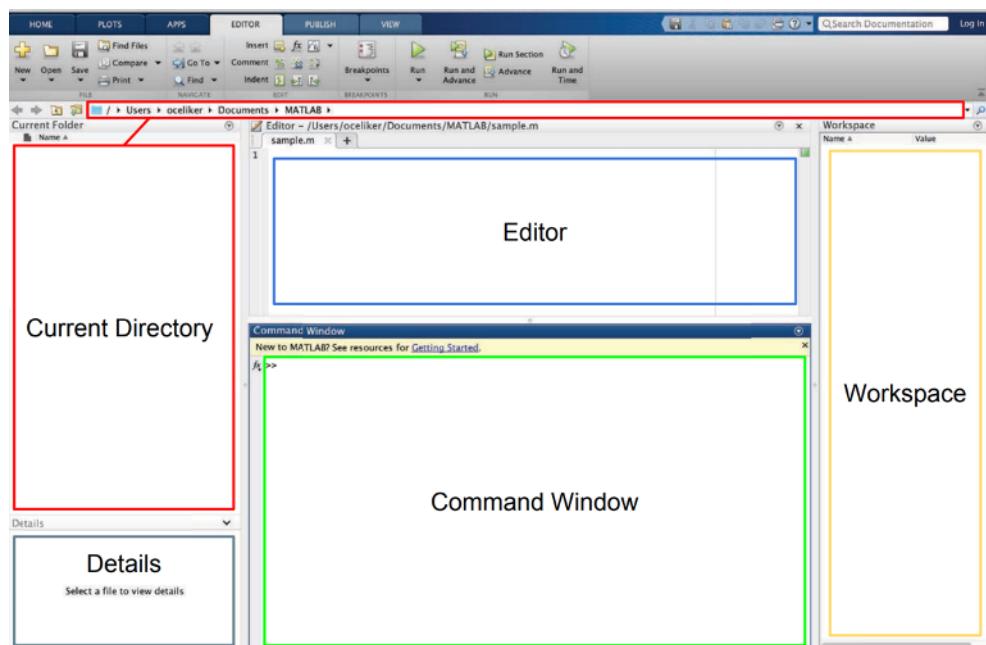


مقدمه

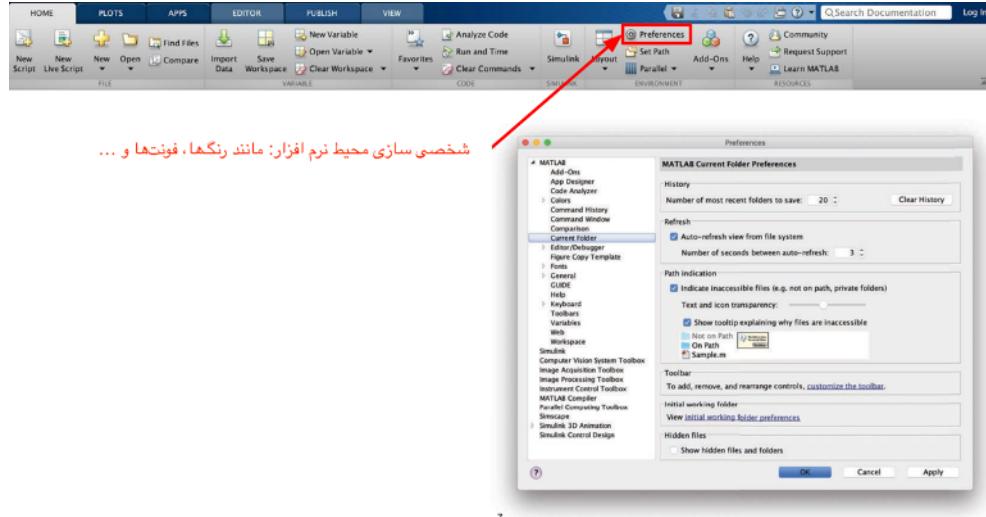
به متلب (MATLAB) می‌توان مانند یک ماشین حساب گرافیکی بسیار قدرتمند نگاه کرد. متلب (مانند پایتون) یک زبان برنامه نویسی تفسیر شده (interpreted) است که خط به خط اجرا می‌شود.

محیط

در صفحه‌ی اصلی هر قسمت به این شکل می‌باشد:



هر یک از این قسمت‌ها استفاده‌ی مشخصی دارد، برای مثال شما کدهای خود را در قسمت Editor می‌نویسید و خروجی آن‌ها در قسمت Command Window نمایش داده می‌شود. در قسمت Workspace متغیرهایی که در برنامه‌ی خود آن‌ها را تعریف کرده‌اید به همراه مشخصاتی مانند: ابعاد، (Dimension)، مقدار (Value) و ... را نشان داده می‌شود. در قسمت Current Directory آدرسی که در آن فایل کد شما وجود دارد نمایش داده می‌شود و در قسمت Details شما با انتخاب کردن یک فایل، مشخصات مربوط به آن را می‌توانید ببینید.



منابع مناسب برای یادگیری

برخی دستورات در متلب وجود دارند که به شما کمک می‌کنند تا این زبان را به صورت خودخوان بهتر یاد بگیرید.

help sin ●

○ این دستور علاوه بر نمایش توضیحات مختصر درمورد دستور (در اینجا sin)، لیستی از توابع مربوط و لینک‌های به داکیومنت‌ها را نمایش می‌دهد.

doc sin ●

○ این دستور نسخه‌ی خواناتری از مطالب را نسبت به دستور قبل و همراه با تعدادی مثال و توضیحات بهتر فراهم می‌کنند.

● سایت اختصاصی متلب نیز منبع بسیار خوبی برای یادگیری این زبان می‌باشد.

دستورات کاربردی

● تعریف بردار

```

1 >> v = [3, 2, 5]
2 >> u = [2, -1, 0]
3 v =
4     3     2     5
5 u =
6     2    -1     0

```

● جمع، تفریق، ضرب و تقسیم یک اسکالر با ماتریس

```

1 >> a = [ 10 12 23 ; 14 8 6; 27 8 9];
2 >>b = 2;
3 >>c = a + b
4 >>d = a - b
5 >>e = a * b
6 >>f = a / b
7 c =
8      12      14      25
9      16      10       8
10     29      10      11
11 d =
12      8      10      21
13     12       6       4
14     25       6       7
15 e =
16     20      24      46
17     28      16      12
18     54      16      18
19 f =
20      5.0      6.0     11.5
21      7.0      4.0      3.0
22     13.5      4.0      4.5

```

● جمع دو بردار

```

1 >>v = [3, 2, 5]
2 >>u = [2, -1, 0]
3 >>v + u
4 ans =
5      5      1      5

