خودآموز اُکتاو

سید ناصر رضوی n.razavi@tabrizu.ac.ir

۱۳۹۵

اکتاو

- معرفی اکتاو.
  - یک زبان برنامه‌نویسی سطح بالا و رایگان
  - مناسب برای محاسبات عددی
  - مناسب برای پیاده‌سازی سریع

عملیات ریاضی


```
>> 5 + 6
ans = 11

>> 3 - 2
ans = 1

>> 5 * 8
ans = 40

>> 1 / 2
ans = 0.50000

>> 2 ^ 6
ans = 64
```

عملیات منطقی و مقایسه‌ای


```
>> 1 == 2       % equality
ans = 0

>> 1 ~= 2       % inequality
ans = 1

>> 3 >= 1       % greeter than or equal
ans = 1

>> 1 && 0       % logical AND
ans = 0

>> 1 || 0       % logical OR
ans = 1
```

متغیرها


```
>> a = 3
 a = 3

>> a = 3;    % semicolon suppressing output

>> b = 'hi';

>> c = (3 >= 1)
 c = 1
```

متغیرها


```
>> a = pi
a = 3.1416

>> disp(a)
3.1416

>> disp(sprintf('2 decimals: %.2f', a))
2 decimals: 3.14

>> disp(sprintf('6 decimals: %.6f', a))
6 decimals: 3.141593

>> format long
>> a
3.14159265358979
```

بردارها و ماتریس‌ها


```
>> A = [1 2; 3 4; 5 6]          % 3 by 2 matrix
A =
   1   2
   3   4
   5   6

>> v = [1 2 3]                  % row vector
v =
   1   2   3

>> v = [1; 2; 3]                % column vector
v =
   1
   2
   3
```

بردارها و ماتریس‌ها

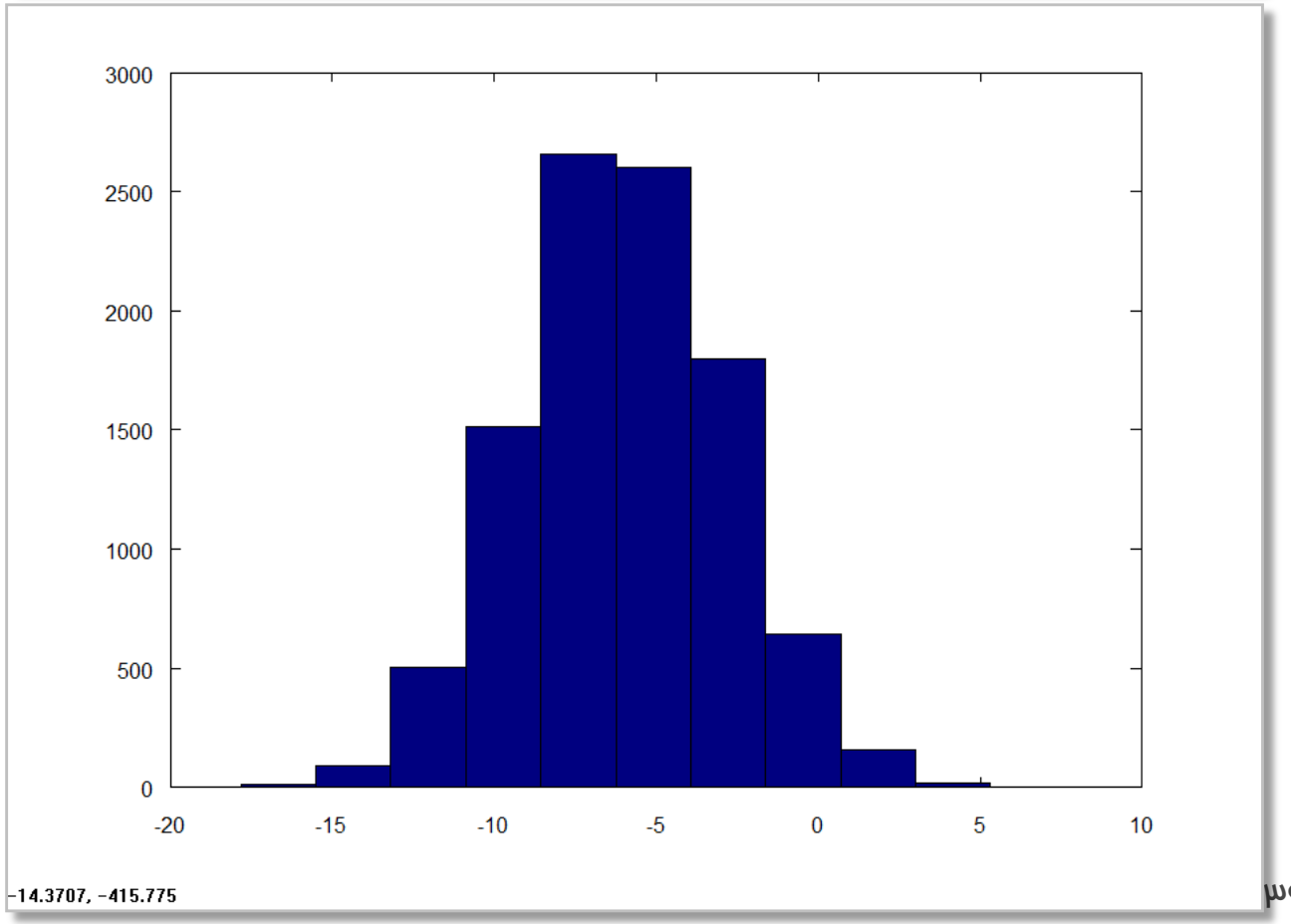

```
>> v = 1 : 0.2 : 2
v = 1.0   1.2   1.4   1.6   1.8   2.0

>> v = 1 : 6
v = 1   2   3   4   5   6

>> ones(2, 3)
ans =
   1   1   1
   1   1   1

>> c = 2 * ones(2, 3)
c =
   2   2   2
   2   2   2
```

بردارها و ماتریس‌ها


```
>> w = ones(1, 3)
w =
   1   1   1

>> w = zeros(1, 3)
w =
   0   0   0

>> w = rand(1, 3)
w =
   0.91477   0.14359   0.84860

>> w = randn(1, 3)
w =
   -0.33517   1.26847   -0.28211
```

