What's*Best!*®

Version 17.0

User's Manual

Taking your spreadsheet beyond "What If?"

1415 North Dayton Street, Chicago, Illinois 60642 USA
Phone: (312) 988-7422 Fax: (312) 988-9065
info@lindo.com
www.lindo.com

COPYRIGHT

The What's*Best!*® software and its related documentation are copyrighted. You may not copy the What's*Best!*® software or related documentation except in the manner authorized in the related documentation or with the written permission of LINDO Systems Inc.

TRADEMARKS

What's*Best!*® and LINDO are registered trademarks of LINDO Systems Inc. Other product and company names mentioned herein are the property of their respective owners.

DISCLAIMER

LINDO Systems Inc. warrants that on the date of receipt of your payment, the enclosed computer media contains an accurate reproduction of the What's*Best!*® software and that the copy of the related documentation is accurately reproduced. Due to the inherent complexity of computer programs and computer models, the What's*Best!*® program may not be completely free of errors. You are advised to verify your answers before basing decisions on them. NEITHER LINDO SYSTEMS INC. NOR ANYONE ELSE ASSOCIATED IN THE CREATION, PRODUCTION, OR DISTRIBUTION OF THE WHAT'S*BEST!*® SOFTWARE MAKES ANY OTHER EXPRESSED WARRANTIES REGARDING THE DISKS OR DOCUMENTATION AND MAKES NO WARRANTIES AT ALL, EITHER EXPRESSED OR IMPLIED, REGARDING THE WHAT'S*BEST!*® SOFTWARE, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR OTHERWISE. Further, LINDO Systems Inc. reserves the right to revise this software and related documentation and make changes to the content hereof without obligation to notify any person of such revisions or changes.

ACKNOWLEDGEMENTS

We gratefully acknowledge Professor Sam Savage for his contributions to early releases of What's*Best!*®.

Copyright © 2020, LINDO Systems, Inc.

Published by

1415 North Dayton Street
Chicago, Illinois 60642
USA
Sales & Information: (312) 988-7422
info@lindo.com
Technical Support: (312) 988-9421
tech@lindo.com
www.lindo.com

Contents

CONTENTS ..... **III**
PREFACE ..... **XIV**
1 GETTING STARTED WITH WHAT'SBEST! **1**
What is What'sBest! ? 1
Optimization Models 2
Non-Optimization Models 2
Models with Integer Restrictions 2
System Requirements 3
Installation Overview 4
The What'sBest! Interactive Environment 7
The ABC's: Three Steps to What'sBest! 9
TUTORIAL 10
The XYZ Production Problem 10
The ABC's 12
What's Best If 18
Working While Solving 19
The Next Step 19
2 ABC'S: THE BASICS OF WHAT'SBEST! **20**
Adjustable... 21
Make Adjustable 21
Remove Adjustable 22
Make Adjustable & Free or Remove Free 22
Free 23
Best... 24
Single Objective Optimization 24
Multiple Objective (Lexico/Pre-emptive Priority) Optimization 25
Constraints... 28
Left Hand Side (LHS):, Right Hand Side (RHS):, and Stored in: 28
Constraint-related Problems 32
Explicitly Specifying Convexity to the Solver 33
Solve or 34
Getting the Best Results 37
3 ADDITIONAL COMMANDS **41**
Options and Solvers 41
Integers…|Integer-Binary 43
Refers to 43
Binary-WBBIN 44
General-WBINT 44
Integer Names in Workbook 44
Runtime Concerns in Integer Problems 44
Integers…|Special Ordered Set 45
Select Any Cells 45

Special Ordered Set ... 45
Cardinality ... 46
List of Selected Cells ... 46
Place the Function in Cells ... 46
Integers…|Semi-continuous ... 47
Select Any Cells ... 48
List of Selected Cells ... 48
Place the Function in Cells ... 48
Integers…|All-different ... 49
Options...|General ... 52
Feasibility Tolerance ... 53
Iteration Limit ... 53
Runtime Limit ... 53
Constraint Display ... 53
Auto Select Free/Int/Omit Ranges ... 54
Minimize Excel® on Solve ... 54
Hide Status Window on Solve ... 55
Linearization ... 55
Delta Coefficient ... 55
Big M Coefficient ... 55
Status Report ... 56
Solution Report ... 56
Beginning or End ... 57
Debuggers ... 58
Warnings ... 58
Update Links ... 59
Delete Reports ... 59
Options...|Linear Solver ... 60
Scale Model ... 60
Model Reduction ... 60
Solver Method ... 60
Pricing ... 61
Options...|Nonlinear Solver ... 62
Crash Initial Solution / Use LINDO Crash ... 62
SLP Direction ... 63
SLP Solver ... 63
Quadratic Recognition ... 63
Presolve ... 63
Selective Constraint Evaluation ... 63
Steepest Edge ... 64
Starting Point ... 64
Optimality Tolerance ... 64
Iteration Limit for Slow Progress ... 64
Derivatives ... 65
Options...|Global Solver ... 66
Global Solver ... 66
Multistart Attempts ... 67

Optimality 67
Delta 67
Variable Bound 67
Use of Bound 67
Branching Direction 68
Box Selection 68
Algebraic Reformulation 68
Options...|Integer Pre-Solver 69
Heuristics Level 70
Heuristics Cutoff Criterion 70
Probing Level 70
Constraint Cuts 71
Options...|Integer Solver 73
Branching Direction 74
Integrality 74
LP Solver 75
Optimality 76
Tolerances 77
Usage Guidelines for K-Best Solutions 82
Options...| Stochastic Solver 89
Stochastic Modeling Support 89
Step/Tab 1 "core model" 90
Step/Tab 2 "sequencing" 91
Step/Tab 3 "distributions or correlations" 93
Step/Tab 4 "scenario sampling" 96
Tab Reports 98
Tab Chance-Constrained 106
Frame Specifications 107
Use Simulation Format 109
Options...|Reset to Default 111
Advanced...|Dual... 112
For Cell Range 113
Report on Type 113
Dual Value 113
Upper Range and Lower Range 113
Report Information in 113
Usage Guidelines for Dual Values 114
Dual Value of a Constraint Cell - Shadow Price 114
Dual Value of an Adjustable Cell - Reduced Cost 116
Dual Value of Zero and Multiple Optima 119
Dual Values in Nonlinear Problems 119
Dual Values in Integer Problems 119
Valid Ranges for Dual Values 119
Constraint Ranges 120
Adjustable Cell Ranges 122
Advanced...|Omit... 126
OMIT Names in Workbook and Refers to 126

Usage Guidelines for Omit 127
What To Omit 127
What Not To Omit 127
Advanced...|Function Support 128
Function Support 128
Advanced...|Statistical Functions 129
Advanced...|String Support 129
String Support 129
Maximum String Length 130
Advanced…|Convert Model Format 130
Advanced...|Scenario Viewer 132
Current Selected Worksheet to Convert 133
Advanced...|Parameters 133
LINDO API Parameters 135
Export Models 135
Folder Path 136
About What'sBest! 138
Toolbar / Ribbon Bar 140
License Key Entry 141
Register... 143
CheckUpdate... 144
Language 146
4 WHAT'SBEST! MACROS: THE VBA INTERFACE 147
Usage Guidelines for Macros in VBA 147
Introduction to the VBA Interface of What'sBest! 147
Object Browser 147
Macro Recorder 148
Calling Procedures 148
Error messages you might encounter learning VBA. 149
Beyond VBA / VB 149
Embedding What'sBest! in a Visual Basic® / Visual Studio® Project 149
Concealing and Protecting a Model from the User 150
Tutorial on the VBA Interface 151
Building and Solving a Basic Model 151
Solving Multiple Problems with a Looping Macro 154
VBA Interface: Procedures 155
wbAddAdjustableStyle 156
wbAddBestStyle 156
wbAddWBMenu 156
wbAdjust 157
wbBest 158
WBBIN 159
wbConstraint 160
wbConvertModelFormat 162
wbDeleteReports 162
wbDeleteWBMenu 163
wbDualValue 163

wbError, and Error Codes 165
wbError 166
WBFREE 173
WBINT 174
wbInteger 175
wbIntegerAllDiff 176
wbIntegerCard 177
wbIntegerSemic 178
wbIntegerSos 179
wbIntegerSolverKBestReport 180
WBOMIT 181
WBProgressInfoRequest 182
wbResetOptionsToDefault 184
wbSetFunctionSupport 184
wbSetGeneralOptions 185
wbSetGlobalOptions 189
wbSetIntegerOptions 191
wbSetIntegerPreSolverOptions 195
wbSetLinearOptions 198
wbSetNonlinearOptions 199
wbSetStochasticOptions 201
wbSetStochasticSupport 204
wbSetStringSupport 204
wbSolve 205
wbStochasticChanceConstrained 207
wbStochasticFunction 209
wbStochasticHistogram 210
wbStochasticReport 212
wbStochasticStageScenario 213
wbUpdateLinks 214
5 FUNCTIONS AND OPERATORS 216
Supported Functions and Operators 216
Functions and Operators in Excel Recognized by What'sBest! 216
Functions Added by What'sBest! 217
Distributions/Correlations for Stochastic Programming 218
Linearization 219
Global Solver 219
Range Name Usage 220
Functions Descriptions 220
WBALLDIFF 220
WBCARD 220
WBDUAL 221
WBERLANG_B 221
WBERLANG_C 222
WBEXPOINV 222
WBINNERPRODUCT 223
WBLMTD 223

WBLOGABEXPX 224
WBLOGSUMAEXP 225
WBLOGSUMEXP 226
WBMULTINV 227
WBNORMSL 227
WBPOSD 228
WBQUADPRODUCT 228
WBRLMTD 229
WBSEMIC 229
WBSIGNPOWER 230
WBSOS1 230
WBSOS2 231
WBSOS3 231
WBTRIAINV 232
WBUNIFINV 232
WBUPPER 233
WBXPOWDIVAB 234
6 OVERVIEW OF MATHEMATICAL MODELING 235
Introduction 235
Linear vs. Nonlinear Expressions and Linearization 235
Linear Expressions 235
Nonlinear Expressions 236
Linearization 236
The Solution Process: Determining Optima 237
Local Optima vs. Global Optima 237
Convexity 239
Smooth vs. Non-smooth Expressions 240
Solution Outcomes 242
Case 1: Optimization Models 242
Case 2: Non-optimization Models 243
Guidelines for Modeling with What'sBest! 244
General Modeling Guidelines 244
Nonlinear Modeling Guideline 245
Scale the Model to a Reasonable Range of Units 245
Simplify Relationships 246
Reduce Integer Restrictions 246
Guidelines for Stochastic Modeling 247
Multistage Decision Making Under Uncertainty 247
Recourse Models 249
Defining an SP Model in What'sBest! : Simple 2-Stage Example 250
Defining an SP Model in What'sBest! : Multi-Stage Example 258
Sampling a Scenario Tree 265
Guidelines for Chance-Constrained Modeling 266
Solving Second Order Cone Programs 273
7 SAMPLE MODELS 277
Blending 281
Application Profile 281

Chance-Constrained Blending 286
Application Profile 286
Box Design 290
Flow Network Modeling 293
Bond Portfolio Optimization 301
Application Profile 301
Lockbox Location 307
Markowitz Portfolio Problem 311
Application Profile 311
Portfolio with Transaction Costs 315
Application Profile 315
Portfolio - Minimizing Downside Risk 318
Application Profile 318
Portfolio - Scenario Model 321
Seasonal Sales Factoring 327
Exponential Smoothing 331
Application Profile 331
Linearization Option: Construction Cost Estimation 336
Application Profile 336
Stratified Sampling 340
Application Profile 340
Car Pricing 343
Media Buying 347
Application Profile 347
Multi-Period Inventory Management 350
Application Profile 350
Product Mix 353
Application Profile 353
The Building Block Method 357
Application Profile 357
Waste Minimization in Stock Cutting 362
Application Profile 362
Plant Location 367
Staff Scheduling 372
Application Profile 372
Staff Scheduling: Preferred Assignment 376
Application Profile 376
Staff Scheduling: Two Stage Fixed Shift 383
Application Profile 383
Stage 1: Cost Minimization 383
Stage 2: Preference Maximization 388
Pipeline Optimization 392
Application Profile 392
Shipping Cost Reduction 396
Application Profile 396
Traffic Congestion Cost Minimization 399
Application Profile 399

Truck Loading 402
Application Profile 402
Crop Allocation Under Uncertainty 406
Application Profile 406
Put Option 413
Application Profile 413
8 TROUBLESHOOTING 423
FAQ Frequently Asked Questions 423
Error Messages & Warnings 440
Errors from VBA Code 440
Runtime Errors 441
Errors from an embedded application 442
Invalid Outside Procedure 442
ADDINLINK - Multiple Add-in Links 442
ARITHERR - Arithmetic Error 443
BLKCELL - Blank Cell Warning 443
ENTIRERANGE - Entire Range Selection 444
EXLVER - Excel® File Format 444
FORMULA1 - Formula Parsing 445
FORMULA2 - Unsupported Functions 445
FORMULA3 - External Reference 446
FUNCADDIN - Failed to Access the Add-in 447
FUNCMACRO - Failed to Access the Macro 447
FUNCSERVER - Failed to Access the Server 448
HIDSNAME - Hidden Solvers Names 449
IKBMOD - K-Best Model Format 450
IKBREP - K-Best WBIKB_REP Format 451
INDEX - Unsupported Index Usage 451
INDICMOD - Indicator Model Cell Reference 452
INFEASIBLE - No Feasible Solution Found 453
INFLARG - Large Infeasibility 454
INTERRUPT - Solver Interrupt 454
INVALBEST – Invalid Best Cell 455
INVMOD - Invalid Model 456
IRRECONST - Irreconcilable Constraints 456
ITRLIM - Iteration Limit Reached 457
LICCAP1 - Constraint Limit 457
LICCAP2 - Adjustable Cell Limit 458
LICCAP3 - Integer Cell Limit 458
LICCAP4 - Nonlinear Adjustable Cell Limit 459
LICCAP5 - Global Adjustable Cell Limit 460
LICKEY1 - Failed to Process License Key 460
LICKEY2 - Pending License Expiration 461
LICKEY3 - Expired License Key 461
LICKEY4 - Pending Expiration of Option Trial 461
LICKEY5 - License Key Dongle Required 462
LICOPT1 - Global Solver Not Licensed 462

LICOPT2 - Nonlinear Solver Not Licensed 463
LICOPT3 - Barrier Solver Not Licensed 463
LICOPT4 - Mixed Integer Solver Not Licensed 464
LICOPT5 - Stochastic Solver Not Licensed 464
LICOPT6 - Conic Solver Not Licensed 464
LOOKUP - Unsupported Lookup Usage 465
MATCH Unsupported Match Usage 465
MBEST – Multiple Best Cells 466
MEMORY - Insufficient Memory 466
MODERR - Parsing Cell Formula 467
NAME - Unsupported Value in the Name List 468
NAMECONFLICT - Name Cell with Conflicting Properties 468
NLINCELL - Nonlinearities Present 469
NOADJ - No Adjustable Cells 469
NOBEST - No Best Cell 470
NOCONST - No Constraint Cells 471
NUMERICAL - Numerical Error 471
OMITTED - Omitted Cell Reference 472
QUAPREC - Quadratic Recognition 472
RUNTIME - Runtime Limit Reached 473
SOLVERGOP - Global Solver Use 473
SOLVERR - Solver Error 474
SPCONFLICT - Stochastic Conflicting Declarations 475
SPCORR - Stochastic WBSP_CORR Format 475
SPDIST1 - Stochastic WBSP_DIST 1 Format 476
SPDIST2 - Stochastic WBSP_DIST 2 Format 476
SPDIST3 - Stochastic WBSP_DIST 3 Format 477
SPERR - Stochastic Error 478
SPHIST - Stochastic WBSP_HIST Format 478
SPRAND - Stochastic WBSP_RAND Format 479
SPREP - Stochastic WBSP_REP Format 479
SPSTSC - Stochastic WBSP_STSC Format 480
SPVAR - Stochastic WBSP_VAR Format 480
STRARG - String Arguments Found 481
STRLIST - String List Error 481
STRRES - String Result of Formula 482
SUMIF - Unsupported SUMIF Usage 482
SUMIFS - Unsupported SUMIFS Usage 483
TEMPFILE - Error Managing Temp Files 484
UNBOUNDED - Unbounded Model 484
UNDEFREF - Undefined Reference 485
WBALLDIFFFORM – All-different Cell Format 485
WBCARDFORM - Cardinality Cell Format 486
WBDUFORM - Dual Cell Format 486
WBPOSDFORM - Semi-definite Positive Cell Format 487
WBLOFORM - Lower Range Cell Format 487
WBSEMICFORM - Semi-continuous Cell Format 488

WBSOSFORM - Special Ordered Sets Cell Format 488
WBUPFORM - Upper Range Cell Format 489
APPENDIX A: INSTALLATION DETAILS 491
Installation 491
What'sBest! Example Files 491
Setup Type 491
What'sBest! Add-in Files 492
Location of the Add-In Files 492
Installation Related Problems 492
Settings for Microsoft® Excel® 2007, 2010, 2013, 2016, 2019 / Office365® 493
Settings for Microsoft® Excel® 1997, 2000, 2002, 2003 493
Add-ins 495
Location of the Add-In Files and Update Links 495
Uninstall Files 499
Disclaimer 500
Acknowledgement 501
APPENDIX B: CONTACTING LINDO SYSTEMS 503
www.lindo.com 503
INDEX 505

Preface

What'sBest!® is an add-in to Microsoft® Excel® that allows you to build large scale optimization models in a free form layout within a spreadsheet. *What'sBest!*® combines the proven power of Linear, Nonlinear (convex and non-convex/Global), Quadratic, Quadratically Constrained, Second Order Cone, Semi-Definite, Stochastic, and Integer optimization with Excel® -- the most popular and flexible business modeling environment in use today. *What'sBest!*® 17.0 includes major solver enhancements to increase speed and reliability on broad classes of models.

What's New in Release 17

- **Support for Multiple Objective Optimization**
  - Support has been added to handle multiple objectives for linear and integer models using a strategy known as Lexicographic/Preemptive optimization. The feature allows the user to specify an ordered list of objectives. *What'sBest!* will sequentially solve the model by optimizing the 1st objective, then constraining the 1st objective to its optimal value and optimize the 2nd objective, then constraining the 1st and 2nd objectives and optimize the 3rd objective, and so on down the objective list.

- **Improved Support for using Network and Cloud Drives**
  - A feature has been added to allow users to more easily solve models stored on network and cloud drives.

- **Improved Support for Excel® Functions**
  - Support for the SUMIFS() formula has been added.
  - Support for the SUMIF() formula has been extended.

- **Improved User Interface**
  - The format and layout of many dialog boxes have been simplified with smarter choice of defaults but with no loss of functionality.

- **Simplified and Improved Installation Process**
  - The new smart installation process checks for 32-bit vs 64-bit Excel® and installs the appropriate version automatically.

- **Support for User VBA Progress Callback**
  - This feature has been added to allow users to define a callback procedure (via the VBA interface) for accessing the progress information during the solution process.

- **Global Solver Improvements**
  - Improved performance for models with convex-concave functions.
  - Models with ratio/fractional terms, including MIP, tend to solve faster, sometimes order of magnitude.
  - Non-convex quadratically constrained models are solved substantial faster.
  - Improved recognition and exploitation of convexity of various composite functions, e.g.,

ln(a + b * exp(x))	WBLOGABEXPX
ln(a_1 * exp(x_1) + . . . +a_n * exp(x_n))	WBLOGSUMAEXP
ln(exp(x_1) + . . . + exp(x_n))	WBLOGSUMEXP
x^*a*/(*b* - *x*)	WBXPOWDIVAB

- **Significant Speed Improvements on Quadratic/Conic Models**
  - Support has been added for exponential and power cones.
  - The models with e^x and x^k terms are solved faster.

- **Integer Solver Improvements**
  - Better performance on many integer models has been achieved through improved selection of default settings for cut generation and solution heuristics.

- **Improved Performance on Linear Models**
  - Enhancements to the Simplex solvers allow linear models with many more variables than constraints to solve in a fraction of the time previous releases required.
  - Performance has been improved for linear models when using multiple cores with concurrent execution of Primal, Dual, and Barrier solvers.

Please do not hesitate to contact us if you have problems or questions. As always, we welcome your feedback and suggestions. The *What'sBest!*® development team wishes you the *Best* in all your optimization endeavors!

Copyright © 2020, LINDO Systems, Inc.

1 Getting Started with What's*Best!*

What is What'sBest! ?

What's*Best!* makes available to your Excel® spreadsheet program a highly developed solver capable of performing linear and nonlinear optimization on the most difficult of problems. What's*Best!* gives you access to this solver from within Excel®, and may either be run directly or called from within Visual Basic®.

People in business, finance, science, math, and many other fields use What's*Best!* every day to model and solve problems in production, financial planning, personnel scheduling, resource allocation, portfolio management, stock cutting, inventory control… It's a long list to which you will want to add your own application. To demonstrate some of the range of applications for What's*Best!*, sample models are provided with the software and many are explained in Chapter 8, *Sample models.*

Optimization Models

Optimization helps you find the answer that yields the most desirable result — the one that attains the highest profit, output, or happiness, or the one that achieves the lowest cost, waste, or discomfort. Often these problems involve making the most efficient use of your resources — including money, time, machinery, staff, inventory, etc.

Optimization problems are classified as linear or nonlinear depending on whether the relationships in the problem are linear with respect to the variables. What's*Best!* can solve both linear and nonlinear problems, with optional integer restrictions. For more information on optimization and the solution process, please refer to *Overview of Mathematical Modeling*. For additional reference, we recommend the LINGO textbook, *Optimization Modeling with LINGO*, or the LINDO textbook, *Optimization Modeling with LINDO*, both by Linus Schrage and available through LINDO Systems.

Non-Optimization Models

Non-optimization models do not have a cell defining an objective function that is to be maximized or minimized. A common purpose of such models is to solve an equation. What's*Best!* can find values that satisfy sets of equations or satisfy circular references. Similarly, What's*Best!* can find values to accomplish a desired result — commonly called goal seeking or backsolving. The sample model *Flow Network Modeling* is an example of a non-optimization model.

Models with Integer Restrictions

Users frequently need to find answers that consist of whole units (i.e., variables expressible as integers). In personnel scheduling, for example, the most efficient use of workers may call for 2.37 persons on a shift, but it's difficult to find .37 of a person who is capable of meaningful work. Therefore, What's*Best!* allows you to restrict values to be whole numbers (or general integers). Binary integer (0/1) restrictions can also be helpful in yes/no or on/off decisions.

System Requirements

To install and run What's*Best!*, check that you have the following:

Software

- Microsoft® Windows® 10 or previous Windows® XP, Vista®, 7, 8
- Microsoft® Excel® 32-bit 2019 / Office365®, and previous versions 2002, 2003, 2007, 2010, 2013, 2016
- or Microsoft®Excel® 64-bit versions 2019 / Office365®, or older 2010, 2013, 2016 with latest Service Packs
- Microsoft®Excel®, 'PC' version
- Microsoft® .NET Framework 4.0

Hardware

- Pentium-class PC
- 500 MB of RAM
- 50 MB of free disk space

An Internet connection is required to download the latest version of What's*Best!*. You can also contact LINDO Systems to obtain a copy. Add-ins, library and executable files from LINDO Systems are digitally signed. Additional information can be found in the *Installation Overview*.

Installation Overview

To get up and running quickly, first close all programs and simply run **WB-WINDOWS-17.0.exe** and follow the prompts in the dialog boxes that follow (see Section *Installation* for more information on installing with **WB-WINDOWS-17.0.zip**).

The setup program offers *Default* and *Specified* installation. The *Default* installation option analyzes your system for the critical information required to successfully install What's*Best!*. When you select *Default* as the installation type in the initial dialog box, the add-in file is copied into a directory entitled *Library* within the main Excel® directory.

The *Specified* installation option is available for situations that require more information from the user and for users who prefer controlling the details of installation. See the section entitled *Installation* for more information.

Default installation is recommended for most environments. However, we recommend you select *Specified* installation under the following situations:

- You have more than one copy/release of Excel® installed on your system.
- Excel® is installed on a network server rather than a local drive.
- Your system runs on a non-English version of Windows®.

If you are updating from an earlier release, the location that you specify for the program files may affect how your existing models are interpreted. Carefully read the information displayed during installation, as well as the Location of the *Add-In Files and Update Links* section.

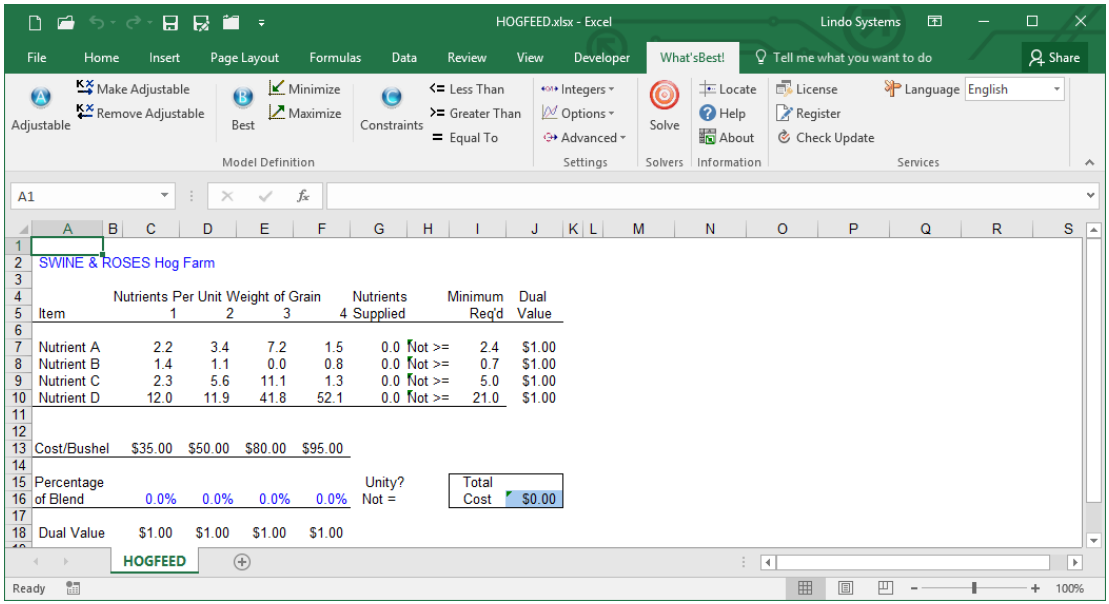
As the last step in installing What's*Best!*, Excel® will open and you will find a new toolbar (What's*Best!*) and menu (*WB!*) specific to the What's*Best!* program. These are discussed in the *The What'sBest! Environment: Menu and Toolbar* section. If you do not find the *WB!* menu present, please go to *Tools*|*Add-ins* and check the 'What's*Best!*' entry or try reinstalling What's*Best!*.

You may be prompted for a license key the first time the software runs. Please copy-and-paste the license key received via email directly into the space provided (*Ctrl-C* to copy, *Ctrl-V* to paste).

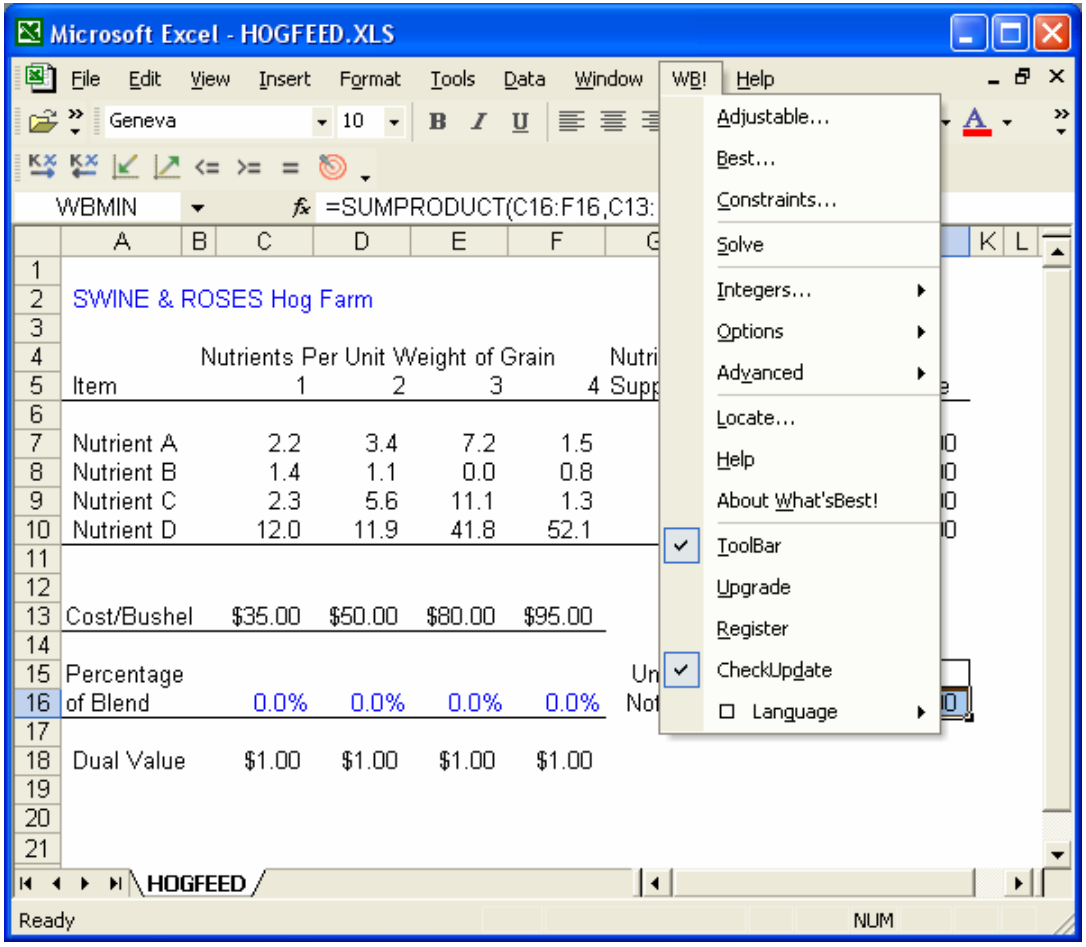
If you would like to run a trial version of What's*Best!* (150 constraints, 300 variables, and 30 integers), no license key is required, and just click *Trial* here.

If you have misplaced your license key, you will need to contact LINDO Systems.

You can now start building and solving models using the What's*Best!* menu and toolbar, integrated in the Ribbon bar in Excel® version 2016:



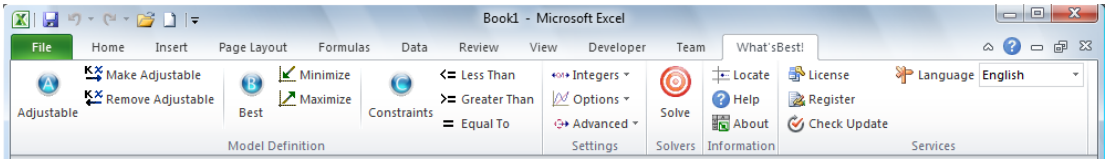
In Excel® version 2002, the menu and toolbar show as follow:



The What'sBest! Interactive Environment

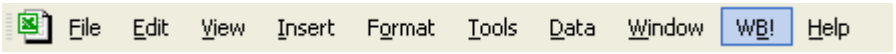
What's*Best!* offers two interactive methods to access commands, the What's*Best!* menu and toolbar. This section describes how to effectively use this interface.

In Excel® versions 2007 and later, What's*Best!* shows an additional tab in the Ribbon layout:



In Excel® version 2002-2003:

1) The What's*Best!* menu (*WB!*) is embedded in the main Excel® menu bar and appears as follows:



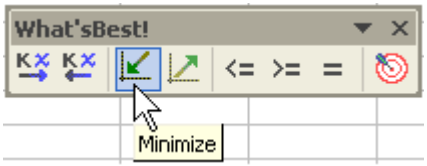
The *WB!* menu contains all the commands discussed in *ABC's: Basic Functions, and Additional Commands*.

2) The What's*Best!* toolbar can float on your screen or be repositioned to a preferred part of the Excel® window and appears as follows:



The buttons on the toolbar correspond to the menu commands *Adjustable…*, *Best…*, *Constraints…*, and *Solve*, which are discussed in *ABC's: Basic Functions*.

Of these two access methods, the What's*Best!* menu provides for full interactive use of the entire range of What's*Best!* commands. For faster access, the What's*Best!* toolbar offers rapid one-click access to the most commonly used operations. The What's*Best!* toolbar also makes tool tips available. To learn the function of a particular toolbar button, just move the cursor over the button for a second and a statement of the button's function will appear as follows:



When What's*Best!* is first installed, the following events should occur:
1) the What's*Best!* add-in is loaded;
2) the What's*Best!* menu is inserted in the Excel® menu bar; and
3) the toolbar appears in floating (undocked) mode. The user may then reposition the What's*Best!* toolbar to a preferred part of the Excel® window.

To remove the toolbar from view, either go to WB!|Toolbar or to View|Toolbars on the main Excel® menu bar and uncheck the What's*Best!* toolbar. Reverse the process to return the toolbar to view. Should you wish to remove the What's*Best!* toolbar, you would use View|Toolbars|Customize… command, select the What's*Best!* toolbar, and press the Delete button. You would then have to reinstall What's*Best!* in order to restore the toolbar.

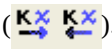
What's*Best!* advises the user as to its progress via the Excel® status bar displayed at the bottom of the Excel® window. This bar normally displays "Ready", but displays other messages when either Excel® or What's*Best!* is performing some activity that may be prolonged.

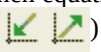
For further information on the What's*Best!* add-in, see the *Add-ins* section.

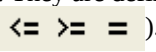
Developing a Model in What'sBest!

The ABC's: Three Steps to What's*Best!*

There are three steps to setting up a model to be solved by What's*Best!*. Throughout this help file, you'll find them referred to again and again. We call them the ABC's (Adjustable, Best, and Constraints):

A. Identify Adjustable Cells: The *adjustable cells* are the cells in the worksheet that What's*Best!* can adjust in its quest for a solution. In mathematical programming terms, these are called variables. These can be defined using either the menu command (*Adjustable…*) or the toolbar button ().

B. Define Best: The *best cell* is the goal, or objective, of your solution. Typically, this is to maximize or minimize an adjustable cell or some function of the adjustable cells. What's*Best!* allows only one best cell in the model. No best cell is needed when equation solving or goal seeking. The menu command (*Best…*) or the toolbar button () can be used to define the best cell.

C. Specify Constraints: The *constraint cells* identify any limitations in a model. For example, "Raw materials used in production must be less-than-or-equal-to raw materials on hand". The constraint cells enforce these restrictions. They are defined by either the menu command (*Constraints…*) or the toolbar buttons ().

Once you've specified the ABC's, you can solve () your worksheet model and find the best answer to your problem.

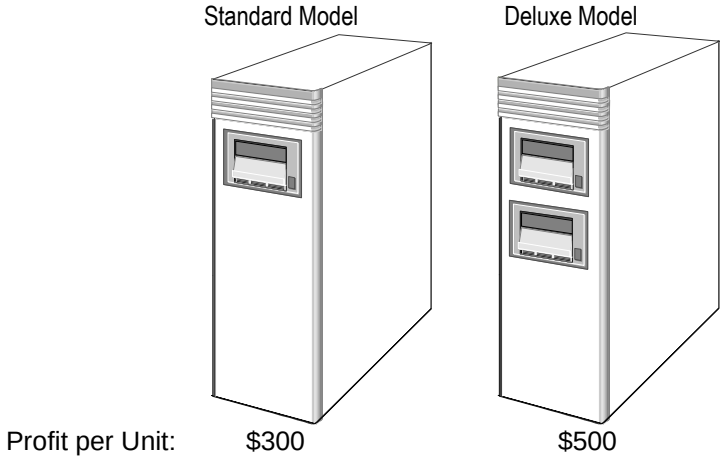
TUTORIAL

This tutorial will introduce you to What's*Best!* by showing you how to solve a simple linear problem called the *XYZ Production Problem*. You should open the sample problem file *XYZ.XLS* located in the *WB* directory in your C: drive, so you can finish formulating the model in this tutorial (The other sample models have been completely formulated for you and are ready to solve).

Please note that the following discussion uses Excel® naming conventions (such as $C$5:$D4) throughout.

The XYZ Production Problem

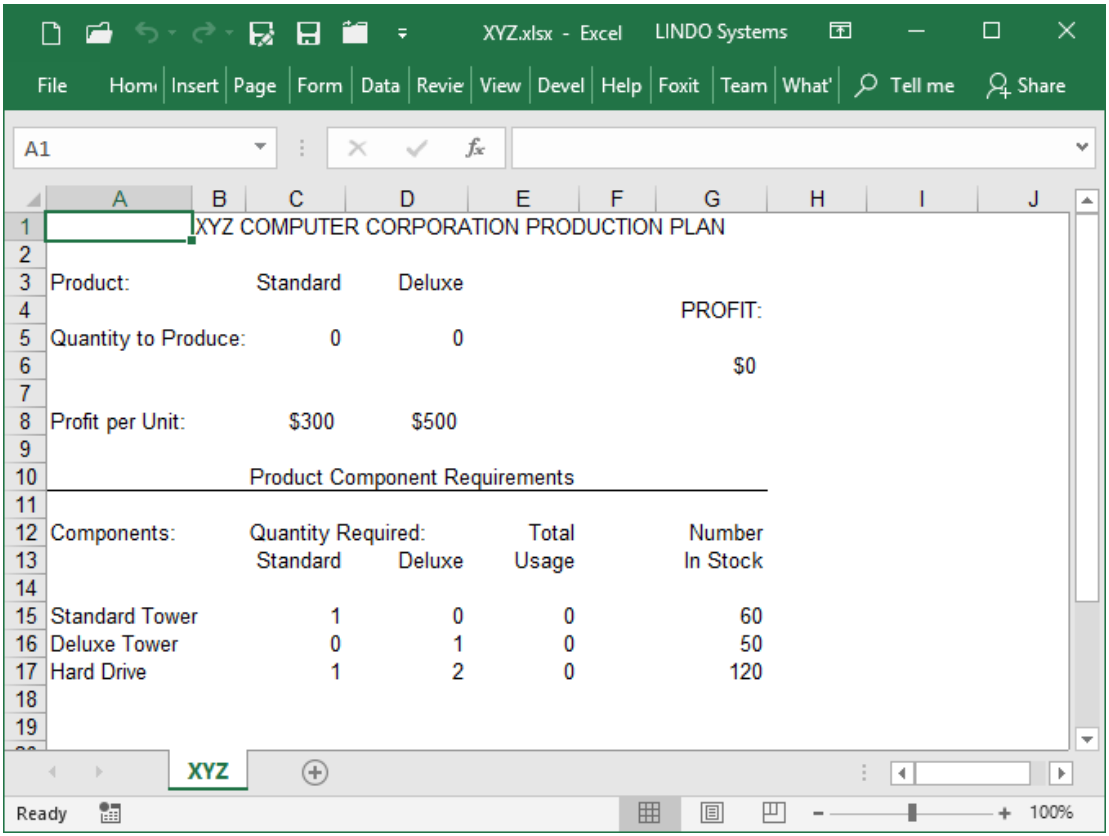
XYZ Corporation builds two computer models – the *Standard* and the *Deluxe*. The Standard has a profit per unit of $300, and the Deluxe has a profit per unit of $500. The two models are produced from three components: the Standard computer tower, the Deluxe computer tower, and a hard drive.



Problem What combination of Standard and Deluxe computer towers will maximize XYZ's profit from the components currently in stock?

Unless specified otherwise during installation, the *XYZ.XLS* file was copied to the *WB* subdirectory. If you haven't already done so, open the *XYZ.XLS* sample file:

The *XYZ* Worksheet Before Optimization



Examine the layout and logic of the model. You might want to experiment with various "What If?" projections. For example, try to adjust the *Quantity to Produce* in both cells (C5 and D5), so as to maximize profit (G6) without allowing your *Total Usage* (E15:E17) to exceed the number of components in stock (G15:G17).

For example, a sensible production plan might be to make as many *Deluxe* models as possible (since these yield the highest per unit profit). Then, with what is left, make as many *Standard* models as possible. This production plan would use all 50 *Deluxe* computer towers (E16), 20 *Standard* computer towers (E15), and all 120 hard drives (E17) currently in stock. It would result in a total profit of $31,000 (G6). However, this solution can be improved by using What's*Best!*.

The ABC's

Now, let's apply the ABC's to this spreadsheet to show how What's*Best!* provides you with the best possible answer.

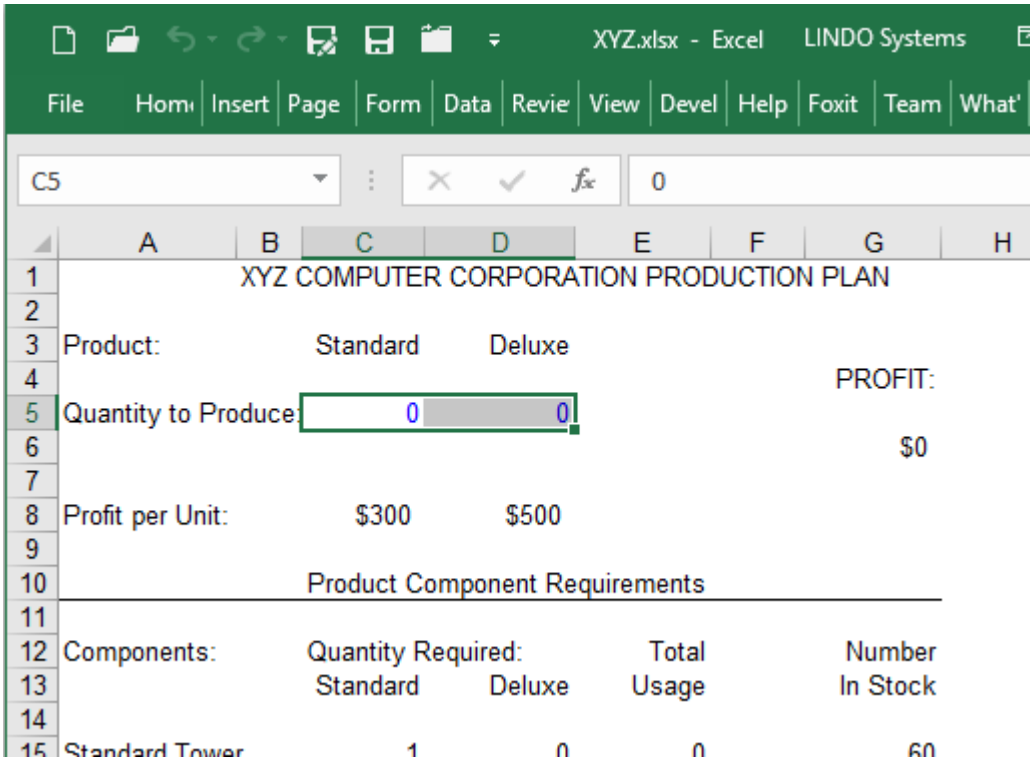
A. Determine Adjustable Cells

In this example, we first want What's*Best!* to adjust the value for *Quantities to Produce* for both models of computer (in cells C5:D5). What's*Best!* requires that numbers be placed in all adjustable cells. You may simply enter zeroes in these cells, although any number will suffice. Next, specify that cells C5 and D5 are adjustable by selecting both cells and either:

1) choose *Adjustable…* from the *WB!* menu, and click *OK*, or
2) use the *Make Adjustable* toolbar button, or to *Remove Adjustable*.

If you use the *WB!* menu to access the *Adjustable* dialog box, you'll notice that the *Adjustable* dialog box has $C$5:$D$5, the current selection, entered in the *Refers To:* text box. In the drop-down box, the default setting *Make Adjustable* is already selected, so you need only click *OK*. What's*Best*! identifies an adjustable cell by applying an *Adjustable* style, distinguished by a default font color of blue (as a visual reminder) and unlocked cell status as follows:

The Adjustable Cells in *XYZ*



B. Define Best

XYZ Corporation's goal is to maximize profit, which may be expressed as:

Total Profit = (*Quantity of Standard Models Produced*)
TIMES (*Profit per Unit of Standard Models*)
PLUS (*Quantity of Deluxe Models Produced*)
TIMES (*Profit per Unit of Deluxe Models*)

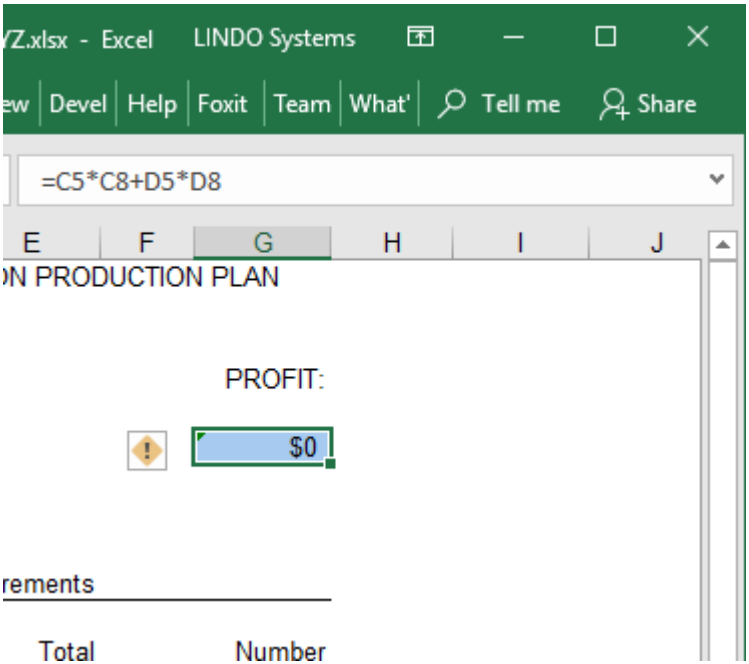
This formula appears in cell G6 as =C5*C8+D5*D8. This is the sumproduct of C5:D5 and C8:D8, and could also be entered as =*SUMPRODUCT*(C5:D5,C8:D8). This function can be especially useful when doing similar operations on larger ranges.

To make G6 the objective (best cell), move the cursor to that cell and either:

1) choose *Best...* from the *WB!* menu, select *Maximize***,** and click *OK*, or
2) use the *Maximize* toolbar button .

If you use the *Best* dialog box via the menu, you'll notice that the right text box on the *Best* dialog box indicates $G$6, the current selection.

The Objective (Best Cell) in *XYZ*



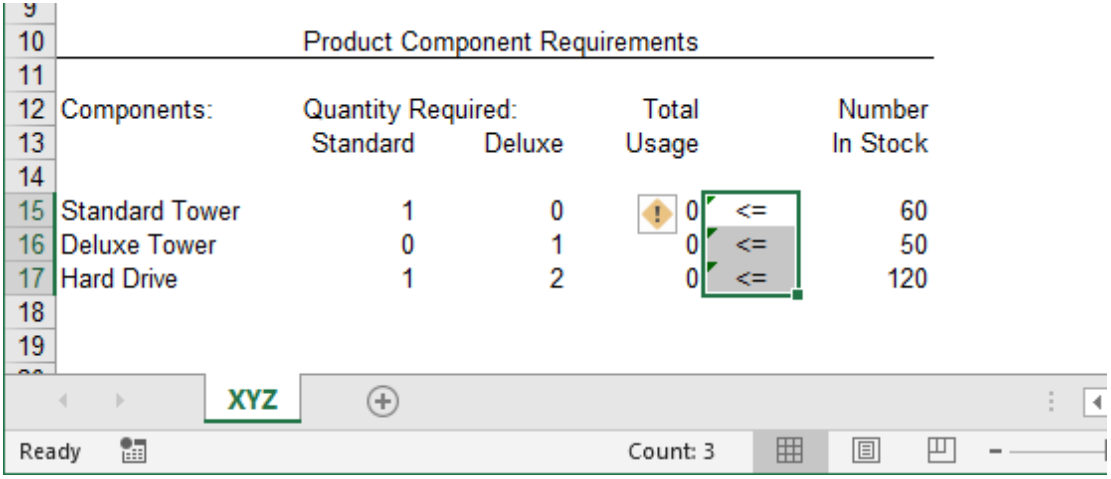
C. Specify Constraints

The constraints for this problem are that the *Total Usage* of components (E15:E17) must be less-than-or-equal-to the number in stock (G115:G17).

The formula for *Total Usage* of *Standard* tower components is =C5*C15+D5*D15. Cells E16 and E17 have similar formulas for the Deluxe tower and hard drives.

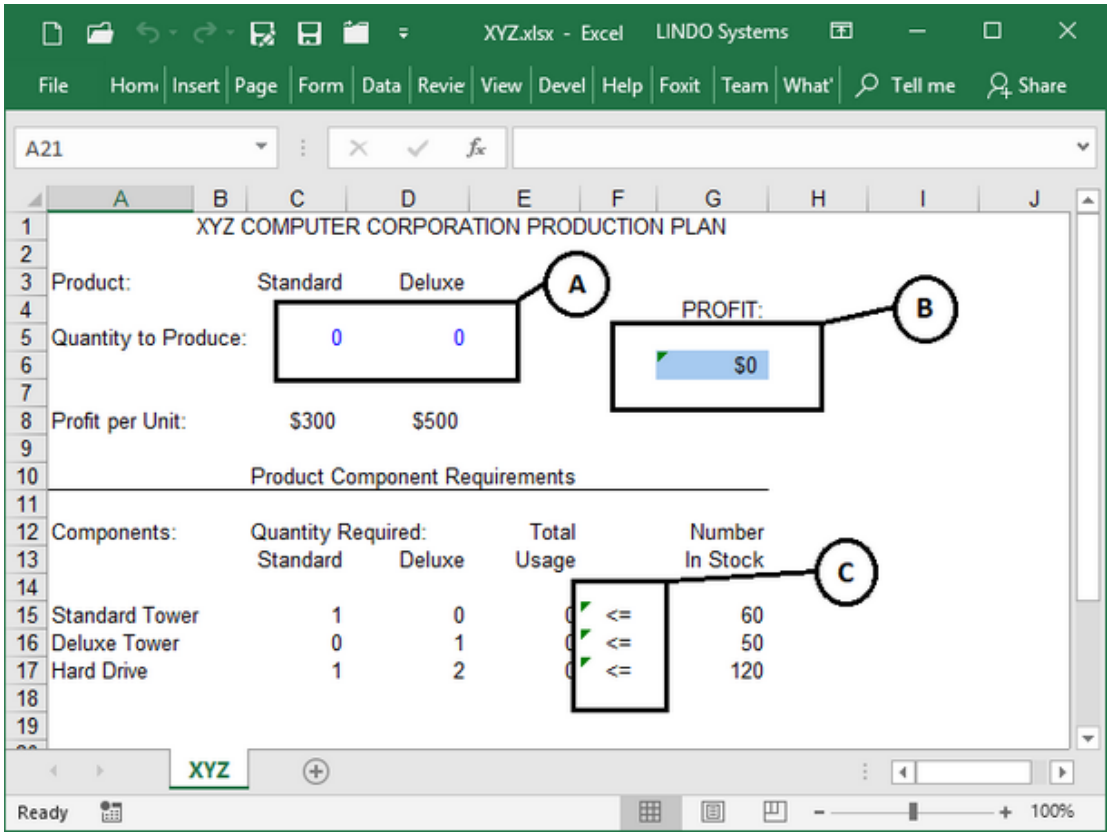
To specify the constraints, highlight the range F15:F17, then choose *Constraints…* from the *WB!* menu, and click *OK*. Note that the *Constraints* dialog box has E15:E17 entered as *Left-hand side (LHS)*, =G15:G17 entered as *Right-Hand Side (RHS)*, $F$15:$F$17 entered in *Store in:*, and <= (less-than-or-equal-to), being the default, is entered as the constraint type. Alternately, you may use the <= toolbar button after selecting the range to store the constraints. See *The ABC's* for more information on these defaults.

Automatic Generation of a Constraint



At this point, let's briefly summarize what we have done so far:

The *XYZ* Worksheet *Before* Optimization

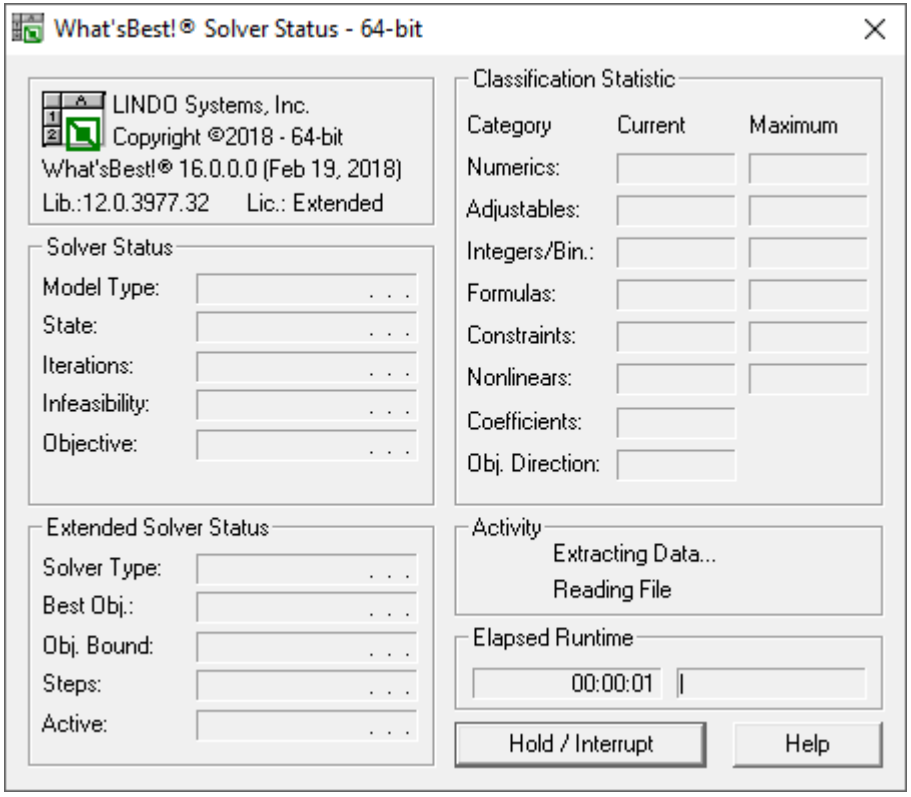


A. Set *Adjustable cells*: This lets What's*Best!* know that the spreadsheet cells denoting *Quantity to Produce* (C5:D5) are the variables and can therefore be adjusted as What's*Best!* seeks an optimal answer.

B. Define *Best*: As XYZ's goal is to maximize profit, the total profit (G6) was specified as the objective of optimization. This calls for maximizing the value of the formula in cell G6 through manipulation of the values in the adjustable cells (C5:D5).

C. Specify *Constraints*: XYZ's limitations are simply that *Total Usage* of inventory parts (E15:E17) be less-than-or-equal-to the total *Number in Stock* (G15:G17). These constraints are specified by means of the formulas in F15:F17.

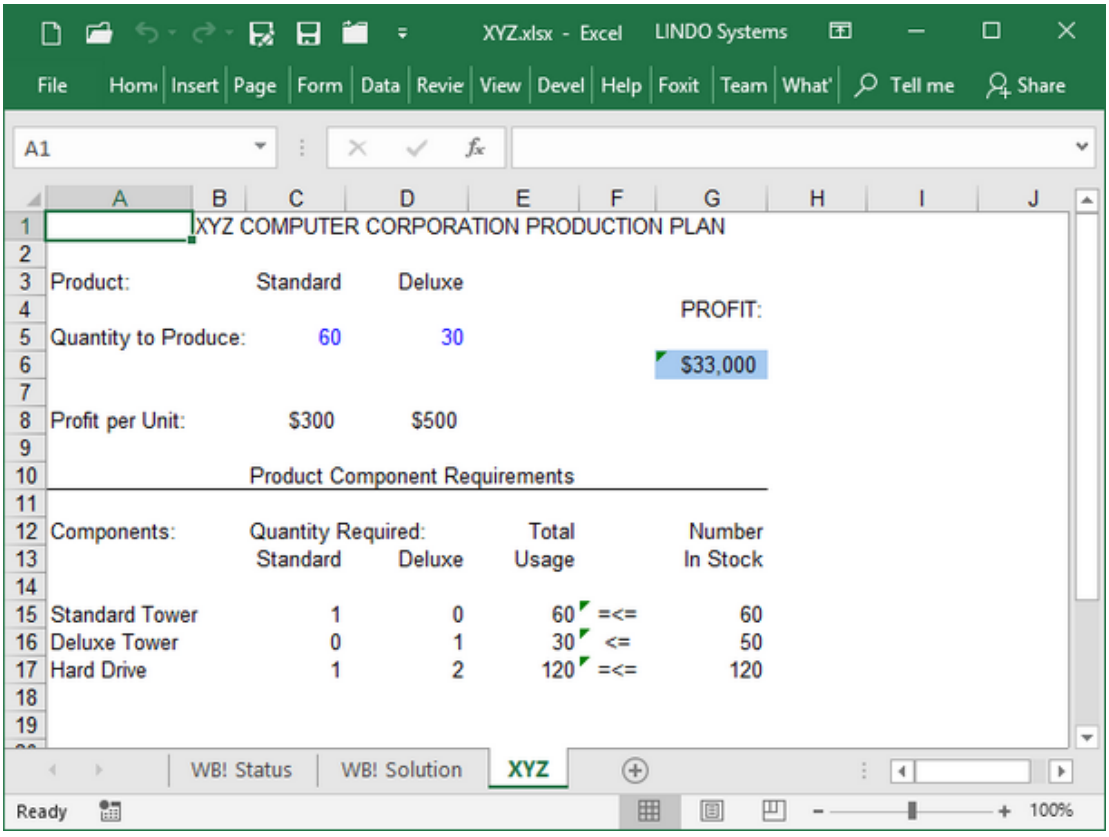
Now, you are ready to solve the XYZ model with What's*Best!*. When you choose *Solve* from the *WB!* menu or press on the toolbar, the What's*Best!* solver is started and the solver status window appears like the one shown below:



The solver status window contains useful information about the properties of your model and the progress of What's*Best!* toward a solution. The information about model properties will also appear on the *WB! Status* worksheet tab added to your workbook after the model is solved, so don't worry if the window doesn't remain long enough for you to read the information it contains. See the *Solve* section for detailed information about the solver status window.

The worksheet soon reappears indicating the highest possible profit:

The *XYZ* Worksheet *After* Optimization



The What's*Best!* solution yields the best possible profit attainable given your resources and constraints. It also tells you the appropriate quantities to produce and the total usage of each component. Note that profit is now $33,000–considerably higher than the $31,000 attained by simply building as many Deluxe models as possible.

What's Best If

It's often useful to explore your optimization model from various angles. For example, what if you learn that the 20 Deluxe computer towers remaining from the What's*Best!* answer found above (G16 - E16) will be obsolete after the current production run? In that case, it might be better to maximize the utilization of this inventory than to maximize profit. Try this by making *Total Usage* of Deluxe computer towers (cell E16) your best cell to be maximized. This is done by selecting cell E16, choosing *Best...* from the *WB!* menu, clicking *OK* and then re-solving.

What's*Best!* will maximize the value of cell E16 without regard to the former objective of maximizing *Total Profit*. Note that the resulting production of 50 Deluxe units completely exhausts the obsolete inventory, but that profit has dropped from $33,000 to $25,000.

Now, suppose for financial reasons profit must be at least $32,000. This constraint can be manually entered by simply typing =*WB*(G6,">=",32000) in a convenient cell (in this case, let's use H6)**.**

Optimizing again with the new constraint in place results in the production of 40 units of each model, a profit of $32,000, and a surplus of 10 Deluxe towers.

Working While Solving

This simple model solves very quickly. While larger models are solving, you may want to do other work as you wait for your solution.

If you want to work in Excel® while your model is being solved, then just minimize the model being solved and open another copy of Excel® (or whatever application you wish to you use). Simply leave the minimized Excel® icon on the taskbar at the bottom of the screen, and your model will continue to solve in the background.

The Next Step

The *ABC's: Basic Functions* and *Advanced Functions* sections explain the What's*Best*! commands in depth. To learn more about the principles of optimization modeling, *see Overview of Mathematical Modeling*. If you'd like to move on to more sophisticated problems, see *Sample Models*.

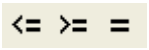
2 ABC's: The Basics of What'sBest!

This chapter describes how to use the three basic commands that are used in building almost every model for What's*Best!*. We'll refer to them as the ABC's – Adjustable, Best, Constraints. These three commands are the first commands at the top of the main *WB!* menu item on the Excel® menu. In addition to the menu commands, you may use the corresponding toolbar buttons on the What's*Best!* toolbar.

The sections below include in depth discussion of the use of these commands:

Adjustable… 

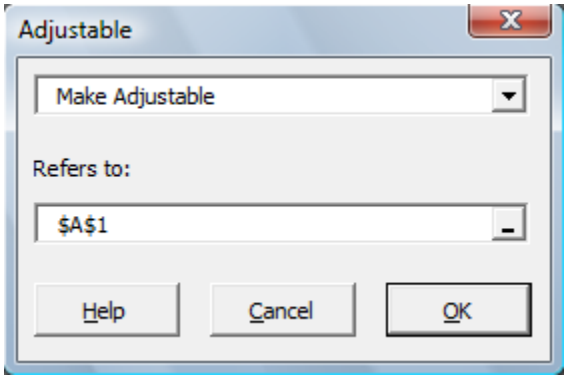
Best… 

Constraints… 

After you have created your model by setting up the ABC's, you can *Solve*  your worksheet model and find the best answer to your problem.

Adjustable...

The dialog box posted by the *Adjustable* command appears as follows:



This command allows you to set properties of selected cells in your workbook. Specifically, it is used to identify to What's*Best!* the cells that may be adjusted by the solver in it's search for a solution that optimizes the objective cell while satisfying all constraints.

Enter the range of cells whose properties you wish to set in the *Refers To:* field (the current selection of cells is already entered). Next, select the property you wish to apply to the cells from the drop-down list at the top of the dialog box. You have the following options:

1) *Make Adjustable*
2) *Remove Adjustable*
3) *Make Adjustable & Free* or *Remove Free*

See below for more detail on these options. Clicking *OK* will apply the change in properties.

Make Adjustable



Select *Make Adjustable* to specify the range indicated in the *Refers To:* text box as adjustable cells.

Before What's*Best!* can solve a problem, it must know what cells it can change in its search for the optimal solution. These are the adjustable cells. By default, What's*Best!* assumes the adjustable cells can be set to any non-negative value.

Adjustable cells, also called decision variables, usually represent quantities or activities over which you have direct control. Traditionally, mathematicians refer to them as variables. Some examples might be:

- the number of televisions to produce
- the amount of money to be spent on advertising
- the number of shares to purchase of a particular stock

- the location of your next service facility

Inappropriate adjustable cells would be things you have no control over, such as:

- the total demand for televisions
- the price of advertising to be purchased
- the risk of a particular stock
- the local building restrictions

Cells containing equations are also inappropriate as adjustable cells. What's*Best!* cannot rewrite an equation (although the value returned by an equation will change, if the equation depends upon any adjustable cells).

What's*Best!* will not alter the contents of cells specified as adjustable that contain equations, text, or blanks. In other words, *the cells must be numeric*. Therefore, be sure to enter a numeric value in each of your adjustable cells.

Note: When the adjustable quality is applied to a cell, What's*Best!* assigns a predefined *Adjustable* style to the cell. This style is automatically made available to each of your What's*Best!* models and can be customized (e.g., font color) as you wish.

Remove Adjustable



Select *Remove Adjustable* to return an adjustable cell to its fixed (non-adjustable) state. By default, a cell is fixed until it has been specified as adjustable. When you instruct What's*Best!* to remove a cell's adjustable status, the *Adjustable* style is removed from the cell and you will see the font color revert from blue (or the font color you defined for the *Adjustable* style) to the original color.

After solving a model, you may wish use *Remove Adjustable* to fix some of the adjustable cells at their new values, turning them into constants, and solve again for the remaining adjustable cells.

Make Adjustable & Free or Remove Free

Select this to take the referenced cells and either set them as adjustable and free (capable of assuming negative as well as positive values) or remove their free status. Making this selection and clicking *OK* will post the *Free* dialog box to facilitate setting or removing the free status. For further discussion, see the section entitled *Free*.

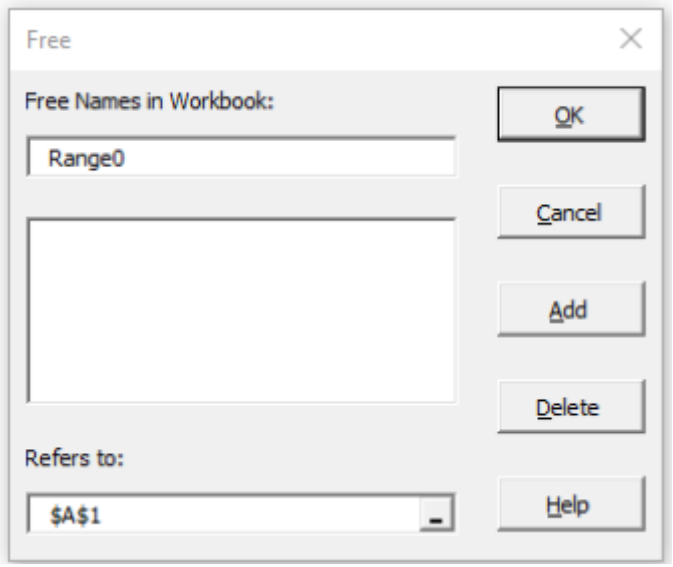
Refers To

Specify the range of cells that are to be made adjustable or fixed in this text box by using the button on the right edge to bring up a cursor for cell selection. Alternately, you can accept the currently selected cells, which What's*Best!* has automatically placed in this box. You can also type the correct cell range directly into the text box.

Note: Adjustable cells should not be locked or hidden on a protected sheet.

Free

The dialog box posted by selecting the *Make Adjustable & Free or Remove Free* command in the drop-down list in the Adjustable dialog box appears as follows:



By default, adjustable cells are restricted to being greater-than-or-equal-to zero. However, you may override this default lower bound of zero on an adjustable cell by making it free. Adjustable cells that may assume negative as well as positive values are referred to as free variables. To create an adjustable and free cell(s), specify the cell or cell range you wish to change in the *Refers to:* field. Then, enter a name in the *FREE Names in Workbook:* field. You can choose any combination of letters for your range name. Once this is done, pressing the *Add* button causes What's*Best!* to assign a *WBFREE* range name to the selected cells, which will appear in the list on the Free dialog box. For instance, if you used the name "BuySell" in the *FREE Names in Workbook:* text box, a range name *WBFREEBuySell* would be assigned to the selected cells.

If the cells are already adjustable, you can make them adjustable and free by following the steps above. Alternately, if you are removing a free variable designation for a given cell range, simply select the name of the range to remove, and press the *Delete* button.

A negative adjustable cell representing the number of televisions to produce would not be meaningful. However, there are scenarios in which allowing an adjustable cell to assume negative values would be appropriate. For example, if you have the ability to either buy or sell a stock, you could create a single adjustable cell representing your decision to buy/sell shares. A positive value would represent the amount to buy and a negative value the amount to sell.

For more information, see the *Adjustable* command above.

Best...

For optimization models, you must define at least one best cell (equation-solving and goal-seeking models do not require a best cell, as discussed below). The best cell, sometimes called the objective of optimization, is the cell whose value is to be minimized or maximized during optimization. The best cell must be an adjustable cell or a formula depending upon an adjustable cell defining exactly what it is you want to optimize.

The two most common objectives of optimization models are to minimize cost or maximize profit, but you can choose to maximize or minimize any adjustable cell or any equation that depends upon one or more adjustable cells. Other frequent targets for minimization are waste, conflict, time required, surplus, and risk. A maximization cell could be something tangible, an equation representing total output, or something less concrete, such as a calculation to estimate employee job satisfaction or the effectiveness of customer service.
In a scenario with multiple goals, you can either

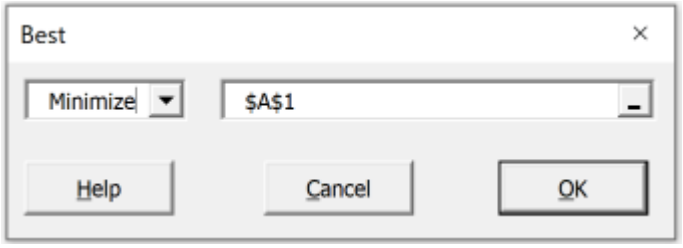
- employ the multi-objective optimization feature of What'sBest! (available for linear problems with\without integer restrictions), or
- convert the problem to a model with one objective. To do so, consider maximizing or minimizing one goal and constraining the others or combining goals into a single equation to be maximized or minimized. For example, if the goals for your factory are to maximize production output and minimize costs, there are clearly trade-offs between the two goals. You can choose to:

    1) Maximize output while constraining costs to be less than some value;
    2) Minimize costs while constraining output to be greater than some level; or
    3) Combine your goals into a single equation and maximize total value of the output minus the cost.

Both of the single and multiple objective cases are described in detail below.

Single Objective Optimization

If the user selects one cell, the dialog box posted by the *Best...* command appears as follows:



This command is used to specify the cell in your model to maximize or minimize.

Select the objective sense you wish to apply from the drop-down box on the left of the dialog box. Possible direction options are:

1) *Minimize*
2) *Maximize*
3) *None*

Then, enter the cell you wish to set in the text box on the right of the dialog box. What's*Best!* defaults to entering the current selection. Clicking *OK* will define the best cell.

Minimize or Maximize 

Select *Minimize* or *Maximize* to specify whether your objective is to determine the smallest or largest value for the best cell, respectively. These choices may also be made using the *Minimize*  and *Maximize*  toolbar buttons.

None

Select *None* to void any previously specified best cell. If no best cell is specified in a model, What's*Best!* will try to find a solution that satisfies all constraints and relationships in the model without trying to maximize or minimize any particular cell. This is referred to as goal-seeking or backsolving.

If you solve a model in which no best cell has been specified, a "No Best Cell Specified" warning appears in the status report after the solution process has completed. This warning is intended to notify the user that the specification of a best cell may have been inadvertently omitted. You may turn this warning on or off by the *No Best Cell* checkbox in the *General Options* dialog box posted by *Options...*|*General* command.

Cell range

Specify the cell to be maximized or minimized in this text box by using the button on the right edge to bring up a cursor for cell selection. Alternately, you may accept the currently selected cell, which is automatically selected by What's*Best!*. You may also type the cell range to be specified as the best cell directly into the text box.

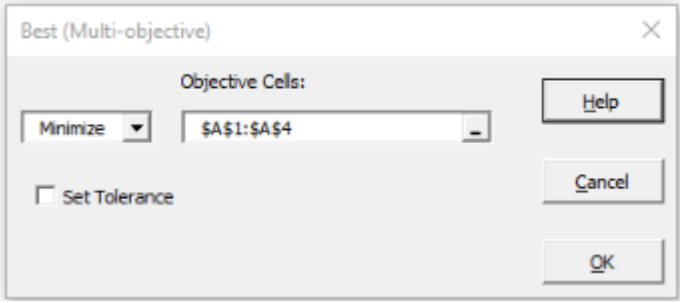
Multiple Objective (Lexico/Pre-emptive Priority) Optimization

What's*Best*! allows you to specify two or more objectives and then "Lexico Optimize" them. Suppose we are maximizing three objectives. The Lexico optimization approach is to:

+ Maximize the 1st objective;
+ constrain 1st objective to its optimal value;
+ Maximize the 2nd objective, subject to the constraint on the 1st objective,
+ constrain 2nd objective to its optimal value;
+ Maximize the 3rd objective, subject to the constraints on the previous 2 objectives,

In Goal Programming, this kind of optimization is called Pre-emptive Priority optimization.

If the user selects multiple cells, the dialog box posted by the *Best…* command appears as follows:



This command is used to specify the cells in your model to maximize or minimize and the corresponding optimality tolerances.

Minimize or Maximize

Select *Minimize* or *Maximize* to specify whether your objective is to determine the smallest or largest value for the best cell, respectively. These choices may also be made using the *Minimize* and *Maximize* toolbar buttons.

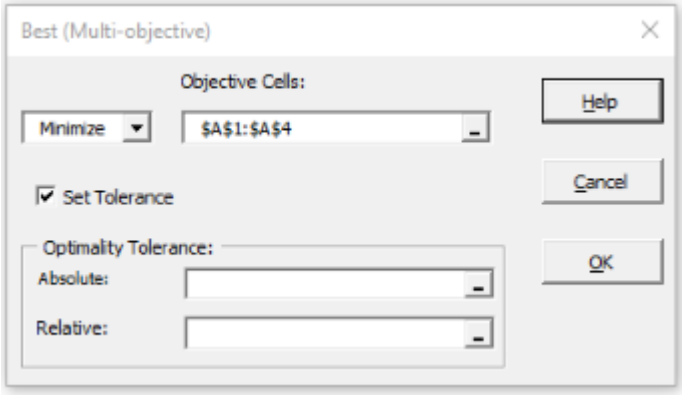
Note: In case of using the *Minimize* and *Maximize* buttons, you are not able to specify the tolerances and all selected cells are considered as the best cells.

Objective Cells:

Specify the range of cells to be maximized or minimized in this text box by using the button on the right edge to bring up a cursor for cell selection. Alternately, you may accept the default range, which is automatically selected by What's*Best!*. You may also type the range of cells to be specified as the best cell directly into the text box. The specified range should be a contiguous single row range or contiguous column range of cells. If it is a row range, the optimization order is from left to right. If it is a column range the order is from top to down. Depending upon the direction of optimization, the row range is called either WBMAXLEX or WBMINLEX.

You are also allowed to set absolute and relative tolerances for each objective. These values play two roles:

1) They are considered as the absolute and relative optimality tolerances corresponding to that objective (see Options...|Integer Solver)
2) They specify the maximum deviation allowed for this objective function from its true optimal value in the formulation of the next subproblem. Higher values allow a wider range of admissible solutions. Checking the *Set Tolerance* box displays the tolerance cells as follows:



Absolute:

You can specify the absolute optimality tolerance corresponding to each objective. You can select either nothing or a range with the same size as the best cells range.

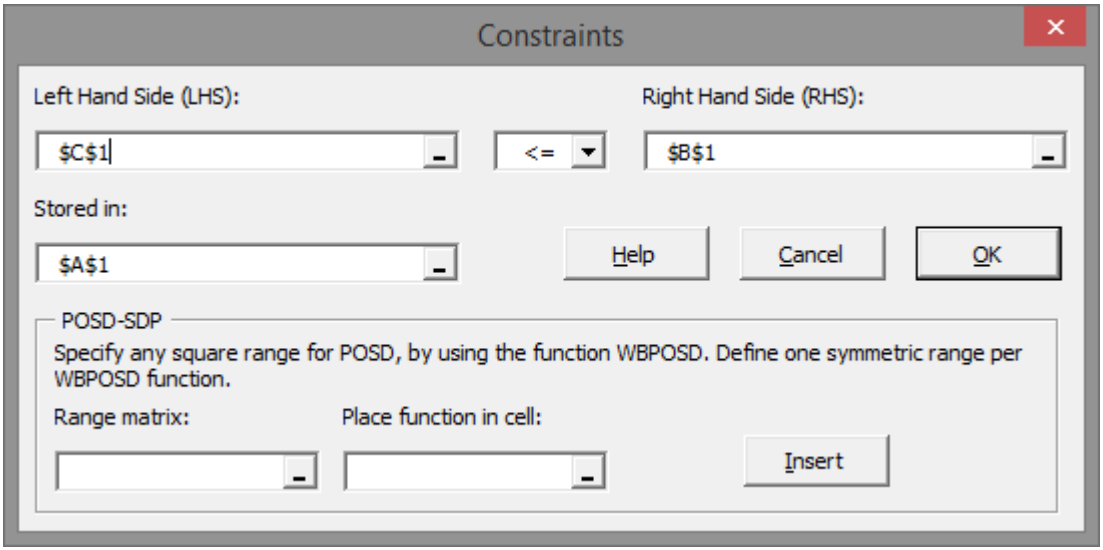
Relative:

You can specify the relative optimality tolerance corresponding to each objective. You can select either nothing or a range with the same size as the best cells range. If you select nothing, these tolerances are set to 0 by default.

During the solve process, the status window will display the objective index being optimized, its current value, and its bound.

Constraints...

The dialog box posted by the *Constraints* command appears as follows <= >= = :



This command is used to specify the resource constraints in your model.

To define a constraint, you must supply the left-hand side of the constraint, the right-hand side of the constraint, and the host cell where the constraint should appear. These go in the *Left Hand Side (LHS):*, *Right Hand Side (RHS):*, and *Stored in:* fields, respectively.

Then, select <= (less-than-or-equal-to), >= (greater-than-or-equal-to), = (equal-to), <c= (convex), >c= (concave), =c (convex), in order to specify the direction of the constraint. If no type is selected, What's*Best!* will default to <= (less-than-or-equal-to). Selecting *None* will clear the cell. Clicking *OK* will apply the constraints. The user can explicitly specify the constraint convexity to the Solver, by integrating a "c" in the direction; see the *Usage Guidelines for Constraints* part for additional information. You need be concerned with <c= or >c= only if using the Global solver.

Note:	What's*Best!* uses the loose inequality operators rather than the strict inequality operators of > (greater-than) and < (less-than).

Left Hand Side (LHS):, Right Hand Side (RHS):, and Stored in:

Specify the cells or ranges to be constrained in these text boxes. What's*Best!* attempts to automatically fill in these boxes based upon what the current selection is. The cells just to the left of the currently selected cells are entered as the Left-Hand Side of the constraint. The cells just to the right of the currently selected cells are entered as the Right-Hand Side of the constraint. The currently selected cells become the Stored in: range displayed. When a row is selected, it is assumed the cells just above the row are the left-hand side and the cells just below are the right-hand side of the equation.

If these are not the ranges of cells you want, you can specify the cells or ranges by using the button on the right edge of each text box to bring up a cursor for cell selection or type the ranges in. However, you should use the following table when selecting cells or ranges:

If the Left-hand side includes a…	then Stored In should have a…	and the Right-hand side should have a…
single cell	single cell	single cell
*range	same shape and size range	single cell
*single cell	range	same shape and size range
range	same shape and size range	same shape and size range
*You can relate a left-hand side range to a single right-hand side cell or a right-hand side range to a single left-hand side cell.		

What's*Best!* inserts a worksheet function called *WB* in the host cell(s) specified in the *Stored In:* text box (also known as the constraint cells). By clicking on one of the constraint cells and looking at the Excel® formula box, you can see the *WB* function that was created. For example, select the B1 cell, choose the *WB*|*Constraints…* command, and press *OK* on the *Constraints* dialog box to insert the formula =*WB*(A1, "<=", C1) into the cell B1. This formula indicates that the value in A1 should be less-than-or-equal-to that in C1. This could also be accomplished by selecting cell B1 and pressing the <= toolbar button or typing the =*WB*(A1, "<=",C1) formula directly into the cell.

Although it is typically easier to use the What's*Best!* commands, constraints can be created in a What's*Best!* model by copying or modifying existing constraints. What's*Best!* will treat copied constraint cells the same as any constraint cell created by the menu or toolbar *Constraints* command.

Note: It is not recommended to lock or hide constraint cells on a protected sheet.

POSD-SDP Constraint Range

To allow the use of the Semi-Definite Positive (SDP) capability with the LINDO Solver, specify the selection of the range matrix to be constrained by a Positive Semi-Definite function (POSD), as =WBPOSD(range). The range defining the matrix should be symmetric with the adjustable cells typically set to be free type (although not required), and greater than one cell.

A sample case is provided with the file "WBPOSDcase.XLSX". This case is based on a covariance matrix, and shows the lower triangular matrix, its constraints, and the non-default option keeping the quadratic recognition on to minimize the sum of squared differences. Also, the POSD feature is not compatible with nonlinear constraints and objective. To retrieve compatibility, the model needs to be convex quadratic, with the adjustable cells set to be free type in the matrix, and the Quadratic Recognition should be turned on.

Usage Guidelines for Constraints

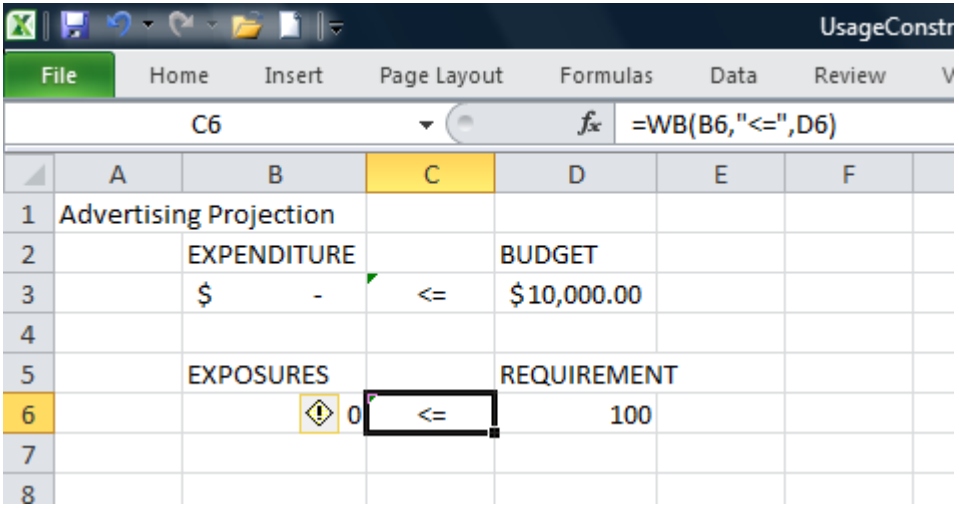
In the real world, decisions are always constrained in some fashion. In the steel business, for example, you work within a budget to acquire finite amounts of ore, coke, alloy metals and power. Orders must be delivered on time and up to specifications,or you risk losing business. Such limits to doing business are referred to as constraints in the language of optimization modeling. With What's*Best!*, you can use the operators >= (greater-than-or-equal-to), <= (less-than-or-equal-to), or = (equal-to) to express your particular constraints.

What's*Best!* allows you to enter constraints using the four following methods:

1) The *Constraints* dialog box.
2) The three constraint toolbar buttons.
3) Manually entering constraint functions as cell formulas.
4) Using VBA to automatically enter constraint functions as cell formulas.

The first two methods only allow the entry of cell references (The dialog box method offers an error-checking interface for out-of-range and other errors). The last two methods allow you the flexibility of entering complex expressions as well as cell references. To learn how to use VBA to enter constraints, see the section titled *wbConstraint*.

Let's consider the following constraints: a resource limitation constraint, "Advertising expenditures must be no more than (<=) $10,000", and a performance requirement constraint, "Advertising exposures must be at least (>=) 100".



In your What's*Best!* model, if an adjustable cell representing the advertising expenditure was in cell B3 and D3 was a fixed cell with a value of 10,000, you could enter the constraint formula =*WB*(B3,"<=",D3) anywhere in the spreadsheet to enforce the first constraint. We've entered it in cell C3, where it makes visual sense. Likewise, in cell C6, the formula =*WB*(B6,">=",D6) enforces the second constraint.

What's*Best!* would then optimize so as to satisfy these constraints. With B3 selected as the objective to minimize, the program will select at least 100 exposures that add up to the lowest possible expenditure. With B6 selected as the objective to maximize, the program will select the highest number of exposures over 100 whose summed cost is less than $10,000.

Alternatively, you could do away with cell D3 and use the formula =*WB*(B3,"<=",10000). The *Constraints* command lets you enter single-cell references in the left-hand side and single-cell references, single-cell range names, or values in the right-hand side of your formula. When entering a constraint formula manually, you have the additional option of entering an expression (which evaluates to a number, not an array) on either side of the Indicator, such as =*WB*(B3-B11,"<=", D3+D11).

Constraint Display

Using the *Constraint* option in the *Display* box under the *Options…|General* command, you can set the method of constraint display to either *Indicator* or *Slack*. If you set constraint display Indicator mode (the default), then the constraint is shown in the cell by one of the following eight visual indicators: **<=**, **>=**, **=<=**, **=>=**, **Not <=**, **Not >=**, **Not =**, or **=**. If you set What's*Best!* to display the constraint in *Slack* mode, then a numeric value known as the *slack* is returned. The slack is the amount by which a constraint is not violated.

For example, What's*Best!* would display constraint values based upon the following table:

Constraint	Slack Mode	Indicator Mode
Tightly satisfied	0	=<=
Satisfied	*positive number*	<=
Not satisfied	*negative number*	Not <=

Constraint-related Problems

Constraints must be properly formulated or they can be a source of problems. Some of the problems are discussed below.

Too Few Constraints - Unbounded Solutions

Let's return for a moment to the XYZ Production model presented in the *Tutorial* of Getting Started. Had you forgotten to include constraints in the model and tried to solve, What's*Best!* would have given an error message indicating Solution Status: UNBOUNDED in the WB! Status worksheet. An unbounded solution error occurs when a model is not properly constrained. Profit could be increased without limit by producing an infinite number of computers! What's*Best!* knows you've left out one or more constraints and indicates that the problem is unbounded.

In an actual production situation, there may be constraints other than raw material or component availability, such as limited plant capacity or labor supply. Excluding any of these could lead to an unbounded solution. Also, What's*Best!* might give you an unbounded error message if you maximize the wrong cell, or maximize the objective cell when in fact you really wanted to minimize it.

Anytime What's*Best!* finds the objective heading toward infinity, you'll receive the Solution Status: UNBOUNDED error message. For details, see the topic *UNBOUNDED* in Troubleshooting.

No Feasible Solution Found - Linear

What if you add a constraint that contradicts one or more of the existing constraints? In the XYZ problem, if you add a requirement that the number of Deluxe computers to be produced is greater than 60, What's*Best!* will return the Solution Status: INFEASIBLE message in the "WB! Status" worksheet. The original constraint requires that no more than 50 Deluxe computer towers be used. However, this new constraint requires production of at least 60 Deluxe computers, exceeding the limitation of the first constraint.

Anytime What's*Best!* can't find a solution to a linear problem that satisfies all constraints, you'll see the *INFEASIBLE* error message. By examining the state of the constraints in the returned infeasible solution, you may be able to determine how to reformulate and eliminate the source of infeasibility.

No Feasible Solution Found - Nonlinear

In a nonlinear problem, the *Solution Status: INFEASIBLE* message may not mean that there is no solution that satisfies all constraints. It only indicates that none can be found from the starting values you've entered in the adjustable cells. If you know or suspect there is a feasible answer to the problem, enter adjustable cell values that yield a feasible or near-feasible solution and try solving from that starting point. Alternatively, using the Global Solver option may help.

One other possible course to follow is to view the constraints in slack mode rather than Indicator mode. This helps you determine which constraints are furthest from being satisfied. Then, enter "educated guesses" for the adjustable cells, values that cause the constraints to come closer to being satisfied, and try solving again from these starting values.

Explicitly Specifying Convexity to the Solver

One of the important considerations for the solver in finding a global optimum is the exploitation of convexity. In layman's terms, if an optimization problem is convex, then a carefully implemented "hill climbing" search algorithm (or valley finding for a minimization problem) will find a global optimum. More generally, if the feasible region is convex, then any strict local optimum can justifiably be declared a global optimum. Our solver is fairly sophisticated in its ability to identify convex expressions. Nevertheless, there may be some constraints that you know have convex feasible regions but you suspect might not be identified as convex by our solver. What's*Best!* allows special constraint types, "<c=", ">c=", and "=c" to allow users to identify constraints that have convex feasible regions. For example, suppose we have the constraints:

x^3 - 2*x*y + y^3 <= 6;

x >= .5;

y >= .5;

The feasible region of this set of constraints happens to be convex. The user is justified in writing:

x^3 - 2*x*y + y^3 <c= 6;

More generally, if $f(x)$ is any function on the left hand side and the user writes:

$f(x)$ <c= b;

The solver will assume that the feasible set for this constraint is convex. This is true for example if $f(x)$ is a convex or quasi-convex function. Loosely speaking, a quasi-convex function is uni-modal; if you pour water into it, only one puddle would form. A convex function is also quasi-convex.

If you have the constraint $f(x) >= b$, and you know that $f(x)$ is a concave function then you are allowed to write:

$f(x)$ >c= b;

Finally, if you have the constraint $f(x) = b$, and you know that $f(x)$ is a quasi-convex function then you are allowed to write:

$f(x)$ =c= b;

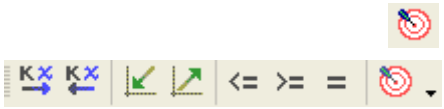
The interpretation is that we require $f(x) = b$, however, the feasible region to the relaxation, $f(x) <= b$, is convex.

Similarly, if you have the constraint $f(x) = b$, and you know that $f(x)$ is a quasi-concave function then you are allowed to write:

$-f(x)$ =c= $-b$;

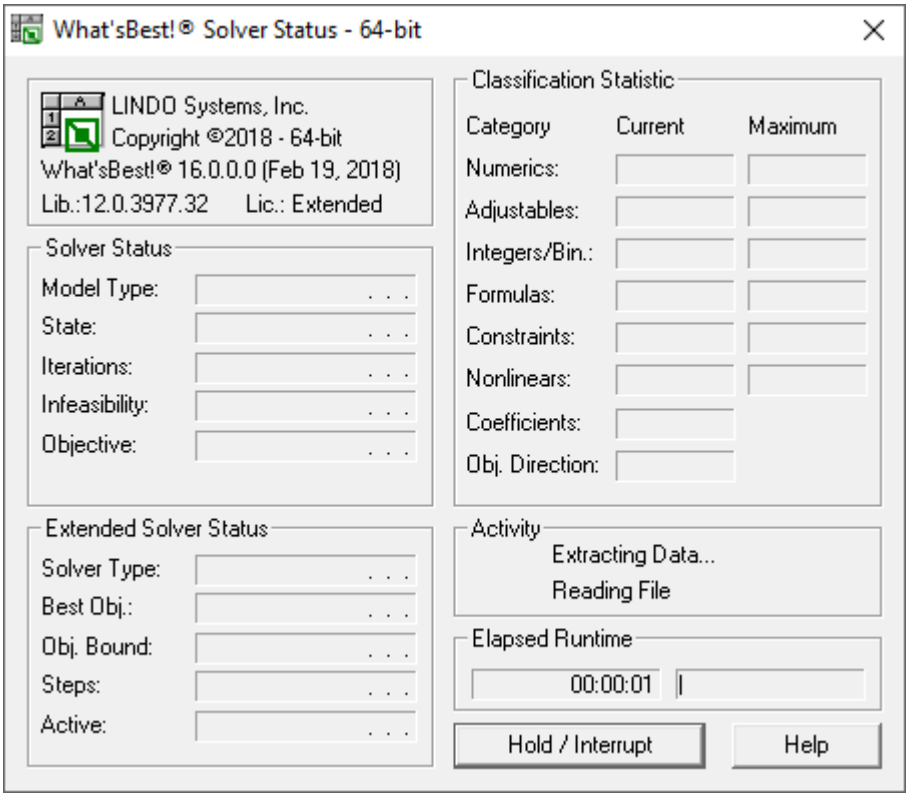
The interpretation is that we require $f(x) = b$, however, the feasible region to the relaxation, $-f(x) <= -b$, or, $f(x) >= b$, is convex.

Solve *or*



on the What's*Best!* toolbar

Once the ABC's (Adjustable, Best, and Constraint cells) have been specified, you are ready to solve your model. Choosing the *Solve* command prompts the What's*Best!* solver to begin examining and solving your model. During the time that the solver is computing, it displays the *Solver Status* window, which appears as shown below.



The solver status window remains present for the duration of the solution process, and is updated periodically. Once the solver completes its run, it returns to the workbook to install a new solution and inserts a new worksheet entitled *WB! Status*. This new worksheet is referred to as the status report.

The status report is an easily understood analysis of the model and solution. It provides a hard copy of the information appearing in the solver status window. The section entitled *Getting the Best Results* discusses the information contained in the status report in further detail.

You may use the *InterruptSolver* button on the solver status window to interrupt the solver while it is solving. If you do, you can not expect any useful information to be returned unless you have an integer

problem. In the case of an integer problem, the best solution to the point of interruption is returned. For linear problems and non-linear problems, the returned values will not be useful.

The definitions of the information in the What's*Best!* solver status box are shown below.

Model Type: This shows one of six categories your model falls into: *Integer/Quadratic, Integer/Nonlinear, Integer/Linear, Quadratic, Nonlinear, Linear*. For further information on these categories see the section entitled *Linear vs. Nonlinear Expressions and Linearization* in *Overview of Mathematical Modeling*.

State: This shows the state of the current solution. Possible states include *Globally Optimal, Infeasible, Unbounded, Feasible, Infeasible or Unbounded, Near Optimal, Locally Optimal, Locally Infeasible, Cutoff, Numerical Error, Undetermined, Unknown Error*. (See *Overview of Mathematical Modeling* for more information on these states).

Tries: This shows the current number of tries or iterations used in solving the model. The value is updated periodically in the course of solving.

Infeasibility: This shows the total amount by which all constraints are violated. When this value is reported as zero, all constraints are satisfied. However, on integer models, all integer restrictions may not be satisfied.

Objective: This shows the current value of the cell to be maximized or minimized.

Solver Type: This shows the type of specialized solver in use, if any, and will be either *Global*, *Branch-and-Bound*, or *Multistart*.

Best Obj.: This shows the objective value of best feasible solution found.

Obj. Bound: This shows the theoretical bound on the objective for integer or nonlinear programming models. What's*Best!* initially determines this value by solving the problem without integer restrictions. It then tightens this bound as it searches more branches of the solution tree. The objective bound is useful information in that it is a limit on how good a value the objective can attain. On long-running models, you might want to interrupt the solver once the best solution is sufficiently close to the theoretical limit.

Steps: This shows the number of solver iterations.

Active: This shows the number of pending subproblems to be solved.

Numerics: This shows the total number of numeric cells in the model.

Adjustables: This shows the total number of adjustable cells in the model.

Integers/Bin.: This shows the number of adjustables cells with integer/binary restrictions.

Formulas: This shows the number of formula cells.

Constraints: This shows the total number of constraint cells in the model.
Nonlinears: This shows the number of adjustable cells that appear nonlinearly in the model. We say that a variable appears nonlinearly in a formula when that formula is nonlinear with respect to changes in the variable. For instance, consider the nonlinear expression *X+Y^2*. This expression is nonlinear with respect to *Y*, but it is linear with respect to *X*. Thus, *Y* would be included in the nonlinear count, while *X* would not.

Coefficients: This shows the total number of coefficients in the best cell and all formulas dependent on adjustable cells. Mathmaticians refer to this as the number of nonzero elements.

Obj. Direction: This shows the direction of the best cell.

Activity: After the Verifying part, this shows the stage of the solution process the solver is currently performing.

- ♦ Extracting Data… / Check license, copy file
- ♦ Extracting Data… / Reading file
- ♦ Extracting Data… / Storing relevant formulas
- ♦ Extracting Data… / Creating Instruction List
- ♦ Building the Model…
- ♦ Solving…
- ♦ *Debugging... / Finding Infeasible Cells or Finding Unbounded Cells* if Debugger requested
- ♦ Writing Solution…

During the last stage, What's*Best!* writes the solved values of the adjustable cells directly into your spreadsheet and writes the requested reports.

Elapsed Runtime: This shows the length of time the solver has been running in hours, minutes, and seconds.

Note:	Variables or Optimizables is defined as the total number of adjustable cells, constraint cells, and cells containing formulas dependent on adjustable cells that are used by the objective function.

Getting the Best Results

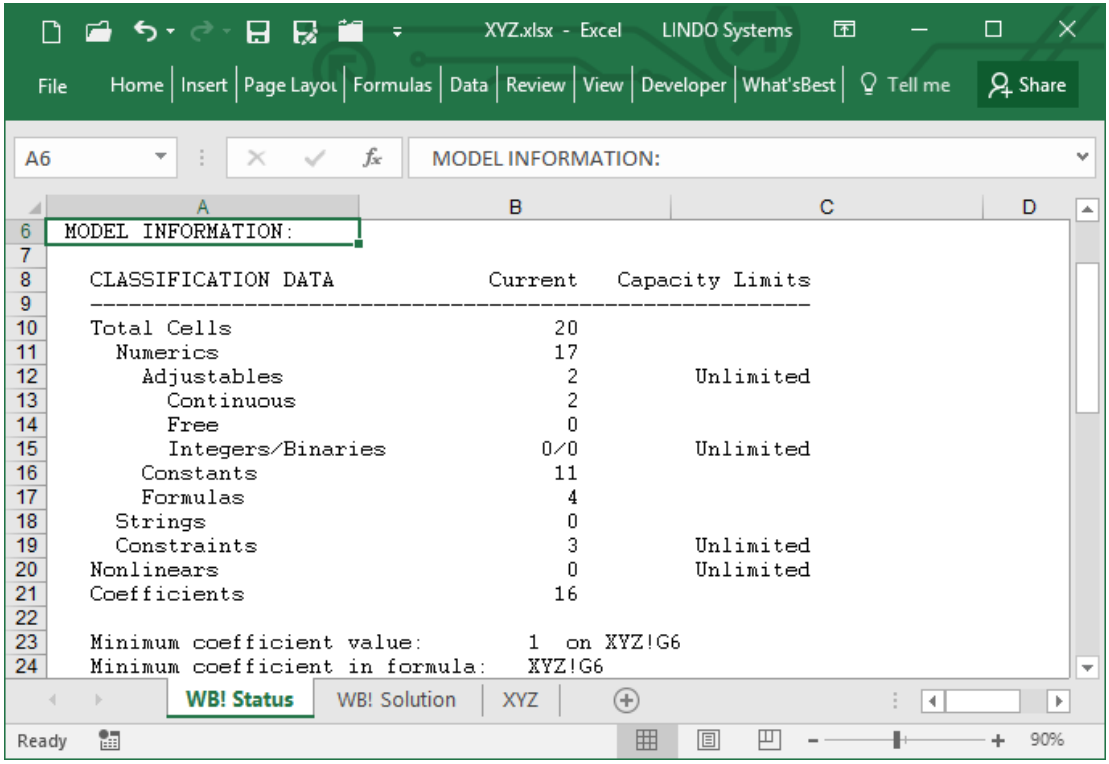
You can obtain the results of your call to solve the model by either examining the values displayed in the spreadsheet or by looking at the status report (*WB! Status*) and/or the solution report (*WB! Solution*).

In some cases, such as when you have a pretty good estimate of what the objective should be, you might feel satisfied after inspecting the best cell. However, even under such ideal conditions, it is a good idea to confirm that your model solved successfully by examining the status report. The status report provides detailed feedback about your model and the solution outcome.

If you wish a more detailed view of your model, you may request the solution report as well. The solution report provides a detailed analysis of the adjustable, best, and constraint cells, showing the initial and final values, as well as the locations of the cells. The generation of the status report and solution report is controlled by settings at the bottom of the *General Options* dialog box.

If an error or warning situation was encountered that prevented the solver from running to completion, then, by default, the solver opens up the status report to give you the error or warning message. This message describes the problem the solver encountered and offers suggestions for correcting it. The user may change the error and warning situations under which the status report is automatically opened at the bottom of the *General Options* dialog box.

The status report generated by solving the completed sample model *XYZ.XLS* is shown below:




```
MODEL INFORMATION:

   CLASSIFICATION DATA              Current    Capacity Limits
   ------------------------------------------------------------
   Total Cells                         20
     Numerics                          17
       Adjustables                      2          Unlimited
         Continuous                     2
         Free                           0
         Integers/Binaries            0/0          Unlimited
       Constants                       11
       Formulas                         4
     Strings                            0
     Constraints                        3          Unlimited
   Nonlinears                           0          Unlimited
   Coefficients                        16

   Minimum coefficient value:         1  on XYZ!G6
   Minimum coefficient in formula:    XYZ!G6
```

The status report is organized as follows. First, final values for the fields appearing in the solver status window are presented. At its conclusion, the status report appends critically important information such as the solution status. The fields in the report and the significance of the returned values are as follows:

Classification Data:

Total Cells: This shows the total number of numeric cells in the model. These are the cells useful to the model, containing the Numerics, Strings, and the Constraints.

Numerics: This shows the number of cells displaying a numeric value. These are Adjustable, Constant, and Formula cells.

Adjustables: This shows the total number of adjustable cells in the model.

Continuous: This shows the number of Adjustable cells with a continuous range [0, +infinity[.

Free: This shows the number of Adjustable cells with a free range,]-infinity, +infinity[.

Integers: This shows the number of Adjustable cells with integer restrictions on a range [0, +infinity[.

Binaries: This shows the number of Adjustable cells with binary restrictions [0,1].

Constants: This shows the number of cells containing a constant number, or an unsupported function converted into its constant value. This does not count the constant arguments within a formula.

Formulas: This shows the number of cells containing a usefull formula to the model. This does not count the constant arguments within a formula.

Strings: This shows the number of cells containing a string of characters, or an unsupported function converted into its string value. This does not count the string arguments within a formula.

Constraints: This shows the total number of constraint cells in the model.

Nonlinears: This shows the number of adjustable cells that appear nonlinearly in the model. We say that a variable appears nonlinearly in a formula when that formula is nonlinear with respect to changes in the variable. For instance, consider the nonlinear expression X + Y^2. This expression is nonlinear with respect to Y, but it is linear with respect to X. Thus, Y would be counted in the nonlinear count, while X would not.

Coefficients: This shows the total number of coefficients in the best cell and all formulas dependent on adjustable cells.

Variables: This shows the total number of adjustable cells, constraint cells, and cells containing formulas dependent on adjustable cells that influence the best cell. This information appears after an automatic linearization.

Minimum and Maximum coefficients: These are the largest and smallest absolute values of coefficients appearing in the model. When the ratio of these two values becomes very large the model is said to be poorly scaled. Poorly scaled models can lead to numerical round off errors in the solver.

These values are supplied with their location to assist you in tracking them down. Note that "<RHS>" signifies the Right Hand Side of a constraint. If the location given is "…", this means that no specific cell location is available and that the coefficient appears in a constraint added to the model as part of the linearization option.

Model Type: What's*Best!* examines your model to determine which of six types your model falls under: *Integer/Quadratic, Integer/Nonlinear, Integer/Linear, Quadratic, Nonlinear, Linear*. For further information on these categories, see the section entitled *Linear vs. Nonlinear Expressions and Linearization.*

Solution Status: This field gives the status of the final solution. The list of possible states is *Globally Optimal, Infeasible, Unbounded, Feasible, Infeasible* or *Unbounded, Near Optimal, Local Optimal, Locally Infeasible, Cutoff, Numerical Error, Unknown, Unloaded, Loaded* and *Unknown Error*. (See *Solution Outcomes* in *Overview of Mathematical Modeling* for more information).

Optimality Condition: This shows the optimality condition for nonlinear models, either *Satisfied* or *Uncertain* (See *Solution Outcomes* for more information).

Objective Value: Assuming your model contains a best cell to be maximized or minimized, this field displays its final value.

Direction: Assuming your model contains a best cell, this field displays its direction. The direction will be listed as either *Maximize* or *Minimize*.

Solver Type: This shows the type of any extended solver in use over and above the standard linear and nonlinear solvers. Possible extended solvers are: *Global*, *Branch-and-Bound*, or *Multistart*.

Tries: This shows the current number of tries (or iterations) needed to solve the model.

Infeasibility: This shows the total amount by which all constraints are violated. When this value is reported as zero, all constraints are satisfied. However, on integer models, all integer restrictions may not be satisfied.

Best Objective Bound: Assuming a best cell is present, this shows the theoretical bound on the objective for integer programming models or any model solved with the global solver.

Steps: This shows the number of solver steps required by any extended solver.

Active: This shows the number of pending subproblems to be solved by any extended solver.

Solution Time: This shows the length of time the solver has been running in hours, minutes, and seconds.

Additional detailed information about your model can be found by generating a solution report. You can enable the solution report option via the *Solution Report* drop-down box on the *General Options* dialog box.

3 Additional Commands

What's*Best!* has a number of additional commands to allow you to do such things as perform sensitivity analysis, identify integers variables, change the type or appearance of a report, or locate a particular type of cell. Others will wish to tailor the performance of the solvers by manipulating various option settings. This chapter discusses these additional commands and options.

Options and Solvers

Some options in What's*Best!* pertain to the overall operating parameters of the What's*Best!* solvers or to the user interface and data display. These general options (such as *Iteration limit* or *Runtime Limit*) are set in the *General Options* dialog box. Other options modulate the performance of a particular solver. For example, *Model Reduction* is a linear solver option and is set in the *Linear Solver Options* dialog box.

For most models, the default option settings should offer the best performance in a standard running environment. However, as you build more complex models, you may benefit from trying new option settings or be required to change the settings to get a valid of good solution. In order to effectively use the options, the user should understand the four-solver architecture of What's*Best!* and the types of options that are available.

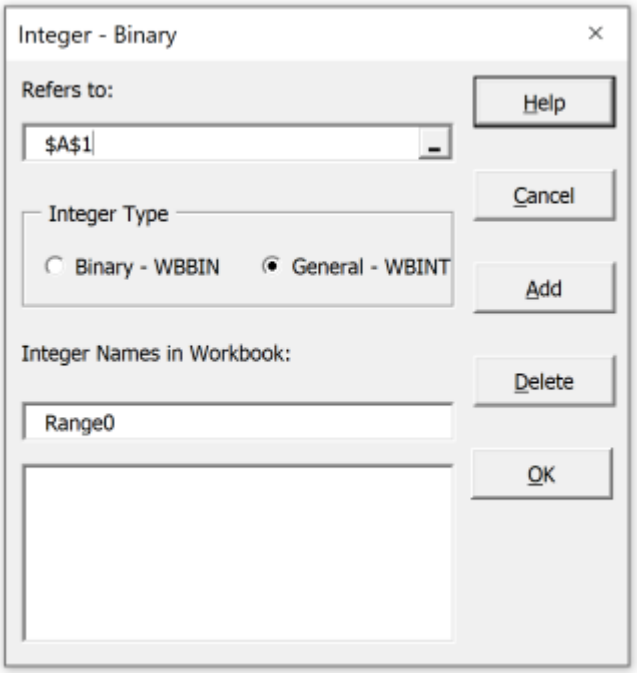
The four classes of solvers available in What's*Best!* are linear, nonlinear, global, and integer. The downloadable and solver suite versions of What's*Best!* are supplied with a linear solver, a nonlinear solver, global solver, and an integer solver. Larger versions of What's*Best!* are equipped with linear and integer solvers only. A nonlinear solver and a global nonlinear solver are provided optionally for additional fees. The linear solver generally runs in primal simplex mode, but it can be set to dual simplex or the barrier method (if the barrier option is purchased).

What's*Best!* options apply only to the current model. That means options are defined as Excel® names in the workbook and are not saved for other workbooks. The What's*Best!* option names usually start with ‘WBxxx” and thus are separated alphabetically from the user defined names. The *Reset to Default* command resets the workbook options, but not options defined in the *Advanced Parameters* window. The workbook options are *General Options*, *Linear Solver Options*, *Nonlinear Solver Options*, *Global Solver Options*, *Integer Pre-Solver Options*, *Integer Solver Options*, and *Stochastic Solver*.

The algorithms used by the solvers are as follows. The linear solver in What's*Best!* uses either a primal simplex, dual simplex, or barrier method. The *Solver Method* default setting of *Solver decides* selects the primal simplex method. The What's*Best!* nonlinear solver uses a generalized reduced gradient (GRG) algorithm. The What's*Best!* global solver combines a series of range bounding techniques within a brand-and-bound framework to find global solutions to non-convex NLPs. Finally, the integer solver provided in What's*Best!* uses a branch-and-bound algorithm, also referred to as the branch-and-bound manager.

Integers…|Integer-Binary...

The *Integers...|Integer-Binary* command allows you to restrict your adjustable cells to being integers. What's*Best!* allows you to define integers in either *binary* or *general* format. A binary integer will return a zero or one, whereas any nonnegative, whole number {0,1,2,3…} will be returned for a general integer. The *Integer* dialog box appears as follows:



To create an integer range, specify the cell or cell range you wish to make integer in the *Refers to:* text box. Then, enter a name in the *Integer Names in Workbook:* text box. Next, select the *Binary - WBBIN* or *General - WBINT* radio button in the *Integer Type* box. Once this is done, pressing the *Add* button causes What's*Best!* to assign a *WBBIN* or *WBINT* range name to the selected cells, which will appear in the list on the *Integer* dialog box.

If you decide later you would like to remove the integer restriction, simply select the *Integer* command, click on the name of the integer range to remove, and press the *Delete* button. It is important to remember that integer restrictions can increase solution times as discussed in the section entitled *Runtime Concerns*.

Refers to

Specify the range of cells that are to be made integer in this text box. What's*Best!* will automatically fill the *Refers to:* text box in with the range of the currently selected cells. If this is not the range of cells you want, you can type the correct range in or select the button on the right edge of the text box to bring up a cursor for cell selection.

Binary-WBBIN

Select the *Binary* radio button in the *Integer Type* box to cause What's*Best!* to constrain the current range of cells specified in the *Refers to:* text box to have a value of zero or one. Binary integer variables (0/1) are useful in making Yes/No, Open/Close, or Buy/Sell decisions and formulating piecewise linear functions. If a cell is specified as binary, What's*Best!* will find the best solution that returns a 0 or 1 in that cell.

General-WBINT

Select the *General* radio button in the *Integer Type* box to cause What's*Best!* to constrain the range of cells specified in the *Refers To:* text box to be a positive, whole number. General integer variables (0,1,2…) can be useful when answers with fractions are of limited or no value. Some examples are personnel scheduling, buy/sell decisions involving round lots, and discrete loading models. If a cell is specified as general integer, What's*Best!* will find the best solution that returns a whole number in that cell.

Integer Names in Workbook

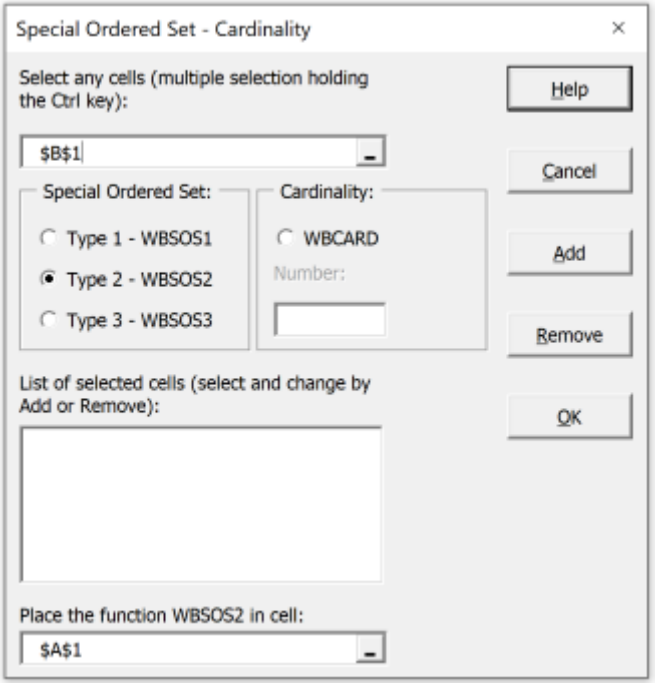
What's*Best!* uses range names to specify which cells in a model should be integers. You can choose any combination of letters for your range name. When the range name is created, What's*Best!* adds *WBBIN* or *WBINT* to the front of the name to represent, respectively, binary integers and general integers. For instance, if you used the name "Quantity" in the *Integer Names in Workbook:* textbox for a general integer, a range name *WBINTQuantity* would be assigned to the selected cells and appear in the list on the *Integer* dialog box.

Runtime Concerns in Integer Problems

The use of integer variables can considerably increase the time required to find optimal solutions. In most nonlinear models that include integer variables, your chances of reaching an optimal solution in a reasonable amount of time are diminished further, unless the problem is extremely rudimentary or constrained in very specific ways. To improve performance, you may wish to use options such as *Tolerance* and/or *Hurdle (Known IP)*. Such options can significantly decrease the solution time on some integer problems. For details on these options, see the section entitled *Options...*|*Integer Solver*.

Integers...|Special Ordered Set

The *Integers...|Special Ordered Set* command allows you to support Special Ordered Sets (SOS) variables of type 1, 2 and 3, and Cardinality sets of variable via the following dialog box:



Select Any Cells

This reference field allows you to select any of the Adjustable or Variable cells that will be part of the set. You can select multiple range by holding the Control (Ctrl) key. Then click on Add to store them in the List of Selected Cells field.

Special Ordered Set

The properties of the three SOS types are, via the WBSOSx function:

SOS Type	***Property***
WBSOS1	At most one variable belonging to an SOS1 set will be > 0.

WBSOS2	At most two variables in an SOS2 set can be > 0. If two variables are nonzero, then the variables will be adjacent to one another. SOS2 sets are particularly useful for implementing piecewise-linear functions in models.
WBSOS3	Exactly one variable from a given SOS3 set will be equal to 1. All remaining variables will be equal to 0.

The syntax for the WBSOSx declarations is as follows, written in the selected cell of the spreadsheet:

=WBSOS1(range reference, ...)

Note: Each variable in an SOS set counts against the integer variable limit imposed in limited versions of What's*Best!*. SOS sets are supported for linear models only.

Cardinality

Related to the SOS capability discussed above, What's*Best!* also supports cardinality sets of variables via the WBCARD function. The cardinality feature allows you to specify a set of variables with a cardinality of *N*, meaning that, at most, *N* of the variables in the set will be allowed to be nonzero.

As with SOS sets, cardinality sets help the integer solver branch more efficiently, and they reduce the number of variables and constraints in your models.

The syntax for the WBCARD declarations is as follows, written in the selected cell of the spreadsheet:

=WBCARD(number, range reference, ...)

Note: Each variable in a CARD set counts against the integer variable limit imposed in limited versions of What's*Best!*. CARD sets are supported for linear models only.

List of Selected Cells

This field will list all the cells that are part of an SOS or CARD set. You can modify the list by clicking on Add or Remove buttons.

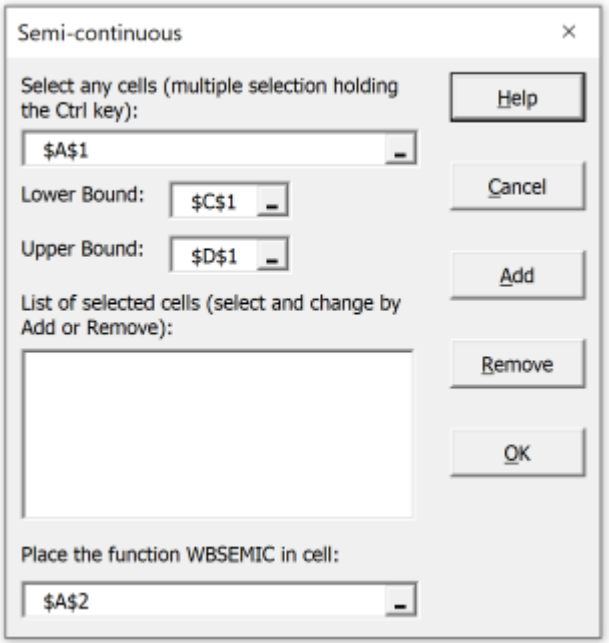
Place the Function in Cells

Specify the host range of cell you would like to see the SOS or CARD function to be placed in the spreadsheet. What's*Best!* automatically fills this box with the current cell selection.

If the selection already contains a WBSOSx or a WBCARD function, then What's*Best!* will display the cells argument into the List of Selected Cells.

Integers...|Semi-continuous

The *Integers...|Semi-continuous* command allows you to support Semi-continuous variables via the following dialog box:



Many models require certain variables to either be 0 or lie within some nonnegative range, e.g., 10 to 20. Variables with this property are said to be semi-continuous. Modeling semi-continuity in What's*Best!* in the past meant having to add an additional 0/1 variable and two additional constraints.

The syntax for the WBSEMIC declaration is as follows, written in the selected cell of the spreadsheet:

=WBSEMIC(lower bound, upper bound, range references, ...)

This will restrict the variable in the range references, to be either 0 or to lie within the range [lower_bound, upper_bound].

Note: Each semi-continuous variable will be counted against any integer variable limit for your installation.

Select Any Cells

This reference field allows you to select any of the Adjustable or Variable cells that will be part of the set. You can select multiple ranges by holding the Control (Ctrl) key. Then click on Add to store them in the List of Selected Cells field.

Lower Bound, Upper Bound

This field allows you to select the lower and the upper bounds of the Adjustable cells of the set. You can select numeric values, or cells references. These references cannot be variable numbers.

List of Selected Cells

This field will list all the cells that are part of a Semi-continuous set. You can modify the list by clicking on Add or Remove buttons.

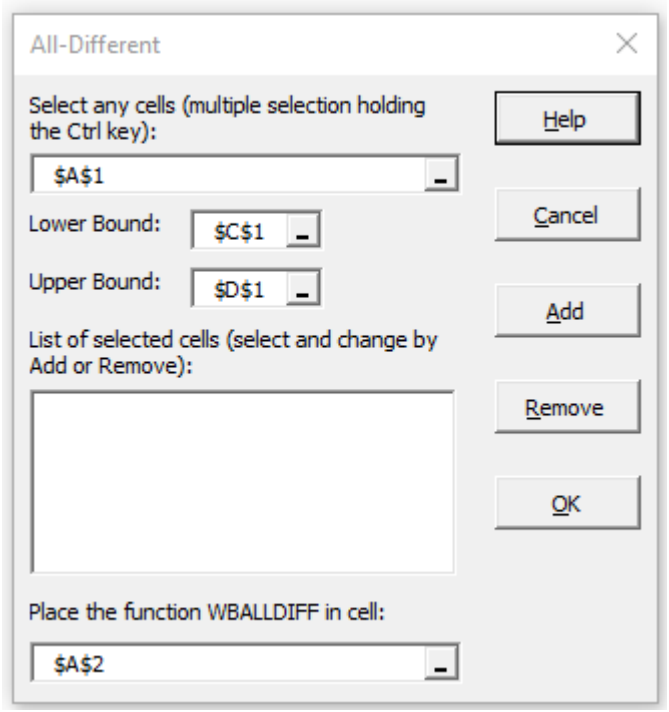
Place the Function in Cells

Specify the host range of cell you would like to see the SEMIC function to be placed in the spreadsheet. What's*Best!* automatically fills this box with the current cell selection.

If the selection already contains a WBSEMIC function, then What's*Best!* will display the cells argument into the List of Selected Cells.

Integers...|All-different

The *Integers...|All-different* command allows you to support All Different variables via the following dialog box:



Many models require certain integer variables to be all different and lie within some range. Variables with this property are said to be All Different. Modeling with all differences in What's*Best!* in the past meant having to add an additional integer variables and constraints.

The syntax for the WBALLDIFF declaration is as follows, written in the selected cell of the spreadsheet:

=WBALLDIFF(lower bound, upper bound, range references, ...)

This will restrict the variable in the range references, and lie within the range [lower_bound, upper_bound].

Note:	Each all-different variable will be counted against any integer variable and constraint limits for your installation.

Select Any Cells

This reference field allows you to select any of the Integer Adjustable or Variable cells that will be part of the set. You can select multiple ranges by holding the Control (Ctrl) key. Then click on Add to store them in the List of Selected Cells field.

Lower Bound, Upper Bound

This field allows you to select the lower and the upper bounds of the Adjustable/Integer cells of the set. You can select numeric values, or cells references. These references cannot be variable numbers.

Cells reference must also be declared as Integer or Binary.

List of Selected Cells

This field will list all the cells that are part of a All-different set. You can modify the list by clicking on Add or Remove buttons.

Place the Function in Cells

Specify the host range of cell you would like to see the ALLDIFF function to be placed in the spreadsheet. What's*Best!* automatically fills this box with the current cell selection.

If the selection already contains a WBALLDIFF function, then What's*Best!* will display the cells argument into the List of Selected Cells.

A sample case is provided with the file "WBALLDIFF_Sudoku.xlsb". This model shows the Sudoku puzzle case, where the user fills out a 9x9 grid with the digits 1, 2, … , 9 so each digit appears once in:

a) each column,
b) each row,
c) each of the nine 3x3 subsquares.

All Adjustable cells should be Integers, and some of the cell entries may be pre-specified as constant.

Options...

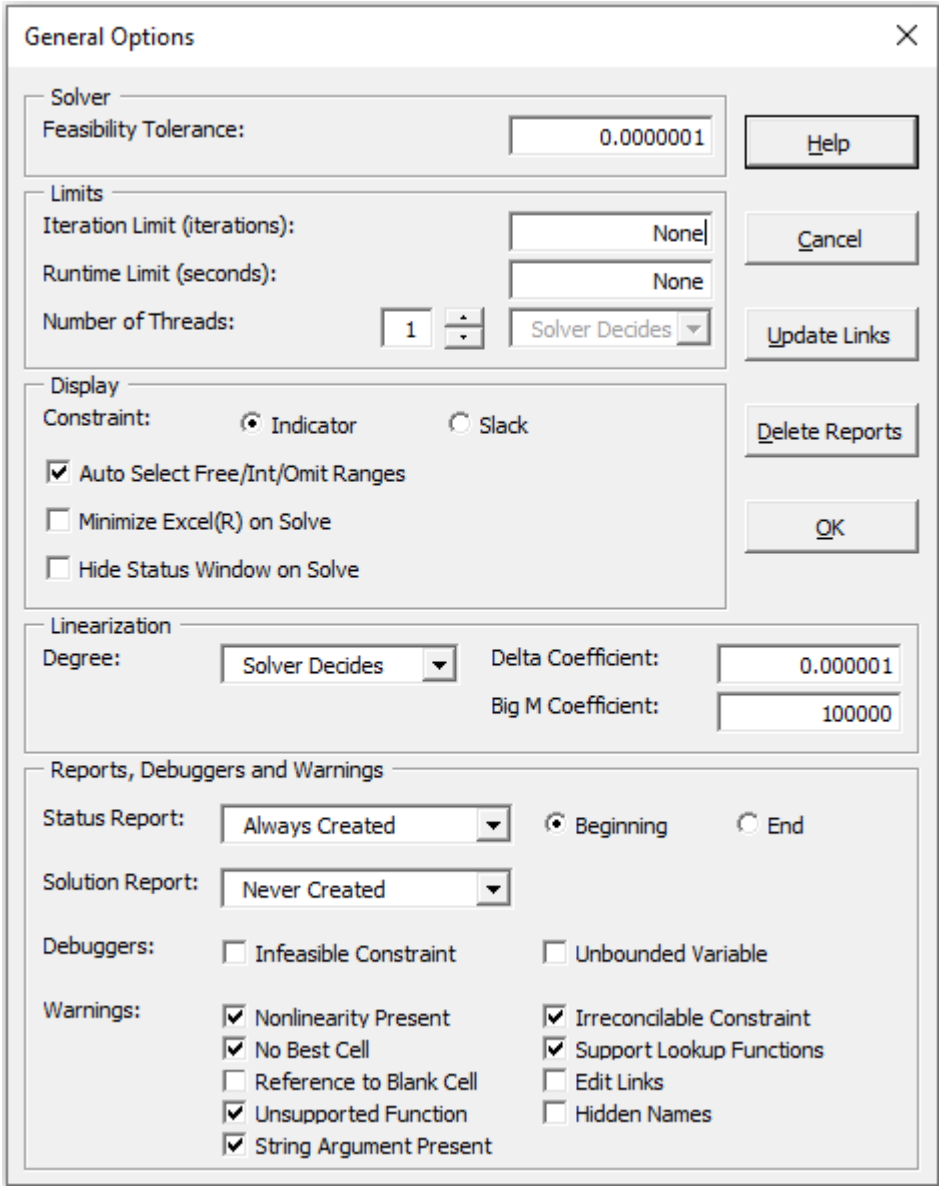
The *Options* command allows you to customize the operating parameters of the What's*Best!* solvers, the user interface, how What's*Best!* runs and how your model is displayed. There are several sub-commands under the *Options* command on the *WB!* menu. They are:

- General...
- Linear Solver...
- Nonlinear Solver...
- Global Solver...
- Integer Pre-Solver...
- Integer Solver...
- Stochastic Solver
- Reset to Default

From the dialog boxes posted by these commands, you can set the options by simply selecting the options you would like and clicking the *OK* button. If you wish to restore the options to their default values, you should use the *WB!*|*Options…*|*Reset to Default* command to return all What's*Best!* options to their default values.

Options...|General

The dialog box posted by the *Options...*|*General* command appears as follows:



The *General Options* dialog box contains options that let you control the ways in which What's*Best!* displays, processes, and saves information. All of the options in the *General Options* dialog are saved with the workbook and are therefore called workbook options. All workbook options are reset to default settings every time you create a new workbook. You may also use the *Reset to Default* command to return all workbook options in the current model to their default values, with the exception of the Advanced Parameters options. Discussion of the *General Options* follows below.

Feasibility Tolerance

You can specify the amount of violation that will be tolerated in a model's constraints with the *Feasibility Tolerance* text box. Increasing the feasibility tolerance can allow you to find a feasible solution to a model that was previously infeasible. For example, changing a feasibility tolerance would be useful if you have a poorly scaled model with very large and very small coefficients that is nearly feasible (You would know that the model is nearly feasible if you found the slack values to be small). The best solution would be to rescale the model, but setting the feasibility tolerance offers a simpler alternative. By entering a larger feasibility tolerance, you allow the solver to tolerate slight constraint violation, such as a few pennies in a budget of tens-of-thousands, in order to reach a feasible solution.

The default value for feasibility tolerance is 0.0000001.

Iteration Limit

You can specify a limit on the number of tries or iterations undertaken during the solution process with the *Iteration Limit (Iterations)* textbox. If this limit is encountered before a solution has been found, then the solution search is terminated and, if the model contains integer variables, the best integer solution found up to that point is returned. If this limit is encountered and the model does not contain integer variables, the returned solution is meaningless. Please note that returning to any previously-established best solution may take some time.

The default value is *None*, signifying that no iteration limit is imposed.

Runtime Limit

You can specify the maximum amount of time in seconds the solver should spend searching for a solution with the *Runtime Limit (seconds)* text box. If this runtime limit is encountered before a solution has been found, then the solution search is terminated and, if the model contains integer variables, the best integer solution found up to that point is returned. If this limit is encountered and the model does not contain integer variables, the returned solution is meaningless. Please note that returning to any previously established best solution might take some time.

The default value is *None*, signifying that no runtime limit imposed.

Constraint Display

Select Slack or Indicator to prompt What's*Best!* to return the slack value or indicator display for the constraint cells.

Indicator mode places the sign of the equation (>=, <=, =) in the constraint cell. If an indicator is tightly satisfied (i.e., the constraint contains no slack), then What's*Best!* will return an equal sign before the indicator (i.e., "=<="). If an indicator is not satisfied at all, What's*Best!* will return a "Not" before the indicator (i.e., "Not =").

Slack mode displays the slack value of each constraint. The slack value of a constraint tells you how close you are to satisfying the constraint. A positive slack is the amount by which the constraint is overly-satisfied. A zero slack value means the constraint is exactly satisfied. A negative slack is the amount by which the constraint is violated. Computing the slack of a constraint involves simply subtracting one side of the constraint from the other side.

The default setting is indicator mode.

Auto Select Free/Int/Omit Ranges

Check this option if you want the *Free*, *Integer*, or *Omit* ranges selected as you scroll through any list of What's*Best!* defined named range names (these range names begin with WB). With this option checked (on), the range corresponding to a particular name is selected when you click on that name on the list. This enables you to easily identify the range corresponding to a particular name.

The default setting is on.

Minimize Excel® on Solve

Check this option if you would like Excel® to be minimized while the What's*Best!* solver is running.

The default setting is off.

Number of Threads

This feature will tell the What's*Best!* solver to run on multiple threads, so to allocate calculation tasks on different processes. This number can be set between 1 and 32, depending on the computer capabilities.

The default setting is 1.

There is also a choice for the threads:

Solver Decides - Use the best performing multithreading solver on average (either concurrent or parallel) for model type when the number of threads is greater than 1; otherwise, it runs single threaded.

Parallel Preferred - Prefer parallel over concurrent, but use concurrent if an appropriate parallel solver does not exist.

Parallel Only - The solver will run the parallel mode.

Concurrent Preferred - Prefer concurrent over parallel, but use parallel if an appropriate concurrent solver does not exist.

Concurrent Only - The solver will run the concurrent mode.

Hide Status Window on Solve

Check this option if you would like to hide the Status Window while the What's*Best!* solver is running.

The default setting is off.

Linearization

The linearization options offer significant performance gains to some nonlinear models by replacing nonlinear operations with mathematically equivalent linear operations. An example of such performance gains may be seen in the sample model entitled *Linearization Option and Construction Cost Estimation*.

Degree

Many nonlinear operations, such as abs(x) and min(x,y), can be replaced by linear operations that are mathematically equivalent. The ultimate goal is to replace all the nonlinear operations in a model with equivalent linear ones, thus allowing you to use the faster and more robust linear solvers in What's*Best!*. We refer to this process as *linearization*.

Specify the extent to which What's*Best!* should attempt to linearize models with the *Degree* drop-down box. The available options here are *Solver Decides*, *None*, *Mathematical*, and *Mathematical, Logical.* Under the *None* option, no linearization occurs. With the *Mathematical* option, What's*Best!* linearizes *ABS(), MAX(),* and *MIN()* function references as well as any product of binary variables and at most one continuous variable. The *Mathematical, Logical* option is equivalent to the *Mathematical* option plus it will linearize *AND(), IF(), INT(), NOT(), OR(), SIGN(), SUMIF(), VLOOKUP(),* and *HLOOKUP()* functions and all logical operators (<=, =, >=, and <>). Under the *Solver Decides* option, What's*Best!* will do maximum linearization if the number of adjustable cells doesn't exceed 12. Otherwise, What's*Best!* will not perform any linearization. What's*Best!* defaults to using *Solver Decides*.

The linearization process may add a considerable number of constraints and variables to the mathematical program generated to optimize your model. The number of such constraints and variables added is noted in the *WB! Status* report.

Delta Coefficient

You can specify how closely you want the additional constraints added as part of linearization to be satisfied with the *Delta Coefficient* text box.

What's*Best!* defaults to a *Delta Coefficient* coefficient of 0.000001.

Big M Coefficient

Specify the *Big M* value to use during linearization with this text box. When What's*Best!* linearizes a model, it will add *forcing constraints* to the mathematical program generated to optimize your model. These forcing constraints are of the form:

$f(\text{Adjustable Cells}) = M \bullet y$

where *M* is the *Big M Coefficient* and *y* is a 0/1 variable. The idea being that, if some activity in the adjustable cells is occurring, the forcing constraint will drive *y* to take on the value of 1. Given this, if we set the *Big M* value to be too small, we may end up with an infeasible model. Therefore, the astute reader might then conclude that it would be smart to make *Big M* quite large, thereby minimizing the chance of an infeasible model. Unfortunately, setting *Big M* to a large number can lead to numerical roundoff problems in the solver resulting in infeasible or sub-optimal solutions. So, getting a good value for the *Big M Coefficient* may take some experimenting.

The default value for *Big M* is 100,000.

Status Report

Use the Status Report option to specify under what conditions, if any, you would like a status report created when a model is solved. The available options here are: *Never Created*, *Always Created*, *Only on Error/Warning*. The *Never Created* selection means that What's*Best!* will not insert a status report. The *Always Created* is selected by default, and What's*Best!* will insert a status report worksheet entitled *WB! Status* into the current workbook every time you solve. By selecting *Only on Error/Warning*, What's*Best!* will insert a status report worksheet entitled *WB! Status* only if an error or warning from calculation is encountered. If there are no errors or warnings, then no status report is inserted.

The status report contains the model's classification statistics, options selected, and any error messages or warnings generated during the last attempt to solve the model. To view the status report, just click on the *WB! Status* tab of your workbook. The information presented in the status report is discussed in the *Solve* section or the *Solution Outcomes* section in *Overview of Mathematical Modeling*.

The What's*Best!* default is *Always Created*, meaning that a status report is created and inserted into the workbook upon each solve.

Solution Report

Select the *Solution Report* checkbox to tell What's*Best!* you would like a solution report created when the model is solved. The next time you solve, What's*Best!* will generate a worksheet called *WB! Solution* containing different listings.

Never Created: the solver will skip the report; this is the default selection.

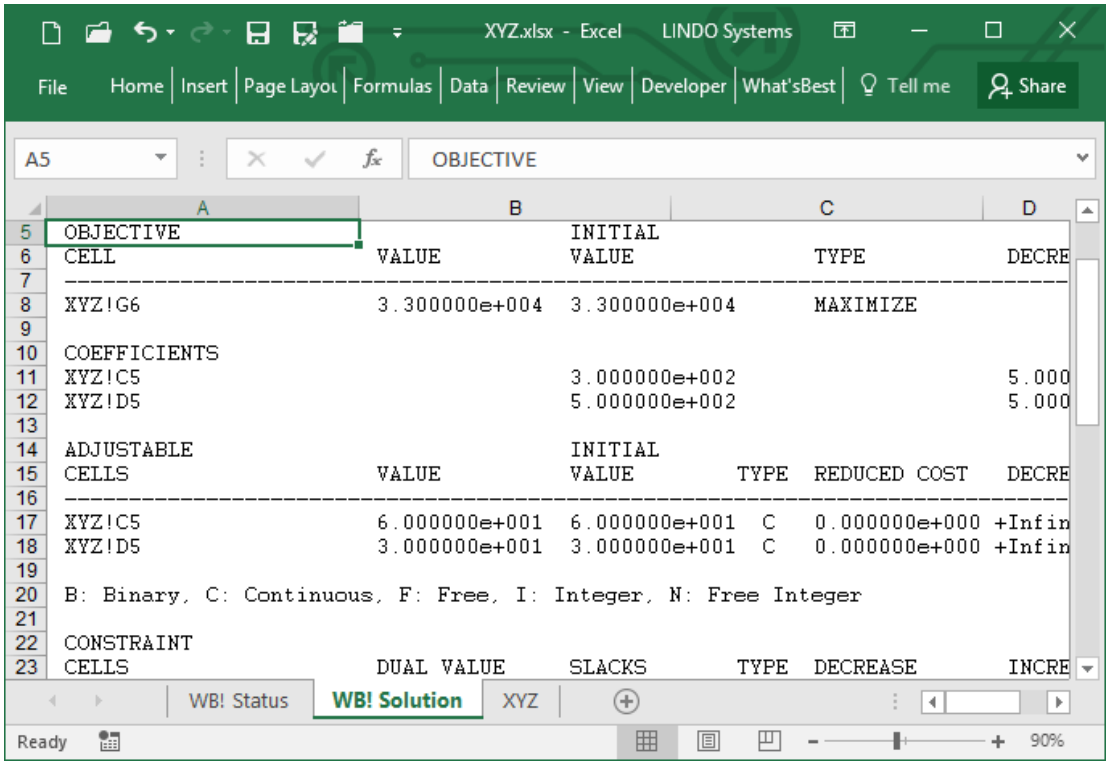
Always Created: this report tells the Adjustable/Best/Constraint cells and their locations, values, types, and formulas, as well as a sensitivity analysis with the dual values for the adjustable cells, reduced cost with range, and the constraint cells, shadow price with range.

Always Created Detail: for linear models, there will be additional information for the coefficients involved in the objective cell formula, showing the initial values and the relative ranges for sensitivity analysis on the objective value.

Full Report: in addition to the detailed report, the last part shows a summary of the location for the Adjustable, the Constraint, and the Nonlinear block of cells, to ease the understanding of the model. This part may take a few moments to be generated.

Click on the *WB! Solution* tab to see the solution report.

Here's an example of the solution report "*WB! Solution*" for the XYZ sample model covered earlier:



	A	B	C	D
5	OBJECTIVE		INITIAL	
6	CELL	VALUE	VALUE	TYPE DECRE
7	----------	----------	----------	----------
8	XYZ!G6	3.300000e+004	3.300000e+004	MAXIMIZE
9				
10	COEFFICIENTS			
11	XYZ!C5		3.000000e+002	5.000
12	XYZ!D5		5.000000e+002	5.000
13				
14	ADJUSTABLE		INITIAL	
15	CELLS	VALUE	VALUE TYPE	REDUCED COST DECRE
16	----------	----------	----------	----------
17	XYZ!C5	6.000000e+001	6.000000e+001 C	0.000000e+000 +Infin
18	XYZ!D5	3.000000e+001	3.000000e+001 C	0.000000e+000 +Infin
19				
20	B: Binary, C: Continuous, F: Free, I: Integer, N: Free Integer			
21				
22	CONSTRAINT			
23	CELLS	DUAL VALUE	SLACKS TYPE	DECREASE INCRE

Note that creating solution reports may dramatically increase runtimes on large models.

Beginning or End

Select the *Beginning* or *End* radio button to tell What's*Best!* whether to place any reports (status and/or solution report) at the beginning of the worksheet tabs (far left) or at the end of all worksheet tabs (far right).

The default is to insert the reports at the *Beginning*.

Debuggers

There are two debuggers:

♦ *Infeasible Constraint* - this debugger will list the infeasible constraints. These constraints are contributing to the infeasibility of the model.

♦ *Unbounded Variable* - this debugger will list the unbounded variables. These constraints are contributing to unbound the model.

Selecting the debuggers may increase the running time to provide a status report, since the solver has to identify the problematic areas for the constraints and variables.

Warnings

These checkboxes allow you to select the warnings you wish to display in the status report. The warnings are:

♦ *Nonlinearity Present* - warning that nonlinear functions or formulas are present in the model. Nonlinear functions can adversely affect the speed and performance of the What'sBest! solver. When possible, nonlinearities should be avoided. See Overview of Mathematical Modeling for a discussion of nonlinear relationships.

♦ *No Best Cell* - warning that no objective (best cell) is specified for the model. The omission of a best cell may be intentional (e.g., in a goal seeking model) or unintentional (e.g., in an optimization model).

♦ *Reference to Blank Cell* - warning that blank cells are referenced in formulas. The reference of a blank cell in a formula may be intentional (e.g., summing across a large range that contains a few blank cells) or unintentional (e.g., accidentally referencing A21 instead of A12).

♦ *Unsupported Function* - warning that an unsupported function appears in the model. The value of a cell containing an unsupported function will not be recalculated during the solution process.

♦ *String Argument Present* - warning that a text argument appears in a function that is expecting a numeric argument. Unexpected text arguments are treated as if they have a numeric value of zero.

♦ *Irreconcilable Constraint* - warning that a violated constraint does not depend upon adjustable cells, and, therefore, cannot be reconciled.

♦ *Support Lookup Functions* - warning that the Lookup functions will be read and processed; otherwise, these functions will be treated as unsupported functions.

♦ *Edit Links* - warning that the interface will check for any suspicious add-in links that may be corrupted. The user will need to reset the links via the Microsoft® Excel® add-in options, the Edit Links feature, and the Update Links of What'sBest! before saving the file.

♦ *Hidden Names* - warning that notifies about names generated by Excel® or any other add-ins, and are not showing via the Name Manager menu of Excel®. These names are saved with the workbook, and may interfere with the reading of the model. They can be deleted via the menu Reset To default.

The default is to display all warnings except the Reference to Blank Cell and the Infeasible Constraint debugger warnings.

Update Links

Select the *Update Links* button to update all What's*Best!* functions in the current workbook to point to the present location of the What's*Best!* add-in file (WBA.XLA or WBA.XLAM).

When a file is opened that contains What's*Best!* functions created on a system in which the What's*Best!* program files were in a different location, Excel® displays a message about "automatic links" and asks: "Update all Linked Information?". You may reply *No* to this dialog box given that the Excel® link update won't update the links to the What's*Best!* add-in. If you now look at a cell containing one of the What's*Best!* functions, you will see that it contains the path of the What's*Best!* add-in file at the time the function was created. For example, a function of the form:

=*WB*(A1,"<=",B1)

has a prepended path and appears something like this:

='C:\PROGRAM FILES\MICROSOFT OFFICE\OFFICE10\LIBRARY\wba.xla'!WB(A1,"<=",C1)

Or in Excel® 2007:

='C:\PROGRAM FILES\MICROSOFT OFFICE\OFFICE12\LIBRARY\LINDOWB\wba.xlam'!WB(A1,"<=",C1)

The *Appendix* section of this manual gives additional information on the location of the Add-in files.

In the case of the sample model files, the prepended path may not coincide with the path to your What's*Best!* add-in file. For this reason, Excel® prompts you to update links and, after declining this, you should use the What's*Best! Update Links* button (in the *General Options* dialog box). The *Update Links* command in What's*Best!* finds the path to your What's*Best!* add-in file and then returns the equations to the previous form without a prepended path. If you leave the paths in their incorrect form, then Excel® fails to find your What's*Best!* add-in file and it inserts an error code of #REF! into each of the cells having an incorrect path.

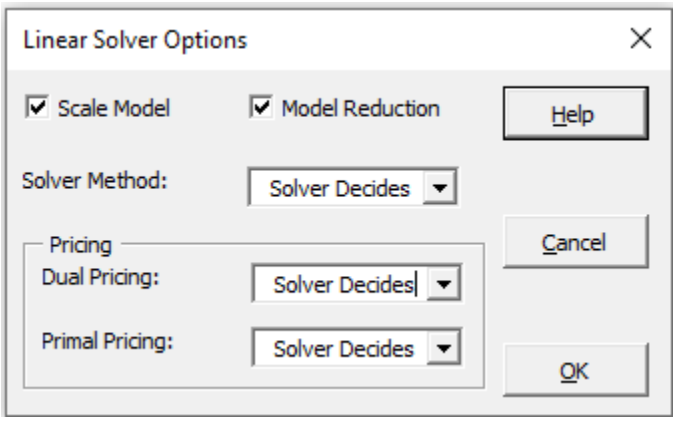
After updating is completed, the workbook/sheet should be saved.

Delete Reports

Click on the *Delete Reports* button to remove the *WB! Status, the WB! Solution, WB!_ Stochastic* and the *WB!_Histogram* reports from the workbook you have open. The next time you solve the model, new reports will be generated according to your specifications in the *Status Report* and the *Solution Report* drop-down boxes. You might want to use the *Delete Reports* button if you have already solved the model and want to save it without the What's*Best!* reports, especially when the reports are at the end of many worksheets.

