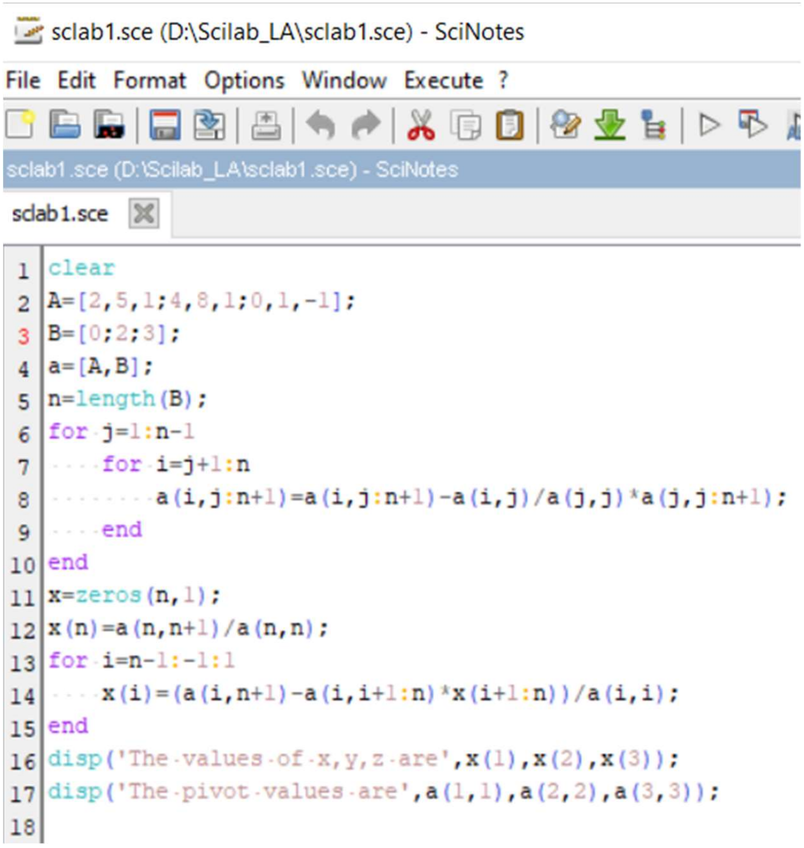
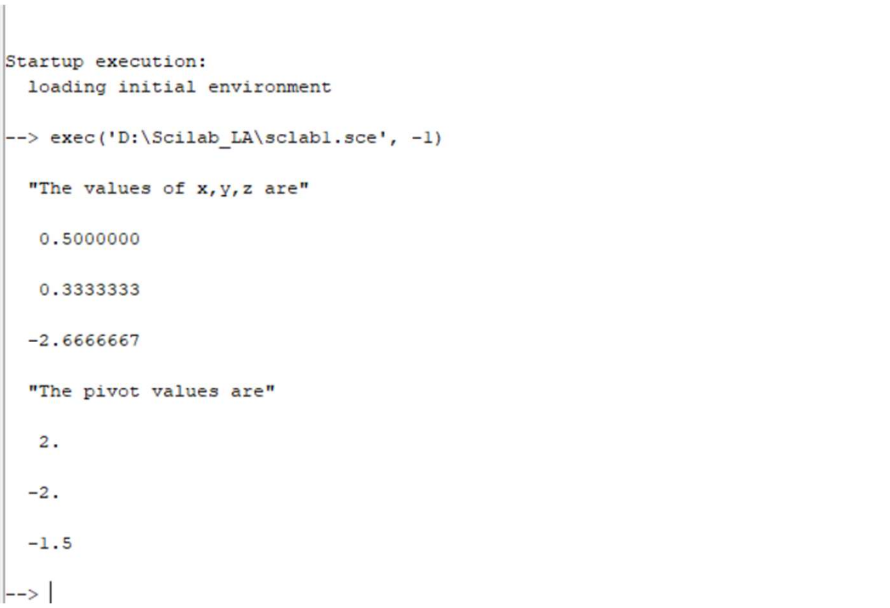
1) **Gaussian Elimination** - Solve the following system of equations by Gaussian Elimination. Identify the pivots in each case.

