Updated on March 19, 2020

MATLAB Optimization Solvers

Prakash Kotecha, Associate Professor
Debasis Maharana, Teaching Assistant &
Remya Kommadath, Teaching Assistant
Indian Institute of Technology Guwahati

MATLAB inbuilt functions: Linear & Mixed Integer Linear Programming: https://www.youtube.com/watch?v=L3J1aIeeWTE
MATLAB inbuilt functions: Nonlinear & Mixed Integer Nonlinear Programming: https://www.youtube.com/watch?v=my-J6YsMTSQ
MATLAB Optimization Tool: Options, Output Function, Vectorization, Parallelization: https://www.youtube.com/watch?v=c9zQY_E75bM
Additional resources: tinyurl.com/sksopti, tinyurl.com/sksoptivid

Optimization solvers (MATLAB 2019a)

Function	Objective function	Constraint handling: Linear	Constraint handling: Nonlinear	Integer variable	Algorithm	Remarks	Type
linprog	linear	✓	×	×	a) 'dual-simplex' (default) b) 'interior-point-legacy' c) 'interior-point'		LP
intlinprog	linear	✓	×	✓	Branch and bound		MILP
quadprog	quadratic	✓	×	×	a) 'interior-point-convex' (default) b) 'trust-region-reflective'	If symmetric matrix is not a positive definite, quadprog issues a warning and uses the symmetrized version (H + H')/2 instead	QP
fminunc	nonlinear	×	×	×	a) 'quasi-newton' (default) b) 'trust-region'	Suitable for continuous differentiable unbounded problems	NLP
fminsearch	nonlinear	×	×	×	'Nelder-Mead simplex'	Solve non differentiable, discontinuous unbounded problem	NLP
fmincon	nonlinear	✓	✓	×	a) 'interior-point' (default) b) 'trust-region-reflective' c) 'sqp' d) 'sqp-legacy' e) 'active-set'	Gradient-based method: objective function and constraints are continuous and also have continuous first derivatives	NLP
patternsearch	nonlinear	✓	✓	×	Pattern search	Can solve single and multi-objective problems (paretosearch)	NLP
particleswarm	nonlinear	×	×	×	Particle swarm optimization	Can solve single objective optimization problems	NLP
simulannealbnd	nonlinear	×	×	×	Simulated annealing	Can solve single objective optimization problems	NLP
ga	nonlinear	✓	✓	✓	Genetic algorithm	Can solve single and multi-objective problems (gamultiobj) Cannot handle integer variables if equality constraints are present	MINLP

Linear programming

Linear objective function

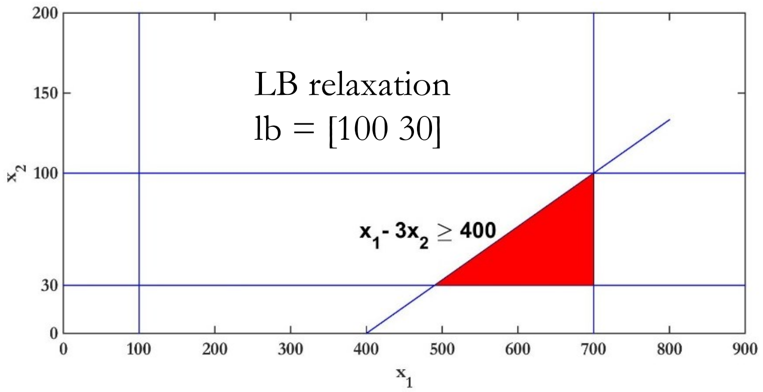
Linear equality and inequality constraints

Bound constraints

MATLAB function: linprog

Introduced before R2006a

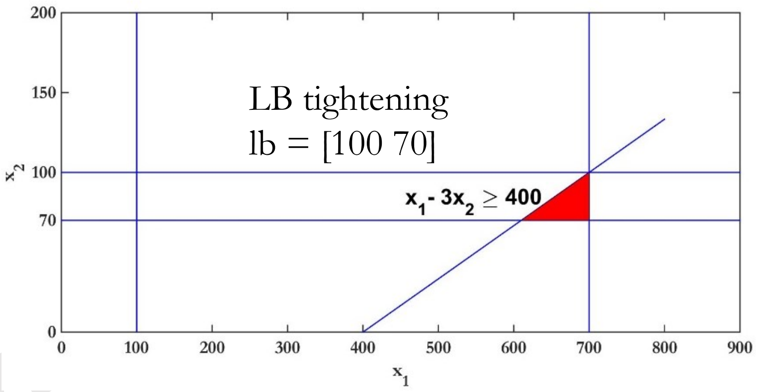
Relaxation and tightening of search space

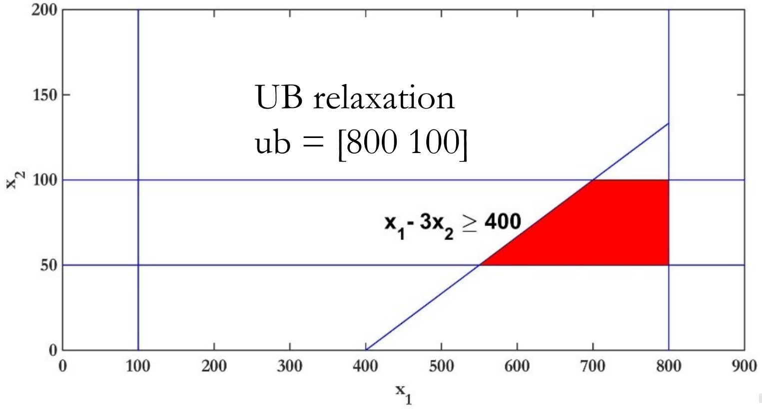


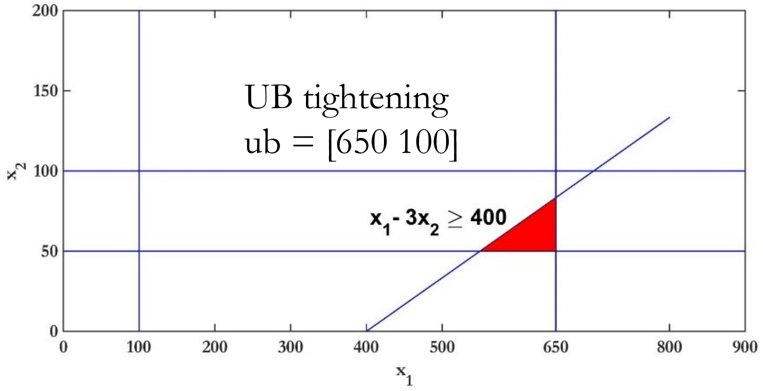
$$\text{Min } f = 3x_1 + 2x_2$$

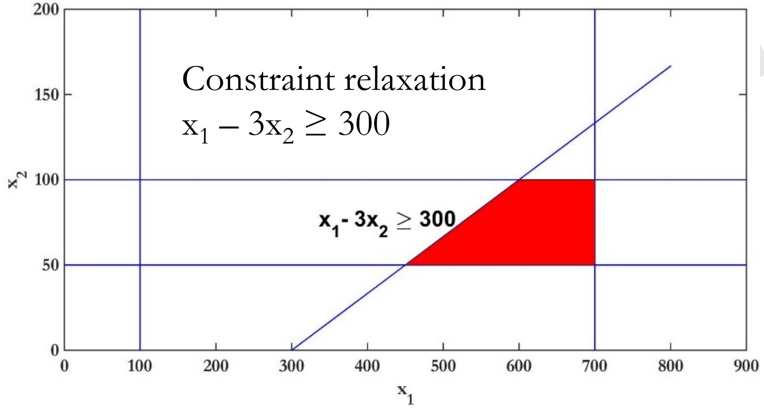
$$\text{lb} = [100\ 50]\ ;\ \text{ub} = [700\ 100]$$

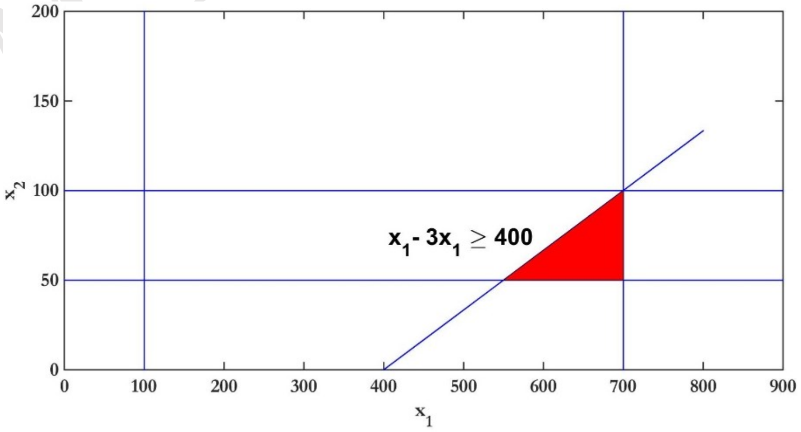
$$\text{st.}\ \ x_1 - 3x_2 \geq 400$$

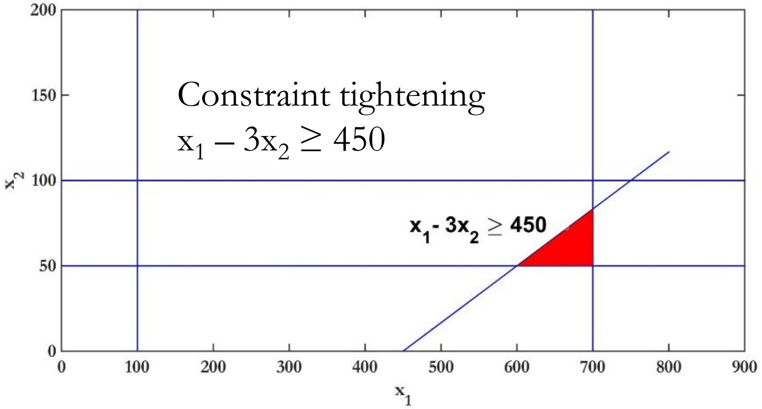












Linear programming

Three neighbouring cities discharge pollutants A and B into the river after pollution treatment. The state government has set up a treatment plant that treats the pollutants from each city.

	Amount of A (in tons) per ton of waste	Amount of B (in tons) per ton of waste	Cost
City 1	0.1	0.45	$15/ton
City 2	0.2	0.25	$10/ton
City 3	0.4	0.3	$20/ton
Required reduction	At least 30 tons	At least 40 tons	

Determine the amount of waste treated from each city that will minimize the cost of pollutants by desired amount.