Options...|Linear Solver

The dialog box posted by the *Options...|Linear Solver* command appears as follows:



This command allows you to set a number of options controlling the function of the linear solver.

Scale Model

Select the *Scale Model* checkbox to rescale the matrix coefficients, so the ratio of the largest to the smallest coefficients is reduced. Rescaling reduces the likelihood of roundoff error and thereby promotes numerical stability and accuracy.

The default setting is to enable scaling.

Model Reduction

Enable the *Model Reduction* checkbox to have the solver identify and remove extraneous variables and constraints from the formulation before solving. In some cases, this greatly reduces the size of the model to be solved to the user's benefit. In other cases, reduction only adds to the solution time without significantly decreasing the size of the model. Because *Model Reduction* can offer significant improvement in performance, this is one of the more significant options. However, it is extremely difficult to predict whether a model will solve better or worse with reduction, so the user is advised to try both settings to determine which setting offers better performance.

The default is to enable reduction.

Solver Method

Indicate the method you would like the linear solver in What's*Best!* to employ using the *Solver Method* drop-down box. Possible choices include: *Solver Decides*, *Primal Simplex*, *Dual Simplex*, and *Barrier*. The simplex method moves along the surface of the feasible region toward the optimal solution while

the barrier method moves through the interior region. The *Solver Decides* setting currently defaults to the primal simplex method.

As a rough guideline, primal simplex tends to do better on sparse models with fewer rows than columns. Dual simplex tends to do well on sparse models with fewer columns than rows. The barrier method is most effective on densely structured or very large models.

The barrier method is an optional feature in What's*Best!*. However, most versions ship with a 30-day trial license for the barrier solver. If your version of What's*Best!* doesn't currently include the barrier option, you may add it at any time. Please contact LINDO Systems for information. If the barrier solver is chosen without a license, What's*Best!* will default to using the primal simplex method.

The default setting for *Solver Method* is *Solver Decides*.

Pricing

Select the type of pricing for the dual simplex and primal simplex algorithms in the *Dual Pricing* and *Primal Pricing* drop-down boxes, respectively. Pricing is a way of assessing the relative attractiveness of variables for selection in the simplex algorithm.

Dual Pricing

One pricing algorithm for dual simplex is called *Partial*, according to which the dual simplex solver prefers variables offering the highest absolute rate of improvement toward the objective. This preference is acted upon regardless of how far other variables have to move in tandem with the chosen variables. Since these other variables may quickly hit a bound, the resulting gain for the objective may actually be quite small. The pricing mode called *Steepest Edge* takes a different approach of more painstakingly selecting variables according to the actual improvement in the objective. Because the objective's gain is higher per iteration with *Steepest Edge*, fewer iterations–of somewhat longer duration–are required.

The default value for *Dual Pricing* is *Solver Decides*.

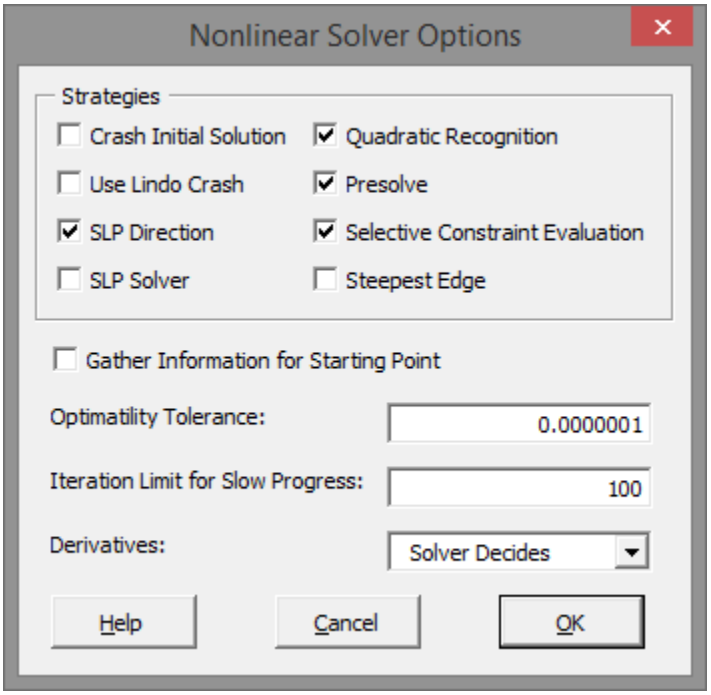
Primal Pricing

For the primal simplex algorithm, there are two primary methods. The *Partial* method prices a small subset of variables each iteration and intermittently prices out all the variables to determine a new subset of interesting variables. *Devex* prices out all columns during each iteration using a steepest-edge approximation. Use of the thorough steepest-edge algorithm means that *Devex* results in fewer overall iterations, though each iteration is longer than it would be under the *Partial* method. Devex is useful with degenerate models, but it is generally difficult to predict in advance what method to use.

The default value for Primal Pricing is *Solver Decides*.

Options...|Nonlinear Solver

This command allows you to set a number of options controlling the nonlinear solver. The dialog box posted by the *Options...|Nonlinear Solver* command appears as follows:



The nonlinear solver is an option on larger versions of What's*Best!*. You can try the nonlinear solver in the Demo version of What's*Best!* or contact LINDO Systems to purchase a nonlinear solver license for a larger version. For more information regarding the difference between linear and nonlinear models, see *Linear vs. Nonlinear Expressions and Linearization.*

Crash Initial Solution / Use LINDO Crash

Check the Crash Initial Solution or Use LINDO Crash checkbox to have What's*Best!*'s nonlinear solver invokes a heuristic for generating a "good" starting point when solving a model. If this initial point is relatively good, subsequent solver iterations should be reduced along with overall runtimes. The two checkboxes invoke two different internal strategies for the purpose of collecting data.

The default setting for Crash Initial Solution and Use LINDO Crash is off.

SLP Direction

Check the *SLP Direction* checkbox to have What's*Best!*'s nonlinear solver use *successive linear programming* (SLP) techniques to compute new search directions. This technique uses a linear approximation in search computations in order to speed iteration times.

In general, the number of total iterations will tend to rise when this method is used. However, runtimes will tend to be shorter.

The default setting for *SLP Direction* is on.

SLP Solver

The nonlinear solver is provided with a SLP Solver. The user is advised to try this solver which may offer better performance for some models. This option should not be used with the Multistart feature of the Global Solver.

The default setting for the *SLP Solver* is off.

Quadratic Recognition

Check the *Quadratic Recognition* checkbox to have What's*Best!*'s nonlinear solver use algebraic preprocessing to determine if an arbitrary nonlinear model is actually a quadratic programming (QP) model. If a model is found to be a QP model, then it can be passed to the faster barrier or conic solver. Note that the QP solver is not included with the standard version of What's*Best!*, but comes as part of the barrier option.

The default setting for *Quadratic Recognition* is on.

Presolve

Check the *Presolve* checkbox to have What's*Best!*'s nonlinear solver identifiy and remove extraneous variables and constraints from the formulation before solving. In some cases, this greatly reduces the size of the model, thus reducing the solving time.

The default setting for *Presolve* is on.

Selective Constraint Evaluation

Check the *Selective Constraint Evaluation* checkbox to have What's*Best!*'s nonlinear solver evaluate constraints only on an as needed basis. Thus, not every constraint will be evaluated each iteration. This generally leads to faster solution times, but can also lead to problems in models that have functions that are undefined in certain regions.

For example, What's*Best!* may not evaluate a constraint for many iterations only to find that the solver has moved to a point in a region where the constraint is no longer defined. In this case, there may not

be a valid point for the solver to retreat to and the solution process terminates with an error. Turning off *Selective Constraint Evaluation* eliminates these errors.

The default setting for *Selective Constraint Evaluation* is on.

Steepest Edge

When What's*Best!* is not in *Steepest-Edge* mode, the nonlinear solver will tend to select variables that offer the highest *absolute* rate of improvement to the objective, regardless of how far other variables may have to move per unit of movement in the newly introduced variable. The problem with this strategy is that other variables may quickly hit a bound resulting in little gain to the objective. When the steepest-edge option is enabled, the nonlinear solver spends a little more time in selecting variables by looking at the rate at which the objective will improve *relative* to movements in the other nonzero variables. Therefore, on average, each iteration will lead to larger gains in the objective. In general, the *Steepest-Edge* option will result in fewer iterations. However, each iteration will take longer.

The default setting for *Steepest Edge* is off.

Starting Point

The *Gather Information for Starting Point* option may be useful when solving nonlinear models. This option computes the slack values for the constraint cells and uses these values internally to compute an improved starting point. After solving, the spreadsheet goes back to the original Slack/Indicator display.

The default value for *Gather Information for Starting Point* is off.

Optimality Tolerance

Specify the *Optimality Tolerance* in this text box. The nonlinear solver operates by taking each variable and making small changes to assess the rate of improvement in the objective function. The nonlinear solver's *Optimality Tolerance* is a tolerance placed upon that rate of improvement calculation for the objective function.

If, for a given variable, the computed value proves to be less-than-or-equal-to the *Optimality Tolerance*, then the solver does not attempt further adjustments to that variable's value. Decreasing the *Optimality Tolerance* toward zero will tend to make the solver run longer, but may lead to better solutions to poorly-formulated or poorly-scaled models.

The default setting for *Optimality Tolerance* is 0.0000001.

Iteration Limit for Slow Progress

Specify an integer limit upon the number of successive iterations made without a significant improvement in the objective function in the *Iteration Limit for Slow Progress* text box. In other words, if the value has not improved by the *N*th iteration (*N* having been set as the *Iteration Limit for Slow Progress*), then the solver terminates the solution process. In cases when the model has a "flat"

objective function around the optimal solution, a large iteration limit may be required to allow for sufficient iterations to move to the optimal solution.

The default setting for the *Iteration Limit for Slow Progress* is 100.

Derivatives

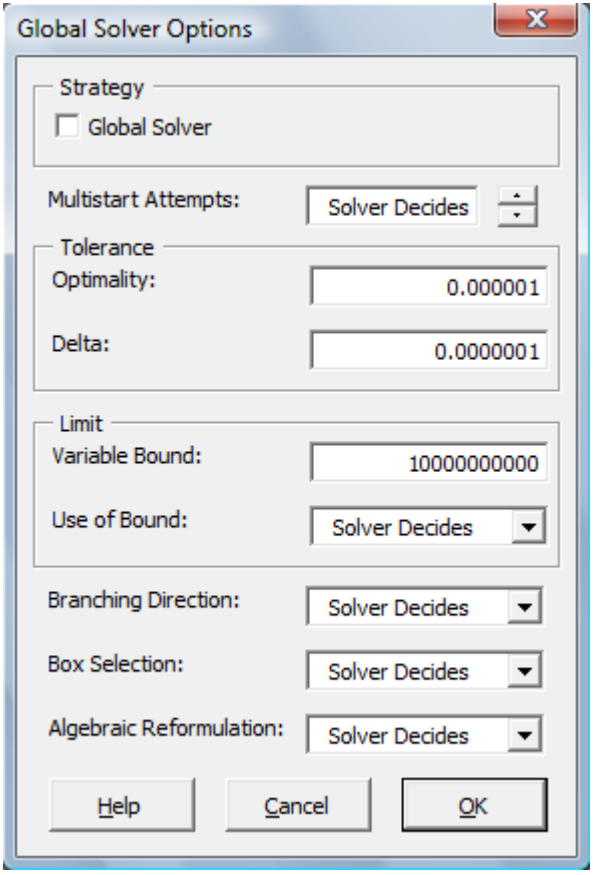
Use the drop down menu to control the derivative method in the nonlinear solver. There are two different methods to use: *Numerical* or *Analytical* derivatives. Numerical derivatives are computed using finite differences. There are two choices available: *Central Differences* and *Forward Differences*. Analytical derivatives are computed directly by symbolically analyzing the arithmetic operations in a constraint. Again, the two choices are *Backward Analytical*, and *Forward Analytical*.

By default, What's*Best!* uses backward analytical derivatives. However, different derivatives can improve speed and precision on many nonlinear models. Experimenting with the various derivative options may be required to determine which method works the best for a particular model.

The default setting for the *Derivatives* is 0 or *Solver Decides*.

Options...|Global Solver

The dialog box posted by the *Options...|Global Solver* command appears as follows:



This command allows you to set a number of options controlling the function of the global solver.

Global Solver

Check the *Global Solver* checkbox to have the global solver partition the original nonconvex/nonlinear problem into several convex/linear subproblems. The global solver will then use a branch-and-bound technique to exhaustively search over these subproblems for the globally optimal solution. The global solver will rigorously find a global optimum if allowed to run to completion. But be aware that runtimes may increase dramatically (particularly on large models) when selecting the global option.

For more information on the concept of global versus local optimality refer to section *Local Optima vs Global Optima.*

The default setting for *Global Solver* is off.

Multistart Attempts

Use the *Multistart Attempts* text box to have the What's*Best!*'s nonlinear solver invoke a heuristic for generating multiple good starting points. The multistart capability is useful for quickly finding better solutions when a non-convex model has several local optima.

Note: The multistart capability is an optional component of the What's*Best!* solver and is packaged as part of the global solver option. Your particular installation may not have the global option enabled, in which case enabling *Multistart Attempts* will cause a warning to be generated when solving. If you are interested in adding the multistart capability, please contact LINDO Systems.

The default setting for *Multistart Attempts* is *Solver Decides*.

Optimality

Specify the global solver's optimality tolerance in the *Optimality* text box in the *Tolerance* box. This tolerance indicates that candidate solutions must beat the incumbent solution by at least this amount to become the new best solution.

The default setting for *Optimality* is 0.000001.

Delta

Specify the delta tolerance the global solver's convexification process in the *Delta* text box in the *Tolerance* box. This tolerance is a measure of how closely the additional constraints added as part of convexification should be satisfied.

The default setting for *Delta* is 0.0000001.

Variable Bound

Specify the maximum magnitude of variable bounds used in the global solver's convexification process in the *Variable Bound* text box in the *Limit* box. Any variable bound with a magnitude in excess of this value will be truncated to this setting. Setting this parameter appropriately helps the global solver focus on more productive domains.

The default setting for *Variable Bound* is 10000000000.

Use of Bound

Specify how to impose the above variable bound limit on the model variables in the *Use of Bound* drop-down box in the *Limit* box. Possible values are: *Solver Decides*, *None*, *All Variables*, or *Selected Variables*.

The default setting for *Use of Bound* is 0 or *Solver Decides*.

Branching Direction

Specify the direction to branch first when branching on a variable with the *Branching Direction* drop-down box. The branch variable is selected as the one that holds the largest magnitude in the measure. Possible values are: *Solver Decides*, *Absolute Width*, *Local Width*, *Global Width*, *Global Distance*, *Absolute Violation*, or *Relative Violation*.

The default setting for *Branching Direction* is 0 or *Solver Decides*.

Box Selection

Specify the node selection rule for choosing between all active nodes in the global solver's branch-and-bound tree in the *Box Selection* drop-down box. Possible values are: *Solver Decides*, *Depth First, Worst Bound, or Best Bound*.

The default setting for *Box Selection* is 0 or *Solver Decides*.

Algebraic Reformulation

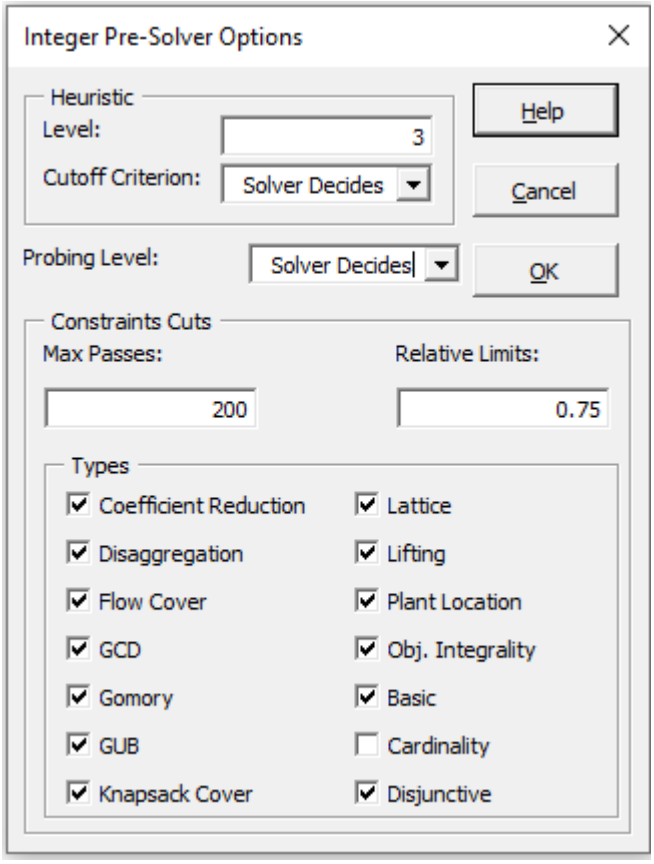
Control the extent of algebraic reformulation (e.g., whether $x*(y+x)$ is replaced by $x*y+x^2$) with the *Algebraic Reformulation* drop-down box. The algebraic reformulation process is crucial in building a tight convex envelope to enclose the nonlinear/nonconvex functions, thereby helping to improve overall convergence. Possible values are: *Solver Decides*, *None*, *Minimum*, *Medium*, or *Maximum*.

The default setting for *Algebraic Reformulation* is *Solver Decides*.

Note: The global solver needs the nonlinear option and the mixed integer option to operate

Options...|Integer Pre-Solver

The dialog box posted by the *Options…*|*Integer Pre-Solver* command appears as follows:



This command allows you to set a number of options controlling the function of the integer pre-solver.

Integer programming models are inherently difficult to solve. This is due to the many possible combinations of values that the integer variables may assume. In fact, the possible number of combinations grows *exponentially* with the number of integer variables. Given this rate of growth in complexity, it doesn't take a whole lot of integer variables to create a problem that is very tough to solve. The integer pre-solver carefully examines integer models using a number of heuristic procedures. The goal of these procedures is to reduce the number of integer variable combinations that need to be examined. In some cases, the pre-solver can deduce a model's solution without the need to pass it on to the full-blown integer solver.

The integer pre-solver options available to the user are listed below.

Heuristics Level

Use the *Heuristics Level* drop-down box to control the level of integer programming heuristics. These heuristics use the continuous solution at each node in the branch-and-bound tree to attempt to quickly find a good integer solution. If an integer solution better than the incumbent is found, then it is used to fix or tighten, global and local variable bounds.

There are four options available: *None*, *Low*, *Medium*, and *High*. The *None* option disables heuristics. The remaining three options allow you to set the level of the amount of heuristics performed to a low, medium, or high level.

What's*Best!* defaults to performing a high level of heuristics.

Heuristics Cutoff Criterion

The *Heuristic Cutoff Criterion* is used to control the criterion for terminating heuristics. Choices here are *Solver Decides*, *Time*, and *Iterations*. Under the Time setting, What's*Best!* terminates heuristics after a certain amount of elapsed time. The Iterations option terminates heuristics after a certain number of iterations. In general, the Time setting results in the fastest performance. However, due to shifting computational loads on a machine, solution paths may change under the Time setting from one solve to the next, potentially resulting in non-reproducible solutions. If reproducibility of a runs is a concern, then the Iterations option should be selected. Under the Solver Decides setting, What's*Best!* chooses the most appropriate strategy. What's*Best!* defaults to *Solver Decides*.

Probing Level

Specify the level to which you would like probing applied to your model with the *Probing Level* drop-down box. *Level 1* is the lowest level of probing, while *Level 9* is the highest. Setting the *Probing Level* to *None* will turn probing off.

The *Probing Level* option is applicable to mixed integer linear models. Probing examines the integer variables to deduce tighter variable bounds and right-hand side values. This process is referred to as tightening the formulation and can result in the integer variables taking on values that are closer to being integral when the continuous relaxation of the model is solved. Probing can often tighten an integer model enough to dramatically speed solution times. In other cases, probing cannot adequately tighten a model and the expense of the probing step simply slows down the overall solution time.

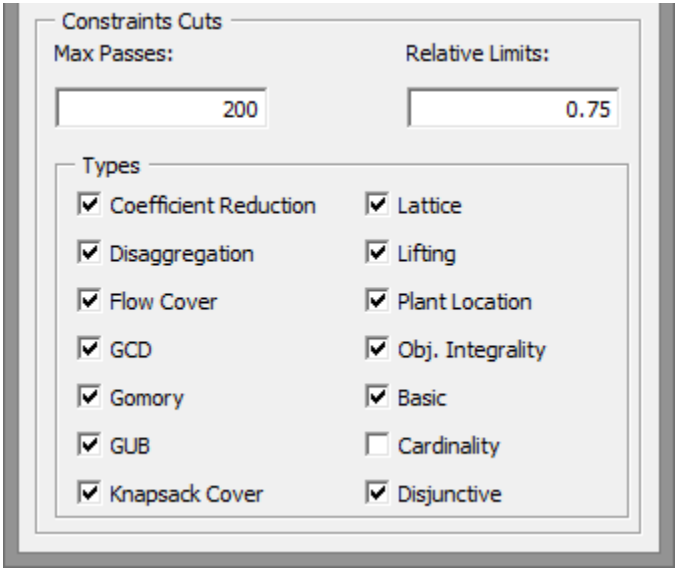
Settings are:

- None: no probing,
- Level 1: minimum, simple presolving,
- Level 2: probing,
- Level 3: coefficient reduction,
- Level 4: elimination,
- Level 5: dual reductions,
- Level 6: use dual information,
- Level 7: binary row presolving,
- Level 8: row aggregation,
- Level 9: maximum pass.

The default setting for *Probing Levels* is *Solver decides,* which results in the solver setting probing to a level it believes will offer the best result.

Constraint Cuts

The options contained in the *Constraint Cuts* box on the *Integer Pre-Solver* dialog box:



can be used to influence the solver's cut generation phase on linear models.

What's*Best!* 's integer programming pre-solver performs extensive evaluation of your model in order to add constraint cuts. Constraint cuts are used to "cut" away sections of the feasible region of the continuous model (i.e., the model with integer restrictions dropped) that are not contained in the feasible region to the integer model. On most integer models, this will accomplish two things. First, solutions to the continuous problem will tend to be more naturally integer. Thus, the branch-and-bound solver will have to branch on fewer variables. Secondly, the bounds derived from intermediate solutions will tend to be tighter, allowing the solver to "fathom" (i.e., drop from consideration)

branches higher in the branch-and-bound tree. These improvements should dramatically speed solution times on most integer models.

Max Passes

Set the maximum number of passes allowed by the integer pre-solver to any non-negative, integer value with the *Max Passes* text box. The integer pre-solver makes iterative passes through a model determining appropriate constraint cuts to append to the formulation. In general, the benefit of each successive pass declines. At some point, additional passes will only add to the total solution time. Thus, What's*Best!* imposes a limit on the maximum number of passes. The default limit is 200 passes.

Relative Limit

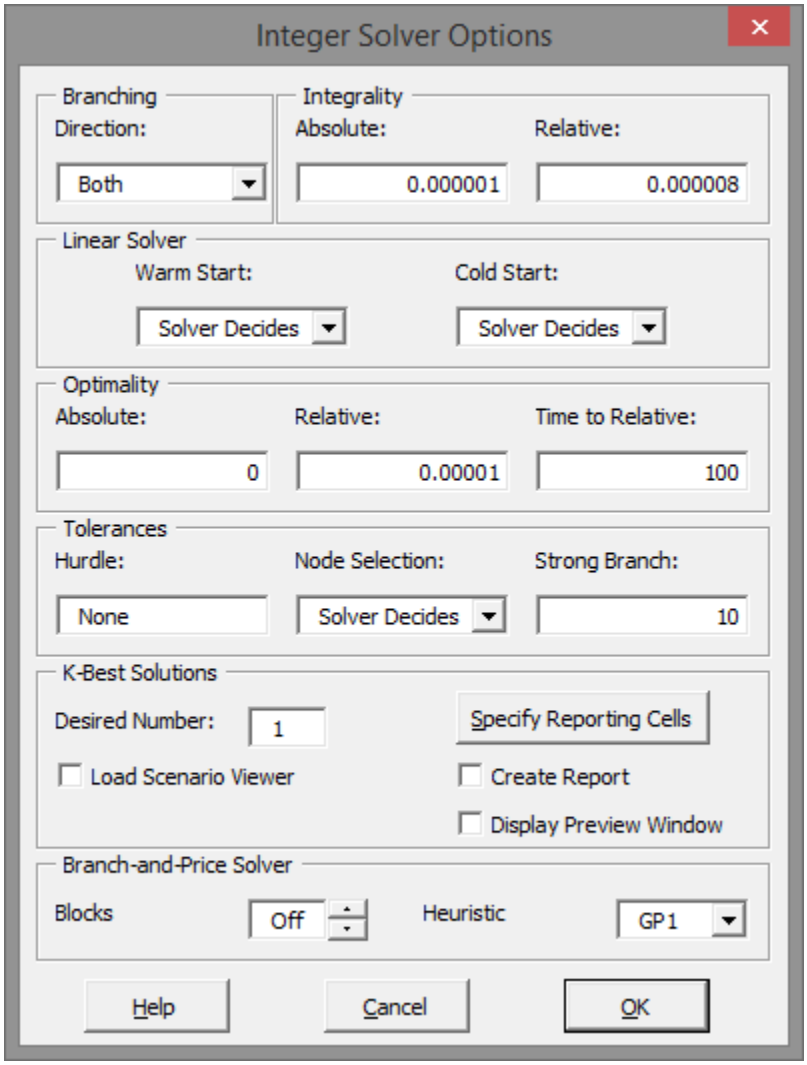
Set the relative constraint cut limit by changing it to any desired fractional value with the *Relative Limits* text box. Most integer programming models benefit from the addition of some constraint cuts. However, at some point additional cuts take more time to generate than they save in solution time. For this reason, What's*Best!* imposes a relative limit on the number of constraint cuts that are generated. The default limit is set to 0.5 times the number of true constraints in the original formulation.

Types

Enable or disable any of the different strategies What's*Best!* uses for generating constraint cuts with the checkboxes in the *Types* box. It is beyond the scope of this help file to go into the details of the various strategies. Interested readers may refer to any good text on integer programming. What's*Best!* defaults to enabling all cut generation strategies: *Coefficient Reduction, Disaggregation, Flow Cover, GCD, Gomory, GUB, Knapsack Cover, Lattice, Lifting, Plant Location, Objective Integrality, Basic, Cardinality, Disjunctive.*

Options...|Integer Solver

The dialog box posted by the *Options...|Integer Solver* command appears as follows:



This command allows you to set a number of options controlling the function of the integer solver.

To understand the operation of the integer options, it is useful to understand how integer problems are normally solved. With the default option settings, What's*Best!* solves integer problems using the following steps:

1. What's*Best!* begins by solving the *continuous relaxation* (i.e. the original model with integer restrictions removed). This gives an optimistic bound on the objective of the true integer model, because the objective of the integer-restricted model could never be better than the objective of the continuous approximation.

2. After solving the continuous relaxation, What's*Best!* uses a process called *branch-and-bound* to find the optimal integer solution. Branch-and-bound implicitly enumerates all possible integer solutions in an intelligent fashion, minimizing the number of solutions that have to be explicitly examined. However, the number of potential solutions grows exponentially with the number of integer adjustable cells. Thus, models with a large number of integers may take a *very long time* to solve.

Options such as the optimality tolerances set limits upon how exhaustively this branch-and-bound search will be carried out. *Optimality Tolerance* is particularly useful in some integer problems as a means of significantly decreasing the solution time.

Branching Direction

Use the *Direction* drop-down box in the *Branching* box to govern the preferred direction of branching. Branching occurs when the branch-and-bound manager forces an integer variable that is currently fractional to an integer value. When the branching direction is set to *Up*, the branch-and-bound manager will branch on a fractional integer variable by first forcing it to the next largest integer. The reverse is true when this option is set to *Down*. When the option is set to *Both*, the branch-and-bound manager makes an educated guess as to the best initial branching direction for each fractional variable.

The default setting is *Both*.

Integrality

Due to the potential for round-off error on digital computers, it is not always possible to find exact integer values for the integer variables. The two tolerances in the *Integrality* box, *Absolute* and *Relative*, control the amount of deviation from integrality that will be tolerated.

Absolute

Specify the absolute amount of violation from integrality that is acceptable in the *Absolute* text box. Specifically, if *I* is the closest integer value to *X*, *X* will be considered an integer if:

$|X - I|$ <= Absolute Integrality Tolerance.

The default value for this tolerance is .000001. Although one might be tempted to set this tolerance to 0, this may result in feasible models being reported as infeasible.

Relative

Specify the relative amount of violation from integrality that is acceptable in the *Relative* text box. Specifically, if *I* is the closest integer value to *X*, *X* will be considered an integer if:

$$\frac{|X - I|}{|X|} <= \textit{Relative Integrality Tolerance}.$$

The default value for the relative integrality tolerance is .000008. Although one might be tempted to set this tolerance to 0, this may result in feasible models being reported as infeasible.

LP Solver

When solving a mixed linear integer programming model, the branch-and-bound solver solves a linear programming (LP) model at each node of the solution tree. You may choose between the primal simplex, dual simplex, or barrier (assuming the barrier option was purchased with your license) solver for handling these linear programs. The two options in the *LP Solver* box, *Warm Start* and *Cold Start*, control this choice of LP solver based on whether there is a starting basis (warm start) or no starting basis (cold start).

Warm Start

Use the *Warm Start* option to control the linear solver that is used by the branch-and-bound solver at each node of the solution tree when a starting basis is present to use as an initial starting point. The *Cold Start* option, discussed below, determines the solver to use when a previous solution does not exist. The available options are:

- *Solver Decides* – What's*Best!* chooses the most appropriate solver.
- *Barrier* – What's*Best!* uses the barrier method, assuming you have purchased a license for the barrier solver. Otherwise, the dual solver will be used.
- *Primal* – The primal solver will be used exclusively.
- *Dual* – The dual solver will be used exclusively.

In general, *Solver Decides* will yield the best results. The barrier solver can't make use of a pre-existing solution, so *Barrier* usually won't give the best results. In general, *Dual* will be faster than *Primal* for reoptimization during branch-and-bound.

Cold Start

Use the *Cold Start* option to control the linear solver that is used by the branch-and-bound solver at each node of the solution tree when a previous solution is not present. The *Warm Start* option, discussed above, determines the solver to use when a previous solution does exist. The available options are:

- *Solver Decides* – What's*Best!* chooses the most appropriate solver at each node.
- *Barrier* – What's*Best!* uses the barrier method, assuming you have purchased a license for the barrier solver. Otherwise, the dual solver will be used.
- *Primal* – The primal solver will be used exclusively.
- *Dual* – The dual solver will be used exclusively.

In general, *Solver Decides* will yield the best results. However, experimentation with the other options may be fruitful.

Optimality

The *Optimality* box contains three tolerances: *Absolute*, *Relative*, and *Time to Relative*. These tolerances allow you to tradeoff solution time vs. solution quality. Ideally, we'd always want the solver to find the best solution to a model. Unfortunately, integer programming problems are very complex and the extra computation required to find an optimum solution can be prohibitive. On larger integer models, the alternative of getting a solution within a few percentage points of the true optimum after several minutes of runtime, as opposed to the true optimum after several days, makes the use of these tolerances quite attractive.

Absolute

Use the *Absolute* text box to set the absolute optimality tolerance. This is a positive value *r*, indicating to the branch-and-bound solver that it should only search for integer solutions with objective values at least *r* units better than the best integer solution found so far. In many integer programming models there are huge numbers of branches with roughly equivalent potential. This tolerance helps keep the branch-and-bound solver from being distracted by branches that can't offer a solution significantly better than the incumbent solution.

In general, you shouldn't have to set this tolerance. Occasionally, particularly on poorly formulated models, you might need to increase this tolerance slightly to improve performance. In most cases, you should experiment with the relative optimality tolerance (discussed below) in order to improve performance.

The default value for the *Absolute* optimality tolerance is 0.0.

Relative

Use the *Relative* text box to set the relative optimality tolerance. This is a value *r*, ranging from 0 to 1, indicating to the branch-and-bound solver that it should only search for integer solutions with objective values at least 100**r*% better than the best integer solution found so far. The end results of modifying the search procedure in this way are twofold. First, on the positive side, solution times can be improved tremendously. Second, on the negative side, the final solution obtained by What's*Best!* may not be the true optimal solution. You will, however, be guaranteed the solution is within 100**r*% of the true optimum on linear integer models.

Typical values for the relative optimality tolerance would be in the range .01 to .05. In other words, you would be happy to get a solution within 1% to 5% of the true optimal value. On large integer models, the alternative of getting a good solution within a few percentage points of the true optimum after several minutes of runtime, as opposed to the true optimum after several days, makes the use of this tolerance quite attractive.

The default value for the relative optimality tolerance is 0.00001.

Note: Generally speaking, increasing the relative optimality tolerance is the change that will most likely improve runtimes on integer models.

Time to Relative

If an integer programming model is relatively easy to solve, then we would like to have the solver press on to the true optimal solution without immediately resorting to a relative optimality tolerance (discussed above). On the other hand, if after running for a while, it becomes apparent that the optimal solution won't be immediately forthcoming, then we might want the solver to switch to using a relative optimality tolerance. The *Time to Relative* tolerance can be used in this manner. This tolerance is the number of seconds before the branch-and-bound solver begins using the relative optimality tolerance. So, for the first *n* seconds, where *n* is the value of the *Time to Relative* tolerance, the branch-and-bound solver will not use the relative optimality tolerance and will attempt to find the true optimal solution to the model. Thereafter, the solver will use the relative optimality tolerance to confine its search.

The default value for the *Time to Relative* tolerance is 100 seconds.

Tolerances

The *Tolerances* group box contains three miscellaneous tolerances for controlling the branching strategy used by the branch-and-bound solver. The three tolerances are *Hurdle*, *Node Selection*, and *Strong Branch*. We discuss each of these in the sections that follow.

Hurdle

Enter a value in the *Hurdle* text box if you know the objective value of a solution to a model. This value is used in the branch-and-bound manager to narrow the search for the optimum. More specifically, What's*Best!* will only search for integer solutions in which the objective is better than the *Hurdle* value.

Any user-supplied *Hurdle* value comes into play when What's*Best!* is searching for an initial integer solution. At this point, the solver can ignore branches in the search tree with objective values worse than the *Hurdle* value because a better solution exists (i.e., the *Hurdle*) on some alternate branch. Depending on the problem, a good *Hurdle* value can greatly reduce solution time. However, if you set a *Hurdle* value *better* than the optimal integer solution, What's*Best!* will return an infeasible error message because no feasible answer can satisfy the bound you have set.

Once What's*Best!* finds an initial integer solution, the *Hurdle* tolerance no longer has an effect. At this point, the relative optimality tolerance comes into play.

Hurdle differs from the relative optimality tolerance. If you use *Hurdle* by itself and an integer answer is found, it will yield the true optimal integer solution.

For example, if you're confident the best integer answer is at least 95 for a maximization problem, just enter 95 as the value in the *Hurdle* field. What's*Best!* won't consider any branch that doesn't yield an answer of at least 95. If, in fact, the best integer answer is only 93, you will eventually get the message "Solution Status: No Feasible Solution Found".

The default value for *Hurdle* is *None*.

Note: When entering a hurdle value, be sure that a solution exists that is at least as good or better than your hurdle. If such a solution does not exist, What's*Best!* will not be able to find a feasible solution to the model.

Node Selection

The branch-and-bound solver has a great deal of freedom in deciding how to search the branch-and-bound solution tree. Use the *Node Selection* drop-down box to control the order in which the solver selects branch nodes in the tree.

If you examine the pull down list for the node selection option, you will see the following options:

- *Solver decides* – This is the default option. What's*Best!* makes an educated guess regarding the best node to branch on.
- *Depth first* – What's*Best!* spans the branch-and-bound tree using a depth first strategy.
- *Worst bound* – What's*Best!* picks the node with the worst bound.
- *Best bound* – What's*Best!* picks the node with the best bound.

In general, *Solver decides* will offer the best results. Experimentation with the other options may be beneficial with some classes of models.

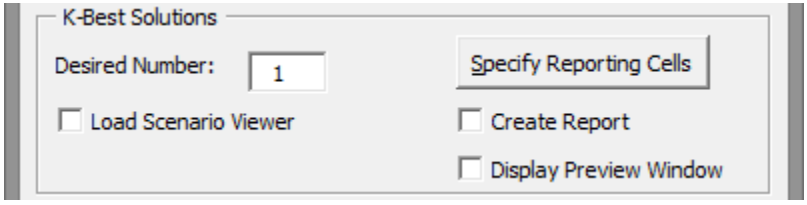
Strong Branch

Use the *Strong Branch* text box to specify a more intensive branching strategy during the first *n* levels of the branch-and-bound tree, where *n* is the option's setting. During these initial levels, What's*Best!* picks a subset of the fractional variables as branching candidates. What's*Best!* then performs a tentative branch on each of the variables in the subset, selecting as the final candidate the variable that offers the greatest improvement in the bound on the objective. Although strong branching is useful in tightening the bound quickly, it does take additional computation time. So, you may want to try different settings to determine what works best for your model.

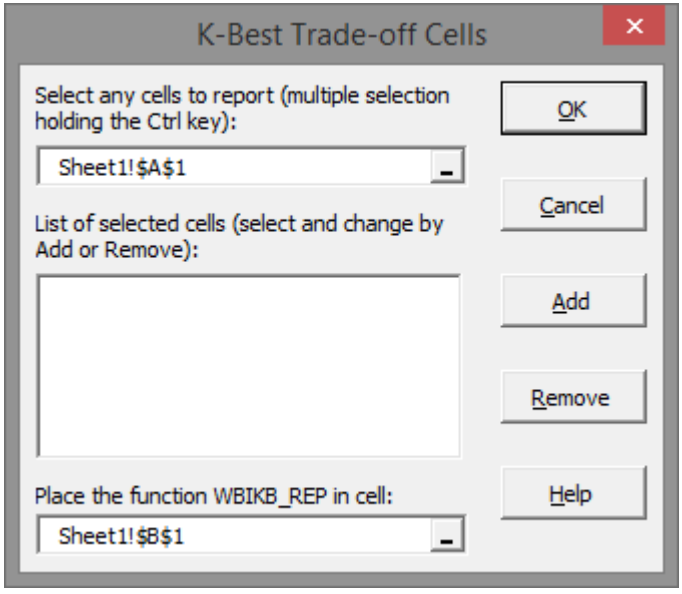
The default *Strong Branch* setting is 10 levels.

K-Best Solutions

The *Desired Number* parameter is used to set the number of solutions desired as part of the K-Best solutions feature of *What'sBest!*'s mixed integer solver. The *Specify Reporting Cells* is used for selecting the trade-off cells.



Whenever this value is greater than 1, say K, *What'sBest!* will return up to K unique solutions to the model, with respect to the 0/1 variables. These solutions will have the property that they are the next best solutions available in terms of their objective values. If there are K or less alternate optima in the 0/1 variables, then these solutions will be listed. Less than K solutions may be returned if a sufficient number of feasible solutions do not exist. Please refer to section *Usage Guidelines for K-Best Solutions* for an example of the use of the K-Best feature.



The default *Desired Number* is 1, with no reporting cells specified.

Create Report Using the Function WBIKB_REP():
The general format is =WBIKB_REP(cells_to_be_reported), e.g. WBIKB_REP(Sheet1!$B$1,Sheet1!$B$2), meaning the cells B1 and B2 of Sheet1 will appear in the K-Best Report. After solution, a new tab,WB!_KBest, will be created. It will have one column for every cell to be reported, and one row for every run.

Select the checkbox to tell What's*Best!* you would like a KBest report created when the model is solved. The next time you solve, What's*Best!* will generate a worksheet called WB!_KBest containing a listing of the cells and their values. Click on the WB!_KBest tab to see the KBest report.

Display Preview Window:
Select this preview window when you want to look at the cells value after solving, before returning to the spreadsheet. These cells are being specified via the WBIKB_REP() functions, and can be anywhere in the workbook. The Usage Guidelines for K-Best Solutions gives an example of the output.

Load Scenario Viewer:
The scenario viewer will give the list of all the Adjustable cells, but only for the worksheet tab currently selected, via the Advanced...|Scenario Viewer menu. This tool can be useful to scroll through the different K-Best solutions without writing additional WBIKB_REP() functions. The Usage Guidelines for K-Best Solutions shows an output of the solutions.

Branch-and-Price Solver

The Branch-and-Price (BNP) Solver contains two parameters: Blocks and Heuristic.



The BNP solver is a mixed integer programming solver for solving models with block structures like the following:

Minimize: Σ c(k) * x(k)

Subject To:

Σ A(k) * x(k) = d (linking constraints)

x(k) in X(k), for all k (decomposition structure)

where d, c(k) and x(k) are vectors and A(k) is a matrix with appropriate dimensions. x(k) contains decision variables and X(k) denotes a linear feasible domain for x(k).

The BNP solver is a hybrid of branch-and-bound, column generation, and Lagrangean relaxation methods. It can help to find either the optimal solution or a better lower bound (the Lagrangean bound) for a minimization problem. Based on the decomposition structure, the solver divides the original problem into several subproblems, or blocks, and solves them (almost) independently, exploiting parallel processing if multiple cores are available.

BNP may perform better than the default MIP solver if: a) the number of linking constraints is small, b) the number of blocks is large and they are of approximately the same size, and c) the number of available processors (or cores) is large, e.g., 4 or more. Also, there may be some models for which BNP finds a good solution and good bound more quickly than the default MIP algorithm, although it may take longer to prove optimality.

The Blocks option for the BNP solver controls the number of subproblems, or blocks, that the model will be partitioned into. Possible setting for the Blocks parameter are:

♦ Off - This will disable the BNP solver, in which case, the standard MIP solver will be used to solve all mixed integer linear programs.

♦ N - A positive integer, greater-than-or-equal-to 2, indicating the number of independent blocks to try and partition the model into via one of the graph partitioning algorithms provided by What's*Best!*. The actual heuristic used is chosen with the Heuristic parameter.

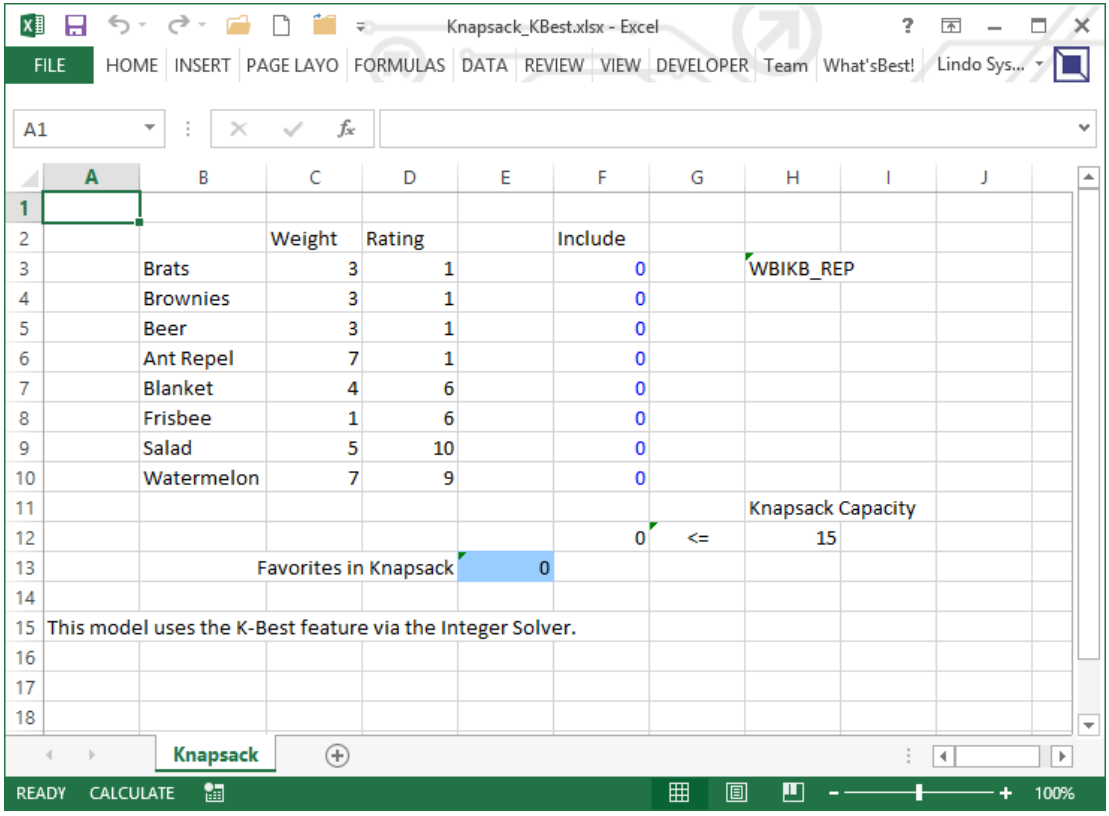
The default setting for Blocks is Off, i.e., the BNP solver will not be used on integer programming models.

The Block Heuristic parameter controls the heuristic used to partition the model into blocks. You may currently select from two graph partitioning algorithms named simply GP1 and GP2, with the default setting being GP1.

Usage Guidelines for K-Best Solutions

As an example, you are packing a basket for a picnic you will be taking with a friend. You've constructed a list of items you would like to carry with you on the picnic. Each item has a weight associated with it and your knapsack is limited to carrying no more than 15 pounds. You have also come up with a 1 to 10 rating for each item, which indicates how strongly your friend wants to include the particular item in the knapsack for the picnic. The information are listed in the model below, "KNAPSACK_KBEST.xlsx".

The *KNAPSACK_KBEST* Worksheet *Before* Solving



	A	B	C	D	E	F	G	H	I	J
1										
2			Weight	Rating		Include				
3		Brats	3	1		0		WBIKB_REP		
4		Brownies	3	1		0				
5		Beer	3	1		0				
6		Ant Repel	7	1		0				
7		Blanket	4	6		0				
8		Frisbee	1	6		0				
9		Salad	5	10		0				
10		Watermelon	7	9		0				
11								Knapsack Capacity		
12						0	<=	15		
13			Favorites in Knapsack		0					
14										
15	This model uses the K-Best feature via the Integer Solver.									
16										
17										
18										

Where:

Cell F12 has the formula: =SUMPRODUCT(C3:C10,WBBINf), where WBBINf is the range for the adjustable binary cells in column F.

Cell E13 has the formula: =SUMPRODUCT(D3:D10,F3:F10), objective cell to maximize.

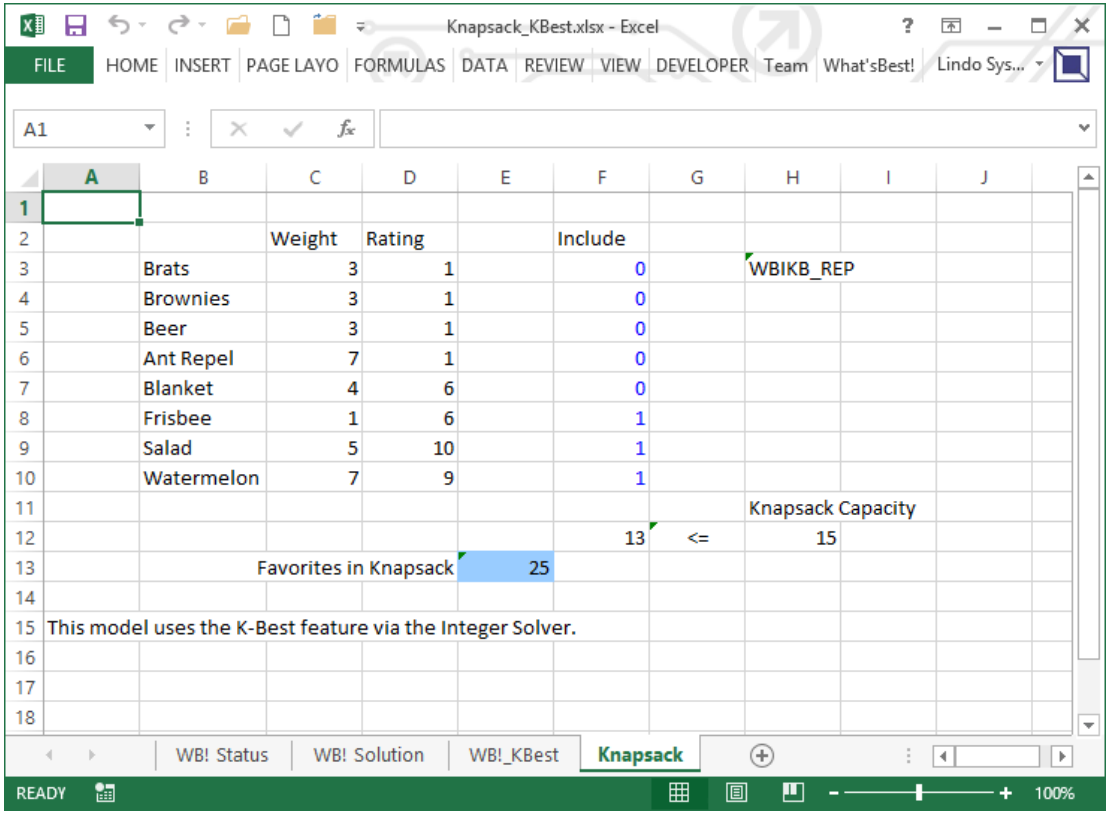
Cell G12 is the capacity constraint: =WB(F12,"<=",H12).

Say our friend's ratings of the candidate picnic items is given in the data section above, and the rating is different than yours:

Brats	8
Brownies	10
Beer	9
Ant Repel	2
Blanket	3
Frisbee	6
Salad	4
Watermelon	10

If we solve the model as is, thus solely maximizing our preferences, we get the following solution:

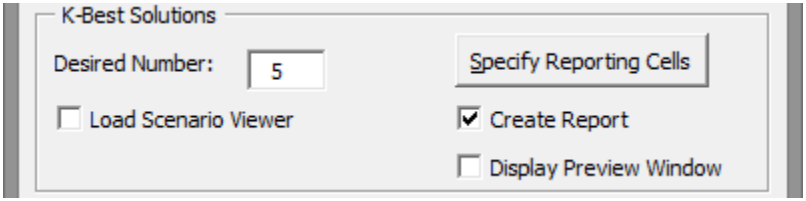
The *KNAPSACK_KBEST* Worksheet *After* Solving



	A	B	C	D	E	F	G	H	I	J
1										
2			Weight	Rating		Include				
3		Brats	3	1		0		WBIKB_REP		
4		Brownies	3	1		0				
5		Beer	3	1		0				
6		Ant Repel	7	1		0				
7		Blanket	4	6		0				
8		Frisbee	1	6		1				
9		Salad	5	10		1				
10		Watermelon	7	9		1				
11								Knapsack Capacity		
12						13	<=	15		
13				Favorites in Knapsack	25					
14										
15	This model uses the K-Best feature via the Integer Solver.									
16										
17										
18										

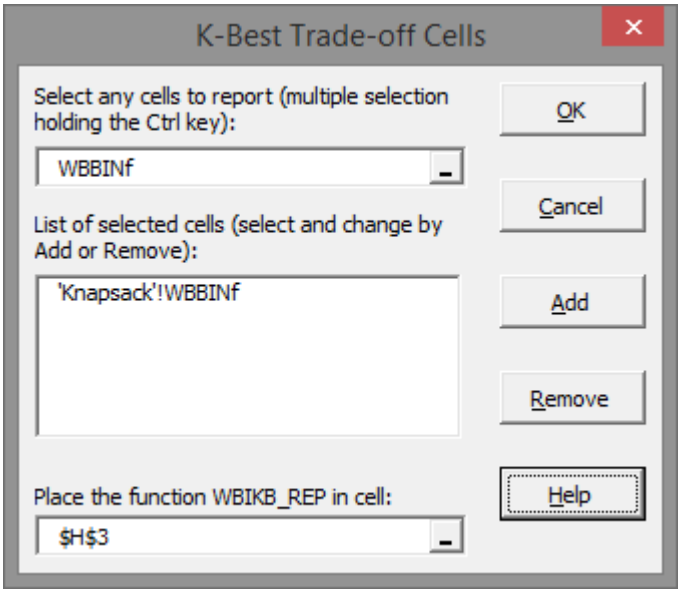
As indicated by the adjustable cells, a few of our favorite items are included in the optimal basket. However, we are wondering if there isn't another combination of items that our friend might like almost as much that includes most of our favorite items.

To investigate this question, we set the Desired Number parameter of the K-Best Solutions box to 5, via the *Options...|Integer Solver*.



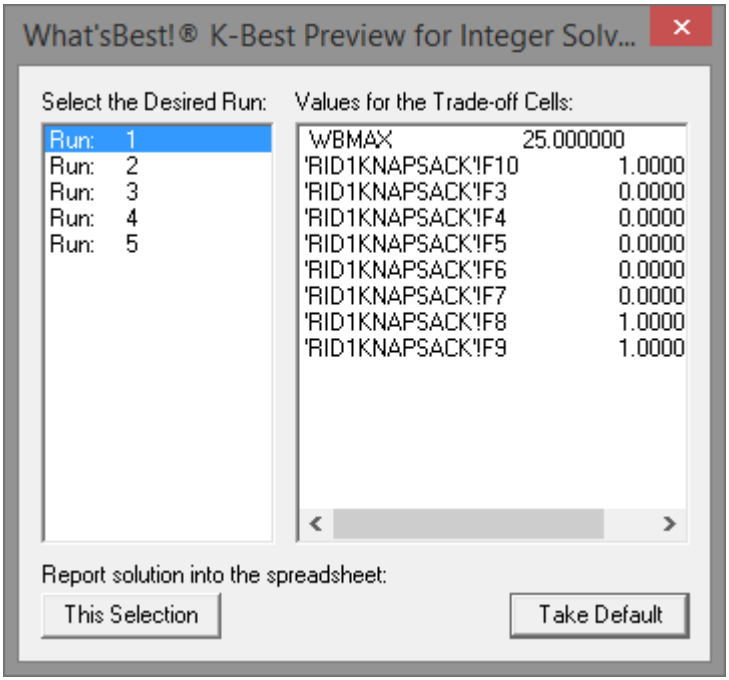
We also define the cells to look at via the button **Specify Reporting Cells**, here all the Adjustable cells: select the range of cells, or cell by cell, select the cell to refer, then click **Add**. You can modify or delete an entry by clicking on the cell reference in the list.

Cell H3 has the formula: =WBIKB_REP(F3:F10)



This means that we would like to generate the 5 best solutions to the model. We then click **OK** and then run the Solve command. The solver sees that the K-Best feature is being requested, and it automatically generates the 5 best solutions to the model.

With the "Display Preview Window" checked, we are presented with the following dialog box:



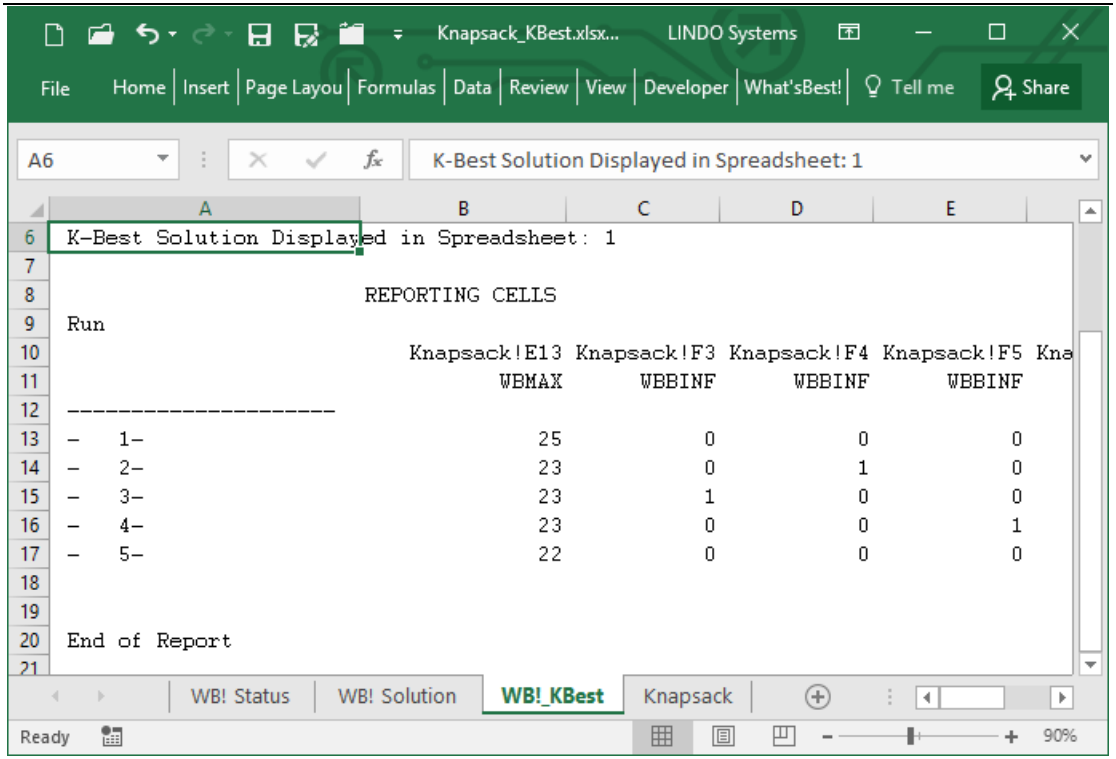
In the Preview window, we see that the solver was able to find 5 feasible next-best solutions to the model. The solutions are ranked in order by their objective values.

There is also a column labeled Trade-off Cells, which lists the value in each solution of a designated trade-off variable. Any scalar variable in a model can be selected as the tradeoff variable. In this example, there are 9 variables, and the objective cell WBMAX is automatically selected as a tradeoff variable. The idea behind the trade-off variable is that it allows you to weigh the trade-offs in a model's objective value with a secondary goal. In this case, our secondary goal is the number of our favorite items in the picnic basket. In particular, we see that there are three solutions with slightly worse objective values (23 vs. 25) that include one of our favorite items.

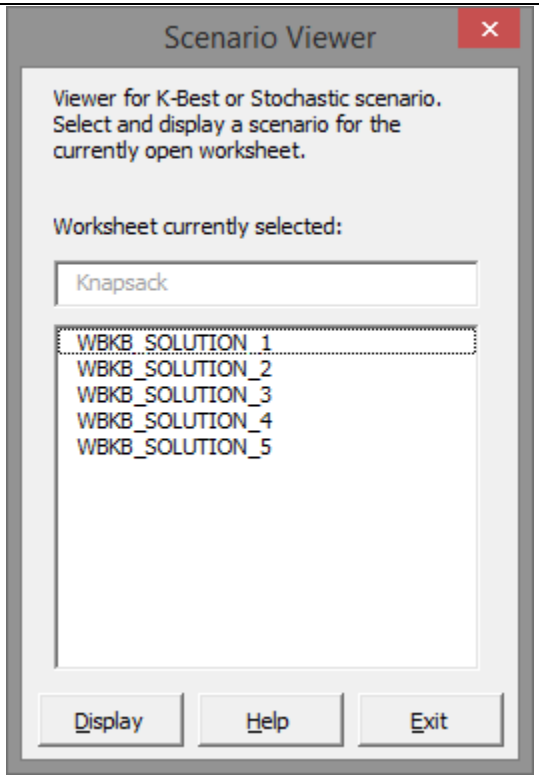
For example, if we selected the solution "Run: 2" and pressed the **This Selection** button, we'd see the following solution containing one of our favorite items, Brownies (cell F4). The **Take Default** button selects the "Run: 1", which is the solution of the objective cell.

These buttons allow you to display selected solutions returned by the K-Best solver. Once a final solution is selected, all subsequent status and solution reports will be based on that particular solution.

Here's an example of the K-Best report for the KNAPSACK_KBEST sample model, when the option was selected before solving.



With the "Load Scenario Viewer" checked, the user will then need to open the dialog box via the Advanced...|Scenario Viewer menu. First select the "knapsack" tab, then select the Scenario Viewer via the menu. The solutions are being presented for all the Adjustable cells in the currently selected "knapsack" tab. Finally display or double click the solution to write the values directly into the worksheet.



Note: The "Preview Window" shows the values specified via the function WBIKB_REP() for any cells in the workbook. The "Scenario Viewer" menu shows the Adjustable cells for only the selected worksheet tab.

Options...|Stochastic Solver

The *Options...*|*Stochastic Solver* command allows you to support Stochastic Modeling via the series of dialog boxes.

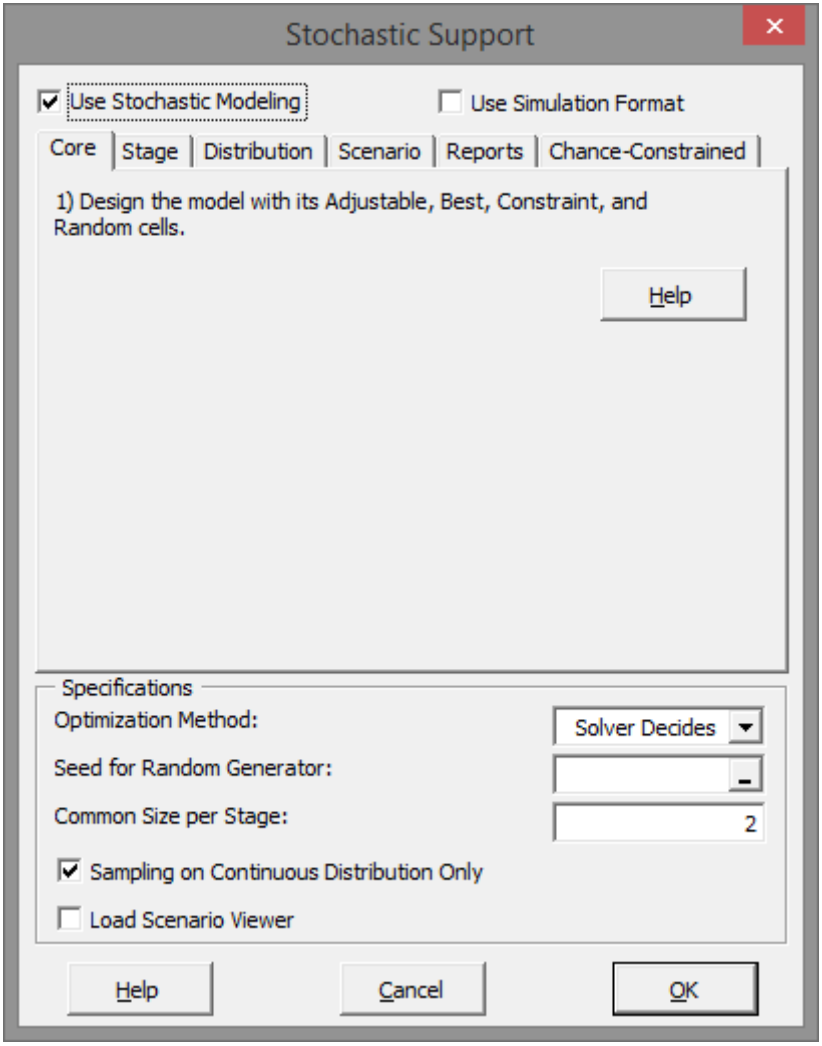
Note: The user can directly enter or modify the stochastic functions in the spreadsheet without reopening the dialog box.

Stochastic Modeling Support

By checking this option, the model will be processed as a stochastic model. Otherwise, What's*Best!* will treat the core model, even if some stochastic functions are set in the spreadsheet.

The default setting is False (unchecked).

Step/Tab 1 "core model"

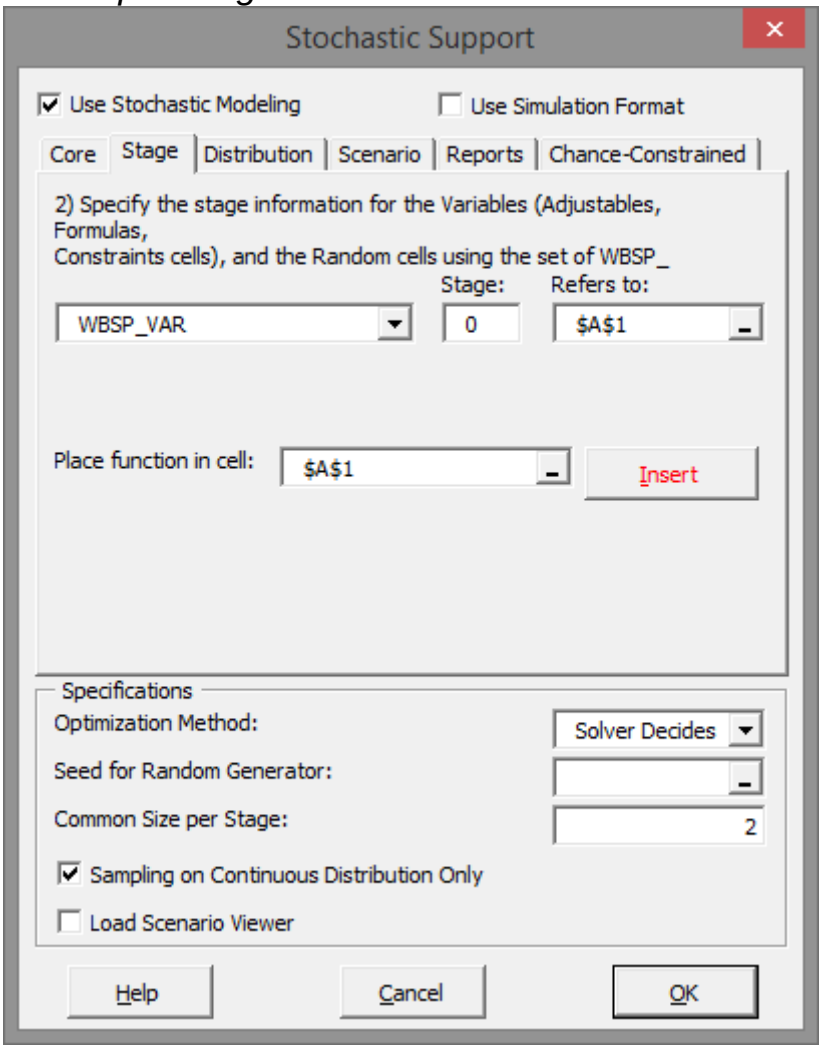


This step simply reminds you to first formulate a standard deterministic "core" model (Adjustable, Best, Constraint). A cell that represents a Random quantity is simply filled with an arbitrary number.

Cells containing equations or strings are inappropriate as random cells. What's*Best!* cannot rewrite an equation, although the value returned by an equation will change, if the equation depends upon any adjustable cells. Therefore, be sure to enter a numeric value in each of your random cells.

Note:	The Random cells will be identified during the Step 2 below.

Step/Tab 2 "sequencing"



This step provides the related pieces of the sequencing of decision and random events, or “staging”.

The general sequence of events is as follows: Stage 0: We make initial decisions; Stage 1, beginning: First set of random variables are observed, Stage 1, end: We make additional decision or calculations based on observations so far, Stage 2, beginning, the second set of random variables are observed, etc.. This staging information is inserted into the sheet with the WBSP_VAR and WBSP_RAND functions as follows.

In the drop-down list, you can select a WBSP_ function with its associated arguments.

Variables:
"WBSP_VAR"

Use this function to associate a Stage to an Adjustable, Constraint, or formula cell.

The format is =WBSP_VAR(*stage,cells_with_this_stage*), e.g., =WBSP_VAR(0,Sheet1!$A$2) means cell A2 in Sheet1 is in Stage 0. The entry “Place function in cell:” allows you to place this information in a any cell in the workbook. Typically you place this information near the decision cell so that you can quickly observe the staging information of a cell without calling up any special menus.

Randoms:
"WBSP_RAND"

Use this function to associate a Stage to a Random cell.

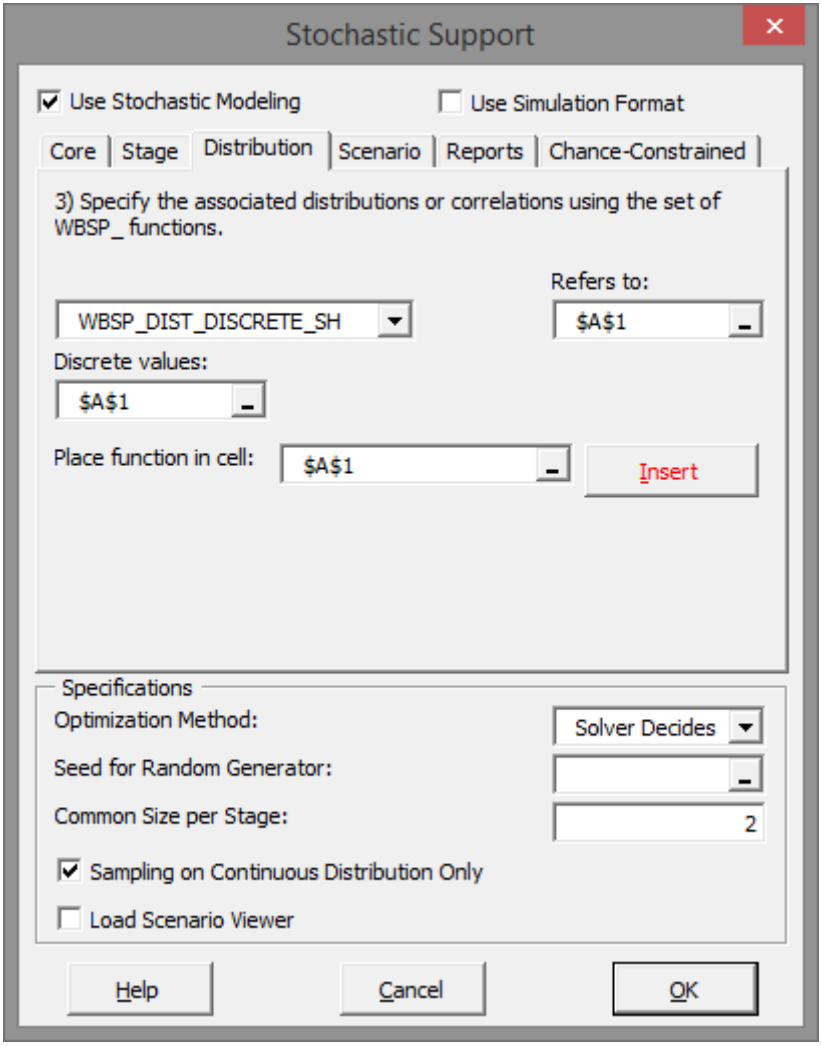
The format is =WBSP_RAND(*stage,random_cells_with_this_stage*), e.g., =WBSP_RAND(1,Sheet1!$A$2), meaning cell A2 in Sheet1 will be at Stage 1.

The "NONE" choice will actually delete the selected cell.

Refers To:
Specify the random cell on which to apply the distribution. Alternately, you can accept the currently selected cells, which What's*Best!* has automatically placed in this box. You can also type the correct cell range directly into the text box.

Place function in cell:
Specify the cell in which to write the WBSP_ function. Alternately, you can accept the currently selected cells, which What's*Best!* has automatically placed in this box. You can also type the correct cell range directly into the text box.

Step/Tab 3 "distributions or correlations"



This step provides the distributions of the random variables, so to specify the distribution information about the random cells.

The format is =WBSP_DIST_*distribution*(*parameter_cells*,*random_cells*), e.g., =WBSP_DIST_LOGNORMAL(C2,D2,Sheet1!$A$2), meaning the random cell A2 in Sheet1 will follow a Lognormal distribution with the value in C2 for the Mean (argument 1), and the value in D2 for the Standard Deviation sigma (argument 2). As before, the entry “Place function in cell:” allows you to place this information in a any cell in the workbook. Typically you place this information near the random cell(s) so that you can quickly observe the distribution information of a cell without calling up any special menus.

Distributions with one argument:

"WBSP_DIST_DISCRETE_SH" argument 1: Discrete values, with scenarios read horizontally

"WBSP_DIST_DISCRETE_SV" argument 1: Discrete values, with scenarios read vertically

"WBSP_DIST_CHISQUARE" argument 1: Degrees freedom

"WBSP_DIST_EXPONENTIAL" argument 1: Rate

"WBSP_DIST_GEOMETRIC" argument 1: Success probabilities

"WBSP_DIST_LOGARITHMIC" argument 1: p-Factor

"WBSP_DIST_POISSON" argument 1: Mean

"WBSP_DIST_STUDENTS_T" argument 1: Degrees freedom

Distributions with two arguments:

"WBSP_DIST_DISCRETE_SH_W" argument 1: Discrete values, argument 2: Weight probabilities,
with scenarios read horizontally

"WBSP_DIST_DISCRETE_SV_W" argument 1: Discrete values, argument 2: Weight probabilities,
with scenarios read vertically

"WBSP_DIST_BETA" argument 1: Shape 1, argument 2: Shape 2

"WBSP_DIST_BINOMIAL" argument 1: Sample size, argument 2: Success probabilities

"WBSP_DIST_CAUCHY" argument 1: Location, argument 2: Scale

"WBSP_DIST_GAMMA" argument 1: Shape, argument 2: Scale

"WBSP_DIST_GUMBEL" argument 1: Location, argument 2: Scale

"WBSP_DIST_F_DISTRIBUTION" argument 1: Degrees freedom 1, argument 2: Degrees freedom 2

"WBSP_DIST_LAPLACE" argument 1: Location, argument 2: Scale

"WBSP_DIST_LOGISTIC" argument 1: Location, argument 2: Scale

"WBSP_DIST_LOGNORMAL" argument 1: Mean, argument 2: Sigma

"WBSP_DIST_NEGATIVEBINOMIAL" argument 1: r-Factor, argument 2: Success probabilities

"WBSP_DIST_NORMAL" argument 1: Mean, argument 2: Sigma

"WBSP_DIST_PARETO" argument 1: Scale, argument 2: Shape

"WBSP_DIST_SYMMETRICSTABLE" argument 1: Alpha, argument 2: Success probabilities

"WBSP_DIST_UNIFORM" argument 1: Upper limit, argument 2: Upper limit

"WBSP_DIST_WEIBULL" argument 1: Scale, argument 2: Shape

Distributions with 3 arguments:

"WBSP_DIST_HYPERGEOMETRIC" argument 1: Population, argument 2: Defective, argument 3: Size

"WBSP_DIST_TRIANGULAR" argument : Lower, argument 2: Upper, argument 3: Mode

Correlations:

You can also specify the correlation information about the random cells. Correlation is optional.

The format is =WBSP_CORR_correlation(matrix,random_cells), e.g., =WBSP_CORR_SPEARMAN(M19:N20,Sheet1!$B$13:$B$14), meaning the random cells B13 and B14 in Sheet1 will follow a spearman correlation with the matrix defined in the range M19:N20. This is a 2-column and 2-row matrix for the 2 random cells to correlate. As before, the entry “Place function in cell:” allows you to place this information in a any cell in the workbook. Typically you place this information near the random cell(s) so that you can quickly observe the correlation information of a cell without calling up any special menus.

"WBSP_CORR_KENDALL" argument 1: matrix

"WBSP_CORR_PEARSON" argument 1: matrix

"WBSP_CORR_SPEARMAN" argument 1: matrix

The "NONE" choice will actually delete the selected cell.

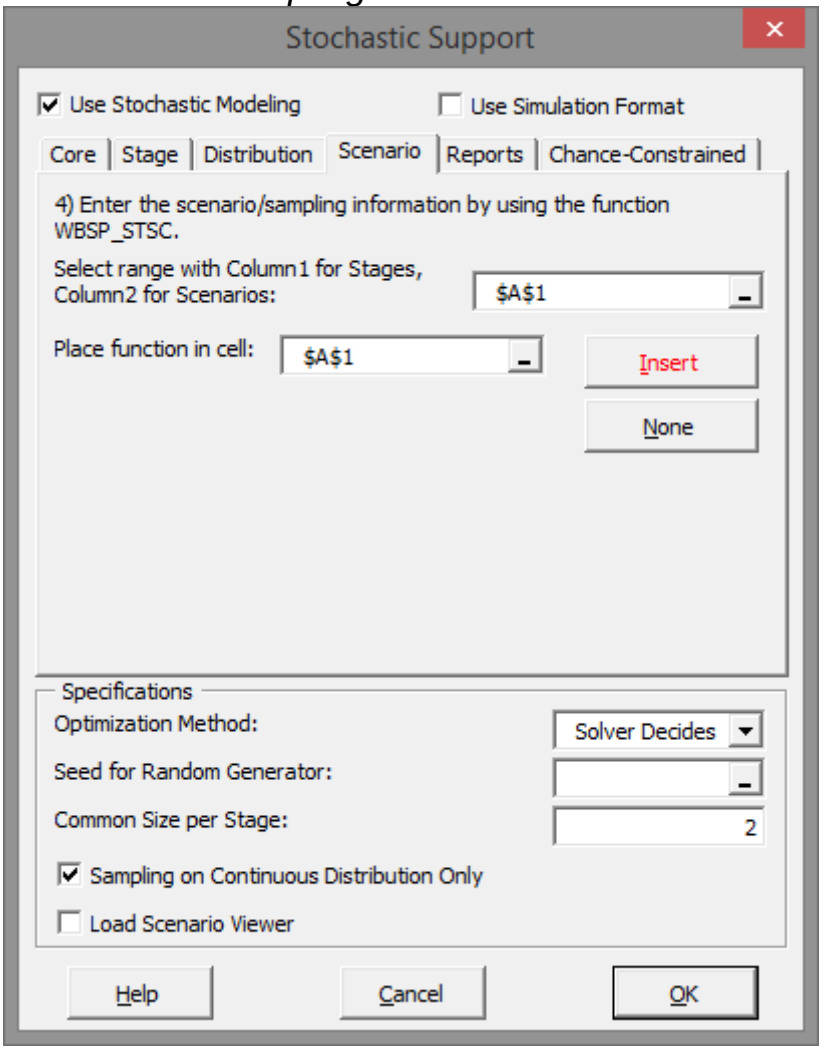
Refers To:

Specify the random cell on which to apply the distribution. Alternately, you can accept the currently selected cells, which What's*Best!* has automatically placed in this box. You can also type the correct cell range directly into the text box.

Place function in cell:

Specify the cell in which to write the WBSP_ function. Alternately, you can accept the currently selected cells, which What's*Best!* has automatically placed in this box. You can also type the correct cell range directly into the text box.

Step/Tab 4 "scenario sampling"



This command allows you to specify the Scenario/Sampling information for the stochastic part.

The general format is =WBSP_STSC(*stages_and_sample_sizes*), e.g, WBSP_STC(B1:C3), means range B1:B3 lists the Stages, and range C1:C3 lists sample size for each stage.

Sampling with WBSP_STSC() has to be specified if the model contains a continuous distribution. With a discrete distribution, there is no necessary call for sampling because it takes the discrete values directly from the discrete table. Otherwise sampling will be applied on the discrete distribution too.

In case of a model with both distributions, continuous and discrete, so the user may want to apply sampling only on the continuous, with the option “Sampling on Continuous Distribution Only” via the Stochastic Solver options box.

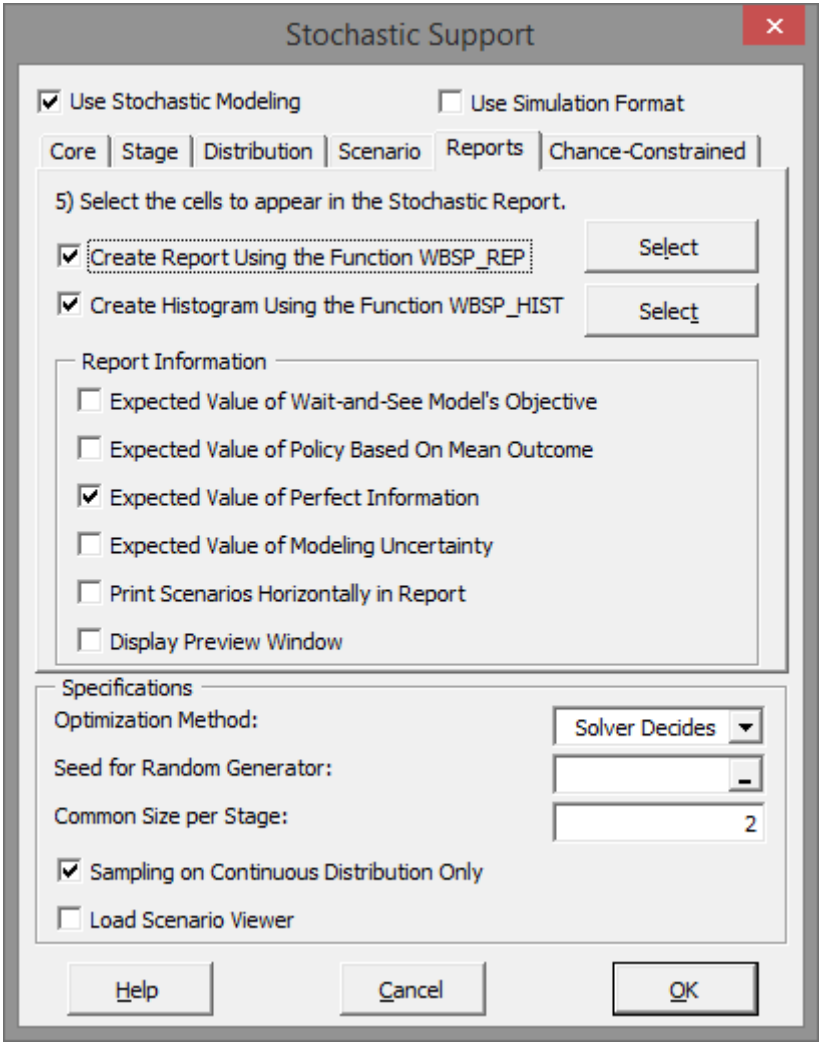
Select Range:
Select the a two-column range, where the column1 is for the Stage in ascending order, and the second column for the number of scenarios associated to the stage.

Place function in cell:
Specify the cell in which to write the "WBSP_STSC" function.

Set button:
Place the function in the spreadsheet.

None button:
Delete the selected cell.

Tab Reports



The solution of an SP model can produce a lot of information. You are probably interested in only a small portion of these results. These features allow you to specify the cells for which to report the values in the *WB!_Stochastic* and the *WB!_Histogram* sheets.

Create Report Using the Function WBSP_REP():

The general format is =WBSP_REP(*cells_to_be_reported*), e.g. WBSP_REP(Sheet1!$B$1,Sheet1!$B$2), meaning the cells B1 and B2 of Sheet1 will appear in the *Stochastic Scenario Report*. After solution, a new tab,*WB!_Stochastic*, will be created. It will have one column for every cell to be reported, and one row for every scenario.

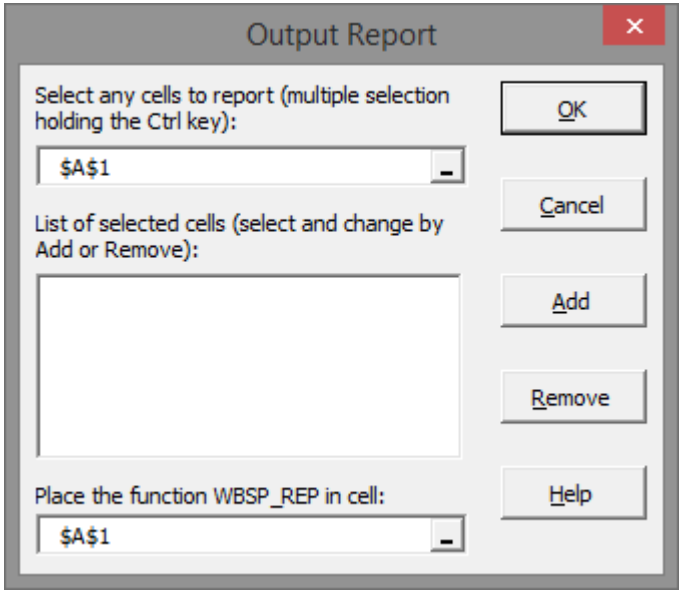
Select the checkbox to tell What's*Best!* you would like a stochastic report created when the model is solved. The next time you solve, What's*Best!* will generate a worksheet called *WB!_Stochastic* containing a listing of the cells and their locations, and values of the variables solution or the random outcomes for the scenario path. Click on the *WB!_Stochastic* tab to see the stochastic report.

Select button:

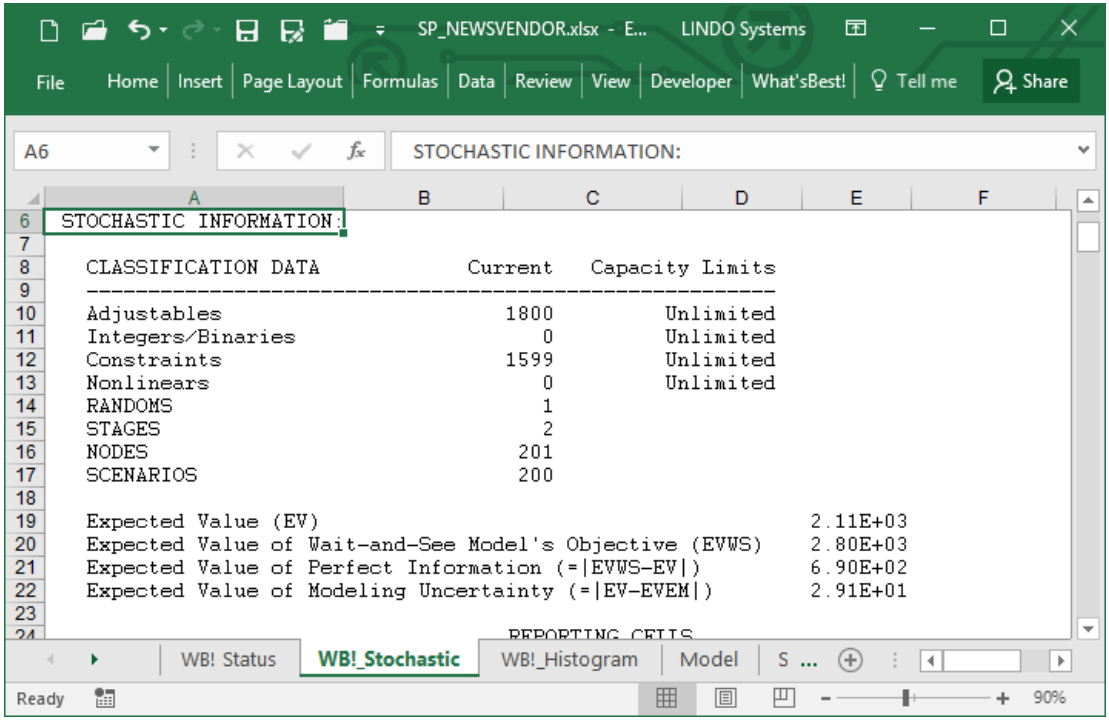
This button will open a new window.

To create the list of reporting cells, specify the cell or cell range in the *Select any cell to report:* field. Once this is done, pressing the *Add* button causes the selected cells to appear in the *List of selected cells* box. You can decide to remove any selected cells by simply going to the list box, select the cell reference to remove from the list of names, and click on the *Remove* button.

The "WBSP_REP" function will be placed in the current selection of the field *Place the function WBSP_REP in cell:*



Here's an example of the stochastic report for the SP_NEWSVENDOR sample model:



Once created, the *WB!_Stochastic* report will stay on the workbook, and the solver will just clear the sheet during successive runs. This makes possible for other sheets to keep their cells reference from this report for displaying a specific data.

Create Histogram Using the Function WBSP_HIST():

The general format is =WBSP_HIST(number_of_bins,cells_to_be_reported), e.g. WBSP_HIST(0,Sheet1!$B$1), meaning the solver decides of the number of bins, and cell B1 Sheet1 will appear in the Stochastic Histogram Report. After solution, a new tab,WB!_Histogram, will be created. It will have four columns reporting the Bin Lows, Bin Highs, Mid-Points, and Bin Counts.

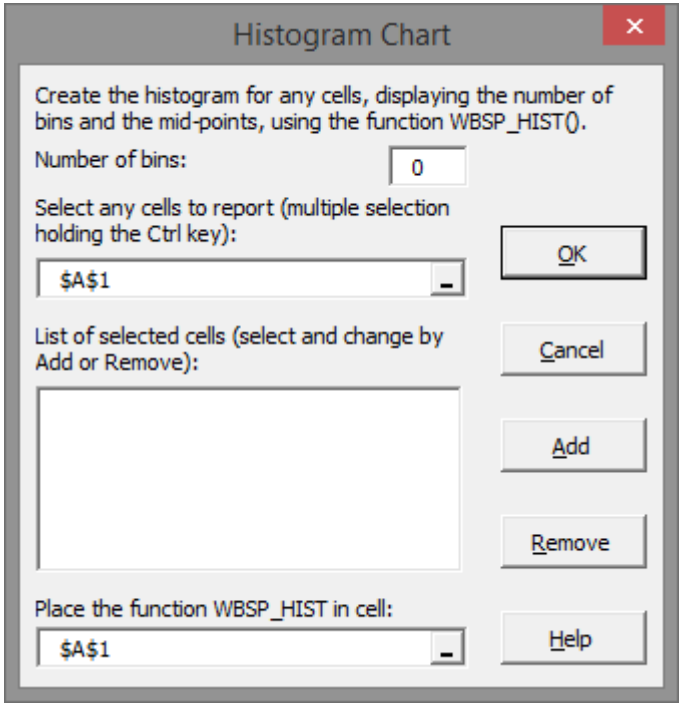
Select the checkbox to tell What's*Best!* you would like a stochastic histogram created when the model is solved. The next time you solve, What's*Best!* will generate a worksheet called WB!_Histogram containing a listing of the data, and the histogram graph for the Bin Counts against the Mid-Points values. Click on the WB!_Histogram tab to see the stochastic histogram report.

Select button:

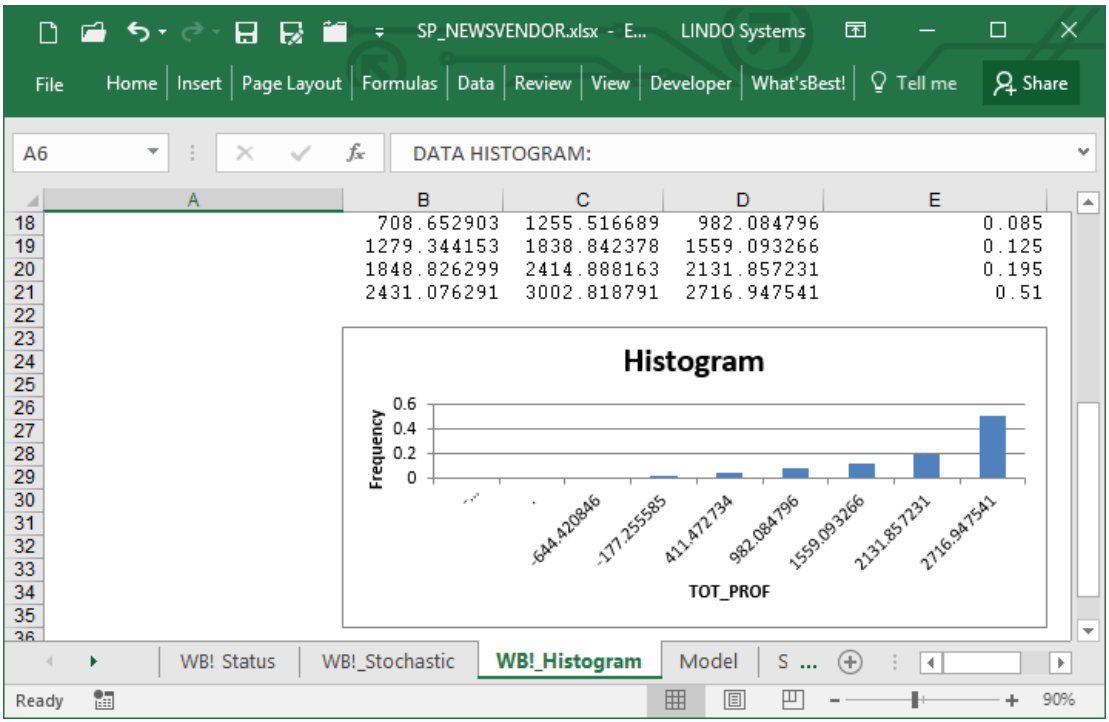
This button will open a new window.

To create the list of reporting cells, specify the cell or cell range in the Select any cell to report: field, and the number of bins. Once this is done, pressing the Add button causes the selected cells to appear in the List of selected cells box. You can decide to remove any selected cells by simply going to the list box, select the cell reference to remove from the list of names, and click on the Remove button.

The "WBSP_HIST" function will be placed in the current selection of the field *Place the function WBSP_HIST in cell*:



Here's an example of the histogram report for the SP_NEWSVENDOR sample model:



Once created, the *WB!_Histogram* report will stay on the workbook, and the solver will just clear the sheet during successive runs. This makes possible for other sheets to keep their cells reference from this report for displaying a specific data.

The solver writes a series of solution in the created tab *'WB!_Stochastic', and 'WB!_Histogram'*, with the expected value, and displays the first default scenario, or a selected scenario, on the spreadsheet.

Please refer to the section *Guidelines for Stochastic Modeling* for a tutorial and information on how to use Stochastic Support effectively.

Expected Value of Objective

The *Expected Value of Objective (EV)* is the expected value for the model's objective over all the scenarios, and is the same as the reported objective value for the model. This value is provided by default in the report.

The expected value of a cell can be reported directly anywhere in the spreadsheet by use of the *WBSP_MEAN* function. Place *=WBSP_MEAN(cellref, 0)* in any available cell. After solving, the 0 will be replaced by the expected value of cellref, and this value will be displayed in the cell containing *=WBSP_MEAN( )*. For instance, enter =WBSP_MEAN($B$22, 0) in any cell. After solving, this function will display the Weighted Mean value of 2106.49622433202 for the objective cell in B22.

Calculation for Expected Value of Wait-and-See Model's Objective

The Expected Value of Wait-and-See Model's Objective (EVWS) reports the expected value of the objective if we could wait and see the outcomes of all the random variables before making our decisions. Such a policy would allow us to always make the best decision based on the outcomes for the random variables, and, of course, is not possible in practice. For a minimization, it's true that EVWS <= EV, with the converse holding for a maximization. Technically speaking, EVWS is a relaxation of the true Stochastic Programming model, obtained by dropping the nonanticipativity constraints. Possible flag values are Disable (False), or Enable (True).

The default setting is off, the EVWS is not calculated.

Calculation for Expected Value of Policy Based On Mean Outcome

This Expected Value of Policy Based On Mean Outcome (EVEM) is the expected true objective value if we (mistakenly) assume that all random variables will always take on exactly their mean values. EVEM is computed using a two-step process. First, the values of all random variables are fixed at their means, and the resulting deterministic model is solved to yield the optimal values for the stage 0 decision variables. Next, a) the stage 0 variables are fixed at their optimal values from the previous step, b) the random variables are freed up, c) the nonanticipativity constraints are dropped, and d) this wait-and-see model is solved. EVEM is the objective value from this EVWS model. Possible flag values are Disable (False), or Enable (True).

The default setting is off, the EVEM is not calculated.

Calculation for Expected Value of Perfect Information

This Expected Value of Perfect Information (EVPI) is the absolute value of the difference between EV and EVWS. This corresponds to the expected improvement to the objective were we to obtain perfect information about the random outcomes. As such, this is a expected measure of how much we should be willing to pay to obtain perfect information regarding the outcomes of the random variables. Possible flag values are Disable (False), or Enable (True).

The default setting is to enable calculation.

Calculation for Expected Value of Modeling Uncertainty

This Expected Value of Modeling Uncertainty (EVMU) is the absolute value of the difference EV - EVEM. It is a measure of what we can expect to gain by taking into account uncertainty in our modeling analysis, as opposed to mistakenly assuming that random variables always take on their mean outcomes. Possible flag values are Disable (False), or Enable (True).

The default setting is off, the EVMU is not calculated.

Note: The above approach for computing EM and EVMU makes unambiguous sense only for models with a stage 0 and a stage 1. If there are later random variables in stages 2, 3, etc., then there are complications. For example, for decisions in later stages, we have seen the outcomes from the random variables in earlier stages, so considering these random variables to take on their mean value does not make sense. For models with additional stages beyond 0 and 1, EVMU will merely be an approximation of the true expected value of modeling uncertainty.

Note: Computing these expected value statistics can be very time consuming for large models.

Print Scenarios Horizontally in Report

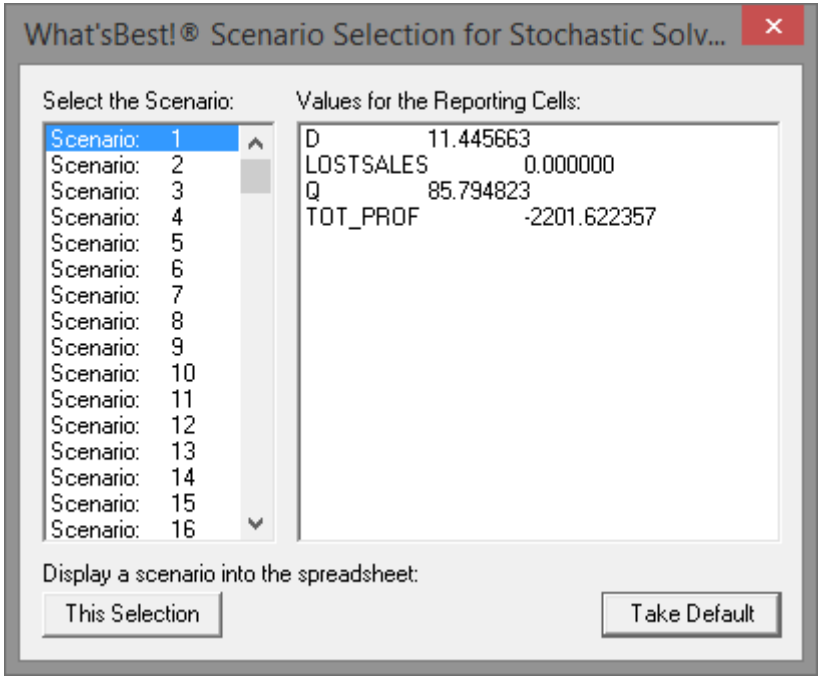
This flag enables to print the scenarios along a row in the WB!_Stochastic report, so the reporting cells will be displayed along a column. It is useful for models reporting numerous cells for a few scenarios.

The default setting is off, meaning the scenarios will appear vertically in the report.

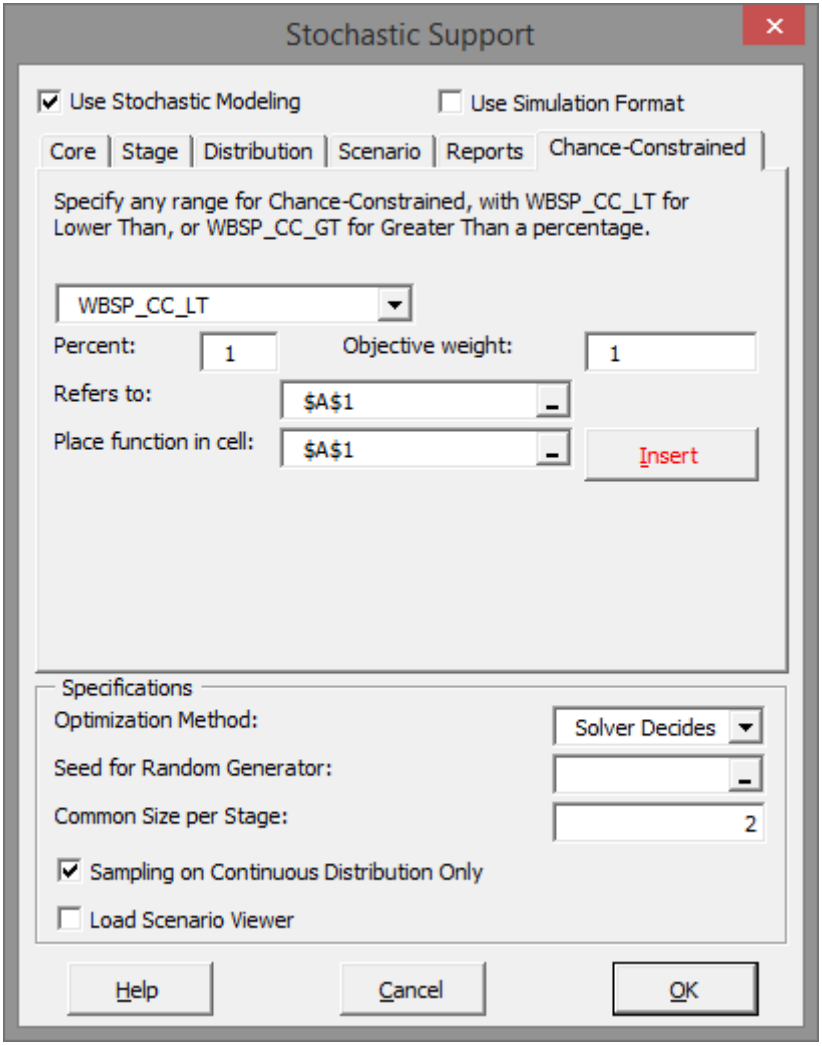
Display Preview Window

Once the solver has completed the stochastic calculation, the user can have a preview of the cells to report by selecting the scenario number. By default, the solution values of the scenario 1 will written back into the spreadsheet, but another scenario solution can be selected. This will allow a closer look of a particular scenario when reading the spreadsheet.

The cells are selected via the same WBSP_REP() function, for either an Adjustable, Variable, Number, Constraint, Random Parameter, or any Dual references. All the scenarios will be anyway written in the *WB!_Stochastic* sheet, is this report was previously requested.



Tab Chance-Constrained



This page is useful to setup a stochastic model in Chance-Constrained format, which is the second major class of models in stochastic programming. The goal in Chance-Constrained Programming is to make an optimal decision prior to realization of random data, while controlling the chance that constraints are violated in one stage.

Extensive explanation on Chance-Constrained Programming are given in the *Guidelines for Chance-Constrained Modeling.*

Specify Chance-Constrained Using the Function WBSP_CC_LT() or WBSP_CC_GT():

The general format is =WBSP_CC_LT(*percent*, *weight*, *constraint cells*), where LT for "Lower Than", or =WBSP_CC_GT(*percent*, *weight*, *constraint cells*) with GT for "Greater Than".

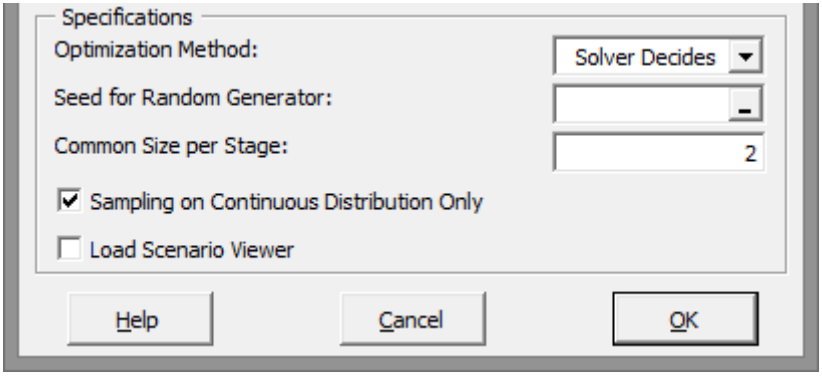
Percent:
This is the amount by which the constraints assigned to this set must be satisfied in the scenarios. Otherwise, the scenario will be considered as unsatisfied, according to to the direction function.

Objective Weight:
The amount by to give priority to the objective value.

Refers To:
Specify the constraint cells on which to apply the Chance-Constrained feature. You can select multiple ranges in the same function which will be treated as being in the same set. Otherwise, specifying different functions, the solver will treat the Chance-Constrained functions as different set. The reference ranges are entered into the text box.

Place function in cell:
Specify the cell in which to write the WBSP_ function. The reference ranges have to be entered into the text box.

Frame Specifications
The Stochastic Solver option allows you to solve models in which some cells are random variables. The frame posted by the Options...|Stochastic Solver command appears as follows:



Optimization Method:

Indicates the method you would like the stochastic solver in What's*Best!* to employ using the Optimization Method drop-down box. Possible choices include: Solver Decides, Free, Deterministic Equivalent, Nested Benders Decomposition, Augmented Lagrangian Decomposition, and Simple Benders Decomposition. The Solver Decides setting currently defaults to the free method.

The default setting for Solver Method is Solver Decides.

Seed for Random Generator:

This is the Seed to initialize the random number generator. Possible values are positive integers, and can be set via a single cell reference, or a number.

The default setting for Seed is 1031.

Common Size per Stage:

This is the default number of scenarios to generate per stage. Possible values are positive integers.

The logical states will behave as follow:

	"Sampling on Continuous Only" Checked	**"Sampling on Continuous Only" Not Checked**
Function =WBSP_STSC() Not Defined	Use of Common Size per Stage for Continuous, and the Defined Table Size for Discrete Distributions	Use of Common Size per Stage
Function =WBSP_STSC() Defined	Use of Function Argument for Continuous, and the Defined Table Size for Discrete Distributions	Use of Function Argument

The default setting for Common Sample Size is 2.

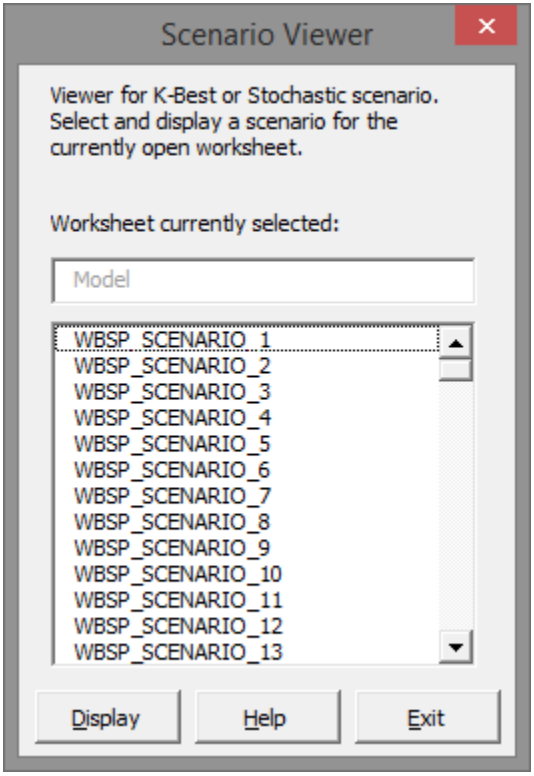
Sampling on Continuous Distribution Only

In case of a model with both distributions, continuous and discrete, this flag enables the sampling technique only on the continuous distribution, leaving the discrete distribution with its own set of data. Sampling can be applied with the spreadsheet function WBSP_STSC().

The default setting is on; meaning sampling is applied on continuous distributions.

Load Scenario Viewer:

The scenario viewer will give the list of all the Adjustable and Parameters cells, but only for the worksheet tab currently selected, via the Advanced...|Scenario Viewer menu. This tool can be useful to scroll through the different Stochastic Programming scenarii without writing additional WBSP_REP() functions.



With the "Load Scenario Viewer" checked, the user will then need to open the dialog box via the Advanced...|Scenario Viewer menu. First select the "model" tab, then select the Scenario Viewer via the menu. The scenarii are being presented for all the Adjustable and Parameter cells in the currently selected "model" tab. Finally display or double click the scenario to write the values directly into the worksheet.

Note: The "Preview Window" shows the values specified via the function WBSP_REP() for any cells in the workbook. The "Scenario Viewer" menu shows the Adjustable and Parameter cells for only the selected worksheet tab.

Use Simulation Format

By checking this option, the model will be processed for simulation. Based on a stochastic model, the user can set up the random parameters and their associated distributions at a stage 0, eventually any additional formulation, and What's*Best!* will calculate the values of the reporting cells. There is

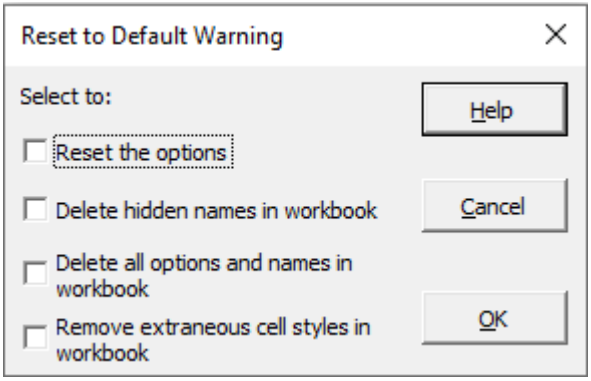
no need of defining an adjustable, constraint, or objective cells. This feature will be useful for knowing the behavior of some cells along the scenarios, and to display their histograms.

The default setting is False (unchecked).

Options...|Reset to Default

This command resets all workbook options to their default values. Such options can be seen in the *Insert|Name|Define… or Name Manager* menu as *WBxxx* names. This command does not restore options defined in the *Advanced Parameter* window to default values. See the section entitled *Options and Solvers* for more information about workbook options.

Upon first use, the *Reset to Default* command displays the following dialog box:



"Reset the options" will remove the *WBxxx* names in the workbook. These names belong to What's*Best!* and were created via the *Options* dialog boxes.

"Deleting hidden names" will only remove names in the workbook that are not visible or listed via the Name Manager of Excel®.

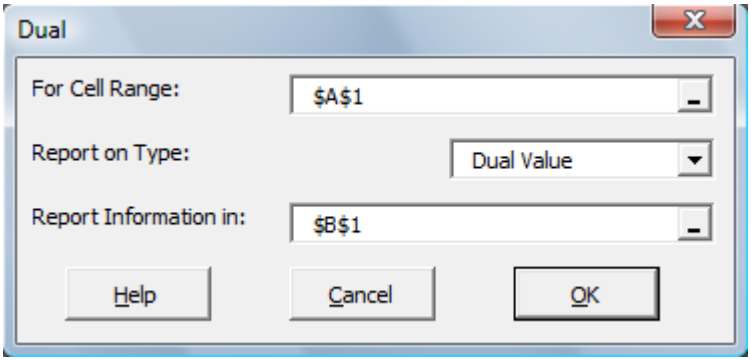
"Deleting all names and options" will actually remove any names from the workbook.

These names could have been defined by What's*Best!*, the user, Excel®, or other applications saving names in this workbook file. Removing these visible or invisible names will clean up the file, and the user has to redefine the useful names.

"Remove extraneous cell styles in workbook" will actually remove any non built-in styles from the workbook, but keeps the "Adjustable" style. These user's defined styles can be seen via the menu Home→Cell Styles, and may have been duplicated by thousands, slowing down the process of adding a new style.

Advanced...|Dual...

The dialog box posted by the *Advanced...|Dual* command appears as follows:



The *Dual* command allows for reporting of dual values for the chosen cells or range of cells. If you are requesting dual values for a range of cells, then the range in *Report Information in:* should be identical in dimension to the range in *For Cell Range:*.

What's*Best!* provides three dual value options in the *Report on Type* box. The first, *Dual Value*, lets you define the dual value of a cell or range of cells. The formula is *=WBDUAL(cell, 0)*. The 0 will be replaced by the dual value of cell after solving the model and this value will be displayed in the cell containing *=WBDUAL( )*. The other two, *Upper Range* and *Lower Range*, let you define the upper and lower ranges for dual values. The formulas are *=WBUPPER(cell, 0)* and *=WBLOWER(cell, 0)*. The study of these values and their effect on the solution of the model is called sensitivity analysis. Please refer to the section *Usage Guidelines for Dual Values* for a tutorial and information on how to use this option effectively.

For Cell Range

This is the target range of cells you wish dual reporting on. What's*Best!* will automatically fill this box with the cells directly to the left of the current selection of cells, or above them if the entire row is selected. If this is not the range of cells you want, you can type the correct range in or select the button on the right edge of the text box to bring up a cursor for cell selection.

Report on Type

Dual Value

Select *Dual Value* to prompt What's*Best!* to return the dual value of the adjustable or constraint cells specified in *For Cell Range:*.

To enter a dual value function, put the cursor in the cell for which you want to find the dual value and select *Dual Value*. Then, specify the cell in which you want the dual information displayed in the *Report Information in:* box.

Note:	When taking the dual value of a constraint, be sure to specify the cell containing the constraint *function*, not the right-hand side of the constraint.

Upper Range and Lower Range

Select *Upper Range* or *Lower Range* to cause What's*Best!* to return the upper and lower valid ranges for a dual value. That is, the upper and lower bounds of the range over which the dual value of a constraint or adjustable cell stays the same.

Note:	Calculating upper and lower ranges may significantly increase solution times, so it is good to weigh the benefits of the additional information against the amount of extra time it will take to get that information. This runtime consideration is particularly relevant for large models.

Report Information in

Specify the host range of cells you would like to see the dual information placed in the *Report Information in* text box. What's*Best!* will automatically fill this box with the current selection of cells.

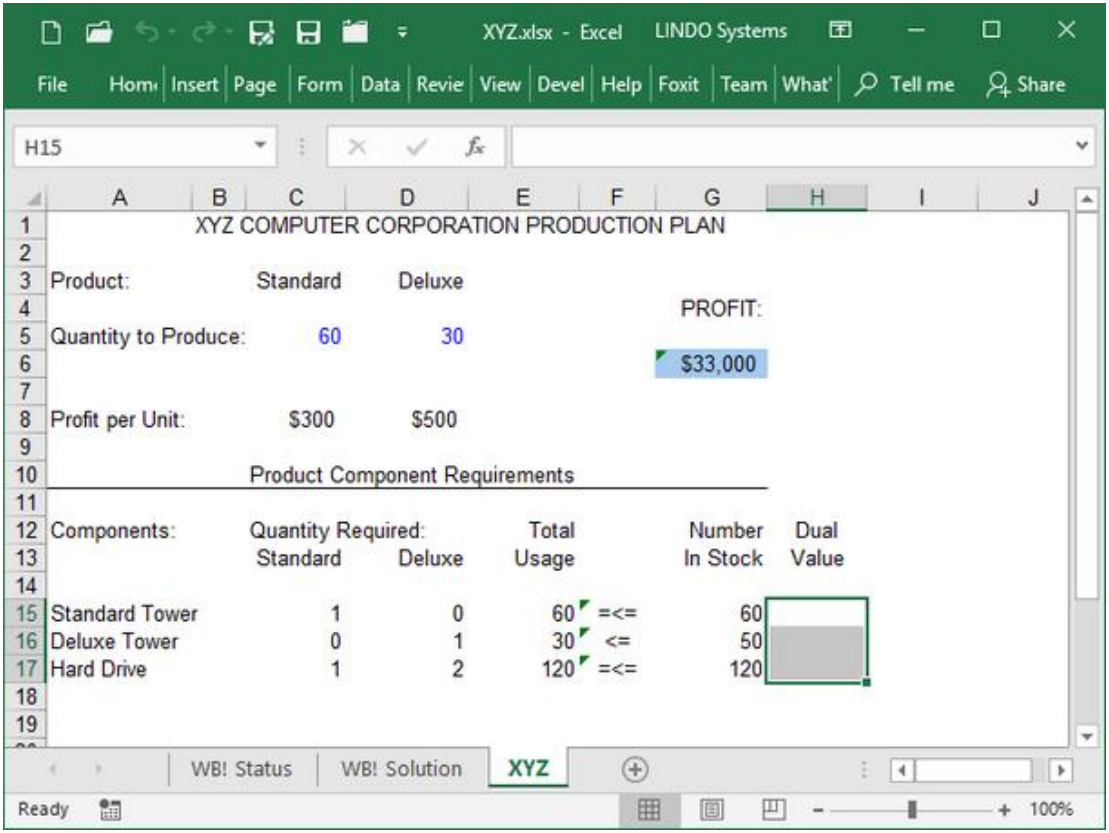
Usage Guidelines for Dual Values

Dual values can provide valuable sensitivity information. They can give you a feel for how sensitive a solution is to a particular constraint cell or adjustable cell. What's*Best!* can provide dual information on any adjustable cell, constraint cell, or any cell that is a function of adjustable cells.

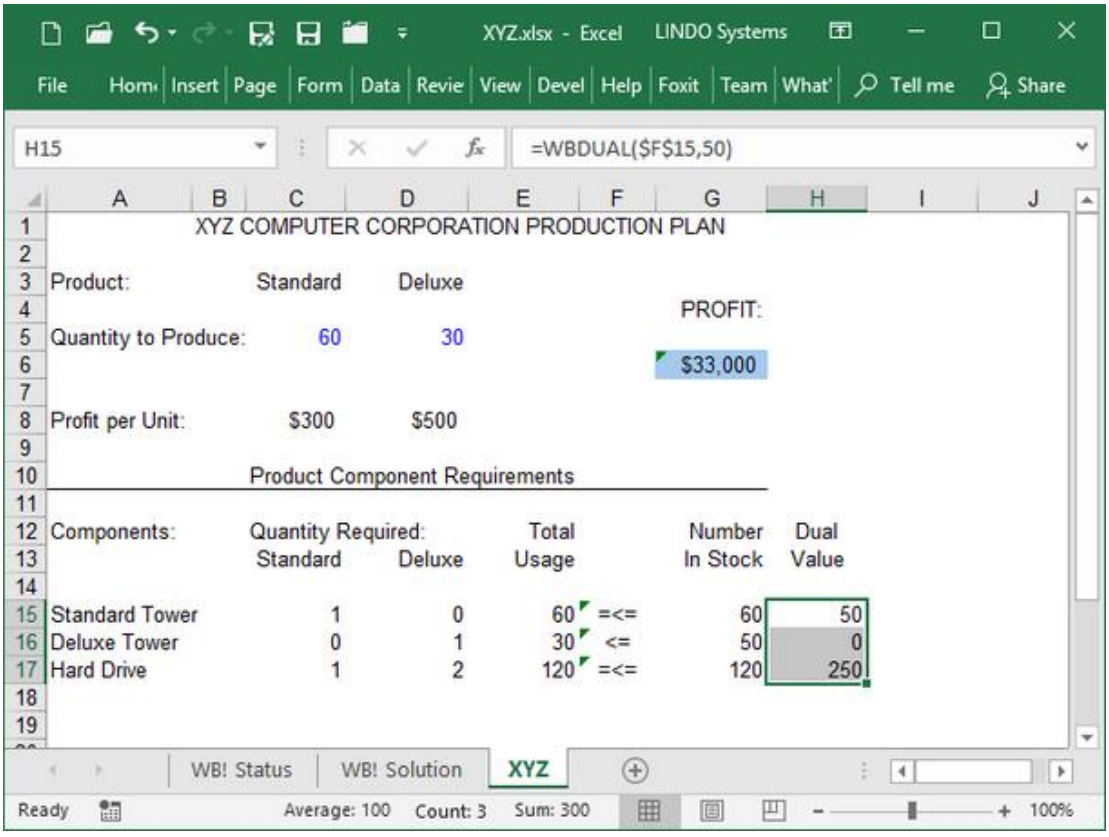
Dual Value of a Constraint Cell - Shadow Price

The dual value of a constraint is the rate of *improvement* in the best cell as the right-hand side of the constraint is increased. The dual value of a constraint is often referred to as the shadow price because it foretells the price you should be willing to pay for that item.

For example, in the XYZ sample model referred to earlier in *Getting Started*, dual values can be specified for the constraints in cells F15:F17. These could be put in any convenient place on the spreadsheet. We chose H15:H17 below:



The dual cells are inserted by highlighting the cells H15:H17, clicking on *WB!|Advanced...|Dual*, selecting F15:F17 for the *For Cell Range:* box, and clicking *OK*. When we re-solve, the dual values for the stock constraints will appear as in the following.



The dual value of the constraint function *WB*(E15,"<=",G15) is shown in cell H15 as 50 (Note: You must re-solve your model to allow What's*Best!* to update any new dual cells). This means the value of the best cell would improve by $50 for each unit increase in the right-hand side of the constraint equation. In other words, changing the cell G15 to 61 and re-solving the model would return a new solution with a maximum profit of $33,050 (assuming a valid range on the dual value of at least one). This is also the maximum price at which you should be willing to purchase an additional Standard computer tower.

Note: In What's*Best!* we use the convention of having dual values on constraints return the *rate of improvement* in the objective for an increase in the constraint's right-hand side value. A positive dual implies that the objective will improve when increasing the right-hand side, while a negative implies that the objective will suffer. On maximization problems, as we have shown above, a positive dual value means that increasing the right-hand side of the constraint will cause the optimal objective to improve, or *increase*. On the other hand, for minimization models, if the sign of the dual is positive, then the optimal objective value will *decrease* as the right-hand side increases.

Note: If the constraint on Deluxe computer tower usage *WB*(E16,"<=",G16) is not tight (i.e., E16 is not equal to G16, but is less than 50), then the dual value is 0. Increasing the right-hand side from 50 to 51 would not affect the optimal solution and consequently would not change the best cell value.

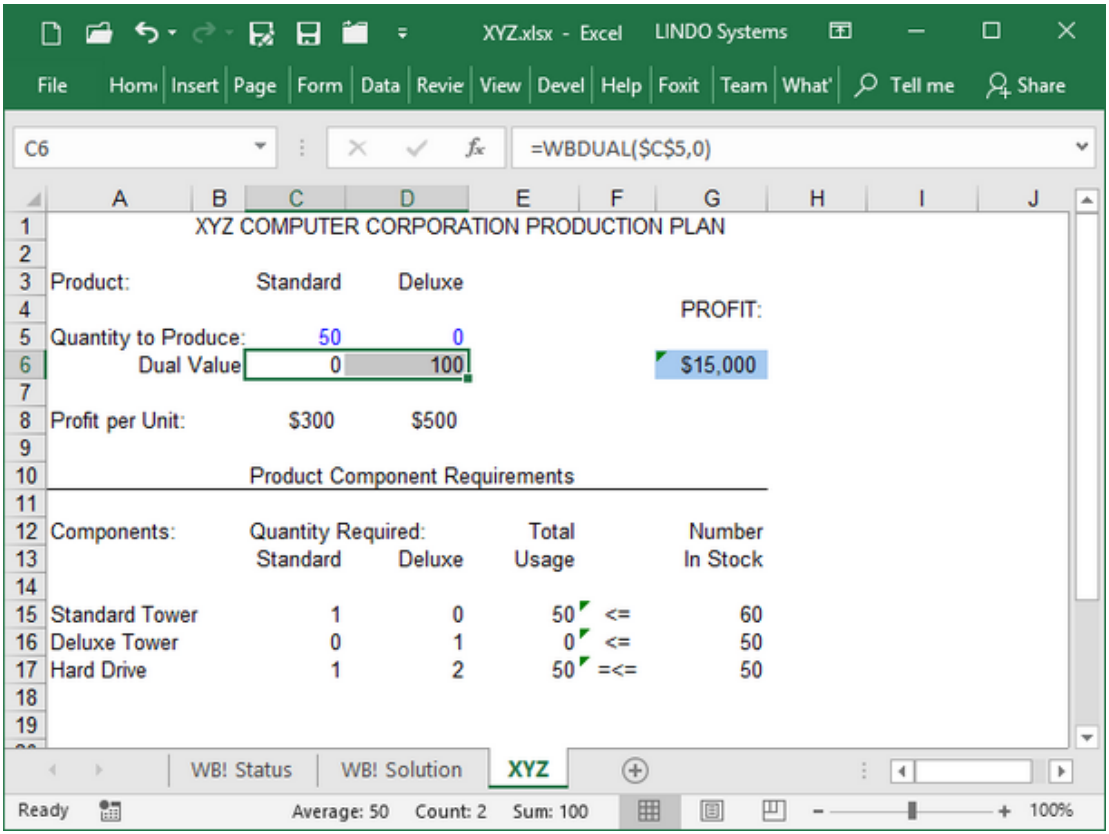
Note: When taking the dual value of a constraint, be sure to specify the cell containing the constraint *function*, not the right-hand side of the constraint.

Dual Value of an Adjustable Cell - Reduced Cost

The dual value of an adjustable cell is the rate at which the best cell would be *penalized* if an adjustable cell with an optimal value of zero were forced into the solution–even though it is not optimal to do so. If the adjustable cell is a positive number, the dual value will always be 0. The dual value of an adjustable cell is often referred to as the *reduced cost* because it is the amount by which the cost of producing that adjustable item would have to be reduced in order to make it profitable.

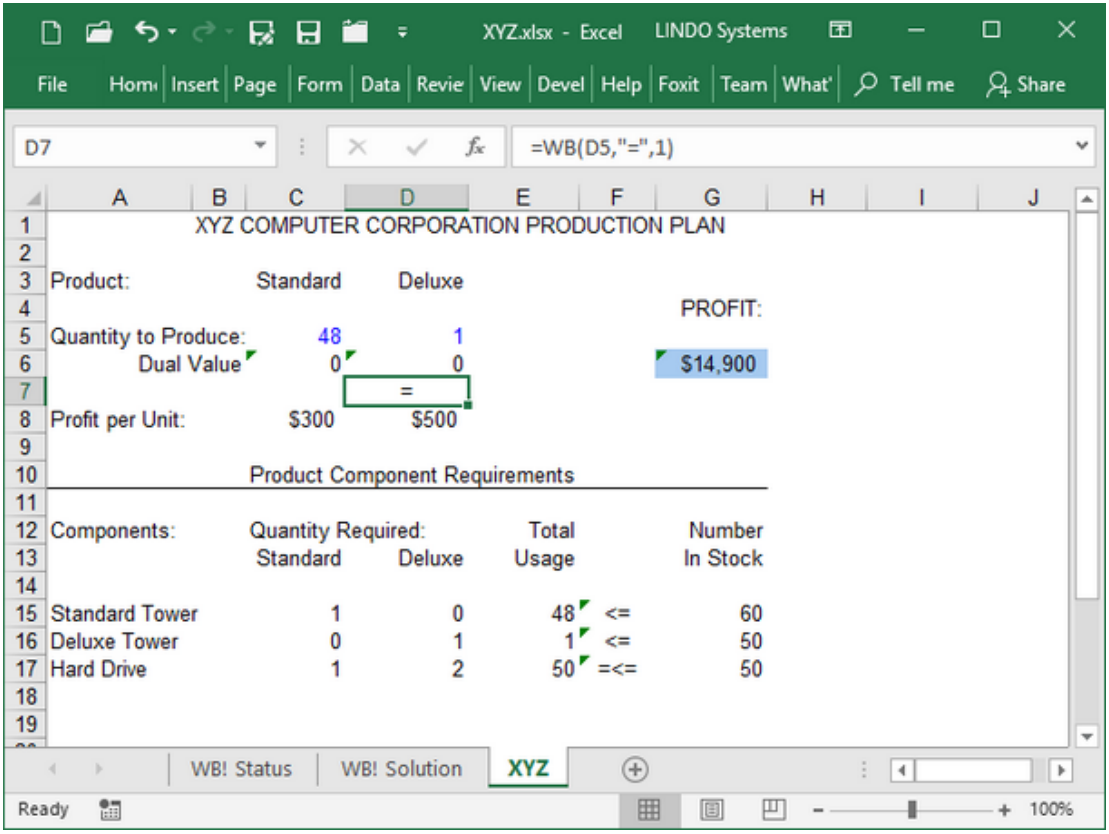
The dual values for the adjustable cells can be placed in the model by clicking on the cells to put the information in, choosing *Advanced...|Dual*, entering the adjustable cells in the *For Cell Range* edit box, and clicking *OK*. The dual values can be placed in any convenient location in the spreadsheet.

For illustration, we continue with the XYZ model, changing the available number of hard drives (G17) to 50 and re-solving, causing the number of Deluxes (adjustable cell D5) to drop to 0 and the total profit to be reduced to $15,000. We also requested dual values on our two adjustable cells, placing the dual values in cells C6:D6. After re-solving we get the following results:



The dual value of adjustable cell D5 is now 100. This means the value of the best cell would be penalized (in this case decrease) by $100 for each unit increase in D5. In other words, forcing the number of Deluxe computers from 0 to 1 would return a new solution with a profit of $14,900 ($15,000 - $100) (assuming a valid range on the dual value of at least one—see the *Valid Ranges for Dual Values section below*).

In the window below, we show the re-solved model with the Deluxe adjustable cell forced to be 1 by putting the constraint =*WB*(D5,"=",1) in cell D7. As you can see, the dual value is now zero because the variable is in the solution. The quantity of Deluxe computers to produce could also be forced to one by choosing *Remove Adjustable* in the *Adjustable* dialog box or the toolbar button to make D5 into a non-adjustable cell and entering 1 as a constant value.



Note: It is important to note that the dual value of a particular constraint or adjustable cell assumes all other information in the model remains unchanged. That is, given the dual information, you cannot predict the effect on the best cell of simultaneously changing two, or more, constraints.

Dual Value of Zero and Multiple Optima

Dual values are generally positive for adjustable cells that have a zero value in the optimal solution. An exception to this occurs when there are multiple optima (i.e., more than one combination of adjustable cell values that yield the same optimal objective value) because no penalty is incurred in moving from one optimal solution to an alternative one. If your model yields zero dual values for any of the adjustable cells, it's likely that there are multiple optima for your model.

Dual Values in Nonlinear Problems

Dual values in a nonlinear problem can be interpreted usefully only for small changes in the right-hand side. It may not be possible to make assumptions about the effect of large changes in a nonlinear model. You may want to investigate the effect of such changes manually by inserting proposed changes and re-solving.

Users with knowledge of calculus can think of the dual value as a derivative — the rate of change of the best cell value with respect to changes in either the right-hand side of a constraint or an adjustable cell value. As with the derivative of a linear function, a dual value in a linear model has a constant value over a range. In nonlinear models, the dual value is similar to the derivative of a nonlinear function–it is valid at the point where it's evaluated, but it may change immediately as you move from that point.

Dual Values in Integer Problems

Dual values for problems with integer variables cannot be usefully interpreted because of the way the branch-and-bound technique solves integer models. If a dual value is requested in an integer model, What's*Best!* will return a value. However, this value is of no practical use.

Valid Ranges for Dual Values

You may also wish to determine the upper and lower ranges of any specified dual value. Changing the right-hand side of a constraint or using more or less of a resource, as described above, eventually will cause a change in the dual value. The valid range for a dual value is the amount of change in the right-hand side of a constraint, or in resource use in an adjustable cell, that is possible before a change in the dual value will occur.

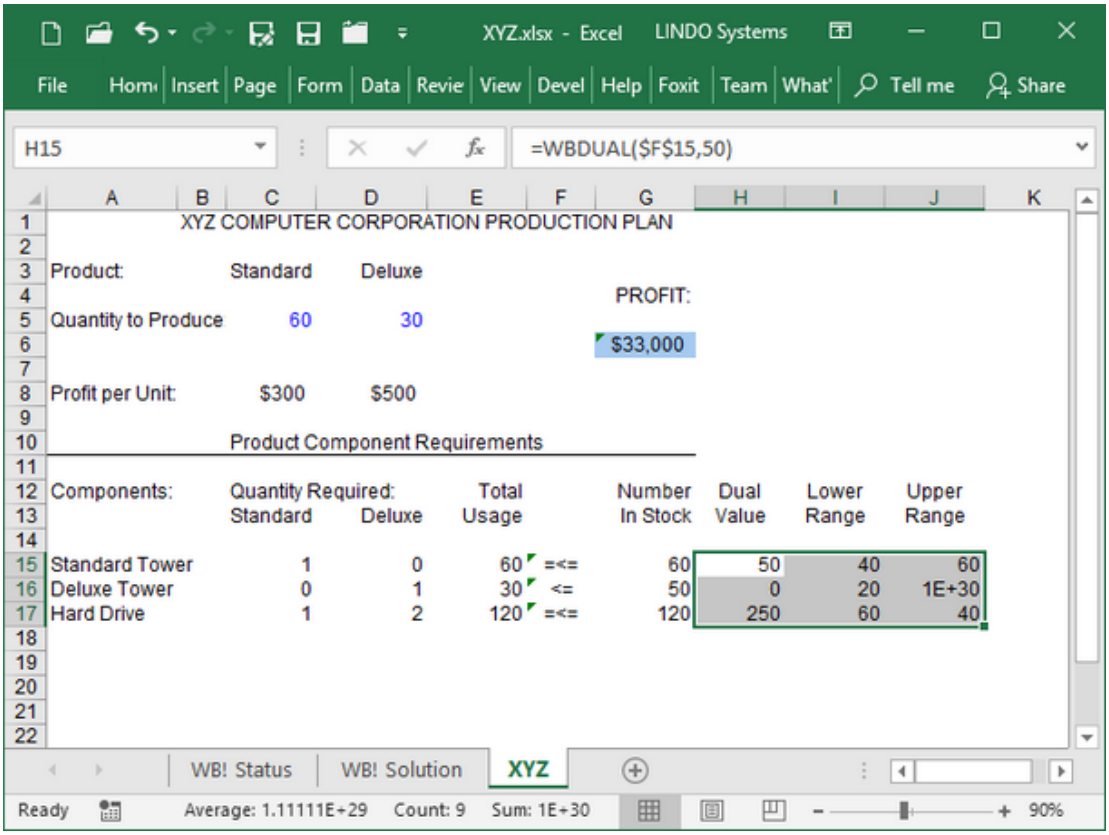
The range over which the use or availability of a resource, or the right-hand side of a constraint, can decrease before the dual value changes is called the *lower range*. The range over which the use or availability of a resource, or the right-hand side of a constraint, can increase before the dual value changes is called the *upper range*.

Constraint Ranges

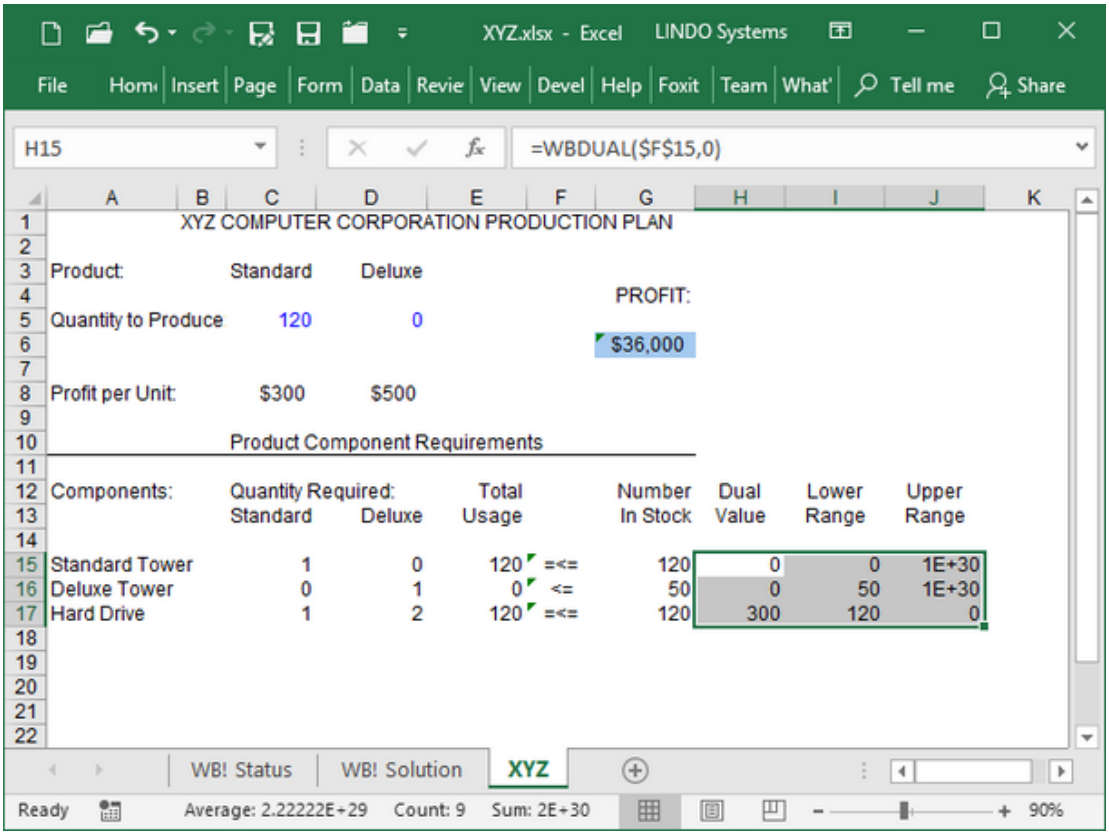
For example, building on the XYZ example, an upper and lower range can be determined for the dual values of the stock constraints. These ranges are put into the model by highlighting the cells you would like the range to appear in, selecting *Advanced...|Dual*, specifically selecting the constraint cells (F15:F17) to display in *For Cell Range*, choosing *Upper Range* or *Lower Range*, whichever is appropriate, and clicking *OK*.

Note: The default value entered in the *For Cell Range:* is G15:G17, which is not the range of the constraint cells. You must make the correction to F15:F17 in order to obtain the true range values.

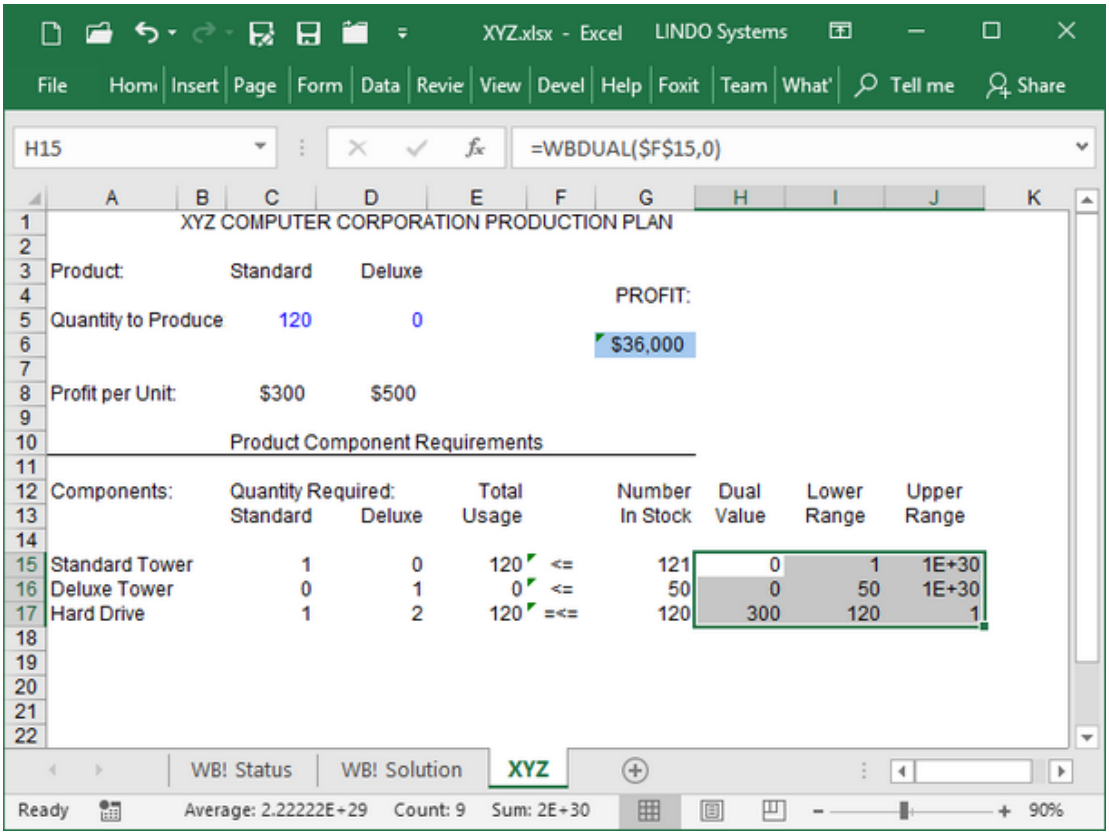
In the illustration below, we used cells I15:I17 and J15:J17 for the upper and lower ranges respectively and then re-solved. Again, we could have placed the dual value cells in any convenient location in the workbook.



