

# NUMERICAL OPTIMIZATION

KON428E Introduction to Optimal Control

Prof. Dr. İbrahim EKSİN

Prof. Dr. Müjde GÜZELKAYA

Araş. Gör. Emre DİNCEL

# Outline

- **Numerical Optimization**
  - Hooke & Jeeves Algorithm
  - Genetic Algorithm
  - Big Bang-Big Crunch Optimization
- **Discussion About The Final Project**

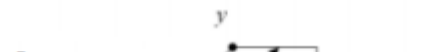
# Numerical Optimization

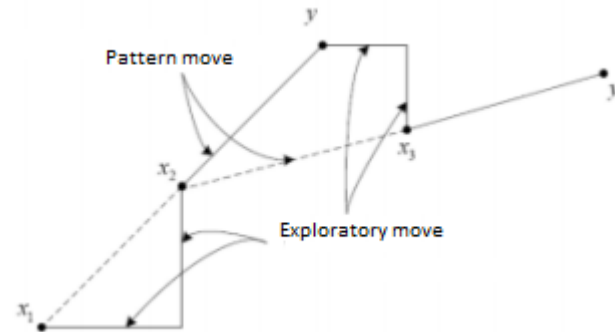
- **Optimization**
  - The selection of a best element (with regard to some criteria) from some set of available alternatives.
- **Numerical Optimization**
  - Fitness Function
  - Constraint Function
  - Iterative Solution
$$x(k) = x(k-1) + \text{correction function}$$

# Numerical Optimization

- **Local Optimization Methods**
  - Hooke & Jeeves Algorithm
- **Global Optimization Methods**
  - Genetic Algorithm
  - Big Bang – Big Crunch Optimization Algorithm

# Hooke & Jeeves Algorithm

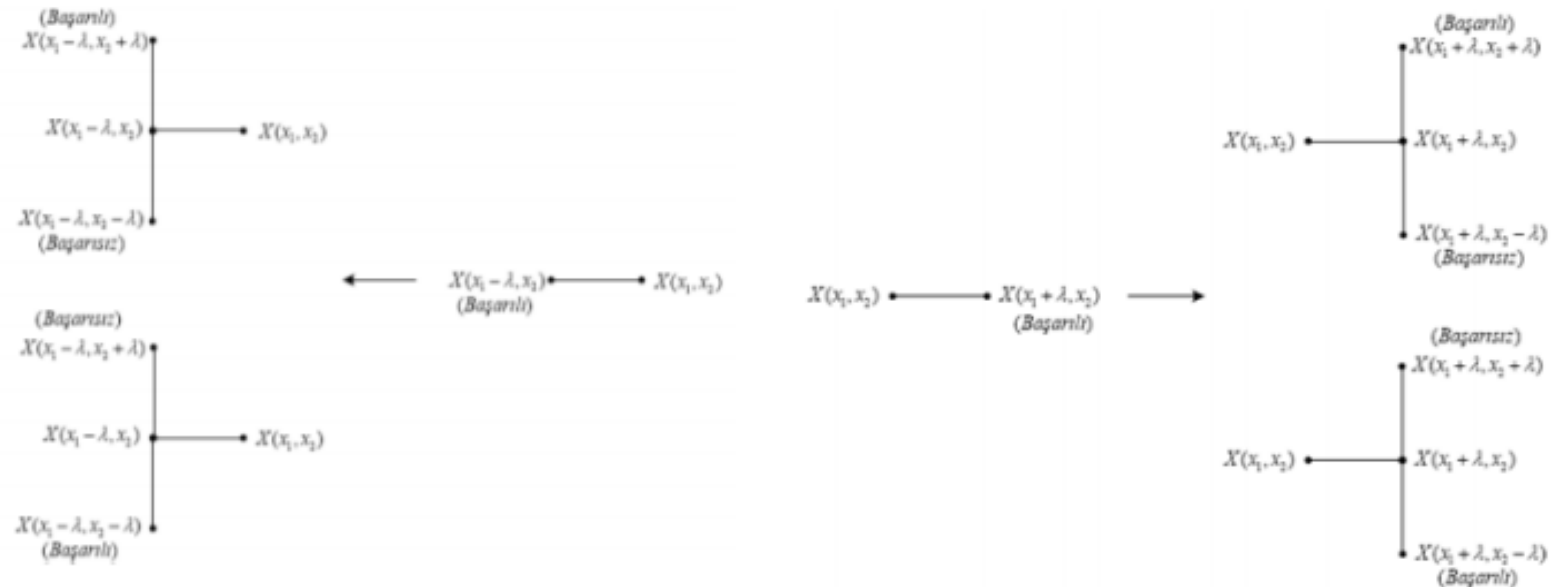
- Aim is to find the minimum value of function  $f(x)$
  - **2 type of search:**
    - Exploratory search
    - Pattern search
- 



