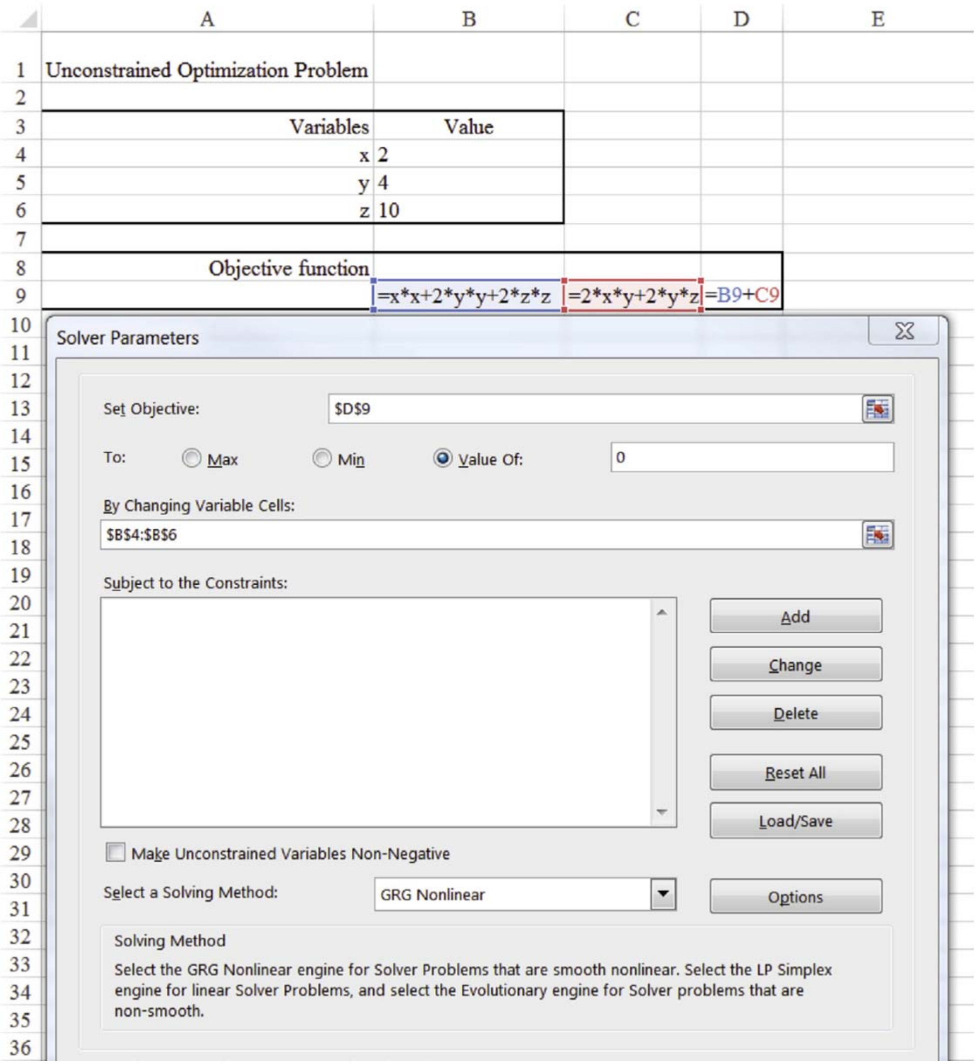
EXCEL vs. MATLAB

	A	B	C	D	E
1	solution of KKT conditions				
2	variables				
3	x1	1			1.732051
4	x2	-2			1.732051
5	u	2			0.5
6	s	0			0
7	equations				
8	2*x1-3*x2+2*u*x1	12	=	0	-2.5E-07
9	-3*x1+2*x2+2*u*x2	-15	=	0	5.48E-08
10	x1^2+x2^2-6+s^2	-1	=	0	-1.2E-07
11	u*s	0	=	0	0
12	s^2	0	>=	0	0
13	u	2	>=	0	0.5

```
Function F=kktsystem(x)
F=[2*x(1)-3*x(2)+2*x(3)*x(1);
2*x(2)-3*x(1)+2*x(3)*x(2);
x(1)^2+x(2)^2-6+x(4)^2;
x(3)*x(4)];
x0=[1;1;1;1];
options=optimset('Display','iter')
x=fsolve(@kktsystem,x0,options)
```