```

- نرم بردار (انواع نرم مانند نرم ۲، نرم بی‌نهایت، نرم فربونیوس و ...)
- در کد داده شده بردار A به این صورت می‌باشد: $A = -2i + 6j$

```

1 >>A = [ 10 12 23 ; 14 8 6; 27 8 9]
2 >>norm(A, 1)
3 >>norm(A,inf)
4 >>norm(A, 2)
5 >>norm(A, 'fro')
6 ans = 51
7 ans = 45
8 ans = 40.942
9 ans = 44.079

```

● زاویه‌ی بین بردار و محور x

```

1 >>atan2(6, -2)*180/pi
2 ans = 108.43

```

● مثال بیشتر از نرم‌ها

○ در کد داده شده بردار A به این صورت می‌باشد:

$$A = -2i + 6j$$

```

1 # Create a vector and calculate the magnitude.
2 >>v = [1 -2 3];
3 >>n = norm(v)
4
5 # Calculate the 1-norm of a vector, which is the sum of the element
   magnitudes.
6 >>v = [-2 3 -1];
7 >>n = norm(v, 1)
8
9 # Calculate the distance between 2 points
10 >>a = [0 3];
11 >>b = [-2 1];
12
13 >>d = norm(b-a)
14

```

```

15 # Calculate the 2-norm which is the largest singular value
16 >>X = [2 0 1; -1 1 0; -3 3 0];
17 >>n = norm(X)
18 n = 3.7417
19 n = 6
20 d = 2.8284
21 n = 4.7234

```

● زاویه‌ی بین بردار و محور x

```

1 >>atan2(6, -2)*180/pi
2 ans = 108.43

```

● محاسبه‌ی زاویه‌ی بین دو بردار

○ در مطلب دستور مشخصی برای محاسبه‌ی زاویه‌ی بین دو بردار وجود ندارد، بنابراین برای محاسبه‌ی این مقدار به این شکل عمل می‌کنیم.

```

1 >>u = [1 1 0]
2 >>v = [0 1 0]
3 >>CosTheta = max(min(dot(u,v)/(norm(u)*norm(v)),1),-1);
4 >>ThetaInDegrees = real(acosd(CosTheta))
5 u =
6 1 1 0
7 v =
8 0 1 0
9 ThetaInDegrees = 45.000

```

● بردار سطري و ستوني

```

1 >># row vector
2 >>r = [7 8 9 10 11]
3 >># column vector
4 >>c = [7; 8; 9; 10; 11]
5 r =
6 7 8 9 10 11

```

```

7 c =
8
9
10
11
12

```

● دسترسی به یک درایه‌ی مشخص از بردار

```

1 >>v = [ 1; 2; 3; 4; 5; 6];
2 >>v(3)
3 ans = 3

```

● بردار صفر و بردار یک

```

1 >># row and column vector of zeros
2 >>n = 8;
3 >>zeros(1,n)
4 >>zeros(n,1)
5 ans =
6 0 0 0 0 0 0 0 0
7 ans =
8 0
9 0
10 0
11 0
12 0
13 0
14 0
15 0
16 >># row and column vector of ones
17 >>ones(1,n)
18 >>ones(n,1)
19 ans =

```

```
20    1   1   1   1   1   1   1
21 ans =
22   1
23   1
24   1
25   1
26   1
27   1
28   1
29   1
```

● ضرب داخلی دو بردار

```
1 >>a = [-2, 6]
2 >>b = [6, 3]
3 >>dot (a, b)
4 ans = 6
```

● محاسبه مجموع تمام درایه‌های یک بردار

```
1 >>x = [8 5 6]
2 >>y = [5; 7; 9]
3 >>sum(x)
4 >>sum(y)
5 ans = 19
6 ans = 21
```

● دسترسی به یک درایه مشخص از ماتریس

```
1 >>a = [ 1 2 3 4 5; 2 3 4 5 6; 3 4 5 6 7; 4 5 6 7 8];
2 >>a(2,5)
3 ans = 6
```

● دسترسی به یک ستون مشخص از ماتریس

```
1 >>A = [ 1 2 3 4 5; 2 3 4 5 6; 3 4 5 6 7; 4 5 6 7 8];
```

```

2 >>v = A(:,4)
3 v =
4 4
5 5
6 6
7 7

```

● دسترسی به یک سطر مشخص از ماتریس

```

1 >>A = [ 1 2 3 4 5; 2 3 4 5 6; 3 4 5 6 7; 4 5 6 7 8];
2 >>v = A(3,:)
3 v =
4 3 4 5 6 7

```

● ایجاد زیر ماتریس‌های یک ماتریس

```

1 >>a = [ 1 2 3 4 5; 2 3 4 5 6; 3 4 5 6 7; 4 5 6 7 8];
2 >>sa = a(2:3, 2:4)
3 sa =
4 3 4 5
5 4 5 6

```

● ماتریس صفر و ماتریس یک

```

1 >>zeros(5)
2 >>ones(4,3)
3 ans =
4 0 0 0 0 0
5 0 0 0 0 0
6 0 0 0 0 0
7 0 0 0 0 0
8 0 0 0 0 0
9 ans =
10 1 1 1
11 1 1 1

```

12	1	1	1
13	1	1	1

● ماتریس همانی

```

1 >>eye(4)
2 Diagonal Matrix
3
4
5
6
    1   0   0   0
    0   1   0   0
    0   0   1   0
    0   0   0   1

```

● magic ماتریس

- دستور magic ماتریس مربعی را ایجاد می‌کند که جمع تمام درایه‌ها در هر سطر، ستون و قطر اصلی برابر یک مقدار برابر می‌باشد. در داخل پرانتز که ورودی دستور magic می‌باشد، ابعاد این ماتریس مربعی مشخص می‌شود و باید حتماً بزرگتر مساوی ۳ باشد.

```

1 >>magic(4)
2 ans =
3
4
5
6
    16     2     3    13
    5    11    10     8
    9     7     6    12
    4    14    15     1

```

● به هم چسباندن ماتریس‌ها به صورت افقی و عمودی

```

1 >>A = [ 1 2 3; 4 5 6; 7 8 9];
2 >>B = [ 0 0 0; 1 1 1; 2 2 2];
3 >>[A, B]
4 >>[A;B]
5 ans =
6
7
8
9
    1     2     3     0     0     0
    4     5     6     1     1     1
    7     8     9     2     2     2
ans =

```

10	1	2	3
11	4	5	6
12	7	8	9
13	0	0	0
14	1	1	1
15	2	2	2

● ماتریس افزوده

```

1 >>A = [ 1 2 3; 4 5 6; 7 8 9];
2 >>b = [ 1 ; -1 ; 0 ];
3 >>[A, b]
4 ans =
5   1   2   3   1
6   4   5   6  -1
7   7   8   9   0

```

● ابعاد یک ماتریس

```

1 >>A = [ 1 2 3 4 5; 2 3 4 5 6; 3 4 5 6 7; 4 5 6 7 8];
2 >>size(A)
3 ans =
4   4   5