- **Procedure**
  - Initial point and step length is determined.
  - Exploratory and pattern move is applied.
  - If the value of function  $f(x)$  decreases after the exploratory move, it is called as successful point else it is called as unsuccessful point.

# Hooke & Jeeves Algorithm

- Exploratory Search:**



# Hooke & Jeeves Algorithm

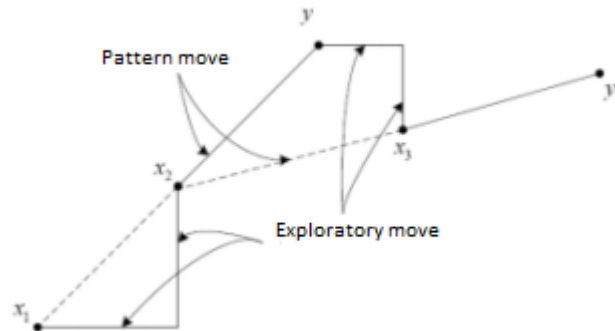
- After the exploratory search;
  - If a successful point is found, pattern search is done.
  - If all possible points are found to be unsuccessful, step length is halved and exploratory search is repeated.
- Algorithm is terminated if the minimum step length is reached.

# Hooke & Jeeves Algorithm

- **Pattern Search**

- Let  $b_2$  be a successful point after the exploratory search.
- The new point, which is obtained by pattern search, is given as follows:

$$p_1 = b_2 + (b_2 - b_1) = 2b_2 - b_1$$





# Hooke & Jeeves Algorithm

- Algorithm continues with the exploratory search near the point  $p_1$  obtained in pattern search.
- If any of the points in the exploratory search is found to be successful, the new reference point is taken as this successful point.
- Otherwise, exploratory search is done near the previous point ( $p_2$ ).

# Example

Hooke & Jeeves metodu kullanarak aşağıda verilen fonksiyonun;

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

$b_1 = [0, 0]$  başlangıç değeri

$h_1 = h_2 = 1$  adım aralığı

$h_1 = h_2 < \frac{1}{4}$  durdurma şartı

için minimum noktasını bulunuz.

# Example

*Gerek koşul*

$$\left. \begin{array}{l} \frac{\partial f}{\partial x_1} = 6x_1 - 2x_2 + 4 = 0 \\ \frac{\partial f}{\partial x_2} = 2x_2 - 2x_1 + 3 = 0 \end{array} \right\} \underbrace{x_1 = -1.75 \quad x_2 = -3.25}_{\text{durağan nokta}}$$

*Yeter koşul*

$$\left. \begin{array}{ll} \frac{\partial f}{\partial x_1 \partial x_1} = 6 & \frac{\partial f}{\partial x_1 \partial x_2} = -2 \\ \frac{\partial f}{\partial x_2 \partial x_1} = -2 & \frac{\partial f}{\partial x_2 \partial x_2} = 2 \end{array} \right\} H = \begin{bmatrix} 6 & -2 \\ -2 & 2 \end{bmatrix} = 8 > 0$$

