CSE/ECE 848 Introduction to Evolutionary Computation

Module 1, Lecture 2, Part 4
Existing Point-based Optimization Methods

Kalyanmoy Deb, ECE

Koenig Endowed Chair Professor

Recall Module 1, Lecture 2, Part 1-3

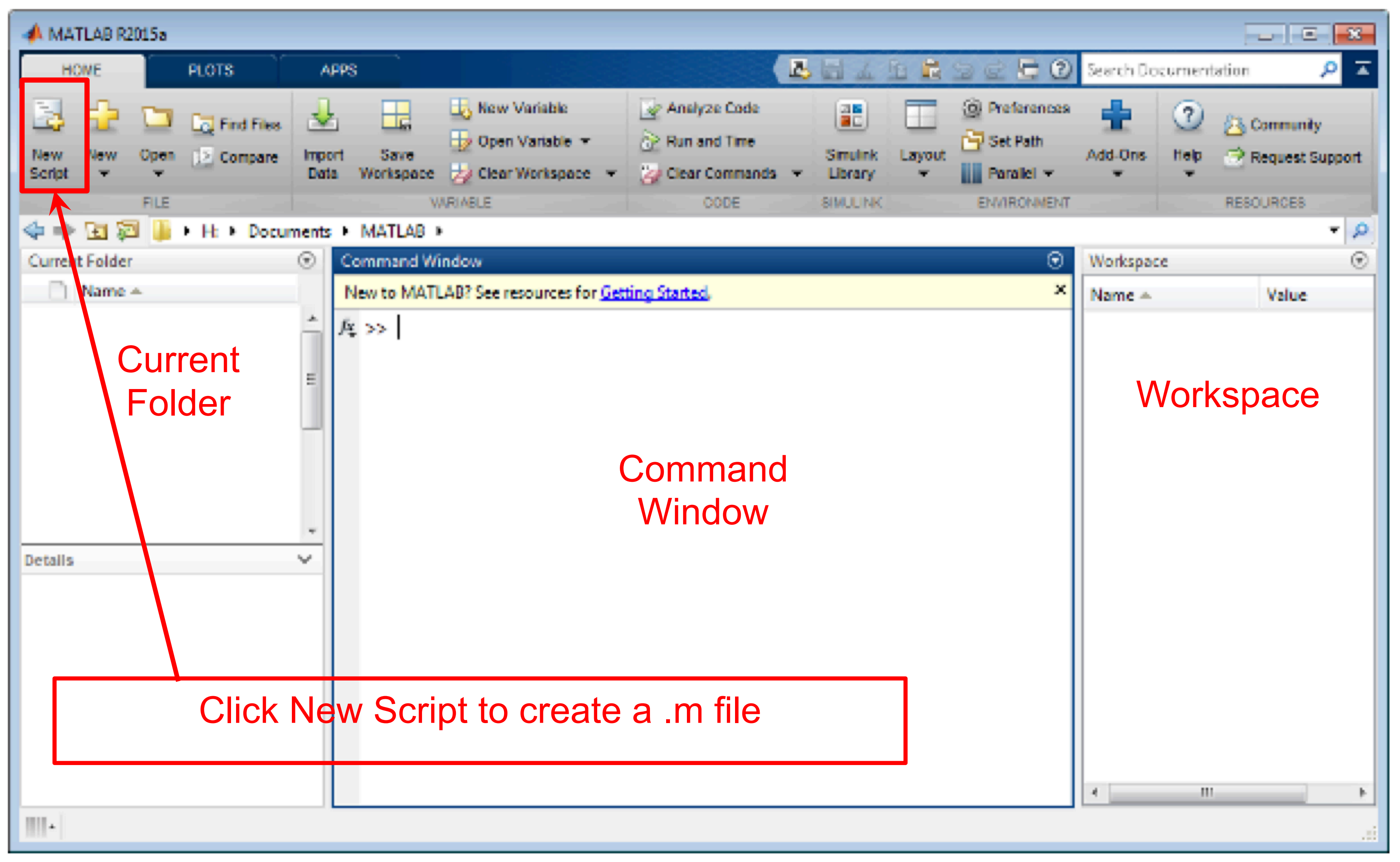
- Point-based Optimization:
  - Single-variable optimization
  - Multi-variable optimization
  - Constrained optimization