```

● تولید ماتریس تصادفی

○ این دستور، ماتریسی را با ابعاد دلخواه (که به عنوان ورودی دریافت کرده است) از اعداد بین صفر و یک که به صورت تصادفی و با توزیع نرمال انتخاب شده‌اند، ایجاد می‌کند.

```

1 >>rand(3, 5)
2 ans =
3   0.932033   0.550139   0.862495   0.278611   0.523965
4   0.174390   0.629709   0.070209   0.820340   0.164239
5   0.563489   0.925484   0.669851   0.235872   0.512230

```

○ می‌توان بازه‌ای که اعداد از میان آن‌ها انتخاب می‌شوند را به این صورت تغییر داد. (در این مثال بازه از $[1, 0]$ به $[a, b]$ انتخاب شده است.)

```

1 >>n = 5
2 >>a = 10; b = 50;
3 >>x = a + (b-a) * rand(n)
4 x =
5   42.404   26.376   42.025   49.172   32.335
6   22.132   42.763   19.159   38.625   49.540
7   30.809   24.667   44.044   34.035   41.164
8   45.186   34.191   15.899   13.022   40.696
9   14.895   21.697   21.219   20.747   29.461

```

```

1 # r = 5*5
2 >>r = rand(5)
3
4 # r = 10*1
5 >>r = -5 + (5+5)*rand(10,1)
6
7 # r = 1*5 and numbers within the specified range
8 >>r = randi([10 50],1,5)
9
10
11 r =
12
13   0.82793   0.14622   0.72625   0.48841   0.58569
14   0.30831   0.29407   0.85688   0.40753   0.28158
15   0.77949   0.52338   0.90414   0.44349   0.2931
16   0.42873   0.61348   0.6872    0.14451   0.74523
17   0.49481   0.42957   0.59661   0.97652   0.98902
18
19 r =
20
21   2.8643

```

```

22      2.5203
23      -4.0009
24      0.7432
25      -3.1002
26      -4.129
27      0.053462
28      -1.9304
29      -2.6819
30      4.5101
31
32 r =
33
34      49          19          15          49          11

```

● محاسبه مجموع تمام درایه‌های یک ماتریس

```

1 >>A = [ 1 2 3; 4 5 6; 7 8 9];
2 >>sum(A(:))
3 ans = 45

```

● جمع و تفریق ماتریس‌ها

```

1 >>a = [ 1 2 3 ; 4 5 6; 7 8 9];
2 >>b = [ 7 5 6 ; 2 0 8; 5 7 1];
3 >>c = a + b
4 >>d = a - b
5 c =
6      8      7      9
7      6      5     14
8     12     15     10
9 d =
10    -6     -3     -3
11     2      5     -2
12     2      1      8

```

● ضرب نقطه به نقطه دو ماتریس

```

1 >>A = [ 1 2 3 ; 4 5 6; 7 8 9];
2 >>B = [ 7 5 6 ; 2 0 8; 5 7 1];
3 >>A.*B
4 ans =
5     7     10     18
6     8      0     48
7    35     56      9

```

● ضرب دو ماتریس

```

1 >>a = [ 1 2 3; 2 3 4; 1 2 5]
2 >>b = [ 2 1 3 ; 5 0 -2; 2 3 -1]
3 >>prod = a * b
4 prod =
5     18     10     -4
6     27     14     -4
7     22     16     -6

```

● به توان رساندن یک ماتریس

```

1 >>a = [ 1 2 3; 2 3 4; 1 2 5]
2 >>a ^ 2
3 >>a ^ 3
4 ans =
5     8     14     26
6    12     21     38
7    10     18     36
8 ans =
9     62    110    210
10    92    163    310
11    82    146    282

```

● به توان رساندن درایه‌های یک ماتریس

```

1 >>a = [ 1 2 3; 2 3 4; 1 2 5];
2 >>a.^2
3 ans =
4   1     4     9
5   4     9    16
6   1     4    25

```

● عملگر \ و /

```

1 >>a = [ 1 2 3 ; 4 5 6; 7 8 9];
2 >>b = [ 7 5 6 ; 2 0 8; 5 7 1];
3 >>c = a / b
4 >>d = a \ b
5 c =
6   -0.5254    0.6864    0.6610
7   -0.4237    0.9407    1.0169
8   -0.3220    1.1949    1.3729
9 d =
10  -3.2778   -1.0556   -4.8611
11  -0.1111    0.1111   -0.2778
12   3.0556    1.2778    4.3056

```

● محاسبه ترانهادهی یک ماتریس

```

1 >>A = [ 10 12 23 ; 14 8 6; 27 8 9]
2 >>B = A '
3 A =
4   10    12    23
5   14     8     6
6   27     8     9
7 B =
8   10    14    27
9   12     8     8
10  23     6     9

```

```

11
12 >>A = [ 1+2i 3+4i; 5+6i 7+8j]
13 >>B = A . '
14 >>C = A '
15 A =
16   1 + 2i   3 + 4i
17   5 + 6i   7 + 8i
18 B =
19   1 + 2i   5 + 6i
20   3 + 4i   7 + 8i
21 C =
22   1 - 2i   5 - 6i
23   3 - 4i   7 - 8i

```

● محاسبه دترمینان یک ماتریس

```

1 >>a = [ 1 2 3; 2 3 4; 1 2 5]
2 >>det(a)
3 ans = -2

```

● دریافت قطر ماتریس و نشان دادن آن به صورت یک بردار

```

1 >>A = [ 1 2 3; 2 3 4; 1 2 5]
2 >>diag(A)
3 ans =
4   1
5   3
6   5

```

● محاسبه اثر ماتریس (trace)

```

1 >>A = [ 1 2 3; 2 3 4; 1 2 5]
2 >>trace(A)
3 ans = 9

```

● محاسبه رتبه ماتریس

```

1 >>A = [ 1 2 3; 2 3 4; 1 2 5]
2 >>rank(A)
3 ans = 3

```

● محاسبه وارون ماتریس

```

1 >>A = [1, 2; 3, 4]
2 >>inv(A)
3 >>inv(A)*A
4 ans =
5 -2.0000 1.0000
6 1.5000 -0.5000
7 ans =
8 1.0000 0
9 0.0000 1.0000

```

● محاسبه شبیه وارون ماتریس

```

1 >>A = magic(8)
2 >>A = A(:,1:6)
3 >>pinv(A)
4 >>pinv(A)*A

```

● محاسبه ماتریس بالا مثلثی و پایین مثلثی یک ماتریس

```

1 >>A = [1 2 3; 4 5 6; 7 8 9]
2 >>upperDiagram = triu(A,1)
3 >>lowerDiagram = tril(A,-1)
4 A =
5 1 2 3
6 4 5 6
7 7 8 9
8 upperDiagram =
9 0 2 3
10 0 0 6

```

```

11      0   0   0
12 lowerDiagram =
13      0   0   0
14      4   0   0
15      7   8   0