# Example

$E(b_1)$ :

$$f[1, 0] = 7 \quad (F)$$

$$f[-1, 0] = -1 \quad (S)$$

$$f[-1, 1] = 5 \quad (F)$$

$$f[-1, -1] = -5 \quad (S)$$

Yeni referans noktamız,

$$b_2 = [-1, -1] \rightarrow f(b_2) = -5$$

$$p(b_2): \quad p_1 = 2b_2 - b_1 = 2[-1, -1] - [0, 0] = [-2, -2]$$

# Example

$E(p_1)$ :

$$f[-1, -2] = -7 \quad (S)$$

$$f[-1, -1] = -5 \quad (F)$$

$$f[-1, -3] = -7 \quad (F)$$

Yeni referans noktamız,

$$b_3 = [-1, -2] \rightarrow f(b_3) = -7$$

$$p(b_3): \quad p_2 = 2b_3 - b_2 = 2[-1, -2] - [-1, -1] = [-1, -3]$$

# Example

$E(p_2)$ :

$$f[0, -3] = 0 \quad (F)$$

$$f[-2, -3] = -8 \quad (S)$$

$$f[-2, -2] = -6 \quad (F)$$

$$f[-2, -4] = -8 \quad (F)$$

Yeni referans noktamız,

$$b_4 = [-2, -3] \rightarrow f(b_4) = -8$$

$$p(b_4): \quad p_3 = 2b_4 - b_3 = 2[-2, -3] - [-1, -2] = [-3, -4]$$

# Example

$E(p_3)$ :

$$f[-2, -4] = -8 \quad (F)$$

$$f[-4, -4] = 4 \quad (F)$$

$$f[-3, -5] = -5 \quad (F)$$

$$f[-3, -3] = -3 \quad (F)$$

Referans noktamız değişmedi,

$$b_4 = [-2, -3] \rightarrow f(b_4) = -8$$

# Example

$E(b_4)$ :

$$f[-1, -3] = -7 \quad (F)$$

$$f[-3, -3] = -3 \quad (F)$$

$$f[-2, -2] = -6 \quad (F)$$

$$f[-2, -4] = -8 \quad (F)$$

$E(b_4)$  yani  $b_4$  civardaki aramada da tüm noktalar başarısız olduğundan adım aralığı yarıya düşürülüp tekrar civar araması yapılmalı

$$b_4 = [-2, -3] \qquad h_1 = h_2 = \frac{1}{2}$$



# Example

$E(b_4)$ :

$$f[-1.5, -3] = -8.25 \quad (F)$$

$$f[-1.5, -2.5] = -8 \quad (F)$$

$$f[-1.5, -3.5] = -8 \quad (F)$$

Yeni referans noktamız,

$$b_5 = [-1.5, -3] \rightarrow f(b_5) = -8.25$$

$$p(b_5): \quad p_4 = 2b_5 - b_4 = 2[-1.5, -3] - [-2, -3] = [-1, -3]$$

# Example

$E(b_5)$ :

$$f[-1, -3] = -7 \quad (F)$$

$$f[-2, -3] = -8 \quad (F)$$

$$f[-1.5, -2.5] = -8 \quad (F)$$

$$f[-1.5, -3.5] = -8 \quad (F)$$

$b_5$  civarındaki aramada tüm noktalar başarısız olduğundan adım aralığı yarıya düşürülüp tekrar civar araması yapılmalı

$$b_5 = [-1.5, -3] \quad h_1 = h_2 = \frac{1}{4}$$

# Example

$E(b_5)$ :

$$f[-1.25, -3] = -7.8125 \quad (F)$$

$$f[-1.75, -3] = -8.3125 \quad (S)$$

$$f[-1.75, -2.75] = -8.125 \quad (F)$$

$$f[-1.75, -3.25] = -8.375 \quad (S)$$

Yeni referans noktamız,

$$b_6 = [-1.75, -3.25] \rightarrow f(b_6) = -8.375$$

$$p(b_6): \quad p_5 = 2b_6 - b_5 = 2[-1.75, -3.25] - [-1.5, -3] = [-2, -3.5]$$

# Example

$E(p_5)$ :