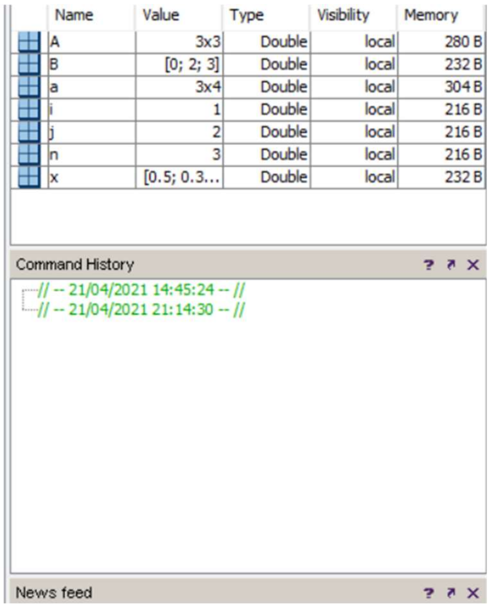
i) 2x+ 5y+z=0, 4x+ 8y+z=2,y–z =3




```
clear
A=[2,5,1;4,8,1;0,1,-1];
B=[0;2;3];
a=[A,B];
n=length(B);
for j=1:n-1
    for i=j+1:n
        a(i,j:n+1)=a(i,j:n+1)-a(i,j)/a(j,j)*a(j,j:n+1);
    end
end
x=zeros(n,1);
x(n)=a(n,n+1)/a(n,n);
for i=n-1:-1:1
    x(i)=(a(i,n+1)-a(i,i+1:n)*x(i+1:n))/a(i,i);
end
disp('The values of x,y,z are',x(1),x(2),x(3));
disp('The pivot values are',a(1,1),a(2,2),a(3,3));
```



```
Startup execution:
  loading initial environment

--> exec('D:\Scilab_LA\sclab1.sce', -1)

  "The values of x,y,z are"

   0.5000000

   0.3333333

  -2.6666667

  "The pivot values are"

   2.

  -2.

  -1.5

-->
```



Name	Value	Type	Visibility	Memory
A	3x3	Double	local	280 B
B	[0; 2; 3]	Double	local	232 B
a	3x4	Double	local	304 B
i	1	Double	local	216 B
j	2	Double	local	216 B
n	3	Double	local	216 B
x	[0.5; 0.3...	Double	local	232 B

ii) 2x+3y+z = 8,4x+ 7y+5z=20,-2y+2z =0

```
clear
A=[2,3,1;4,7,5;0,-2,2];
B=[8;20;0];
a=[A,B];
n=length(B);
for j=1:n-1
    for i=j+1:n
        a(i,j:n+1)=a(i,j:n+1)-a(i,j)/a(j,j)*a(j,j:n+1);
    end
end
x=zeros(n,1);
x(n)=a(n,n+1)/a(n,n);
for i=n-1:-1:1
    x(i)=(a(i,n+1)-a(i,i+1:n)*x(i+1:n))/a(i,i);
end
disp('The values of x,y,z are',x(1),x(2),x(3));
disp('The pivot values are',a(1,1),a(2,2),a(3,3));
```

```
--> exec('D:\Scilab_LA\sclab2.sce', -1)

  "The values of x,y,z are"

   2.

   1.

   1.

  "The pivot values are"

   2.

   1.

   8.

-->
```

2) **LU decomposition of a matrix** –

Factorize the following matrices as A = LU

(i) $A=\begin{pmatrix} 2 & 3 & 1 \\ 4 & 7 & 5 \\ 1 & -2 & 2 \end{pmatrix}$ (ii) $A=\begin{pmatrix} 2 & -3 & 0 \\ 4 & -5 & 1 \\ 2 & -1 & -3 \end{pmatrix}$

i)

scilab4.sce

```
clear;
A=[2,3,1;4,7,5;1,-2,2];
U=A;
disp("The given matrix A=",A);
m=det(U(1,1));
n=det(U(2,1));
a=n/m;
U(2,:)=U(2,:)-U(1,:)/(m/n);
n=det(U(3,1));
b=n/m;
U(3,:)=U(3,:)-U(1,:)/(m/n);
m=det(U(2,2));
n=det(U(3,2));
c=n/m;
U(3,:)=U(3,:)-U(2,:)/(m/n);
disp('The upper triangular matrix is U=',U);
L=[1,0,0;a,1,0;b,c,1];
disp('The lower triangular matrix is L=',L);
```

```
> exec('D:\Scilab_LA\scilab4.sce', -1)

"The given matrix A="

 2.   3.   1.
 4.   7.   5.
 1.  -2.   2.

"The upper triangular matrix is U="

 2.   3.   1.
 0.   1.   3.
 0.   0.   12.

"The lower triangular matrix is L="

 1.    0.    0.
 2.    1.    0.
 0.5  -3.5   1.
```