Adapted from Introduction to Applied Optimization by Urmila Diwekar

Linear programming

Let x_1 = amount of waste treated from city 1

x_2 = amount of waste treated from city 2

x_3 = amount of waste treated from city 3

Decision variables

	Reduction in A /ton	Reduction in B /ton	Cost
City 1	0.1	0.45	\$15/ton
City 2	0.2	0.25	\$10/ton
City 3	0.4	0.3	\$20/ton
Required reduction	30 tons	40 tons	

Minimize Cost, $Z = 15\,x_1 + 10\,x_2 + 20\,x_3$

Objective function

subject to

Reduction in A: $0.1\,x_1 + 0.2\,x_2 + 0.4\,x_3 \geq 30$

Reduction in B: $0.45\,x_1 + 0.25\,x_2 + 0.3\,x_3 \geq 40$

Constraints

Flow rates: $x_1, x_2, x_3 \geq 0$

Bound constraints

MATLAB function: *linprog*

linprog

Min Z = 15 x_1 + 10 x_2 + 20 x_3

$0.1\,x_1 + 0.2\,x_2 + 0.4\,x_3 \geq 30$
$0.45\,x_1 + 0.25\,x_2 + 0.3\,x_3 \geq 40$

$x_1, x_2, x_3 \geq 0$

Min Z = 15 x_1 + 10 x_2 + 20 x_3

$-0.1\,x_1 - 0.2\,x_2 - 0.4\,x_3 \leq -30$
$-0.45\,x_1 - 0.25\,x_2 - 0.3\,x_3 \leq -40$

$x_1, x_2, x_3 \geq 0$

$$\min_x f^T x \ \ subject\ \ to \begin{cases} Ax \leq b \\ Aeq\,x = beq \\ lb \leq x \leq ub \end{cases} \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix}$$

f, x, b, beq, lb, and ub are vectors and A and Aeq are matrices

```
[X,FVAL,EXITFLAG,OUTPUT,LAMBDA]  = linprog(Z,A,b,Aeq,beq,lb,ub,options)
```

Input:

$$Z = [15 \quad 10 \quad 20]$$

$$A = \begin{bmatrix} -0.1 & -0.2 & -0.4 \\ -0.45 & -0.25 & -0.3 \end{bmatrix} \quad b = \begin{bmatrix} -30 \\ -40 \end{bmatrix}$$

$$lb = [0 \quad 0 \quad 0]$$

Output:

x	Solution vector
fval	Objective function value at solution
exitflag	Algorithm stopping condition
output	Information such as number of iterations, algorithm used, exit message etc., of the optimization process
lambda	Lagrange multipliers

MATLAB code

Create a script file to solve the problem using *linprog*

Input:

$$Z = \begin{bmatrix} 15 & 10 & 20 \end{bmatrix}$$

$$A = \begin{bmatrix} -0.1 & -0.2 & -0.4 \\ -0.45 & -0.25 & -0.3 \end{bmatrix} \quad b = \begin{bmatrix} -30 \\ -40 \end{bmatrix}$$

$$lb = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

```
clc; clear % Clear command window and workspace respectively

lb = zeros(1,3);%Lower bounds

A = [-0.1 -0.2 -0.4; -0.45 -0.25 -0.3];% coefficients of inequality constraints
b = -[30 40]';% RHS of inequality constraints

Z = [15 10 20];% Coefficients of objective function

% Calling the solver
[X, FVAL, EXITFLAG, OUTPUT, LAMBDA]  = linprog(Z, A, b, [], [], lb);
```

Output:

X	[7.6923 146.1538 0]	
FVAL	1576.9231	
LAMBDA.lower	[0 0 6.1538]	Shadow price for change in lower bound
LAMBDA.upper	[0 0 0]	Shadow price for change in upper bound
LAMBDA.ineqlin	[11.5385 30.7692]	Shadow price for change in RHS of inequality constraints

Result analysis

LAMBDA.lower = [0 0 6.1538]

- Optimal value of x_3 lies on lower bound
- One unit decrease in lb of x_3 results in 6.1538 units improvement in objective value

LAMBDA.ineqlin = [11.5385 30.7692]

- One unit decrease in b_1 results in 11.5385 units improvement in objective value, similarly one unit decrease in b_2 causes 30.7692 unit improvement in objective value

X	[7.6923 146.1538 0]
FVAL	1576.9231
LAMBDA.lower	[0 0 6.1538]
LAMBDA.upper	[0 0 0]
LAMBDA.ineqlin	[11.5385 30.7692]

Decreasing the value of lb ➔ Increased the search space

Increasing the value of lb ➔ Reduced the search space

Decreasing the value of b ➔ Reduced the search space

Increasing the value of b ➔ Increased the search space

lb = [0 0 0]

$$b = \begin{bmatrix} -30 \\ -40 \end{bmatrix}$$

lb_3	b_1	b_2	Lagrange multipliers	FVAL
$lb_3 + 1 = 1$	-	-	6.1538	1576.9231 + 6.1538 = 1583.0769
$lb_3 - 1 = -1$	-	-	6.1538	1576.9231 – 6.1538 = 1570.7692
-	$b_1 + 1 = -29$	-	11.5385	1576.9231 – 11.5385 = 1565.3846
-	$b_1 - 1 = -31$	-	11.5385	1576.9231 + 11.5385 = 1588.4616
-	-	$b_2 + 1 = -39$	30.7692	1576.9231 – 30.7692 = 1546.1538
-	-	$b_2 - 1 = -41$	30.7692	1576.9231 + 30.7692 = 1607.6923

Result analysis

Input:

$$Z = [15 \quad 10 \quad 20]$$

$$A = \begin{bmatrix} -0.1 & -0.2 & -0.4 \\ -0.45 & -0.25 & -0.3 \end{bmatrix} \quad b = \begin{bmatrix} -30 \\ -40 \end{bmatrix}$$

$$lb = [0 \quad 0 \quad 0]$$

X	[7.6923 146.1538 0]
FVAL	**1576.9231**
LAMBDA.lower	[0 0 **6.1538**]
LAMBDA.upper	[0 0 0]
LAMBDA.ineqlin	[**11.5385** **30.7692**]

$$Z = [15 \quad 10 \quad 20]$$

$$A = \begin{bmatrix} -0.1 & -0.2 & -0.4 \\ -0.45 & -0.25 & -0.3 \end{bmatrix} \quad b = \begin{bmatrix} -30 \\ -40 \end{bmatrix}$$

$$lb = [0 \quad 0 \quad 1]$$

FVAL= 1583.08

$$Z = [15 \quad 10 \quad 20]$$

$$A = \begin{bmatrix} -0.1 & -0.2 & -0.4 \\ -0.45 & -0.25 & -0.3 \end{bmatrix} \quad b = \begin{bmatrix} -29 \\ -40 \end{bmatrix}$$

$$lb = [0 \quad 0 \quad 0]$$

FVAL= 1565.38

$$Z = [15 \quad 10 \quad 20]$$

$$A = \begin{bmatrix} -0.1 & -0.2 & -0.4 \\ -0.45 & -0.25 & -0.3 \end{bmatrix} \quad b = \begin{bmatrix} -30 \\ -39 \end{bmatrix}$$

$$lb = [0 \quad 0 \quad 0]$$

FVAL= 1546.15

$$Z = [15 \quad 10 \quad 20]$$

$$A = \begin{bmatrix} -0.1 & -0.2 & -0.4 \\ -0.45 & -0.25 & -0.3 \end{bmatrix} \quad b = \begin{bmatrix} -30 \\ -40 \end{bmatrix}$$

$$lb = [0 \quad 0 \quad -1]$$

FVAL= 1570.77

$$Z = [15 \quad 10 \quad 20]$$

$$A = \begin{bmatrix} -0.1 & -0.2 & -0.4 \\ -0.45 & -0.25 & -0.3 \end{bmatrix} \quad b = \begin{bmatrix} -31 \\ -40 \end{bmatrix}$$

$$lb = [0 \quad 0 \quad 0]$$

FVAL= 1588.46

$$Z = [15 \quad 10 \quad 20]$$

$$A = \begin{bmatrix} -0.1 & -0.2 & -0.4 \\ -0.45 & -0.25 & -0.3 \end{bmatrix} \quad b = \begin{bmatrix} -30 \\ -41 \end{bmatrix}$$

$$lb = [0 \quad 0 \quad 0]$$

FVAL= 1607.69

Linear programming

Let $X = [x_1,\ x_2,\ x_3]$ be the vector of decision variables.

Minimize $x_1 + 2x_2 - 3x_3$ **Objective function**

Subject to

$$x_1 + 2x_2 \le 3$$
$$x_2 + x_3 \le 2$$

Inequality constraints

$$x_1 + x_2 + x_3 = 4$$

Equality constraints

Constraints

$$0 \le x_1 \le 5$$
$$-1.5 \le x_2 \le 3$$
$$0 \le x_3 \le 2$$

Bound constraints

MATLAB function: *linprog*

linprog

$$\min_x\ f^T x \ \ subject\ \ to \begin{cases} Ax \le b \\ Aeq\,x = beq \\ lb \le x \le ub \end{cases}$$

Input:

$$Z = \begin{bmatrix} 1 & 2 & -3 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

$$Aeq = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \qquad beq = \begin{bmatrix} 4 \end{bmatrix}$$

$$lb = \begin{bmatrix} 0 & -1.5 & 0 \end{bmatrix} \qquad ub = \begin{bmatrix} 5 & 3 & 2 \end{bmatrix}$$

MATLAB code

Create a script file to solve the problem using *linprog*

```
clc; clear % Clear the command window and workspace respectively
Z = [1 2 -3]; % Coefficients of objective function
A = [1 2 0; 0 1 1]; % coefficients of inequality constraints
b = [3;2]; % RHS of inequality constraints

Aeq = [1 1 1]; % coefficients of equality constraints
beq = 4; % RHS of equality constraints

lb = [0 -1.5 0]; % Lower bound of decision variables
ub = [5 3 2]; % Upper bound of decision variables

% Calling the solver
[X, FVAL, EXITFLAG, OUTPUT, LAMBDA] = linprog(Z, A, b, Aeq, beq, lb, ub);
```

$$Z = \begin{bmatrix} 1 & 2 & -3 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

$$Aeq = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \qquad beq = \begin{bmatrix} 4 \end{bmatrix}$$

$$lb = \begin{bmatrix} 0 & -1.5 & 0 \end{bmatrix} \quad ub = \begin{bmatrix} 5 & 3 & 2 \end{bmatrix}$$

Output:

`X`	`[3.5 -1.5 2]`	
`FVAL`	`-5.5`	
`LAMBDA.lower`	`[0 1 0]`	Shadow price for change in lower bound
`LAMBDA.upper`	`[0 0 4]`	Shadow price for change in upper bound
`LAMBDA.ineqlin`	`[0 0]`	Shadow price for change in RHS of inequality constraints
`LAMBDA.eqlin`	`-1`	Shadow price for change in RHS of inequality constraints

Result analysis

FVAL	-5.5
LAMBDA.lower	[0 1 0]
LAMBDA.upper	[0 0 4]
LAMBDA.eqlin	-1

lb = [0 **-1.5** 0]

ub = [5 3 **2**]

$b_{eq} = [4]$

lb_2	ub_3	b_{eq}	Lagrange multipliers	FVAL
$lb_2 + 1 = -0.5$	-	-	1	$-5.5 + 1 = -4.5$
$lb_2 - 1 = -2.5$	-	-	1	$-5.5 - 1 = -6.5$
-	$ub_3 + 1 = 3$	-	4	$-5.5 - 4 = -9.5$
-	$ub_3 - 1 = 1$	-	4	$-5.5 + 4 = -1.5$
-	-	$b_{eq} + 1 = 5$	-1	$-5.5 - (-1) = -4.5$
-	-	$b_{eq} - 1 = 3$	-1	$-5.5 + (-1) = -6.5$

Result analysis

$Z = [1 \quad 2 \quad -3]$

$A = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$

$Aeq = [1 \quad 1 \quad 1] \qquad beq = [4]$

$lb = [0 \quad -1.5 \quad 0] \qquad ub = [5 \quad 3 \quad 2]$

X	**[3.5 -1.5 2]**
FVAL	**-5.5**
LAMBDA.lower	**[0 1 0]**
LAMBDA.upper	**[0 0 4]**
LAMBDA.ineqlin	**[0 0]**
LAMBDA.eqlin	**-1**

$Z = [1 \quad 2 \quad -3]$

$A = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$

$Aeq = [1 \quad 1 \quad 1] \qquad beq = [4]$

$lb = [0 \quad -0.5 \quad 0] \qquad ub = [5 \quad 3 \quad 2]$

$FVAL = -4.5$

$Z = [1 \quad 2 \quad -3]$

$A = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$

$Aeq = [1 \quad 1 \quad 1] \qquad beq = [4]$

$lb = [0 \quad -1.5 \quad 0] \qquad ub = [5 \quad 3 \quad 3]$

$FVAL = -9.5$

$Z = [1 \quad 2 \quad -3]$

$A = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$

$Aeq = [1 \quad 1 \quad 1] \qquad beq = [5]$

$lb = [0 \quad -1.5 \quad 0] \qquad ub = [5 \quad 3 \quad 2]$

$FVAL = -4.5$

$Z = [1 \quad 2 \quad -3]$

$A = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$

$Aeq = [1 \quad 1 \quad 1] \qquad beq = [4]$

$lb = [0 \quad -2.5 \quad 0] \qquad ub = [5 \quad 3 \quad 2]$

$FVAL = -6.5$

$Z = [1 \quad 2 \quad -3]$

$A = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$

$Aeq = [1 \quad 1 \quad 1] \qquad beq = [4]$

$lb = [0 \quad -1.5 \quad 0] \qquad ub = [5 \quad 3 \quad 1]$

$FVAL = -1.5$

$Z = [1 \quad 2 \quad -3]$

$A = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$

$Aeq = [1 \quad 1 \quad 1] \qquad beq = [3]$

$lb = [0 \quad -1.5 \quad 0] \qquad ub = [5 \quad 3 \quad 2]$

$FVAL = -6.5$

Mixed Integer Linear Programming

Linear objective function

Linear equality and inequality constraints

Bound constraints
&
Integer variables

MATLAB function: *intlinprog*
Introduced in R2014a

Mixed Integer Linear Programming

- Reddy Mikks company produces interior and exterior paints from raw materials, M1 and M2.
- Daily demand for interior paint cannot exceed that for exterior paint by more than 1 unit.
- Maximum daily demand for the interior paint is 2 units.
- Determine optimum quantity of interior and exterior paints that maximizes total daily profit.

