JASMINDER KAUR.(MAT/19/109).
ANUMEET KAUR.(MAT/19/111).

LINEAR ALGEBRA

TOPICS:

- *MATRICES.*
- *PERFORMING GAUSSIAN ELIMINATION.*
- *MATRIX OPERATION.*
- *MINORS AND COFACTORS.*
- *WORKING WITH LARGE MATRICES.*
- *SOLVING SYSTEM OF LINEAR EQUATIONS*
- *VECTOR SPACES.*
- *EIGEN VALUES AND EIGEN VECTORS.*

MATRICES

Matrices are denoted by capital letters, but in Mathematica we use lowercase letters, since capitals are reserved for built-in functions.

```
In[3]:= mat1 = {{2, 3, 4, 5, 6, 7}, {1, 1, 1, 1, 1, 1},
   {4, 5, 4, 5, 4, 5}, {11, 2, 2, 2, 2, 2}, {0, 0, 0, 0, 1}}
Out[3]= {{2, 3, 4, 5, 6, 7}, {1, 1, 1, 1, 1, 1}, {4, 5, 4, 5, 4, 5}, {11, 2, 2, 2, 2, 2}, {0, 0, 0, 0, 1}}
```

Here each row is enclosed in curly brackets with entries separately by commas, the rows are separately by commas,the entire matrix is enclosed in curly brackets.

The command **MatrixForm** will produce nicely formatted rectangular array with bracket-son the sides. **MatrixForm** command is used we want a nice look of the matrix:

```
In[1]:= mat2 = {{1, 2, 3, 4}, {4, 5, 6, 7}, {7, 8, 9, 10}}
Out[1]= {{1, 2, 3, 4}, {4, 5, 6, 7}, {7, 8, 9, 10}}
```

In[2]:=
```
mat2 // MatrixForm
```

Out[2]//MatrixForm=

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 5 & 6 & 7 \\ 7 & 8 & 9 & 10 \end{pmatrix}$$

$Post is a global variable whose value, if set, is a function that will be applied to every output generated in the current session. The simplest setting would be something like **$Post:=MatrixForm**, which would put every output cell into MatrixForm. The command **If[MatrixQ[#], MatrixForm[#], #]&** is an example of something called a pure function.

In[4]:=
```
$Post := If[MatrixQ[#], MatrixForm[#], #] &
```

In[6]:=
```
mat2
```

Out[6]//MatrixForm=

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 5 & 6 & 7 \\ 7 & 8 & 9 & 10 \end{pmatrix}$$

Dimensions command returns a list containing the number of rows and columns in the matrix.

In[11]:=
```
Dimensions[mat2]
```

Out[11]= {3, 4}

To get a 3 X 4 matrix with random integer entries we write:

In[12]:=
```
RandomInteger[40, {3, 4}]
```

Out[12]//MatrixForm=

$$\begin{pmatrix} 28 & 20 & 40 & 1 \\ 33 & 27 & 9 & 7 \\ 26 & 8 & 3 & 20 \end{pmatrix}$$

The Array command works much like the Table command but uses a function (either built-in or user-defined) rather than an expression to compute the entries. For a function f that takes two arguments, the command Array[f, {m, n}] gives the mln matrix whose i, j th entry is f (i, j). For example, using the built-in function Min for f produces a matrix where each entry is the minimum of the row number and column number of that entry's position:

In[13]:=
```
Array[Min, {4, 5}]
```

Out[13]//MatrixForm=

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 & 2 \\ 1 & 2 & 3 & 3 & 3 \\ 1 & 2 & 3 & 4 & 4 \end{pmatrix}$$

In[14]:=
```
Clear[f]
```

In[19]:=
```
f[i_, j_] := i^3 + j^2;
```

In[20]:=
```
Array[f, {2, 3}]
```
Out[20]//MatrixForm=

$$\begin{pmatrix} 2 & 5 & 10 \\ 9 & 12 & 17 \end{pmatrix}$$

We can use the Array command to produce a general 3X4 matrix whose i, jth entry (the entry in row i and column j) is a_ij.

