MATLAB Optimization Toolbox (optimtool)

Dr. Rajesh Kumar

PhD, PDF (NUS, Singapore)
SMIEEE (USA), FIET (UK) FIETE, FIE (I), LMCSI, LMISTE
Professor, Department of Electrical Engineering
Malaviya National Institute of Technology, Jaipur, India,
Mobile: (91)9549654481
rkumar.ee@mnit.ac,in, rkumar.ee@gmail.com
Web:http://drrajeshkumar.wordpress.com/

Contents

- Minimization algorithm
  - fgoalattain
  - fmincon
  - fminimax
  - fminunc
- Equation solving
  - fsolve
  - fseminf
- Linear programming
  - linprog
  - intlinprog
- Least square problems
  - lsqlin
  - lsqnonlin
  - lsqcurvefit (curve fitting)
- Quadratic programming
  - quadprog
- Global Optimization Toolbox
  - ga (genetic algorithm)
  - particaleswarm (Particle swarm optimization)
  - simulannealbnd (simulated annealing algorithm)
  - gamultiobj (multi-objective ga)
  - patternsearch

fgoalattain

- Solve multiobjective goal attainment problems

$$\underset{x,y}{\text{minimize}} \begin{cases} F(x) - \text{weight}.y \le \text{goal} \\ c(\text{x}) \le 0 \\ ceq(x) = 0 \\ A.x \le b \\ Aeq.x = beq \\ lb \le x \le ub \end{cases}$$

fgoalattain

- Solve multiobjective goal attainment problems

$$\underset{x,y}{\text{minimize}} \begin{cases} F(x) - \text{weight}.y \le \text{goal} & \\ c(\mathrm{x}) \le 0 & \text{Nonlinear inequality constraints} \\ ceq(x) = 0 & \text{Nonlinear equality constraints} \\ A.x \le b & \text{Linear inequality constraints} \\ Aeq.x = beq & \text{Linear equality constraints} \\ lb \le x \le ub & \text{Range of x} \end{cases}$$

- Where
  - *weight*, *goal*, *b*, and *beq* are vectors
  - *A* and *Aeq* are matrices
  - *c*(*x*), *ceq*(*x*), and *F*(*x*) are functions that return vectors
  - *F*(*x*), *c*(*x*), and *ceq*(*x*) can be nonlinear functions

fgoalattain

- Example An output feedback controller, K is designed producing a closed loop system

$$\dot{x} = \left(A + BKC\right)x + Bu$$

$$y = Cx$$

With design consideration, close loop poles [-5,-3,-1] and gain -4< K <4

fgoalattain

- <u>Example</u> An output feedback controller, K is designed producing a closed loop system

$$\dot{x} = \left(A + BKC\right)x + Bu$$

$$y = Cx$$

With design consideration, close loop poles [-5,-3,-1] and gain -4< K <4

$$A = \begin{bmatrix} -0.5 & 0 & 0 \\ 0 & -2 & 10 \\ 0 & 1 & -2 \end{bmatrix}; B = \begin{bmatrix} 1 & 0 \\ -2 & 2 \\ 0 & 1 \end{bmatrix}; C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{goal}=\begin{bmatrix} -5 & -3 & -1 \end{bmatrix}; \text{weight} = abs(\text{goal});$$

$$\text{K0} = \begin{bmatrix} -1 & -1 \\ -1 & -1 \end{bmatrix};$$

$$lb = \begin{bmatrix} -4 & -4 \\ -4 & -4 \end{bmatrix}; ub = \begin{bmatrix} 4 & 4 \\ 4 & 4 \end{bmatrix};$$

fgoalattain

- Create function file, eigfun.m.

```
function F = eigfun(K,A,B,C)
F = sort(eig(A+B*K*C)); % Evaluate objectives
```

- Next enter the system matrix and invoke an optimization routine

```
A= [-0.5 0 0; 0 -2 10; 0 1 -2];
B = [1 0; -2 2; 0 1];
C = [1 0 0; 0 0 1];
K0 = [-1 -1; -1 -1]; % Initialize controller matrix
goal = [-5 -3 -1];    % Set goal values for the eigenvalues
weight = abs(goal);   % Set weight for same percentage
lb = -4*ones(size(K0)); % Set lower bounds
ub = 4*ones(size(K0));  % Set upper bounds
```

fgoalattain

- Create function file, eigfun.m.

```
function F = eigfun(K,A,B,C)
F = sort(eig(A+B*K*C)); % Evaluate objectives
```

- Next enter the system matrix and invoke an optimization routine

```
A= [-0.5 0 0; 0 -2 10; 0 1 -2];
B = [1 0; -2 2; 0 1];
C = [1 0 0; 0 0 1];
K0 = [-1 -1; -1 -1]; % Initialize controller matrix
goal = [-5 -3 -1];    % Set goal values for the eigenvalues
weight = abs(goal);   % Set weight for same percentage
lb = -4*ones(size(K0)); % Set lower bounds
ub = 4*ones(size(K0));  % Set upper bounds

options = optimoptions('fgoalattain','Display','iter');

[K,fval,attainfactor]=fgoalattain(@(K)eigfun(K,A,B,C),...
K0,goal,weight,[],[],[],[],lb,ub,[],options)
```

fgoalattain

- Result

```
K =
    -4.0000 -0.2564
    -4.0000 -4.0000

fval =
    -6.9313
    -4.1588
    -1.4099

attainfactor =
    -0.3863
```

fmincon

- Find minimum of constrained nonlinear multivariable function

$$\underset{x}{\text{minimize}}\ f(x) \quad \begin{cases} c(\mathrm{x}) \le 0 \\ ceq(x) = 0 \\ A.x \le b \\ Aeq.x = beq \\ lb \le x \le ub \end{cases}$$