	Exterior paint	**Interior paint**	**Availability**
M1	6	4	24
M2	1	2	6
Profit	5	4	

Mixed Integer Linear Programming

Let x_1 = Units of exterior paint produced daily

x_2 = Units of interior paints produced daily

Decision variables

Maximize Profit, $Z = 5x_1 + 4x_2$

Objective function

	Ext. paint	Int. paint	Availability
M1	6	4	24
M2	1	2	6
Profit	5	4	

Daily demand for interior paint cannot exceed that for exterior paint by more than 1 Unit.

Subject to

$$6x_1 + 4x_2 \le 24$$
$$x_1 + 2x_2 \le 6$$

Raw material constraints

$$x_2 \le x_1 + 1$$

Demand constraint

Constraints

$$x_1, x_2 \ge 0$$
$$x_2 \le 2$$

Bound constraints

MATLAB function: *intlinprog*

Input:

$$Z = \begin{bmatrix} -5 & -4 \end{bmatrix}$$

$$A = \begin{bmatrix} 6 & 4 \\ 1 & 2 \\ -1 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 24 \\ 6 \\ 1 \end{bmatrix}$$

$$lb = \begin{bmatrix} 0 & 0 \end{bmatrix} \quad ub = \begin{bmatrix} \text{inf} & 2 \end{bmatrix}$$

intlinprog

Max Z = $5x_1 + 4x_2$

$6x_1 + 4x_2 \le 24$
$x_1 + 2x_2 \le 6$
$x_2 \le x_1 + 1$

$x_1, x_2 \ge 0$
$x_2 \le 2$

Min Z = $-5x_1 - 4x_2$

$6x_1 + 4x_2 \le 24$
$x_1 + 2x_2 \le 6$
$-x_1 + x_2 \le 1$

$x_1, x_2 \ge 0$
$x_2 \le 2$

linprog

$$\min_x f^T x \ \ subject \ \ to \begin{cases} Ax \le b \\ Aeq\,x = beq \\ lb \le x \le ub \end{cases}$$

$$\min_x f^T x \ \ subject \ \ to \begin{cases} x(intcon) \ \ are \ \ integers \\ Ax \le b \\ Aeq\,x = beq \\ lb \le x \le ub \end{cases} \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix}$$

$f, x, intcon, b, beq, lb,$ and ub are vectors, and A and Aeq are matrices

```
[X, FVAL, EXITFLAG, OUTPUT]  = intlinprog(Z,intcon,A,b,Aeq,beq,lb,ub,X0,options)
```

Input:

$$Z = \begin{bmatrix} -5 & -4 \end{bmatrix}$$

$$A = \begin{bmatrix} 6 & 4 \\ 1 & 2 \\ -1 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 24 \\ 6 \\ 1 \end{bmatrix}$$

$$lb = \begin{bmatrix} 0 & 0 \end{bmatrix} \qquad ub = \begin{bmatrix} \inf & 2 \end{bmatrix}$$

Output:

x	Solution vector
fval	Objective function value at solution
exitflag	Algorithm stopping condition
output	Information such as number of relative gap, absolute gap, number of integer feasible points determined, number of nodes in branch and bound algorithm etc.

MATLAB code

Create a script file to solve the problem using *intlinprog*

```
clc; clear

Z = -[5 4];   % coefficients of objective function

A = [6 4; 1 2; -1 1];   % coefficients of linear inequality constraints
b = [24; 6; 1];   % RHS of linear inequality constraints

lb = [0 0];   % lower bounds of the variables
ub = [inf 2];% upper bounds of the variables

intcon = [1 2];% Indices of integer variables

% Calling solver
[x, fval] = intlinprog(Z, intcon, A, b, [], [], lb, ub);
```

Input:

$$Z = [-5 \quad -4]$$

$$A = \begin{bmatrix} 6 & 4 \\ 1 & 2 \\ -1 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 24 \\ 6 \\ 1 \end{bmatrix}$$

$$lb = [0 \quad 0] \qquad ub = [\inf \quad 2]$$

Output:

```
X          [4   0]
FVAL       -20
```

Solve using linprog

Create a script file to solve the problem using *linprog*

```
clc; clear
Z = [-5 -4];   % coefficients of objective function
A = [6 4; 1 2; -1 1]; %coefficients of linear inequality constraints
b = [24; 6; 1];   % RHS of linear inequality constraints
lb = [0 0];   % lower bounds of the variables
ub = [inf 2];% upper bounds of the variables
intcon = [1 2];% Indices of integer variables
% Calling solver
[x, fval] = linprog(Z, A, b, [], [], lb, ub);
```

Input:

$$Z = \begin{bmatrix} -5 & -4 \end{bmatrix}$$

$$A = \begin{bmatrix} 6 & 4 \\ 1 & 2 \\ -1 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 24 \\ 6 \\ 1 \end{bmatrix}$$

$$lb = \begin{bmatrix} 0 & 0 \end{bmatrix} \qquad ub = \begin{bmatrix} \inf & 2 \end{bmatrix}$$

linprog solution:

X	[3 1.5]
FVAL	-21

Ceil the solution

X	[3 2]
FVAL	-23
AX - b	[2 1 -2]

Infeasible solution

Floor the solution

X	[3 1]
FVAL	-19
AX - b	[-2 -1 -2]

Sub-optimal solution

intlinprog solution:

X	[4 0]
FVAL	-20

Quadratic Programming

Quadratic objective function

Linear equality and inequality constraints

Bound constraints

MATLAB function: *quadprog*

Introduced before R2006a

Quadratic Programming

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

Objective function

Quadratic

- Subject to

$$2x_1 + x_2 \le 6$$
$$x_1 - 4x_2 \le 0$$
$$x_1 \ge 0, x_2 \ge 0$$

Constraints

Linear

Bound constraints

- MATLAB function: *quadprog*

quadprog

$$\min f(x) = -4x_1 + x_1^2 - 2x_1x_2 + 2x_2^2$$
$$2x_1 + x_2 \le 6$$
$$x_1 - 4x_2 \le 0$$
$$x_1 \ge 0, x_2 \ge 0$$

$$\min_x \frac{1}{2} x^T H x + f^T x \text{ such that } \begin{cases} Ax \le b \\ Aeq\, x = beq \\ lb \le x \le ub \end{cases} \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix}$$

$H, A,$ *and* Aeq *are matrices, and* $f, b, beq, lb, ub,$ *and* x *are vectors.*

If symmetric matrix (H) is not a positive definite, quadprog issues a warning and uses the symmetrized version $(H + H^T)/2$ instead


```
[X, FVAL, EXITFLAG, OUTPUT, LAMBDA] = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0,options)
```

Input:

$f = \begin{bmatrix} -4 & 0 \end{bmatrix} \quad H = \begin{bmatrix} 2 & -2 \\ -2 & 4 \end{bmatrix}$

$A = \begin{bmatrix} 2 & 1 \\ 1 & -4 \end{bmatrix} \quad b = \begin{bmatrix} 6 \\ 0 \end{bmatrix}$

$lb = \begin{bmatrix} 0 & 0 \end{bmatrix}$

$$H = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} \end{bmatrix}$$

$$H = \begin{bmatrix} 2 & -2 \\ -2 & 4 \end{bmatrix}$$

Output:

x	Solution vector
fval	Objective function value at solution
exitflag	Algorithm stopping condition
output	Information such as number of iterations, algorithm used, exit message etc., of the optimization process
lambda	Lagrange multipliers at the solution negative of the associated "shadow price."

MATLAB code

Create a script file to solve the problem using *quadprog*

```
clc; clear % Clear the command window and workspace respectively

f = [-4; 0];          % Linear terms in the objective function
H = [2 -2; -2 4];     % Hessian matrix

A = [2 1; 1 -4];      % Coefficients of linear inequality
b = [6; 0];           % RHS of linear inequality

Aeq = [];             % Coefficients of linear equality
beq = [];             % RHS of linear equality

lb = [0 0];           % Lower bounds of the variables
ub = [];              % Upper bounds of the variables

% Calling the solver
[x, fval, exitflag, output, lambda] = quadprog(H, f, A, b, Aeq, beq, lb, ub)
```

Input:

$$f = \begin{bmatrix} -4 & 0 \end{bmatrix} \quad H = \begin{bmatrix} 2 & -2 \\ -2 & 4 \end{bmatrix}$$

$$A = \begin{bmatrix} 2 & 1 \\ 1 & -4 \end{bmatrix} \quad b = \begin{bmatrix} 6 \\ 0 \end{bmatrix}$$

$$lb = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

Output:

x	[2.4615 1.0769]	
FVAL	-6.7692	
LAMBDA.lower	[0.3871e^{-11} 0.8824e^{-11}]	Shadow price for change in lower bound
LAMBDA.upper	[0 0]	Shadow price for change in upper bound
LAMBDA.ineqlin	[0.6154 0]	Shadow price for change in RHS of inequality constraints

Unconstrained Non-linear Programming

Non-linear objective function

MATLAB function: fminunc and fminsearch

Introduced before R2006a

Unconstrained Non-linear Programming

Rosenbrock's function

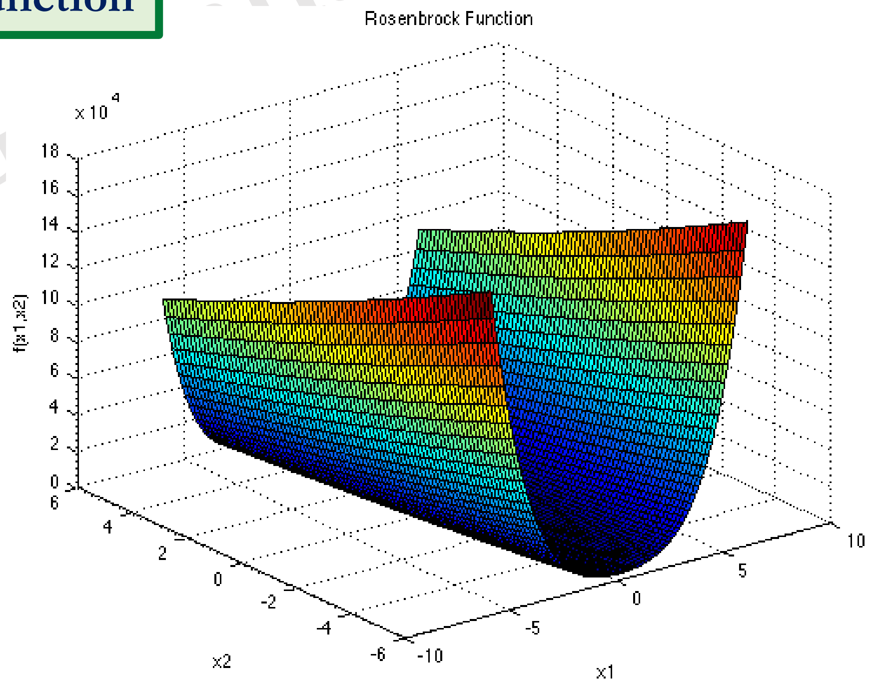
$$\text{Minimize } f(x) = 100\left(x_2 - x_1^2\right)^2 + \left(x_1 - 1\right)^2$$

Non-linear function

Gradient:

$$\nabla f(x) = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \dfrac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} -400\left(x_2 - x_1^2\right)x_1 + 2\left(x_1 - 1\right) \\ 200\left(x_2 - x_1^2\right) \end{bmatrix}$$

MATLAB function: *fminunc*



$$f^* = 0 \quad x^* = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

Continuous differentiable function

fminunc

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

$$\textit{where } f(x) \textit{ is the objective function.}$$

Objective function file returns a scalar

If user provides the gradient information of *f*(*x*), then the objective function file returns two scalars (function value and gradient value)

```
[x, fval, exitflag, output, grad, hessian] = fminunc(fun,x0,options)
```

Input:

fun	Objective function handle
x0	Initial point of decision variables
options	Structure containing the optimization options such as choice of algorithms, display options, maximum iterations/ function evaluations, plot functions, output functions etc.