In[23]:=
```
IdentityMatrix [4]
```
Out[23]//MatrixForm=

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

In[24]:=
```
DiagonalMatrix [{a, b, c, d}]
```
Out[24]//MatrixForm=

$$\begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & d \end{pmatrix}$$

PERFORMING GAUSSIAN ELIMINATION

A matrix is in reduced row echelon form if the first nonzero entry in each row is a 1 with only 0 above and beneath it. Furthermore, the rows must be arranged so that if one row begins with more 0s than another, then that row appears beneath the other. Any matrix can be put into reduced row echelon form by performing successive elementary row operations: multiplying a row by a nonzero constant, replacing a row by its sum with a multiple of another row, or interchanging two rows.

In[25]:=
```
mat = {{1, 1, 4, 25}, {2, 1, 0, 7}, {-3, 0, 1, -1}}
```
Out[25]//MatrixForm=

$$\begin{pmatrix} 1 & 1 & 4 & 25 \\ 2 & 1 & 0 & 7 \\ -3 & 0 & 1 & -1 \end{pmatrix}$$

In[26]:=
```
RowReduce [mat]
```
Out[26]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 5 \end{pmatrix}$$

MATRIX OPERATION

If two matrices have the same dimensions, we can compute their sum by adding the corresponding entries of the two matrices. In Mathematica, as in ordinary mathematical notation, we use the + operator for matrix sums:

In[28]:=
```
mat3 = {{1, 0, 0}, {2, 3, 4}, {-1, 5, -1}};
```

In[29]:=
```
mat4 = {{2, 2, 3}, {0, 0, 1}, {5, 5, 5}};
```

In[30]:=
```
mat3 + mat4
```

Out[30]//MatrixForm=

$$\begin{pmatrix} 3 & 2 & 3 \\ 2 & 3 & 5 \\ 4 & 10 & 4 \end{pmatrix}$$

In[31]:=
```
mat3 + mat4
```

Out[31]//MatrixForm=

$$\begin{pmatrix} 3 & 2 & 3 \\ 2 & 3 & 5 \\ 4 & 10 & 4 \end{pmatrix}$$

Scalar Multiplication:

In[32]:=
```
5 * mat3
```

Out[32]//MatrixForm=

$$\begin{pmatrix} 5 & 0 & 0 \\ 10 & 15 & 20 \\ -5 & 25 & -5 \end{pmatrix}$$

The i, jth entry of the product of the matrix a with the matrix b is the dot product of the row of a with the j th column of b. Multiplication is only possible if the number of columns of a is equal to the number of rows of b.

In[33]:=
```
mat3.mat4
```

Out[33]//MatrixForm=

$$\begin{pmatrix} 2 & 2 & 3 \\ 24 & 24 & 29 \\ -7 & -7 & -3 \end{pmatrix}$$

The symbol * will simply multiply corresponding entries in the two matrices.mat3*mat4

In[36]:=
```
mat3 * mat4
```

Out[36]//MatrixForm=

$$\begin{pmatrix} 2 & 0 & 0 \\ 0 & 0 & 4 \\ -5 & 25 & -5 \end{pmatrix}$$

The **Transpose** command will produce the transpose of a matrix , the matrix obtained by switching the rows and columns of that matrix:

In[37]:=
```
Transpose[mat3]
```

Out[37]//MatrixForm=

$$\begin{pmatrix} 1 & 2 & -1 \\ 0 & 3 & 5 \\ 0 & 4 & -1 \end{pmatrix}$$

To find a power of a matrix use the command **MatrixPower**. The first argument is the

matrix, and the second argument is the desired power:

In[38]:=
```
MatrixPower[mat3, 10]
```

Out[38]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 \\ 10\,249\,364 & 36\,166\,989 & 20\,498\,728 \\ 7\,834\,130 & 25\,623\,410 & 15\,668\,261 \end{pmatrix}$$

The **inverse** of a square matrix, if it exists, is the matrix whose product with the original matrix is the identity matrix. A matrix that has an inverse is said to be nonsingular. You can find the inverse of a nonsingular matrix with the Inverse command:

In[39]:=
```
Inverse[mat3]
```

Out[39]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 \\ \frac{2}{23} & \frac{1}{23} & \frac{4}{23} \\ -\frac{13}{23} & \frac{5}{23} & -\frac{3}{23} \end{pmatrix}$$

The **determinant** of a square matrix is a number that is nonzero if and only if the matrix is nonsingular.

