MATLAB PROJECT 3

Please include this page in your Group file, as a front page. Type in the group number and the names of all members WHO PARTICIPATED in this project.

GROUP # 26

FIRST & LAST NAMES (UFID numbers are NOT required):

1. James Luberisse
2. Anthony Khmarin
3. Hao Lin
4. William Liu
5. Sean Madsen
6. Ryan Houston

By including your names above, each of you had confirmed that you did the work and agree with the work submitted.

Exercise 1

```
type rowspace
```

```
function []=rowspace(A,B)
a = rref(A);
b = rref(B);
if(size(A,2) == size(B,2))
    fprintf('Row A and Row B are subspaces of R^%i\n',size(A,2))
    fprintf('The dimension of Row A is %i\n',rank(A));
    fprintf('The dimension of Row B is %i\n',rank(B));
    if(rank(A) >= size(A,2) && (rank(B) >= size(B,2)))
        fprintf('Row A and Row B spans the whole R^%i\n',size(B,2))
        return
    else
        if(rank(A) == rank(B))
                pa = zeros(size(A,2),rank(A));
                ta = transpose(a);
                for i = 1:rank(A)
                    pa(:,i) = ta(:,i);
                end

                pb = zeros(size(A,2),rank(A));
                tb = transpose(b);
                for i = 1:rank(A)
                    pb(:,i) = tb(:,i);
                end

            if(pa == pb)
                disp('A basis for Row A and Row B is')
                P = pa
            else
                disp('Row A and Row B have the same dimensions but are not equal')
            end
        else
            disp('Row A and Row B have diffrent dimensions and cannot be equal')
        end
    end

else
    disp('The Rowspaces of A and B are of diffrent vector spaces')
end
```

```
C=[2 6 2 4 -6;-4 -9 -7 -2 3;-2 -5 -3 -2 3;3 8 9 -1 4]
```

```
C = 4×5
     2     6     2     4    -6
    -4    -9    -7    -2     3
    -2    -5    -3    -2     3
     3     8     9    -1     4
```

```
%(a)
A=C, B=rref(C)
```

```
A = 4×5
     2     6     2     4    -6
    -4    -9    -7    -2     3
    -2    -5    -3    -2     3
     3     8     9    -1     4
B = 4×5
     1     0     0     0    -2
     0     1     0     1    -1
     0     0     1    -1     2
```

```
     0     0     0     0     0
```

```
rowspace(A,B)
```

```
Row A and Row B are subspaces of R^5
The dimension of Row A is 3
The dimension of Row B is 3
A basis for Row A and Row B is
P = 5×3
     1     0     0
     0     1     0
     0     0     1
     0     1    -1
    -2    -1     2
```

```
%(b)
A=A',B=B'
```

```
A = 5×4
     2    -4    -2     3
     6    -9    -5     8
     2    -7    -3     9
     4    -2    -2    -1
    -6     3     3     4
B = 5×4
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     1    -1     0
    -2    -1     2     0
```

```
rowspace(A,B)
```

```
Row A and Row B are subspaces of R^4
The dimension of Row A is 3
The dimension of Row B is 3
Row A and Row B have the same dimensions but are not equal
```

```
% Since the transpose of the matrices have diffrent
% row spaces, this indicates that the elementry row
% operations done by rref(C) changed the column space
% of C

%(c)
A=[C;zeros(2,5)], B=rref(C)
```

```
A = 6×5
     2     6     2     4    -6
    -4    -9    -7    -2     3
    -2    -5    -3    -2     3
     3     8     9    -1     4
     0     0     0     0     0
     0     0     0     0     0
B = 4×5
     1     0     0     0    -2
     0     1     0     1    -1
     0     0     1    -1     2
     0     0     0     0     0
```

```
rowspace(A,B)
```

```
Row A and Row B are subspaces of R^5
```

```
The dimension of Row A is 3
The dimension of Row B is 3
A basis for Row A and Row B is
P = 5×3
     1     0     0
     0     1     0
     0     0     1
     0     1    -1
    -2    -1     2
```

```
%(d)
A=magic(3);B=magic(4);
rowspace(A,B)
```

```
The Rowspaces of A and B are of diffrent vector spaces
```

```
%(e)
A=magic(4);B=hilb(4);
rowspace(A,B)
```

```
Row A and Row B are subspaces of R^4
The dimension of Row A is 3
The dimension of Row B is 4
Row A and Row B have diffrent dimensions and cannot be equal
```

```
%(f)
A=magic(5);B=hilb(5);
rowspace(A,B)
```

```
Row A and Row B are subspaces of R^5
The dimension of Row A is 5
The dimension of Row B is 5
Row A and Row B spans the whole R^5
```

```
%(g)
A=magic(5);B=[A;eye(3,5)];
rowspace(A,B)
```

```
Row A and Row B are subspaces of R^5
The dimension of Row A is 5
The dimension of Row B is 5
Row A and Row B spans the whole R^5
```

Exercise 2

Part 1:

```
type shrink
```

```
function B=shrink(A)
[~,pivot]=rref(A);
B=A(:,pivot);
end
```

```
A=magic(4)
```

```
A = 4×4
    16     2     3    13
     5    11    10     8
     9     7     6    12
```

```
     4    14    15     1
```

```
rref(A)
```

```
ans = 4×4
     1     0     0     1
     0     1     0     3
     0     0     1    -3
     0     0     0     0
```

```
[r,pivot]=rref(A)
```

```
r = 4×4
     1     0     0     1
     0     1     0     3
     0     0     1    -3
     0     0     0     0
pivot = 1×3
     1     2     3
```

```
P=A(:,pivot)
```

```
P = 4×3
    16     2     3
     5    11    10
     9     7     6
     4    14    15
```

The first is A, the second is the reduced row echelon form of A, then it splits up the result and the pivot columns, and makes a matrix P out of only the pivot columns of A.

```
B=shrink(A)
```

```
B = 4×3
    16     2     3
     5    11    10
     9     7     6
     4    14    15
```

This forms a basis for the column space of A because it only includes the pivot columns of A.

```
R=rref(transpose(A)), BC=colspace(sym(A))
```

```
R = 4×4
     1     0     0     1
     0     1     0     3
     0     0     1    -3
     0     0     0     0
BC =
```

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 3 & -3 \end{pmatrix}$$

colspace(sym(A)) generates a symbolic matrix by first transforming A into a symbolic matrix, and then finding the columns of A that are pivot columns.

```
DC=double(BC)
```

```
DC = 4×3
     1     0     0
     0     1     0
     0     0     1
     1     3    -3
```