Output:

x	Solution vector
fval	Objective function value at solution
exitflag	Algorithm stopping condition
output	Information such as number of iterations and function count, algorithm used, exit message etc.,
grad	Gradient at the solution
hessian	Approximate Hessian

MATLAB code

Create a function file of the objective function

```
function [f,g] = rosenbrockwithgrad(x)
% Calculate objective f
f = 100*(x(2) - x(1)^2)^2 + (1-x(1))^2;

% Determination of gradient
g = [-400*(x(2)-x(1)^2)*x(1)+2*(x(1)-1);
    200*(x(2)-x(1)^2)];
```

Create a script file to solve the problem using *fminunc*

```
clc;clear

x0 = [-1,2]; % Initial guess
fun = @rosenbrockwithgrad; % objective function handle

% calling the solver
[x, fval, exitflag, output, grad, hessian] = fminunc(fun, x0);
```

$$Minimize\ f(x) = 100\left(x_2 - x_1^2\right)^2 + \left(x_1 - 1\right)^2$$

$$\nabla f(x) = \begin{bmatrix} -400\left(x_2 - x_1^2\right)x_1 + 2\left(x_1 - 1\right) \\ 200\left(x_2 - x_1^2\right) \end{bmatrix}$$

Output:

X	FVAL
[0.9999 0.9999]	$1.2262e^{-10}$

fminsearch

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

$$\textit{where } f(x) \textit{ is the objective function}$$

Objective function file returns a scalar

```
[x, fval, exitflag, output] = fminsearch(fun, x0, options)
```

Input:

fun	Objective function handle
x0	Initial point of decision variables
options	Structure containing the optimization options such as choice of algorithms, display options, maximum iterations/ function evaluations, plot functions, output functions etc.

Output:

x	Solution vector
fval	Objective function value at solution
exitflag	Algorithm stopping condition
output	Information such as number of iterations and function count, algorithm used and exit message.

MATLAB code

$$Minimize\ f(x) = 100\left(x_2 - x_1^2\right)^2 + \left(x_1 - 1\right)^2$$

Create a function file of the objective function

```
function f = rosenbrock(x)


% Calculate objective f
f = 100*(x(2) - x(1)^2)^2 + (1-x(1))^2;
```

Output:

X	FVAL
[0.9999 0.9999]	$1.7062e^{-10}$

Create a script file to solve the problem using *fminsearch*

```
clc; clear

fun = @rosenbrock; % Objective function
x0 = [-1, 2]; % Initial guess

% Calling the solver
[x, fval, exitflag, output] = fminsearch (fun, x0);
```

Bound Constrained Non-linear Programming

Non-linear objective function

Bound constraints

MATLAB function: simulannealbnd and particleswarm

simulannealbnd: Introduced in R2007a

particleswarm: Introduced in R2014b

Bound Constrained Non-linear Programming

Rastrigin Function

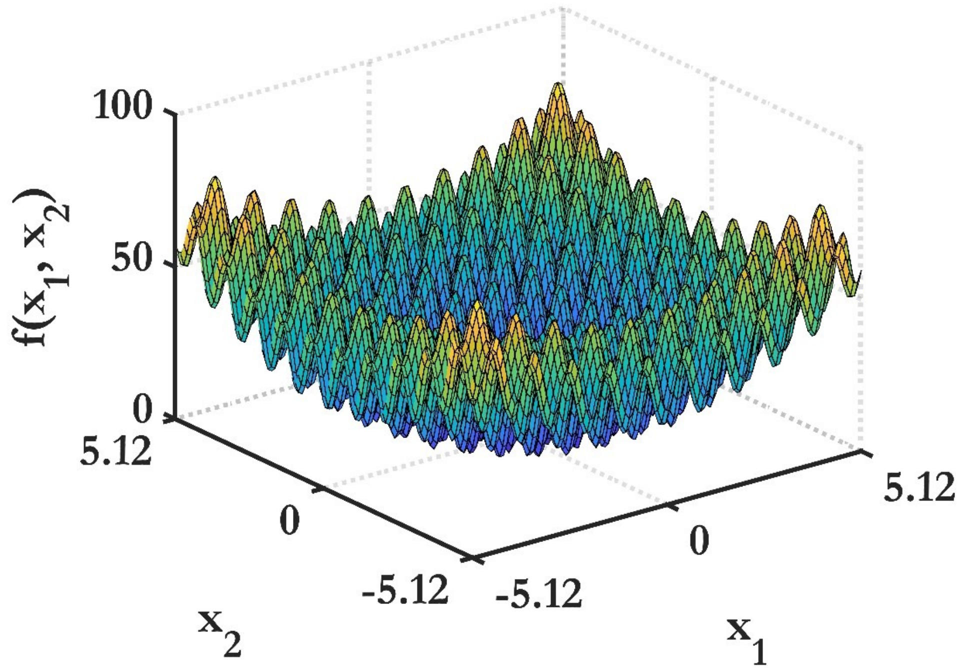
$$f(x) = 10D + \sum_{i=1}^{D}\left[x_i^2 - 10\cos(2\pi x_i)\right]$$

Non-linear function

Domain:

$$-5.12 \le x_i \le 5.12 \quad \forall i \in 1,2,...,D$$

MATLAB function: simulannealbnd, particleswarm



$$f^* = 0 \quad x^* = [0 \quad 0]$$

simulannealbnd

$$\min_x f(x)$$
$$lb \le x \le ub$$
$$\textit{where } lb \textit{ and } ub \textit{ are scalars / vectors}$$
$$\textit{indicating the bounds of decision variables}$$

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

Input:

fun	Objective function handle
*x*0	Initial point of decision variables
lb	Lower bounds of decision variables
ub	Upper bounds of decision variables
options	Structure containing options such as display options, maximum iterations/function evaluations, plot functions, **initial temperature** etc.

Output:

x	Solution vector
fval	Objective function value at solution
exitflag	Algorithm stopping condition
output	Information such as number of iterations and function count, state of random number generator and exit message.

MATLAB code

Create a function file of the objective function

```
function F = Rastrigin(x)

D = length(x);
F = 0;

for d = 1:D
  F = F + x(d)^2 - 10*cos(2*pi*x(d)) + 10;
end
```

Create a script file to solve the problem using *simulannealbnd*

```
clc; clear
rng(1, 'twister') % For reproducibility
FUN = @Rastrigin; % Objective function handle
D = 2; % Dimension of the problem
LB = -5.12*ones(1,D); % Lower bounds
UB = 5.12*ones(1,D); % Upper bounds
x0 = 2*ones(1,D); % Initial point
% Calling the solver
[X, FVAL, EXITFLAG, OUTPUT] =
        simulannealbnd(FUN, x0, LB, UB);
```

$$f(x) = \sum_{i=1}^{D}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$$

$$-5.12 \le x_i \le 5.12 \quad \forall i \in 1,2,\ldots,D$$

Output:

X	FVAL
[$0.5724e^{-0.5}$ $0.0168e^{-0.5}$]	$6.5066e^{-09}$

particleswarm

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

$$lb \leq x \leq ub$$

where lb and ub are scalars / vectors indicating the bounds of decision variables

```
[x, fval, exitflag, output]  = particleswarm(fun,nvar,lb,ub,options)
```

Input:

fun	Objective function handle
nvar	Dimension of the function
lb	Lower bounds of decision variables
ub	Upper bounds of decision variables
options	Structure containing optimization options such as display options, maximum iterations/function evaluations, plot functions, inertia range, social and self adjustment coefficients etc.

Output:

x	Solution vector
fval	Objective function value at solution
exitflag	Algorithm stopping condition
output	Information such as number of iterations and function count, state of random number generator and exit message.

MATLAB code

Create a function file of the objective function

```
function F = Rastrigin(x)

D = length(x);
F = 0;

for d = 1:D
  F = F + x(d)^2 - 10*cos(2*pi*x(d)) + 10;
end
```

$$f(x) = \sum_{i=1}^{D}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$$

$$-5.12 \le x_i \le 5.12 \quad \forall i \in 1,2,\ldots,D$$

Create a script file to solve the problem using *particleswarm*

```
clc; clear
rng(1, 'twister') % For reproducibility
FUN = @Rastrigin; % Objective function handle
D = 2; % Dimension of the problem
LB = -5.12*ones(1,D); % Lower bounds
UB = 5.12*ones(1,D); % Upper bounds

% Calling the solver
[X, FVAL, EXITFLAG, OUTPUT] =
        particleswarm(FUN, D, LB, UB)
```

Output:

X	FVAL
[0. 0876$e^{-0.7}$ -0. 5884$e^{-0.7}$]	7.0344e^{-13}

Constrained Non-linear Programming

Linear/Non-linear objective function

Linear equality and inequality constraints
Non-linear equality and inequality constraints

Bound constraints

MATLAB function: fmincon, ga, patternsearch
Introduced before R2006a

Constrained Non-linear Programming

Minimize $f(X) = x_1 + x_2 + x_3$ **Objective function**

Subject to:

$$0.0025(x_4 + x_6) - 1 \le 0$$
$$0.0025(-x_4 + x_5 + x_7) - 1 \le 0$$
$$0.01(-x_5 + x_8) - 1 \le 0$$

Linear inequality constraints

$$100x_1 - x_1x_6 + 833.33252x_4 - 83333.333 \le 0$$
$$x_2x_4 - x_2x_7 - 1250x_4 + 1250x_5 \le 0$$
$$x_3x_5 - x_3x_8 - 2500x_5 + 1250000 \le 0$$

Non-linear inequality constraints

Bound constraints:

$$100 \le x_1 \le 10000$$
$$1000 \le x_2, x_3 \le 10000$$
$$10 \le x_i \le 1000, i = 4, 5, \ldots, 8$$

Bound constraints

MATLAB function: *fmincon*

$$A = \begin{bmatrix} 0 & 0 & 0 & 0.0025 & 0 & 0.0025 & 0 & 0 \\ 0 & 0 & 0 & -0.0025 & 0.0025 & 0 & 0.0025 & 0 \\ 0 & 0 & 0 & 0 & -0.01 & 0 & 0 & 0.01 \end{bmatrix} \quad b = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

fmincon

```
[x,fval,exitflag,output,lamda,grad,hessian] = fmincon(fun,X0,A,b,Aeq,beq,lb,ub,nonlcon,options)
```

$$\min_{x} f(x) \;\; subject \;\; to \begin{cases} c(x) \le 0 \\ ceq(x) = 0 \\ Ax \le b \\ Aeq\, x = beq \\ lb \le x \le ub \end{cases}$$

$f,\ x,\ b,\ beq,\ lb,$ and ub are vectors, and A and Aeq are matrices

$c(x)$ and $ceq(x)$ are functions that return vectors

Input:

fun	Objective function handle
X0	Initial point of the decision variables
A	Coefficients of linear inequality constraints
b	RHS of linear inequality constraints
Aeq	Coefficients of linear equality constraints
beq	RHS of linear equality constraints
lb	Lower bounds of decision variables
ub	Upper bounds of decision variables
nonlcon	Non linear constraints function handle
options	Structure containing options such as display options, maximum iterations/function evaluations, plot functions, choice of algorithm etc.