- Syntex

```
[x,fval]= fmincon(fun,x0,A,b,Aeq,beq,lb,ub)
```

fmincon

- <u>Example</u> Find the minimum value of Rosenbrock's function

$$100\left(x_2 - x_1^2\right)^2 + \left(1 - x_1\right)^2$$

with following constraints

$$x_1 + 2x_2 \le 1$$

at starting point (-1,2)

fmincon

- Example Find the minimum value of Rosenbrock's function

$$100\left(x_2 - x_1^2\right)^2 + \left(1 - x_1\right)^2$$

  with following constraints

$$x_1 + 2x_2 \leq 1$$

  at starting point (-1,2)

- Matlab Code

```
x0 = [-1,2];
A = [1,2];
b = 1;
x = fmincon(fun,x0,A,b)
```

fmincon

- <u>Example</u> Find the minimum value of Rosenbrock's function

$$100\left(x_2 - x_1^2\right)^2 + \left(1 - x_1\right)^2$$

with following constraints

$$x_1 + 2x_2 \leq 1$$

at starting point (-1,2)

- Matlab Code

```
x0 = [-1,2];
A = [1,2];
b = 1;
x = fmincon(fun,x0,A,b)
```

- Solution

```
x =
      0.5022     0.2489
```

fminimax

- Finds the minimum of a problem specified by

$$\min_x \max_i f_i(x) \begin{cases} c(\mathrm{x}) \le 0 \\ ceq(x) = 0 \\ A.x \le b \\ Aeq.x = beq \\ lb \le x \le ub \end{cases}$$

- Syntex

```
[x,fval]= fminimax(fun,x0,A,b,Aeq,beq,lb,ub)
```

fminimax

- Example Find values of *x* that minimize the maximum value of

$$\left[f_1(x), f_2(x), f_3(x), f_4(x), f_5(x)\right]$$

where

$$f_1(x) = 2x_1^2 + x_2^2 - 48x_1 - 40x_2 + 304$$

$$f_2(x) = -x_1^2 + 3x_2^2$$

$$f_3(x) = x_1 + 3x_2 - 18$$

$$f_4(x) = -x_1 - x_2$$

$$f_5(x) = x_1 + x_2 - 8$$

at starting point (0.1,0.1)

fminimax

- First, write a file that computes the five functions at x

```
function f = myfun(x)
f(1)= 2*x(1)^2+x(2)^2-48*x(1)-40*x(2)+304;
f(2)= -x(1)^2 - 3*x(2)^2;
f(3)= x(1) + 3*x(2) -18;
f(4)= -x(1)- x(2);
f(5)= x(1) + x(2) - 8;
```

fminimax

- First, write a file that computes the five functions at x

```
function f = myfun(x)
f(1)= 2*x(1)^2+x(2)^2-48*x(1)-40*x(2)+304;
f(2)= -x(1)^2 - 3*x(2)^2;
f(3)= x(1) + 3*x(2) -18;
f(4)= -x(1)- x(2);
f(5)= x(1) + x(2) - 8;
```

- Next, invoke an optimization routine

```
x0 =[0.1; 0.1]; % Make a starting guess solution
[x,fval] = fminimax(@myfun,x0);
```

fminimax

- First, write a file that computes the five functions at x

```
function f = myfun(x)
f(1)= 2*x(1)^2+x(2)^2-48*x(1)-40*x(2)+304;
f(2)= -x(1)^2 - 3*x(2)^2;
f(3)= x(1) + 3*x(2) -18;
f(4)= -x(1)- x(2);
f(5)= x(1) + x(2) - 8;
```

- Next, invoke an optimization routine

```
x0 =[0.1; 0.1]; % Make a starting guess solution
[x,fval] = fminimax(@myfun,x0);
```

- After seven iterations, the solution is

```
x =
    4.0000
    4.0000
fval =
    0.0000 -64.0000 -2.0000 -8.0000 -0.0000
```

fminunc

- Finds the minimum of a problem specified by

$$\min_{x} f(x)$$

  where $f(x)$ is a function that returns a scalar
  x is a vector or a matrix

- Syntex
  `[x,fval]= fminunc(fun,x0)`

fminunc

- Example Minimize the function

$$f(x) = 3x_1^2 + 2x_1x_2 + x_2^2 - 4x_1 + 5x_2$$

at starting point (1,1)

fminunc

- <u>Example</u> Minimize the function

$$f(x) = 3x_1^2 + 2x_1x_2 + x_2^2 - 4x_1 + 5x_2$$

  at starting point (1,1)

- Matlab Code

```
fun = @(x)3*x(1)^2 + 2*x(1)*x(2) + x(2)^2 - 4*x(1)
+ 5*x(2);

x0 = [1,1];
[x,fval] = fminunc(fun,x0);
```

fminunc

- Example Minimize the function

$$f(x) = 3x_1^2 + 2x_1x_2 + x_2^2 - 4x_1 + 5x_2$$

at starting point (1,1)

- Matlab Code

```
fun = @(x)3*x(1)^2 + 2*x(1)*x(2) + x(2)^2 - 4*x(1)
+ 5*x(2);

x0 = [1,1];
[x,fval] = fminunc(fun,x0);
```