$$f[-2.25, -3.5] = -7.8125 \quad (F)$$

$$f[-1.75, -3.5] = -8.3125 \quad (F)$$

$$f[-2, -3.75] = -8.1875 \quad (F)$$

$$f[-2, -3.25] = -8.1875 \quad (F)$$

Referans noktamız değişmedi,

$$b_6 = [-1.75, -3.25] \rightarrow f(b_6) = -8.375$$

# Example

$E(b_6)$ :

$$f[-1.5, -3.25] = -8.1875 \quad (F)$$

$$f[-2, -3.25] = -8.1875 \quad (F)$$

$$f[-1.75, -3] = -8.3125 \quad (F)$$

$$f[-1.75, -3.5] = -8.3125 \quad (F)$$

$E(b_6)$  tümüyle başarısız olduğundan adım aralığı yarıya düşürülür,

$$h_1 = h_2 = \frac{1}{8}$$

Durdurma kriteri sağlandığından algoritmadan çıkılır.

$$b_6 = [-1.75, -3.25] \rightarrow f(b_6) = -8.375$$

# Big Bang-Big Crunch Optimization\*

## **Step 1 (Big Bang):**

- Initial population is generated.

## **Step 2:**

- The fitness function values for all candidates are calculated.

## **Step 3: (Big Crunch)**

- Center of mass is calculated and is chosen as a new point (or the best candidate is chosen as a new point).

## **Step 4: (Big Bang)**

- New candidates are randomly spread over near the point obtained in previous step.

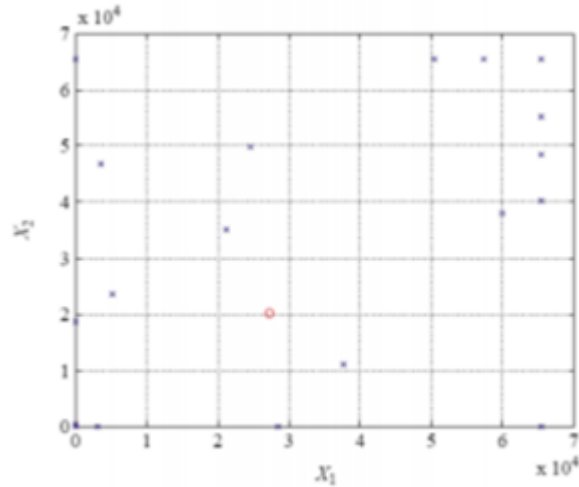
## **Step 5:**

- Return back to step 2 and repeat until stopping criteria is provided.

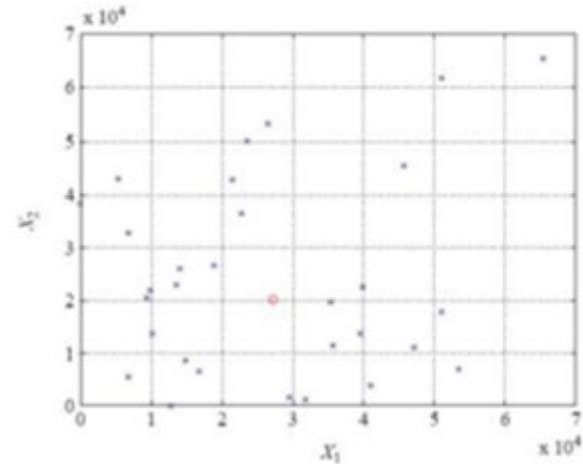
\*O.K.Erol and I.Eksin, "A new optimization method: Big Bang–Big Crunch"

Advances in Engineering Software, 37, 106–111, 2006 <http://www3.itu.edu.tr/~okerol/BBBC.html>