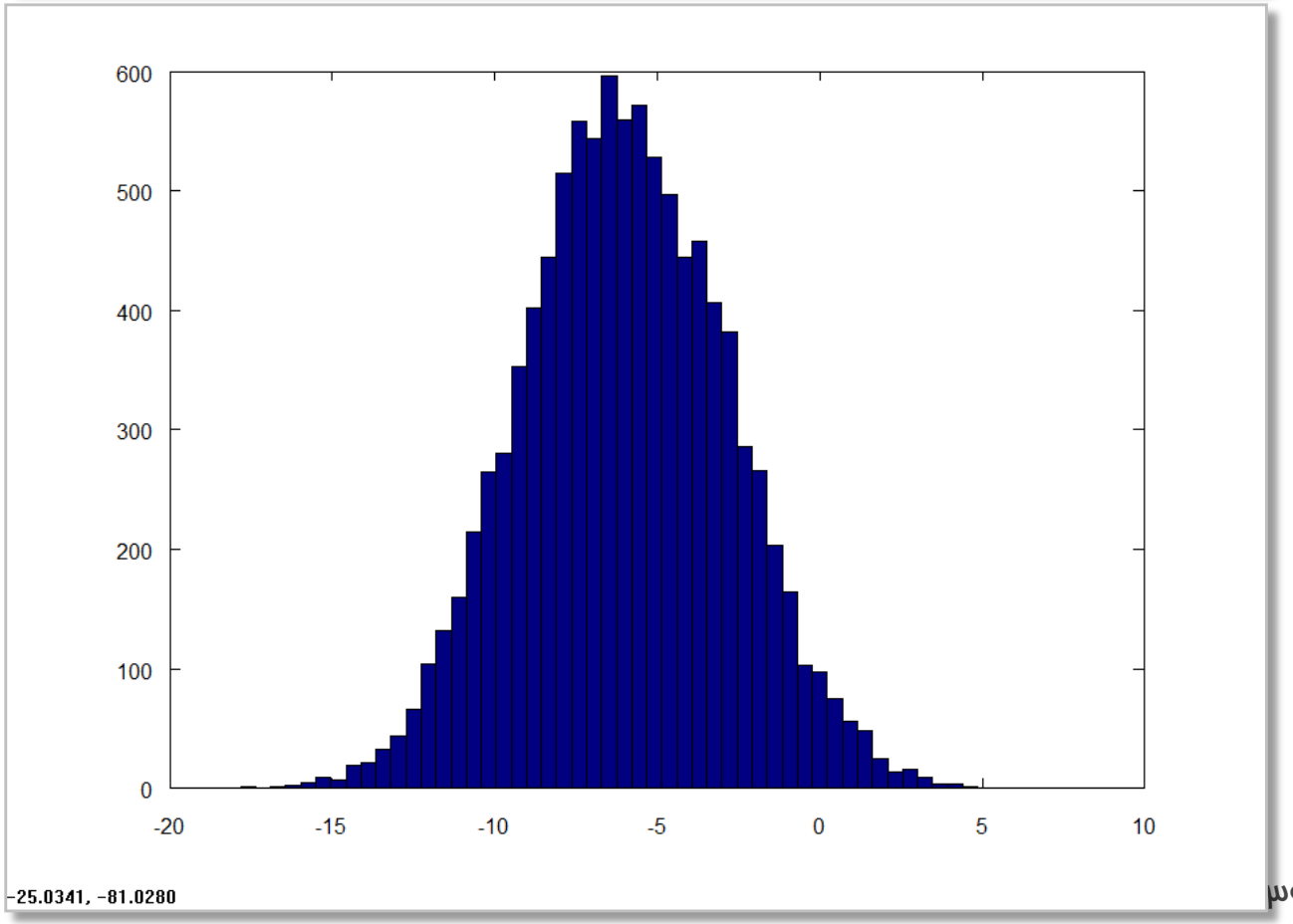
بردارها و ماتریس‌ها


```
>> w = -6 + sqrt(10) * (randn(1, 10000));
>> hist(w)
```



بردارها و ماتریس‌ها


```
>> w = -6 + sqrt(10) * (randn(1, 10000));
>> hist(w, 50)
```



بردارها و ماتریس ها


```
>> I = eye(3)
I =
Diagonal Matrix
   1   0   0
   0   1   0
   0   0   1

>> I = eye(4)
I =
Diagonal Matrix
   1   0   0   0
   0   1   0   0
   0   0   1   0
   0   0   0   1
```

۱۳ کار با داده‌ها

ابعاد ماتریس


```
>> A = [1 2; 3 4; 5 6]          % 3 by 2 matrix
A =
   1   2
   3   4
   5   6
>> size(A)
ans = 3   2
>> size(A, 1)                   % No. of rows
ans =  3
>> rows(A)
ans = 3
>> size(A, 2)                   % No. of columns
ans =  2
>> columns(A)
ans = 2
```

طول بردار


```
>> v = [1 2 3 4 5]
>> length(v)
ans =  5

>> A = [1 2; 3 4; 5 6];
A =
   1   2
   3   4
   5   6

>> length(A)            % longer dimension
ans =  3

>> length([1; 2; 3; 4; 5])
ans =  5
```

خواندن داده‌ها از فایل


```
>> pwd
ans =  d:\Octave\3.2.4_gcc-4.4.0\bin

>> cd 'C:\Users\IRANDATA\Desktop'

>> pwd
ans =  C:\Users\IRANDATA\Desktop

>> ls           % list directories and files
featuresX.dat
priceY.dat
...

>> load featuresX.dat
>> load priceY.dat
```

خواندن داده‌ها از فایل


```
>> who
variables in the current scope:
A       I       ans     c       priceY
C       a       b       featuresX
>> featuresX
2140    3
1600    3
2400    3
1416    2
...
>> priceY
3999
3299
3690
...
```

بررسی محتویات حافظه


```
>> whos
variables in the current scope:
  Attr   Name           Size              Bytes  Class
  ====   ====           ====              =====  =====
         A               3x2                 48  double
         C               2x3                 48  double
         I               3x3                 72  double
         a               1x1                  8  double
         ans             1x2                 16  double
         b               1x2                  2  char
         c               1x1                  1  logical
         featuresX      47x2                752  double
         priceY         47x1                376  double
         v               1x4                 32  double
         w               1x10000          80000  double
```

بررسی محتویات حافظه


```
>> clear featuresX
>> whos
variables in the current scope:
  Attr  Name            Size              Bytes  Class
  ====  ====            ====              =====  =====
        A                3x2                 48  double
        C                2x3                 48  double
        I                6x1                 48  double
        a                1x1                  8  double
        ans              1x2                 16  double
        b                1x2                  2  char
        c                1x1                  1  logical
        priceY          47x1                376  double
        v                1x4                 32  double
        w                1x10000          80000  double
```

ذخیره داده‌ها در فایل


```
>> v = priceY(1:10)
3999
3299
3690
2320
5399
2999
3149
1989
2120
2425

>> save hello.mat v % save as binary
```

خواندن داده‌ها از فایل


```
>> clear                % clear all variables
>> who
>> load hello.mat
>> who
v
>> v
3999
3299
3690
2320
5399
2999
3149
...
```

ذخیره‌ی داده‌ها در یک فایل متنی


```
>> save hello.txt v -ascii % save as text
```

عملگر کالن


```
>> A = [1 2; 3 4; 5 6];

>> A(3, 2)
ans =  6

>> A(2, :)    % every element in 2nd row
ans =
   3   4

>> A(:, 2)    % every element in 2nd column
ans =
   2
   4
   6
```

عملگر کالن


```
>> A
A =
   1   2
   3   4
   5   6
>> A([1 3], :)
ans =
   1   2
   5   6
>> A(:, 2) = [10; 11; 12]
A =
   1   10
   3   11
   5   12
```

عملگر کالن


```
>> A = [A, [100; 101; 102]]  % append a column
A =
   1   10   100
   3   11   101
   5   12   102
>> v = A(:)   % put all elements in a vector
v =
     1
     3
     5
    10
    11
    12
   100
   101
```

اتصال ماتریس‌ها به یکدیگر


```
>> A = [1 2; 3 4; 5 6];
>> B = [11 12; 13 14; 15 16];
>> C = [A B]                          % C = [A, B]
C =
    1    2   11   12
    3    4   13   14
    5    6   15   16
>> C = [A; B]
C =
    1    2
    3    4
    5    6
   11   12
   13   14
   15   16
```

انجام محاسبات بر روی داده‌ها

عملیات ماتریسی


```
>> A = [1   2;  3  4;  5  6];
>> B = [11 12; 13 14; 15 16];
>> C = [ 1  1;  2  2];

>> A * C                  % Matrix multiplication
ans =
    5     5
   11    11
   17    17

>> A .* B                 % element-wise multiplication
ans =
   11    24
   39    56
   75    96
```

عملیات ماتریسی


```
>> A = [1 2; 3 4; 5 6];

>> A .^ 2       % element-wise squaring
ans =
    1    4
    9   16
   25   36

>> v = [1; 2; 3]

>> 1 ./ v       % element-wise reciprocal
ans =
   1.00000
   0.50000
   0.33333
```

عملیات ماتریسی


```
>> v = [1; 2; 3];

>> log(v)        % element-wise logarithms
ans =
    0.00000
    0.69315
    1.09861

>> exp(v)         % element-wise exponentiation
ans =
    2.7183
    7.3891
   20.0855

>> abs(v);       % element-wise absolute value
```