- After a few iterations, fminunc returns the solution

```
x =
    2.2500    -4.7500
fval =
    -16.3750
```

fseminf

- Finds the minimum of a problem specified by

$$\underset{x}{\text{minimize}}\ f(x) \quad \begin{cases} K_i(x, w_i) \le 0, 1 \le i \le n \\ c(\mathrm{x}) \le 0 \\ ceq(x) = 0 \\ A.x \le b \\ Aeq.x = beq \\ lb \le x \le ub \end{cases}$$

$K_i(x,w_i) \le 0$ are continuous functions of both x and an additional set of variables $w_1, w_2, \ldots, w_n$

- Syntex

```
[x,fval]= fseminf(fun,x0,ntheta,seminfcon,…
                              A,b,Aeq,beq,lb,ub)
```

  – `ntheta` –Number of semi-infinite constraints
  – `Seminfcon` – Semi-infinite constraint function

fseminf

- Example Minimizes the function

$$\left(x-1\right)^2$$

subject to the constraints

$$0 \le x \le 2$$

$$g(x,t) = \left(x - \tfrac{1}{2}\right) - \left(t - \tfrac{1}{2}\right)^2 \le 0 \text{ for all } 0 \le t \le 1$$

fseminf

- Example Minimizes the function

$$(x-1)^2$$

subject to the constraints

$$0 \le x \le 2$$

$$g(x,t) = \left(x - \tfrac{1}{2}\right) - \left(t - \tfrac{1}{2}\right)^2 \le 0 \text{ for all } 0 \le t \le 1$$

- The unconstrained objective function is minimized at $x = 1$. However, the constraint,

$$g(x,t)^2 \le 0 \text{ for all } 0 \le t \le 1$$

implies $x \le \tfrac{1}{2}$

fseminf

- Write the objective function as an anonymous function

```
objfun = @(x)(x-1)^2;
```

- Write the semi-infinite constraint function,
  - Includes the nonlinear constraints ([ ] in this case)
  - Initial sampling interval for *t* (0 to 1 in steps of 0.01 in this case)
  - The semi-infinite constraint function *g*(*x*, *t*):

```
function [c, ceq, K1, s] = seminfcon(x,s)
% No finite nonlinear inequality and equality
constraints
c = []; ceq = [];
% Sample set
if isnan(s) % Initial sampling interval
    s = [0.01 0];
end
t = 0:s(1):1;
% Evaluate the semi-infinite constraint
K1 = (x - 0.5) - (t - 0.5).^2;
```

fseminf

- Call `fseminf` with initial point 0.2, and view the result:

```
x = fseminf(objfun,0.2,1,@seminfcon)

x =
    0.5000
```

fsolve

- Solves a nonlinear problem specified by

$$F(x) = 0$$

where *F(x)* is a function that returns a vector value

- Syntex

```
[x,fval]= fsolve(fun,x0)
```

fsolve

- Example Solve two nonlinear equations in two variables. The equations are:-

$$e^{-e^{-(x_1+x_2)}} = x_2\left(1+x_1^2\right)$$

$$x_1\cos\left(x_2\right)+x_2\sin\left(x_1\right)=\frac{1}{2}$$

fsolve

- <u>Example</u> Solve two nonlinear equations in two variables. The equations are:-

$$e^{-e^{-(x_1+x_2)}} = x_2\left(1+x_1^2\right)$$

$$x_1\cos\left(x_2\right)+x_2\sin\left(x_1\right)=\frac{1}{2}$$

- Converting in *F(x)*=0

$$e^{-e^{-(x_1+x_2)}} - x_2\left(1+x_1^2\right)=0$$

$$x_1\cos\left(x_2\right)+x_2\sin\left(x_1\right)-\frac{1}{2}=0$$

fsolve

- Matlab Code for function file

```
function F = root2d(x)
F(1) = exp(-exp(-(x(1)+x(2)))) - x(2)*(1+x(1)^2);
F(2) = x(1)*cos(x(2)) + x(2)*sin(x(1)) - 0.5;
```

fsolve

- Matlab Code for function file

```
function F = root2d(x)
F(1) = exp(-exp(-(x(1)+x(2)))) - x(2)*(1+x(1)^2);
F(2) = x(1)*cos(x(2)) + x(2)*sin(x(1)) - 0.5;
```

- Solve the system of equations starting at the point [0,0].

```
fun = @root2d;
x0 = [0,0];
x = fsolve(fun,x0)
```

fsolve

- Matlab Code for function file

```
function F = root2d(x)
F(1) = exp(-exp(-(x(1)+x(2)))) - x(2)*(1+x(1)^2);
F(2) = x(1)*cos(x(2)) + x(2)*sin(x(1)) - 0.5;
```

- Solve the system of equations starting at the point [0,0].

```
fun = @root2d;
x0 = [0,0];
x = fsolve(fun,x0)
```

- Results

```
x =
    0.3532      0.6061
```

linprog

- Solve linear programming problems

$$\min_{x} f^T x \begin{cases} A.x \leq b \\ Aeq.x = beq \\ lb \leq x \leq ub \end{cases}$$