In[40]:=
```
Det[mat3]
```

Out[40]= −23

The **trace** of a matrix is the sum of the entries along the main diagonal. The trace of a matrix may be calculated with the command **Tr**:

In[41]:=
```
Tr[mat4]
```

Out[41]= 7

In[47]:=
```
s4 = SparseArray[{{1, 1} → a, {2, 3} → b, {5, 2} → c}, {5, 5}, 2]
```

MINORS AND COFACTORS

If A is a square matrix then the minor Mij of entry aij is the determinant of the submatrix that remains after the ith row and jth column are deleted from A . M = (Mij) is the matrix of minors . But the command **Minors** will return a matrix whose ijth entry is the determinant of the submatrix that remains after the (n - i + 1)st row and (n - j + 1)st column are deleted from A .

In[1]:=
```
Clear[a];
mat5 = Array[a_ ## &, {3, 3}];
mat5 // MatrixForm
```

Out[3]//MatrixForm=

$$\begin{pmatrix} a\_ & 2\,a\_ & 3\,a\_ \\ 2\,a\_ & 4\,a\_ & 6\,a\_ \\ 3\,a\_ & 6\,a\_ & 9\,a\_ \end{pmatrix}$$

This is the matrix returned by the built in **Minors** command :

In[4]:=
```
Minors[mat5] // MatrixForm
```

Out[4]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

The custom command below will give us the traditional matrix of minors .

In[5]:=
```
minorsMatrix[m_List?MatrixQ] := Map[Reverse, Minors[m], {0, 1}]
```

In[6]:=
```
minorsMatrix[mat5] // MatrixForm
```

Out[6]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

To get a single minor say M23 we can simply ask for that entry from the output of the **minorsMatrix** command :

In[7]:=
```
minorsMatrix[mat5][[2, 3]]
```

Out[7]=
```
0
```

The matrix of cofactors is the matrix whose ijth entry is (-1)^(i+j) Mij . We can use **minorsMatrix** command to compute a matrix of cofactors .

In[8]:=
```
cofactorsMatrix[m_List?MatrixQ] :=
Table[(1)^(i + j), {i, Length[m]}, {j, Length[m]}] * minorsMatrix[m]
```

In[9]:=
```
cofactorsMatrix[mat5] // MatrixForm
```

Out[9]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Relationship between Inverse of a matrix and its adjoint :
A^(-1) = 1/det(A) * adj(A).

In[13]:=
```
Clear[mat];
mat = RandomInteger[9, {4, 4}];
mat // MatrixForm
```

Out[15]//MatrixForm=

$$\begin{pmatrix} 5 & 9 & 6 & 4 \\ 5 & 6 & 4 & 1 \\ 4 & 5 & 7 & 2 \\ 7 & 8 & 4 & 1 \end{pmatrix}$$

In[16]:=
```
(1 / Det[mat]) Transpose [cofactorsMatrix [mat]] // MatrixForm
```

Out[16]//MatrixForm=

$$\begin{pmatrix} -\frac{1}{10} & \frac{87}{20} & 1 & -\frac{11}{4} \\ -\frac{1}{10} & \frac{77}{20} & 1 & -\frac{9}{4} \\ -\frac{1}{10} & -\frac{23}{20} & 0 & \frac{3}{4} \\ -\frac{3}{10} & -\frac{99}{20} & -1 & \frac{11}{4} \end{pmatrix}$$

In[7]:=
```
%.mat // MatrixForm
```

Out[7]//MatrixForm=

$$\{\{0, 0, 0\}, \{0, 0, 0\}, \{0, 0, 0\}\}.\text{mat}$$

WORKING WITH LARGE MATRICES

A large matrix with only a few nonzero entries you can use the **SparseArray** command to enter, store, and work with the matrix efficiently. To create a **SparseArray** simply give the position and value for each nonzero entry of the matrix as follows:

In[1]:=
```
s1 = SparseArray [{{1, 1} → a, {2, 3} → b, {5, 2} → c, {6, 7} → d}]
```

Out[1]=

SparseArray []

In[2]:=
```
s1 // MatrixForm
```

Out[2]//MatrixForm=

$$\begin{pmatrix} a & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & b & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & c & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & d \end{pmatrix}$$

In[3]:=
```
s2 = SparseArray [{{1, 1}, {2, 3}, {5, 2}, {6, 7}} → {a, b, c, d}]
```

Out[3]=

SparseArray []