Output:

x	Solution vector
fval	Objective function value at solution
exitflag	Algorithm stopping condition
output	Information such as number of iterations, algorithm used, exit message etc.,
lambda	Lagrange multipliers at the solution
grad	Gradient at the solution
hessian	Approximate Hessian

MATLAB code

$$Min\ f(X) = x_1 + x_2 + x_3$$

$$100x_1 - x_1x_6 + 833.33252x_4 - 83333.333 \le 0$$
$$x_2x_4 - x_2x_7 - 1250x_4 + 1250x_5 \le 0$$
$$x_3x_5 - x_3x_8 - 2500x_5 + 1250000 \le 0$$

Create a function file of the objective function

```
function f = ObjectiveFunction(x)

f = sum(x(1:3));
```

Create a function file of the objective function

```
function [C, Ceq] = NLCon(x)

C(1) = 100*x(1) - x(1)*x(6) + 833.33252*x(4) - 83333.333;

C(2) = x(2)*x(4) - x(2)*x(7) - 1250*x(4) + 1250*x(5);

C(3) = x(3)*x(5) - x(3)*x(8) - 2500*x(5) + 1250000;

Ceq = []; % nonlinear equality constraints are absent
```

MATLAB code

Create a function file of the objective function

```
clc; clear;
A = [0 0 0 0.0025 0 0.0025 0 0;
0 0 0 -0.0025 0.0025 0 0.0025 0;
0 0 0 0 -0.01 0 0 0.01]; %Linear inequalityconstraints
b = ones(3,1); % RHS of linear inequality constraints

lb = [100 1000 1000 10*ones(1,5)]; % Lower bounds
ub = [10000 10000 10000 1000*ones(1,5)]; % Upper bounds
x0 = lb; % Initial point of variables

fun = @ObjectiveFunction; % Objective function handle
Nonlcon = @NLCon; % Nonlinear constraint function handle

% calling the solver
[X, FVAL] = fmincon(fun, x0, A, b, [], [], lb, ub, Nonlcon);
```

$$A = \begin{bmatrix} 0 & 0 & 0 & 0.0025 & 0 & 0.0025 & 0 & 0 \\ 0 & 0 & 0 & -0.0025 & 0.0025 & 0 & 0.0025 & 0 \\ 0 & 0 & 0 & 0 & -0.01 & 0 & 0 & 0.01 \end{bmatrix} \quad b = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$100 \le x_1 \le 10000$$
$$1000 \le x_2, x_3 \le 10000$$
$$10 \le x_i \le 1000, i = 4, 5, \ldots, 8$$

Output:

X	[579.31 1359.97 5109.97 182.02 295.60 217.98 286.42 395.60]
FVAL	7049.25

Constrained Non-linear Programming

- Maximize $f(x) = \left[\left(1 + x_1^2\right) + x_2^2\right]^{-1}$ **Objective function**

subject to

$$\left.\begin{aligned} x_1^2 + x_2^2 &\geq 4 \\ x_1^2 + x_2^2 &\leq 16 \end{aligned}\right\}$$ **Non-linear inequality constraints**

$$x_1 - x_2 = 3$$ **Linear equality constraints**

$$-10 \leq x_1, x_2 \leq 10$$ **Bound constraints**

- MATLAB function: *ga, patternsearch*

ga

```
[x,fval,exitflag,output,Population,Score] =
ga(fun,nvar,A,b,Aeq,beq,lb,ub,nonlcon,intcon,options)
```

$$\min_{x} f(x) \text{ subject to } \begin{cases} x(intcon) \text{ are integers} \\ c(x) \le 0 \\ c_{eq}(x) = 0 \\ Ax \le b \\ A_{eq}x = b_{eq} \\ lb \le x \le ub \end{cases}$$

intcon, b, beq, lb, and ub are vectors, and A and Aeq are matrices

c(x) and ceq(x) are functions that return vectors

Input:

fun	Objective function handle
nvar	Number of decision variables
A	Coefficients of linear inequality constraints
b	RHS of linear inequality constraints
A_{eq}	Coefficients of linear equality constraints
b_{eq}	RHS of linear equality constraints
lb	Lower bounds of decision variables
ub	Upper bounds of decision variables
nonlcon	Non linear constraints function handle
intcon	Indices of integer variables
options	Structure containing optimization options such as display options, maximum iterations/function evaluations, plot functions, choice of algorithm, function and probability of crossover and mutation, etc.

Output:

x	Solution vector
fval	Objective function value at solution
exitflag	Algorithm stopping condition
output	Information such as number of iterations, algorithm used, exit message, state of random number generator etc.,
Population	Final population
Score	Objective function value of final population

MATLAB code

Create a function file of the objective function

```
function f = ObjectiveFn(x)

f = -((1 + x(1)^2) + x(2)^2)^-1;
```

Create a function file of the objective function

```
function [c, ceq] = NLConstraints(x)

ceq = [];
c(1) = 4 - x(1)^2 - x(2)^2;
c(2) = x(1)^2 + x(2)^2 - 16;
```

Output:

X	[1.4994 -1.4996]
FVAL	-0.1819

Create a script file to solve the problem using *ga*

```
clc; clear;
rng(1,'twister')
fun = @ObjectiveFn; % Objective function handle
Aeq = [1 -1]; % Coefficients of equality constraints
beq = 3; % RHS of equality constraints
lb = [-10 -10]; % Lower bound
ub = [10 10];% Upper bound
nvar = length(lb); % No. of decision variables
nonlcon = @NLConstraints;% Nonlinear constraint
function handle
% Calling the solver
[x, fval] = ga(fun, nvar, [], [], Aeq, beq,
              lb, ub, nonlcon);
```

$$f(x) = \left[\left(1 + x_1^2\right) + x_2^2\right]^{-1}$$
$$x_1^2 + x_2^2 \geq 4$$
$$x_1^2 + x_2^2 \leq 16$$
$$x_1 - x_2 = 3$$
$$-10 \leq x_1, x_2 \leq 10$$

patternsearch

```
[x,fval,exitflag,output] = patternsearch(fun,X0,
        A,b,Aeq,beq,lb,ub,nonlcon,options)
```

$$\min_x f(x) \quad subject \;\; to \begin{cases} c(x) \le 0 \\ c_{eq}(x) = 0 \\ Ax \le b \\ A_{eq}x = b_{eq} \\ lb \le x \le ub \end{cases}$$

f, x, b, beq, lb, and ub are vectors, and A and Aeq are matrices
$c(x)$ and $ceq(x)$ are functions that return vectors

Input:

fun	Objective function handle
X0	Initial point of the decision variables
A	Coefficients of linear inequality constraints
b	RHS of linear inequality constraints
A_{eq}	Coefficients of linear equality constraints
b_{eq}	RHS of linear equality constraints
lb	Lower bounds of decision variables
ub	Upper bounds of decision variables
nonlcon	Non linear constraints function handle
options	Structure containing options such as display options, maximum iterations/function evaluations, plot functions, choice of algorithm, tolerance and maximum size of mesh etc.

Output:

X	Solution vector
fval	Objective function value at solution
exitflag	Algorithm stopping condition
output	Information such as number of iterations, algorithm used, exit message etc.,

MATLAB code

Create a function file of the objective function

```
function f = ObjectiveFn(x)

f = -((1 + x(1)^2) + x(2)^2)^-1;
```

Create a function file of the objective function

```
function [c, ceq] = NLConstraints(x)

ceq = [];
c(1) = 4 - x(1)^2 - x(2)^2;
c(2) = x(1)^2 + x(2)^2 - 16;
```

Output:

X	[1.4995 -1.4995]
FVAL	-0.1819

Create a script file to solve the problem using *patternsearch*

```
clc; clear
rng(1,'twister') % For reproducibility

fun = @ObjectiveFn; % Objective function handle
Aeq = [1 -1]; % Coefficients of equality constraints
beq = 3; % RHS of equality constraints
lb = [-10 -10]; % Lower bound
ub = [10 10];% Upper bound
x0 = [-3 -3];% Initial point
nonlcon = @NLConstraints;% Nonlinear constraint function
handle
% Calling the solver
[x, fval] = patternseach(fun, nvar, [], [],
                Aeq, beq, lb, ub, nonlcon);
```

$$f(x) = \left[\left(1 + x_1^2\right) + x_2^2\right]^{-1}$$
$$x_1^2 + x_2^2 \geq 4$$
$$x_1^2 + x_2^2 \leq 16$$
$$x_1 - x_2 = 3$$
$$-10 \leq x_1, x_2 \leq 10$$

Constrained Mixed Integer Non-linear Programming

Linear/Non-linear objective function

Linear equality and inequality constraints
&
Non-linear equality and inequality constraints

Bound constraints
&
Integer variables

MATLAB function: ga
Introduced before R2006a

Constrained Mixed Integer Non-linear Programming

$$Minimize\ f(x,y) = -y + 2x - \ln\left(\frac{x}{2}\right)$$

Non-linear objective function

subject to

$$-x - \ln\left(\frac{x}{2}\right) + y \le 0$$

Non-linear inequality constraints

$$0.5 \le x \le 1.5$$
$$y \in \{0,1\}$$

Bound constraints

- MATLAB function: *ga*

MATLAB code

$$\text{Minimize } f(x,y) = -y + 2x - \ln\left(\frac{x}{2}\right)$$

$$-x - \ln\left(\frac{x}{2}\right) + y \le 0$$

$$0.5 \le x \le 1.5$$

$$y \in \{0,1\}$$

Create a function file of the objective function

```
function f = ObjFunInt(X)
% x = X(1) and y = X(2)
f = -X(2)+2*X(1)-log(X(1)/2);
```

Create a function file of the objective function

```
function [c,ceq] = nonlconInt(X)
% x = X(1) and y = X(2)
c = -X(1)-log(X(1)/2)+X(2);
ceq = [];
```

Create a script file to solve the problem using *ga*

```
clc;clear
rng(1,'twister') % For reproducibility

fun = @ObjFunInt; % objective function handle
lb = [0.5,0]; % Lower bound
ub = [1.5,1]; % Upper bound

intcon = 2; % Index of integer variable
nvar = length(lb); % Problem dimension

nonlcon = @nonlconInt;% Nonlinear constraint function
handle

[x, fval] = ga(fun, nvar, [], [], [], [], lb,
               ub, nonlcon, intcon);
```

Output:

X	[1.3770 1]
FVAL	2.1272

Restriction of *ga*

- *ga* cannot solve a problem with equality constraints and integer variables

> **Note**
>
> When `IntCon` is nonempty, `Aeq` and `beq` must be an empty entry (`[]`), and `nonlcon` must return empty for `ceq`. For more information on integer programming, see Mixed Integer Optimization.

- No CreationFcn option, CrossoverFcn option, MutationFcn option, InitialScoreMatrix option.

- To obtain integer variables, ga uses special creation, crossover, and mutation functions.