عملیات ماتریسی


```
>> v = [1; 2; 3];

>> v + ones(length(v), 1)                % element-wise increment
ans =
   2
   3
   4

>> v + 1                                  % element-wise increment
ans =
   2
   3
   4
```

عملیات ماتریسی


```
>> A = [1 2; 3 4; 5 6]
ans =
   1   2
   3   4
   5   6

>> A'                    % transpose
ans =
   1   3   5
   2   4   6
```

عملیات ماتریسی


```
>> a = [1 15 2 0.5]
a =
   1.00000   15.00000   2.00000   0.50000

>> val = max(a)
val = 15

>> [val, ind] = max(a)
val =   15
ind =   2

>> a < 3                          % element-wise comparison
ans =   1   0   1   1

>> find(a < 3)                    % find elements less than 3
ans =   1   3   4
```

عملیات ماتریسی


```
>> A = magic(3)                    % 3 by 3 magic square
A =
   8   1   6
   3   5   7
   4   9   2

>> [r, c] = find(A >= 7)
r =
   1
   3
   2
c =
   1
   2
   3
```

عملیات ماتریسی


```
>> a
a =
   1.00000   15.00000   2.00000   0.50000
>> sum(a)
ans =   18.500
>> prod(a)
ans =   15
>> floor(a)
ans =
   1   15   2   0
>> ceil(a)
ans =
   1   15   2   1
```

عملیات ماتریسی


```
>> A = magic(3)                    % 3 by 3 magic square
A =
   8   1   6
   3   5   7
   4   9   2

>> max(A, [], 1)                   % column-wise maximum
ans =
   8   9   7

>> max(A, [], 2)                   % row-wise maximum
ans =
   8
   7
   9
```

عملیات ماتریسی


```
>> max(A)                % column-wise maximum
ans =
   8    9    7

>> max(max(A))           % maximum element in the matrix
ans =    9

>> max(A(:))
ans =    9

>> sum(A, 1)             % sum of columns
ans =
   15   15   15

>> sum(A, 2)             % sum of rows
```

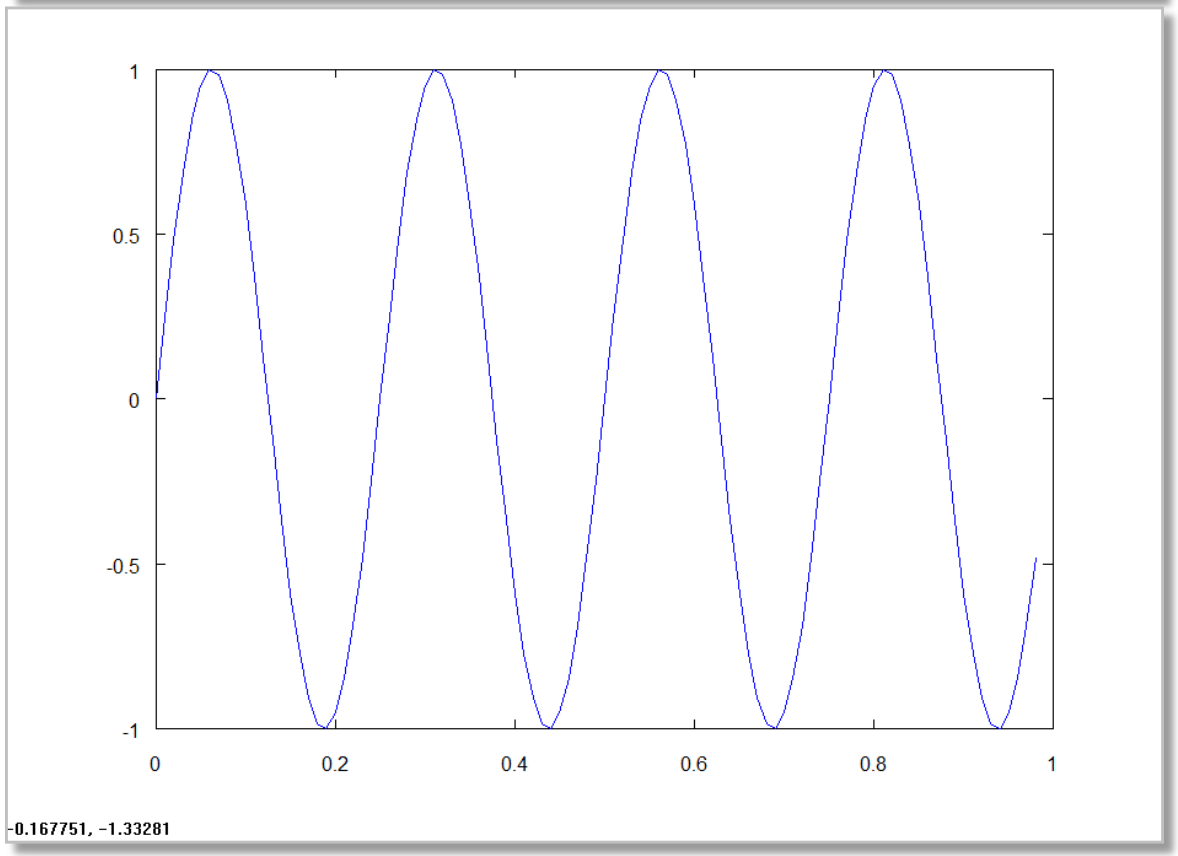
عملیات ماتریسی


```
>> sum(sum(A .* eye(3)))                % sum of main diagonal
ans =    15

>> sum(sum(A .* flipud(eye(3))))        % sum of other diagonal
ans =    15

>> pinv(A)                              % pseudo-inverse
ans =
    0.147222   -0.144444    0.063889
   -0.061111    0.022222    0.105556
   -0.019444    0.188889   -0.102778
```

ترسیم نمودارهای ساده

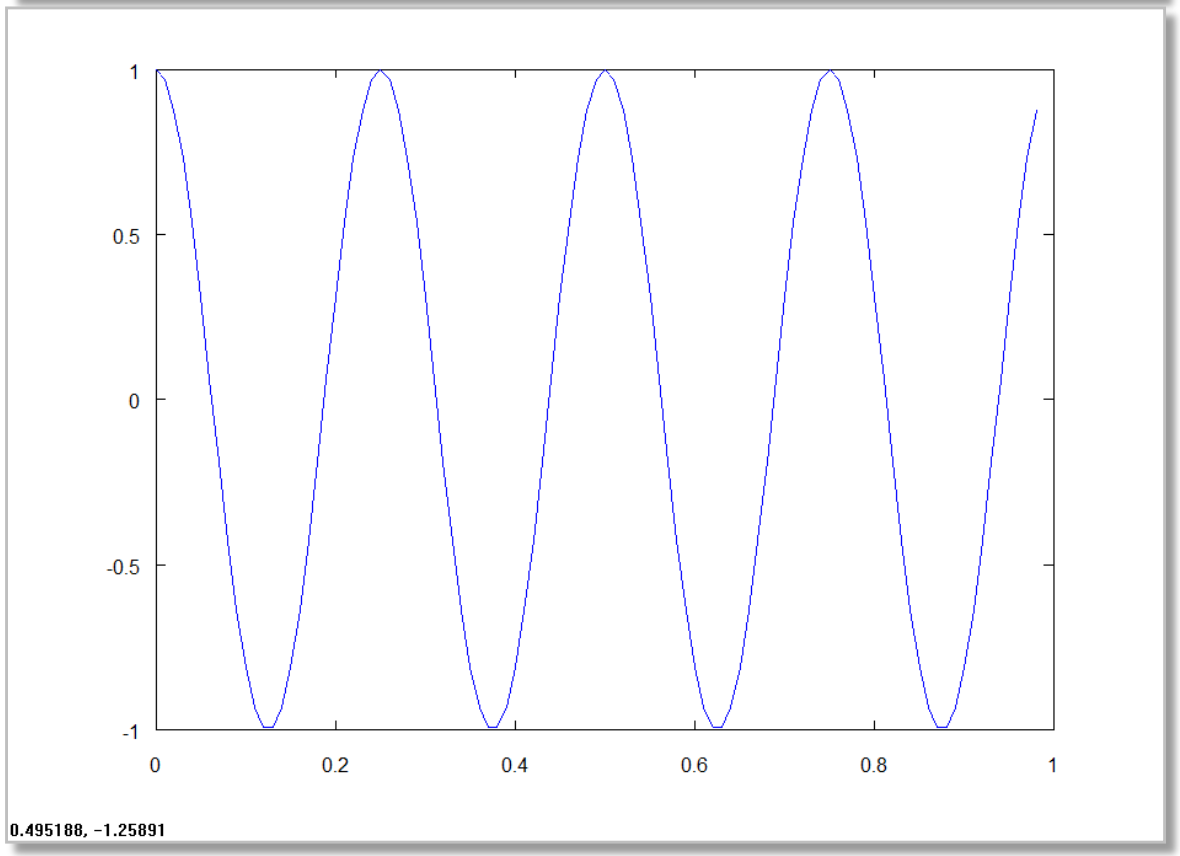
رسم نمودار


```
>> t = 0 : 0.01 : 0.98;
>> y1 = sin(2 * pi * 4 * t);
>> plot(t, y1);
```