In[6]:=
```
s2 // MatrixForm
```

Out[6]//MatrixForm=

$$\begin{pmatrix} a & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & b & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & c & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & d \end{pmatrix}$$

```
In[7]:= s3 = SparseArray[{{1, 1}, {2, 3}, {5, 2}, {6, 7}} → {a, b, c, d}, {8, 10}]
```

```
Out[7]= SparseArray[ 
                     Specified elements : 4
                     Dimensions : {8, 10} ]
```

```
In[8]:= s3 // MatrixForm
```

```
Out[8]//MatrixForm=
```

$$\begin{pmatrix} a & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & b & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & c & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & d & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

```
In[9]:= s4 = SparseArray[{{1, 1} → a, {2, 3} → b, {5, 2} → c}, {5, 5}, 2]
```

```
Out[9]= SparseArray[ 
                     Specified elements : 3
                     Dimensions : {5, 5}
                     Default : 2 ]
```

```
In[10]:= s4 // MatrixForm
```

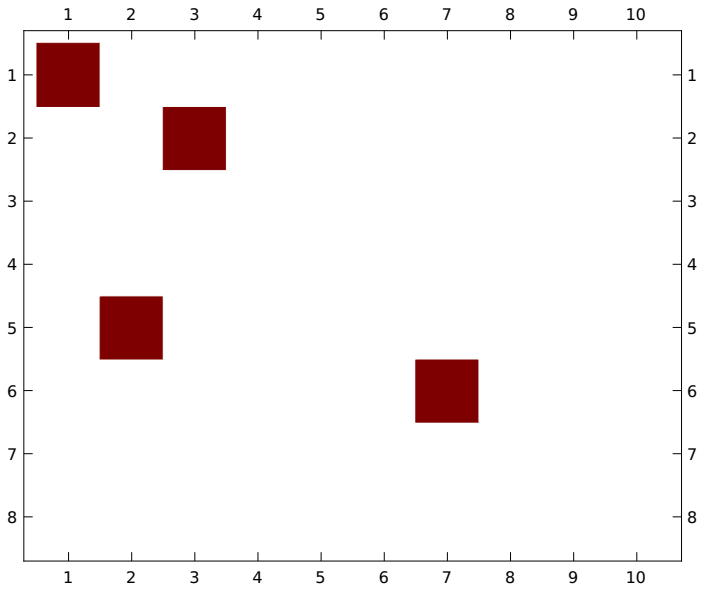
```
Out[10]//MatrixForm=
```

$$\begin{pmatrix} a & 2 & 2 & 2 & 2 \\ 2 & 2 & b & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & c & 2 & 2 & 2 \end{pmatrix}$$

Large matrix makes a mess if we ask for a numerical output, but a picture can tell us a lot about our matrix. The plots will automatically color entries with larger values in a dark color.

In[11]:=
```
MatrixPlot[s3]
```

Out[11]=



SOLVING SYSTEM OF LINEAR EQUATIONS

Nonhomogeneous Systems of Linear Equations:

To solve a system of linear equations in the form m x =b, where m is the coefficient matrix, x is a column vector of variables, and b is a column vector. Such a system is called **nonhomogeneous** when b is a vector with at least one nonzero entry. Mathematica offers several options for solving such a system, and we will explore each in turn. In this first example m is a nonsingular matrix and the system has a unique solution. Enter the equation m x= b by typing m.x==b. Note how Mathematica interprets this equation:

In[47]:=
```
m = {{1, 5, -4, 1}, {3, 4, -1, 2}, {3, 2, 1, 5}, {0, -6, 7, 1}};
x = {x1, x2, x3, x4};
```

In[53]:=
```
b = {{1}, {2}, {3}, {4}};
```

In[54]:=
```
m.x == b
```

Out[54]=
```
{x1 + 5 x2 - 4 x3 + x4, 3 x1 + 4 x2 - x3 + 2 x4, 3 x1 + 2 x2 + x3 + 5 x4, -6 x2 + 7 x3 + x4} ==
 {{1}, {2}, {3}, {4}}
```

We can interpret this as a list of four linear equations , each in four variables :
To check that **m** is nonsingular :

In[51]:=
```
Det[m]
```

Out[51]=
```
35
```

To form the augmented matrix we use the **ArrayFlatten** command and to find the row reduced echelon form of the matrix use **RowReduce** command.