- *ga* uses only the binary tournament selection function (SelectionFcn option)

- There are no hybrid functions that support integer constraints. So *ga* does not use hybrid functions when there are integer constraints

Mixed Integer Optimization https://in.mathworks.com/help/gads/mixed-integer-optimization.html#bs1cifg

Solving MINLP using ga

Problem type	$\begin{aligned}&\text{minimize } f(x)\\&\text{subject to}\\&\quad Ax\le b\\&\quad A_{eq}x=b_{eq}\\&\quad C(x)\le 0\\&\quad C_{eq}(x)=0\\&x(intcon)\text{ are integer variable}\end{aligned}$	$\begin{aligned}&\text{minimize } f(x)\\&\text{subject to}\\&\quad Ax\le b\\&\quad A_{eq}x=b_{eq}\\&\quad C(x)\le 0\\&x(intcon)\text{ are integer variable}\end{aligned}$	$\begin{aligned}&\text{minimize } f(x)\\&\text{subject to}\\&\quad Ax\le b\\&\quad C(x)\le 0\\&\quad C_{eq}(x)=0\\&x(intcon)\text{ are integer variable}\end{aligned}$	$\begin{aligned}&\text{minimize } f(x)\\&\text{subject to}\\&\quad Ax\le b\\&\quad C(x)\le 0\\&x(intcon)\text{ are integer variable}\end{aligned}$	$\begin{aligned}&\text{minimize } f(x)\\&\text{subject to}\\&\quad Ax\le b\\&\quad A_{eq}x=b_{eq}\\&\quad C(x)\le 0\\&\quad C_{eq}(x)=0\\&x\text{ are continuous variable}\end{aligned}$
Solve using ga	✘	✘	✘	✓	✓
	MINLP problem	MINLP problem without nonlinear equality constraints	MINLP problem without linear equality constraints	MINLP problem without equality constraints	NLP problem No integer variables

Input and output arguments of different solvers

[x, fval, exitflag, output, lambda] = **linprog**(Z, A, b, Aeq, beq, lb, ub, options)

[x, fval, exitflag, output] = **intlinprog**(Z, intcon, A, b, Aeq, beq, lb, ub, x0, options)

[x, fval, exitflag, output, lambda] = **quadprog**(H, f, A, b, Aeq, beq, lb, ub, x0, options)

[x, fval, exitflag, output, grad, hessian] = **fminunc**(fun, x0, options)

[x, fval, exitflag, output] = **fminsearch**(fun, x0, options)

[x, fval, exitflag, output] = **simulannealbnd**(fun, x0, lb, ub, options)

[x, fval, exitflag, output] = **particleswarm**(fun, nvar, lb, ub, options)

[x, fval, exitflag, output, lamda, grad, hessian] = **fmincon**(fun, x0, A, b, Aeq, beq, lb, ub, nonlcon, options)

[x, fval, exitflag, output] = **patternsearch**(fun, x0, A, b, Aeq, beq, lb, ub, nonlcon, options)

[x, fval, exitflag, output, population, score] = **ga**(fun, nvar, A, b, Aeq, beq, lb, ub, nonlcon, intcon, options)

Optimization options

➢Optimization options are specified as the output of the structure 'optimoptions'.

➢Can be used to modify or view the default setting of any optimization solvers.

➢Consider minimization of Rastrigin function using *ga*.

$$f(x) = \sum_{i=1}^{D} \left[x_i^2 - 10\cos(2\pi x_i) + 10 \right]$$

$$-5.12 \le x_i \le 5.12 \quad \forall i \in 1, 2, ..., D$$

➢Using optimoptions,

- change crossover probability to 0.6,
- include plot function to plot best iteration versus objective value.

Optimization options

Options for ga, Integer ga, and gamultiobj

Option	Description	Values
ConstraintTolerance	Determines the feasibility with respect to nonlinear constraints. Also, max(sqrt(eps),ConstraintTolerance) determines feasibility with respect to linear constraints. For an options structure, use TolCon.	Positive scalar | {1e-3}
CreationFcn	I* Function that creates the initial population. Specify as a name of a built-in creation function or a function handle. See Population Options.	{'gacreationuniform'} | {'gacreationlinearfeasible'}* | Custom creation function
CrossoverFcn	I* Function that the algorithm uses to create crossover children. Specify as a name of a built-in crossover function or a function handle. See Crossover Options.	{'crossoverscattered'} for ga, {'crossoverintermediate'}* for gamultiobj | 'crossoverheuristic' | 'crossoversinglepoint' | 'crossovertwopoint' | 'crossoverarithmetic' | Custom crossover function
CrossoverFraction	The fraction of the population at the next generation, not including elite children, that the crossover function creates.	Positive scalar | {0.8}
Display	Level of display.	'off' | 'iter' | 'diagnose' | {'final'}
DistanceMeasureFcn	Function that computes distance measure of individuals. Specify as a name of a built-in distance measure function or a function handle. The value applies to decision variable or design space (genotype) or to function space (phenotype). The default 'distancecrowding' is in function space (phenotype). For gamultiobj only. See Multiobjective Options. For an options structure, use a function handle, not a name.	{'distancecrowding'} means the same as {@distancecrowding,'phenotype'} | {@distancecrowding,'genotype'} | Custom distance function
EliteCount	**NM** Positive integer specifying how many individuals in the current generation are guaranteed to survive to the next generation. Not used in gamultiobj.	Positive integer | {ceil(0.05*PopulationSize)} | {0.05*(default PopulationSize)} for mixed-integer problems
FitnessLimit	**NM** If the fitness function attains the value of FitnessLimit, the algorithm halts.	Scalar | {-Inf}
FitnessScalingFcn	Function that scales the values of the fitness function. Specify as a name of a built-in scaling function or a function handle. Option unavailable for gamultiobj.	{'fitscalingrank'} | 'fitscalingshiftlinear' | 'fitscalingprop' | 'fitscalingtop' | Custom fitness scaling function
FunctionTolerance	The algorithm stops if the average relative change in the best fitness function value over MaxStallGenerations generations is less than or equal to FunctionTolerance. If StallTest is 'geometricWeighted', then the algorithm stops if the *weighted* average relative change is less than or equal to FunctionTolerance. For gamultiobj, the algorithm stops when the geometric average of the relative change in value of the spread over options.MaxStallGenerations generations is less than options.FunctionTolerance, and the final spread is less than the mean spread over the past options.MaxStallGenerations generations. See gamultiobj Algorithm. For an options structure, use TolFun.	Positive scalar | {1e-6} for ga, {1e-4} for gamultiobj
HybridFcn	I* Function that continues the optimization after ga terminates. Specify as a name or a function handle. Alternatively, a cell array specifying the hybrid function and its options. See ga Hybrid Function. For gamultiobj, the only hybrid function is @fgoalattain. See gamultiobj Hybrid Function. See When to Use a Hybrid Function.	Function name or handle | 'fminsearch' | 'patternsearch' | 'fminunc' | 'fmincon' | {[]} or 1-by-2 cell array | {@solver, hybridoptions}, where solver = fminsearch, patternsearch, fminunc, or fmincon {[]}
InitialPenalty	**NM** I* Initial value of penalty parameter	Positive scalar | {10}
InitialPopulationMatrix	Initial population used to seed the genetic algorithm. Has up to PopulationSize rows and N columns, where N is the number of variables. You can pass a partial population, meaning one with fewer than PopulationSize rows. In that case, the genetic algorithm uses CreationFcn to generate the remaining population members. See Population Options For an options structure, use InitialPopulation.	Matrix | {[]}

MATLAB code

Create a function file of the objective function

```
function F = Rastrigin(x)

D = length(x);
F = 0;

for d = 1:D
   F = F + x(d)^2 - 10*cos(2*pi*x(d))
       + 10;
end
```

Create a script file to solve the problem using *ga*

```
clc; clear
rng(1, 'twister') % For reproducibility

FUN = @Rastrigin; % Objective function handle

D = 2; % Dimension of the problem
LB = -5.12*ones(1,D); % Lower bounds
UB = 5.12*ones(1,D); % Upper bounds
% Change the default settings
options = optimoptions('ga','PlotFcn',
@gaplotbestf,'CrossoverFraction',0.6);

% Calling the solver
[x, fval] = ga(FUN, D, [], [], [], [], LB,
              UB, [], [], options);
```

$$f(x) = \sum_{i=1}^{D}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$$

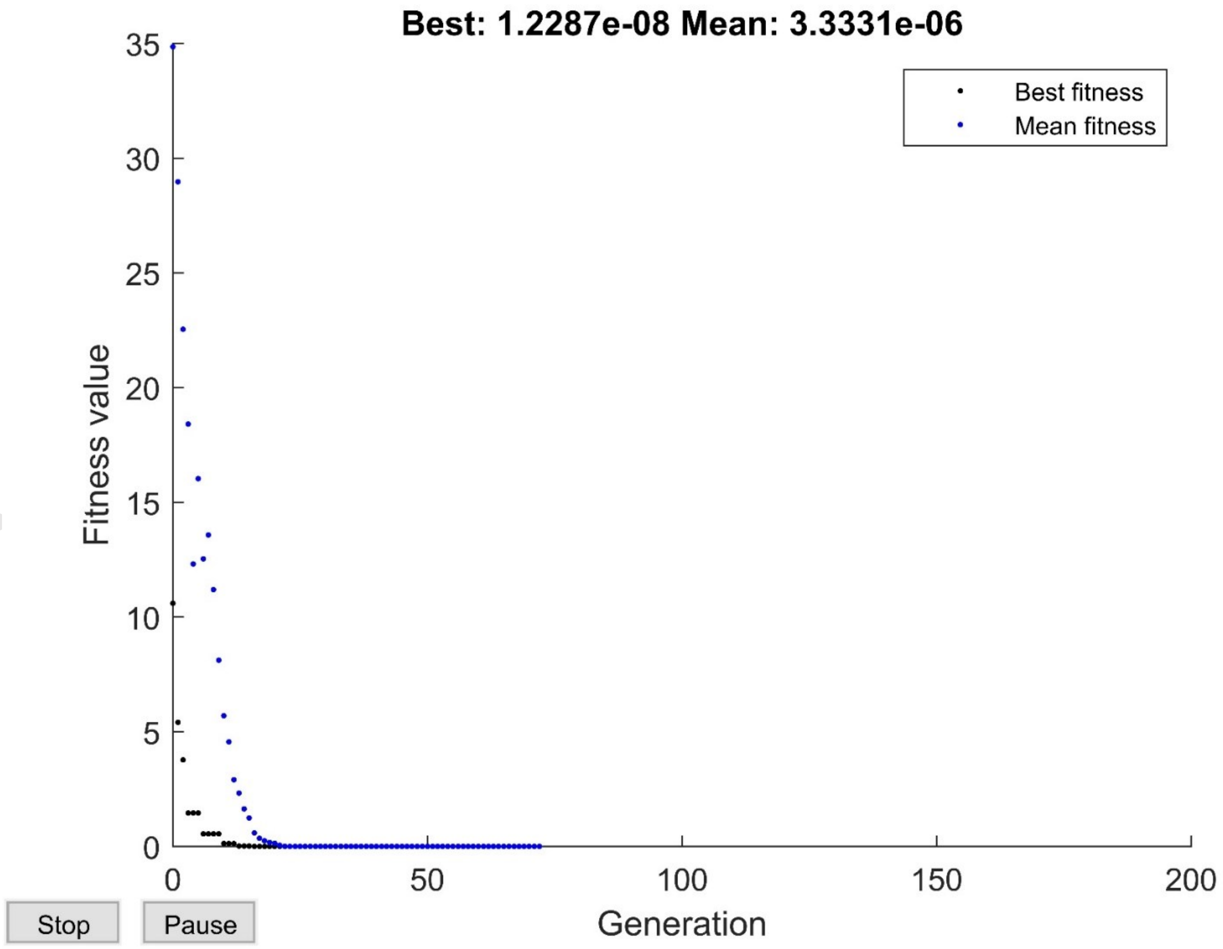
$$-5.12 \le x_i \le 5.12 \quad \forall i \in 1,2,...,D$$

Results

Output:

X $[0.7856e^{-05} \quad -0.0458e^{-05}]$

FVAL $1.2287e^{-08}$



OutputFcn in optimization toolbox

- To retrieve output from an optimization algorithm in every iteration

- Syntax: options = optimoptions(@solvername, 'OutputFcn', @outputfunction)

- For *ga,* MATLAB passes the options, state, and flag data to the output function, and it returns state, options, and optchanged data.