ii)

sclab5.sce

```
clear;
A=[2,-3,0;4,-5,1;2,-1,-3];
U=A;
disp("The given matrix A=",A);
m=det(U(1,1));
n=det(U(2,1));
a=n/m;
U(2,:)=U(2,:)-U(1,:)/(m/n);
n=det(U(3,1));
b=n/m;
U(3,:)=U(3,:)-U(1,:)/(m/n);
m=det(U(2,2));
n=det(U(3,2));
c=n/m;
U(3,:)=U(3,:)-U(2,:)/(m/n);
disp('The upper triangular matrix is U=',U);
L=[1,0,0;a,1,0;b,c,1];
disp('The lower triangular matrix is L=',L);
```

```
--> exec('D:\Scilab_LA\sclab5.sce', -1)

 "The given matrix A="

  2.  -3.   0.
  4.  -5.   1.
  2.  -1.  -3.

 "The upper triangular matrix is U="

  2.  -3.   0.
  0.   1.   1.
  0.   0.  -5.

 "The lower triangular matrix is L="

  1.   0.   0.
  2.   1.   0.
  1.   2.   1.
```

3) The Gauss-Jordan method of calculating A^{-1} – Find the inverse of the following matrices

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

i)

```
clear;
A=[1,0,0;1,1,1;0,0,1];
n=length(A(1,:));
aug=[A,eye(n,n)];
N=1:n;
for i=1:n
    dummy1=N;
    dummy1(i)=[];
    index(i,:)=dummy1;
end
for j=1:n
    [dummy2,t]=max(abs(aug(j:n,j)));
    lrow=t+j-1;
    aug([j,lrow],:)=aug([lrow,j],:);
    aug(j,:)=aug(j,:)/aug(j,j);
    for i=index(j,:)
        aug(i,:)=aug(i,:)-aug(i,j)/aug(j,j)*aug(j,:);
    end
end
inv_A=aug(:,n+1:2*n);
disp(inv_A)
```

```
--> exec('D:\Scilab_LA\sclab8.sce', -1)

   1.    0.    0.
  -1.    1.   -1.
   0.    0.    1.

-->
```

ii)

```
clear;
A=[2,-1,0;-1,2,-1;0,-1,2];
n=length(A(1,:));
aug=[A,eye(n,n)];
N=1:n;
for i=1:n
    dummy1=N;
    dummy1(i)=[];
    index(i,:)=dummy1;
end
for j=1:n
    [dummy2,t]=max(abs(aug(j:n,j)));
    lrow=t+j-1;
    aug([j,lrow],:)=aug([lrow,j],:);
    aug(j,:)=aug(j,:)/aug(j,j);
    for i=index(j,:)
        aug(i,:)=aug(i,:)-aug(i,j)/aug(j,j)*aug(j,:);
    end
end
inv_A=aug(:,n+1:2*n);
disp(inv_A)
```

```
--> exec('D:\Scilab_LA\sclab9.sce', -1)

  0.75   0.5   0.25
  0.5    1.    0.5
  0.25   0.5   0.75
```

iii)

```
clear;
A=[0,0,1;0,1,1;1,1,1];
n=length(A(1,:));
aug=[A,eye(n,n)];
N=1:n;
for i=1:n
    dummy1=N;
    dummy1(i)=[];
    index(i,:)=dummy1;
end
for j=1:n
    [dummy2,t]=max(abs(aug(j:n,j)));
    lrow=t+j-1;
    aug([j,lrow],:)=aug([lrow,j],:);
    aug(j,:)=aug(j,:)/aug(j,j);
    for i=index(j,:)
        aug(i,:)=aug(i,:)-aug(i,j)/aug(j,j)*aug(j,:);
    end
end
inv_A=aug(:,n+1:2*n);
disp(inv_A)
```

```
--> exec('D:\Scilab_LA\sclab10.sce', -1)

   0.  -1.   1.
  -1.   1.   0.
   1.   0.   0.
```

4) Span of the Column Space of A –

Identify the columns that span the column space of A in the following cases.