```

● تمام جایگشت‌های ممکن برای یک ماتریس

```

1 >>v = [2 4 6];
2 >>P = perms(v)
3
4 P = 6×3
5
6      6     4     2
7      6     2     4
8      4     6     2
9      4     2     6
10     2     6     4
11     2     4     6

```

● متعامد بودن یا نبودن ماتریس

```

1 % Driver Code
2 >>a = [0,1 ; -1,0];
3 >>b = [2,3 ; 4,-3]
4
5 % Function to check if a matrix is orthogonal
6 >>function isOrthogonal = checkOrthogonal(a)
7 >>     [m, n] = size(a);
8 >>     if m ~= n
9 >>         isOrthogonal = false;
10 >>     return;
11 >> end
12

```

```

13 % Find transpose
14 >> trans = a';
15
16 % Find product of a and its transpose
17 >> prod = a * trans;
18
19 % Check if product is identity matrix
20 >> identity = eye(n);
21 >> isOrthogonal = isequal(prod, identity);
22 >> end
23 >> if checkOrthogonal(a)
24     disp('Yes');
25 >> else
26     disp('No');
27 >> end
28 >> if checkOrthogonal(b)
29     disp('Yes');
30 >> else
31     disp('No');
32 >> end
33
34 Yes
35 No

```

● ماتریس‌های خودتوان

```

1 % Driver function
2 >> mat = [2, -2, -4; -1, 3, 4; 1, -2, -3];
3 >> mat2 = [2, -1, -3; 1, 3, -4; 4, 2, -3];
4
5 % Function for matrix multiplication
6 >> function res = multiply(mat)
7 >>     N = size(mat, 1);

```

```
8 >> res = zeros(N, N);
9 >> for i = 1:N
10 >>     for j = 1:N
11 >>         for k = 1:N
12 >>             res(i, j) = res(i, j) + mat(i, k) * mat(k, j);
13 >>         end
14 >>     end
15 > end
16
17
18 % Function to check idempotent property of matrix
19 >>function isIdempotent = checkIdempotent(mat)
20 >> N = size(mat, 1);
21 >> res = multiply(mat);
22 >> isIdempotent = isequal(mat, res);
23 >>end
24 % checkIdempotent function call
25 >>if checkIdempotent(mat)
26 >>     disp('Idempotent Matrix');
27 >>else
28 >>     disp('Not Idempotent Matrix');
29 >>end
30 >>if checkIdempotent(mat2)
31 >>     disp('Idempotent Matrix');
32 >>else
33 >>     disp('Not Idempotent Matrix');
34 >>end
35
36 Idempotent Matrix
37 Not Idempotent Matrix
```

● ماتریس پوچ توان

```
1 % Driver function
2 >>mat = [5, -3, 2; 15, -9, 6; 10, -6, 4]
3 >>flag = 1;
4
5 >>N = size(mat, 1);
6 >>res = zeros(N, N);
7
8 >>for i = 1:N
9 >>    for j = 1:N
10 >>        for k = 1:N
11 >>            res(i, j) = res(i, j) + mat(i, k) * mat(k, j);
12 >>        end
13 >>    end
14 >>end
15 >>for i = 1 : 3
16 >>    for j = 1 : 3
17 >>        if res(i,j) != 0
18 >>            disp('Not Nilpotent Matrix')
19 >>        flag = 0;
20 >>    endif
21 >> endfor
22 >>end
23
24 >>if flag == 1
25 >>    disp('Nilpotent Matrix');
26 >>end
27
28 mat =
29
30      5          -3           2
31     15          -9           6
```

32	10	-6	4
33			
34	Nilpotent Matrix		

● مرتب کردن عناصر یک آرایه

```

1 >>v = [ 23 45 12 9 5 0 19 17] % horizontal vector
2 >>sort(v) % sorting v
3 ans =
4      0      5      9     12     17     19     23     45

```

● مرتب کردن درایه‌های یک ماتریس (سطری و ستونی)

```

1 >>m = [2 6 4; 5 3 9; 2 0 1] % two dimensional array
2 >>sort(m, 1) % sorting m along the row
3 >>sort(m, 2) % sorting m along the column
4 m =
5      2      6      4
6      5      3      9
7      2      0      1
8 ans =
9      2      0      1
10     2      3      4
11     5      6      9
12 ans =
13     2      4      6
14     3      5      9
15     0      1      2

```

● حل دستگاه معادلات خطی

$$(Ax = b)$$

○ روش اول:

```

1 >>A = [3, 4, 5; 2, -3, 7; 1, -6, 1]
2 >>b = [2; -1; 3]

```

```

3 >>x = A \ b
4 X =
5   2.6196
6  -0.2283
7  -0.9891

```

○ روش دوم:

```

1 >>A = [3, 4, 5; 2, -3, 7; 1, -6, 1]
2 >>b = [2; -1; 3]
3 >>x = inv(A) * b
4 X =
5   2.6196
6  -0.2283
7  -0.9891

```

○ روش سوم:

```

1 >>A = [3, 4, 5; 2, -3, 7; 1, -6, 1]
2 >>b = [2; -1; 3]
3
4 >>X = linsolve(A,b)
5 X =
6   2.6196
7  -0.2283
8  -0.9891

```

● محاسبه مقادیر ویژه و شعاع طیفی

```

1 >>A = [0 1 2;
2     1 0 -1;
3     2 -1 0]
4 >>e = eig(A)
5 >>max(abs(eig(A)))
6 e =

```

```

7 -2.7321
8 0.7321
9 2.0000
10 ans = 2.7321

```

● محاسبه عدد حالت

```

1 >>A = [0.333, 0.333; 0.333, 0.3]
2 >>B = inv(A)
3
4 # k1
5 >>norm(A, 1) * norm(B, 1)
6
7 # k2
8 >>norm(A, 2) * norm(B, 2)
9
10 # k-infinity
11 >>norm(A, inf) * norm(B, inf)
12
13 # k-fro
14 >>norm(A, 'fro') * norm(B, 'fro')
15
16 A =
17
18 0.333 0.333
19 0.333 0.3
20
21 B =
22
23 -27.3 30.303
24 30.303 -30.303
25
26 ans = 40.364

```

```

27 ans = 38.437
28 ans = 40.364
29 ans = 38.463

```

● مشخص کردن خوشحالی یا بدحالی مسأله

```

1 >>A = [2, 2; 2, 2.01];
2 <<b = [1, -1];
3
4 >>delta_A = [-0.001, 0; 0, 0];
5 >>delta_B = [0.0001; 0.0001];
6
7 >>d = 4;
8 >>k = 2;
9 >>result = 10^(d - k - 1)
10
11 >>k_1 = norm(A,1) * norm(inv(A),1)
12
13 >>if k_1 > 10
14     disp("ill-conditioned");
15 else
16     disp("well-conditioned");
17 end
18
19 result = 10
20 k_1 = 804
21 ill-conditioned