- Part 4: Matlab codes and their use
  - A Brief Introduction to Matlab
  - Demonstration of use of optimization routines
    - fminbnd() and fminsearch()
  - Mention of other related Matlab routines

Introduction to Matlab

- MATLAB is a matrix processing software
- It can execute most matrix algebra, such as eigenvalue and determinant calculations
- It has macros to execute various other numerical computations, such as root finding, ODE solution, etc.
- Optimization, too
- It has several tool-boxes for doing advanced computations

Matlab Desktop Basics



How to Start Working in MATLAB?

1. Directly type commands in Command Window. Press enter to execute
2. Open a new script and start typing commands. When done, type RUN

- A row vector is entered: a = [5 1 4];
- A column vector, b = [5; 1; 4];
- Transpose of a is a’ (a column vector)
- 2nd element of a is a(2) (= 1)
- To comment a line, use % in the beginning
- No semicolon at the end will print the value
- On Search Doc, type a command to see details

Matrix Representation in Matlab

- A 2x3 matrix is represented as
  - A = [5 4 3; 6 1 2] = [[5 4 3];[6 1 2]];
  - A = $\begin{bmatrix} 5 & 4 & 3 \\ 6 & 1 & 2 \end{bmatrix}$

```
A =
     5     4     3
     6     1     2
```

- Product of A and b: x = A*b
- x = $\begin{bmatrix} 5 & 4 & 3 \\ 6 & 1 & 2 \end{bmatrix} \begin{Bmatrix} 5 \\ 1 \\ 4 \end{Bmatrix} = \begin{Bmatrix} 41 \\ 39 \end{Bmatrix}$

```
x =
    41
    39
```

- $a .* b$: $[2,3,4] \ .* \ [1,2,0.5] \ = \ [2,6,2]$
- Random #: y = rand(2, 3) produces 2x3 matrix of random numbers within [0, 1]

```
y =
    0.3221    0.5079    0.7588
    0.5376    0.9347    0.9630
```

Example: Linear Equation Solver

- Simultaneous linear equations in matrix:
- $2x_1 + 3x_2 - x_3 = 4$
- $5x_1 - 2x_2 + 2x_3 = 8$
- $x_1 + 2x_2 + x_3 = 10$

$$\begin{bmatrix} 2 & 3 & -1 \\ 5 & -2 & 2 \\ 1 & 2 & 1 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 4 \\ 8 \\ 10 \end{Bmatrix}$$

- A = [2 3 -1; 5 -2 2; 1 2 1]; b = [4; 8; 10];
- To solve *x*-vector, type ***x*** = inv(A)*b
- Results ***x*** =[0.727; 2.364; 4.546]
- Eigenvalues of A: type [**u, v**] = eig(A)
- Results **v** = [-4.886; 4.323; 1.562]

```
x =
    0.7273
    2.3636
    4.5455
```

```
v =
    0.4166   -0.5663   -0.1374
   -0.8800   -0.6197    0.3312
    0.2283   -0.5434    0.9335
```