The ranges for the Standard computer tower stock constraint are 60 upper and 40 lower. That is, the dual value will stay at 50 as the number in stock varies from 20 (60 - 40) through 120 (60 + 60). If the right-hand side of the equation is increased to the limit of 120 and the model is re-solved, the profit increases to $36,000 ($33,000 + (50*60)) as shown in the following:



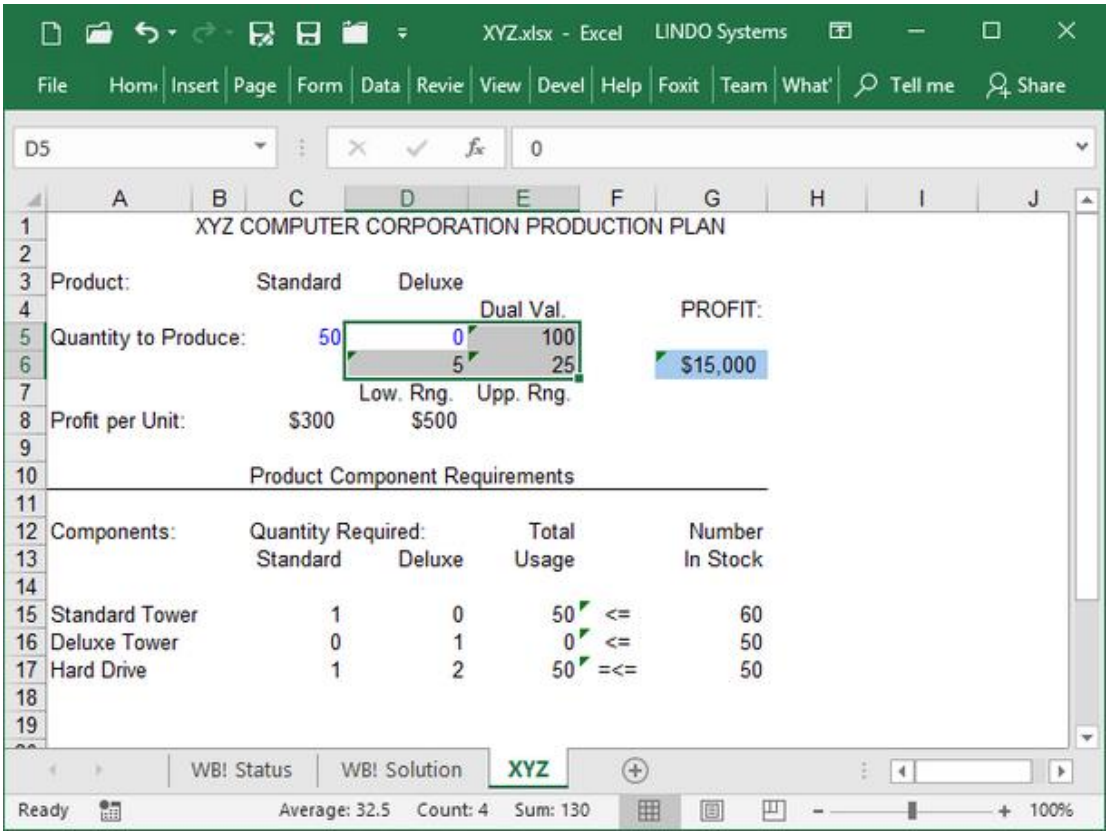
Increasing the right-hand side of this constraint beyond 120 exceeds the upper range for the original dual value of 50 and a dual value of 0 is now in effect. Since the dual value is $0, changing cell G15 to 121 and re-solving the model does not increase the profit beyond $36,000 as shown in the following:



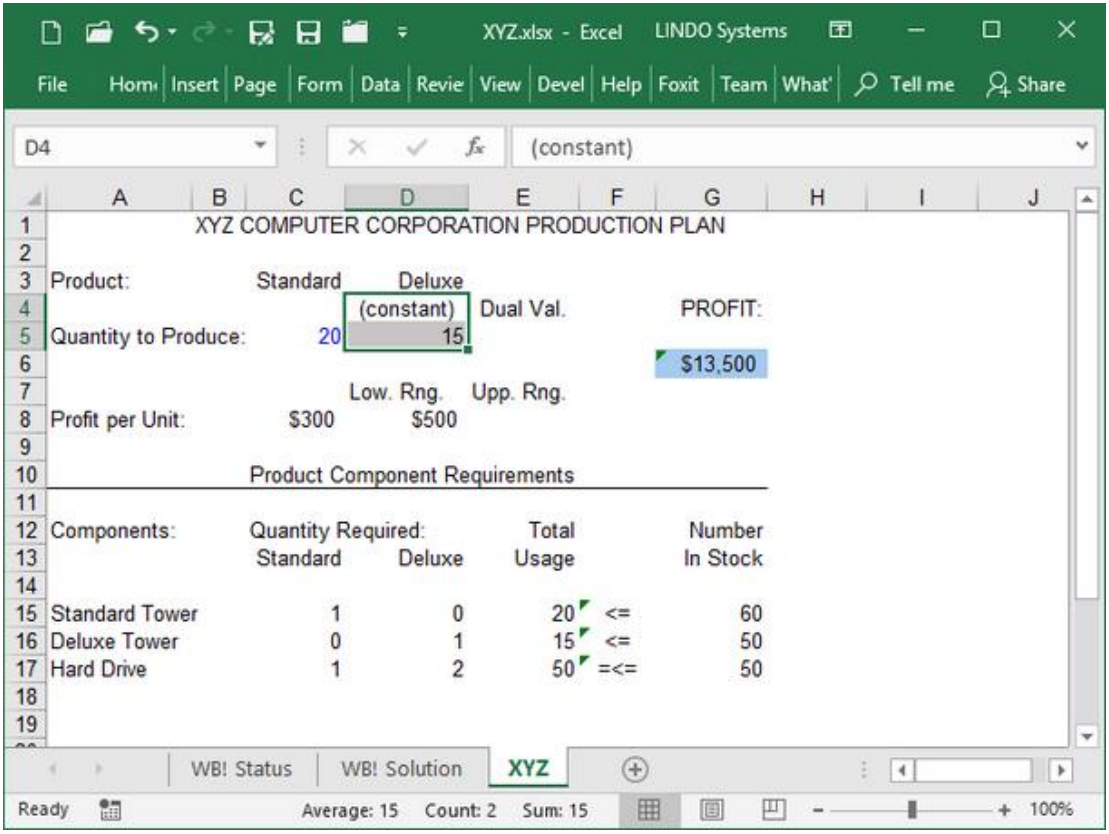
Adjustable Cell Ranges

In the XYZ Production example, the adjustable cells refer to the production of Standard and Deluxe computer tower models. The dual value of each of these adjustable cells is the amount by which the profit of producing the models would have to increase in order to make it profitable to put them into the solution. The upper and lower range of the dual of an adjustable cell is the range over which the adjustable cell could change with its dual value remaining the same.

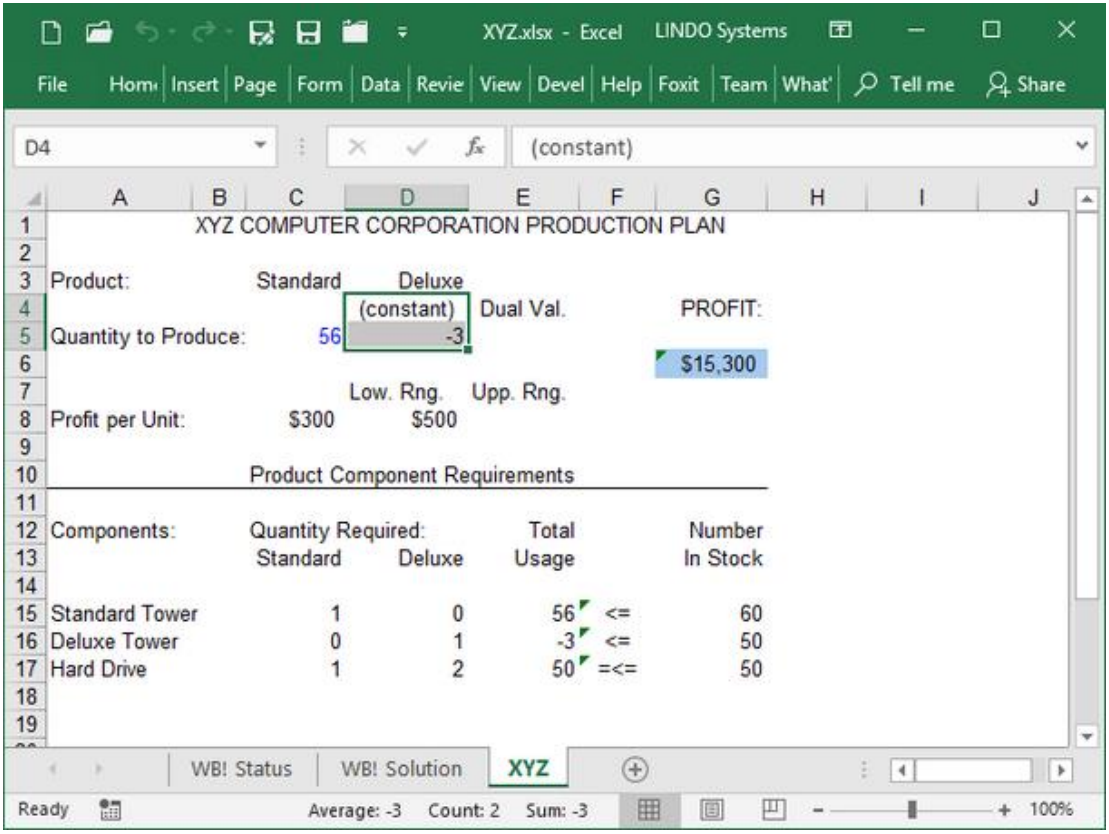
We will use the example referred to earlier for the adjustable cell dual value where the number of hard drives in stock is reduced to 50. An upper and lower range value can be put in the model by selecting the cell to put the dual information in, selecting *Advanced...*|*Dual*, selecting the adjustable cells (D5) to enter in the *For cell range:*, choosing *Upper Range* or *Lower Range*, whichever is appropriate, and clicking *OK*. In the box below, we used cells D6 and E6 for the lower and upper ranges respectively and then re-solved.



The ranges for the number of Deluxe computer towers to produce are 25 upper and 5 lower. That is, the dual value will stay at 100 as the quantity of Deluxe computers to produce varies from -5 through 25. If cell D5 (Quantity to Produce) is increased to 15, made non-adjustable using the *Remove Adjustable* command via the *Adjustable* dialog box or the toolbar button, and the model is re-solved, then profit decreases to $13,500 ($15,000 - (100*15)) as shown in the following.



Note that, since the dual value formulas now refer to a non-adjustable cell, What's*Best!* ignores these cells and they are not changed. If the quantity of Deluxe computers to produce is decreased to -3 and the model is re-solved, the profit increases to $15,300 ($15,000 - (100*-3)) and the number of Standard to produce increases to 56 as shown in the following. This might occur if there are three obsolete Deluxe models and they are taken apart for the common parts used in the Standard model.



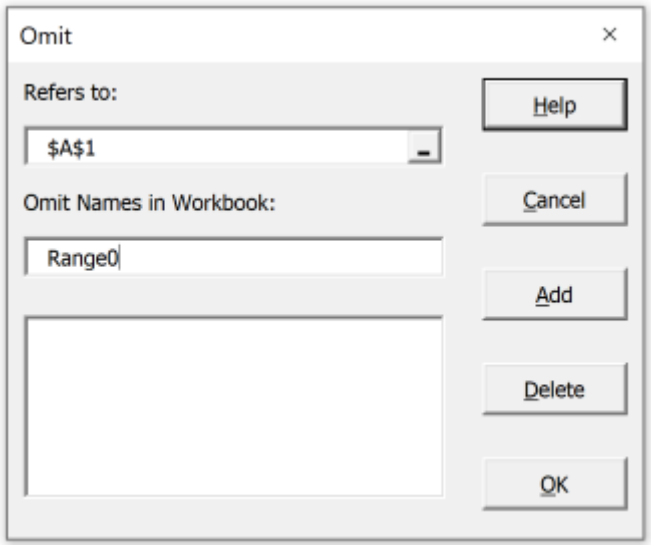
In this problem, if the original lower bound is exceeded by one (-6), the dual value is reduced to zero. If the original upper bound is exceeded (26 or more), the number of hard drives in stock is insufficient and the model is infeasible. Either way, the dual value changes when the range is exceeded.

In some problems, the range of a dual value may be very small or zero. When this occurs, you may find that different starting solutions result in different dual values. This may also indicate that more than one combination of the adjustable cells will result in the optimal value of the best cell. In this case, your model is said to have multiple optima. In any event, before making use of dual values in decisions of economic importance, you should investigate the range over which they're valid.

To retrieve dual values for all adjustable and constraint cells without entering dual value formulas in the worksheet, you may request a solution report with the *Options…*|*General* command.

Advanced...|Omit...

The dialog box posted by the *Advanced…|Omit* command appears as follows:



The *Omit* command allows you to specify a range of cells on the workbook to be ignored by the What's*Best!* solver while it is seeking a solution. These omitted areas do not contribute toward any limits What's*Best!* imposes in restricted size versions.

If used properly, omit ranges can decrease the solution time. However, care should be exercised in applying omit ranges. Please refer to the section entitled *Usage Guidelines for Omit*.

OMIT Names in Workbook and Refers to

What's*Best!* uses range names to specify the cell range to be omitted. The target cell range is indicated in the *Refers to:* box. What's*Best!* automatically fills the *Refers to:* box with the range of the currently selected cells. If this is not the range of cells you want omitted, you can type the correct range in or select the button on the right edge of the text box to bring up a cursor for cell selection. Next, enter any word of your choice in the *OMIT Names in Workbook* box and click on the *Add* button. The name you entered should then appear in the box below with a *WBOMIT* prefix.

If you decide later you would like to remove the omit restriction, simply go to the *Omit* dialog box, select the name of the omit range to remove from the list of names, and click on the *Delete* button.

Usage Guidelines for Omit

What To Omit

You may use the *Advanced…|Omit* command to remove parts of your workbook from What's*Best!*'s consideration, thereby decreasing the time to reach a solution. If you can identify cells with a numeric value or equations that are extraneous to the problem being solved, then you can place them in an omit range. For example, some cells in the spreadsheet may be used for evaluation and reporting and are not used in finding the solution. If these cells contain unsupported spreadsheet functions, placing them in an omit range will eliminate the error message caused by the equations. If they contain equations that depend upon adjustable cells, placing them in an omit range can shorten the time required for solving.

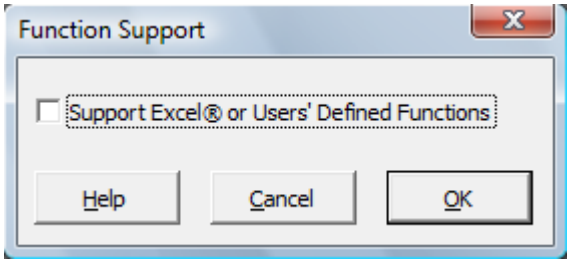
What Not To Omit

What's*Best!* will ignore the values and equations of every cell within an omit range. Therefore, it is very important that information pertinent to the problem is not included in an omit range. For example, you should be especially careful not to include:

- *Adjustable cells* - Any adjustable cell within an omit range will not be adjusted during the solution process.
- *Constraint cells* - Any constraint cells included in an omit range will be ignored during the solution process.
- *Cell Precedent to Equations Outside Omit Range* - An equation outside of any omit range that references a cell within an omit range will cause the solution to be aborted with an error message and the unsolved model will be returned.

Advanced...|Function Support

This feature allows you to build models using not only Microsoft® Excel® built-in functions, but also user defined functions. The user defined functions can be VBA macros, add-in file .XLA, macros calling a user add-in file .XLA, .XLAM, or a library file .XLL. This feature is useful if the user has to develop his own function and wants these functions to be integrated in the optimization model.
The function support and add-in names are entered using the following dialog box:



Function Support

By checking this option, every function will be supported in the model. Otherwise, What's*Best!* will treat user written functions as unsupported functions, and will read the cell as a constant equal to its numeric value at the start of optimization.

This support of user functions will make the model bigger, more complex, and so may make solution time longer.

In Microsoft® Excel® 2002, you may need to adjust a setting from the menu "Tools|Options|Security|MacroSecurity|TrustedSources" so as to grant any project access. Thus grant any project access. Check the box "Trust Access to Visual Basic Project", save your model and reopen Excel®.

In Microsoft® Excel® 2007 and later, adjust the security setting from the Excel® Options, then click on "TrustCenter|TrustCenterSettings". Enable macros and grant the same access in the "Macro Settings" tab.

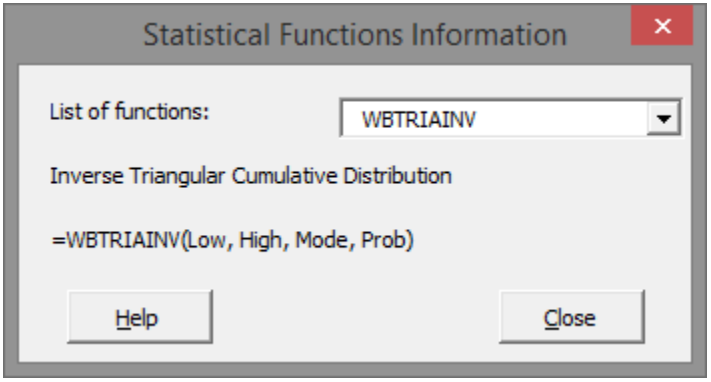
In case of your add-in has been digitally signed, you will need to enable this add-in to run your model. In some situations, a time delay may occur after enabling the add-in.

Refer to the What's*Best!* sample file *Shipping_MacroFunctions.xlsm* for an example of valid user defined function usage.

The default setting is False (unchecked).

Advanced...|Statistical Functions

This feature allows you to view the specific statistical functions available for What's*Best!*.

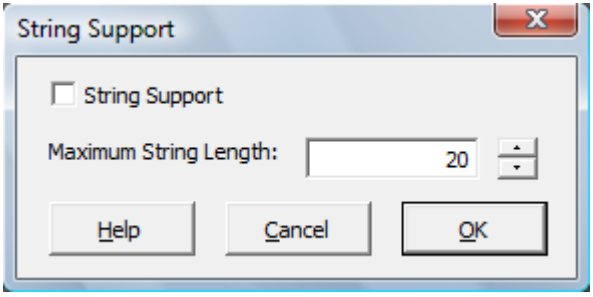


See the part Supported Functions and Operators for additional information.

Advanced...|String Support

This feature allows you to build models using strings as arguments, e.g., in functions such as IF( ), VLOOKUP( ), SUMIF( ), and SUMIFS().

String support related settings are entered using the following dialog box:



String Support

By checking this option, string arguments will be supported in the model. Otherwise, What's*Best!* will treat strings as arguments with a numeric value of zero.

String support may make the model bigger so it may need more resources to be solved.

The default setting is False (unchecked), which means string arguments will be read as a 0 value number.

Maximum String Length

This length can be set between 1 and 255. By making this setting small, you reduce the memory required by the string support feature, however, making the max string length too small may truncate two different strings, such as “New_York” and “New_Jersey”, make them appear to be identical, and lead to incorrect results.

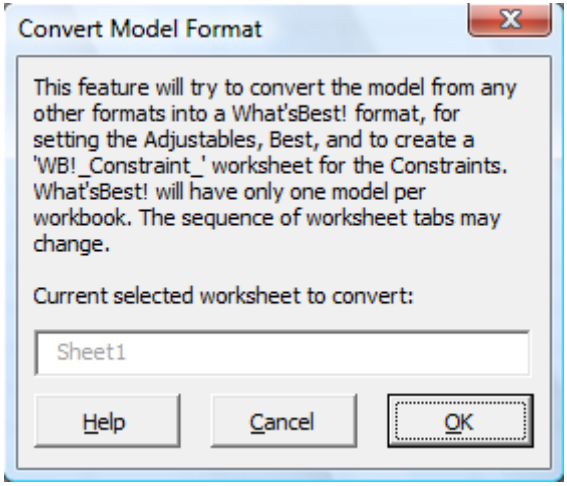
The maximum string length default value is 20 characters.

Note: If the *String Support* option is set to FALSE, then the *Maximum String Length* setting will be disregarded.

String operations are allowed only in variations of compare, such as "=", "< >" (symbol for not equal), ">", "<", ">=", "<=", SUMIF(), SUMIFS(), HLOOKUP(), and VLOOKUP() functions. Any other operations will return an error message. Also, a formula cell should return a numeric number rather than a string value.

Advanced...|Convert Model Format

This feature allows you to convert a non-What’s*Best!* format spreadsheet optimization model to What's*Best!* format. It will try to extract any existing data or information to set the Adjustable, Best cells, and it will create a new tab *'WB!_Constraint_Sheet1'* at the end of the workbook for inserting the constraint functions.



Current Selected Worksheet to Convert

What's*Best!* can have one model per workbook, but other spreadsheet modeling interfaces can have one model per worksheet in the same workbook. The user needs to select the current worksheet to convert, then invoke the feature. The feature will apply the Adjustable Style for the decision variable cells, the WBMIN or WBMAX Name for the Objective cell, and create the '=WB()' functions in a separate tab *'WB!_Constraint_Sheet1'* at the end of the workbook.

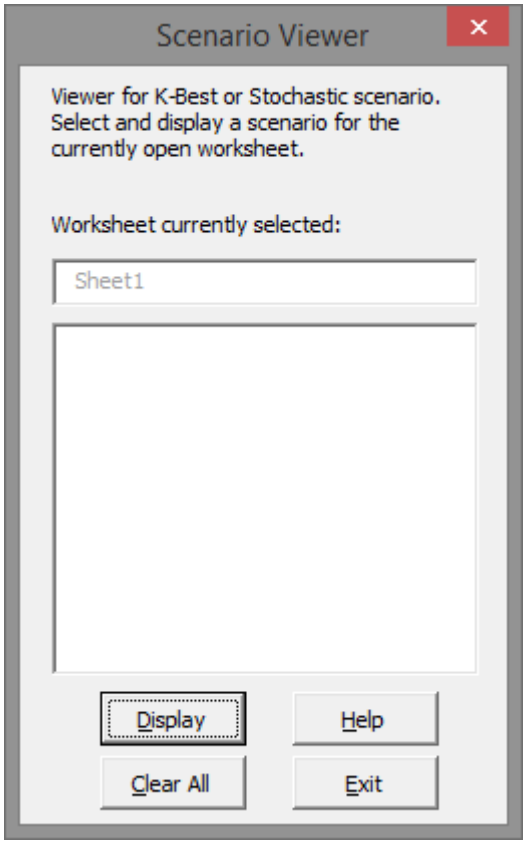
The user may need to reset the tolerances via the numerous *Additional Commands* dialog box for the type of the model. What's*Best!* will then detect automatically the type of the model once the user starts the solver via the *Solve* button.

For instance, the workbook has several tabs, but you want to convert the model in '*Sheet1'*. Select the tab *'Sheet1'*, go to Advanced... and Convert Model Format via the WB! menu, and press OK. You will see the Adjustable cells with a blue-like font, the Best cell with a blue background, and the tab *'WB!_Constraint_Sheet1'* for the constraints functions, such as =WB(Sheet1!A1, "<=", Sheet2!C1).

Note: This feature creates a new model format, but does not remove the existing format. Any cells difference that could not be transported will be listed in the status report.

Advanced...|Scenario Viewer

This dialog box is useful to display the different solution values for a specific worksheet tab, either for an Integer K-Best model or a Stochastic Programming model. This tool works for the Adjustable and Constraints cells written in the same worksheet tab.



For an Integer model designed with the K-Best feature via the Options...|Integer Solver menu, first check for the Load Scenario Viewer, then run the model. After solving, select the worksheet tab containing all the Adjustable, Constraint, and Dual functions cells, and open the Scenario Viewer dialog box. The user will see all the K-Best successive solutions starting with the name WBKB_SOLUTION_x for the specified worksheet. For instance, if there are 5 K-Best solutions, the dialog box displays WBKB_SOLUTION_1 to 5. The user can then select the solution, and the viewer will display the values directly onto the worksheet.

Similarly, this feature works also for a Stochastic Programming model after selecting the Load Scenario Viewer via the the Options...|Stochastic Solver menu. After solving, the dialog box displays all the scenarii as WBSP_SCENARIO_x, where x goes from 1 to the maximum number of scenarii calculated in the model. Again, first select the worksheet tab where all the Adjustable, Parameters, Constraint, Dual or Mean functions cells are located, then invoke the Scenario Viewer.

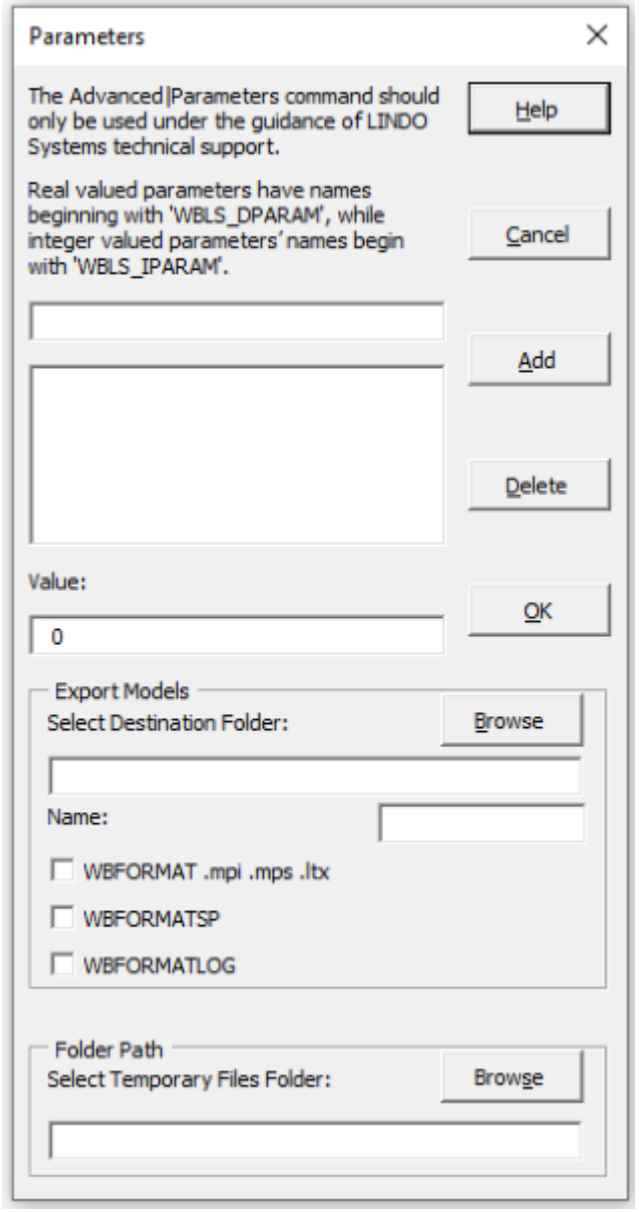
Current Selected Worksheet to Convert

The default worksheet shown is the current tab being selected at the opening of the dialog box. If the user wants to display another sheet, close the dialog box, select a new tab containing the Adjustable, Parameters, Constraint cells, then reopen the Scenario Viewer via the menu.

This feature cannot be invoked via the VBA interface, but loading the scenarii or solutions are possible via the wbSetIntegerOptions, or wbSetStochasticOptions.

Advanced...|Parameters

The dialog box posted by the *Advanced…|Parameters* command appears as follows:


Parameters
The Advanced|Parameters command should only be used under the guidance of LINDO Systems technical support.
Help
Real valued parameters have names beginning with 'WBLS_DPARAM', while integer valued parameters' names begin with 'WBLS_IPARAM'.
Cancel
Add
Delete
Value:
0
OK
Export Models
Select Destination Folder:
Browse
Name:
WBFORMAT .mpi .mps .ltx
WBFORMATSP
WBFORMATLOG
Folder Path
Select Temporary Files Folder:
Browse

LINDO API Parameters

This feature allows you to set arbitrary parameters in the LINDO API, which is the solver engine used by What's*Best!*. In order to solve your model successfully, there may be rare occasions when you will need to set an API parameter that is not available through the normal What's*Best!* interface. If this is found to be the case, you will be provided with a list of settings by a LINDO Systems Technical support representative after investigation of the model. Then, you should enter the parameter names and their values in the Parameters form.

Note: The *Advanced…|Parameters* command should only be used with guidance from LINDO Systems technical support.

The WB!|Options|Reset to Default command will not reset the custom parameters defined in this window. Instead, you must delete them individually using the Delete button pictured above.

There is also a hidden name that the user can setup, as WBCALCxxx with a range value, which actually tells What's*Best!* to read the cells in the range as constant values, and ignoring any formulas in these cells. This name can be useful to bypass the reading of an external link to the model contained in a cell, or an unsupported function, but may also break any dependencies to other parts of the model when What's*Best!* reads and parses the model from the spreadsheet.

All the What's*Best!* names can be defined, modified, or deleted via the Name Manager of Excel®.

Users accessing What's*Best!* via VBA macros can set customer parameters by using the standard VBA code, for instance:

```
ActiveWorkbook.Names.Add Name:="WBLS_DPARAM_ANYPARAM", _
    RefersToR1C1:="=1.23"
```

Export Models

In addition, What's*Best!* will accept two names for providing the instruction list and a parameter file in a standard format.

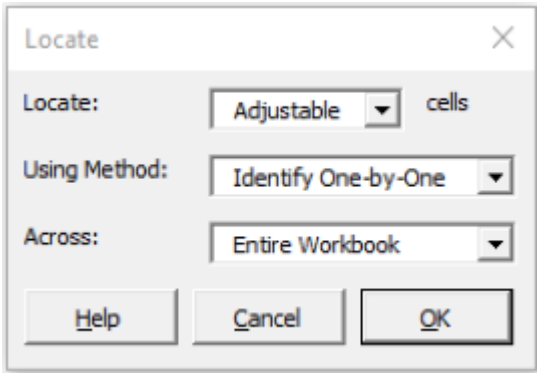
- WBFORMAT, will produce a .MPI, .MPS, or .LTX file format in the specified destination; for instance:
  "C:\Temp\modelname.mpi". The parameter file is provided as a .PAR and .CSV.
- WBFORMATSP, will produce the formatted files related to a Stochastic Program model; for instance:
  "C:\Temp\stochasticmodelname" (without extension), for .MPI, .SMPI, .STOCH, .TIME, _1.STOCH. The parameters file is provided as _PARAM.TXT and _PARAM._PARAM.CSV.
- WBFORMATLOG, will produce the formatted files related to an output log file from the calculation library; for instance:
  "C:\Temp\logname.txt".

Folder Path

What'sBest! generates some temporary files when it solves the model and deletes them at the end of the solution process. The default location for these temporary files is the directory containing the model. You may not have write privileges for storing temporary files in this folder (for example when you retrieve the model from a network drive), so you should specify another local directory for storing these temporary files.

Locate...

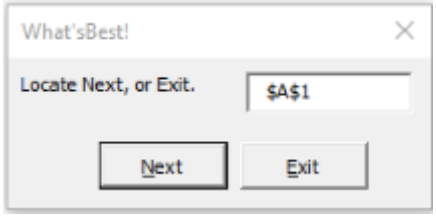
The dialog box posted by the *Locate* command appears as follows:



Specify the type of cell you would like to find in the Locate: drop-down box to search the worksheet or workbook for Adjustable, Best, Constraint, Violated, Dual, WBCALC, or WBOMIT cells.

If you choose to locate *Adjustable*, *Best*, *Constraint*, *Violated*, *Dual*, *WBCALC*, or *WBOMIT* cells, then specify whether you wish to simultaneously Select *All Such Cells* or Identify *One-by-One* with the *Using Method* drop-down box. The search can focus either across the *Worksheet* or the *Workbook*.

To Identify *One-by-One*, What's*Best!* proceeds to search by row following by column until it finds the first such cell. If there are more cells of the same type across the worksheet or workbook, then the following dialog box appears:

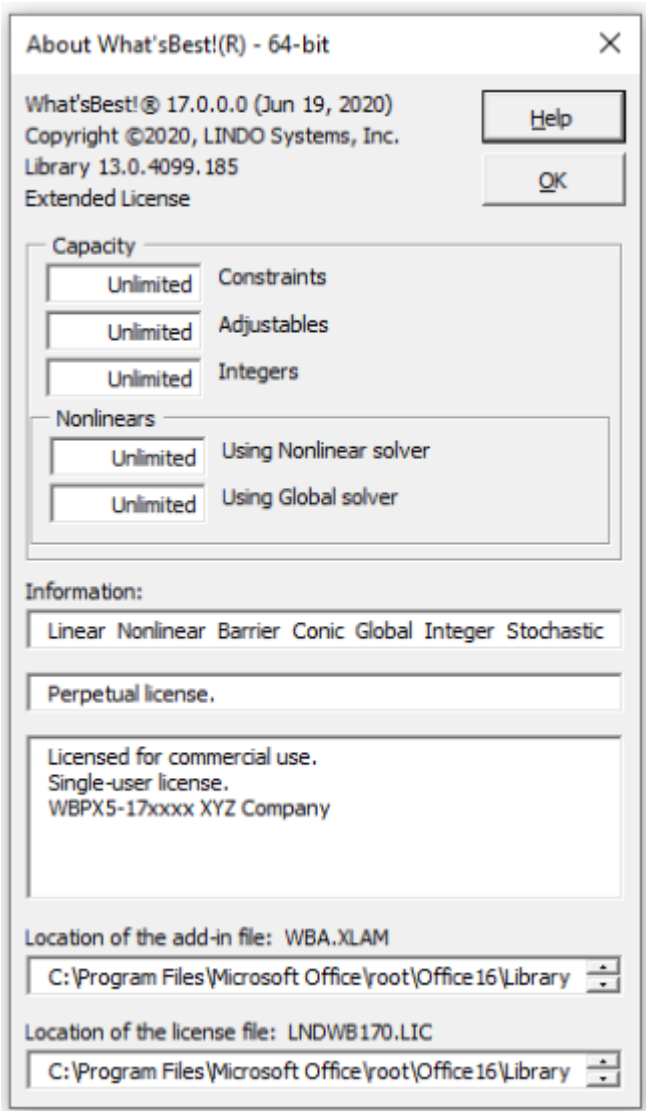


For example, if the current cell is A1, you are searching for adjustable cells one-by-one, and A2 and B1 are both adjustable cells, the next cell selected will be B1.

Finally, select whether you want What's*Best!* to search across the *Entire Workbook* or *This Worksheet* in the *Across:* drop-down box. If no cells of the type you requested are found in the worksheet or workbook, then What's*Best!* lets you know.

About What'sBest!

The dialog box posted by the *About* What's*Best!* command appears as follows:



The *About What'sBest!* dialog box provides information about which version of What's*Best!* you are running and its capacity according to the license key you possess (If you do not have a license key, you can run a Demonstration/Trial version of What's*Best!*). In addition, the box at the bottom of the dialog box supplies the location of the What's*Best!* add-in file (*WBA.XLA* or *WBA.XLAM*) with the necessary accompanying files to run What's*Best!*, and the location of the license file (*LNDWBxxx.LIC*) which has to be in the same location of the add-in. The capacity and size for the different versions of What's*Best!* are as follows:

Name	**Constraints**	**Variables**	**Integers**	**Nonlinear**	**Global**
Trial Version	150	300	30	30	5
Personal Version	250	500	50	50	5
Commercial Version	1000	2000	200	200	10
Professional Version	4000	8000	800	800	20
Industrial Version	16000	32000	3200	3200	50
Extended Version	*Unlimited*	*Unlimited*	*Unlimited*	*Unlimited*	*Unlimited*

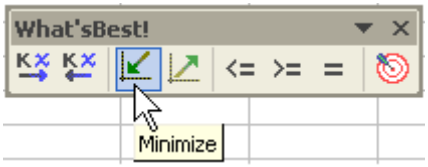
Toolbar / Ribbon Bar

In Excel® versions 2002-2003, use the *ToolBar* command to toggle the What's*Best!* toolbar on and off. When What's*Best!* is installed, the toolbar appears in floating (undocked) mode as follows:



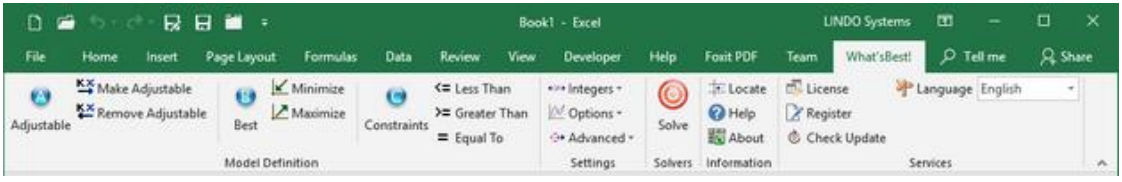
The user may then reposition the What's*Best!* toolbar to a preferred part of the Excel® window. Select the *ToolBar* command to turn the toolbar off. Select it again to turn the toolbar back on. A checkmark will appear next to the *ToolBar* command when the toolbar is on.

The What's*Best!* toolbar offers rapid one-click access to eight of the most commonly used operations. The What's*Best!* toolbar also makes tool tips available. To learn the function of a particular toolbar button, just move the cursor over the button for a second and a statement of the button's function will appear.



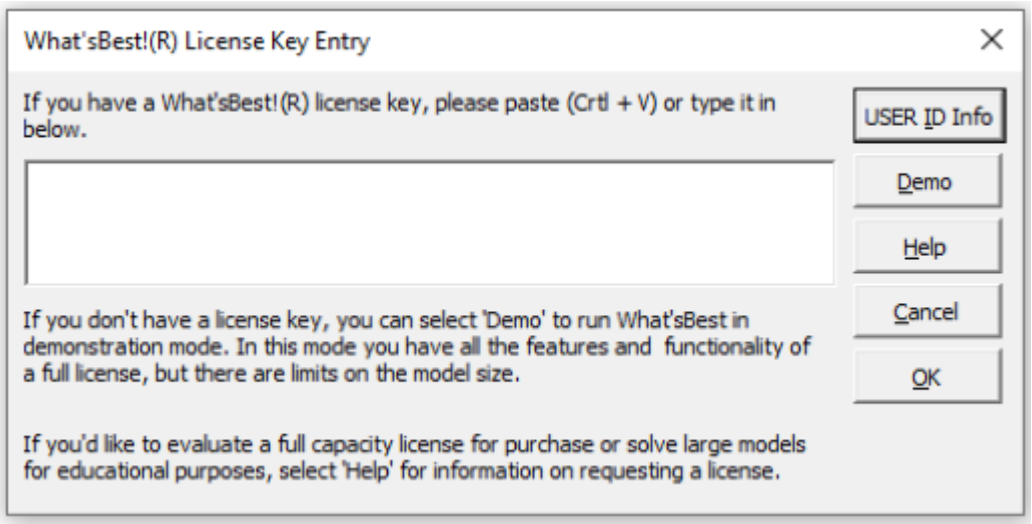
Should you wish to remove the What's*Best!* toolbar, you would use *View*|*Toolbars*|*Customize…* command, select the What's*Best!* toolbar, and press the *Delete* button. You would then have to reinstall What's*Best!* in order to restore the toolbar.

In Excel® versions 2007 and later, What's*Best!* just shows an additional tab in the Ribbon Bar layout, with an easier access to the dialog boxes and the shortcuts:



License Key Entry

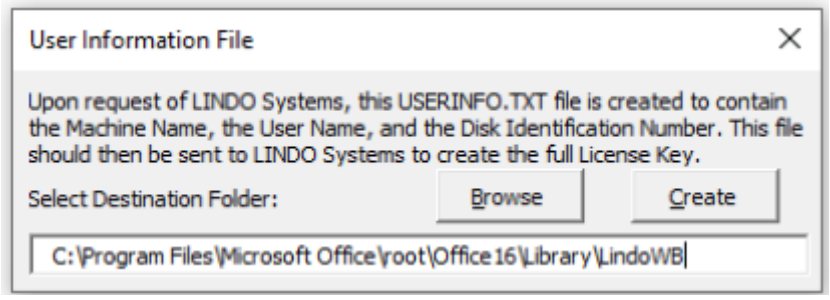
The dialog box posted by the License command appears as follows:



Use this command to input your What's*Best!* license key.

If you have already requested and received a license key, even a trial key, you may copy-and-paste it directly into the space provided (*Ctrl-C* to copy, *Ctrl-V* to paste), then select *OK*.

This command can also be used to upgrade What's*Best!* to handle larger problems (greater number of constraints, variables, and integers) or to add options to solve other types of problems. To upgrade, simply obtain a new license key from LINDO Systems and enter it here, then select *OK*.
The User Information File is created upon request of LINDO Systems, by clicking the *USER ID Info* button. It contains the Machine Name, the User Name, and the Disk Identification Number. This file USERINFO.TXT can then be sent to sales@lindo.com, so to create the full license key for academic purpose or a temporary Evaluation license for commercial purpose. By default, this file will be created in the What's*Best!* add-in folder, but a different folder can be selected, usually in the local disk drive.

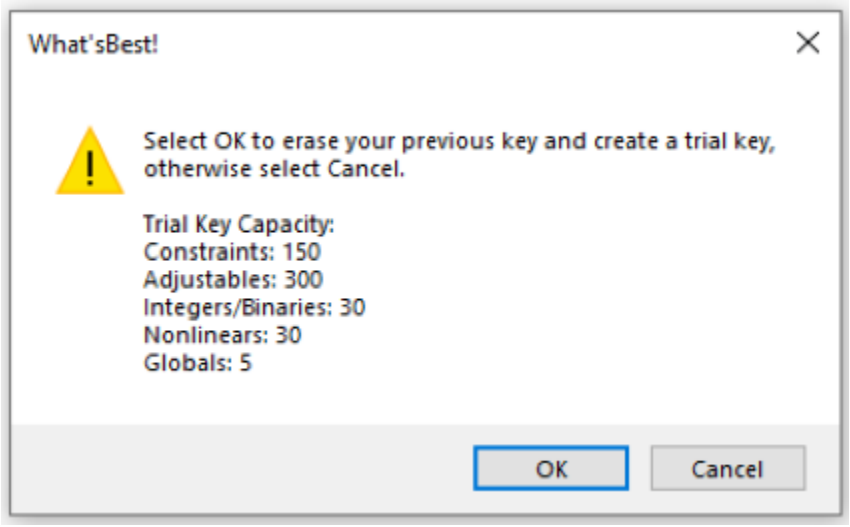


Also, you may need to verify where your license key is stored. The license file, *LNDWBxxx.LIC*, is located in the same directory as the What's*Best!* add-in files.

The About What's*Best!* will show your license capacity currently installed.

If you have lost your license key, please contact LINDO Systems.

Selecting the Demo button puts What'sBest in demonstration mode. This mode provides all of the features and functionality of a full license but limits the model size to 150 constraints, 300 total variables, 30 integer variables, 30 nonlinear variables, and 5 Global variables.

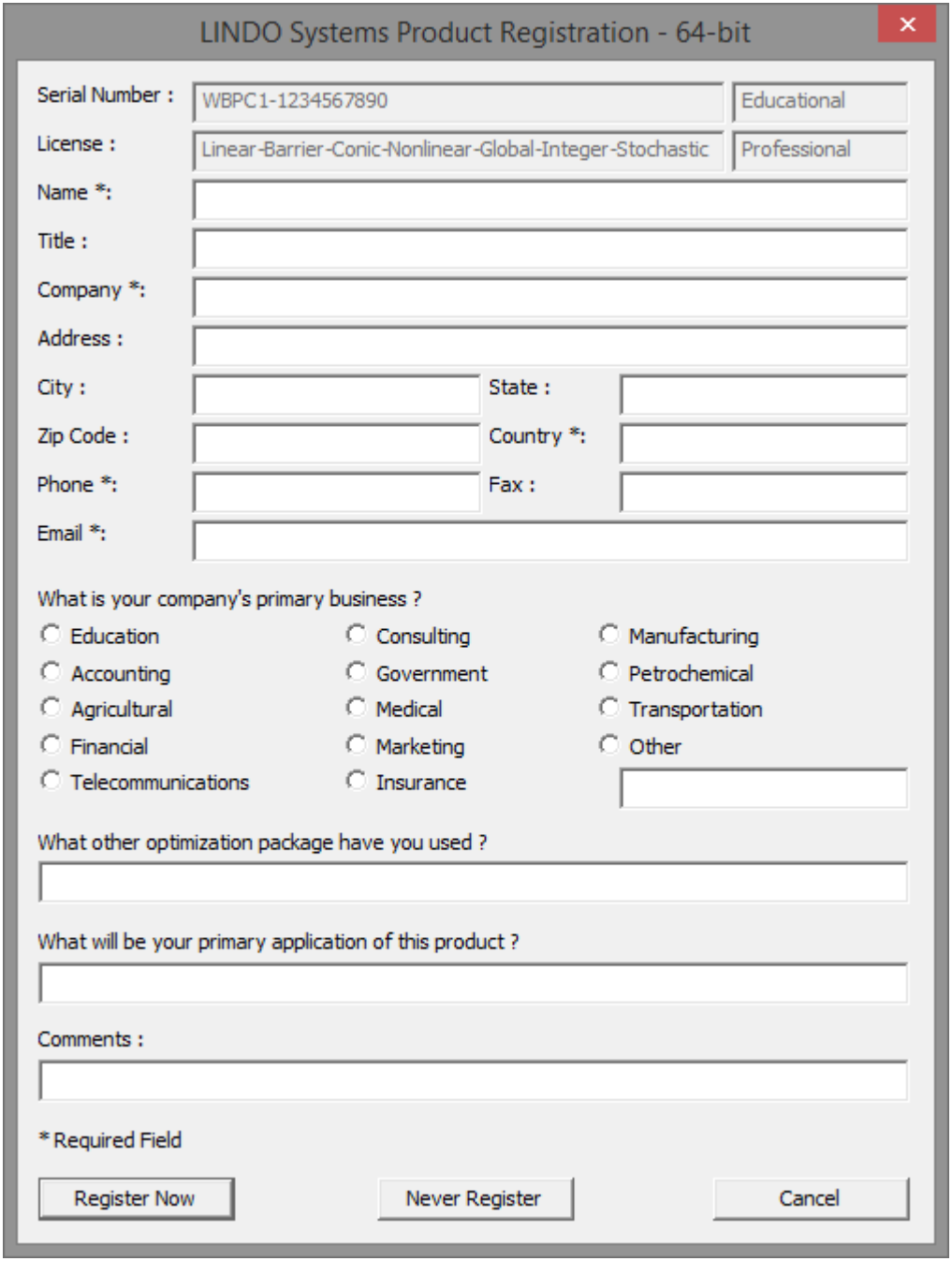


For the academic purpose, if you want to solve homework models, we recommend you select the Demo button. The demonstration mode generally provides sufficient capacity for most homework models. If you are a student or professor needing to solve larger models, you can request an unrestricted capacity Educational Research license. To request a license, select the *USER ID Info* button to create a file named USERINFO.TXT. Then, send this file as an attachment to sales@lindo.com from your educational email account. We will generate and send you back an Educational Research license key.

For the commercial evaluation, if you want to try What'sBest on larger models, you can request an unrestricted capacity temporary Evaluation license. To request a temporary Evaluation license, select the *USER ID Info* button to create a file named USERINFO.TXT. Then, send this file as an attachment to sales@lindo.com. We will generate and send you back a temporary Evaluation license key.

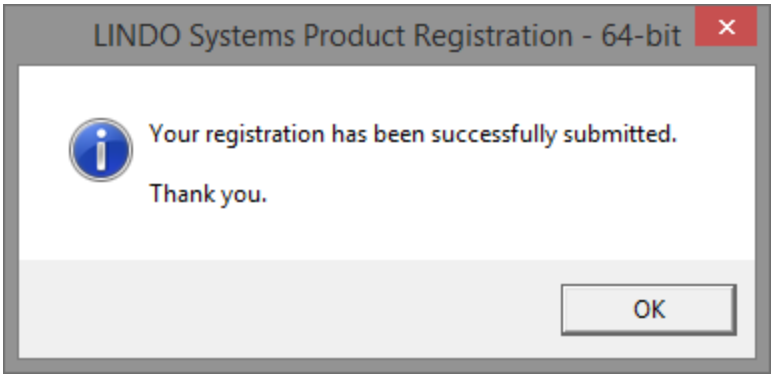
Register...

The dialog box posted by the Register command appears as follows:



Use the Register command to register your version of What's*Best!* online. You will need a connection to the Internet for this command to work. Enter the personal information you deem relevant and select the Register button. Your information will be sent directly to LINDO Systems via the Internet.

Once your registration is complete, the following dialog box will appear on your screen:



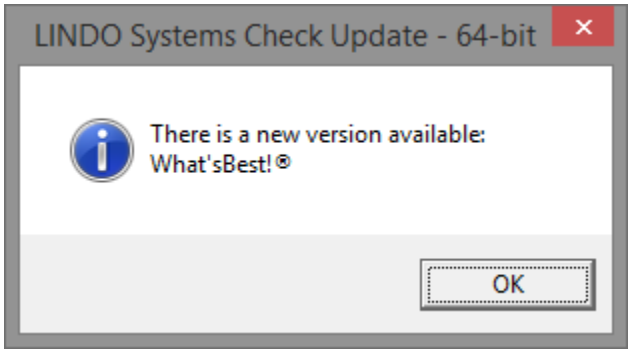
Select the OK button to be returned to the main Excel® environment.

LINDO Systems is constantly working to make our products faster and easier to use. Registering your software ensures that you will be kept up-to-date on the latest enhancements and other product news. You can also register through the mail or by fax using the registration card included with your software package.

CheckUpdate...

Turn the *CheckUpdate* command on to have What's*Best!* automatically check every time you start the software if there is a more recent version of What's*Best!* available for download on the LINDO Systems website. ***You will need an Internet connection to the Internet for this command to work***.

When you issue the *CheckUpdate* command, or start a version of What's*Best!* with *CheckUpdate* enabled, What's*Best!* will search the Internet to see if an updated version of the software is available for download. If you currently have the most recent version, then you will be returned to the main What's*Best!* environment. If a newer build of the software is available, you will be presented with the following dialog box:



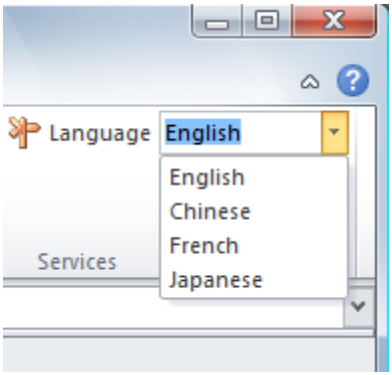
A checkmark next to the command in the WB menu indicates that the command is still enabled.You will need to download the new version of the software from the Website at the address indicated. You can disable this option by checking off the command item.

Upon completion of downloading, you will be prompted to begin installation of the new update. Close Excel®, and double click on the file to open the install wizard, which guides you through the setup process to update What's*Best!*. For your convenience, your license key will be automatically transferred to the new version during installation.

Keeping your software up to date helps ensure that you are using the most recent version of the software and that compatibility and operational problems are kept to a minimum.

Language

This feature allows the user to select the language of his choice without reinstalling the add-in.



After selecting the language, the user can revert to the previous setting, because the language words will stay displayed in the native language. Make sure that your system has installed the international language capabilities. This can be verified via the Control Panel of the Windows® operating system.

Note:

English (US)
Chinese (Pinyin)
French (France)
Japanese (Katakana)

4 What'sBest! Macros: The VBA Interface

This section discusses the VBA (Visual Basic® for Applications) interface of What's*Best!*. The VBA interface allows the user to write VBA code or macros that execute What's*Best!* commands. By using this interface, you can build optimization applications that utilize all the power and functionality of What's*Best!* in Excel® or any other application offering VBA control.

The section entitled *Usage Guidelines for Macros in VBA* introduces some features of the VBA interface such as the *Object Browser*, how to embed your What's*Best!* model in a Visual Basic® project, as well as how to hide the model. The procedures or functions (such as *Adjust*, *Best*, and *Constraint*) provided by What's*Best!* are discussed in *VBA Interface: Procedures*. The section entitled *Tutorial on the VBA Interface* is provided to help you get started writing VBA. The rest of the section describes the syntax, arguments, and error codes of particular procedures.

Usage Guidelines for Macros in VBA

Introduction to the VBA Interface of What'sBest!

The very first step in creating a macro for What's*Best!* is to create a reference to the What's*Best!* add-in, so you can use the add-in's attributes and procedures. This reference to What's*Best!* is made by checking the *WBA.XLA*, or *WBA.XLAM*, box under *Tools*|*References* from within the *Visual Basic® Editor* (The *Visual Basic® Editor* is called via *Tools*|*Macros* from the main Excel® menu bar). If you do not create this reference, then when your code attempts to call the What's*Best!* attributes or procedures Excel® returns the error message *Sub or Function not defined*.

Object Browser

Excel®'s *Object Browser* is a useful tool for developing macros that run What's*Best!*. The *Object Browser* provides for display of all publicly declared objects within a given unit of code, such as a module or the What's*Best!* add-in (*WBA.XLA* or *WBA.XLAM*). Once you've selected such an object, the *Object Browser* provides a view of all objects, procedures, and attributes that are available to any macros you write for that object. For example, by selecting the What's*Best!* add-in object (*WBA.XLA*) within the *Object Browser*, you can view the particular attributes and procedures of What's*Best!* that you can employ in a macro.

The *Object Browser* is a feature of the *Visual Basic® Editor* and is opened from the Excel® menu by *Tools|Macro|Visual Basic Editor*. Having opened the *Visual Basic® Editor*, select *View|Object Browser* to start the *Object Browser*. From the *Object Browser*, you designate What's*Best!* as your object by selecting *WBA.XLA* for the Library/Workbooks (instead of *<All Libraries>*). If you do not see *WBA.XLA* in the drop-down list, then you must check in the What's*Best!* add-in via the Excel® menu's or Ribbon's *Tools|Add-ins* or re-install the What's*Best!* add-in.

After you've selected *WBA.XLA* in the *Object Browser*, you should see *WBUsers* among the *Classes* listed. This class is your primary source of What's*Best!* procedures. As you select this class, the right window should display all of the *Members* (procedures) available to you from that class.

Macro Recorder

Excel®'s *Macro Recorder* can be a very valuable tool for creating macros of tasks in Excel®. Unfortunately, it generally does not work well in conjunction with Excel® add-ins such as What's*Best!*. The *Macro Recorder* of Excel® 97 and later versions is unable to record the What's*Best!* add-in commands, so we recommend you do not try to use it to generate VBA code. You can use Excel®'s *Object Browser* described above to more quickly and accurately write your VBA code.

Calling Procedures

Visual Basic® for applications is quite liberal in the syntax it allows when calling procedures. For the sake of simplicity, the calling procedures used for illustration below will stick to one or two conventions depending upon whether or not named arguments are used. For example, to call the procedure to set cells D8:E10 as adjustable without named arguments would be:

```
wbAdjust "D8:E10"
```

The same procedure call with named arguments would be:

```
wbAdjust AdjRange:= "D8:E10"
```

Note: VBA is case-insensitive to the procedure name when it is called, so the procedure could alternately be called with its name typed entirely in upper case (i.e., as *WBADJUST* instead of *wbAdjust*).

Named arguments such as shown above are necessary when you are skipping over some arguments. With some programming languages, it is possible to skip an argument by leaving it blank. This is not possible with VBA. Named arguments must be used instead:

INVALID: wbSetGeneralOptions (, , , , , , , , , , , False, True, True, , , , , ,)

VALID: wbSetGeneralOptions goStatusReport:=True, goSolutionReport:=False, goWrnBlank:=True

When calling procedures with multiple arguments, using named arguments allows you to list the arguments in any order you choose. Using named arguments can also make the code easier to read.

Error messages you might encounter learning VBA.

When initially learning to run What's*Best!* using VBA code, the user may encounter the following error messages, which are easily resolved:

Sub or Function not defined - This error message is often the result of not having *WBA.XLA* checked as an available reference. Checking it as a reference makes the What's*Best!* procedures available to your Visual Basic® code. Instructions for setting this reference are provided at the beginning of this section.

Invalid outside procedure - This message is often the result of using parenthesis without the *CALL* keyword. Either put the keyword *CALL* in front of procedure calls where you use parenthesis or remove the parenthesis.

Beyond VBA / VB

Model developers may prefer the ability to provide the user with a more elaborate, polished interface than can be provided using Excel® workbooks running Visual Basic® macros. Developers can create such applications using the full featured development environments of Visual Basic® or Microsoft® Access® that run What's*Best!* with Excel® running in the background. The Access® or VB application can easily prepare the data for the optimization model, insert it into an Excel® workbook, and run What's*Best!*, so that Excel® is completely hidden from the user.

The development of the application can progress in natural stages. The optimization model can first be prototyped and tested in Excel®. Once the structure of the model is decided upon, Visual Basic® code can be written to automate the process of creating the model within Excel®. Finally, the code can be incorporated into an Access® or Visual Basic® project in which the What's*Best!* model is built based upon a data set provided by the application.

Embedding What'sBest! in a Visual Basic® / Visual Studio® Project

We again return to the XYZ Tutorial problem to look at how it could be incorporated into a Visual Basic® (VB) program. Assuming that Visual Basic® is installed on the computer, extract the WB_VB project from the WB_VB.zip file of the *WB* directory. Start the Visual Basic® or Visual Studio® and

choose *File|Open Project* to select the "WB_VB.sln" project. There are WB_Module1.vb, WB_Form1.vb, WB_Form1.vb[Design], and the WB_VB project properties.

This sample project will demonstrate a basic dialog box with a button. A function WB_VBBuildXYZ() will actually open Microsoft Excel®, initialize the link to the What's*Best!* add-in, and open the file C:\WB\XYZ.XLSX. It will then complete the model by setting the Adjustable, Best, and Constraint cells, then solves it. The solution will be displayed back onto the worksheet, with a message box for the returned status. The user can then close the messages.

The code clarify the basics for opening, building, and solving a model. The user can look at this project as a template for further developments, eventually adjusting on the platform, .NET, either 32 or 64-bit.

Concealing and Protecting a Model from the User

What's*Best!* allows model developers to hide aspects of a model that might be proprietary and protect sections of a model from inadvertent or unauthorized modification, while allowing the user to access and modify input data. While this does not require the use of Visual Basic®, these techniques can easily be used in conjunction with VBA.

Information in a What's*Best!* model that is to be hidden from the user must reside on one or more worksheets that are separate from the worksheet(s) to which the user has access. Similarly, information that is to be protected must reside on one or more worksheets that are separate from the rest of the model.

To demonstrate how to conceal and protect a model from the user, we will modify the Product Mix model. First, we rename the *Prodmix* sheet as *Model*. Second, we will create two new worksheets that have all the information that the user needs: one called *Inputs* and the other called *Outputs*. When we are done, the *Model* sheet will be entirely hidden and protected from the user. It will contain data and formulas that are considered proprietary.

We now populate the *Inputs* and *Outputs* sheets as follows. First, we copy all of the data from A5:G6 on the *Model* sheet to the *Inputs* sheet. We now set the *Profit/ Unit* for each of the products to unlocked cells on this *Inputs* sheet. We can then protect the *Inputs* sheet with the *Tools|Protect Sheet* command and enter a password.

Turning to the *Outputs* sheet, we copy and change the "Quantity to Produce" for each of the products to formulas that reference the *Model* sheet. We can then protect the *Outputs* sheet from the user, so nothing can be changed. First, select the "Quantity to Produce" cells and choose *Format|Cells*. On the Protection tab, select *Hidden* (leave the cells *Locked*), so that the user can not see the formulas in these cells. Then, choose *Tools|Protection|Protect Sheet* and enter a password.

On the *Model* sheet, we change the "Profit/ Unit" for each of the products to formulas that reference the *Inputs* sheet.

The last thing to do is to hide and protect the *Model* sheet. To do this select the entire worksheet and choose *Format|Cells*. On the *Protection* tab select *Hidden*. Then, with the entire spreadsheet still selected, choose *Format|Row|Hide* and *Format|Column|Hide*. Next, protect the sheet by giving the

command *Tools|Protection|Protect Sheet* and enter a password. The final step is to hide the *Model* sheet with the command *Format|Sheet|Hide*.

Now, the user will only see the *Inputs* and *Outputs* sheets. A clever user would be able to determine that there is a *Model* sheet but he or she would not be able to view or modify it. In the WB subdirectory, there is a file called PRODMIX2.XLS, which has the changes described above.

Note: It is not possible to use Excel®'s *Tools|Protection|Protect Structure* command with What's*Best!*. This command does not allow the *WB! Status* or *WB! Solution* worksheets to be added to the workbook.

Tutorial on the VBA Interface

This section provides a tutorial on the basics of running What's*Best!* commands using VBA (Visual Basic® for Applications). It includes detailed discussion and step-by-step examples of a variety of different common applications combining Visual Basic® and What's*Best!*.

Note: The first step in running any What's*Best!* from VBA is to create a reference to What's*Best!*. This reference is made by checking the *WBA.XLA* box under *Tools|References* from within the *Visual Basic® Editor* (The *Visual Basic® Editor* is called via *Tools|Macros* from the main Excel® menu bar). If you do not create this reference, then any attempt to use the What's*Best!* attributes or procedures will produce the error message *Sub or Function not defined*.

Building and Solving a Basic Model

Let's revisit the XYZ Production problem discussed in the *Tutorial* of *Getting Starting*. We will look at how the ABC's of a model can be specified using Visual Basic®. Unless specified otherwise, a sample file named *XYZVBA.XLSM* was copied to the *WB* subdirectory during installation. Along with the main XYZ worksheet, there are two VBA modules called *ABC's* and *Dual*.

The ABC's module has three procedures in it: *BuildXYZ*, which uses Visual Basic® to build and solve the model; *MaxDeluxe*, which changes the objective to maximizing the use of Deluxe computer towers and solves the new model; and *MaxDeluxe2*, which uses the same objective as *MaxDeluxe*, but imposes a restriction that profit must be greater-than-or-equal-to $32,000. You will notice that the code to build the model is very straightforward. For information on the complete syntax for each of the functions, refer to the section on *Procedures* .

The *BuildXYZ* procedure is listed below:

```
Sub BuildXYZ()

    Dim lngSolutionStatus As Long

    'Go to line label Handler for code that handles error
    On Error GoTo Handler

    'Make Quantities to Produce (C5:D5) Adjustable
```

```
wbAdjust "C5:D5"

'Make Total Profit (G6) the best cell to Maximize
wbBest "G6", "Maximize"

'Constrain Total Usage (E15:E17) to be less than Number in
'Stock (G15:G17)
wbConstraint "E15:E17", "<=", "G15:G17", "F15:F17"

'Solve the model
wbSolve lngSolutionStatus

'Display the results
Select Case lngSolutionStatus
    Case 1
        MsgBox "The model is: Globally Optimal"
    Case 2
        MsgBox "The model is: Globally Optimal, " &_
        Range("WBMAX")
    Case 3
        MsgBox "The model is: Infeasible"
    Case 4
        MsgBox "The model is: Unbounded"
    Case 5
        MsgBox "The model is: Feasible"
    Case 6
        MsgBox "The model is: Infeasible or Unbounded"
    Case 7
        MsgBox "The model is: Near Optimal"
    Case 8
        MsgBox "The model is: Locally Optimal"
    Case 9
        MsgBox "The model is: Locally Infeasible"
    Case 10
        MsgBox "The model is: Cutoff"
    Case 11
        MsgBox "The model is: Numerical Error"
    Case 12
        MsgBox "The model is: Unknown"
    Case 13
        MsgBox "The model is: Unloaded"
    Case 14
        MsgBox "The model is: Loaded"
    Case Else
        MsgBox "Solution status unknown"
End Select

Exit Sub
```

```
Handler:
    MsgBox "The error description is " & wbError(Err)

End Sub
```

In the above example, any line starting with a single quotation mark is a comment.

Note: If Excel® returns the error message *Sub or Function not defined*, then What's*Best!* needs to be set as an available reference. To do this, choose *Tools*|*References* from the Visual Basic® Editor and check *WBA.XLA* in the list. Now, you should be able to run the procedure without error.

The procedure *BuildXYZ* shown above is very simple. It sets C5:D5 to be adjustable cells, makes the *Total Profit* in G6 the cell to maximize, and constrains *Total Usage* (E15:E17) to be less than the *number in stock* (G15:G17). When these steps have been performed, the macro code solves the model and displays the results.

To make this example more readable, the ranges passed to the What's*Best!* procedures are text references. Alternately, the ranges could also have been passed as objects to allow the developer to work with row and column numbers instead of the text. The above code presented with object references would look like this:

```
'Make Quantities to Produce (C5:D5) Adjustable
 wbAdjust Range(Cells(5,3), Cells(5,4))

'Make Total Profit (G6) the best cell to Maximize
 wbBest Cells(6,7), "Maximize"

'Constrain Total Usage (E15:E17) to be less than Number in
'Stock (G15:G17)
 wbConstraint Range(Cells(15,5), Cells(17,5)), "<=", _
   Range(Cells(15,7), Cells(17,7)),  Range(Cells(15,6), _
    Cells(17,6))
```

The two other procedures in the *ABC's* module, *MaxDeluxe* and *MaxDeluxe2*, simply change the XYZ model slightly and solve it again. The *Dual* module contains a couple of subroutines that demonstrate putting dual value functions in the model.

In the examples above, none of the arguments used the named argument syntax. To do so, simply list the argument name followed by a colon and equal sign and then the argument. To set the range C5:D5 as adjustable with the named argument syntax, enter:

```
wbAdjust AdjRange:="C5:D5", AdjChoice:="Adjustable"
```

Using the named argument syntax can make the code easier to read and, when calling procedures with multiple arguments, allows you to list the arguments in any order you choose.

Solving Multiple Problems with a Looping Macro

When automating a process using macros, you may want to run the same macro many times on different data while storing the optimal solution for each data set. For example, you may have a database filled with information on the return on a group of investments at a range of interest rates. What's*Best!* can be used to determine the optimal portfolio in each situation by using a looping macro. Using the *PMIXMAC.XLS* sample file, which was copied to the *WB* subdirectory during installation, let's look at how a looping macro might be created. Use the Visual Basic® Editor to go to the *Looping Macro* module in your workbook and your screen should look like the following:

```
Sub Macroloop()

    'This macro will increment the Profit/Unit of Product 2
    'from 45-60, solve the model, copy the Total Profit and Quantity Produced
    'of Products 1-6 to a table, and repeat.

    Dim Count As Integer
    Dim TableRow As Integer

    'Initialize the profit of product 2 to 45
    Sheets("PRODMIX").Select
    Range("C6") = 44

    'Evaluate the effect of changing the profit on product 2 by
    'increasing it by one unit and resolving. Solve the model,
    'save the results in a table, and increment the profit on
    'product 2.

    For Count = 1 To 20

        'Copy a new value to the profit of product 2
        Range("C6") = Range("C6") + 1
        'Solve the current model
        Application.Run macro:="WBUsers.wbSolve"
        'Set TableRow to Count plus 2, since the table will
        'start in row 3
        TableRow = Count + 2
        'Copy the profit per unit for product 2 to the table
        Sheets("PRODMIX").Select
        Cells(TableRow, 12) = Range("C6")
        'Copy the solution so it can be pasted into the table
        Range("B8:G8").Copy
        'Paste the solution to the table. We paste in the values
        'only to prevent the cells.
        'in the table from being treated as Adjustable
        Range(Cells(TableRow, 13), Cells(TableRow, 18)).PasteSpecial Paste:=xlValues
        'Save the Total Profit in the table
        Cells(TableRow, 19) = Range("A3")

    Next Count

End Sub
```

Lines beginning with a single quotation mark are comments.

The macro shown here performs range analysis by incrementing the profit/unit of product 2 from 45 to 64. The *For...Next* form of looping in Visual Basic® is used here, but other looping forms such as *Do...Loop* and *For Each...Next* forms are available. Please consult the Visual Basic® or Excel® Help for more information.

After running the macro, the adjustable cells and the best cell for each increment are copied to the table in the range L2:S22. The row is incremented each time through the variable *Table* that is always two greater than the *Count* (the iteration number). In this way, a convenient report is created providing sensitivity analysis of the profit/unit of Product 2.

As shown here, there are changes in the optimal solution when the profit/unit for product 2 is $50 or greater. In addition to simply having the macro increment the profit/unit for product 2 through a range, the profit/unit values could have been read from a database or a text file.

This model is a looping version of the *Product Mix* model.

VBA Interface: Procedures

What's*Best!* provides a set of Visual Basic® Procedures that allow models to be built and automated using Visual Basic® for Applications (VBA) commands. These procedures are the same ones that are called when What's*Best!* commands are selected from the *WB!* menu or the What's*Best!* toolbar. For example, the *Solve* procedure invoked in a macro is also invoked when you press the  button on the What's*Best!* toolbar.

If you wish to automate the creation of a What's*Best!* model, or if you wish to incorporate What's*Best!* as a part of a larger application, then you will find these procedures to be invaluable. They make it relatively easy to incorporate What's*Best!* into a Microsoft® Excel® or Access® application, or any other application offering Visual Basic® for Applications.

All of the following procedures with arguments requiring cells can accept either range objects or the address of the cell(s).

Note: VBA is case-insensitive to the procedure name when it is called, so any procedure could alternately be called with its name typed entirely in lower case (i.e., as "wbaddadjustablestyle" instead of "wbAddAdjustableStyle").

All of the What's*Best!* procedures available are listed below with details on their syntax, arguments, error codes, and any return values. Some also include examples of using the procedure or remarks. The procedures with arguments that require cells can accept either range objects or the address of the cell(s). Argument with square brackets ([and]) around them are optional - non-bracketed arguments are required.

wbAddAdjustableStyle

This routine is used to add the adjustable style to a workbook that does not have one yet.

Syntax: wbAddAdjustableStyle()

The wbAddAdjustableStyle procedure has no arguments.

Error Codes	Description
Adj_CreateStyleError	Error Creating an Adjustable style

Remarks: Because this procedure is called by the wbAdjust procedure, in most cases, it should not be necessary for the developer to call it directly.

wbAddBestStyle

This routine is used to add the best style to a workbook that does not have one yet.

Syntax: wbAddBestStyle()

The wbAddBestStyle procedure has no arguments.

Error Codes	Description
Best_CreateStyleError	Error Creating a Best style

Remarks: Only one cell, the objective cell, should use this style.

wbAddWBMenu

This routine is used to add the *WB!* menu item to the Excel® menu.

Syntax: wbAddWBMenu()

The wbAddWBMenu procedure has no arguments.

Error Code	Description
AddWBMenuError	Error in adding the *WB!* menu

Remarks: This procedure and the wbDeleteWBMenu procedure allows a developer to remove and add the *WB!* menu as desired.

wbAdjust

For additional discussion of the functionality provided, see the section entitled *Adjustable*, which refers to the dialog box interface that calls this procedure.

This routine is also called with appropriate arguments by two of the toolbar buttons.

This routine is used to either set cells as adjustable or to remove the adjustable attribute. All arguments are optional.

Syntax: wbAdjust([AdjRange], [AdjChoice], [NoErrDialog])

Argument	*Required*	*Description*
AdjRange	No	This is the range or address of the cells to be rendered adjustable. If omitted, the default value for *AdjRange* is the current selection.
AdjChoice	No	This is a string to indicate *Adjustable* or *Reset*. *AdjChoice* accepts the text "Adjust" to set the *AdjRange* as Adjustable or "Reset" to set the *AdjRange* to no longer be adjustable. If omitted, the default value for *AdjChoice* is "Adjust". For backwards compatibility with earlier versions of What's*Best!*, *AdjChoice* also accepts the numbers 1 for *Adjustable* and 2 for *Reset*.
NoErrDialog	No	Any argument passed here causes all What's*Best!* error dialog boxes from the *wbAdjust* routine to be suppressed. Instead, the error number is delivered in this return argument. Pass an integer to use any possible returned What's*Best!* error number. Useful in an embedded application of What's*Best!*.

Error Codes	*Description*
Adj_BadAdjRangeArg	Bad *AdjRange* argument
Adj_BadAdjChoiceArg	Bad *AdjChoice* argument
Adj_ProtectedSheetError	Unable to set cells as adjustable on a protected sheet
Adj_CreateStyleError	Error creating an *Adjustable* style
Adj_CheckStyleUnlockedError	Error checking that the *Adjustable* style is unlocked
Adj_CheckStyleProtectionError	Error checking that the *Adjustable* style includes protection

Remarks: An adjustable cell must contain a number or the cell is ignored by the solver. The *Adjust* procedure formats the cell(s) with a style called *Adjustable*, which carries a blue font color as a visual reminder. All adjustable cells will return non-negative values unless they are in a *WBFREE* range.

Example: To set the cells D8:E10 as adjustable, enter the macro

```
wbAdjust "D8:E10"
```

or

```
wbAdjust Range(Cells(8,4),cells(10,5))
```

To reset the cells D8:D10 to no longer be adjustable, enter:

```
wbAdjust "D8:D10", "Reset"
```

or

```
wbAdjust Range(Cells(8,4),cells(10,4)), "Reset"
```

wbBest

For additional discussion of the functionality provided, see the section entitled *Best*, which refers to the dialog box that calls this procedure.

This routine is also called with appropriate arguments by the either of the toolbar buttons displayed above.

This routine is used to create a best cell, replacing any previously defined best cell. All arguments are optional.

Syntax: wbBest([BestCell], [BestChoice], [NoErrDialog])

Argument	*Required*	*Description*
BestCell	No	This is the cell or address of the cell to be the best cell. If omitted, the default value for *BestCell* is the current cell.
BestChoice	No	This is a string to indicate whether the best cell should be minimized or maximized. *BestChoice* accepts "MIN" or "MINIMIZE" (upper or lower case) for minimize, "MAX" or "MAXIMIZE" (upper or lower case) for maximize, and "NONE" (upper or lower case) to remove any previous best cell. If omitted, the default value for *BestChoice* is minimize. For backwards compatibility with earlier versions of What's*Best!*, *BestChoice* also accepts the numbers 1 for *Minimize*, 2 for *Maximize*, and 3 for *None*.
NoErrDialog	No	Any argument passed here causes all What's*Best!* error dialog boxes from the *wbBest* routine to be suppressed. Instead, the error number is delivered in this return argument. Pass an integer to get any possible returned What's*Best!* error number. Useful in an embedded application of What's*Best!*.

Error Codes	Description
Best_BadBestCellArg	Bad *BestCell* argument
Best_BadBestChoiceArg	Bad *BestChoice* argument
Best_ProtectedSheetError	Unable to set the best cell in a protected sheet
Best_CreateStyleError	Error creating a *Best* style

Remarks: Only one cell can be specified as a best cell.

Example: To set I17 as your objective to minimize, enter:

```
wbBest "I17", "Minimize"
```

To set F17 as the objective to maximize, enter:

```
wbBest "F17", "Maximize"
```

To reset cell F17 to "None", enter:

```
wbBest "F17", "None"
```

WBBIN

For additional discussion of the functionality provided, see the section entitled *Integer*, which refers to the *Binary* option button that calls this procedure.

This routine is used to build binary integer ranges named *WBBIN*. The adjustable cells contained in these ranges will be set to either 0 or 1 by the What's*Best!* solver.

Syntax: WBBIN(IntName, [Refers_to])

Argument	Required	Description
IntName	Yes	*IntName* is a string that refers to the range being made integer.
Refers_to	No	*Refers_to* is the range or address of the cells to be specified as integer. If omitted, the default is the current selection.

Error Codes	Description
Int_BadIntNameArg	Bad *IntName* argument
Int_BadRefers_toArg	Bad *Refers_to* argument
Int_ProtectedSheetError	Unable to set integer cells on a protected sheet
Int_CreateError	Error setting integer cells

Remarks: Integer variables can dramatically increase total solution times.

Example:
To constrain cell F7 to be a binary integer using the range name OnOff, enter:

```
WBBIN "OnOff", "F7"
```

To Remove:
If you wish to delete a *WBBIN* range name using VBA instead of the *Delete* button on the *Integer-Binary* dialog box, you can use:
ActiveWorkbook.Names("WBBINOnOff").Delete.
This statement removes the *WBBIN* range name, thereby returning the adjustable cells contained in the former range to normal status.

wbConstraint

For additional discussion of the functionality provided, see the section entitled *Constraints*, which refers to the dialog box that calls this procedure.

This routine is also called with appropriate arguments by the three toolbar buttons displayed above.

This routine inserts a constraint formula within the selected cell.

Syntax : wbConstraint(LHS, Direction, RHS, ConLoc, [NoErrDialog])

Argument	***Required***	***Description***
LHS	Yes	*LHS* is the left-hand side of the constraint equation.
Direction	Yes	This is a string to indicate what type of constraint: "<=" (less-than-or-equal-to), ">=" (greater-than-or-equal-to) , "=" (equal-to), "<c=" (convex), ">c=" (concave), or "=c" (convex). For backward compatibility with earlier versions of What's*Best!*, *Direction* also accepts the numbers 1 for "<=", 2 for "=", and 3 for ">=", 4 for "<c=", 5 for "=c", 6 for ">c=", 7 for "None".
RHS	Yes	*RHS* is the right-hand side of the constraint equation. It can be a range or a value (see the *Remarks* below).
ConLoc	Yes	*ConLoc* is the range to store the constraints.
NoErrDialog	No	Any argument passed causes all What's*Best!* error dialog boxes from the *wbConstraint* routine to be suppressed. Instead, the error number is delivered in this return argument. Pass an integer to get any possible returned What's*Best!* error number. Useful in an embedded application of What's*Best!*.

Error Codes	***Description***
Con_BadLHSArg	Bad *LHS* argument
Con_BadDirectionArg	Bad *Direction* argument

Con_BadRHSArg	Bad *RHS* argument
Con_BadConLocArg	Bad *ConLoc* argument
Con_RangeSizeError	LHS must be same size as ConLoc
Con_ProtectedSheetError	Unable to enter constraints on a protected sheet

Remarks: LHS and ConLoc ranges must be the same size.

RHS can be a range or a value.

Example: To enter the constraint H8 >= 10 into cell I8, enter:

```
wbConstraint "H8", ">=", "10", "I8"
```

or

```
wbConstraint Cells(8,8), ">=", "10", Cells(8,9)
```

To constrain H8:H10 to be less-than-or-equal-to J8:J10 and store the constraints in I8:I10, enter:

```
wbConstraint "H8:H10", "<=", "J8:J10", "I8:I10"
```

or

```
wbConstraint Range(Cells(8,8),Cells(10,8)), "<=", _
  Range(Cells(8,10),Cells(10,10)), _
  Range(Cells(8,9),Cells(10,9))
```

wbConstraintPOSD

For additional discussion of the functionality provided, see the section entitled *Constraints...*, which refers to the dialog box that calls this procedure.

This routine is also called with appropriate arguments by the three toolbar buttons displayed above.

This routine inserts a constraint formula within the selected cell.

Syntax: wbConstraintPOSD(RangeMatrix, Place_in_cell, [NoErrDialog])

Argument	***Required***	***Description***
RangeMatrix	Yes	RangeMatrix is a cell reference for indicating the range of the matrix.
Place_in_cell	Yes	Place_in_cell is a cell reference to specify the cell in which to write the WBPOSD function.
NoErrDialog	No	Any argument passed causes all What's*Best!* error dialog boxes from the wbConstraintPOSD routine to be suppressed. Instead, the error number is delivered in this return argument. Pass an integer to get any possible returned What's*Best!* error number. Useful in an embedded application of What's*Best!*.

Error Codes Value	Description
Con_BadRangeMatrixArg	Bad RangeMatrix argument
Con_BadPlaceInCellArg	Bad PlaceInCell argument

Remarks:
RangeMatrix ranges must be a symmetric range, more than one cell.

Example:

To enter the POSD constraint for the matrix A1:C3, and the function =WBPOSD() in cell D1, enter:

```
wbConstraintPOSD "A1:C3", "D1"
```

or

```
wbConstraintPOSD Range(Cells(1,1),Cells(3,3)), Cells(1,4)
```

wbConvertModelFormat

This routine allows you to convert a non-What's*Best!* format spreadsheet optimization model to What's*Best!* format. It will try to extract any existing cells in the original model to set the Adjustable, Best cells, and it will create a new tab 'WB!_Constraint_Sheet1' at the end of the workbook for inserting the constraint functions.

Syntax: *wbConvertModelFormat()*
The *wbConvertModelFormat* procedure has no arguments.

Remarks: What's*Best!* can have one model per workbook, but other spreadsheet modeling interfaces may have one model per worksheet in the same workbook. We recommend to select the active worksheet containing the model to convert before invoking this function. You will see the Best and Adjustable cells with a new format, and a additional sheet for the =WB() constraints in the order they were found in the model. Also, you may need to look at the options and other settings for tolerances that were not converted.

wbDeleteReports

This routine is called by the *Delete Reports* button on the *General Options* dialog box posted by the *Options…*|*General* command.

This routine is used to delete any What's*Best!* report worksheets (*WB! Status* and *WB! Solution*) in the current workbook.

Syntax: wbDeleteReports()

The *wbDeleteReports* procedure has no arguments.

Error Codes Value	Description
Rep_DeleteReportsError	Error deleting one of the reports

wbDeleteWBMenu

This routine is used to delete the *WB!* menu item from Excel®'s menu.

Syntax: *wbDeleteWBMenu()*

The *wbDeleteWBMenu* procedure has no arguments.

Remarks: This procedure, together with the *wbAddWBMenu* procedure, allows a developer to remove and add the *WB!* menu as desired.

wbDualValue

For additional discussion of the functionality provided, see the section entitled *Advanced…|Dual*, which refers to the dialog box interface that calls this procedure.

This routine is used to build the dual formulas. All arguments are optional.

Syntax: wbDualValue([DualCells], [DualLoc], [DualChoice])

Argument	*Required*	*Description*
DualCells	No	*DualCells* is the range or cell that dual value information is desired for. If omitted, the default is the offset of one cell to the left of the current selection.
DualLoc	No	*DualLoc* is the range or cell where the information is to be displayed. If omitted, the default is the current selection.
DualChoice	No	*DualChoice* is a string indicating the type of information desired. *Dual* or *Dual Value* for dual value, *Upper* or *Upper Range* for upper range, and *Lower* or *Lower Range* for lower range. If omitted, the default is *Dual*. For backwards compatibility with earlier versions of What's*Best!*, *DualChoice* also accepts the numbers 1 for *Dual*, 2 for *Lower*, and 3 for *Upper*.

Error Codes	*Description*
Dual_BadDualsCellArg	Bad *DualCells* argument
Dual_BadDualLocArg	Bad *DualLoc* argument
Dual_BadDualChoice	Bad *DualChoice* argument
Dual_ProtectedSheetError	Unable to enter dual formulas on a protected sheet

Remarks: On integer models, dual value information is only of limited use.

Example: To create the dual value for cell E5 in cell F5, enter:

```
wbDualValue "E5", "F5", "Dual Value"
```

To create the upper and lower ranges for cell E5 displayed in cells F6 and F7, respectively, enter:

```
wbDualValue "E5", "F6", "Upper Range"
wbDualValue "E5", "F7", "Lower Range"
```

wbError, and Error Codes

It is very important to handle any possible What's*Best!* errors using some form of the On Error statement as shown in the code example below when error-handling code is in place. If a What's*Best!* VBA procedure encounters an error, then your code handles the particular error. For example, you can obtain a description of the error and provide it to the user, or perhaps provide the user with further information, as the code example below demonstrates. However, if your code that calls the What's*Best!* procedure does not have error-handling, then Excel® handles any What's*Best!* error with a run-time error message that can sometimes disguise the error. For details, see the section entitled *Run-time Errors*.

Code Sample One (below) sets the selected cell(s) as adjustable and solves the model, while providing the user with information in case there is an error setting adjustable cells on a protected sheet. The code also displays a description of any other error.

NOTE ON CODE SAMPLE ONE: What's*Best!* has defined error numbers beginning at 30001. These error numbers should not be used in your code, since they are subject to change. Instead, use the error codes stored in the wbErr structure, as shown in the example below. What's*Best!* provides a function to return the error text for these wbErr codes. This function is similar to Excel®'s Error() function. The wbError() function takes an error code or error number as an argument and returns the error text.

```
' CODE SAMPLE ONE
Sub mySub
On Error GoTo ErrorHandler                 'Trap any What'sBest!
error
wbAdjust                                        'Make current cell
adjustable
wbSolve                                       'Solve the model
Exit Sub
ErrorHandler:                                  'Process any
What'sBest! error
If (Err = wbErr.Adj_ProtectedSheetError) Then
MsgBox "To the user: Please unprotect all the sheets in your models
using the  Tools|Protect command"
Else
If Err >= 30000 Then
   MsgBox "The error description is " & _
     wbError(Err)
Else
   MsgBox "The error description is " & _
     Err.Description
```

```
End If

End If

Exit Sub

End Sub
```

If you are running What's*Best!* within a VB project, you should use an alternative method for error handling. This method will prevent possible What's*Best!* errors from being transformed into unusable Visual Basic® runtime errors. The following *Code Sample Two* can be used as a workaround for the Visual Basic® project file WB_VB.sln, available in the *WB* directory.

```
' CODE SAMPLE TWO

Sub VBABuildXYZ

{Omitted code that created an Excel® object named gobjExcel and
opens the problem to be solved}

Dim intError As Integer

gobjExcel.Run "wbAdjust", "C5:D5", , intError

If (intError > 0) Then

GoTo Quit

End If

{Omitted code similar to above for wbBest, wbConstraint, and
wbSolve, as well as code for displaying the results of solve}

Exit Sub

Quit:

Dim strError as String

gobjExcel.Run "wbError", intError, strError

MsgBox "Error # = " & intError & ", Description: " & _

        strError

Exit Sub

End Sub
```

wbError

The error numbers and the codes stored in the wbErr structure are as follows:

Number	Code
30001	Adj_BadAdjRangeArg
30002	Adj_BadAdjChoiceArg

30003	Adj_ProtectedSheetError
30004	Adj_CreateStyleError
30005	Adj_CheckStyleUnlockedError
30006	Adj_CheckStyleProtectionError
30007	Free_BadRefers_toArg
30008	Free_ProtectedSheetError
30009	Free_CreateError
30010	Best_BadBestCellArg
30011	Best_BadBestChoiceArg
30012	Best_ProtectedSheetError
30013	Best_CreateStyleError
30020	Con_BadLHSArg
30021	Con_BadDirectionArg
30022	Con_BadRHSArg
30023	Con_BadConLocArg
30024	Con_RangeSizeError
30025	Con_ProtectedSheetError
30026	Con_BadRangeMatrixArg
30027	Con_BadPlaceInCellArg
30030	Solve_UnmappedDrive
30031	Solve_NoDLL
30032	Solve_BadSolutionStatusArg
30033	Solve_BadExcelOpenArg
30034	Solve_MultipleMaxMinError
30035	Solve_ProtectedWorkBookError
30036	Solve_ReadLINDOWBRCError
30037	Solve_SaveTempFileError
30038	Solve_MinimumFileFormatError
30039	Solve_UnableToLoadSolverError
30040	Solve_ErrorRunningSolver

30050	IntSemic_BadLowerBoundArg
30051	IntSemic_BadUpperBoundArg
30052	IntSemic_BadArgListArg
30053	IntSemic_BadRefersToArg
30054	IntSemic_ProtectedSheetError
30060	IntCard_BadCardNumberArg
30061	IntSos_BadTypeSosArg
30062	IntSos_BadArgListArg
30063	IntSos_BadRefersToArg
30064	IntSos_ProtectedSheetError
30065	IntAllDiff_BadLowerBoundArg
30066	IntAllDiff_BadUpperBoundArg
30067	IntAllDiff_BadArgListArg
30068	IntAllDiff_BadRefersToArg
30069	IntAllDiff_ProtectedSheetError
30070	Int_BadRefers_toArg
30071	Int_BadIntChoiceArg
30072	Int_ProtectedSheetError
30073	Int_CreateError
30080	Opt_BadFeasibilityToleranceArg
30081	Opt_BadIterLimitArg
30082	Opt_BadRuntimeLimitArg
30083	Opt_BadNumberThreadsLimitArg
30084	Opt_BadThreadChoiceArg
30085	Opt_BadSelectRefersToArg
30086	Opt_BadMinimizeExcelArg
30087	Opt_BadHideStatusArg
30088	Opt_BadConstraintIndSlackArg
30089	Opt_BadLinearizationDegArg
30090	Opt_BadDeltaCoeffArg

30091	Opt_BadBigMCoeffArg
30092	Opt_BadStatusReportArg
30093	Opt_BadReportsLocationArg
30094	Opt_BadSolutionReportArg
30095	Opt_BadInfeasConstraintArg
30096	Opt_BadUnboundVarArg
30097	Opt_BadNonlinearityArg
30098	Opt_BadNoObjectiveArg
30099	Opt_BadBlankCellsArg
30100	Opt_BadUnsupportedFunctionArg
30101	Opt_BadStringArgumentArg
30102	Opt_BadIrreConstraintArg
30103	Opt_BadSupLookupFctArg
30104	Opt_BadEditLinksArg
30105	Opt_BadHiddenNamesArg
30110	LinOpt_BadScaleModelArg
30111	LinOpt_BadReductionArg
30112	LinOpt_BadLinearSolverMethodArg
30113	LinOpt_BadDualPricingArg
30114	LinOpt_BadPrimalPricingArg
30120	NonlinOpt_BadStratCrashInitialArg
30121	NonlinOpt_BadStartUseLINDOCrashArg
30122	NonlinOpt_BadStratSLPDirectionArg
30123	NonlinOpt_BadStratSLPSolverArg
30124	NonlinOpt_BadStratQuadraticRecognitionArg
30125	NonlinOpt_BadStratPresolveArg
30126	NonlinOpt_BadStratSelectiveConstraintArg
30127	NonlinOpt_BadStratSteepestEdgeArg
30128	NonlinOpt_BadStartingPointArg
30129	NonlinOpt_BadOptimalityToleranceArg

30130	NonlinOpt_BadItLimitSlowProgArg
30131	NonlinOpt_BadDerivativesArg
30140	GlobalOpt_BadGlStratGlobalArg
30141	GlobalOpt_BadGlMultistartAttemptsArg
30142	GlobalOpt_BadGlOptimalityToleranceArg
30143	GlobalOpt_BadGlDeltaToleranceArg
30144	GlobalOpt_BadGlVariableBoundLimitArg
30145	GlobalOpt_BadGlUseBoundLimitArg
30146	GlobalOpt_BadGlBranchingDirectionArg
30147	GlobalOpt_BadGlBoxSelectionArg
30148	GlobalOpt_BadGlAlgebraicReformulationArg
30150	IntPreSolvOpt_BadHeuristicLevelArg
30151	IntPreSolvOpt_BadCutoffCriterionArg
30152	IntPreSolvOpt_BadProbingLevelArg
30153	IntPreSolvOpt_BadMaxCutsPassesArg
30154	IntPreSolvOpt_BadRelativeCutsLimitArg
30155	IntPreSolvOpt_BadCoefficientReductionCutsArg
30156	IntPreSolvOpt_BadDisaggregationCutsArg
30157	IntPreSolvOpt_BadFlowCoverCutsArg
30158	IntPreSolvOpt_BadGCDCutsArg
30159	IntPreSolvOpt_BadGomoryCutsArg
30160	IntPreSolvOpt_BadGUBCutsArg
30161	IntPreSolvOpt_BadKnapsackCoverCutsArg
30162	IntPreSolvOpt_BadLatticeCutsArg
30163	IntPreSolvOpt_BadLiftingCutsArg
30164	IntPreSolvOpt_BadPlantLocationCutsArg
30165	IntPreSolvOpt_BadObjIntegralityCutsArg
30166	IntPreSolvOpt_BadBasicCutsArg
30167	IntPreSolvOpt_BadCardinalityCutsArg
30168	IntPreSolvOpt_BadDisjunctiveCutsArg

30170	IntOpt_BadBranchingDirectionArg
30171	IntOpt_BadAbsoluteIntegralityArg
30172	IntOpt_BadRelativeIntegralityArg
30173	IntOpt_BadWarmStartArg
30174	IntOpt_BadColdStartArg
30175	IntOpt_BadAbsoluteOptimalityArg
30176	IntOpt_BadRelativeOptimalityArg
30177	IntOpt_BadTimeToRelativeArg
30178	IntOpt_BadHurdleArg
30179	IntOpt_BadNodeSelectionArg
30180	IntOpt_BadStrongBranchArg
30181	IntOpt_BadKBestArg
30182	IntOpt_BadKBestLoadScenarioArg
30183	IntOpt_BadKBestReportArg
30184	IntOpt_BadKBestDisplayPreviewArg
30185	IntOpt_BadBNPBlocksArg
30186	IntOpt_BadBNPHeuristicArg
30187	IKBest_BadArgListRepArg
30188	IKBest_BadRefersToRepArg
30189	IKBest_ProtectedSheetRepError
30190	StoOpt_BadAdvStochasticSupportArg
30191	StoOpt_BadAdvSimulationSupportArg
30192	StoOpt_BadFunctionNameArg
30193	StoOpt_BadRefersToArg
30194	StoOpt_BadPercentArg
30195	StoOpt_BadObjWeightArg
30196	StoOpt_BadStageArg
30197	StoOpt_BadCellRange1Arg
30198	StoOpt_BadCellRange2Arg
30199	StoOpt_BadCellRange3Arg

30200	StoOpt_BadPlaceInCellArg
30201	StoOpt_BadCellRange1CellRange2Arg
30202	StoOpt_BadCellRange1CellRange2CellRange3Arg
30203	StoOpt_BadRangeChoiceArg
30204	StoOpt_BadArgListArg
30205	StoOpt_BadNumberBinsArg
30206	StoOpt_ProtectedSheetRepError
30207	StoOpt_BadStochasticSolverMethodArg
30208	StoOpt_BadStochasticSeedArg
30209	StoOpt_BadStochasticSamplSizeArg
30210	StoOpt_BadStochasticSamplContArg
30211	StoOpt_BadStochasticEVWSArg
30212	StoOpt_BadStochasticEVEMArg
30213	StoOpt_BadStochasticEVPIArg
30214	StoOpt_BadStochasticEVMUArg
30215	StoOpt_BadStochasticRepScenHorizArg
30216	StoOpt_BadUsePreviewWindowArg
30217	StoOpt_BadLoadScenarioViewerArg
30220	Dual_BadDualCellsArg
30221	Dual_BadDualLocArg
30222	Dual_BadDualChoiceArg
30223	Dual_ProtectedSheetError
30224	Omit_BadRefers_toArg
30225	Omit_ProtectedSheetError
30226	Omit_CreateError
30227	AdvFunc_BadAdvFunctionSupportArg
30228	AdvStr_BadAdvStringSupportArg
30229	AdvStr_BadAdvStringLengthArg
30230	Rep_DeleteReportsError
30231	Rep_FormatError

30232	Rep_OpenError
30233	UpdateLinksProtectedError
30234	UpdateLinksHiddenError
30235	CheckAdjustableStyleError
30236	Invalid_Data_Type
30237	CleanUpFilesError
30238	AddWBMenuError
30239	ResetOptionsToDefaultError

WBFREE

For additional discussion of the functionality provided, see the section entitled *Free*, which refers to the dialog box that calls this procedure.

This routine builds *WBFREE* ranges, which are areas of the worksheet where adjustable cells are allowed to be negative.

Syntax: WBFREE(FreeName, [Refers_to])

Argument	***Required***	***Description***
FreeName	Yes	*FreeName* is a string that refers to the range being allowed to be negative.
Refers_to	No	*Refers_to* is the range or address of the cells to be allowed to be negative. If omitted, the default is the current selection.

Error Codes	***Description***
Free_BadRefers_toArg	Bad *Refers_to* argument
Free_ProtectedSheetError	Unable to set free cells on a protected sheet
Free_CreateError	Error setting free cells

Example:
To allow cell G8 to be positive or negative using the range name BuySell, enter:

```
WBFREE "BuySell", "G8"
```

To Remove:

If you wish to delete a *WBFREE* range name using VBA instead of the *Delete* button on the *Free* dialog box, you can use:
ActiveWorkbook.Names("WBFREEBuySell").Delete.
This statement removes the *WBFREE* range name, thereby returning the adjustable cells contained in the former range to normal status.

WBINT

For additional discussion of the functionality provided, see the section entitled *Integer*, which refers to the *Integer* dialog box having a *General* option button that calls this procedure.

This routine is used to build general integer ranges named *WBINT*. The adjustable cells contained in these ranges will be set to non-negative, integral values by the What's*Best!* solver.

Syntax: WBINT(IntName, [Refers_to])

Argument	*Required*	*Description*
IntName	Yes	*IntName* is a string that refers to the range being made integer.
Refers_to	No	*Refers_to* is the range or address of the cell(s) to be specified as integer. If omitted, the default is the current selection.

Error Codes	*Description*
Int_BadIntNameArg	Bad *IntName* argument
Int_BadRefers_toArg	Bad *Refers_to* argument
Int_ProtectedSheetError	Unable to set integer cells on a protected sheet
Int_CreateError	Error setting integer cells

Remarks: The use of integer variables can dramatically increase total solution time.

Example:
To constrain cell F6 to be a general integer using the range name staff, enter:

```
WBINT "staff", "F6"
```

To Remove:
If you wish to delete a *WBBINT* range name using VBA instead of the *Delete* button on the *Integer-Binary* dialog box, you can use:
ActiveWorkbook.Names("WBINTstaff").Delete.
This statement removes the *WBINT* range name, thereby returning the adjustable cells contained in the former range to normal status.

wbInteger

For additional discussion of the functionality provided, see the section entitled *Integer*, which refers to the *Integer* dialog box that calls this procedure.

This routine is used to build the integer ranges *WBBIN* and *WBINT*. Adjustable cells contained in these ranges will be restricted to integral values by the What's*Best!* solver. If you wish to delete an integer range name using VBA instead of the *Delete* button on the *Integer* dialog box, you can use: *ActiveWorkbook.Names("WBBINmyRange").Delete*. This statement removes the integer range name, thereby returning the adjustable cells contained in the former range to normal status.

Syntax: wbInteger([IntName], [Refers_to], [IntChoice])

Argument	*Required*	*Description*
IntName	No	*IntName* is a string that refers to the range being made integer.
Refers_to	No	*Refers_to* is the range or address of the cell(s) to be specified as integer. If omitted, the default value is the current selection.
IntChoice	No	*IntChoice* is a string to indicate if the cells should be binary or general integer. *IntChoice* accepts *BIN* or *BINARY* for binary (0/1) and *INT* or *GENERAL* for general integer. If omitted, the default is binary. For backwards compatibility with earlier versions of What's*Best!*, *IntChoice* also accepts the numbers 1 for binary and 2 for general.

Error Codes	*Description*
Int_BadIntNameArg	Bad *IntName* argument
Int_BadRefers_toArg	Bad *Refers_to* argument
Int_BadIntChoiceArg	Bad *IntChoice* argument
Int_ProtectedSheetError	Unable to set integer cells on a protected sheet
Int_CreateError	Error setting integer cells

Remarks: Integer variables can dramatically increase the solution time.

Example: To constrain cell F6 to be a general integer using the range name *staff*, enter:

```
wbInteger "staff", "F6", "General"
```

To constrain cell F7 to be a binary integer using the range name *OnOff*, enter:

```
wbInteger "OnOff", "F7", "Binary"
```

Note: The second of these two examples could be done without the *IntChoice* argument of "Binary", which is the default.

wbIntegerAllDiff

For additional discussion of the functionality provided, see the section entitled *Integer…|All-different*, which refers to the All-different dialog box that calls this procedure.

wbIntegerAllDiff (Procedure for VBA)
This routine is used to build the All-different ranges *WBALLDIFF*. Adjustable/Integer cells contained in these ranges will be restricted to integral values by the What's*Best!* solver. If you wish to delete a ALLDIFF function using VBA instead of deleting it on the spreadsheet, you can use: *Selection.Clear*. This statement clears the selected cell from any of its content.

Syntax: wbIntegerAllDiff(LowerBound, UpperBound, ArgList, Refers_to, [NoErrDialog])

Argument	***Required***	***Description***
LowerBound	Yes	*LowerBound* is a number indicating the lower bound in the range.
UpperBound	Yes	*UpperBound* is a number indicating the upper bound in the range.
ArgList	Yes	*ArgList* is a string list of range or cells reference separated by a comma.
Refers_to	Yes	*Refers_to* is the range or address of the cell(s) to be specified as ALLDIFF.
NoErrDialog	No	Any argument passed here causes all What's*Best!* error dialog boxes from the wbIntegerAllDiff routine to be suppressed. Instead, the error number is delivered in this return argument. Pass an integer to use any possible returned What's*Best!* error number. Useful in an embedded application of What's*Best!*.

Error Codes	***Description***
IntAllDiff_BadLowerBoundArg	Bad *LowerBound* argument
IntAllDiff_BadUpperBoundArg	Bad *UpperBound* argument
IntAllDiffSemic_BadArgListArg	Bad *ArgList* argument
IntAllDiff_BadRefersToArg	Bad *Refers_to* argument
IntAllDiff_ProtectedSheetError	Unable to set AllDiff cells on a protected sheet

Remarks:
Integer variables can dramatically increase the solution time.

Example:
To constrain cell F6 to be a general integer using the range name *staff*, enter:
wbIntegerAllDiff 10, 20, "Sheet1!$B$1,Sheet1!$C$1:$C$3", "Sheet1!$A$1"
The cell A1 then contains the function:
=WBALLDIFF(10,20,Sheet1!$B$1,Sheet1!$C$1:$C$3)

wbIntegerCard

For additional discussion of the functionality provided, see the section entitled *Integer…|Special Ordered Set*, which refers to the Special Ordered Set - Cardinality dialog box that calls this procedure.

This routine is used to build the Cardinality ranges WBCARD. Adjustable cells contained in these ranges will be restricted to integral values by the What's*Best!* solver. If you wish to delete a CARD function using VBA instead of deleting it on the spreadsheet, you can use: *Selection.Clear*. This statement clears the selected cell from any of its content.

Syntax: wbIntegerCard(CardNumber, ArgList, Refers_to, [NoErrDialog])

Argument	***Required***	***Description***
CardNumber	Yes	CardNumber is an integer meaning that, at most, N of the variables in the set will be allowed to be nonzero.
ArgList	Yes	ArgList is a string list of range or cells reference separated by a comma.
Refers_to	Yes	Refers_to is the range or address of the cell(s) to be specified as SOS.
NoErrDialog	No	Any argument passed here causes all What's*Best!* error dialog boxes from the wbIntegerCard routine to be suppressed. Instead, the error number is delivered in this return argument. Pass an integer to use any possible returned What's*Best!* error number. Useful in an embedded application of What's*Best!*.

Error Codes	***Description***
IntCard_BadCardNumberArg	Bad CardNumber argument
IntSos_BadArgListArg	Bad ArgList argument
IntSos_BadRefersToArg	Bad Refers_to argument
IntSos_ProtectedSheetError	Unable to set integer cells on a protected sheet

Remarks:

Integer variables can dramatically increase the solution time.

Example:

To constrain cell F6 to be a general integer using the range name staff, enter:

wbIntegerCard 3, "Sheet1!$B$1,Sheet1!$C$1:$C$3", "Sheet1!$A$1"

The cell A1 then contains the function:

=WBCARD(3,Sheet1!$B$1,Sheet1!$C$1:$C$3)

wbIntegerSemic

For additional discussion of the functionality provided, see the section entitled *Integer…|Semi-continuous*, which refers to the Semi-continuous dialog box that calls this procedure.

This routine is used to build the Semi-continuous ranges WBSEMIC. Adjustable cells contained in these ranges will be restricted to integral values by the What's*Best!* solver. If you wish to delete a SEMIC function using VBA instead of deleting it on the spreadsheet, you can use: *Selection.Clear*. This statement clears the selected cell from any of its content.

Syntax: wbIntegerSemic(LowerBound, UpperBound, ArgList, Refers_to, [NoErrDialog])

Argument	***Required***	***Description***
LowerBound	Yes	LowerBound is a number indicating the lower bound in the range.
UppererBound	Yes	UpperBound is a number indicating the upper bound in the range.
ArgList	Yes	ArgList is a string list of range or cells reference separated by a comma.
Refers_to	Yes	Refers_to is the range or address of the cell(s) to be specified as SEMIC.
NoErrDialog	No	Any argument passed here causes all What's*Best!* error dialog boxes from the wbIntegerSemic routine to be suppressed. Instead, the error number is delivered in this return argument. Pass an integer to use any possible returned What's*Best!* error number. Useful in an embedded application of What's*Best!*.

Error Codes	***Description***
IntSemic_BadLowerBoundArg	Bad LowerBound argument

IntSemic_BadUpperBoundArg	Bad UpperBound argument
IntSemic_BadArgListArg	Bad ArgList argument
IntSemic_BadRefersToArg	Bad Refers_to argument
IntSemic_ProtectedSheetError	Unable to set integer cells on a protected sheet

Remarks:

Integer variables can dramatically increase the solution time.

Example:

To constrain cell F6 to be a general integer using the range name staff, enter:

wbIntegerSemic 10, 20, "Sheet1!$B$1,Sheet1!$C$1:$C$3", "Sheet1!$A$1"

The cell A1 then contains the function:

=WBSEMIC(10,20,Sheet1!$B$1,Sheet1!$C$1:$C$3)

wbIntegerSos

For additional discussion of the functionality provided, see the section entitled *Integer…|Special Ordered Set*, which refers to the *Special Ordered Set - Cardinality* dialog box that calls this procedure.

This routine is used to build the SOS ranges *WBSOSx*. Adjustable cells contained in these ranges will be restricted to integral values by the What's*Best!* solver. If you wish to delete a SOS function using VBA instead of deleting it on the spreadsheet, you can use: *Selection.Clear*. This statement clears the selected cell from any of its content.

Syntax: wbIntegerSos(TypeSos, ArgList, Refers_to, [NoErrDialog])

Argument	***Required***	***Description***
TypeSos	Yes	TypeSos is an integer that refers to the type of the SOS: - "1" for type 1, - "2" for type 2, - "3" for type 3.
ArgList	Yes	ArgList is a string list of range or cells reference separated by a comma.
Refers_to	Yes	Refers_to is the range or address of the cell(s) to be specified as SOS.
NoErrDialog	No	Any argument passed here causes all What's*Best!* error dialog boxes from the wbIntegerSos routine to be suppressed. Instead, the error number is delivered in this return argument. Pass an integer to use any possible returned What's*Best!* error number. Useful in an embedded application of What's*Best!*.

Error Codes	Description
IntSos_BadTypeSosArg	Bad TypeSos argument
IntSos_BadArgListArg	Bad ArgList argument
IntSos_BadRefersToArg	Bad Refers_to argument
IntSos_ProtectedSheetError	Unable to set integer cells on a protected sheet

Remarks:

Integer variables can dramatically increase the solution time.

Example:

To constrain cell F6 to be a general integer using the range name *staff*, enter:

wbIntegerSos 1, "Sheet1!$B$1,Sheet1!$C$1:$C$3", "Sheet1!$A$1"

The cell A1 then contains the function:

=WBSOS1(Sheet1!$B$1,Sheet1!$C$1:$C$3

wbIntegerSolverKBestReport

For additional discussion of the functionality provided, see the section entitled *Options...|Integer Solver*, and *Usage Guidelines for K-Best Solutions*, which refers to the dialog box that calls this procedure.

This routine is used to build *WBIKB_REP* function with ranges, which are areas of the worksheet that What's*Best!* will see the trade-off cells while solving. This function will need to be used in conjunction with the option *KBest* defined via the *wbSetIntegerOptions*.

Syntax: wbIntegerSolverKBestReport(ArgList, Refers_to, [NoErrDialog])

Argument	Required	Description
KBestName	Yes	*ArgList* is the range or address of the cell(s) to be read.
Refers_to	Yes	*Refers_to* is a cell reference to specify in which to write the *WBIKB_REP* function.
NoErrDialog	No	Any argument passed here causes all What's*Best!* error dialog boxes from *the wbIntegerSolverKBestReport* routine to be suppressed. Instead, the error number is delivered in this return argument. Pass an integer to use any possible returned What's*Best!* error number. Useful in an embedded application of What's*Best!*.

Error Codes	Description
IKBest_BadArgListRepArg	Bad *ArgList* argument
IKBest_BadRefersToRepArg	Bad *Refers_to* argument
IKBest_ProtectedSheetRepError	Unable to set KBest cells on a protected sheet

Example:

To define an KBest trade-off range F3:F10, and place the function WBIKB_REP in H3, enter:

```
wbIntegerSolverKBestReport "F3:F10", "H3"
```

See sample *Knapsack_KBest.xlsx*.

WBOMIT

For additional discussion of the functionality provided, see the section entitled *Advanced…|Omit*, which refers to the dialog box that calls this procedure.

This routine is used to build *WBOMIT* ranges, which are areas of the worksheet that What's*Best!* will ignore while solving. If you wish to delete a *WBOMIT* range name using VBA instead of the *Delete* button on the *Omit* dialog box, you can use: *ActiveWorkbook.Names("WBOMITmyRangeName").Delete*. This statement removes the *WBOMIT* range name.

Syntax: WBOMIT(OmitName, [Refers_to])

Argument	Required	Description
OmitName	Yes	*OmitName* is a string that refers to the range that is to be ignored.
Refers_to	No	*Refers_to* is the range or address of the cell(s) to be ignored. If omitted, the default is the current selection.

Error Codes	Description
Omit_BadOmitNameArg	Bad *OmitName* argument
Omit_BadRefers_toArg	Bad *Refers_to* argument
Omit_ProtectedSheetError	Unable to set omit cells on a protected sheet
Omit_CreateError	Error setting omit cells

Remarks: Cells that are in an omit range can't be referenced by any cell outside of an omit range.

Example:
To define an omit range D10:H20 using the range name *Report*, enter:

```
WBOMIT "Report", "D10:H20"
```

To Remove:
If you wish to delete a *WBOMIT* range name using VBA instead of the *Delete* button on the *Omit* dialog box, you can use:
ActiveWorkbook.Names("WBOMITReport").Delete.
This statement removes the *WBOMIT* range name, thereby returning the adjustable cells contained in the former range to normal status.

WBProgressInfoRequest

This routine is used to set 1) the name of the user progress callback procedure, and 2) the time interval between callbacks in seconds

Syntax: WBProgressInfoRequest(callbackName, [timeIntervalParam])

Argument	***Required***	***Description***
callbackName	Yes	*callbackName* is a string storing the name of the user progress callback procedure. See below for more details.
timeIntervalParam	No	*frequencyParam* is a positive number indicating the time between callbacks in seconds. The default value is 1 second.

What'sBest! enables the user to define a callback procedure for accessing the progress information during the solution process. This information includes that displayed in the status window. The required steps to do this are:

1) A public procedure (with an arbitrary name) without any arguments should be defined. This procedure represents the user defined callback (see the *userProgressCallback* procedure of the "CustomVBAInterface.xlsm" sample case).

2) The name of the callback procedure defined in the previous step and the time interval between callbacks (optional) should be set via the *WBProgressInfoRequest* routine. The format is (let's say the name of the callback procedure is *userProgressCallback)*:
WBProgressInfoRequest "userProgressCallback"

   Note that the name should be set before calling the *wbSolve* procedure.

Then, the updated progress information can be accessed safely within the user progress callback procedure (defined in step 1) via the global variables described below. The first and second columns represent the names and types of the global variables, respectively. The third column shows the status window items corresponding to each variable. For example, the *APIProgNumerics* is a long variable storing the value of the "Numerics" item of the status window.

Variable	*Type*	*Item*
APIProgNumerics	Long	Numerics
APIProgAdjustables	Long	Adjustables
APIProgIntegersBin	Long	Integers/Bin.
APIProgFormulas	Long	Formulas
APIProgConstraints	Long	Constraints
APIProgNonLinears	Long	Nonlinears
APIProgCoefficients	Long	Coefficients
APIProgObjDirection	String	Obj. Direction
APIProgActivity	String	Activity
APIProgModelType	String	Model Type
APIProgState	String	State
APIProgIterations	Long	Iterations
APIProgInfeasibility	Double	Infeasibility
APIProgObjective	Double	Objective
APIProgSolverType	String	Solver Type
APIProgBestObj	Double	Best Obj.
APIProgObjBound	Double	Obj. Bound
APIProgSteps	Long	Steps
APIProgActive	Long	Active

The value -999999 for the double and long variables and “” for the string variables show that the corresponding items in the status window are blank.

The user can also interrupt the solution process via VBA interface. To do this, the value of the global long variable APIInterrupt should be set to 1 (its default value is 0).

A sample case is provided with the file "CustomVBAInterface.xlsm". This is a variant of the “XYZVBA.xlsm” sample in which a customized status window is displayed instead of the traditional one. This customized status window employs the progress callback feature.

wbResetOptionsToDefault

For further discussion of the functionality provided, see the section entitled *Options…|Reset To Default*.

This procedure has no arguments and resets all of the workbook options to their default values.

Syntax: wbResetOptionsToDefault()

Error Code	Description
ResetOptionsToDefaultError	Error in reseting options to default

wbSetFunctionSupport

This routine can be used to set the What's*Best!* function support option seen in the *Function Support* dialog box. There is only one argument required. For additional discussion of the options available through this routine, see the section entitled *Advanced...|FunctionSupport*.

Syntax: wbSetFunctionSupport(AdvFunctionSupport)

Argument	Required	Default	Description
AdvFunctionSupport	Yes	0 (False)	*AdvFunctionSupport* is a True/False flag indicating whether or not to use this support.

Error Codes	Description
AdvFunc_BadAdvFunctionSupportArg	Bad *AdvFunctionSupport* argument

Example:

If one wished to set all the function support options, the simplest syntax would be:

```
wbSetFunctionSupport True
```

or by the named argument:

```
wbSetFunctionSupport AdvFunctionSupport:=True
```

Note: The *AdvFunctionSupport* argument should be set to TRUE in order to use the other arguments.

wbSetGeneralOptions

This routine is used to set the What's*Best!* general options. These options constitute the majority of the options seen in the General Options dialog box. All arguments are optional. For additional discussion of the options available through this routine, see the section entitled *Options… |General*.

Syntax: wbSetGeneralOptions([goFeasTol], [goIterLimit], [goRuntimeLimit], [goNumberThreadsLimit], [goThreadChoice], [goIndSlack], [goAutoSelectFreeIntOmit], [goMinimizeExcel], [goHideStatusWindow], [goLinearizationDegree], [goLinearizationDelta], [goLinearizationBigM], [goStatusReport], [goStatusReportBegEnd], [goSolutionReport], [goWrnInfeasConst], [goWrnUnboundVar], [goWrnNonlinear], [goWrnBest], [goWrnBlank], [goWrnFunction], [goWrnStringArg], [goWrnIrreConst], [goWrnSupLookupFct], [goWrnEditLinks], [goWrnHiddenNames])

Argument	Required	Default	Description
goFeasTol	No	0.0000001	*goFeasibilityTol* is a positive number indicating the amount of violation tolerated in constraints.
goIterLimit	No	None	*goIterLimit* is an integer indicating how many tries to allow in solving the model.
goRuntimeLimit	No	None	*goRuntimeLimit* is an integer indicating the maximum number of seconds to be used in solving the model.
goNumberThreadsLimit	No	1	*goNumberThreadsLimit* is an integer specifying the number of threads, between 1 and 32.
goThreadChoice	No	0	*goThreadChoice* is an integer specifying the choice for the threads: 0: Solver Decides, 1: Parallel Preferred, 2: Parallel Only, 3: Concurrent Preferred, 4: Concurrent Only.
goIndSlack	No	1 (Indicator)	*goIndSlack* is an integer indicating what form to display constraints in (1-Indicator, 2 – Slack).
goAutoSelectFreeIntOmit	No	1 (True)	*goAutoSelectFreeInt* is a True/False flag indicating whether to automatically select Free, Integer or Omit range names.
goMinimizeExcel	No	0 (False)	*goMinimizeExcel* is a True/False flag indicating whether to minimize Excel® during the solution process.

goHideStatusWindow	No	0 (False)	*goHideStatusWindow* is a True/False flag indicating whether to minimize the status window during the solution process.
goLinearizationDegree	No	0 (Solver Decides)	*goLinearizationDegree* is an integer indicating the degree of linearization to use: 0: Solver Decides, 1: None, 2: Mathematical, 3: Mathematical, Logical.
goLinearizationDelta	No	0.000001	*goLinearizationDelta* is a number indicating the Delta coefficient.
goLinearizationBigM	No	100000	*goLinearizationBigM Coefficient* is a number indicating the Big M coefficient.
goStatusReport	No	1 (Always Created)	*goStatusReport* is an integer indicating when to show the status report: 0: Never Created, 1: Always Created, 2: Only on Error/Warning.
goStatusReportBegEnd	No	0 (Beginning)	*goStatusReportBegEnd* is a flag indicating the location of the reports in the workbook: 0: Beginning, 1: End.
goSolutionReport	No	0 (Never Created)	*goSolutionReport* is an integer indicating whether to show the solution report: 0: Never Created, 1: Always Created, 2: Always Created with Details, 3: Full Report.
goWrnInfeasConst	No	0 (False)	*goWrnInfeasConst* is a True/False flag indicating whether to provide warning of infeasible constraints or not.
goWrnUnboundVar	No	0 (False)	*goWrnUnboundVar* is a True/False flag indicating whether to provide warning of unbounded variables or not.
goWrnNonlinear	No	1 (True)	*Nonlinearity* is a True/False flag indicating whether to provide warning of nonlinear formulas or not.
goWrnBest	No	1 (True)	*goWrnBest* is a True/False flag

			indicating whether to provide warning on presence of Best cell.
goWrnBlank	No	0 (False)	*goWrnBlank* is a True/False flag indicating whether to provide warning of blank cells referenced in formulas or not.
goWrnFunction	No	1 (True)	*goWrnFunction* is a True/False flag indicating whether to provide warning of unsupported functions or not.
goWrnStringArg	No	1 (True)	*goWrnStringArg* is a True/False flag indicating whether to provide warning of string, text argument in formulas.
goWrnIrreConst	No	1 (True)	*goWrnIrreConst* is a True/False flag indicating whether to provide warning of irreconciliable constraints or not.
goWrnSupLookupFct	No	1 (True)	*goWrnSupLookupFct* is a True/False flag indicating whether to provide warning of supported lookup functions or not.
goWrnSupLookupFct	No	1 (True)	*goWrnSupLookupFct* is a True/False flag indicating whether to provide warning of supported lookup functions or not.
goWrnEditLinks	No	0 (False)	*goWrnEditLinks* is a True/False flag indicating whether to display any suspicious add-in links that may be corrupted in the workbook.
goWrnHiddenNames	No	0 (False)	*goWrnHiddenNames* is a True/False flag that notifies about names generated by Excel® or any other add-ins, and are not showing via the Name Manager menu of Excel®.

Error Codes	***Description***
Opt_BadFeasibilityToleranceArg	Bad Feasibility Tolerance argument
Opt_BadIterLimitArg	Bad Iteration Limit argument
Opt_BadRuntimeLimitArg	Bad Runtime Limit argument
Opt_BadSelectRefersToArg	Bad Select Refers To argument
Opt_BadNumberThreadsLimitArg	Bad Number Threads Limit To argument
Opt_BadThreadChoiceArg	Bad Thread Choice argument

Opt_BadMinimizeExcelArg	Bad Minimize Excel® argument
Opt_BadHideStatusArg	Bad Hide Status argument
Opt_BadConstraintIndSlackArg	Bad Constraint Indicator argument
Opt_BadLinearizationDegArg	Bad Linearization Degree argument
Opt_BadDeltaCoeffArg	Bad Delta coefficient argument
Opt_BadBigMCoeffArg	Bad Big M coefficient argument
Opt_BadStatusReportArg	Bad Status Report argument
Opt_BadInfeasConstraintArg	Bad *Infeasible Constraint* argument
Opt_BadUnboundVarArg	Bad *Unbounded Variable* argument
Opt_BadNonlinearityArg	Bad *Nonlinearity* argument
Opt_BadNoObjectiveArg	Bad *No Objective* argument
Opt_BadBlankCellsArg	Bad *Blank Cells* argument
Opt_BadUnsupportedFunctionArg	Bad *Unsupported Function* argument
Opt_BadStringArgumentArg	Bad *String Argument* argument
Opt_BadIrreConstraintArg	Bad *Irreconcilable Constraint* argument
Opt_BadSupLookupFctArg	Bad *Support Lookup Functions* argument
Opt_BadHiddenNamesArg	Bad *Hidden Names* argument

Example:

Arguments should be provided as follows:

If one wished to set all the options, the simplest syntax would be:

```
wbSetGeneralOptions 0.0000002, 77, 88, 1, 0,  2, True, False, False, _
1, 0.2, 222222, 1, 1, 1, False, True, False, False, _
False, True, False, False, True, False, False
```

To set one or more options, named arguments should be used:

```
wbSetGeneralOptions goSolutionReport:=1, goWrnBlank:=True
```

wbSetGlobalOptions

This routine is used to set the What's*Best!* global solver options seen in the *Global Solver Options* dialog box. For additional discussion of the options available through this routine, see the section entitled *Options. . . |Global Solver*. All arguments are optional.

Syntax: wbSetGlobalOptions([GlStratGlobal], [GlMultistartAttempts], [GlOptimalityTolerance], [GlDeltaTolerance], [GlVariableBoundLimit], [GlUseBoundLimit], [GlBranchingDirection], [GlBoxSelection], [GlAlgebraicReformulation])

Argument	*Required*	*Default*	*Description*
GlStratGlobal	No	0 (False)	*GlStratGlobal* is a True/False flag indicating whether or not to use this strategy.
GlMultistartAttempts	No	0 (Solver Decides)	*GlMultistartAttempts* is a positive number indicating the multistart attempts: 0: *Solver Decides*, 1: *Off*, 2 or more: number of attempts.
GlOptimalityTolerance	No	0.000001	*GlOptimalityTolerance* is a number between 0 and 1 indicating the optimality tolerance.
GlDeltaTolerance	No	0.0000001	*GlDeltaTolerance* is a number between 0 and 1 indicating the delta tolerance.
GlVariableBoundLimit	No	1000000000	*GlVariableBoundLimit* is a positive integer indicating the variable bound limit.
GlUseBoundLimit	No	0 (Solver Decides)	*GlUseBoundLimit* indicates the use of bound limit:

			0: *Solver Decides*, 1: *None*, 2: *All Variables*, 3: *Selected Variables*.
GlBranchingDirection	No	0 (Solver Decides)	*GlBranchingDirection* indicates the branching direction: 0: *Solver Decides*, 1: *Absolute Width*, 2: *Local Width*, 3: *Global Width*, 4: *Global Distance*, 5: *Absolute Violation*, 6: *Relative Violation*.
GIBoxSelection	No	0 (Solver Decides)	*GlBoxSelection* indicates the box selection: 0: *Solver Decides*, 1: *Depth First*, 2: *Worst Bound*, 3: *Best Bound.*
GlAlgebraicReformulation	No	0 (Solver Decides)	*GlAlgebraocReformulation* indicates the algebraic reformulation: 0: *Solver Decides*, 1: *None*, 2: *Minimum*, 3: *Medium*, 4: *Maximum*.

Error Codes	***Description***
GlobalOpt_BadGlStratGlobalArg	Bad *Global Solver* argument.
GlobalOpt_BadGlMultistartAttemptsArg	Bad *Multistart Attempts* argument.
GlobalOpt_BadGlOptimalityToleranceArg	Bad *Optimality Tolerance* argument.
GlobalOpt_BadGlDeltaToleranceArg	Bad *Delta Tolerance* argument.
GlobalOpt_BadGlVariableBoundLimitArg	Bad *Variable Bound Limit* argument.
GlobalOpt_BadGlUseBoundLimitArg	Bad *Use of Bound Limit* argument.
GlobalOpt_BadGlBranchingDirectionArg	Bad *Branching Direction* argument.
GlobalOpt_BadGlBoxSelectionArg	Bad *Box Selection* argument.
GlobalOpt_BadGlAlgebraicReformulationArg	Bad *Algebraic Reformulation* argument.

Example: If one wished to set all the options, the following would work:

```
wbSetGlobalOptions True, 4, 0.0001, 0.000001, 100000000,_
1, 2, 1, 2
```

To set individual options (here setting *Multistart Attempts* to 4), named arguments should be used:

```
wbSetGlobalOptions GlMultistartAttempts:=4
```

Note: The global solver needs the nonlinear option and the mixed integer option to operate.

wbSetIntegerOptions

This routine is used to set the What's*Best!* integer options displayed in the *Integer Solver Options* dialog box. All arguments are optional. For additional discussion of the options available through this routine, see the section entitled *Options. . .* |*Integer Solver*.

Syntax: wbSetIntegerOptions([BranchingDirection], [AbsoluteIntegrality], [RelativeIntegrality], [WarmStartLP], [ColdStartLP], [AbsoluteOptimality], [RelativeOptimality], [TimeToRelativeOptimality], [HurdleTolerance], [NodeSelectionTolerance], [StrongBranchTolerance], [KBest], [KBestLoadScenario] ,[KBestReport], [KBestDisplayPreview],[BNPBlocks],[BNPHeuristic])

Argument	*Required*	*Default*	*Description*
BranchingDirection	No	0 (Both)	*BranchingDirection* is an integer indicating the preferred direction of branching: 0: *Both*, 1: *Up*, 2: *Down*.
AbsoluteIntegrality	No	0.000001	*AbsoluteIntegrality* is a fractional value indicating the absolute amount of violation from integrality that is acceptable for the integer variables.
RelativeIntegrality	No	0.000008	*RelativeIntegrality* is a fractional value indicating the relative amount of violation from integrality that is acceptable for the integer variables.
WarmStartLP	No	0 (Solver Decides)	*WarmStartLP* is an integer value indicating the linear solver to use during branch-and-bound when a starting basis is present: 0: *Solver Decides*, 1: *Barrier*, 2: *Primal*, 3: *Dual*.

ColdStartLP	No	0 (Solver Decides)	*ColdStartLP is* an integer value indicating the linear solver to use during branch-and-bound when a starting basis is *not* present: 0: *Solver Decides*, 1: *Barrier*, 2: *Primal*, 3: *Dual*.
AbsoluteOptimality	No	0	*AbsoluteOptimality* is a positive value *r*, indicating to the branch-and-bound solver that it should only search for integer solutions with objective values at least *r* units better than the best integer solution found so far.
RelativeOptimality	No	0.00001	*RelativeOptimality* is a value *r*, ranging from 0 to 1, indicating to the branch-and-bound solver that it should only search for integer solutions with objective values at least 100**r*% better than the best integer solution found so far.
TimeToRelativeOptimality	No	100	*TimeToRelativeOptimality* is the number of seconds before the branch-and-bound solver resorts to using the *RelativeOptimality* tolerance.
HurdleTolerance	No	None	*HurdleTolerance* is a value of a known integer solution. What's*Best!* will only search for any feasible integer solution if it is at least as good as the bound specified in *HurdleTolerance* (0: None).
NodeSelectionTolerance	No	0 (Solver Decides)	*NodeSelectionTolerance* controls the order in which the branch-and-bound solver selects branch nodes in the tree: 0: *Solver Decides*, 1: *Depth First*, 2: *Worst Bound*, 3: *Best Bound*.
StrongBranchTolerance	No	10	The *StrongBranchTolerance* option uses a more intensive branching strategy during the first *n* levels of the branch-and-bound tree, where *n* is the option's setting. During these initial levels, the solver picks a subset of the fractional

			variables as branching candidates, performs a tentative branch on each of the variables in the subset, and selects as the final candidate the variable that offers the greatest improvement in the bound on the objective.
KBest	No	1	*KBest* is an integer indicating the number of solutions desired as part of the K-Best solutions feature. The WBKBEST function is used to specify the trade-off cells.
KBestLoadScenario	No	False (Off)	*KBestLoadScenario* is a boolean for loading the Scenario Window with the K-Best solutions. The Scenario Window is then selected via the menu.
KBestReport	No	False (Off)	*KBestReport* is a boolean indicating the generation of the K-Best report. The reporting cells may be defined with the function *wbIntegerSolverKBestReport.*
KBestDisplayPreview	No	False (Off)	*KBestDisplayPreview* is a boolean indicating the display of the Preview Window for the K-Best solutions upon solving . The reporting cells may be defined with the function *wbIntegerSolverKBestReport.*
BNPBlocks	No	0 or 1 (Off)	*BNPBlocks* is an integer indicating the number of blocks defined with the Branch-and-Price solver, and should be greater or equal than 2.
BNPHeuristic	No	1 (GP1)	*BNPHeuristic* is an integer indicating the type of heuristic to use as part of the Branch-and-Price solver, where 1 is GP1, and 2 for GP2.

Error Codes	***Description***
IntOpt_BadBranchingDirectionArg	Bad *BranchingDirection* argument.
IntOpt_BadAbsoluteIntegralityArg	Bad *AbsoluteIntegrality* argument.
IntOpt_BadRelativeIntegralityArg	Bad *RelativeIntegrality* argument.

IntOpt_BadWarmStartArg	Bad *WarmStart* argument.
IntOpt_BadColdStartArg	Bad *ColdStart* argument.
IntOpt_BadAbsoluteOptimalityArg	Bad *AbsoluteOptimality* argument.
IntOpt_BadRelativeOptimalityArg	Bad *RelativeOptimality* argument.
IntOpt_BadTimeToRelative	Bad *TimeToRelative* argument.
IntOpt_BadHurdleArg	Bad *Hurdle* argument.
IntOpt_BadNodeSelectionArg	Bad *NodeSelection* argument.
IntOpt_BadStrongBranchArg	Bad *StrongBranch* argument.
IntOpt_BadKBestArg	Bad *IntOpt_BadKBestArg* argument.
IntOpt_BadKBestLoadScenarioArg	Bad *IntOpt_BadKBestLoadScenarioArg* argument.
IntOpt_BadKBestReportArg	Bad *IntOpt_BadKBestReportArg* argument.
IntOpt_BadKBestDisplayPreviewArg	Bad *IntOpt_BadKBestDisplayPreviewArg* argument.
IntOpt_BadBNPBlocksArg	Bad *IntOpt_BadBNPBlocksArg* argument.
IntOpt_BadBNPHeuristicArg	Bad *IntOpt_BadBNPHeuristicArg* argument.

Example:

If one wished to set all the options, the following would work:

```
wbSetIntegerOptions 1, .000001, .00001, 3, 1, .00008, _
.01, 120, 0, 1, 20, 1, False, False, False, 0, 1
```

To set individual options (here setting the *RelativeOptimality* tolerance to 10%), named arguments should be used:

```
wbSetIntegerOptions RelativeOptimality:=.1
```

wbSetIntegerPreSolverOptions

This routine is used to set the What's*Best!* integer pre-solver options seen in the *Integer Pre-Solver Options* dialog box. For additional discussion of the options available through this routine, see the section entitled *Options. . .* |*Integer Pre-Solver*. All arguments are optional.

Syntax: wbSetIntegerPreSolverOptions([HeuristicLevel], [ProbingLevel], [MaxCutsPasses], [RelativeCutsLimit], [CoefficientReduction], [Disaggregation], [FlowCover], [GCD], [Gomory], [GUB], [KnapsackCover], [Lattice], [Lifting], [PlantLocation], [ObjIntegrality], [Basic], [Cardinality], [Disjunctive])

Argument	*Required*	*Default*	*Description*
HeuristicLevel	No	3 (High)	*HeuristicLevel* controls the level of integer programming heuristics used by the integer solver. These heuristics use the continuous solution at each node in the branch-and-bound tree to attempt to quickly find a good integer solution: 0: *None*, 1: *Low*, 2: *Medium*, 3: *High*.
Cutoff Criterion	No	0 (Solver Decides)	The *Cutoff Criterion* is used to control the criterion for terminating heuristics: 0: *Solver Decides*, 1: *Time*, 2: *Iterations*.
ProbingLevel	No	0 (Solver Decides)	*ProbingLevel* controls the amount of probing that occurs in integer models. Probing involves taking a close look at the integer variables in a model and deducing tighter variable bounds and right-hand side values. In many cases, probing can tighten an integer model sufficiently to speed overall solution times: 0: *Solver Decides*, 1: *None*, 2-10: increasing *levels* of probing.
MaxCutsPasses	No	200	*MaxCutsPasses* sets the maximum number of iterative passes through a model determining appropriate constraint cuts to append to the formulation.
RelativeCutsLimit	No	0.5	*RelativeCutsLimit* is a fractional value imposing a relative limit on the number of constraint cuts that are generated. The default

			limit is set to 0.5 times the number of true constraints in the original formulation.
CoefficientReduction	No	1 (True)	*CoefficientReduction* is used to enable or disable coefficient reduction cuts (False: *Disable*, True: *Enable*).
Disaggregation	No	1 (True)	*Disaggregation* used to enable or disable disaggregation cuts (False: *Disable*, True: *Enable*).
FlowCover	No	1 (True)	*FlowCover* is used to enable or disable flow cover cuts (0: *Disable*, 1: *Enable*).
GCD	No	1 (True)	*GCD* is used to enable or disable greatest common denominator cuts (0: *Disable*, 1: *Enable*).
Gomory	No	1 (True)	*Gomory* is used to enable or disable Gomory cuts (0: *Disable*, 1: *Enable*).
GUB	No	1 (True)	*GUB* is used to enable or disable generalized upper bound cuts (0: *Disable*, 1: *Enable*).
KnapsackCover	No	1 (True)	*KnapsackCover* is used to enable or disable knapsack cover cuts (0: *Disable*, 1: *Enable*).
Lattice	No	1 (True)	*Lattice* is used to enable or disable lattice cuts (0: *Disable*, 1: *Enable*).
Lifting	No	1 (True)	*Lifting* is used to enable or disable lifting cuts (0: *Disable*, 1: *Enable*).
PlantLocation	No	1 (True)	*PlantLocation* is used to enable or disable plant location cuts (0: *Disable*, 1: *Enable*).
ObjIntegrality	No	0 (False)	*ObjIntegrality* is used to enable or disable the objective integrality cuts (0: *Disable*, 1:*Enable*)
Basic	No	1 (True)	*Basic* is used to enable or disable the basic cuts (0: *Disable*, 1: Enable)
Cardinality	No	0 (False)	*Cardinality* is used to enable or disable the objective cardinality cuts (0: *Disable*, 1: *Enable*)
Disjunctive	No	1 (True)	*Disjunctive* is used to enable or disable the objective disjunctive cuts (0: Disable, 1: Enable).

Error Codes	***Description***
IntPreSolvOpt_BadHeuristicLevelArg	Bad *Heuristic Level* argument.
IntPreSolvOpt_BadCutoffCriterionArg	Bad *Cutoff Criterion* argument.
IntPreSolvOpt_BadProbingLevelArg	Bad *Probing Level* argument.
IntPreSolvOpt_BadMaxCutsPassesArg	Bad *Max Cuts Passes* argument.
IntPreSolvOpt_BadRelativeCutsLimitArg	Bad *Relative Cuts Limit* argument.
IntPreSolvOpt_BadCoefficientReductionCutsArg	Bad *Coefficient Reduction Cuts* argument.
IntPreSolvOpt_BadDisaggregationCutsArg	Bad *Disaggregation Cuts* argument.
IntPreSolvOpt_BadFlowCoverCutsArg	Bad *Flow Cover Cuts* argument.
IntPreSolvOpt_BadGCDCutsArg	Bad *GCD Cuts* argument.
IntPreSolvOpt_BadGomoryCutsArg	Bad *Gomory Cuts* argument.
IntPreSolvOpt_BadGUBCutsArg	Bad *GUB Cuts* argument.
IntPreSolvOpt_BadKnapsackCoverCutsArg	Bad *Knapsack Cover Cuts* argument.
IntPreSolvOpt_BadLatticeCutsArg	Bad *Lattice Cuts* argument.
IntPreSolvOpt_BadLiftingCutsArg	Bad *Lifting Cuts* argument.
IntPreSolvOpt_BadPlantLocationCutsArg	Bad *Plant Location Cuts* argument.
IntPreSolvOpt_BadObjIntegralityCutsArg	Bad *Objective Integrality Cuts* argument.
IntPreSolvOpt_BadBasicCutsArg	Bad *Basic Cuts* argument.
IntPreSolvOpt_BadCardinalityCutsArg	Bad *Cardinality Cuts* argument.
IntPreSolvOpt_BadDisjunctiveCutsArg	Bad *Disjunctive* Cuts argument.

Example: If one wished to set all the options, the following would work:

```
wbSetIntegerPreSolverOptions 2, 1, 6, 20, .3, _
False, False, True, True, False, True, True, _
False, False, True, True, False, True, True
```

To set individual options (here setting the *ProbingLevel* to 4), named arguments should be used:

```
wbSetIntegerPreSolverOptions ProbingLevel:=4
```

wbSetLinearOptions

This routine is used to set the What's*Best!* linear solver options, which can be seen in the *Linear Solver Options* dialog box. All arguments are optional. For additional discussion of the options available through this routine, see the section entitled *Options…|Linear Solver*.

Syntax: wbSetLinearOptions([LinearSolverMethod], [ScaleModel], [Reduction], [PrimalPricing], [DualPricing])

Argument	*Required*	*Default*	*Description*
LinearSolverMethod	No	0 (Solver Decides)	*LinearSolverMethod* is an integer indicating the linear solver method to employ: 0: *Solver Decides*, 1: *Primal Simplex*; 2: *Dual Simplex*.
ScaleModel	No	1 (True)	*ScaleModel* is a True/False flag indicating whether to scale model.
Reduction	No	1 (True)	*Reduction* is a True/False flag indicating whether to use model reduction.
PrimalPricing	No	0 (Solver Decides)	*PrimalPricing* is an integer indicating what primal pricing method to use: 0: *Solver Decides*, 1: *Devex*; 2: *Partial*.
DualPricing	No	0 (Solver Decides)	*DualPricing* is an integer indicating what dual pricing method to use: 0: *Solver Decides*, 1: *Partial*, 2: *Steepest Edge*.

Error Codes	*Description*
LinOpt_BadScaleModelArg	Bad ScaleModel argument
LinOpt_BadReductionArg	Bad Reduction argument
LinOpt_BadLinearSolverMethodArg	Bad LinearSolverMethod argument
LinOpt_BadDualPricingArg	Bad DualPricing argument
LinOpt_BadPrimalPricingArg	Bad PrimalPricing argument

Example: If one wished to set all the linear solver options, the simplest syntax would be:

```
wbSetLinearOptions 3, False, True, 2, 2
```

To set one or more options, named arguments should be used:

```
wbSetLinearOptions Reduction:=True
```

wbSetNonlinearOptions

This routine is used to set the What's*Best!* nonlinear solver options seen in the Nonlinear Solver Options dialog box. All arguments are optional. For additional discussion of the options available through this routine, see the section entitled Options…|Nonlinear Solver.

Syntax: *wbSetNonlinearOptions*([StratCrashInitial], [StratUseLINDOCrash], [StratSLPDirection], [StratSLPSolver], [StratQuadraticRecognition], [StratPresolve], [StratSelectiveConstraint], [StratSteepestEdge], [StartingPoint], [OptimalityTolerance], [ItLimitSlowProg], [Derivatives])

Argument	*Required*	*Default*	*Description*
StratCrashInitial	No	0 (False)	*StratCrashInitial* is a True/False flag indicating whether or not to use this strategy.
StratUseLINDOCrash	No	0 (False)	*StratUseLINDOCrash* is a True/False flag indicating whether or not to use this strategy.
StratSLPDirection	No	1(True)	*StratSLPDirection* is a True/False flag indicating whether or not to use this strategy.
StratSLPSolver	No	0 (False)	*StratSLPSolver* is a True/False flag indicating whether or not to use this strategy.
StratQuadraticRecognition	No	1(True)	*StratQuadraticRecognition* is a True/False flag indicating whether or not to use this strategy.
StratPresolve	No	1(True)	*StratPresolve* is a True/False flag indicating whether or not to use this strategy.
StratSelectiveConstraint	No	1(True)	*StratSelectiveConstraint* is a True/False flag indicating whether or not to use this strategy.
StratSteepestEdge	No	0 (False)	*StratSteepestEdge* is a True/False flag indicating whether or not to use this strategy.
StartingPoint	No	0 (False)	*StartingPoint* is a True/False flag indicating whether or not to use this feature.
OptimalityTolerance	No	0.0000001	*OptimalityTolerance* is a positive number indicating the optimality tolerance.

ItLimitSlowProg	No	100	*ItLimitSlowProg* is an integer greater than 2 indicating the iteration limit for slow progress.
Derivatives	No	0 (Solver Decides)	*Derivatives* can take 5 values: 0 for Solver Decides, 1 for Backward Analytical, 2 for Forward Analytical, 3 for Central Differences, 4 for Forward Differences.

Error Codes	***Description***
NonlinOpt_BadStratCrashInitialArg	Bad StratCrashInitial argument
NonlinOpt_BadStratUseLINDOCrashArg	Bad StratUseLINDOCrash argument
NonlinOpt_BadStratSLPDirectionArg	Bad StratSLPDirection argument
NonlinOpt_BadStratSLPSolverArg	Bad StratSLPSolver argument
NonlinOpt_BadStratQuadraticRecognitionArg	Bad StratQuadraticRecognition argument
NonlinOpt_BadStratPresolveArg	Bad StratPresolve argument
NonlinOpt_BadStratSelectiveConstraintArg	Bad StratSelectiveConstraint argument
NonlinOpt_BadStratSteepestEdgeArg	Bad StratSteepestEdge argument
NonlinOpt_BadStartingPointArg	Bad StartingPoint argument
NonlinOpt_BadOptimalityToleranceArg	Bad OptimalityTolerance argument
NonlinOpt_BadItLimitSlowProgArg	Bad ItLimitSlowProg argument
NonlinOpt_BadDerivativesArg	Bad Derivatives argument

Example:

If one wished to set all the nonlinear options, the simplest syntax would be:

```
wbSetNonlinearOptions True, True, True, True, True, True, False, True, False, 0.0001, 200, 1
```

To set one or more options, named arguments should be used:

```
wbSetNonlinearOptions OptimalityTolerance:=0.0001
```

wbSetStochasticOptions

This routine can be used to set the What's*Best!* stochastic solver options seen in the *Stochastic Solver* dialog box. All arguments are optional. For additional discussion of the options available through this routine, see the section entitled *Options…|Stochastic Solver*.

Syntax: wbSetStochasticOptions([StochasticSolverMethod], [StochasticSeed], [StochasticSamplSize], [StochasticSamplCont], [StochasticEVWS], [StochasticEVEM], [StochasticEVPI], [StochasticEVMU], [StochasticRepScenHoriz], [UsePreviewWindow], [LoadScenarioViewer])

Argument	*Required*	*Default*	*Description*
StochasticSolverMethod	No	0 (Solver Decides)	StochasticSolverMethod is an integer indicating the stochastic solver method to employ: 0: Solver Decides, 1: Free, 2: Deterministic Equivalent, 3: Nested Benders Decomposition, 4: Augmented Lagrangian Decomposition, 5: Simple Benders Decomposition.
StochasticSeed	No	1031	StochasticSeed is a cell reference or an integer greater than 1 indicating the seed value for random generator.
StochasticSamplSize	No	2	StochasticSamplSize is an integer greater than 1 indicating the default number of scenarios per stage.
StochasticSamplCont	No	True	StochasticSamplCont is a True/False flag indicating whether or not to use this option.
StochasticEVWS	No	True	StochasticEVWS is a True/False flag indicating whether or not to use this option.
StochasticEVEM	No	True	StochasticEVEM is a True/False flag indicating whether or not to use this option.
StochasticEVPI	No	True	StochasticEVPI is a True/False flag indicating whether or not to use this option.
StochasticEVMU	No	False	StochasticEVMU is a True/False flag indicating whether or not to use this option.

StochasticRepScenHoriz	No	False	StochasticRepScenHoriz is a True/False flag indicating whether or not to use this option.
UsePreviewWindow	No	False	*UsePreviewWindow* is a boolean indicating the display of the Preview Window for the Stochastic scenarii upon solving . The reporting cells may be defined with the function *wbStochasticReport.*
LoadScenarioViewer	No	False	*LoadScenarioViewer* is a boolean for loading the Scenario Window with the Stochastic scenarii. The Scenario Window is then selected via the menu.