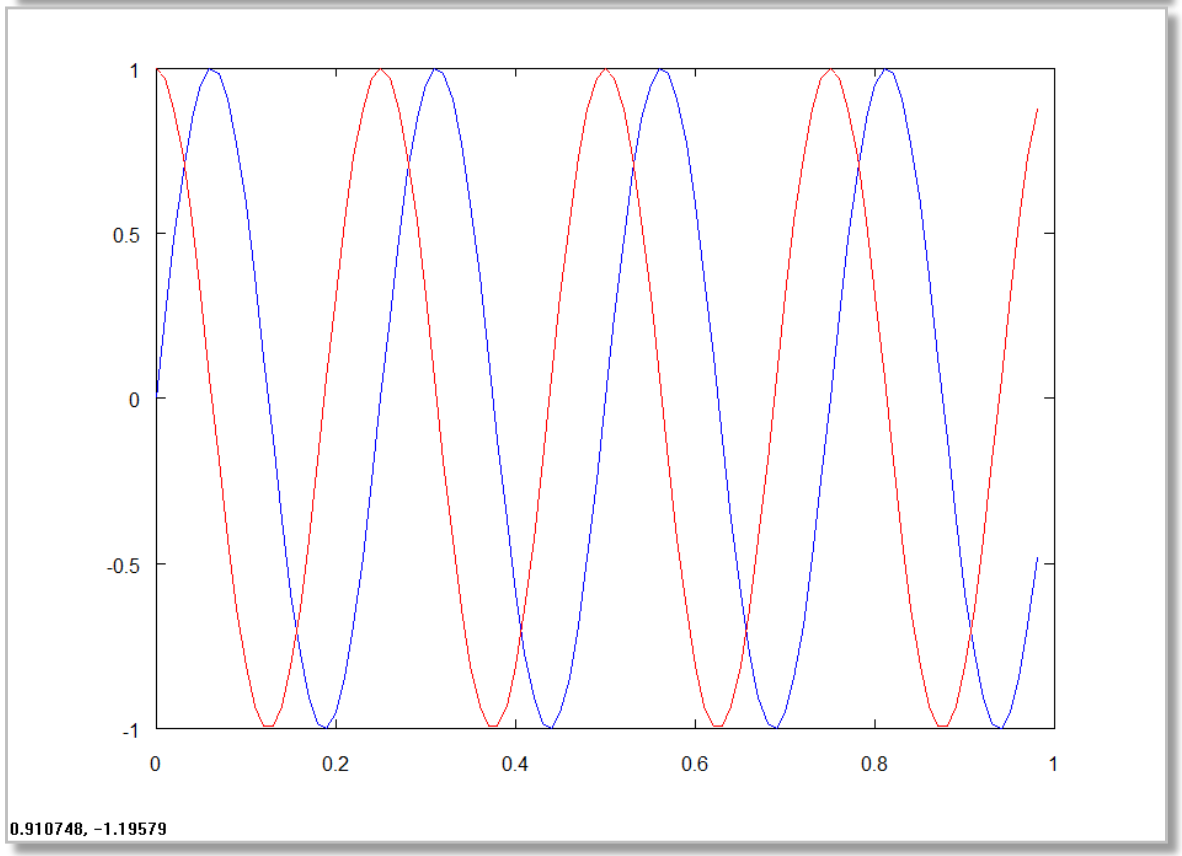
رسم نمودار


```
>> t = 0 : 0.01 : 0.98;
>> y2 = cos(2 * pi * 4 * t);
>> plot(t, y2);
```



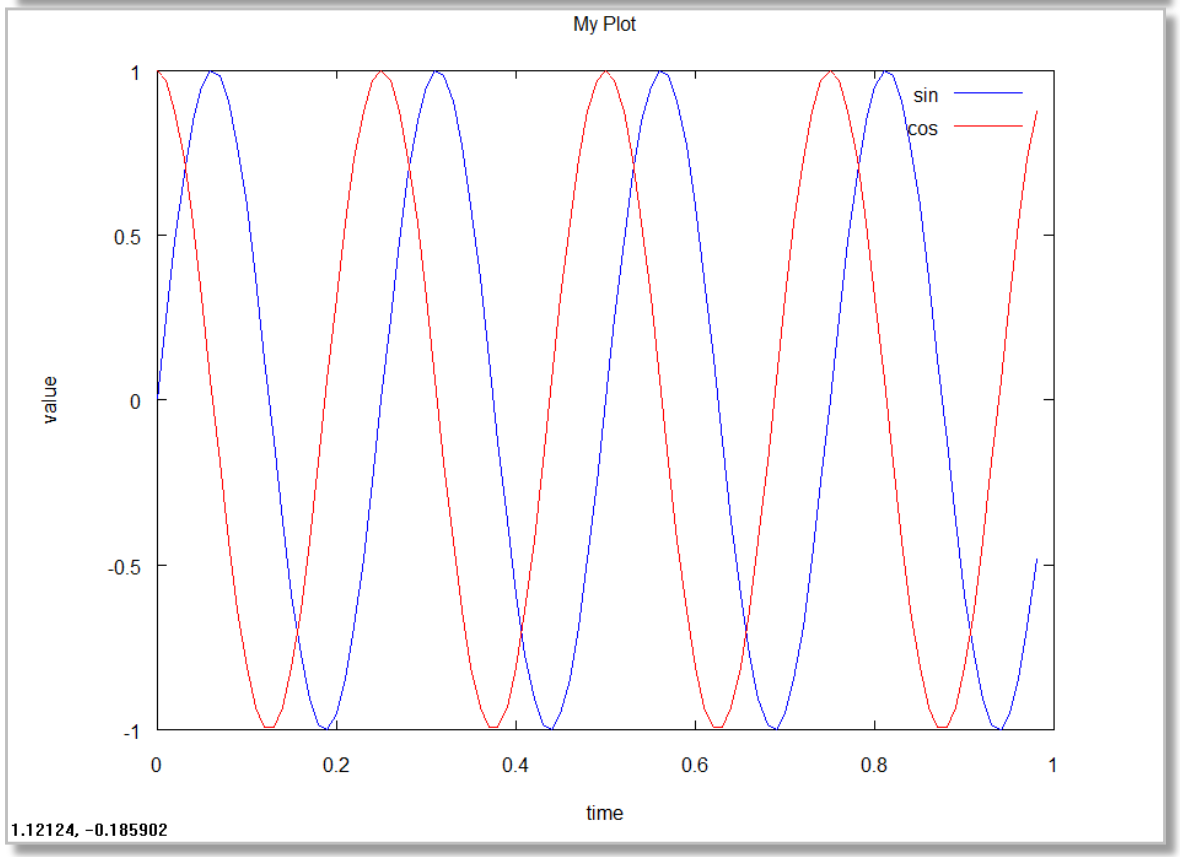
رسم نمودار


```
>> plot(t, y1);
>> hold on;
>> plot(t, y2, 'r');
```