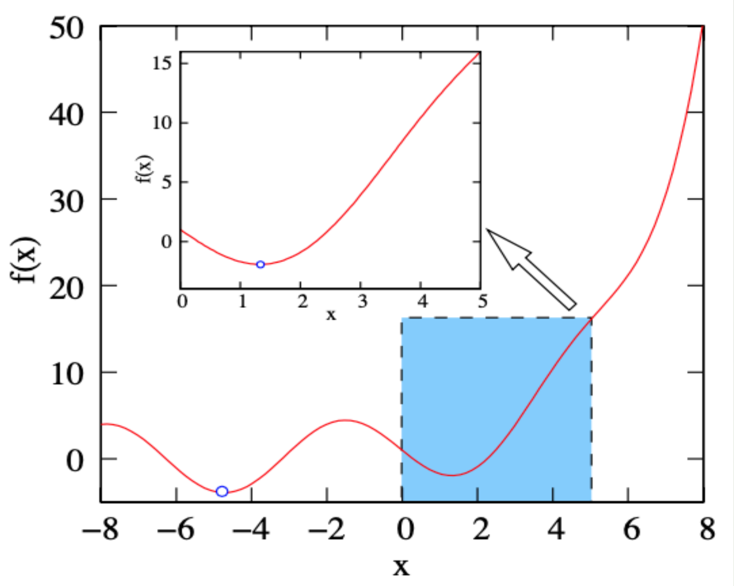
```
u =
   -4.8856         0         0
         0    4.3232         0
         0         0    1.5624
```

Single-variable Minimization using fminbnd()

- $\text{Min}_x\ f(x)$ such that $x1 < x < x2$
- x = fminbnd(fun,x1,x2,options)
- options = optimset('Display','iter');
- x=fminbnd(@(x) exp(0.5*x)-4*sin(x), 0, 5, options)

Func-count	x	f(x)	Procedure
1	1.90983	-1.17386	initial
2	3.09017	4.48277	golden
3	1.18034	-1.89465	golden
4	1.29707	-1.93834	parabolic
5	1.32289	-1.94012	parabolic
6	1.32605	-1.94014	parabolic
7	1.3258	-1.94014	parabolic
8	1.32577	-1.94014	parabolic
9	1.32584	-1.94014	parabolic

Optimization terminated: the current x satisfies the termination criteria using OPTIONS.TolX of 1.000000e-04



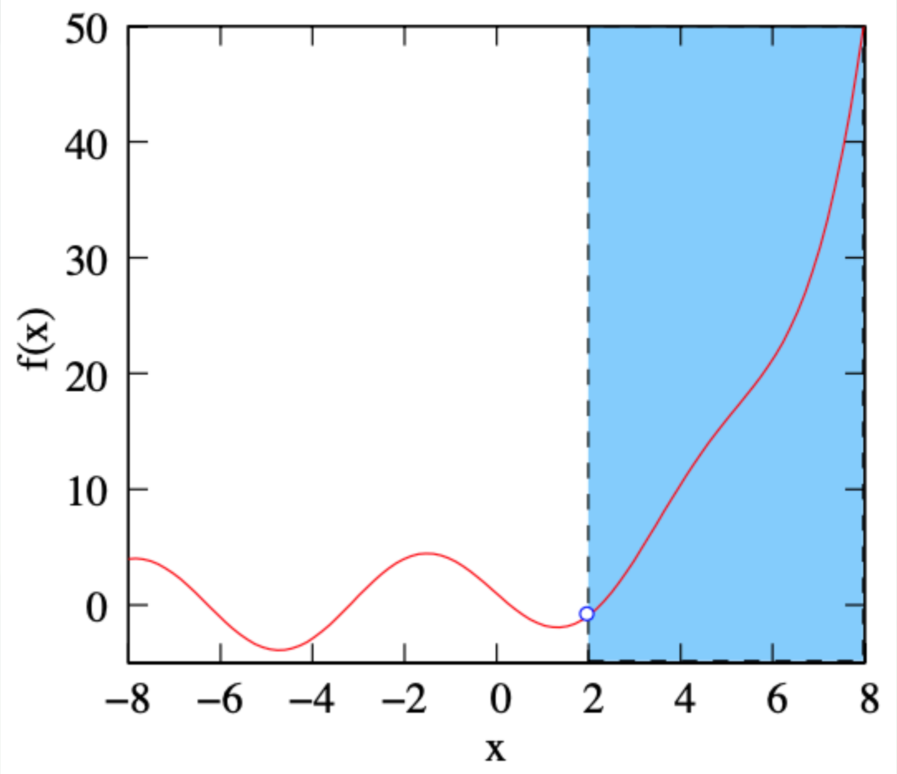
x=fminbnd(@(x)exp(0.5*x)-4*sin(x),-8,8)
x =
1.3258
Starting x_0= -1.889