In[55]:=
```
ArrayFlatten[{{m, b}}] // MatrixForm
```

Out[55]//MatrixForm=

$$\begin{pmatrix} 1 & 5 & -4 & 1 & 1 \\ 3 & 4 & -1 & 2 & 2 \\ 3 & 2 & 1 & 5 & 3 \\ 0 & -6 & 7 & 1 & 4 \end{pmatrix}$$

In[56]:=
```
RowReduce[%] // MatrixForm
```

Out[56]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & -\frac{127}{35} \\ 0 & 1 & 0 & 0 & \frac{141}{35} \\ 0 & 0 & 1 & 0 & \frac{139}{35} \\ 0 & 0 & 0 & 1 & \frac{13}{35} \end{pmatrix}$$

The command **LinearSolve** provides a quick means for solving systems that have a single solution :

In[57]:=
```
LinearSolve[m, b]
```

Out[57]=

$$\left\{\left\{-\frac{127}{35}\right\}, \left\{\frac{141}{35}\right\}, \left\{\frac{139}{35}\right\}, \left\{\frac{13}{35}\right\}\right\}$$

LinearSolve command also used to form a function for matrix **m** that can be applied to any vector **b**.

In[60]:=
```
Clear[f]
f = LinearSolve[m]
```

Out[61]=

LinearSolveFunction []

In[62]:=
```
f[b]
```

Out[62]=

$$\left\{\left\{-\frac{127}{35}\right\}, \left\{\frac{141}{35}\right\}, \left\{\frac{139}{35}\right\}, \left\{\frac{13}{35}\right\}\right\}$$

Or we can solve the system **m x = b** for **x** by multiplying both sides on the left by **m^(-1)** , to get **x = m(-1)b.**

In[63]:=
```
Inverse[m].b
```

Out[63]=

$$\left\{\left\{-\frac{127}{35}\right\}, \left\{\frac{141}{35}\right\}, \left\{\frac{139}{35}\right\}, \left\{\frac{13}{35}\right\}\right\}$$

Now use the command **Solve** to this system. When we use the Table/Matrix New... dialogue box to create **m** , **x** , and **b** both **mx** and **b** are lists of lists. The **Solve** command takes a list of equations as its first argument and a list of variables as its second argument -- it unfortunately cannot accept lists of lists . There is a simple solution . We will have to re-enter **x** and **b** without using the Table/Matrix New... dialogue box , expressing each as a single list . Then the equation **mx = b** is acceptable as input to the Solve command .

In[69]:=
```
Clear[x, b];
x = {x1, x2, x3, x4};
b = {1, 2, 3, 4};
m.x == b
```

Out[72]= $\{x1 + 5\,x2 - 4\,x3 + x4,\ 3\,x1 + 4\,x2 - x3 + 2\,x4,\ 3\,x1 + 2\,x2 + x3 + 5\,x4,\ -6\,x2 + 7\,x3 + x4\} == \{1, 2, 3, 4\}$

In[73]:=
```
Solve[m.x == b, x]
```

Out[73]= $\left\{\left\{x1 \to -\frac{127}{35},\ x2 \to \frac{141}{35},\ x3 \to \frac{139}{35},\ x4 \to \frac{13}{35}\right\}\right\}$

As inconsistent system of equations has no solutions. If we use **Solve** command on this type of system , the output will be an empty set of curly brackets :

In[74]:=
```
Clear[m, x, b]
```

In[77]:=
```
m = {{1, 1, 1}, {1, 1, 1}, {1, -1, -1}};
x = {x1, x2, x3};
b = {1, 2, -1};
Solve[m.x == b, x]
```

Out[80]= {}

If we row reduce , we can see the inconsistency in the system :

In[19]:=
```
ArrayFlatten[{{m, Transpose[{b}]}}]
```

Out[19]= {{1, 1, 1, 1}, {1, 1, 1, 2}, {1, -1, -1, -1}}

In[20]:=
```
RowReduce[%] // MatrixForm
```

Out[20]//MatrixForm=

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

The last row represents the impossible equation 0 = 1 .
If we use the **LinearSolve** command with an inconsistent system you will be told off :

In[81]:=
```
LinearSolve[m, b]
```

Out[81]= LinearSolve[{{1, 1, 1}, {1, 1, 1}, {1, -1, -1}}, {1, 2, -1}]

And if we find the inverse of **m** we will me told off again:

In[22]:=
```
Inverse[m].b
```

Out[22]= Inverse[{{1, 1, 1}, {1, 1, 1}, {1, -1, -1}}].{1, 2, -1}

The remaining possibility for a system of equations is that there are an infinite number of solutions .
The **Solve** command displays the solution set in this situation .