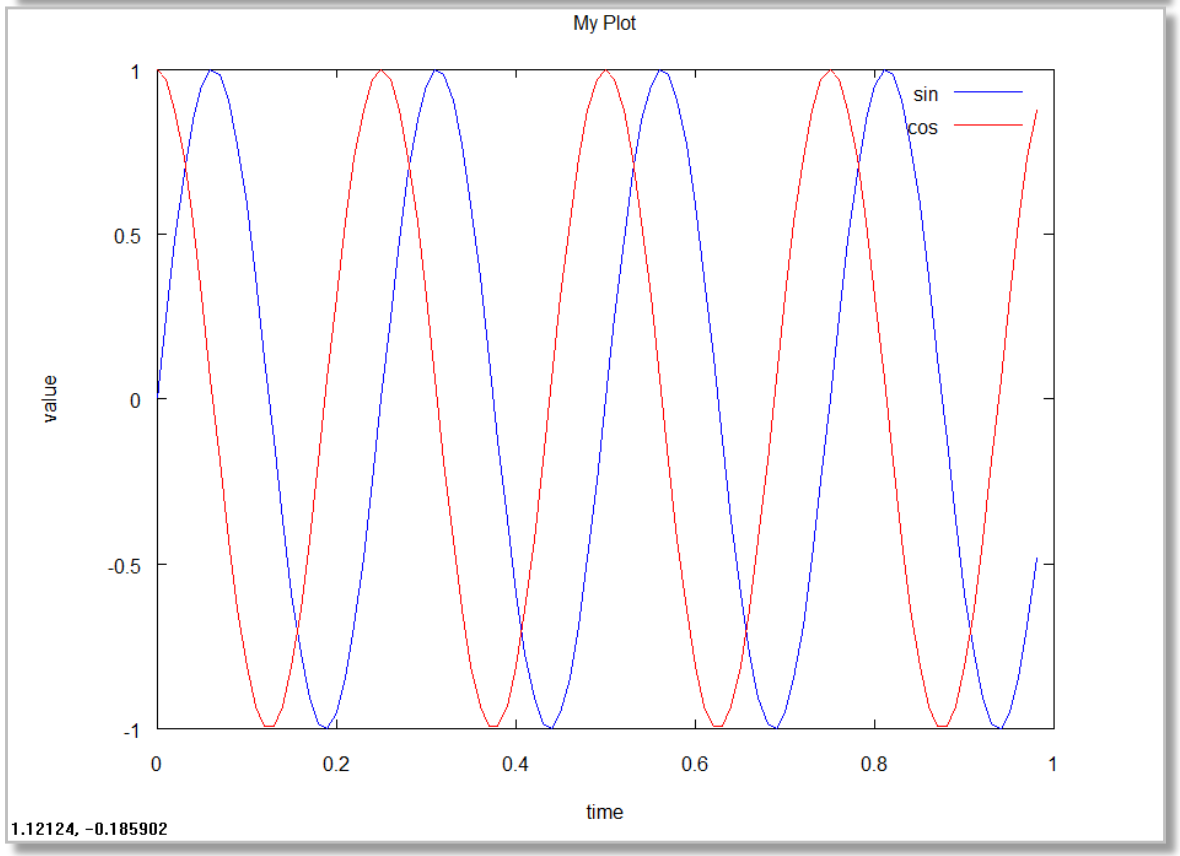
رسم نمودار


```
>> xlabel('time'), ylabel('value')
>> legend('sin', 'cos')
>> title('My Plot');
```



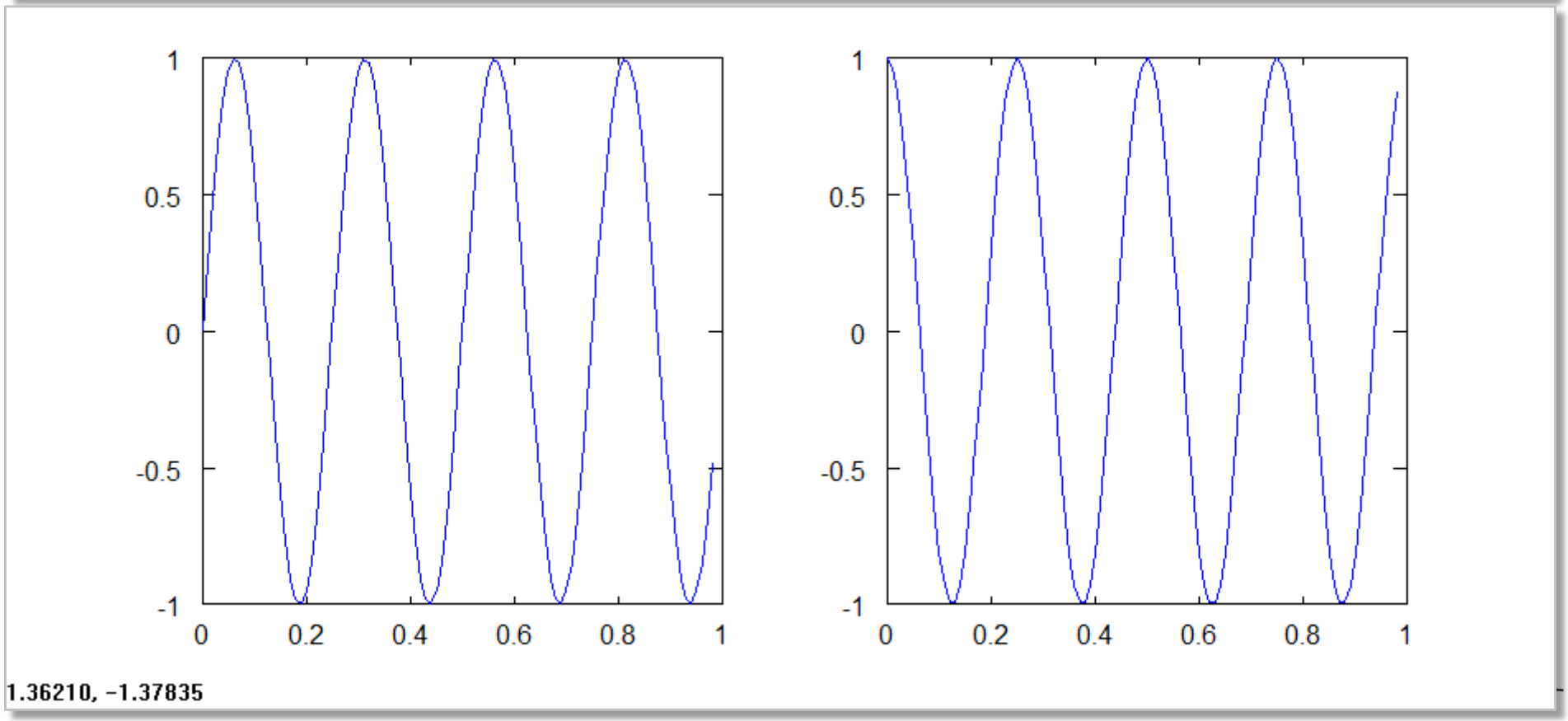
رسم نمودار


```
>> cd 'C:\Users\IRANDATA\Desktop';
>> print -dpng 'MyPlot.png'
```