Unconstrained Optimization Problem

$$\underset{x,y,z}{\text{Min}}\ f\left(x,y,z\right)=x^2+2y^2+2z^2+2xy+2yz$$



Linear Programming Problem

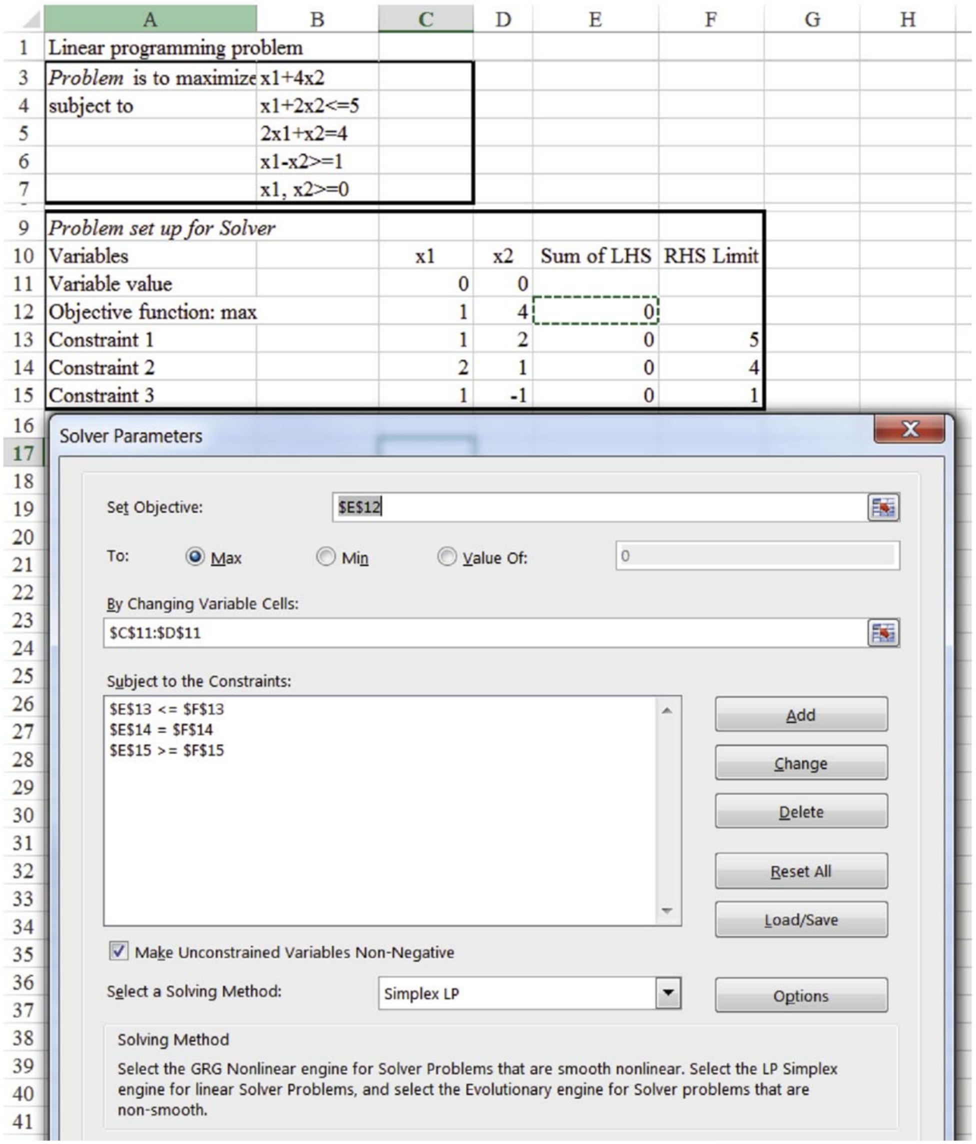
$$\underset{\mathbf{x}}{\text{Maximize}} \quad z = x_1 + 4x_2$$

$$\text{subject to} \quad x_1 + 2x_2 \le 5$$

$$2x_1 + x_2 = 4$$

$$x_1 - x_2 \ge 1$$

$$x_1, x_2 \ge 0$$



Answer and Sensitivity Report

Microsoft Excel 15.0 Answer Report
Worksheet: [Figure 6.8(LP)-2015.xlsx]Figure 6.8
Report Created: 7/11/2015 2:51:41 PM
Result: Solver found a solution. All Constraints and optimality conditions are satisfied.
Solver Engine
Engine: Simplex LP
Solution Time: 0 Seconds.
Iterations: 2 Subproblems: 0
Solver Options
Max Time 100 sec, Iterations 100, Precision 0.000001
Solve Without Integer Constraints, Assume NonNegative

Objective Cell (Max)

Cell	Name	Original Value	Final Value
$E$12	Objective function: max Sum of LHS	0	4.333333333

Variable Cells

Cell	Name	Original Value	Final Value	Integer
$C$11	Variable value x1	0	1.666666667	Contin
$D$11	Variable value x2	0	0.666666667	Contin

Constraints

Cell	Name	Cell Value	Formula	Status	Slack
$E$13	Constraint 1 Sum of LHS	3	$E$13<=$F$13	Not Binding	2
$E$14	Constraint 2 Sum of LHS	4	$E$14=$F$14	Binding	0
$E$15	Constraint 3 Sum of LHS	1	$E$15>=$F$15	Binding	0