Part 2:

```
type noll.m
```

```
function [C,p] = noll(A)
[m,n]=size(A);
C=[];
p=0;
if rank(A) == n
    fprintf('the null space is the zero subspace of R^%i\n',n)
    C = zeros(n,1);
else
    [~,pivot_c] = rref(A);
    S=1:n;
    nonpivot_c = setdiff(S,pivot_c);
    q=numel(nonpivot_c);
    j=1:q;
    C=zeros(n,q);
    Aref = rref(A);
    colOfZero = 0;
    for k = 1:n
        if Aref(:,k) == zeros(m,1)
            colOfZero = 1 + colOfZero;
        else
            break;
        end
    end
    C(1+colOfZero:m+colOfZero,j)= -Aref(1:m,nonpivot_c(j));
    for i = 1:q
     C(nonpivot_c(i),i) = 1;
    end
    p = n - rank(A)
    C = C(1:n,:);
    fprintf('the null space is a %i-dimentional subspace of R^%i\n',p,n)
    fprintf('A basis for Nul A is formed by the columns of the matrix')
    end
end
```

```
A=magic(5)
```

```
A = 5×5
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```

```
[C,p]=noll(A)
```

```
the null space is the zero subspace of R^5
C = 5×1
     0
     0
     0
     0
     0
```

```
p = 0
```

```
%(a)
A=[2 -4 -2 3; 6 -9 -5 8; 2 -7 -3 9; 4 -2 -2 -1;  -6 3 3 4]
```

```
A = 5×4
     2    -4    -2     3
     6    -9    -5     8
     2    -7    -3     9
     4    -2    -2    -1
    -6     3     3     4
```

```
[C,p]=noll(A)
```

```
p = 1
the null space is a 1-dimentional subspace of R^4
A basis for Nul A is formed by the columns of the matrix
C = 4×1
    0.3333
   -0.3333
    1.0000
         0
p = 1
```

```
N=null(A,'r' )
```

```
N = 4×1
    0.3333
   -0.3333
    1.0000
         0
```

```
isequal(C,N)
```

```
ans = logical
   1
```

```
%(b)
A=ones(5)
```

```
A = 5×5
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
```

```
[C,p]=noll(A)
```

```
p = 4
the null space is a 4-dimentional subspace of R^5
A basis for Nul A is formed by the columns of the matrix
C = 5×4
    -1    -1    -1    -1
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
p = 4
```

```
N=null(A,'r' )
```

```
N = 5×4
    -1    -1    -1    -1
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

```
isequal(C,N)
```

```
ans = logical
   1
```

```
%(c)
A=[magic(4),ones(4,1)]
```

```
A = 4×5
    16     2     3    13     1
     5    11    10     8     1
     9     7     6    12     1
     4    14    15     1     1
```

```
[C,p]=noll(A)
```

```
p = 2
the null space is a 2-dimentional subspace of R^5
A basis for Nul A is formed by the columns of the matrix
C = 5×2
   -1.0000   -0.0588
   -3.0000   -0.1176
    3.0000    0.0588
    1.0000         0
         0    1.0000
p = 2
```

```
N=null(A,'r' )
```

```
N = 5×2
   -1.0000   -0.0588
   -3.0000   -0.1176
    3.0000    0.0588
    1.0000         0
         0    1.0000
```

```
isequal(C,N)
```

```
ans = logical
   1
```

The null space function works by finding rational basis vectors for the matrix from the RREF.

```
A=[2 -4 -2 3; 6 -9 -5 8; 2 -7 -3 9; 4 -2 -2 -1; -6 3 3 4]
```

```
A = 5×4
     2    -4    -2     3
     6    -9    -5     8
     2    -7    -3     9
     4    -2    -2    -1
    -6     3     3     4
```

```
BR=double(colspace(sym(A')))
```

```
BR = 4×3
    1.0000         0         0
         0    1.0000         0
   -0.3333    0.3333         0
         0         0    1.0000
```

```
q = rank(BR)
```

```
q = 3
```

```
n = rank(A)
```

```
n = 3
```

Exercise 3

```
type polyspace.m
```

```
function P=polyspace(B,Q,rB)
format
n = numel(B);
P=zeros(n);


% Fill the array with the polynomial.
for i = 1:n
    P(:,i) = sym2poly(B(i));
end

P=closetozeroroundoff(P,7);
fprintf('matrix of E-coordinate vectors of the polynomials in B is\n')
display(P)


%Check to see if P forms a basis for R^n.

if (rank(P) == n )

   fprintf('the polynomials in B form a basis for P%d\n',n-1)
   q = sym2poly(Q);
   q = closetozeroroundoff(q, 7);
   B = rref([P transpose(q)]);
   qB = B(:,end);
   fprintf('the B-coordinate vector of Q is\n')
   display(qB)


   R = P*rB;
   R = closetozeroroundoff(R, 7);
   R = poly2sym(R);
   fprintf('the polynomial whose B-coordinates form the vector rB is\n')
   display(R)


else
    sprintf('the polynomials in B do not form a basis for P%d',n-1)
    A=rref(P);
    fprintf('the reduced echelon form of P is\n')
    disp(A)

end
end
```

```
type closetozeroroundoff.m
```

```
function B=closetozeroroundoff(A,p)
A(abs(A)<10^-p)=0;
B=A;
end
```

```
syms x
%(a)
B=[x^3+3*x^2,10^(-8)*x^3+x,10^(-8)*x^3+4*x^2+x,x^3+x]
```

B =

$$\left(x^3+3\,x^2 \quad \frac{x^3}{100000000}+x \quad \frac{x^3}{100000000}+4\,x^2+x \quad x^3+x\right)$$

```
Q=10^(-8)*x^3+x^2+6*x
```

Q =

$$\frac{x^3}{100000000}+x^2+6\,x$$

```
rB=[2;-3;1;0]
```

```
rB = 4×1
     2
    -3
     1
     0
```

```
P=polyspace(B,Q,rB);
```

```
matrix of E-coordinate vectors of the polynomials in B is
P = 4×4
     1     0     0     1
     3     0     4     0
     0     1     1     1
     0     0     0     0
ans =
'the polynomials in B do not form a basis for P3'
the reduced echelon form of P is
    1.0000         0         0    1.0000
         0    1.0000         0    1.7500
         0         0    1.0000   -0.7500
         0         0         0         0
```

```
%(b)
B=[x^3-1,10^(-8)*x^3+2*x^2,10^(-8)*x^3+x,x^3+x]
```

B =

$$\left(x^3-1 \quad \frac{x^3}{100000000}+2\,x^2 \quad \frac{x^3}{100000000}+x \quad x^3+x\right)$$

```
P=polyspace(B,Q,rB);
```

```
matrix of E-coordinate vectors of the polynomials in B is
P = 4×4
     1     0     0     1
     0     2     0     0
```

```
     0     0     1     1
    -1     0     0     0
the polynomials in B form a basis for P3
the B-coordinate vector of Q is
qB = 4×1
         0
    0.5000
    6.0000
         0
the polynomial whose B-coordinates form the vector rB is
```