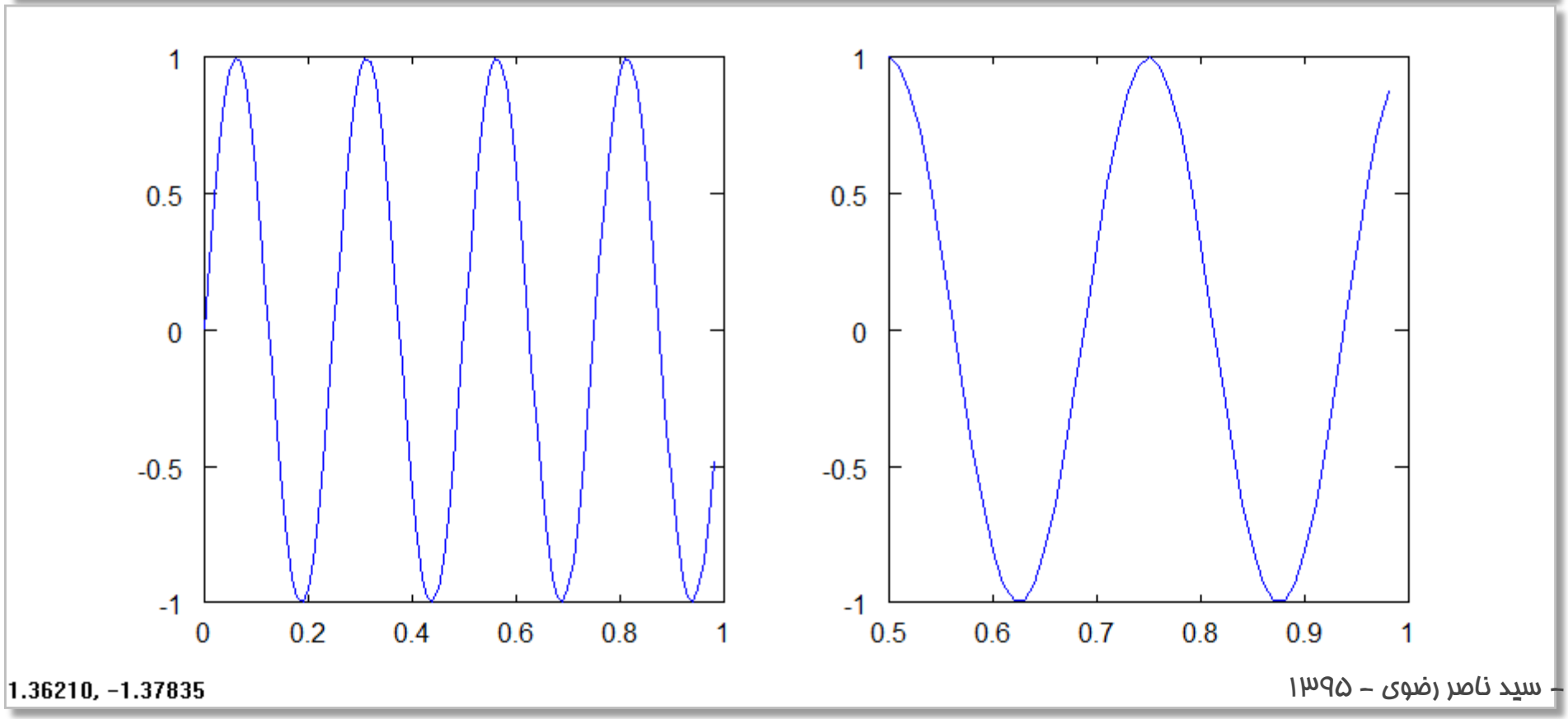
رسم نمودار


```
>> subplot(1, 2, 1);          % divides plot a 1x2 grid, access first
>> plot(t, y1);
>> subplot(1, 2, 2); plot(t, y2);
```



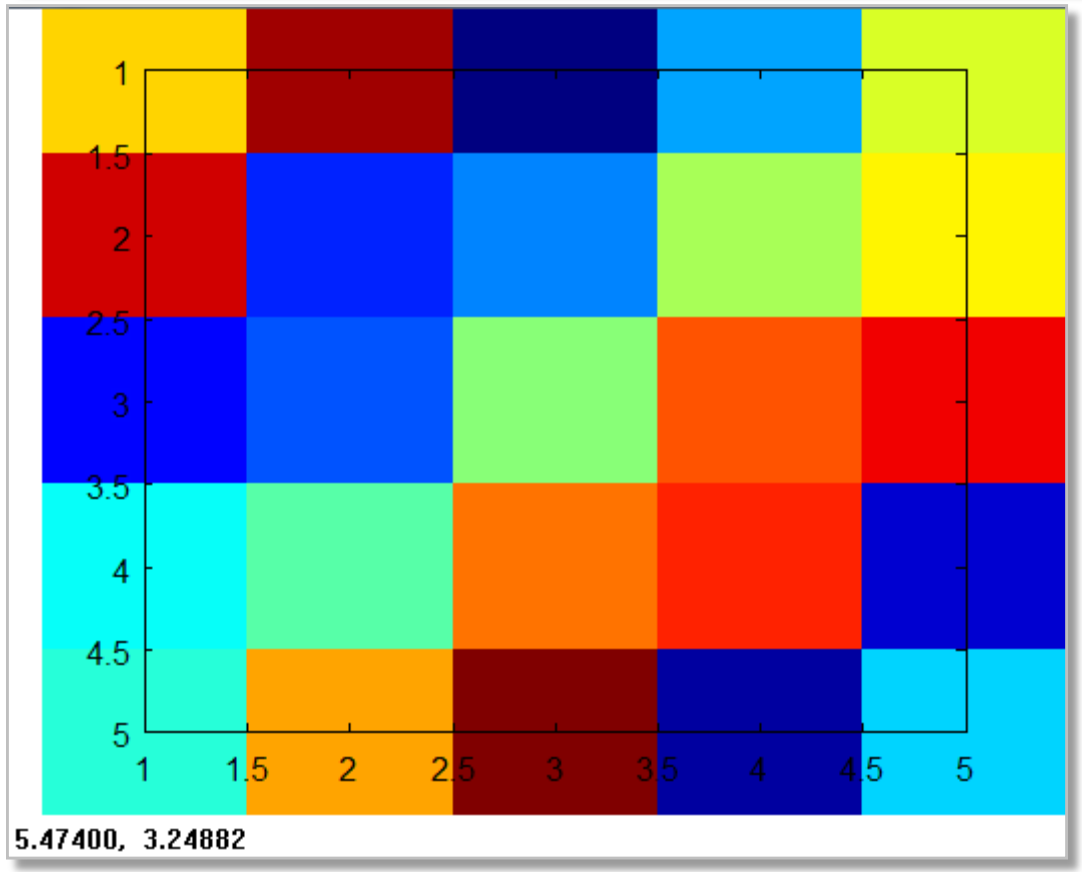
رسم نمودار


```
>> axis([0.5 1 -1 1]); % change axis scale
```