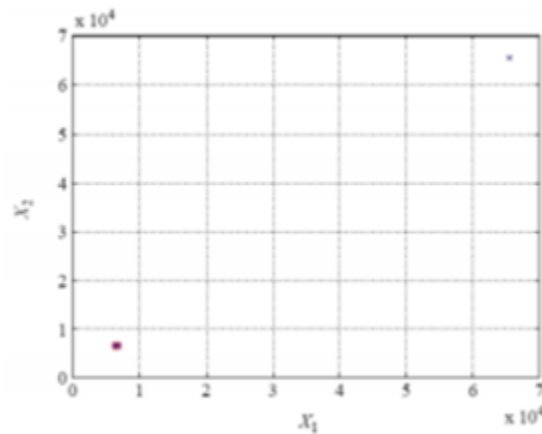
# Big Bang-Big Crunch Optimization



The distribution of candidates over 2D search space



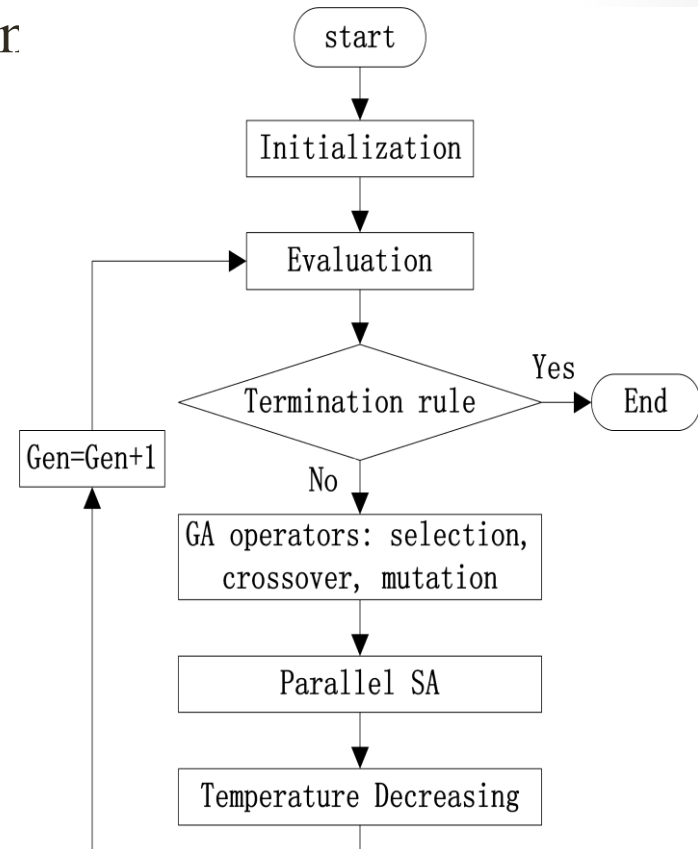
The new distribution after 4 iterations



The new distribution after 500 iterations

# Genetic Algorithm

- Charles Darwin's Theory of Evolution
- Global search algorithm
- Population based
- Natural selection
  - Only the stronger individuals live.
- Genetic operators
  - Variations in population





# Genetic Algorithm

- Optimization Toolbox in MATLAB (gatool)

The screenshot displays the MATLAB Optimization Tool (gatool) interface, which is divided into two main panels: "Problem Setup and Results" and "Options".