```

● تصویر دو بردار

```

1 >>v = [1,1]
2 >>u = [2,0]
3
4 >>proj = (dot(u,v)/(norm(u,2)^2))*u

```

```

5
6 v =
7
8     1         1
9
10 u =
11
12     2         0
13
14 proj =
15
16     1         0

```

● محاسبه بردارهای ویژه یک ماتریس

```

1 >>A = [10 -6 2;
2      -6 7 -4;
3      2 -4 3];
4 >>[V,D,W] = eig(A);
5 >>disp("Right eigenvectors :")
6 >>disp(V);
7
8 Diagonal matrix of Eigenvalues :
9 Diagonal Matrix
10
11   0.1618          0          0
12       0      3.8694          0
13       0          0     15.9688
14 Right eigenvectors :
15   0.2411    0.6460   -0.7242
16   0.6389    0.4561    0.6195
17   0.7305   -0.6121   -0.3028
18

```

```

19 >>A = [ 10 12 23 ; 14 8 6; 27 8 9]
20 >>poly(A)
21
22 A =
23
24 10 12 23
25 14 8 6
26 27 8 9
27
28 ans =
29
30 1.0000e+00 -2.7000e+01 -5.9500e+02 1.7200e+03

```

● حل چندجمله‌ای ویژه

```

1 >>[X,e] = polyeig(6,-5,1)
2 X =
3 1.0000 0.6667
4 e =
5 2.0000
6 3.0000

```

● محاسبه مقادیر منفرد

```

1 >>A = [-5, 1; 8, 0; 4, -7];
2 >>S = svds(A)
3
4 S =
5
6 10.975
7 5.8774

```

● محاسبه بردارهای منفرد

```

1 >>A = [-1,0,2;1,2,-2]

```

```

2
3 >> [U,S,V] = svd(A)
4
5 >> first = V(:, 1)
6 >> second = V(:, 2)
7 >> third = V(:, 3)
8
9 first =
10
11      0.3946
12      0.47059
13     -0.7892
14
15 second =
16
17     -0.21046
18      0.88235
19      0.42091
20
21 third =
22
23      0.89443
24      1.0143e-16
25      0.44721

```

● حل چندجمله‌ای ویژه

```

1 >> [X,e] = polyeig(6,-5,1)
2 X =
3      1.0000    0.6667
4 e =
5      2.0000
6      3.0000

```

● گرام اشمیت

```
1 >>A = [1, 3, 2; -1, 0, 6; 3, 8, 5]
2
3 >>[m, n] = size(A)
4 >>Q = zeros(m, n)
5 >>R = zeros(n, n)
6
7 >>for j=1 : n
8   v = A(:,j);
9   for i=1 : j-1
10     R(i, j) = Q(:,j)'*A(:,j);
11     v = v - R(i,j) * Q(:,i);
12   endfor
13   R(j, j) = norm(v)
14   Q(:,j) = v/R(j,j)
15 >>end
16 >>Q
17 >>R
18 >>Q * R
19 Q =
20
21      0.30151      0.35112      0.24807
22     -0.30151          0      0.74421
23      0.90453      0.93633      0.62017
24
25 R =
26
27      3.3166          0          0
28          0      8.544          0
29          0          0      8.0623
```

```

30
31 ans =
32
33     1         3         2
34     -1        0         6
35     3         8         5

```

● محاسبه تجزیه QR

```

1 >>A = magic(5)
2 >>[Q, R] = qr(A)
3 A =
4
5
6
7 Q =
8
9
10
11 R =
12
13
14

```

8	1	6
3	5	7
4	9	2

-0.848	0.522	0.090
-0.318	-0.365	-0.874
-0.424	-0.770	0.476

-9.434	-6.254	-8.162
0	-8.239	-0.965
0	0	-4.631

● محاسبه تجزیه SVD

```

1 >>A = [1 2; 3 4; 5 6]
2 >>[U, S, V] = svd(A)
3 A =
4
5
6
7 U =
8

```

1	2
3	4
5	6

-0.22985	0.88346	0.40825
----------	---------	---------

```

9      -0.52474      0.24078      -0.8165
10     -0.81964      -0.4019      0.40825
11 S =
12 Diagonal Matrix
13      9.5255          0
14          0          0.5143
15          0          0
16 V =
17      -0.61963      -0.78489
18      -0.78489      0.61963

```

- رسم دستگاه معادلات خطی دو بعدی
- روش اول:

```

1 >>x = linspace(1, 3, 100)
2 >>y = 4 - 3*x
3 >>plot(x,y)
4 >>hold on
5 >>y = (x - 6)/2
6 >>plot(x, y)
7 ans =
8      0.2673
9      -0.5345
10     0.8018

```

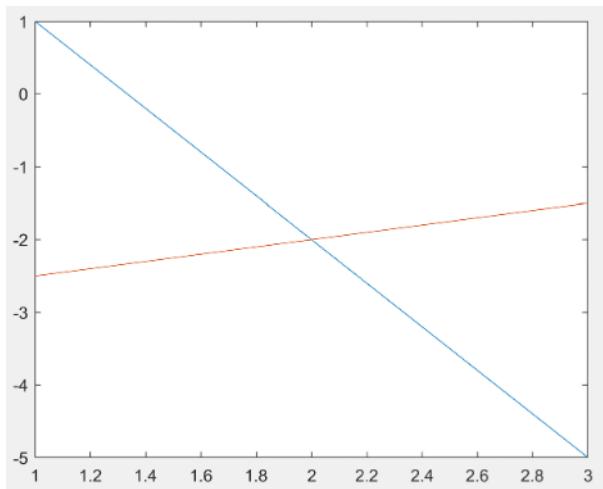
- روش دوم:

```

1 >>A = magic(8)
2 >>A = A(:, 1:6)
3 >>rank(A)
4 >>pinv(A)
5 ans =
6      0.2673
7      -0.5345

```

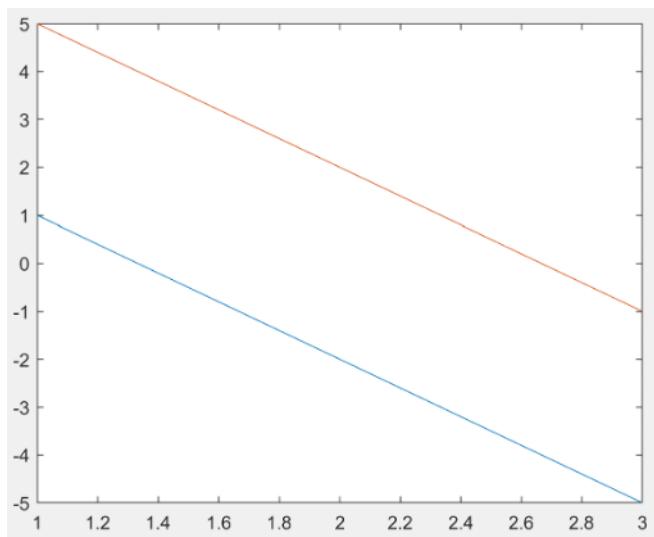
8 0.8018



```

1 >>x = linspace(1, 3, 100)
2 >>y1 = 4 - 3*x
3 >>y2 = 8 - 3*x
4 >>figure
5 >>plot(x,y1,x,y2)