R = $2x^3 - 6x^2 + x - 2$

```
%(c)
B=[x^3+1,10^(-8)*x^3+x^2+1,10^(-8)*x^3+x+1,10^(-8)*x^3+1]
```

B =

$$\left(x^3 + 1 \quad \frac{x^3}{100000000} + x^2 + 1 \quad \frac{x^3}{100000000} + x + 1 \quad \frac{x^3}{100000000} + 1\right)$$

```
Q=10^(-8)*x^3+2*x^2+x+6
```

Q =

$$\frac{x^3}{100000000} + 2x^2 + x + 6$$

```
rB=[0;-3;1;0]
```

```
rB = 4×1
     0
    -3
     1
     0
```

```
P=polyspace(B,Q,rB);
```

```
matrix of E-coordinate vectors of the polynomials in B is
P = 4×4
     1     0     0     0
     0     1     0     0
     0     0     1     0
     1     1     1     1
the polynomials in B form a basis for P3
the B-coordinate vector of Q is
qB = 4×1
     0
     2
     1
     3
the polynomial whose B-coordinates form the vector rB is
```

R = $-3x^2 + x - 2$

```
%(d)
B=[x^4+x^3+x^2+1,10^(-8)*x^4+x^3+x^2+x+1,10^(-8)*x^4+x^2+x+1,10^(-8)*x^4+x+1,10^(-8)*x^4+1]
```

B =

$$\left(x^4 + x^3 + x^2 + 1 \quad \frac{x^4}{100000000} + x^3 + x^2 + x + 1 \quad \frac{x^4}{100000000} + x^2 + x + 1 \quad \frac{x^4}{100000000} + x + 1 \quad \frac{x^4}{100000(}\right.$$

```
Q=10^(-8)*x^4+3*x^3-1
```

Q =

$$\frac{x^4}{100000000} + 3\,x^3 - 1$$

```
rB=diag(magic(5))
```

```
rB = 5×1
    17
     5
    13
    21
     9
```

```
P=polyspace(B,Q,rB);
```

```
matrix of E-coordinate vectors of the polynomials in B is
P = 5×5
     1     0     0     0     0
     1     1     0     0     0
     1     1     1     0     0
     0     1     1     1     0
     1     1     1     1     1
the polynomials in B form a basis for P4
the B-coordinate vector of Q is
qB = 5×1
     0
     3
    -3
     0
    -1
the polynomial whose B-coordinates form the vector rB is
```

R = $17\,x^4 + 22\,x^3 + 35\,x^2 + 39\,x + 65$

Exercise 4

```
type closetozeroroundoff.m
```

```
function B=closetozeroroundoff(A,p)
A(abs(A)<10^-p)=0;
B=A;
end
```

```
type quer
```

```
function [Q,R] = quer(A)
format
[m,n]=size(A);

[Q,R]=qr(A)
if (closetozeroroundoff(A-Q*R,7)==0)
    disp('the product of Q and R forms a decomposition of A');
else
    disp('no, it cannot be true!');
end

q=0;
r=0;
if(inv(Q) == transpose(Q))
```

```
    q=1;
    disp('Q is a unitary matrix');
    %check that Q is unitary
end
if (istriu(R))
    r=1;
    disp('R is an upper-triangular matrix');
    %check R is an upper-triangular matrix
end
if(q==1 && r==1)
    disp('Q*R forms an orthogonal-triangular decomposition of A');
else
    disp('Q*R does NOT forms an orthogonal-triangular decomposition of A. What is wrong?!');
end

if(m == n)
    k=1;
    P=closetozeroroundoff(A-triu(A),7);
while(P~=zeros(m,n))
    A=R*Q;
    [Q,R]=qr(A);
    P=closetozeroroundoff(A-triu(A),7);
    k=k+1;
end
disp('the matrix B');
disp(A);
disp('the number of iterations:');
disp(k);

disp('the main diagonal of the matrix B');
for i=1:m
    disp(A(i,i));
end

E = eig(A);
disp('the eigenvalues of the input matrix A')
disp(E)
end
end
```

```
%(a)
A=randi(10,3,4)
```

```
A = 3×4
     4     7     9    10
     2     5     6     3
     3     4     6     8
```

```
[Q,R] = quer(A);
```

```
Q = 3×3
   -0.7428   -0.0531   -0.6674
   -0.3714   -0.7967    0.4767
   -0.5571    0.6020    0.5721
R = 3×4
   -5.3852   -9.2848  -12.2559  -12.9987
         0   -1.9476   -1.6466    1.8945
         0         0    0.2860   -0.6674
the product of Q and R forms a decomposition of A
R is an upper-triangular matrix
Q*R does NOT forms an orthogonal-triangular decomposition of A. What is wrong?!
```

```
%(b)
A=ones(5,4);
```

```
[Q,R] = quer(A);
```

```
Q = 5×5
   -0.4472    0.8944   -0.0000         0    0.0000
   -0.4472   -0.2236    0.8660         0   -0.0000
   -0.4472   -0.2236   -0.2887   -0.5774   -0.5774
   -0.4472   -0.2236   -0.2887    0.7887   -0.2113
   -0.4472   -0.2236   -0.2887   -0.2113    0.7887
R = 5×4
   -2.2361   -2.2361   -2.2361   -2.2361
         0   -0.0000   -0.0000   -0.0000
         0         0   -0.0000   -0.0000
         0         0         0         0
         0         0         0         0
the product of Q and R forms a decomposition of A
R is an upper-triangular matrix
Q*R does NOT forms an orthogonal-triangular decomposition of A. What is wrong?!
```

```
%(c)
A=diag([2,2,4,4])
```

```
A = 4×4
     2     0     0     0
     0     2     0     0
     0     0     4     0
     0     0     0     4
```

```
[Q,R] = quer(A);
```

```
Q = 4×4
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
R = 4×4
     2     0     0     0
     0     2     0     0
     0     0     4     0
     0     0     0     4
the product of Q and R forms a decomposition of A
Q is a unitary matrix
R is an upper-triangular matrix
Q*R forms an orthogonal-triangular decomposition of A
the matrix B
     2     0     0     0
     0     2     0     0
     0     0     4     0
     0     0     0     4

the number of iterations:
     1

the main diagonal of the matrix B
     2

     2

     4

     4

the eigenvalues of the input matrix A
     2
     2
```

```
     4
     4
```

```
%(d)
A=triu(magic(4))
```

```
A = 4×4
    16     2     3    13
     0    11    10     8
     0     0     6    12
     0     0     0     1
```

```
[Q,R] = quer(A);
```

```
Q = 4×4
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
R = 4×4
    16     2     3    13
     0    11    10     8
     0     0     6    12
     0     0     0     1
the product of Q and R forms a decomposition of A
Q is a unitary matrix
R is an upper-triangular matrix
Q*R forms an orthogonal-triangular decomposition of A
the matrix B
    16     2     3    13
     0    11    10     8
     0     0     6    12
     0     0     0     1

the number of iterations:
     1

the main diagonal of the matrix B
    16

    11

     6

     1

the eigenvalues of the input matrix A
    16
    11
     6
     1
```

```
%(e)
A=triu(magic(4),-1)
```

```
A = 4×4
    16     2     3    13
     5    11    10     8
     0     7     6    12
     0     0    15     1
```

```
[Q,R] = quer(A);
```

```
Q = 4×4
   -0.9545    0.2436   -0.0011    0.1722
   -0.2983   -0.7794    0.0034   -0.5509
         0   -0.5772   -0.0051    0.8166
         0         0    1.0000    0.0062
R = 4×4
  -16.7631   -5.1900   -5.8462  -14.7944
         0  -12.1270  -10.5268   -9.9956
         0         0   15.0003    0.9524
         0         0         0    7.6357
the product of Q and R forms a decomposition of A
R is an upper-triangular matrix
Q*R does NOT forms an orthogonal-triangular decomposition of A. What is wrong?!
the matrix B
    16     2     3    13
     5    11    10     8
     0     7     6    12
     0     0    15     1

the number of iterations:
     1

the main diagonal of the matrix B
    16

    11

     6

     1

the eigenvalues of the input matrix A
   25.0011
   12.0417
    7.4036
  -10.4464
```

```
%(f)
A=tril(magic(4),1)
```

```
A = 4×4
    16     2     0     0
     5    11    10     0
     9     7     6    12
     4    14    15     1
```

```
[Q,R] = quer(A);
```

```
Q = 4×4
   -0.8230    0.4186    0.1367   -0.3590
   -0.2572   -0.5155   -0.7600   -0.3009
   -0.4629   -0.1305   -0.0997    0.8711
   -0.2057   -0.7363    0.6275   -0.1478
R = 4×4
  -19.4422  -10.5955   -8.4352   -5.7607
         0  -16.0541  -16.9816   -2.3024
         0         0    1.2138   -0.5692
         0         0         0   10.3048
the product of Q and R forms a decomposition of A
R is an upper-triangular matrix
Q*R does NOT forms an orthogonal-triangular decomposition of A. What is wrong?!
the matrix B
    16     2     0     0
```

```
     5    11    10     0
     9     7     6    12
     4    14    15     1

the number of iterations:
     1

the main diagonal of the matrix B
    16

    11

     6

     1

the eigenvalues of the input matrix A
   25.4509
   14.9803
   -7.7521
    1.3209
```

```
%(g)
A=[1 1 4;0 -4 0;-5 -1 -8]
```

```
A = 3×3
     1     1     4
     0    -4     0
    -5    -1    -8
```

```
[Q,R] = quer(A);
```

```
Q = 3×3
   -0.1961    0.1887    0.9623
         0   -0.9813    0.1925
    0.9806    0.0377    0.1925
R = 3×3
   -5.0990   -1.1767   -8.6291
         0    4.0762    0.4529
         0         0    2.3094
the product of Q and R forms a decomposition of A
R is an upper-triangular matrix
Q*R does NOT forms an orthogonal-triangular decomposition of A. What is wrong?!
the matrix B
     1     1     4
     0    -4     0
    -5    -1    -8

the number of iterations:
     1

the main diagonal of the matrix B
     1

    -4

    -8

the eigenvalues of the input matrix A
   -3.0000
   -4.0000
   -4.0000
```