- Syntax

```
x = linprog(fun,A,b,Aeq,beq,lb,ub)
```

linprog

- <u>Example</u> Linear programming

$$\min_x \left(-x_1 - \frac{1}{3}x_2\right) \text{ subject to } \begin{cases} x_1 + x_2 \le 2 \\ x_1 + x_2/4 \le 1 \\ x_1 - x_2 \le 2 \\ -x_1/4 - x_2 \le 1 \\ -x_1 - x_2 \le -1 \\ -x_1 + x_2 \le 2 \end{cases}$$

linprog

- Example Linear programming

$$\min_x \left(-x_1 - \frac{1}{3}x_2\right) \text{ subject to } \begin{cases} x_1 + x_2 \le 2 \\ x_1 + {x_2}/{4} \le 1 \\ x_1 - x_2 \le 2 \\ -{x_1}/{4} - x_2 \le 1 \\ -x_1 - x_2 \le -1 \\ -x_1 + x_2 \le 2 \end{cases}$$

linprog

- <u>Example</u> Linear programming

$$\min_{x}\left(-x_1-\frac{1}{3}x_2\right) \text{ subject to } \begin{cases} x_1+x_2 \le 2 \\ x_1+{x_2}/{4} \le 1 \\ x_1-x_2 \le 2 \\ -{x_1}/{4}-x_2 \le 1 \\ -x_1-x_2 \le -1 \\ -x_1+x_2 \le 2 \end{cases}$$

- Matlab Code

```
% Objective Function
f = [-1 -1/3];
A = [1 1; 1 1/4; 1 -1;-1/4 -1; -1 -1; -1 1];
B = [2 1 2 1 -1 2];
% calling linprog
x = linprog(f,A,b,[],[],[],[])
```

linprog

- Results

```
Optimal solution found.
 x =
    0.6667
    1.3333
```

intlinprog

- Mixed-integer linear programming (MILP)

$$\min_{x} f^T x \begin{cases} x(\text{intcon}) \text{are integers} \\ A.x \le b \\ Aeq.x = beq \\ lb \le x \le ub \end{cases}$$

- Syntax
  `x = intlinprog(f,intcon,A,b,Aeq,beq,lb,ub)`
  – `intcon –integer variables`

intlinprog

- <u>Example</u> Mixed-integer linear programming (MILP)

$$\min_{x}(8x_1 + x_2) \text{ subject to } \begin{cases} x_2 \text{ is an integer} \\ x_1 + 2x_2 \geq -14 \\ -4x_1 - x_2 \leq -33 \\ 2x_1 + x_2 \leq 20 \end{cases}$$

intlinprog

- <u>Example</u> Mixed-integer linear programming (MILP)

$$\min_{x}(8x_1 + x_2) \text{ subject to } \begin{cases} x_2 \text{ is an integer} \\ x_1 + 2x_2 \geq -14 \\ -4x_1 - x_2 \leq -33 \\ 2x_1 + x_2 \leq 20 \end{cases}$$

- Matlab Code

```
% Objective Function and integer variable
f = [8;1];
intcon = 2;
A = [-1,-2; -4,-1; 2,1];
B = [14;-33;20];
% calling intlinprog
x = intlinprog(f,intcon,A,b)
```

intlinprog

- Results

```
LP: Optimal objective value is 59.000000
 x =
    6.5000
    7.0000
```

lsqcurvefit

- Solve nonlinear curve-fitting (data-fitting) problems in least-squares sense

$$\min_x \left\| F(x, xdata) - ydata \right\|_2^2 = \min_x \sum_i \left(F(x, xdata) - ydata \right)^2$$

Nonlinear least-squares solver finds coefficients *x* that solve the problem.
Where user defined data should be feed in vector form

$$F(x, xdata) = \begin{bmatrix} F(x, xdata(1)) \\ F(x, xdata(2)) \\ \vdots \\ F(x, xdata(k)) \end{bmatrix}$$

- Syntax

```
x = lsqcurvefit(fun,x0,xdata,ydata,lb,ub)
```

lsqcurvefit

- <u>Example</u> Simple exponential fit $\text{ydata} = x_1 e^{x_2 \text{xdata}}$

  xdata = [0.9 1.5 13.8 19.8 24.1 28.2 35.2 60.3 74.6 81.3];
  ydata = [455.2 428.6 124.1 67.3 43.2 28.1 13.1 -0.4 -1.3 -1.5];

lsqcurvefit

- <u>Example</u> Simple exponential fit $\text{ydata} = x_1 e^{x_2 \text{xdata}}$

  xdata = [0.9 1.5 13.8 19.8 24.1 28.2 35.2 60.3 74.6 81.3];
  ydata = [455.2 428.6 124.1 67.3 43.2 28.1 13.1 -0.4 -1.3 -1.5];

- Matlab code

```
xdata = [0.9 1.5 13.8 19.8 24.1 28.2 35.2 60.3 …
74.6 81.3];
ydata = [455.2 428.6 124.1 67.3 43.2 28.1 13.1 …
-0.4 -1.3 -1.5];
% Create a simple exponential decay model
fun = @(x,xdata)x(1)*exp(x(2)*xdata);
% Fit the model using the starting point
x0 = [100,-1]

x = lsqcurvefit(fun,x0,xdata,ydata,[],[])
```

lsqcurvefit

- Results

```
Local minimum possible.
x =
    498.8309 -0.1013
```

lsqlin

- Solve constrained linear least-squares problems

$$\min_x \frac{1}{2}\|C.x - d\|_2^2 \quad \begin{cases} A.x \le b \\ Aeq.x = beq \\ lb \le x \le ub \end{cases}$$

  solves the linear system C*x = d

- Syntax

```
x = lsqlin(C,d,A,b,Aeq,beq,lb,ub)
```

lsqlin

- Example Find the x that minimizes the norm of C*x - d for an over determined problem with linear equality and inequality constraints and bounds.