```



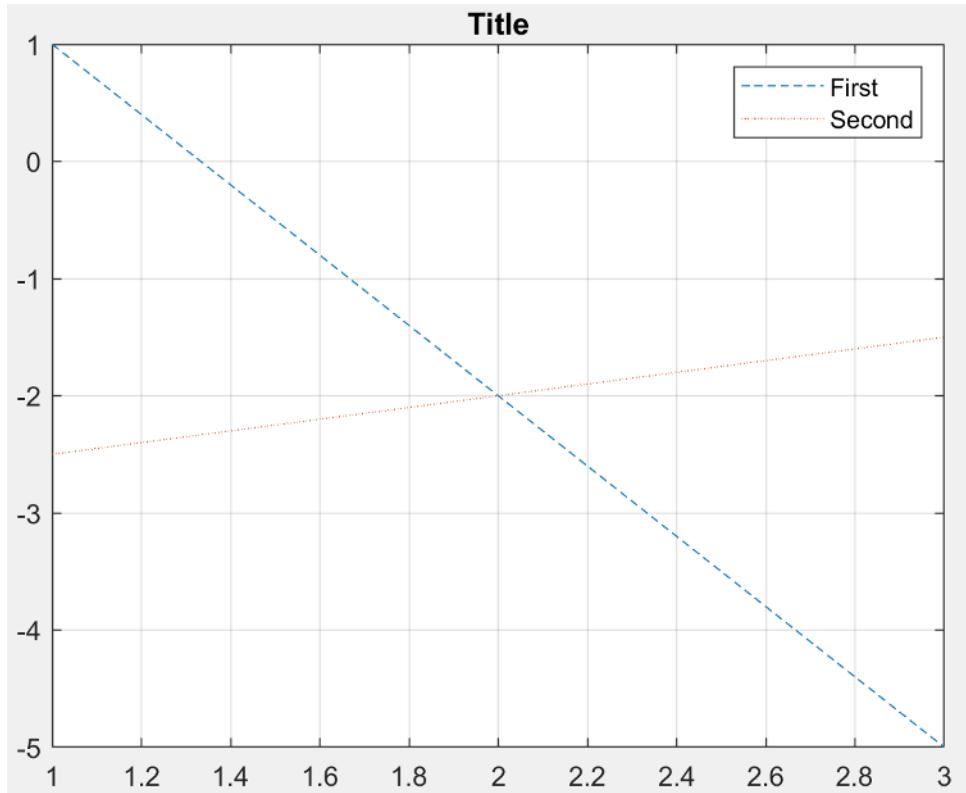
- تغییر ظاهر نمودار رسم شده
- تغییر خطوط رسم شده

- ایجاد و نشان دادن اسم هر خط
- ایجاد و نشان دادن اسم نمودار
- نمایش پس زمینه به صورت چهارخانه

```

1 >>x = linspace(1, 3, 100)
2 >>y1 = 4 - 3*x
3 >>y2 = (x - 6)/2
4 >>figure
5 >>plot(x,y1, '--', x,y2, ':'), legend('First', 'Second'), title('Title'), grid
     on

```



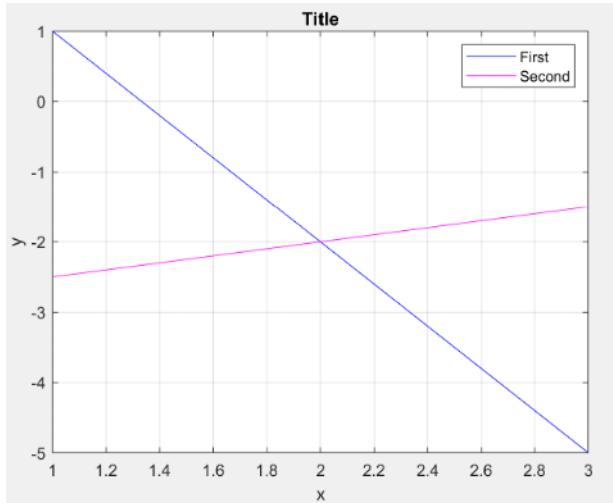
- تغییر رنگ خطوط رسم شده

```

1 >>x = linspace(1, 3, 100)
2 >>y1 = 4 - 3*x
3 >>y2 = (x - 6)/2
4 >>figure

```

```
5 >> plot(x,y1,'b',x,y2,'m'), legend('First', 'Second'), xlabel('x'), ylabel('y'), title('Title'), grid on
```

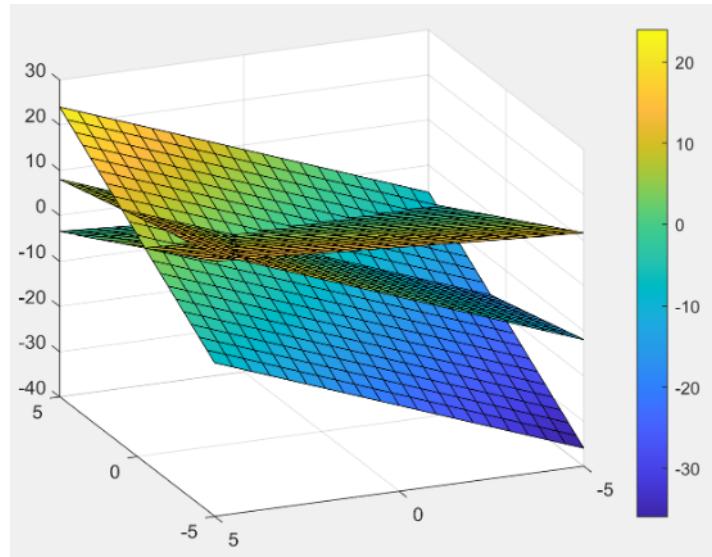


Color Name	Short Name	RGB Triplet	Hexadecimal Color Code	Appearance
"red"	"r"	[1 0 0]	"#FF0000"	
"green"	"g"	[0 1 0]	"#00FF00"	
"blue"	"b"	[0 0 1]	"#0000FF"	
"cyan"	"c"	[0 1 1]	"#00FFFF"	
"magenta"	"m"	[1 0 1]	"#FF00FF"	
"yellow"	"y"	[1 1 0]	"#FFFF00"	
"black"	"k"	[0 0 0]	"#000000"	
"white"	"w"	[1 1 1]	"#FFFFFF"	
"none"	Not applicable	Not applicable	Not applicable	No color