(1) $A=\begin{pmatrix} 1 & 3 & 3 & 2 \\ 2 & 6 & 9 & 7 \\ -1 & -3 & 3 & 4 \end{pmatrix}$ (2) $A=\begin{pmatrix} 2 & 4 & 6 & 4 \\ 2 & 5 & 7 & 6 \\ 2 & 3 & 5 & 2 \end{pmatrix}$

i)

```
clear;
disp('The given matrix A=')
a=[1,3,3,2;2,6,9,7;-1,-3,3,4]
a(2,:)=a(2,:)-(a(2,1)/a(1,1))*a(1,:);
a(3,:)=a(3,:)-(a(3,1)/a(1,1))*a(1,:);
disp(a)
a(3,:)=a(3,:)-(a(3,2)/a(2,2))*a(2,:);
disp(a)
a(1,:)=a(1,:)/a(1,1)
a(2,:)=a(2,:)/a(2,2)
disp(a)
for i=1:3
    for j=i:4
        if(a(i,j)<>0)
            disp('column',j,'is a pivot column');
            break
        end
    end
end
```

```
--> exec('D:\Scilab_LA\sclab11.sce', -1)

 "The given matrix A="

  1.   3.   3.   2.
  0.   0.   3.   3.
  0.   0.   6.   6.

  1.    3.    3.    2.
  0.    0.    3.    3.
  Nan   Nan   Nan   Nan

  1.    3.    3.    2.
  Nan   Nan   Inf   Inf
  Nan   Nan   Nan   Nan

 "column"

  1.

 "is a pivot column"

 "column"

  2.

 "is a pivot column"

 "column"

  3.

 "is a pivot column"
```

ii)

```
clear;
disp('The given matrix A=')
a=[2,4,6,4;2,5,7,6;2,3,5,2]
a(2,:)=a(2,:)-(a(2,1)/a(1,1))*a(1,:);
a(3,:)=a(3,:)-(a(3,1)/a(1,1))*a(1,:);
disp(a)
a(3,:)=a(3,:)-(a(3,2)/a(2,2))*a(2,:);
disp(a)
a(1,:)=a(1,:)/a(1,1)
a(2,:)=a(2,:)/a(2,2)
disp(a)
for i=1:3
    for j=i:4
        if(a(i,j)<>0)
            disp('column',j,'is a pivot column');
            break
        end
    end
end
```

```
--> exec('D:\Scilab_LA\sclab12.sce', -1)

 "The given matrix A="

  2.   4.   6.   4.
  0.   1.   1.   2.
  0.  -1.  -1.  -2.

  2.   4.   6.   4.
  0.   1.   1.   2.
  0.   0.   0.   0.

  1.   2.   3.   2.
  0.   1.   1.   2.
  0.   0.   0.   0.

 "column"

  1.

 "is a pivot column"

 "column"

  2.

 "is a pivot column"
```

5) The Four Fundamental Subspaces –

Find the four fundamental subspaces of

$$A=\begin{pmatrix} 1 & 2 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{pmatrix}$$

```
clear;
disp('A=')
a=[1,2,0,1;0,1,1,0;1,2,0,1];
[m,n]=size(a);
disp('m=',m);
disp('n=',n);
[v,pivot]=rref(a);
disp(rref(a));
disp(v);
r=length(pivot);
disp('rank=',r);
cs=a(:,pivot);
disp('Column Space=',cs);
ns=kernel(a);
disp('Null Space=',ns);
rs=v(1:r,1)';
disp('Row Space=',rs);
lns=kernel(a');
disp('Left null space=',lns);
```

```
--> exec('D:\Scilab_LA\sclab13.sce', -1)

 "A="

 "m="

  3.

 "n="

  4.

  1.   0.  -2.   1.
  0.   1.   1.   0.
  0.   0.   0.   0.

  1.   0.  -2.   1.
  0.   1.   1.   0.
  0.   0.   0.   0.

 "rank="

  2.

 "Column Space="

  1.   2.
  0.   1.
  1.   2.

 "Null Space="

  3.909D-17  -0.8660254
 -0.4082483   0.2886751
  0.4082483  -0.2886751
  0.8164966   0.2886751

 "Row Space="

  1.   0.

 "Left null space="

 -0.7071068
  1.106D-16
  0.7071068
```

6) Projections by Least Squares

Solve Ax = b by least squares where

$$A=\begin{pmatrix}1 & 0\\ 0 & 1\\ 1 & 1\end{pmatrix} \quad b=\begin{pmatrix}1\\ 1\\ 0\end{pmatrix}$$

```
clear;close;clc;
A=[1 0;0 1;1 1];
disp(A,'A=');
b=[1;1;0];
disp(b,'b=');
x=(A'*A)\(A'*b);
disp(x,'x=');
C=x(1,1);
D=x(2,1);
disp(C,'C=');
disp(D,'D=');
disp('The line of best fit is b=C+Dt');