In[82]:=
```
Clear[m, x, b];
m = {{2, 3, -4}, {4, 6, -8}, {1, -1, -1}};
x = {x1, x2, x3};
b = {8, 16, 1};
Solve[m.x == b, x]
```

Out[86]=
$$\left\{\left\{x2 \to \frac{4}{7} + \frac{2\,x1}{7},\ x3 \to -\frac{11}{7} + \frac{5\,x1}{7}\right\}\right\}$$

In a system having an infinite number of solutions it will return only one of them , giving no warning that there are others . In this example it returns only the solution where x3 = 0 :

In[26]:=
```
LinearSolve [m, b]
```

Out[26]=
$$\left\{\frac{11}{5},\ \frac{6}{5},\ 0\right\}$$

Row reduction gives the solution with little possibility for confusion :

In[27]:=
```
ArrayFlatten [{{m, Transpose [{b}]}}]
```

Out[27]=
```
{{2, 3, -4, 8}, {4, 6, -8, 16}, {1, -1, -1, 1}}
```

In[28]:=
```
RowReduce [%] // MatrixForm
```

Out[28]//MatrixForm=
$$\begin{pmatrix} 1 & 0 & -\frac{7}{5} & \frac{11}{5} \\ 0 & 1 & -\frac{2}{5} & \frac{6}{5} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Thus for each value assumed by x3 , there is a solution with x1 = $\frac{11}{5} + \frac{7}{5}$ x3 , and x2 = $\frac{6}{5} + \frac{2}{5}$ x3.

In short we have to be very careful while using the command **LinearSolve** unless we know that we have a nonsingular matrix and have a single solution . to check this we can use the **Det** command keeping in mind that a singular matrix has determinant zero .

Homogeneous Systems of Equations :

A system of equations of the form **mx = 0** , where **m** is the coefficient matrix , **x** is a column vector of variables and **0** is the zero vector is called homogeneous . Note that **x = 0** is a solution to any homogeneous system . Now assume that **m** is a square matrix . The system of linear equation has a solution if and only if **m** is nonsingular . A homogeneous system will have only the trivial solution **x = 0** , while if **m** is singular the system will have an infinite number of solutions . The set of all solutions to a homogeneous system is called the null space of m :

In[93]:=
```
Clear[m, x, b];
m = {{0, 2, 2, 4}, {1, 0, -1, -3}, {2, 3, 1, 1}, {-2, 1, 3, -2}};
x = {x1, x2, x3, x4};
b = {{0}, {0}, {0}, {0}};
Det[m]
```

Out[97]= 0

In[98]:=
```
RowReduce [ArrayFlatten [{{m, b}}]] // MatrixForm
```

Out[98]//MatrixForm=

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

This reduced form of the augmented matrix tells us that x1 = x3 , x2 = -x3 and x4 = 0 . That is any vector of the form (t , -t , t , 0) where t is a real number is a solution and the vector (1 , -1 , 1 , 0) forms a basis for the solution space .

The command **NullSpace** gives a set of basis vector for the solution space of the homogeneous equation **mx = b** .

In[32]:=
```
NullSpace [m]
```

Out[32]= {{1, -1, 1, 0}}

USING LINEAR SOLVE AND NULLSPACE TO SOLVE THE NONHOMOGENEOUS SYSTEMS

We know that the **LinearSolve** command will only return one solution when a matrix equation **mx = b** has an infinite number of solutions . If we were to take the sum of the solution vector provided by **LinearSolve** with any vector in the null space of **m** , we would get another solution vector.

In[33]:=
```
Clear[m, b];
m = {{0, 2, 2, 4}, {1, 0, -1, -3}, {2, 3, 1, 1}}; b = {2, 0, 0};
LinearSolve [m, b]
```

Out[35]= {-9, 7, 0, -3}

In[36]:=
```
NullSpace [m]
```

Out[36]= {{1, -1, 1, 0}}

This show that there are number of infinite solutions . For each number of t , there is a solution (-9 , 7 , 0 , -3) + t (1 , -1 , 1 , 0) or x1 = -9 + t , x2 = 7-t , x3 = t and x4 = -3 .

In[99]:=
```
RowReduce [ArrayFlatten [{{m, b}}]] // MatrixForm
```

Out[99]//MatrixForm=

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

VECTOR SPACES

Span and Linear Independence :

Suppose we are given a set {**v1 , v2 , v3 , ..., vn** } of vectors . Any vector that can be expressed in the form **a1v1 + a2v2 + a3v3 +.....+ anvn** is said to be in the span of the vectors **v1 , v2 , v3, ...,vn** where the coefficients **ai** are scalars .