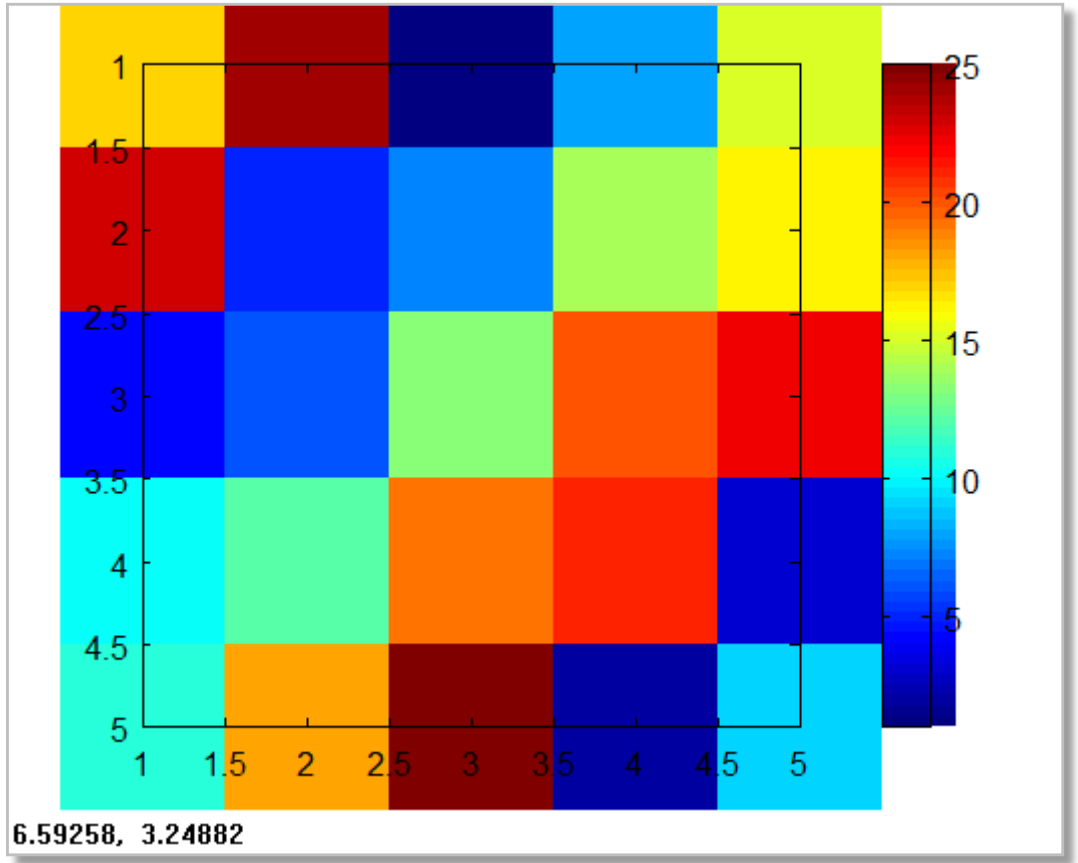
رسم نمودار


```
>> clf;                 % clear current figure
>> A = magic(5);        % 5x5 magic square
>> imagesc(A);          % a way to visualize a matrix
```



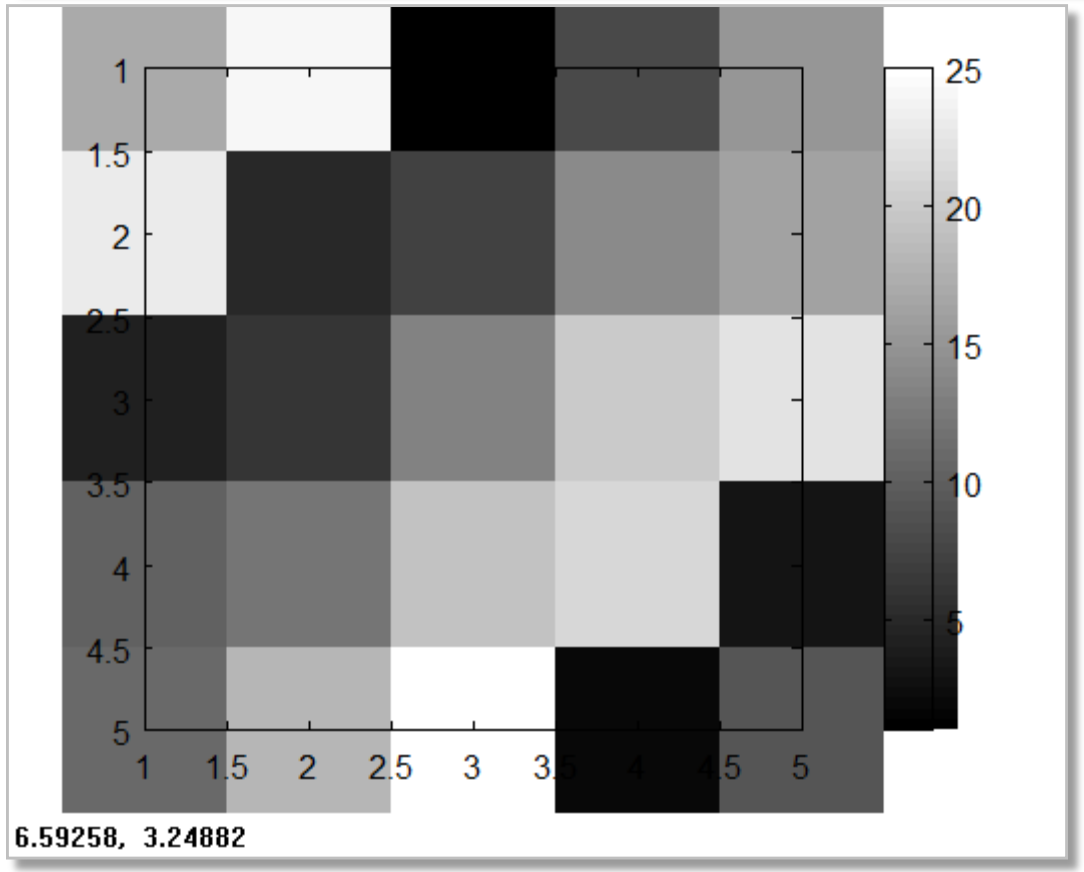
رسم نمودار


```
>> clf;                          % clear current figure
>> A = magic(5);                 % 5x5 magic square
>> imagesc(A), colorbar;
```



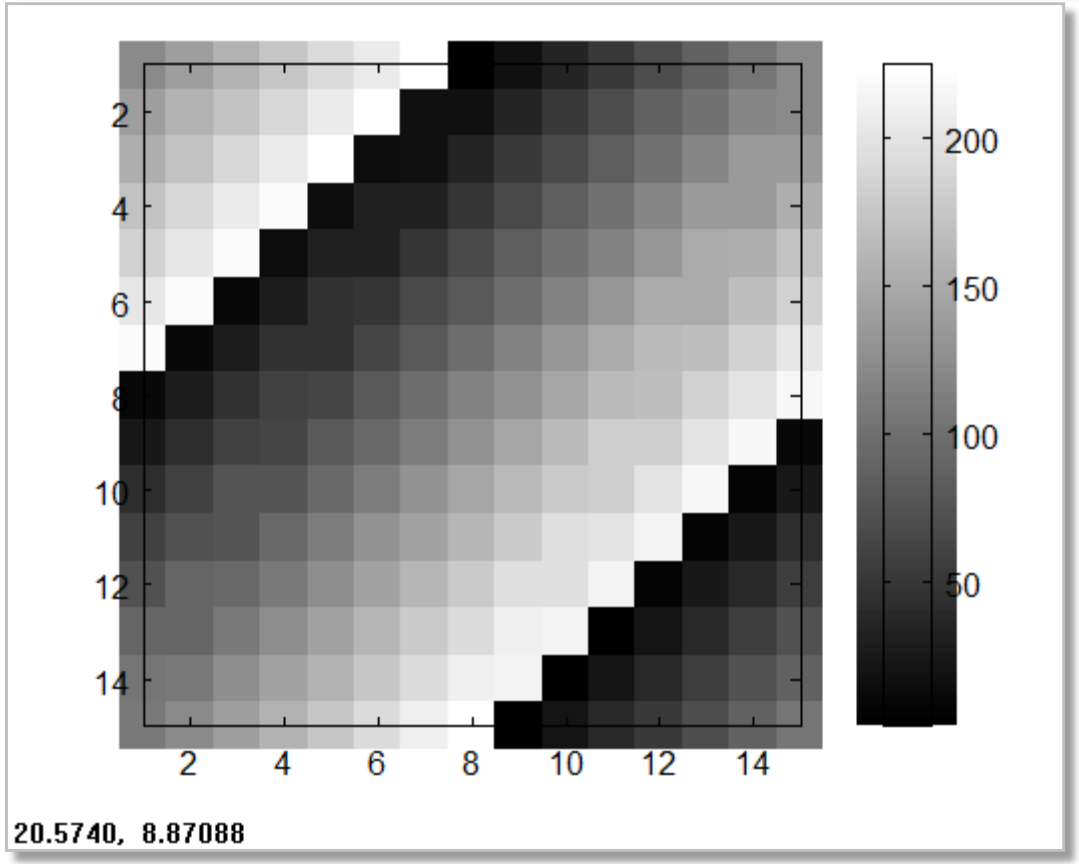
رسم نمودار


```
>> clf;                                          % clear current figure
>> A = magic(5);                                 % 5x5 magic square
>> imagesc(A), colorbar, colormap gray;
```