x=fminbnd(@(x)exp(0.5*x)-4*sin(x),-12,8)
x =
-4.7242
Starting x_0= -4.361

fminbnd() continued

- Can not get the boundary points as minimum
- x=fminbnd(@(x)exp(0.5*x)-4*sin(x),2,8,options)
- Func-count x f(x) Procedure

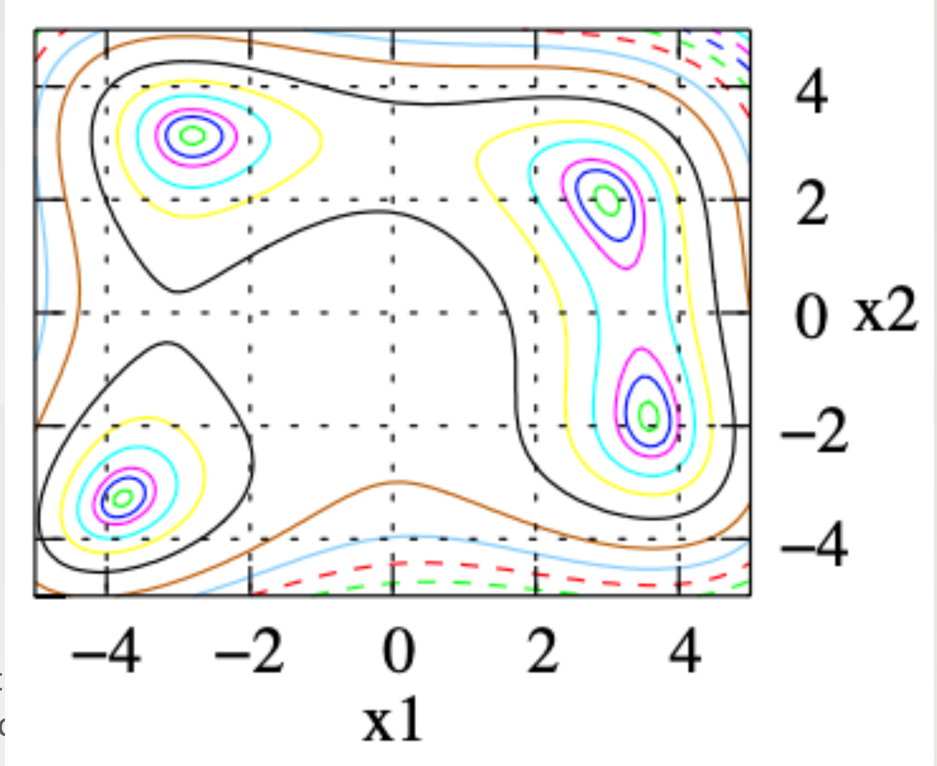
Func-count	x	f(x)	Procedure
1	4.2918	12.2011	initial
2	5.7082	19.5341	golden
3	3.41641	6.60452	golden
4	2.87539	3.15869	golden
5	2.54102	1.30221	golden
…			
20	2.0004	-0.917708	golden
21	2.00025	-0.918167	golden
22	2.00015	-0.91845	golden
23	2.00009	-0.918625	golden
24	2.00006	-0.918733	golden



- $x^* = 2.00006$

Multi-variable Unconstrained Minimization using fminsearch()