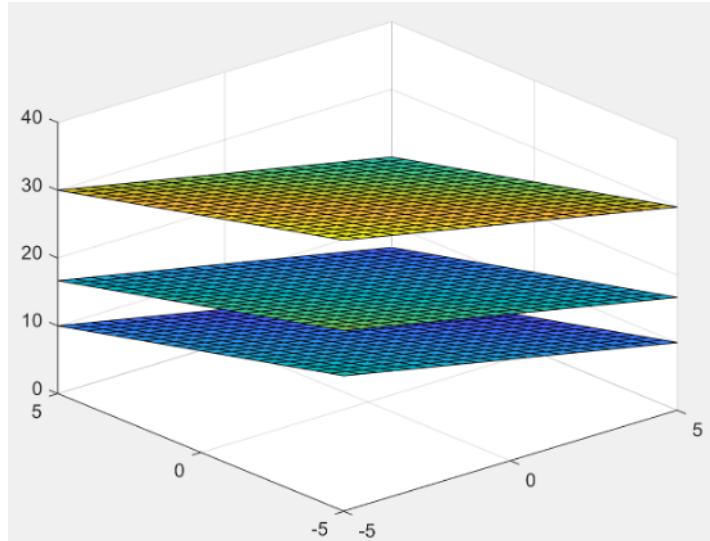
● رسم دستگاه معادلات خطی سه بعدی

```
1 >> # single point
2 >> [x,y] = meshgrid(-5:0.5:5)
3 >> z = ( 4 + 2*x - 6*y)/(-2)
4 >> surf(x,y,z)
5 >> hold on
6 >> z = (8 - 4*x + y)/2
7 >> surf(x,y,z)
8 >> hold on
9 >> z = (-6 + 3*x + 3*y)
```

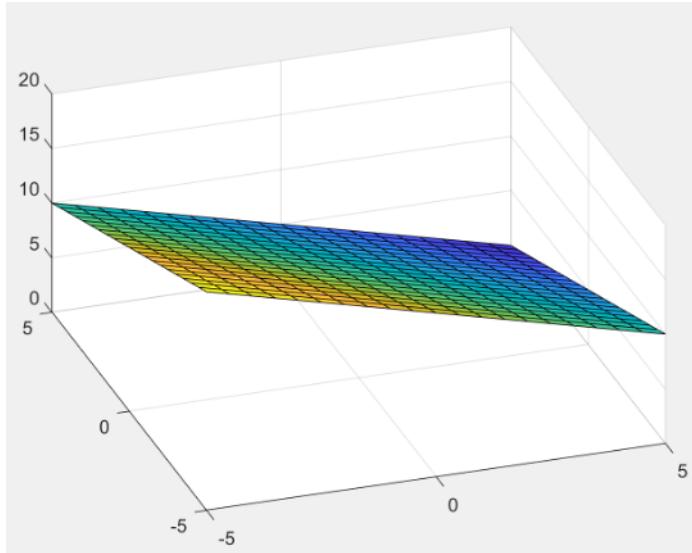
10 >>surf(x,y,z)



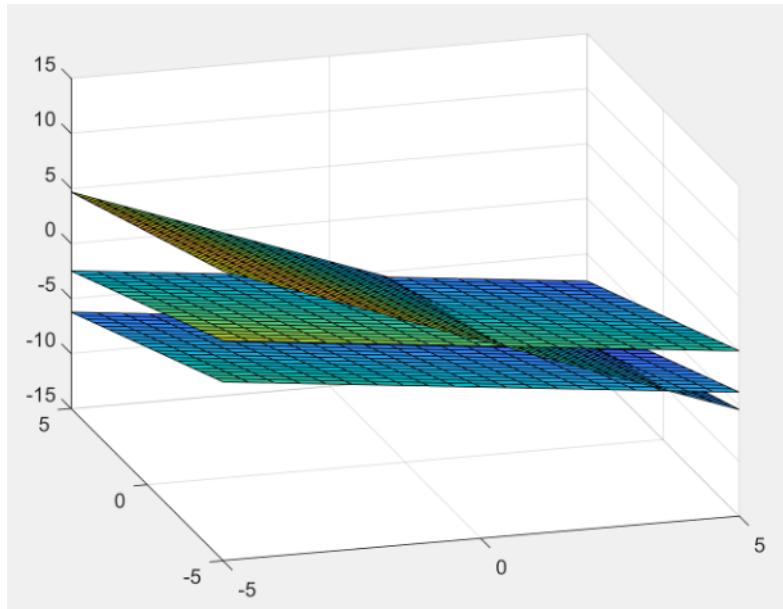
```
1 >># parallel
2 >>[x,y] = meshgrid(-5:0.5:5)
3 >>z = 10 - x - y
4 >>surf(x,y,z)
5 >>hold on
6 >>z = (60 - 2*x - 2*y)/2
7 >>surf(x,y,z)
8 >>hold on
9 >>z = (50 - 3*x - 3*y)/3
10>>surf(x,y,z)
```



```
1 >># Coincident planes
2 >>[x,y] = meshgrid(-5:0.5:5)
3 >>z = 10 - x - y
4 >>surf(x,y,z)
5 >>hold on
6 >>z = (20 - 2*x - 2*y)/2
7 >>surf(x,y,z)
8 >>hold on
9 >>z = (30 - 3*x - 3*y)/3
10>>surf(x,y,z)
```



```
1 >># Two planes intersecting with the third plane parallel to them
2 >>[x,y] = meshgrid(-5:0.5:5)
3 >>z = (-2*x-3*y-5)/4
4 >>surf(x,y,z)
5 >>hold on
6 >>z = (-2*x-3*y-8)/4
7 >>surf(x,y,z)
8 >>hold on
9 >>z = (-5*x-2*y-1)/3
10>>surf(x,y,z)
```



● بدست آوردن فضای پوچ یک ماتریس

```

1 >>A = [ 1 2 3; 4 5 6; 7 8 9]
2 >>N = null(A)
3 >>A * N
4 >>norm(A * N)
5 >>A = ones(3)
6 >>x1 = null(A)
7 >>A * x1
8 >>norm(A * x1)

9
10 A =
11
12      1          2          3
13      4          5          6
14      7          8          9
15
16 N =
17
18      -0.40825

```

```

19      0.8165
20     -0.40825
21
22 ans =
23
24 -1.1102e-16
25  2.2204e-16
26  7.7716e-16
27
28 ans = 8.1584e-16
29 A =
30
31      1          1          1
32      1          1          1
33      1          1          1
34
35 x1 =
36
37      0.8165      0
38     -0.40825   -0.70711
39     -0.40825    0.70711
40
41 ans =
42
43  3.8858e-16  3.3307e-16
44  3.8858e-16  3.3307e-16
45  3.8858e-16  3.3307e-16
46
47 ans = 8.8644e-16