Error Codes	***Description***
StoOpt_BadStochasticSolverMethodArg	Bad StochasticSolverMethod argument
StoOpt_BadStochasticSeedArg	Bad StochasticSeed argument
StoOpt_BadStochasticSamplContArg	Bad StochasticSamplCont argument
StoOpt_BadStochasticEVWSArg	Bad StochasticEVWS argument
StoOpt_BadStochasticEVEMArg	Bad StochasticEVEM argument
StoOpt_BadStochasticEVPIArg	Bad StochasticEVPI argument
StoOpt_BadStochasticEVMUArg	Bad StochasticEVMU argument
StoOpt_BadStochasticRepScenHorizArg	Bad StochasticRepScenHoriz argument
StoOpt_BadUsePreviewWindowArg	Bad StoOpt_BadUsePreviewWindowArg argument
StoOpt_BadLoadScenarioViewerArg	Bad StoOpt_BadLoadScenarioViewerArg argument

Example:

If one wished to set all the stochastic support options, the simplest syntax would be:

wbSetStochasticOptions 2, 2027, False, False, False, False, True, False, False, False

To set one or more options, named arguments should be used:

wbSetStochasticOptions StochasticEVPI:=False

Note: The *AdvStochasticSupport* argument should be set to TRUE via the *wbSetStochasticSupport* function in order to use these options.

wbSetStochasticSupport

This routine can be used to set the What's*Best!* stochastic support option seen in the Stochastic Support dialog box. There is only one argument. For additional discussion of the options available through this routine, see the section entitled Options...|Stochastic Solver.

Syntax: wbSetStochasticSupport(AdvStochasticSupport)

Argument	***Required***	***Description***
AdvStochasticSupport	Yes	AdvStochasticSupport is a True/False flag indicating whether or not to use this support. Default 0 (False).

Error Codes	**Description**
Sto_BadAdvStochasticSupportArg	Bad AdvStochasticSupport argument

Example:

If one wished to set all the function support options, the simplest syntax would be:

wbSetStochasticSupport True

or by the named argument:

wbSetStochasticSupport AdvStochasticSupport:=True

wbSetStringSupport

This routine can be used to set the What's*Best!* string support option seen in the *String Support* dialog box. The first argument is required, the others are optional. For additional discussion of the options available through this routine, see the section entitled *Advanced. . . |String Support*.

Syntax: wbSetStringSupport(AdvStringSupport, [AdvStringLength])

Argument	Required	Default	Description
AdvStringSupport	Yes	0 (False)	*AdvStringSupport* is a True/False flag indicating whether or not to use this support.
AdvStringLength	No	20	*AdvStringLength* is an integer between 1 and 255 indicating the maximum string length.

Error Codes	Description
AdvStr_BadAdvStringSupportArg	Bad *AdvStringSupport* argument
AdvStr_BadAdvStringLengthArg	Bad *AdvStringLength* argument

Examples:

If one wished to set all the string support options, the simplest syntax would be:

```
wbSetStringSupport True, 10
```

To set one or more options, named arguments should be used:

```
wbSetStringSupport AdvStringSupport:=True
```

Note: The *AdvStringSupport* argument should be set to TRUE in order to use the *AdvStringLength* argument.

wbSolve

For additional discussion of the functionality provided by this routine, see *Solve*.

This routine is used to solve the active model.

Syntax: wbSolve([SolutionStatus], [ExcelOpen], [NoErrDialog])

Argument	Required	Description
SolutionStatus	No	*SolutionStatus* is a return integer on the condition of the solution: Globally Optimal (1), Globally Optimal (2), Infeasible (3), Unbounded (4), Feasible (5), Infeasible or Unbounded (6), Near Optimal (7), Locally Optimal (8), Locally Infeasible (9), Cutoff (10), Numerical Error (11), Unknown (12), Unloaded (13), Loaded (14), Unknown Error (other), where 1000 is an interruption from a calculation error, and 2000 is an interruption by the user.

ExcelOpen	No	*ExcelOpen* is a True/False flag to indicate whether Excel® should remain in the open state or be minimized (True to leave Excel® open, False to minimize Excel®). If this argument is omitted, it defaults to the value of the Minimize Excel® on Solve setting in the General Options dialog box.
NoErrDialog	No	Any argument (e.g., an integer) passed causes all What's*Best!* error dialog boxes from the wbSolve routine to be suppressed. Instead, the error number is delivered in this return argument. Pass an integer to use any possible returned What's*Best!* error number. This is useful in an embedded application of What's*Best!*.

Error Codes	***Description***
Solve_UnmappedDrive	Error on drive
Solve_NoDLL	Error on DLL to access
Solve_BadSolutionStatusArg	Bad SolutionStatus argument
Solve_BadExcelOpenArg	Bad ExcelOpen argument
Solve_MultipleMaxMinError	More than one cell is specified as the objective cell
Solve_ProtectedWorkbookError	Protection can not be on the workbook, only on worksheets
Solve_ReadLINDOWBRCError	Error reading temporary file LINDOWBRC.TXT
Solve_SaveTempFileError	Error saving a copy of the active workbook
Solve_SymbolInSheetNameError	Error on sheet name
Solve_MinimumFileFormatError	The active workbook must be an Excel® 8.0 (BIFF8 or Excel® 97) format or higher
Solve_UnableToLoadSolverError	There was a problem loading the What's*Best!* solver
Solve_ErrorRunningSolver	There was a problem running the What's*Best!* solver

Return Value: The return value is the same of the Solution Status argument.

Example:

```
Sub JustSolve
  On Error goto errorhandler
  Dim lngSolutionStatus as Long
  'Solve the current model
  wbSolve lngSolutionStatus
  'Display the results
  Select Case lngSolutionStatus
    Case 1
      MsgBox "The model is: Globally Optimal"
```

```
        Case 2
          MsgBox "The model is: Globally Optimal, " &_
                  Range("WBMAX")
        Case 3
          MsgBox "The model is: Infeasible"
        Case 4
          MsgBox "The model is: Unbounded"
        Case 5
          MsgBox "The model is: Feasible"
        Case 6
          MsgBox "The model is: Infeasible or Unbounded"
        Case 7
          MsgBox "The model is: Near Optimal"
        Case 8
          MsgBox "The model is: Locally Optimal"
        Case 9
          MsgBox "The model is: Locally Infeasible"
        Case 10
          MsgBox "The model is: Cutoff"
        Case 11
          MsgBox "The model is: Numerical Error"
        Case 12
          MsgBox "The model is: Unknown"
        Case 13
          MsgBox "The model is: Unloaded"
        Case 14
          MsgBox "The model is: Loaded"
        Case Else
          MsgBox "Solution status unknown"
      End Select
      Exit Sub
    Errorhandler:
    Msgbox "Error number is " & Err & ", Description: "
       wbError(Err)
    End Sub
```

wbStochasticChanceConstrained

This routine can be used to set the stochastic support functions seen in the *Stochastic Support* dialog box, at the Chance-Constrained page. All arguments are required, except for the error code, but a 0 value can be placed for unnecessary arguments. For additional discussion of the options available through this routine, see the section entitled *Options...|Stochastic Solver*.

Syntax:

wbStochasticChanceConstrained(FunctionName, Percent, ObjWeight, Refers_to, Place_in_cell, [NoErrDialog])

Argument	*Required*	*Default*	*Description*
FunctionName	Yes	"WBSP_CC_LT"	FunctionName is a string argument indicating the name of the function to implement, either WBSP_CC_LT or WBSP_CC_GT.
Percent	Yes	0	Percent is an integer between 0 and 1 indicating the wether the probability of the set is satisfied.
ObjWeight	Yes	0	ObcjWeight is a number indicating the amont set on the objective value.
Refers_to	Yes	0	Refers_to is the range or address of the cell to copy the chance constraint function.
Place_in_cell	Yes	0	Place_in_cell is a cell reference to specify the cell in which to write the WBSP_ function.
NoErrDialog	No	0	Any argument passed here causes all What's*Best!* error dialog boxes from the wbStochasticChanceConstrained routine to be suppressed. Instead, the error number is delivered in this return argument. Pass an integer to use any possible returned What's*Best!* error number. Useful in an embedded application of What's*Best!*

Error Codes	*Description*
Sto_BadFunctionNameArg	Bad FunctionName argument
Sto_BadPercentArg	Bad Percent argument
Sto_BadObjWeightArg	Bad ObjWeight argument
Sto_BadRefersToArg	Bad RefersTo argument
Sto_BadPlaceInCellArg	Bad PlaceInCell argument

Examples:
To set a Chance-Constrained information to constraint cells, the simplest syntax would be:
wbStochasticChanceConstrained "WBSP_CC_LT", 0.8, 0, "A2:B3", "D4", 0

wbStochasticFunction

This routine can be used to set the stochastic support functions seen in the *Stochastic Support* dialog box. All arguments are required, except for the error code, but a 0 value can be placed for unnecessary arguments. For additional discussion of the options available through this routine, see the section entitled *Options...|Stochastic Solver*.

Syntax: wbStochasticFunction(FunctionName, Stage, CellRange1, CellRange2, CellRange3, Refers_to, Place_in_cell, [NoErrDialog])

Argument	***Required***	***Default***	***Description***
FunctionName	Yes	"WBSP_..."	FunctionName is a string argument indicating the name of the function to implement.
Stage	Yes	0	Stage is an integer between 1 and 255 indicating the stage the cell belongs to.
CellRange1	Yes	0	CellRange1 is a cell reference for indicating the value of the first argument; used for distributions.
CellRange2	Yes	0	CellRange2 is a cell reference for indicating the value of the first argument; used for distributions.
CellRange3	Yes	0	CellRange3 is a cell reference for indicating the value of the first argument; used for distributions.
Refers_to	Yes	0	Refers_to is the range or address of the cell to copy the stochastic function.
Place_in_cell	Yes	0	Place_in_cell is a cell reference to specify the cell in which to write the WBSP_ function.
NoErrDialog	No	0	Any argument passed here causes all What's*Best!* error dialog boxes from the wbStochasticFunction routine to be suppressed. Instead, the error

			number is delivered in this return argument. Pass an integer to use any possible returned What's*Best!* error number. Useful in an embedded application of What's*Best!*.

Error Codes	***Description***
Sto_BadFunctionNameArg	Bad FunctionName argument
Sto_BadStageArg	Bad Stage argument
Sto_BadCellRange1Arg	Bad CellRange1 argument
Sto_BadCellRange2Arg	Bad CellRange2 argument
Sto_BadCellRange3Arg	Bad CellRange3 argument
Sto_BadRefersToArg	Bad RefersTo argument
Sto_BadPlaceInCellArg	Bad PlaceInCell argument
Sto_BadCellRange1CellRange2Arg	Bad CellRange1CellRange2 argument
Sto_BadCellRange1CellRange2CellRange3Arg	Bad CellRange1CellRange2CellRange3 argument

Examples:

To set a stage information to a variable cell, the simplest syntax would be:

```
wbStochasticFunction "WBSP_VAR", 1, 0, 0, 0, "A2", "A3", 0
```

To set a distribution to a random cell:

```
wbStochasticFunction "WBSP_DIST_LOGNORMAL", 0, "C2", "D2", 0, "A2", "A3", 0
```

wbStochasticHistogram

This routine can be used to set the stochastic histogram seen in the Stochastic Support dialog box. All arguments are required, except for the error code. For additional discussion of the options available through this routine, see the section entitled *Options...|Stochastic Solver*.

Syntax: wbStochasticHistogram(NumberBins, Arglist, Place_in_cell, [NoErrDialog])

Argument	***Required***	***Default***	***Description***
NumberBins	Yes	0	*NumberBins* is an integer between 1 and 255 indicating the

			number of bins for the reporting cell.
ArgList	Yes	0	*ArgList* is a listing of cell references for indicating the cells to display in the stochastic report.
Place_in_cell	Yes	0	Place_in_cell is a cell reference to specify the cell in which to write the WBSP_HIST function.
NoErrDialog	No	0	Any argument passed here causes all What's*Best!* error dialog boxes from the wbStochasticHistogram routine to be suppressed. Instead, the error number is delivered in this return argument. Pass an integer to use any possible returned What's*Best!* error number. Useful in an embedded application of What's*Best!*.

Error Codes	***Description***
Sto_BadNumberBinsArg	Bad NumberBins argument
Sto_BadArgListArg	Bad ArgList argument
Sto_BadPlaceInCellArg	Bad PlaceInCell argument
Sto_ProtectedSheetRepError	Bad ProtectedSheetRep argument

Examples:

To set several reporting cells, so to display the histogram report, the simplest syntax would be:

```
wbStochasticHistogram 0, "Sheet1!$C$1"
```

wbStochasticReport

This routine can be used to set the stochastic reporting cells seen in the *Stochastic Support* dialog box. All arguments are required, except for the error code. For additional discussion of the options available through this routine, see the section entitled *Options...|Stochastic Solver*.

Syntax: wbStochasticReport(ArgList, Place_in_cell, [NoErrDialog])

Argument	*Required*	*Default*	*Description*
ArgList	Yes	0	ArgList is a listing of cell references for indicating the cells to display in the stochastic report.
Place_in_cell	Yes	0	Place_in_cell is a cell reference to specify the cell in which to write the WBSP_REP function.
NoErrDialog	No	0	Any argument passed here causes all What's*Best!* error dialog boxes from the wbStochasticFunction routine to be suppressed. Instead, the error number is delivered in this return argument. Pass an integer to use any possible returned What's*Best!* error number. Useful in an embedded application of What's*Best!*.

Error Codes	*Description*
Sto_BadArgListArg	Bad ArgList argument
Sto_BadPlaceInCellArg	Bad PlaceInCell argument
Sto_ProtectedSheetRepError	Bad ProtectedSheetRep argument

Examples:

To set several reporting cells, so to display the stochastic report, the simplest syntax would be:

```
wbStochasticReport "Sheet1!$B$1:$B$3,Sheet1!$C$1", "A1", 0
```

wbStochasticStageScenario

This routine can be used to set the stochastic stage/scenario table seen in the *Stochastic Support* dialog box. All arguments are required, except for the error code. For additional discussion of the options available through this routine, see the section entitled *Options...*|*Stochastic Solver*.

Syntax: wbStochasticStageScenario(CellRange1, Place_in_cell, RangeChoice, [NoErrDialog])

Argument	*Required*	*Default*	*Description*
CellRange1	Yes	0	CellRange1 is a cell reference for indicating the range of the table.
Place_in_cell	Yes	0	Place_in_cell is a cell reference to specify the cell in which to write the WBSP_STSC function.
RangeChoice	Yes	0	RangeChoice is a string "SET" to enter the function, or "NONE" to delete the selected cell.
NoErrDialog	No	0	Any argument passed here causes all What's*Best!* error dialog boxes from the wbStochasticFunction routine to be suppressed. Instead, the error number is delivered in this return argument. Pass an integer to use any possible returned What's*Best!* error number. Useful in an embedded application of What's*Best!*.

Error Codes	*Description*
Sto_BadCellRange1Arg	Bad CellRange1 argument
Sto_BadPlaceInCellArg	Bad PlaceInCell argument
Sto_BadRangeChoiceArg	Bad RangeChoice argument

Examples:

To set a Stage/Scenario table, the simplest syntax would be:

```
wbStochasticStageScenario "Sheet1!$B$1:$C$3", "D1", "SET", 0
```

wbUpdateLinks

For additional discussion of the functionality provided by this routine, see the description of *Update Links* in the section entitled *Options…|General.*

This routine is used to update the links to the WB constraints and functions.

Syntax: wbUpdateLinks()

The *wbUpdateLinks* procedure has no arguments.

Error Codes	***Description***
UpdateLinksProtectedError	Unable to update links on a protected sheet
UpdateLinksHiddenError	Unable to determine if there are constraints in hidden cells on protected sheets

Remarks: This procedure is provided because Excel® does not properly update links of add-in functions.

5 Functions and Operators

What's*Best!* supports many mathematical and logical functions of Excel® and also adds some functions of its own.

What'sBest! supports many, but not all, of the mathematical and logical functions and operators of Excel and also adds some functions of its own. In general, If for any cell, say X1,

+ X1 does not depend upon a WB Adjustable cell, or

+ X1 depends upon a WB Adjustable cell, but does not influence (or is not a precedent of) any WB constraint or Objective cell and X1 is declared to be in a WBOMIT range (see *Advanced|Omit..*.),

then any Excel® function can be used in X1. Otherwise, the user is restricted to a subset of Excel functions as enumerated below.

Supported Functions and Operators

Functions and Operators in Excel Recognized by What'sBest!

The following Excel functions and operators are supported by What'sBest! with limitations as announced in notes 1 and 2:

ABS*(note 1)*	CHIINV	INT*(note 1)*	NORMSDIST	SIGN*(note 1)*
ACOS	COS	LN	NORMSINV	SIN
ACOSH	COSH	LOG*(note2)*	NOT*(note1)*	SINH
AND*(note1)*	EXP	LOGINV	NPV	SQRT
ASIN	EXPONDIST	LOGNORMDIST	OR*(note1)*	STDEV
ASINH	FALSE	MAX*(note1)*	PI	SUM
ATAN	FDIST	MIN*(note1)*	POISSON	TAN
ATAN2	FINV	MOD*(note1)*	PRODUCT	TANH
ATANH	GAMMADIST	MMULT	PV	TRANSPOSE
AVERAGE	GAMMAINV	NEGBINOMDIST	ROUND*(note1)*	TRUE
BINOMDIST	HYPGEOMDIST	NORMDIST	ROUNDDOWN*(note1)*	TRUNC*(note1)*
CHIDIST	IF*(note1)*	NORMINV	ROUNDUP	

^	+
/	<=*(note1)*
=*(note1)*	>=*(note1)*
*	-
<*(note1)*	<>
>*(note1)*	%

The following Excel functions are supported only under certain conditions:

- COUNT(*note 3*)
- HLOOKUP
- INDEX(*note 3*)
- MATCH(*note 3*)
- SUMIF
- SUMIFS
- SUMPRODUCT
- VLOOKUP

Functions Added by What'sBest!

The WBxxx functions are the ones added by What'sBest!. Note that many of these functions cannot be used in an expression, see note 3. They can be used only by themselves in a cell.

- WBALLDIFF*(note3)*
- WBCARD*(note3)*
- WBDUAL(note3)
- WBERLANG_B
- WBERLANG_C
- WBEXPOINV
- WBINNERPRODUCT*(note3)*
- WBLMTD*(note3)*
- WBLOGABEXPX
- WBLOGISTIC
- WBLOGISTICDENS
- WBLOGISTICINV
- WBLOGSUMAEXP
- WBLOGSUMEXP
- WBLOWER*(note3)*
- WBMULTINV
- WBNORMSL
- WBPOSD*(note3)*
- WBQUADPRODUCT*(note3)*
- WBRLMTD*(note3)*
- WBSEMIC*(note3)*
- WBSIGNPOWER*(note3)*

- WBSOS1*(note3)*
- WBSOS2*(note3)*
- WBSOS3*(note3)*
- WBTRIAINV
- WBUNIFINV
- WBUPPER*(note3)*
- WBXPOWDIVAB

Note 1: The function is non-smooth. Its use may result in long solution times, and if the Global Solver is not used, in non-optimal solutions. See the discussion of *Smoothness* in *Overview of Mathematical Modeling*.

Note 2: *LOG* can support one argument assuming base 10, or two arguments.

Note 3: The function can be used only standalone in a cell, not as part of a larger expression. For example, =WBLMTD(X1,X2) is OK, but =X3*WBLMTD(X1,X2)^2 is not.

Note 4: All arguments must evaluate to being numeric, the second argument must be in ascending order, the third argument refers to numeric constant, and the fourth argument is an optional TRUE/FALSE value.

Distributions/Correlations for Stochastic Programming

Refer to the section *Options...|Stochastic Solver* for additional explanation on using these functions.

WBSP_CORR_KENDALL	WBSP_DIST_DISCRETE_SV_W	WBSP_DIST_NEGATIVEBINOMIAL
WBSP_CORR_PEARSON	WBSP_DIST_EXPONENTIAL	WBSP_DIST_NORMAL
WBSP_CORR_SPEARMAN	WBSP_DIST_F_DISTRIBUTION	WBSP_DIST_PARETO
WBSP_DIST_BETA	WBSP_DIST_GAMMA	WBSP_DIST_POISSON
WBSP_DIST_BETABINOMIAL	WBSP_DIST_GUMBEL	WBSP_DIST_STUDENTS_T
WBSP_DIST_BINOMIAL	WBSP_DIST_GEOMETRIC	WBSP_DIST_SYMMETRICSTABLE
WBSP_DIST_CAUCHY	WBSP_DIST_HYPERGEOMETRIC	WBSP_DIST_TRIANGULAR
WBSP_DIST_CHISQUARE	WBSP_DIST_LAPLACE	WBSP_DIST_UNIFORM
WBSP_DIST_DISCRETE_SH	WBSP_DIST_LOGARITHMIC	WBSP_DIST_WEIBULL
WBSP_DIST_DISCRETE_SV	WBSP_DIST_LOGISTIC	WBSP_MEAN
WBSP_DIST_DISCRETE_SH_W	WBSP_DIST_LOGNORMAL	

Linearization

Many of the non-smooth functions and operators supported by What's*Best!* may be automatically smoothed by the solver through a process referred to as *linearization*. Linearization replaces a non-smooth function or operator with a collection of additional linear variables and constraints such that the modified model is mathematically equivalent to the original with the non-smooth functions or operators eliminated. Non-smooth functions and operators that may be eliminated through linearization are:

Functions	Operators
ABS	<
AND	<=
HLOOKUP	<>
IF	=
INT	>
MAX	>=
MIN	
NOT	
OR	
SIGN	
SUMIF	
VLOOKUP	

In addition to the above, What's*Best!* can also linearize products of 0/1 and continuous variables.

For more information on the linearization process, refer to the section entitled *Options...|General*.

Global Solver

Functions currently supported are:

1) Standard smooth functions: x+y, x-y, x*y, log(x), exp(x), sqrt(x), sin(x), cos(x)
2) Continuous, non-smooth: abs(x), max(x,y), min(x,y)
3) Smooth, not quite continuous: x/y, x^y, tan(x), floor(x)
4) Logical and relational operators: if (x,y,z),>=, <=, ==, !=, .AND. , .OR., .NOT.
5) Probability distributions: psn(z) [Normal Distribution], psl(z) [Normal Linear Loss]

Range Name Usage

With regard to the usage of Excel Range names, the following rules apply to example cell X1: If a range name refers to a range containing just one cell, then you may use that range name in the formula just like a single cell reference (You might as well, however, just use the cell name itself, e.g. X2). If the range name refers to a range with one or more cells, then that range name can be used in all the What'sBest! recognized functions and operators described before, e.g., in SUM, SUMIF, SUMPRODUCT, WBINNERPRODUCT, etc.

Functions Descriptions

WBALLDIFF

The WBALLDIFF function is employed for modeling the All Different variables. These are the integer variables restricted to get all different values within some range. The syntax is:

WBALLDIFF(LowerBound, UpperBound, Refers_to)

Argument	Description
LowerBound	the number indicating the lower bound in the range
UpperBound	the number indicating the upper bound in the range
Refers_to	the range or address of the cell(s) to be specified as ALLDIFF

For additional discussion and a sample case for the WBALLDIFF function, see the section entitled Integers|All-different, which refers to the dialog box interface that calls this function.

WBCARD

The WBCARD function is employed for modeling a set of variables with a cardinality of N, i.e., at most N variables belonging to this set are able to get non-zero values. The syntax is:

WBCARD(CardNumber, Refers_to)

Argument	Description
CardNumber	the strictly positive integer indicating the cardinality

Refers_to	the range or address of the cell(s) belonging to the set

A sample case is provided with the file "WBCARD-PRODMIX.xlsx". This model is a variant of the product mix problem in which we are only allowed to produce at most two products of the first three ones and also the same for the next three products. These cardinality constraints are modeled employing the WBCARD function.
For additional discussion of the WBCARD function, see the section entitled *Integers|Special Ordered Set*, which refers to the dialog box interface that calls this function.

WBDUAL

The WBDUAL function is employed for returning the dual values of the adjustable or constraint cells. The syntax is:

WBDUAL(Refers_to, value)

Argument	Description
Refers_to	the reference to the adjustable or constraint cell you want the dual value on
value	an arbitrary number (What'sBest! replaces this value with the actual dual value after solving the model)

For additional discussion of the WBDUAL function, see the section entitled *Advanced|Dual...*, which refers to the dialog box interface that calls this function. Furthermore, a comprehensive sample case for this function is provided in Usage Guidelines for Dual Values.

WBERLANG_B

The WBERLANG_B function returns the Erlang's loss probability for a service system with X servers and an arriving load of A, no queue allowed. The result of WBERLANG_B(A, X) can be interpreted as either the fraction of time all servers are busy or the fraction of customers lost due to all servers being busy when they arrive. The arriving load A is the expected number of customers arriving per unit of time multiplied by the expected time to process one customer. The syntax is:

WBERLANG_B(Load, Num_Servers)

Argument	Description
Load	the non-negative real number indicating the arriving load of the system

Num_Servers	the strictly positive integer indicating the number of servers

A sample case is provided with the file "WBERLANGB-Queue.xlsx". It models an M/M/c/c system for which we aim to minimize the number of required servers, while the fraction of lost customers is required to be less than a given value.

WBERLANG_C

The WBERLANG_C function returns the Erlang's busy probability for a service system with X servers and an arriving load of A, with infinite queue allowed. The result of WBERLANG_C(A, X) can be interpreted as either the fraction of time all servers are busy or the fraction of customers that must wait in the queue. The arriving load A is the expected number of customers arriving per unit of time multiplied by the expected time to process one customer. The syntax is:

WBERLANG_C(Load, Num_Servers)

Argument	**Description**
Load	the non-negative real number indicating the arriving load of the system
Num_Servers	the strictly positive integer indicating the number of servers

A sample case is provided with the file "QueueNservers.xlsx". It models an M/M/c service system for which we aim to minimize the total cost including servers costs and the expected customers waiting costs.

WBEXPOINV

The WBEXPOINV function returns the inverse of an exponential cumulative distribution for a supplied mean and standard deviation. For example, *WBEXPOINV(.5,1)* returns 0.69315. This means that for an exponential distribution with mean 1.0, 50 percent of the outcomes are less-than-or-equal-to 0.69315. The syntax is:

WBEXPOINV(Prob, Mean)

Argument	**Description**
Prob	the probability (greater-than-or-equal-to zero and less-than-or-equal-to 1) at which you want to evaluate the inverse of the exponential cumulative distribution

Mean	the non-negative real number indicating the arithmetic mean of the distribution

WBINNERPRODUCT

The WBINNERPRODUCT function is the equivalent of SUMPRODUCT, but the terms are multiplied row by column and then summed instead of multiplying them row by row. The syntax is:

WBINNERPRODUCT(Range1, Range2)

Argument	Description
Range1	the cell range containing the column or row (*Range1* can only be one dimensional and it must be the same size and shape as *Range2*)
Range2	the cell range containing the column or row (*Range2* can only be one dimensional and it must be the same size and shape as *Range1*)

A sample case is provided with the file "WBINNERPRODUCT-PRODMIX.xlsx". This is a variant of the product mix problem in which the WBINNERPRODUCT function is employed instead of SUMPRODUCT.

WBLMTD

The WBLMTD function represents the log mean temperature difference function. The amount of heat exchanged between the hot stream and the cold stream in a heat exchanger is proportional to WBLMTD(d1, d2) = (d1-d2)/Log(d1/d2). The syntax is:

WBLMTD(d1, d2)

Argument	Description
d1	the number indicating the temperature difference of the two streams at one end
d2	the number indicating the temperature difference of the two streams at the other end

A sample case for this function is provided with the file "HeatXchange_WBLMTD.xlsx".

WBLOGABEXPX

The WBLOGABEXPX function is employed for computing the composite function WBLOGABEXPX(x, a, b) = ln(a + b*exp(x)), where x, a and b represent real values. For example, *WBLOGABEXPX(2, 3, 4)* returns 3.482969. The syntax is:

WBLOGABEXPX(x, a, b)

Argument	Description
x	a real number
a	a real number
b	a real number

WBLOGISTIC

The WBLOGISTIC function returns the cumulative distribution function for a Logistic distribution. The result of WBLOGISTIC(A, B, C) shows the probability that an observation from the Logistic distribution with the location and scale parameters A and B is less than or equal to C. For example, *WBLOGISTIC(5,2,5)* returns 0.5. The syntax is:

WBLOGISTIC(A, B, C)

Argument	Description
A	the number indicating the location parameter
B	the number indicating the scale parameter
C	the number at which you want to evaluate the cumulative distribution function

WBLOGISTICDENS

The WBLOGISTICDENS function returns the probability density function for a Logistic distribution. The result of WBLOGISTICDENS(A, B, C) shows the probability density at C for the Logistic distribution with the location and scale parameters A and B, respectively. For example, *WBLOGISTICDENS(5,2,3)* returns 0.098306. The syntax is:

WBLOGISTICDENS(A, B, C)

Argument	Description
A	the number indicating the location parameter
B	the number indicating the scale parameter
C	the number at which you want to evaluate the probability density

WBLOGISTICINV

The WBLOGISTICINV function returns the inverse of a Logistic cumulative distribution. The result of WBLOGISTICINV(A, B, C) shows the inverse of the Logistic cumulative distribution with the location and scale parameters A and B at the probability C. For example, *WBLOGISTICINV(5,2,0.5)* returns 5. This means that for a Logistic distribution with the location and scale parameters 5 and 2, 50 percent of the observations are less-than-or-equal-to 5. The syntax is:

WBLOGISTICINV(A, B, C)

Argument	Description
A	the number indicating the location parameter
B	the number indicating the scale parameter
C	the probability (greater-than-or-equal-to zero and less-than-or-equal-to 1) at which you want to evaluate the inverse of the Logistic cumulative distribution

WBLOGSUMAEXP

The WBLOGSUMAEXP function is employed for computing the composite function

$$\text{WBLOGSUMAEXP}(x) = \ln\left(\sum_{i=1}^{n} a_i \, exp(x_i)\right),$$

where a is a non-negative vector of size n and x represents a vector of size n. The syntax is:

WBLOGSUMAEXP(a, x)

Argument	Description
a	a cell range containing a column or row of size n (all elements should be non-negative constants)
x	a cell range containing a column or row of size n

WBLOGSUMEXP

The WBLOGSUMEXP function is employed for computing the composite function

$$\text{WBLOGSUMEXP}(x) = \ln\left(\sum_{i=1}^{n} exp(x_i)\right),$$

where x represents a vector of size n. The syntax is:

WBLOGSUMEXP(x)

Argument	Description
x	a cell range containing a column or row of size n

WBLOWER

The WBLOWER function is employed for returning the lower bound of the range over which the dual value of the adjustable or constraint cell does not change. The syntax is:

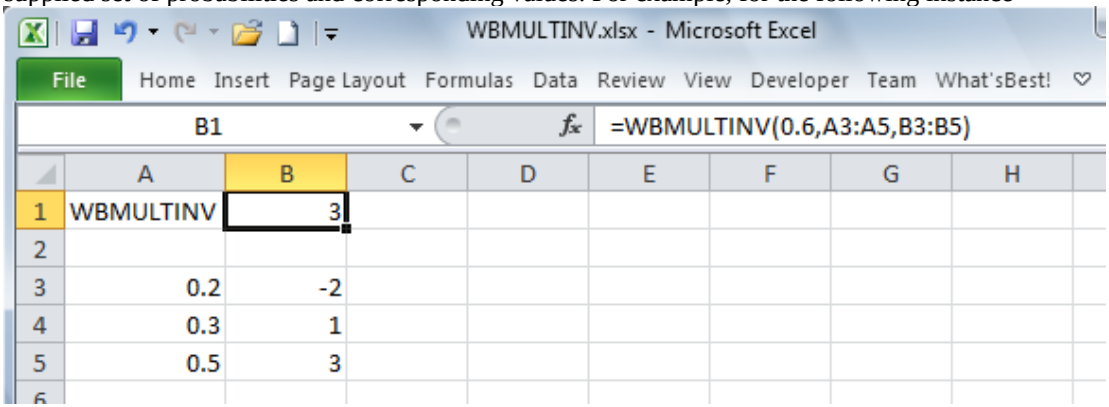
WBLOWER(Refers_to, value)

Argument	Description
Refers_to	the reference to the adjustable or constraint cell you want the lower range for its dual value
value	an arbitrary number (What'sBest! replaces this value with the actual lower range after solving the model)

For additional discussion of the WBLOWER function, see the section entitled *Advanced|Dual...*, which refers to the dialog box interface that calls this function. Furthermore, a comprehensive sample case for this function is provided in Usage Guidelines for Dual Values.

WBMULTINV

The WBMULTINV function returns the inverse of a multinomial cumulative distribution for a supplied set of probabilities and corresponding values. For example, for the following instance



, *WBMULTINV(0.6,A3:A5,B3:B5)* returns 3. This means that the minimum observation of X, such that 60 percent of the observations are less-than-or-equal-to X, is X=3. The syntax is:

WBMULTINV(Prob, ProbRange, ValueRange)

Argument	**Description**
Prob	the probability (greater-than-or-equal-to zero and less-than-or-equal-to 1) at which you want to evaluate the inverse of the multinomial cumulative distribution
ProbRange	the cell range containing the probabilities (*ProbRange* must be a one-dimensional range (i.e., a range of cells in either a single row or a single column))
ValueRange	the cell range containing the values that correspond with each probability (*ValueRange* must be the same size and shape as the *ProbRang)*

WBNORMSL

The WBNORMSL function represents the standard normal linear loss function which returns the expected value of max {0, Z-X}, where Z is a standard normal random variable. In inventory modeling, WBNORMSL(X) is the expected amount that demand exceeds a level X, if demand has a normal distribution. For example, *WBNORMSL(2)* returns 0.008491. The syntax is:

WBNORMSL(Value)

Argument	Description
Value	A real number

WBPOSD

The WBPOSD function is employed for the models in which at least one matrix is restricted to be positive semi-definite. The syntax is:

WBPOSD(Range)

Argument	Description
Range	the range of cells indicating the positive semi-definite matrix (Range must be symmetric with the adjustable cells typically set to be free type (although not required), and more than one cell)

For additional discussion and a sample case for the WBPOSD function, see the section entitled *Constraints...,* which refers to the dialog box interface that calls this function.

WBQUADPRODUCT

The WBQUADPRODUCT function is employed for computing

$$\text{WBQUADPRODUCT}(x,c,Q) = x^T Q x + c^T x,$$

where x and c are vectors of size n and Q represents an n by n matrix. It returns the scalar

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_i\, x_j Q_{ij} + \sum_{i=1}^{n} c_i\, x_i$$

The syntax is:

WBQUADPRODUCT(xRange, linRange, qRange)

Argument	Description
xRange	a cell range containing a column or row of size n
linRange	a cell range containing a column or row of size n
qRange	a cell range containing a matrix of size n by n

A sample case for this function is provided with the file "PortCorrVG_wbquadproduct.xlsx".

WBRLMTD

The WBRLMTD function returns the reciprocal of the WBLMTD function, i.e., WBRLMTD(d1, d2) = 1/WBLMTD(d1, d2). The syntax is:

WBRLMTD(d1, d2)

Argument	Description
d1	the number indicating the temperature difference of the two streams at one end
d2	the number indicating the temperature difference of the two streams at the other end

WBSEMIC

The WBSEMIC function is employed for modeling the semi-continuous variables. These are the variables restricted to either be 0 or lie within some nonnegative range, e.g., 10 to 20. The syntax is:

WBSEMIC(LowerBound, UpperBound, Refers_to)

Argument	Description
LowerBound	the number indicating the lower bound in the range

UpperBound	the number indicating the upper bound in the range
Refers_to	the range or address of the cell(s) to be specified as semi-continuous

A sample case is provided with the file "AstroCosSemic.xls". It models a production planning problem in which one of the products can be either produced within the company in a minimum batch size or outsourced. This constraint is modeled using the WBSEMIC function.
For additional discussion of the WBSEMIC function, see the section entitled Integers|Semi-continuous, which refers to the dialog box interface that calls this function.

WBSIGNPOWER

The function WBSIGNPOWER is employed for computing pressure differential in a network flow model. In these models, the pressure differential between two nodes as a function of the flow, f, between them, and a resistance type parameter p is related to sign(f)*abs(f)^p. For electrical networks, p = 1, and for most other fluids, 1.8 <= p <= 2. The function SIGNPOWER(f, p) returns sign(f)*abs(f)^p. The syntax is:

WBSIGNPOWER(F,p)

Argument	**Description**
F	the number indicating the flow
p	the number indicating the resistance type parameter

A sample case for this function is provided with the file "NetEqlxSignPower.xlsx".

WBSOS1

The function WBSOS1 is employed for modeling the Special Ordered Sets (SOS) of type 1. At most one of the variables belonging to an SOS1 set can be > 0. The syntax is:

WBSOS1(Refers_to)

Argument	**Description**
Refers_to	the range or address of the cell(s) belonging to the SOS1 set

For additional discussion of the WBSOS1 function, see the section entitled *Integers|Special Ordered Set*, which refers to the dialog box interface that calls this function.

WBSOS2

The function WBSOS2 is employed for modeling the Special Ordered Sets (SOS) of type 2. At most two of the variables belonging to an SOS2 set can be > 0 and these variables should be adjacent. In general, the SOS2 sets are applied to model the piecewise linear functions. The syntax is:

WBSOS2(Refers_to)

Argument	Description
Refers_to	the range or address of the cell(s) belonging to the SOS2 set

A sample case is provided with the file "PieceWiseSOS2a.xlsx". It models a problem in which we need to buy a specified amount of commodity. Two vendors each offer their own price curve (piecewise linear) and we should decide about the amount of buying from each to minimize the total cost. The WBSOS2 function is exploited to represent piecewise linear cost curves.
For additional discussion of the WBSOS2 function, see the section entitled *Integers|Special Ordered Set*, which refers to the dialog box interface that calls this function.

WBSOS3

The function WBSOS3 is employed for modeling the Special Ordered Sets (SOS) of type 3. Exactly one of the variables belonging to an SOS3 set gets the value of 1 and the other variables are restricted to be 0. The syntax is:

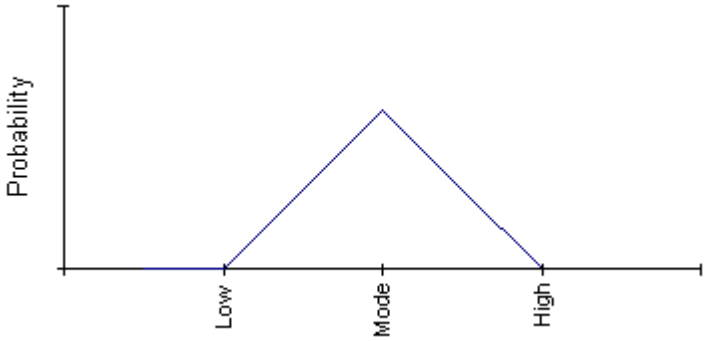
WBSOS3(Refers_to)

Argument	Description
Refers_to	the range or address of the cell(s) belonging to the SOS3 set

A sample case is provided with the file "WBSOS3-assignment.xlsx". This is an assignment problem in which we aim to allocate the jobs to machines such that the total required time for processing the jobs to be minimized. The WBSOS3 function is exploited for modeling the requirements of allocation of only one machine to each job and vice versa.
For additional discussion of the WBSOS3 function, see the section entitled *Integers|Special Ordered Set*, which refers to the dialog box interface that calls this function.

WBTRIAINV

The WBTRIAINV function returns the inverse of a triangular cumulative distribution for supplied low, mode, and high values. For example, *WBTRIAINV(1, 3, 2, 0.7)* returns 2.2254. This means that for a triangular distribution over the range [1,3] with mode 2, 70 percent of the observations are less-than-or-equal-to 2.2254.



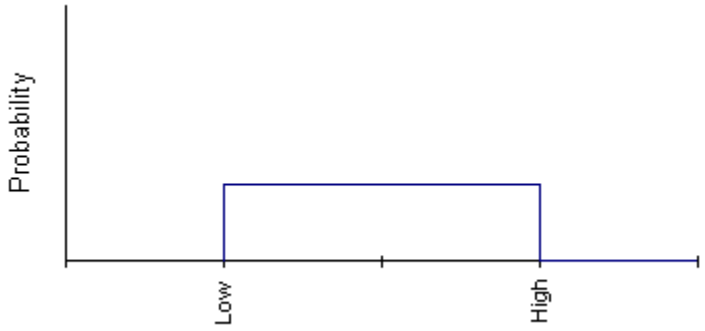
The syntax is:

WBTRIAINV(Low, High, Mode, Prob)

Argument	Description
Low	the number indicating the lower limit of the distribution
High	the number indicating the upper limit of the distribution (*High* must be greater-than-or-equal-to *Mode)*
Mode	the number indicating the mode (peak value) of the distribution (*Mode* must be between the *Low* and *High* values)
Prob	the probability (greater-than-or-equal-to zero and less-than-or-equal-to 1) at which you want to evaluate the inverse of the triangular cumulative distribution

WBUNIFINV

The WBUNIFINV function returns the inverse of a uniform cumulative distribution for supplied low and high values. For example, *WBUNIFINV(.4,1,2)* returns 1.4. This means that for a uniform distribution on the interval [1,2], 40 percent of the observations are less-than-or-equal-to 1.4.



The syntax is:

WBUNIFINV(Prob, Low, High)

Argument	Description
Prob	the probability (greater-than-or-equal-to zero and less-than-or-equal-to 1) at which you want to evaluate the inverse of the uniform cumulative distribution
Low	the number indicating the lower limit of the distribution
High	the number indicating the upper limit of the distribution (*High* must be greater-than-or-equal-to Low*)*

WBUPPER

The WBUPPER function is employed for returning the upper bound of the range over which the dual value of the adjustable or constraint cell does not change. The syntax is:

WBUPPER(Refers_to, value)

Argument	Description
Refers_to	the reference to the adjustable or constraint cell you want the lower range for its dual value
value	an arbitrary number (What'sBest! replaces this value with the actual upper range after solving the model)

For additional discussion of the WBUPPER function, see the section entitled *Advanced|Dual...*, which refers to the dialog box interface that calls this function. Furthermore, a comprehensive sample case for this function is provided in Usage Guidelines for Dual Values.

WBXPOWDIVAB

The WBXPOWDIVAB function is employed for computing the function WBXPOWDIVAB(x, a, b) = x^a/(b-x), where x, a and b represent real values and 0 <= x < b. For example, *WBXPOWDIVAB(2, 3, 6)* returns *2*. The syntax is:

WBXPOWDIVAB(x, a, b)

Argument	Description
x	a non-negative real number
a	a real number
b	a non-negative real number such that x < b

6 Overview of Mathematical Modeling

Introduction

The relationships in your model influence the computation time, the solution methods used by What's*Best!*, and the type of solution returned. This overview very briefly explains some of the different types of expressions entailed and explores how the expression type affects the solution search. An understanding of these basic principles is not required to use What's*Best!*, but it can go a long way in helping you to use the software more effectively.

In the section entitled *Linear vs. Nonlinear Expressions and Linearization*, we consider the two expression types, linear and nonlinear, and how they influence the solution process. The section entitled The Solution Process: Determining optima introduces some of the issues in finding optima. These issues include local versus global optima and the distinction between smooth (continuous) and non-smooth (non-continuous) functions. The type of outcomes generated by What's*Best!* is discussed in the section entitled *Solution Outcomes*. Finally, some suggestions for modeling are offered in the last section *Guidelines for Modeling with What'sBest!* and *Guidelines for Stochastic Modeling*.

Linear vs. Nonlinear Expressions and Linearization

Mathematical expressions can be classified according to their characteristics. The broadest and most common distinction is between linear and nonlinear expressions. What's*Best!* analyzes your model and describes it as linear or nonlinear in the status report under *Model Type*.

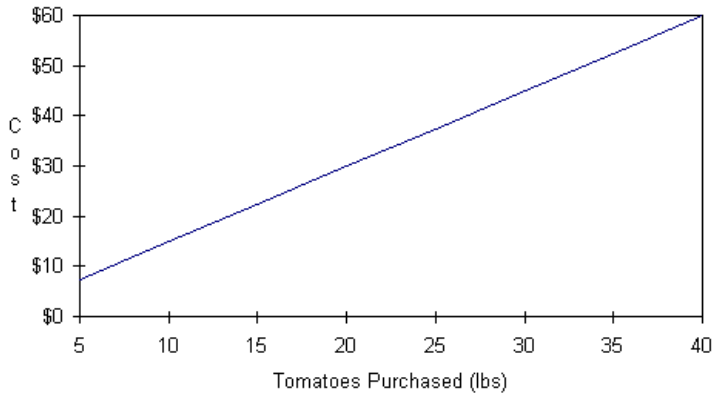
Linear Expressions

If all the terms of an equation are of the first order, the expression is said to be linear. This means the expression doesn't contain a variable squared, cubed, or taken to any power other than one, a term divided by a variable, or variables multiplied by each other. In other words, proportionality exists. That is, for every unit increase or decrease in a variable, the expression increases or decreases by a fixed amount.

In their simplest form, linear formulas are “straight line" relationships. For example, suppose you're buying tomatoes for $1.50 per pound. The expression or function used to calculate the *Cost* (*C*) in terms of the amount of tomatoes purchased (*T*) is:

$$C = 1.5 * T$$

As you might expect, a graph of this expression for cost is a straight line:



Linear expressions can have multiple variables. For example, if you added potatoes (*P*) at $0.75 per pound and apples (*A*) at $1.25 per pound, your cost function would become:

$$C = 1.5 * T + 0.75 * P + 1.25 * A$$

This new cost expression is linear — you could think of it as the sum of three simpler linear expressions.

Nonlinear Expressions

By definition, all expressions that are not linear are nonlinear. Nonlinear expressions include relationships in which variables are squared, cubed, or taken to powers other than one, variables that are multiplied by each other, and many expressions using nonlinear spreadsheet functions such as *IF*, *MAX*, and *MIN*.

Models with nonlinear expressions are intrinsically much more difficult to solve than linear models. Unlike linear models, a badly-formulated nonlinear model may prevent What's*Best!* from finding a solution, even though one in fact exists. Another problem posed by some nonlinear models is that What's*Best!* may detect a locally-optimal solution to the model that appears to be the "best", whereas a better, but also locally-optimal, solution still exists. For more on what you can do to help minimize the occurrence of these undesirable results, see the section titled *Guidelines for Modeling with* What's*Best!*.

Linearization

The presence of any nonlinear expressions in the model will cause the nonlinear solver to be invoked and the model to be classified as nonlinear. You can determine whether your model is linear or nonlinear by the classification under *Model Type* in the solver status window or the status report. The nonlinear models take much longer to solve than a linear model with the same number of constraints. The process of converting a nonlinear expression to a linear expression is called *linearization*. What's*Best!* can perform different degrees of linearization in the pre-processing stage of the solution

process on the *ABS*, *IF*, *MAX*, and *MIN* nonlinear spreadsheet functions, as well as all logical operators. See Chapter 6, *Functions and Operators*, for a comprehensive list.

Ideally, all nonlinear expressions would be converted to linear equivalents, so the linear solver can be used instead of the nonlinear solver. For an example of a simple nonlinear-to-linear conversion, consider the following equation:

$$X / Y = 10$$

As written, this equation is nonlinear, because of the division by *Y*. By simply multiplying both sides of the equation through by *Y*, this can be converted to the equivalent linear equation:

$$X = 10 * Y$$

By utilizing as many nonlinear-to-linear conversions as possible, the nonlinear model might possibly be reduced to a linear model. When a nonlinear model can be fully linearized, the payoff can sometimes be enormous (e.g., finding a solution where none could be found before or reducing solution time from days to seconds). The sample model *Linearization Option and Construction Cost Estimation* demonstrates the benefits of successful linearization.

The What's*Best!* linearization option cannot reduce all nonlinear models to linear models, nor does the effort necessarily improve the solution time. In some exceptional cases, it may be better to run What's*Best!* with linearization disabled. For this reason, the linearization option is provided with some user-adjustable parameters that are set in the *General Options* dialog box. You may wish to try different degrees of linearization to see what works best for your model.

It is highly recommended that you consider performing linearization on your own. If What's*Best!* indicates that your model is nonlinear, it can be beneficial to investigate the expressions in the nonlinear cells and determine whether any could be reformulated in a linear manner.

Note: The locations of all nonlinear cells will be displayed in the staus report, assuming that the *Nonlinearity present* warning has been enabled in the *General Options* dialog box.

The Solution Process: Determining Optima

Local Optima vs. Global Optima

Linear and nonlinear problems are distinguishable by their different kinds of solutions. When the What's*Best!* linear solver finds a solution to a linear optimization model, that solution is a best solution. Such a solution is called a *global optimum* because there is no better feasible solution.

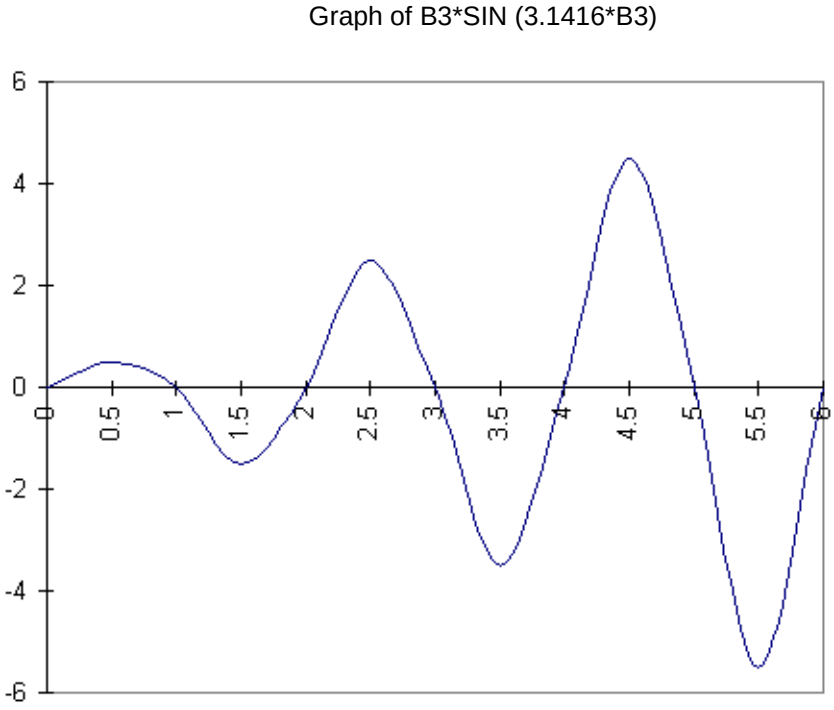
By contrast, conventional nonlinear solvers are only able to identify *local optima* (i.e., solutions for which no better feasible solutions can be found nearby). What's*Best!* tells you what type of solution you have reached by returning either *Globally Optimal* or *Locally Optimal* in the *Solution Status* field of the status report. If a model is nonlinear and the global solver is not used, then the status will always be locally optimal if a solution is found.

It is a property of nonlinear optimization models that they may have several solutions that are locally optimal. Such locally optimal solutions to nonlinear optimization models cannot be assumed to be

globally optimal. The following example illustrates this property and explains how you might explore the various local optima.

Consider the following model. Cell B3 has been specified as an adjustable cell, B4 has been specified to be minimized and contains the equation B3**SIN*(3.1416*B3), and B5 contains the constraint B4 <= 6.

The following graph shows a plot of B4, the expression to be minimized, for values of B3 between 0 and 6. You can see that, if you're searching for a minimum, there are local optima at B3 values of 0, 1.564, 3.529, and 5.518 — in the "valleys". If you use the global solver, it will find a global optimum for this problem at B3 = 5.518, since that is the last valley before 6.



Imagine the graph as a series of hills. You're searching in the dark for the minimum or lowest elevation. If you start your search at B3 = 3, every step to the left takes you uphill and every step right takes you downhill. Therefore, you move to the right in your search for the lowest point. You'll continue to move right as long as this direction leads to lower ground.

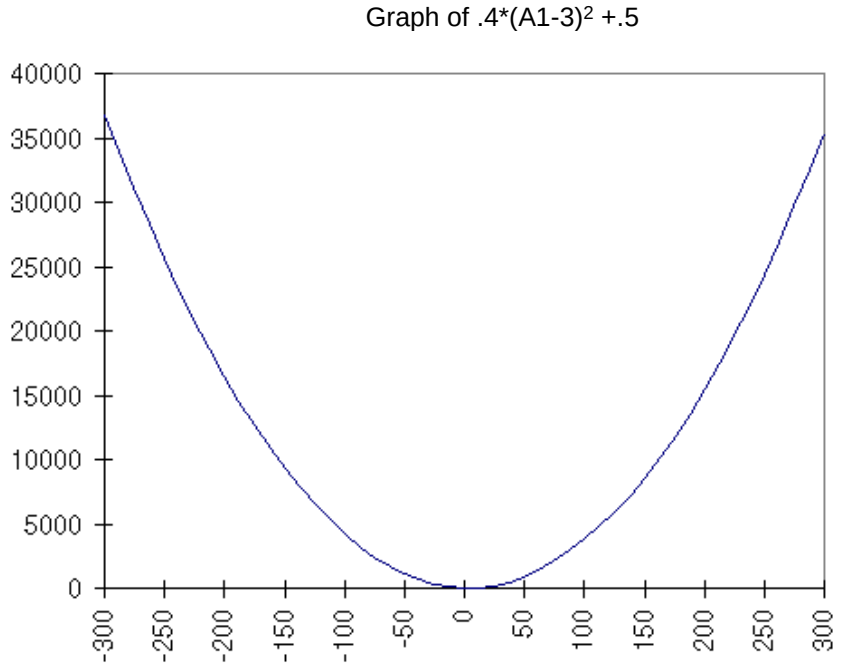
When you reach B3 = 3.529, you will notice a small flat area (slope is equal to zero). Continuing to the right begins to lead uphill and retreating to the left leads up the hill you just came down. You are in a valley, the lowest point in the immediate neighborhood (a local minimum), but is it the lowest possible point? You can't determine that given the information available to you at that particular point.

This is an extreme example and not typical of well-formulated nonlinear problems. Nonlinear solvers do their best to tackle such nonlinear models by seeking a local optimum from the starting points supplied by the user. This provides the user with the opportunity to explore the available local optima.

What's*Best!* takes the initial values of the adjustable cells as starting points in its search for a locally optimum solution. Therefore, if you solve this model with different starting values for B3, and do not use the Global Solver, then What's*Best!* may return a different local minimum. Attempting to solve the model with a starting value between 5 and 6 for B3 is likely to lead to a local minimum at 5.518, which happens to be the global optimum. You may try other starting values to find other local optima. For some problems, you may find that observing the results of solving the model several times with different initial values can help you find the best solution.

Convexity

If a function is convex, then it has a single global optimum. If a function is not convex, it may have multiple local optima. The following example shows a convex function of a single variable:

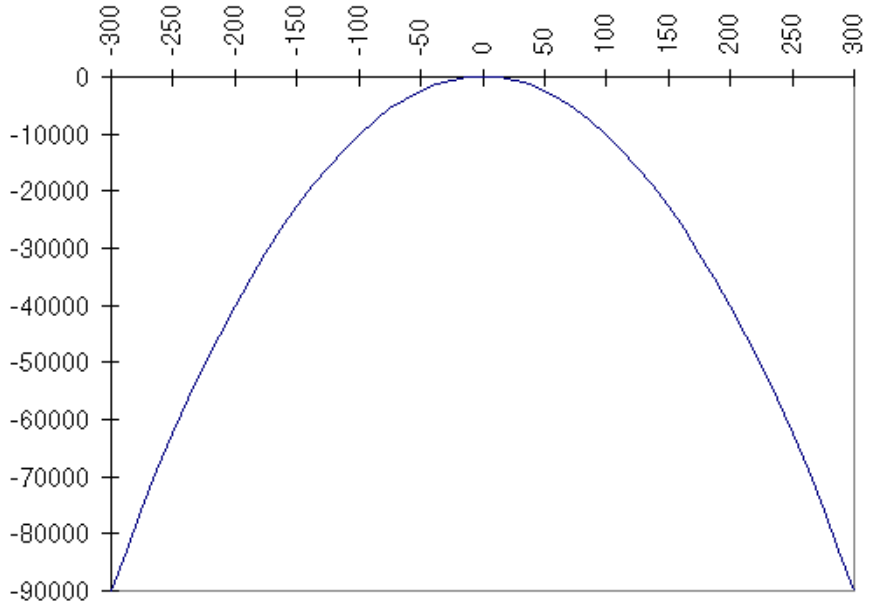


A geometric definition of convexity states that a function is convex if for any two points on or above the function, a straight line connecting the two points lies entirely on or above the function.

As illustrated in the preceding graph, a strictly convex function with no constraints has a single minimum. That is, water drizzled on it will collect in a single puddle at the global minimum. Therefore, minimizing a smooth convex function with no constraints will yield the global optimum regardless of the initial value of the adjustable cells. However, if the function is not convex, there may be more than one local optimum. In this case, the returned answer may be locally optimal, but not globally optimal.

Determining the convexity of a multiple variable problem is no easy task. Mathematicians call a function convex if the matrix of second derivatives is positive definite or has all positive Eigen values. A function is called concave if the matrix of second derivatives is negative definite. A function can be concave in one section and convex in another. The following function is strictly concave:

A Strictly Concave Function: Graph of -(A1²)



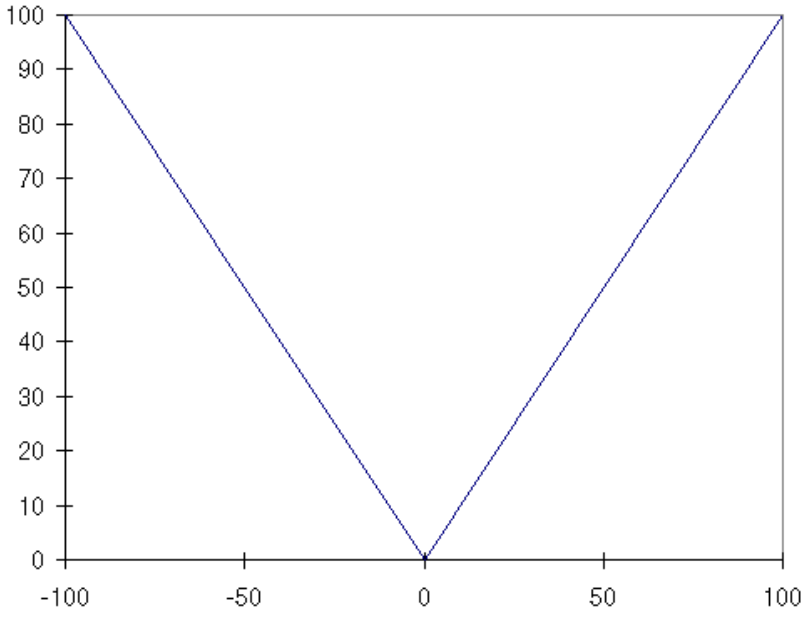
For a mixed function, look back at the graph of *X*SIN* (3.1416*B3) in the preceding section.

Smooth vs. Non-smooth Expressions

Smooth expressions have a defined first derivative (slope or gradient) at every point. Graphically, a smooth function of a single variable can be plotted as a single continuous line with no abrupt bends or breaks.

Non-smooth expressions include nondifferentiable and discontinuous functions. Expressions having one or more points for which the first derivative is not defined are called nondifferentiable. Graphs of nondifferentiable expressions may have abrupt bends at such points. Taking the absolute value of an adjustable cell, *ABS*(A1), is an example of a nondifferentiable expression. This is illustrated in the following graph:

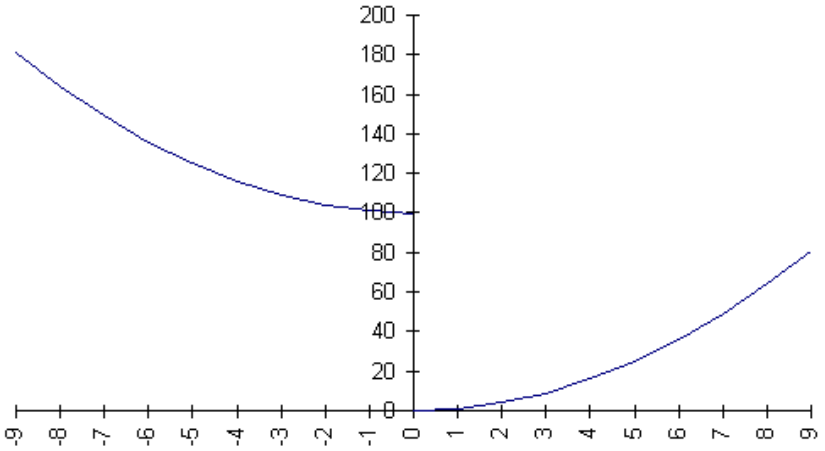
Graph of *ABS*(A1)



Here, there is an abrupt bend in the graph at the point of zero. This can dramatically increase solution times. Additional non-smooth functions are *MAX* and *MIN*.

Discontinuous functions are even more challenging to a nonlinear solver. Discontinuous functions are expressions whose graphs contain breaks. The spreadsheet *IF* function with adjustable cells as arguments is commonly used to express a discontinuous expression, as in the following:

Graph of *IF*(A1>0,A1^2,A1^2+100)



In simplified terms, What's*Best!* searches along the plot of the expression to find a maximum or minimum point representing an optimal solution. Breaks and sharp bends on the plots of non-smooth functions pose challenges to the solver not posed by smooth continuous functions. It is a good practice to avoid formulating your problem with non-smooth functions.

Solution Outcomes

The solution and analysis of the outcome of the solution process is presented in the status report worksheet. The status report worksheet is inserted following the completion of the solution process-unless you have disabled the status report (regarding disabling reports, see the *Reports, Location and Warnings* box on the *General Options* dialog box posted by the *Options…|General* command). The status report is opened by clicking the *WB! Status* tab among your Excel® worksheets.

Unless you have imposed a limit on the number of iterations in the solution search or have interrupted the solver by pressing the *Hold/Interrupt* button on the What's*Best!* solver status window, the message provided in the field *Solution Status* in the status report should be one of those listed below.

Case 1: Optimization Models

If a cell has been specified to be maximized or minimized, there are several possible outcomes of an attempt to solve the model:

Globally Optimal

During the solution search, the solver found the best possible solution that satisfies all the constraints. The message *Solution Status: GLOBALLY OPTIMAL* will be displayed in the status report upon successful solution of a linear model. There is no better answer that satisfies all constraints.

Locally Optimal

A locally optimal outcome applies to a nonlinear model when the solver has found the best solution within a local area that satisfies all of the constraints. In other words, no better solution exists within that local area. This means it is possible there is a locally optimal solution outside this immediate area that provides a better answer. It may be wise to consider re-solving the model with different starting values for the adjustable cells in attempting to investigate other local optima.

Optimality Conditions in Nonlinear Models

After successfully solving a nonlinear model, in addition to reporting a solution status of locally optimal, the status report also indicates whether the optimality conditions are *SATISFIED* or *UNCERTAIN*. A mathematician might refer to these optimality conditions as the Kuhn-Tucker conditions. A layman, on the other hand, might prefer to think of the optimality conditions as the "flatness" conditions.

Geometrically, for functions with no constraints, satisfaction of the optimality conditions means the objective function is flat (the slope is zero) in the neighborhood of the returned solution (i.e., it is at the top of the peak or bottom of the valley). Algebraically, it means that, for each variable, the partial

derivative is zero. Explaining these conditions on models with constraints lies beyond the scope of this help file. It suffices to say that the interpretation can be generalized to constrained models.

There are two primary reasons for the optimality conditions to be reported as uncertain. First, if the optimum (best cell value) lies at a discontinuity or a point where the objective function is non-smooth, the slope is undefined. At a point for which the slope is undefined, the optimality conditions cannot be said to be satisfied – hence they are determined uncertain. Second, if in the search for a solution What's*Best!* completes several iterations without making any significant improvement in the objective function, it will stop even though the optimality conditions are not satisfied. This may be a situation in which the objective function is very nearly flat over an unusually large region.

Whether the optimality conditions are satisfied or uncertain, if the solution is locally optimal, then another local optimum may exist that improves on the current solution. Even though the mathematical conditions to prove optimality may be inconclusive, the value returned in the best cell is locally optimal. Therefore, you should always consider re-solving the model with different values for the adjustable cells to see whether another local optimum improves the solution.

No Feasible Solution Found

The message *Solution Status: INFEASIBLE* is displayed in the status report if What's*Best!* was unable to find a solution that satisfied all the constraints. If the model is linear, then there is no answer that satisfies all constraints. If the model is nonlinear, then either no feasible answer exists or a feasible answer exists, but What's*Best!* was unable to find it. For details, please see the topic *Solution Status: No Feasible Solution.*

Unbounded

The message *Solution Status: UNBOUNDED* is returned if, without violating any constraints, the value of the cell to be minimized can be decreased without limit or the value of the cell to be maximized can be increased without limit. For details, please see the topic *Solution Status: UNBOUNDED.*

Numerical Error

The message *Solution Status: NUMERICAL ERROR* is returned if there was a serious error. For details, please see the topic *Solution Status: NUMERICAL ERROR.*

Case 2: Non-optimization Models

If no best cell has been specified (i.e., there is no objective function), there are only two possible outcomes of an attempt to solve the model. What's*Best!* either finds a feasible solution or reports that a feasible solution was not found. As with nonlinear optimization models, if the model is nonlinear, there may be a feasible solution that What's*Best!* was unable to find. You may want to consider re-solving with different initial values in the adjustable cells.

Guidelines for Modeling with What'sBest!

General Modeling Guidelines

An example showing how to build a simple linear model is provided in the *Tutorial* in *Getting Started*. In addition, the *Sample Models* demonstrate how linear and nonlinear models can be constructed for a variety of applications. As you construct your own model, it may be useful to periodically solve the model and check the status report to determine the type of model you have (linear/nonlinear) and its characteristics. In general, linear models are the fastest group of models to solve and nonlinear models are the most difficult to solve. It is not unusual that a nonlinear model will take many times longer to solve than a linear model with the same number of variables.

If a linear or nonlinear model is close to a solution (i.e., constraints are only slightly violated) then increasing one of the feasibility tolerances may allow you to reach a solution. Tolerances are set in both the *Linear Solver Options* and *Nonlinear Solver Options* dialog boxes.

For a difficult linear model, the options provided in the *Linear Solver Options* dialog box may help you to reach a solution or shorten the solution time. One of the first options you might try is turning on (or off) *Model Reduction*. Some linear problems will solve faster by a different *Solver Method*. The *Barrier* method may be considerably faster on big models, while the two simplex methods will tend to be faster on smaller, sparse models.

Large integer problems can be very difficult to solve and take a great deal of time arriving at a solution. To solve an infeasible problem or to shorten the solution time, you can try various options on the *Integer Solver Options* dialog box. Setting an *Optimality Tolerance* can improve run times dramatically if you are happy being within a certain percentage of the optimal solution. Some problems solve faster when the *Branching Direction* is set to *Up*. Setting the *Application* of the *Constraint Cuts* to *All Nodes* may help solve some problems. Setting the *Probing Levels* may also help shorten solution time. We recommend setting to a level of three or lower.

Nonlinear problems are generally difficult to solve. Setting *a Feasibility Tolerance* or *Optimality Tolerance* may help you to reach a solution. Trying different *Strategies* may also be useful to reaching a solution.

Many nonlinear problems are made unnecessarily difficult or slow to solve by the presence of nonlinear expressions that could be rewritten in linear form. The *Linearization* option is a tool for automatically rewriting these nonlinear expressions in linear form. Linearization can be a great performance boost if all nonlinear formulas in a model can be linearized, such that the final model becomes totally linear. If linearization can only be applied to a subset of the nonlinear formulas, then performance may erode.

The options described above for solving nonlinear problems are important aids to solving nonlinear problems. However, if the problem is poorly-formulated, then the options can only help alleviate the effects of poor formulation. The following section provides some suggestions for good formulation of nonlinear problems.

Nonlinear Modeling Guideline

Nonlinear models can be extremely complex to solve. It can pay off in terms of solution speed and reliability to spend a little extra time to make sure the model is formulated in a way that is most efficient to solve. This section gives some general guidelines to consider when building and solving nonlinear models.

Supply Good Initial Values

The initial adjustable cell values can affect the path What's*Best!* takes to the solution. Starting with initial adjustable cell values near the optimal solution can reduce the solution time.

In many situations, you may not know good initial values. However, when reasonable values are known, they should be used. If near-optimal values are not known, simply starting with values that satisfy all constraints may be helpful. Starting from any reasonable answer may be better than starting with zeroes in the adjustable cells.

Always start with adjustable cell values that avoid undefined values in adjustable-dependent equations. For example, if an equation divides by an adjustable cell, then a value of zero for that adjustable cell makes the equation take an undefined value. Start the solve attempt with a nonzero value for that adjustable cell.

Consider changing the initial values and re-solving the model if: 1) you suspect there is an answer better than the answer returned by What's*Best!*; 2) you know a feasible solution exists even though What's*Best!* returns the message "No Feasible Solution Found".

Use Reasonable Bounding Constraints

Adding constraints that bound the upper and/or lower values of adjustable cells to fall within a reasonable range of values may help shorten the solution time.

For example, assume you know that reasonable values for an adjustable cell A1 are between 500 and 1000. Values outside of this range are not mathematically impossible-they just don't make sense. Add the constraints: *WB*(A1,">=", 500) and *WB*(A1,"<=",1000). This ensures that What's*Best!* won't waste computation time investigating solutions you would view as meaningless. Similarly, adding bounds to restrict expressions from investigating values at or very near undefined regions (i.e., division by zero) may reduce computation time and increase the reliability of the solution.

Scale the Model to a Reasonable Range of Units

Try to model your problem so that the units involved are of similar orders of magnitude. Mathematically manipulating numbers that differ in size by a large amount can increase the computation time and may introduce problems due to round-off error.

For example, consider a financial problem with equations expressing an interest rate of 8.5% (.085) and budget constraints of $12,850,000. The difference in magnitude between these numbers is on the order of 10 to the 9th (1/100th compared to 10,000,000). A difference of 10 to the 4th or less between the largest and smallest units would be preferable. In this case, the budget could be expressed in units of millions of dollars. That is, $12.85 to represent $12,850,000. This lowers the difference in magnitude of the units of these numbers to 10 to the 4th (1/100th compared to 10).

Simplify Relationships

Whenever possible, use linear rather than nonlinear relationships. Some nonlinear expressions can be reformulated in a linear manner. A simple example is a constraint on the ratio of two adjustable cells. Consider the constraint: *WB*(A1/B1,"<=", .25) where A1 and B1 are adjustable cells that each must be greater than zero.

The relationship A1/B1 is nonlinear. If you multiply both sides of the constraint by B1, the constraint becomes: *WB*(A1,"<=", .25*B1). This is mathematically equivalent to the original constraint, but it is now a linear rather than nonlinear constraint.

As much as possible avoid non-smooth expressions, notably spreadsheet functions like *IF*, *MAX*, *MIN*, *ABS*, and table lookup functions. Models with non-smooth relationships are generally more difficult to solve. If you can, approximate the non-smooth relationship with a smooth expression.

Reduce Integer Restrictions

Minimizing the use of integer restrictions can drastically reduce the solution time. In instances involving larger numbers, you may find that solving the model without integer restrictions and then rounding yields acceptable answers in a fraction of the time the integer model requires.

Guidelines for Stochastic Modeling

So far, we worked with deterministic mathematical programs where model parameters (e.g. coefficients, bounds, etc.) are known constants. A stochastic program (SP) is a mathematical program (linear, nonlinear or mixed-integer) in which some of the model parameters are not known with certainty and the uncertainty can be described with known probability distributions. Applications arise in several industries:

- Multi-period financial portfolio planning with uncertain prices, interest rates, and exchange rates,
- Exploration planning for petroleum companies,
- Fuel purchasing when facing uncertain future fuel demand,
- Fleet assignment: vehicle type to route assignment in face of uncertain route demand,
- Electricity generator unit commitment in face of uncertain demand,
- Hydro management and flood control in face of uncertain rainfall,
- Optimal time to exercise for options in face of uncertain prices,
- Capacity and Production planning in face of uncertain future demands and prices,
- Foundry metal blending in face of uncertain input scrap qualities,
- Product planning in face of future technology uncertainty,
- Revenue management in the hospitality and transport industries.

Multistage Decision Making Under Uncertainty

Stochastic programs fall into two major categories a) Multistage Stochastic Programs with Recourse, and b) Chance-Constrained Programs. What's*Best!* capabilities are extended to solve models in both categories.

In this chapter, the term 'stochastic program' refers to a multistage stochastic model with recourse. The term 'stage', 'time' and 'period' are used interchangeably, unless otherwise is stated. The terms 'random parameter' and 'stochastic parameter' are also used interchangeably.

Multistage decision making under uncertainty involves making optimal decisions for a T-stage horizon before uncertain events (random parameters) are revealed while trying to protect against unfavorable outcomes that could be observed in the future.

In its most general form, a multistage decision process with T+1 stages follows an alternating sequence of decisions and random events (*decisions-events-decisions-…-events-decisions*).

0.1) in stage 0, we make a decision $x0$, taking into account that…

1.0) at the beginning of stage 1, "Nature" takes a set of random decisions $ù1$, leading to realizations of all random events in stage 1, and…

1.1) at the end of stage 1, having seen nature's decision, as well as our previous decisions, we make a recourse decision $x1(x0, ù1)$, taking into account that …

2.0) at the beginning of stage 2, "Nature" takes a set of random decisions $ù2$, leading to realizations of all random events in stage 2, and…

2.1) at the end of stage 2, having seen nature's decision, as well as our previous decisions, we make a recourse decision $x2$ $(x0, ù1, x1, ù2)$, taking into account that …

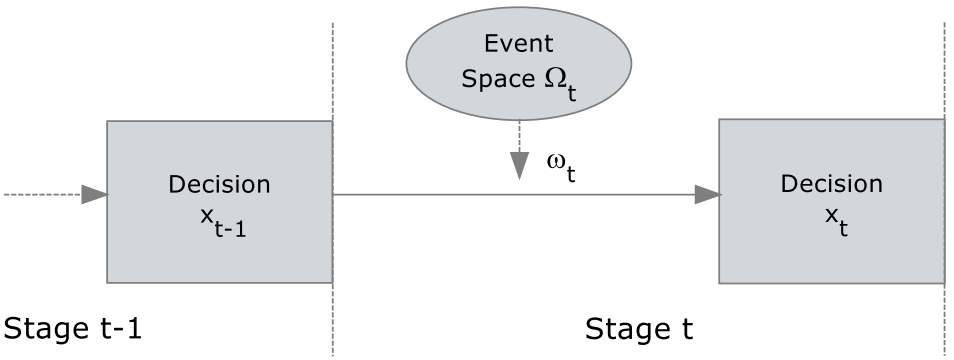
:

:

T.0) At the beginning of stage T, "Nature" takes a random decision, $ùT$, leading to realizations of all random events in stage T, and…

T.1) at the end of stage T, having seen all of nature's T previous decisions, as well as all our previous decisions, we make the final recourse decision $xT(x0, ù1, x1, ù2, ..., ùT,)$.

This relationship between the decision variables and realizations of random data can be illustrated as follows:



Each decision, represented with a rectangle, corresponds to an uninterrupted sequence of decisions until the next random event. And each random observation corresponds to an uninterrupted sequence of random events until the next decision point.

Note:	A stage is the pair: random event, followed by a decision. The initial stage 0 is special in that it consists only of our first decision. The final stage may be special in that it may consist of just a random event with no explicit final decision.

Recourse Models

The decision taken in stage-0 is called the initial decision, whereas decisions taken in succeeding stages are called 'recourse decisions'. Recourse decisions are interpreted as corrective actions that are based on the actual values the random parameters realized so far, as well as the past decisions taken thus far. Recourse decisions provide latitude for obtaining improved overall solutions by realigning the initial decision with possible realizations of uncertainties in the best possible way.

Restricting ourselves to linear multistage stochastic programs for illustration, we have the following form for a multistage stochastic program with (T+1) stages.

Minimize (or maximize) $c0x0 + E1[c1x1 + E2[\ c2x2 \ldots + ET[\ cTxT\ ] \ldots]]$

Such that

$$
\begin{array}{l}
A00x0 \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \sim\ b0 \\
A(\grave{u}1)10x0 \qquad + A(\grave{u}1)\ 11x1 \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \sim b(\grave{u}1)1 \\
A(\grave{u}1,\ldots,\grave{u}2)20x0 \ + A(\grave{u}1,\ldots,\grave{u}2)21x1 + A(\grave{u}1,\ldots,\grave{u}2)22x2 \qquad\qquad\qquad\qquad \sim b(\grave{u}1,\ldots,\grave{u}2)2 \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad

Scenario Tree

When the probability distributions for all random variables are discrete and finite, then there are only a finite number of outcomes in each stage. With each random parameter fixed to one of its possible outcomes, one can create a scenario representing one possible realization of the future. Enumeration of all possible combinations of outcomes allows us to represent all scenarios in a tree, with each scenario being a path from the root of the tree to one of its leaves. The nodes visited by each path correspond to values assumed by random parameters in the model.

Defining an SP Model in What'sBest! : Simple 2-Stage Example

There is a relatively straightforward five step process for setting up an SP model in What's*Best*!. All the information about the SP features is stored explicitly/openly on the spreadsheet. The five steps are:

Step 1) Define a core (or deterministic) model as a regular deterministic What's*Best*! model. You can "plug" regular numbers in a random cell to check results.

Step 2) Specify staging information about the sequence of decisions and random events. This information is stored directly on the spreadsheet using special What's*Best*! functions. These functions and formats are:
Decision variable and their stages are identified by: WBSP_VAR(stage, cell_list) and
Random variables and their stage are identified by: WBSP_RAND(stage, cell_list);

Step 3) Distribution information about random cells is stored in
WBSP_DIST_distribution(table, cell_list);
where distribution specifies the distribution, e.g., NORMAL.

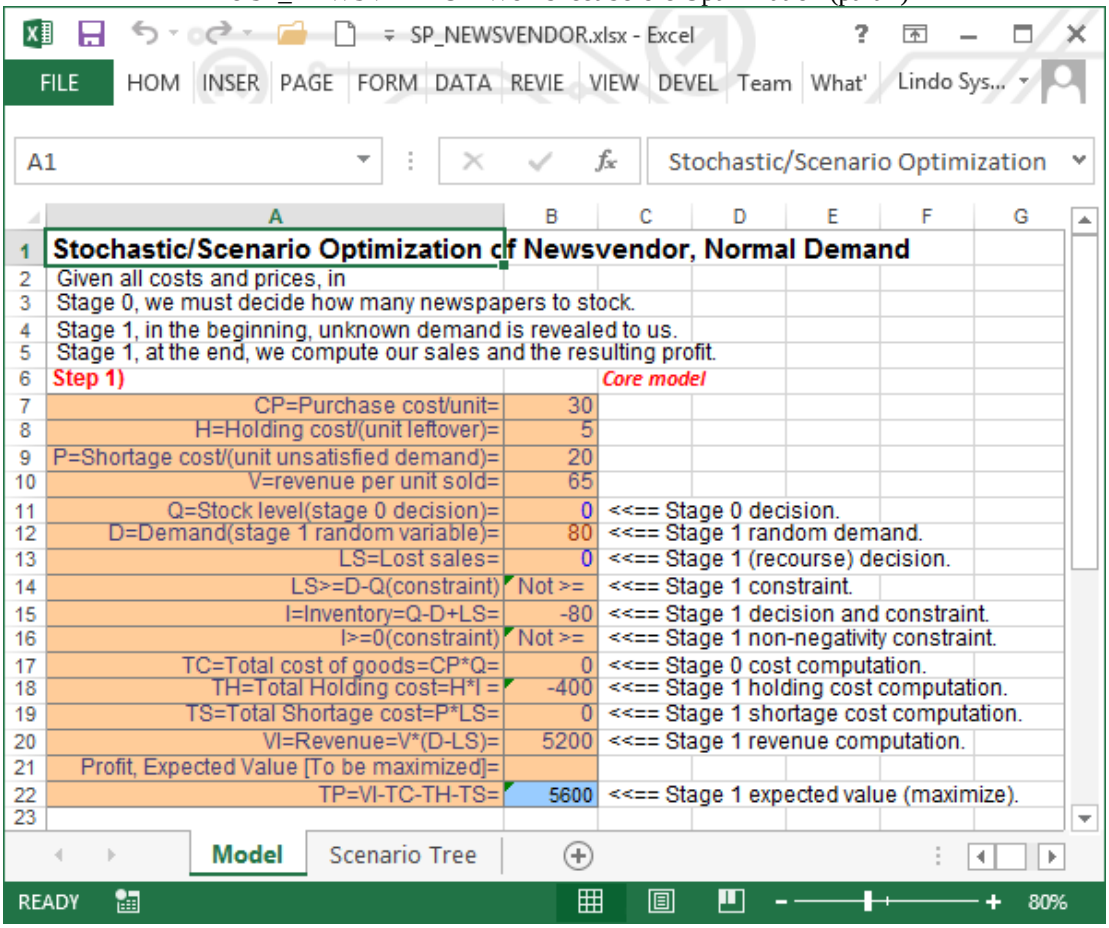
Step 4) Sample size for each stage is stored in
WBSP_STSC(table);

Step 5) Cells to be reported are listed in
WBSP_REP(cell_list) or WBSP_HIST(bins, cell_list);

You may type these functions in directly, or you may use the *Options...*|*Stochastic Solver* drop-down menu in What's*Best*! to guide you through the steps.

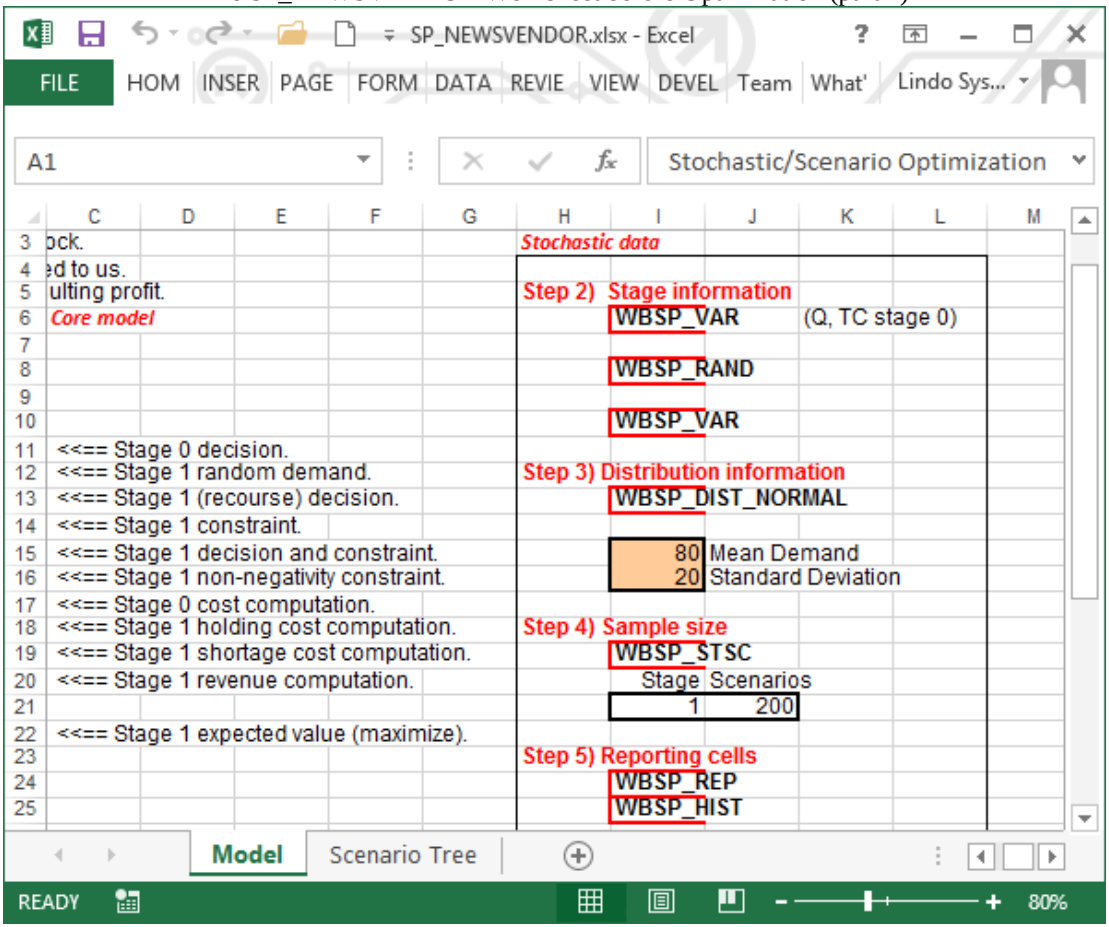
We illustrate the formulation of an SP model in What's*Best*! with perhaps the simplest SP model possible, the one product newsvendor problem (model *SP_NEWSVENDOR.XLSX*). This model has just two stages. In stage 0 (half stage) we must decide how much to stock, *Q*, of a single product. At the beginning of stage 1, demand, D, is revealed. At the end of stage 1 we sell the minimum(*D*,*Q*), enjoy our sales revenue and pay a penalty, if any, for unsatisfied demand or pay a holding cost for leftover inventory.

The SP_NEWSVENDOR Worksheet before Optimization (part 1)



	A	B	C
1	**Stochastic/Scenario Optimization of Newsvendor, Normal Demand**		
2	Given all costs and prices, in		
3	Stage 0, we must decide how many newspapers to stock.		
4	Stage 1, in the beginning, unknown demand is revealed to us.		
5	Stage 1, at the end, we compute our sales and the resulting profit.		
6	**Step 1)**		*Core model*
7	CP=Purchase cost/unit=	30	
8	H=Holding cost/(unit leftover)=	5	
9	P=Shortage cost/(unit unsatisfied demand)=	20	
10	V=revenue per unit sold=	65	
11	Q=Stock level(stage 0 decision)=	0	<<== Stage 0 decision.
12	D=Demand(stage 1 random variable)=	80	<<== Stage 1 random demand.
13	LS=Lost sales=	0	<<== Stage 1 (recourse) decision.
14	LS>=D-Q(constraint)	Not >=	<<== Stage 1 constraint.
15	I=Inventory=Q-D+LS=	-80	<<== Stage 1 decision and constraint.
16	I>=0(constraint)	Not >=	<<== Stage 1 non-negativity constraint.
17	TC=Total cost of goods=CP*Q=	0	<<== Stage 0 cost computation.
18	TH=Total Holding cost=H*I =	-400	<<== Stage 1 holding cost computation.
19	TS=Total Shortage cost=P*LS=	0	<<== Stage 1 shortage cost computation.
20	VI=Revenue=V*(D-LS)=	5200	<<== Stage 1 revenue computation.
21	Profit, Expected Value [To be maximized]=		
22	TP=VI-TC-TH-TS=	5600	<<== Stage 1 expected value (maximize).
23			

The SP_NEWSVENDOR Worksheet before Optimization (part 2)



For this specific problem, the five steps mentioned earlier are:
Step 1) The core model is described numerically in column B and in words in columns A and C.

Step 2) We specify cell B11 as a stage 0 decision variable by inserting the expression:
=WBSP_VAR(0,B11)
into cell I6. We can enter this expression directly, or using the Stochastic Support dialog box.
We specify cell B12 as a stage 1 random variable by inserting the expression:
=WBSP_RAND(1,B12) into cell I8.
The cell B13 is the recourse decision adjustable at stage 1:
=WBSP_VAR(1,B13)
Step 3) We tell What'*sBest*! that demand has a Normal distribution with mean 80 and standard deviation 20 by inserting the expression:
=WBSP_DIST_NORMAL(I15,I16,B12) into cell I13.

Step 4) We tell What's*Best*! how many scenarios to use in each stage (i.e., the sample size) by inserting the expression:

=WBSP_STSC(I21:J21) into cell I19.

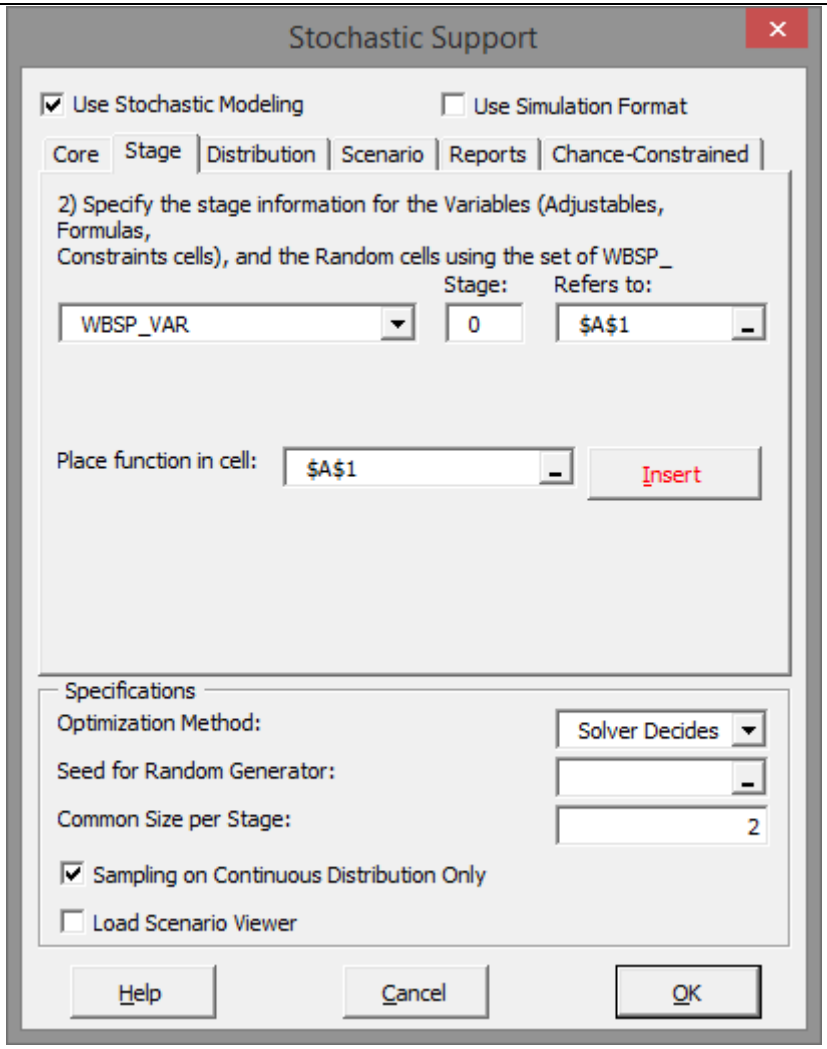
Step 5) We specify that we want a scenario-by-scenario report on the amount ordered, the demand, lost sales, and profit by placing the expression:

=WBSP_REP(B11,B12,B13,B22) in cell I24.

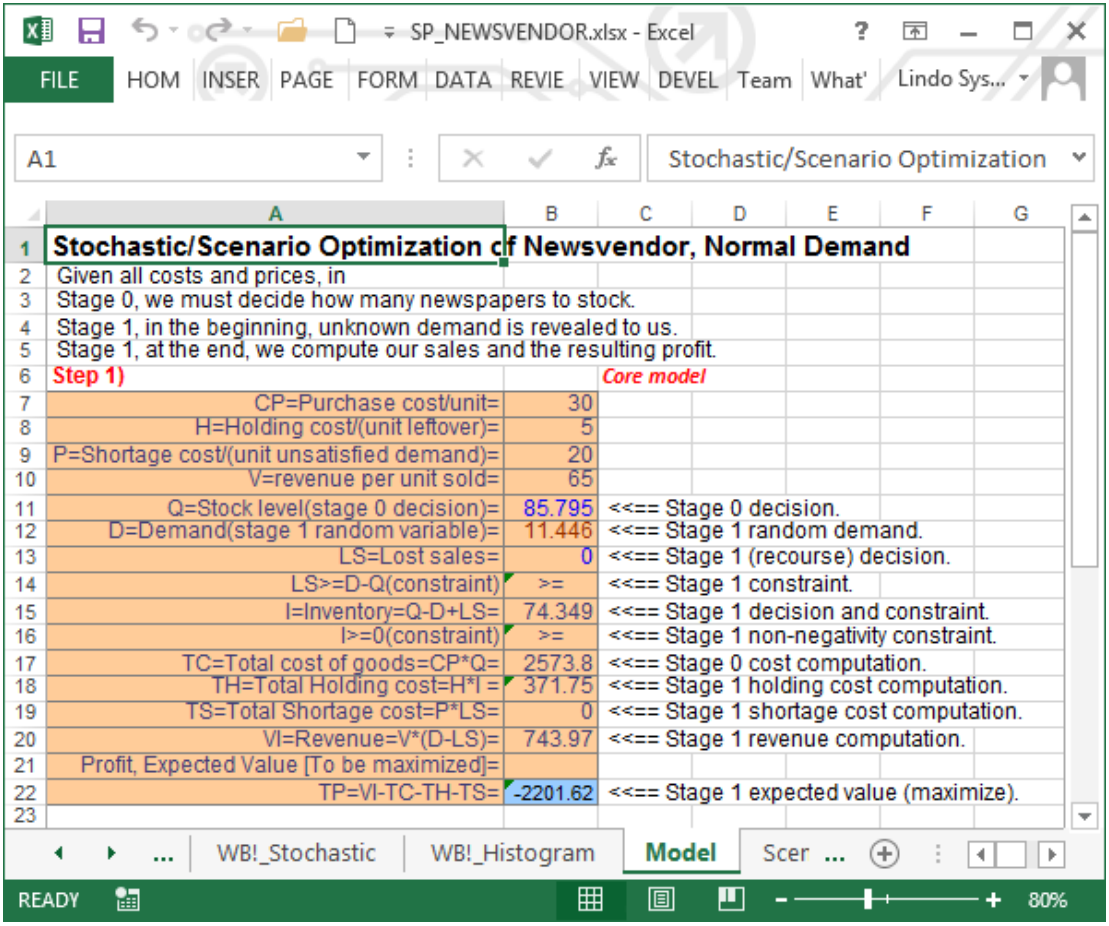
We request a nine bin histogram of Profit by placing the expression:

=WBSP_HIST(9,B22) in cell I25.

There is also an *Options...*|*Stochastic Solver* dialog box for setting various SP related options:

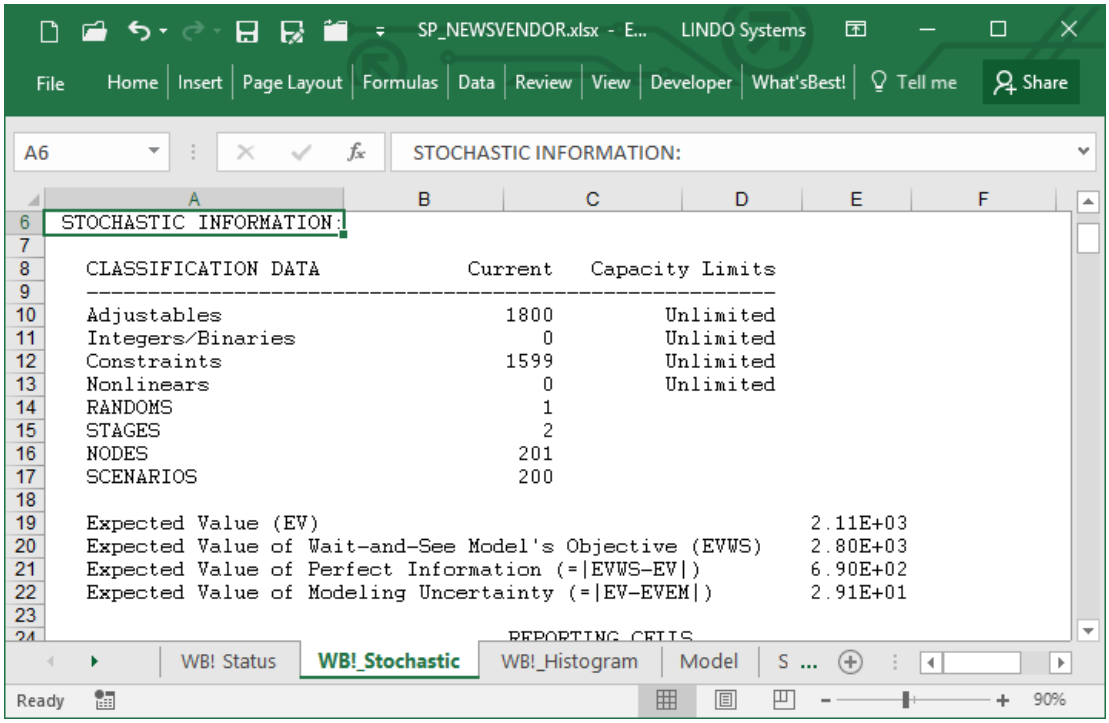

Stochastic Support
Use Stochastic Modeling
Use Simulation Format
Core
Stage
Distribution
Scenario
Reports
Chance-Constrained
2) Specify the stage information for the Variables (Adjustables, Formulas,
Constraints cells), and the Random cells using the set of WBSP_
Stage:
Refers to:
WBSP_VAR
0
$A$1
Place function in cell:
$A$1
Insert
Specifications
Optimization Method:
Solver Decides
Seed for Random Generator:
Common Size per Stage:
2
Sampling on Continuous Distribution Only
Load Scenario Viewer
Help
Cancel
OK

The SP_NEWSVENDOR Worksheet After Optimization



The solver writes a series of solution in the created tab *'WB!_Stochastic', and 'WB!_Histogram'*, with the expected value, and displays the first default scenario, or a selected scenario, on the spreadsheet.

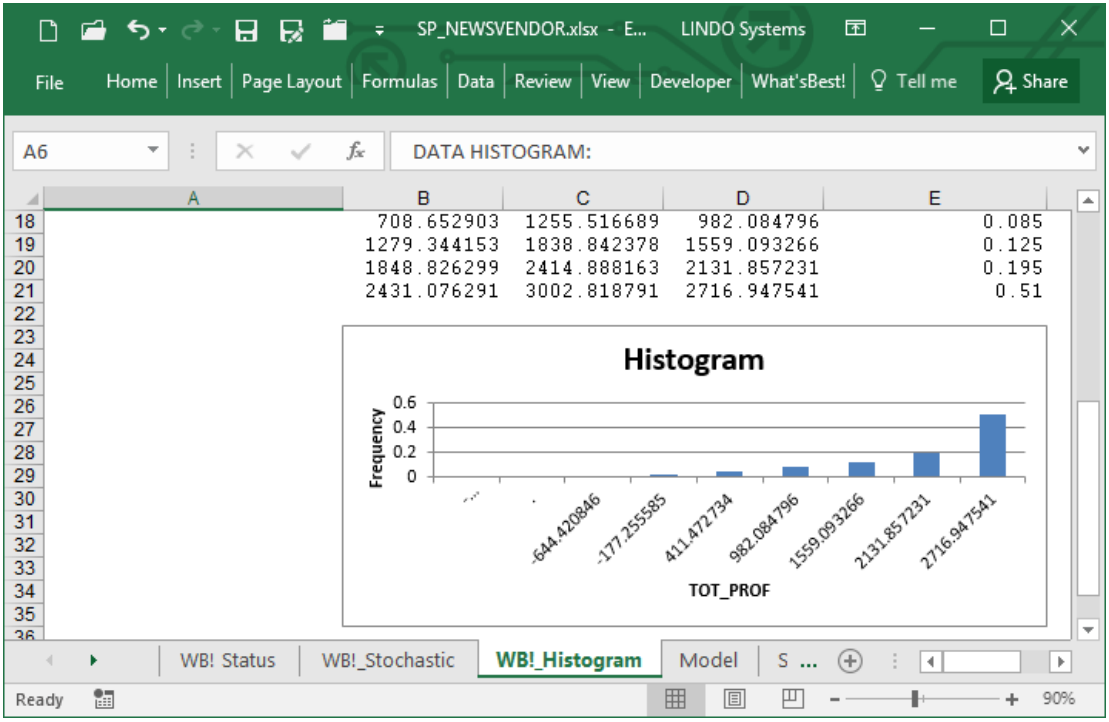
The scenario-by-scenario report is generated on the *WB!_Stochastic* tab. A portion of it appears below.



	A	B	C	D	E	F
6	STOCHASTIC INFORMATION:					
7						
8	CLASSIFICATION DATA		Current	Capacity Limits		
9	----------					
10	Adjustables		1800	Unlimited		
11	Integers/Binaries		0	Unlimited		
12	Constraints		1599	Unlimited		
13	Nonlinears		0	Unlimited		
14	RANDOMS		1			
15	STAGES		2			
16	NODES		201			
17	SCENARIOS		200			
18						
19	Expected Value (EV)				2.11E+03	
20	Expected Value of Wait-and-See Model's Objective (EVWS)				2.80E+03	
21	Expected Value of Perfect Information (=|EVWS-EV|)				6.90E+02	
22	Expected Value of Modeling Uncertainty (=|EV-EVEM|)				2.91E+01	
23						
24			REPORTING CELLS			

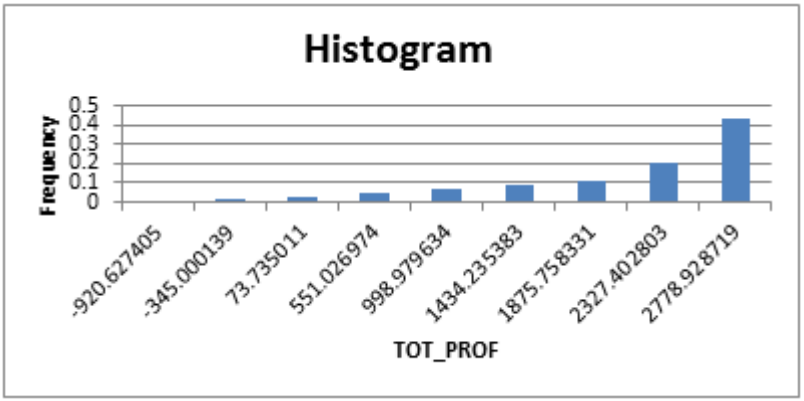
Any of the standard statistical tools in Excel can by used to analyze the scenario data.

The histogram generated on the *WB!_Histogram* tab is:



Notice that even though the driving random variable, Demand, has a Normal distribution, the total profit has a highly skewed, decidedly non-Normal, distribution.

The histogram graph is:



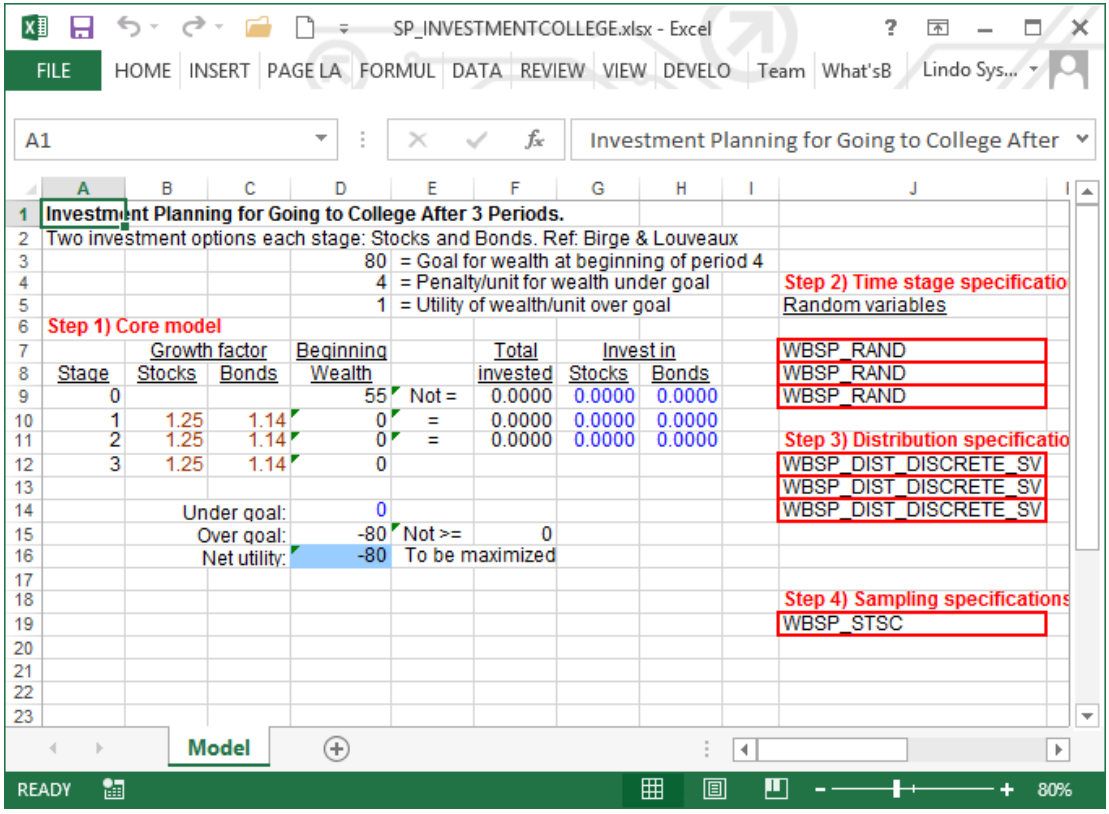
Defining an SP Model in What'sBest! : Multi-Stage Example

Next we give a slightly more complex example that illustrates how to formulate an SP with more than two stages (model *SP_INVESTMENTCOLLEGE.XLSX*). This example was originally given by Birge and Louveaux in their book, *Introduction to Stochastic Programming*. You want to set aside $55,000 now, hoping that it will grow to $80,000 in three periods so as to support your child in college. Only two investments are available, Stocks, and Bonds. At the beginning of each period or stage you (re)allocate you money between these two investments. Each year, the return on each investment is a random variable, given by the table:

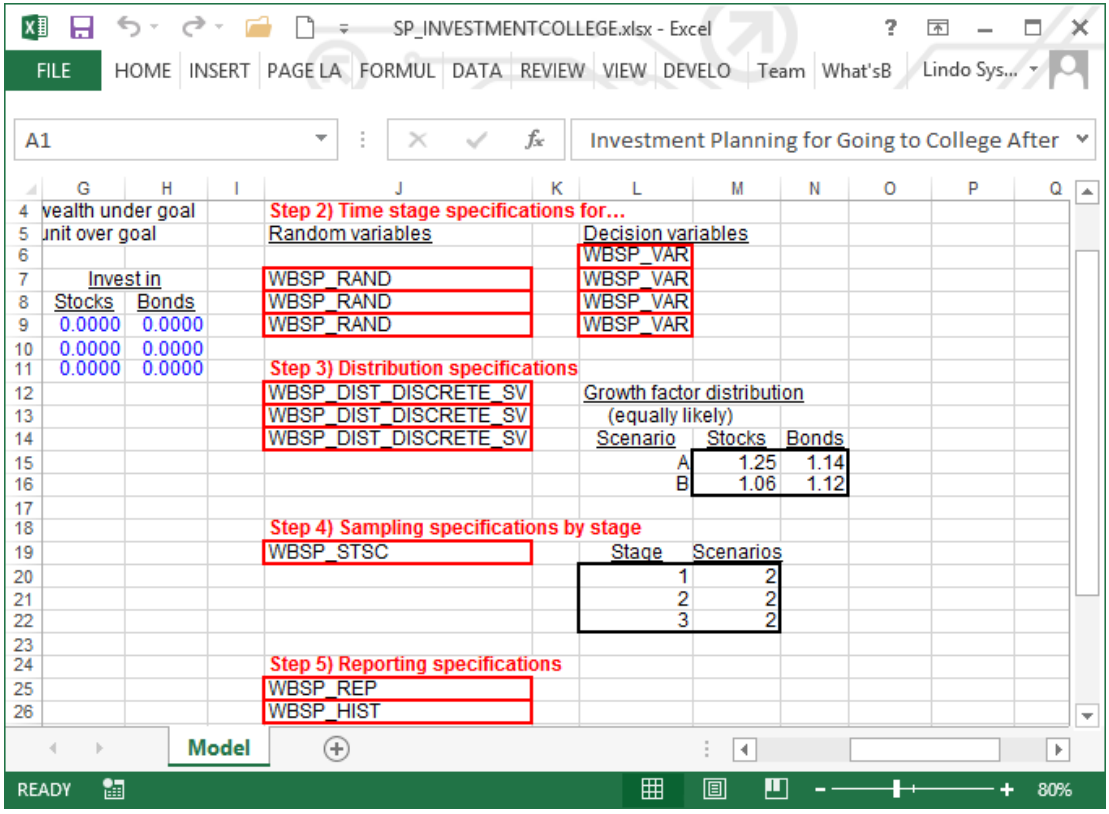
	Stocks	Bonds
Scenario 1	1.25	1.14
Scenario 2	1.06	1.12

Stocks have the higher expected return, so if we just wished to maximize expected return, we would put all our money in Stocks at the beginning of each stage. Stocks, however, also have the greater variability, so if we are concerned about achieving our goal of $80,000, then Bonds might seem more attractive with their lower variability in return.

The SP_INVESTMENTCOLLEGE Worksheet Before Optimization (part 1)



The SP_INVESTMENTCOLLEGE Worksheet Before Optimization (part 2)



Stepping through the five steps again for this multistage example,
Step 1) The core model is described in the range A3:H16.
The growth factors for Stocks and Bonds are in columns B and C. They will be declared random in step 2 below. The beginning wealth before each allocation decision is given in column D. At the beginning it is a constant 55. In subsequent periods the wealth available comes previous period investments and is given by the formulae:

D10=SUMPRODUCT(G9:H9,B10:C10)
D11=SUMPRODUCT(G10:H10,B11:C11)
D12=SUMPRODUCT(G11:H11,B12:C12)

The decision variables of how much to invest in Stocks and Bonds each period is in columns G and H.
The total amount invested is computed in column F and is given by:

F9=SUM(G9:H9)
F10=SUM(G10:H10)
F11=SUM(G11:H11)

We require the amount invested each period to equal the amount available.
This is enforced by the constraints in column E:

E9=WB(D9,"=",F9)
E10=WB(D10,"=",F10)

E11=WB(D11,"=,F11)

The amount over goal at the end is computed with:

D15=D12-D3+D14

The constraint in E15 constrains this to be nonnegative.

E15=WB(D15,">=",F15)

The amount under goal is in D15.

The net utility to be maximized is given by:

D16=D5*D15-D4*D14

Step 2) We specify cells G9, H9 as a stage 0 (half stage) decision variable by inserting the expression:

=WBSP_VAR(0,G9,H9)

into cell L6. We specify cells B10, C10 as stage 1 random variables that are observed at the beginning of stage 1 by inserting in cell J7 the expression:

=WBSP_RAND(1,B10,C10).

Similar expressions are in cells L7, L8, L9, J8, and J9 for the remaining stages 1 to 3.

The cell D14 is the recourse decision adjustable at stage 3:

=WBSP_VAR(3,D14)

Step 3) We tell What's*Best*! that the growth factors have a joint discrete distribution as listed in the scenario table S13:T14 by inserting the expressions:

=WBSP_DIST_DISCRETE_SV(M15:N16,B10,C10)

=WBSP_DIST_DISCRETE_SV(M15:N16,B11,C11)

=WBSP_DIST_DISCRETE_SV(M15:N16,B12,C12)

into cells J12, J13, and J14.

Step 4) We tell What's*Best*! how many scenarios to use in each stage (i.e., the sample size) by inserting the expression:

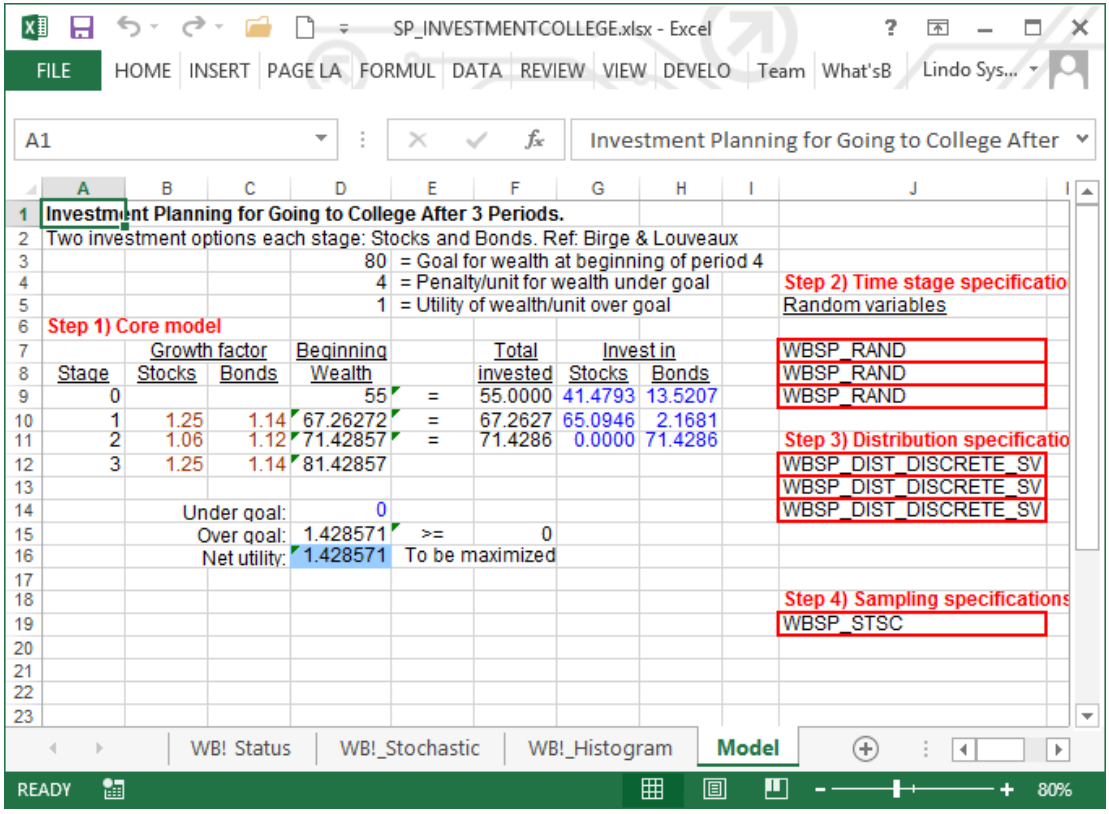
=WBSP_STSC(L20:M22) into cell J19.

Step 5) We specify that we want a scenario-by-scenario report of the wealth and the allocation to Stocks and Bonds each period, by placing the expression:

=WBSP_REP(F9,G9,H9,Wealth1,G10,H10,Wealth2,G11,H11,D12,D14,D15,D16)

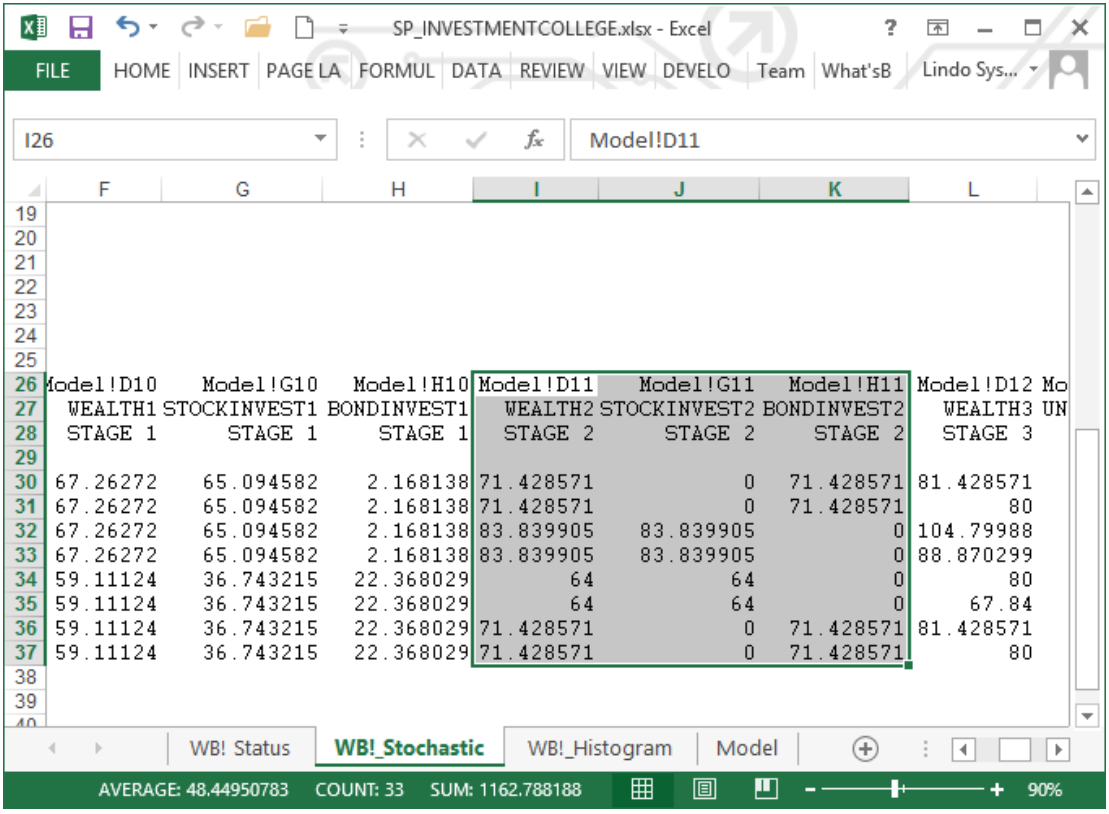
into cell J25. Similarly, the cell J26 contains the function for the histogram, over 10 bins, as =WBSP_HIST(10,D12).

The SP_INVESTMENTCOLLEGE Worksheet After Optimization



The solver writes a series of solution in the created tab *'WB!_Stochastic', and 'WB!_Histogram'*, with the expected value, and displays the first default scenario, or a selected scenario, on the spreadsheet.

The scenario-by-scenario report is generated on the *WB!_Stochastic* tab:



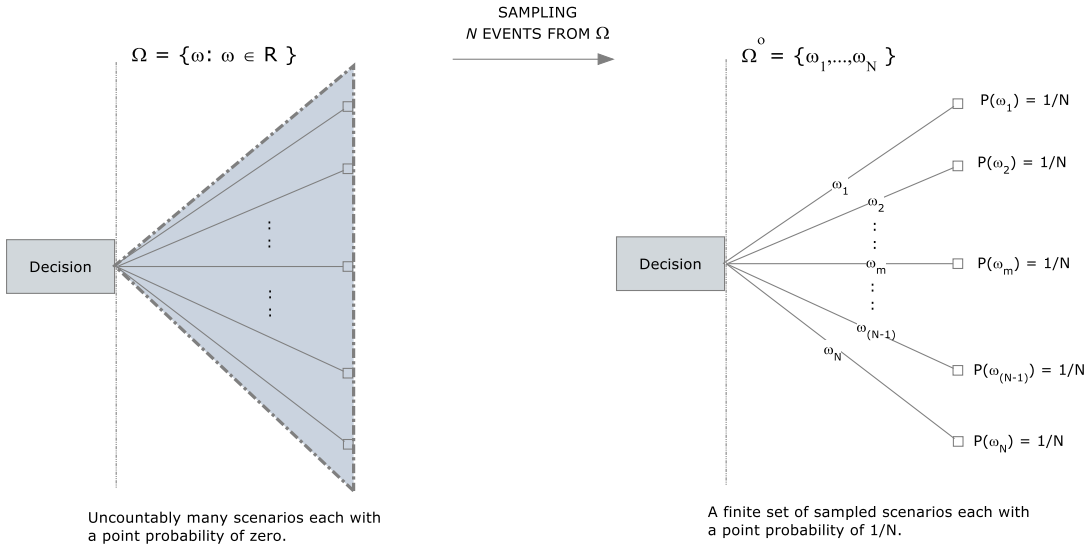
	F	G	H	I	J	K	L
26	Model!D10	Model!G10	Model!H10	Model!D11	Model!G11	Model!H11	Model!D12 Mo
27	WEALTH1	STOCKINVEST1	BONDINVEST1	WEALTH2	STOCKINVEST2	BONDINVEST2	WEALTH3 UN
28	STAGE 1	STAGE 1	STAGE 1	STAGE 2	STAGE 2	STAGE 2	STAGE 3
30	67.26272	65.094582	2.168138	71.428571	0	71.428571	81.428571
31	67.26272	65.094582	2.168138	71.428571	0	71.428571	80
32	67.26272	65.094582	2.168138	83.839905	83.839905	0	104.79988
33	67.26272	65.094582	2.168138	83.839905	83.839905	0	88.870299
34	59.11124	36.743215	22.368029	64	64	0	80
35	59.11124	36.743215	22.368029	64	64	0	67.84
36	59.11124	36.743215	22.368029	71.428571	0	71.428571	81.428571
37	59.11124	36.743215	22.368029	71.428571	0	71.428571	80

The scenario report we obtain is show above. Our expected utility at the end of the three periods is -1.514085. This means that there were some outcomes under which our final wealth fell short of our target wealth of $80. This report gives complete detail on what to do each period. For example, at the beginning we should split our $55 by investing $41.479 into Stocks and $13.521 in Bonds. Notice the interesting policy in period\stage 2. If our wealth is well below $80, i.e., 64, or if our wealth is above $80, then we invest all of our portfolio into Stocks. If our wealth is intermediate, i.e., 71.428571, then we invest all portfolio into bonds. The reasoning is that if we know we are going to fall short of our target or if we have already achieved our target, then we might as well invest in the instrument with highest expected returns, Stocks. On the other hand, if our beginning wealth stage 2 is $71.428571, then if we put all our money into bonds, then we are guaranteed to (just) achieve our target of $80.

Monte Carlo Sampling

In stochastic programming where one or more stochastic parameters have continuous or discrete but infinite event space, there are an infinite number of possible scenarios, making the model computationally intractable. For such cases Monte Carlo sampling (also called pre-sampling) can be used to approximate the problem to work with a finite scenario tree. As illustrated in the figure below, if the model has a single stochastic parameter with a continuous distribution such as the Normal Distribution; one can discretize the event space simply by generating *N* sample points and construct a finite and tractable scenario tree. This is also true for discrete distributions with infinite event space like the Poisson distribution.

Note: Sampling a scenario tree prior to the optimization process is also called pre-sampling. This is to distinguish this type of sampling from the one that is used during optimization process. In What's*Best!*, sampling refers to pre-sampling unless otherwise is stated.



Note: Because the point probability of each scenario in the original model is zero, it is customary to set the probabilities of sampled scenarios to 1/*N*. However, the user can always define customized sampling approaches to work with different scenario probabilities.

Given the parametric distribution of each stochastic parameter, What's*Best!*'s sampling routines can be used to generate univariate samples from these distributions efficiently. The user has the option to use antithetic-variates or Latin-hypercube sampling to reduce the sample variance.

Generating Dependent Samples

In certain situations, the modeler may require some of the random variables to be dependent on each other. There are several ways of doing this in What's*Best*!. You can specify an explicit, arbitrary discrete joint distribution table form any of:

WBSP_DIST_DISCRETE_SV(*table, rand_vars*) (Equi-probable, sequenced vertically)
WBSP_DIST_DISCRETE_SH(*table, rand_vars*) (Equi-probable, sequenced horizontally)
WBSP_DIST_DISCRETE_SV_W(*table,prob,rand_vars*) (Weighted prob., sequenced vertically)
WBSP_DIST_DISCRETE_SH_W(*table,prob,rand_vars*) (Weighted prob., sequenced horizontally)

Another common way to characterize dependencies among variables drawn from standard univariate distributions such as the Normal or Poisson is by standard correlation measures. What's*Best*! supports three correlation types:

WBSP_CORR_PEARSON(*matrix,rand_vars*) (Pearson's linear correlation)
WBSP_CORR_SPEARMAN(*matrix,rand_vars*) (Spearman's rank correlation)
WBSP_CORR_KENDALL(*matrix,rand_vars*) (Kendall's rank correlation)

Sampling a Scenario Tree

For stochastic programs with infinite event space, What's*Best!* offers an easy to use function to create finite scenario trees implicitly with user-specified dimensions. This is especially handy when there are several stochastic parameters and the task of explicit sampling becomes tedious. In this context, the user can specify the dimensions of a scenario tree by either of the following methods:

Specify the number of nodes per stage: In this method, the user should provide an integer array of length *T* (number of stages in the model) and give in each position the number of nodes to be created in that stage. By default stage-0 will always one node, thus the 0th index in the array will be one. Other positions in the array, corresponding to the number of nodes in stages 1,2,…*T*-1, may take any positive integer values. In this framework, each node represents a block realization of all the stochastic parameters in that stage and will have a conditional probability of 1/*Nt*, where *Nt* represents the number of nodes in stage *t*.
Specify the sample size per stochastic parameter: In the method, the user should provide an integer array of length *S* (the number stochastic parameters in the model), and give in each position the sample size for that stochastic parameter.

In either case, What's*Best!* will automatically construct a finite scenario tree with specified dimensions.

In a case where both stochastic parameters are normally distributed each belonging to a different stage. Therefore, creating *N* nodes per stage has the same effect as creating *N* samples per stochastic parameter whenever there is a single stochastic parameter per stage.

Note: Sampling a scenario tree is not limited to stochastic parameters that follow parametric distributions. It is also possible to use sampling for models, which already have a finite scenario tree. This is especially useful when the original tree is finite but still too big to handle computationally. For instance, a 2-stage model may have 30 stochastic parameters with two outcomes each. This will correspond to a scenario tree with 2^{30} = 1.0737e+009 scenarios. Sampling is essential for models with scenario trees this big.

Guidelines for Chance-Constrained Modeling

The second major class of models in stochastic programming is Chance-Constrained Programs (CCP). A CCP model is a) similar to general stochastic programs in that the model contains random quantities with known distributions, but b) simpler in that the model has just a single decision stage and a single random outcome stage.

The goal in CCP is to make an optimal decision prior to realization of random data, while controlling the chance that constraints are violated. Consider an LP with random matrix E and random right-handside w:

Min c x
E x_i = w_i *i* = 1...m

If we required all *m* realizations of E *x* = w to be satisfied, then we would get a very conservative/expensive solution *x*, or no feasible solution at all. The distinctive feature of CCP is that we require that E *x* = w be satisfied with some prespecified probability, $0 < p < 1$, as opposed to it being satisfied for all possible realizations of (E, w).

Defining a CCP Model in What'sBest! : Simple Feed Mixer Example

There is a relatively straightforward six-step process for setting up an SP/CCP model in What's*Best*!. All the information about the SP features is stored explicitly/openly on the spreadsheet, then the CCP part. The six steps are:

Step 1) Define a core (or deterministic) model as a regular deterministic What's*Best*! model. You can "plug" regular numbers in a random cell to check results.

Step 2) Specify staging information about the sequence of decisions and random events. This information is stored directly on the spreadsheet using special What's*Best*! functions. These functions and formats are:
Decision variable and their stages are identified by: WBSP_VAR(stage, cell_list), where stage is 0, and
Random variables and their stage are identified by: WBSP_RAND(stage, cell_list), where stage is 1;

Step 3) Distribution information about random cells is stored in

WBSP_DIST_distribution(table, cell_list);
where distribution specifies the distribution, e.g., NORMAL.

Step 4) Sample size for the 1 stage is stored in
WBSP_STSC(table);

Step 5) Cells to be reported are listed in
WBSP_REP(cell_list) or WBSP_HIST(bins, cell_list);

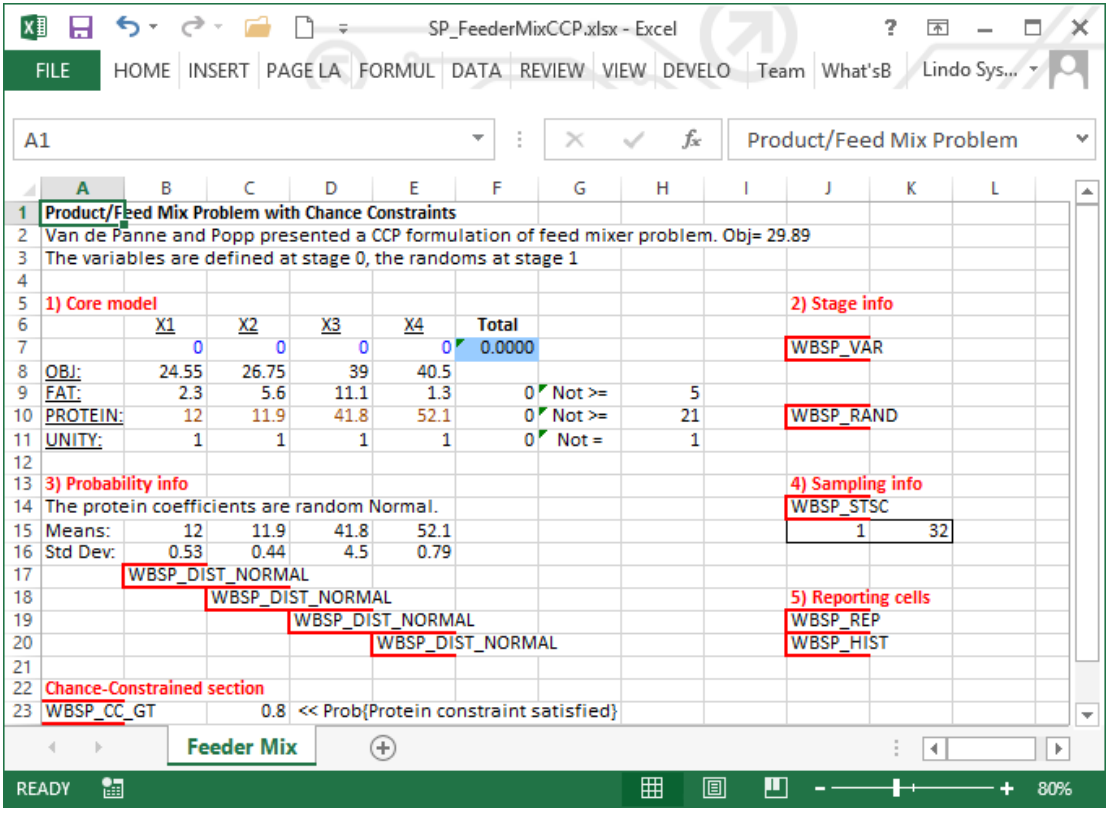
Step Chance-Constrained) Cells to be reported are listed in
WBSP_CC_LT(percent, weight, constraint cells) for "Less Than", or WBSP_CC_GT(percent, weight, constraint cells) for "Greater Than". One function will define one set of Chance Constraint.

You may type these functions in directly, or you may use the Options...|Stochastic Solver drop-down menu in What's*Best*! to guide you through the steps.

We illustrate the formulation of an SP/CCP model in What's*Best*! with perhaps the simplest SP/CCP model possible, the Feeder Mix problem (model *SP_FEEDERMIXCCP.XLSX*). This model has just 1 stage. In stage 0 (half stage) we must decide how much X1..X4. At the beginning of stage 1, Protein, is revealed. At the end of stage 1 we minimize the Objective.

This model is an excerpt of the blending model, *HOGCHANC.XLSX*, explained in the Chance-Constrained Blending example.

The SP_FEEDERMIXCCP Worksheet before Optimization



For this specific problem, the five steps mentioned earlier are:
Step 1) The core model is described numerically in the area B7 toH11.

Step 2) We specify cell B7:E7 as a stage 0 decision variable by inserting the expression:

=WBSP_VAR(0,B7:E7)

into cell J7. We can enter this expression directly, or using the Stochastic Support dialog box.

We specify cell B10:E10 as a stage 1 random variable by inserting the expression:

=WBSP_RAND(1,B10:E10) into cell J10.

Step 3) We tell What's*Best*! that the protein coefficient has a Normal distribution with different Means and Standard Deviation by inserting the 4 separated expressions:

=WBSP_DIST_NORMAL(B15,B16,B10) into cell B17,

=WBSP_DIST_NORMAL(C15,C16,C10) into cell C18,

=WBSP_DIST_NORMAL(D15,D16,D10) into cell D19,

=WBSP_DIST_NORMAL(E15,E16,E10) into cell E20.

Step 4) We tell What's*Best*! how many scenarios to use in each stage (i.e., the sample size) by inserting the expression:

=WBSP_STSC(J15:K15) into cell J14.

Step 5) We specify that we want a scenario-by-scenario report on the Protein amount against its cosntraint by placing the expression:

=WBSP_REP(F7,F10,H10) into cell J19.

We request a histogram of Protein Amount by placing the expression:

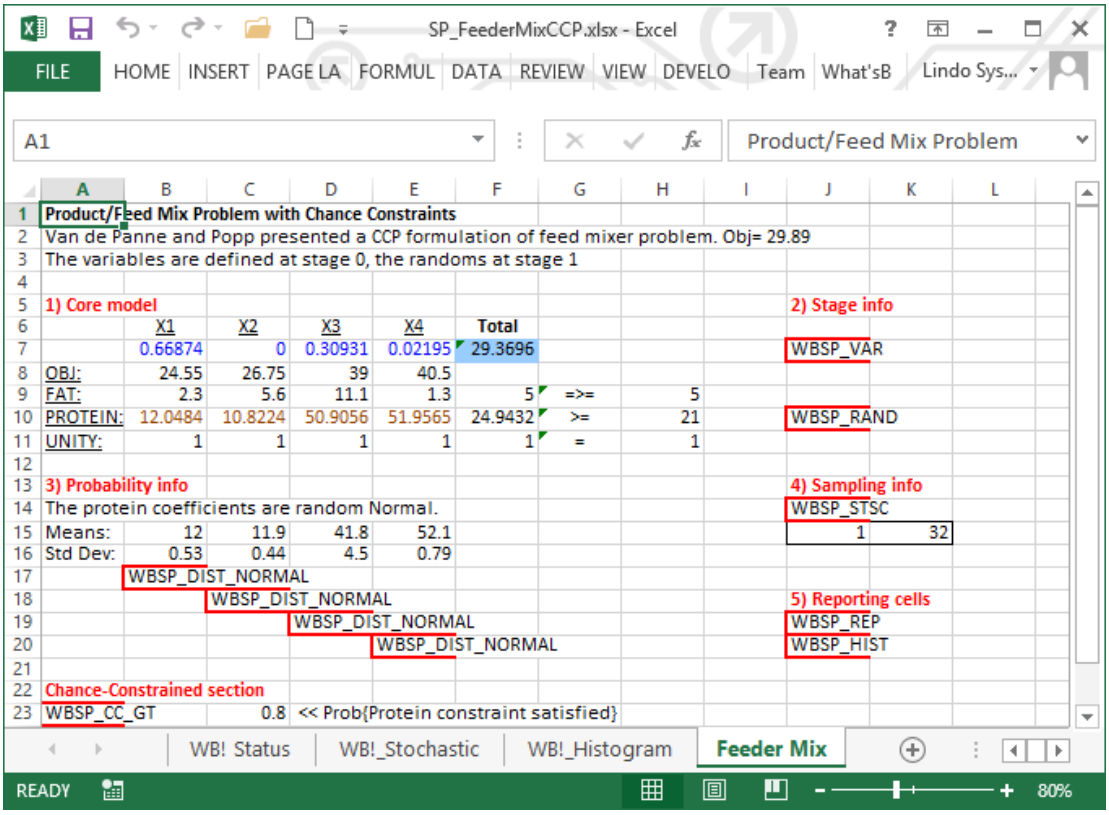
=WBSP_HIST(0,'Feeder Mix'!F10) into cell J20.

Chance-Constrained We specify that we want more than 80% probability to have the Protein amount constraint satisfied to consider the model feasible, by placing the expression:

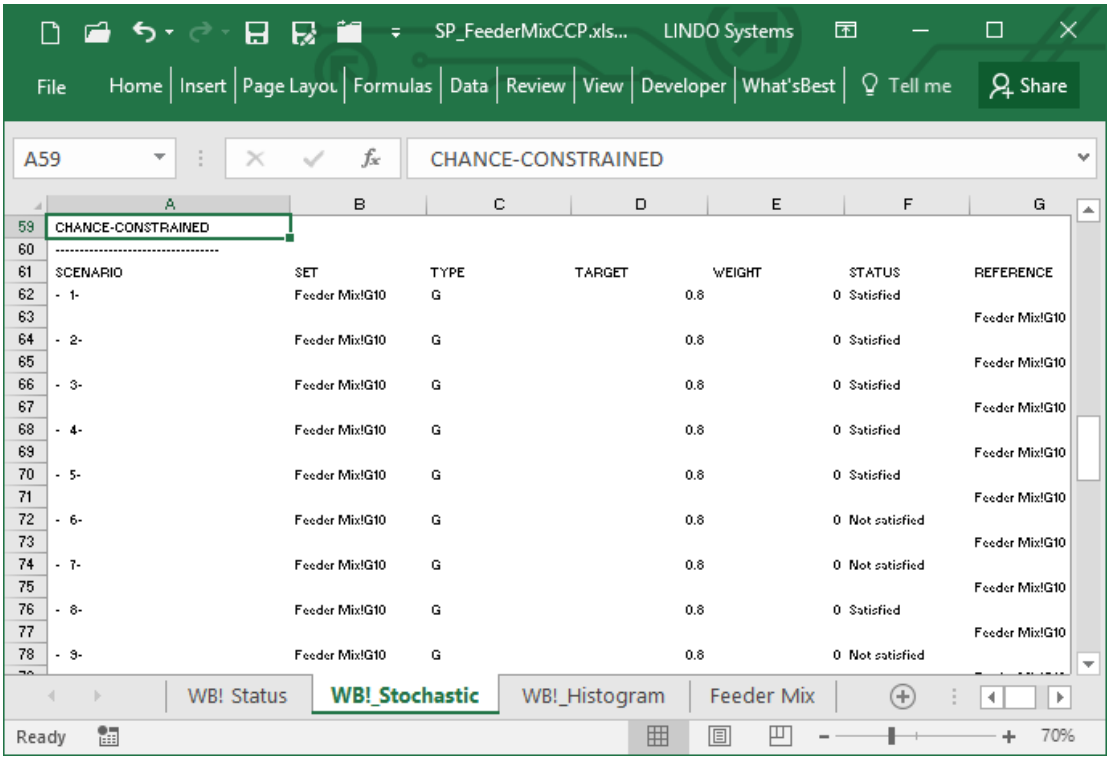
=WBSP_CC_GT(C23,0,G10) into cell A23.

All the Expected Value calculation will be turned off in this type of model.

The SP_FEEDERMIXCCP Worksheet After Optimization



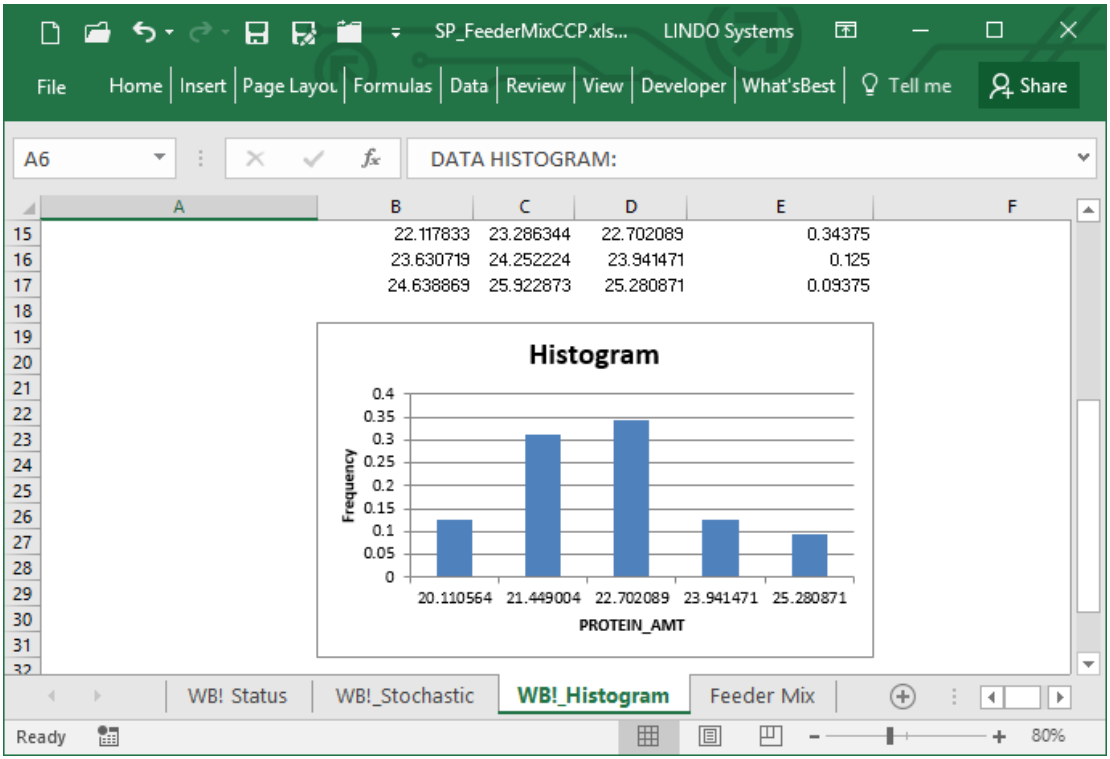
The scenario-by-scenario report is generated on the *WB!_Stochastic* tab. A portion of it appears below.



	A	B	C	D	E	F	G
59	CHANCE-CONSTRAINED						
60	----------						
61	SCENARIO	SET	TYPE	TARGET	WEIGHT	STATUS	REFERENCE
62	- 1-	Feeder Mix!G10	G		0.8	0 Satisfied	
63							Feeder Mix!G10
64	- 2-	Feeder Mix!G10	G		0.8	0 Satisfied	
65							Feeder Mix!G10
66	- 3-	Feeder Mix!G10	G		0.8	0 Satisfied	
67							Feeder Mix!G10
68	- 4-	Feeder Mix!G10	G		0.8	0 Satisfied	
69							Feeder Mix!G10
70	- 5-	Feeder Mix!G10	G		0.8	0 Satisfied	
71							Feeder Mix!G10
72	- 6-	Feeder Mix!G10	G		0.8	0 Not satisfied	
73							Feeder Mix!G10
74	- 7-	Feeder Mix!G10	G		0.8	0 Not satisfied	
75							Feeder Mix!G10
76	- 8-	Feeder Mix!G10	G		0.8	0 Satisfied	
77							Feeder Mix!G10
78	- 9-	Feeder Mix!G10	G		0.8	0 Not satisfied	

Any of the standard statistical tools in Excel can by used to analyze the scenario data.