  - **options:** structure containing the settings used in ga.
  - **state:** Structure containing information such as generation, start time, stop flag, best score in each generation, current population and scores etc. about the current generation.
  - **flag:** current status ('init', 'iter', 'done' etc.) of the algorithm

- Output function syntax: [state,options,optchanged] = outputfunction(options,state,flag)

- optchanged: flag indicating changes to options (if options are changed, optchanged = 1 else optchanged = 0)

MATLAB code

Create a function file of the objective function

```
function F = Rastrigin(x)

D = length(x);
F = 0;

for d = 1:D
    F = F + x(d)^2 -
        10*cos(2*pi*x(d)) + 10;
end
```

$$f(x)=\sum_{i=1}^{D}\left[x_i^2-10\cos(2\pi x_i)+10\right]$$

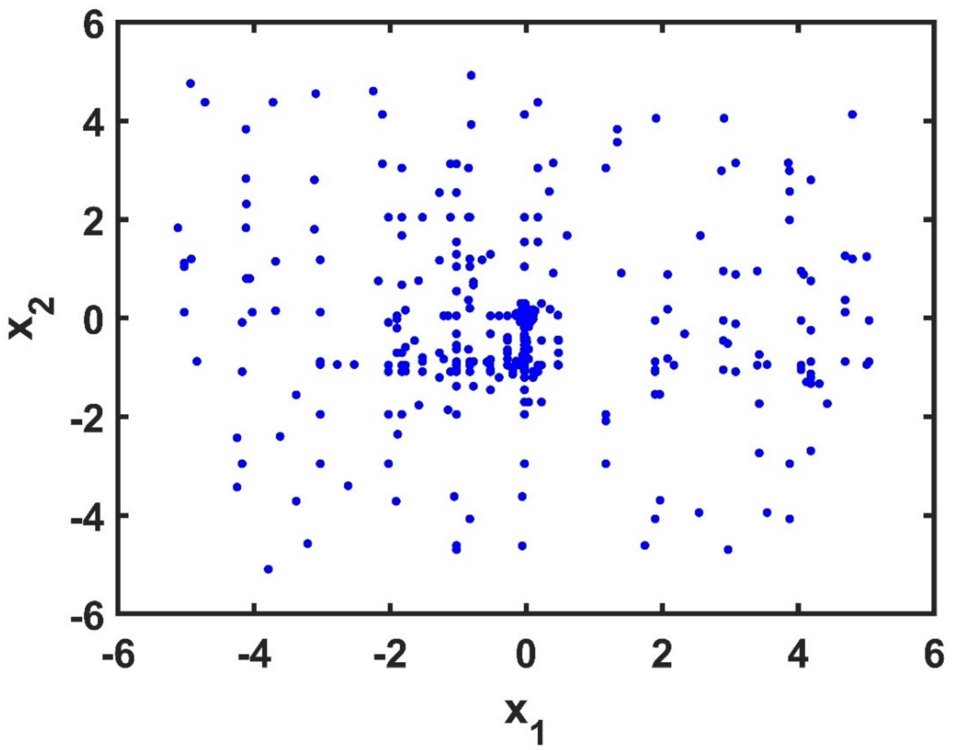
Create an output function which plots the population in every generation

```
function [state, options, optchanged] =
outputFnExample(options, state, flag)

optchanged = false; % Flag to indicate the change in options

currentPop = state.Population;% current population
plot(currentPop(:,1), currentPop(:,2), ‘b.');
hold on;
drawnow
xlabel('x_1')
ylabel('x_2')
end
```

MATLAB code

Create a script file to solve the problem using *ga*

```
clc;clear
rng(1, 'twister') % For reproducibility
FUN = @Rastrigin; % Objective function handle
D = 2; % Dimension of the problem
LB = -5.12*ones(1,D); % Lower bounds of the problem
UB = 5.12*ones(1,D); % Upper bounds of the problem
% Change the default settings using optimoptions
options = optimoptions('ga', 'OutputFcn',
@outputFnExample, 'CrossoverFraction', 0.6);
% Calling the solver
[x, fval] = ga(FUN, D, [], [], [], [],LB, UB,
            [], [], options);
```



$$-5.12 \le x_i \le 5.12 \quad \forall i \in 1,2,...,D$$

Output:

x $[0.7856e^{-05} \;\; -0.0458e^{-05}]$

fval $1.2287e^{-08}$

Vectorization of fitness function

- Vectorization increases the speed of execution.
- For using the vectorized option, the fitness function must
  - accept a matrix with arbitrary number of rows (population)
  - return the fitness vector (fitness of the population)
- Vectorized code of Rastrigin function

$$f(x) = 10D + \sum_{i=1}^{D}\left[x_i^2 - 10\cos(2\pi x_i)\right]$$

$$-5.12 \le x_i \le 5.12 \quad \forall i \in 1,2,\ldots,D$$

```
function F = RastriginVectorized(X)
[N,D] = size(X); % Number of decision variables
F = zeros(N,1); % Initial value of fitness
for n = 1:N
    for d = 1:D
        F(n) = F(n) + X(n, d)^2 - 10*cos(2*pi*X(n,d)) + 10;
    end
end
```

Vectorized Code ⇦ Accepts X matrix (N x D) Returns F vector (N x 1)

```
function F = Rastrigin(x)
D = length(x); % Number of decision variables
F = 0; % Initial value of fitness
for d = 1:D
    F = F + x(d)^2 - 10*cos(2*pi*x(d)) + 10;
end
```

Not a vectorized Code ⇦ Accepts x vector (1 x D) Returns F scalar (1 x 1)

MATLAB code

Create a script file to solve the problem using ***ga***

```
clc;clear
rng(1, 'twister') % For reproducibility
D = 20; % Dimension of the problem
LB = -5.12*ones(1,D); % Lower bounds of the problem
UB = 5.12*ones(1,D); % Upper bounds of the problem
% Calling the solver without vectorization option
options = optimoptions('ga','PopulationSize',2000);
FUN = @Rastrigin; % Objective function handle

tic
[x,fval] = ga(FUN, D , [], [], [], [], LB, UB,[],[],options);
noVec = toc

% Calling the solver with vectorization option
options = optimoptions('ga','UseVectorized',true,'PopulationSize',2000);
FUN = @RastriginVectorized; % Objective function handle

tic
[x, fval] = ga(FUN, D, [], [], [], [], LB, UB, [], [], options);
vec = toc
```

Comparison of elapsed time		
Use vectorized	false	true
Elapsed time	17.28	13.65

Machine configuration
Intel core i7 @3.4GHz with 24 GB RAM

Parallel evaluation of fitness function

- Option is available to compute the fitness and non-linear constraint function in parallel.
- Cannot use vectorized and parallel computation options simultaneously.

	UseVectorized = false	**UseVectorized = true**
UseParallel = false	Serial	Vectorized
UseParallel = true	Parallel	Vectorized

- For using the parallel option, the fitness and non-linear constraint functions need not be vectorized.
- Syntax: options = optimoptions(@SolverName, 'UseParallel', true, 'UseVectorized', false);
- Parallel option available in solvers such as ga, particleswarm, patternsearch, etc.
- Demo of optimizing an ODE in parallel: https://in.mathworks.com/help/gads/optimize-an-ode-in-parallel.html

https://in.mathworks.com/help/gads/genetic-algorithm-options.html#f17234

MATLAB code

Create a function file of the objective function

```
function F = Rastrigin(x)

D = length(x);
F = 0;

for d = 1:D
  F = F + x(d)^2 -  10*
      cos(2*pi*x(d)) + 10;

end

pause(0.01) % simulate an expensive function by pausing
```

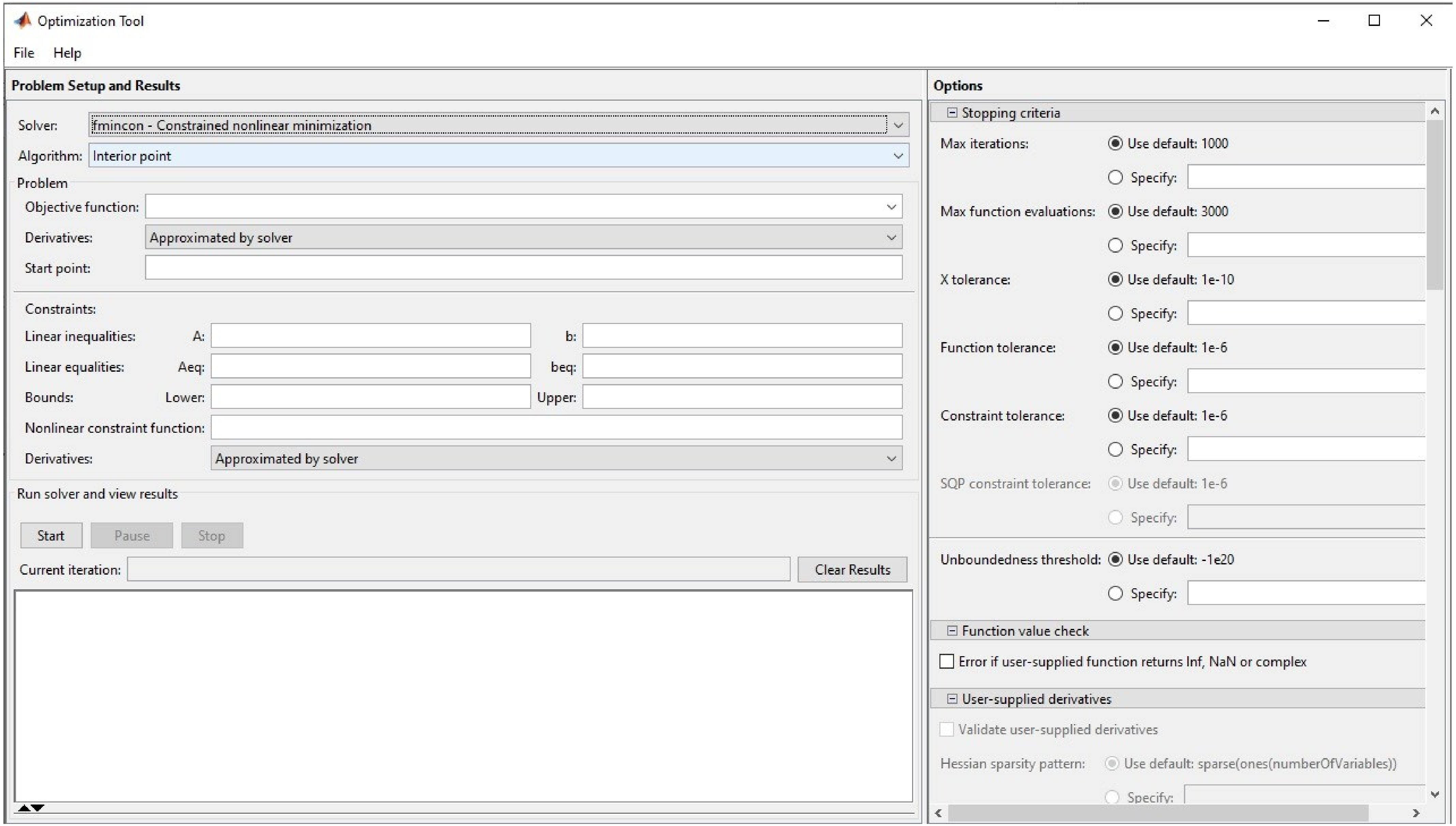
Comparison of elapsed time		
Use parallel	false	true
Elapsed time	853.026	123.61

Create a script file to solve the problem using *ga*

```
clc;clear
rng(1, 'twister') % For reproducibility

FUN = @Rastrigin; % Objective function handle
D = 20; % Dimension of the problem
LB = -5.12*ones(1,D); % Lower bounds of the problem
UB = 5.12*ones(1,D); % Upper bounds of the problem

tic
[x, fval] = ga(FUN, D , [], [], [], [], LB, UB);
woPar = toc;

options = optimoptions('ga', 'UseParallel', true,
          'UseVectorized', false);

parpool % To start parallel pool
tic
[x, fval] = ga(FUN, D, [], [], [], [], LB, UB, [], [], options);
wPar = toc;
```

Machine configuration
Intel core i7 @3.4GHz with 24 GB RAM

Optimtool


Optimization Tool
File Help
Problem Setup and Results
Solver: fmincon - Constrained nonlinear minimization
Algorithm: Interior point
Problem
Objective function:
Derivatives: Approximated by solver
Start point:
Constraints:
Linear inequalities: A: b:
Linear equalities: Aeq: beq:
Bounds: Lower: Upper:
Nonlinear constraint function:
Derivatives: Approximated by solver
Run solver and view results
Start Pause Stop
Current iteration: Clear Results
Options
Stopping criteria
Max iterations: Use default: 1000
Specify:
Max function evaluations: Use default: 3000
Specify:
X tolerance: Use default: 1e-10
Specify:
Function tolerance: Use default: 1e-6
Specify:
Constraint tolerance: Use default: 1e-6
Specify:
SQP constraint tolerance: Use default: 1e-6
Specify:
Unboundedness threshold: Use default: -1e20
Specify:
Function value check
Error if user-supplied function returns Inf, NaN or complex
User-supplied derivatives
Validate user-supplied derivatives
Hessian sparsity pattern: Use default: sparse(ones(numberOfVariables))
Specify:

Constrained Multi-objective Optimization

Linear/Non-linear objective function

Linear equality and inequality constraints
&
Non-linear equality and inequality constraints

Bound constraints

MATLAB functions:
gamultiobj (Introduced in R2007b)
paretosearch (Introduced in R2018b)

gamultiobj

```
[x,fval,exitflag,output,Population,Score] = 
gamultiobj(fun,nvar,A,b,Aeq,beq,lb,ub,nonlcon,options)
```

$$\min_{x} f_i(x) \quad i = 1,2,\ldots,M$$

$$subject\ \ to \begin{cases} c(x) \le 0 \\ c_{eq}(x) = 0 \\ Ax \le b \\ A_{eq}x = b_{eq} \\ lb \le x \le ub \end{cases}$$

$b,\ beq,\ lb,\ and\ ub\ are\ vectors,\ and\ A\ and\ Aeq\ are\ matrices$

$c(x)\ and\ ceq(x)\ are\ functions\ that\ return\ vectors$

Input:

fun	Objective function handle
nvar	Number of decision variables
A	Coefficients of linear inequality constraints
b	RHS of linear inequality constraints
A_{eq}	Coefficients of linear equality constraints
b_{eq}	RHS of linear equality constraints
lb	Lower bounds of decision variables
ub	Upper bounds of decision variables
nonlcon	Non linear constraints function handle
options	Structure containing optimization options such as display options, maximum iterations/function evaluations, plot functions, choice of algorithm, function and probability of crossover and mutation, etc.

