OPTIMUM DESIGN

Dr. / Ahmed Nagib Elmekawy

Lecture 6

Optimum Design with MATLAB

In this I describe the capabilities of the Optimization Toolbox in MATLAB to solve linear, quadratic, and nonlinear programming problems. We start by describing the basic capabilities of this toolbox.
Some operators and syntax used to enter expressions and data are described. In subsequent sections, we illustrate the use of the program for unconstrained and constrained optimization problems. Then some engineering design optimization problems are solved using the program

Optimization Toolbox

The Optimization Toolbox for MATLAB can solve unconstrained and constrained optimization problems.
The Optimization Toolbox must be installed in the computer in addition to the basic MATLAB program before it can be used.

Optimization Toolbox

TABLE 7.1 Optimization Toolbox Functions

Problem type	Formulation	MATLAB function
One-variable minimization in fixed interval	Find $x \in [x_L\ x_U]$ to minimize $f(x)$	`fminbnd`
Unconstrained minimization	Find $\mathbf{x}$ to minimize $f(\mathbf{x})$	`fminunc` `fminsearch`
Constrained minimization: Minimize a function subject to linear inequalities and equalities, nonlinear inequalities and equalities, and bounds on the variables	Find $\mathbf{x}$ to minimize $f(\mathbf{x})$ subject to $\mathbf{Ax} \le \mathbf{b}$, $\mathbf{Nx} = \mathbf{e}$ $g_i(\mathbf{x}) \le 0, i = 1$ to m $h_j = 0, j = 1$ to p $x_{iL} \le x_i \le x_{iU}$	`fmincon`
Linear programming: minimize a linear function subject to linear inequalities and equalities	Find $\mathbf{x}$ to minimize $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x}$ subject to $\mathbf{Ax} \le \mathbf{b}$, $\mathbf{Nx} = \mathbf{e}$	`linprog`
Quadratic programming: Minimize a quadratic function subject to linear inequalities and equalities	Find $\mathbf{x}$ to minimize $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{Hx}$ subject to $\mathbf{Ax} \le \mathbf{b}$, $\mathbf{Nx} = \mathbf{e}$	`quadprog`

UNCONSTRAINED OPTIMUM DESIGN PROBLEMS

Example 7.1

Find x to

Minimize

$$f(x) = 2 - 4x + e^x, -10 \le x \le 10$$

Example 7.1

Find x to

Minimize

$$f(x) = 2 - 4x + e^x, -10 \le x \le 10$$

To solve this problem, we write an m-file that returns the objective function value. Then we invoke fminbnd, the single-variable minimization function, in fixed intervals through another m-file

Example 7.1

```
% All comments start with %
% File name: Example7_1.m
% Problem: minimize f(x) = 2 − 4x + exp(x)
  clear all
% Set lower and upper bound for the design variable
  Lb = -10; Ub = 10;
% Invoke single variable unconstrained optimizer fminbnd;
% The argument ObjFunction7_1 refers to the m-file that
% contains expression for the objective function
  [x,FunVal,ExitFlag,Output] = fminbnd('ObjFunction7_1',Lb,Ub)
```

Example 7.1

```
% File name: ObjFunction7_1.m
% Example 7.1 Single variable unconstrained minimization

function f = ObjFunction7_1(x)
f = 2 - 4*x + exp(x);
```

The output from the function is

x = 1.3863, FunVal = 0.4548, ExitFlag = 1 > 0 (ie, minimum was found),
output = (iterations: 14, funcCount: 14, algorithm: golden section search, parabolic interpolation).

UNCONSTRAINED OPTIMUM DESIGN PROBLEMS

Example 7.2 Multivariable unconstrained Minimization

Consider a two-variable problem:
Minimize

$$f(\mathrm{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \text{ starting from } \mathrm{x}^{(0)} = (-1.2,\ 1.0)$$

Solve the problem using different algorithms available in the Optimization Toolbox.

