```
% LMtestGOP1: Set up and solve the following nonlinear model with LINDO API's
%             global optimizer.
%
%             minimize  f(x) =  x*sin(x*pi) + 10
%             subject to
%                               0 <= x <= 10;
%
% Usage:  LMtestGOP1

% Remarks:
% 1) Model is loaded via LSloadInstruct(). See LINDO-API manual for details.
% 2) Calls (optional) LMcbLocal.m callback function if multi-start solver is used.
%
% Copyright (c) 2001-2002
%
% LINDO Systems, Inc.            312.988.7422
% 1415 North Dayton St.          info@lindo.com
% Chicago, IL 60622              http://www.lindo.com
%
% Last update Sep 02, 2003 (MKA)
%


lindo;
clc;
help LMtestGOP1;

pause (0.01);

% set verbose
verbose = 1;

% Init model size
nCons = 1; % number of constraints
nVars = 1; % number of variables
nObjs = 1; % number of objectives
nNums = 2; % number of real numbers
iObjsen = 1;

% Init bounds and rhs
csense = ['L'];
vtype = ['C'];
adL = [0];
adU = [10];
adX = [5];

% Init instruction list
anConBeg = [];
anConLen = [];
anObjBeg = [];
anObjLen = [];
anIns =[];
nInsLen=[];
anVarNdx = [];
adNumVal = [10; atan(1)*4];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% objective: x*sin(x*pi) + 10
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ikode  = 1;
tmpIns = [EP_PUSH_VAR;  0;  ...
          EP_PUSH_NUM;  1;  ...
          EP_MULTIPLY;      ...
          EP_SIN     ;      ...
          EP_PUSH_VAR;  0;  ...
          EP_MULTIPLY;      ...
          EP_PUSH_NUM;  0;  ...
          EP_PLUS];
```

```
anIns    = [anIns; tmpIns];
anObjBeg = [anObjBeg; ikode];
anObjLen = [anObjLen; length(tmpIns)];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% constraint 1: x - 15 < = 0
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ikode  = ikode + length(tmpIns);
tmpIns = [EP_PUSH_VAR;  0; ...
          EP_PUSH_NUM;  0; ...
          EP_MINUS];

anIns    = [anIns; tmpIns];
anConBeg = [anConBeg; ikode];
anConLen = [anConLen; length(tmpIns)];
ikode    = ikode + length(tmpIns);

% length of instruction list
nInsLen = length(anIns);

if (ikode-1 ~= nInsLen)
   fprintf('Error in instruction lists');
end;
anConBeg = anConBeg -1;
anObjBeg = anObjBeg -1;


% Read license key from a license file
[MY_LICENSE_KEY,nErr] = mxlindo('LSloadLicenseString',MY_LICENSE_FILE);

% Create a LINDO environment
[iEnv,nErr]=mxlindo('LScreateEnv',MY_LICENSE_KEY);
if nErr ~= LSERR_NO_ERROR, return; end;


% Declare and create a model
[iModel,nErr]=mxlindo('LScreateModel',iEnv);
if nErr ~= LSERR_NO_ERROR, return; end;


if verbose > 0
   fprintf('\n Default Nonlinear Optimizer started... \n\n');
end;

% Set callback function to display local solutions (see LMcbLP.m)
if verbose > 0
   fprintf('\n%10s %15s %15s %15s %15s
\n','ITER','PRIMAL_OBJ','DUAL_OBJ','PRIMAL_INF','DUAL_INF');
   [nErr] = mxlindo('LSsetCallback', iModel, 'LMcbLP', 'Dummy string');
end;

% Set NLP print level to 1
[nErr] = mxlindo('LSsetModelIntParameter', iModel, LS_IPARAM_NLP_PRINTLEVEL, 1);

[nErr] = mxlindo('LSsetModelIntParameter', iModel, LS_IPARAM_NLP_PRELEVEL, 126);

% Set NLP solver
[nErr] = mxlindo('LSsetModelIntParameter', iModel, LS_IPARAM_NLP_SOLVER,
LS_NMETHOD_CONOPT);

% Set maximum number of local searches when multistart solver is used
%[nErr] = mxlindo('LSsetModelIntParameter', iModel, LS_IPARAM_NLP_MAXLOCALSEARCH, 1);

% Turn linearization on
[nErr] = mxlindo('LSsetModelIntParameter', iModel, LS_IPARAM_NLP_LINEARZ, 1);
```

```
[nErr] = mxlindo('LSloadInstruct',iModel,nCons,nObjs,nVars,nNums,iObjsen,...
   csense,vtype,anIns,nInsLen,anVarNdx,adNumVal,adX,anObjBeg,anObjLen,...
   anConBeg,anConLen,adL,adU);

%keep the dimension info
[n, nErr] = mxlindo('LSgetInfo',iModel,LS_IINFO_NUM_VARS);
[m, nErr] = mxlindo('LSgetInfo',iModel,LS_IINFO_NUM_CONS);
[LPNz, nErr] = mxlindo('LSgetInfo',iModel,LS_IINFO_NUM_NONZ);
[QCNz, nErr] = mxlindo('LSgetInfo',iModel,LS_IINFO_NUM_QC_NONZ);
[NLPNz, nErr] = mxlindo('LSgetInfo',iModel,LS_IINFO_NUM_NLP_NONZ);
[NLPobjNz, nErr] = mxlindo('LSgetInfo',iModel,LS_IINFO_NUM_NLPOBJ_NONZ);
if nErr ~= LSERR_NO_ERROR, return; end;

% Optimize model
[solStatus,nErr] = mxlindo('LSsolveGOP', iModel);
[solStatus,nErr] = mxlindo('LSgetModelIntParameter',iModel,LS_IPARAM_STATUS);

if (solStatus == LS_STATUS_OPTIMAL)
   if verbose > 0
      fprintf('\n A (global) optimal solution is found ...  \n\n');
   end;

   % Get solution
   [x,nErr]=mxlindo('LSgetPrimalSolution',iModel);
   [obj,nErr]=mxlindo('LSgetObjective',iModel);

   % Report
   fprintf(' Objective = %11.5f\n',obj);
   for i=1:n,
      fprintf(' x[%d]      = %11.5f\n',i,x(i));
   end;
else
   if verbose > 0
      fprintf('\n Optimizer failed....  \n\n');
   end;
end;

% Exit LINDO API
[nErr]=mxlindo('LSdeleteEnv',iEnv);
```