The histogram generated on the *WB!_Histogram* tab is:



Notice that even though the driving random variable, Protein amount, has a Normal distribution.

The solver writes a series of solution in the created tab *'WB!_Stochastic', and 'WB!_Histogram'*, with the expected value, and displays the first default scenario, or a selected scenario, on the spreadsheet.

Solving Second Order Cone Programs

The optimization capabilities of What's*Best!* extend to the solution of second-order-cone problems (SOCP) of the following form

Optimize ||A0 *x* + b0 ||+ *c0 x*

subject to:

||Ai *x* + bi ||+ *ci x* - *di* ? 0 for *i* = 0,1,…,*m*-1,

Lj <= *xj* <= *Uj* for *j* = 0,1,…,*n*-1,

xj is integer for *j* in a specified *J C*= {0,…, *n*-1}

where

Optimize is either minimize or maximize,

Ai are matrices of appropriate dimensions *i*=0,…,*m*-1,

bi and *ci* are vectors of constants,

di are constants,

x = {*x*0, *x*2,…,*xn*-1}, is an *n*-vector of decision variables.

"?" is one of the relational operators "<=", "=", or ">=".

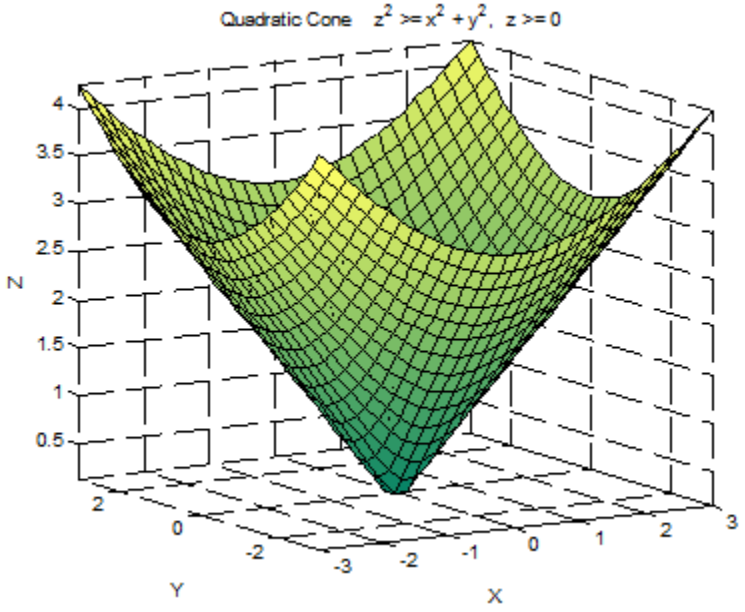
Without the integrality restrictions, SOCPs are nonlinear convex problems that include linear and convex quadratically constrained quadratic programs as special cases. Several decision problems in engineering design and control can be formulated as SOCP. What's*Best!* solves this class of problems using the so-called *conic optimizer*, which uses an interior-point algorithm. To solve a convex problem using What's*Best!*, it may be advantageous to cast the problem (e.g. a QCQP) as a SOCP and use the conic optimizer.

To motivate the second-order cone problems and common forms of quadratic cones, consider the following two constraints:

x^2 + *y*^2 – *z*^2 <= 0,

z >= 0

Geometrically, the feasible region defined by these two constraints is an *ice cream cone*, with the point of the cone at (0,0,0). The feasible region for the constraint *x*^2 + *y*2 ^- *z*^2 <= 0 by itself is not convex. The feasible region consists of two ice cream cones, one right side up, the other upside down, and with their pointy ends touching. The constraint *z* >= 0 eliminates the upside down cone and leaves the *quadratic cone* illustrated in Figure 1. Second-order cone problems are essentially a generalization of linear models defined over polyhedral cones to ones defined over quadratic cones.



More generally, in *n* dimensions, a simple quadratic cone (ice-cream cone) constraint is of the form:

-*x*0^2 + *x*1^2 + *x*2^2 +… + *xn*^2 <= 0;

*x*0 >= 0;

Second-order cone constraints are more general than they might at first appear. For another conic form, consider the constraints:

-*uv* + *x*^2 <= 0,

u, *v* >= 0.

The first constraint by itself describes a nonconvex feasible region (colored blue and green) illustrated in Figure 2. The three constraints together, however, describe a convex feasible region (colored green only) called the *rotated quadratic cone*.

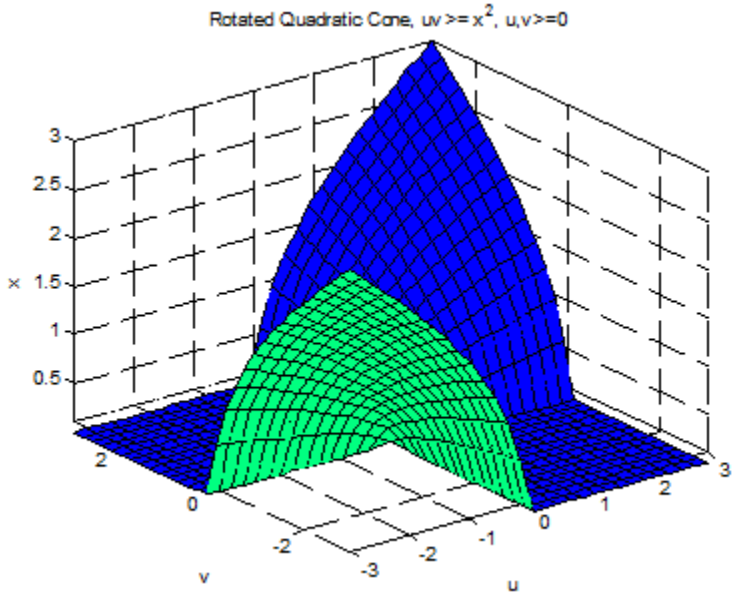


Figure 2. Rotated Quadratic Cone

More generally, in *n* dimensions, the rotated quadratic cone constraint in standard form is:

-2*x*0 *x*1 + *x*2^2 + *x*3^2 +… + *xn*^2 <= 0;

*x*0, *x*1>= 0;

In both simple and rotated quadratic cones, a variable can appear in at most one cone constraint. If naturally you would like to have a variable, say *x*2, appear in two cone constraints, then you must introduce an extra copy of the variable, say *y*2 , for the second cone constraint and then connect the two with the linear constraint *x*2 - *y*2 = 0.

Notice, using a standard transformation, rotated quadratic cone constraints can be shown to be equivalent to quadratic cone constraints:

$y = (u - v)/2$,

$z = (u + v)/2$,

$x^2 + y^2 - z^2 <= 0$,

$z >= 0$.

7 Sample Models

Introduction

This section provides sample models for problems in manufacturing, finance, engineering, scheduling, etc. They are real-world problems that demonstrate the practical use of the elegant concepts of linear and nonlinear programming.

A major strength and advantage of the spreadsheet environment compared to the more traditional methods of mathematical modeling is the ease with which relationships among data are visualized.

We've assembled a wide range of sample spreadsheet models for you to investigate. Some of them are very nearly full-fledged applications. All of them clearly demonstrate the principles of modeling with What's*Best!*. We've organized them in the following groups.

These models are located in the default folder after installation: "C:\WB"

Blending Models

Blending - Blending elements with various qualities into the lowest-cost product to meet designated quality requirements (HOGFEED.XLSX).

Chance-constrained Blending - As above, but quality content in the constituent elements varies at random, making the model nonlinear (HOGCHANC.XLSX).

Engineering Models

Box Design - A simple nonlinear model that finds dimensions for a cabinet to meet various design requirements (BOX.XLSX).

Flow Network Modeling - Calculating flow and pressure across a complex network (FLOWNET.XLSX).

Financial Models

Bond Portfolio Optimization - A multi-period model that recommends bond purchases to minimize costs while providing a specified cash flow (BONDS.XLSX).

Lockbox Location - Locating postal lockboxes to minimize "float" while still serving all customers (LOCKBOX.XLSX).

Markowitz Portfolio Problem - Selecting assets to meet a desired return at minimum variance (MARKOWIT.XLSX).

Portfolio with Transaction Costs - Adjusting a portfolio of assets in such a way that desired return after broker's fees and other transaction costs is met with minimum variance (PORTCOST.XLSX).

Minimizing Downside Risk - Purchasing and maintaining a portfolio of assets, so as to minimize the risk of losing value (DNRISK.XLSX).

Portolio Scenario Model - Demonstrating the differences among minimizing three different measures of risk (PORTSCEN.XLSX).

Forecasting Models

Seasonal Sales Factoring - Determining the effect of seasonal factors on historical sales to improve forecasting (SEASON.XLSX).

Exponential Smoothing - Two models illustrating one technique for using historical data to predict future sales (SIMXPO.XLSX, SMOOTH.XLSX).

Linearization Option and Construction Cost Estimation Models

Linearization Option and Construction Cost Estimation – This model demonstrates the performance gains resulting from using the What's*Best!* linearization option, and illustrates a case of estimating construction costs (LINEARZ.XLSX).

Marketing Models

Stratified Sampling - Determining the least-costly polling sample likely to give reliable results (SAMPLEWB.XLSX).

Car Pricing - A nonlinear pricing model - illustrating a case in which sales of products are interdependent (PRICING.XLSX).

Media Buying - Purchasing advertising media space to meet an exposure target at minimum cost (MEDIA.XLSX).

Production Models

Multi-Period Inventory Management - Managing inventory across multiple periods to minimize holding costs while maintaining sufficient stock (INVENT.XLSX).

Product Mix - Using available resources to manufacture a mix of products that yield the highest profit (PRODMIX.XLSX, PMIXMAC.XLSM).

The Building Block Method - Combining production and shipping models into one large model, a common approach to problem solving (BLOCK.XLSX).

Waste Minimization in Stock Cutting - Cutting sheet or coil materials to varying lengths while minimizing waste (CUTSTOCK.XLSX).

Plant Locating - Demonstrating how to locate plants or warehouse facilities to minimize shipping expenses while meeting demand (PLANTLOC.XLSX).

Staff Scheduling Models

Staff Scheduling - Meeting personnel needs at minimum cost (STAFF.XLSX).

Staff Sched.: Preferred Assignment - Covering staff needs while meeting employee preferences for job or shift assignments (ASSIGN.XLSX).

Staff Sched.: Two Stage Fixed Shift - Meeting two objectives, in this case minimizing cost and maximizing employee satisfaction, in scheduling (FIXED1.XLSX, FIXED2.XLSX).

Transportation Models

Pipeline Optimization - Moving resources along routes with limited capacities at minimum expense (PIPELINE.XLSX).

Shipping Cost Reduction - Minimizing shipping costs on routes with fixed costs while meeting demand (SHIPPING.XLSX).

Traffic Congestion Cost Minimization - Minimizing shipping costs on a network whose routes have costs that vary with the amount of traffic (TRAFFIC.XLSX).

Truck Loading - Packing a container with objects of varying sizes to maximize efficiency, a "knapsack" problem (TRUCK.XLSX).

Stochastic Models

Newsvendor Stock - A stochastic model to maximize net profit by deciding how much stock to hold, over 2 stages and 1 random parameter for the demand (SP_NEWSVENDOR.XLSX).

Investment Planning College - A multi-period stochastic model planning investments for going to college, after 3 Periods. (SP_INVESTMENTCOLLEGE.XLSX).

Crop Allocation Under Uncertainty - A stochastic model to maximize revenue by allocating crops to grow on a specified land area, over 2 stages and 1 random parameter for the weather condition (SP_CROPALLOC.XLSX).

Put Option - A stochastic model to maximize wealth by selling an option at the right time, over a multi-stage structure and random parameters for the stock returns (SP_PUTOPTION.XLSX).

Models Showing Features

Integer Options for K-Best/Knapsack: Knapsack_KBest.xlsx, see the K-Best Solutions feature.

Function Support: Shipping_MacroFunctions.XLSM, see the Function Support option.

Chance Constrained in Stochastic Models: SP_FeederMixCCP.xlsx, see the Page Chance-Constrained feature.

SUMIF function: NetFloSimple.xlsx, see the SUMIF usage restrictions.

String Support: VehicleRouting_String.xlsx, see the String Support option.

LOOKUP functions: VlookupWBSolver.xlsx, see the LOOKUP usage restrictions.

Constraint POSD-SDP: WBPOSDcase.xlsx, see the Semi-Definite Positive feature.

WBALLDIFF function: WBALLDIFF_Sudoku.xlsb, see the All-Different feature.

WBCARD function: WBCARD-PRODMIX.xlsx, see the Cardinality function.

WBERLANG_B function: WBERLANGB-Queue.xlsx, see the ERLANG_B function.

WBERLANG_C function: QueueNservers.xlsx, see the ERLANG_C function.

WBINNERPRODUCT function: WBINNERPRODUCT-PRODMIX.xlsx, see the Inner Product function.

WBLMTD function: HeatXchange_WBLMTD.xlsx, see the Logarithmic Mean Temperature Difference function.

WBSEMIC function: AstroCosSemic.xls, see the Semi-Continuous function.

WBSIGNPOWER function: NetEqlxSignPower.xlsx, see the Sign Power function.

WBSOS2 function: PieceWiseSOS2a.xlsx, see the Special Ordered Set function.

WBSOS3 function: WBSOS3-Assignment.xlsx, see the Special Ordered Set function.

WBQUADPRODUCT function: PortCorrVG_wbquadproduct.xlsx, see the Quad Product function.

Tutorial models for the ABC case: XYZ.xlsx, and via the VBA code :XYZVBA.xlsm.

Tutorial model for the user progress callback feature: CustomVBAInterface.xlsm

Blending

File name: HOGFEED.XLSX *TYPE: LINEAR OPTIMIZATION*

Application Profile

The manufacture of animal feed is an example of a blending problem. The question is: how do you mix a number of raw components into a minimum cost final blend that has specified amounts of certain ingredients?

In the case of hogfeed, the issue is vital. How can raw feedstocks be combined to produce a high nutrition hogfeed at lowest cost? For someone with several thousand hungry hogs, the quality of this answer can easily spell the difference between profit and loss. Since feedstock prices are typically volatile, the complexity of the question and the daily importance of reliable solutions is even more apparent.

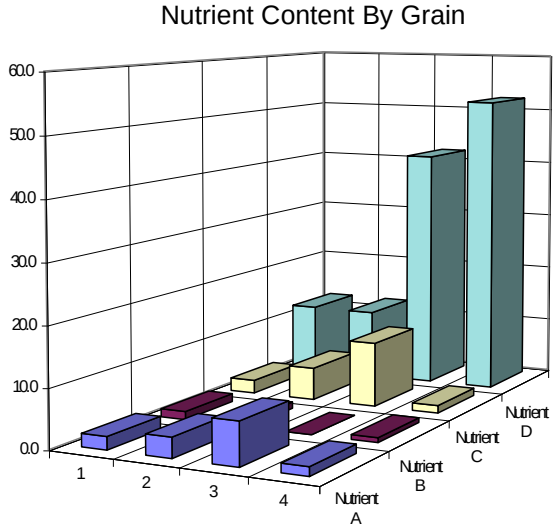
Blending problems are also of interest well beyond the field of animal husbandry. Examples include finding the least costly mix of ores that will produce an alloy with specified characteristics or deciding on the most efficient combination of advertising media purchases to reach target audience percentages in multiple demographic groups.

The Problem in Words

You are the manager of the Swine & Roses (S&R) Hogfarm and your hogfeed must meet certain minimum requirements for four nutrients, *A*, *B*, *C*, and *D*, to assure fast growth and large, robust animals.

Background

The nutrients are found in four different grains. Each grain contains a different combination with a price that fluctuates with market conditions. The nutrient content by grain is shown in the following graph.



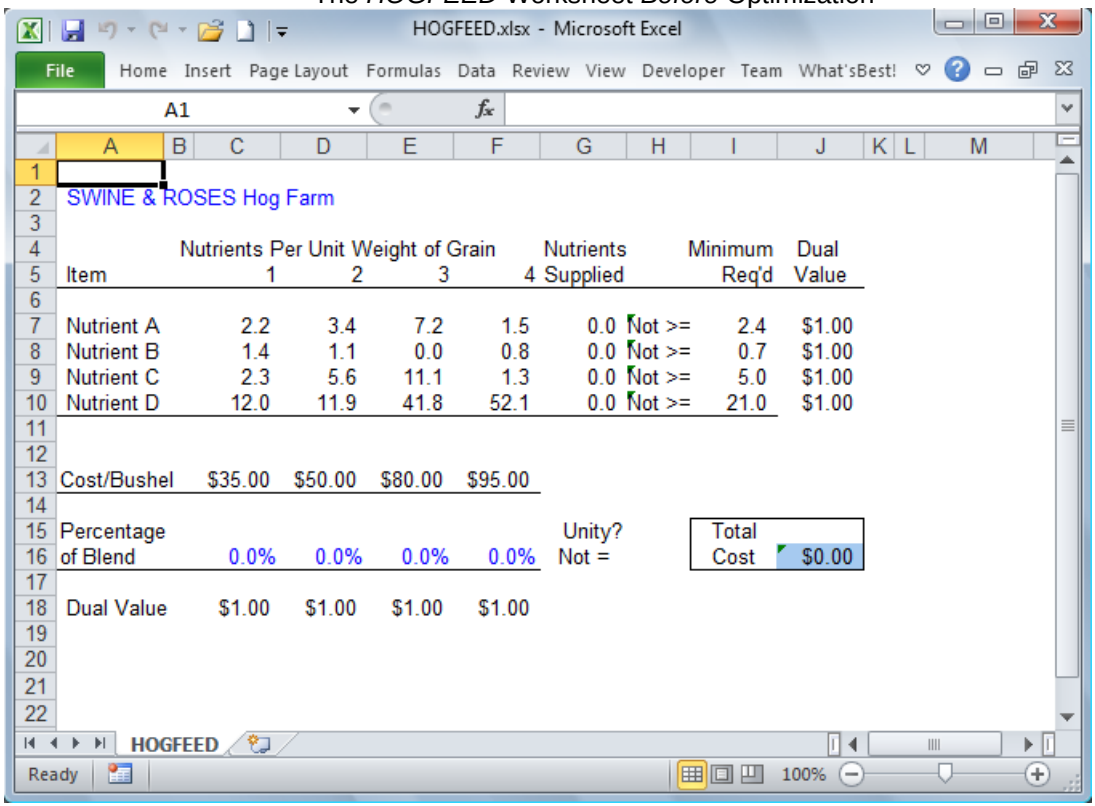
Objective of Optimization

The objective is to determine how much of each grain S&R should buy at today's prices to meet their nutritional requirements at lowest cost. This always has a chance of being wrong if price fluctuates unexpectedly.

The Worksheet

Let's open the *HOGFEED* sample file and look over the ABC's.

The *HOGFEED* Worksheet *Before* Optimization



A. Determine Adjustable Cells

The cells What's*Best!* is free to change in this model are the *Percentage of Blend* contributed by each grain (C16:F16).

B. Define Best

The best solution to this problem meets nutritional needs at the lowest cost. *Total Cost*, J16, contains the formula *SUMPRODUCT*(C16:F16,C13:F13). That's the *Percentage of Blend* times the *Cost Per Bushel* for each grain.

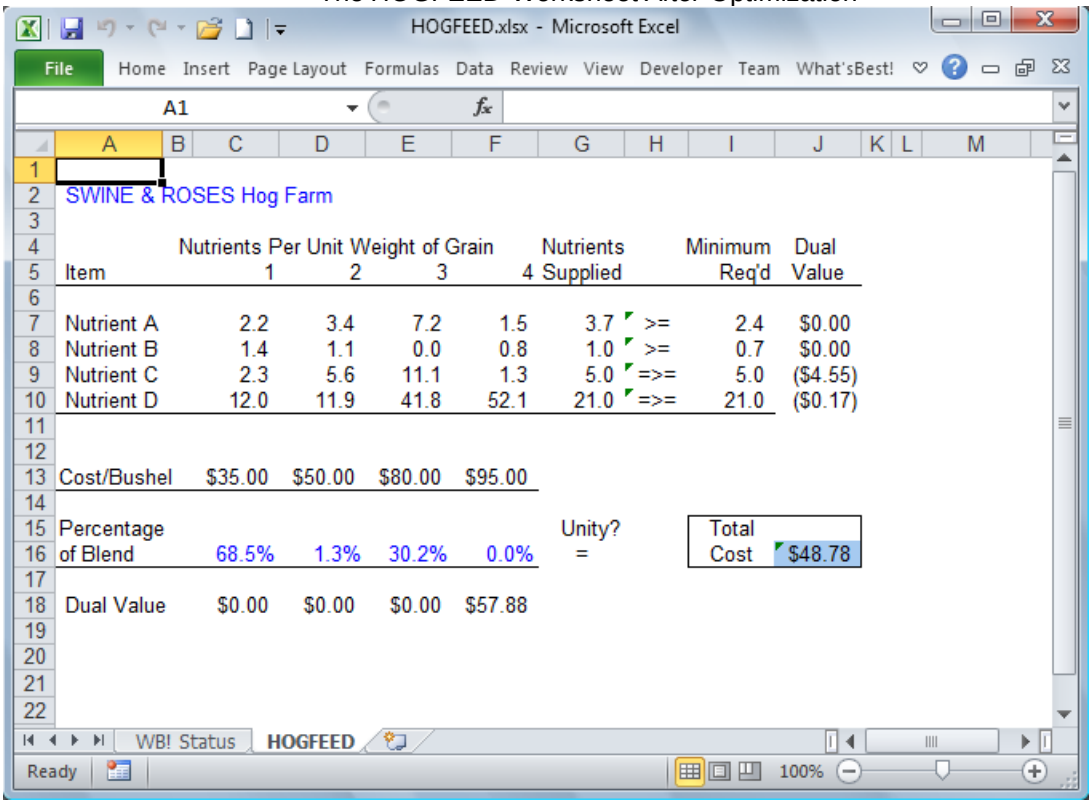
C. Specify Constraints

There are two limitations on the solution to this problem. First, the final mix must contain the minimum required levels of nutrients. This is enforced by a constraint for each nutrient (H7:H10). Each constraint will return the "Not >=" indicator until the minimum requirement (I7:I10) for that nutrient is met in the *Nutrients Supplied* column (G7:G10). Second, the constraint in G16 forces the percentages in C16:F16 to sum to 100%.

The *Nutrients Supplied* cell for *Nutrient A* (G7) contains *SUMPRODUCT*(C7:F7,$C$16:$F$16), which sums the product of the *Percentage of Blend* and the amount of *Nutrient A* for each grain. Each grain contains different combinations of each nutrient (C7:F10).

Now, let's solve the model with What's*Best!* to find the optimal solution.

The *HOGFEED* Worksheet *After* Optimization



The What's*Best!* solution has a total cost of $48.78 per bushel of the optimal blend. Grain 4, which is not a bargain at $95 per bushel, is not purchased.

D. Dual Values

Dual values have been included in two locations in this worksheet (C18:F18 and J7:J10).

The dual value entries in cells J7 through J10 show the reward (i.e., how much money could be saved) if each respective constraint (H7:H10) were relaxed by one unit. In practical terms, this means that you could save $4.55 per bushel if the requirement for *Nutrient C* were reduced to 4. If slightly less sprightly hogs are tolerable in return for such a savings, relaxing this nutritional requirement would be a good decision. Likewise, if the requirement for *Nutrient D* were reduced to 20, the cost/bushel would go down by $.17. Note that the dual values are zero for *Nutrients A* and *B*. Lowering the requirement by one unit offers no savings because there is already an excess of *Nutrients A* and *B* in your optimum blend, since constraints H7 and H8 are not tightly satisfied.

The other dual values in the *Swine & Roses* model (C18:F18) show the amount by which the price of an unused grain would have to be reduced before it would be cost effective to use it in the minimum cost blend. For example, the dual value for *Grain 4* is $57.88 (F18). This tells you how much the price of *Grain 4* would have to be reduced in order for it to appear in the optimal solution of *HOGFEED*. This is a good number to have handy when negotiating to get a lower price. For more information on dual values, see the section on *Dual Values* in Chapter 3, *Additional Commands*.

Chance-Constrained Blending

File name: HOGCHANC.XLSX *TYPE: NONLINEAR OPTIMIZATION*

Application Profile

In the previous *HOGFEED* blending example, we showed you a deterministic model — the nutrient content of the different grains to be incorporated in the finished feed were known and constant. However, what if the nutrient content is not so reliable? What if the content varies at random? This model is nonlinear and perhaps more likely to occur in the real world.

The Problem In Words

You must determine how much of four available grains to include in blended feed (in circumstances under which the content of one of the nutrients varies at random) to meet nutrient requirements with some degree of certainty.

Background

You've had several samples of the four grains tested and you find that their content of *Nutrient D* varies randomly and is normally distributed. You want to find a new blend that ensures the minimum requirements for *Nutrient D* are met at least 95% of the time.

You've calculated the mean and variance for each grain (the content means are the same as in the original *HOGFEED* model). The equation that calculates *Nutrient D* now looks like this:

(*Mean Nutrient D in the blend*) - *Z**(*standard deviation Nutrient D*) >= 21.

For 95% confidence in the content of *Nutrient D*, set *Z* = 1.645 (Refer to any elementary text on statistics for a discussion of *Z* values).

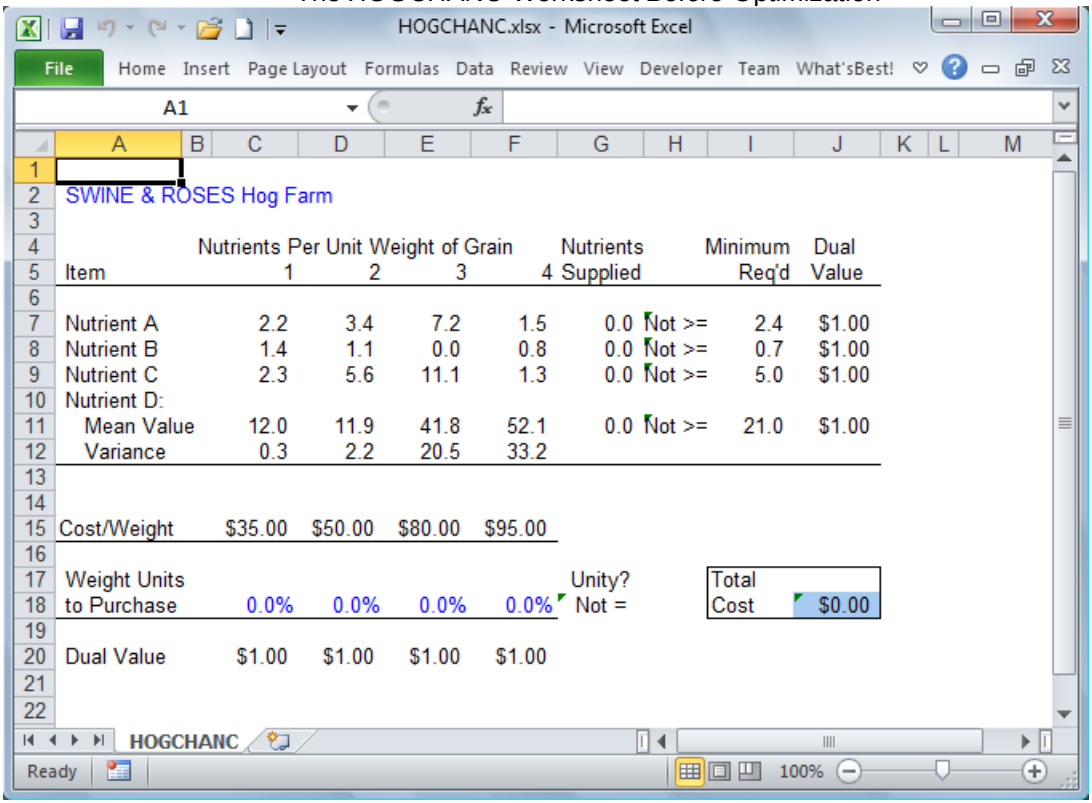
Objective of Optimization

The objective is to determine how much of each grain you should buy at today's prices to meet their nutritional requirements at lowest cost.

The Worksheet

Let's look at the *HOGCHANC* worksheet as supplied in your sample files.

The *HOGCHANC* Worksheet *Before* Optimization



A. Determine Adjustable Cells

The cells What's*Best!* is free to manipulate in this model are the *Weight Unit Percentages to Purchase* for each grain (C18:F18).

B. Define Best

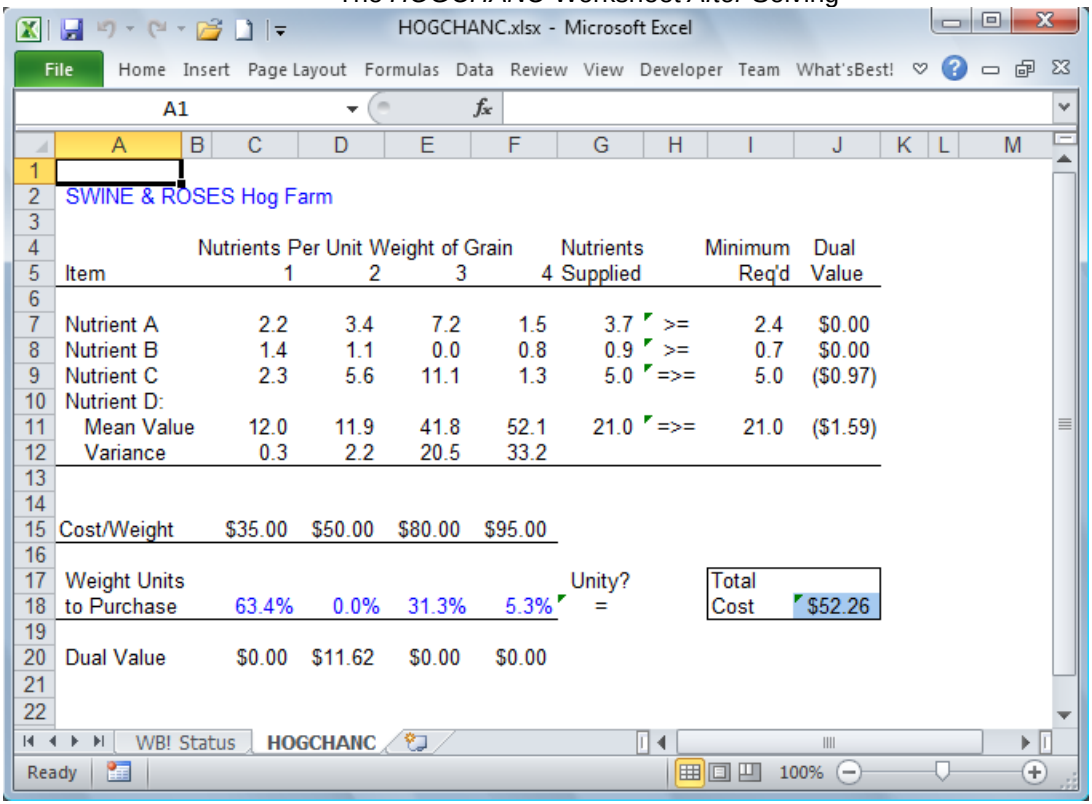
The optimal solution to this problem is the one that meets needs at lowest cost. If you examine the formula in the *Total Cost* cell (J18), *SUMPRODUCT*(C18:F18,C15:F15), you see that every change in price or purchase quantities has a direct bearing on cost. What's*Best!*, of course, is only free to change the purchase quantities in C18:F18.

C. Specify Constraints

The limitation to this problem is that the final mix must contain the minimum required levels of nutrients for Swine & Rose's hogs. This limitation is enforced by creating a constraint cell for each nutrient (H7:H9, H11). Each constraint will return the "Not >=" indicator until the *Minimum Required* (I7:I9, I11) for that nutrient is met in the *Nutrients Supplied* column (G7:G9, G11).

Now, let's solve the problem with What's*Best!*.

The *HOGCHANC* Worksheet *After* Solving



This What's*Best!* solution costs $52.26 per bushel. It's more expensive than the *HOGFEED* solution in the previous example because you have compensated here for the variability in the amount of *Nutrient D* in the various component grains. It's now necessary to use the relatively more expensive *Grain 4*, rich in *Nutrient D*, to reach a 95% confidence in fulfilling all your nutritional requirements.

D. Dual Values

Dual values are included in two locations in this worksheet (C20:F20 and J7:J9,J11).

The dual values in cells J7:J9,J11 show the shadow prices or reward (how much money you would save) if each respective requirement (I7:I9, I11) is relaxed by one unit. This means, for example, that you would save $.97 per bushel if the requirement for *Nutrient C* is reduced to 4. A reduction to 4.9 saves you $0.97. If slightly less sprightly hogs are tolerable in return for such a saving, relaxing this nutritional requirement would be a good decision. Note that the dual values are zero for *Nutrients A* and *B*. Lowering the requirement by one unit offers no savings because there is already an excess of *Nutrients A* and *B* in your optimum blend since constraints H7 and H8 are not tightly satisfied.

The other dual values in the Swine & Roses model (C20:F20) show the amount by which the price of an unused grain would have to be reduced before it would be cost effective to use it in the minimum cost blend. For example, the dual value for *Grain 2* is $11.62 (D20). This tells you how much the price

of *Grain 2* would have to be reduced in order for it to appear in the optimal solution of *HOGCHANC*. This is a good number to have handy when negotiating to get a lower price.

Note: In nonlinear models, the ranges over which dual values are valid may be very small. Before basing a pricing or purchasing decision on a dual price or reduced cost in a nonlinear model, you should test the returned value by making the specified change to the variable or constraint, using *Adjustable|Remove Adjustable* on the variable, and re-solving.

Box Design

File name: BOX.XLSX *TYPE: NONLINEAR OPTIMIZATION*

The Problem in Words

As a manufacturer of electronic equipment, you must design a cabinet for your new product that meets the specifications of various departments within your organization at minimum cost.

Background

Your engineering department has determined that the equipment requires a volume of at least 1512 cubic inches and that a minimum surface area of 888 square inches will suffice to dissipate heat.

Marketing will find it easiest to sell the finished device if the footprint of the cabinet is no more than 252 square inches.

Finally, your designers have decreed that, for aesthetic reasons, the ratio of height to width should be .618 ± .1. That is, between .518 and .718, inclusively.

The sheet metal from which the cabinet will be made costs $.05 per square inch. Extra labor required on the front and back panels raises the cost for these components to $.10 per square inch.

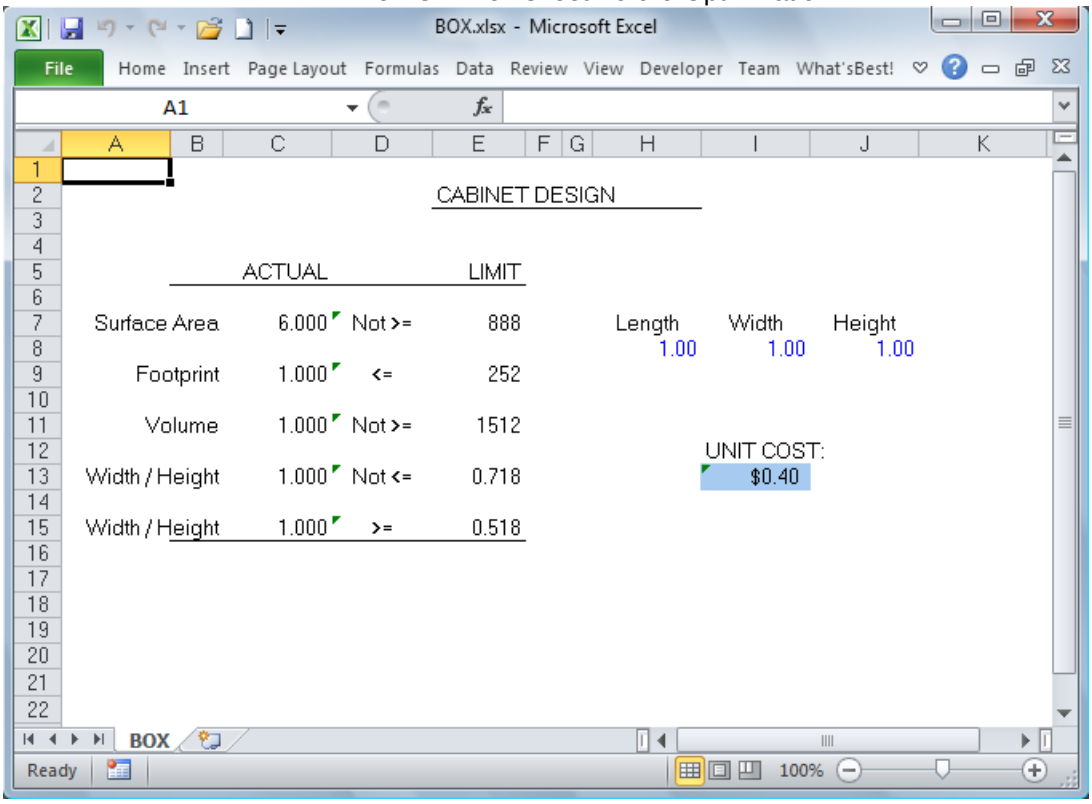
Objective of Optimization

The objective is to determine the specifications of the box with the smallest production cost.

The Worksheet

Let's open the *BOX* sample file. Let's look at how the requirements are translated to the spreadsheet format.

The *BOX* Worksheet *Before* Optimization



A. Determine Adjustable Cells

The adjustable cells in the BOX model are the *Length*, *Width*, and *Height* in H8:J8. These cells are further identified as ranges *L*, *W*, and *H*, respectively.

B. Define Best

The best solution is the minimum cost cabinet that satisfies all the constraints. This objective is calculated in cell I13 by means of the following formula:

2*(0.05*(*Length***Width*+*Length***Height*)+0.1*(*Height***Width*))

The interpretation of which is: The cost of the side/top/bottom panels ***** the sum of the areas of one side and one top/bottom panel **+** the cost of front/back panels ***** the area of a front/back panel all multiplied by 2.

C. Specify Constraints

The constraints are enforced by means of the formulas in D7, D9, D11, D13, and D15.

In D7, the formula *WB*(C7,">=",E7) forces the total *Surface Area* of the cabinet in C7, calculated by the formula 2*(*Length***Width*+*Length***Height*+*Width***Height*), to be at least as great as the lower limit of 888 square inches in E7.

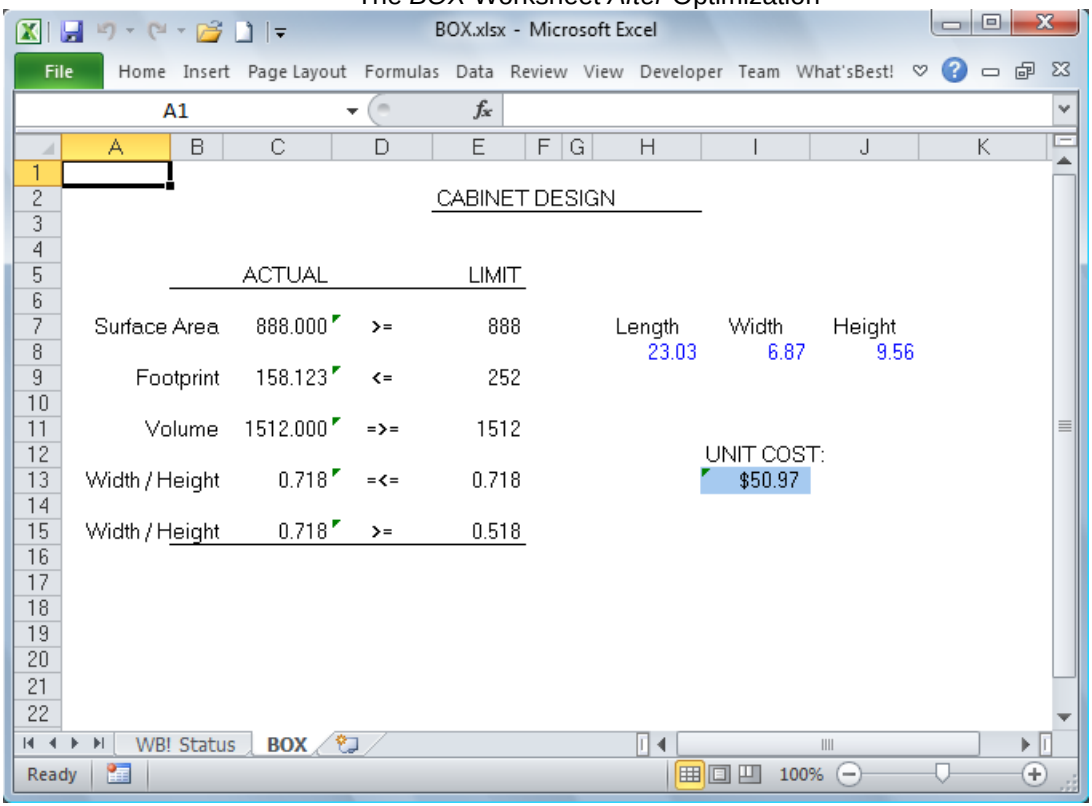
In D9, the formula *WB*(C9,"<=",E9) forces the *Footprint* in C9 to be no greater-than the upper limit of 252 square inches in E9. The *Footprint* is calculated by the formula *Width***Length*.

In D11, the formula *WB*(C11,">=", E11) forces the *Volume* in C11, calculated by the formula *Length* * *Width* * *Height*, to be at least as great as the lower limit of 1512 cubic inches required to fit the electronic equipment in the cabinet.

Finally, in D13 and D15, the formulas *WB*(C13,"<=", E13) and *WB*(C15, ">=", E15) force the ratio of *Height* to *Width* to fall between .718 and .518, where C13 and C15 both calculate *Width*/*Height*.

Now, you're ready to solve the model and find the optimum design.

The *BOX* Worksheet *After* Optimization



The resulting 23.03 x 6.87 x 9.56 inch box is the lowest-cost design ($50.97 per unit) that meets all the requirements of your various departments.

Flow Network Modeling

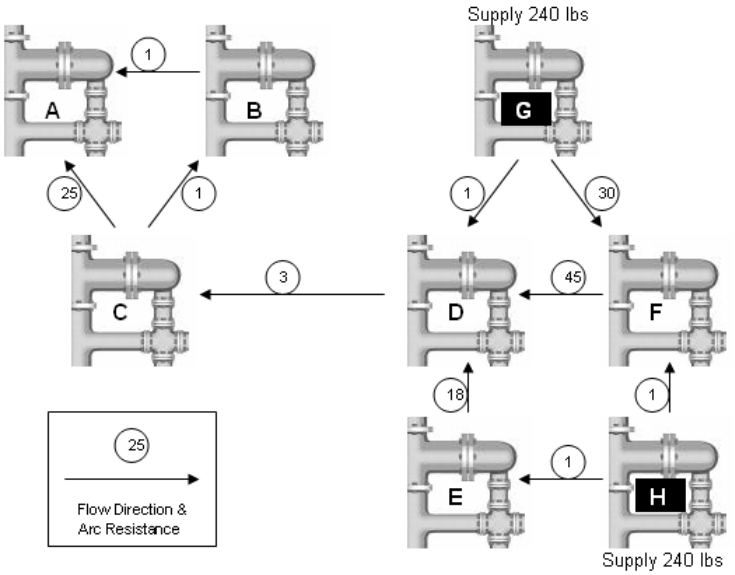
File name: FLOWNET.XLSX *TYPE: NONLINEAR*

The Problem in Words

You are the manager of a network of pipelines and you need to determine a balance of flows along each arc of the network while satisfying demand at the nodes.

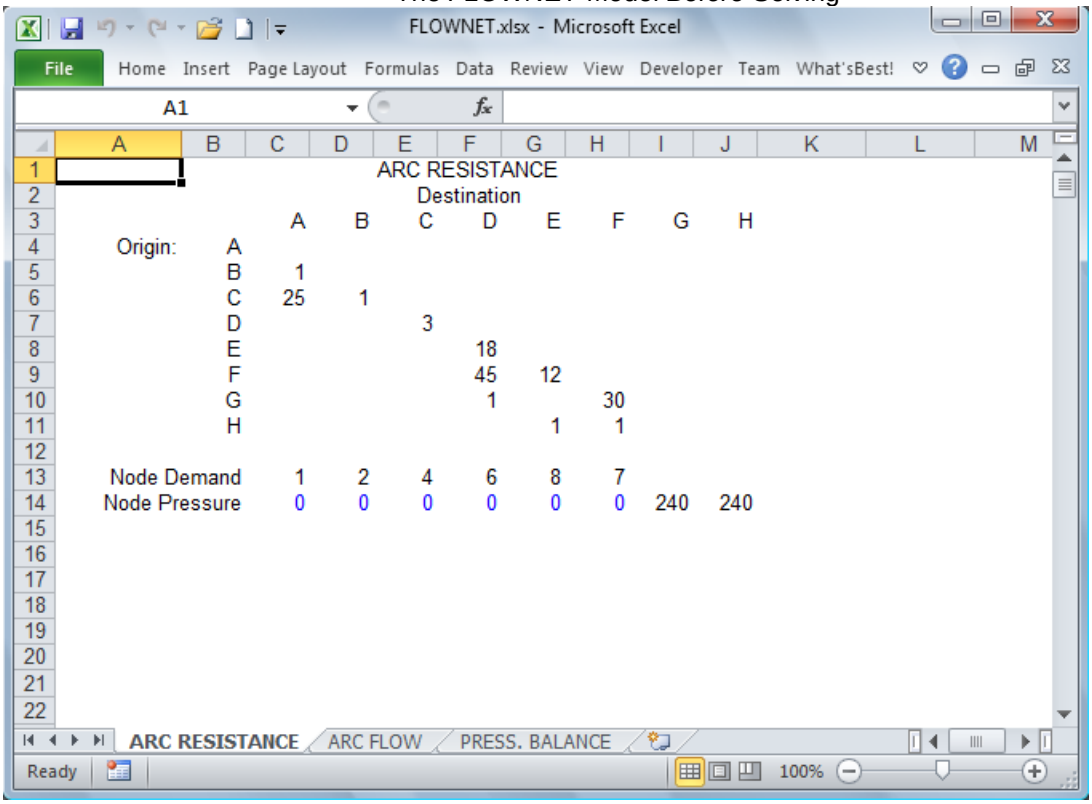
Background

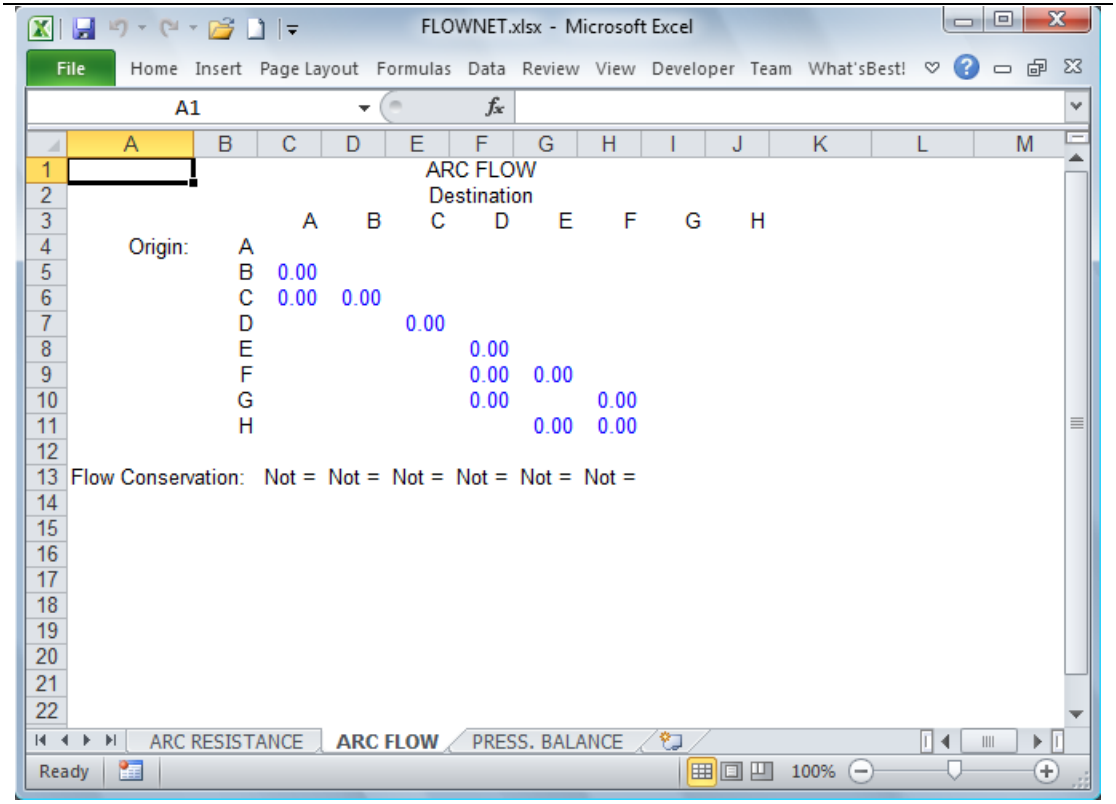
In the *FLOWNET* model, you have two supply nodes with connections to a total of six other nodes. The pressure at the supply nodes *G* and *H* is fixed at 240 lbs/square inch (PSI). There's a known demand at each of the six other nodes in C13:H13 of the *Arc Resistance* worksheet. You need to calculate the pressure at each node and the flow across each arc. The following diagram helps visualize this.

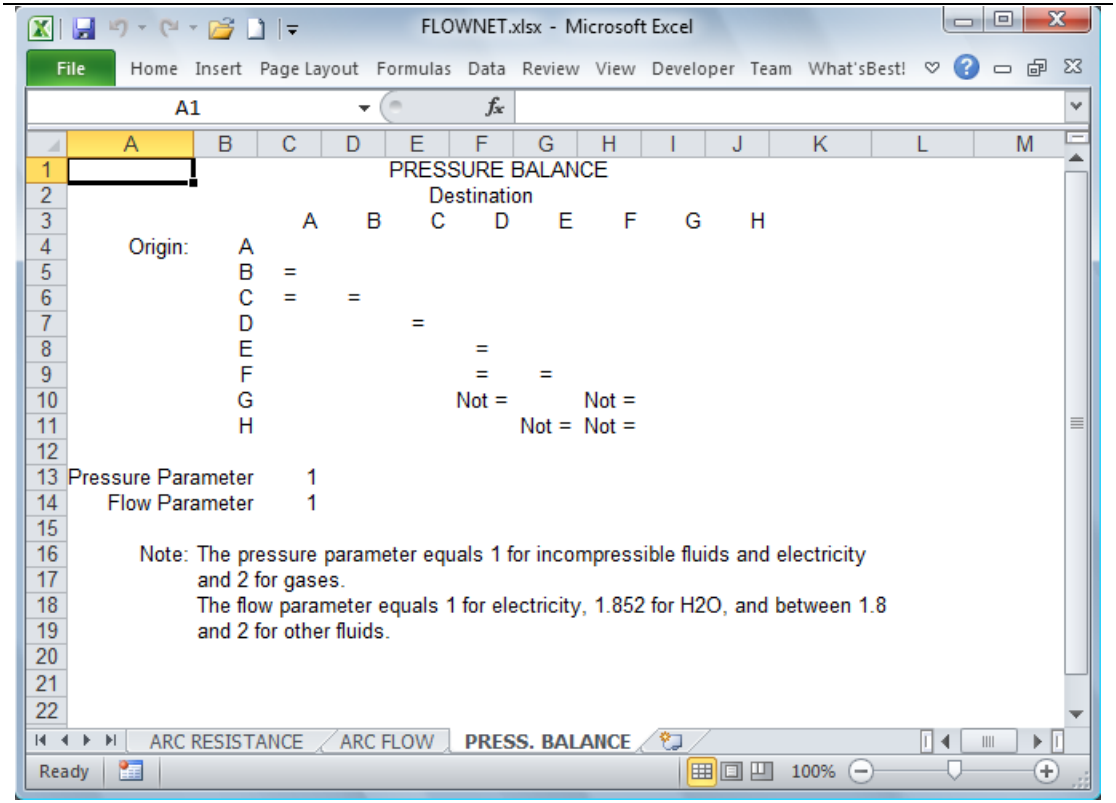


The Worksheet

The *FLOWNET* Model *Before* Solving




FLOWNET.xlsx - Microsoft Excel
File Home Insert Page Layout Formulas Data Review View Developer Team What'sBest!
A1
ARC FLOW
Destination
A B C D E F G H
Origin:
A
B 0.00
C 0.00 0.00
D 0.00
E 0.00
F 0.00 0.00
G 0.00 0.00
H 0.00 0.00
Flow Conservation: Not = Not = Not = Not = Not = Not =
ARC RESISTANCE
ARC FLOW
PRESS. BALANCE
Ready
100%



A. Determine Adjustable Cells

The adjustable cells are the *Pressures* at each *Node* (*Arc Resistance* C15:H15) and the *Flow* along each arc (the cells outlined above in the range C4:J11 of the *Arc Flow* worksheet).

To facilitate modeling the problem, nonadjustable zeroes have been entered for arcs where there is no flow. This increases the number of numeric cells in the model. In your own models, you may want to minimize the number of numeric cells.

B. Define Best

No best cell is defined in this example, since our objective is simply to balance the flows along the arcs and satisfy demand at the nodes.

C. Specify Constraints

The constraints in this model are twofold. First, conservation of flow requires that the amount entering an arc must equal the amount exiting that arc (*Arc Flow* C13:H13). As an example, here's the formula in C13:

WB(*SUM*(C4:C11),"=", 'ARC RESISTANCE'! C13+*SUM*(C4:H4))

Second, you must account for pressure lost along each arc due to resistance. These constraints are found in cells C5:H11 of the *Pressure Balance* worksheet. The formula in C5, for instance, requires that pressure applied equals pressure lost on the *B*⇒*A* arc:

WB('ARC RESISTANCE'! D14^$C$13- 'ARC RESISTANCE'! C14 ^$C$13 ,
"=", 'ARC RESISTANCE'! C5* 'ARC FLOW'! C5^$C$14)

Disregarding the exponents, which in this case are 1, the formula is:

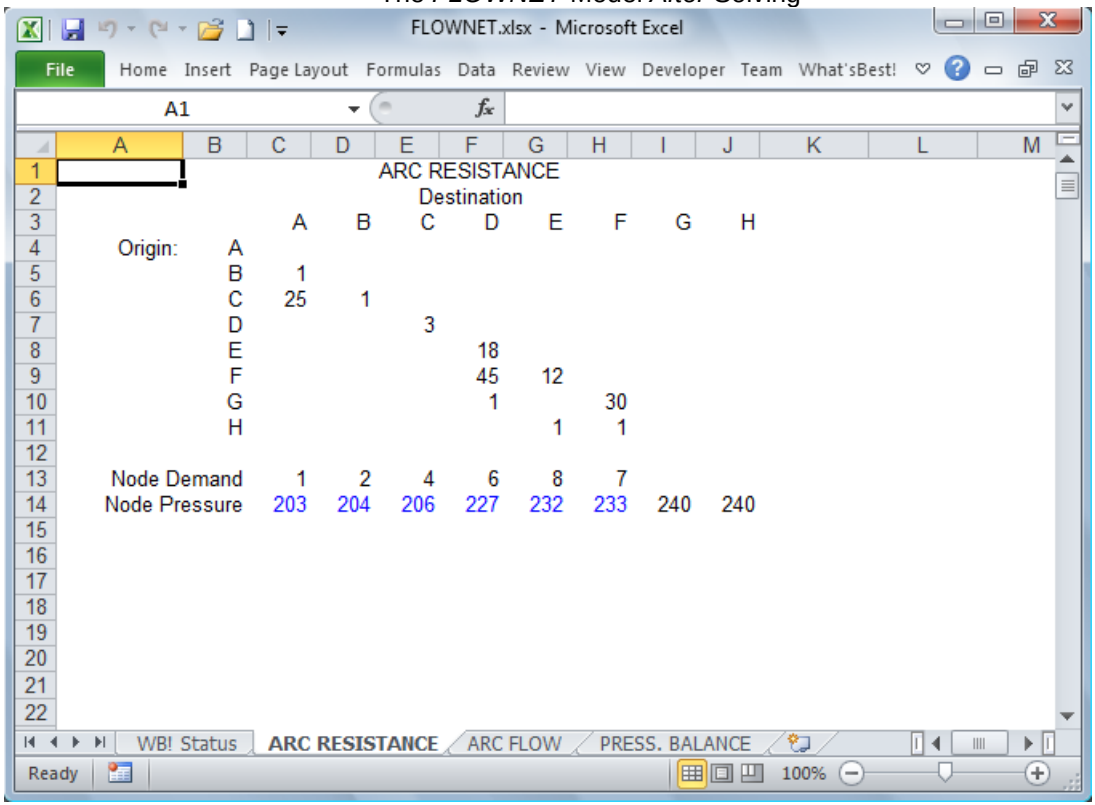
WB('ARC RESISTANCE'! D14- 'ARC RESISTANCE'! C14,"=",
'ARC RESISTANCE'! C5* 'ARC FLOW'! C5)

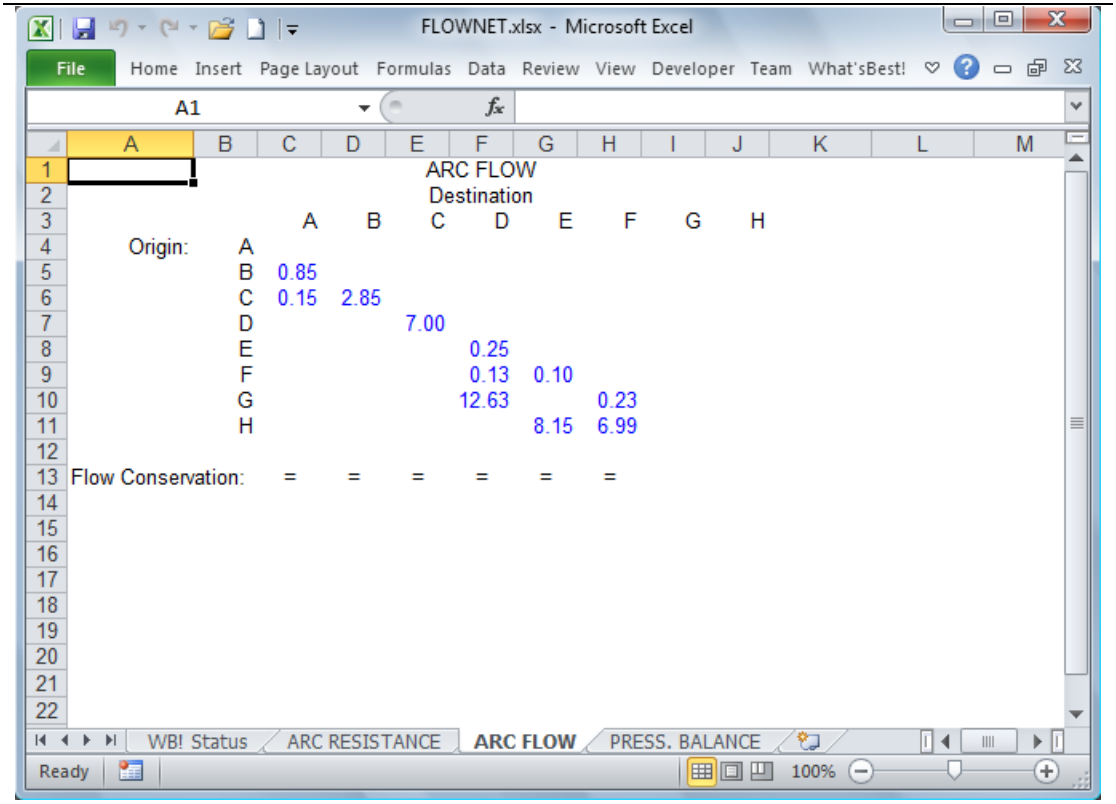
Thus, loss in pressure along arc *B*⇒*A* (pressure at *B* minus pressure at *A*) must equal resistance along arc *B*⇒*A* times flow along arc *B*⇒*A*.

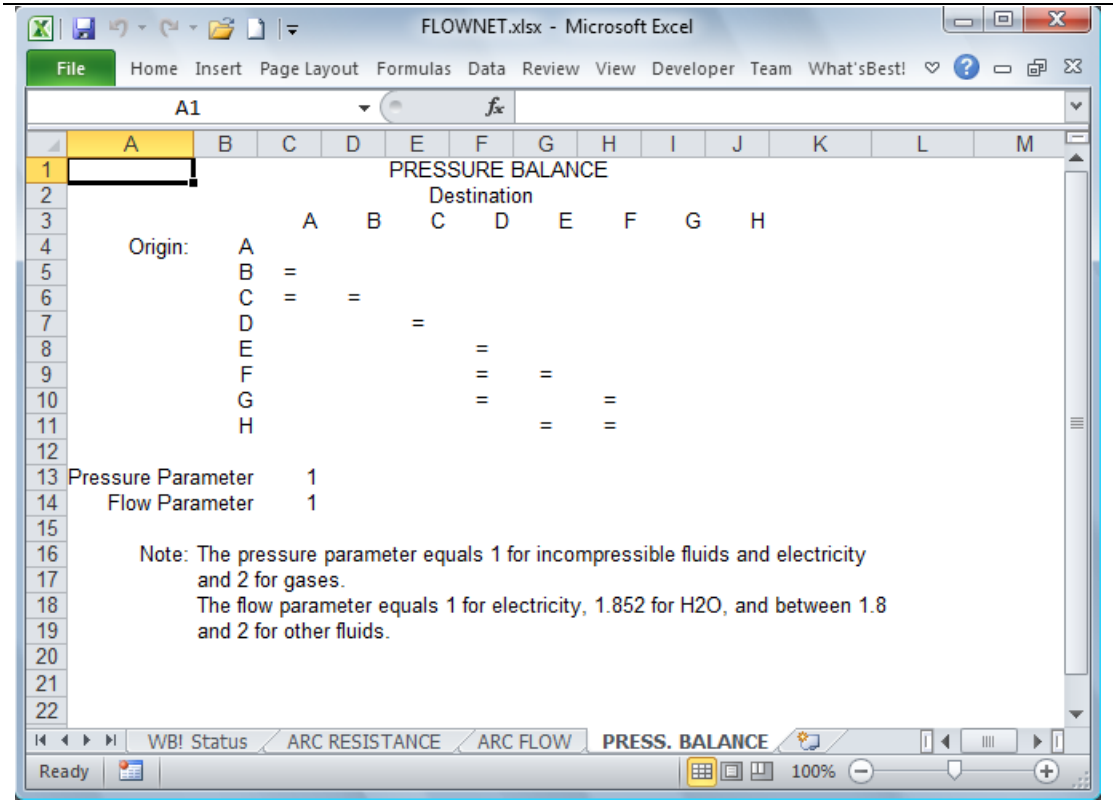
Note: The *Pressure* parameter is 1 for incompressible fluids and electricity and 2 for gases. The *Flow* parameter is 1 for electricity, 1.852 for H_2O, and between 1.8 and 2 for other fluids.

Now, you're ready to solve the model and determine the balance of flows. After solving, the *WB! Status* worksheet will open in order to show you the *Nonlinearity present* warning and *No Best Cell* warnings. These warnings may be turned off from the *General Options* dialog box. The three model worksheets now show the flows as shown below.

The *FLOWNET* Model *After* Solving




FLOWNET.xlsx - Microsoft Excel
File
Home
Insert
Page Layout
Formulas
Data
Review
View
Developer
Team
What'sBest!
A1
ARC FLOW
Destination
A
B
C
D
E
F
G
H
Origin:
A
B 0.85
C 0.15 2.85
D 7.00
E 0.25
F 0.13 0.10
G 12.63 0.23
H 8.15 6.99
Flow Conservation: = = = = = =
WB! Status
ARC RESISTANCE
ARC FLOW
PRESS. BALANCE
Ready
100%


FLOWNET.xlsx - Microsoft Excel
File Home Insert Page Layout Formulas Data Review View Developer Team What'sBest!
A1
PRESSURE BALANCE
Destination
A B C D E F G H
Origin:
A
B =
C = =
D =
E =
F = =
G = =
H = =
Pressure Parameter 1
Flow Parameter 1
Note: The pressure parameter equals 1 for incompressible fluids and electricity and 2 for gases.
The flow parameter equals 1 for electricity, 1.852 for H2O, and between 1.8 and 2 for other fluids.
WB! Status
ARC RESISTANCE
ARC FLOW
PRESS. BALANCE
Ready
100%

Bond Portfolio Optimization

File name: BONDS.XLSX *TYPE: LINEAR OPTIMIZATION FLOWNET*

Application Profile

This multi-period model is an example of dynamic modeling. "Dynamic" means that decisions made in this period affect not only this period's returns (or costs), but future decisions and returns as well. For this reason, multi-period problems cannot be treated as if they were merely a collection of single-period problems.

In many multi-period modeling problems, liquid or cash-like assets are treated like other commodities, so holding cash is just like keeping inventory. The following illustrates the major features of multi-period models in a financial context.

The Problem in Words

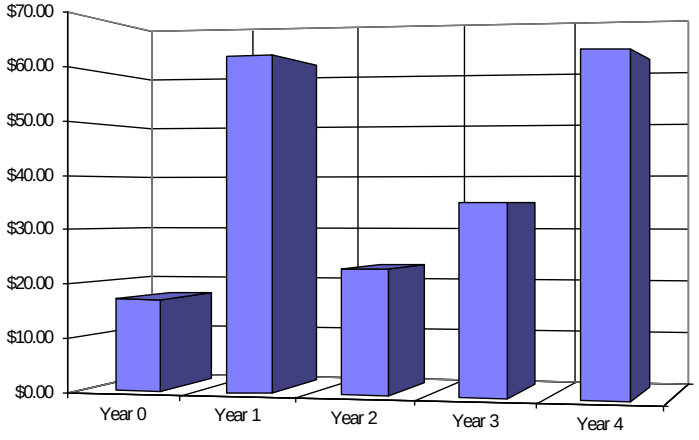
You're a financial advisor for an investment firm. Your client needs enough cash to cover commitments for the next five years. You've concluded that to meet these financial obligations, he should invest in some very low-risk securities such as government bonds. You must recommend bonds to buy to minimize his total cost now, yet cover his cash flow requirements. This is also an example of how to defease debt — how to wipe it off the books.

Background

Your client's specific cash flow needs are as follows:

Year	0	1	2	3	4
Cash Flow Need ($M)	$17.0	$62.0	$23.0	$35.0	$62.0

In graph form, the data look like this:



You've counseled your client to buy government bonds not only because of their low risk, but also because a bond owner receives a fixed set of cash payouts. For each year until the bond matures, your client receives an interest payment and, at maturity. He also gets back the face value or principal of the bond.

Generally, a broad spectrum of such investments are available on a given day. To keep it simple, let's assume you come across the following quotations for nine U.S. Treasury bonds:

Year to Maturity	Asking Price (Rate/1000)	Interest Rate (%)
1 Year	0.996	10.5
	0.997	10.5
2 Year	0.923	10.75
	0.987	11.2
3 Year	0.993	11.8
	1.061	12.0
	0.883	10.0
4 Year	1.102	12.6
	0.889	10.2

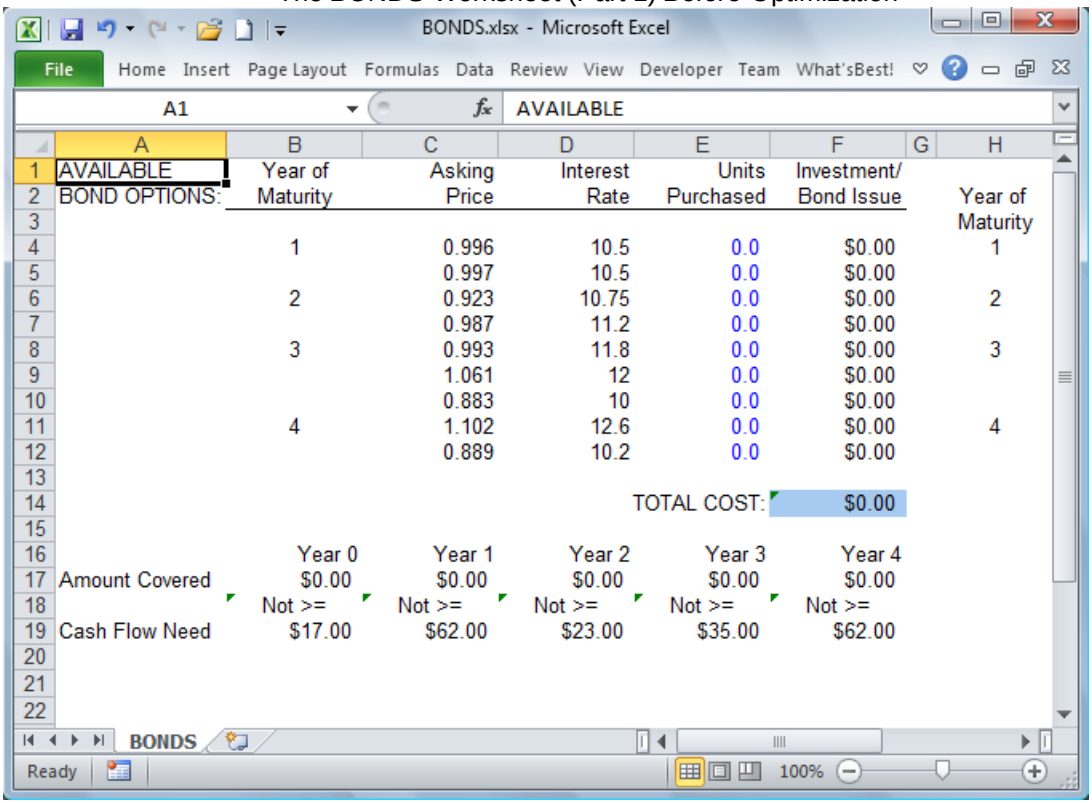
Objective of Optimization

You want to minimize your client's initial investment, while covering his cash flow needs over the next five years. In other words, you wish to know the minimum amount of cash that should be allocated among all the available bonds, which will maintain the required cash inflow over the next five years.

The Worksheet

To get started, let's examine the *BONDS* worksheet. Look it over for a minute to see how it's designed.

The *BONDS* Worksheet (Part 1) *Before* Optimization



	A	B	C	D	E	F	G	H
1	AVAILABLE	Year of	Asking	Interest	Units	Investment/		
2	BOND OPTIONS:	Maturity	Price	Rate	Purchased	Bond Issue		Year of Maturity
3								
4		1	0.996	10.5	0.0	$0.00		1
5			0.997	10.5	0.0	$0.00		
6		2	0.923	10.75	0.0	$0.00		2
7			0.987	11.2	0.0	$0.00		
8		3	0.993	11.8	0.0	$0.00		3
9			1.061	12	0.0	$0.00		
10			0.883	10	0.0	$0.00		
11		4	1.102	12.6	0.0	$0.00		4
12			0.889	10.2	0.0	$0.00		
13								
14					TOTAL COST:	$0.00		
15								
16		Year 0	Year 1	Year 2	Year 3	Year 4		
17	Amount Covered	$0.00	$0.00	$0.00	$0.00	$0.00		
18		Not >=	Not >=	Not >=	Not >=	Not >=		
19	Cash Flow Need	$17.00	$62.00	$23.00	$35.00	$62.00		
20								
21								
22								

A. Determine Adjustable Cells

Adjustable cells are the units of bonds purchased for each bond quoted (E4:E12).

B. Define Best

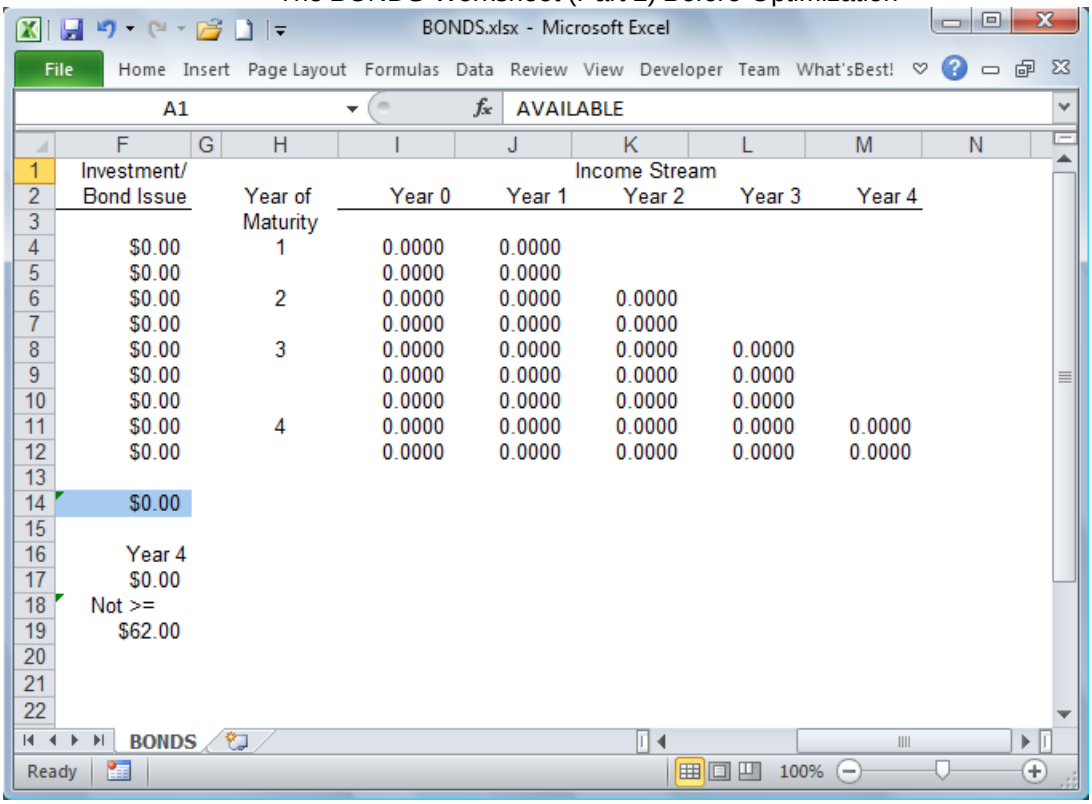
The Best possible solution is the particular mix of units of bonds purchased that result in minimum initial investment in cell F14. Examine the formula for the *Total Cost* cell: *SUM*(F4:F12). This formula sums the amount invested in each bond issued for each of the nine bonds offered.

C. Specify Constraints

The constraints in this problem (B18:F18) require that the total cash inflows in B17:F17 (i.e., from the interest and/or principal income each year) remain greater-than-or-equal-to cash outflows needed per period in B19:F19.

The cells B17:F17, *Amount Covered*, contains formulas that add the cash received from bond interest payments as well as the repaid principal (when applicable) for each year. This *Income Stream* is detailed in cells I4:M12 and appears as follows:

The *BONDS* Worksheet (Part 2) *Before* Optimization

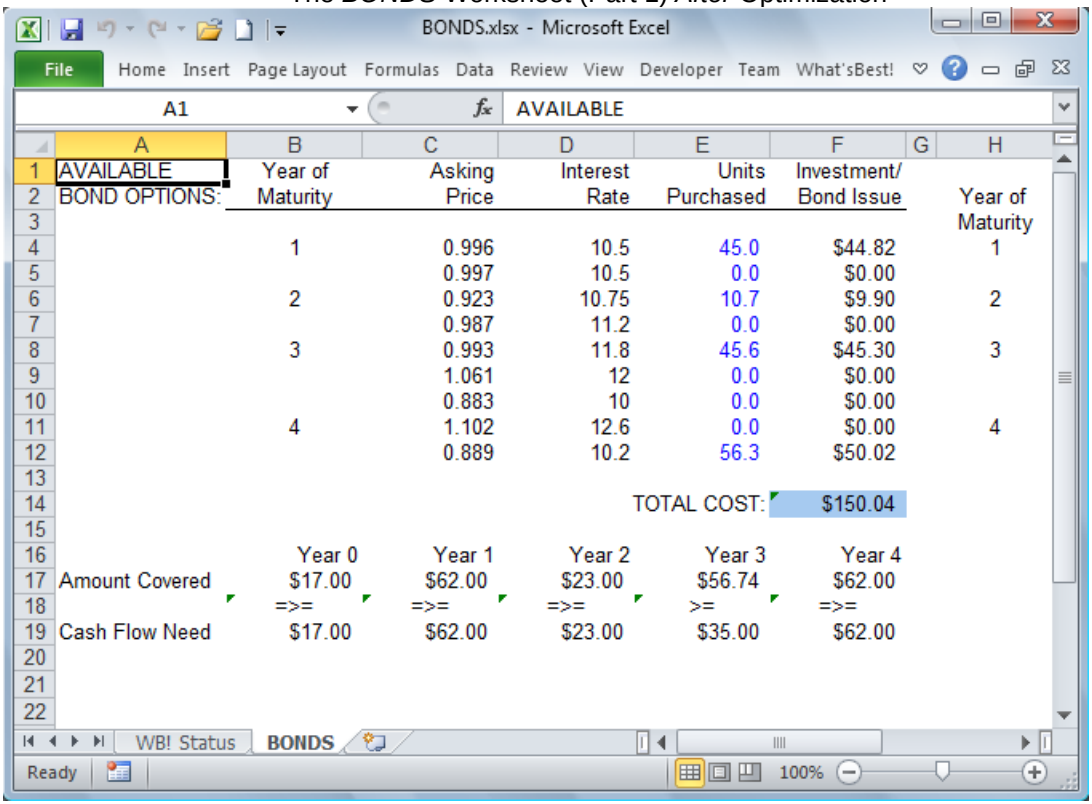


For instance, suppose that in the beginning of year 0 your client invests in 10 units of a 10 percent 3 year U.S. Treasury bond (see the bond listed in C10:F10). A "10.0" would then be inserted in cell E10. Each unit represents a $1000 face value or principal.

Thus, in the current year, your client will receive an interest payment of $1.00, or 10 units * .10 rate of interest, as calculated in cell I10 (E10*D10/100). Next year and the year after, he will continue to receive interest payments of $1.00 (J10 and K10). In three years, when the bond matures, he will receive another interest payment of $1.00 and additionally a repayment of the principal of 10 units, for a total of $11.00 as calculated in cell L10 (E10*(1+D10/100)).

You're now ready to solve the model and find the best possible solution to this problem.

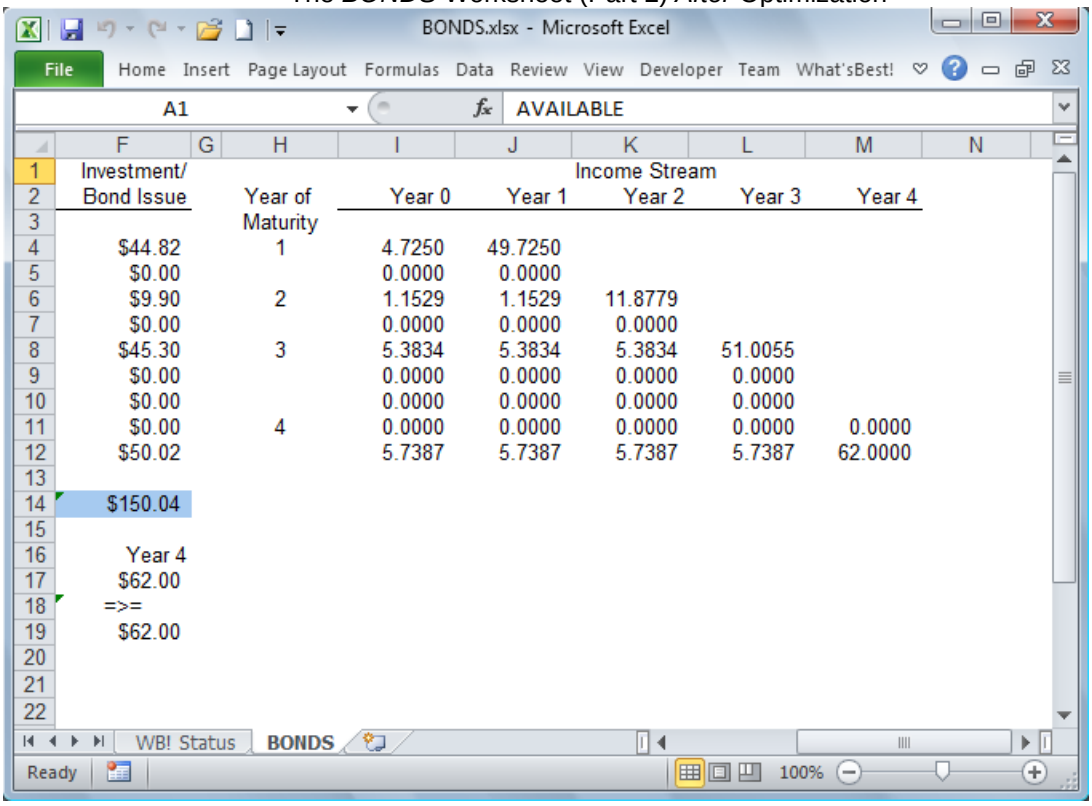
The *BONDS* Worksheet (Part 1) *After* Optimization



	A	B	C	D	E	F	G	H
1	AVAILABLE	Year of	Asking	Interest	Units	Investment/		
2	BOND OPTIONS:	Maturity	Price	Rate	Purchased	Bond Issue		Year of Maturity
3								
4		1	0.996	10.5	45.0	$44.82		1
5			0.997	10.5	0.0	$0.00		
6		2	0.923	10.75	10.7	$9.90		2
7			0.987	11.2	0.0	$0.00		
8		3	0.993	11.8	45.6	$45.30		3
9			1.061	12	0.0	$0.00		
10			0.883	10	0.0	$0.00		
11		4	1.102	12.6	0.0	$0.00		4
12			0.889	10.2	56.3	$50.02		
13								
14					TOTAL COST:	$150.04		
15								
16		Year 0	Year 1	Year 2	Year 3	Year 4		
17	Amount Covered	$17.00	$62.00	$23.00	$56.74	$62.00		
18		=>=	=>=	=>=	>=	=>=		
19	Cash Flow Need	$17.00	$62.00	$23.00	$35.00	$62.00		
20								
21								
22								

What's*Best!* returns a total cost of $150.04 million, with only one cash surplus in *Year 3* of $21.74 million. The *Income Stream* for each bond also appears in its optimized form:

The *BONDS* Worksheet (Part 2) *After* Optimization



	F	G	H	I	J	K	L	M	N
1	Investment/					Income Stream			
2	Bond Issue		Year of Maturity	Year 0	Year 1	Year 2	Year 3	Year 4	
3									
4	$44.82		1	4.7250	49.7250				
5	$0.00			0.0000	0.0000				
6	$9.90		2	1.1529	1.1529	11.8779			
7	$0.00			0.0000	0.0000	0.0000			
8	$45.30		3	5.3834	5.3834	5.3834	51.0055		
9	$0.00			0.0000	0.0000	0.0000	0.0000		
10	$0.00			0.0000	0.0000	0.0000	0.0000		
11	$0.00		4	0.0000	0.0000	0.0000	0.0000	0.0000	
12	$50.02			5.7387	5.7387	5.7387	5.7387	62.0000	
13									
14	$150.04								
15									
16	Year 4								
17	$62.00								
18	=>=								
19	$62.00								
20									
21									
22									

Lockbox Location

File name: LOCKBOX.XLSX *TYPE: LINEAR OPTIMIZATION*

The Problem in Words

As a corporate cash manager or officer of a bank whose clients must deal with the problem of optimizing collections from customers located at large distances from your home office, you must decide where to locate postal lockboxes, so that customers' deposits can be credited with a minimum of expense caused by mail delay (or "float time") while minimizing operating costs associated with the locations. In addition, each customer must be assigned to exactly one location.

Background

Your customers, located in *Seattle*, *Los Angeles*, *Houston*, *Philadelphia*, and *Miami*, whose deposits must be credited with a minimum of float time, may be assigned to potential lockbox locations in *New York*, *Atlanta*, *Cincinnati*, *Denver*, or *St. Louis*. It may also be more cost-efficient to assign them to the home office.

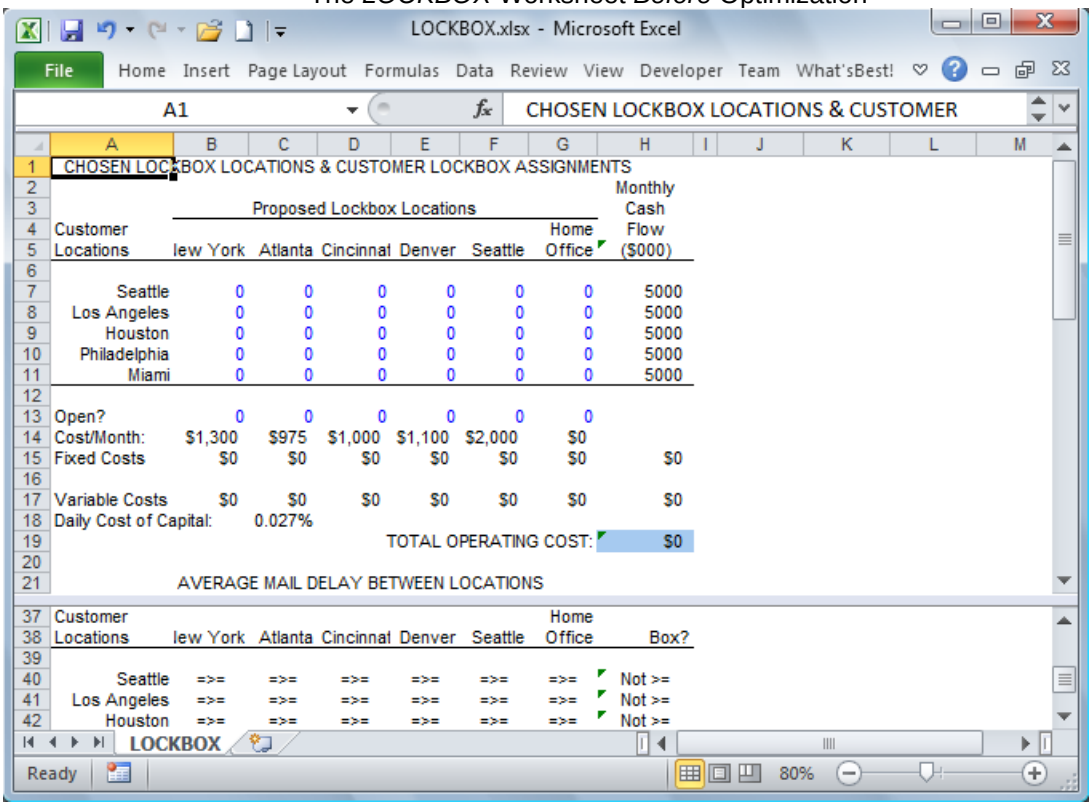
Objective of Optimization

Each customer must be assigned to a postal lockbox, so as to minimize the sum of float-time expense and monthly fixed operating costs.

The Worksheet

Let's open the *LOCKBOX* sample file and look at its design and formulas.

The *LOCKBOX* Worksheet *Before* Optimization



In cell C18, the *Daily Cost of Capital* is entered. In this example, the cost was calculated based on a hypothetical 10% annual rate for a 360-day year, prorated to .027% daily. This figure is important in calculating the expense due to float time caused by mail delay.

In H7:H11, the *Monthly Cash Flow* from each customer is shown. Cells B27:F31 display the average mail delay, in days, between customer sites and proposed lockbox locations. In G27:G31, the average delay from customer locations to the home office is found. Finally, in B14:G14, the fixed operating *Cost/Month* associated with each potential lockbox site is entered. Note that no fixed cost specific to the crediting of deposits is associated with the home office (cell G14), since this is part of the assumed cost of doing business.

A. Determine Adjustable Cells

The adjustable cells indicate whether or not cash is to be routed from a given customer to a proposed lockbox location (B7:G11) and whether or not a lockbox is opened at a given site (B13:G13). These cells display a figure to show if a customer is assigned to a lockbox location (1) or not (0).

B. Define Best

The best cell is the sum of all operating and float costs in H19. A cost summary broken down into subtotals of fixed and variable costs is found in A15:H19. Let's look at the formulas used to determine these costs. Considering first the variable costs associated with a lockbox in New York City, the formula in cell B17 is:

```
(B7*B27*$H$7+B8*B28*$H$8+B9*B29*$H$9+B10*B30*$H$10
+B11*B31*$H$11)*1000*$C$18
```

Thus, for the NYC lockbox, the average mail delay from each customer location (B27:B31) is multiplied by the cell indicating whether a lockbox has been assigned (B7:B11, a zero or a one), then by the amount of cash flow from the given customer location (H7:H11). The sum of these products is then multiplied by 1,000, to compensate for the fact that monthly cash flow figures are abbreviated by three digits, and by the *Daily Cost of Capital* in cell C18.

Fixed costs are calculated simply by multiplying the cost per month for each proposed lockbox (cell B14 in the case of NYC) by the cell indicating whether the location has been opened (B13).

These costs per location are added to produce subtotals in cells H15 and H17. The sum of the subtotals, *Total Operating Costs*, appears in H19, the objective cell.

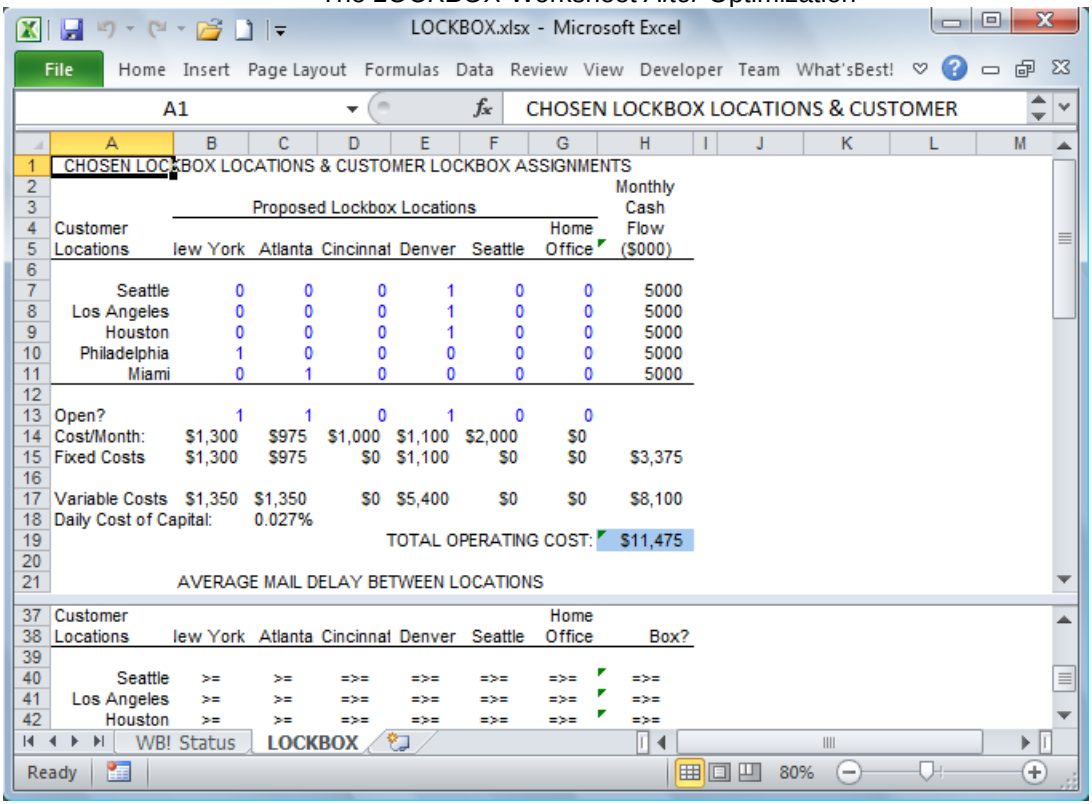
C. Specify Constraints

The constraints are of two types. First, cells B40:G44 prevent a customer from being assigned to a location where no lockbox has been opened. For instance, move the cursor to cell B40 and observe the formula there (*WB* (B$13, ">=", B7)). It requires that B13, a 1 or 0 depending on whether or not the New York lockbox is opened, be greater-than-or-equal-to B7, a 1 or 0 depending on whether or not the Seattle customer is assigned to the New York Lockbox. The value in B7 cannot be greater-than that in B13. If the Seattle customer is assigned to a lockbox located in New York, the location must be open or else the constraint in cell B40 will be unsatisfied.

The constraints in cells H40:H44 ensure that each customer is assigned to a lockbox by requiring that the sum of lockbox assignments for a given customer be at least 1.

With all constraints in place, we're ready to solve the model.

The *LOCKBOX* Worksheet *After* Optimization



After optimization, What's*Best!* has returned a minimized cost of $11,475 in cell H19 and the optimal solution recommends that lockboxes be opened in New York, Atlanta, and Denver.

Note: This model is an example of a class of models that returns naturally integer answers even though the adjustable cells have not been specified as integer. Since the use of 0/1 integer adjustable cells can cause dramatic increases in computation time, it is best to use naturally integer models whenever possible. Space here prohibits a detailed explanation of this phenomenon. Interested readers may refer to Chapter 8, *Introductions to Operations Research*, by Hillier & Lieberman, 7th ed., from McGraw-Hill, and to Chapter 14, *Optimization Modeling with LINGO*, by Linus Schrage, from Duxbury Press. The latter text is available from LINDO Systems. Call (312) 988-7422 for more information.

Markowitz Portfolio Problem

File name: MARKOWIT.XLSX *TYPE: NONLINEAR OPTIMIZATION*

Application Profile

In the March, 1952, issue of *Journal of Finance*, Harry M. Markowitz published an article called *Portfolio Selection*. In it, he demonstrated how to reduce the standard deviation of returns on asset portfolios by selecting assets with prices that don't fluctuate in exactly the same ways. At the same time, he laid down some basic principles for establishing an advantageous relationship between risk and return, and his work is still in use forty years later.

The Problem in Words

You're considering investing in three assets and historical data reveal that the return from each asset has fluctuated over time. You want to reduce variability, or risk, by spreading your investment over the three stocks.

Background

From the historical data, you have calculated an expected return, the variance of the return rate, and the covariance of the return between the different assets. Variance is a measure of the fluctuation in the return — the higher the variance, the riskier the investment. The covariance is a measure of the correlation of return fluctuations of one stock with the fluctuations of another. High covariance indicates that an increase in one stock's return is likely to correspond to an increase in the other. A low covariance means the return rates are relatively independent and a negative covariance means that an increase in one stock's return is likely to correspond to a decrease in the other.

You have a target return of 15%. What percentages of your funds should you invest in each of the three assets to achieve this target and minimize the variance (or risk) of the portfolio? As an additional safety feature, you decide to invest no more than 75% in any single asset.

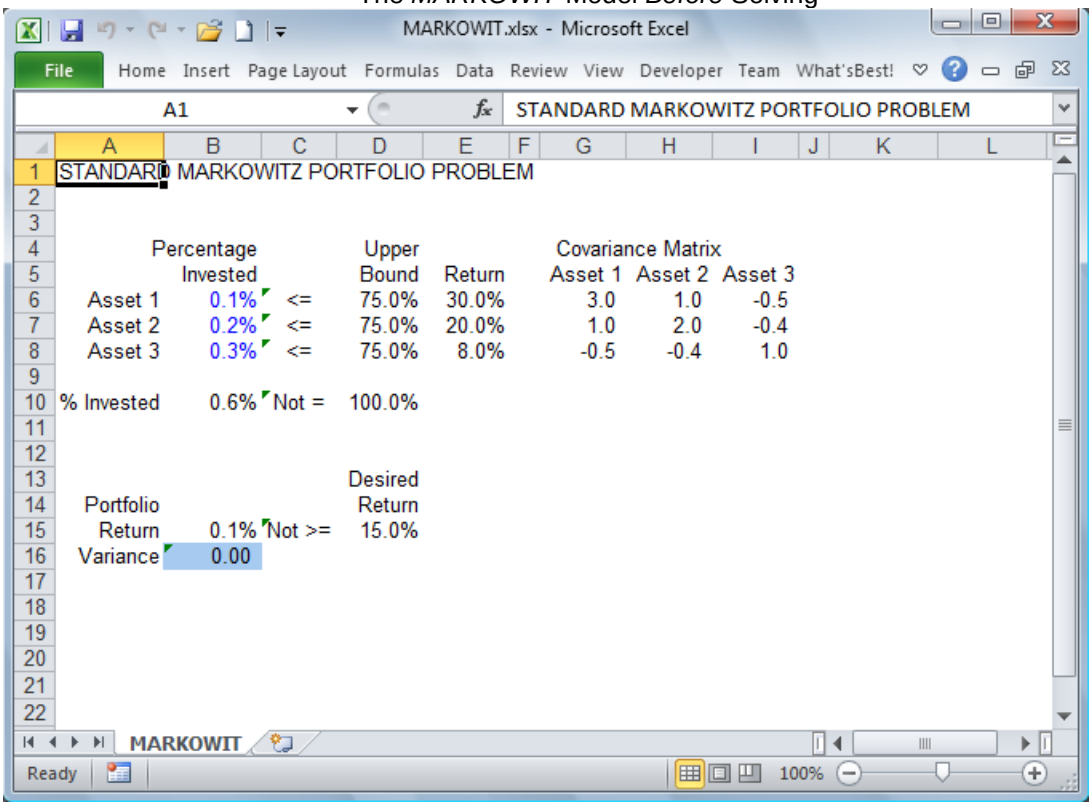
Objective of Optimization

The objective is to determine the percent to invest in each asset while minimizing risk of the entire portfolio.

The Worksheet

Let's look at the *MARKOWIT* sample file as supplied.

The *MARKOWIT* Model *Before* Solving



A. Determine Adjustable Cells

The adjustable cells are B6:B8, the *Percentages Invested* in each of the three stocks.

B. Define Best

The best solution is the one that minimizes the risk of the entire portfolio. In the Markowitz model, a portfolio's variance is used as the measure of risk. Using the variance gives a relatively greater penalty to outcomes that are far from the mean.

The variance of a portfolio is:

$$\Sigma_i \Sigma_j x_i x_j \sigma_{ij}$$

…where:

x_i is the percentage invested in *Asset i*

σ_{ij} for $i \neq j$ is the covariance between *Assets i* and *j*

for $i=j$ is the variance of *Asset i*

This is computed by means of the following formula in cell B16:

```
(G6*B6^2+G7*B6*B7+G8*B6*B8)+(H6*B7*B6+H7*B7^2
   +H8*B7*B8)+(I6*B6*B8+I7*B7*B8+I8*B8^2)
```

C. Specify Constraints

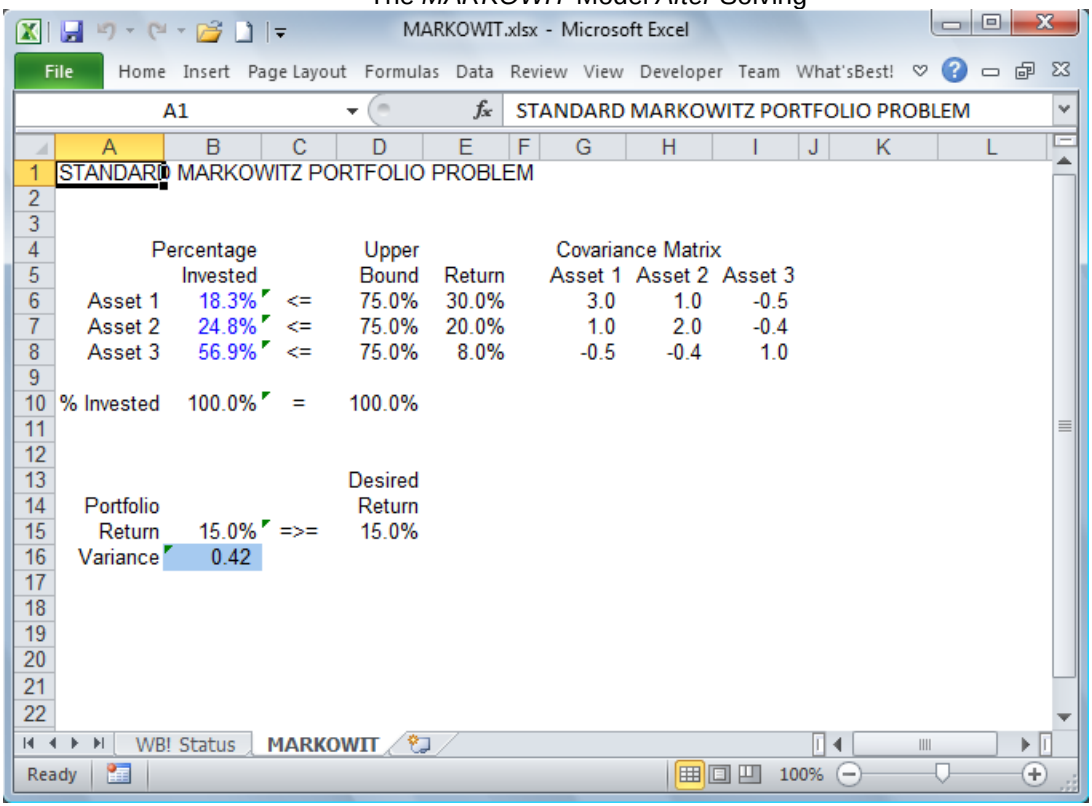
There are three types of constraints in the Markowitz model. The first, in C6:C8, requires that no percentage invested in any single stock exceed the *Upper Bound* for each stock of 75% (D6:D8). For example, C6 has *WB*(B6,"<=",D6).

The second, in C15, is *WB*(B15,">=",D15). This requires that the combined return on the three *Assets* (B15) be greater-than-or-equal-to the desired return in cell D15 (15%).

The third, in C10, is *WB*(B10,"=",D10). This requires the sum of the percentages invested in each stock (B10) add up to 100% (D10).

Now, let's solve the model. After solving, the *WB! Status* worksheet will open in order to show you the *Nonlinearity present* warning. (This warning can be shut off from the *General Options* dialog box.) Your solved model now appears as follows.

The *MARKOWIT* Model *After* Solving



What's*Best!* returns a minimized variance of 0.42.

An interesting exercise is to solve this model for different levels of *Desired Return* in cell D15, recording the portfolio variance at each step. A graph of the resulting data, illustrating the tradeoff between risk and return, is called the efficient frontier.

Portfolio with Transaction Costs

File name: PORTCOST.XLSX *TYPE: NONLINEAR OPTIMIZATION*

Application Profile

As the expected return and variances of each asset change, the optimal portfolio is almost certain to change. Adjusting your investments with every change may cheer up your broker, but the commissions or transaction costs will slowly erode the value of your portfolio. Typically, there's a cost associated with each sale or purchase. Using this model, you need to take these transaction costs into account in calculating how to adjust your portfolio.

The Problem in Words

According to your estimates, your current holdings will yield a 12.2% return. You are interested in adjusting your portfolio to lower the variance, yet achieve a return of at least 9.5%.

Background

Your portfolio currently consists of 50% of *Asset 1*, 30% of *Asset 2*, and 20% of *Asset 3*. You pay a transaction fee at the beginning of the period that is a percentage of the amount bought or sold. The transaction fee is 1.0% for *Asset 1*, 1.5% for *Asset 2*, and 2.0% for *Asset 3*. You have estimated the expected return and the covariance terms for each asset.

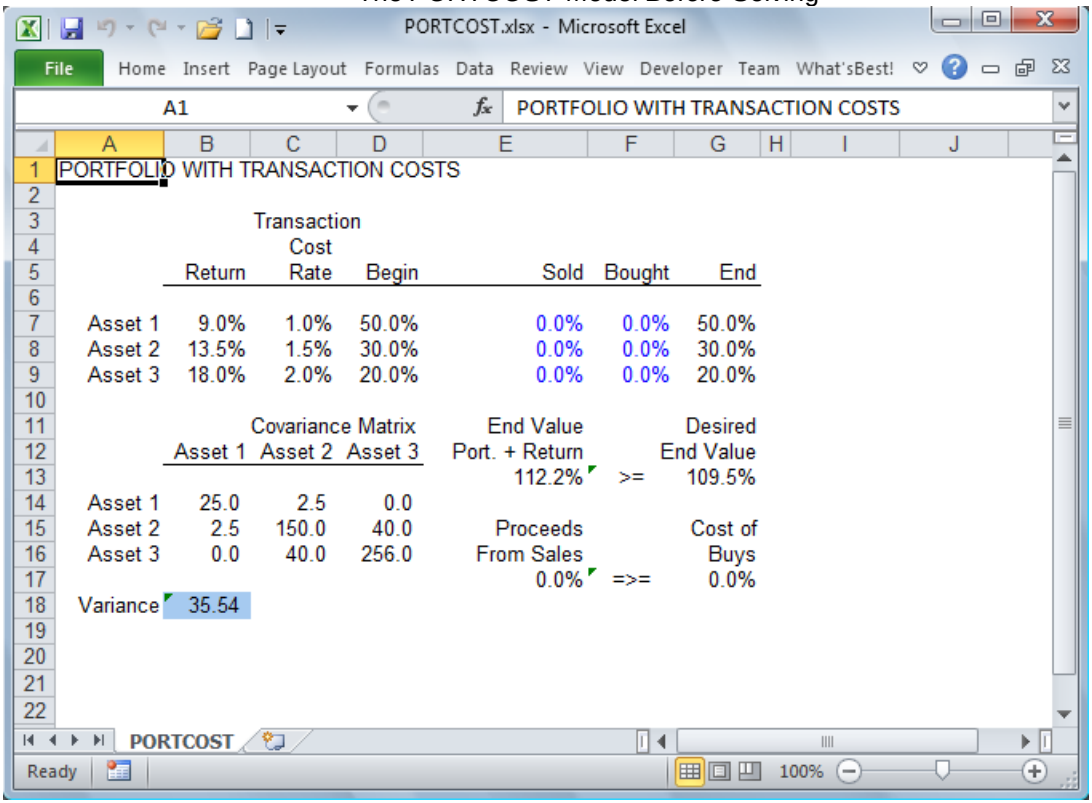
Objective of Optimization

The objective is to determine the percent to invest in each asset while minimizing the risk of the entire portfolio.

The Worksheet

Let's look at the *PORTCOST* sample file.

The *PORTCOST* Model *Before* Solving



A. Determine Adjustable Cells

The adjustable cells are the percentages of each asset to be *Bought* or *Sold*, in cells E7:F9.

B. Define Best

The best solution minimizes the variance of the portfolio. This is accomplished in cell B18 by means of the following formula explained in the previous sample model:

```
(G7^2*B14+G7*G8*B15+G7*G9*B16)+(G8*G7*C14+G8^2*C15
+G8*G9*C16)+(G9*G7*D14+G9*G8*D15+G9^2*D16)
```

C. Specify Constraints

Cell E13 calculates the value of the ending portfolio plus the return over the period with the formula:

```
(1+B7)*G7+(1+B8)*G8+(1+B9)*G9
```

The constraint in cell F13, *WB*(E13,">=",G13), ensures that the *End Value Port. + Return* (E13) is greater-than-or-equal-to the *Desired End Value* (G13) of 109.5% of the original portfolio value.

The cash required to buy any asset must come from selling some asset. The *Proceeds From Sales* is calculated in E17 by the formula:

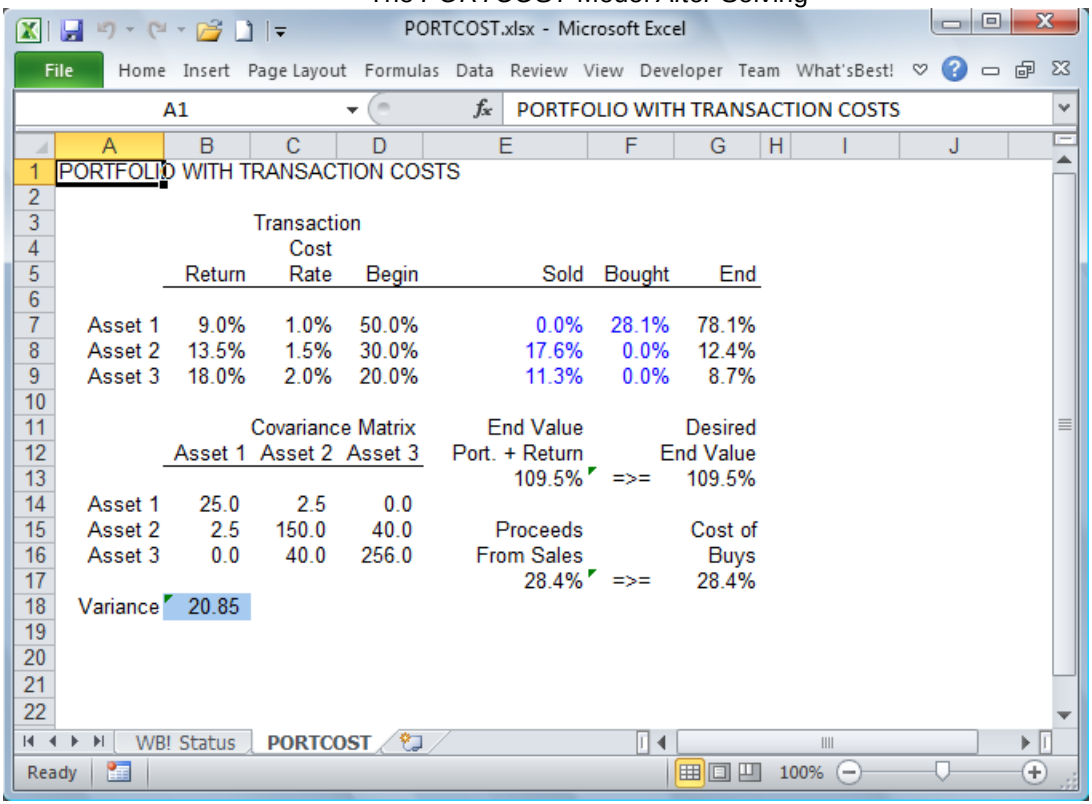
E7*(1-$C$7)+E8*(1-$C$8)+E9*(1-$C$9)

The constraint in cell F17, *WB*(E17,">=",G17), requires the *Proceeds from Sales* to be at least as great as the *Cost of Buys* in G17, calculated by the formula:

F7*(1+$C$7)+F8*(1+$C$8)+F9*(1+$C$9)

Now, let's solve the model. After solving, the *WB! Status* worksheet will open in order to show you the *Nonlinearity present* warning. (This warning can be shut off from the *General Options* dialog box.) Your solved model now appears as follows:

The *PORTCOST* Model *After* Solving



In the optimal solution, you are holding 78.1% of *Asset 1*, 12.4% of *Asset 2*, and 8.7% of *Asset 3*. These amounts are percentages of the value of your original portfolio. You may notice that the percentages add to only 99.2% rather than 100%. This is because .8% of the value of the original portfolio has gone to the transaction costs required to shift your investment levels. Note also that the variance has been lowered to 20.85 from 35.54.

Portfolio - Minimizing Downside Risk

File name: DNRISK.XLSX *TYPE: NONLINEAR OPTIMIZATION*

Application Profile

An obvious objective and common strategy in investing is to minimize the risk of losing ground. This model demonstrates how to accomplish this.

The Problem in Words

You're considering investing in three different assets. You need to determine how much to invest in each of three assets to minimize the risk of having to sell your baseball cards.

Background

You have forecasts for the expected return of each asset under seven equally likely scenarios — there is a transportation strike, interest rates rise or fall, crops fail, Cubs win the World Series, Bulls win NBA finals, etc. You know that if you invest 100% of your capital in *Asset 2*, the average return under all possible scenarios (your expected return) is 19.9%.

You plan to live on the return from your portfolio. After figuring your living expenses, you find that a return below 11% will force you to start selling off your beloved baseball card collection to live in the manner to which you've become accustomed. Call that 11% your threshold return.

The return on *Asset 2* is below your threshold return of 11% in a couple of scenarios, and under *Scenario 7* you actually lose money on your portfolio—perhaps forcing you to sell the entire baseball card collection. While the 19.9% return is very appealing, the thought of parting with even some of your baseball cards is frightening. You've decided you would like an average or expected return of at least 13% (call this your desired return), but you want to minimize the likelihood that you'll have to sell off some of those valuable cards.

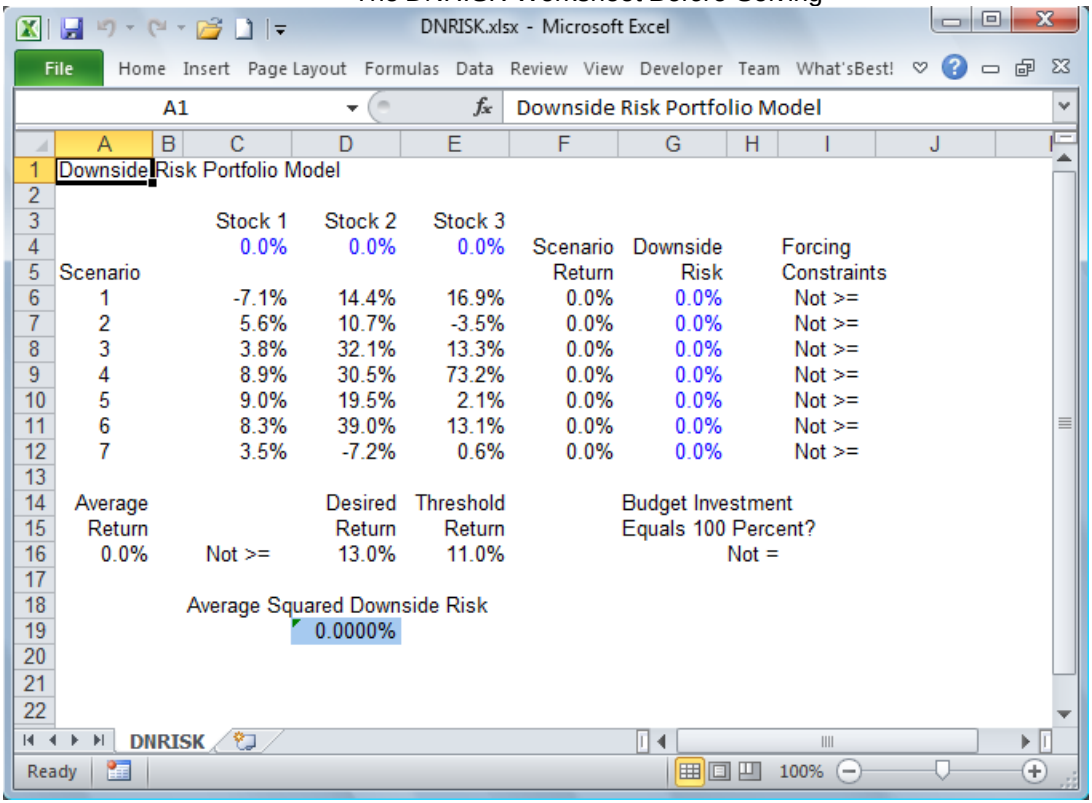
Objective of Optimization

The objective is to invest for a profit while minimizing the risk of diminishing your baseball card collection.

The Worksheet

Let's look at the *DNRISK* sample file.

The *DNRISK* Worksheet *Before* Solving



A. Determine Adjustable Cells

The adjustable cells are the percentages of your capital to invest in each of the three assets, in cells C4:E4, and the percentages of *Downside Risk* for each *Scenario* in cells G6:G12.

B. Define Best

The best cell is the average of the sum of the squares of the downside risk percentages (D19). The sum of squares is used to increase the relative penalty on outcomes that are far below the return threshold.

C. Specify Constraints

There are three kinds of constraints in this model.

The *Forcing Constraints* in I6:I12 force the *Downside Risk* for each *Scenario* to be either the *Threshold Return* minus *Scenario Return* when the *Scenario Return* is less-than the *Threshold Return*, or zero when the *Scenario Return* is more than the Threshold Return. For example, the formula in I6 is:

WB(G6,">=",$E$16-F6)

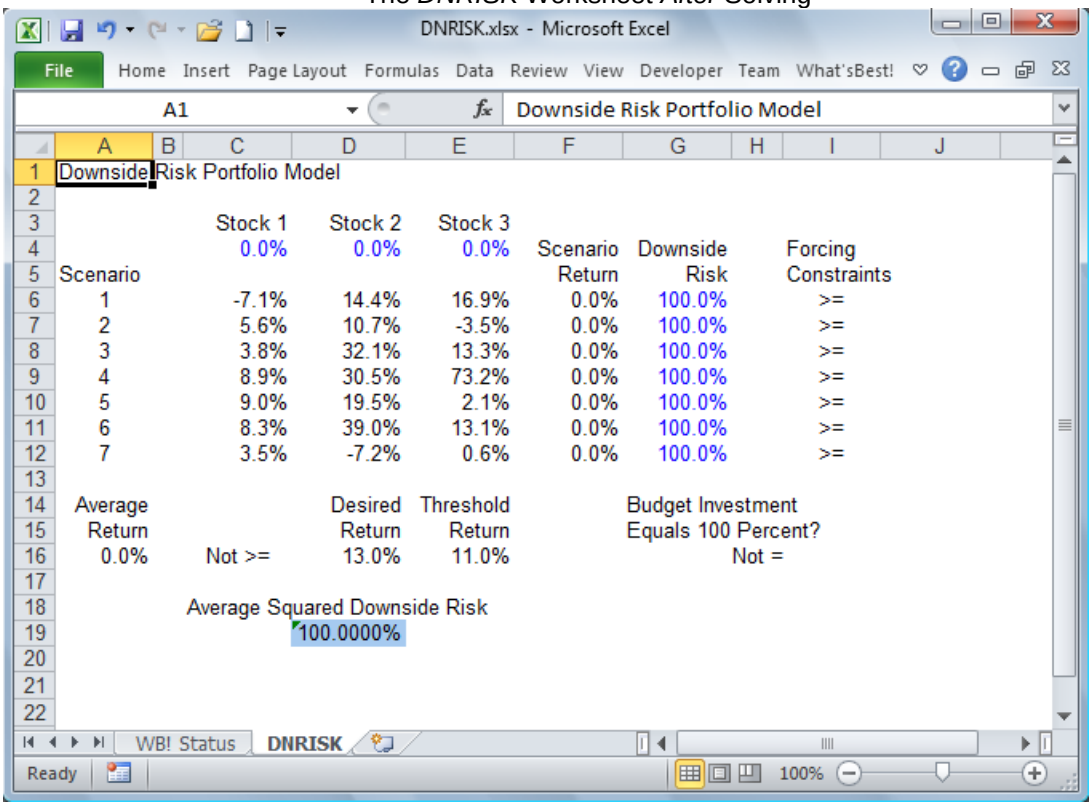
Since G6 is an adjustable cell, which is prohibited from returning a negative value, and since this is a minimization problem, it will return the smallest possible nonnegative value. If E16 (*Threshold Return*) is less-than F6 (*Scenario Return*), E16-F6 is a negative number — so the constraint will be satisfied by zero, which is greater-than any negative number.

The constraint in H16 forces the sum of percentages invested in each stock to be 100.

The constraint in C16 forces the average return for the scenarios to be greater-than-or-equal-to the *Desired Return*.

Now, let's solve the model. After solving, the *WB! Status* worksheet will open in order to show you the *Nonlinearity present* warning. (This warning can be shut off from the *General Options* dialog box.) Your solved model now appears as follows.

The *DNRISK* Worksheet *After* Solving



	A	C	D	E	F	G	I
1	Downside Risk Portfolio Model						
3		Stock 1	Stock 2	Stock 3			
4		0.0%	0.0%	0.0%	Scenario	Downside	Forcing
5	Scenario				Return	Risk	Constraints
6	1	-7.1%	14.4%	16.9%	0.0%	100.0%	>=
7	2	5.6%	10.7%	-3.5%	0.0%	100.0%	>=
8	3	3.8%	32.1%	13.3%	0.0%	100.0%	>=
9	4	8.9%	30.5%	73.2%	0.0%	100.0%	>=
10	5	9.0%	19.5%	2.1%	0.0%	100.0%	>=
11	6	8.3%	39.0%	13.1%	0.0%	100.0%	>=
12	7	3.5%	-7.2%	0.6%	0.0%	100.0%	>=
14	Average		Desired	Threshold		Budget Investment	
15	Return		Return	Return		Equals 100 Percent?	
16	0.0%	Not >=	13.0%	11.0%		Not =	
18		Average Squared Downside Risk					
19		100.0000%					

The *Average Squared Downside Risk* in cell D19 has been minimized to .2828%.

Portfolio - Scenario Model

File name: PORTSCEN.XLSX *TYPE: NONLINEAR OPTIMIZATION*

The Problem in Words

There are a variety of ways to measure the risk of a portfolio. In this model, you need to observe the changes in the optimal investment allocations resulting from the minimization of risk as measured in three different ways and determine the best measure of risk for your lifestyle.

Background

Once again, there are three assets, this time with twelve equally likely expected return scenarios. Scenarios are outcomes of events with an influence on your analysis: Fed raises interest rates by 0.5 point, Fed lowers interest rates by 0.25 point, housing starts up, etc. This model calculates three different measures of risk: variance, semi-variance, and downside risk. You can minimize each of the three and get different levels of each asset that achieve your target return of 15% — each offers the "lowest risk", depending on which of the three measures you select.

Variance is a measure of the fluctuation of the expected return. However, this measure of risk considers a scenario that returns 5% *above* the target to be as desirable as a scenario that returns 5% *below* the target. You're probably only worried about the risk that the return will be below your target return.

Both *Semi-variance* and *downside risk* look only at the risk that your return will be below the target. Downside risk minimizes the difference between the target and returns below the target. The semi-variance minimizes the difference between the target and the square of the returns below the target. Therefore, it puts a higher weight on larger differences below the target.

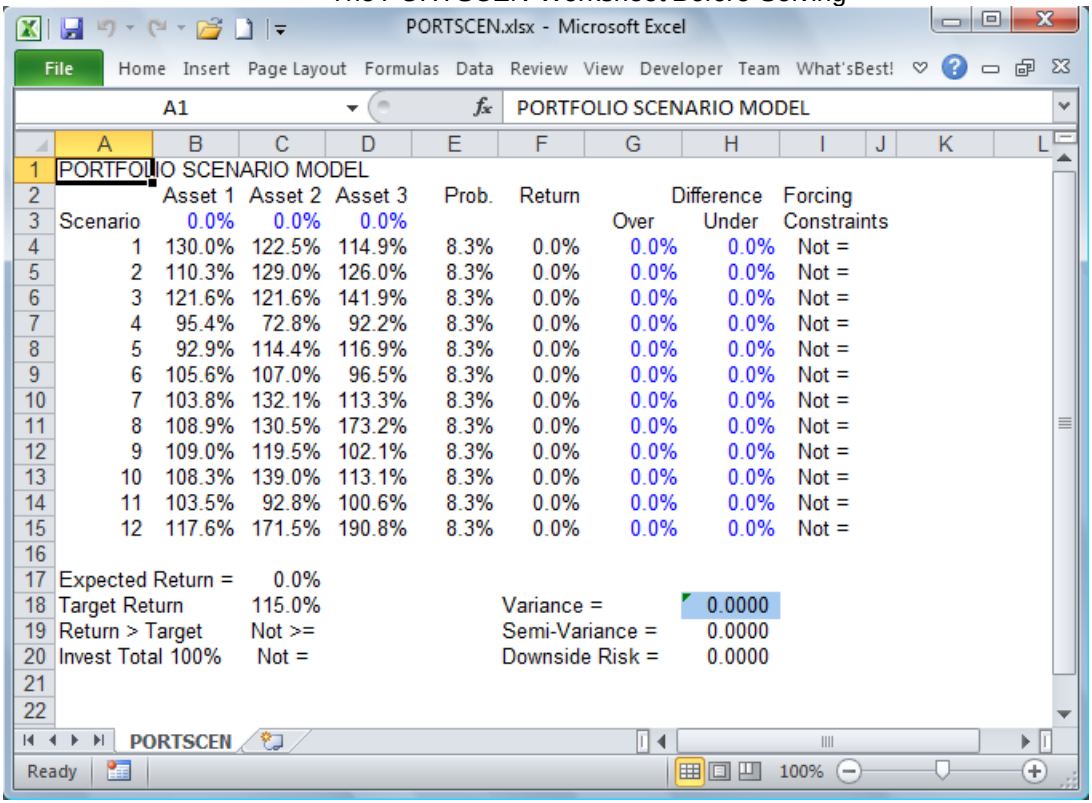
Objective of Optimization

The objective is to determine the percent to invest in each asset while minimizing the risk of the entire portfolio using the risk measure that is most important to you.

The Worksheet

Let's look at the supplied *PORTSCEN* sample file:

The *PORTSCEN* Worksheet *Before* Solving



	A	B	C	D	E	F	G	H	I
1	PORTFOLIO SCENARIO MODEL								
2		Asset 1	Asset 2	Asset 3	Prob.	Return		Difference	Forcing
3	Scenario	0.0%	0.0%	0.0%			Over	Under	Constraints
4	1	130.0%	122.5%	114.9%	8.3%	0.0%	0.0%	0.0%	Not =
5	2	110.3%	129.0%	126.0%	8.3%	0.0%	0.0%	0.0%	Not =
6	3	121.6%	121.6%	141.9%	8.3%	0.0%	0.0%	0.0%	Not =
7	4	95.4%	72.8%	92.2%	8.3%	0.0%	0.0%	0.0%	Not =
8	5	92.9%	114.4%	116.9%	8.3%	0.0%	0.0%	0.0%	Not =
9	6	105.6%	107.0%	96.5%	8.3%	0.0%	0.0%	0.0%	Not =
10	7	103.8%	132.1%	113.3%	8.3%	0.0%	0.0%	0.0%	Not =
11	8	108.9%	130.5%	173.2%	8.3%	0.0%	0.0%	0.0%	Not =
12	9	109.0%	119.5%	102.1%	8.3%	0.0%	0.0%	0.0%	Not =
13	10	108.3%	139.0%	113.1%	8.3%	0.0%	0.0%	0.0%	Not =
14	11	103.5%	92.8%	100.6%	8.3%	0.0%	0.0%	0.0%	Not =
15	12	117.6%	171.5%	190.8%	8.3%	0.0%	0.0%	0.0%	Not =
16									
17	Expected Return =		0.0%						
18	Target Return		115.0%			Variance =		0.0000	
19	Return > Target		Not >=			Semi-Variance =		0.0000	
20	Invest Total 100%		Not =			Downside Risk =		0.0000	
21									
22									

A. Determine Adjustable Cells

The adjustable cells in the model are the percentages of capital invested in each *Asset* (B3:D3), and the differences *Over* and *Under* in G4:H15. The *Overs* are forced to be the amount by which a *Return* is greater-than the *Expected Return*. The *Unders*, similarly, are the amount by which a *Return* is less-than the *Expected Return*. These adjustable cells are forced to take these values by the constraints in I4:I15, which we explain in detail under *C. Specify Constraints* below.

B. Define Best

You can choose one of three best cells in this model. We'll show solutions for all three minimization objectives:

- Variance in H18,
- Semi-Variance in H19, and
- Downside Risk in H20.

The *Variance* is calculated in H18 with the formula:

```
E4*(G4+H4)^2+E5*(G5+H5)^2+E6*(G6+H6)^2+E7*(G7+H7)^2+E8
*(G8+H8)^2+E9*(G9+H9)^2+E10*(G10+H10)^2+E11*(G11+H11)^2
+E12*(G12+H12)^2+E13*(G13+H13)^2+E14*(G14+H14)^2+E15
*(G15+H15)^2
```

This is equivalent to the sumproduct of the probabilities and the squared values of the difference between *Return* and *Target Return*.

The *Downside Risk* is calculated as the sumproduct of the probabilities in *Column E* and the *Unders* in *Column H*. The *Semi-variance* is formulated as the sumproduct of the probabilities and the squares of the *Unders*.

C. Specify Constraints

There are three types of constraints in the *PORTSCEN* model. The first requires a bit of thought.

The *Forcing Constraints* in I4:I15 force the difference between *Over* (G4:G15) and *Under* (H4:H15) to be equal-to the difference between the *Expected Return* (F17) and *Return* (F4:F15). For instance, in I4 the formula is: *WB*(G4-H4,"=",F4-$C$17).

To find the semi-variance and downside risk, we need to calculate the amount by which each *Scenario Risk* (*SR*) is below the *Target Return* (*TR*). A shrewd spreadsheet builder will recognize that this calculation can be done using an *IF* function of the form:

IF(*SR*<*TR*,*TR*-*SR*,0)

This function returns the difference between the target return and the scenario return if the scenario return is less-than the target return. Otherwise, it returns zero.

The bend at the origin of this *IF* statement makes it a non-smooth function. A smooth function has no breaks or sharp bends. Most *IF* functions that depend directly or indirectly on adjustable cells are non-smooth, and it's wise to avoid non-smooth functions, since they can be difficult to solve reliably (see Chapter 7, *Overview of Mathematical Modeling*, for a discussion of non-smooth relationships).

By adding adjustable cells and carefully formulated constraints, it turns out, you can avoid the need for most non-smooth *IF* functions. You can choose to use *IF* functions instead of finding ways around them, but you run the risk of suboptimal answers from the solver.

In this case, we've emulated the *IF* function using two adjustable cells and one constraint. For each of the scenarios, we've added an adjustable cell in columns *G* and *H*. Then, we use a constraint to force the adjustable cell in column *G* to be equal-to the amount the scenario return is *Over* the target return, or zero if the return is less-than the target. The adjustable cell in column *H* is likewise required to be equal-to the amount the scenario return is *Under* the target return, or zero if the return is greater-than the target.

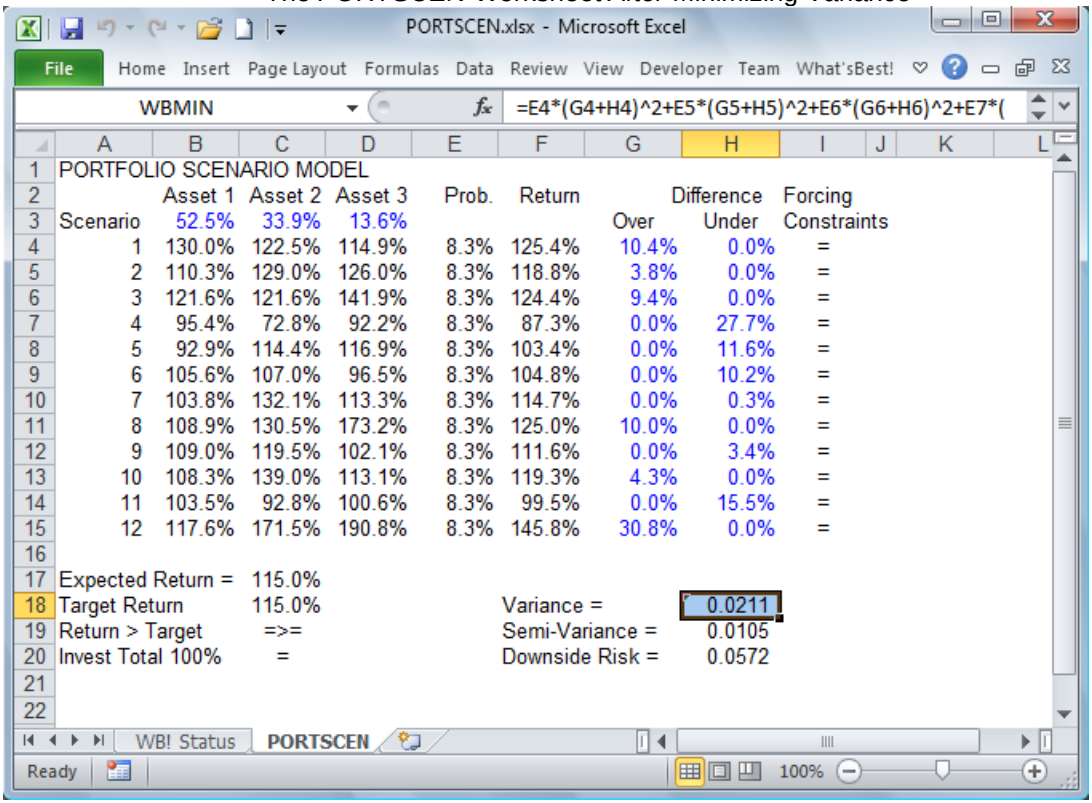
The constraints in cells I4:I15 force the difference between the adjustable cells G4:H15 to equal the difference between the return for this scenario (F4:F15) and the *Target Return* (C19).

For example, for scenario 1 the constraint in cell I4 reads: *WB*(G4-H4,"=",F4-$C$18). If *Target Return* (C18) is 115% and the scenario return (F4) is 120%, the constraint forces G4-H4 to be equal-to 5%. As adjustable cells, G4 and H4 are implicitly nonnegative. Since the objective is to minimize a function of these adjustable cells, they're forced to the smallest values that satisfy all constraints. Thus, G4 will be forced to 5% and H4 will be forced to 0%. Similarly, if the scenario return (F4) is 110%, then the constraint forces G4-H4 to return -5%. G4 is forced to 0% and H4 to 5%. If you experiment with the asset allocations in cells G4:H15, you can confirm that the forcing constraints work properly.

In C19, the constraint formula: *WB*(C17,">=",C18) forces the *Expected Return* in C17 to be at least as great as the *Target Return* in C18. In C20, a constraint requires that 100 percent of the funds are invested. That is, that B3:D3 sum to 1.

Now, you're ready to solve the model. After solving, the *WB! Status* worksheet will open in order to show you the *Nonlinearity present* warning. (This warning can be disabled from the *General Options* dialog box.) The solutions for all three objectives follow.

The *PORTSCEN* Worksheet *After* Minimizing Variance

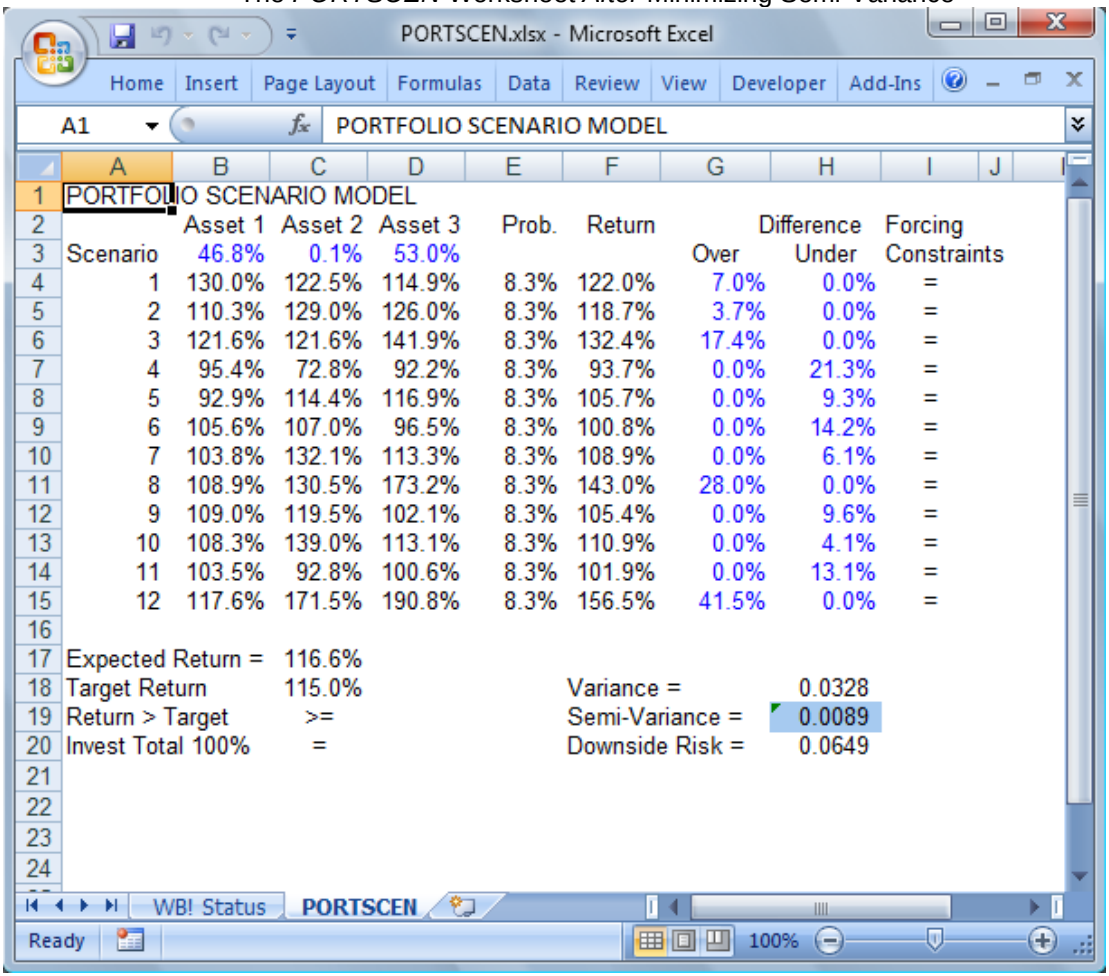


	A	B	C	D	E	F	G	H	I
1	PORTFOLIO SCENARIO MODEL								
2		Asset 1	Asset 2	Asset 3	Prob.	Return		Difference	Forcing
3	Scenario	52.5%	33.9%	13.6%			Over	Under	Constraints
4	1	130.0%	122.5%	114.9%	8.3%	125.4%	10.4%	0.0%	=
5	2	110.3%	129.0%	126.0%	8.3%	118.8%	3.8%	0.0%	=
6	3	121.6%	121.6%	141.9%	8.3%	124.4%	9.4%	0.0%	=
7	4	95.4%	72.8%	92.2%	8.3%	87.3%	0.0%	27.7%	=
8	5	92.9%	114.4%	116.9%	8.3%	103.4%	0.0%	11.6%	=
9	6	105.6%	107.0%	96.5%	8.3%	104.8%	0.0%	10.2%	=
10	7	103.8%	132.1%	113.3%	8.3%	114.7%	0.0%	0.3%	=
11	8	108.9%	130.5%	173.2%	8.3%	125.0%	10.0%	0.0%	=
12	9	109.0%	119.5%	102.1%	8.3%	111.6%	0.0%	3.4%	=
13	10	108.3%	139.0%	113.1%	8.3%	119.3%	4.3%	0.0%	=
14	11	103.5%	92.8%	100.6%	8.3%	99.5%	0.0%	15.5%	=
15	12	117.6%	171.5%	190.8%	8.3%	145.8%	30.8%	0.0%	=
16									
17	Expected Return =	115.0%							
18	Target Return	115.0%				Variance =	0.0211		
19	Return > Target	=>=				Semi-Variance =	0.0105		
20	Invest Total 100%	=				Downside Risk =	0.0572		
21									
22									

Variance minimizes to .0211.

Solutions for downside risk and semi-variance appear below.

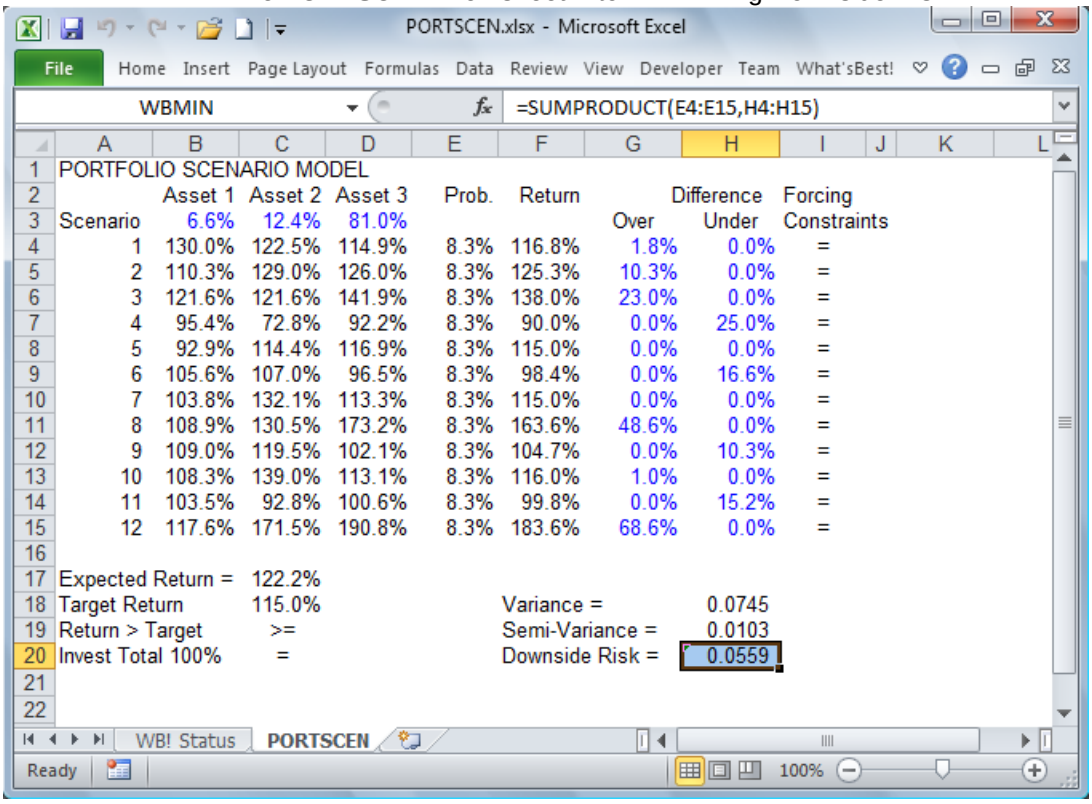
The *PORTSCEN* Worksheet *After* Minimizing Semi-Variance



	A	B	C	D	E	F	G	H	I
1	PORTFOLIO SCENARIO MODEL								
2		Asset 1	Asset 2	Asset 3	Prob.	Return		Difference	Forcing
3	Scenario	46.8%	0.1%	53.0%			Over	Under	Constraints
4	1	130.0%	122.5%	114.9%	8.3%	122.0%	7.0%	0.0%	=
5	2	110.3%	129.0%	126.0%	8.3%	118.7%	3.7%	0.0%	=
6	3	121.6%	121.6%	141.9%	8.3%	132.4%	17.4%	0.0%	=
7	4	95.4%	72.8%	92.2%	8.3%	93.7%	0.0%	21.3%	=
8	5	92.9%	114.4%	116.9%	8.3%	105.7%	0.0%	9.3%	=
9	6	105.6%	107.0%	96.5%	8.3%	100.8%	0.0%	14.2%	=
10	7	103.8%	132.1%	113.3%	8.3%	108.9%	0.0%	6.1%	=
11	8	108.9%	130.5%	173.2%	8.3%	143.0%	28.0%	0.0%	=
12	9	109.0%	119.5%	102.1%	8.3%	105.4%	0.0%	9.6%	=
13	10	108.3%	139.0%	113.1%	8.3%	110.9%	0.0%	4.1%	=
14	11	103.5%	92.8%	100.6%	8.3%	101.9%	0.0%	13.1%	=
15	12	117.6%	171.5%	190.8%	8.3%	156.5%	41.5%	0.0%	=
16									
17	Expected Return =	116.6%							
18	Target Return	115.0%				Variance =	0.0328		
19	Return > Target	>=				Semi-Variance =	0.0089		
20	Invest Total 100%	=				Downside Risk =	0.0649		
21									
22									
23									
24									

After solving, the semi-variance is minimized to .0089.