رسم نمودار


```
>> clf;                                  % clear current figure
>> A = magic(15);                        % 15x15 magic square
>> imagesc(A), colorbar, colormap gray;
```



دستورات کنترلی و توابع

دستورات شرطی

حلقه ها

توابع

دستورات شرطی


```
>> r = rand
r = 0.72666

>> if r <= 0.5,
>     disp('Head');
>  else
>     disp('Tail');
>  end;
Tail
```

حلقه for


```
>> h = 0; t = 0;

>> for i = 1 : 1000,
>       if rand < 0.5,
>          h = h + 1;
>       else
>          t = t + 1;
>      end;
>  end;

>> disp(sprintf('%3d heads, %3d tails', h, t));
491 heads, 509 tails
```

حلقه for


```
>> result = zeros(1, 6);

>> for i = 1 : 1000,
>       r = 1 + floor(6 * rand);
>       result(r) = result(r) + 1;
>   end;

>> result
result =
162   176   167   177   158   160
```

حلقه for


```
>> v = zeros(1, 10);

>> for i = 1 : 10,
>       v(i) = 2 ^ i;
>   end;

>> v
v =   2   4   8   16   32   64   128   256   512   1024

>> % display 1 to 10
>> indices = 1 : 10
>> for i = indices,
>       disp(i);
>   end;
```

حلقه while


```
>> v
v =
   2   4   8   16   32   64   128   256   512   1024

>> i = 1;
>> while (i <= 5),
>       v(i) = 100;
>       i = i + 1;
>  end;

>> v
v =
   100   100   100   100   100   64   128   256   512   1024
```

حلقه while


```
>> i = 1;
>> while true,
>       v(i) = 999;
>       i = i + 1;
>       if i == 6,
>           break;
>       end;
>  end;

>> v
v =
   999   999   999   999   999    64   128   256   512  1024
```

توابع

squareThisNumber.m

```
function y = squareThisNumber(x)
   y = x ^ 2;
```

```
>> squareThisNumber(5)
ans =   25

>> % tell to octave where it should search for your files
>> addpath('C:\Users\IRANDATA\Desktop');
>> cd 'C:\'

>> squareThisNumber(5);
ans =   25
```

توابع با چند خروجی

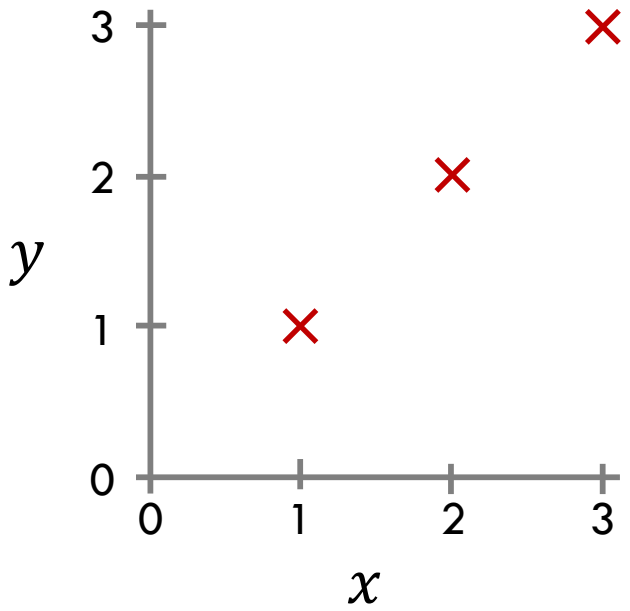
squareAndCubeThisNumber.m

```
function [y1, y2] = squareAndCubeThisNumber(x)
    y1 = x ^ 2;
    y2 = x ^ 3;
```

```
>> squareAndCubeThisNumber(5)
ans =   25

>> [s, c] = squareAndCubeThisNumber(5)
s =   25
c =  125

>>
```

توابع

- هدف. تعریف یک تابع به منظور محاسبه تابع هزینه $J(\theta)$



توابع

costFunctionJ.m

```
function J = costFunctionJ(X, y, theta)

   % X is "design matrix" containing our training examples
   % y contains the correct output for each training examples
   % theta is the hypothesis parameters


   predictions = X * theta;                % prediction of hypothesis
   sqrError = (predictions - y) .^ 2;      % squared errors
   J = 0.5 * sum(sqrError);
```

توابع

costFunctionJ.m

```
function J = costFunctionJ(X, y, theta)

   % X is "design matrix" containing our training examples
   % y contains the correct output for each training examples
   % theta is the hypothesis parameters


   predictions = X * theta;                % prediction of hypothesis
   errors = (predictions - y);             % error of predictions
   J = 0.5 * errors' * errors;
```

توابع


```
>> X = [1 1; 1 2; 1 3];
>> y = [1; 2; 3];
>> theta = [0; 1]       ;        % h(x) = θ0 + θ1x = 0 + 1 * x = x

>> j = costFunctionJ(X, y, theta)
j =   0

>> theta = [0; 0.5];    % h(x) = θ0 + θ1x = 0 + 0.5 * x = 0.5 * x
>> j = costFunctionJ(X, y, theta)
j =   1.75

>> theta = [0; 0];      % h(x) = θ0 + θ1x = 0 + 0 * x = 0
>> j = costFunctionJ(X, y, theta)
j =   7.0
```

برداری‌سازی محاسبات

برداری‌سازی

- مثال برداری‌سازی.

$$h_\theta(x) = \sum_{j=0}^{n} \theta_j x_j = \theta^T x$$

Unvectorized implementation

```
prediction = 0.0;
for j = 1:n + 1,
    prediction = prediction +
                 theta(j) * x(j);
end;
```

Vectorized implementation

```
prediction = theta' * x;
```

ساده‌تر و کاراتر

برداری‌سازی

- مثال برداری‌سازی.

$$h_\theta(x) = \sum_{j=0}^{n} \theta_j x_j = \theta^T x$$

Unvectorized implementation (JAVA)

```
double prediction = 0.0;
for (int j = 0; j <= n; j++)
    prediction += theta[j] * x[j];
```

Vectorized implementation (JAVA)

```
double prediction =
    theta.transpose().times(x);
```

با استفاده از بسته‌ی JAMA

برداری‌سازی

- مثال برداری‌سازی. گرادیان کاهشی

$$\theta_j = \theta_j - \alpha \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) \cdot x_j^{(i)}$$

- به عنوان مثال برای دو ویژگی داریم:

$$\theta_0 = \theta_0 - \alpha \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) \cdot x_0^{(i)}$$

$$\theta_1 = \theta_1 - \alpha \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) \cdot x_1^{(i)}$$

$$\theta_2 = \theta_2 - \alpha \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) \cdot x_2^{(i)}$$

برداری‌سازی

- مثال برداری‌سازی. گرادیان کاهشی

$$\theta_j = \theta_j - \alpha \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) \cdot x_j^{(i)}$$

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \boldsymbol{\delta}$$

$$\delta = \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) \cdot x^{(i)}$$

$$\begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} - \alpha \begin{bmatrix} \delta_0 \\ \delta_1 \\ \vdots \\ \delta_n \end{bmatrix}$$

$$\delta = X^T(X\theta - y)$$