Exercise 5

Part 1

```
type eluinv
```

```
function [L, U] = eluinv(A)
[m,n] = size(A);
[L,U] = lu(A)

if closetozeroroundoff(A-L*U,7) == 0
    disp('Yes, I got a factorization')
end

if m~=n
    return
else

%Code checks to see if U echelon form of A
mat = sum(U~=0,2);
tf = true;
for i=1:size(mat)-1
    if mat(i,:)<mat(i+1,:)
        tf = false;
    end
end

if tf == true
    disp('U is an echelon form of A')
else
    disp('U is not an echelon form of A? What is wrong?!')
    return
end

%Code checks to see if A is invertible
if rank(A)~=n
    sprintf('A is not invertible')
    return
end

%Calculate inverses
L1 = rref([L eye(n)]);
invL = L1(:,n+1:2*n)

U1 = rref([U eye(n)]);
invU = U1(:,n+1:2*n)

invA = invU * invL;
disp('the inverse of A calculated using LU factorization is')
disp(invA)

P = inv(A);
if closetozeroroundoff(invA-P,7) == 0
    disp('Yes, LU factorization works for calculating the inverses')
else
    disp('LU factorization does not work for me!?')
end

end
```

```
type closetozeroroundoff
```

```
function B=closetozeroroundoff(A,p)
A(abs(A)<10^-p)=0;
```

```
B=A;
end
```

```
% (a)
A = [1 1 4 3;0 -4 0 2;-5 -1 -8 1]
```

```
A = 3×4
     1     1     4     3
     0    -4     0     2
    -5    -1    -8     1
```

```
[L, U] = eluinv(A);
```

```
L = 3×3
   -0.2000   -0.2000    1.0000
         0    1.0000         0
    1.0000         0         0
U = 3×4
   -5.0000   -1.0000   -8.0000    1.0000
         0   -4.0000         0    2.0000
         0         0    2.4000    3.6000
Yes, I got a factorization
```

```
% (b)
A = [1 1 4;0 -4 0;-5 -1 -8;2 3 -1]
```

```
A = 4×3
     1     1     4
     0    -4     0
    -5    -1    -8
     2     3    -1
```

```
[L, U] = eluinv(A);
```

```
L = 4×3
   -0.2000   -0.2000   -0.5714
         0    1.0000         0
    1.0000         0         0
   -0.4000   -0.6500    1.0000
U = 3×3
   -5.0000   -1.0000   -8.0000
         0   -4.0000         0
         0         0   -4.2000
Yes, I got a factorization
```

```
% (c)
A = [1 1 4;0 -4 0;-5 -1 -8]
```

```
A = 3×3
     1     1     4
     0    -4     0
    -5    -1    -8
```

```
[L, U] = eluinv(A);
```

```
L = 3×3
   -0.2000   -0.2000    1.0000
         0    1.0000         0
    1.0000         0         0
U = 3×3
   -5.0000   -1.0000   -8.0000
```

```
         0   -4.0000         0
         0         0    2.4000
Yes, I got a factorization
U is an echelon form of A
invL = 3×3
         0         0    1.0000
         0    1.0000         0
    1.0000    0.2000    0.2000
invU = 3×3
   -0.2000    0.0500   -0.6667
         0   -0.2500         0
         0         0    0.4167
the inverse of A calculated using LU factorization is
   -0.6667   -0.0833   -0.3333
         0   -0.2500         0
    0.4167    0.0833    0.0833

Yes, LU factorization works for calculating the inverses
```

```
% (d)
A = magic(6)
```

```
A = 6×6
    35     1     6    26    19    24
     3    32     7    21    23    25
    31     9     2    22    27    20
     8    28    33    17    10    15
    30     5    34    12    14    16
     4    36    29    13    18    11
```

```
[L, U] = eluinv(A);
```

```
L = 6×6
    1.0000         0         0         0         0         0
    0.0857    0.8893   -0.7306    0.1953    1.0000         0
    0.8857    0.2261   -0.3797   -1.0000   -0.0000    1.0000
    0.2286    0.7739    0.3797    1.0000         0         0
    0.8571    0.1154    1.0000         0         0         0
    0.1143    1.0000         0         0         0         0
U = 6×6
   35.0000    1.0000    6.0000   26.0000   19.0000   24.0000
         0   35.8857   28.3143   10.0286   15.8286    8.2571
         0         0   25.5884  -11.4435   -4.1131   -5.5247
         0         0         0    7.6416   -5.0305    5.2221
         0         0         0         0    5.2718   10.5435
         0         0         0         0         0    0.0000
Yes, I got a factorization
U is an echelon form of A
ans =
'A is not invertible'
```

```
% (e)
A = [2 1 -3 1;0 5 -3 5;-4 3 3 3;-2 5 1 3]
```

```
A = 4×4
     2     1    -3     1
     0     5    -3     5
    -4     3     3     3
    -2     5     1     3
```

```
[L, U] = eluinv(A);
```

```
L = 4×4
```

```
   -0.5000    0.5000         0    1.0000
         0    1.0000         0         0
    1.0000         0         0         0
    0.5000    0.7000    1.0000         0
U = 4×4
   -4.0000    3.0000    3.0000    3.0000
         0    5.0000   -3.0000    5.0000
         0         0    1.6000   -2.0000
         0         0         0         0
Yes, I got a factorization
U is an echelon form of A
ans =
'A is not invertible'
```

```
% (f)
A = magic(5)
```

```
A = 5×5
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```