The *PORTSCEN* Worksheet *After* Minimizing Downside Risk



	A	B	C	D	E	F	G	H	I	J	K	L
1	PORTFOLIO SCENARIO MODEL											
2		Asset 1	Asset 2	Asset 3	Prob.	Return		Difference	Forcing			
3	Scenario	6.6%	12.4%	81.0%			Over	Under	Constraints			
4	1	130.0%	122.5%	114.9%	8.3%	116.8%	1.8%	0.0%	=			
5	2	110.3%	129.0%	126.0%	8.3%	125.3%	10.3%	0.0%	=			
6	3	121.6%	121.6%	141.9%	8.3%	138.0%	23.0%	0.0%	=			
7	4	95.4%	72.8%	92.2%	8.3%	90.0%	0.0%	25.0%	=			
8	5	92.9%	114.4%	116.9%	8.3%	115.0%	0.0%	0.0%	=			
9	6	105.6%	107.0%	96.5%	8.3%	98.4%	0.0%	16.6%	=			
10	7	103.8%	132.1%	113.3%	8.3%	115.0%	0.0%	0.0%	=			
11	8	108.9%	130.5%	173.2%	8.3%	163.6%	48.6%	0.0%	=			
12	9	109.0%	119.5%	102.1%	8.3%	104.7%	0.0%	10.3%	=			
13	10	108.3%	139.0%	113.1%	8.3%	116.0%	1.0%	0.0%	=			
14	11	103.5%	92.8%	100.6%	8.3%	99.8%	0.0%	15.2%	=			
15	12	117.6%	171.5%	190.8%	8.3%	183.6%	68.6%	0.0%	=			
16												
17	Expected Return =	122.2%										
18	Target Return	115.0%				Variance =		0.0745				
19	Return > Target	>=				Semi-Variance =		0.0103				
20	Invest Total 100%	=				Downside Risk =		0.0559				
21												
22												

Minimized downside risk comes to .0559.

Minimization of the three measures of risk yield the following portfolios:

	Asset 1	Asset 2	Asset 3
Minimize Variance	52.5%	33.9%	13.6%
Minimize Semi-Variance	46.8%	.1%	53.1%
Minimize Downside Risk	6.6%	12.4%	81.0%

It is interesting to observe that, although minimizing all three measures of risk (*Variance*, *Semi-variance*, and *Downside Risk*) individually produces a similar level of risk, it also produces rather different portfolio allocation and expected returns. This illustrates why it is important to consider the measure of risk that is most appropriate for your investment goals.

Seasonal Sales Factoring

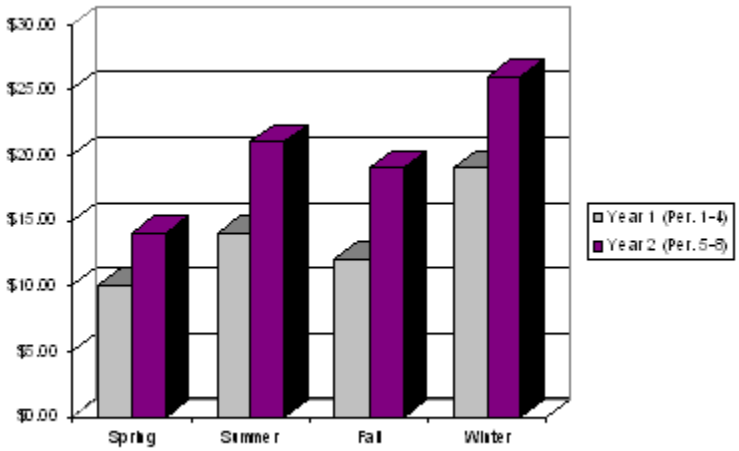
File name: SEASON.XLSX *TYPE: NONLINEAR OPTIMIZATION*

The Problem in Words

You sell a seasonal product and you would like to find out how the seasons affect sales and your sales growth. This information would help you in forecasting sales and making future ordering decisions.

Background

This model calculates seasonal factors that measure the effect each season has on sales and fits a straight line to the normalized data by minimizing the squared error. The *trend*, or slope of the line, is the sales growth per season. The *base* is the sales figure at time zero. You have the sales data by season for the last two years:



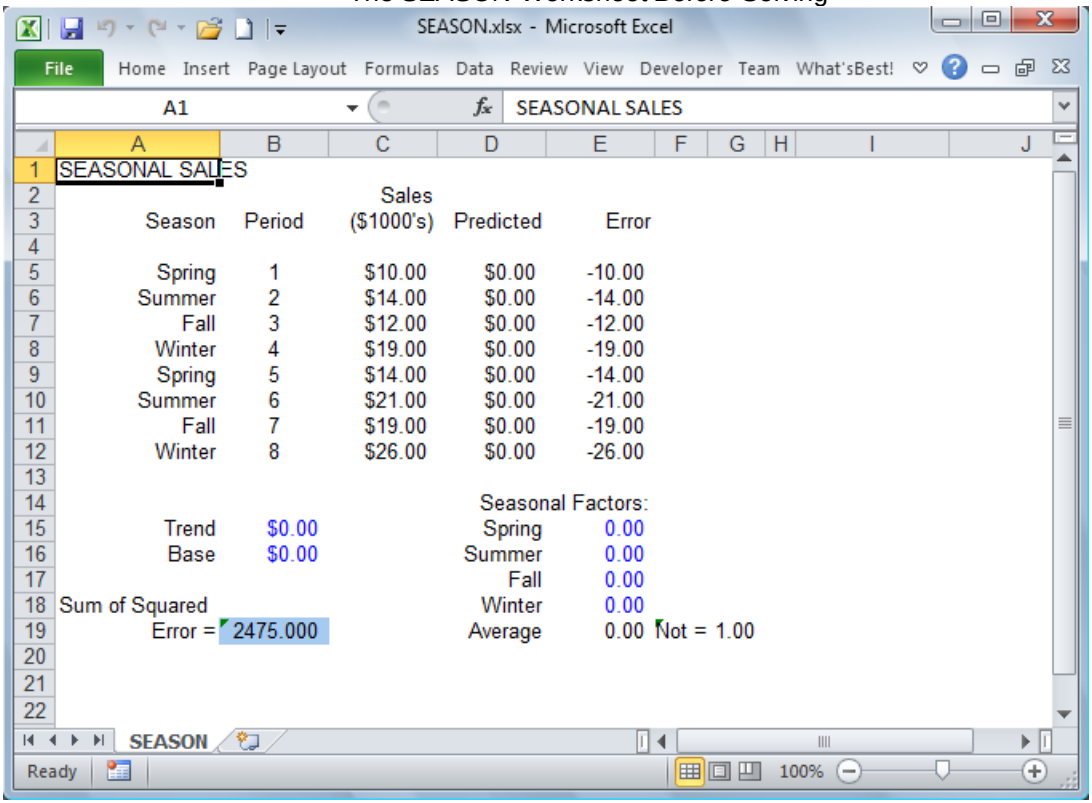
Objective of Optimization

The objective is to predict sales for the next two years based on the *Base*, *Trend*, and *Seasonal Factors* while minimizing the error.

The Worksheet

Let's look at the *SEASON* sample file.

The *SEASON* Worksheet *Before* Solving



	A	B	C	D	E
1	SEASONAL SALES				
2			Sales		
3	Season	Period	($1000's)	Predicted	Error
4					
5	Spring	1	$10.00	$0.00	-10.00
6	Summer	2	$14.00	$0.00	-14.00
7	Fall	3	$12.00	$0.00	-12.00
8	Winter	4	$19.00	$0.00	-19.00
9	Spring	5	$14.00	$0.00	-14.00
10	Summer	6	$21.00	$0.00	-21.00
11	Fall	7	$19.00	$0.00	-19.00
12	Winter	8	$26.00	$0.00	-26.00
13					
14				Seasonal Factors:	
15	Trend	$0.00		Spring	0.00
16	Base	$0.00		Summer	0.00
17				Fall	0.00
18	Sum of Squared			Winter	0.00
19	Error =	2475.000		Average	0.00 Not = 1.00
20					
21					
22					

A. Determine Adjustable Cells

The adjustable cells in the model are the *Trend* in B15, the *Base* in B16, and the four *Seasonal Factors* in E15:E18.

B. Define Best

The best cell is the *Sum of Squared Error* in B19. The formula there is:

SUMPRODUCT(E5:E12,E5:E12)

This is the sum of the range of *Errors* in E5:E12 squared.

C. Specify Constraints

The only constraint (F19) is the requirement that the *Seasonal Factors* average to 1. This is enforced by the formula:

WB(E19,"=",G19)

where E19 contains *AVERAGE*(E15:E18).

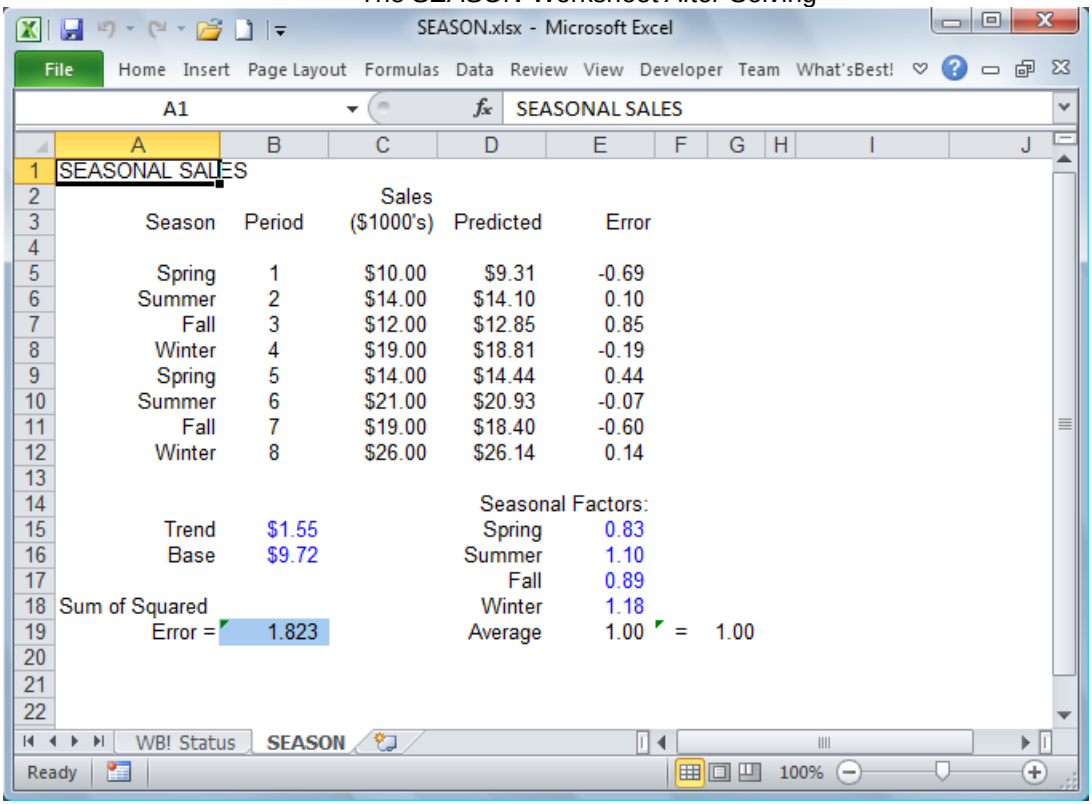
The equations to calculate the predicted points (D5:D12) are of the form:

Predicted = *Seasonal Factor* * (*Base Point* + *Period* * *Trend*)

The *Error* terms are the *Predicted* points minus the *Sales* data points. For example, the *Error* for *Spring* of *Period 1* in E5 is found with D5-C5.

Now, let's solve the model. After solving, the *WB! Status* worksheet will open in order to show you the *Nonlinearity present* warning. (This warning can be disabled from the *General Options* dialog box.) Your solved model now appears as follows:

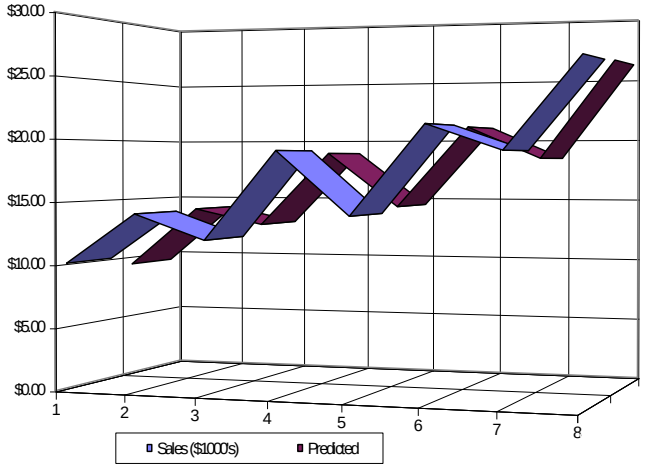
The *SEASON* Worksheet *After* Solving



	A	B	C	D	E
1	SEASONAL SALES				
2			Sales		
3	Season	Period	($1000's)	Predicted	Error
4					
5	Spring	1	$10.00	$9.31	-0.69
6	Summer	2	$14.00	$14.10	0.10
7	Fall	3	$12.00	$12.85	0.85
8	Winter	4	$19.00	$18.81	-0.19
9	Spring	5	$14.00	$14.44	0.44
10	Summer	6	$21.00	$20.93	-0.07
11	Fall	7	$19.00	$18.40	-0.60
12	Winter	8	$26.00	$26.14	0.14
13					
14				Seasonal Factors:	
15	Trend	$1.55		Spring	0.83
16	Base	$9.72		Summer	1.10
17				Fall	0.89
18	Sum of Squared			Winter	1.18
19	Error =	1.823		Average	1.00 = 1.00
20					
21					
22					

The spring seasonal factor is .83. In other words, spring sales are 83% of the average season. The trend of the solved model is $1.55. This means that, after the effects of season are taken into account, sales are increasing at an average rate of $1,550 per season.

Actual sales compared to predicted sales are shown in the graph below:



Exponential Smoothing

File name: SIMXPO.XLSX, SMOOTH.XLSX *TYPE: NONLINEAR OPTIMIZATION*

Application Profile

Exponential smoothing is one technique that you can use to predict events in the future by studying events in the past. By employing weighted averages to "smooth" past values, it lets you forecast the value in the next period.

The basic model for exponential smoothing is:

$$P_{t+1} = P_t + \alpha * (Y_t - P_t)$$

where

P_{t+1} = predicted value at period (t+1)
P_t = predicted value in period t
Y_t = actual value at period t
α (alpha) = the smoothing constant

If *Alpha* is set to 1, the forecast for the next period is based entirely on the actual value from the last period. If *Alpha* is set to 0, the actual value from the last period is completely ignored. Since neither of these cases will provide much insight into future data, we'll constrain *Alpha* to be between .01 and .99.

The Problem in Words

In order to minimize costly overstocking and inventory holding, your retail outlet needs useful predictions of future sales.

Background

For this simple exponential smoothing problem, you have sales data (in $1,000's) for eight months. You need to find *Alpha*, the smoothing constant, that minimizes the sum of the error, which in this case is the difference between the actual and predicted sales for each period.

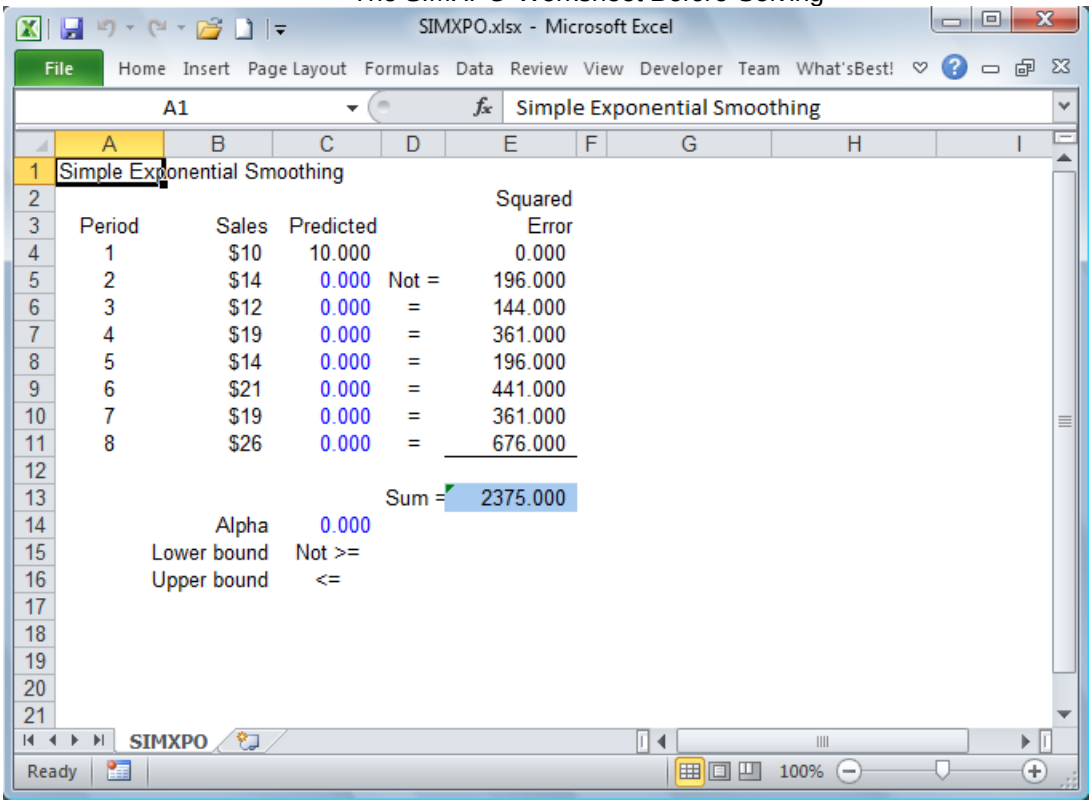
Objective of Optimization

The objective is to determine projected sales and the *Alpha* smoothing constant while minimizing the squared error.

The Worksheet

Let's look at the supplied sample file called *SIMXPO*.

The *SIMXPO* Worksheet *Before* Solving



A. Determine Adjustable Cells

The adjustable cells are the *Predicted* sales in C5:C11, and the *Alpha* smoothing term in C14. Note that *Period 1* in cell C4 is fixed at 10, since there is no previous period on which to base a prediction.

B. Define Best

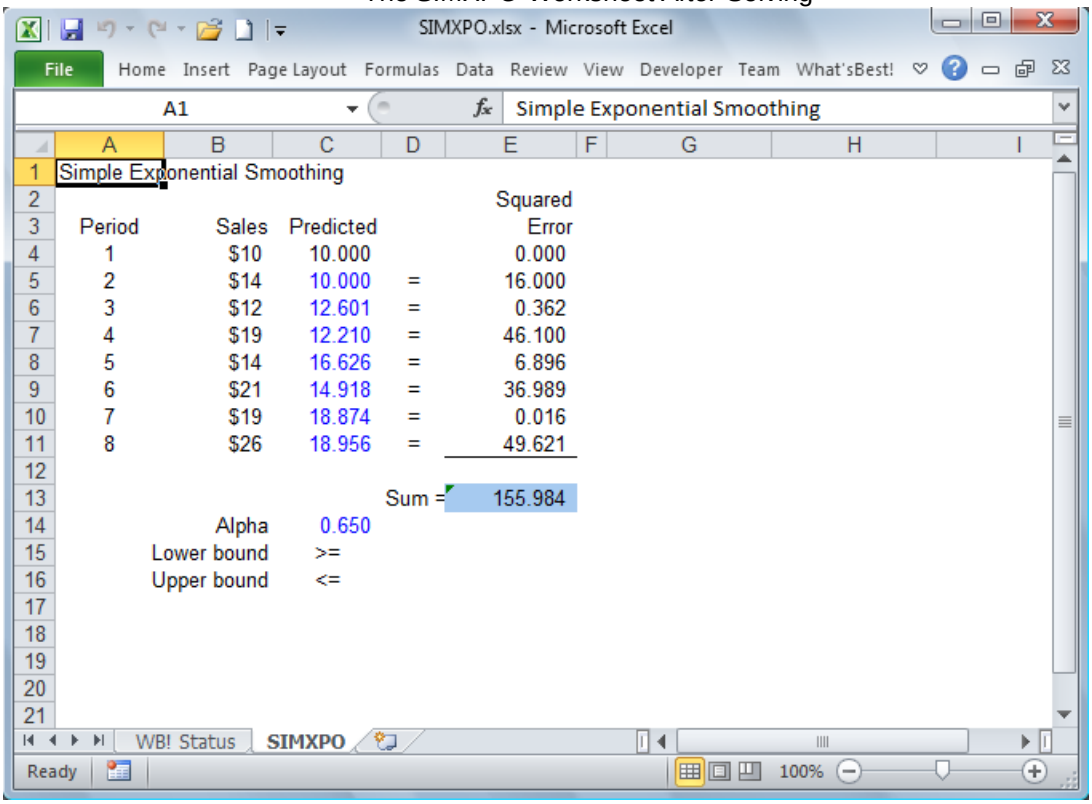
The best cell is the minimized *Sum* of *Squared Errors* (differences between *Sales* and *Predicted* in cells E4:E11) in E13.

C. Specify Constraints

The constraints in D5:D11 reproduce the basic model for exponential smoothing (see preceding page) for each *Predicted* period. In C15:C16, the *Alpha* term is bounded by .001 and .999.

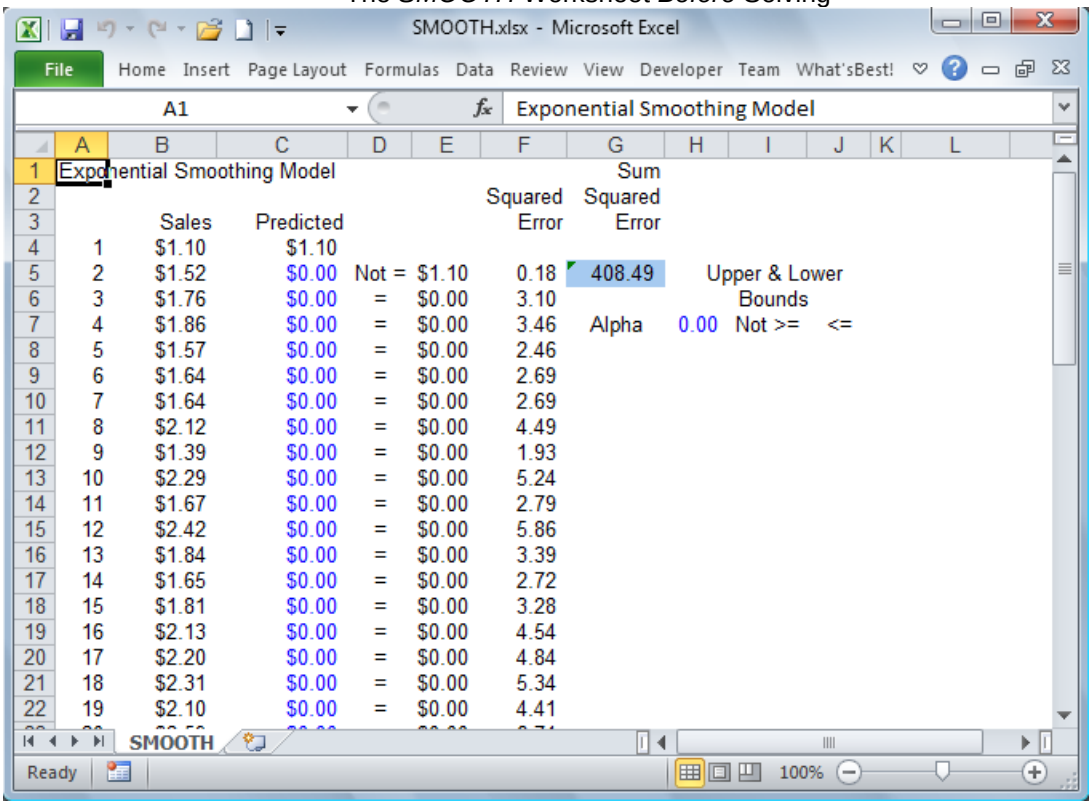
Now, you are ready to solve the model. After solving, the *WB! Status* worksheet will open in order to show you the *Nonlinearity present* warning. (This warning can be disabled from the *General Options* dialog box.) Your solved model now appears as follows:

The *SIMXPO* Worksheet *After* Solving



Following is another sample file called *SMOOTH*, which uses the same principles to analyze a much larger set of data covering 2 years:

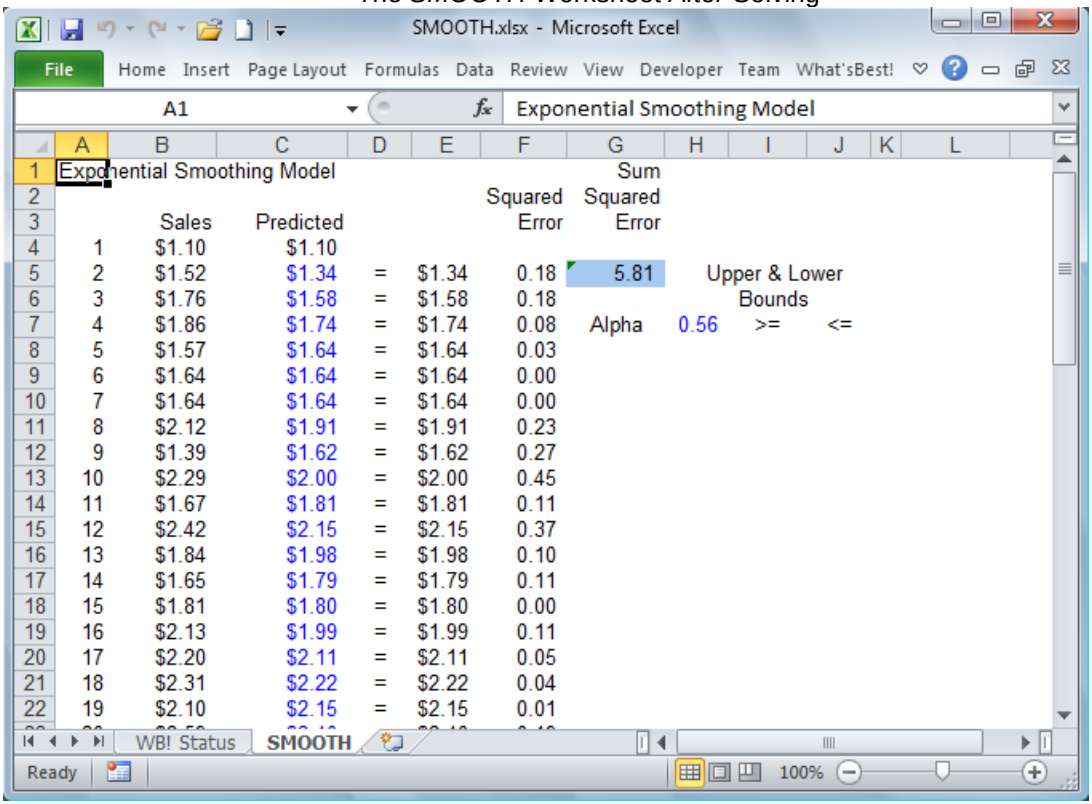
The *SMOOTH* Worksheet *Before* Solving



	A	B	C	D	E	F	G	H	I	J	K	L
1	Exponential Smoothing Model						Sum					
2						Squared	Squared					
3		Sales	Predicted			Error	Error					
4	1	$1.10	$1.10									
5	2	$1.52	$0.00	Not =	$1.10	0.18	408.49		Upper & Lower			
6	3	$1.76	$0.00	=	$0.00	3.10			Bounds			
7	4	$1.86	$0.00	=	$0.00	3.46	Alpha	0.00	Not >=	<=		
8	5	$1.57	$0.00	=	$0.00	2.46						
9	6	$1.64	$0.00	=	$0.00	2.69						
10	7	$1.64	$0.00	=	$0.00	2.69						
11	8	$2.12	$0.00	=	$0.00	4.49						
12	9	$1.39	$0.00	=	$0.00	1.93						
13	10	$2.29	$0.00	=	$0.00	5.24						
14	11	$1.67	$0.00	=	$0.00	2.79						
15	12	$2.42	$0.00	=	$0.00	5.86						
16	13	$1.84	$0.00	=	$0.00	3.39						
17	14	$1.65	$0.00	=	$0.00	2.72						
18	15	$1.81	$0.00	=	$0.00	3.28						
19	16	$2.13	$0.00	=	$0.00	4.54						
20	17	$2.20	$0.00	=	$0.00	4.84						
21	18	$2.31	$0.00	=	$0.00	5.34						
22	19	$2.10	$0.00	=	$0.00	4.41						

The solution is shown below.

The *SMOOTH* Worksheet *After* Solving



		Sales	Predicted			Squared Error
4	1	$1.10	$1.10			
5	2	$1.52	$1.34	=	$1.34	0.18
6	3	$1.76	$1.58	=	$1.58	0.18
7	4	$1.86	$1.74	=	$1.74	0.08
8	5	$1.57	$1.64	=	$1.64	0.03
9	6	$1.64	$1.64	=	$1.64	0.00
10	7	$1.64	$1.64	=	$1.64	0.00
11	8	$2.12	$1.91	=	$1.91	0.23
12	9	$1.39	$1.62	=	$1.62	0.27
13	10	$2.29	$2.00	=	$2.00	0.45
14	11	$1.67	$1.81	=	$1.81	0.11
15	12	$2.42	$2.15	=	$2.15	0.37
16	13	$1.84	$1.98	=	$1.98	0.10
17	14	$1.65	$1.79	=	$1.79	0.11
18	15	$1.81	$1.80	=	$1.80	0.00
19	16	$2.13	$1.99	=	$1.99	0.11
20	17	$2.20	$2.11	=	$2.11	0.05
21	18	$2.31	$2.22	=	$2.22	0.04
22	19	$2.10	$2.15	=	$2.15	0.01

Linearization Option: Construction Cost Estimation

File name: LINEARZ.XLSX *TYPE: NONLINEAR/LINEAR OPTIMIZATION*

Application Profile

This example demonstrates how one might develop a model for estimating construction costs, as well as performance gains resulting from What's*Best!*'s exclusive linearization option. Note, we classify this model as both nonlinear and linear. As you will see, the model, as formulated, is nonlinear. However, after invoking the linearization option in What's*Best!*, the solver automatically converts this nonlinear model to an equivalent linear model.

The Problem in Words

You are a contractor specializing in home construction. You have the opportunity to bid on a job that involves building a number of homes in a new housing development. In order to place an intelligent bid, you need to calculate your expected cost on a per home basis. Based on past experience, you know the following three factors primarily determine the construction cost of a house: total square footage, number of bedrooms, and number of baths.

You believe there is a linear relationship between these three factors and total cost. Specifically, you have come up with the following equation:

$$Total\ Cost = \beta_1 + (\beta_2 * Sq.\ Feet) + (\beta_3 * Bedrooms) + (\beta_4 * Bathrooms) \qquad (i)$$

You pull up the following historical data from projects your company has completed in the past:

Sq. Feet	Beds	Baths	Cost($000)
1500	1	1	78.5
1600	1	1	105.8
2200	2	1	149.1
2600	3	2	173.8
3000	3	2	224.4
3500	5	2	267.0
4000	4	4	302.7
5200	2	3	302.0
8200	4	4	438.7
8700	6	4	490.4

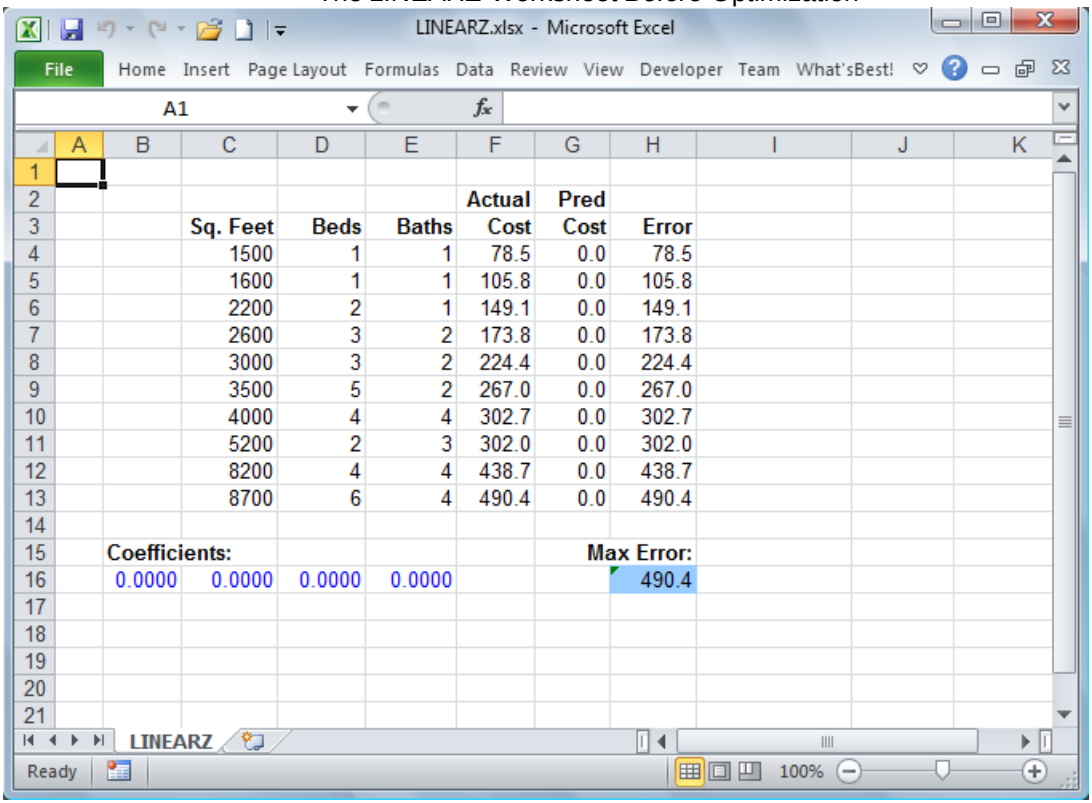
Objective of Optimization

You will use your historical data to estimate values for the beta coefficients (β_1, β_2, β_3, and β_4) in formula (*i*). You want your estimation function to avoid the case where it might be a poor predictor on one type of home. Thus, you have decided you want your model's objective to minimize the maximum error in your cost estimations. Note, that this process of fitting a linear function to a set of data points is referred to as *linear regression*.

The Worksheet

A picture of the worksheet used to solve this problem is listed below. You can find it in your sample models directory under the name *LINEARZ* sample file:

The *LINEARZ* Worksheet *Before* Optimization



	A	B	C	D	E	F	G	H	I	J	K
1											
2						Actual	Pred				
3			Sq. Feet	Beds	Baths	Cost	Cost	Error			
4			1500	1	1	78.5	0.0	78.5			
5			1600	1	1	105.8	0.0	105.8			
6			2200	2	1	149.1	0.0	149.1			
7			2600	3	2	173.8	0.0	173.8			
8			3000	3	2	224.4	0.0	224.4			
9			3500	5	2	267.0	0.0	267.0			
10			4000	4	4	302.7	0.0	302.7			
11			5200	2	3	302.0	0.0	302.0			
12			8200	4	4	438.7	0.0	438.7			
13			8700	6	4	490.4	0.0	490.4			
14											
15		Coefficients:						Max Error:			
16		0.0000	0.0000	0.0000	0.0000			490.4			
17											
18											
19											
20											
21											

A. Determine Adjustable Cells

The adjustable cells in the model are the beta coefficients. We need What's*Best!* to pick the values for the coefficients that minimize the maximum prediction error. There are four beta *Coefficients*, so we added four adjustable cells to represent them in the range B16:E16. B16 is the "intercept" coefficient, C16 is the coefficient for square feet, D16 is the coefficient on the number of bedrooms, and, finally, E16 is the coefficient on the number of baths.

B. Define Best

We want to minimize the *Max Error*, which is computed in cell H16 with the formula *MAX*(H4:H13). The range H4:H13 contains the amount of *Error* in our forecast for each of the historical cost observations. For example, cell H4 contains the formula *ABS*(F4-G4). In words, this is the absolute value of the difference of the *Actual Cost* (F4) minus the *Predicted Cost* (G4). This gives us the amount of forecast error on the first observation. Cell H16 simply takes the maximum of all these error terms. This, of course, is our objective, and we tell What's*Best!* to minimize it.

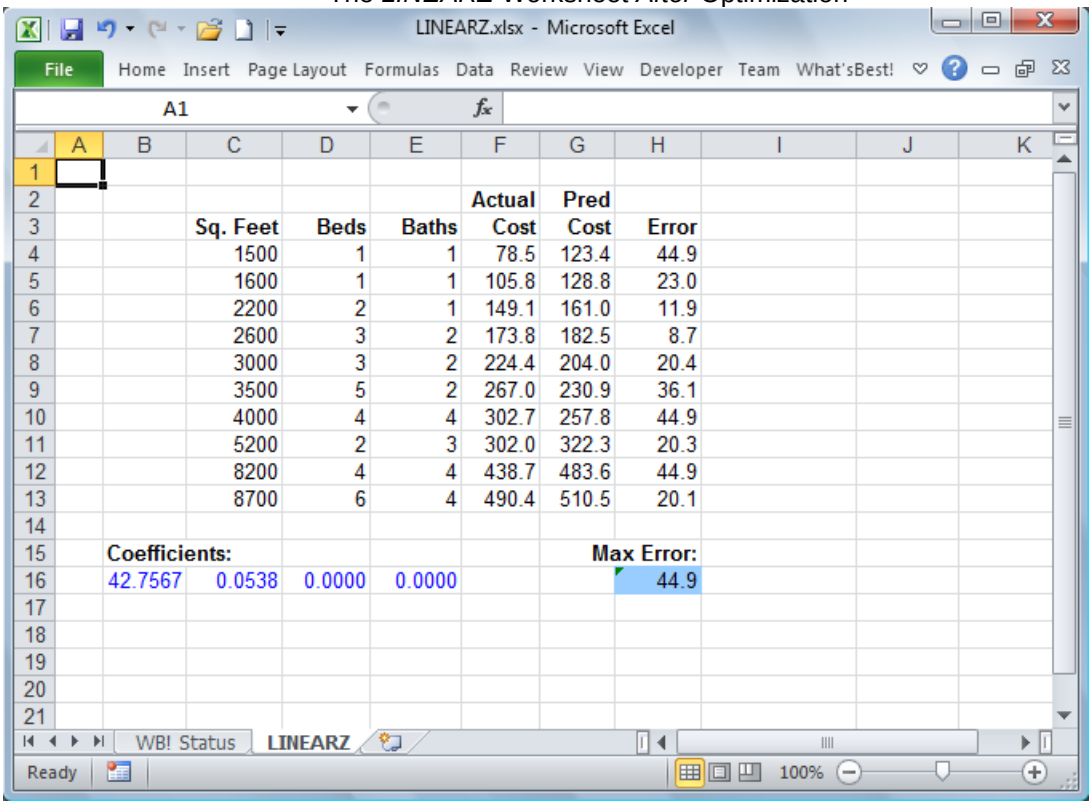
C. Specify Constraints

This is an unusual model in that it doesn't make use of any limited resources, so no constraints are required.

You are almost ready to solve the model, but first you will need to set a couple of options on the *General Options* tab. First, you will need to set the *Linerization Degree* to *Maximum* to have the What's*Best*! linearizer automatically convert this otherwise nonlinear model to linearity. Next you will need to reduce the *Big M Coefficient* from its default value of 100,000 to 1,000. Linearized models are very sensitive to this parameter. If *Big M* is too large, you can end up with poorly scaled models. On the other hand, setting this parameter too low will result in infeasible models. For this particular model a setting of 1,000 works well.

After solving, you'll see the following result.

The *LINEARZ* Worksheet *After* Optimization



	A	B	C	D	E	F	G	H	I	J	K
1											
2						Actual	Pred				
3			Sq. Feet	Beds	Baths	Cost	Cost	Error			
4			1500	1	1	78.5	123.4	44.9			
5			1600	1	1	105.8	128.8	23.0			
6			2200	2	1	149.1	161.0	11.9			
7			2600	3	2	173.8	182.5	8.7			
8			3000	3	2	224.4	204.0	20.4			
9			3500	5	2	267.0	230.9	36.1			
10			4000	4	4	302.7	257.8	44.9			
11			5200	2	3	302.0	322.3	20.3			
12			8200	4	4	438.7	483.6	44.9			
13			8700	6	4	490.4	510.5	20.1			
14											
15		Coefficients:						Max Error:			
16		42.7567	0.0538	0.0000	0.0000			44.9			
17											
18											
19											
20											
21											

Thus, our optimal solution minimizes the maximum error at 17.8 using the following formula:

$$\textit{Total Cost} = 5.6896 + (0.0373 * \textit{Sq. Feet}) + (19.5877 * \textit{Beds}) + (15.0613 * \textit{Baths})$$

Linearization

Note that as formulated, this model is nonlinear. This is the result of the *ABS()* and *MAX()* functions used in computing the error terms and the objective value, respectively. Both *ABS()* and *MAX()* can create problems for the nonlinear solver in that they aren't continuously differentiable. Both functions have "kinks" in their graphs at various points that the nonlinear solver is unable to "see" around. With a fair amount of effort, one could linearize this model by adding a number of additional variables and constraints. However, the linearization option in What's*Best!* can do this for you automatically.

For illustration, we ran this model twice—once with linearization off and once with it on—using the two *Linearization Degree* settings of *None* and *Maximum*. The adjustable cells were reset to 0 for each run. Here are some comparative results:

Linearization:	**Off**	**On**
Adjustables	4	4
Constraints	0	61
Integers (Bin)	0	20
Variables	25	76
Nonlinears	20	0
Solution Time (sec)	1	<1

As you can see, the model is considerably larger with linearization enabled, because it has internally increased the number of variables (from 25 to 76), integers (from 0 to 20), and constraints (from 0 to 61). The crucial observation, however, is that the number of nonlinears goes from 20 to 0. Thus, the linearization completely removes all the nonlinearities from the model. This allows What's*Best!* to invoke its faster and more accurate linear solver, which reduces total solution time down to less than 1 second.

In conclusion, if your model has nonlinearities that can all be handled by the What's*Best!* linearizer, solution times can be reduced dramatically by engaging the linearization option. For more information on setting the linearization options, see *Selecting Options*. For more information on the linearization process, see *Overview of Mathematical Modeling*. For a list of the functions that can be linearized automatically by What's*Best!*, see *Functions and Operators*.

Stratified Sampling

File name: SAMPLEWB.XLSX *TYPE: NONLINEAR OPTIMIZATION*

Application Profile

This model is useful wherever you need to determine an optimal statistical sampling strategy. Opinion polling and market research are two areas in which models similar to this one might be used.

The Problem in Words

In your work in market research for an appliance manufacturer, you must determine the consumer's trend toward purchasing your product. You need to determine two population means using the smallest (least costly) sample likely to give reliable results. To reduce error, you break up the population by income groups. You're asking two questions:

- How much do you plan to spend on major appliances next year?
- On a scale of 1 to 100, how likely are you to buy our brand?

Background

The population data looks like this:

			Variance in Response	
Stratum	**Income Group**	**Population**	**Question 1**	**Question 2**
1	$50,001 +	400,000	5	1
2	35,001 - 50,000	300,000	5	2
3	22,501 - 35,000	200,000	5	4
4	under 22,500	100,000	5	8

You have determined your tolerable upper limits of variance for both sample means, and you know the variance by question for each group.

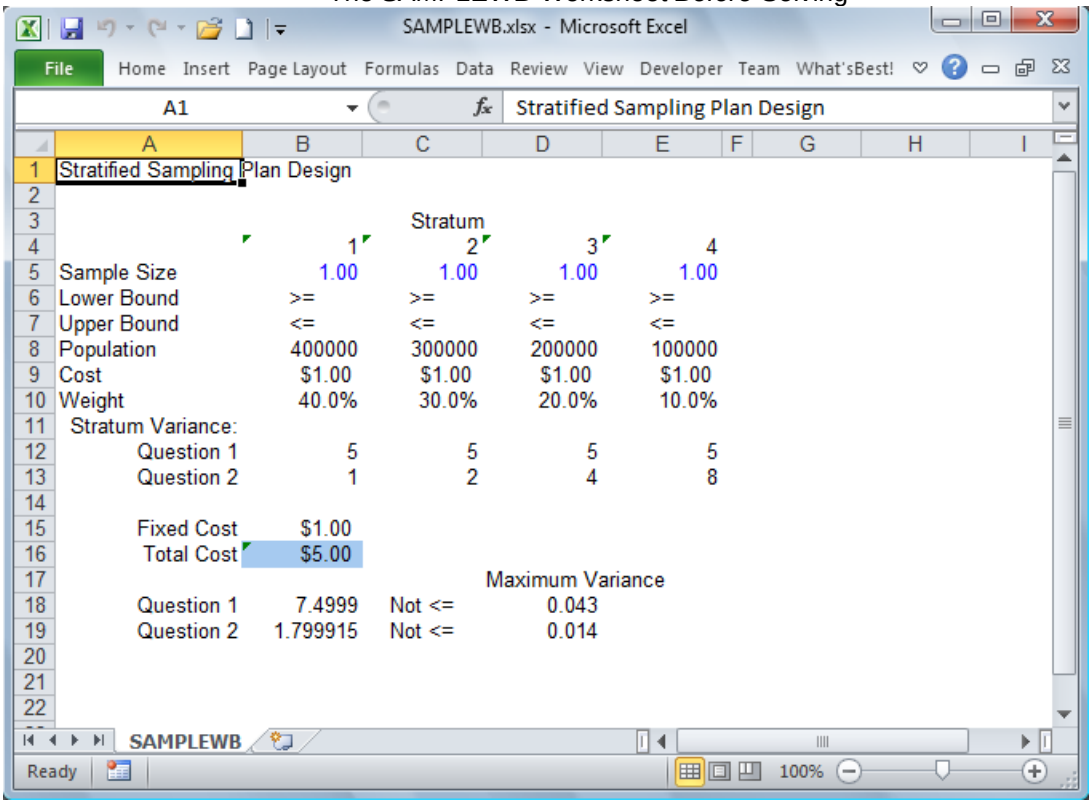
Objective of Optimization

The objective is to determine the best sample size for each stratum while minimizing the cost to obtain the sample.

The Worksheet

Let's look at the sample file called *SAMPLEWB*:

The *SAMPLEWB* Worksheet *Before* Solving



	A	B	C	D	E
1	Stratified Sampling Plan Design				
2					
3			Stratum		
4		1	2	3	4
5	Sample Size	1.00	1.00	1.00	1.00
6	Lower Bound	>=	>=	>=	>=
7	Upper Bound	<=	<=	<=	<=
8	Population	400000	300000	200000	100000
9	Cost	$1.00	$1.00	$1.00	$1.00
10	Weight	40.0%	30.0%	20.0%	10.0%
11	Stratum Variance:				
12	Question 1	5	5	5	5
13	Question 2	1	2	4	8
14					
15	Fixed Cost	$1.00			
16	Total Cost	$5.00			
17				Maximum Variance	
18	Question 1	7.4999	Not <=	0.043	
19	Question 2	1.799915	Not <=	0.014	

A. Determine Adjustable Cells

The adjustable cells are B5:E5, the *Sample Sizes* for each stratum.

B. Define Best

The best cell is the minimized cost of sampling in cell B16. The formula here adds a *Fixed Cost* (a nominal $1.00 in this example) to the sum of sample sizes times the cost per unit sampled.

C. Specify Constraints

There are four types of constraints in the model.

In B6:E6, a *Lower Bound* of .001 is placed on the sample sizes. This ensures that optimizer errors caused by division by zero will not occur.

In B7:E7, the sample size is required to be no larger than the population from which it is taken.

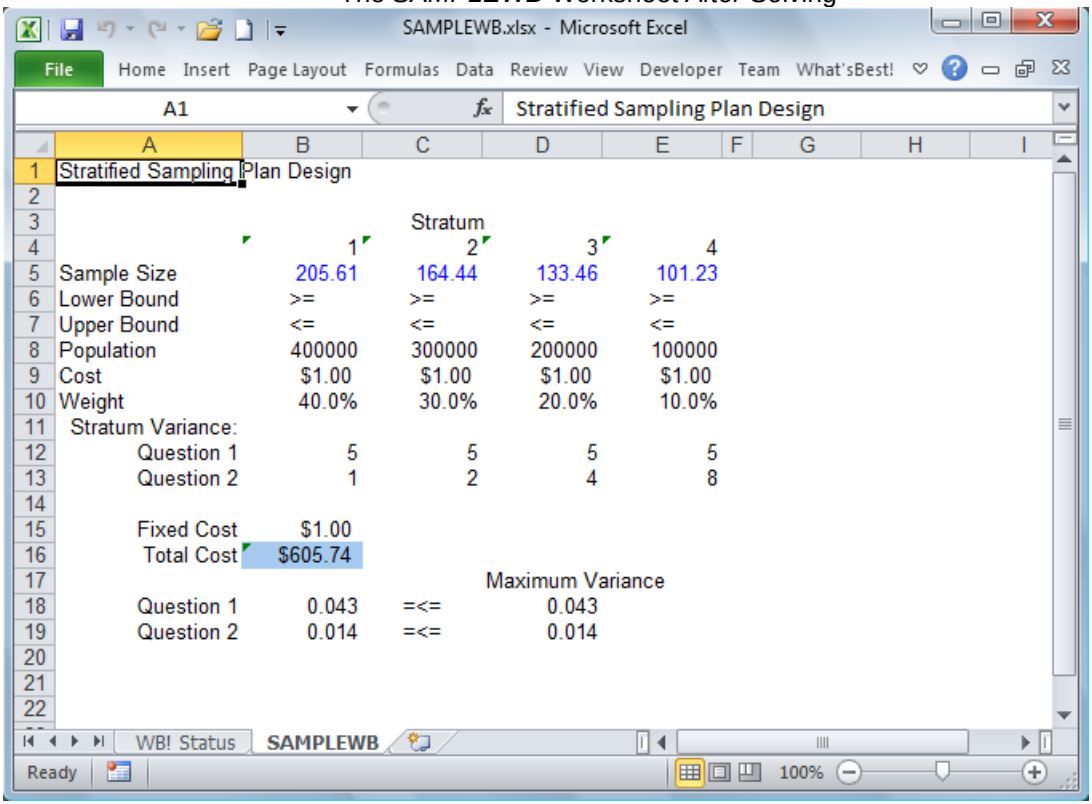
In C18 and C19, the variances on the two measured features are bounded on the upper end, so as to force reliable estimates. The variances are computed in B18 and B19. For instance, the variance computed for *Question 1* in B18 is found by means of the formula:

($B$10^2*B12^2)/$B$5+($C$10^2*C12^2)/$C$5+($D$10^2*D12^2)/$D$5+
($E$10^2*E12^2)/$E$5-($B$10*B12^2/$B$8+$C$10*C12^2/$C$8+
$D$10*D12^2/$D$8+$E$10*E12^2/$E$8)

That is, for each *Stratum*, its *Weight*^{2} times *Variance*^{2} is divided by its sample size. The resulting figures are summed. For each stratum, from that result is subtracted the *Weight* times *Variance*^{2} divided by *Population* size. This formulation is from *Sampling Techniques*, W.G. Cochran, 2nd Ed., Wiley, New York, 1963, to which interested readers may refer.

Now, you are ready to solve the model. After solving, the *WB! Status* worksheet will open in order to show you the *Nonlinearity* present warning. This warning can be shut off from the *General Options* dialog box. Your solved model now appears as follows:

The *SAMPLEWB* Worksheet *After* Solving



	A	B	C	D	E
1	Stratified Sampling Plan Design				
2					
3			Stratum		
4		1	2	3	4
5	Sample Size	205.61	164.44	133.46	101.23
6	Lower Bound	>=	>=	>=	>=
7	Upper Bound	<=	<=	<=	<=
8	Population	400000	300000	200000	100000
9	Cost	$1.00	$1.00	$1.00	$1.00
10	Weight	40.0%	30.0%	20.0%	10.0%
11	Stratum Variance:				
12	Question 1	5	5	5	5
13	Question 2	1	2	4	8
14					
15	Fixed Cost	$1.00			
16	Total Cost	$605.74			
17				Maximum Variance	
18	Question 1	0.043	=<=	0.043	
19	Question 2	0.014	=<=	0.014	
20					
21					
22					

The solution tells you the minimal sample sizes that yield reliable results within the tolerances you've set. This data could be used to calculate the probable success of a marketing plan or in forecasting demand for next year's production.

Car Pricing

File name: PRICING.XLSX *TYPE: NONLINEAR OPTIMIZATION*

The Problem in Words

You manufacture three automobile models, each with its own price structure and miles-per-gallon rating, at your five plants.

An increase in sales of any one model results in a small decrease in sales of the two others. The models compete for production capacity in your five plants. Also, the federal government has imposed an lower limit on the average "fleet mileage" of your entire production, which makes sales of low-mileage cars more desirable.

You need to set the prices of the three models of automobiles, so profit is maximized at the five plants, while staying within the federal fleet gas mileage requirement of 24 miles per gallon.

Background

Your known values are the cost of manufacturing each model at each plant, the miles per gallon for each model, and the annual production capacity at each plant.

You also know price-quantity relationships among the models, called *demand curves* by economists. These demands are dependent upon the price of each car related to the prices of the other cars.

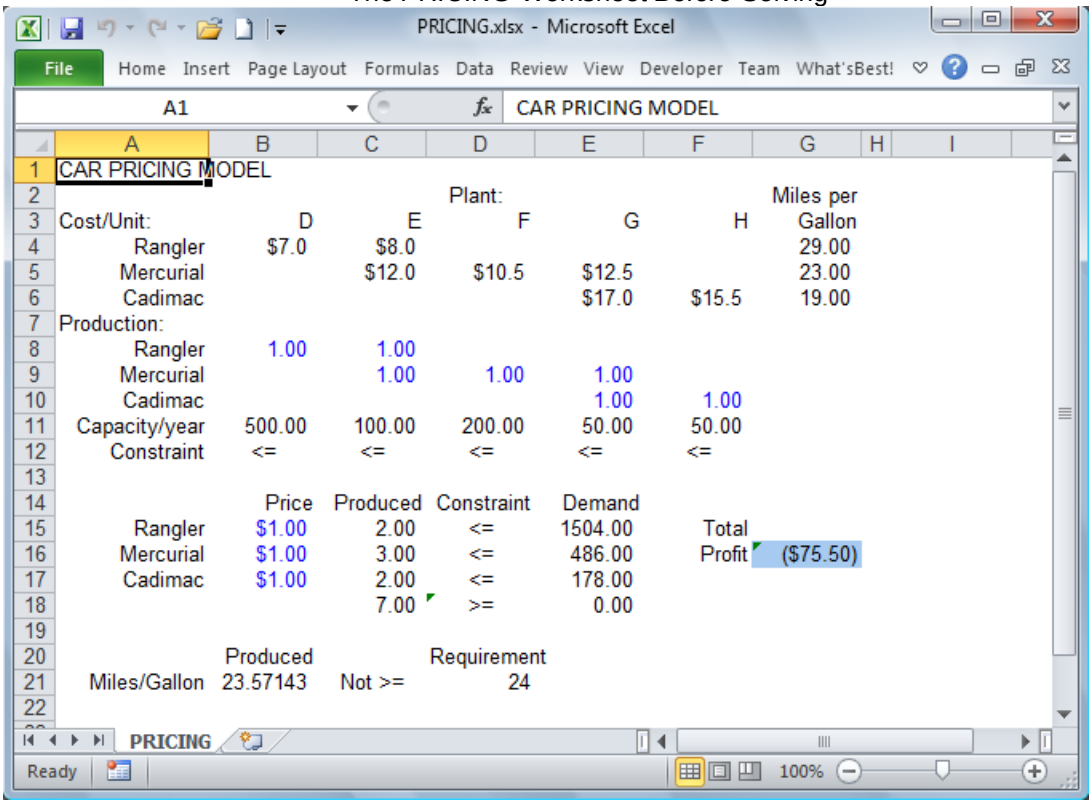
Objective of Optimization

The objective is to determine the best price for each car to maximize profit while conforming to all regulations and constraints.

The Worksheet

Let's look at the sample file called *PRICING*.

The *PRICING* Worksheet *Before* Solving



	A	B	C	D	E	F	G	H	I
1	CAR PRICING MODEL								
2				Plant:			Miles per		
3	Cost/Unit:	D	E	F	G	H	Gallon		
4	Rangler	$7.0	$8.0				29.00		
5	Mercurial		$12.0	$10.5	$12.5		23.00		
6	Cadimac				$17.0	$15.5	19.00		
7	Production:								
8	Rangler	1.00	1.00						
9	Mercurial		1.00	1.00	1.00				
10	Cadimac				1.00	1.00			
11	Capacity/year	500.00	100.00	200.00	50.00	50.00			
12	Constraint	<=	<=	<=	<=	<=			
13									
14		Price	Produced	Constraint	Demand				
15	Rangler	$1.00	2.00	<=	1504.00	Total			
16	Mercurial	$1.00	3.00	<=	486.00	Profit	($75.50)		
17	Cadimac	$1.00	2.00	<=	178.00				
18			7.00	>=	0.00				
19									
20		Produced		Requirement					
21	Miles/Gallon	23.57143	Not >=	24					
22									

A. Determine Adjustable Cells

The adjustable cells in the model are the quantity of each automobile to produce at each plant (B8:F10) and the *Price* (in thousands) to charge for each unit (B15:B17).

B. Define Best

The best cell is the maximized *Total Profit* in G16. The formula there is:

SUMPRODUCT(B15:B17,C15:C17)-(B4*B8+C4*C8+C5*C9+D5*D9+
E5*E9+E6*E10+F6*F10)

This is profit minus production costs.

C. Specify Constraints

The constraints in the model enforce the three requirements.

In B12:F12, yearly production at each plant (the sum of car models produced at a given plant) is required to be no greater-than that plant's capacity. For instance, the constraint for *Plant D* (B12), is *WB*(*SUM*(B8:B10),"<=",B11).

In D15:D17, the total quantity *Produced* of each auto (C15:C17) is forced to be no greater-than the quantity required according to that model's demand curve formula (E15:E17). The estimates for these demand curves have been supplied by your econometrics department. For example, the demand curve formula for the *Mercurial* (E16) is:

```
495-13*B16+3*B15+B17
```

Note that the demand curves for the three makes of auto are interdependent (the sales of one influence demand for the other two).

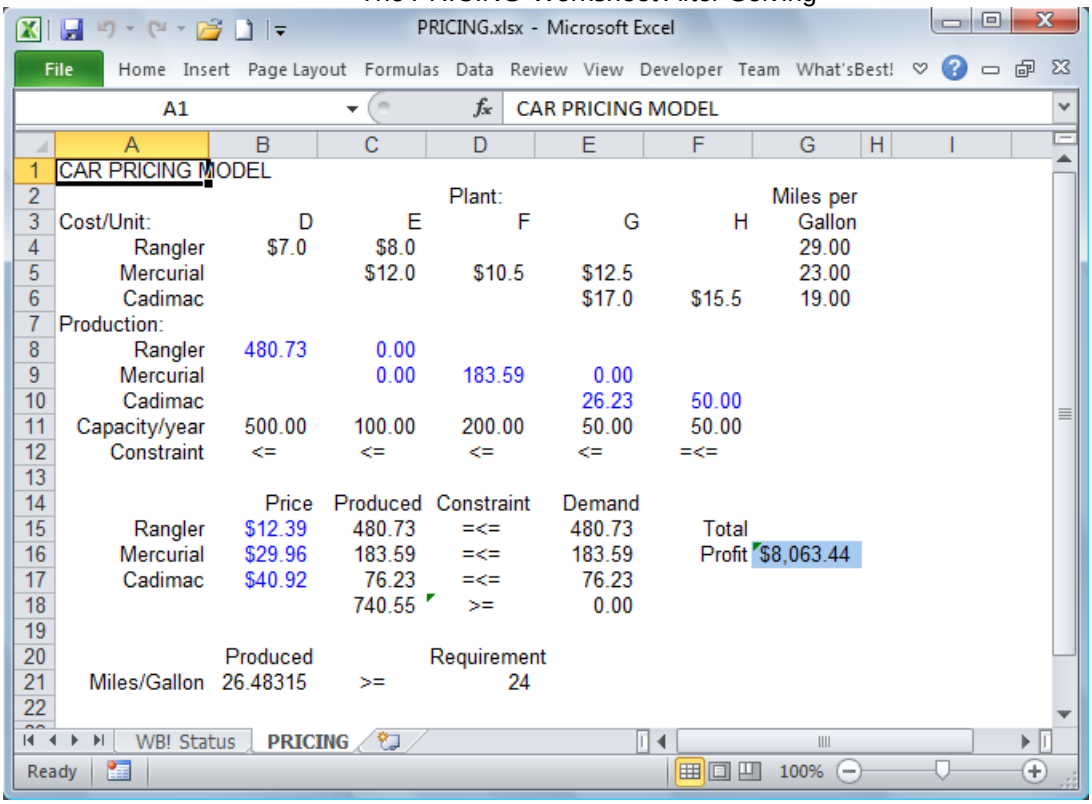
Finally, cell C20 contains the constraint *WB*(B20,">=",D20), which forces the average *Miles/Gallon* across the produced quantity of all three makes (B20) to be at least 24. The formula in B20 is:

```
SUMPRODUCT(G4:G6,C15:C17)/SUM(C15:C17)
```

Note: The federal guideline is expressed in miles per gallon. The constraint could more properly be expressed as *gallons per mile*: as if each car sold were driven one mile, a measure taken of total gallons used, and the total cars sold divided by total gallons used. This is because using miles per gallon assumes each car running until the fuel tank is empty. Obviously, the higher-MPG car would go further on a tank of fuel, skewing the overall production in its favor.

Now, you are ready to solve the model. After solving, the *WB! Status* worksheet will open in order to show you the *Nonlinearity present* warning. (This warning can be shut off from the *General Options* dialog box.) Your solved model appears as follows.

The *PRICING* Worksheet *After* Solving



	A	B	C	D	E	F	G	H	I
1	CAR PRICING MODEL								
2				Plant:			Miles per		
3	Cost/Unit:	D	E	F	G	H	Gallon		
4	Rangler	$7.0	$8.0				29.00		
5	Mercurial		$12.0	$10.5	$12.5		23.00		
6	Cadimac				$17.0	$15.5	19.00		
7	Production:								
8	Rangler	480.73	0.00						
9	Mercurial		0.00	183.59	0.00				
10	Cadimac				26.23	50.00			
11	Capacity/year	500.00	100.00	200.00	50.00	50.00			
12	Constraint	<=	<=	<=	<=	=<=			
13									
14		Price	Produced	Constraint	Demand				
15	Rangler	$12.39	480.73	=<=	480.73	Total			
16	Mercurial	$29.96	183.59	=<=	183.59	Profit	$8,063.44		
17	Cadimac	$40.92	76.23	=<=	76.23				
18			740.55	>=	0.00				
19									
20		Produced		Requirement					
21	Miles/Gallon	26.48315	>=	24					
22									

Media Buying

File name: MEDIA.XLSX *TYPE: LINEAR OPTIMIZATION*

Application Profile

Media selection (or buying) is the problem of finding the best way to deliver the desired number of advertising exposures to the target audience by determining the most efficient combination of advertising media purchases. Even for small media budgets, there can be literally thousands of possible media schedules from which to choose. The task is to select from this set an effective schedule to become the media plan.

What's*Best!* provides a natural format for analyzing a media selection problem. You can use it to find the media mix that will maximize the number of effective exposures subject to a set of constraints: the ad budget, minimum and maximum media availabilities, minimum desired exposure rates, etc.

A great advantage of What's*Best!* is that new plans can be generated quickly to show the significance of changes made in problem specifications.

The Problem in Words

You're a media buyer for an advertising agency. You need to minimize the cost while satisfying minimum exposure needs.

Background

There are six target markets, *Groups* 1 through 6, and each requires a specified number of media exposures. There are five different media sources: the *Times*, *Mirror*, *Tribune*, *Herald*, and *Post*, over which the advertising dollars are to be allocated. Each of the media sources offers various combinations of exposures per dollar spent for each of the different target groups.

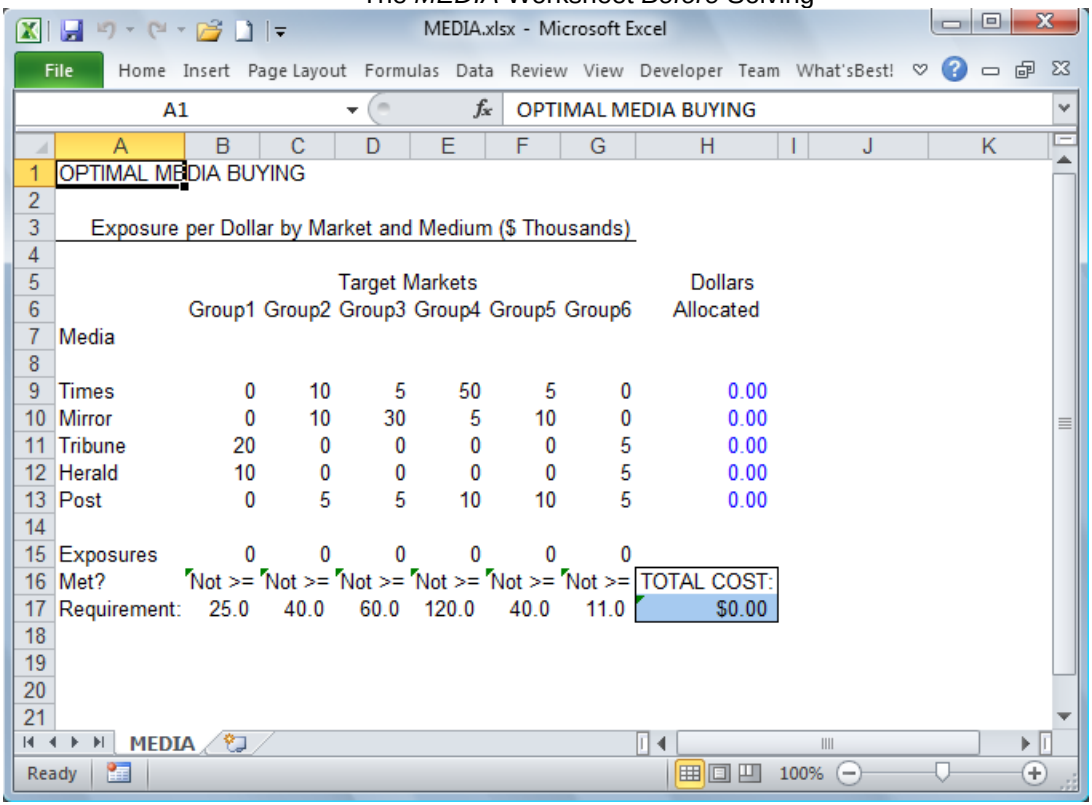
Objective of Optimization

The objective is to determine how much of each of the media sources should be purchased to meet the exposure requirements at lowest cost.

The Worksheet

Let's open the *MEDIA* sample file and look at its structure and formulas.

The *MEDIA* Worksheet *Before* Solving



	A	B	C	D	E	F	G	H
1	OPTIMAL MEDIA BUYING							
2								
3	Exposure per Dollar by Market and Medium ($ Thousands)							
4								
5				Target Markets				Dollars
6		Group1	Group2	Group3	Group4	Group5	Group6	Allocated
7	Media							
8								
9	Times	0	10	5	50	5	0	0.00
10	Mirror	0	10	30	5	10	0	0.00
11	Tribune	20	0	0	0	0	5	0.00
12	Herald	10	0	0	0	0	5	0.00
13	Post	0	5	5	10	10	5	0.00
14								
15	Exposures	0	0	0	0	0	0	
16	Met?	Not >=	Not >=	Not >=	Not >=	Not >=	Not >=	TOTAL COST:
17	Requirement:	25.0	40.0	60.0	120.0	40.0	11.0	$0.00
18								
19								
20								
21								

The exposures per dollar are listed for each media source and each media source in B9:G13. Notice that each media source reaches different combinations of target groups per dollar spent. Cells B15:G15 show the minimum required exposures for each of the target groups.

A. Determine Adjustable Cells

The adjustable cells in this model are the *Dollars Allocated* to each media source (cells H9 through H13).

B. Define Best

The best solution is the one that meets *Requirements* at the lowest cost. The *Total Cost* cell (H17) shows that every change in *Dollars Allocated* will have a direct bearing on cost.

C. Specify Constraints

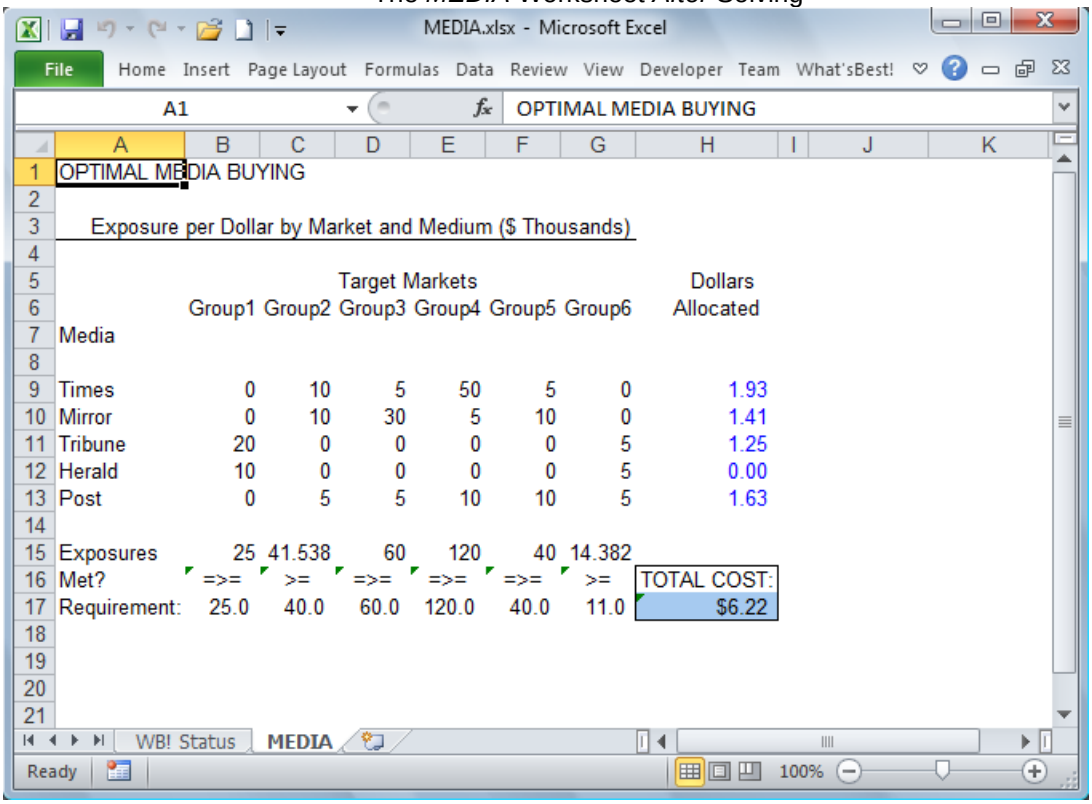
The constraints are that the minimum *Requirements* be met. The *Exposures* formulas in B17:G17 are the sumproduct of the number of exposures per dollar for a media source and the amount in dollars being allocated to that source. The formulas in the row *Met?* (B16:G16) force these amounts to be greater-than-or-equal-to the required exposures for each targeted group (B15:G15).

"What If?" vs. What's*Best!*

Work on a conventional "What If?" spreadsheet solution to this problem by adjusting the *Dollars Allocated* shown in cells H9:H13. You've reached a viable solution when cells B16:G16 return ">=" or "=>=". Judge any solution that meets all the exposure requirements by minimizing *Total Cost* (H18). When you arrive at a reasonable answer, jot down your total cost figure.

Now, let's solve the model with What's*Best!*.

The *MEDIA* Worksheet *After* Solving



As you can see, the What's*Best!* solution has a total cost of $6,220. It results in surplus exposures in only two target markets, *Group* 2 and *Group* 6. It proposes no media purchases from the *Herald*, which, offering only 15 total exposures per dollar, is not a bargain.

D. Dual Values

Dual values can be applied to cells H9:H13, giving the cost penalty associated with using a media source when the optimizer determines zero dollars should be allocated for that source. In this example, the optimizer determines that zero dollars should be allocated for the *Herald*. By using the *Advanced*|*Dual* command, you would find that the total cost, $6,220, will be increased by $500 for every $1,000 spent on the *Herald*. Dual values for cells B16:G16 would give the dollar cost of requiring a thousand more exposures.

Multi-Period Inventory Management

File name: INVENT.XLSX *TYPE: LINEAR OPTIMIZATION*

Application Profile

What if your product is made from materials that are in limited supply and require special or expensive storage facilities? In such a situation, you have to keep purchasing and inventory costs down, but still keep enough stock to meet demand. If you use perishable resources or materials subject to supply limitations, you may encounter problems like this one.

The Problem in Words

Your production demands vary over thirteen time periods. Raw materials, available at varying costs from three different suppliers, must occasionally be held in sufficient quantity from time period to time period to meet production needs. A holding cost associated with inventory held over further complicates the picture. How can you determine how much inventory to hold and which supplier to use while minimizing total cost and meeting your customers' demand for the product?

Background

You know the unit demand for each of the thirteen time periods. For instance, the demand for *Time Period 1* must be met by using some combination of sources 1, 2, and 3. The ending inventory is equal-to starting inventory plus the sum of amounts purchased from each source less the unit demand.

You must not only satisfy the demand for each period, but also minimize the amount of inventory held over. Obviously, it's in your interest to purchase as much as possible from your least expensive source, *Source 1*. However, demand in some periods exceeds *Source 1*'s capacity.

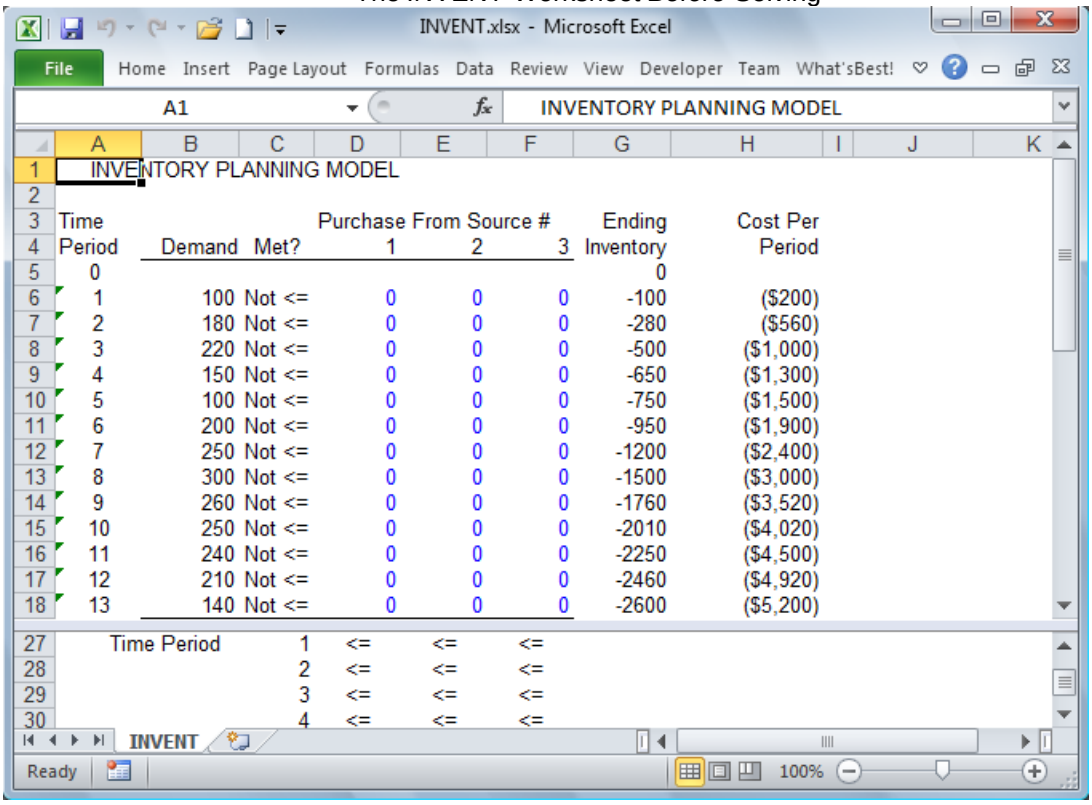
Objective of Optimization

The objective in the *Inventory* model is to minimize the total cost and determine how much material to buy from each source in each time period to meet customer demand at lowest purchase and holding costs.

The Worksheet

Let's look at the *INVENT* worksheet and note that *Source 1* is obviously the best bargain (D21). However, as demand rises from period to period (B6:B18), *Source 1*, with a capacity of 180, won't suffice to meet demand.

The *INVENT* Worksheet *Before* Solving



A. Determine Adjustable Cells

The adjustable cells (D6:F18) are the amounts purchased for each time period from each of the three supply sources.

B. Define Best

The best solution to this inventory problem consists of the combination of purchases from each source in each time period that results in the minimum *Total Cost* (H21). The formula in this cell, *SUM*(H6:H18), is the sum of period costs. Each period cost is composed of the sum of the purchasing and inventory holding costs for that period.

C. Specify Constraints

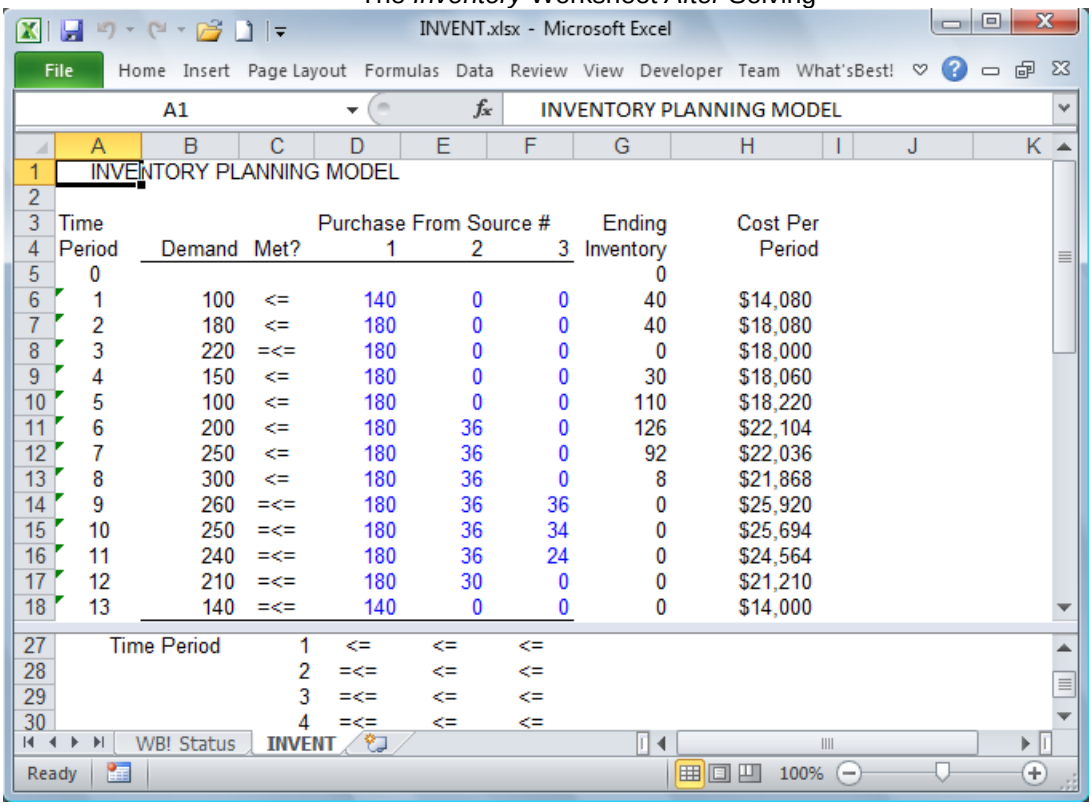
The constraints are that there must be sufficient supply for each time period to satisfy your demand and the capacity of the three suppliers must not be exceeded.

The first constraints are found in C6:C18. Look at the formula in C6 (*WB*(B6,"<=",G5+*SUM*(D6:F6)). The amount held over from the period before (G5) is added to the sum of purchases from the three sources, and required to be greater-than-or-equal-to total demand for period 1 (B6).

The second group of constraints (D27:F39) is that supply sources not be used beyond their capacity, is in cells D27:F39. In cell D27, for instance, is the formula *WB*(D6,"<=",$D$20). D6, the amount purchased from *Source 1* in *Period 1*, is required to be less-than-or-equal-to D20, the supply capacity per period.

Now, let's solve the model.

The *Inventory* Worksheet *After* Solving



	A	B	C	D	E	F	G	H
1	INVENTORY PLANNING MODEL							
2								
3	Time			Purchase From Source #			Ending	Cost Per
4	Period	Demand	Met?	1	2	3	Inventory	Period
5	0						0	
6	1	100	<=	140	0	0	40	$14,080
7	2	180	<=	180	0	0	40	$18,080
8	3	220	=<=	180	0	0	0	$18,000
9	4	150	<=	180	0	0	30	$18,060
10	5	100	<=	180	0	0	110	$18,220
11	6	200	<=	180	36	0	126	$22,104
12	7	250	<=	180	36	0	92	$22,036
13	8	300	<=	180	36	0	8	$21,868
14	9	260	=<=	180	36	36	0	$25,920
15	10	250	=<=	180	36	34	0	$25,694
16	11	240	=<=	180	36	24	0	$24,564
17	12	210	=<=	180	30	0	0	$21,210
18	13	140	=<=	140	0	0	0	$14,000

27	Time Period	1	<=	<=	<=
28		2	=<=	<=	<=
29		3	=<=	<=	<=
30		4	=<=	<=	<=

What's*Best!* has arrived at a minimized *Total Cost* of $263,836 (H21).

This simple model is intended to demonstrate the basic principles in modeling inventory. Add fixed ordering and shipping costs for each source, etc., for a model more reflective of the real world.

Product Mix

File name: PRODMIX.XLSX, PMIXMAC.XLSM *TYPE: LINEAR OPTIMIZATION*

Application Profile

In product mix problems, the objective is to find the most profitable allocation of a set of limited resources over a set of desired products or activities. Virtually every manufacturer faces this situation in some form or another.

This example involves the conversion of raw materials into manufactured goods. An agricultural example might involve the use of land, seed, water, labor, and fertilizer so as to maximize farm output. Product mix problems often serve as the "building blocks" for more complex applications.

The Problem in Words

You are a manufacturer producing six products from six materials with each product requiring a different combination and amount of raw materials.

Background

You know the profit per unit of each product and the quantity of each raw material required for a single unit of each product. For instance, each unit of product 2 contributes $45 to profit and uses 4 units of steel, 5 of wood, 3 of plastic, etc. in its manufacture.

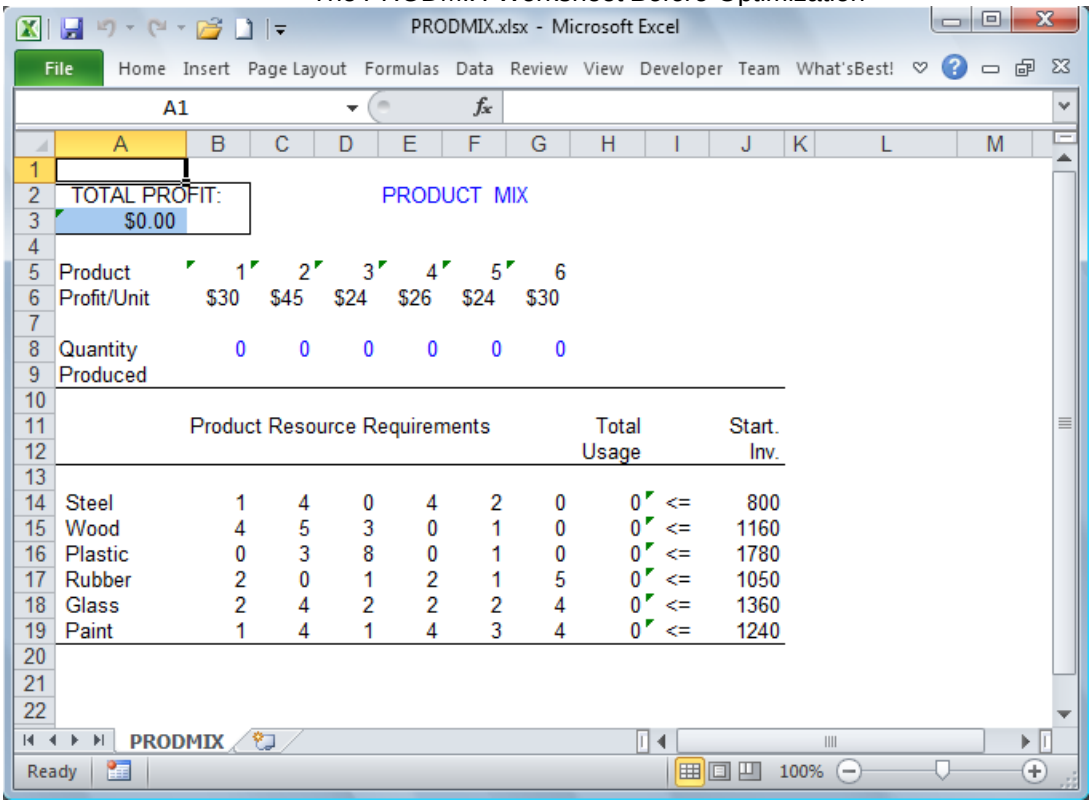
Objective of Optimization

The objective of optimization is to maximize the profits that can be realized from manufacturing products using only the current inventory of raw materials.

The Worksheet

Let's look at the *PRODMIX* sample file.

The *PRODMIX* Worksheet *Before* Optimization



A. Determine Adjustable Cells

The adjustable cells in this model (B8:G8) contain the *Quantity Produced* for each product.

B. Define Best

The best solution is the one that results in the maximum *Total Profit* (A3). Take a moment to examine the formula in this cell:

SUMPRODUCT(B6:G6,B8:G8)

That is interpreted as *Profit/Unit* of *Product 1* * *Quantity Produced* of *Product 1* + *Profit/Unit* of *Product 2* * *Quantity Produced* of *Product 2* and so on. You want to find a combination of values (B8:G8) that will yield the highest *Total Profit* (A3).

Note:	The *SUMPRODUCT* method of expression eliminates the need for lengthy formulas, while simplifying later addition of other products. Inserting rows (not appending) within the ranges allows you to enlarge them without rewriting the Total Cost formula. In writing small models, you may find it easier to write such formulas without *SUMPRODUCT*. In large models, or ones you intend to enlarge, it's probably best to use *SUMPRODUCT*.

C. Specify Constraints

The constraints in this problem (I14:I19) must reflect the requirement that total usage of raw materials remain less-than-or-equal-to the quantities of raw materials in stock. This necessitates a separate constraint for each of the six raw materials. For instance, the constraint in I14, *WB* (H14, "<=", J14) requires that the *Total Usage* (H14) be less-than-or-equal-to the *Starting Inventory* (J14).

Total Usage (H14:H19) is determined based on the *Product Resource Requirements* defined within the model (B14:G19). For instance, the formula in H14 for *Total Steel Usage* is *SUMPRODUCT*(B14:G14,B$8:G$8). That is interpreted as the *Steel* requirement per unit of *Product 1* * *Quantity Produced* of *Product 1* + *Steel* requirement per unit of *Product 2* * *Quantity Produced* of *Product 2* and so on.

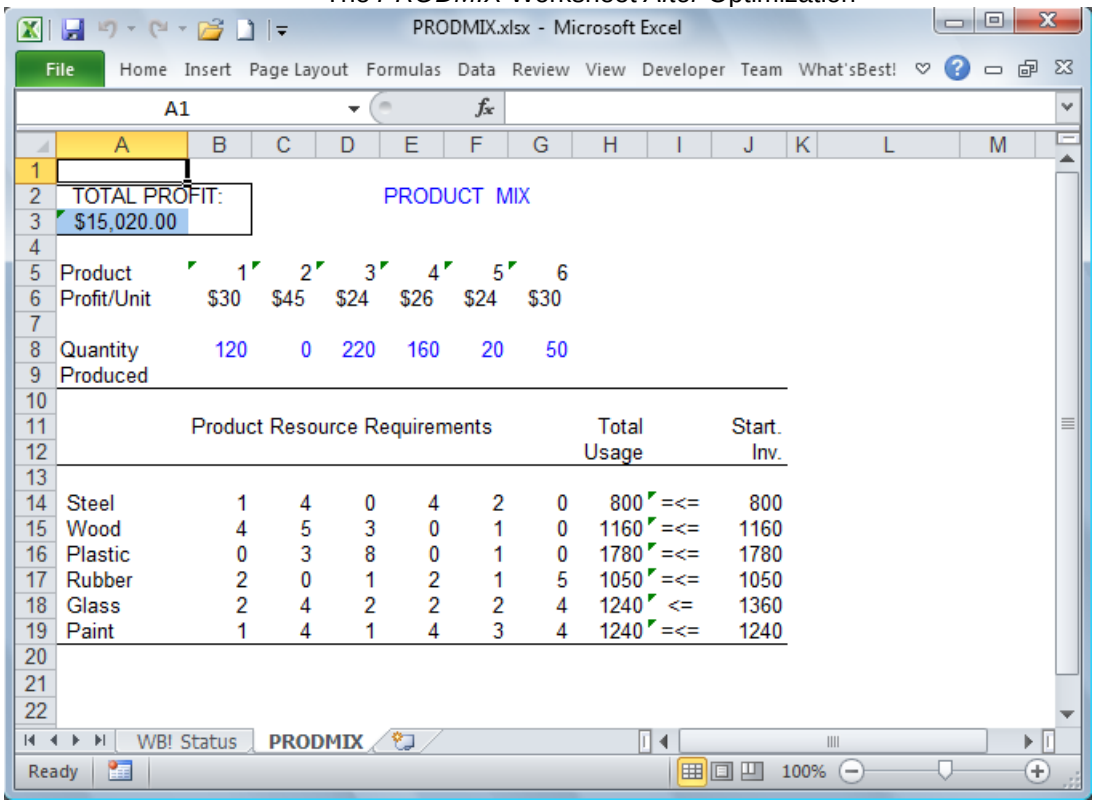
"What if?" vs. What's*Best!*

Spend a few minutes working with the model to try to maximize *Total Profit* (A3) by inserting and adjusting the *Quantities Produced* (B8:G8) for each product. As you go through these "What If?" projections, be sure to use no more of any raw material than you currently have in your *Starting Inventory*.

As you near exhaustion of any raw material, look at the *Product Resource Requirements* table (B14:G19) for products that require relatively little of that material. Also, remember to use *Profit/Unit* (B6:G6) as a guide to the effect of incremental production of each product. Judge each of your hypothetical solutions by the *Total Profit* (A3) generated. Once you arrive at a reasonable solution, jot down its total profit.

Now that you've experimented on your own, you're ready to solve the model.

The *PRODMIX* Worksheet *After* Optimization



The best possible solution to this problem is a total profit of $15,020. Note how effectively What's*Best!* used the available raw materials — and how quickly this optimal solution was calculated. Note also how many of each product What's*Best!* tells you to produce and that the solution manufactures none of the product with the highest profit contribution! This nonintuitive result, which you'd have difficulty finding through "what-if", is a good indication of the power of What's*Best!*.

The Building Block Method

File name: BLOCK.XLSX *TYPE: LINEAR OPTIMIZATION*

Application Profile

This example uses the full power of your spreadsheet software to develop larger applications using smaller ones as building blocks. In this way, comprehensive models can be created that can simultaneously cover many different aspects of a business situation.

The Problem in Words

You run three plants. Each plant is capable of producing six products from six raw materials. You also operate two steel mills that can ship steel as a raw material to any of your plants. You want to know the amount of steel each mill should ship to each plant and the number of products each plant should produce to maximize your total profit.

Background

The *PRODMIX* (shown earlier) and *SHIPPING* (shown later) worksheets are combined in the *BLOCK* problem, which involves maximizing the profit from several manufacturing plants while minimizing the cost of shipping raw materials from multiple sites. The cost of shipping one unit of steel from each of the mills to each of the plants is shown in the table below:

Shipping Costs Per Unit of Steel

From:	Steel	Steel
To:	**Mill 1**	**Mill 2**
Plant A	$2	$5
Plant B	$3	$4
Plant C	$5	$6

Mill 1 has a maximum total capacity of 1200 units and *Mill 2* has a maximum total capacity of 1800 units. Each plant has an 800 unit minimum requirement of steel for the period that has been specified by each plant manager. The actual starting inventory of steel at each plant is determined by the total number of units shipped to that plant from the two steel mills. The starting inventories of the other five raw materials at each plant are listed below:

Resource	Units on Hand
Wood	1160
Plastic	1780
Rubber	1050
Glass	1360
Paint	1240

The profit contribution and product resource requirements for each of the six products are the same as in the *Product Mix* problem.

Objective of Optimization

The objective of optimization is to maximize total profits. The total profit is the sum of individual plant profits, minus the sum of the shipping costs from the individual steel mills.

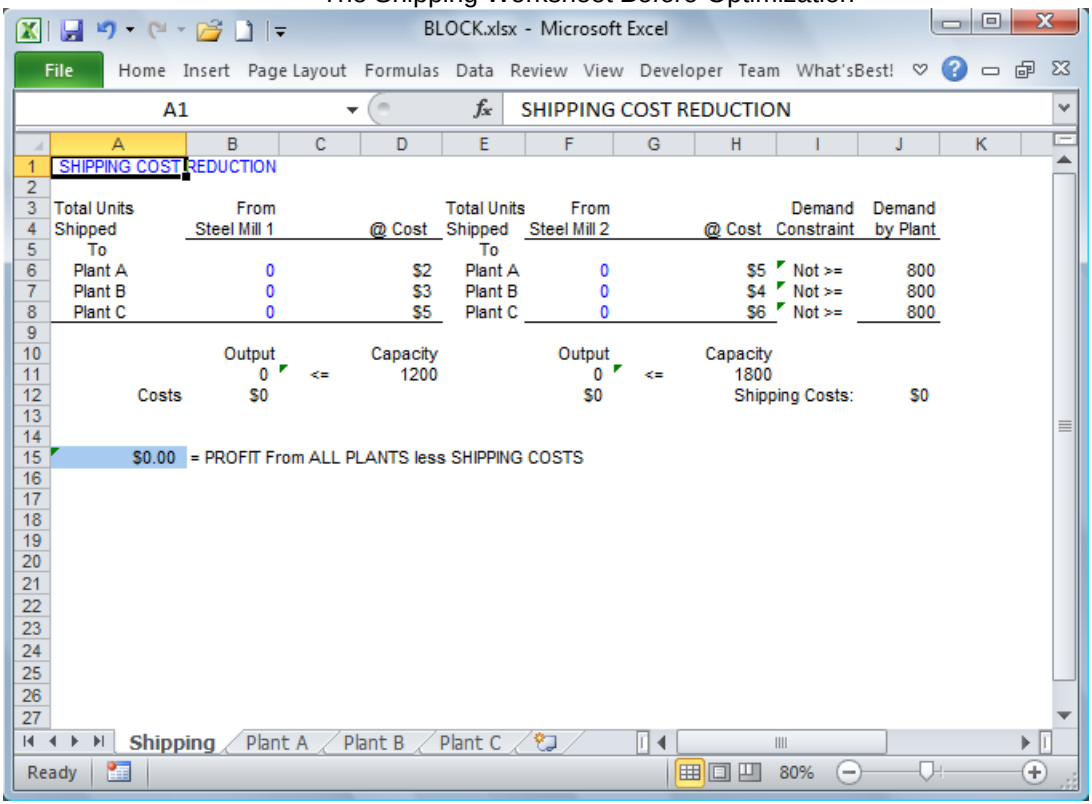
The Worksheet

Let's look at the *BLOCK* sample file.

The *BLOCK* worksheet was built by directly combining the *SHIPPING* and *PRODMIX* worksheets. After combining one copy of the *Shipping Cost Reduction* model with three copies of the *Product Mix* model, we performed only slight modifications to link the four individual problems into one comprehensive model. The model is laid out in a 3D workbook on four separate worksheets—a shipping sheet and three separate manufacturing sheets.

The first worksheet is *Shipping*, built by copying the original *Shipping Cost Reduction* problem into cell A1 of a new workbook. This sheet gives information on the amount of steel that has been shipped from each steel mill to each manufacturing plant, a breakdown of shipping costs, and total costs.

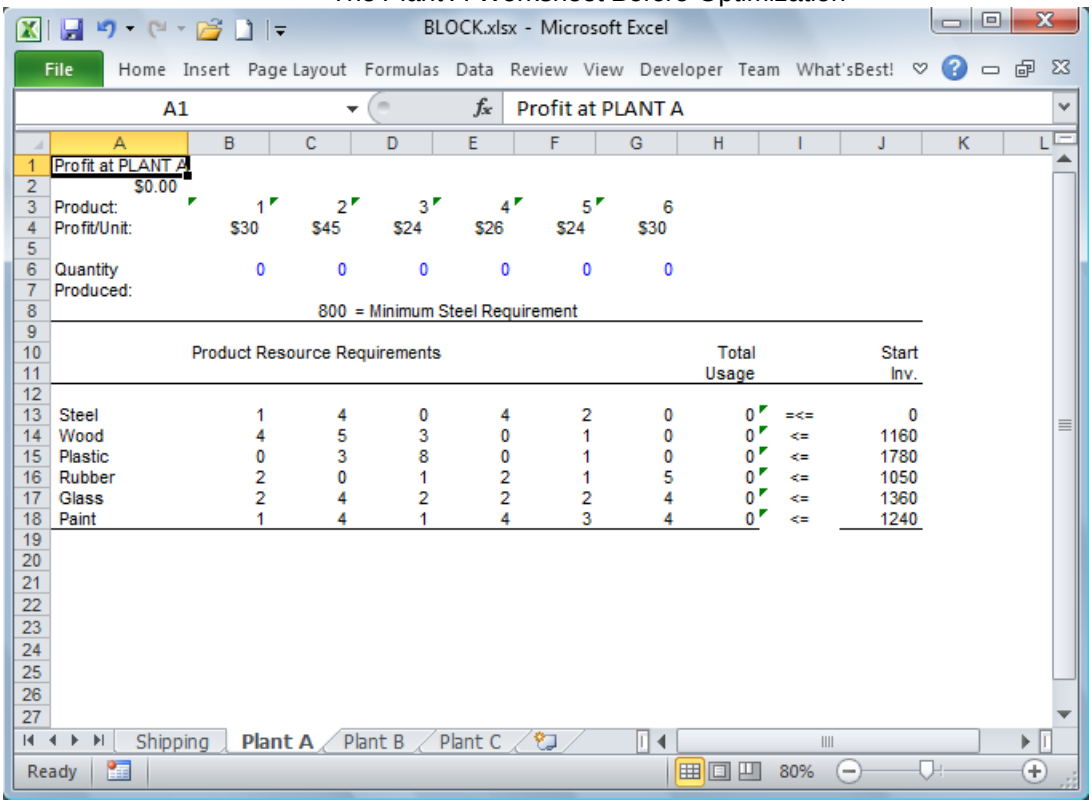
The Shipping Worksheet *Before* Optimization



The remaining three worksheets—*Plant A*, *Plant B*, and *Plant C*—are the manufacturing plant screens. They were built by placing the *PRODMIX* model into each of the three tabbed worksheets. Each sheet represents one of the three manufacturing plants.

With the exception of steel, the starting inventories of all the raw materials are the same as in the original problem. The starting inventory of steel for each plant now becomes the amounts shipped from the steel mills to that plant. For example, the formula in cell J13 of *Plant A* worksheet, *Starting Steel Inventory* for steel at *Plant A*, is =Shipping! B6+Shipping! F6. Each plant's sheet tells the quantity of each item produced, inventory total usage amounts, and the profit contribution from that plant.

The *Plant A* Worksheet *Before* Optimization



	A	B	C	D	E	F	G	H	I	J
1	Profit at PLANT A									
2	$0.00									
3	Product:	1	2	3	4	5	6			
4	Profit/Unit:	$30	$45	$24	$26	$24	$30			
5										
6	Quantity	0	0	0	0	0	0			
7	Produced:									
8		800 = Minimum Steel Requirement								
9										
10		Product Resource Requirements						Total		Start
11								Usage		Inv.
12										
13	Steel	1	4	0	4	2	0	0	=<=	0
14	Wood	4	5	3	0	1	0	0	<=	1160
15	Plastic	0	3	8	0	1	0	0	<=	1780
16	Rubber	2	0	1	2	1	5	0	<=	1050
17	Glass	2	4	2	2	2	4	0	<=	1360
18	Paint	1	4	1	4	3	4	0	<=	1240

With the exception of the best cell, the ABC's in the *BLOCK* model are the same as in its *PRODMIX* and *SHIPPING* components.

A. Determine Adjustable Cells

The adjustable cells (B6:B8, F6:F8 of the *Shipping* worksheet) are the quantities shipped from each mill to each plant, and the quantities to produce of each product from each plant (B6:G6 of the *Plant A, B, & C* worksheets).

B. Define Best

The best solution is the shipping and production decision that maximizes profit at all three plants less all shipping costs. This formula is in cell A15 of the *Shipping* worksheet.

C. Specify Constraints

The constraints are to ship at least the minimum amount of steel required by each plant (I6:I8 of the *Shipping* worksheet), to ship no more steel than the capacity of each mill (C11, G11 of the *Shipping* worksheet), and to use no more raw material at each plant than has been shipped or is being held in stock (I13:I18 of the *Plant A*, *B* & *C* worksheets).

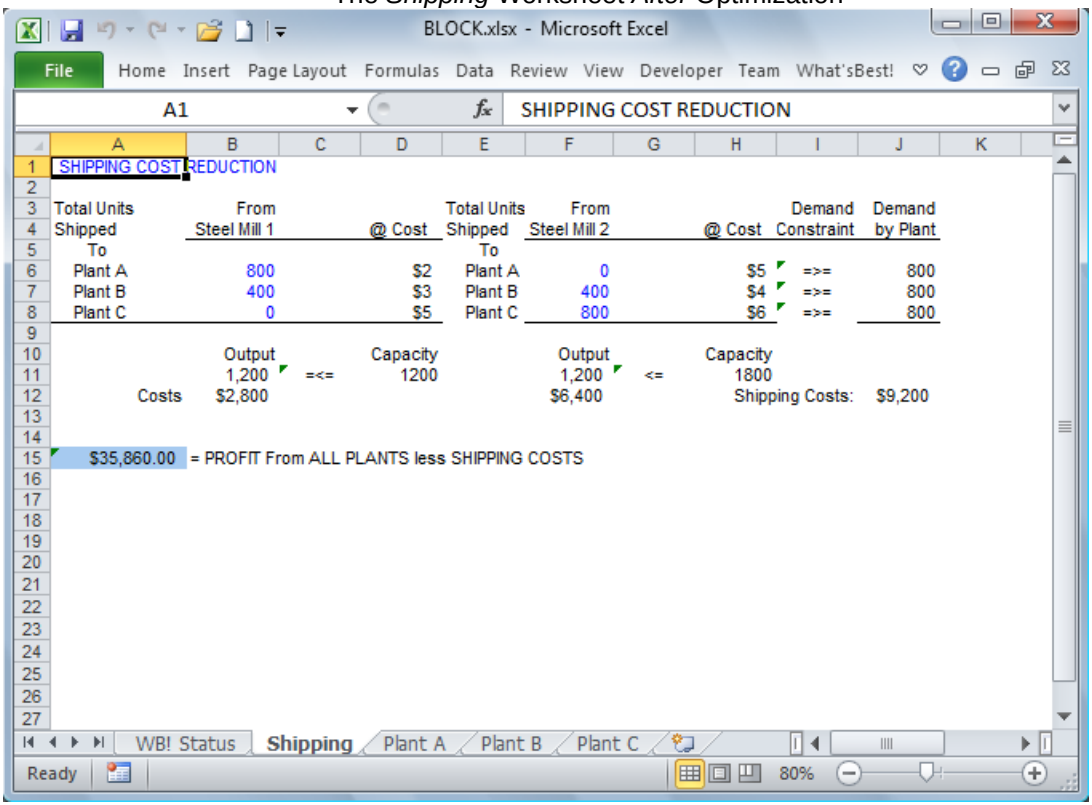
"What If?" vs. What's*Best!*

If you spend a few minutes working with the spreadsheet, you'll quickly understand how complex this problem really is. You must make six decisions on the quantities of steel to ship from each mill to each plant and eighteen decisions on the amount of each product you will produce at each plant (six products times three plants). As your spreadsheet recalculates your "What If?" projections, be sure to keep in mind the following guidelines:

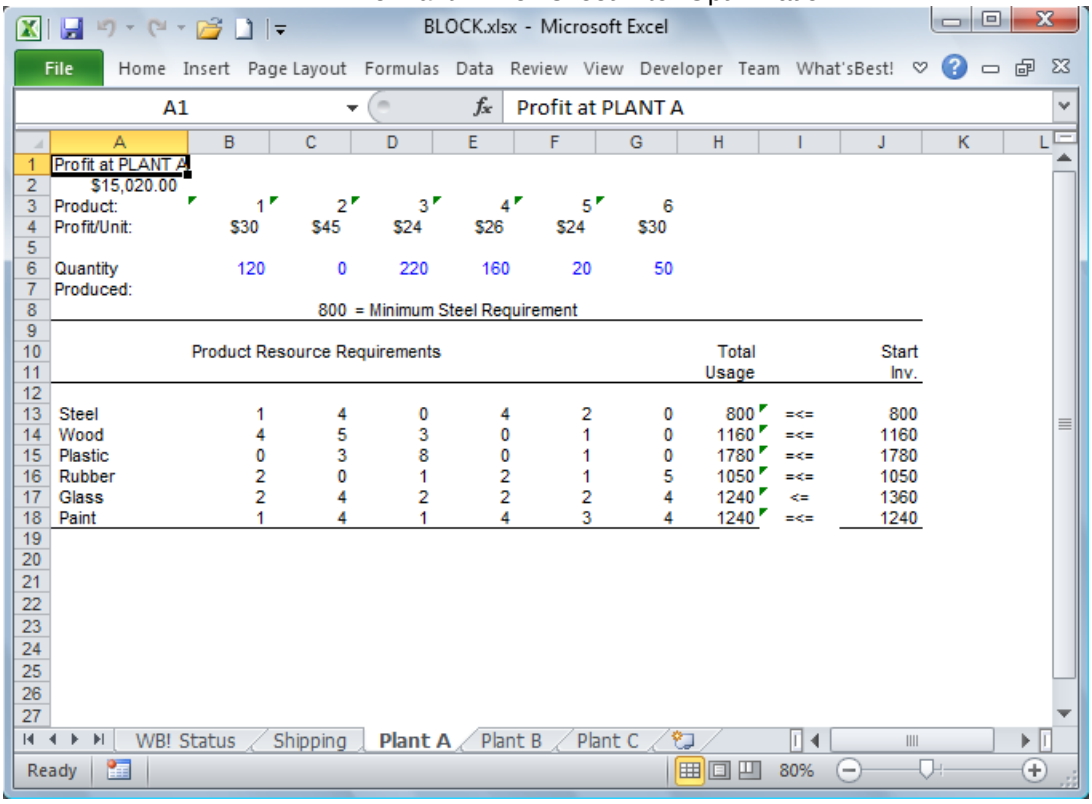
- Each plant must receive its minimum steel requirement.
- Each steel mill cannot ship more than its capacity.
- The use of any raw material cannot exceed the starting inventory.

After you've experimented on your own, you're ready to solve.

The *Shipping* Worksheet *After* Optimization



The *Plant A* Worksheet *After* Optimization



The best possible answer to this problem is a total profit of $35,860. Note how What's*Best!* successfully met the minimum steel needs of each plant while efficiently using the other raw materials on hand.

Waste Minimization in Stock Cutting

File name: CUTSTOCK.XLSX *TYPE: LINEAR OPTIMIZATION*

Application Profile

Consider a product produced in rolls or sheets — fabric, sheet metal, or carpeting. A manufacturer of this product is constantly faced with a problem. His machinery produces rolls of a single standard width, but he must sell to an unstandardized marketplace. This model applies linear programming to the problem of cut patterns. Given the sizes required, the amount of each size demanded, and the cut patterns available, the model minimizes the waste cost associated with meeting demand.

The Problem in Words

You're the production manager in a sheet metal plant. You turn out rolls of sheet metal of a particular width. Your customers use your product in different ways requiring rolls of different widths. To accommodate their needs, you can cut a roll of sheet metal into more than one narrower roll. Your goal is to waste as little of the sheet metal as possible by choosing patterns that best utilize the width of a roll and closely matching the length requirements of various sizes. Each wasted edge strip and each length of metal cut to an unneeded width adds to your total waste cost. You must select cut patterns that meet customer needs and minimize total waste costs.

Background

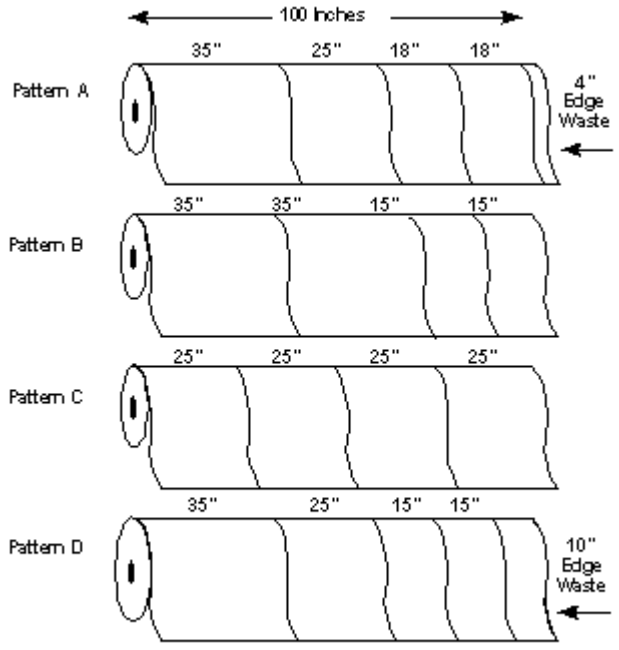
Your factory produces sheet metal in various widths that you cut from 100 inch widths. In order to meet customer demand, you must produce rolls of these widths and footage:

Width	Total Footage
35"	1,020
25"	3,000
18"	967
15"	1,450

For each roll of 100" metal, you select one of four cut patterns:

Pattern	A	B	C	D
Rolls	35"	35"	25"	35"
cut into	25"	35"	25"	25"
widths	18"	15"	25"	15"
of:	18"	15"	25"	15"

Schematically, the patterns look like this:



Each length of 100" metal must be cut into one of these patterns, but how many feet should be cut using each pattern?

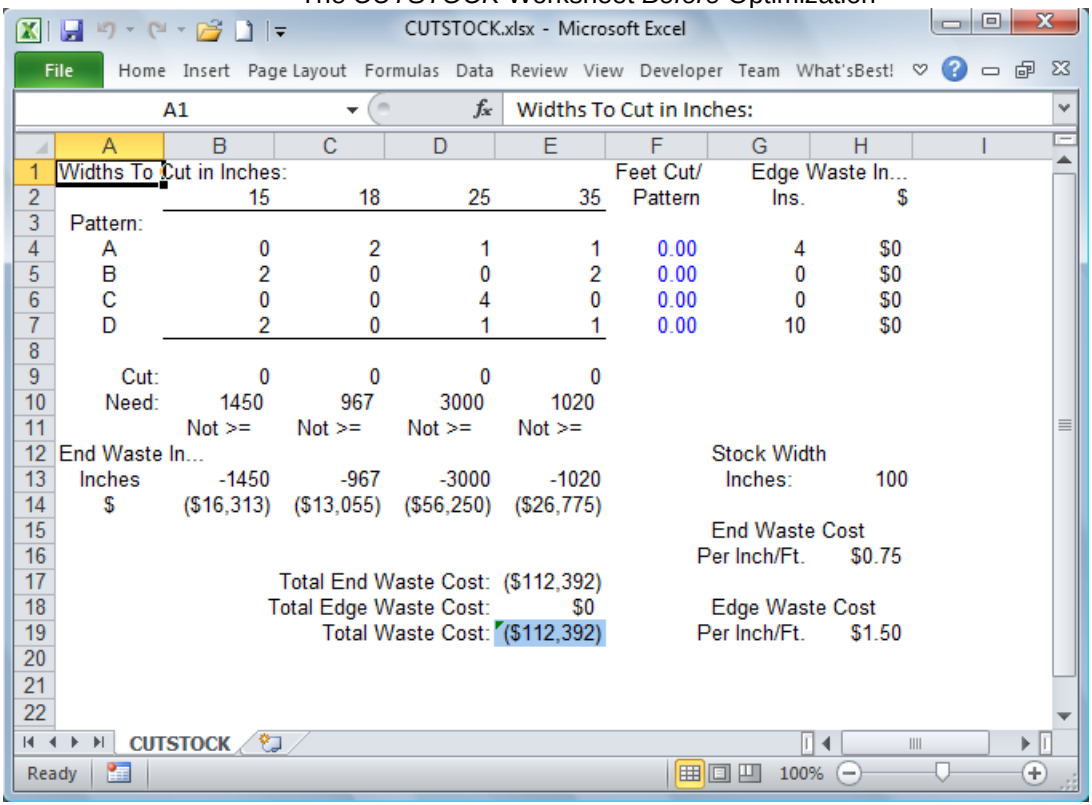
Objective of Optimization

Your objective is to minimize the total waste cost — the sum of edge waste and end waste. How many feet of each pattern should you cut in order to meet customer demand at the lowest cost?

The Worksheet

Let's examine the *CUTSTOCK* sample file and see how the edge waste and end waste are calculated.

The *CUTSTOCK* Worksheet *Before* Optimization



	A	B	C	D	E	F	G	H
1	Widths To Cut in Inches:					Feet Cut/	Edge Waste In...	
2		15	18	25	35	Pattern	Ins.	$
3	Pattern:							
4	A	0	2	1	1	0.00	4	$0
5	B	2	0	0	2	0.00	0	$0
6	C	0	0	4	0	0.00	0	$0
7	D	2	0	1	1	0.00	10	$0
8								
9	Cut:	0	0	0	0			
10	Need:	1450	967	3000	1020			
11		Not >=	Not >=	Not >=	Not >=			
12	End Waste In...						Stock Width	
13	Inches	-1450	-967	-3000	-1020		Inches:	100
14	$	($16,313)	($13,055)	($56,250)	($26,775)			
15							End Waste Cost	
16							Per Inch/Ft.	$0.75
17				Total End Waste Cost:	($112,392)			
18				Total Edge Waste Cost:	$0		Edge Waste Cost	
19				Total Waste Cost:	($112,392)		Per Inch/Ft.	$1.50
20								
21								
22								

In B4:B7, the number of cuts of the 15 inch width is entered for each pattern (e.g., pattern *A* has no 15 inch cuts, and pattern *B* has two, as shown in the schematic on the previous page). The total *Cut* cells (B9:E9) display the footage cut to each width. In G4:G7, *Edge Waste* in inches is shown. *Edge waste* is the total of inches cut per pattern subtracted from the stock width of 100 inches. *End Waste* in inches (B13:E13) is the amount cut less the amount needed.

A. Determine Adjustable Cells

The adjustable cells in this model are the *Feet Cut/Pattern* (F4:F7). They contain the number of feet to be cut into each of the four available patterns.

B. Define Best

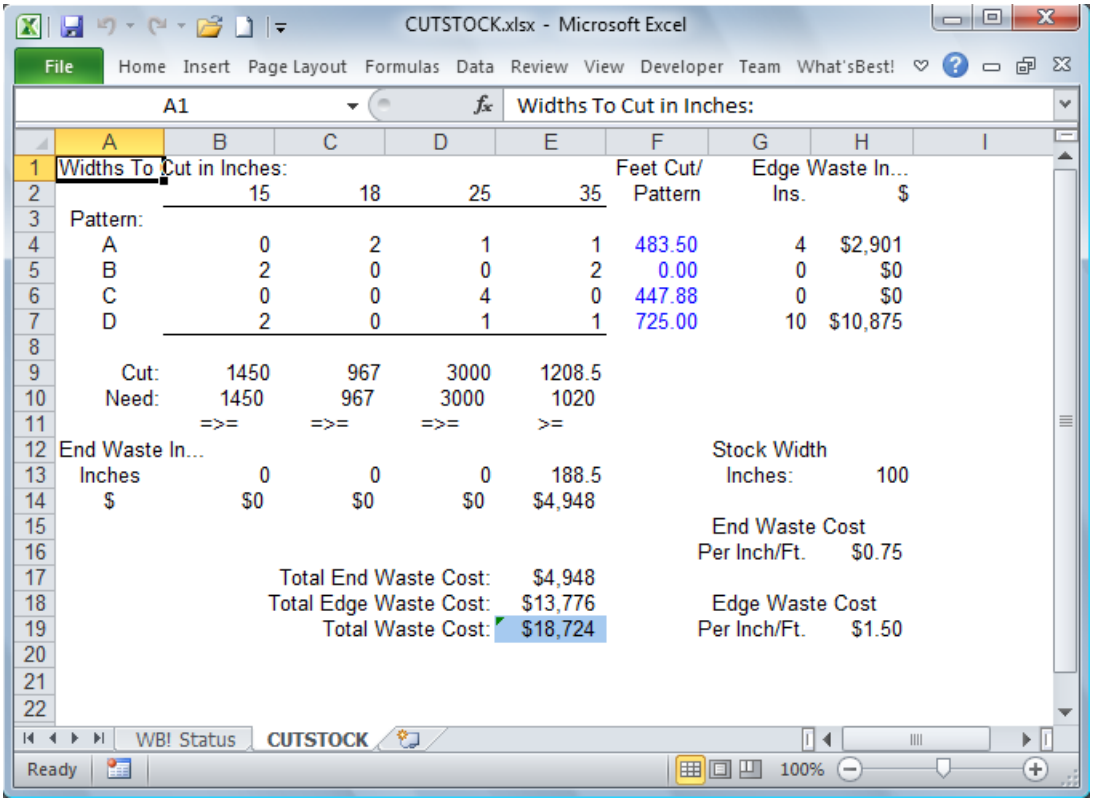
The best solution to this worksheet is the combination of cut patterns that minimizes total *Waste Costs* while meeting the required footage for each width. The value of *Total Waste Costs* is located in cell E19, and is the sum of the edge waste costs resulting from each cut pattern, and the end waste costs of excess footage of any width produced.

C. Specify Constraints

The constraints in this problem are simply that the production of sheet metal in each of the four widths must meet the amount demanded. The requirements for 15, 18, 25, and 35 inch sheet metal appear in B10:E10. The actual footage of each width resulting from the selected *Cut* patterns is tallied using formulas in B9:E9. B11:E11 contain constraints forcing these requirements to be met. The formula for 15" *End Waste* in B11, for example, is *WB*(B9,">=",B10). This ensures that actual footage (B9) is at least equal-to the required amount (B10).

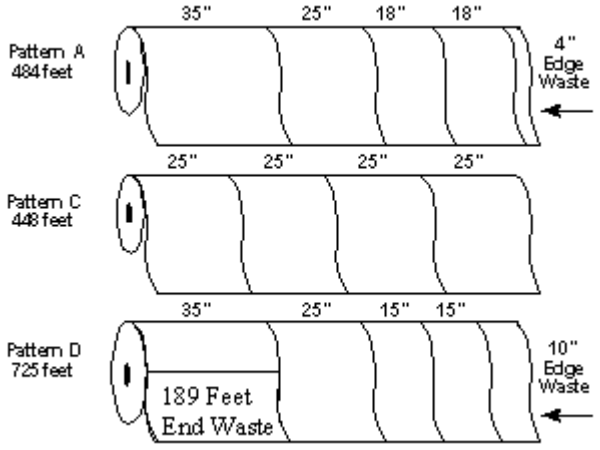
Now, you're ready to solve the model.

The *CUTSTOCK* Worksheet *After* Optimization



The solution to the *CUTSTOCK* model is a *Total Waste Cost* of $18,724 ($13,776 *Edge Waste* (E18) and $4,948 *End Waste* (E17)). Rounding may give a slightly different answer. Here's a schematic of the solution.

Note: If multiple raw material types were available, this model would be reformulated to minimize total material costs. Otherwise, very expensive material might be purchased, although it would be used efficiently. Another model might also take into account the value of *End Waste* that could be re-entered in inventory.



Note that the optimized cutting plan includes no footage cut to *Pattern B*, in cell G5. Also, end waste occurs only in the 35" width (E13).

Plant Location

File name: PLANTLOC.XLSX *TYPE: LINEAR OPTIMIZATION*

This problem is related to the class of network or routing problems. In such problems, the variables usually include multiple choices of transportation routes to and from a number of points of origin and destination, with differing shipping and operating costs associated with each possible route.

The *Plant Location* problem is similar to the *SHIPPING* sample model, but allows greater latitude of decision-making in that the points of origin (plant locations) are variable. Manufacturers and wholesale businesses are likely to encounter problems of this sort in matching existing customer demand to product availability and minimal transportation costs.

Background

Your firm has a choice of five locations in which to operate a manufacturing facility. Six markets exist with a demand for your product. Each potential plant has an associated monthly operating cost, and shipping routes to the demand cities have varying costs. In addition, each potential plant will have a production capacity that must not be exceeded.

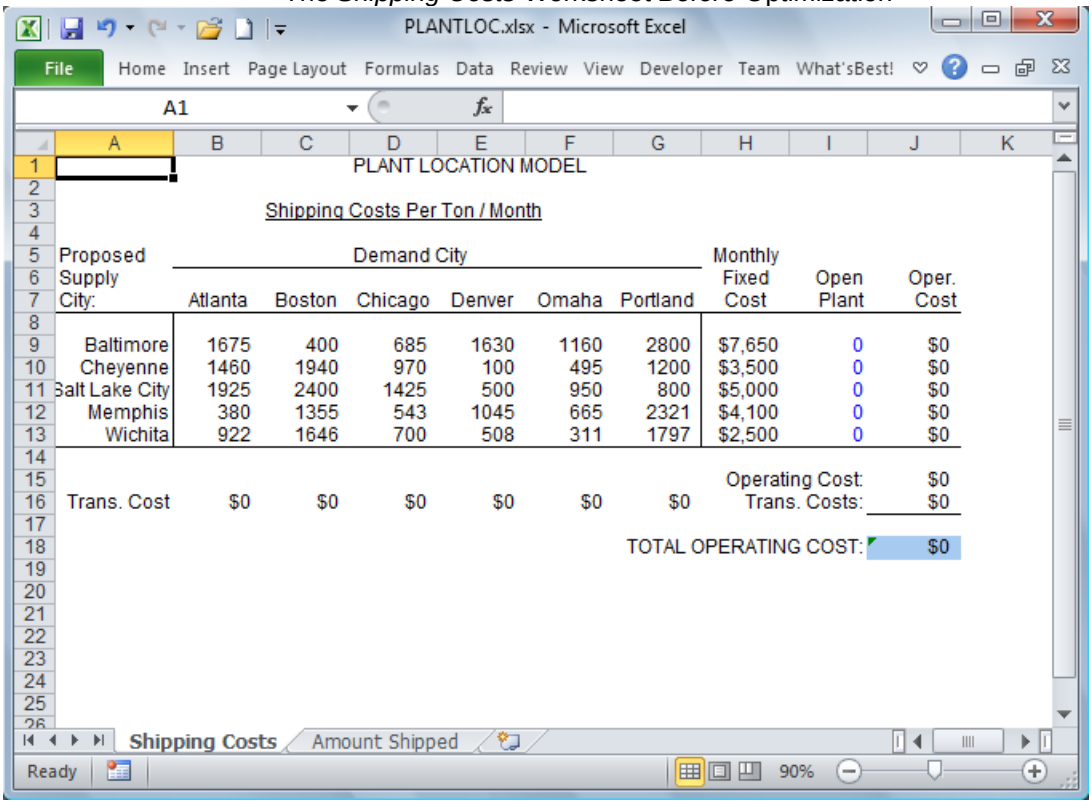
Objective of Optimization

The objective is to locate plants in such a way that demand is satisfied in each target city, potential plant capacity is not exceeded, and overall operating costs are minimized.

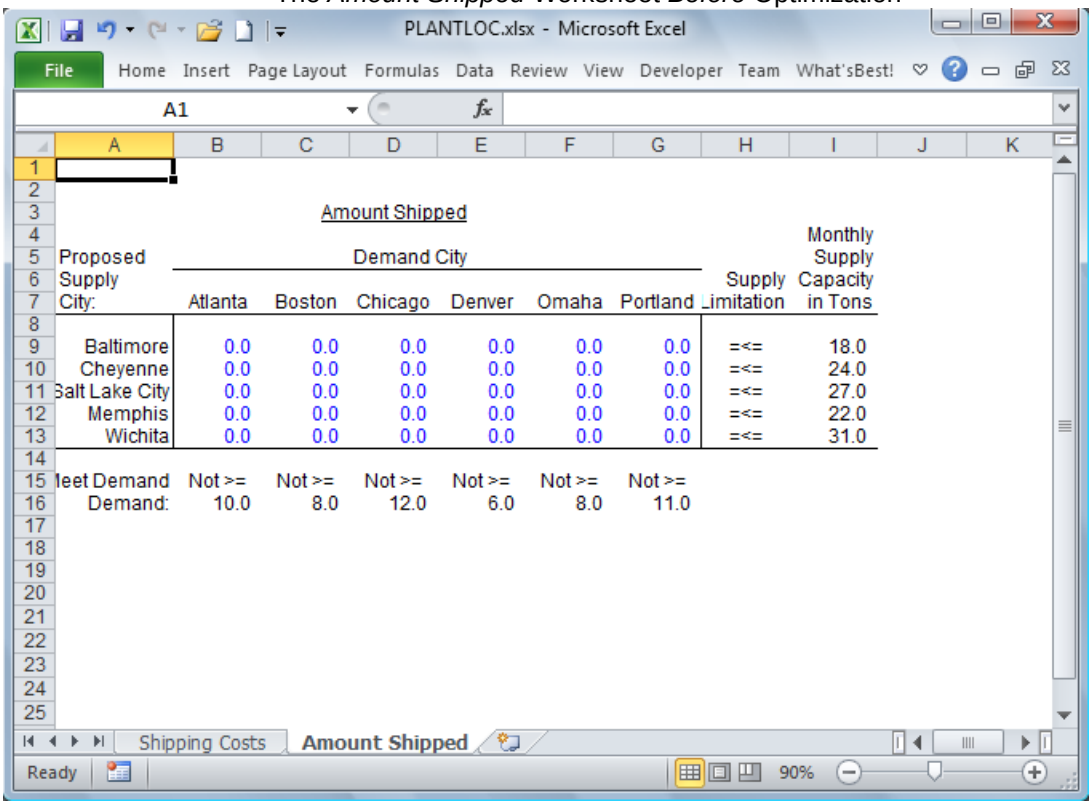
The Worksheet

Let's open the *PLANTLOC* sample and examine its layout and formulas.

The *Shipping Costs* Worksheet *Before* Optimization



The *Amount Shipped* Worksheet *Before* Optimization



A. Determine Adjustable Cells

The adjustable cells in *PLANTLOC* are the quantities shipped from each potential plant to each demand city (B9:G13 on the *Amount Shipped* worksheet) and the number of plants open at each location (I9:I13 of the *Shipping Costs* worksheet).

In addition, since either a whole plant or none at all must be erected at any location, I9:I13 of the *Shipping Costs* worksheet are specified as binary integers. What's*Best!* will return only a zero or one as values in these cells.

B. Define Best

The best solution consists of minimizing *Total Operating Cost* in cell J18 of the *Shipping Costs* worksheet. This cell contains the sum of the two subtotals for *Operating Costs* and *Transportation* costs in cells J15 and J16 of the *Shipping Costs* worksheet, respectively. The *Transportation* cost subtotal, in turn, is the sum of the values in cells B16:G16 of the *Shipping Costs* worksheet. The formulas in these cells represent the total shipping costs to the city in question from all plant locations. For instance, the formula in cell B16 is *SUMPRODUCT*(B9:B13, 'Amount Shipped'! B9:B13).

C. Specify Constraints

The two blocks of constraints in the *Plant Locating* problem ensure that demand is satisfied in the six cities and that production capacity at each of the five potential plants is not exceeded.

First, since demand in B16:G16 of the *Amount Shipped* worksheet must be met, the constraints in B15:G15 of the same worksheet require the sum of amount shipped to each demand city (*Atlanta*, *Boston*, etc.) to be greater-than-or-equal-to the demand for that city.

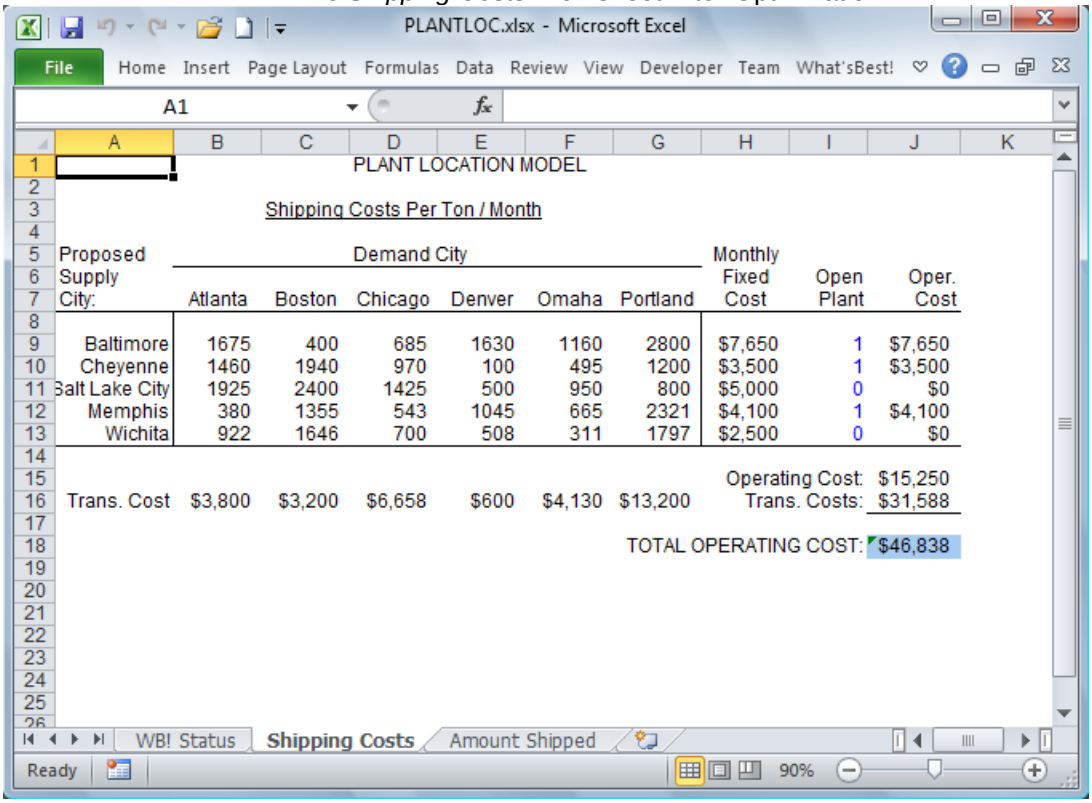
The second constraint is that the amount shipped from each *Supply City* (*Baltimore*, *Cheyenne*, etc.) must not exceed its supply capacity. To enforce this constraint, the constraints in H9:H13 of the *Amount Shipped* worksheet require that the sum of materials shipped from each *Supply City* is less-than-or-equal-to the *Monthly Supply Capacity* for that city (I9:I13 of the *Amount Shipped* worksheet). Let's look at the formula in H9 of the *Amount Shipped* worksheet:

WB(*SUM*(B9:G9),"<=",I9*'Shipping Costs'! I9)

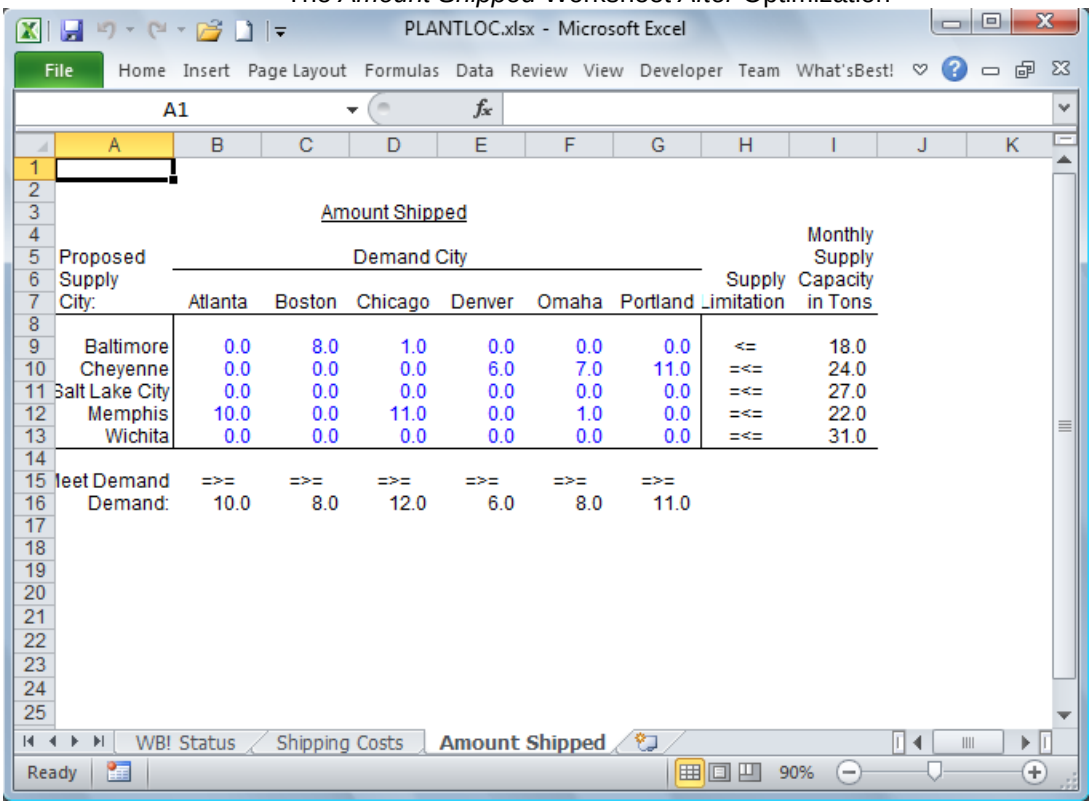
The total amount shipped from Baltimore, *SUM*(B9:G9), can be no more than I9*Shipping Costs I9, Baltimore's capacity. Since *Shipping Costs* I9 is either 1 or 0, depending on whether or not a plant is open in Baltimore, its use in the formula returns a nonzero value only if the plant is opened.

Now, let's solve the model.

The *Shipping Costs* Worksheet *After* Optimization



The *Amount Shipped* Worksheet *After* Optimization



PLANTLOC.xlsx - Microsoft Excel

File Home Insert Page Layout Formulas Data Review View Developer Team What'sBest!

A1

Amount Shipped

Proposed Supply City:	Demand City: Atlanta	Boston	Chicago	Denver	Omaha	Portland	Supply Limitation	Monthly Supply Capacity in Tons
Baltimore	0.0	8.0	1.0	0.0	0.0	0.0	<=	18.0
Cheyenne	0.0	0.0	0.0	6.0	7.0	11.0	=<=	24.0
Salt Lake City	0.0	0.0	0.0	0.0	0.0	0.0	=<=	27.0
Memphis	10.0	0.0	11.0	0.0	1.0	0.0	=<=	22.0
Wichita	0.0	0.0	0.0	0.0	0.0	0.0	=<=	31.0
Meet Demand	=>=	=>=	=>=	=>=	=>=	=>=		
Demand:	10.0	8.0	12.0	6.0	8.0	11.0		

WB! Status | Shipping Costs | Amount Shipped

Ready 90%

Plants are open in *Baltimore*, *Cheyenne*, and *Memphis* (I9, I10, and I12 of the *Shipping Costs* worksheet). *Total Operating Cost* in J18 of the *Shipping Costs* worksheet has been minimized to the optimally low figure of $46,838. Since the constraint in H9 of the *Amount Shipped* worksheet is not binding, you know that additional sales out of *Baltimore* are possible.

Staff Scheduling

File name: STAFF.XLSX *TYPE: LINEAR OPTIMIZATION*

Application Profile

In *Staff Scheduling* problems, the goal is to meet specified manpower requirements at minimum cost. In general, the schedule must meet certain conditions, such as those imposed by regulations or union contracts, including minimum shift length, number of work breaks, or maximum overtime hours.

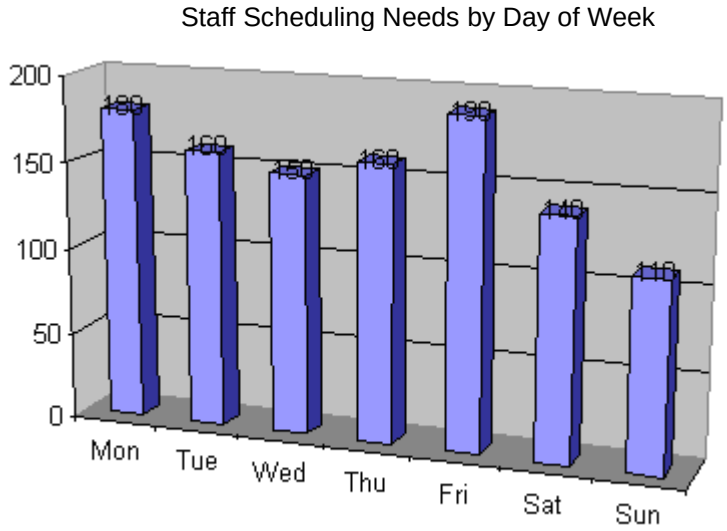
These models have applications in the scheduling of airline flight crews, hospital and office staffs, and restaurant personnel, to name just a few.

The Problem in Words

You are running a business with daily staff needs on varying work loads each day of the week.

Background

The daily staff needs range from 110 to 190 people as shown below:



In addition, there's a labor requirement. Employees must work a five consecutive day workweek, followed by two days off. Thus, the allowable shifts are Monday through Friday, Tuesday through Saturday, Wednesday through Sunday, etc. Each employee earns $200 per week.

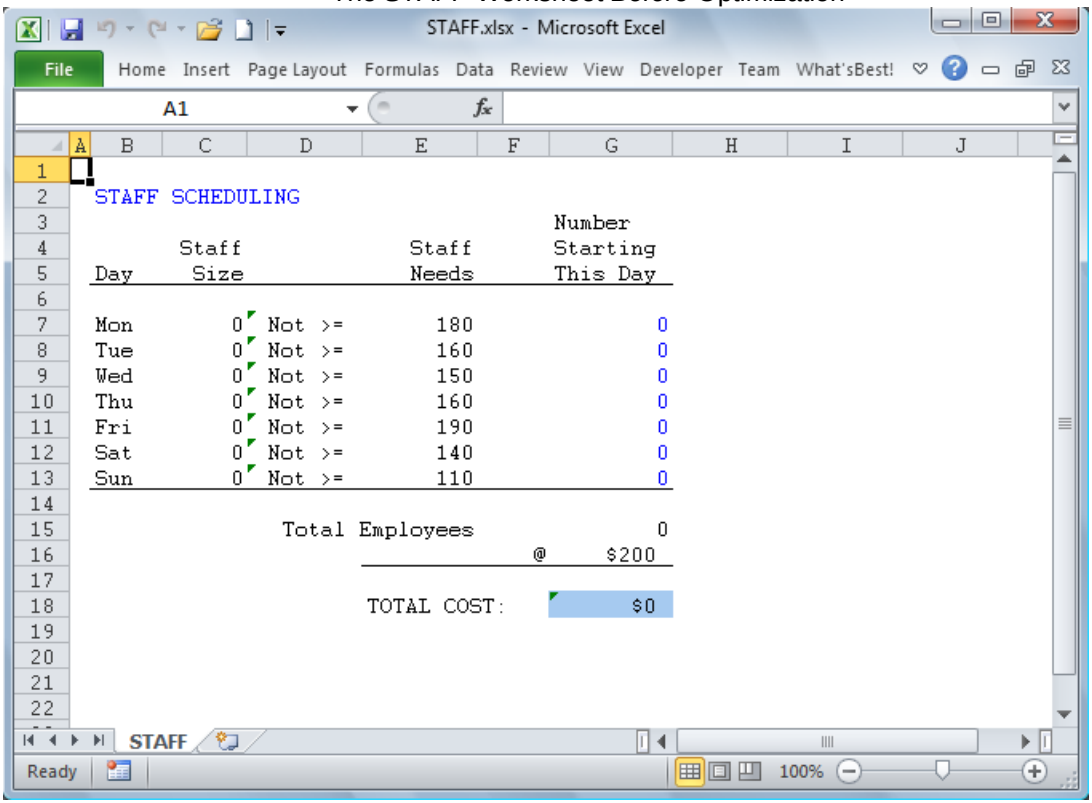
Objective of Optimization

The objective is to cover staff needs with five-day shifts at minimum weekly payroll cost.