We can determine whether a given vector **b** is in the span of the vectors **v1 , v2 , v3, ...,vn** by letting **m** be the matrix whose columns are **v1 , v2 , v3, ...,vn** and then determining whether the equation **mx = b** has a solution . A solution **x** if exists provides values for the scalars **ai** .

For example , in real three space , is the vector **b** = (1,2,3) in the span of the vectors **v1** = (10,4,5) , **v2** = (4,4,7) , **v3** = (8,1,0) ?

In[1]:=
```
Clear[v1, v2, v, b, m, c];
v1 = {10, 4, 5};
v2 = {4, 4, 7};
v3 = {8, 1, 0};
b = {1, 2, 3};
m = Transpose [{v1, v2, v3}];
c = LinearSolve [m, b]
```

Out[7]= $\left\{\frac{3}{2}, -\frac{9}{14}, -\frac{10}{7}\right\}$

We can check that $\frac{3}{2}$ v1 - $\frac{9}{14}$ v2 - $\frac{10}{7}$ v3 = b.

In[8]:=
```
c[[1]] v1 + c[[2]] v2 + c[[3]] v3
```

Out[8]= {1, 2, 3}

A set of vectors {**v1 , v2 , v3 , ..., vn** } is said to be linearly independent if every vector in their span can be expressed in a unique way as a linear combination **a1v1 + a2v2 + a3v3 +.....+ anvn** . The only way to express the zero vector as such a linear combination is to have each coefficient **ai = 0** . If it is possible to write **a1v1 + a2v2 + a3v3 + ...+anvn = 0** with atleast one of the ai ≠ 0 then the set of vectors {**v1 , v2 , v3 , ... , vn** } is linearly dependent .

To check whether a set of vectors {**v1 , v2 , v3 , ... , vn** } is linearly independent , let **m** be the matrix whose columns are **v1 , v2, v3, ..., vn** and chcek that the equation **mx = 0** has the only trivial solution :

In[9]:=
```
NullSpace [m]
```

Out[9]= {}

Yes , these are linearly independent vectors . Alternatively , we could check that the matrix whose rows are **v1 , v2 , v3 , ... , vn** is nonsingular :

In[10]:=
```
Det[{v1, v2, v3}]
```

Out[10]= 14

Bases:

A basis for a vector space is a set of linearly independent vectors whose span includes every vector in the vector space. Given a spanning set of vectors **v1, v2, v3, ..., vn** for a vector space we can easily obtain a basis for that space. Form a matrix whose rows are the vectors **v1, v2, v3, ..., vn** and rowreduce:

In[8]:=
```
Clear[v1, v2, v3, v4, m, a, b, c];
v1 = {2, 1, 15, 10, 6};
v2 = {2, -5, -3, -2, 6};
v3 = {0, 5, 15, 10, 0};
v4 = {2, 6, 18, 8, 6};
m = {v1, v2, v3, v4};
RowReduce[m] // MatrixForm
```

Out[14]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & -2 & 3 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The nonzero rows of this matrix form a basis for the space spanned by the set **v1, v2, v3, v4**. This space is also called the row space of the matrix **m**.
We can also find a basis consisting of a subset of the original vectors. If we row-reduce the matrix whose columns are the vectors **v1, v2, v3, ..., vn**, then the columns containing the leading 1s will form a basis for the column space, and the corresponding columns from the original matrix will also form a basis for the column space.

In[29]:=
```
Clear[v1, v2, v3, v4, m, a, b, c];
v1 = {2, 1, 15, 10, 6};
v2 = {2, -5, -3, -2, 6};
v3 = {0, 5, 15, 10, 0};
v4 = {2, 6, 18, 8, 6};
m = Transpose[{v1, v2, v3, v4}];
RowReduce[m] // MatrixForm
```

Out[35]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & \frac{5}{6} & 0 \\ 0 & 1 & -\frac{5}{6} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

The vectors (1, 0, 0, 0, 0) , (0, 1, 0, 0, 0) , and (0, 0, 1, 0, 0) form a basis for the column space of **m** . The vectors from the same columns in m will also form a basis for the column space . So , **v1 , v2** and **v4** will form a basis for the space spanned by the set {**v1, v2, v3, v4**} .