```
[L, U] = eluinv(A);
```

```
L = 5×5
    0.7391    1.0000         0         0         0
    1.0000         0         0         0         0
    0.1739    0.2527    0.5164    1.0000         0
    0.4348    0.4839    0.7231    0.9231    1.0000
    0.4783    0.7687    1.0000         0         0
U = 5×5
   23.0000    5.0000    7.0000   14.0000   16.0000
         0   20.3043   -4.1739   -2.3478    3.1739
         0         0   24.8608   -2.8908   -1.0921
         0         0         0   19.6512   18.9793
         0         0         0         0  -22.2222
Yes, I got a factorization
U is an echelon form of A
invL = 5×5
         0    1.0000         0         0         0
    1.0000   -0.7391         0         0         0
   -0.7687    0.0899         0         0    1.0000
    0.1443   -0.0336    1.0000         0   -0.5164
   -0.0613   -0.1111   -0.9231    1.0000   -0.2464
invU = 5×5
    0.0435   -0.0107   -0.0140   -0.0343    0.0012
         0    0.0493    0.0083    0.0071    0.0127
         0         0    0.0402    0.0059    0.0031
         0         0         0    0.0509    0.0435
         0         0         0         0   -0.0450
the inverse of A calculated using LU factorization is
   -0.0049    0.0512   -0.0354    0.0012    0.0034
    0.0431   -0.0373   -0.0046    0.0127    0.0015
   -0.0303    0.0031    0.0031    0.0031    0.0364
    0.0047   -0.0065    0.0108    0.0435   -0.0370
    0.0028    0.0050    0.0415   -0.0450    0.0111

Yes, LU factorization works for calculating the inverses
```

```
% (g)
A = hilb(5)
```

```
A = 5×5
    1.0000    0.5000    0.3333    0.2500    0.2000
    0.5000    0.3333    0.2500    0.2000    0.1667
    0.3333    0.2500    0.2000    0.1667    0.1429
    0.2500    0.2000    0.1667    0.1429    0.1250
    0.2000    0.1667    0.1429    0.1250    0.1111
```

```
[L, U] = eluinv(A);
```

```
L = 5×5
    1.0000         0         0         0         0
    0.5000    1.0000    1.0000         0         0
    0.3333    1.0000         0         0         0
    0.2500    0.9000   -0.6000    0.5000    1.0000
    0.2000    0.8000   -0.9143    1.0000         0
U = 5×5
    1.0000    0.5000    0.3333    0.2500    0.2000
         0    0.0833    0.0889    0.0833    0.0762
         0         0   -0.0056   -0.0083   -0.0095
         0         0         0    0.0007    0.0015
         0         0         0         0   -0.0000
Yes, I got a factorization
U is an echelon form of A
invL = 5×5
    1.0000         0         0         0         0
   -0.3333         0    1.0000         0         0
   -0.1667    1.0000   -1.0000         0         0
   -0.0857    0.9143   -1.7143         0    1.0000
   -0.0071    0.1429   -0.6429    1.0000   -0.5000
invU = 5×5
    0.0000   -0.0001   -0.0004   -0.0007   -0.0140
         0    0.0001    0.0019    0.0084    0.2688
         0         0   -0.0018   -0.0210   -1.1760
         0         0         0    0.0140    1.7920
         0         0         0         0   -0.8820
the inverse of A calculated using LU factorization is
   1.0e+05 *

    0.0002   -0.0030    0.0105   -0.0140    0.0063
   -0.0030    0.0480   -0.1890    0.2688   -0.1260
    0.0105   -0.1890    0.7938   -1.1760    0.5670
   -0.0140    0.2688   -1.1760    1.7920   -0.8820
    0.0063   -0.1260    0.5670   -0.8820    0.4410

LU factorization does not work for me!?
```

Part 2

```
type msystem.m
```

```
function [X1,X2,X] = msystem(A,B)
[~,n] = size(A);
[~,p] = size(B);
[L,U] = lu(A);

%first algorithm
X1 = inv(A) * B;
%second algorithm
X2 = A\B;
%third algorithm
y = rref([L B]);
Y = y(:,n+1:n+p);
x = rref([U Y]);
```

```
X = x(:,n+1:n+p);

if closetozeroroundoff(X1-X2,0) == 0
    if closetozeroroundoff(X1-X,0) == 0
    disp('The solutions calculated by different methods match')
    end
else
    disp('There is a problem with my code?!')
end


end
```

```
% (a)
A = [1 1 4; 0 -4 0; -5 -1 -8], B=magic(3)
```

```
A = 3×3
     1     1     4
     0    -4     0
    -5    -1    -8
B = 3×3
     8     1     6
     3     5     7
     4     9     2
```