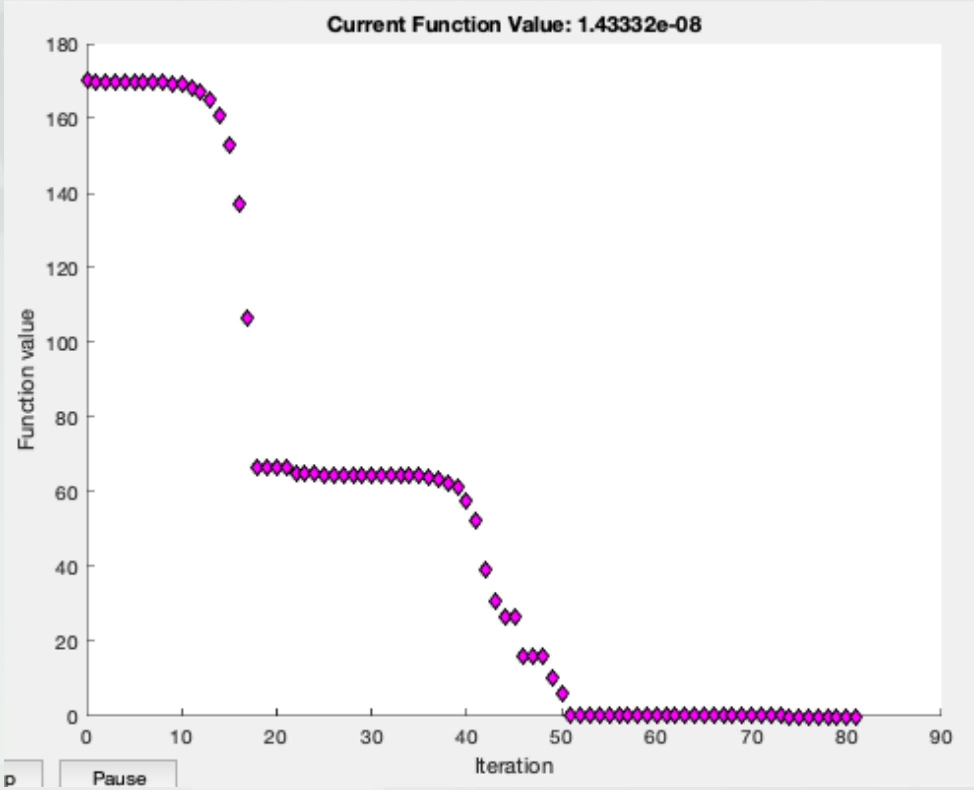
- Min$_x$ $f(x)$, x is a vector or a matrix
- x = fminsearch(fun,x0,options)
- fun = @(x) (x(1)^2+x(2)-11)^2+(x(1)+x(2)^2-7)^2
- options = optimset('PlotFcns',@optimplotfval);
- x0 = [0, 0]
- x=fminsearch(fun,x0,options)
- x = 3.0000   2.0000



Multi-variable Unconstrained Minimization using fminsearch()

- Display of Iterations:

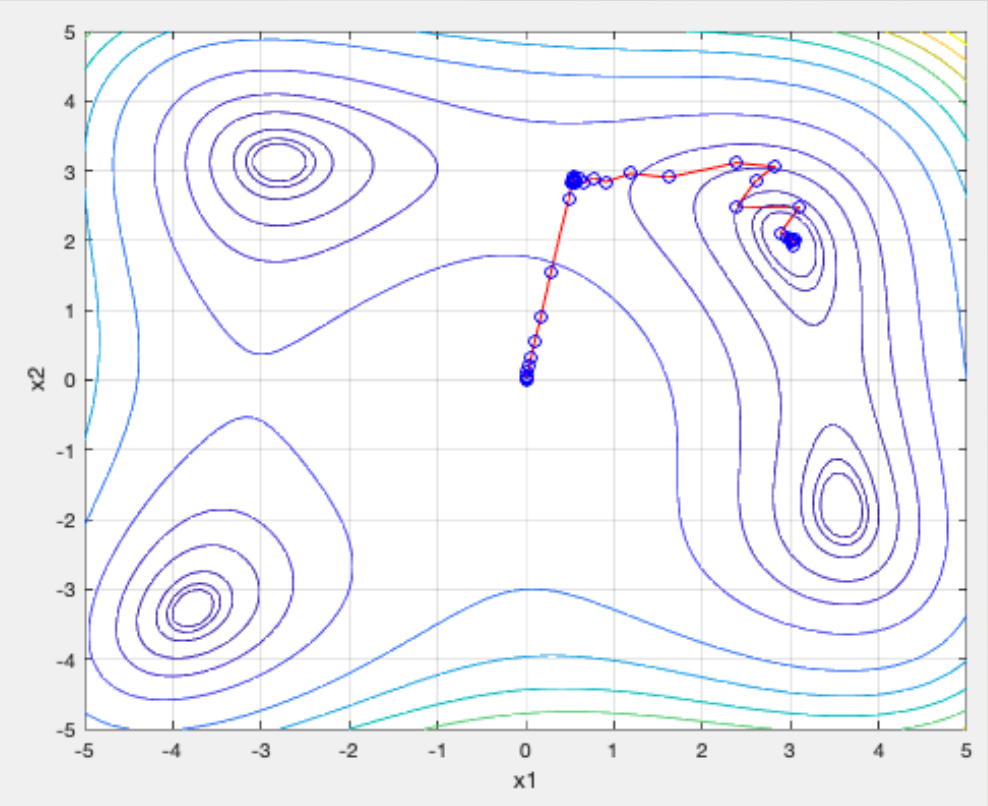
Iteration	Func-count	min f(x)	Procedure
0	1	170	
1	3	169.994	initial simplex
2	5	169.986	expand
3	7	169.978	expand
…			
18	37	66.782	expand
19	38	66.782	reflect
20	40	66.782	contract inside
21	42	66.782	contract inside
22	44	64.7839	contract inside
23	45	64.7839	reflect
24	47	64.7839	contract inside
…			
78	151	7.83007e-08	contract inside
79	153	3.59737e-08	contract inside
80	155	3.59737e-08	contract inside
81	157	1.43332e-08	contract inside



Optimization terminated:
 the current x satisfies the termination criteria using OPTIONS.TolX of 1.000000e-04
 and F(X) satisfies the convergence criteria using OPTIONS.TolFun of 1.000000e-04

fminsearch() with plots in 2D

```
% call myproblem(x0)
[x, fval, history] = myproblem([0,0]),
% plot history of points
plot(history(:,1),history(:,2),'r-'); hold on;
plot(history(:,1),history(:,2),'bo');
grid on; xlabel('x1'); ylabel('x2');
% plot contour
x = linspace(-5,5);
y = linspace(-5,5);
[X,Y] = meshgrid(x,y);
Z = (X.^2+Y-11).^2+(X+Y.^2-7).^2;
v=[800 700 600 500 400 300 200 100 ...
   50 25 10 5 3];
contour(X,Y,Z,v); hold off;
```



myproblem.m file

```
function [x, fval, history] = myproblem(x0)
   history = x0;
   options = optimset('OutputFcn', @myoutput, ...
      'Display','Iter', 'PlotFcns',@optimplotfval);
   [x fval] = fminsearch(@objfun, x0,options);
   function stop = myoutput(x,optimvalues,state);
      stop = false;
      if isequal(state,'iter')
        history = [history; x];
      end
   end
```

```
function z = objfun(x)
    z = (x(1)^2+x(2)-11)^2+(x(1)+x(2)^2-7)^2;
  end
end
```

```
x  =   3.0000    2.0000
fval = 1.4333e-08
```

End of Module 1, Lecture 2, Part 4

- Introduction to Matlab
- Single-variable optimization code **fminbnd()**
- Multi-variable optimization code **fminsearch()**

- Part 5:
  - Constrained optimization code **fmincon()**
  - Structured programming using Matlab
    - Linear programming **linprog()**
    - Integer/Discrete programming **intlinprog()**
    - Quadratic programming **quadprog()**