- Matlab code

```
C = [0.9501 0.7620 0.6153 0.4057;
     0.2311 0.4564 0.7919 0.9354;
     0.6068 0.0185 0.9218 0.9169;
     0.4859 0.8214 0.7382 0.4102;
     0.8912 0.4447 0.1762 0.8936];
d = [0.0578; 0.3528; 0.8131; 0.0098; 0.1388];
A =[0.2027 0.2721 0.7467 0.4659;
     0.1987 0.1988 0.4450 0.4186;
     0.6037 0.0152 0.9318 0.8462];
```

lsqlin

- Matlab code conti.

```
b =[0.5251; 0.2026; 0.6721];
Aeq = [3 5 7 9];
beq = 4;
lb = -0.1*ones(4,1);
ub = 2*ones(4,1);

% call lsqlin to solve the problem
x = lsqlin(C,d,A,b,Aeq,beq,lb,ub)
```

lsqlin

- Matlab code conti.

```
b =[0.5251; 0.2026; 0.6721];
Aeq = [3 5 7 9];
beq = 4;
lb = -0.1*ones(4,1);
ub = 2*ones(4,1);

% call lsqlin to solve the problem
x = lsqlin(C,d,A,b,Aeq,beq,lb,ub)
```

- Results

```
x = -0.1000 -0.1000 0.1599 0.4090
```

lsqnonlin

- To solve nonlinear least-squares curve fitting problems

$$\min_x \|f(x)\|_2^2 = \min_x \sum_{i=1}^{n} f_i(x)^2$$

- Trust region reflective algorithm or Levenburg Marquardt (set in Options)
- lsqnonlin stops because the final change in the sum of squares relative to its initial value is less than the default value of the function tolerance.

- Syntax

```
x = lsqnonlin(fun,x0,lb,ub)
```

lsqnonlin

- Example Fit a simple exponential decay curve to data. The model considered is:

$$y = e^{-1.3t} + \varepsilon$$

where $0 \le t \le 3$, and ε is normally distributed noise, mean 0 and standard deviation 0.05.

lsqnonlin

- <u>Example</u> Fit a simple exponential decay curve to data. The model considered is:

$$y = e^{-1.3t} + \varepsilon$$

where $0 \le t \le 3$, and ε is normally distributed noise, mean 0 and standard deviation 0.05.

- Matlab code

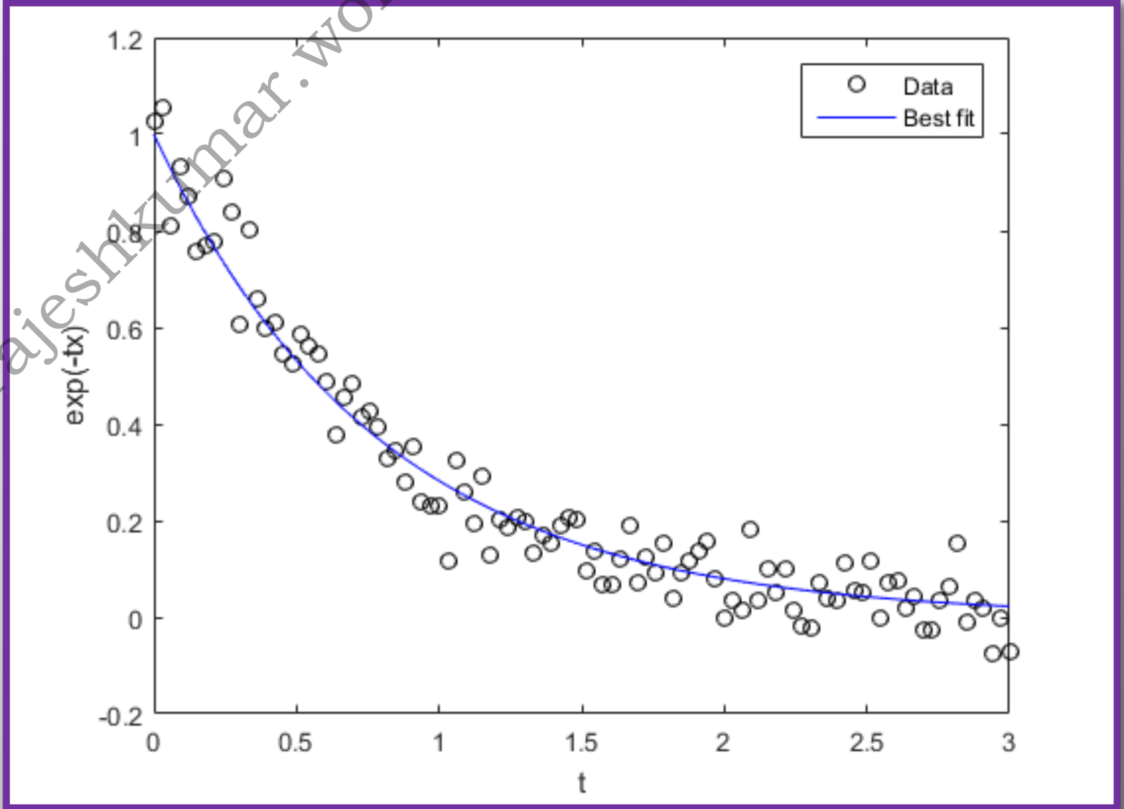
```
rng default; % for reproducibility
d = linspace(0,3);
y = exp(-1.3*d) + 0.05*randn(size(d));
fun = @(r)exp(-d*r)-y;
x0 = 4;

x = lsqnonlin(fun,x0,[],[])
```

lsqnonlin

- Results

```
plot(d,y,'ko',d,exp(-x*d),'b-')
legend('Data','Best fit')
xlabel('t')
ylabel('exp(-tx)')
```



quadprog

- Finds a minimum for a quadratic programming problem that is specified by:

$$\min_x \frac{1}{2} x^T H x + f^T x \begin{cases} Ax \le b \\ Aeq.x \le beq \\ lb \le x \le ub \end{cases}$$