Microsoft Excel 15.0 Sensitivity Report
Worksheet: [Figure 6.8(LP)-2015.xlsx]Figure 6.8
Report Created: 7/11/2015 2:51:41 PM

Variable Cells

Cell	Name	Final Value	Reduced Cost	Objective Coefficient	Allowable Increase	Allowable Decrease
$C$11	Variable value x1	1.666666667	0	1	7	1E+30
$D$11	Variable value x2	0.666666667	0	4	1E+30	3.5

Constraints

Cell	Name	Final Value	Shadow Price	Constraint R.H. Side	Allowable Increase	Allowable Decrease
$E$13	Constraint 1 Sum of LHS	3	0	5	1E+30	2
$E$14	Constraint 2 Sum of LHS	4	1.666666667	4	2	2
$E$15	Constraint 3 Sum of LHS	1	-2.333333333	1	1	2

Optimum Design with MATLAB

- Introduction to Optimization Toolbox
- Unconstrained optimum design problem
- Constrained optimum design problem
- Optimum design examples
  - Location of maximum shear stress for two spherical bodies in contact
  - Column design for minimum mass
  - Flywheel design for minimum mass

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 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 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`

Syntax

```
[x,FunValue,ExitFlag,Output]=fminX('ObjFun',...,options)
```

Argument	Description
x	The solution vector or matrix found by the optimization function. If `ExitFlag` > 0 then x is a solution, otherwise x is the latest value from the optimization routine.
`FunValue`	Value of the objective function, `ObjFun`, at the solution x.
`ExitFlag`	The exit condition for the optimization function. If `ExitFlag` is positive then the optimization routine converged to a solution x. If `ExitFlag` is zero then the maximum number of function evaluations was reached. If `ExitFlag` is negative then the optimization routine did not converge to a solution.
`Output`	The `Output` structure contains several pieces of information about the optimization process. It provides the number of function evaluations (`Output.iterations`), the name of the algorithm used to solve the problem (`Output.algorithm`), and Lagrange multipliers for constraints, etc.

Unconstrained Optimization: Single-Variable

$$\underset{x}{\text{Min}}\ f\left(x\right)=2-4x+e^{x},\ \ -10\leq x\leq 10$$

```
% 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)
```

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

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

`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 Optimization: Multivariable

$$\underset{\mathbf{x}}{\text{Min}}\ f(\mathbf{x}) = 100\left(x_2 - x_1^2\right)^2 + \left(1 - x_1\right)^2,\ \mathbf{x}^{(0)} = (-1.2, 1.0) \rightarrow \mathbf{x}^* = (1.0, 1.0),\ f\left(\mathbf{x}^*\right) = 0$$

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

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

`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 (see Subsection 11.9.3 for details of this algorithm). Thus it is a *nongradient-based method* (*direct search method*) that can be used for problems where the cost function is not differentiable.

Since `fminunc` does require the gradient value, with the option `LargeScale` set to `off`, it uses the BFGS quasi-Newton method (refer to chapter: More on Numerical Methods for Unconstrained Optimum Design for details) with a mixed quadratic and cubic line search procedure. The DFP formula (refer to chapter: More on Numerical Methods for Unconstrained Optimum Design, for details),

```
% 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)
% 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)
```

```
% 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 Optimization

$$\left.\begin{aligned}
&\underset{\mathbf{x}}{\text{Minimize}} \quad f(\mathbf{x}) = (x_1 - 10)^2 + (x_2 - 20)^2 \\
&\text{subject to} \quad g_1(\mathbf{x}) = 100 - (x_1 - 5)^2 + (x_2 - 5)^2 \le 0 \\
&\qquad\qquad\quad g_2(\mathbf{x}) = -82.81 - (x_1 - 6)^2 + (x_2 - 5)^2 \le 0 \\
&\qquad\qquad\quad 13 \le x_1 \le 100, \;\; 0 \le x_2 \le 100
\end{aligned}\right\} \rightarrow \mathbf{x}^* = (14.095, 0.84296), \; f(\mathbf{x}^*) = -6961.8$$

`Active constraints`: 5, 6 [ie, g(1) and g(2)]
$\mathbf{x}$ = (14.095,0.843), `FunVal` = −6.9618e + 003, `ExitFlag` = 1 > 0 (ie, minimum was found)
`output` = (iterations: 6, funcCount: 13, stepsize: 1, algorithm: medium scale: SQP, quasi-Newton, line-search).

```
% 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];

% 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)
```

% File name: ObjAndGrad7_3.m

```
function [f, gf] = ObjAndGrad7_3(x)
```

% f returns value of objective function; gf returns objective fun

% 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
```

% 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 = [ ];
```

% 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
```