The syntax for the functions fminsearch and fminunc used to solve a multivariable unconstrained optimization problem is given as follows:

```
[x,FunValue,ExitFlag,Output]=fminsearch('ObjFun',x0,options)

[x,FunValue,ExitFlag,Output]=fminunc('ObjFun',x0,options)
```

where

`ObjFun` = the name of the m-file that returns the function value and its gradient if programmed
`x0` = the starting values of the design variables
`options` = a data structure of parameters that can be used to invoke various conditions for the optimization process

fminsearch uses the Simplex search method of Nelder–Mead, which does not require numerical or analytical gradients of the objective function. Thus it is a nongradient-based method (direct search method) that can be used for problems where the cost function is not differentiable.
fminunc does require the gradient value, with the option LargeScale set to off, it uses the BFGS quasi-Newton method

To solve this problem, we write an m-file that returns the objective function value.Then, the unconstrained minimization function fminsearch or fminunc is invoked through execution of another m-file

Example 7.2

TABLE 7.5 m-File for Unconstrained Optimization Routines for Example 7.2

```
% File name: Example7_2
% Rosenbruck valley function with analytical gradient of
% the objective function
  clear all
  x0 = [-1.2 1.0]';  Set starting values
% Invoke unconstrained optimization routines
% 1. Nelder-Mead simplex method, fminsearch
   % Set options: medium scale problem, maximum number of function evaluations
   % Note that "..." indicates that the text is continued on the next line
     options = optimset('LargeScale', 'off', 'MaxFunEvals', 300);
             [x1, FunValue1, ExitFlag1, Output1] =  ...
              fminsearch ('ObjAndGrad7_2', x0, options)
```

Example 7.2

```
% 2. BFGS method, fminunc, dafault option
  % Set options: medium scale problem, maximum number of function evaluations,
  % gradient of objective function
    options = optimset('LargeScale', 'off', 'MaxFunEvals', 300,…
              'GradObj', 'on');
    [x2, FunValue2, ExitFlag2, Output2] = …
              fminunc ('ObjAndGrad7_2', x0, options)
% 3. DFP method, fminunc, HessUpdate = dfp
  % Set options: medium scale optimization, maximum number of function evaluation,
  % gradient of objective function, DFP method
    options = optimset('LargeScale', 'off', 'MaxFunEvals', 300, …
              'GradObj', 'on', 'HessUpdate', 'dfp');
    [x3, FunValue3, ExitFlag3, Output3] = …
              fminunc ('ObjAndGrad7_2', x0, options)
```

Example 7.2

TABLE 7.6 m-File for Objective Function and Gradient Evaluations for Example 7.2

```
% File name: ObjAndGrad7_2.m
% Rosenbrock valley function
  function [f, df] = ObjAndGrad7_2(x)
% Re-name design variable x
  x1 = x(1); x2 = x(2);  %
% Evaluate objective function
  f = 100*(x2 - x1^2)^2 + (1 - x1)^2;
% Evaluate gradient of the objective function
  df(1) = -400*(x2-x1^2)*x1 - 2*(1-x1);
  df(2) = 200*(x2-x1^2);
```

CONSTRAINED OPTIMUM DESIGN PROBLEMS

The general constrained optimization problem treated by the function **fmincon**.The procedure for invoking this function is the same as for unconstrained problems except that an m-file containing the constraint functions must also be provided. If analytical gradient expressions are programmed in the objective function and constraint functions m-files, they are declared through the options command. Otherwise, fmincon uses numerical gradient calculations based on the finite difference method. E xample 7.3 shows the use of this function for an inequality-constrained problem. E qualities, if present, can be included similarly.

UNCONSTRAINED OPTIMUM DESIGN PROBLEMS

Example 7.3

Solve the problem:

Minimize

$$f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$$

subject to

$$g_1(x) = 100 - (x_1 - 5)^2 - (x_2 - 5)^2 \leq 0$$

$$g_2(x) = -82.81 - (x_1 - 6)^2 - (x_2 - 5)^2 \leq 0$$

$$13 \leq x_1 \leq 100,\ 0 \leq x_2 \leq 100$$

Example 7.3

Find x to

Minimize

$$f(x) = 2 - 4x + e^x, -10 \le x \le 10$$

The script m-file in Table 7.7 invokes the function fmincon with appropriate arguments and options. The function mfile in Table 7.8 contains the cost function and its gradient expressions, and the function m-file in Table 7.9 contains the constraint functions and their gradients.