- Syntax

```
x = quadprog(H,f,A,b,Aeq,beq,lb,ub,options)
```

    - options – optimtool options for specific function

quadprog

- Example Solve the given the quadratic programming problem:

$$f(x)=\frac{1}{2}x_1^2+x_2^2-x_1x_2-2x_1-6x_2$$

subject to

$$\begin{aligned} x_1+x_2 &\le 2 \\ -x_1+2x_2 &\le 2 \\ 2x_1+x_2 &\le 3 \\ 0\le x_1;\,0 &\le x_2 \end{aligned}$$

quadprog

- Example Solve the given the quadratic programming problem:

$$f(x) = \frac{1}{2}x_1^2 + x_2^2 - x_1 x_2 - 2x_1 - 6x_2$$

subject to

$$x_1 + x_2 \le 2$$
$$-x_1 + 2x_2 \le 2$$
$$2x_1 + x_2 \le 3$$
$$0 \le x_1; 0 \le x_2$$

- Matlab Code

```
H = [1 -1; -1 2]; f = [-2; -6];
A = [1 1; -1 2; 2 1]; b = [2; 2; 3];
lb = zeros(2,1);
options = optimoptions('quadprog', 'Algorithm',
'interior-point-convex','Display','off');
[x,fval]=
quadprog(H,f,A,b,[],[],lb,[],[],options);
```

quadprog

- Result

```
x =
    0.6667
    1.3333
fval =
    -8.2222
```

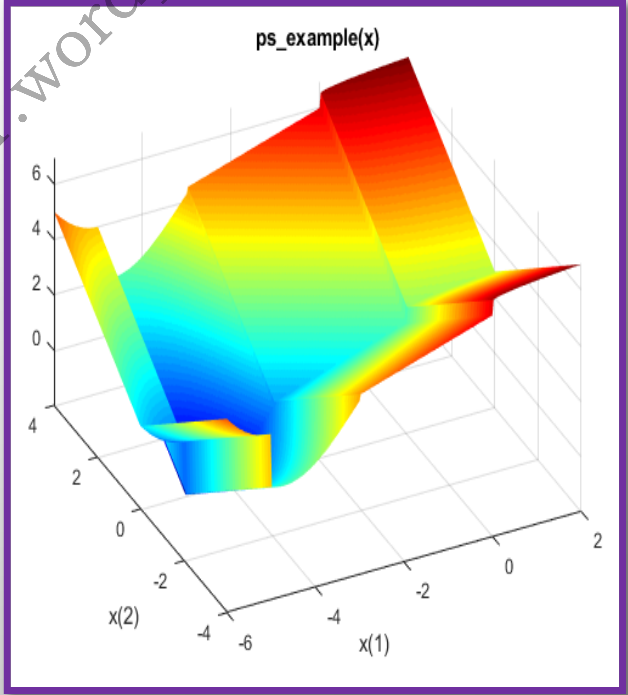
Genetic Algorithm (GA)

- Genetic algorithm solver for mixed-integer or continuous-variable optimization, constrained or unconstrained
- Syntax
  `x = ga(fun,nvars,A,b,Aeq,beq,lb,ub)`
  – `nvars` is the dimension (number of design variables) of fun

Genetic Algorithm (GA)

- <u>Example</u> Use the genetic algorithm to minimize the ps_example function {inbuilt in Matlab}, with following constraints :-

$$x_1 + x_2 \geq 1$$
$$x_2 = x_1 + 5$$
$$1 \leq x_1 \leq 6$$
$$-3 \leq x_2 \leq 8$$



Genetic Algorithm (GA)

- Rearranging constraints

$$x_1 + x_2 \geq 1$$
$$-x_1 + x_2 = 5$$
$$1 \leq x_1 \leq 6$$
$$-3 \leq x_2 \leq 8$$

- Matlab Code

```
A = [-1 -1];     b = -1;
Aeq = [-1 1];    beq = 5
lb = [1 -3];     ub = [6 8];%Set bounds lb and ub
fun = @ps_example;

x = ga(fun,2,A,b,Aeq,beq)
```

Genetic Algorithm (GA)

- Results

```
x =
    -2.0000    2.9990
```

particleswarm

- Bound constrained optimization using **Particle Swarm Optimization (PSO)**
- Minimizes objective function subject to constraints
- MATLAB built in function *particleswarm*
  - May define as an unconstrained problem or find solution in a range
  - Allows to specify various options using *optimoptions*

- Syntax

```
x = particleswarm(fun,nvars,lb,ub,options)
```

particleswarm

- Example the objective function.

$$fun = @(x)x(1)*\exp(-norm(x)^2)$$

Set bounds on the variables

$$lb = [-10, -15]\ \text{and}\ ub = [15, 20]$$

particleswarm

- Example the objective function.