```

● بدست آوردن برد یک ماتریس

```
>>A = [1 0 1; -1 -2 0; 0 1 -1]
```

```

2 >>r = rank(A)
3 >>Q = orth(A)
4
5 ans=
6 r = 3
7 Q =
8
9      0.12      0.80971     -0.57443
10     -0.90175    -0.15312     -0.40422
11      0.41526    -0.5665     -0.71179

```

تجزیه LU ●

```

1 >>A = [4, 3, 2; 6, 3, 1; 0, 5, 6]
2
3 >>[L, U, P] = lu(A)
4 >>L
5 >>U
6 >>P
7
8 L =
9
10     1         0         0
11     0         1         0
12     0.66667     0.2         1
13
14 U =
15
16     6         3         1
17     0         5         6
18     0         0     0.13333
19
20 P =

```

21
 22 Permutation Matrix
 23

24 0 1 0
 25 0 0 1
 26 1 0 0

● تجزیه PLU

```

1 >>A = [8 2 3; 3 5 6; 7 5 9]
2 >>[L, U] = lu(A)
3 >>P' * L * U
4
5 A =
6
7      8          2          3
8      3          5          6
9      7          5          9
10
11 L =
12
13      1          0          0
14      0.375       1          0
15      0.875       0.76471     1
16
17 U =
18
19      8          2          3
20      0          4.25       4.875
21      0          0          2.6471
22
23 P =
24

```

```

25 Permutation Matrix
26
27 1 0 0
28 0 1 0
29 0 0 1
30
31 ans =
32
33 8 2 3
34 3 5 6
35 7 5 9

```

● تجزیه چولسکی

```

1 >>A = [8 2 3; 3 5 6; 7 5 9]
2 >>R = chol(A)
3 >># LL^T
4 >>R' * R
5
6 A =
7
8 8 2 3
9 3 5 6
10 7 5 9
11
12 R =
13
14 2.8284 0.70711 1.0607
15 0 2.1213 2.4749
16 0 0 1.3229
17
18 ans =
19

```

20	8	2	3
21	2	5	6
22	3	6	9

● مشخص کردن متقارن معین مثبت بودن ماتریس
 ○ راه حل اول:

```

1 >># Try Choleski decomposition
2 >>A = [1 -1 0; -1 5 0; 0 0 7]
3 >>try chol(A)
4 >>     disp( 'Matrix is symmetric positive definite.' )
5 >> catch ME
6 >>     disp( 'Matrix is not symmetric positive definite.' )
7 >> end
8
9 >>A = [5 2 6; 0 5 9; -9 -8 7]
10 >>try chol(A)
11 >>     disp( 'Matrix is symmetric positive definite.' )
12 >> catch ME
13 >>     disp( 'Matrix is not symmetric positive definite.' )
14 >> end
15
16 A =
17
18
19
20
21
22 ans =
23
24
25
26

```

18	1	-1	0
19	-1	5	0
20	0	0	7
24	1	-1	0
25	0	2	0
26	0	0	2.6458

```

27
28 Matrix is symmetric positive definite.
29 A =
30
31      5          2          6
32      0          5          9
33     -9         -8          7
34
35 Matrix is not symmetric positive definite.

```

راه حل دوم:

```

1 >> # Check eigenvalues
2 >> # using d>=0 you can check whether a matrix is a symmetric positive
   semidefinite matrix
3 >>A = [1 -1 0; -1 5 0; 0 0 7]
4 >>tf = issymmetric(A)
5 >>d = eig(A)
6 >>isposdef = all(d > 0)
7 >>B = [5 2 6; 0 5 9; -9 -8 7]
8 >>tf = issymmetric(B)
9 >>d = eig(B)
10 >>isposdef = all(d > 0)
11
12 A =
13
14      1          -1          0
15     -1           5          0
16      0           0          7
17
18 tf = 1
19 d =
20

```

```

21      0.76393
22      5.2361
23      7
24
25 isposdef = 1
26 B =
27
28      5          2          6
29      0          5          9
30      -9         -8         7
31
32 tf = 0
33 d =
34
35      3.7543 +      0i
36      6.6229 +     11.288i
37      6.6229 -     11.288i
38
39 isposdef = 1

```

● حلقه‌ها

```

1 >># for loop
2 >>x = ones(1, 10)
3 >>for n=2:6
4 >>    x(n) = 2 * x(n - 1)
5 >>end
6 >># while loop
7 >>n = 1;
8 >>max = 1;
9 >>while max < 100
10 >>    n = n + 1
11 >>    max = max * n;

```

```

12 >>end
13 >># Nested Loops
14 >>A = zeros(2, 3)
15 >>for m = 1:2
16 >>    for n = 1:3
17 >>        A(m, n) = 2 * (m + n)
18 >>    end
19 >>end

```

● پیاده سازی روش تکراری ژاکوبی برای حل دستگاه معادلات خطی ○ روش ماتریسی

```

1 >>A = [10 1 -1; 3 -5 4; 1 -3 10];
2 >>b = [9; 5; 25];
3 >>n = length(A);
4 >>L = tril(A, -1);
5 >>U = triu(A, 1);
6 >>D = diag(diag(A));
7 >>x0 = zeros(n,1);
8 >>MJ = -D \ (L+U);
9 >>cJ = D \ b;
10 >>k = 0;
11
12 >>while k <= 5
13 >>    k = k + 1
14 >>    X = MJ * x0 + cJ
15 >>    x0 = X;
16 >>end
17
18 >>e = A \ b
19
20 k =      1
21 X =

```

22
23 0.9
24 -1
25 2.5
26
27 $k = 2$
28 $X =$
29
30 1.25
31 1.54
32 2.11
33
34 $k = 3$
35 $X =$
36
37 0.957
38 1.438
39 2.837
40
41 $k = 4$
42 $X =$
43
44 1.0399
45 1.8438
46 2.8357
47
48 $k = 5$
49 $X =$
50
51 0.99919
52 1.8925

```

53      2.9491
54
55 k = 6
56 X =
57
58      1.0057
59      1.9588
60      2.9678
61
62 e =
63
64      1
65      2
66      3

```

روش نقطه‌ای

```

1 >>A = [10 1 -1; 3 -5 4; 1 -3 10];
2 >>b = [9; 5; 25];
3 >>a = A;
4 >>n = length(A);
5 >>x0 = zeros(n, 1);
6 >>Iteration = 10;
7 >>for i = 1:n
8 >>    x(i) = ((b(i) - a(i,[1:i - 1, i + 1:n])) * x0([1: i - 1, i + 1:n]))/a(i,
9 >>    i))
10 >>    end
11 >>x1 = x';
12 >>for k=1:Iteration
13 >>    for i=1:n
14 >>        xx(i) = ((b(i) - a(i, [1:i-1,i+1:n])) * x1([1:i-1,i+1:n]))/a(i,i))
15 >>    end
15 >> x1=xx';

```

```
16 >> end  
17  
18 xx =  
19  
20      1.0001      1.9989      2.9993
```