Example 7.3

TABLE 7.7 m-File for Constrained Minimizer fmincon for Example 7.3

```
% File name: Example7_3
% Constrained minimization with gradient expressions available
% Calls ObjAndGrad7_3 and ConstAndGrad7_3

  clear all

% Set options; medium scale, maximum number of function evaluation,
% gradient of objective function, gradient of constraints, tolerances
% Note that three periods "..." indicate continuation on next line

  options = optimset ('LargeScale', 'off', 'GradObj', 'on',...
  'GradConstr', 'on', 'TolCon', 1e-8, 'TolX', 1e-8);

% Set bounds for variables

  Lb = [13; 0]; Ub = [100; 100];
```

Example 7.3

```
% Set initial design
  x0 = [20.1; 5.84];
% Invoke fmincon; four [ ] indicate no linear constraints in the problem
  [x,FunVal, ExitFlag, Output] = …
  fmincon('ObjAndGrad7_3',x0,[ ],[ ],[ ],[ ],Lb, …
  Ub,'ConstAndGrad7_3',options)
```

Example 7.3

TABLE 7.8 m-File for Objective Function and Gradient Evaluations for Example 7.3

```
% File name: ObjAndGrad7_3.m

  function [f, gf] = ObjAndGrad7_3(x)

% f returns value of objective function; gf returns objective function gradient
% Re-name design variables x

  x1 = x(1); x2 = x(2);

% Evaluate objective function

  f = (x1-10)^3 + (x2-20)^3;

% Compute gradient of objective function

  if nargout > 1
          gf(1,1) = 3*(x1-10)^2;
          gf(2,1) = 3*(x2-20)^2;
  end
```

Example 7.3

TABLE 7.9 Constraint Functions and Their Gradients Evaluation m-File for Example 7.3

```
% File name: ConstAndGrad7_3.m

  function [g, h, gg, gh] = ConstAndGrad7_3(x)

% g returns inequality constraints; h returns equality constraints
% gg returns gradients of inequalities; each column contains a gradient
% gh returns gradients of equalities; each column contains a gradient
% Re-name design variables

  x1 = x(1); x2 = x(2);

% Inequality constraints

  g(1) = 100-(x1-5)^2-(x2-5)^2 ;
  g(2) = -82.81+ (x1-6)^2 + (x2-5)^2;

% Equality constraints (none)

  h = [ ];
```

Example 7.3

```
% Gradients of constraints

if nargout > 2
        gg(1,1) = -2*(x1-5);
        gg(2,1) = -2*(x2-5);
        gg(1,2) = 2*(x1-6);
        gg(2,2) = 2*(x2-5);
        gh = [];
end
```

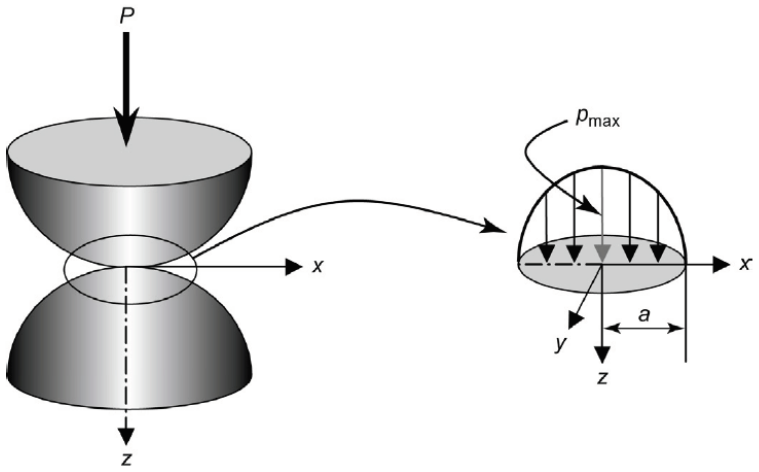
OPTIMUM DESIGN EXAMPLES WITH MATLAB

Location of Maximum Shear Stress for Two Spherical Bodies in Contact

There are many practical applications where two spherical bodies come into contact with each other, as shown in Fig. 7.1. We want to determine the maximum shear stress and its location along the z-axis for a given value of the Poisson's ratio of the material, v = 0.3.

Definition of Design Variables

The only design variable for the problem is the normalized depth for the location of the maximum shear stress α $(= z/a)$.



Optimization Toolbox

The Optimization Toolbox for MATLAB can solve unconstrained and constrained optimization problems.
The Optimization Toolbox must be installed in the computer in addition to the basic MATLAB program before it can be used.