$$fun = @(x)x(1)*\exp(-norm(x)^2)$$

  Set bounds on the variables

$$lb = [-10, -15] \text{ and } ub = [15, 20]$$

- Matlab Code

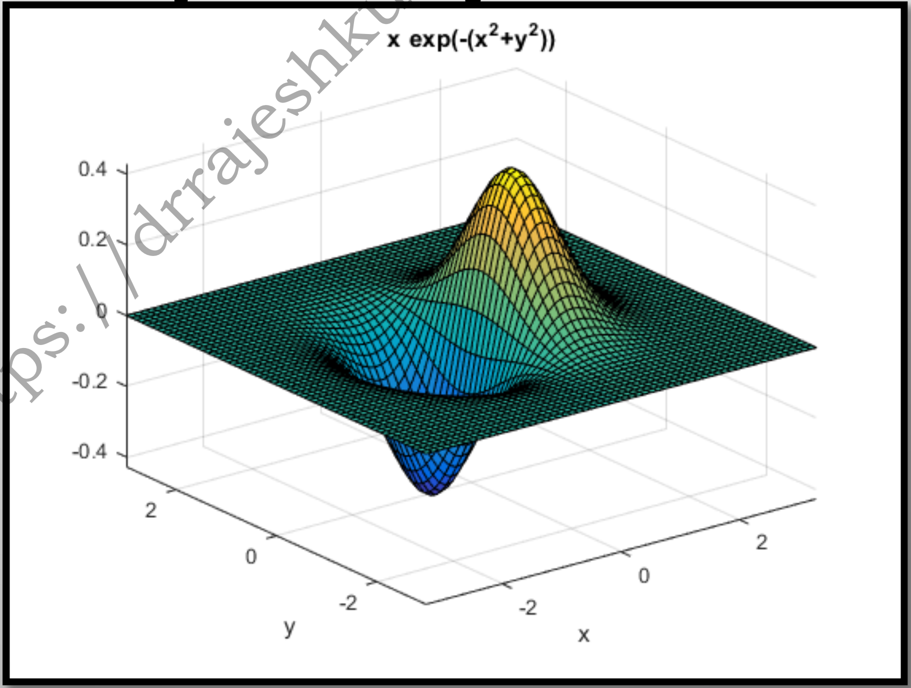
```
%Define the objective function.
fun = @(x)x(1)*exp(-norm(x)^2);

%Call particleswarm to minimize the function.
rng default % For reproducibility
nvars = 2;
x = particleswarm(fun,nvars)
```

particleswarm

- Result

```
x =
    629.4474    311.4814
%This solution is far from the true minimum, as
%you see in a function plot.
fsurf(@(x,y)x.*exp(-(x.^2+y.^2)))
```



simulannealbnd

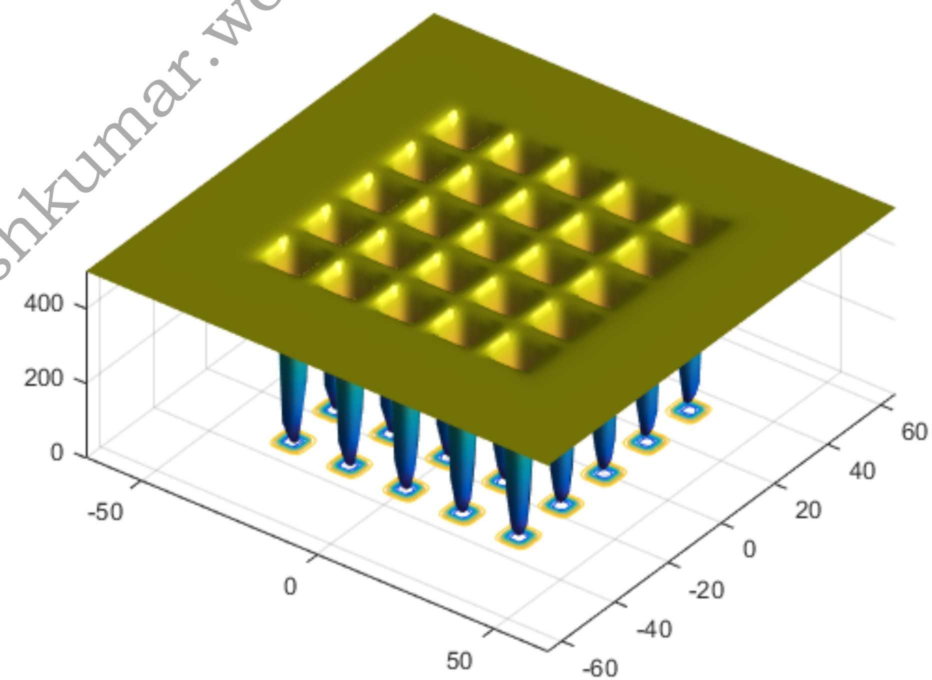
- Finds the minimum of a given function using the Simulated Annealing Algorithm.
- May define upper and lower bounds
- Useful in minimizing functions that may have many local minima

- Syntax

```
[x,fval] = simulannealbnd(fun,x0,lb,ub,options)
```

simulannealbnd

- <u>Example</u> Minimize *De Jong's fifth function* → 2 D function with many local minima.
- Starting point → [0,0]
- UB is 64 and LB is -64



De Jong's Fifth Function Plot – Many Local Minima

simulannealbnd (example)

No Lower and Upper Bounds

```
fun = @dejong5fcn;
x0 = [0 0];
x = simulannealbnd(fun,x0)
```

Output

```
x =

  -31.9785  -31.9797
```

With LB and UB

```
fun = @dejong5fcn;
x0 = [0 0];
lb = [-64 -64];
ub = [64 64];
x = simulannealbnd(fun,x0,lb,ub)
```

Output

```
x =

    0.0009  -31.9644
```

Plot De Jong's function and assign fcn = @dejong5fcn

In main program, simulannealbnd with starting [0,0]

Set LB and UB. *Call simlannealbnd* with LB and UB

Note: The *simulannealbnd* algorithm uses the MATLAB® random number stream, so different results may be obtained after each run

Multi-objective GA

- Find Pareto front of multiple fitness functions using genetic algorithm
- Syntax

```
x = gamultiobj(fitnessfcn,nvars,A,b,Aeq,beq,lb,ub)
```