```
[X1,X2,X] = msystem(A,B)
```

```
The solutions calculated by different methods match
X1 = 3×3
   -6.9167   -4.0833   -5.2500
   -0.7500   -1.2500   -1.7500
    3.9167    1.5833    3.2500
X2 = 3×3
   -6.9167   -4.0833   -5.2500
   -0.7500   -1.2500   -1.7500
    3.9167    1.5833    3.2500
X = 3×3
   -6.9167   -4.0833   -5.2500
   -0.7500   -1.2500   -1.7500
    3.9167    1.5833    3.2500
```

```
% (b)
A = magic(5), B = randi(10,5,4)
```

```
A = 5×5
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
B = 5×4
     8     6     5     4
     4     8     1     6
     6    10     4     2
     1     2     2     7
     1     6     8     3
```

```
[X1,X2,X] = msystem(A,B)
```

```
The solutions calculated by different methods match
X1 = 5×4
   -0.0426    0.0485   -0.0856    0.2347
    0.1824   -0.0515    0.1971    0.0328
```

```
   -0.1718    0.0985    0.1615    0.0344
    0.0824   -0.0515   -0.1490    0.1943
    0.2574    0.4485    0.1837   -0.1576
X2 = 5×4
   -0.0426    0.0485   -0.0856    0.2347
    0.1824   -0.0515    0.1971    0.0328
   -0.1718    0.0985    0.1615    0.0344
    0.0824   -0.0515   -0.1490    0.1943
    0.2574    0.4485    0.1837   -0.1576
X = 5×4
   -0.0426    0.0485   -0.0856    0.2347
    0.1824   -0.0515    0.1971    0.0328
   -0.1718    0.0985    0.1615    0.0344
    0.0824   -0.0515   -0.1490    0.1943
    0.2574    0.4485    0.1837   -0.1576
```

```
% (c)
A = magic(3), B = [magic(3), eye(3)]
```

```
A = 3×3
     8     1     6
     3     5     7
     4     9     2
B = 3×6
     8     1     6     1     0     0
     3     5     7     0     1     0
     4     9     2     0     0     1
```

```
[X1,X2,X] = msystem(A,B)
```

```
The solutions calculated by different methods match
X1 = 3×6
    1.0000         0   -0.0000    0.1472   -0.1444    0.0639
         0    1.0000         0   -0.0611    0.0222    0.1056
         0    0.0000    1.0000   -0.0194    0.1889   -0.1028
X2 = 3×6
    1.0000         0         0    0.1472   -0.1444    0.0639
         0    1.0000         0   -0.0611    0.0222    0.1056
         0         0    1.0000   -0.0194    0.1889   -0.1028
X = 3×6
    1.0000         0         0    0.1472   -0.1444    0.0639
         0    1.0000         0   -0.0611    0.0222    0.1056
         0         0    1.0000   -0.0194    0.1889   -0.1028
```

Exercise 6

```
type markov
```

```
function q = markov(P,x0)
format
n = size(P,1);
q=[];

%Determine stochastic
if sum(P,1) == ones(1,n)
    Q = null(P-eye(n), 'r');
    q = Q/sum(Q,1);
    counter = 0;
    x = x0;
    while closetozeroroundoff(q-x,7) ~= 0
        x = P * x0;
        x0 = x;
        counter = counter + 1;
```

```
    end
    fprintf("Number of iterations = %d", counter)
else
    disp('P is not a stochastic matrix')
    return
end

end
```

```
%(a)
P=[.6 .3;.5 .7], x0=[.4;.6]
```

```
P = 2×2
    0.6000    0.3000
    0.5000    0.7000
x0 = 2×1
    0.4000
    0.6000
```

```
q=markov(P,x0);
```

```
P is not a stochastic matrix
```

```
%(b)
P=[.5 .3;.5 .7],x0=[.5;.5]
```

```
P = 2×2
    0.5000    0.3000
    0.5000    0.7000
x0 = 2×1
    0.5000
    0.5000
```

```
q=markov(P,x0);
```

```
Number of iterations = 9
```

```
%(c)
P=[.9 .2;.1 .8], x0=[.11;.89]
```

```
P = 2×2
    0.9000    0.2000
    0.1000    0.8000
x0 = 2×1
    0.1100
    0.8900
```

```
q=markov(P,x0);
```

```
Number of iterations = 44
```

```
%(d)
P=[.9 .2;.1 .8], x0=[.90;.10]
```

```
P = 2×2
    0.9000    0.2000
    0.1000    0.8000
x0 = 2×1
    0.9000
    0.1000
```

```
q=markov(P,x0);
```

```
Number of iterations = 42
```

```
%(e)
P=[.90 .01 .09;.01 .90 .01;.09 .09 .90], x0=[.5; .3; .2]
```

```
P = 3×3
    0.9000    0.0100    0.0900
    0.0100    0.9000    0.0100
    0.0900    0.0900    0.9000
x0 = 3×1
    0.5000
    0.3000
    0.2000
```

```
q=markov(P,x0);
```

```
Number of iterations = 71
```

```
%(f)
P=magic(5); P=P.*1./sum(P), x0=randi(10,5,1);x0=x0.*1./sum(x0)
```

```
P = 5×5
    0.2615    0.3692    0.0154    0.1231    0.2308
    0.3538    0.0769    0.1077    0.2154    0.2462
    0.0615    0.0923    0.2000    0.3077    0.3385
    0.1538    0.1846    0.2923    0.3231    0.0462
    0.1692    0.2769    0.3846    0.0308    0.1385
x0 = 5×1
    0.2500
    0.2500
    0.2857
    0.1786
    0.0357
```