The Worksheet

Let's look at the *STAFF* worksheet included in the sample files. Look it over for a minute and examine its layout and formulas to see how it has been designed.

The *STAFF* Worksheet *Before* Optimization



A. Determine Adjustable Cells

The adjustable cells in this model contain the number of people whose five-day workweeks begin on each day of the week. These are shown in the *Number Starting This Day* column (G7:G13).

B. Define Best

The best solution is the one that minimizes total payroll costs. In this problem, cost is equal-to the number of employees times their weekly salary. This is shown in cell G18. The formula for *Total Employees* (G15) is the sum of employees starting each day. The weekly salary per employee has been entered in cell G16. This means that the formula for *Total Cost* in cell G18 is G15*G16.

C. Specify Constraints

The requirement of any solution to this problem is that *Staff Size* is constrained to be greater-than-or-equal-to *Staff Needs*. Without such a constraint, the job for What's*Best!* would be easy. The least expensive solution would always be to hire nobody!

Take a look at the *Staff Size* column (C7:C13). The formula for *Staff Size* equals the *Number Starting This Day* plus the total of the number starting for the preceding four days. Remember, each employee has shifts on five consecutive days, so any given day's *Staff Size* equals the five-day total.

For example, the formula for the *Staff Size* on Monday (C7) reads:

G7+G10+G11+G12+G13

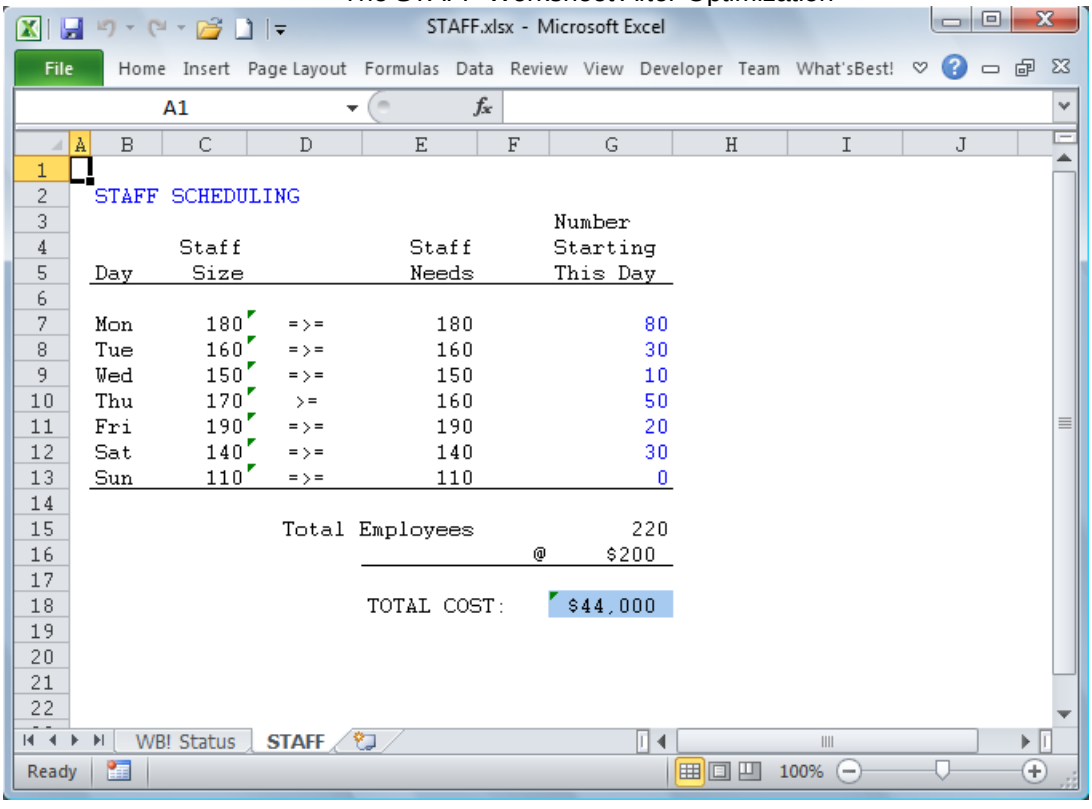
The value in C7 must always be greater-than-or-equal-to the *Staff Needs* for Monday. To ensure this result, a greater-than (">=") constraint was entered in cells D7:D13 with C7:C13 on the left-hand side and E7:E13 on the right-hand side of the equation.

"What If?" vs. What's*Best!*

Try a "What If?" solution to this problem by entering initial estimates of the numbers of employees who will start their five-day workweek on each day (G7:G13). As you meet *Staff Needs* for some days, you begin to violate the constraints for others. Adjust your entries for *Number Starting This Day* to satisfy all the constraints, but be sure to meet all staff requirements. After each estimate, judge your solutions by minimizing *Total Cost* (G18). When you find a solution you might be satisfied with, note your final figure for *Total Cost*.

Now that you have arrived at a "What If?" spreadsheet solution, you're ready to solve the model.

The *STAFF* Worksheet *After* Optimization



The best possible solution to this problem is a *Total Cost* of $44,000. Note that the constraints in D7:D13 are all tightly satisfied. That is, there is no overstaffing at all.

D. Dual Values

The best solution to this problem recommends starting some people on each day of the week except Sunday. If someone on your staff can only work Sunday through Thursday, you can analyze the model to find what the cost would be of starting this person on Sunday instead of some other day. Do this by using the *Advanced|Dual* command to display in cell H13 the dual value of cell G13, the *Number Starting* on Sunday. Then, solve the model again (Refer to the section *About Dual Values* in Chapter 3, *Additional Commands*, for more information on dual values).

After solving, a dual value of $67 (0.666) appears in cell H13. This indicates that the total cost would increase by $67 if one person were to start a workweek on Sunday. You can confirm this by placing a "1" in cell G13, and using the *Remove Adjustable* command to lock the value in place. Then, solve once again.

The total cost of this solution has increased to $44,067 — an increase of $67 from the previous best solution, exactly as predicted by the dual value in cell H13. However, you get fractional numbers of employees. To get integer answers, you could round the fractional cell values. One possibility is as follows:

	Round From:	**To:**
Cell G8, Number Starting on Monday	79.33333	80
Cell G10, Number Starting on Wednesday	19.33333	20
Cell G11, Number Starting on Thursday	39.33333	39
Cell G13, Number Starting on Saturday	29.33333	29

After entering these new integer values in the adjustable cells, you'll find a new solution with a total of 221 employees and $44,200 as the minimum cost. Therefore, dual values in this case provide us with the theoretical (since fractional people are imaginary) minimum increase in total cost ($67), instead of the actual increase in total cost ($200) for a feasible solution.

Note: Before using any dual values in a pricing, hiring, or purchasing decision, be sure to investigate the ranges over which they are valid. In some problems, dual values may be valid over only an infinitesimal range. See *Additional Commands*, for a discussion of valid ranges for dual values.

Staff Scheduling: Preferred Assignment

File name: ASSIGN.XLSX *TYPE: LINEAR OPTIMIZATION*

Application Profile

Staff scheduling is usually a very costly and laborious job. In institutions where there are several daily shifts and the staff can rotate between shifts during the same period, the task is even more complicated. The scheduler must match the organization's requirements with the preferences of individual employees. Larger numbers of employees and longer time periods make the task more difficult.

This problem involves scheduling several staff members in an organization that operates in multiple time shifts each day, seven days a week.

The Problem in Words

Your business works in three shifts — day, evening, and night. For each shift, you must meet certain minimum staffing levels. Each staff member must be assigned a specific number of shifts and has individual preferences for working the various shifts. Moreover, because of the nature of the work, each employee must have an adequate rest period between shifts in order to perform the job effectively.

This model demonstrates how you can meet your staffing requirements while satisfying your employees' scheduling preferences to the greatest possible degree.

Background

Your specific staffing needs for a particular week are as follows:

	Staff Members Required						
	Mon	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**	**Sun**
Day	2	1	1	3	1	1	1
Eve	1	1	0	0	2	1	1
Nite	1	0	0	0	1	1	1

Your staff members — *Nixon*, *Ford*, *Bush*, and *Reagan* — must each be assigned to five shifts for the week, and have individually selected their preferred shifts on a descending scale of 5 to 1. These rankings appear on the *Preferences* worksheet of the model.

In addition to considering your staff's preferences, you must abide by two rules in scheduling the shifts:

- Staff members cannot work more than one shift in a single day
- After working a shift, a staff member can't work the next two shifts

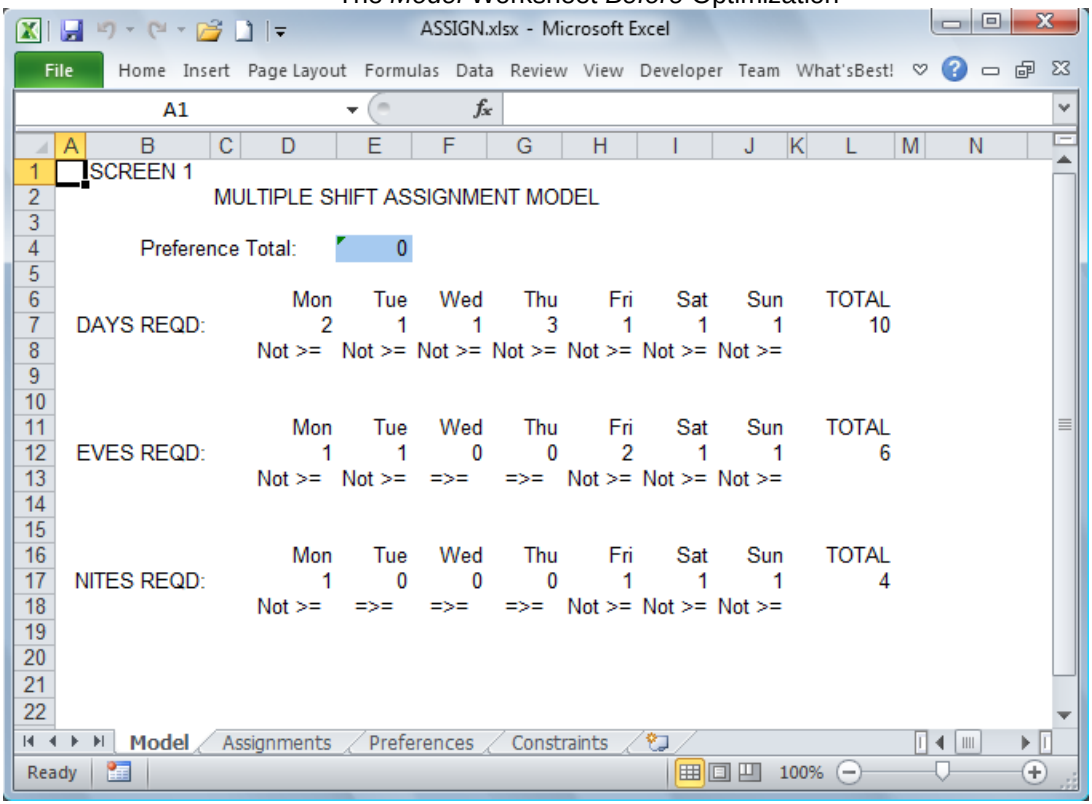
Objective of Optimization

The objective is to maximize overall staff preference by assigning employees to the shifts they deem most desirable.

The Worksheet

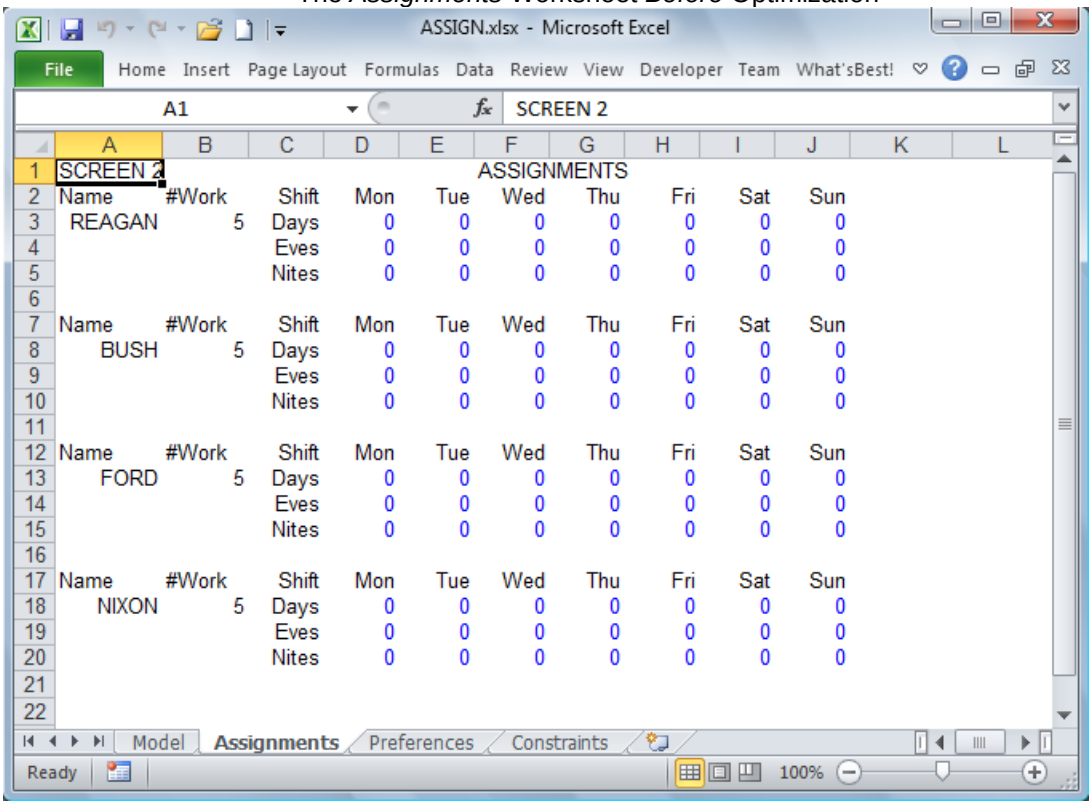
Let's open the *ASSIGN* sample file and look it over. The model consists of four worksheets in one workbook.

The *Model* Worksheet *Before* Optimization



The *Model* worksheet shows the number of people required for each shift for the week and constraints ensuring these needs are met. The best cell also appears in this screen.

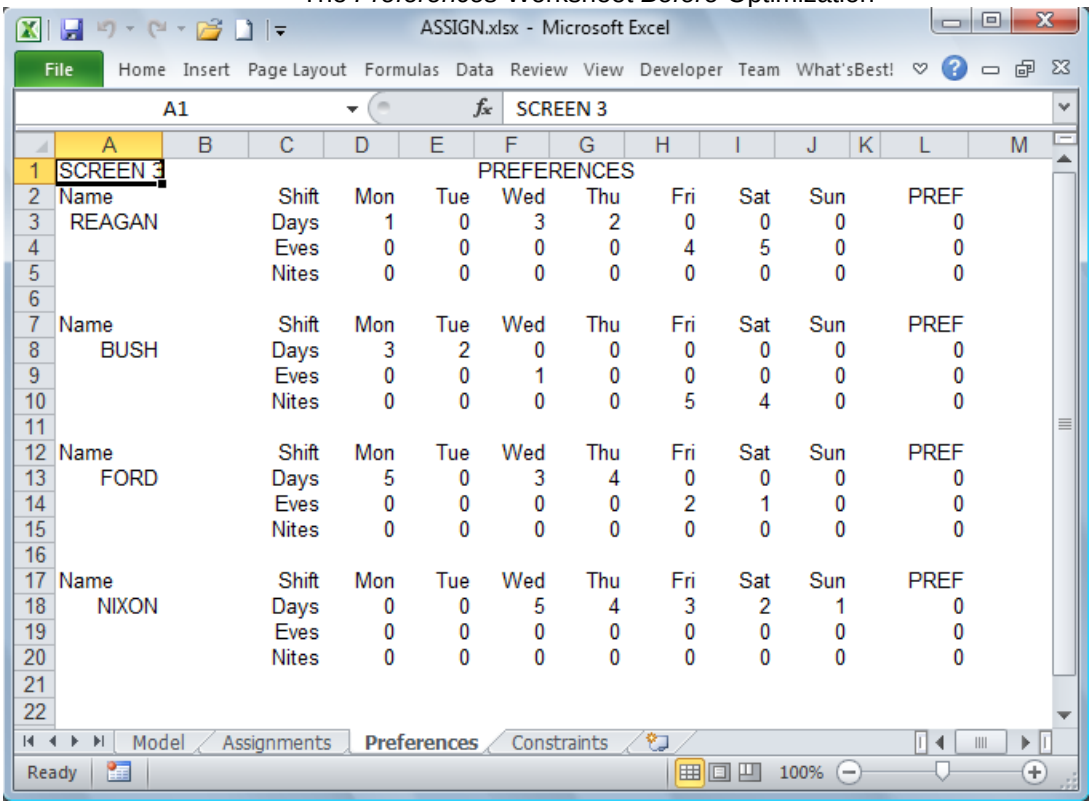
The *Assignments* Worksheet *Before* Optimization



	A	B	C	D	E	F	G	H	I	J	K	L
1	SCREEN 2					ASSIGNMENTS						
2	Name	#Work	Shift	Mon	Tue	Wed	Thu	Fri	Sat	Sun		
3	REAGAN	5	Days	0	0	0	0	0	0	0		
4			Eves	0	0	0	0	0	0	0		
5			Nites	0	0	0	0	0	0	0		
6												
7	Name	#Work	Shift	Mon	Tue	Wed	Thu	Fri	Sat	Sun		
8	BUSH	5	Days	0	0	0	0	0	0	0		
9			Eves	0	0	0	0	0	0	0		
10			Nites	0	0	0	0	0	0	0		
11												
12	Name	#Work	Shift	Mon	Tue	Wed	Thu	Fri	Sat	Sun		
13	FORD	5	Days	0	0	0	0	0	0	0		
14			Eves	0	0	0	0	0	0	0		
15			Nites	0	0	0	0	0	0	0		
16												
17	Name	#Work	Shift	Mon	Tue	Wed	Thu	Fri	Sat	Sun		
18	NIXON	5	Days	0	0	0	0	0	0	0		
19			Eves	0	0	0	0	0	0	0		
20			Nites	0	0	0	0	0	0	0		
21												
22												

The *Assignments* worksheet shows individual employees who are assigned to weekly shifts. A 1 indicates that a staff member has been assigned to a particular shift. A 0 means he isn't working that shift. The *#Work* cell by each name specifies the exact number of shifts each one should work during the week. Since no employee can work more than 1 shift during any 24 hour period, the maximum number of shifts any employee can work during the week is 7.

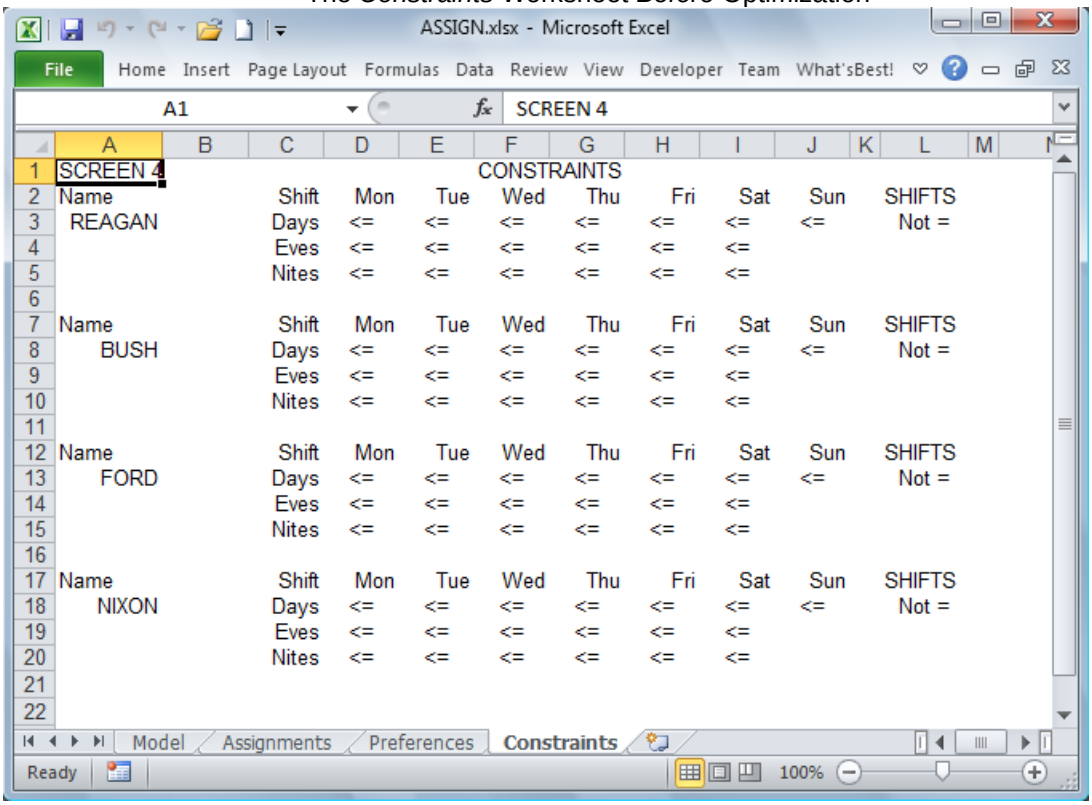
The *Preferences* Worksheet *Before* Optimization



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	SCREEN 3					PREFERENCES							
2	Name		Shift	Mon	Tue	Wed	Thu	Fri	Sat	Sun		PREF	
3	REAGAN		Days	1	0	3	2	0	0	0		0	
4			Eves	0	0	0	0	4	5	0		0	
5			Nites	0	0	0	0	0	0	0		0	
6													
7	Name		Shift	Mon	Tue	Wed	Thu	Fri	Sat	Sun		PREF	
8	BUSH		Days	3	2	0	0	0	0	0		0	
9			Eves	0	0	1	0	0	0	0		0	
10			Nites	0	0	0	0	5	4	0		0	
11													
12	Name		Shift	Mon	Tue	Wed	Thu	Fri	Sat	Sun		PREF	
13	FORD		Days	5	0	3	4	0	0	0		0	
14			Eves	0	0	0	0	2	1	0		0	
15			Nites	0	0	0	0	0	0	0		0	
16													
17	Name		Shift	Mon	Tue	Wed	Thu	Fri	Sat	Sun		PREF	
18	NIXON		Days	0	0	5	4	3	2	1		0	
19			Eves	0	0	0	0	0	0	0		0	
20			Nites	0	0	0	0	0	0	0		0	
21													
22													

Each employee ranks his five most preferred shifts for the week on the *Preferences* worksheet. Employees can specify preferences for more than 5 shifts during the week as long as they follow the same procedure (e.g., each employee can rank his first seven choices from 7 to 1). However, each employee would still be assigned the exact number of shifts specified in column *B* of the *Assignments* worksheet.

The *Constraints* Worksheet *Before* Optimization



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	SCREEN 4					CONSTRAINTS							
2	Name		Shift	Mon	Tue	Wed	Thu	Fri	Sat	Sun		SHIFTS	
3	REAGAN		Days	<=	<=	<=	<=	<=	<=	<=		Not =	
4			Eves	<=	<=	<=	<=	<=	<=				
5			Nites	<=	<=	<=	<=	<=	<=				
6													
7	Name		Shift	Mon	Tue	Wed	Thu	Fri	Sat	Sun		SHIFTS	
8	BUSH		Days	<=	<=	<=	<=	<=	<=	<=		Not =	
9			Eves	<=	<=	<=	<=	<=	<=				
10			Nites	<=	<=	<=	<=	<=	<=				
11													
12	Name		Shift	Mon	Tue	Wed	Thu	Fri	Sat	Sun		SHIFTS	
13	FORD		Days	<=	<=	<=	<=	<=	<=	<=		Not =	
14			Eves	<=	<=	<=	<=	<=	<=				
15			Nites	<=	<=	<=	<=	<=	<=				
16													
17	Name		Shift	Mon	Tue	Wed	Thu	Fri	Sat	Sun		SHIFTS	
18	NIXON		Days	<=	<=	<=	<=	<=	<=	<=		Not =	
19			Eves	<=	<=	<=	<=	<=	<=				
20			Nites	<=	<=	<=	<=	<=	<=				
21													
22													

The *Constraints* worksheet contains constraint formulas that ensure each employee works no more than one shift in any 24 hour period.

A. Determine Adjustable Cells

The adjustable cells in this model are cells D3:J5, D8:J10, D13:J15, and D18:J20 of the *Assignments* worksheet. These are the schedules for the individual staff members.

B. Define Best

The best solution maximizes the total staff preference (E4 of the *Model* worksheet), which is the sum of the individual preference totals. Individual totals are the product of the shift assignment variables (0 or 1) and the preference values assigned to each shift.

C. Specify Constraints

The constraints in this problem are:

- Staffing requirements for each shift must be met.
- Each staff member must work exactly the prescribed number of shifts during the week—no more and no less.
- After working one shift, a staff member can't be assigned to the next two consecutive shifts or work more than one shift in a single day.

To assure that the staffing requirements for each shift are met, the constraint formulas in D8:J8 of the *Model* worksheet require the number of people assigned to each shift to be greater-than-or-equal-to the number required. On the Monday *Days* shift (Model D8), for instance, the formula is:

```
=WB(ASSIGNMENTS! D3+ASSIGNMENTS! D8+ ASSIGNMENTS!
D13+ASSIGNMENTS! D18,">=",D7)
```

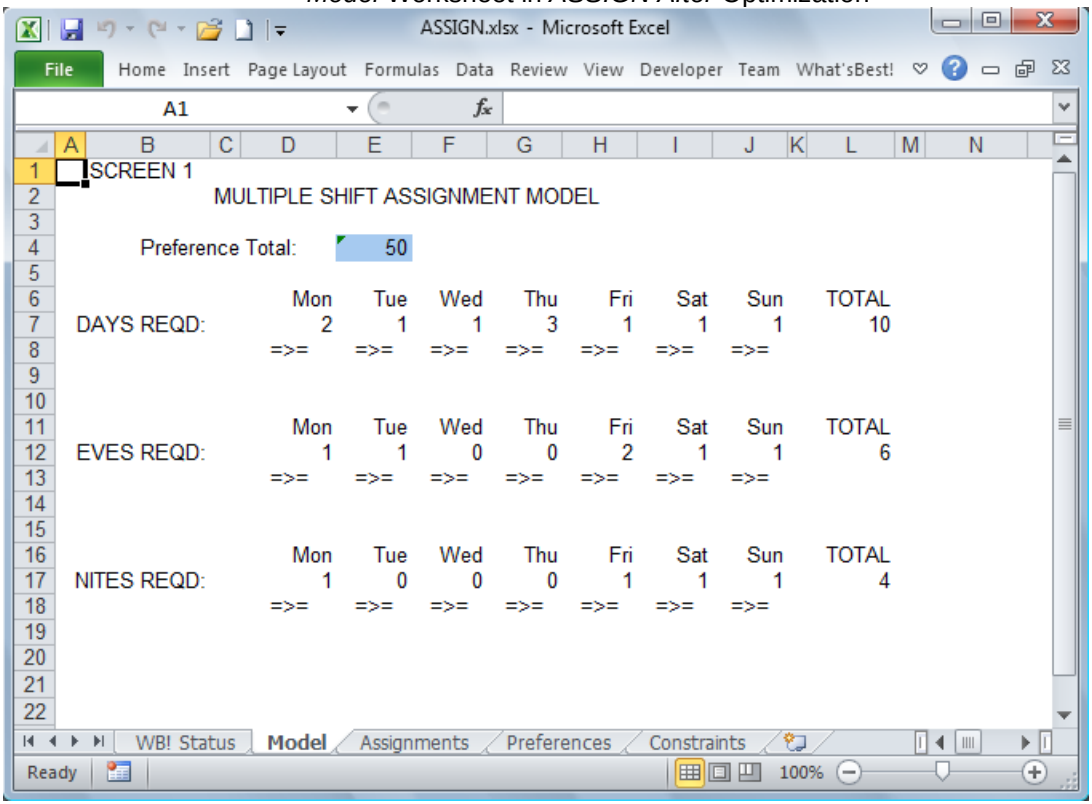
This adds the shift assignment variables for the Monday *Day* shift for all four staff members and forces this sum to be greater-than-or-equal-to (at least as great as) the number of people required for that shift.

The model uses an equality constraint for each staff member to force the number of shifts assigned each week to equal the number he's expected to work. The formula in cell L3 of the *Constraints* worksheet, for example, *WB*(ASSIGNMENTS! B3 ,"=", *SUM*(ASSIGNMENTS! D3:J3)), forces this equality for *Reagan*. Assigning him to more than 5 shifts causes the constraint to return "Not =".

Finally, the model must assure that, after working one shift, a staff member isn't assigned to either of the following two shifts and that each staff member is assigned to only one shift per day. This is accomplished by summing the assignment variables (0's and 1's) for each set of three consecutive shifts and forcing that total to be less-than-or-equal-to one. If, for instance, a staff member were assigned to two shifts out of a consecutive block of three, the constraint would be violated. Examine the formula in D3 and D4 of the *Constraints* worksheet, *WB*(*SUM*(ASSIGNMENTS! D3:D5),"<=", 1) and *WB*(ASSIGNMENTS! D4+ASSIGNMENTS! D5+ ASSIGNMENTS! E3,"<=", 1), which excludes *Reagan* from working more than one of the three Monday shifts or more than one of three shifts beginning with Monday *Evening*, respectively.

You're now ready to solve the problem.

Model Worksheet in *ASSIGN After* Optimization



The solution returns a *Preference Total* of 50. The scheduling requirements are tightly satisfied — no overstaffing has occurred — and each employee has been assigned to three or more preferred shifts.

Staff Scheduling: Two Stage Fixed Shift

File name: FIXED1.XLSX, FIXED2.XLSX *TYPE: LINEAR OPTIMIZATION*

Application Profile

Many organizations staff their personnel according to a number of non-overlapping daily shifts. Manufacturing plants, hospitals, and banks usually have 2 or 3 shifts per day with prespecified durations and starting times. For example, a hospital might have 3 daily shifts, each 8 hours in length, with 8:00 am, 4:00 pm, and midnight as the starting times. In this model, you need to fill these shifts at the lowest cost (i.e., with minimal overstaffing).

Another consideration is that worker preferences for certain shifts should be satisfied as much as possible. In fact, these may be not only preferences, but requirements under union work agreements. Here's a two-stage model for this specific situation.

Stage 1: Cost Minimization

The Problem in Words

The first stage is to determine the lowest-cost schedule that meets all staffing requirements. You can determine this schedule independent of specific personnel availability or characteristics.

You know staffing requirements for each day of the week. It's not important how these daily requirement figures are obtained, as long as they accurately reflect staffing needs. They could be based on the output of another model (such as a queuing model) or an extrapolation of historical figures. We also make the following assumptions in this model:

- There is only one skill level of staff — employees are interchangeable.
- Staffing requirements or *Full Time Equivalent* requirements (FTEs) can be met with either full time employees working five days per week or part time (or pool) employees that can be hired on a daily basis.
- Each full time employee cannot work more than 4 days in a row.

You could incorporate other assumptions with minor changes to the model.

Background

An employee's schedule for one time period is called a *work pattern*. Each employee can be scheduled under one of several work patterns. For example, a full time employee working five days a week can either work Monday through Friday or Tuesday through Saturday. These work patterns can be represented as follows:

Pattern #	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	1	1	1	1	1	0	0
2	0	1	1	1	1	1	0

A one indicates that the employee is working, and a zero that he is off.

There may be many patterns consistent with an organization's work rules. The first step in scheduling is to find a representative sample of patterns that correspond with organization rules or policies and meet other necessary criteria. The example shown here displays 10 patterns. Patterns 11 and 12 may be written in (or any of the supplied patterns may be written over). This process need not be repeated unless there is a change in the policies or rules.

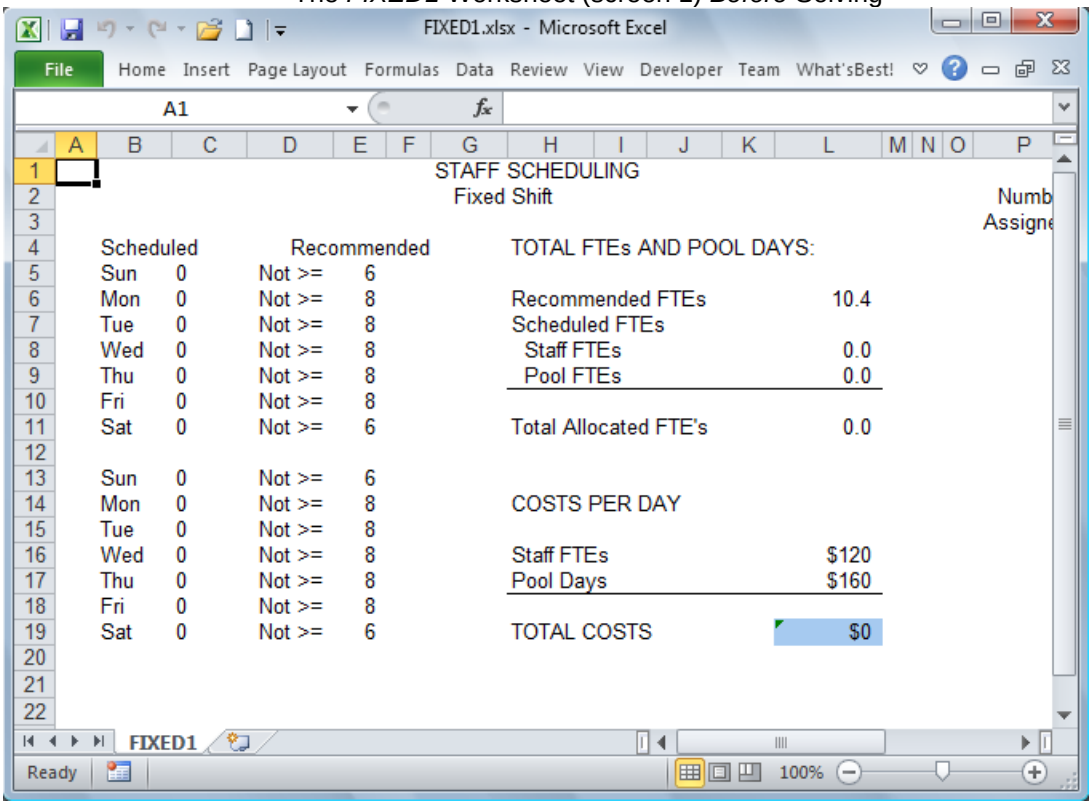
Objective of Optimization

The objective is to minimize staffing costs for a single shift. The model creates the minimum cost schedule based on daily requirement figures, average cost per employee category, and the work patterns. It chooses the best out of all work patterns that meet the staffing requirements. It also specifies how many people to schedule according to each work pattern and, if desired, how many 'pool' people to hire on a temporary basis to satisfy requirements.

The Worksheet

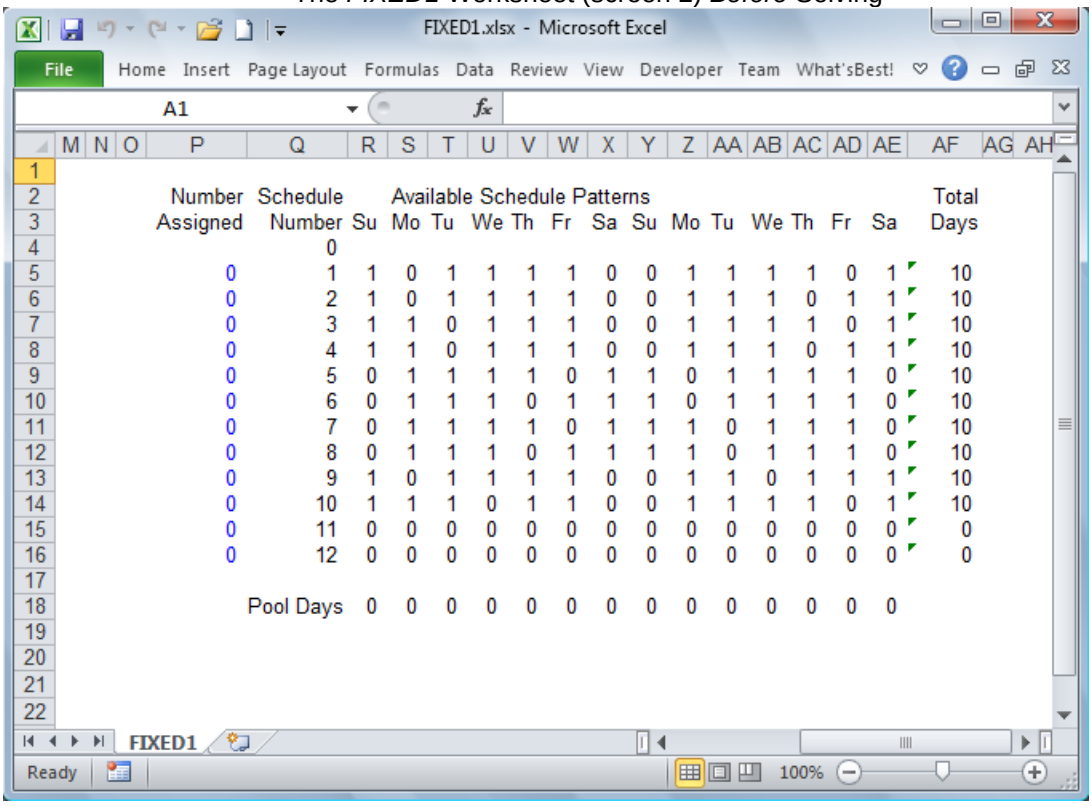
The model is formulated on the left and far right of the worksheet *FIXED1*. The first screen (A1:N20) contains the staffing and full time equivalent (FTE) requirements and coverage, and cost information:

The *FIXED1* Worksheet (screen 1) *Before* Solving



The second screen (P1:AG20) consists of possible individual schedules (R5:AE16), work patterns, and the adjustable cells (P5:P16):

The *FIXED1* Worksheet (screen 2) *Before* Solving



	Number Assigned	Schedule Number	Available Schedule Patterns Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Total Days
4		0															
5	0	1	1	0	1	1	1	1	0	0	1	1	1	1	0	1	10
6	0	2	1	0	1	1	1	1	0	0	1	1	1	0	1	1	10
7	0	3	1	1	0	1	1	1	0	0	1	1	1	1	0	1	10
8	0	4	1	1	0	1	1	1	0	0	1	1	1	0	1	1	10
9	0	5	0	1	1	1	1	0	1	1	0	1	1	1	1	0	10
10	0	6	0	1	1	1	0	1	1	1	0	1	1	1	1	0	10
11	0	7	0	1	1	1	1	0	1	1	1	0	1	1	1	0	10
12	0	8	0	1	1	1	0	1	1	1	1	0	1	1	1	0	10
13	0	9	1	0	1	1	1	1	0	0	1	1	0	1	1	1	10
14	0	10	1	1	1	0	1	1	0	0	1	1	1	1	0	1	10
15	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18		Pool Days	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Screen 2 depicts 10 work patterns that incorporate the assumptions and work rules outlined earlier. Each work pattern represents an employee working 5 days a week, 10 days per two week time period. Employees get one out of two weekends off and nobody works more than 4 days at a stretch.

Cell L16 contains the average daily cost per full time employee and cell L17 contains the average daily cost per pool employee. These estimates should be based on accounting and payroll data and can be updated on an as-needed basis.

A. Determine Adjustable Cells

The adjustable cells are the number of people scheduled according *to* each work pattern (P5:P16). Pool employees can be added to the model by making cells R18:AE18 adjustable and re-solving.

B. Define Best

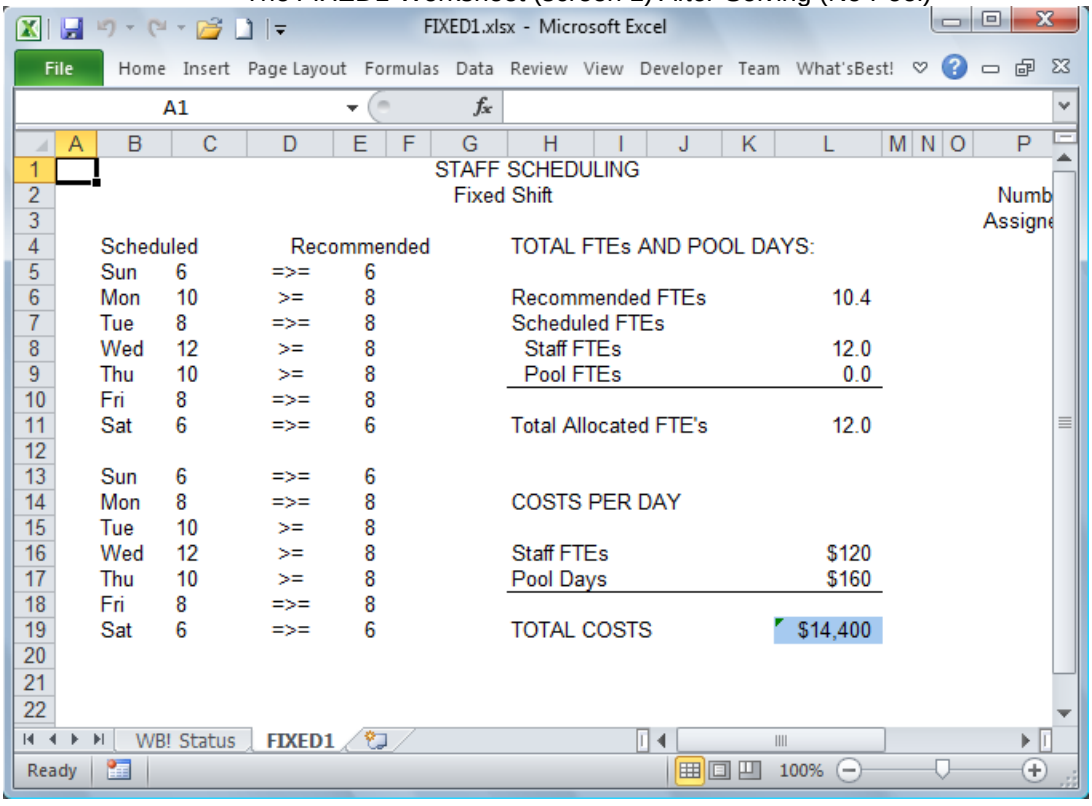
The best solution is the one that results in the minimum total cost in cell L19. Cell L6 contains the aggregate requirement level. L8 and L9 contain the minimum cost staffing levels with breakdown between full time and pool employees. Cell L11 contains the total employees scheduled and cell L19 is the objective of optimization, the total cost figure.

C. Specify Constraints

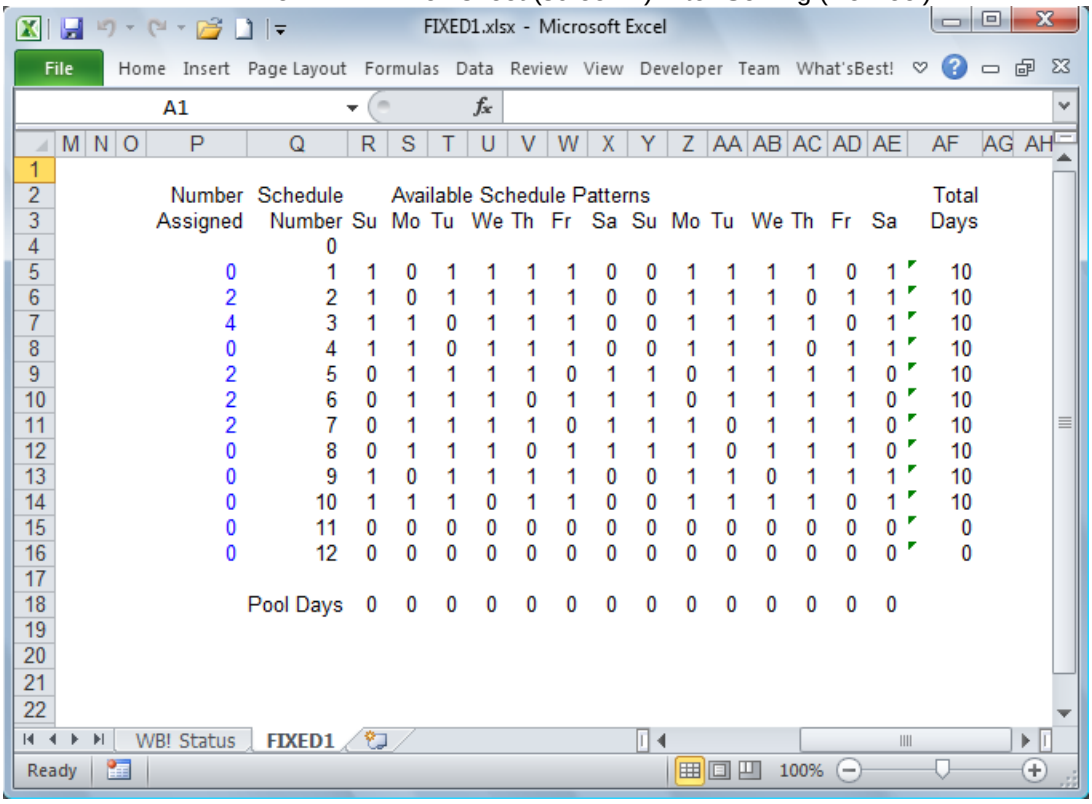
The constraints in cells D5:D11 and D13:D19 require that the employees scheduled (C5:C11 and C13:C19) for each day of the scheduling period be greater-than-or-equal-to staffing needs for that day (E5:E11 and E13:E19).

With the ABC's in place, go ahead and solve the model.

The *FIXED1* Worksheet (screen 1) *After* Solving (No Pool)



The *FIXED1* Worksheet (screen 2) *After* Solving (No Pool)



	Number Assigned	Schedule Number	Available Schedule Patterns Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Total Days
4		0															
5	0	1	1	0	1	1	1	1	0	0	1	1	1	1	0	1	10
6	2	2	1	0	1	1	1	1	0	0	1	1	1	0	1	1	10
7	4	3	1	1	0	1	1	1	0	0	1	1	1	1	0	1	10
8	0	4	1	1	0	1	1	1	0	0	1	1	1	0	1	1	10
9	2	5	0	1	1	1	1	0	1	1	0	1	1	1	1	0	10
10	2	6	0	1	1	1	0	1	1	1	0	1	1	1	1	0	10
11	2	7	0	1	1	1	1	0	1	1	1	0	1	1	1	0	10
12	0	8	0	1	1	1	0	1	1	1	1	0	1	1	1	0	10
13	0	9	1	0	1	1	1	1	0	0	1	1	0	1	1	1	10
14	0	10	1	1	1	0	1	1	0	0	1	1	1	1	0	1	10
15	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17																	
18		Pool Days	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

The above screens show the solution when only the cells for full time employees are adjustable. The total cost is $14,400. The full time staff needed to cover these requirements is 12 people.

Note: Though we've illustrated the model without pool employees as part of the solution, you can also use it if you have access to part-time or pool employees and need not rely only on full-timers to meet staffing needs. Just make the cells representing pool (R18:AE18) adjustable before solving. This will result in a $1,760 saving in this case since, with pool cells adjustable, there's much more flexibility in matching the *Scheduled* personnel with *Recommended* levels.

Sometimes the lowest cost solution results in fractional values for the adjustable cells. Although these fractional numbers provide the lowest cost, they are not practical for implementation purposes, since it's difficult to locate 2/3 of a person who is in any condition to work.

Fortunately, though, realistic whole number solutions can be obtained by rounding the fractional answers without sacrificing much in total cost. If the program returns non-integer values in cells P5:P16, or in R18:AE18 when the Pool is used, you can manually round these values to whole numbers. You could also use the *Integer* command to force whole number answers, but this can significantly increase the solution time. The objective in rounding these cells should be to make sure that the *Scheduled* >= *Recommended* constraints in D5:D11 and D13:D19 aren't violated and to have a final cost as close as possible to the one before you started rounding.

Now that the lowest-cost schedule has been found, the next step is to assign individual employees to the work patterns selected in this stage by preference.

Stage 2: Preference Maximization

The Problem in Words

The second stage model assigns existing staff members to the work patterns selected in stage 1. You know the preferences of each staff member from the ranks of all work patterns selected in stage 1 and the model assigns individuals to work patterns so as to maximize cumulative preference of the group. This model has provisions for specifying employee preference scores for work patterns, and individual seniority levels, performance indices, or a composite of the two.

Objective of Optimization

The objective of this model is to maximize the cumulative preference of the group while assigning staff members to the minimum-cost work patterns selected in stage 1.

The Worksheet

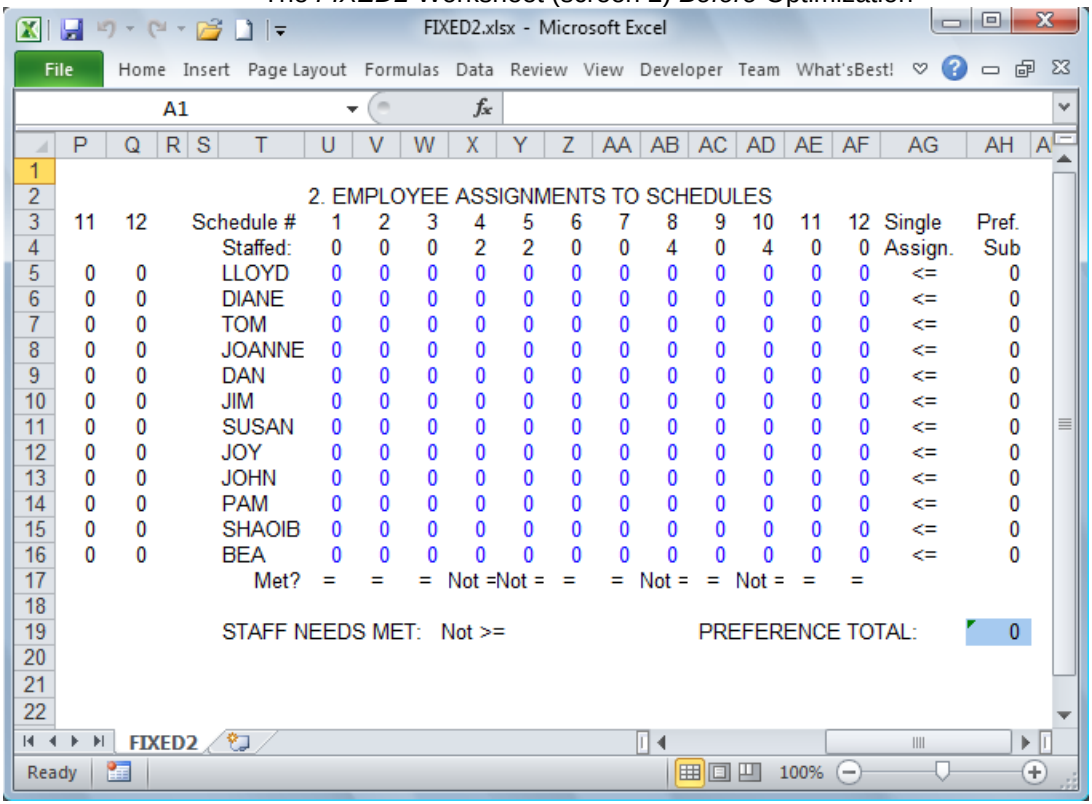
This model is also in 2 screens of the worksheet *FIXED2*. Screen 1 contains space for up to 12 individual employee names (B5:B16), the employee seniority-performance index (D5:D16) and space for work pattern preferences (F5:Q16).

Screen 2 contains the adjustable cells, employee names and their assignments and a preference subtotal for each employee.

You must now copy the optimal work patterns selected information from cells P5:P16 of the Stage 1 model after optimization and enter into cells U4:AF4. For this example, we've used the optimal scheduling generated without using *Pool FTE's*.

After the data from the first stage is entered, the model looks like this:

The *FIXED2* Worksheet (screen 2) *Before* Optimization



A total of 10 people need to be scheduled, 4 each for Schedules 2 and 8, and 2 each for schedules 3 and 5.

Each person's preference score is multiplied by his or her seniority index to get a weighted preference score. Employees are assigned to work patterns according to these weighted scores, with the highest-score employees assigned first. The only exception to this rule occurs when assigning an employee with a high score to a first choice. It may be necessary to assign employees with low weighted scores to their lowest preferences. In such instances, the program would assign the senior employee to the second or third choice so as to benefit the junior staff members. This is in keeping with the objective of maximizing group preference rather than individual preferences.

To follow strict seniority in staff assignment, just use a different scale for seniority/performance scores. For example, vary the scores between zero and 100 rather than 10 and 20. The first scoring scheme would place a much higher emphasis on senior employees than the latter one.

You can tailor the model to meet specific needs with only slight modifications in the structure of input data. The same is true for the stage 1 model.

A. Determine Adjustable Cells

The adjustable cells in U5:AF16 of *FIXED1* show whether a person is assigned to a work pattern. There are 12 adjustable cells corresponding to each individual being assigned, indicating what one of the 12 possible work patterns that person is assigned to. A 1 in an employee's row indicates that he is assigned that column's schedule number.

B. Define Best

The best solution is the one that maximizes total preference score for the group. Total preference is shown in AH19. This is the sum of the preference subtotals for each employee.

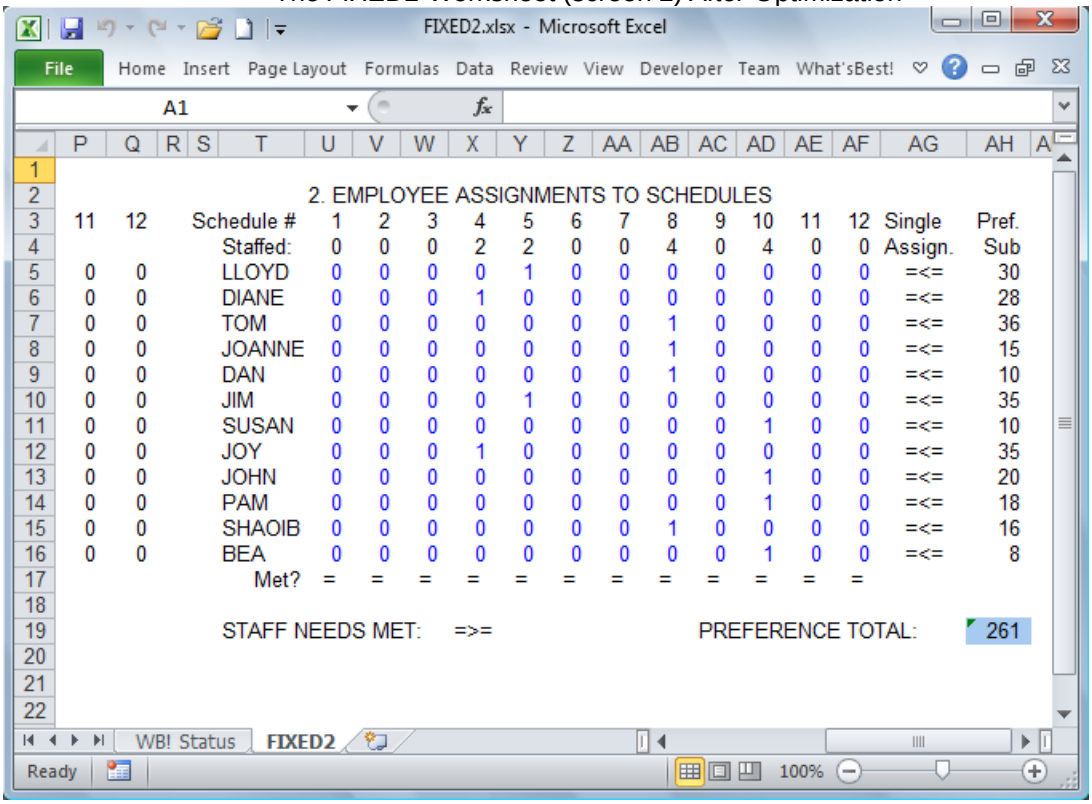
C. Specify Constraints

Constraints are of the following three types.

- No more work patterns must be assigned than recommended by the stage 1 optimization (X19),
- Employees must be assigned to only the work patterns selected in stage 1 (U17:AF17), and
- Each person should be assigned to only one work pattern (AG5:AG16).

Now, let's solve the model:

The *FIXED2* Worksheet (screen 2) *After* Optimization



The maximized preference total in cell AH19 is 150 and the staffing needs established by stage 1 are met. To verify that assignments have been made according to seniority and preference, go back to screen 1 and observe that *Tom*, in row *7*, with a seniority level of 9, has rated shift *8* as his highest preference. After solving, *Tom* has indeed been assigned to shift *8*.

Pipeline Optimization

File name: PIPELINE.XLSX *TYPE: LINEAR OPTIMIZATION*

Application Profile

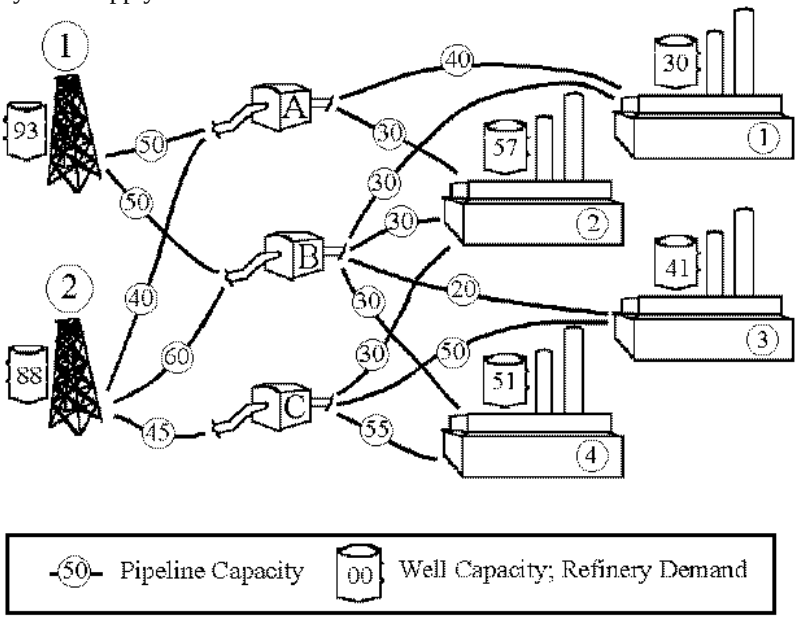
This model is an example of a network problem — requiring the movement of resources, at minimum cost, along different routes with which varying costs are associated. With the addition of limits to capacity along the routes, it becomes "capacitated". Oil and gas pipelines, truck and air routes may be utilized at their highest cost efficiency by applying the principles demonstrated in this model.

The Problem in Words

As the operator of an oil supply network, you must choose among several available pipelines from two wells to three pumping stations and from the pumping stations to four refineries. Wells have a monthly supply capacity that must not be exceeded; capacities of the pipelines may be limited, and costs vary among the pipelines. In addition, monthly refinery demand must be fully met.

Background

You must decide how many barrels per month to pump along each pipeline. At present, no pipeline is operating between *Well 1* and *Pump C*, between *Pump A* and *Refineries* 3 and 4, and between *Pump C* and *Refinery* 1. The decision on how much material to send along a given pipeline is governed by the monthly cost per unit on that pipeline and by the necessity of satisfying refinery demand without exceeding monthly well supply.



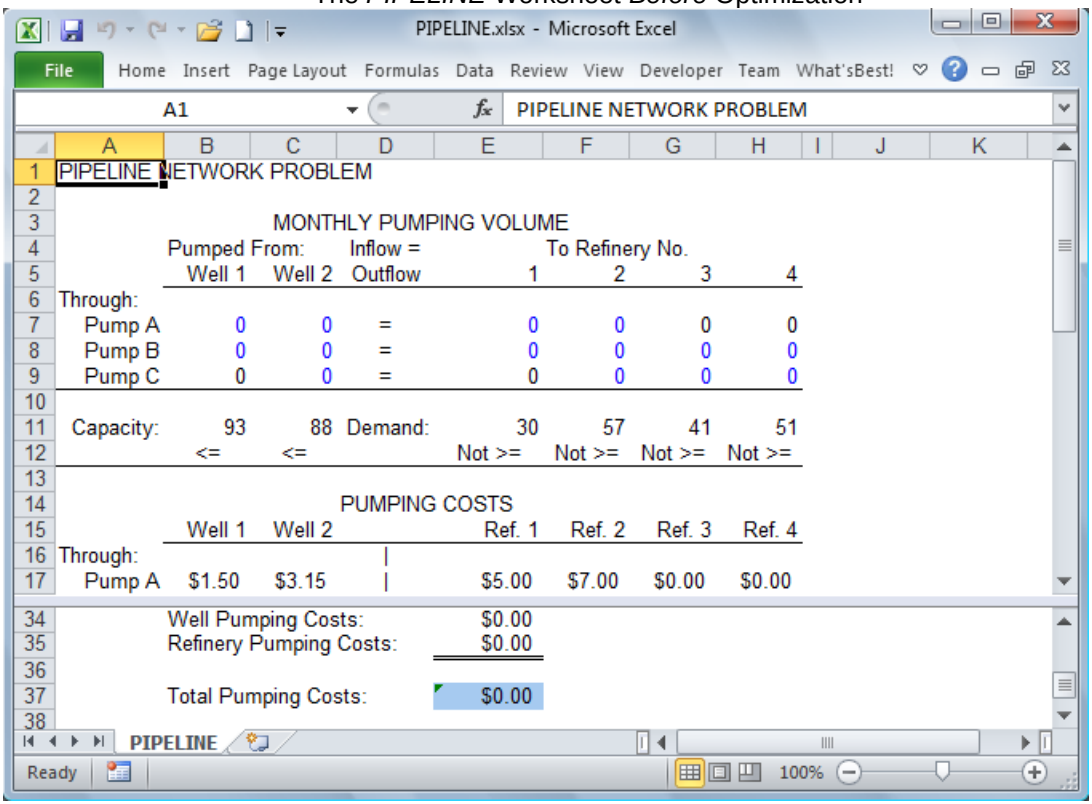
Objective of Optimization

The objective in this model is to minimize the *Total Pumping Cost* without exceeding either the monthly output capacity of the wells or the capacity of the pipelines, while meeting demand at each refinery.

The Worksheet

Open the *PIPELINE* sample file and let's have a look at it.

The *PIPELINE* Worksheet *Before* Optimization



A. Determine Adjustable Cells

The adjustable cells are the amounts sent from the two *Wells* to the three *Pumps* (B7:B8, and C7:C9) and the amounts sent from the three *Pumps* to the *Refineries* (E7:E8, F7:F9, G8:G9, H8:H9). Cells B9, E9, G7, and H7 are fixed cells with a value of zero representing pipelines that are not open.

B. Define Best

The best solution to this problem is the one that minimizes *Total Pumping Cost* in cell E37. The formula in E37 is the sum of the two subtotals for *Well* and *Refinery* pumping costs. Each subtotal in turn is the sumproduct of pumping costs along each open pipeline (from the *Wells* to the *Pumps* in the first instance and from the *Pumps* to *Refineries* in the second) and amounts carried on that pipeline.

C. Specify Constraints

The constraints in the *PIPELINE* model must perform several functions. First, supply capacity at the *Wells* must not be exceeded. In cells B12 and C12, constraint formulas require that the sum of amounts pumped from each *Well* is less-than-or-equal-to the *Capacities* (B11 and C11).

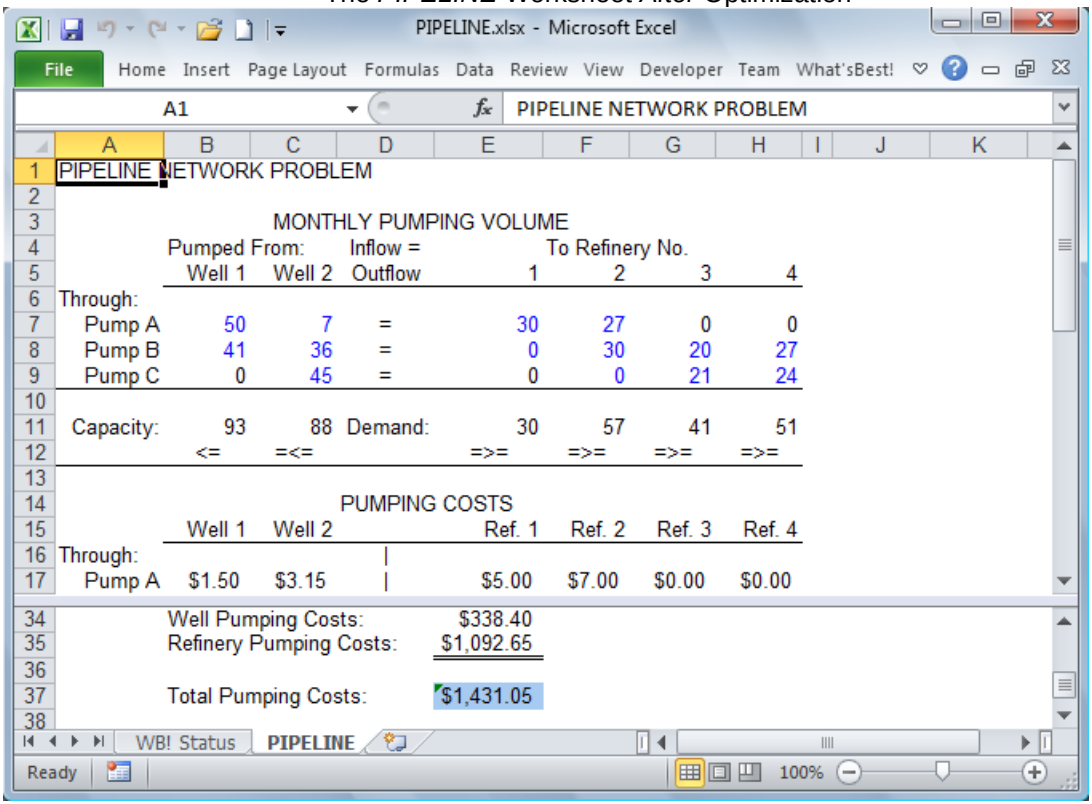
Second, the amount received at each *Pump* must equal the amount pumped out. Such constraints are known as conservation constraints. To achieve this, cells D7:D9, contain constraint formulas forcing the sum of amounts pumped from the *Wells* to each *Pump* to equal the sum of amounts pumped from that Pump to the *Refineries*.

Third, demand at the *Refineries* must be satisfied. These constraints are in cells E12:H12 and require that the sum of amounts pumped from the *Pumps* to each *Refinery* is greater-than-or-equal-to the demands in E11:H11.

Finally, the capacities of the pipelines cannot be exceeded. The constraint formulas in B29:C31 and E29:H31, one for each “arc”, accomplish this result.

Now, let's solve the model.

The *PIPELINE* Worksheet *After* Optimization



What's*Best!* has returned a minimized *Total Pumping Cost* of $1,431.05. Note that all *Pumps* are in use.

Shipping Cost Reduction

File name: SHIPPING.XLSX *TYPE: LINEAR OPTIMIZATION*

Application Profile

This problem is typical of the class of network problems that generally involve the shipping of goods through transportation networks or of oil or gas through systems of pipelines.

The particular shipping example illustrates the use of optimization to provide the least-cost distribution for a business that produces a product at many locations, transports it to intermediate warehouses, and then to final consumption or sales locations. Manufacturers, utilities, retail chains, and other distributors are all likely users of shipping optimization systems modeled along similar lines.

The Problem in Words

You work in the shipping department of a manufacturing business and need to minimize shipping costs while meeting demand without exceeding capacity.

Background

In this case, two steel mills produce the same product. This product must be sent out to three plant locations. Each location has a demand for the product that must be met and each steel mill has limited manufacturing capacity. Finally, shipping costs vary from each steel mill to each plant.

Shipping Costs Per Unit of Steel

From:	Steel	Steel
To:	**Mill 1**	**Mill 2**
Plant A	$200	$500
Plant B	$300	$400
Plant C	$500	$600

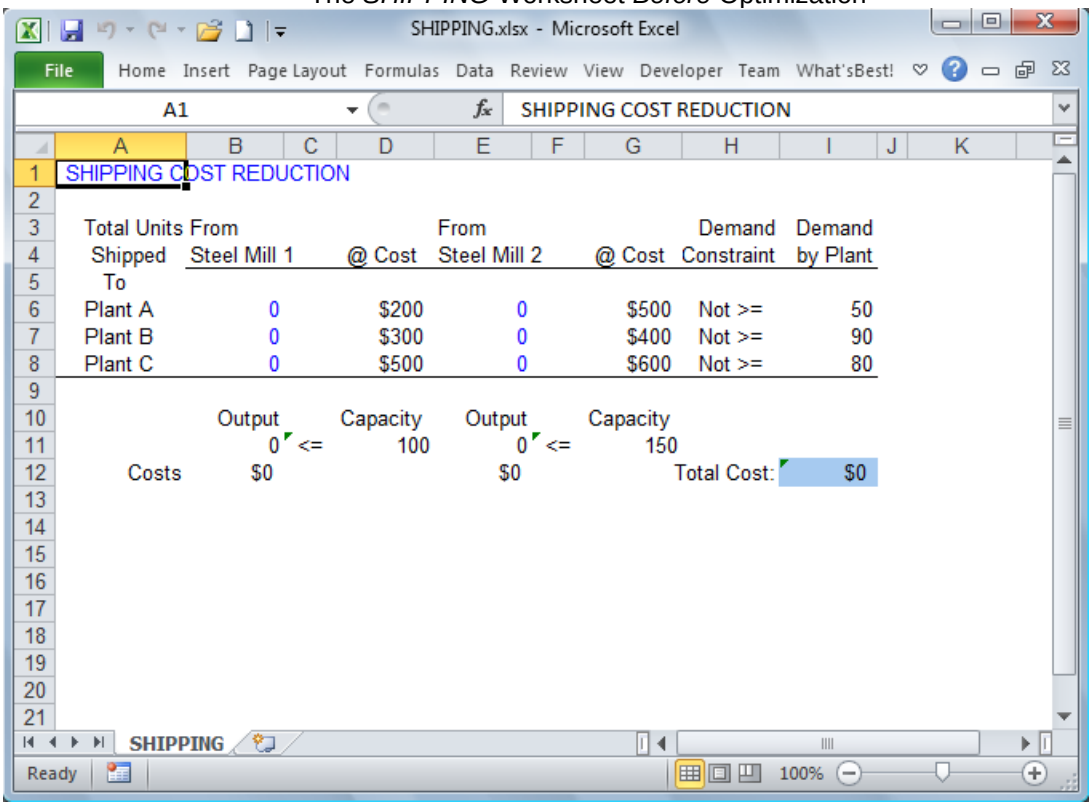
Objective of Optimization

The objective is to meet all plant demand without exceeding steel mill capacity at minimum shipping cost.

The Worksheet

Let's look at the *SHIPPING* sample file.

The *SHIPPING* Worksheet *Before* Optimization



A. Determine Adjustable Cells

The adjustable cells in this model are the quantities to be shipped from each steel mill to each plant (B6:B8 and E6:E8).

B. Define Best

The best solution minimizes *Total Cost* in cell I12. The *Total Cost* formula is the sum of the two *Cost* per *Mill* cells in B12 and E12. Each of these is the sumproduct of the *Units Shipped* from each *Mill* to each *Plant* (B6:B8 and E6:E8) and the cost associated with each shipping arc (D6:D8 and G6:G8).

C. Specify Constraints

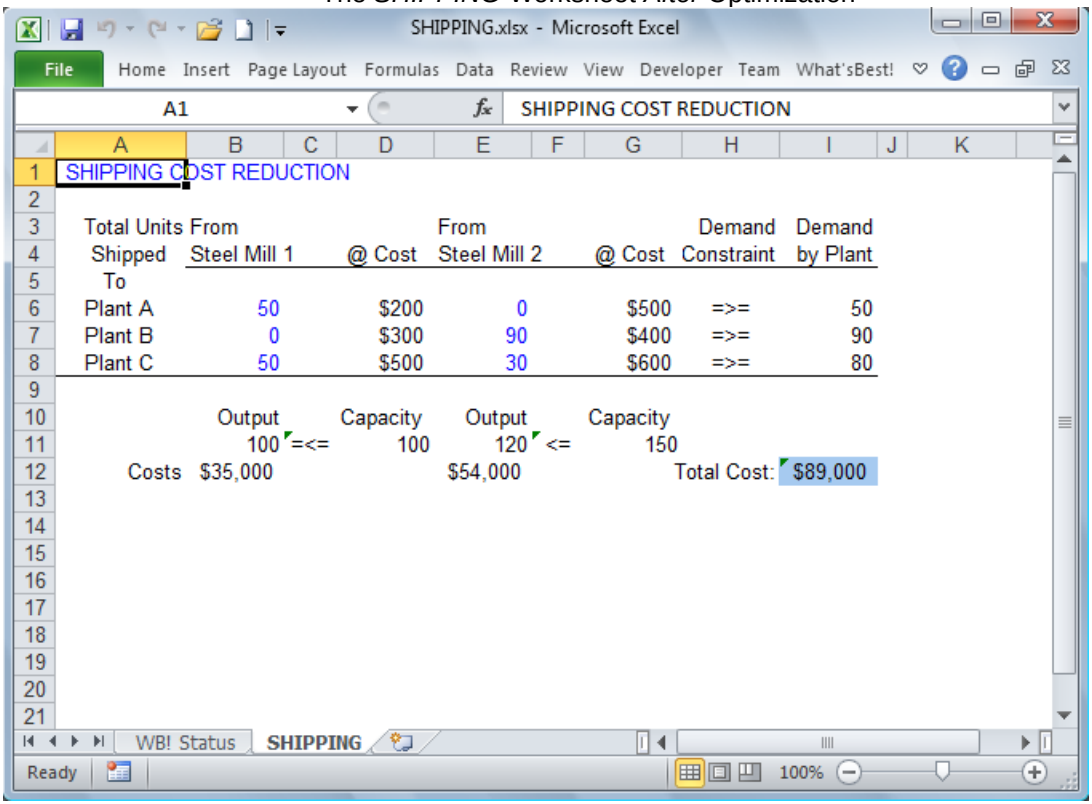
There are two sets of constraints. First, each plant location requires certain levels of shipments. Second, the production of each steel mill must not exceed its capacity.

The constraints in H6:H8 force the total shipped to each plant to be greater-than-or-equal-to that plant's demand (I6:I8).

To guard against exceeding the capacity of a *Steel Mill*, the capacity constraints in C11 and F11 require that a *Steel Mill's* output (the sum of amounts shipped from it) be less-than-or-equal-to that *Mill's* capacity.

Now, let's solve the model.

The *SHIPPING* Worksheet *After* Optimization



The least-cost solution results in shipping costs of $89,000.

D. Dual Values

The solution recommends shipping nothing from *Mill 2* to *Plant A*. To find out how much your total cost will increase if you transport anything between these two points, use the *Advanced|Dual* command to display in an empty cell the dual value for cell E6. Then, optimize again and you'll find a dual value of 200. This means that your *Total Cost* would increase by $200 if you were to ship one unit along that arc.

If you perform the same operation for cell B7, the amount shipped from *Mill 1* to *Plant B*, you'll find a dual value of zero. Dual values are generally positive for adjustable cells that have a zero value in the solution. An exception to this occurs in cases where there are multiple solutions, as in the current problem. Because no penalty is incurred in moving from one optimal solution to another, such a problem is said to have multiple optima — there is more than one combination of adjustable cells that give the same minimum cost.

Traffic Congestion Cost Minimization

File name: TRAFFIC.XLSX *TYPE: NONLINEAR OPTIMIZATION*

Application Profile

In the earlier *SHIPPING* network problem, the cost to transport your product from a supply point to a demand point was fixed. However, in some network problems, the costs vary with the amount transported along each arc. If you've ever driven to or from a major city during rush hour, you've experienced this phenomenon. As the number of cars on the road increases, the cost, in terms of time required, of getting from point *A* to point *B* increases. In many cases, the cost does not increase linearly. For example, doubling the traffic on a lightly traveled road may not double the travel time, but doubling the traffic again may effectively grind the flow of vehicles nearly to a halt.

The Problem in Words

As Quartermaster of a military base, you need to distribute uniforms from three warehouses to four intake centers within the base.

Background

You know that the time required to go from a particular warehouse to a particular unit obeys the following formula:

Time = *Rate* * *Flow* / (1 - *Flow* / *Limit*)

Where:

Rate = time required to transport one unit if there is no congestion along this route
Flow = amount of product moving along this route
Limit = the maximum amount that can be moved along this route

You also know the rates and limits for each route, or arc, in the network.

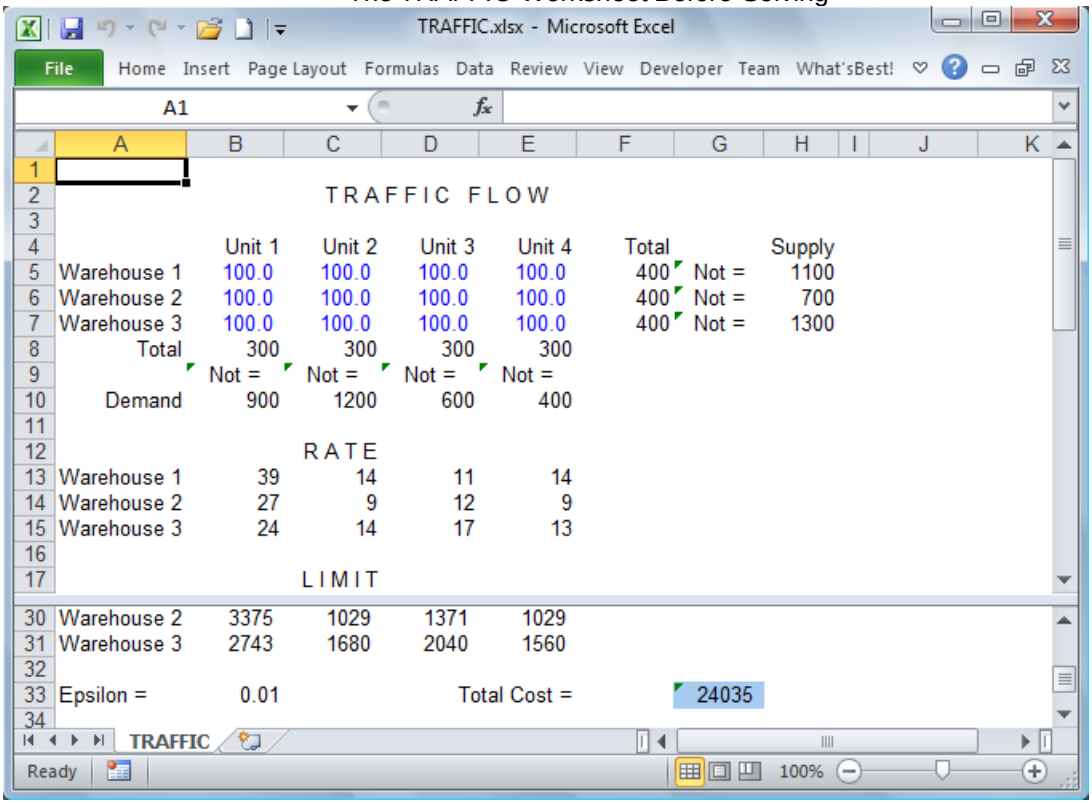
Objective of Optimization

The objective is to ship all the uniforms to the *Intake Centers* at minimum cost, while satisfying demand at each center.

The Worksheet

Let's look at the sample file *TRAFFIC*.

The *TRAFFIC* Worksheet *Before* Solving



A. Determine Adjustable Cells

The adjustable cells are the amounts shipped along each arc from *Warehouse* to *Intake Center*, in B5:E7.

B. Define Best

The best solution, in C36, is the minimized sum of all time penalties.

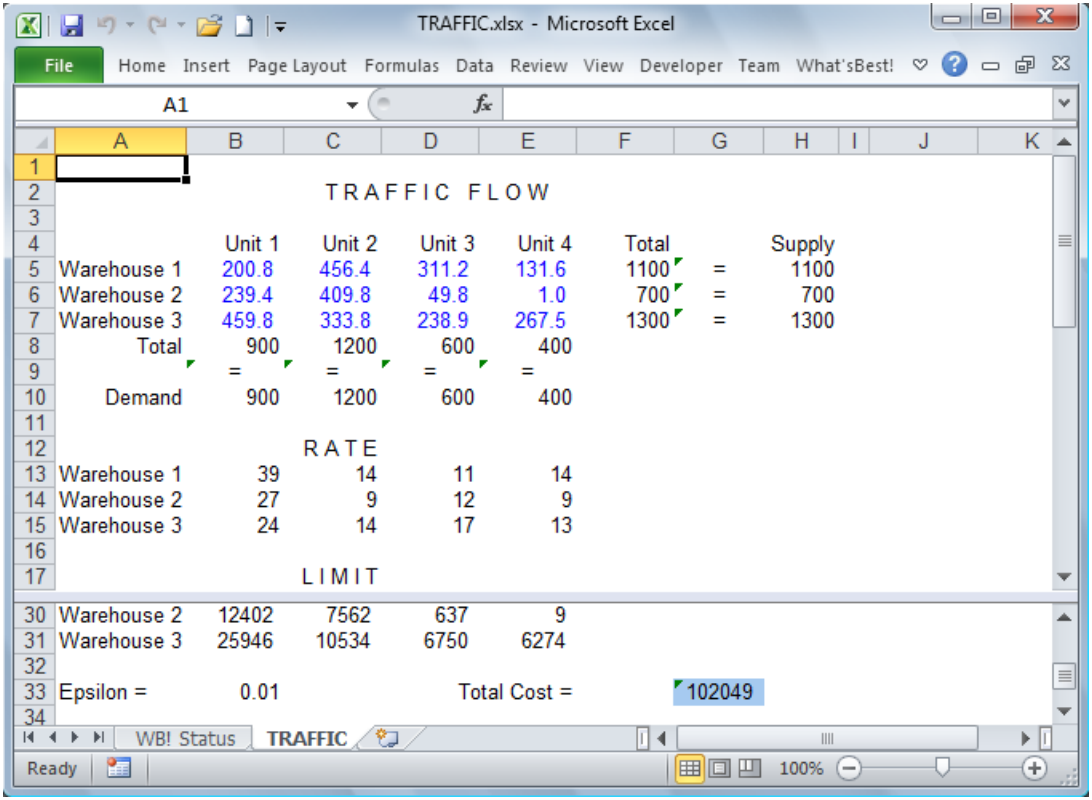
C. Specify Constraints

There are three groups of constraints in the model. In G5:G7, the *Total* shipped from each *Warehouse* (F5:F7) is forced to be equal-to its total *Supply* (H5:H7). In B9:E9, the amount shipped to each intake center (B8:E8) is forced to be equal-to its demand (B10:E10). In B23:E25, the amount shipped along each route (B5:E7) is required to be less-than-or-equal-to that route's maximum capacity minus *Epsilon*, .01, in cell B33 (see note below).

Note: As traffic *Flow* approaches the *Limit*, the *Time Penalty* goes to infinity. Observe, for instance, the formula for *Warehouse 1⇒ Unit 1* in cell B23: B13*B5/(1-B5/B18). When *Flow* (B5) = *Limit* (B18), the formula to calculate time is undefined because it results in division by zero. The user should build models that avoid such mathematically undefined regions. Therefore, rather than constraining the flow along each route to be less-than-or-equal-to its limit, we've constrained it to be less-than-or-equal-to the limit minus a small amount we call *Epsilon*.

Now, let's solve the model:

The *TRAFFIC* Worksheet *After* Solving



Truck Loading

File name: TRUCK.XLSX *TYPE: LINEAR OPTIMIZATION*

Application Profile

This is an example of the *Knapsack* class of problems, in which a number of things — boxes, books, or Bradley Fighting Vehicles — must be efficiently or profitably packed in a container (trucks, crates, or C130 aircraft). The objective can be to minimize wasted space in the container, to maximize or minimize total load weight, or, as in this case, to maximize the value of the load.

The Problem in Words

You have to decide which items to load on a truck. Each item must be shipped in its entirety or not at all (i.e., 25% of an item cannot be shipped). The *Truck Loading* problem illustrates a situation in which non-integer (fractional) answers are not acceptable. To demonstrate potential problems involved in rounding fractional answers, we'll solve the problem using two different methods.

In the first case, we'll use What's*Best!* to optimize by the conventional method. The solution may suggest fractional answers, but we'll just round the fractions to the nearest whole number that does not violate any constraints. In the second case, we'll find an integer solution using the binary Integer command.

Background

Items are to be loaded onto a truck with a 10,000 pound capacity. However, the items currently scheduled for shipment will exceed the capacity of the truck, so you have to make a yes/no decision as to whether each item gets shipped.

Each item has an associated dollar value and weight as shown below.

Dollar Value and Weight for Each Item

Item	Value	Weight
1	$22,500	7,500 lbs.
2	$24,000	7,500 lbs.
3	$ 8,000	3,000 lbs.
4	$ 9,500	3,500 lbs.
5	$11,500	4,000 lbs.
6	$ 9,750	3,500 lbs.

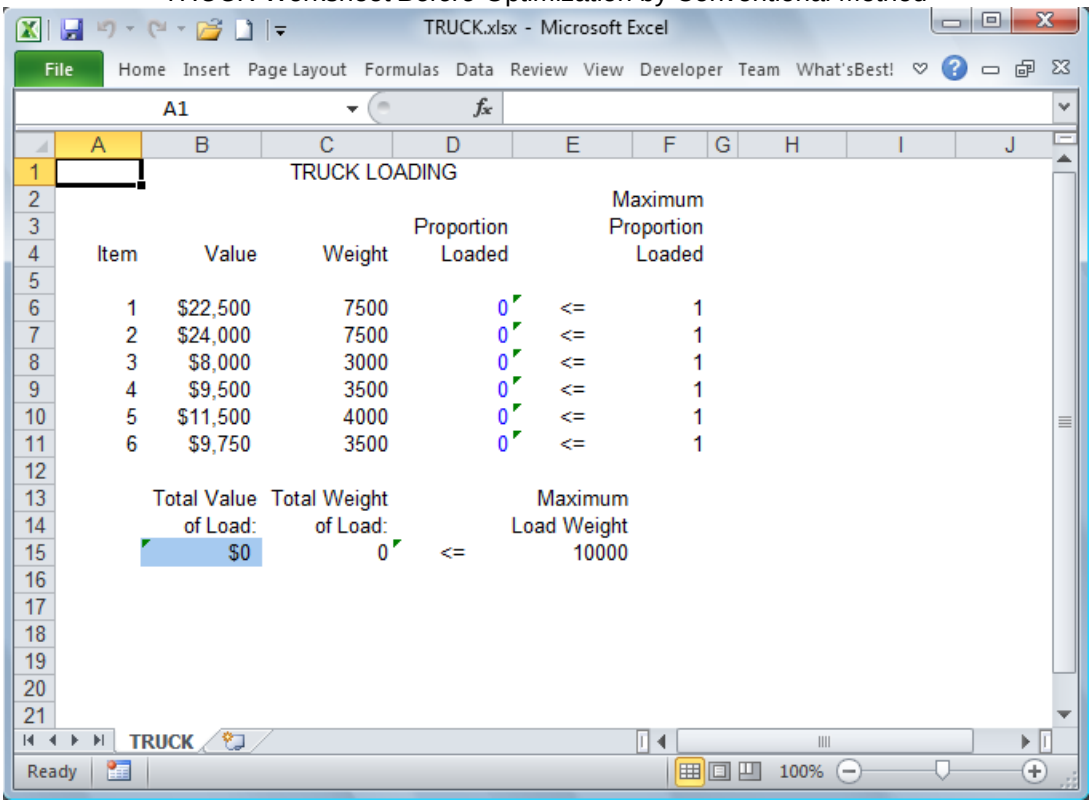
Objective of Optimization

The objective of optimization is to maximize the total value of the items loaded onto the truck without exceeding the truck's weight capacity.

The Worksheet

Let's look at the *TRUCK* sample file. The six items scheduled for shipment (A6:A11) along with their corresponding dollar *Values* (B6:B11) and *Weights* (C6:C11) appear in the worksheet.

TRUCK Worksheet *Before* Optimization by Conventional Method



A. Determine Adjustable Cells

The adjustable cells are the *Proportion Loaded* of each item, in D6:D11.

B. Define Best

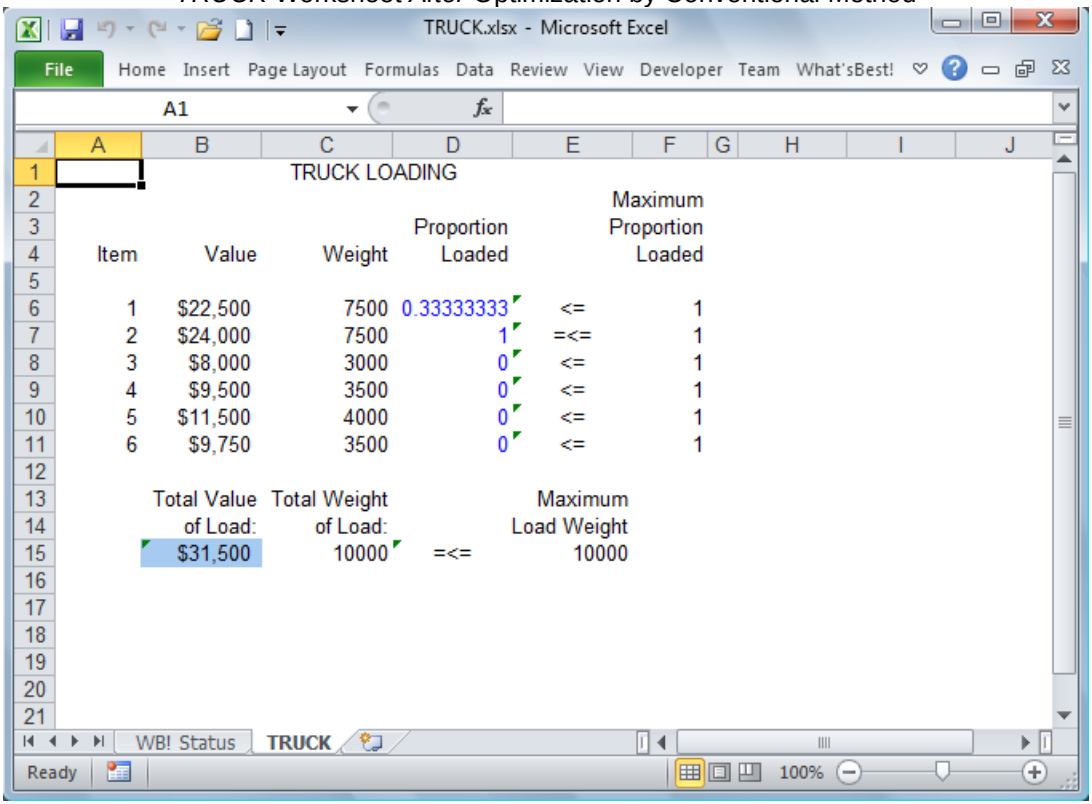
The best solution is the one that maximizes the value of the load in cell B15. The formula there, *SUMPRODUCT*(B6:B11,D6:D11), multiplies each individual item's value times the *Proportion Loaded* and sums them.

C. Specify Constraints

There are two constraints. The first, in D15, specifies that the *Total Weight of Load* (C15) remain less-than-or-equal-to the *Maximum Load Weight* (E15). The second, in E6:E11, requires that the number of each item loaded onto the truck is less-than-or-equal-to 1.

Now, let's solve the worksheet.

TRUCK Worksheet *After* Optimization by Conventional Method



You'll notice that the What's*Best!* solution recommends loading 33% of item 1 and all 100% or ("1") of item 2. While this is the best solution if fractions of items can be loaded, it may be difficult to load (or sell) one-third of a piano. To eliminate the fractions, you could do some "What If?" trials and round them in the direction that doesn't violate the maximum load constraint.

If you round the proportion up to 1.00, the weight of the load increases to 15,000 pounds. This violates the 10,000 pounds maximum weight constraint. Instead, round the fraction of item 1 down to zero. This drops the value of the load to 7,500 pounds, leaving 2,500 pounds of load capacity unused. No other item weighs less-than 2,500 pounds, so the truck can't be filled up further.

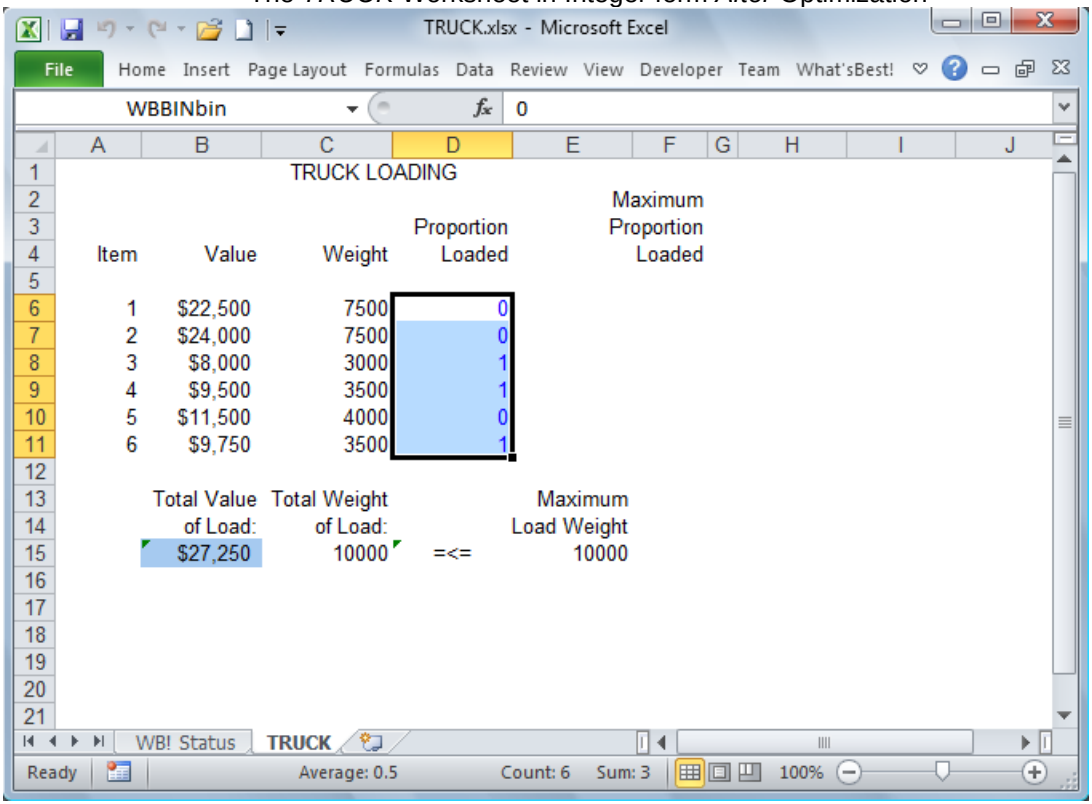
Rounding fractional weights, as in the conventional method, the truck would carry only item 2 for a load weight of 7,500 pounds and a load value of $24,000.

The Binary Integer Method

This problem can be converted very simply to give an integer solution — one with no fractional weights for an item. Use the binary *Integer* command to specify that the adjustable cells in D6:D11 cannot be anything other than zero or one. You could also eliminate the constraints in E6:E11 and the constants in F6:F11, which are now superfluous.

The re-optimized solution should look like the following:

The *TRUCK* Worksheet in Integer form *After* Optimization



Note that What's*Best!* has returned a value of 1 in the *Proportions Loaded* cells of *Items* 3, 4, and 6 and a 0 for the remaining items. This results in a total load weight of 10,000 pounds, and a total load value of $27,250. This total load value is $3,250 greater-than that achieved by rounding fractional answers.

Rounding versus Integer Solutions

The important things to notice about the integer solution are that the truck's capacity is more fully utilized and, more importantly, that the value of the load is $3,250 higher.

As a rule, rounding of fractional answers is effective when it results in large integers. When the integers are small, as in this problem, rounding is less effective.

Crop Allocation Under Uncertainty

File name: SP_CROPALLOC.XLSX *TYPE: LINEAR OPTIMIZATION - STOCHASTIC*

Application Profile

The stochastic farm allocation model is an example of a single stage stochastic optimization problem. In this problem, a random event occurs at stage 1 of the process. To handle this random event, a decision has to be made at stage 0. This decision tries to optimize an objective function over all the possible realizations of the random event.

The Problem in Words

A farmer can grow wheat, corn or beans on his 500 acres of farm land. He requires 200 tons of wheat and 240 tons of corn to feed his cattle. For domestic use, he requires 100 tons of beans.

He can either grow these crops or buy them from the market. The plantation cost for the crops is given by $150 /acre (wheat), $230 /acre (corn) and $260 /acre (beans). The crops can be sold in the market at $170 /ton (wheat), $150 /ton (corn) and $200 /ton (beans). Any shortfall can be purchased from the market at $238 /ton (wheat) and $210 /ton (corn) and $270 /ton (beans).

The yields (in tons /acre) of the crops depend on the weather. The farmer believes that the weather can be good, fair, bad, or very bad with equal probability. If the yield of a crop is 'y' in fair weather, then the variation of the yield with weather is given by:

Good	1.2y
Fair	y
Bad	0.8y
Very bad	0.6y

The yields (in tons/acre) of the various crops under the various weather scenarios are:

	Wheat	Corn	Beans
Good	3	3.6	24
Fair	2.5	3	20
Bad	2	2.4	16
Very bad	1.5	1.8	12

The farmer has to decide how much land to allocate to each crop so as to maximize his profit.

The Worksheet

The worksheet has two sections:

1) A section where we enter the data, the adjustables, constraints, randoms, and the objective function for the problem. This is similar to the section that we create for solving a deterministic optimization problem.

2) A section where we enter the stochastic information.

Information can be entered directly onto the spreadsheet or via the set of dialog boxes.

Step 1 (core model):

1 Data and Formulas

Specify the quantity required, plantation cost, selling price and cost price for all the four crops. Also specify the total area of the farm.

Enter the formulas associated with the plantation and purchase cost, and amount from selling.

2 Adjustable Cells

For each crop, the adjustable cells correspond to the following:

a) Area Allocated
b) Quantity Harvested
c) Quantity Purchased

3 Objective Function

The objective function is the profit given by:

Profit = Amount obtained from Selling the Excess Crop - Planting Cost - Purchase Cost

4 Constraint Cells

a) For each crop c, the quantity of crop purchased plus the product of the yield and the area allocated to that crop is equal to the quantity sold and required.

i.e. Quantity(c) Harvested + Purchased = Quantity Sold + Required
where Quantity Harvested = Area Allocated * Random Yield per Acre

b) The sum of the areas allocated to all the crops should be less than or equal to the 'total area' of the farm.

i.e. sum(area(c)) <= total area

5 Random Parameters

Finally the 'Yield per Acre' cells will be read as random parameters.

Step 2 (stochastic information for stage):

Initial Decision Variables

Specify the Area Allocated to each crop as an initial variable. This is done by using the function WBSP_VAR. This function requires two parameters:

i) stage of the variable
ii) cell address of the variable

In this problem, the Area Allocated to each crop is an adjustable that has to occur at stage 0.

Random Parameters

Enter the yield information for all the crops under the various weather scenarios.

Then declare the Yields of each crop as a random variable. This is done by using the function WBSP_RAND. This function requires two parameters:

i) stage of the variable
ii) cell address of the variable

In this problem, the Yields per Acre of the crop are the random variables that are realized at stage 1.

Stage Information for Recourse Variables

Declare the stage information for each recourse variables.

The following variables (adjustables) are assigned a stage 1 using the function WBSP_VAR:

a) Quantity Sold
b) Quantity Purchased

The stage is 1 because these variables depend on the random variables and therefore take a value only at stage 1.

Stage Information for the Constraints and the Objective Function

The What's*Best!* interface will automatically assign the objective function and all the constraints at stage 1, which depend on the stage value of the previous variables.

The user can also assign these variables to a stage 1, using the WBSP_VAR function.

Step 3 (distribution information for the random parameters):

Declare a joint discrete distribution for the yields of the various crops. This is done by using the function WBSP_DIST_DISCRETE_SV. This function takes in two parameters:

i) the set of possible values of the random variable (set vertically)
ii) the cell address of the random variable

Step 4 (scenario information):

Specify the scenario information by using the function WBSP_STSC. This function requires a two-column table as an argument:

1) Column 1 for the number of stages in ascending order
2) Column 2 for the respective number of scenarios

In this problem, the number of stages is 1 and the number of scenarios is 4.

Step 5 (reporting cells, histogram):

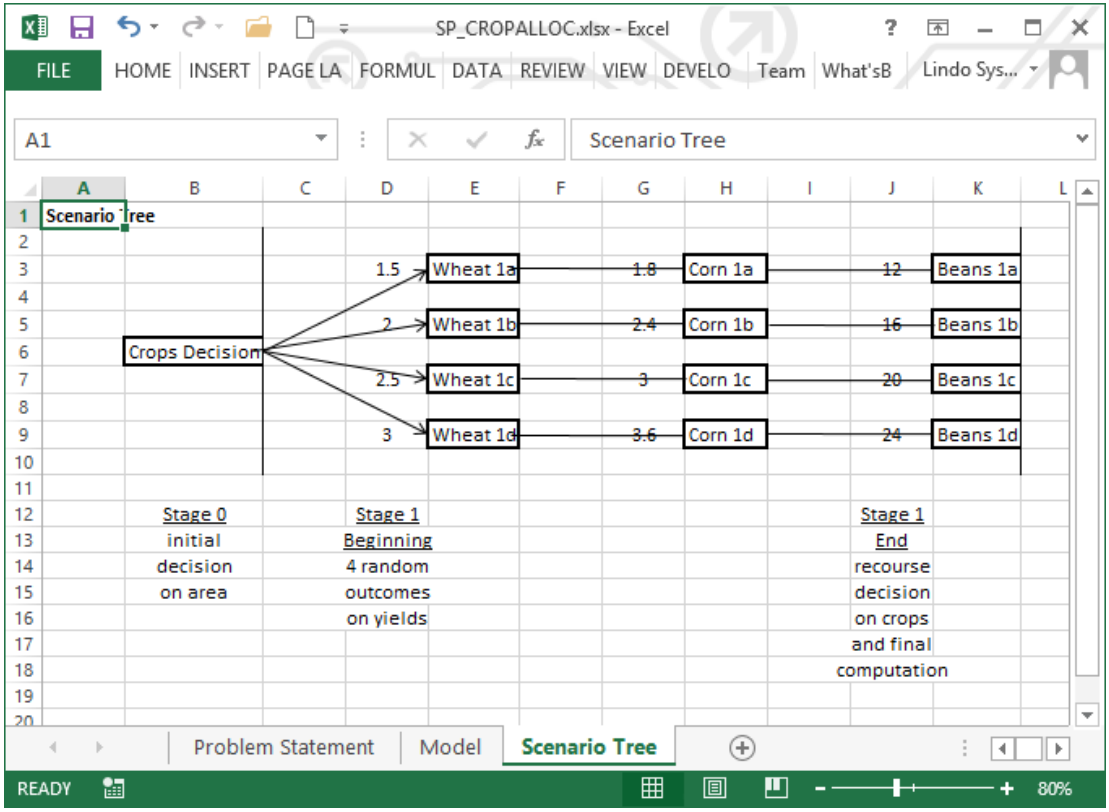
Use the function WBSP_REP, and eventually the WBSP_HIST, to specify the cells that you wish to report. This Reporting function takes in one parameter that is the address of the cell that we wish to report in the final solution. The Histogram function will need the number of bins to display the appropriate table, which can be calculated automatically.

Here we are reporting the area allocated to each crop and the yields for the various crops, and the histogram for the objective cell.

Scenario Tree:

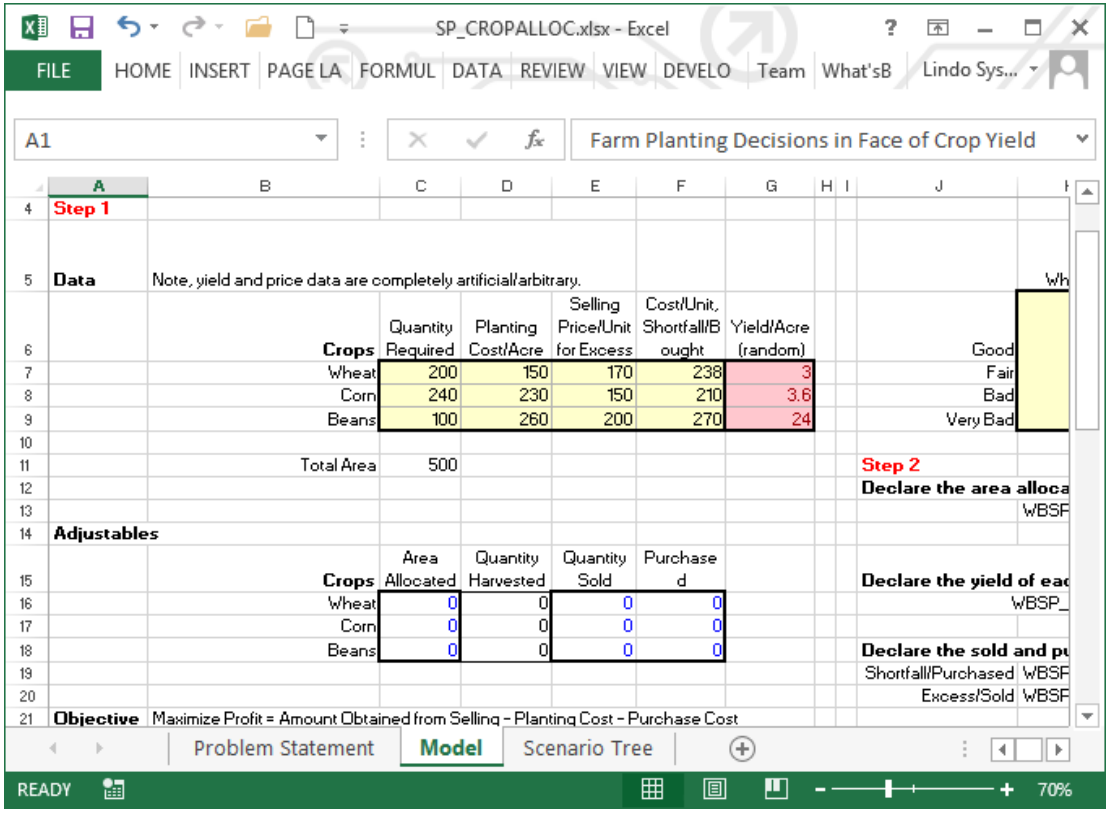
The Scenario Tree illustrates the concept of modeling under uncertainty.

- In Stage 0, the allocated area has to be decided for each crop to grow.
  In Stage 1, the crop yield is revealed depending on the weather, and as a recourse decision, the
- quantity to produce, sell, and purchase, have to be decided.
- The objective is to maximize the total expected profit at the end of planning horizon (Stage 1).

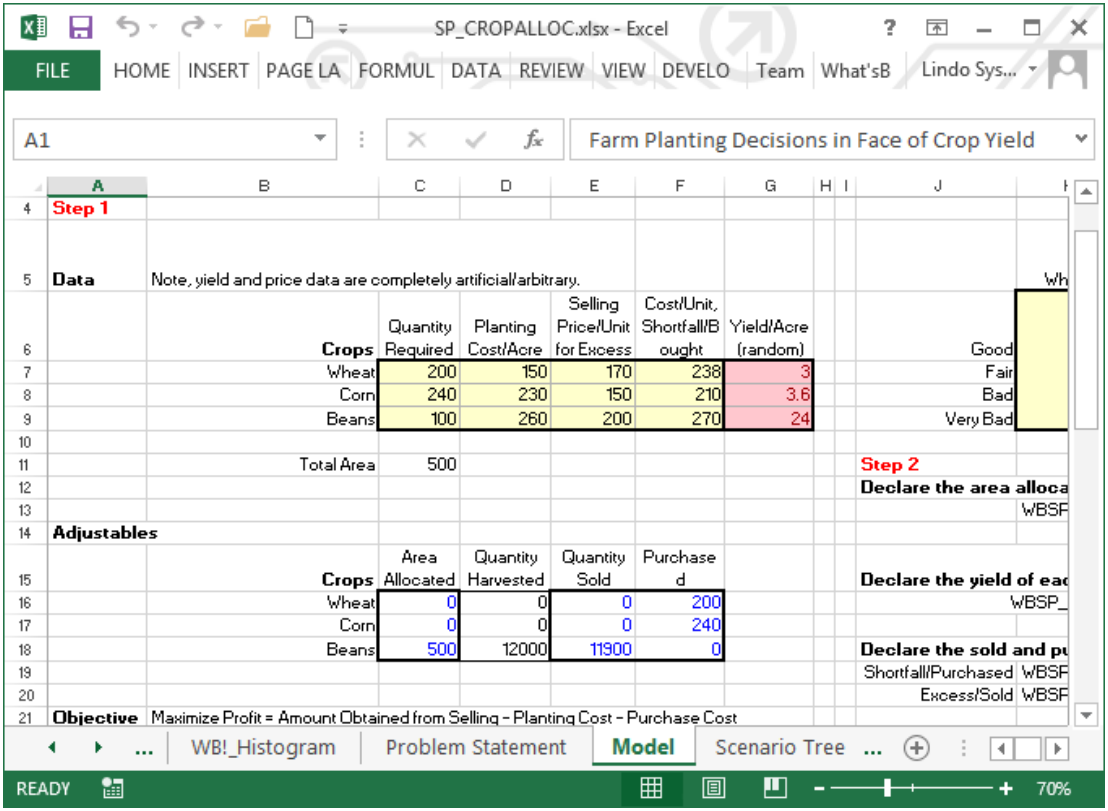

SP_CROPALLOC.xlsx - Excel
FILE
HOME
INSERT
PAGE LA
FORMUL
DATA
REVIEW
VIEW
DEVELO
Team
What'sB
Lindo Sys...
A1
Scenario Tree
Scenario Tree
Crops Decision
1.5
2
2.5
3
Wheat 1a
Wheat 1b
Wheat 1c
Wheat 1d
1.8
2.4
3
3.6
Corn 1a
Corn 1b
Corn 1c
Corn 1d
12
16
20
24
Beans 1a
Beans 1b
Beans 1c
Beans 1d
Stage 0
initial
decision
on area
Stage 1
Beginning
4 random
outcomes
on yields
Stage 1
End
recourse
decision
on crops
and final
computation
Problem Statement
Model
Scenario Tree
READY
80%

Optimization

The SP_CROPALLOC Worksheet Before Optimization:



The SP_CROPALLOC Worksheet After Optimization:



The solver writes a series of solution in the created tab *'WB!_Stochastic'* with the expected value, and displays the first scenario on the spreadsheet.

Put Option

File name: SP_PUTOPTION.XLSX *TYPE: NONLINEAR OPTIMIZATION - STOCHASTIC*

Application Profile

The holder of an American put option has the right to sell a specified stock at any time (the American feature) between now and a specified expiration date at a specified strike price. The holder makes a profit in the period of exercise if the strike price exceeds the market price of the stock at the time of sale. Wealth is invested at the risk free rate, and the stock return for a specific period is the uncertain parameter.

This is a multi-period stochastic problem under the uncertainty of a stock return.

The Problem in Words

A trader of an American style option has the right to sell a stock.

There is an initial price of the stock of $100, a strike price of $99, a risk free rate of 4%. The trader can sell this stock over a 5 time period.

Given all prices, at the beginning of stage n, the market makes a random outcome. At the end of stage n, having seen all of market's n previous outcomes, as well as all the previous decisions, the trader makes the next decision.

The trader has to decide when to decide to sell it once so to generate a profit.

The Worksheet

The worksheet has two sections:

1) A section where we enter the data, the adjustables, the randoms, the constraint, and the objective function for the problem. This is similar to the section for solving a deterministic optimization problem.

2) A section where we enter the stochastic information. Information can be entered directly onto the spreadsheet or via the set of dialog boxes.

Step 1 (core model)

1 Data and Formulas

Specify the initial price ($100), the strike price ($99), and the risk free rate (5%).

The price stock of this period = price of the previous period * (1 + the stock return of this period). This is written in the cells C18 to C23.

The wealth of this period = the wealth of the previous period * (1 + the risk free rate) + the decision to sell * (strike price - price stock of this period). This is written in cells E18 to E23.

2 Adjustable Cell

There are 6 adjustable cells in cells E18 to E23, one for each period. The meaning is 1 for selling, 0 for holding.

3 Objective Function

The objective function is to maximize the wealth at the end of period 5, cell E28.

4 Constraint Cell

There is one constraints in D26 of the model. The total number of selling decisions has to be one.

5 Random Cell

There are 5 random parameters, in cells B19 to B23, one at each period, expressing the uncertainty of the stock return.

Step 2 (stochastic information for stage):

Decision Variables

Specify the stage information for the decision to sell or to hold, which is a decision variable (adjustable). This is done by using the function WBSP_VAR. This function requires two parameters:

i) stage of the variable
ii) cell address of the variable

In this problem, the decision in cell D18 is a recourse variable that has to be decided at stage 0:
=WBSP_VAR(0,D18)

Then the other adjustables belong to their respective period. For instance, D23 belongs to the stage 5.

Random Parameters

Specify the stage information for the stock return. This is done by using the function WBSP_RAND. This function requires two parameters:

i) stage of the variable
ii) cell address of the parameter

In this problem, the return in cell B19 occurs randomly at stage 1:
=WBSP_RAND(1,B19)

Then, the random cell in B23 occurs at stage 5.

Other Variables

The What's*Best!* interface will automatically assign the objective function and all the constraints at stage 1, which depend on the stage value of the previous variables. The user can also assign these variables to a stage 1, using the WBSP_VAR function.

For instance, the following variables are assigned a stage 1, because they are dependent of the random parameter which is at stage 1.

a) Price of stock this period (cell C19)
b) Wealth this period (cell E19)

It would have been possible to define these variables for taking a value only at stage 1:
=WBSP_VAR(1,C19,D19,E19)

Also, the constraint in D26 belongs to stage 5.

Step 3 (distribution information for the random parameters):

Specify a discrete distribution for the stock return parameter. This is done by using the function WBSP_DIST_DISCRETE_SV. This function takes in two parameters:

i) the set of possible discrete values for the scenarios (set vertically)

ii) the cell address of the random parameter

The random demand in cell B19 will take the values entered in the range S19:S22, with equal probabilities:

=WBSP_DIST_DISCRETE_SV($S$19:$S$22,B19)

Step 4 (scenario information):

Specify the number of scenarios per stage by using the function WBSP_STSC. This function requires a two-column table as an argument:

1) Column 1 for the number of stages in ascending order

2) Column 2 for the respective number of scenarios

In this problem, there are 5 stages, with 2 scenarios each, set in the range J28:K32:

=WBSP_STSC(J28:K32)

It happens the discrete table for the distribution has only 4 possible values, so the random parameters will select 2 among the 4 outcomes with replacement. The option "Sampling on Continuous Only" needs to be turned off to take advantage of the sampling feature.

Step 5 (reporting cells):

Use the function WBSP_REP to specify the cells that should appear in the Stochastic report. This function takes a series of arguments that are the address of the cells that should be reported in the final solution:

=WBSP_REP(C19:C23,D18:D23,Wlth0,Wlth1,Wlth2,Wlth3,Wlth4,Wlth5)

Here the stochastic report will display the outcomes of scenarios for all the random cells, the price of stocks for any periods, the decision to sell, and the ending wealth to maximize.

Use the function WBSP_HIST to generate a histogram of the Present Value, in a 8-bin graph:

=WBSP_HIST(8,E28)

Scenario Tree

The Scenario Tree illustrates the concept of modeling under uncertainty.

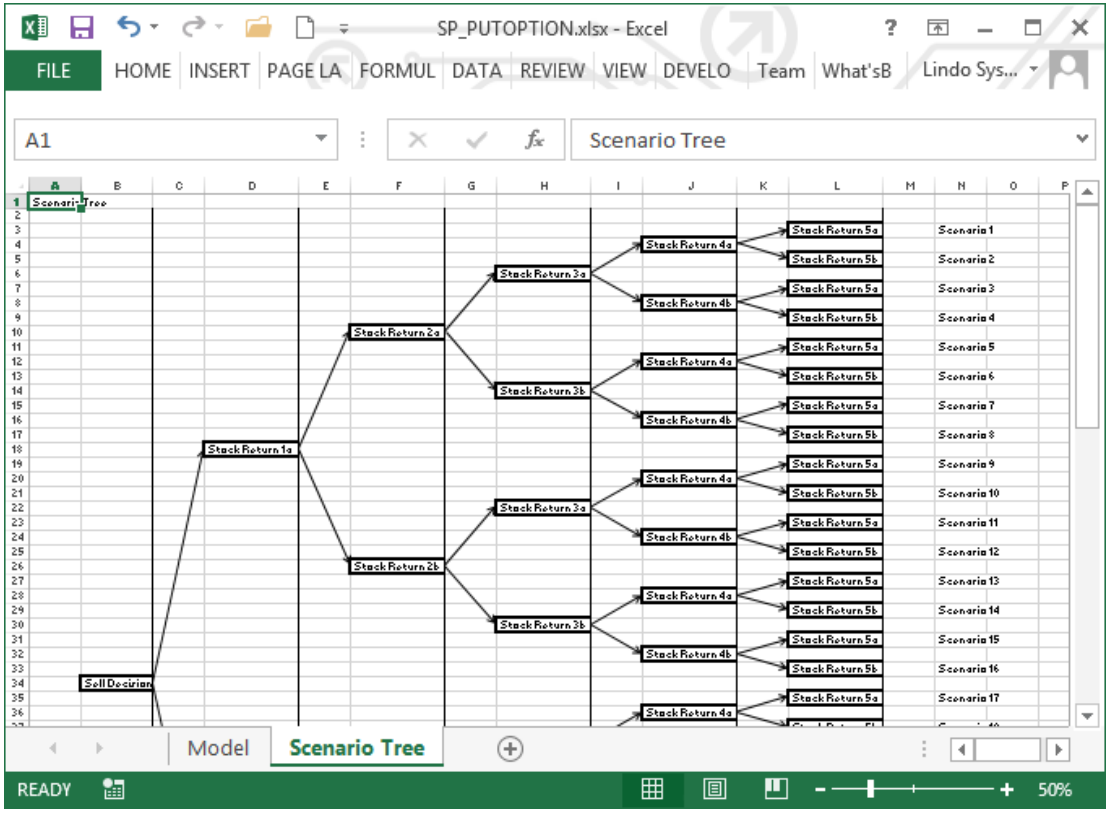
In Stage 0, an initial decision has to be made for selling or keeping the stock.

In Stage 1, the stock return is revealed depending on the market wealth, and as a recourse decision, the sell or keep action has to be decided.

From Stage 2 to 4, the same sequence is applied.

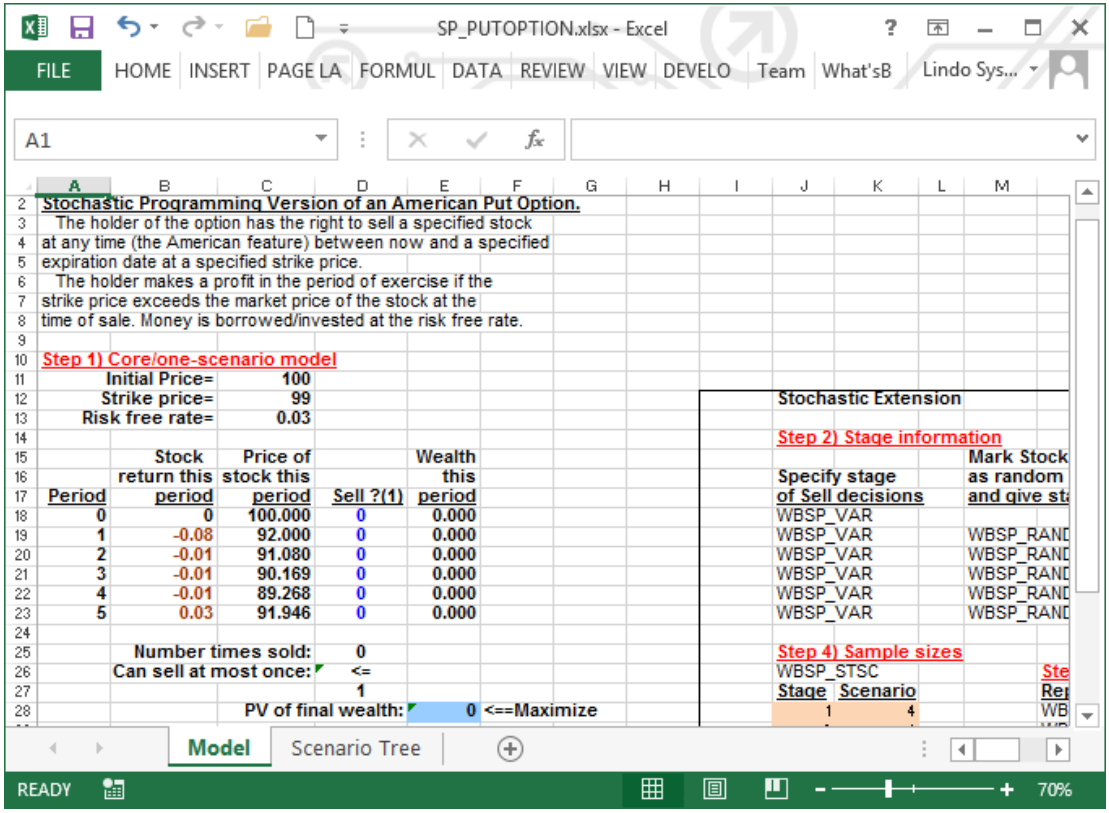
In Stage 5, the stock return is revealed, a recourse decision is made, as well as the final computation.

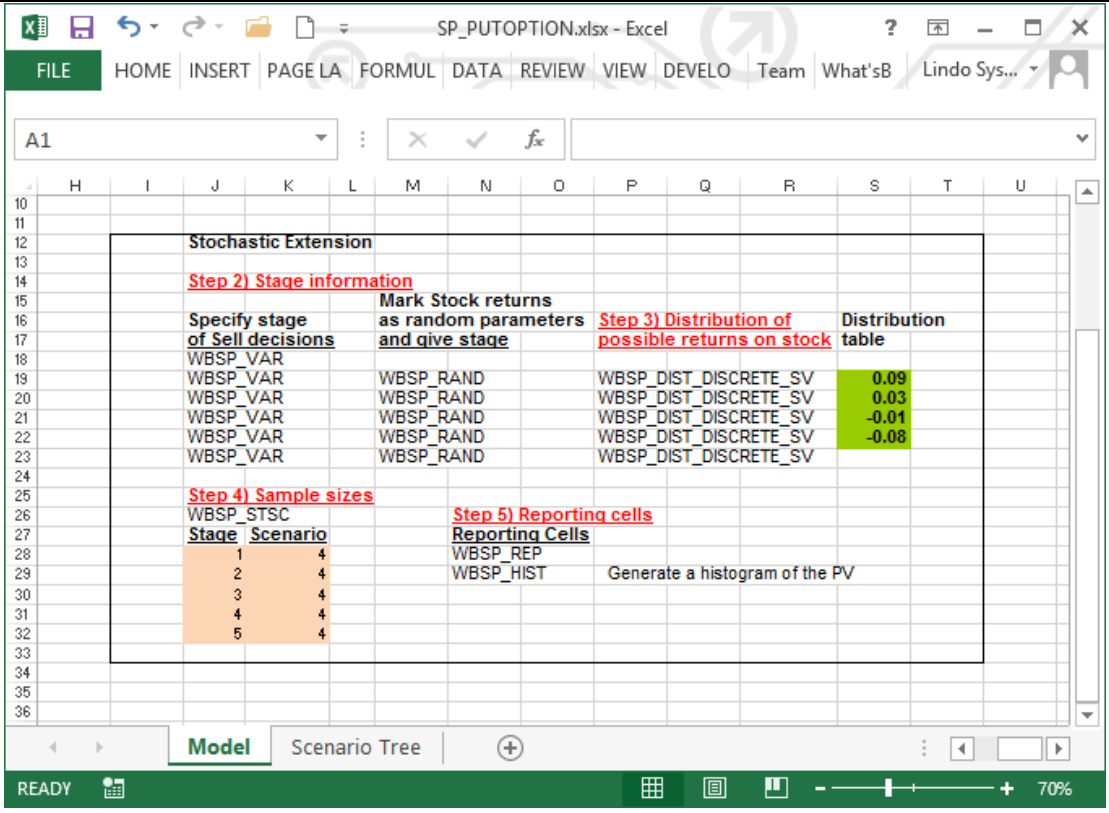
The objective is to maximize the total expected profit at the end of planning horizon (Stage 5).



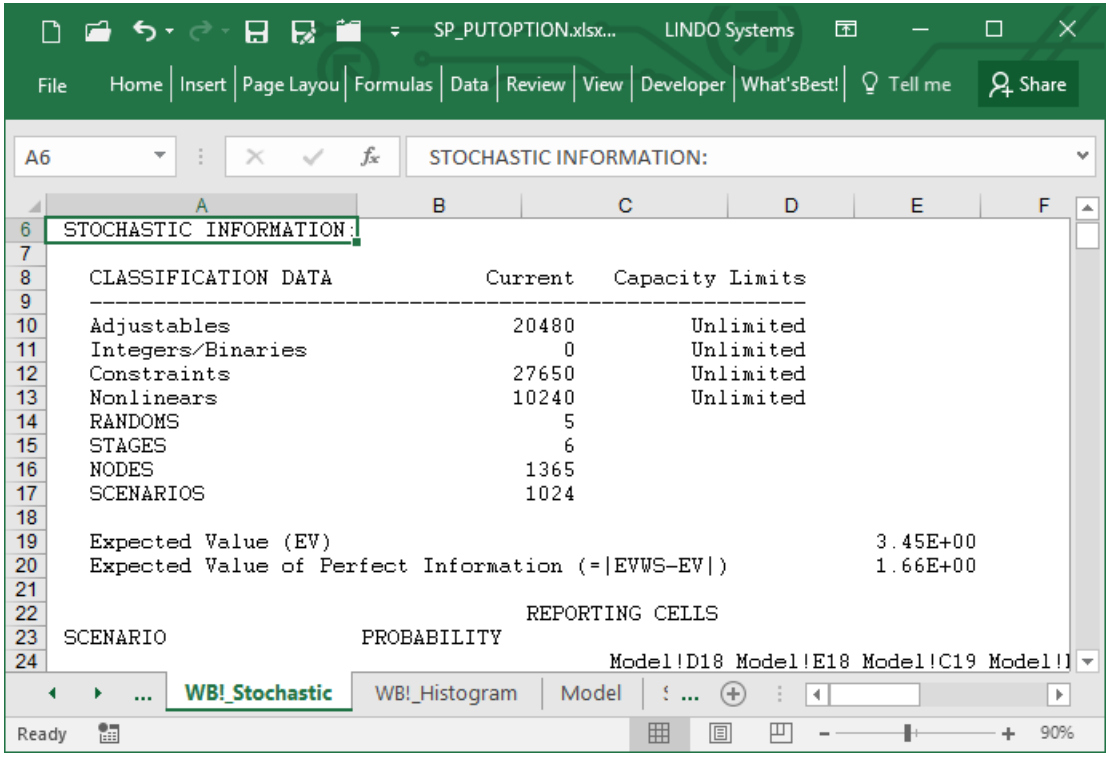
Optimization

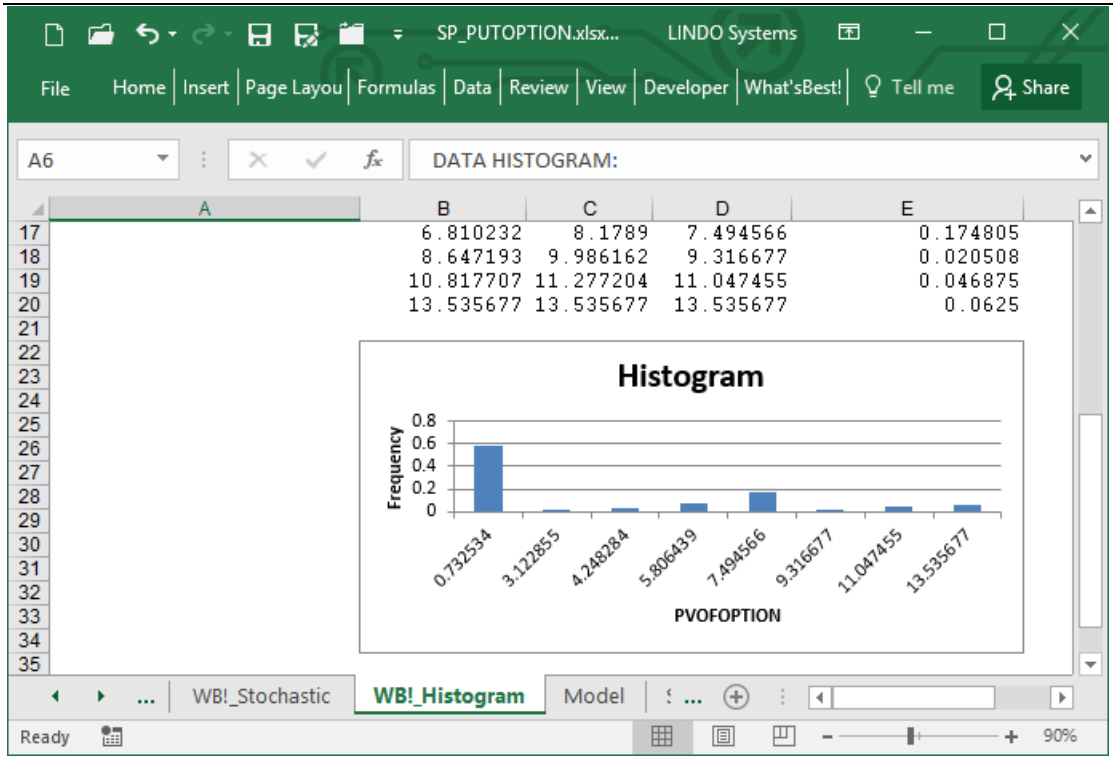
The SP_PUTOPTION Worksheet Before Optimization




SP_PUTOPTION.xlsx - Excel
FILE
HOME
INSERT
PAGE LA
FORMUL
DATA
REVIEW
VIEW
DEVELO
Team
What'sB
Lindo Sys...
A1
Stochastic Extension
Step 2) Stage information
Specify stage of Sell decisions
Mark Stock returns as random parameters and give stage
Step 3) Distribution of possible returns on stock
Distribution table
WBSP_VAR
WBSP_RAND
WBSP_DIST_DISCRETE_SV
0.09
0.03
-0.01
-0.08
Step 4) Sample sizes
WBSP_STSC
Stage
Scenario
Step 5) Reporting cells
Reporting Cells
WBSP_REP
WBSP_HIST
Generate a histogram of the PV
Model
Scenario Tree
READY
70%

The SP_PUTOPTION Worksheet After Optimization





The solver writes a series of solution in the created tab '*WB!_Stochastic*' with the expected value, and displays the first scenario on the spreadsheet.

8 Troubleshooting

Troubleshooting is divided into two major sections, *Frequently Asked Questions* and *Error Messages & Warnings*.
The *Frequently Asked Questions and General Operational Problems* section lists general problems and symptoms that may occur due to some operational problem, especially in loading and running What's*Best!* within Excel®. This can be reviewed as a Technical Support.
Error Messages & Warnings is a list of messages that may be encountered in building and solving a model. Most of these messages are returned in the WB! Status report.
The Appendix section gives additional information on the installation process.
If you encounter an error message that is not discussed here, please contact LINDO Systems.

FAQ Frequently Asked Questions

Frequent Questions and General Operating Problems.

Content

♦ The solution returned from What's*Best!* is not optimal.

♦ An Excel® error appears claiming that a workbook cannot be opened under High Security Level.

♦ An Excel® error appears referring to "multiple copies of WBA.XLA, WBA.XLAM".

♦ When I try to close Excel®, an error appears stating, "This workbook is currently referenced by another workbook and cannot be closed".

♦ The What's*Best!* menu does not load.

♦ There are Excel® error codes of #REF! in What's*Best!* cells.

♦ What are the system requirements to install What's*Best!*?

♦ How do I install What's*Best!* add-in on my computer?

♦ Where are the What's*Best!* add-in files installed on my computer?

♦ What can I do if I receive an error message?

♦ Can I protect my workbook from viewing?

♦ How large can my model be?

♦ How do I fix the "Error Opening File" error message?

♦ How do I fix the "Error in Auto_add: #=5: Invalid procedure or call argument" error message?

♦ How do I fix the "Error in returning solution in cell…" error message?

♦ When using Function Support, the system seems to hang with the message "Server Busy".

♦ What Excel® file format should be used?

♦ What 32-bit or 64-bit format should be installed?

♦ Error in Managing Temporary Files/TEMPFILE?

♦ Error Starting Solving.

♦ After installing, the "About What's*Best!*" shows all 0 in the Library version.

♦ Menu and Toolbar/Ribbon not functioning.

♦ Menu and Toolbar/Ribbon not showing after installation.

♦ There are two Toolbars.

♦ Some Update functionalities on the Menu, and Toolbar do not work.

♦ What's*Best!* can solve a .XLS file format, but not a .XLSX.

♦ Error 1605 when starting Installation.

♦ Error 1618 when starting Installation.

♦ Error Setting Initialization File.

♦ Automation Error.

♦ It takes very long to set cells to "Adjustable".

♦ User's account privileges.

* * * * * * * *

The solution returned from What'sBest! is not optimal.

Problem: There is a solution that yields a better value in the best cell and satisfies all constraints. The model may be incorrectly formulated or, with a nonlinear model, the returned solution may be a local optimum rather than a global optimum.

Suggestions: The following actions may help you in checking your formulation and/or improving the solution returned by What's*Best!*:

♦ Use the Warnings feature to enable all warning messages (set via Options...|General) and re-solve the model. Check the returned status report for any indication of a problem. A warning you consider trivial may be a sign of a more serious underlying problem.

♦ Look at all of the constraints in the model to ensure that each was formulated properly, all cell references are correct, and their signs (<=, >=, and =) are correct.

♦ If you are using the Omit feature, carefully check the contents of each omitted range. For information on the Omit feature, see the section entitled Advanced…|Omit.

♦ If you know of a better solution than the one returned by What's*Best!*, input the new adjustable cell values and check that all the constraints are satisfied and the best cell has improved.

♦ If the model contains nonlinear relationships (check the model statistics in the status report), consider re-solving the model with different starting values for the adjustable cells. Also, check the scaling and consider tightening the bounds on your adjustable cells. The section entitled Guidelines for Modeling with What's*Best!* in Overview of Mathematical Modeling may be helpful in building better nonlinear models.

* * * * * * * *

An Excel® error appears claiming that a workbook cannot be opened under High Security Level.

Problem: This error occurs when Excel® tries to open the What's*Best!* add-in. What's*Best!* makes use of Excel® macros to implement the =WB() constraint function.

Suggestions: The safest option is to put Excel® into Medium security mode. To do this, run the Tools|Macro|Security command in Excel® and set the Security Level to Medium. The problem with this approach is that you will be prompted by Excel® each time it loads as to whether or not you wish to allow the What's*Best!* add-in to be loaded. An alternative is to run the Tools|Macro|Security command, select the Trusted Sources tab, then check the Trust all installed add-ins and templates button.

* * * * * * * *

An Excel® error appears referring to "multiple copies of WBA.XLA, WBA.XLAM".

Problem: You have a toolbar left from a previous version of What's*Best!*, which must be deleted before you reinstall What's*Best!*.

Suggestions: First, uninstall What's*Best!* from your machine with the Uninstall program. You can search for the word "uninstall" to find a shortcut entitled "uninstall What's*Best!*", and double-click it. Otherwise, you can manually remove What's*Best!* related files from your machine. The location of the What's*Best!* files is indicated at the bottom of the About What's*Best!* dialog box. Then, to delete the old toolbar, use the View|Toolbars|Customize… (Excel® 2002) command to bring up a list of the available toolbars, or right click on the ribbon to select Delete (Excel® 2007 and later). Scroll down to the What's*Best!* toolbar, which may or may not be checked off, and click on (highlight) the words "What's*Best!*". Then, click the Delete button to remove the old toolbar. Now, install What's*Best!* and the new toolbar will be installed without conflict with the old toolbar.

* * * * * * * *

When I try to close Excel®, an error appears stating, "This workbook is currently referenced by another workbook and cannot be closed".

Problem: You have set a reference to WBA.XLA or WBA.XLAM in the VBA Editor that must be removed before the What's*Best!* add-in can be removed and Excel® can close.

Suggestions: From the Visual Basic® Editor (Tools|Macro|Visual Basic Editor) go to Tools|References to open the list of references. Unselect WBA.XLA or WBA.XLAM and you should be able to close Excel®.

* * * * * * * *

The What'sBest! menu does not load.

Problem: The add-in is not set to load or there is a conflict such that it is unable to load.

Suggestions: In Excel®, to set the What's*Best!* add-in to load, choose Tools|Add-ins from the Excel® menu and select What's*Best!* from the list of add-ins. If What's*Best!* is not in the list, press the Browse button to locate the What's*Best!* add-in file, WBA.XLA or WBA.XLAM, so it can be loaded. The default location for it is the Library subdirectory below the Excel® directory. If it is not there, it will probably be found in the WB subdirectory.

If you have a laptop machine and are running Excel®, it is possible that the files required to support add-ins have not been installed. If the WB! menu item does not appear on the menu after trying to set

the What's*Best!* add-in to load, run the installation program for Excel® and choose the Typical installation option when prompted.

* * * * * * * *

There are Excel® error codes of #REF! in What'sBest! cells.

Problem: The What's*Best!* add-in provides special functions to Excel®, giving it the ability to express constraints and dual cells. As it does with all add-in functions, Excel® internally stores the location of the add-in program files whenever you use one of the What's*Best!* add-in functions in a model. As a result, if you create a model with What's*Best!* add-in functions and later open it using a copy of Excel® that has the What's*Best!* program files installed in a different location, then Excel® will not properly handle the What's*Best!* functions, and displays the #REF! error code**.**

Suggestions: Close the workbook and then reopen it using the following procedure. When Excel® opens the workbook and detects formulas with incorrect paths to your What's*Best!* add-in, it will present you with a message about automatic links and then ask: "Do you want to update all linked information?". Answer "Yes" and use the Browse button to find the WBA.XLA or WBA.XLAM file in your LIBRARY subdirectory of your main Excel® directory. If this does not remove the #REF! error, then use the What's*Best!* Update Links button on the General Options dialog box. This should correctly update the path. If you fail to update the links with the What's*Best!* Update Links button, then Excel® will place the error code of #REF! into the cells with the incorrect path to WBA.XLA, or WBA.XLAM.

If you frequently exchange models with other What's*Best!* users that have their program files in a different location, or you have updated or re-installed the What's*Best!* program files in a different location, please refer to the discussion of the Update Links button under Options...|General.

* * * * * * * *

What are the system requirements to install What'sBest!?

To install and run What's*Best!*, check that you have the following:

Software

♦ Microsoft® Windows® 10 or previous Windows® XP, Vista®, 7, 8

♦ Microsoft® Excel® 32-bit 2019 / Office365®, and previous versions 2002, 2003, 2007, 2010, 2013, 2016

♦ or Microsoft® Excel® 64-bit versions 2019 / Office365®, or older 2010, 2013, 2016 with latest Service Packs

♦ Microsoft® Excel®, 'PC' version

♦ Microsoft® .NET Framework 4.0

Hardware

♦ Pentium-class PC

♦ 500 MB of RAM

♦ 50 MB of free disk space

Make sure you have administrative privileges to install files on your default drive, System, and Program Files folders.

An Internet connection is required to download the latest version of What's*Best!*. You can also contact LINDO Systems to obtain a copy. Additional information can be found via the Help command on the What's*Best!* menu.

* * * * * * * *

How do I install What'sBest! add-in on my computer?

If you are installing What's*Best!* from the original CD, open the What's*Best!* folder click on 'setup.exe'. If you downloaded the demonstration version from the website, simply run the executable or unzip the zip file and then run the executable. Excel® should be closed during the install process.

An installation program will then commence and will verify if What's*Best!* has already been installed. If it has, you will be asked you if you would like to remove the previous version or to write-over it.

In the next step, you will be presented with the standard license agreement for What's*Best!*, which you must agree to. What's*Best!* will then confirm that your system requirements are adequate. If so, you will then choose a destination directory for the What's*Best!* sample files.

Next, you will be given a choice between a Default and a Specified setup for the add-in files. The Default setup is recommended for English language versions of Windows®. This will install the What's*Best!* add-in files in Excel®'s Library folder. On the other hand, should you choose the Specified setup, the add-in files will be installed in a directory that you specify. The Specified option is recommended for non-English versions of Windows®, or network installations.

At this point, What's*Best!* has enough information to begin copying files. Once the files are done copying, a Finish button should appear. After clicking Finish, Excel® will open. You may receive a message that 'wbintr.xls contains macros'. You must then select the Enable Macros button in order to finish installation.

* * * * * * * *

Where are the What'sBest! add-in files installed in my computer?