Multi-objective GA

- Example Compute the Pareto front for a simple multi-objective problem. There are two objectives and two decision variables x

$$f_1(x) = \|x\|^2$$

$$f_2(x) = 0.5\left\|x - \begin{bmatrix} 2 \\ 1 \end{bmatrix}\right\|^2 + 2$$

Multi-objective GA

- Example Compute the Pareto front for a simple multi-objective problem. There are two objectives and two decision variables x

$$f_1(x) = \|x\|^2$$

$$f_2(x) = 0.5\left\| x - \begin{bmatrix} 2 \\ 1 \end{bmatrix} \right\|^2 + 2$$

- Matlab code

```
fitnessfcn = @(x)[norm(x)^2,0.5*norm(x(:)-[2;-
1])^2+2];

rng default % For reproducibility
x = gamultiobj(fitnessfcn,2,[],[],[],[],[],[])
```

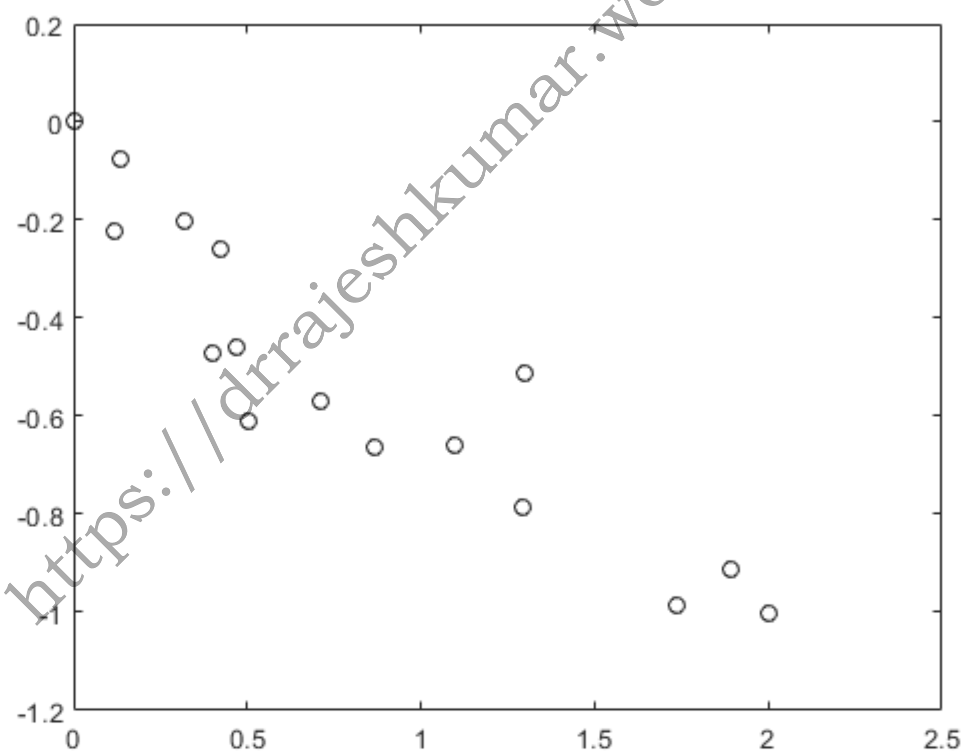
Multi-objective GA

- Results

```
Optimization terminated: average change in the spread of
Pareto solutions less than options.FunctionTolerance.
x =
    -0.0072  0.0003
     0.0947 -0.0811
     1.0217 -0.6946
     1.1254 -0.0857
    -0.0072  0.0003
     0.4489 -0.2101
     1.8039 -0.8394
     0.5115 -0.6314
     1.5164 -0.7277
     1.7082 -0.7006
     1.8330 -0.9337
     0.7657 -0.6695
     0.7671 -0.4882
     1.2080 -0.5407
     1.0075 -0.5348
     0.6281 -0.1454
     2.0040 -1.0064
     1.5314 -0.9184
```

Multi-objective GA

- Plot the solution points
  `plot(x(:,1),x(:,2),'ko')`



patternsearch

- Find a local minimum a given function using pattern search subject to
  - linear equalities/inequalities,
  - lower/upper bounds,
  - non-linear inequalities/equalities
- Specific options may be given using *optimoptions*
- Syntax

```
x = patternsearch(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon)
```

patternsearch

- <u>Example1</u> Minimize an unconstrained problem using a user function and *patternsearch* solver

$$y = \exp(-x(1)^2 - x(2)^2) * (1 + 5x(1) + 6 * x(2) + 12x(1)\cos(x(2)));$$

Function Code

```
function[y] = obj1(x)
y = exp(-x(1)^2-x(2)^2)*(1+5*x(1) + 6*x(2) + 12*x(1)*cos(x(2)));
```

Main Code

```
fun = @obj1;
x0 = [0,0];
x = patternsearch(fun,x0)
```

Output

```
x =
   -0.7037   -0.1860
```

Define a user function (obj1) that describes the objective

In main program, call the function *obj1* and define starting point

Use *patternsearch* to find the minima

patternsearch

- Example 2 Solving the previous problem with lower bound (LB) and upper bound (UB) specified and starting at [1,-5]:

$$0 \le x_1 \le \infty$$
$$-\infty \le x_2 \le -3$$

Main Code

```
fun = @obj1;
lb = [0,-Inf];
ub = [Inf,-3];
A = [];
b = [];
Aeq = [];
beq = [];
x0 = [1,-5];
x =
patternsearch(fun,x0,A,b,Aeq,beq,lb,ub)
```

Output

```
x =
   0.1880   -3.0000
```