```
q=markov(P,x0);
```

```
Number of iterations = 11
```

```
%(g)
x0=q
```

```
x0 = 5×1
    0.2000
    0.2000
    0.2000
    0.2000
    0.2000
```

```
q=markov(P,x0);
```

```
Number of iterations = 0
```

Exercise 7

```
type transf_1.m
```

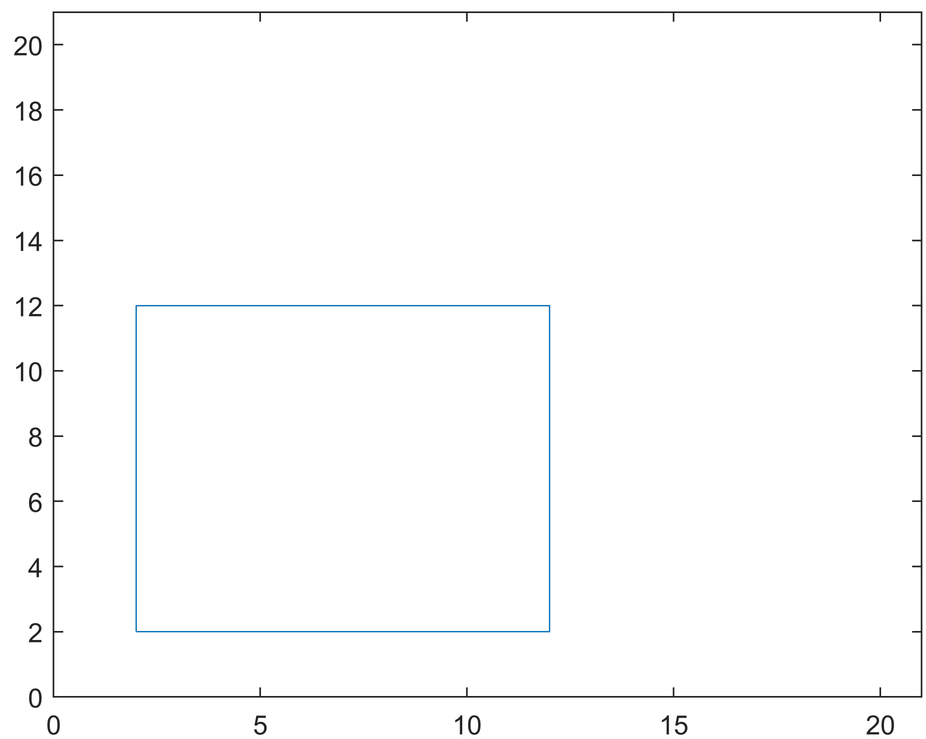
```
function C=transf_1(A,E)
C=A*E;
```

```
x=C(1,:);
y=C(2,:);
plot(x,y)
v=[0 21 0 21];
axis(v)
end
```

```
type polygon.m
```

```
function Area = polygon(E)
A=eye(2);
C=transf_1(A,E);
n=size(E,2)-1;
Area = 0;
for i=1:n
Area = Area + (E(1,i)*E(2,i+1) - E(2,i)*E(1,i+1));
end
Area= Area/2;
end
```

```
%(a)
E=[2 12 12 2 2;2 2 12 12 2];
Area = polygon(E)
```

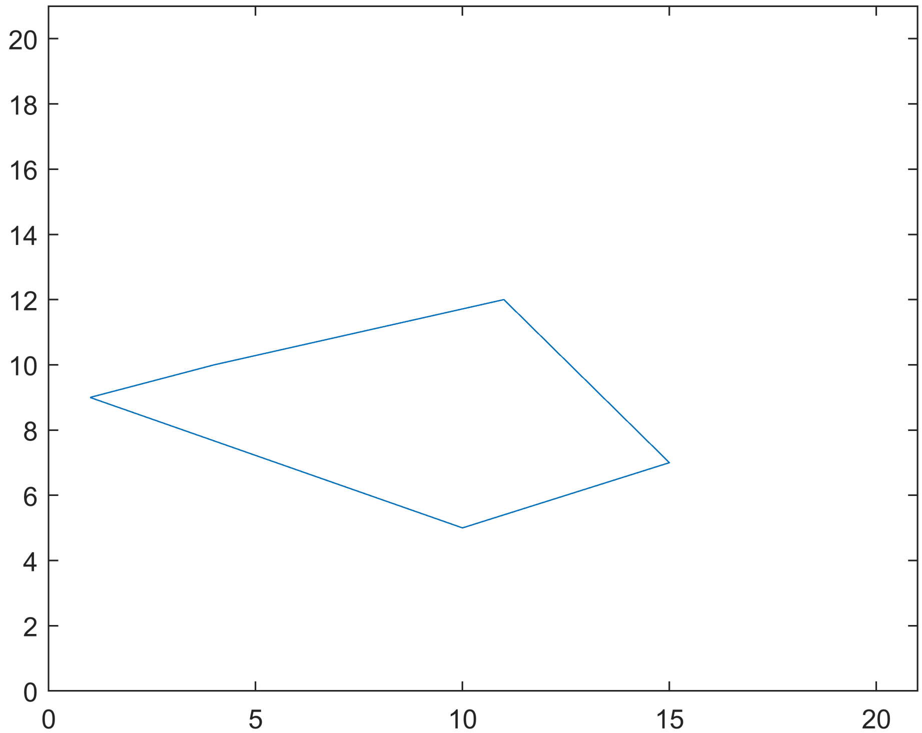