```

```
  1.   0.
  0.   1.
  1.   1.

 "A="

  1.
  1.
  0.

 "b="

  0.3333333
  0.3333333

 "x="

  0.3333333

 "C="

  0.3333333

 "D="

 "The line of best fit is b=C+Dt"

-->
```

7) The Gram-Schmidt Orthogonalization - Apply the Gram – Schmidt process to the following set of vectors and find the orthogonal matrix: (1,1,0), (1, 0,1), (0,1,1)

sclab16.sce

```
clear;close;clc;
A=[1 1 0;1 0 1;0 1 1];
disp(A,'A=');
[m,n]=size(A);
for k=1:n
    V(:,k)=A(:,k);
    for j=1:k-1
        R(j,k)=V(:,j)'*A(:,k);
        V(:,k)=V(:,k)-R(j,k)*V(:,j);
    end
    R(k,k)=norm(V(:,k));
    V(:,k)=V(:,k)/R(k,k);
end
disp(V,'Q=');
```

```
 1.   1.   0.
 1.   0.   1.
 0.   1.   1.

"A="

 0.7071068   0.4082483  -0.5773503
 0.7071068  -0.4082483   0.5773503
 0.          0.8164966   0.5773503

"Q="

-->
```

8) Eigen values and Eigen vectors of a given square matrix – Find the Eigen values and the corresponding Eigen vectors of the following matrices.

$$(i)\ \begin{pmatrix} 8 & -6 & 2 \\ -6 & 7 & -4 \\ 2 & -4 & 3 \end{pmatrix}$$

```
clc;close;clear;
A=[8,-6,2;-6,7,-4;2,-4,3];
lam=poly(0,'lam');
lam=lam
charMat=A-lam*eye(3,3)
disp('The characteristic matrix is',charMat);
charPoly=poly(A,'lam');
disp('The characteristic polynomial is',charPoly);
lam=spec(A);
disp('The eigen values of A are',lam);
function[x,lam]=eigenvectors(A)
    [n,m]=size(A);
    lam=spec(A);
    x=[];
    for k=1:3
        B=A-lam(k)*eye(3,3);
        C=B(1:n-1,1:n-1);
        b=-B(1:n-1,n);
        y=C\b;
        y=[y;1];
        y=y/norm(y);
        x=[x y];
    end
endfunction
get ('eigenvectors')
[x,lam]=eigenvectors(A)
disp('The eigen vectors of A are',x);
```

```
"The characteristic matrix is"

8 -lam  -6       2
-6       7 -lam  -4
2       -4       3 -lam

"The characteristic polynomial is"

-7.128D-14 +45lam -18lam² +lam³

"The eigen values of A are"

 1.584D-15
 3.
 15.

"The eigen vectors of A are"

 0.3333333  -0.6666667   0.6666667
 0.6666667  -0.3333333  -0.6666667
 0.6666667   0.6666667   0.3333333
```

ii)

$$\begin{pmatrix} 2 & 2 & 1 \\ 1 & 3 & 1 \\ 1 & 2 & 2 \end{pmatrix}$$

```
clc;close;clear;
A=[2,2,1;1,3,1;1,2,2];
lam=poly(0,'lam');
lam=lam
charMat=A-lam*eye(3,3)
disp('The characteristic matrix is',charMat);
charPoly=poly(A,'lam');
disp('The characteristic polynomial is',charPoly);
lam=spec(A);
disp('The eigen values of A are',lam);
function[x,lam]=eigenvectors(A)
    [n,m]=size(A);
    lam=spec(A);
    x=[];
    for k=1:3
        B=A-lam(k)*eye(3,3);
        C=B(1:n-1,1:n-1);
        b=-B(1:n-1,n);
        y=C\b;
        y=[y;1];
        y=y/norm(y);
        x=[x y];
    end
endfunction
get ('eigenvectors')
[x,lam]=eigenvectors(A)
disp('The eigen vectors of A are',x);
```

```
"The characteristic matrix is"

2 -lam  2       1
1       3 -lam  1
1       2       2 -lam

"The characteristic polynomial is"

-5 +11lam -7lam² +lam³

"The eigen values of A are"

 1. + 0.i
 5. + 0.i
 1. + 0.i

"The eigen vectors of A are"

 0.           0.5773503   0.
-0.4472136    0.5773503  -0.4472136
 0.8944272    0.5773503   0.8944272
```