Output:

x	Solution vector
fval	Objective function value at solution
exitflag	Algorithm stopping condition
output	Information such as number of iterations, algorithm used, exit message, state of random number generator etc.,
Population	Final population
Score	Objective function value of final population

Multi-objective optimization problem (KUR)

$$\text{minimize } f_1(x) = \sum_{i=1}^{D-1} -10\exp\left(-0.2\sqrt{x_i^2 + x_{i+1}^2}\right)$$

$$\text{minimize } f_2(x) = \sum_{i=1}^{D-1} \left|x_i\right|^{0.8} + 5\sin\left(x_i^3\right)$$

$$-5 \le x_i \le 5 \quad i = 1, 2, 3$$

Objective functions

Multi-objective Optimization using evolutionary algorithms, Wiley
https://in.mathworks.com/help/gads/gamultiobj.html

MATLAB code

Create a function file of the objective function

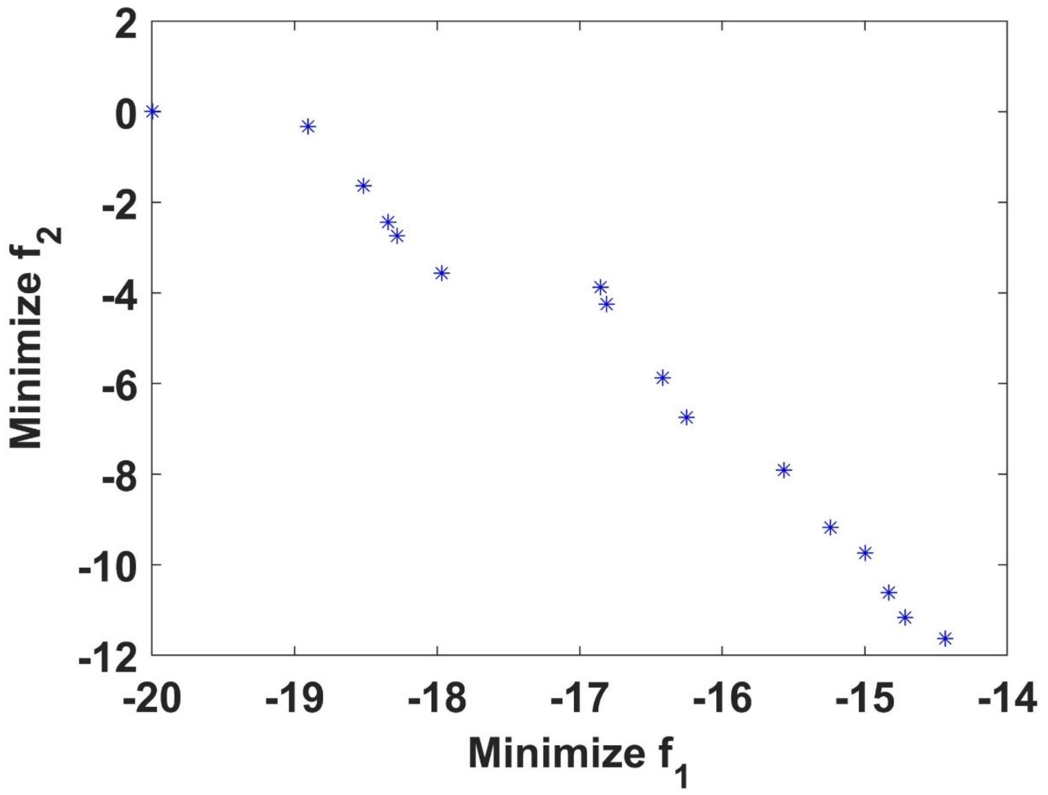
```
function f = kur_MO(x)

D = length(x);

f(1) = sum(-10*exp(-0.2*sqrt(x(1:D-1).^2 + x(2:D).^2)));
f(2) =  sum(abs(x(1:D)).^0.8 + 5*sin(x(1:D).^3));
```

$$\text{minimize } f_1(x) = \sum_{i=1}^{D-1} -10\exp\left(-0.2\sqrt{x_i^2 + x_{i+1}^2}\right)$$

$$\text{minimize } f_2(x) = \sum_{i=1}^{D-1} |x_i|^{0.8} + 5\sin\left(x_i^3\right)$$

Create a script file to solve the problem using *gamultiobj*

```
clc;clear

ub = [5 5 5];
lb = -ub;
nvars = length(lb);
fitnessfcn = @kur_MO;

[x,fval,exitflag,output,population,scores] = ...
gamultiobj(fitnessfcn,nvars,[],[],[],[],lb,ub);

plot(fval(:,1),fval(:,2),'b*')
xlabel('Minimize f_1')
ylabel('Minimize f_2')
```



paretoseach

```
[x,fval,exitflag,output,residuals]=
paretosearch(fun,nvar,A,b,Aeq,beq,lb,ub,nonlcon,options)
```

$$\min_{x} f_i(x) \quad i = 1,2,\ldots M$$

$$\text{subject to} \begin{cases} c(x) \le 0 \\ c_{eq}(x) = 0 \\ Ax \le b \\ A_{eq}x = b_{eq} \\ lb \le x \le ub \end{cases}$$

b, beq, lb, and ub are vectors, and A and Aeq are matrices

c(x) and ceq(x) are functions that return vectors

Input:

fun	Objective function handle
nvar	Number of decision variables
A	Coefficients of linear inequality constraints
b	RHS of linear inequality constraints
A_{eq}	Coefficients of linear equality constraints
b_{eq}	RHS of linear equality constraints
lb	Lower bounds of decision variables
ub	Upper bounds of decision variables
nonlcon	Non linear constraints function handle
options	Structure containing optimization options such as display options, maximum iterations/function evaluations, plot functions, choice of algorithm, function and probability of crossover and mutation, etc.

Output:

x	Solution vector
fval	Objective function value at solution
exitflag	Algorithm stopping condition
output	Information such as number of iterations, algorithm used, exit message, state of random number generator etc.,
residuals	structure containing the constraint values at the solution points x

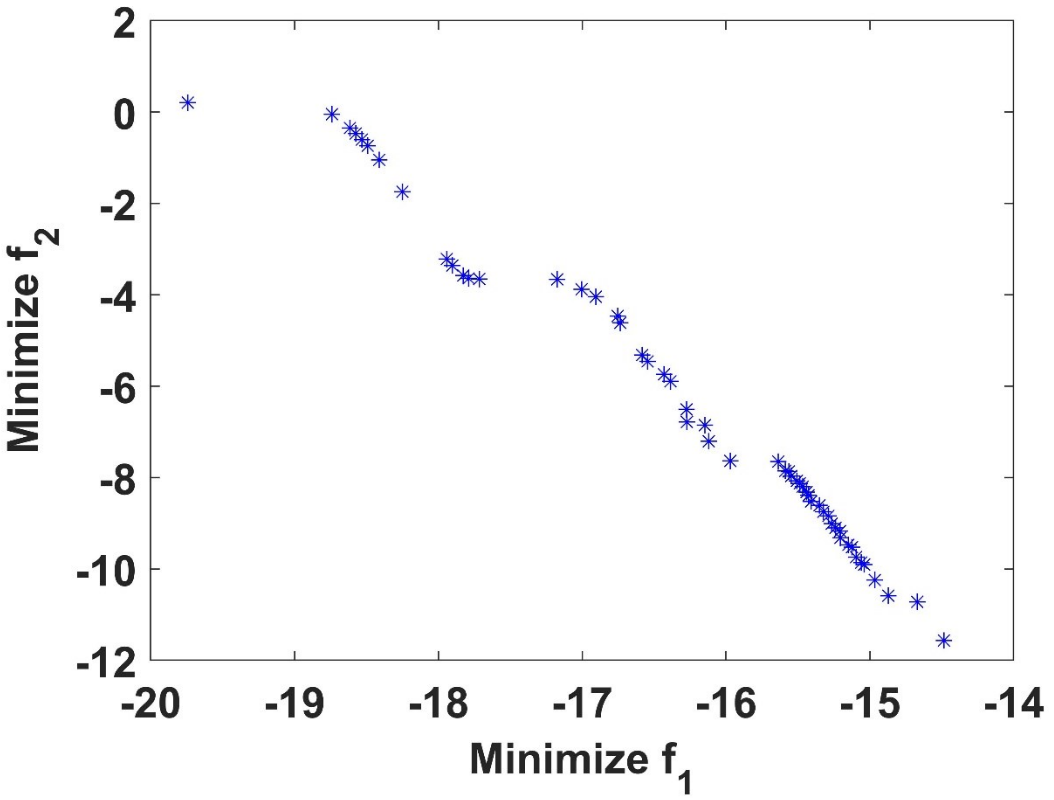
MATLAB code

Create a function file of the objective function

```
function f = kur_MO(x)

D = length(x);

f(1) = sum(-10*exp(-0.2*sqrt(x(1:D-1).^2 + x(2:D).^2)));
f(2) =  sum(abs(x(1:D)).^0.8 + 5*sin(x(1:D).^3));
```

Create a script file to solve the problem using ***paretosearch***

```
clc;clear

ub = [5 5 5];
lb = -ub;
nvars = length(lb);
fitnessfcn = @kur_MO;

[x,fval,exitflag,output,residuals] = ...
paretosearch(fitnessfcn,nvars,[],[],[],[],lb,ub);

plot(fval(:,1),fval(:,2),'b*')
xlabel('Minimize f_1')
ylabel('Minimize f_2')
```



Closure

- MATLAB optimization solvers for
  - LP problems – *linprog*
  - MILP problems – *intlinprog*
  - Unconstrained NLP problems – *fminunc* and *fminsearch*
  - Bound constrained NLP problems – *simulannealbnd* and *particleswarm*
  - Constrained NLP problems – *fmincon, patternsearch* and *ga*
  - MINLP problems (without equality constraints) – *ga*
- Options in optimization toolbox
- Output function in optimization toolbox
- Vectorization of fitness function
- Parallel evaluation of fitness function
- Multi-objective optimization problems – *gamultiobj* and *paretosearch*

Optimization software

Solver	LP	MILP	NLP	MINLP	Constraint Programming
IBM ILOG CPLEX Optimization Studio	✓	✓	✗	✗	✓
GAMS	✓	✓	✓	✓	✗
MATLAB Optimization Toolbox	✓	✓	✓	✓*	✗

* MINLP solver in MATLAB (ga) cannot solve an MINLP problem if integer constraints are present

MATLAB: https://in.mathworks.com/products/get-matlab.html?s_tid=gn_getml

GAMS: https://www.gams.com/download/

IBM ILOG CPLEX : https://www.ibm.com/in-en/products/ilog-cplex-optimization-studio

Thank You !!!