**Problem Setup and Results:**

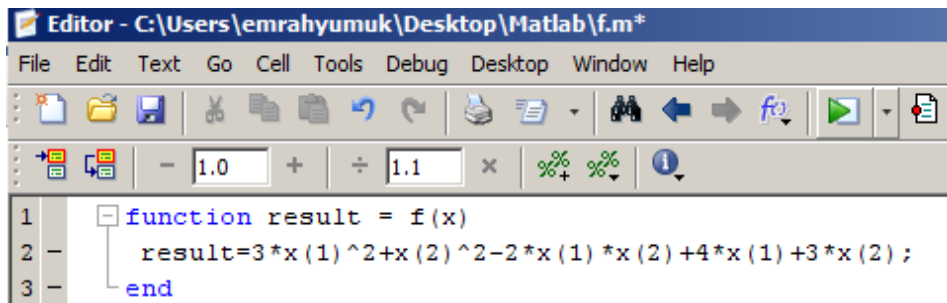
- Solver:** A dropdown menu showing "ga - Genetic Algorithm".
- Problem:**
  - Fitness function:** An empty text input field.
  - Number of variables:** An empty text input field.
- Constraints:**
  - Linear inequalities:** Two input fields labeled "A:" and "b:".
  - Linear equalities:** Two input fields labeled "Aeq:" and "beq:".
  - Bounds:** Two input fields labeled "Lower:" and "Upper:".
  - Nonlinear constraint function:** An empty text input field.
- Run solver and view results:**
  - A checkbox labeled "Use random states from previous run" which is currently unchecked.

**Options:**

- Population:**
  - Population type:** A dropdown menu showing "Double Vector".
  - Population size:** Two radio buttons: "Use default: 20" (selected) and "Specify:" followed by an empty text input field.
  - Creation function:** A dropdown menu showing "Use constraint dependent default".
- Initial population:** Two radio buttons: "Use default: []" (selected) and "Specify:" followed by an empty text input field.
- Initial scores:** Two radio buttons: "Use default: []" (selected) and "Specify:" followed by an empty text input field.
- Initial range:** Two radio buttons: "Use default: [0;1]" (selected) and "Specify:" followed by an empty text input field.

# Genetic Algorithm

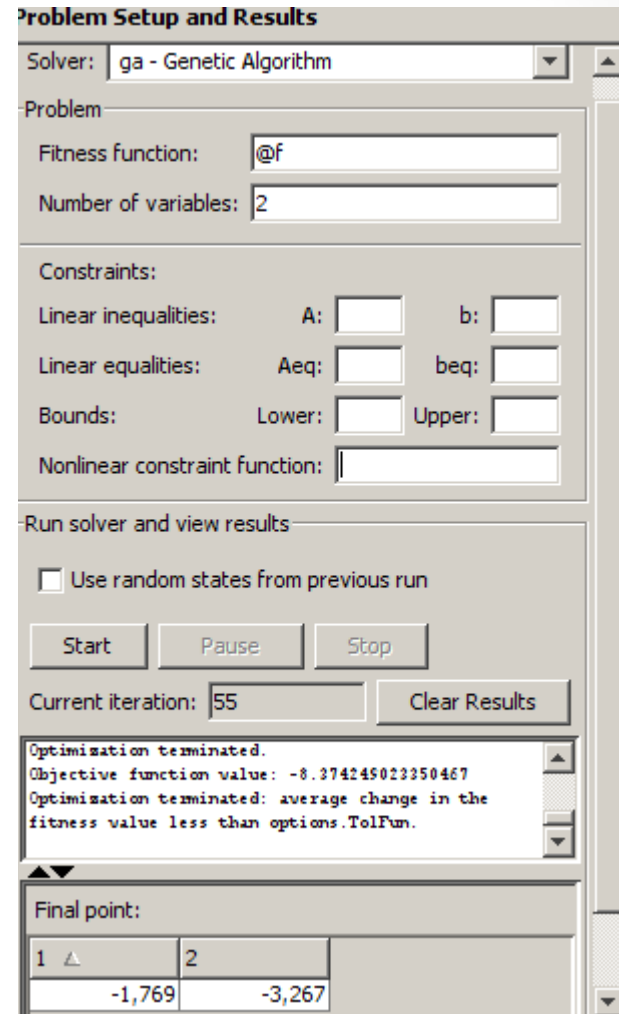
- Function name and the file name should be the same.