The Default installation will automatically transfer the program files to the Library subdirectory of your main Excel® directory, usually "C:\Program Files\Microsoft Office\Office10\Library\LindoWB" for Excel® 2002, or in "C:\Program Files\Microsoft Office 16\root\Office16\Library\LindoWB" for Excel® 2016. You should find the following files:

CONOPTx.DLL
LIBIOMP5MD.DLL
LINDOx_0.DLL
LINDOCU_xx.DLL
LINDOPRx.DLL
LINDOWBEF.DLL
LINDOWBIL.DLL
LNDWBxxx.LIC
MOSEKx_0.DLL
README.WRI
VCREDIST_x86.EXE
WBA.XLA (Excel® 2003) or WBA.XLAM (Excel® 2007 and later)
WBINTR.XLS
WBOPT.DLL
WBOPTLINK.EXE
WBUNCHADD.EXE

The Help file will be stored in the same directory:

LINDOWBHELP.CHM

Finally, sample workbooks will be installed in C:\WB.

The Specified installation differs from the Default installation in that it lets you choose the specific location for the What's*Best!* add-in files.

The 64-bit files are identified with the '64' inside the file name.

The Appendix section of this manual gives additional information on the location of the Add-in files.

* * * * * * * *

What can I do if I receive an error message?

What's*Best!* displays two kinds of error messages: those generated by Excel® and those generated by What's*Best!* itself.

Errors from Excel® (e.g., "Illegal Operation" or "Runtime Error") typically result from operational problems in the loading and running of What's*Best!*. You should check that your model is pointing to the right What's*Best!* add-in file and that there is no loss of links. To do this, check the Tools|Add-ins list and Browse to find What's*Best!*. If you are developing macros with the Visual Basic® Editor and using What's*Best!* functions, verify that the Tools|References command is referring to the correct WBA.XLA file. In Excel® 2007 and later, go to the OfficeButton|ExcelOptions|Add-Ins|Go, to browse to the correct location.

What's*Best!* generated errors (e.g., "Error building the model") are typically encountered when building and solving a model. Check to see that all adjustable, best, and constraint cells are correctly specified. Also, if you have moved your model from one computer to another, try running the General…|Options… command and clicking the Update Links button. You may also not nest any What's*Best!* functions in a single expression.

* * * * * * * *

Can I protect my workbook from viewing?

You can password protect worksheets, data, and macros from viewing. However, you may not protect any of the adjustable, dual or range cells. What's*Best!* will still be able to solve the model, but it won't be able to access these protected cells in order to write the results.

* * * * * * * *

How large can my model be?

The size model you can solve will depend primarily on the size license you purchased. For a list of the various sizes and their specific limitations see section About What's*Best!*.

Other indirect limitations are memory and time. Your model may physically fit within the limits of your version of What's*Best!*, but, particularly for very large models, there may not be enough random access memory available to successfully solve your model. Also, certain classes of models are very difficult to solve. A tough model may take more time to solve than you are willing to wait. For more information on what makes a model tough and techniques for making models easier to solve, refer to the section Overview of Mathematical Modeling.

* * * * * * * *

How do I fix the "Error Opening File" error message?

This error message results from temporary files that couldn't be closed from the last run of the What's*Best!* Solver.

Go to the folder where your model file is located and remove any What's*Best!* temporary files. These files are named:

LINDOWBS.XLS

LINDOWBX.XLS
LINDOWBS.XLSB
LINDOWBX

LINDOWBRC.TXT

LINDOWBSOLN.TXT
LINDOWBSTATUS.PRN
LINDOWBSOLN.PRN
LINDOWBSTOC.PRN
LINDOWBSTOH.PRN
LINDOWBKBST.PRN

* * * * * * * *

How do I fix the "Error in Auto_add: #=5: Invalid procedure or call argument" error message?

This error will appear when an Excel® function or VBA statement cannot be executed properly. In the case of What's*Best!*, this is typically due to missing program components. Attempting to attach the What's*Best!* add-ins over a network can lead to this problem in that not all of the What's*Best!* add-in files will get copied to the local machine. For this reason, we recommend installing What's*Best!* on every platform you intend to use it on. Running a complete install on each machine will guarantee that all the correct add-in files are copied to the correct folder.

* * * * * * * *

How do I fix the "Error in returning solution in cell…" error message?

Usually, this error appears when What's*Best!* could not access the worksheet in order to copy the solution back to the adjustable cells, or the temporary solution file could not be read. First, verify that your worksheet is not protected or locked. Also, make sure Excel® is calling the right add-in in the Library folder.

* * * * * * * *

When using Function Support, the system seems to hang with the message "Server Busy".

Usually, this Windows® message appears when a Microsoft® Office® component could not access the add-in in order to read it and to execute it.

If your add-in has been digitally signed, you will need to enable this add-in to run your model, then retry the call. In some situations, a time delay may occur after enabling the add-in. Also, make sure Excel® is calling the add-in in the right folder.

In Excel® 2002, there is an additional security setting that the user needs to set for accessing and executing a macro from the Solver. Select 'Trust Access to Visual Basic Project' via the 'Tools|Options|Security|MacroSecurity|TrustedSources' menu. Then save the model and restart Excel®.

In Excel® 2007 and later, this feature can be set via the Office® File Button|ExcelOptions|TrustCenter|TrustCenterSettings.

* * * * * * * *

What Excel® file format should be used?

In Excel® 97-2003, the file format is ".XLS" with a maximum size of 256 columns and 65536 rows per sheet.

Staring Excel® version 2007 and later, the new format is either ".XLSX", ".XLSB" for workbook, or ".XLSM" for macro-enabled workbook, with a maximum size of 16384 columns and 1048576 rows per sheet. You can save your file with any of these formats.

* * * * * * * *

What 32-bit or 64-bit format should be installed?

What's*Best!* depends on the bit format of Excel®, not Windows®. The bit format is clearly indicated via the File-Help menu of Excel®. For instance, you may have installed a 32-bit format of Excel® on a 64-bit version of Windows®. In this case, you should install a 32-bit What's*Best!*.

What's*Best!* 64-bit should be installed with Excel® 64-bit, on a 64-bit operating system. Otherwise, install the 32-bit version of What's*Best!*.

The Installation topic in the Appendix section shows additional settings to verify in Excel® for running an add-in.

* * * * * * * *

Error in Managing Temporary Files/TEMPFILE?

There is a topic named TEMPFILE for the "Error Managing Temporary Files" message in this section.

What's*Best!* needs to create temporary files for building data listing, and returning solutions or reports. Such message implies a non access to the folder or its files: network location without mapping a drive, a file format .XLSX without the Microsoft® .NET Framework library in the system, a wrong 32 or 64-bit installation, or missing user's privileges.

The user may just try a sample file in the "C:\WB" folder.

Being able to run a .XLS file but not a a .XLSX means your system is missing the .NET Framework, in which situation you can run a Windows® Update to download it.

Also, you can verify the access rights then moving the model file in this same "C:\WB" location. This simple location will be accessed by What's*Best!*. In some situations, you can map your network location via the Windows® Explorer, then open the file from there.

Finally, another instance of Excel® may still be running as a back task, then your model file handler will still be on hold. This eventually happens if you run a large file via a macro loop in VBA code.

So this message happens when the solver couldn't access some directories of libraries to create the temporary files. This could be due to some missing admin rights, a directory path name with symbols, or a What's*Best!* installation in a wrong Excel® directory.

* * * * * * * *

Error Starting Solving.

The solver was unable to load the model or the calculation library.

- If you have installed the 64-bit version of What's*Best!*®, confirm you are running a 64-bit version of Excel® via the menu File->Account->About.

- If your model file was retrieved from a network drive, try saving the model to a local drive and resolving.

- Try resetting the link to the What's*Best!*® add-in file by selecting the menu File->Options. Then select 'Add-ins' and 'Manage: Excel® Add-ins' and 'Go'. From the Add-ins dialog box, choose 'Browse' to the sub folder ...\Library\LindoWB of the Excel® directory, select WBA.XLAM, and 'OK'.

_ Verify the location of the model; put it in "C:\WB" folder, run a sample from there.

_ Select the "Advanced-Function Support" to run the model via an different internal track.

_ Verify the situation of a 32-bit Excel® on a 64-bit Windows®; in which case a 32-bit What's*Best!* should be installed.

_ Eventually reset the add-in link to "WBA.XLAM" via the Excel® menu, and the VB Editor menu.

A firewall or system scan may block the execution of the .EXE file. The "Advanced-Function Support" will run the model bypassing the .EXE, making the add-in as a .DLL, but with Excel® memory limitation left.

* * * * * * * *

After installing, the "About What'sBest!" shows all 0 in the Library version.

The Verify the Excel® version, not Windows®, for the 32-bit or 64-bit. Then install What's*Best!*, 32 or 64-bit.

What's*Best!* depends on Excel®, 32-bit or 64-bit, not Windows®. The user may have installed a 32-bit Excel® on a 64-bit Windows®.

* * * * * * * *

Menu and Toolbar/Ribbon not functioning.

Make sure to have installed the add-in files in the corresponding Excel® directory, then reset the link via the Excel® Options, Add-ins, and Browse to the WBA file. Usually, the directory is: Excel® and What's*Best!* C:\ProgramFiles\MicrosoftOffice\Office...\Library.

This happens when Excel® cannot activate the add-in, even if it is checked. An add-in installed in another Excel® directory will confuse on which library file to activate, so the functionalities of the add-in will not work properly.

* * * * * * * *

Menu and Toolbar/Ribbon not showing after installation.

- Settings for Microsoft® Excel® 2007-2016

Via the Ribbon, select:

Office® Button

Excel® options

Popular

- Show Developer tab in the Ribbon

Office® Button

Excel® options

Manage Excel® Add-ins Go

- Check "_What's*Best!*"

- or Browse to "WBA.XLAM", usually in "C:\Program Files\Microsoft Office 16\root\Office16\Library\LindoWB"

Office® Button

Excel® options

Trust Center

Trust Center Settings

Macro Settings

- Enable all macros (possibly)

- Trusted Publishers to "LINDO Systems Inc"

- Settings for Microsoft® Excel® 1997-2003

Via the Menubar, select:

Tools

Add-ins

- Check "_What's*Best!*"

- or Browse to "WBA.XLA", usually in "C:\Program Files\Microsoft Office\Office11\Library\LindoWB"

Tools

Customize

Toolbars

- Check "_What's*Best!*"

Tools

Options

Security

Macro Security

- Security Level to medium

- Trusted Publishers to "LINDO Systems Inc"

Depending on the installation of Excel®, What's*Best!* may not be able to catch up the Excel® directory, and so could not activate the add-in link automatically.

* * * * * * * *

There are two Toolbars.

- Settings for Microsoft® Excel® 2007-2016

Via the Ribbon, select:

Right click on the extra toolbar from the previous version.

A pop up label will indicate the release number of the toolbar.

Select Remove from the sub menu.

- Settings for Microsoft® Excel® 1997-2003

Via the Menubar, select:

Tools

Customize

Toolbars

- Select "_What's*Best!*" and click on Delete

A previous installation of What's*Best!* was not successfully removed, so an extra toolbar is still showing up in the screen.

Remove the custom settings file for the toolbar 'Excel15.xlb' or later version. Restart Excel® to generate a new file;

location of 'Excel15.xlb' in "C:\Users<username>\AppData\Roaming\Microsoft\Excel"

* * * * * * * *

Some Update functionalities on the Menu, and Toolbar do not work.

In the Windows® Registry, What's*Best!* creates three keys in the following locations:

"HKEY_CURRENT_USER-Software-Lindo Systems, Inc." for CheckUpdate, Language, and Registration.

What's*Best!* couldn't access the registry to create the keys, so an access error is showing up every time What's*Best!* needs to detect the value of the keys.

* * * * * * * *

What'sBest! can solve a .XLS file format, but not a .XLSX.

A clean installation of Excel® 2007 will solve the issue, by first removing the previous version Excel® '97-2003.

It happens when the system is missing some up to date libraries to process and extract the new Excel® file format .XLSX, .XLSM, or .XLSB. These libraries are related to the .NET environment containing the file structure parsing capabilities.

* * * * * * * *

Error 1605 when starting Installation.

Open the session with full Administrative rights, and run As Administrator from a folder with full access rights.

Missing rights to executes and install a program.

* * * * * * * *

Error 1618 when starting Installation.

Clean up the registry for any remaining components belonging to another What's*Best!*. This situation happens when a previous installation was interrupted before completion, so the Windows® Registry key is still being recorded.

The attached tool should remove any remaining references to What's*Best!* product keys in the registry of Windows®. Download www.lindo.com/downloads/LindoWBClearRegistryKeysexe.zip, and extract the EXE file to run. Then install the newer version of What's*Best!*.

Another What's*Best!* installation still there (or partially) in the system.

* * * * * * * *

Error Setting Initialization File.

Extract all the files from the .ZIP file into a temporary directory, then run the Setup.exe with administrator rights.

The system does not allow running the Setup.exe file directly from the xxx.ZIP file.

* * * * * * * *

Automation Error.

Verify the corresponding Microsoft® .NET version for What's*Best!* is available via the Control Panel and Programs.

Microsoft® .NET are libraries useful for reading .XLSX, .XLSM, .XLSB file formats.

What's*Best!* 10.0 needs Framework .NET 3.5, but later versions need .NET 4.

* * * * * * * *

It takes very long to set cells to "Adjustable".

Check via the What's*Best!* menu "Options-Reset to Default" the checkbox "Remove extraneous cell styles in workbook", and select OK. Then resave the workbook.

The workbook must have a very long list of custom styles, possibly coming from files transferred to different Excel® versions. What's*Best!* has to detect the useful "Adjustable" style when setting cells to "Adjustable".

* * * * * * * *

User's account privileges.

1_ Full Admin rights during installation, Excel® directory, or Specified folder.

2_ Windows® Registry Keys to insert.

3_ Access Read Write during execution, for the model folder.

4_ CheckUpdate needs an Internet access to provide the download message.

Look at installation rights, then execution rights.

Error Messages & Warnings

If What's*Best!* encountered an error during the solution process, then it opens the status report worksheet tab entitled *WB! Status*, as opposed to the worksheet that was open before the run. Based upon the error message returned, you can troubleshoot your model. The default setting for the *General Options* dialog box (*Options…*|*General*) is to produce a status report following each solve command.

Most of the following errors will be found in the status report worksheet after the model is solved.

Errors from VBA Code

Excel® generates these error messages when the *WBA.XLA* file has not been checked off as a reference add-in for your calls to the What's*Best!* VBA interface. To ensure the What's*Best!* procedures are available to your Visual Basic® code, choose *Tools*|*References…* from the Visual Basic® Editor and make sure *WBA.XLA* is checked.

Note: The first step in running What's*Best!* from VBA is to create a reference to What's*Best!*. This reference is made by checking the *WBA.XLA* box under *Tools*|*References* from within the *Visual Basic® Editor* (The *Visual Basic® Editor* is called via *Tools*|*Macros* from the main Excel® menu bar). If you do not create this reference, then any attempt to use the What's*Best!* attributes or procedures will produce the error message *Sub or Function not defined*.

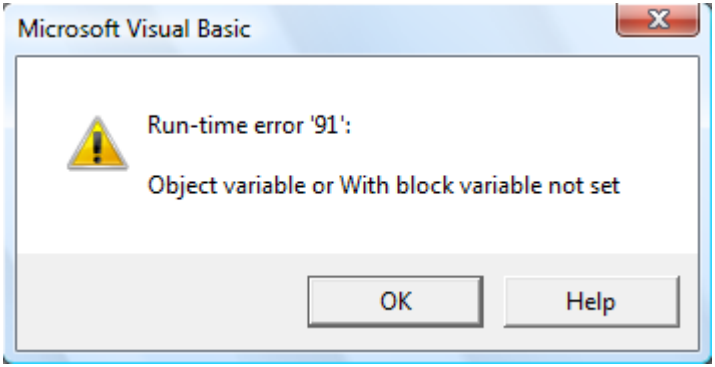
Runtime Errors

If you did not place error handling into your VBA code, then Excel® handles a What's*Best!* error by generating a run-time error message box.

In many instances, this runtime error message provides you with a What's*Best!* error code (a number over 30000). However, in some cases you may only receive a *Run-time error 5: Invalid Procedure Call or Argument* message box like the one shown below. Whether you receive a What's*Best!* error code or an Excel® error code, you must take steps to 1) correct the source of this error; and 2) insert error handling into your code to anticipate any further errors.

Error handling is placed in your VBA code by adding a suitable *On Error* statement before any calls to What's*Best!* routines and inserting some appropriate error handling code. Including error handling in your code will prevent Excel® runtime error messages like that shown above. For an example of such code, see the section entitled *wbError and Error Codes* in chapter *VBA Interface*.

If you don't correct the source of the error, fail to insert error handling, and you again run the routine that produced the error, then Excel® will generate the following, cryptic error message:



This message signifies that Excel® has an unhandled error left over from the previous run and cannot display the new error that was just generated. You must now try to correct the source of the error, but this time you do not have the benefit of a What's*Best!* error code. If you can't manage to guess what the source of the error is, then you must resort to unchecking the What's*Best!* add-in from the list generated by the *Tools|Add-ins* command. This forces the removal of the previous unhandled error. Now, you can return to the add-in list and check the What's*Best!* add-in to reinstate it. You now have another chance to determine the source of the error and insert error handling code.

Errors from an embedded application

In the case of an error running What's*Best!* embedded within a Visual Basic® project, the *Run-time error 5: Invalid Procedure Call or Argument* message box shown above will be displayed. This may be followed by a *Run-time error 440: Automation Error*. To correct this problem, you will need to terminate the application, insert appropriate error handling, and run the application again. The error handling for an embedded application is distinct from normal error handling for a stand-alone Excel® application. A code sample of this error handling is provided toward the end of the section entitled *wbError and Error Codes* in chapter *VBA Interface*.

Invalid Outside Procedure

This is often the result of using parenthesis without the CALL keyword in Visual Basic® code. Either put the keyword CALL in front of the procedure calls where you use parenthesis or remove the parenthesis.

ADDINLINK - Multiple Add-in Links

In the case of multiple installations of *What'sBest!*, the solver may be confused on the source link to the add-in. The following error message will be displayed on the *WB! Status* tab.

Error Message:
ERROR
Multiple Add-in Links (Help reference: ADDINLINK):

The workbook contains multiple links to the What'sBest add-in. Only one source should be defined in the workbook. Make sure to correct the add-in link reference via the menu 'Edit|Links' or 'OfficeButton|Prepare|EditLinksToFiles' and remove the corrupted link. Then update the workbook links via the menu 'WB|Options|General|UpdateLinks', save the file, and reopen it.

Suggestions:
Make sure of the location of the current add-in via the Excel® menu "Tools|Add-ins" or "OfficeButton|ExcelOptions|Add-Ins|Go". You may need to reset the add-in by browsing to the installation location.

Remove any corrupted or wrong add-in link from your workbook via the menu "Edit|Links" or "OfficeButton|Prepare|EditLinksToFiles".

Reset the links on the worksheet to the What's*Best!* functions via the menu "WB!|Options|General|UpdateLinks".

Save the file, close Excel®, and reopen it.

ARITHERR - Arithmetic Error

Many mathematical operations are not defined (e.g., division by zero). *What'sBest!* tries to avoid such errors, but this is not always possible. When an irresolvable numerical error occurs, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Arithmetic Error (Help Reference: ARITHERR)
What'sBest! encountered an undefined arithmetic operation in the cell listed below.
One example of an undefined arithmetic operation would be division by zero, but there are many others. Check the cell below to determine the source of the error.
You must either rewrite the formula or constrain the adjustable cells to avoid the error.
-cell address-

Suggestions:
Examples of undefined arithmetic operations are: division by zero, multiplying by a text string, and evaluating to a number larger than 10^30. If the cells are unrelated to solving the model, include them in a *WBOMIT* range or delete them. Otherwise, change the initial values of the adjustable cells to move away from the undefined region. Also, consider rewriting the model in such a way that all formulas are defined over the entire domain of the adjustable cells.

BLKCELL - Blank Cell Warning

If your model has references to blank cells, the following warning message will be displayed on the *WB! Status* tab.

Warning Message:
****WARNING****
Blank Cell Warning (Help Reference: BLKCELL):
Blank cells have been referenced in formulas. During the solution process, their values have been taken to be zero. This warning can be turned off via the WB|Options|General menu. See cell list below.
(cell addresses listed at bottom of tab)

Suggestions:
You should verify that the blank cells being referenced were actually intended to be blank. If the model looks correct, then you may choose to disable this warning message using the *WB|Options|General* command.

ENTIRERANGE - Entire Range Selection

In the situation where some of the formulas are selecting an entire column or row, the following warning message will be displayed on the *WB! Status* tab.

Warning Message:
****WARNING****
Entire Range Selection (Help Reference: ENTIRERANGE):
A function argument selects an entire column or row in a worksheet, for instance showing A:A or 1:1. This full selection creates several blank cells that may not be necessary to the model, then requesting extensive memory resources for the model, depending on the format of the file as .XLS or .XLSX. A correction would be to select the exact range or cells in the formula.
-cell address-

Suggestions:
What'sBest! will create a 0-value number for any blank cell being referenced by a formula, thus memory allocation for the system. An entire column selected means 1048576 cells, and an entire row means 16384 cells in a .XLSX format. Similarly in a .XLS format, it means 65536 cells for a column, and 255 cells for a row. This warning message will not be displayed if the selection is shorter than these numbers.

EXLVER - Excel® File Format

If your workbook is not saved in a current workbook file format, the following warning message will be displayed on the *WB! Status* tab.

Warning Message:
****WARNING****
Excel® File Format (Help Reference: EXLVER):
The file format you have chosen is from an older release of Excel®. Please resave your file using the latest Excel® file format

Suggestions:
What'sBest! requires workbooks to be saved in Excel® 2002 format, or later. You will need to work with a more current Excel® file format. Resave the file using the *File|Save As* command in Excel®. When presented with the file save dialog box, select a more current file format from the *Save as type* dropdown box.

FORMULA1 - Formula Parsing

If *What'sBest!* was unable to parse a cell formula in your workbook, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Formula Parsing (Help Reference: FORMULA1):
An error occurred while attempting to parse the cell formula listed below.
Check the formula in this cell for potential problems such as complex nested functions or text strings as arguments. Some Excel® functions also require a specific format.
-cell address-

Suggestions:
In rare instances, if a formula becomes too complex What's*Best!* may not be able to successfully parse it. Things to check for are nested functions that are many layers deep, in which case, you may want to break the formula up into multiple cells. Another possibility is that you have used a text string as part of the formula. Finally, some Excel® functions must follow a specific format in order to be parsed by WB.

You may need to contact LINDO Systems for assistance.

FORMULA2 - Unsupported Functions

What'sBest! does not directly support all the functions available in Excel®. Cells in your model can reference Excel® functions that are not supported by *What'sBest!*. However, these cells' value value will be fixed at their calculated values at the start of the solve command. If any unsupported functions references are found in any of the cell formulas, the following warning message will be displayed on the WB! Status tab.

Warning Message:
****WARNING****

Unsupported Functions (Help Reference: FORMULA2):

The cells listed contain spreadsheet functions that are not recognized by What'sBest. The numeric values for these cells are taken from the spreadsheet directly without recalculation. Consider splitting each such cell, say X1, into two cells, one (or more cells), say X2, that contain the unsupported functions but do not depend upon any Adjustable cells, with X1 rewritten so it contains no unsupported functions but may depend upon X2 and Adjustable cells. A WBxxx function cannot be used in a complex expression. It can be used only as the sole element in a cell.

(cell addresses listed at bottom of tab).

Suggestions:
Cells in your model can reference Excel® functions that are not supported by What's*Best!*. However, these cells' values will be fixed at their calculated values at the start of the Solve command. The list of cells referencing unsupported functions may be found on the WB! Status tab at the end of the report. Supported functions and operators are listed in the Supported Functions and Operations section. This fixing by What'sBest! is acceptable in two situations: a) Any cell containing the unsupported function does not depend on an Adjustable cell, or b) If the unsupported cells are used for reporting only, they can be included in a WBOMIT range (see the section entitled: Advanced...|Omit). If condition a) is violated, there are two possible remedies: i) Replace the unsupported functions by equivalent operations supported by What'sBest!, or ii) Split the cell, say B7, containing unsupported functionsinto two cells: one, say B6, that contains the unsupported functions but does not depend on any Adjustable cell, and a second cell, B7, that does not contain any unsupported functions but may depend upon B6 and on Adjustable cells. For example, suppose we have the cell:

B7=INDEX(B3:B5,MATCH(A7,A3:A5,0))*A2

When What'sBest! encounters the MATCH function (which it does not recognize), it stops processing cell B7, generates a warning message, and replaces B7 by whatever is its current value. If any of the cells in the expression for B7 depend upon adjustable cells, then the What'sBest! optimization will probably lead to a non-optimal answer. Suppose, however, that B3:B5, A7, and A3:A5 do not depend upon adjustable cells, but, A2 does. In this case, writing instead:

B6=INDEX(B3:B5,MATCH(A7,A3:A5,0))

B7=B6*A2

will not introduce an error in the What'sBest! formulation because treating B6 as a constant is justified and the expression B6*A2 is understood by What's*Best!*.

Also, the WBxxx functions of What's*Best!* are supported when written as a sole function in a cell, not in an expression.

This warning may be disabled with the Unsupported Function warning checkbox in the General Options dialog box posted by the Options...|General command. Using linked worksheets may also cause this message. For more on this subject, see the Solve topic

FORMULA3 - External Reference

If there are external links to other workbooks that can't be resolved, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
External Reference (Help Reference: FORMULA3):
A link to an external workbook in the cell below could not be resolved. Check the validity of the link or remove the external link.
-cell address-

Suggestions:
An external reference was found that could not be imported. You should check to see if the link is valid and if it can be resolved.

Array formulas can also generate this error in that they may only reference native data contained in the primary workbook. If any of your array formulas have external links, you will need to import the external data into the native workbook.

FUNCADDIN - Failed to Access the Add-in

If the What's*Best!* solver was unable to locate an add-in that is needed to calculate the user's defined functions, the following error message will be displayed on the pop-up window.

Warning Message:
****WARNING****
Failed to Access the Add-in (Help Reference: FUNCADDIN):
The workbook is referring to an add-in that is not at the specified location:
(add-in location)
This is needed to run the user's defined functions from Excel®. Reset any call to this add-in, then save the model and restart Excel®.

Suggestions:
Clear any Excel® call of this add-in via the menu "Tools|add-ins", reset the add-in by browsing to the right location, save the model, and restart Excel®.

In some situations, you may need to remove the add-in from your system to clean up any remaining link in the Excel® Add-ins list.

Also, you can carefully correct the link by accessing the registry editor from the Windows® Start menu, then run "REGEDIT". Usually, the links are displayed in the registry folder "HKEY_CURRENT_USER – Software – Microsoft – Office – 10.0 – Excel – Options".

In Excel® 2007, adjust the security setting from the Excel® Options, then click on "Advanced|TrustCenter|TrustCenterSettings". Enable macros and grant the same access in the "Macro Settings" tab.

In case your add-in has been digitally signed, you will need to enable this add-in to run your model. In some situations, a time delay may occur after enabling the add-in.

Refer to the What's*Best!* sample file *Shipping_MacroFunctions.xls* for an example of valid user defined function usage.

FUNCMACRO - Failed to Access the Macro

If the *What'sBest!* solver is unable to execute a macro that is needed to calculate a user defined function, the following error message will be displayed on the pop-up window.

Warning Message:
****WARNING****
Failed to Access the Macro (Help Reference: FUNCMACRO):
The solver needs to access the VBA code to run the users' defined function from Excel®. Select 'Trust Access to Visual Basic Project' via the 'Tools|Macro|Security|TrustedSources' menu.
Then save the model and restart Excel®.

Suggestions:
In Excel® 2002, there is an additional security setting that the user needs to set for allowing the server to access a user defined macro. Select 'Trust Access to Visual Basic Project' via the 'Tools|Options|Security|MacroSecurity|TrustedSources' menu. Then save the model and restart Excel®.

In Excel® 2007 or later, adjust the security setting from the Excel® Options, then click on "TrustCenter|TrustCenterSettings". Enable macros and grant the same access in the "Macro Settings" tab.

In case your add-in has been digitally signed, you will need to enable this add-in to run your model. In some situations, a time delay may occur after enabling the add-in.

Refer to the What's*Best!* sample file *Shipping_MacroFunctions.xlsm* for an example of valid user defined function usage.

FUNCSERVER - Failed to Access the Server

If the *What'sBest!* solver was unable to access Excel® to calculate the user's defined functions, the following error message will be displayed on the pop-up window.

Warning Message:
****WARNING****
Failed to Access the Server (Help Reference: FUNCSERVER)
The solver needs to access Excel® to run the users' defined functions. Make sure only one Excel® application is running, version 2002 or later, via the Task Manager. Also, verify that the application is not already busy with any macro virus scan.

Suggestions:
The solver could not link into Excel® to execute the functions.

- Make sure to run Excel® version 2002 or later
- Make sure Excel® is not already busy with another process
- Turn off the virus scan for macro applications. In the mean time, select "Retry" from the "Server Busy" window.

HIDSNAME - Hidden Solvers Names

If the *What'sBest!* solver was able to detect hidden names coming from other solvers, and belonging to the workbook, the following message will be displayed on the pop-up window.

Message:
Convert Model Format

What'sBest! has found hidden names in this workbook related to non-What'sBest! model format.

Convert and Solve

Try to Solve without Conversion

Exit

Help

Where:

- 'Convert and Solve' will convert these names into a What'sBest! format for the current selected worksheet, and try to solve. This will create What'sBest! style Adjustable, Best, and Constraint cells in the workbook.

- 'Try to Solve without Conversion' will try to solve as a What'sBest! spreadsheet without conversion.

- 'Exit' to cancel.

- 'Help' to access the manual.

Notes:

- You may remove these hidden names, or after conversion, via the menu:
  What 's*Best!* -> Options -> Reset To Default.

- You can alternatively convert these names into What's*Best!* format via the menu:
  What 's*Best!* -> Advanced -> Convert Model Format.

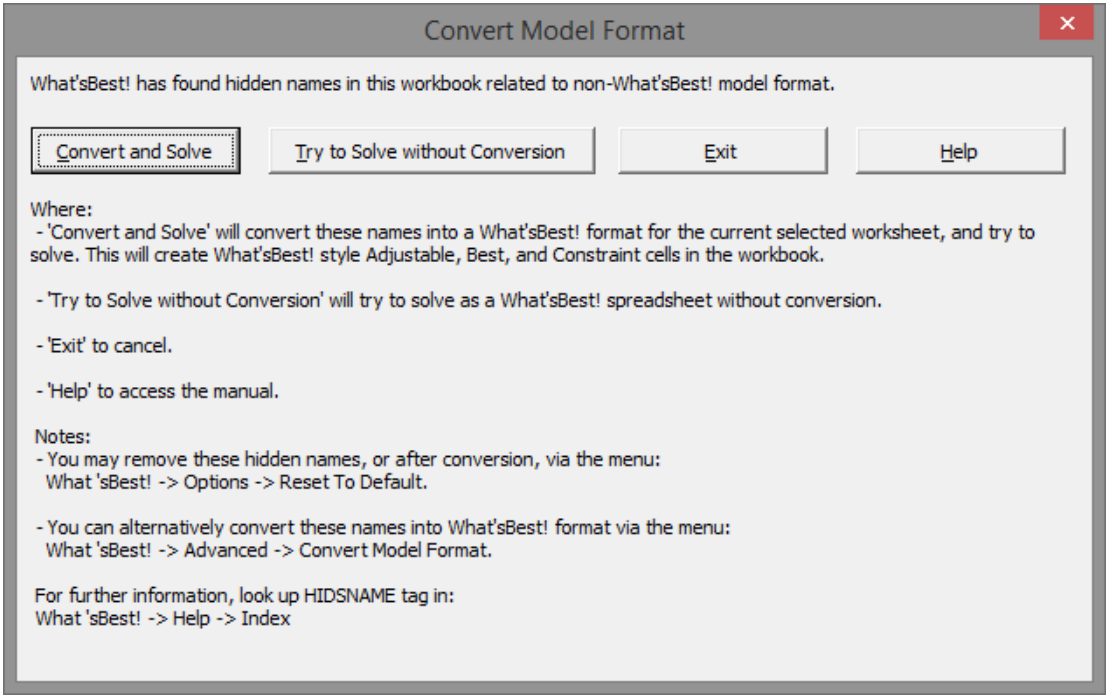
For further information, look up HIDSNAME tag in:
What 's*Best!* -> Help -> Index

Warning Message:
****WARNING****
Hidden Solvers Names (Help Reference: HIDSNAME):
There are hidden names in this workbook related to other model formats for solvers. The interface can convert these names into a What'sBest format, via the menu 'Advanced - Convert Model Format' for the current selected worksheet. This will create any additional Adjustable, Best, or

Constraint cells to review. Otherwise, you may clean up these model names via the menu 'Options - Reset To Default'.

Suggestions:
These hidden names belong to the workbook. They may have been created when the model was being built with other solvers. These names are not visible via the Name Manager, but can be detected. They usually refer to variables, constraints, and objective cells from multiple models in a same workbook. So select the sheet tab from which these names would be converted, or removed.

- You may clean up these hidden model names via the menu “Options – Reset To Default".
- You can convert these names into a *What’sBest!* format, via the menu “Advanced – Convert Model Format”.



IKBMOD - K-Best Model Format

Using the K-Best feature, What’s*Best!* can return the trade-off solution information. Trade-off values tell you how sensitive the successive solutions are to various parts of the model, which can be Mixed-Integer Linear or Nonlinear. If a binary Adjustable cell is not correctly specified, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****

K-Best Model Format (Help Reference: IKBMOD):

A K-Best model is incorrectly formatted. Correct the model by defining at least

one binary Adjustable cell, via the WB|Integers|Integer-Binary menu. Select

'Binary - WBBIN' for the integer type referring to an Adjustable cell.

Suggestions:
This is an incorrectly formatted K-Best model. The model can be Mixed-Integer Linear or Nonlinear, even using the Global Solver, and must contain at least one Adjustable being defined as a binary cell, not general integer cell. For more information on defining a WBBINxxx cell, refer to the section Integers...|Integer-Binary.

IKBREP - K-Best WBIKB_REP Format

Using the K-Best feature, What's*Best!* can return the trade-off solution information. Trade-off values tell you how sensitive the successive solutions are to various parts of the model. The *=WBIKB_REP()* function is used to request trade-off values. If the arguments to this function are not correctly specified, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****

K-Best WBIKB_REP Format (Help Reference: IKBREP):

A WBIKB_REP cell is incorrectly formatted. Correct the formula and the validity of the

arguments in the cell below. The format should be '=WBIKB_REP(cells)', where

'cells' must refer to variable cells.

-cell address-

Suggestions:
There is an incorrectly formatted K-Best Report cell in the model. A reporting cell must use the format: *=WBIKB_REP(cells)*, where *cells* is a reference to the cells you want to report the trade-off value. The cell reference should be to a variable cell. For more information on using K-Best Solution values refer to section *Options...|Integer Solver.*

INDEX - Unsupported Index Usage

What'sBest! supports the INDEX() Excel® function only under certain conditions. If you use an unsupported variant of INDEX function, the following error message will be displayed on the WB! Status tab.

Error Message:
****ERROR****
Unsupported Index Usage (Help Reference: INDEX):
The solver has been halted because the following cell contains an unsupported argument to an INDEX() functions. The second and the third arguments providing the row and column indexes

must be a constant numeric value or a cell with a constant numeric value, but cannot be a variable, an adjustable, or a random parameter.
-cell address-

Suggestions:
The INDEX() function requires two to three arguments as follows: INDEX(range, row index, column index), or just a row or column index if the range shows one column or one row. What'sBest! places the following restrictions on these arguments:

- All arguments must not be a formula,
- the range argument can be an area, or a row selection, or column selection
- the row index and column index arguments must be a constant numeric, or a cell to a constant numeric value, but not a formula or a cell reference
- in case of a single row range, the second argument is the column index
- in case of a single column range, the second argument is the row index

INDICMOD - Indicator Model Cell Reference

If your model has references to WB() constraint cells, without being in a Slack mode, the following warning message will be displayed on the *WB! Status* tab.

Warning Message:

****WARNING****

Indicator Mode Cell Reference (Help Reference: INDICMOD):

The following cells directly refer to WB constraint functions being in Indicator

mode, rather than Slack mode. You need to modify the range selection, or to specify

the Slack mode via the WB|Options|General menu

(cell addresses listed at bottom of tab).

Suggestions:

You should verify that the cells being referenced were actually intended to be in Slack, rather than Indicator mode, checking the option via the *WB!|Options|General* command. Otherwise, you may need to change the range selection in the cells listed at the bottom of the status report.

Some other What's*Best!* functions, such as *WBDUAL()*, *WBLOWER()*, or *WBUPPER()* for the dual values, can have references to *WB()* constraint cells without being in the Slack mode.

INFEASIBLE - No Feasible Solution Found

If the *What'sBest!* solver was unable to find a solution that satisfies all the constraints cells and any integer requirements in your model, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****

No Feasible Solution Found (Help Reference: INFEASIBLE)
There is no solution that satisfies all of the constraints and any integrality conditions in the model. Check to make sure all the constraints are properly formulated. Consider easing constraints in the returned model that are either not satisfied or tight. NOTE: The answer returned is not feasible, therefore the value of the objective function is NOT optimal. The solution returned is only for the purpose of illustrating a scenario with violated constraints so the model can be corrected. See the list of cells below that may be contributing to the infeasibilities.

Suggestions:
A linear optimization model containing constraints that cannot be satisfied simultaneously by What's*Best!* is said to be infeasible. With nonlinear models, it's possible that a feasible solution exists, but What's*Best!* was unable to identify that solution from the starting adjustable cell values you specified. In either case, the solver tries to find a solution to the problem that satisfies as many constraints as possible and returns the appropriate solution status in the status report worksheet entitled *WB! Status*.

If a linear or nonlinear problem is truly infeasible, correcting the model can be a very complicated task. The first step is to investigate those constraints contributing to the infeasibility. These cells have been identified for you by What's*Best!* at the end of the *WB! Status* tab. If you have selected the constraint display type in *General Options* as *Indicator*, then the cells with unsatisfied constraints will display *Not* =, *Not* <=, or *Not* >=. Investigating their relationships to satisfied constraints may help you find the conflicts. Next, find all the constraints that are contradictory to the constraints identified in the first step. Finally, eliminate all the contradictory constraints except the ones that most accurately model your business situation.

If a nonlinear problem returns this error message and you suspect there is a feasible solution, change the starting adjustable cell values to a feasible or near-feasible solution and solve again. Note that a very discontinuous problem may have a feasible solution that can only be "found" by starting at the solution point itself.

If you have the global solver option, you may be able to use the multistart capability to automate the selection of alternate starting points. You may also try invoking the global solver, which, assuming the model is not too large, will always find a feasible point if one exists.

INFLARG - Large Infeasibility

If the infeasibility is too large, the following warning message will be displayed on the *WB! Status tab.*

Warning Message:

****WARNING****

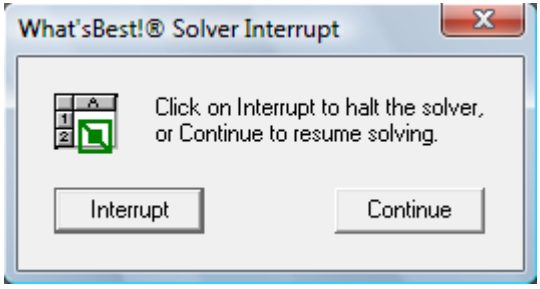
Infeasibility too large for a trusted solution (Help Reference: INFLARG)
Constraint violations exceeding tolerances were found. Check the solution carefully before proceeding. You may be able to resolve this error by decreasing the Feasibility Tolerance in the General Options dialog, or by unchecking the Scale option in the Linear Option dialog box.

Suggestions:

What's*Best!* calculates the amount by which all constraints are violated after the solver returns a solution. In general, this value will be quite small when a feasible or optimal solution is returned. In some cases the total violations, or infeasibilities, may be found to be excessive. When the total infeasibilities exceeds 0.01, What's*Best!* will display this error. Under these conditions, you should check the solution carefully. You may be able to resolve this error by decreasing the Feasibility Tolerance in the *General Options* dialog, or by unchecking the Scale option in the *Linear Option* dialog box.

INTERRUPT - Solver Interrupt

If the *Hold / Interrupt* button is pressed before completion of the solution process, the user can hold the process and continue it some point later, or stop the process via the following dialog box:



By clicking *OK*, the following warning message is displayed on the *WB! Status* tab in the returned workbook.

Warning Message:

```
**************************
*       INTERRUPTED      *
**************************
```

****WARNING****

Solver Interrupt (Help Reference: INTERRUPT).
The solver was interrupted before finding the final solution. Check the solution carefully; it may be sub-optimal or infeasible

Suggestions:
The solver process can be interrupted either by the User, an Iteration Limit, a Runtime Limit, or any software conditions such as a memory limitation. The Iteration or Runtime limits would be set by the user via the Options...|General menu.

When a model is interrupted, there are certain instances when What's*Best!* will be able to return the best solution found so far. The best way to determine if an incumbent solution exists is by checking the *Best Obj* field in the What's*Best! Solver Status* window. If this field contains a numeric value, then there is an incumbent solution that can be returned.

If this field does not contain a numeric value, then a valid solution does not exist. In which case, the values returned by the solver will not be valid. Keep in mind that, at best, the returned solution is most likely sub-optimal. Furthermore, whenever you interrupt the solver you should check the final infeasibility value on the *WB! Status* tab of the returned workbook. An infeasibility value significantly different from zero means that some constraints are violated in the returned answer, and the solution is not valid.

Note: If a feasible integer solution is found for an integer optimization model, restoring the best integer solution may take some time after the initial interrupt. So, don't be concerned if the solver does not interrupt immediately when the model contains integer adjustable cells. The solver is just busy returning to the incumbent solution.

INVALBEST – Invalid Best Cell

Most *What'sBest!* models will have an optimization objective. So, when a model does not contain a valid best cell, the following warning will be displayed on the *WB! Status* tab, although *What'sBest!* may still be able to solve the model to satisfy the constraints.

Warning Message:
****WARNING****
Invalid Best Cell (Help Reference: INVALBEST):
The cell assigned to be minimized/maximized is not adjustable. You need to specify a different Best cell that is either adjustable, or a function of one or more adjustable cells
This warning can be turned off via the WB|Options|General menu.

Suggestions:
A Best cell has been specified in the model, but has no meaning for optimization. Refer to section The ABC's: Three Steps to What'sBest! for information on how to identify a best cell. Occasionally, you may have models that don't require a best cell, where all you want to do is find a set of values for the adjustable cells that satisfy all the constraints. In this case, you should disable this warning by clearing the No Best Cell checkbox on the General Options dialog box.

If there is a WBMAX/WBMIN range and the problem occurs, then it might be:

- A cell with a number but not set adjustable.
- A cell with a formula that is not adjustable dependent.

- A cell with a formula that is adjustable dependent, but the dependency is wrapped up in a cell with unsupported functions.

INVMOD - Invalid Model

If you attempt to solve a model that is clearly not formatted for *What'sBest!*, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Invalid Model (Help Reference: INVMOD):
Your model must have at least one adjustable cell and one formula that is dependent upon an adjustable cell. Refer to the "Getting Started" section of the online help to learn how to format a model for What'sBest.

Suggestions:
The model you are attempting to solve does not have the appropriate formatting for *What'sBest!* You will need to identify the adjustable cells, identify the best cell and input your constraint cells. Refer to section ABC's: Basic Functions for details on how to format a model.

IRRECONST - Irreconcilable Constraints

If a constraint is violated and it does not depend upon any adjustable cells, *What'sBest!* will not be able to satisfy the constraint and it is considered *irreconcilable.* In which case, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Irreconcilable Constraints (Help Reference: IRRECONST)
The cells listed below contain violated constraints that do not depend upon adjustable cells, and, therefore, cannot be reconciled. These constraints were ignored during the solution process. An example of an irreconcilable constraint would be WB(1, "<=", 0). Please check to see that these constraints have been entered correctly and that all adjustable cells have been specified. (cell addresses listed at bottom of tab)

Suggestions:
A simple example of an irreconcilable constraint would be: *@WB(0, ">=", 1).* Clearly, no matter what values are chosen for the adjustable cells, 0 will never be greater than 1. *What'sBest!* will provide you with a list of the irreconcilable cell formulas. You will need to either delete them or correct them so that they are functions of the adjustable cells. To correct them, check the cell formulas for errors and be sure that all the adjustable cells have been specified.

ITRLIM - Iteration Limit Reached

By default, *What'sBest!* does not put a limit on the number of solver iterations. However, by using the *Iteration Limit* option on the *General Options* dialog box posted by the *Options...|General* command, the user may limit the number of iterations. When the program has executed the number of iterations you specified without finding a solution, the following warning message will be displayed on the *WB! Status* tab.

Warning Message:

```
**************************
*       INTERRUPTED       *
**************************
```

****WARNING****

Iteration Limit Reached (Help Reference: ITRLIM).
The limit for the maximum number of iterations was reached before the final solution could be found. Check the solution carefully; it may be sub-optimal or infeasible. The iteration limit is set via the WB|Options|General menu.

Suggestions:

The iteration limit is set through the *WB|Options!General* menu. In some cases, the solver may be able to return the best solution found so far after hitting an iteration limit. However, the same precautions regarding the returned solution apply as when interactively interrupting the solver. For more details, refer to section *Solver Interrupt* above.

LICCAP1 - Constraint Limit

If your model has more constraints than is allowed for your installation, the following error message will be displayed on the *WB! Status* tab.

Error Message:

****ERROR****

Constraint Limit (Help Reference: LICCAP1):
The number of constraints in this model: -number of constraints-
exceeds the limit of: -constraint limit-
The model has more constraints than is allowed by your installation. You will need to either reduce the size of your model or upgrade to a larger version.

Suggestions:

The model exceeds the constraint limit for your installation. The capacity limits of your version may be found by running the *WB!About What'sBest!* command.

One option is to remove constraint cells from the model until it satisfies the constraint limit. Keep in mind that *What'sBest!* defaults to forcing adjustable cells to be non-negative. So, any constraints that put a lower bound of zero on the adjustable cells are not required and may be removed. Also, if the workbook contains multiple independent models, you can separate them into different workbooks in order to meet the limit.

If eliminating constraints is not an alternative, then you may wish to contact LINDO Systems regarding a license upgrade to handle the additional constraints

LICCAP2 - Adjustable Cell Limit

If your model has more adjustable cells than is allowed for your installation, the following error message will be displayed on the *WB! Status* tab.

Error Message:

****ERROR****
Adjustable Cell Limit (Help Reference: LICCAP2):
The number of adjustable cells in this model: -number of adjustables-
exceeds the limit of: -adjustable limit-
The model has more adjustable cells than is allowed by your installation.
You will need to either reduce the size of your model or upgrade to a
larger version.

Suggestions:

The model exceeds the adjustable cell limit for your installation. The capacity limits of your version may be found by running the *WB!About What'sBest!* command.

One option is to remove adjustable cells from the model until it satisfies the limit. Also, if the workbook contains multiple independent models, you can separate them into different workbooks in order to meet the limit.

If eliminating adjustables is not an alternative, then you may wish to contact LINDO Systems regarding a license upgrade to handle the additional constraints.

LICCAP3 - Integer Cell Limit

If your model has more integer cells than is allowed for your installation, the following error message will be displayed on the *WB! Status* tab.

Error Message:

****ERROR****
Integer Cell Limit (Help Reference: LICCAP3):
The number of integer cells in this model: -number of integers-
exceeds the limit of: -integer limit-
The model has more integer cells than is allowed by your installation.
You will need to either reduce the size of your model or upgrade to a
larger version.

Suggestions:

The model exceeds the integer cell limit for your installation. The capacity limits of your version may be found by running the *WB!About What'sBest!* command.
One option is to remove integer cells from the model until it satisfies the limit.

You should also check that there aren't any unintended ranges of integer cells in the model. Integer cells are flagged by *What'sBest!* using range names that begin with "WBINT" and "WBGIN". Review all such range names to be sure that there are no unintended integer cell ranges.

Some classes of models will return naturally integer values. If your model falls in this category, you can remove the integer formatting and let *What'sBest!* solve the model as a continuous problem.

You may also want to review your demands for an integer solution. For instance, in some models you can simply round the solution to the continuous model with little or no ill effects. For instance, if a model calls for producing 1,247,288.33 pencils, you could safely round the number up or down without significant impact. On the other hand, if the model calls for building .6 space stations, rounding up or down will have a huge impact on the objective and feasibility of the constraints. In which case, rounding is not an option.

Check to be sure that all adjustable cells in the model are still required. Also, if the workbook contains multiple independent models, you can separate them into different workbooks in order to meet the limit.

If eliminating integer cells is not an alternative, then you may wish to contact LINDO Systems regarding a license upgrade to handle the additional constraints.

LICCAP4 - Nonlinear Adjustable Cell Limit

If your model has more nonlinear cells than is allowed for your installation, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Nonlinear Adjustable Cell Limit (Help Reference: LICCAP4):
The number of nonlinear adjustable cells in this model: -number of nonlinears-
exceeds the limit of: -nonlinear adjustable cell limit-
The model has more nonlinear adjustable cells than is allowed by your installation.
You will need to either reduce the size of your model or upgrade to a larger version

Suggestions:
The model exceeds the nonlinear adjustable cell limit for your installation. The capacity limits of your version may be found by running the *WB!About What'sBest!* command.

An adjustable cell is considered nonlinear if it appears in a nonlinear manner in one or more cell formulas in your workbook. As an example, suppose that cells *A1* and *B1* are both adjustable cells. Furthermore, suppose that we have the following formula somewhere in the workbook: *SIN(A1) + 3*B1*. The *SIN()* function is a nonlinear function, so this formula causes A1 to be considered a nonlinear adjustable cell. On the other hand, multiplying an adjustable cell by a constant is a linear operation, so this formula alone would not make *B1* a nonlinear adjustable cell. For more information on identifying and solving linear and nonlinear expressions, see *Overview of Mathematical Modeling.*

Evaluate the formulas in which the nonlinear adjustable cells appear to determine if these formulas can be rewritten as linear expressions. If this is not possible, you will need to either reduce the size of your model or upgrade to a larger version of *What'sBest!*.

When possible, formulate your problem using linear expressions. Problems composed entirely of linear relationships solve faster and more reliably. However, in many cases, linear reformulation may not be possible. If your model contains inherently nonlinear expressions, you may choose to turn off this nonlinear warning with the *Nonlinearity present* checkbox in the *General Options* dialog box.

All versions of What's*Best!* are supplied with a linear solver. The downloadable and Solver Suite versions of What's*Best!* provide a nonlinear solver with capacity for a few nonlinear adjustable cells. For the Commercial and more powerful versions of *What'sBest!*, the nonlinear option may be purchased from LINDO Systems.

LICCAP5 - Global Adjustable Cell Limit

If your model has more nonlinear cells than is allowed for your license when you are running the global solver, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Global Adjustable Cell Limit (Help Reference: LICCAP5):
The number of global adjustable cells in this model: -number of globals--
exceeds the limit of: -global adjustable cell limit-
The model has more nonlinear adjustable cells than is allowed by your installation of the global solver. You will need to either reduce the size of your model or upgrade to a larger version.

Suggestions:
The model exceeds the nonlinear adjustable cell limit for your installation of the global solver. This limit is easy to hit on some of the smaller capacity versions of *What'sBest!*, given that they allow only a handful of nonlinear adjustable cells when running the global solver. The capacity limits of your version may be found by running the *WB!About What'sBest!* command. Refer to the previous error message (LICCAP4) for information on the definition of a nonlinear adjustable cell.
You will need to reduce the number of nonlinear adjustable cells in the model or turn off the global solver and use the standard nonlinear solver. If this is not possible, you will need to contact LINDO Systems regarding a license upgrade.

LICKEY1 - Failed to Process License Key

When your *What'sBest!* license key can't be found, read or processed, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Failed to Process License Key (Help Reference: LICKEY1):
Unable to find or read a valid license key. Run the WB|Upgrade command to install a valid license key

Suggestions:
You may need to reinstall your license key with the *WB|License* command. Also, make sure the license file, *LNDWB80.LIC*, is installed in the directory along the What's*Best!* add-in files (typically *\Program Files\Microsoft Office\Office10\Library*). You may need to contact LINDO Systems to obtain a valid license key

LICKEY2 - Pending License Expiration

If your license key will expire in the next day, the following warning message will be displayed on the *WB! Status* tab.

Warning Message:
****WARNING****
Pending License Expiration (Help Reference: LICKEY2):
Your What'sBest! license key will expire at the end of the day.

Suggestions:
You have a temporary license key that will expire at the end of the day. Note that you will still be able to use What's*Best!* after the license expires, but it will revert to a demonstration version with limited capacity. You may want to contact LINDO Systems to obtain a new license key.

LICKEY3 - Expired License Key

When your *What'sBest!* license key has expired, the following warning message will be displayed on the *WB! Status* tab.

Warning Message:
****WARNING****
Expired License Key (Help Reference: LICKEY3):
Your license key has expired. You may wish to contact LINDO Systems to obtain a new license key..

Suggestions:
Your license key has expired. You may continue on running a trial version with limited capacity, or you can contact LINDO Systems to obtain a new license.

LICKEY4 - Pending Expiration of Option Trial

Some versions of *What'sBest!* include trial licenses for some of the optional capabilities (e.g., the global solver). If the trial license for the options is about to expire, the following warning message will be displayed on the *WB! Status* tab.

Warning Message:

****WARNING****

Pending Expiration of Option Trial (Help Reference: LICKEY4):
Your trial period for the optional features will expire at the end of the day.
You can continue to use What'sBest!, but some of the optional features
may no longer be available.

Suggestions:

Some installations come with a free trial period for the optional solvers (Nonlinear, Global and Barrier). In this case, the trial period is about to expire. *What'sBest!* will continue to operate, but you will no longer have access to all the optional solver engines. You may want to contact LINDO Systems to order a license upgrade that includes permanent access to one or more of the optional solvers

LICKEY5 - License Key Dongle Required

Some versions of What's*Best!* include a dongle key with the license. The dongle key is in the form of a key to insert into a USB port of the computer. If this dongle key is required and missing, the following error message will be displayed on the *WB! Status* tab.

Error Message:

****ERROR****

License Key Dongle Required (Help Reference: LICKEY5):

Unable to find or read a valid license key with dongle. Run the

WB|License command to install a valid license key.

Suggestions:

Some licenses require a dongle key at the time of order to insert into a USB port of the computer. If the dongle key is missing, corrupted, or does not match the license file, so you will not be able to run *What'sBest!*. What's*Best!* can still continue to operate on a demo license capacity. You may want to contact LINDO Systems to order a license upgrade that includes permanent access to one or more of the optional solvers including a dongle key.

LICOPT1 - Global Solver Not Licensed

If you attempt to use the global solver option without a license, the following error message will be displayed on the *WB! Status* tab.

Error Message:

****ERROR****

Global Solver Not Licensed (Help Reference: LICOPT1).
The license for your installation does not allow for use of the global solver option.

Suggestions:
The global solver is an optional feature. You've attempted to use the global solver with a copy of What's*Best!* that does not have an appropriate license. You will need to either disable the global option (*WB!*|*Options*|*Global Solver*), or contact LINDO Systems regarding a license upgrade that includes the global option.

Note: The global solver needs the nonlinear option and the mixed integer option to operate.

LICOPT2 - Nonlinear Solver Not Licensed

If you attempt to use the nonlinear solver option without a license, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Nonlinear Solver Not Licensed (Help Reference: LICOPT2).
The license for your installation does not allow for the solution of nonlinear models.

Suggestions:
The nonlinear solver capability is an optional feature. You've attempted to solve a nonlinear model with a copy of *What'sBest!* that does not have the option enabled. You need to either reformulate the model to eliminate all nonlinear cells, or contact LINDO Systems regarding a license upgrade that includes the nolinear solver option. For more information on identifying nonlinear expressions, see *Overview of Mathematical Modeling*.

LICOPT3 - Barrier Solver Not Licensed

If you attempt to use the barrier solver option without a license, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Barrier Solver Not Licensed (Help Reference: LICOPT3).
The license for your installation does not allow for use of the global solver option.

Suggestions:
The barrier solver is an optional feature. You've attempted to use the barrier solver with a copy of What's*Best!* that does not have an appropriate license. You will need to either disable the barrier option (*WB!*|*Options*|*Linear Solver*), or contact LINDO Systems regarding a license upgrade that includes the barrier option.

LICOPT4 - Mixed Integer Solver Not Licensed

If you attempt to use the mixed integer programming solver option without a license, the following error message will be displayed on the *WB! Status* tab.

Error Message:

****ERROR****

Mixed Integer Solver Not Licensed (Help Reference: LICOPT4)

The license for your installation does not allow for use of the mixed integer solver option.

Suggestions:

The mixed integer programming solver is an optional feature. You've attempted to use the mixed integer solver with a copy of *What'sBest!* that does not have an appropriate license. You will need to either disable any integer options (*WB!*|*Options*|*Integer Pre-Solver*, *WB!*|*Options*|*Integer Solver*), or contact LINDO Systems regarding a license upgrade that includes the mixed integer option.

LICOPT5 - Stochastic Solver Not Licensed

If you attempt to use the stochastic solver option without a license, the following error message will be displayed on the *WB! Status* tab.

Error Message:

****ERROR****

Stochastic Solver Not Licensed (Help Reference: LICOPT5):

The license for your installation does not allow for use of the stochastic solver option.

Suggestions:

The stochastic programming solver is an optional feature. You have attempted to set up a stochastic model with a copy of *What'sBest!* that does not have an appropriate license. You will need to either disable any stochastic featuress (*WB!*|*Options*|*Stochastic Solver*, *WB!*|*Advanced*|*Stochastic Support*), or contact LINDO Systems regarding a license upgrade that includes the stochastic option.

LICOPT6 - Conic Solver Not Licensed

If you attempt to use the conic solver option without a license, the following error message will be displayed on the *WB! Status* tab.

Error Message:

****ERROR****

Conic Solver Not Licensed (Help Reference: LICOPT6):

The license for your installation does not allow for use of the conic solver option.

Suggestions:

The Conic Solver, or Second Order Cone Programming (SOCP), is invoked when the model is Convex, with Global option set and checked, the Quadratic Recognition on, and the Conic option set.

The conic programming solver is an optional feature. You have attempted to set up a conic model with a copy of *What'sBest!* that does not have an appropriate license. You will need to contact LINDO Systems regarding a license upgrade that includes the conic option.

LOOKUP - Unsupported Lookup Usage

What'sBest! supports the *VLOOKUP()* and *HLOOKUP()* Excel® functions only under certain conditions. If you use an unsupported variant of either of these lookup functions, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Unsupported Lookup Usage (Help Reference: LOOKUP):
The solver has been halted because the following cell contains an unsupported argument to either VLOOKUP() or HLOOKUP() functions. The first argument can be a number, simple cell, or text, the second argument must be a range where the lookup value is located, the third argument should be a number, and the fourth argument is an optional TRUE/FALSE value. This warning can be turned off via the WB|Options|General menu.
-cell address-

Suggestions:
The *VLOOKUP()* (and the *HLOOKUP()*) function requires three arguments as follows: *VLOOKUP(*lookup_*value, table_array, col_index_num, [range_*lookup*]). What'sBest!* places the following restrictions on these arguments:

- All arguments must not be a formula,
- the *table_array* argument must be the range where the lookup value is located. Furthermore, the lookup value should be in the first column of this range,
- the *col_index_num* argument must be a number, not a formula or a cell reference, and
- *range_lookup* is an optional TRUE / FALSE argument.

Refer to the *What'sBest!* sample file *VlookupWBSolver.xls* for an example of valid *VLOOKUP()* usage.

MATCH Unsupported Match Usage

What'sBest! supports the *MATCH()* Excel® function only under certain conditions. If you use an unsupported variant of *MATCH* function, the following warning message will be displayed on the *WB! Status* tab.

Warning Message:
****WARNING****
Unsupported Match Usage (Help Reference: MATCH):
The solver detected the following cell contains an unsupported argument to a MATCH() function. The first and the second arguments providing the lookup value and the array must be constant numeric values or cells with constant numeric values, but cannot be a variable, an adjustable, or a random parameter. The Solver could anyway proceed by reading the displayed value in the spreadsheet cell.
-cell address-

Suggestions:
The *MATCH()* function requires the arguments as follows: *MATCH(lookup_value, lookup_array, [match_type]).* What's*Best!* places the following restrictions on these arguments:

- All arguments must not be a formula,
- the *lookup_value* argument must be a value, not a formula,
- the *lookup_array* argument must be values, in ascending order for *match_type* =1, or descending =-1, any order=0
- all arguments must evaluate to being numeric rather than text,
- the *match_type* is an optional -1, 0, 1 argument; default being 1.

MBEST – Multiple Best Cells

If there are more than one Objective cell in the model, the following warning message will be displayed on the *WB! Status* tab.

Warning Message:
****WARNING****
Multiple Best Cells (Help Reference: MBEST):
Multiple Best Cells 'WBMAX' or 'WBMIN' were found in the workbook. The model must have no more than one Best/Objective cell defined. Open Excel's Name Manager and review the list of 'WBMAX' or 'WBMIN' names. Delete all extraneous 'WBMAX' or 'WBMIN' names whose scope may be the Workbook or a specific sheet.

Suggestions:
The solver found multiple cells defined as Objective with the name WBMAX or WBMIN. This situation may happen when an objective cell was selected and later the worksheet was deleted, or when the user defines such names locally to a worksheet, instead to a workbook.

The list of names can be reviewed via the Formulas-Name Manager menu of Excel®. The user will need to remove any extraneous WBMAX or WBMIN names to keep no more than one Objective cell.

MEMORY - Insufficient Memory

If there is not enough random access memory on your machine, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Insufficient Memory (Help Reference: MEMORY).
There is not enough random access memory on the machine to solve the mode to completion. Try closing all other applications or adding memory.
- additional message -

Suggestions:
The solver was not able to allocate sufficient memory. Try shutting down any other applications that are open to free up memory. Memory requirements grow along with your model; so reducing the model's size may help too.

Another option is to try and increase the amount of virtual memory available to your computer. Unfortunately, relying on virtual memory rather than true RAM memory will slow the solver down substantially.

If all else fails, you should consider adding more memory to your machine. This assumes that you haven't already added the maximum amount of memory possible to your machine. In the case where you have already maxed out your machine, you will need to reduce the size of the model.

MODERR - Parsing Cell Formula

If *What'sBest!* can't parse a particular cell formula, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Parsing Cell Formula (Help Reference: MODERR):
What'sBest! was not able to parse the formula in the following cell:
-cell address-
-additional message-

Suggestions:
If a formula contains an unexpected, inappropriate, or unrecognized operation, where a comma is expected instead of the multiplication operator, *What'sBest!* may not know how to proceed.

Here is a list of possible causes:

- The formula contains an unsupported combination of cells and areas (e.g., multiplying a range of cells by a numeric value).
- The formula may reference a cell in an error state (e.g., *#NUM!*).
- The cell's formula may exploit some feature unsupported by *What'sBest!*.
- Some Excel® functions need to meet a particular format to be supported by *What'sBest!*.

If the formula still results in this message and the cell is used only for reporting purposes, you may wish to place it in a *WBOMIT* range to force *What'sBest!* to ignore the cell.

NAME - Unsupported Value in the Name List

If *What'sBest!* can't parse a particular name, the following warning message will be displayed on the *WB! Status tab.*

Warning Message:
****WARNING****
Unsupported Value in the Name List (Help Reference: NAME):
Define Name with a numeric value, or a single range. Multiple ranges, Formula,
or external links are not fully supported.
Name will be taken as a - 0 - number.
Name: -name-

Suggestions:
What'sBest! does not support names containing any of the following: links to external workbooks, formulas, symbols, or multiple discontinuous ranges. Verify all names using "Insert|Name|Define…", remove any unused names, eventually copy the external data directly into the current workbook, or define Names with a numeric value, or a single range selection.
You may need to call LINDO Systems for assistance.

NAMECONFLICT - Name Cell with Conflicting Properties

If *What'sBest!* can't decide the property for a particular cell or range, the following warning message will be displayed on the *WB! Status* tab.

Warning Message:
****WARNING****
Name Cell with Conflicting Properties (Help Reference: NAMECONFLICT):
A variable cell appears in two different names with conflicting properties; for instance in a 'WBBINxxx' and 'WBINTxxx'
Mathematically, 'WBBINxxx' is the tighter constraint so it is the binding one, and 'WBINTxxx' is redundant. You may review these names via the Name Manager of Excel®, or the What'sBest menu.

Suggestions:
What'sBest! has detected a cell or a range with different properties: Integer or Binary.The solver will first consider the binary property and display the warning message if the cell is also carrying the Integer property.
The user may want to review the list of these properties via the What'sBest! menu, at Integers-Integer/Binary to correct any range selection for the names WBBINxxx or WBINTxxx.

NLINCELL - Nonlinearities Present

Assuming the *Nonlinearity Present* checkbox is checked, a model containing nonlinear expressions will produce the following warning message on the *WB! Status tab.* Below the message, *What'sBest!* provides the names of the cells containing nonlinear relationships.

Warning Message:
****WARNING****
Nonlinearities Present (Help Reference: NLINCELL):
The cells below contain nonlinear expressions. If these cells are used only for reporting, then, for efficiency, they should be included in a WBOMIT range (refer to documentation). In some cases, nonlinear cells may be linearized automatically by the Linearization option that is set in the General Options dialog box. This warning can be turned off with the Nonlinearity Present checkbox in the General Options dialog box
(cell addresses listed at bottom of tab).

Suggestions:
A cell is considered nonlinear if it applies nonlinear mathematical operations to quantities that depend on the adjustable cells. As an example, suppose that cells A1 and B1 are both adjustable cells. Furthermore, suppose that we have the following formula somewhere in the workbook: A1/B1. Dividing by an adjustable cell is a nonlinear operation, so this term causes the formula containing it to be considered nonlinear. For more information on identifying and solving linear and nonlinear expressions, see *Overview of Mathematical Modeling*.
When possible, formulate your models using linear expressions. Problems composed entirely of linear relationships solve faster and with a higher degree of confidence. Evaluate the formulas in the nonlinear cells to determine if these formulas can be rewritten as linear expressions. In many cases, a nonlinear expression can be rewritten in an equivalent, but linear, manner. Unfortunately, the techniques for doing this are not particularly intuitive. Referring to a good text on mathematical programming may be of help.
In some cases, the Linearization capability in What's*Best!* can automatically reformulate your model "under the hood" to make it entirely linear. However, linearizing a model by hand is always more efficient. Plus, if the Linearization feature does not linearize the model completely, you may end up with a more difficult model to solve.
In many cases, linear reformulation may not be possible. If your model contains inherently nonlinear expressions, you may choose to turn off this nonlinearity warning by clearing the Nonlinearity Present checkbox on the *General Options* dialog box.

NOADJ - No Adjustable Cells

Adjustable cells represent activities that are under the solver's direct control. In other words, these cells have been identified as ones that are allowed to be changed by the solver during the optimization process. If there are no adjustable cells in the model, the following error message is displayed on the *WB! Status* tab.

Error Message:
****ERROR****
No Adjustable Cells (Help Reference: NOADJ):

No cells have been specified as adjustable. You must use the Adjustable command to specify the cells you want What'sBest! to adjust to find the solution. Adjustable cells must contain numbers. Adjustable cells that are blank or contain formulas or text will be ignored by the solver)

Suggestions:
You will need to add at least one adjustable cell to the model. See section *The ABC's: Three Steps to What'sBest!* for information on how to mark cells as being adjustable. Adjustable cells are easy to find in that they are displayed in a blue font and have the Adjustable style applied to them. When identifying your adjustable cells, be sure to place a numeric value (any will do) in each adjustable cell. If an adjustable cell does not contain a numeric value, *What'sBest!* will not attempt to change its value.

NOBEST - No Best Cell

Most *What'sBest!* models will have an optimization objective, i.e., a best cell. It could be profit maximization, cost minimization, or some other criterion that is a function of the adjustable cells. So, when a model does not contain a best cell, the following warning will be displayed on the *WB! Status* tab.

Warning Message:
****WARNING****
No Best Cell (Help Reference: NOBEST).
Either no cell has been specified to be maximized or minimized, or the cell that is marked is not a function of any adjustable cells. If this is an optimization model, use the Best command or the Minimize or Maximize toolbar button to specify a best cell as the objective of the optimization.
This warning can be turned off via the WB|Options|General menu.

Suggestions:
If you failed to indicate a best cell before solving, specify a cell to be maximized or minimized and re-solve. Refer to section *The ABC's: Three Steps to What'sBest!* for information on how to identify a best cell. Occasionally, you may have models that don't require a best cell, where all you want to do is find a set of values for the adjustable cells that satisfy all the constraints. In this case, you should disable this warning by clearing the *No Best Cell* checkbox on the *General Options* dialog box.

If there is no WBMAX or WBMIN range, then no cell has been specified to be maximized or minimized.

NOCONST - No Constraint Cells

Most *What'sBest!* models will have at least one, and most likely many, constraints on the adjustable cells. Constraints place limitations on the adjustable cells and/or functions of the adjustable cells. For instance, resources will never be unlimited. At some usage level resources will be exhausted. Another example would be that orders must be filled. A cost minimization model with no constraints to fill orders would minimize cost by producing nothing, which clearly is not an optimal answer to the real world problem. Constraints are used to indicate these types of limitations. It would be unusual for a model to have no constraints. So, when What'sBest! encounters such a situation, it displays the following warning on the WB! Status tab.

Warning Message:
****WARNING****
No Constraint Cells (Help Reference: NOCONST)
The solver recognized no valid constraints. The model either contained no constraint functions or only constraint functions that did not depend on any adjustable cells. If the model was developed for an earlier version of What'sBest, the constraints may need to be converted to the current format for constraint functions.

Suggestions:
You will need to add at least one constraint cell to the model. Constraints can be created with the *WB!|Constraints* command. You may also want to refer to section *The ABC's: Three Steps to What'sBest!* for detailed information on how to create a constraint cell.

On rare occasions it may actually make sense to have an unconstrained model. This would be a model where you need to find the extreme point of an objective function dependent upon unconstrained adjustable cells. In this case, you may want to create a single, vacuous constraint cell in order to avoid this error message (e.g., *X* >= -1.e20).

NUMERICAL - Numerical Error

In some cases, models will not be numerically stable. In such a case, the solver will not be able to continue when numeric values become too large and the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Numerical Error (Help Reference: NUMERICAL):
The optimizer encountered numerical errors while solving and was unable to continue the optimization. Scaling the model's coefficients so that they don't cover as large a range may be helpful in eliminating this error.

Suggestions:
Typically, this error message occurs when a model is poorly scaled. A model is poorly scaled when the ratio of the largest coefficient to the smallest coefficient in the model's formulas is too high. If this error occurs on a nonlinear model, then you can try a different starting value. If this doesn't seem to help, then you should try rescaling your model. *What'sBest!* displays information on the *WB! Status*

tab regarding the values for the largest and smallest coefficients and where they are located in the workbook. This information can be useful in tracking down the poorly scaled parts of the model. Changing the units of measure in your model may help to improve scaling. For example, if your model contains dollar amounts, you may want to try switching to units of thousands of dollars.

OMITTED - Omitted Cell Reference

Cells within *WBOMIT* ranges (i.e., cells that have been omitted from the optimization via the *WB!|Advanced|Omit* command) cannot be referenced in cell formulas that lie outside *WBOMIT* ranges. When this situation is encountered, What's*Best!* will display the following error message on the *WB! Status* tab.

Error Message:
****ERROR****
Omitted Cell Reference (Help Reference: OMITTED).
The solver has been halted because the following cells are contained in WBOMIT ranges and have been referenced by formulas that are not contained in any WBOMIT range.
-cell addresses-

Suggestions:
There are several options here:

- Remove the omitted cells that are being referenced (i.e., the cells listed in the error message) from all *WBOMIT* ranges.
- Place the cells referencing the omitted cells in *WBOMIT* ranges too.
- Eliminate the references to the omitted cells from the un-omitted cell formulas.

QUAPREC - Quadratic Recognition

A *quadratic program* (QP) is any model that is linear with the exception of product terms involving two variables (e.g., *3*X*Y*). If you have enabled the *Quadratic Recognition* option and you try and solve a quadratic programming model that is non-convex, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Quadratic Recognition (Help Reference: QUAPREC)
The quadratic solver requires that the model be convex. The model does not satisfy this condition. You must disable the quadratic solver by turning off quadratic recognition and then re-solve. This will cause the general purpose, nonlinear solver to be invoked.

Suggestions:
The *Quadratic Recognition* option is controlled with the *WB|Options|Nonlinear Solver* command. When this option is enabled, the nonlinear solver uses algebraic preprocessing to determine if an arbitrary nonlinear model is actually a quadratic program, in which case, it can be passed to the faster

quadratic solver. The quadratic solver assesses the model to determine if it is convex. If the model is found to be non-convex, you will receive this error message. At which point, you must either reformulate the model so that it becomes convex, or turn off quadratic recognition and use the general-purpose, nonlinear solver. Refer to *Overview of Mathematical Modeling* for a discussion of the concept of convexity in math programs.

RUNTIME - Runtime Limit Reached

By default, *What'sBest!* does not put a limit on the total time that the solver will run. However, by using the *Runtime Limit* option on the *General Options* dialog box posted by the *Options...|General* command, the user may limit the number of iterations. When the program exceeds the specified runtime limit, the following warning message will be displayed on the *WB! Status* tab.

Warning Message:

```
**************************
*       INTERRUPTED       *
**************************
```

****WARNING****
Runtime Limit Reached (Help Reference: RUNTIME)
The limit for the maximum runtime was reached before the final solution could be found. Check the solution carefully; it may be sub-optimal or infeasible. The runtime limit is set via the WB|Options|General menu.

Suggestions:

The runtime limit is set through the *WB|Options!General* menu. In some cases, the solver may be able to return the best solution found so far after hitting a runtime limit. However, the same precautions regarding the returned solution apply as when interactively interrupting the solver. For more details, refer to section *Solver Interrupt* above.

The Global Solver for nonlinear model may have limitations when combined with other features, such as the Function Support for Excel® or User's Defined Functions. In such a situation, *What'sBest!* will display the following warning message, and eventually an additional error message, on the *WB! Status* tab.

SOLVERGOP - Global Solver Use

Error Message:

****WARNING****
Global Solver Use (Help Reference: SOLVERGOP):
The Global Solver does not proceed when a User Defined Function or an Unsupported built-in function is found in the instruction list. You may turn off the Function Support via the Advanced menu.
-cell address-

Suggestions:

The Global Solver needs a complete understanding of the function in the search of a global optima. Functions that are not supported by default cannot provide any valuable details. The user may choose alternate features:

- Uncheck the Global Solver to run the Nonlinear solver. Instead of a global solution, the solver may find a local optima.
- Uncheck the Function Support. Unsupported functions will be read as a constant numeric cell, without any further processing.
- Define a WBCALCxxx or WBOMITxxx range, depending if the value of the cell should be part of the model to solve.

SOLVERR - Solver Error

Mathematical programming problems are inherently difficult to solve, and, as a result, the What'*sBest!* solver may encounter an error from which it is unable to recover. This should not be a common occurrence, but should the solver encounter such an error, *What'sBest!* will display the following error message on the *WB! Status* tab.

Error Message:
****ERROR****
Solver Error (Help Reference: SOLVERR).
The solver encountered an error from which it was unable to recover. A brief description of the error follows:
-error description-

Suggestions:
Solver errors should be an uncommon occurrence, and you may need to contact LINDO Systems to resolve the error. Short of that, here are some suggestions:

- If the error appears to be numerical in nature and your model is nonlinear, you may want to try restarting from a different point. This will tend to force the solver to take a different path to the solution, and, hopefully, avoid the error condition.
- Rescale the model so that the ratio of largest to smallest coefficients is reduced to improve numerical stability. The *What'sBest!* status tab displays the value of these coefficients and the cells in which they appear. Often, simply changing units of measure (e.g., dollars to thousands of dollars) will be enough to bring this ratio into line.
- Experimenting with some of the solver parameters via the *WB|Options* command may help solve the problem. However, should this not solve the problem, be sure to restore all options back to their default settings.
- You may be able to choose an alternate solver. For example, with linear models you can choose the primal, dual, or barrier solver. For nonlinear models there are two versions of the standard nonlinear solver, a quadratic solver, and a global solver to choose from.

SPCONFLICT - Stochastic Conflicting Declarations

Using the stochastic feature, the user has to associate a stage information to a variable cell. The *=WBSP_RAND()* and *WBSP_VAR()* functions are used to specify the timing information. If the arguments to these functions are duplicated, or found to have cross references, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Stochastic Conflicting Declarations (Help Reference: SPCONFLICT):
A Stochastic function cell is incorrectly formatted. It appears the argument cell below refers to a cell being specified by other Stochastic functions, leading to contradictory declarations. Correct the formula and the validity of the arguments by tracking its. dependencies or references. The argument cell should be declared only once.
-cell address-

Suggestions:
There is a conflicting declaration of the argument cell in the Stochastic timing functions. The timing functions must use the format: *=WBSP_RAND(arg1, cells)* for randoms, or *=WBSP_VAR(arg1, cells)* for variables, where *arg1* is a numeric value or a cell reference to a numeric value, and *cells* is a reference to the cells you want to apply this timing information. The cell reference should be declared once, without cross or duplicated references. For more information on using the stochastic feature refer to section *Options...|Stochastic Solver*.

SPCORR - Stochastic WBSP_CORR Format

Using the stochastic feature, the user has to associate a distribution to a random cell, either discrete or continuous. The *=WBSP_CORR...()* function is used to request a one-argument correlation. If the arguments to this function are not correctly specified, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Stochastic WBSP_CORR Format (Help Reference: SPCORR):
A WBSP_CORR... cell with 1 argument is incorrectly formatted. Correct the formula and the validity of the arguments in the cell below. The format should be '=WBSP_CORR...(arg1, cells)', where 'arg1' is a reference, and 'cells' must refer to random cells.
-cell address-

Suggestions:

There is an incorrectly formatted distribution function cell in the model. A correlation function cell must use the format: *=WBSP_CORR...(arg1, cells)*, where *arg1* is a cell reference, and *cells* is a reference to the cells you want to apply this correlation type: Kendall, Pearson, or Spearman. The cell reference should be to a random cell. For more information on using the stochastic feature refer to section *Options...|Stochastic Solver*.

SPDIST1 - Stochastic WBSP_DIST 1 Format

Using the stochastic feature, the user has to associate a distribution to a random cell, either discrete or continuous. The *=WBSP_DIST...()* function is used to request a one-argument distribution. If the arguments to this function are not correctly specified, the following error message will be displayed on the *WB! Status* tab.

Error Message:

****ERROR****

Stochastic WBSP_DIST 1 Format (Help Reference: SPDIST1):

A WBSP_DIST... cell with 1 argument is incorrectly formatted. Correct the formula and the validity of the arguments in the cell below. The format should be '=WBSP_DIST...(arg1, cells)', where 'arg1' is a reference, and 'cells' must refer to random cells.

-cell address-

Suggestions:
There is an incorrectly formatted distribution function cell in the model. A distribution function cell must use the format: *=WBSP_DIST...(arg1, cells)*, where *arg1* is a cell reference, and *cells* is a reference to the cells you want to apply this one-argument distribution. The cell reference should be to a random cell. For more information on using the stochastic feature refer to section *Options...|Stochastic Solver*.

SPDIST2 - Stochastic WBSP_DIST 2 Format

Using the stochastic feature, the user has to associate a distribution to a random cell, either discrete or continuous. The *=WBSP_DIST...()* function is used to request a two-argument distribution. If the arguments to this function are not correctly specified, the following error message will be displayed on the *WB! Status* tab.

Error Message:

****ERROR****

Stochastic WBSP_DIST 2 Format (Help Reference: SPDIST2):

A WBSP_DIST... cell with 2 arguments are incorrectly formatted. Correct the formula and the validity of the arguments in the cell below. The format should be '=WBSP_DIST...(arg1, arg2, cells)', where 'arg1' and 'arg2' are references, and 'cells' must refer to random cells.

-cell address-

Suggestions:
There is an incorrectly formatted distribution function cell in the model. A distribution function cell must use the format: *=WBSP_DIST...(arg1, arg2, cells)*, where *arg1* and *arg2* are cell references, and *cells* is a reference to the cells you want to apply this two-argument distribution. The cell reference should be to a random cell. For more information on using the stochastic feature refer to section *Options...|Stochastic Solver*.

SPDIST3 - Stochastic WBSP_DIST 3 Format

Using the stochastic feature, the user has to associate a distribution to a random cell, either discrete or continuous. The *=WBSP_DIST...()* function is used to request a three-argument distribution. If the arguments to this function are not correctly specified, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****

Stochastic WBSP_DIST 3 Format (Help Reference: SPDIST3):

A WBSP_DIST... cell with 3 arguments are incorrectly formatted. Correct the formula and the validity of the arguments in the cell below. The format should be '=WBSP_DIST...(arg1, arg2, arg3, cells)', where 'arg1', 'arg2' and 'arg3' are references, and 'cells' must refer to random cells.

-cell address-

Suggestions:
There is an incorrectly formatted distribution function cell in the model. A distribution function cell must use the format: *=WBSP_DIST...(arg1, arg2, arg3, cells)*, where *arg1*, *arg2* and *arg3* are cell references, and *cells* is a reference to the cells you want to apply this three-argument distribution. The cell reference should be to a random cell. For more information on using the stochastic feature refer to section *Options...|Stochastic Solver*.

SPERR - Stochastic Error

Using the stochastic feature, the user has to associate the stochastic information to a core model. The set of *=WBSP_...()* function is used to set the information. If the arguments of these functions are not correctly specified, or if any piece of information can not be loaded, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****

Stochastic Error (Help Reference: SPERR):

Your model must have at least one formula that is dependent upon an adjustable cell,

and a random cell. Refer to the 'Getting Started' section of the online help to learn

how to format a stochastic model for What'sBest.

-cell address-

-additional message-

Suggestions:
There is an incorrectly formatted stochastic function cell in the model, or the core model is missing the Adjustable, Best Constraint cells. A stochastic function cell must use the format: *=WBSP_...(cells)*, where *cells* is a reference to the cells having the argument value. For more information on using the stochastic feature refer to section *Options...|Stochastic Solver*.

SPHIST - Stochastic WBSP_HIST Format

Using the stochastic feature, *What'sBest!* can return the histogram information on any cells. These values tell you how sensitive the solutions are to various stages of the model. The *=WBSP_HIST()* function is used to request histogram values on any cell. If the arguments to this function are not correctly specified, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Stochastic WBSP_HIST Format (Help Reference: SPHIST):
A WBSP_HIST cell is incorrectly formatted. Correct the formula and the validity of the arguments in the cell below. The format should be '=WBSP_HIST(number, cells)', where the 'number' states the number of desired bins, and 'cells' must refer to any cells.
-cell address-

Suggestions:
There is an incorrectly formatted Stochastic Histogram Report cell in the model. A reporting cell must use the format: *=WBSP_HIST(number, cells)*, where *number* is a numeric number referring to the number of bins, and *cells* is a reference to the cells you want to report the histogram. The *number* can be left to 0, so the solver will decide the number of bins. The *cells* reference can be any cells. For more information on using the Stochastic Histogram values, refer to section *Options...|Stochastic Solver*.

SPRAND - Stochastic WBSP_RAND Format

Using the stochastic feature, the user has to associate a stage information to a random cell. The *=WBSP_RAND()* function is used to specify the timing information. If the arguments to this function are not correctly specified, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****

Stochastic WBSP_RAND Format (Help Reference: SPRAND):

A WBSP_RAND cell is incorrectly formatted. Correct the formula and the validity of the

arguments in the cell below. The format should be '=WBSP_RAND(stage, cells)', where

'stage' is numeric, and 'cells' must refer to random cells. Also, verify the number

of stages in the mode.

-cell address-

-additional message-

Suggestions:
There is an incorrectly formatted random function cell in the model. A random cell must use the format: *=WBSP_RAND(arg1, cells)*, where *arg1* is a numeric value or a cell reference to a numeric value, and *cells* is a reference to the cells you want to apply this timing information. The cell reference should be to a random cell. Also, make sure of the staging consistency in the model. For more information on using the stochastic feature refer to section *Options...|Stochastic Solver.*

SPREP - Stochastic WBSP_REP Format

Using the stochastic feature, *What'sBest!* can return the solution information on any cells. These values tell you how sensitive the solutions are to various stages of the model. The *=WBSP_REP()* function is used to request solution values on variables, and outcomes on random cells. If the arguments to this function are not correctly specified, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****

Stochastic WBSP_REP Format (Help Reference: SPREP):

A WBSP_REP cell is incorrectly formatted. Correct the formula and the validity of the

arguments in the cell below. The format should be '=WBSP_REP(cells)', where

'cells' must refer to random or variable cells.

-cell address-

Suggestions:
There is an incorrectly formatted Stochastic Report cell in the model. A reporting cell must use the format: *=WBSP_REP(cells)*, where *cells* is a reference to the cells you want to report the solutions. The cell reference should be to a random or a variable cell. For more information on using Stochastic Solution values refer to section: *Options...|Stochastic Solver*.

SPSTSC - Stochastic WBSP_STSC Format

Using the stochastic feature, the user has to specify a table of scenario per stage. The *=WBSP_STSC()* function is used to select this two-column table, where stage is in the first column, and scenario on the second column. If the argument to this function is not correctly specified, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****

Stochastic WBSP_STSC Format (Help Reference: SPSTSC):

A WBSP_STSC cell is incorrectly formatted. Correct the formula and the validity of

the arguments in the cell below. The format should be '=WBSP_STSC(cells)', where

'cells' must refer to a two-column range. Also, review the value of the arguments in

the distribution function.

-additional message-

Suggestions:
There is an incorrectly formatted Stochastic/Scenario cell in the model. Such cell must use the format: *=WBSP_STSC(cells)*, where *cells* is a reference to a two-column range. Also, review the value of the arguments in the distribution function; some of them could be out of range. For more information on using Stochastic Solution values refer to section *Options...|Stochastic Solver.*

SPVAR - Stochastic WBSP_VAR Format

Using the stochastic feature, the user has to associate a stage information to a variable cell. The *=WBSP_VAR()* function is used to specify the timing information. If the arguments to this function are not correctly specified, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Stochastic WBSP_VAR Format (Help Reference: SPVAR):
A WBSP_VAR cell is incorrectly formatted. Correct the formula and the validity of the arguments in the cell below. The format should be '=WBSP_VAR(stage, cells)', where 'stage' is numeric, and 'cells' must refer to variable cells.
-cell address-
-additional message-

Suggestions:
There is an incorrectly formatted variable function cell in the model. A variable cell must use the format: *=WBSP_VAR(arg1, cells)*, where *arg1* is a numeric value or a cell reference to a numeric value, and *cells* is a reference to the cells you want to apply this timing information. The cell reference should be to a variable cell. For more information on using the stochastic feature refer to section *Options...|Stochastic Solver*.

STRARG - String Arguments Found

In general, *What'sBest!* expects all function arguments to be numeric. If any unexpected string arguments were found while parsing the workbook, *What'sBest!* will display the following warning message on the *WB! Status* tab.

Warning Message:
****WARNING****
String Arguments Found (Help Reference: STRARG):
Text arguments have been found in formulas. Their values have been taken to be zero. This can lead to infeasible or sub-optimal solutions. Please check the returned solution carefully. If you wish the string arguments to be used, click on the menu Advanced -> String Support -> On -> OK.
On the other hand if you want string arguments to be disregarded, this warning can be turned off via the menu Options -> General.
(cell addresses listed at bottom of tab).

Suggestions:
In many instances, text arguments can be replaced by numeric arguments with the same end effect. See if this is possible in the cell formulas listed as part of this error message.

STRLIST - String List Error

What'sBest! supports string cells and arguments under certain conditions. If your string listing becomes unsupported, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
String List Error (Help Reference: STRLIST):
What'sBest encountered an error while loading all the strings.
You may need to shorten the string length to save memory, omit cells containing strings, or simplify the format of these strings:
-cell address-

Suggestions:
What's*Best!* may have encountered the following restrictions on the string support

- Too many strings to load to the current allowable memory

- String containing unsupported, or exotic, characters
- Avoid formulation containing the double quote "" (null) character
- Remove or omit operations with string of characters
- The maximum length of the string is too large
- The return value of a formula cell should be numeric rather than text
- Omit string cells or sheets that are not relevant to the model

Refer to the *What'sBest!* sample file *VehicleRouting_String.xls* for an example of valid string argument usage.

STRRES - String Result of Formula

What'sBest! supports string cells and arguments under certain conditions, but the formula should return a numeric value instead of a string. The following warning message will be displayed on the *WB! Status* tab.

Warning Message:
****WARNING****
String Result of Formula (Help Reference: STRRES):
What'sBest encountered formulas displaying results in a text form. You can use the String Support feature which allows you to use strings as arguments, but the formula should return a numeric value instead of a string, so their results have been taken to be zero. Otherwise, you may want to set omitted areas to discard these cells (cell addresses listed at bottom of tab).

Suggestions:

What'sBest! may have encountered the following situations on the string support

- The return value of a formula cell should be numeric rather than text
- A numeric result displayed in a text form
- An illegal string operation
- An omitted string cells or sheets that are not relevant to the model

Refer to the *What'sBest!* sample file *VehicleRouting.xls* for an example of valid string argument usage.

SUMIF - Unsupported SUMIF Usage

What'sBest! supports the *SUMIF()* Excel® function only under certain conditions. If you use an unsupported variant of *SUMIF*, the following warning message will be displayed on the *WB! Status* tab.

Warning Message:
****WARNING****
Unsupported SUMIF Usage (Help Reference: SUMIF)
The solver has found cells containing SUMIF() functions with syntax not supported by What'sBest.

The second argument (criterion) can be a number, simple cell reference, or text. Expressions or functions are not permitted. The first and third arguments must be cell ranges, with the same shape and size.

Suggestions:
The *SUMIF()* function requires three arguments as follows: *SUMIF(range, criteria, sum_range)*. *What'sBest!* places the following restrictions on these arguments:

- The *criteria* argument can be a number, simple cell reference, or text. *What'sBest!* does not support *SUMIF* functions that contain mathematical expressions or Excel® function for the criteria argument.
- The *range* and *sum_range* arguments must be cell ranges, with the same shape and size.

Refer to the *What'sBest!* sample file *NetFloSimple.xlsx* for an example of valid *SUMIF()* usage.

SUMIFS - Unsupported SUMIFS Usage

What'sBest! supports the *SUMIFS()* Excel® function only under certain conditions. If you use an unsupported variant of *SUMIFS*, the following warning message will be displayed on the *WB! Status* tab.

Warning Message:
****WARNING****
Unsupported Sumifs Usage (Help Reference: SUMIFS)
The solver has found cells containing SUMIFS() functions with syntax not supported by What'sBest. The criteria can be single cells containing a number, simple cell reference, or text. Expressions or functions are not permitted. The sum range and criteria ranges must be cell ranges, with the same shape and size.

Suggestions:
The format of *SUMIFS()* function is:
SUMIFS(sum_range, criteria_range1, criteria1, [criteria_range2, criteria2], ...).
What'sBest! places the following restrictions on arguments of this function:

- The *criteria* can be a number, simple cell reference, or text. *What'sBest!* does not support *SUMIFS* functions that contain mathematical expressions or Excel® function for the *criteria* arguments.
- The *sum_range* and *criteria_range* arguments must be cell ranges, with the same shape and size.

TEMPFILE - Error Managing Temp Files

What's*Best!* makes use of a number of temporary files when solving your model. If *What'sBest!* encounters a problem managing its temporary files, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Error Managing Temporary Files (Help Reference: TEMPFILE):
An error occurred while attempting to access temporary work files.
Please delete the following files from the folder containing your
workbook: LINDOWBS.XLS, LINDOWBX.XLS, LINDOWBS.XLSB, LINDOWBX, LINDOWBRC.TXT,
LINDOWBSOLN.TXT, LINDOWBSTATUS.PRN, LINDOWBSOLN.PRN, LINDOWBSTOC.PRN,
LINDOWBSTOH.PRN. Also, verify the access rights on this folder to create files. The folder path, name should be shorter, and may not contain any symbols.

Suggestions:
This error occurs when temporary work files used by *What'sBest!* could not be accessed. Go to the folder where your model workbook is located and remove the files listed above. You should then be able to solve your model.

UNBOUNDED - Unbounded Model

If, without violating any constraints, the value of the cell to be maximized can be increased to infinity, or the value of the minimized best cell can be decreased to negative infinity, then the problem is said to be unbounded. In this case, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Unbounded Model (Help Reference: UNBOUNDED)
The value of the cell to be maximized/minimized can be increased/decreased without limit, and without violating a constraint. The problem may be incorrectly formulated. Be sure that the best cell has been correctly specified and all equations and constraints have been correctly formulated. Refer to Help at 'Solution Status: Unbounded'.

Suggestions:
Check the formulation to ensure that the adjustable cells, the best cell, and the constraint cells have been properly specified and no constraints have been left out. The returned worksheet will display a large positive or negative number in the best cell, and may have large numbers in one or more adjustable cells. This information can give you an indication of where formulation errors lie. An incorrectly specified best cell or constraint (i.e., Maximizing a cell that should be minimized or incorrectly specifying a greater-than-or-equal-to constraint as a less-than-or-equal-to constraint) may be causing the problem.

UNDEFREF - Undefined Reference

If *What'sBest!* encounters formulas with *#REF!* type Excel® errors, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Undefined Reference (Help Reference: UNDEFREF)
An invalid cell reference or an invalid output value was found in the formula of the cell listed below. You must correct any illegal references to continue. Running the Update Links command from the WB|Options|General menu may correct the error.
-cell address-

Suggestions:
This error can occur if a cell referenced by another cell in the worksheet is deleted or replaced by moving another cell over it. This error can also occur when a file is opened that contains functions created on a system in which the *What'sBest!* program files were in a different location. The latter problem can be corrected using the *Update Links* button on the *General Options* dialog box posted by the *Options...|General* command. If this is not the result of links needing to be updated, the references in the specified cells need to be corrected to eliminate all *#REF!* errors.

WBALLDIFFFORM – All-different Cell Format

If requested, *What'sBest!* can specify All-different sets. Range values tell you over what range a particular set is valid. The *=WBALLDIFF()* function is used to request All-different ranges. If the arguments to this function are not correctly specified, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Semi-continuous Cell Format (Help Reference: WBALLDIFFFORM):
A All-different cell is incorrectly formatted. Correct the formula and the validity of the arguments in the cell below. The format should be '=WBALLDIFF(lower, upper, cells)', where 'min' and 'max' are constant numbers, and 'cells' must refer to adjustable or variable cells.
-cell address-

Suggestions:
There is an incorrectly formatted *WBALLDIFF* function in the model. A *WBALLDIFF* function cell must use the format: *=WBALLDIFF(lower, upper, cells)*, where *lower* and *upper* are numbers, and *cells* is a reference to the cells you want the all-different on. The cell reference should be to integer, and adjustable or variable cells. For more information on range values refer to section *Integer…|All-different.*

WBCARDFORM - Cardinality Cell Format

If requested, *What'sBest!* can specify Cardinality sets. Range values tell you over what range a particular set is valid. The *=WBCARD()* function is used to request Cardinality ranges. If the arguments to this function are not correctly specified, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****

Cardinality Cell Format (Help Reference: WBCARDFORM):

A Cardinality cell is incorrectly formatted. Correct the formula and the validity of the

arguments in the cell below. The format should be '=WBCARD(number, cells)', where

'number' is numeric, and 'cells' must refer to adjustable or variable cells.

-cell address-

Suggestions:
There is an incorrectly formatted *WBCARD* function in the model. A *WBCARD* function cell must use the format: *=WBCARD(number, cells)*, where *number* is an integer, and *cells* is a reference to the cells you want the CARD on. The cell reference should be to adjustable or variable cells. For more information on range values refer to section: *Integer…|Special Ordered Sets*

WBDUFORM - Dual Cell Format

If requested, *What'sBest!* can return dual solution information. Dual values tell you how how sensitive the final solution is to various parts of the model. The *@WBDUAL()* function is used to request dual values. If the arguments to this function are not correctly specified, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Dual Cell Format (Help Reference: WBDUFORM)
A dual cell is incorrectly formatted. Correct the formula in the cell below. The format should be '=WBDUAL(cell, value)'.
-cell address-

Suggestions:
There is an incorrectly formatted dual cell in the model. A dual cell must use the format: *=WBDUAL(cell, value)*, where *cell* is a reference to the cell you want the dual value on and *value* is a numeric quantity. The cell reference should be to either a constraint cell or an adjustable cell. You may use any numeric value for *value*; *What'sBest!* will replace *value* with the actual dual value the next time you run the solver. For more information on using dual values refer to section *Advanced|Dual*.

WBPOSDFORM - Semi-definite Positive Cell Format

What'sBest! can specify Semi-Definite Positive sets. To allow the use of the Semi-Definite Positive (SDP) capability with the LINDO Solver, specify the selection of the range matrix to be constrained by a Positive Semi-Definite function (POSD), as *=WBPOSD(range)*. If the arguments to this function are not correctly specified, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Semi-definite Positive Cell Format (Help Reference: WBPOSDFORM):
An Semi-definite Positive POSD function cell is incorrectly formatted. Correct the formula and the validity
of the arguments in the cell below. The format should be '=WBPOSD(matrix)', where the selected 'matrix' should
be symmetric, and must refer to free adjustable or variable cells.
-cell address-

Suggestions:
There is an incorrectly formatted *WBPOSD* function in the model. A *WBPOSD* function cell must use the format: *=WBPOSD(matrix)*, where *matrix* is symmetric, and a reference to the cells you want the Semi-definite Positive on. The cell references should be to free, and adjustable or variable cells. For more information on range values refer to section Constraints.

WBLOFORM - Lower Range Cell Format

If requested, What's*Best!* can return ranging information. Range values tell you over what range a particular dual value is valid. The *@WBLOWER()* function is used to request lower ranges. If the arguments to this function are not correctly specified, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Lower Range Cell Format (Help Reference: WBLOFORM)
A lower range cell is incorrectly formatted. Correct the formula in the cell below. The format should be '=WBLOWER(cell, value)'.
-cell address-

Suggestions:
There is an incorrectly formatted lower range cell in the model. A lower range cell must use the format: *=WBLOWER(cell, value)*, where *cell* is a reference to the cell you want the range value on and *value* is a numeric quantity. The cell reference should be to either a constraint cell or an adjustable cell. You may use any numeric value for *value*; *What'sBest!* will replace *value* with the actual lower range value the next time you run the solver. For more information on using range values refer to section *Advanced|Dual.*

WBSEMICFORM - Semi-continuous Cell Format

If requested, *What'sBest!* can specify Semi-continuous sets. Range values tell you over what range a particular set is valid. The *=WBSEMIC()* function is used to request Semi-continuous ranges. If the arguments to this function are not correctly specified, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****

Semi-continuous Cell Format (Help Reference: WBSEMICFORM):

A Semi-continuous cell is incorrectly formatted. Correct the formula and the validity of

the arguments in the cell below. The format should be '=WBSEMIC(lower, upper, cells)', where

'min' and 'max' are constant numbers, and 'cells' must refer to adjustable or variable cells.

-cell address-

Suggestions:
There is an incorrectly formatted WBSEMIC function in the model. A WBSEMIC function cell must use the format: *=WBSEMIC(lower, upper, cells)*, where *lower* and *upper* are numbers, and *cells* is a reference to the cells you want the semic-continuous on. The cell reference should be to adjustable or variable cells. For more information on range values refer to section *Integer…|Semi-continuous.*

WBSOSFORM - Special Ordered Sets Cell Format

If requested, *What'sBest!* can specify Special Ordered Sets. Range values tell you over what range a particular set is valid. The *=WBSOS1()*, *=WBSOS2()*, or *=WBSOS3()* functions are used to request SOS ranges. If the arguments to this function are not correctly specified, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Special Ordered Sets Cell Format (Help Reference: WBSOSFORM):
A SOS cell is incorrectly formatted. Correct the formula and the validity of the
arguments in the cell below. The format should be '=WBSOS1(cells)', where
'cells' must refer to adjustable or variable cells.
-cell address-

Suggestions:
There is an incorrectly formatted WBSOSx function in the model. A WBSOSx function cell must use the format: *=WBSOS1(cells)*, where *cells* is a reference to the cells you want the SOS on. The cell reference should be to adjustable or variable cells. For more information on range values refer to section *Integer…|Special Ordered.*

WBUPFORM - Upper Range Cell Format

If requested, What'*sBest!* can return ranging information. Range values tell you over what range a particular dual value is valid. The *@WBUPPER()* function is used to request upper ranges. If the arguments to this function are not correctly specified, the following error message will be displayed on the *WB! Status* tab.

Error Message:
****ERROR****
Upper Range Cell Format (Help Reference: WBUPFORM)
An upperr range cell is incorrectly formatted. Correct the formula in the cell below.
The format should be '=WBUPPER(cell, value)'.
-cell address-

Suggestions:
There is an incorrectly formatted upper range cell in the model. An upper range cell must use the format: *=WBUPPER(cell, value)*, where *cell* is a reference to the cell you want the range value on and *value* is a numeric quantity. The cell reference should be to either a constraint cell or an adjustable cell. You may use any numeric value for *value*; *What'sBest!* will replace *value* with the actual lower range value the next time you run the solver. For more information on range values refer to section *Advanced|Dual.*

Appendix A: Installation Details

Installation

There are two options to install *What'sBest!* (before installation, please close all applications, Excel® in particular):

1. WB-WINDOWS-17.0.exe: This option is more convenient than using WB-WINDOWS-17.0.zip as the files are unzipped automatically. The first step is running **WB-WINDOWS-17.0.exe**. Then, press the **Unzip** button to unzip the files. In order to unzip the files to a non-default folder, specify the path to that folder before pressing the unzip button.

2. WB-WINDOWS-17.0.zip: This method may be of use if anti-virus software on your computer makes it difficult to run a .exe file. The first step is unzipping the files to an arbitrary folder. Then, open the extracted folder and run **SetupInstall.exe**.

The next steps are the same in both options. If your computer has a typical configuration, you should be able to simply follow the prompts, clicking "Next." For a non-typical configuration, the following is a description of the screens that will be seen in the setup program. Note that the screen entitled *Setup Type* will give a choice between a *Default* or *Specified* installation. If for some reason you are prompted with a screen not described here, follow the directions on it as it has the latest information available.

What'sBest! Example Files

Example files can be installed on a local drive or any drive on the network in which you have write privileges. It is important that What's*Best!* users avoid sharing a directory for Excel® data files. If What's*Best!* users share a common data directory over the network, the data files of one user could be inadvertently overwritten by the data files of another user. If this installation is for multiple users, it is recommended that the example files (**.XLS*) be copied to a private data directory for each user.

Setup Type

When prompted for setup type of *Default* or *Specified*, choosing *Specified* installation will ensure that you are prompted to confirm the location of the version of Excel® you are installing with. It also prompts you for the location of the What's*Best!* program files, which can be on a local drive or any drive on the network in which you have write privileges.

Users who wish to accomplish the following:

- Installing What's*Best!* on a network,

- Installing on systems with multiple versions of Excel®, or
- Operating a non-English version of Windows®.

will need to select a *Specified* installation.

If Excel® is installed on a network server, the What's*Best!* add-ins files should be installed and run locally. If you have a network license or site license, then you can follow the instructions in this section to install What's*Best!* to the server and set up directories for each node. If you have questions regarding installation of What's*Best!*, the type of license you have, or the number of copies licensed to you, please call LINDO Systems.

It is important that What's*Best!* users avoid sharing a directory for spreadsheet data files. When solving a model, What's*Best!* saves a copy of the model and creates a copy of the solved model under the same file names each time. If What's*Best!* users share a common data directory over the network, the data files of one user could be inadvertently overwritten by the data files of another user.

What'sBest! Add-in Files

The setup program will attempt to locate Excel® and find a *Library* folder in the dialog box, but it is possible it may fail or will find a version of Excel® different from the desired one. Therefore, it is important that you confirm that the supplied path is correct and otherwise correct it.

Location of the Add-In Files

This provides you with the choice of where the add-in files are to be located. The recommended choice of the *Library* subdirectory is already selected. See the section entitled *Location of the Add-in Files and Update Links* for details.

Final Instructions

This gives some important instructions to complete the installation of What's*Best!*. When What's*Best!* setup is almost complete, click *Finish* and Excel® will open. You may receive a message that WBINTR.XLS contains macros. You must then select the *ENABLE MACROS* button in order to finish installation. Also, you can choose to view the README file.

Add-ins, library and executable files from LINDO Systems are digitally signed.

Installation Related Problems

Most errors during installation result in clear on-screen messages and instructions for remedying the errors.

If, for instance, you specify installation to a non-existent directory, the setup program asks you if you want to create such a directory now or quit installation. Missing spreadsheet software, found configuration files, and incorrect spreadsheet releases are all handled similarly.

If, after following all on-screen instructions and referring to *Troubleshooting*, you still have trouble installing What's*Best!*, please call LINDO Systems.

Settings for Microsoft® Excel® 2007, 2010, 2013, 2016, 2019 / Office365®

Via the Ribbon, select:

File/Office Button

Excel options

Popular

- Show Developer tab in the Ribbon

File/Office® Button

Excel options

Manage Excel Add-ins Go

- Check _What's*Best!*

- or Browse to WBA.XLAM, usually in:

"C:\Program Files\Microsoft Office 16\root\Office16\Library\LINDOWB"

File/Office® Button

Excel options

Trust Center

Trust Center Settings

Macro Settings

- Enable all macros (possibly)

- Trusted Publishers to LINDO Systems Inc

Settings for Microsoft® Excel® 1997, 2000, 2002, 2003

Via the Menubar, select:

Tools

Add-ins

- Check _What's*Best!*

- or Browse to WBA.XLA, usually in "C:\Program Files\Microsoft Office\Office11\Library\LINDOWB"

Tools

Customize

Toolbars

- Check _What's*Best!*

Tools

Options

Security

Macro Security

- Security Level to medium
- Trusted Publishers to LINDO Systems Inc

Note : A previous add-in link may be remaining under the name "wba" that should be deleted

Add-ins

If you wish to disable (remove) the What's*Best!* add-in and menu, you may do so by going to the Excel® menu's *Tools|Add-ins* dialog. You can then uncheck the What's*Best!* add-in from the list, which disables the add-in while you are still in Excel®. When you reopen Excel®, the *WB!* menu item will not be present. The What's*Best!* add-in and its menu are reenabled by checking the What's*Best!* add-in from the list posted by the *Tools|Add-Ins* dialog. If What's*Best!* does not appear in the current list of add-ins, the *browse* button can be used to locate it. The add-in file *WBA.XLA, or WBA.XLAM,* can be found in the *Library* subdirectory or the *WB* subdirectory on your C: drive (or elsewhere if you so requested). Details are described in *Location of the Add-in Files and Update Links*. (Please note that it is not possible to disable the What's*Best!* menu if the add-in is installed to the *XLSTART* subdirectory).

The What's*Best!* functions are only available when the What's*Best!* add-in is loaded. If you disable (remove) the What's*Best!* add-in from Excel®, then you must manually re-enable it the next time you wish to build or solve a model.

Note 1: Starting with What's*Best!* release 6.0.1, the add-in name is now WBA.XLA (instead of the former WB.XLA name).
Starting What's*Best!* release 10.0.0.0, the add-in name can be either WBA.XLA for Excel® 2002, or WBA.XLAM for Excel® 2007 and later.

Note 2: Add-ins, library and executable files from LINDO Systems are digitally signed. The certificate name of the digital signature is "LINDO Systems Inc".

Location of the Add-In Files and Update Links

If you have already installed What's*Best!*, then you can check the location of the add-in files (WBA.XLA or WBA.XLAM) at the bottom of the *About What'sBest!* dialog box. The installation program will detect if a previous release of What's*Best!* has been installed. Thus, it will ask if you wish to remove it first or want to write over it.

Location of the Add-in Files and Update Links

Your choice of the Default installation option during the setup program will automatically transfer the program files to the Library subdirectory of your main Excel® directory, usually C:\Program Files\Microsoft Office\Office10\Library\LindoWB for Excel® 2002, or in C:\Program Files\Microsoft Office 16\root\Office16\Library\LindoWB for Excel® 2016. You will find the following files:

CONOPTx.DLL
LIBIOMP5MD.DLL

LINDOx_0.DLL
LINDOCU_xx.DLL
LINDOPRx.DLL
LINDOWBEF.DLL
LINDOWBIL.DLL
LNDWBxxx.LIC
MOSEKx_0.DLL
README.WRI
VCREDIST_x86.EXE
WBA.XLA (Excel® 2003) or WBA.XLAM (Excel® 2007 and later)
WBINTR.XLS
WBOPT.DLL
WBOPTLINK.EXE
WBUNCHADD.EXE

The Help file will be stored in the same directory:
LINDOWBHELP.CHM

Finally, sample workbooks will be installed in C:\WB.

The 64-bit files are identified with the '64' inside the file name.

The Specified installation differs from the Default option only with respect to placement of the add-in files. It first check if a Library subdirectory already exists, but then the setup program lets you choose the most convenient place for locating the What's*Best!* add-in files. As a reminder, you might consider the following folders for What's*Best!* add-in files:

LIBRARY - the Library subdirectory of your main Excel® directory, usually a path ending in "C:\Program Files\Microsoft Office 16\root\Office16\Library\LindoWB". This LindoWB subdirectory is the recommended location for installing the What's*Best!* program files.

XLSTART - the XLSTART subdirectory of your main Excel® directory. Unlike using other location options, if installed to the XLSTART directory, the What's*Best!* menu cannot be disabled through the dialog box posted by the Excel® menu command Tools|Add-ins.

WB - the What's*Best!* directory where the sample files are installed.

Note: If you are running Excel® from a server, verify if you have write access to Excel®'s *Library* or *XLSTART* subdirectories.

In the Windows® Registry, What's*Best!* creates two keys in the following locations:

"HKEY_CURRENT_USER-Software-LINDO Systems, Inc." for CheckUpdate, Language, and Registration.

Note: When you re-enable What's*Best!*, it appears on the list given by *Tools|Add-ins…* if the add-in is installed to the *Library* directory. If it is not on the list, you may use the *Browse* button on the *Add-ins* dialog box to specify the location of the *WBA.XLA* or WBA.XLAM add-in file in the *WB* directory.

Here is a list of the default location of the \LindoWB folder:

Excel® 2002: "C:\Program Files\Microsoft Office\Office10\Library\LindoWB"
Excel® 2003: "C:\Program Files\Microsoft Office\Office11\Library\LindoWB"
Excel® 2007: "C:\Program Files\Microsoft Office\Office12\Library\LindoWB"
Excel® 2010: "C:\Program Files\Microsoft Office\Office14\Library\LindoWB"
Excel® 2013: "C:\Program Files\Microsoft Office 15\root\Office15\Library\LindoWB"
Excel® 2016 / 2019 / 365: "C:\Program Files\Microsoft Office 16\root\Office16\Library\LindoWB"
Where a 32 or 64-bit can start as"C:\Program Files\...", but a 32-bit Office® on a 64-bit system would be "C:\Program Files (x86)\...".

Location of the Registry Keys and Product Codes

Keys created by What's*Best!*®:

HKEY_CURRENT_USER
  Software
   LINDO Systems, Inc.
    CheckUpdate
     - WhatsBestCheckUpdate
    Language
     - WhatsBestLanguage
    Registration
     - WhatsBestSerialNumber/WBPC number

The program also looks at HKEY_LOCAL_MACHINE.

A main Key created by the InstallShield® script:

HKEY_CLASSES_ROOT
HKEY_LOCAL_MACHINE
  Look (search) at the registry and the generated folder keys like
   - 32-bit
        Windows® Product code 34F1A1BE25441C049BF703345E07EE2F
        with InstallShield® Product Code {EB1A1F43-4452-40C1-B97F-3043E570EEF2}
  or
  - 64 bit
        Windows® product code 6B15EA4F6F186EF4EBFD44A9E4A6F5F3

with InstallShield® Product Code {F4AE51B6-81F6-4FE6-BEDF-449A4E6A5F3F}

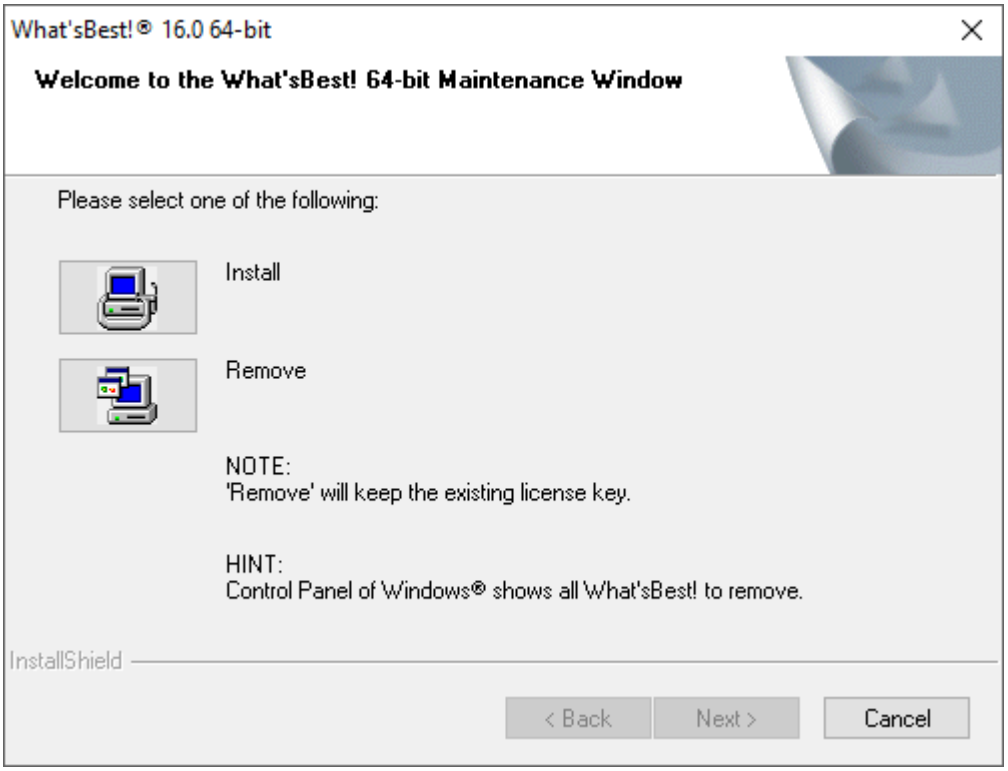
and Wow6432Node for the reflection keys

If these keys or folder keys are altered, the installer will recreate them.

Then run the setup.exe of the What's*Best!* installation you have received, eventually forcing the "multiple copy" which will reset the keys.

Uninstall Files

There are two ways to uninstall What's*Best!*. First, you can rerun the executable file used for the initial installation. Alternatively, you can open the *Control Panel* of your Windows® operating system, select the *Add/Remove Programs* icon, click on What's*Best!* in the list of programs, and click the *Add/Remove* button. Both of these ways will redirect you to a common What's*Best!* maintenance window that resembles the following:



Simply click on the *Remove Only* button. Then, a dialog box will appear to confirm your choice. By clicking *Yes*, your What's*Best!* add-in files will be completely removed. Selecting *No* and then *Cancel* will close the uninstall program.

After uninstalling the software, you might want to verify that Excel® has cleaned up its own lists (e.g., *Toolbar* and *Add-ins* available). To do this, simply open Excel®, and choose the *Tools|Add-ins…*command or the Office® Button, then *Options|Add-ins*. If the What's*Best!* add-in appears in the list of *Add-Ins available*, click on the check box before it. An error message should appear informing you that the *WBA.XLA* or *WBA.XLAM* file can not be found and prompt you to delete it. Thus, click *Yes* and the entry will be removed from the list.

Similarly, to clean up the Excel® list of toolbars, choose the *View|Toolbars|Customize…* command or right click on the Ribbon bar. If the What's*Best!* toolbar appears in the list of *Toolbars:*, select the What's*Best!* toolbar, click the *Delete* button, and click the *OK* button to delete it from the list.

The What's*Best!* add-in can always be reinstalled by launching the installation program again.

Disclaimer

IMPORTANT - Carefully read all the terms and conditions of this agreement before installing this software package. Installing this package indicates your acceptance of these terms and conditions. If you do not accept these terms and conditions, contact LINDO Systems, for instructions on return of this package for a refund.

LINDO Systems License Agreement
Subject to the following terms and conditions, LINDO Systems Inc. (LINDO) hereby grants to you a non-exclusive license to use the LINDO Systems Inc. software program (the "SOFTWARE") contained on the enclosed media and related documentation.

Software License
License
Except as set forth below, LINDO grants to you the right to use the SOFTWARE on any single computer. You may also install a copy for your exclusive use on either a home computer or portable computer. You may store or install a copy of the SOFTWARE on a storage device, such as a network server, used only to install or run the SOFTWARE over an internal network; however, you must acquire and dedicate a license for each individual who will use the SOFTWARE. If the license is installed on a network server or other system that physically allows shared access to the SOFTWARE, you agree to provide technical or procedural methods to prevent use of the SOFTWARE by individuals not specifically licensed to use the SOFTWARE pursuant to this Agreement.

Transfer
The SOFTWARE may be transferred to a single recipient on a permanent basis provided you retain no copies of the SOFTWARE nor documentation (including backup or archival copies) and the recipient agrees to the terms and conditions of this license agreement. At the time of the transfer of the SOFTWARE, you must transfer all media and documentation including any updated media and documentation.

Copyright
The SOFTWARE and its related documentation are copyrighted and protected by US copyright laws and international treaty provisions. You may not use, copy, modify, or transfer the SOFTWARE or related documentation except as expressly provided in the license agreement or with written permission of LINDO Systems Inc.

Restrictions Against Distribution
You may not distribute, lease, sublease, rent, or sublicense the SOFTWARE or related documentation without written permission of LINDO Systems, Inc.

Limited Warranty
LINDO Systems Inc. warrants that the enclosed media and the copy of the related documentation to be free of defects in materials and workmanship for a period of one year from receipt of your payment.

Due to the inherent complexity of computer programs and mathematical models, the SOFTWARE and your mathematical models may not be completely free of errors. You are advised to verify your answers before basing decisions on them. NEITHER LINDO SYSTEMS INC. NOR ANYONE ASSOCIATED WITH THE CREATION, PRODUCTION, OR DISTRIBUTION OF THE SOFTWARE MAKES ANY OTHER EXPRESSED WARRANTIES REGARDING THE MEDIA OR DOCUMENTATION AND MAKES NO WARRANTIES AT ALL, EITHER EXPRESSED OR IMPLIED, REGARDING THE SOFTWARE, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR OTHERWISE

Remedy

LINDO's entire liability and your exclusive remedy for breach of this Limited Warrant shall be, at LINDO's Option, either return of the price paid or replacement of defective media or documentation. In no event shall LINDO Systems Inc. be liable for any damages including but not limited to loss of profit, data, or direct, indirect, special or consequential damages, even if LINDO has been specifically advised of the possibility of such damages.

General

This agreement gives you specific rights, and you may also have other rights that vary from state to state. This License Agreement is governed by, and shall be construed in accordance with, the laws of the State of Illinois.

Trademarks

What's*Best!* and LINDO are registered trademarks of LINDO Systems, Inc. Other product and company names mentioned herein are the property of their respective owners.

Acknowledgement

Portions of LINDO Systems products are based on the independent work of:

LAPACK Users' Guide,
E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra,
J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen.

LAPACK Users' Guide, Society for Industrial and Applied Mathematics, third edition, 1999, Philadelphia, PA.
ISBN 0-89871-447-8.

Appendix B: Contacting LINDO Systems

If you have questions or problems running the software, or you have a suggestion regarding the software or documentation, you can contact LINDO Systems by e-mail, phone, FAX, or mail.

E-mail is often the best means of submitting questions regarding the operation of the software because it allows us to research the questions and provide detailed answers. To resolve a problem running a specific model, we may require that you attach a copy of the model in your email. All models sent to LINDO Systems are treated as strictly confidential.

Sales and Marketing can be reached at:

e-mail:	sales@lindo.com
phone:	(312) 988 7422
FAX:	(312) 988 9065
Mail / Courier:	LINDO Systems, Inc. 1415 North Dayton Street Chicago, IL 60642 -USA-

Tech support is available during business hours: 9:00 AM to 5:00 PM Central Time.

e-mail:	tech@lindo.com
Phone:	(312) 988 9421

General information:

e-mail:	info@lindo.com
phone:	(800) 441 BEST, (800 441 2378)

www.lindo.com

From our website, www.lindo.com, you can download trial versions of any of our optimization products. You can also find a database of sample models and past newsletters to assist you in your modeling efforts. Our website also accommodates online ordering and upgrading information.

Index

#

#REF!, 485

0

0/1 integer variables, 2, 44

A

ABC steps, 9, 12
About What's*Best!*, 138
Absolute, 74
Absolute value function (ABS), 219
Absolute Violation, 68, 190
Absolute Width, 68, 190
Access, 149, 155
Acknowledgements, 2
Active, 40
ADDINLINK, 442
Additional commands, 41
Adjustable button, 21
Adjustable cell, 9, 12
  Dual Value for, 116
Adjustable command, 21
  refers to, 22
Adjustable dialog box, 21, 23
Adjustable-Best-Constraints Steps, 9
Advanced...|Parameters, 128
Advanced…|Dual, 112
Advanced…|Dual…, 112
Advanced…|Function Support, 128
Advanced…|Omit, 126
Advanced…|Omit…, 126
Advanced…|Parameters, 133
Advanced…|Scenario Viewer, 132
Advanced…|Stochastic Support, 130
Advanced…|String Support, 129
Advertising, 347
Algebraic Reformulation, 68
Algorithms, 42
All-different variable, 49
Alpha, 331
American option, 413
And function, 219
Arc, 293
ARITHERR, 443
Asset, 315, 321
ASSIGN.XLSX, 376
Audience, 347

B

Backsolving, 243
Backward analytical, 65
Barrier, 60, 75
Base, 327
Best bound, 78
Best Button, 24
Best cell, 9, 13, 24
Best command, 24
Best dialog box, 24
Best Objective Bound, 40
Beyond VBA, 149
Binary integer variables, 2, 44
Blending Models, 281
BLKCELL, 443
BLOCK.XLSX, 357
Bond portfolio optimization, 301
BONDS.XLSX, 301
Bounding constraints, 245
Box Design, 290
Box Selection, 68
BOX.XLSX, 290
Branching Direction, 68, 74
Budget, 347

C

CALL, 442
callback, 182
Capacity, 343, 367, 392
Car pricing, 343
Cardinality constraint, 46
Cash flow, 301
Cell
  best command, 25
  dual value range, 113
Central Differences, 65
Chance Constrained Modeling, 266
CheckUpdate, 144
Classification Data, 38
Coefficient Reduction, 72
Cold Start, 75
Commercial Version, 139

concave, 28, 33
Concavity, 240
conic programming, 63, 273, 465
Constraint cells, 9
  automatic generation of, 15
  creating, 28
Constraint Cuts, 71
Constraint Ranges, 120
Constraint Related Problems, 32
Constraints, 139
  Button, 28
  command, 28
  dialog box, 28
Constraints..., 28
Contacting Lindo Systems, 503
Container, 402
convert model format, 130
convex, 28, 33
convex,, 239
Convexity, 239
Copyright, 2
Cost, 281, 286, 290, 307, 315, 336, 340, 362, 367, 372, 383, 392, 399
Covariance matrix, 311
Crop Allocation, 406
CROPALLOC.XLSX, 406
CUTSTOCK.XLSX, 362

D

Default, 4
Define Best, 16
Delta, 67
Demand, 350, 362, 399
Depth First, 68, 78
Derivatives, 65
Design, 290
Determine Adjustable cells, 9
Determine Best, 9
Determine Constraints, 9
Devex, 61
Direction, 40
Disaggregation, 72
Disclaimer, 2
Disk space, 3
Distribution, 396
DNRISK.XLSX, 318
Downside risk, 321
Dual, 75
Dual dialog box, 112
Dual Pricing, 61
Dual Value, 113
  of a Constraint Cell, 114
  of an Adjustable cell, 116
  of Zero and Multiple Optima, 119
Dual values, 114
  and Multiple Optima, 119
  in Integer Problems, 119
  in Nonlinear problems, 119
  in Nonlinear Problems, 119
  Valid ranges for, 119

E

Embedding in a Visual Basic project, 149
Engineering Models, 290, 293
Error, 327, 331
Error Codes, 165
Error from VBA code, 440
Error messages
  Iteration Limit, 457, 473
  no feasible solution found, 243
  solution status
    globally optimal, 242
    locally optimal, 237
  Undefined Arithmetic Value, 443
  Unexpected Operation, 467
Error messages & Warnings, 440
  Adjustable Cell Limit, 458
  Barrier Solver Not Licensed, 463
  Constraint Limit, 457
  Dual Cell Format, 486, 487
  Error Managing Temp Files, 484
  Excel File Format, 444
  Expired License Key, 461
  External Reference, 446
  Failed to Access the Add-in, 447
  Failed to Access the Macro, 447
  Failed to Access the Server, 448
  Failed to Process License Key, 460
  Formula Parsing, 445
  Global Adjustable Cell Limit, 460
  Global Solver Not Licensed, 462
  Hidden Solvers Names, 449
  Integer Cell Limit, 458
  Invalid Best Cell, 455
  Invalid Model, 456
  Irreconcilable Constraints, 456
  Iteration Limit Reached, 457
  K-Best Model Format, 450
  Large Infeasibility, 454
  Lower Range Cell Format, 487
  Multiple Best Cells, 466
  Name Cell with Conflicting Properties, 468
  No Adjustable Cells, 451
  No Best Cell, 470

No Constraint Cells, 471
No Feasible Solution Found, 453
Nonlinear Adjustable Cell Limit, 459
Nonlinear Solver Not Licensed, 463
Omitted Cell Reference, 472
Parsing Cell Formula, 467
Pending Expiration of Option Trial, 461
Pending License Expiration, 461
Quadratic Recognition, 472
Runtime Limit Reached, 473
Solver Error, 474
Solver Interrupt, 454
Stochastic Error, 475
String Arguments Found, 481
String List Error, 481
String Result of Formula, 482
Unbounded Model, 484
Undefined Reference, 485
Unsupported Functions, 445
Unsupported Lookup Usage, 465
Unsupported Name, 468
Unsupported Sumif Usage, 482
Unsupported Sumifs Usage, 483
Upper Range Cell Format, 489
Error messages & Warnings
Multiple Add-in Links, 442
Error messages & Warnings
Arithmetic Error, 443
Error messages & Warnings
Blank Cell Warning, 443
Error messages & Warnings
K-Best WBIKB_REP Format, 451
Error messages & Warnings
Indicator Model Cell Reference, 452
Error messages & Warnings
License key Dongle Required, 462
Error messages & Warnings
Mixed Integer Solver Not Licensed, 464
Error messages & Warnings
Stochastic Solver Not Licensed, 464
Error messages & Warnings
Conic Solver Not Licensed, 464
Error messages & Warnings
Stochastic WBSP_CORR Format, 475
Error messages & Warnings
Stochastic WBSP_DIST 1 Format, 476
Error messages & Warnings
Stochastic WBSP_DIST 2 Format, 476
Error messages & Warnings
Stochastic WBSP_DIST 3 Format, 477
Error messages & Warnings
Stochastic Error, 478
Error messages & Warnings
Stochastic WBSP_HIST Format, 478
Error messages & Warnings
Stochastic WBSP_RAND Format, 479
Error messages & Warnings
Stochastic WBSP_REP Foramt, 479
Error messages & Warnings
Stochastic WBSP-STSC Format, 480
Error messages & Warnings
Stochastic WBSP_VAR Format, 480
Error messages & Warnings
Cardinality Cell Format, 486
Error messages & Warnings
Semi-continuous Cell Format, 488
Error messages & Warnings
Special Ordered Sets Cell Format, 488
Estimation Model, 336
Excel, 3, 4, 149, 155
EXLVER, 444
Exponential smoothing, 331
Exposure, 347
Expressions
convex, 239
linear, 235
nonlinear, 236
Nonsmooth, 240
Smooth, 240
Extended Version, 139

F

feasibility tolerance, 53
FIXED1.XLSX, 383
FIXED2.XLSX, 383
Flow, 293, 399
Flow Cover, 72
Flow Network Modeling, 293
FLOWNET.XLSX, 293
Forecasting Models, 327, 331
FORMULA1, 445
FORMULA2, 445
FORMULA3, 446
Forward Analytical, 65
Forward Differences, 65
Free, 23
Deleting designation, 23
variables, 23
FUNCADDIN, 447
FUNCMACRO, 447
FUNCSERVER, 448
Function, 151
Functions, 216–20
convex, 239
linear, 235

nonlinear, 236

G

GCD, 72
General and Operational Problems, 423
General integer variables, 2, 44
General Modeling Guidelines, 244
General Options Dialog Box, 52
Global, 139
Global Distance, 68
Global optima, 242
Global Solver, 66
Global Solver Options Dialog Box, 66
Global Width, 68
Goal seeking, 243
Gomory, 72
GUB, 72

H

Help, 138
Heuristics Level, 70
Hidden cell, 22, 150
Hide model, 150
HIDSNAME, 449
Histogram, 253
HLOOKUP, 219, 465
HOGCHANC.XLSX, 286
HOGFEED.XLS, 281
Hold / Interrupt, 454
Hurdle, 77

I

IF function, 219
IKBMOD, 450
IKBREP, 451
INDICMOD, 452
Industrial Version, 139
Infeasibility, 40
INFEASIBLE, 243, 453
INFLARG, 454
Ingredients, 281
Initial Values, 245
Installation, 4, 491–501
Integer command
  refers to, 43
Integer names in workbook, 43
Integer Pre-Solver Options Dialog Box, 69
Integer problems, 246
  binary, 2, 44
  general, 2, 44
  Solution method in, 73
Integer Solver Options Dialog Box, 73
Integers, 139
Integers…|All-different, 49
Integers…|Integer Binary, 43
Integers…|Semi-continuous, 47
Integers…|Special Ordered Set, 45
Integrality, 74
Interactive environment, 7
INTERRUPT, 454
INVALBEST, 455
Invalid Outside Procedure, 442
INVENT.XLSX, 350
Inventory, 331, 350, 353
Invest, 315, 318
Investment, 321
INVMOD, 456
IRRECONST, 456
Iteration Limit for Slow Progress, 64
ITRLIM, 457

K

K-Best Solutions, 82
Knapsack, 402
Knapsack Cover, 72
Kuhn-Tucker, 242

L

Language, 146
Lattice, 72
Left hand side, 28
LHS, 28
LICCAP1, 457
LICCAP2, 458
LICCAP3, 458
LICCAP4, 459
LICCAP5, 460
License key, 4
License Key, 141
LICKEY1, 460
LICKEY2, 461
LICKEY3, 461
LICKEY4, 461
LICKEY5, 462
LICOPT1, 462
LICOPT2, 463
LICOPT3, 463
LICOPT4, 464
LICOPT5, 464
LICOPT6, 464
Lifting, 72

Lindo Systems Inc., 503
Linear
  expressions, 235
  Formulas, 235
  regression, 336
Linear Solver Options Dialog Box, 60
Linear vs. Nonlinear Expressions and Linearization, 235
linearization, 55
Linearization, 219, 236, 336
LINEARZ.XLSX, 336
Linus Schrage, 2
Load, 402
Local Optima, 237
Local Width, 68
Locally Optimal, 242
Locate…, 137
Location of the Add-in files and Update files, 495
LOCKBOX.XLSX, 307
Locked cell, 22, 150
Logical operators, 216
LOOKUP, 465
Looping Macro, 154
Lower Range, 112, 119
LP Solver, 75

M

Macros, 151
Macros in VBA, 147
Make Adjustable, 21
Make Adjustable & Free or Remove Free, 22, 23
Markowitz Portfolio Model, 311
MARKOWITZ.XLSX, 311
Mathematical functions, 216
MAX function, 219
Max Passes, 72
Maximize Best, 25
MBEST, 466
Media Buying, 347
MEDIA.XLSX, 347
Menu bar, 7
Menu commands, 7
Minimize Best, 25
Minimum and Maximum coefficients, 39
Model Reduction, 60
Model Type, 39
Models
  Non-optimization, 243
  with Integer Restrictions, 2
MODERR, 467
Multi-Period Inventory Management, 350
Multi-period Model, 301
Multiple Objectives, 25
Multiple Optima, 119, 125, 398
Multi-stage, 247
Multistart Attempts, 67

N

NAME, 468
NAMECONFLICT, 468
Network installation, 4
  errors, 396
Network models, 293
Newsvendor, 255
No best cell warning, 243
No Feasible Solution Found, 243
NOADJ, 451
NOBEST, 470
NOCONST, 471
Node, 293
Node Selection, 78
None Best, 25
Nonlinear, 139
  Expressions, 236
  Initial Values, 245
  Magnitude of Model, 245
  Modeling Guidelines, 245
  Models, 64, 242, 245
  Models Starting Point, 64
  Optimization, 237
  Scaling the model, 245
  Solver Options Dialog Box, 62
  Undefined Regions, 245
Non-optimization Models, 2, 243
NOT function, 219
NUMERICAL ERROR, 243

O

Objective, 14
Objective Value, 40
Omit Cells, 181
Omit command
  what not to omit, 127
  what to omit, 127
OMIT names in workbook, 126
OMITTED, 472
Operators, 216–20
Optimality, 67, 76
Optimality conditions, 242
Optimality Tolerance, 64
Optimization, 2
  Models, 2
  Nonlinear, 237

optima, 242
Optimum
Global, 237
Local, 237
Locally, 242
Options and Solvers, 41
Options command, 51
Options...|General, 52
Options...|Global Solver, 66
Options...|Integer Pre-Solver, 69
Options...|Integer Solver, 73
Options...|Linear Solver, 60
Options...|Nonlinear Solver, 62
Options...|Reset to Default, 111
Options…|Stochastic Solver, 89
OR function, 219

P

Parameters Dialog Box, 133
Partial, 61
Personal Version, 139
Pipeline Optimization, 392
PIPELINE.XLSX, 392
Plant Location, 72, 367
PLANTLOC.XLSX, 367
PMIXMAC.XLS, 154
PMIXMAC.XLSM, 353
PORTCOST.XLSX, 315
Portfolio
Minimizing Downside Risk, 318
Scenario Model, 321
Selection, 311
With Transaction Cost, 315
PORTSCEN.XLSX, 321
POSD, 29
POSD-SDP, 29
Positive Definite, 29
Positive Semi-Definite function, 29
Presolve, 63
Pricing, 61
PRICING.XLSX, 343
Primal, 75
Primal Pricing, 61
Probing Level, 70
PRODMIX.XLSX, 353
Product Mix, 353
Professional Version, 139
Profit, 318, 343, 357
Protect Sheet, 22, 29, 150
Put Option, 413
PUTOPTION.XLSX, 413

Q

quadratic cone, 274
Quadratic Recognition, 63
QUAPREC, 472

R

RAM, 3
Ranges, 119
Reasonable Bounding Constraints, 245
Recourse models, 249
Reduced Cost, 116
Refers to, 43
Register, 143
Relationships, 246
Relative, 74
Relative Limit, 72
Relative Violation, 68
Remove adjustable, 22
Restrictions, 246
Return, 301, 315
RHS, 28
Right Hand Side, 28
Risk, 315, 318, 321
Runtime
Concerns, 44
Errors, 441
RUNTIME, 473
runtime limit, 53

S

Sales, 327
SAMPLEWB.XLSX, 340
Sampling, 340
Savage, S., 2
Scale Model, 60
Scaling the Model, 245
Scheduling Model, 372, 376, 383
SDP, 29
SEASON.XLSX, 327
Seasonal Sales Factoring, 327
Selective Constraint Evaluation, 63
Semi-continuous variable, 47
Semi-variance, 321
Set Adjustable cells, 16
Shadow Price, 114
Shipping Cost Reduction, 396
SHIPPING.XLSX, 396
SIMXPO.XLSX, 331
Single objective, 24
SLP Direction, 63

Smooth vs. Nonsmooth Expressions, 240
SMOOTH.XLSX, 331
SOCP, 273
Solution Outcomes, 242
Solution Status, 39, 242
  INFEASIBLE, 243
  Locally Optimal, 242
  NUMERICAL ERROR, 243
  Optimal, 237
  UNBOUNDED, 243
Solution Time, 40
Solve command, 34
Solver decides, 78
Solver Method, 60
Solver Type, 40
SOLVERGOP, 473
SOLVERR, 474
Solving, 34
  Interrupting, 242
  no feasible solution found, 243
  optimality conditions, 242
  Unbounded, 243
Solving Second Order Cone Programs, 273
SOS1, 45
SOS2, 46
SOS3, 46
SPCONFLICT, 475
SPCORR, 475
SPDIST1, 476
SPDIST2, 476
SPDIST3, 477
Special Ordered Set, 45
Specified, 4
Specify Constraints, 16
SPERR, 478
SPHIST, 478
SPRAND, 479
SPREP, 479
SPSTSC, 480
SPVAR, 480
Staff Scheduling, 372
Staff Scheduling Preferred Assignment, 376
Staff Scheduling Two Stage Fixed Shift, 383
STAFF.XLSX, 372
Standard Deviation, 311
Starting What'sBest!, 7
Steepest Edge, 61, 64
Stochastic, 247
Stochastic Monte Carlo Sampling, 264
STRARG, 481
Stratified Sampling, 340
Strictly convex/concave, 240
String Support
  Maximum String Length, 130
STRLIST, 481
Strong Branch, 78
STRRES, 482
Sub or Function not defined, 151, 440
SUMIF, 219, 482
SUMIFS, 483
SUMIFWBSOLVER.XLS, 483
Supported Functions and Operators, 216
System Requirements, 3

T

TEMPFILE, 484
Textbook, 2
The ABC's, 20
The Building Block Method, 357
Time to Relative, 77
Tolerance command, 78
Tolerances, 77
Toolbar, 7
ToolBar command, 140
Tools|Add-ins, 4
Tools|Macros, 151
Tools|References, 151
Trademarks, 2
Traffic Congestion Cost Minimization, 399
TRAFFIC.XLSX, 399
Trend, 327
Trial Version, 139
Tries, 40
Truck Loading, 402
TRUCK.XLSX, 402
Tutorial, 10
  on the VBA Interface, 151
Types, 72

U

UNBOUNDED, 243, 484
Uncertainty, 247
UNDEFREF, 485
Upper Range, 113
Usage Guidelines
  for Dual values, 114
  for Omit, 127
Use of Bound, 67

V

Valid Ranges for Dual Values, 119
Variable Bounds, 67
Variables, 139

integer, 2, 44
Variance, 321
VBA, 149, 151, 155, 175, 181
VBA Interface
Procedures, 155
VBA Interface Tutorial, 151
Visual Basic
Editor, 151
for applications (VBA), 147–214
VLOOKUP, 219

W

Warm Start, 75
Waste Minimization in Stock Cutting, 362
WB formula, 29
WB! Menu, 4, 7, 20
WB! Solution, 37
WB! Status, 37
WB!_Histogram, 59
WB!_Stochastic, 59
wbAddAdjustableStyle, 156
wbAddBestStyle, 156
wbAddWBMenu, 156
wbAdjust, 157
WBALLDIFF, 220
WBALLDIFFFORM, 485
wbBest, 158
WBBIN, 159
WBCARD, 220
WBCARDFORM, 486
wbConstraint, 160
wbConstraintPOSD, 161
wbConvertModelFormat, 162
wbDeleteReports, 162
wbDeleteWBMenu, 163
WBDUAL, 221
WBDUAL Formula, 112
wbDualValue, 163
WBDUFORM, 486
WBERLANG_B, 221
WBERLANG_C, 222
wbError, and Error Codes, 165
WBFREE, 173
WBINNERPRODUCT, 223
WBINT, 174
wbInteger, 175
wbIntegerAllDiff, 176
wbIntegerCard, 177
wbIntegerSemic, 178
wbIntegerSolverKBestReport, 180
wbIntegerSos, 179
WBLMTD, 223
WBLOFORM, 487
WBLOGABEXPX, 224
WBLOGISTIC, 224
WBLOGISTICDENS, 225
WBLOGISTICINV, 225
WBLOGSUMAEXP, 226
WBLOGSUMEXP, 226
WBLOWER, 226
WBLOWER Formula, 112
WBMULTINV, 227
WBNORMSL, 228
WBOMIT, 181
WBPOSD, 29
WBPOSD, 228
WBPOSDFORM, 487
WBProgressInfoRequest, 182
WBQUADPRODUCT, 229
wbResetOptionsToDefault, 184
WBRLMTD, 229
WBSEMIC, 229
WBSEMICFORM, 488
wbSetFunctionSupport, 184
wbSetGeneralOptions, 185
wbSetGlobalOptions, 189
wbSetIntegerOptions, 191
wbSetIntegerPreSolverOptions, 195
wbSetLinearOptions, 198
wbSetNonlinearOptions, 199
wbSetStochasticOptions, 201
wbSetStochasticSupport, 204
wbSetStringSupport, 204
WBSIGNPOWER, 230
wbSolve, 182, 205
WBSOS1, 230
WBSOS2, 231
WBSOS3, 231
WBSOSFORM, 488
WbStochasticChanceConstrained, 207
WbStochasticFunction, 209, 210
wbStochasticReport, 212
wbStochasticStageScenario, 213
WBTRIAINV, 232
WBUNIFINV, 233
wbUpdateLinks, 214
WBUPFORM, 489
WBUPPER, 233
WBUPPER Formula, 112
WBXPOINV, 222
WBXPOWDIVAB, 234
WBxxx, 41
What If projections, 11, 18
Working while Solving, 19
Worst Bound, 68, 78

X

XYZ production problem
adjustable cell ranges, 12
after optimization, 18
best cell, 13
what's best if, 18
XYZ.XLS, 10
XYZVBA.XLSM, 151, 183