```
Area = 100
```

```
%(b)
E=[1 10 15 11 4 1;9 5 7 12 10 9]
```

```
E = 2×6
     1    10    15    11     4     1
     9     5     7    12    10     9
```

```
Area = polygon(E)
```

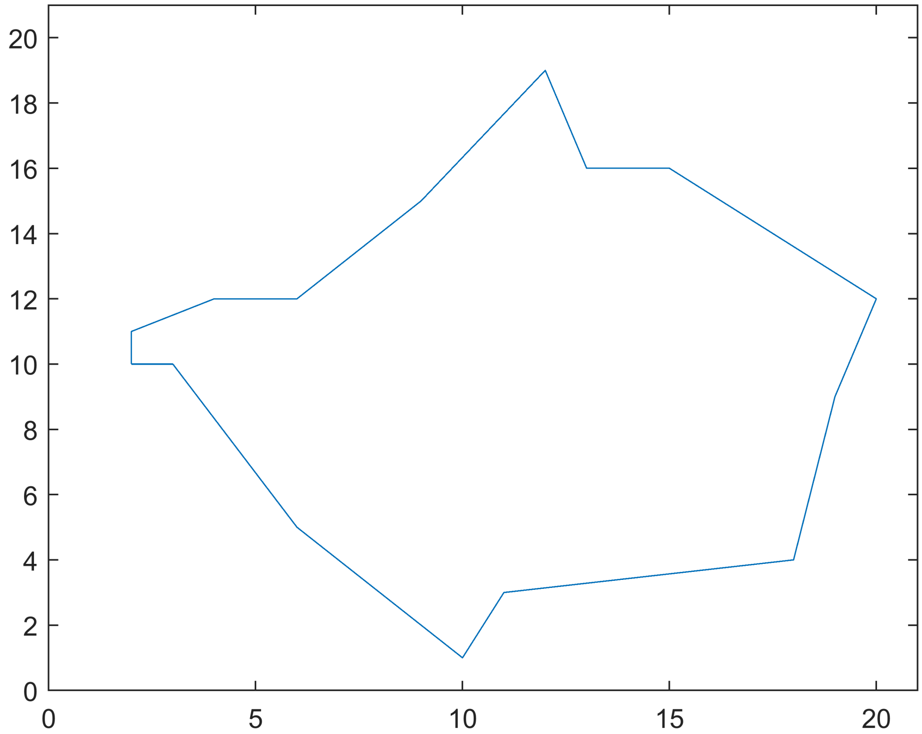



```
Area = 50.5000
```

```
%(c)
A1=randi(10,1,5);
A1=sort(unique(A1), 'ascend');
B1=randi(10,1,size(A1,2));
B1=sort(B1,'descend');
A2=randi([11 20],1,5);
A2=sort(unique(A2), 'ascend');
B2=randi(10,1,size(A2,2));
B2=sort(B2,'ascend');
A3=randi([11 20],1,5);
A3=sort(unique(A3), 'descend');
B3=randi([11 20],1,size(A3,2));
B3=sort(B3,'ascend');
A4=randi(10,1,5);
A4=sort(unique(A4), 'descend');
B4=randi([11 20],1,size(A4,2));
B4=sort(B4,'descend');
E=[A1 A2 A3 A4 A1(1,1);B1 B2 B3 B4 B1(1,1)]
```

```
E = 2×18
     2     3     6     9    10    11    18    19    20    15    13    12     9 ···
    10    10     5     2     1     3     4     9    12    16    16    19    15
```

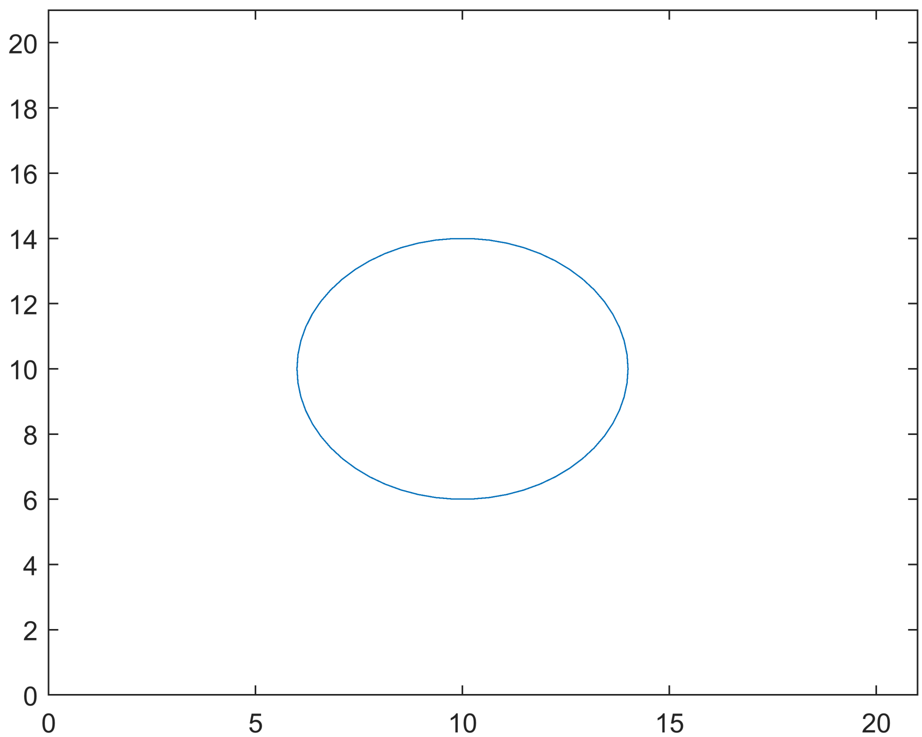
```
Area = polygon(E)
```




```
Area = 170
```

Bonus:

```
clear
R=4;
r0=[10;10];
n=3;
Area = 0;
ACircle = pi*R^2;
while (closetozeroroundoff(Area - ACircle,1)~=0)
   E=zeros(2,n+1);
   for i=1:n+1
       t= 2*pi*(i-1)/n;
       E(1,i)= r0(1,1)+ R*cos(t);
       E(2,i)= r0(2,1) + R*sin(t);
   end
   disp("");
   Area =polygon(E);
   n=n+1;
end
```




```
disp("(1) Approximation of polygon = " +Area);
```

(1) Approximation of polygon = 50.1672

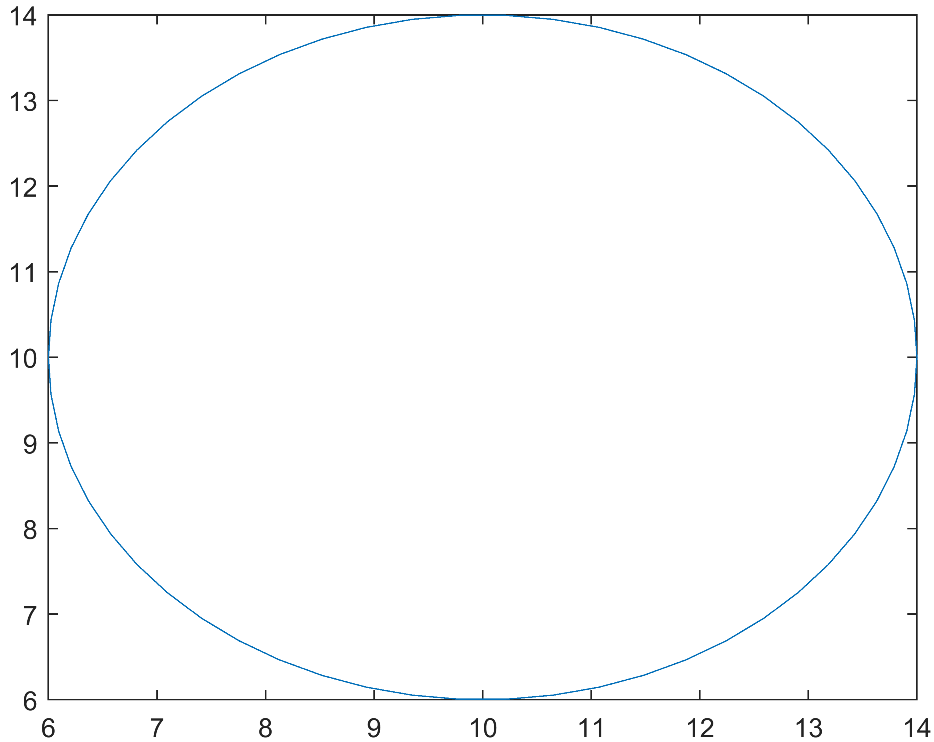
```
disp("(2) Minimum number of vertices needed = " +n);
```

(2) Minimum number of vertices needed = 59

```
x = E(1,:);
y = E(2,:);
disp("(3) Graph of inscribed polygon: ");
```

(3) Graph of inscribed polygon:

```
plot(x,y)
```


14
13
12
11
10
9
8
7
6
6
7
8
9
10
11
12
13
14