Editor - C:\Users\emrahymuk\Desktop\Matlab\f.m\*

File Edit Text Go Cell Tools Debug Desktop Window Help

1 function result = f(x)  
2 result=3\*x(1)^2+x(2)^2-2\*x(1)\*x(2)+4\*x(1)+3\*x(2);  
3 end



**Problem Setup and Results**

Solver: ga - Genetic Algorithm

Problem

Fitness function: @f

Number of variables: 2

Constraints:

Linear inequalities: A: b:

Linear equalities: Aeq: beq:

Bounds: Lower: Upper:

Nonlinear constraint function:

Run solver and view results

☐ Use random states from previous run

Start Pause Stop

Current iteration: 55 Clear Results

Optimization terminated.  
Objective function value: -8.374245023350467  
Optimization terminated: average change in the fitness value less than options.TolFun.

Final point:

1	2
-1,769	-3,267

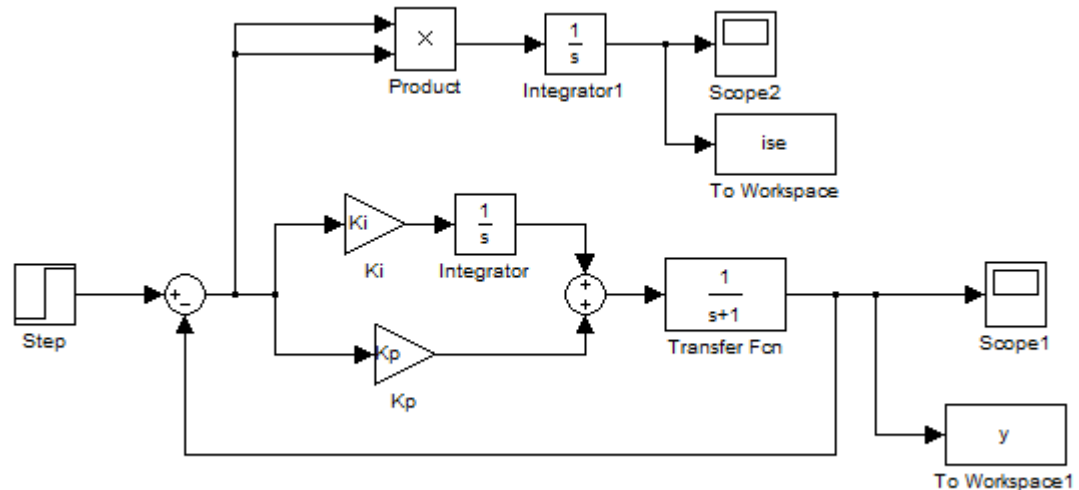
# Genetic Algorithm

**Fitness Function:**

ISE

**Variables:**

$K_p$ ,  $K_i$



Command window:

`>> global  $K_i$   $K_p$`

```
Editor - C:\Users\emrahymuk\Desktop\Matlab\cost.m*
File Edit Text Go Cell Tools Debug Desktop Window Help
[Icons]
- 1.0 + ÷ 1.1 x % + % - i
1 function costValue = cost(x)
2     global Kp;
3     global Ki;
4     Kp=x(1);
5     Ki=x(2);
6     sim('model');
7     costValue=ise(end);
8 end
```

# About the Final Project

1. Hooke & Jeeves algorithm will be coded and will be tested on the example given in presentation.

2. - **Local search method**

Hooke & Jeeves Algorithm

(Choosing of the initial points is critical!)

- **Global optimization method**

BBBC

Genetic Algorithm

Each team will design a PI or PD controller using a local search method and one of the global algorithm given above.

# About the Final Project

$$J_1 = \int e^2(t) dt$$

$$J_2 = \int te^2(t) dt$$

$$J_3 = \text{Overshoot, peak time, settling time etc.}$$

(will be determined by yourself)

- For 3 different fitness function, 3 different controllers will be designed.
- 3. Controller will be redesigned with the saturation element placed in system input.
- 4. For the  $J_1$  fitness function, parameter optimization method will be used and will also be compared with numerical optimization results.