In[38]:=
```
NullSpace[Transpose[{v1, v2, v4}]]
```

Out[38]= {}

Rank And Nullity

The dimension of the null space of a matrix is called the nullity of the matrix. We can find the nullity by using the **Length** command to count the vectors in a basis for the null space:

In[39]:=
```
Length[NullSpace[m]]
```

Out[39]= 1

The rank of a matrix is the common dimension of the row space and the column space. The rank plus the nullity must equal the number of columns in a matrix.

In[100]:=
```
MatrixRank[m]
```

Out[100]= 3

EIGEN VALUES AND EIGEN VECTORS

Finding Eigenvalues and Eigenvectors Automatically

Given an n×n matrix **m** , the non zero vectors v1 such that **mvi = λivi** are the eigenvectors of **m** and the scalars **λi** are the eigenvalues of **m** .There are at most n eigenvalues .

In[102]:=
```
Clear[m]
```

In[103]:=
```
m = Array[Min, {2, 2}];
m // MatrixForm
```

Out[104]//MatrixForm=

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

To get **eigenvalues**:

In[105]:=
```
{λ1, λ2} = Eigenvalues[m]
```

Out[105]= $\left\{\frac{1}{2}\left(3+\sqrt{5}\right), \frac{1}{2}\left(3-\sqrt{5}\right)\right\}$

To get **eigenvectors**:

In[106]:=
```
{λ1, λ2} = Eigenvectors[m]
```

Out[106]= $\left\{\left\{\frac{1}{2}\left(-1+\sqrt{5}\right), 1\right\}, \left\{\frac{1}{2}\left(-1-\sqrt{5}\right), 1\right\}\right\}$

The command **Eigensystem** gives both the eigenvalues and the eigenvectors. The output is a list whose first item is a list of eigenvalues and whose second item is a list of corresponding eigenvectors:

In[112]:=
```
Eigensystem[m]
```

Out[112]= $\left\{\left\{\frac{1}{2}\left(3+\sqrt{5}\right),\ \frac{1}{2}\left(3-\sqrt{5}\right)\right\},\ \left\{\left\{\frac{1}{2}\left(-1+\sqrt{5}\right),\ 1\right\},\ \left\{\frac{1}{2}\left(-1-\sqrt{5}\right),\ 1\right\}\right\}\right\}$

The output of any of these commands be numerical approximations by replacing m with **N[m]**:

In[113]:=
```
Eigensystem[N[m]]
```

Out[113]= {{2.61803 , 0.381966}, {{0.525731 , 0.850651}, {-0.850651 , 0.525731}}}

Even for a simple matrix with integer entries the eigenvalues can be quite complicated and involve complex numbers:

In[115]:=
```
Clear[m];
```

In[116]:=
```
m = Array[Min, {3, 3}];
```

In[117]:=
```
m // MatrixForm
```

Out[117]//MatrixForm=

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

In[118]:=
```
Eigenvalues[m]
```

Out[118]= {Root[5.05 …], Root[0.643 …], Root[0.308 …]}

The eigenvalues here are returned as **Root** objects . The option setting **Cubics -> True** will permit the display of such roots in terms of radicals .

In[119]:=
```
Eigenvalues[m, Cubics → True]
```

Out[119]=
$$\left\{2+\frac{7^{2/3}}{\left(\frac{3}{2}\left(9+i\sqrt{3}\right)\right)^{1/3}}+\frac{\left(\frac{7}{2}\left(9+i\sqrt{3}\right)\right)^{1/3}}{3^{2/3}},\ 2-\frac{7^{2/3}\left(1-i\sqrt{3}\right)}{2^{2/3}\left(3\left(9+i\sqrt{3}\right)\right)^{1/3}}-\frac{\left(1+i\sqrt{3}\right)\left(\frac{7}{2}\left(9+i\sqrt{3}\right)\right)^{1/3}}{2\times 3^{2/3}},\right.$$
$$\left.2-\frac{7^{2/3}\left(1+i\sqrt{3}\right)}{2^{2/3}\left(3\left(9+i\sqrt{3}\right)\right)^{1/3}}-\frac{\left(1-i\sqrt{3}\right)\left(\frac{7}{2}\left(9+i\sqrt{3}\right)\right)^{1/3}}{2\times 3^{2/3}}\right\}$$

In[120]:=
```
Eigenvalues[m] // N
```

Out[120]= {5.04892 , 0.643104 , 0.307979}

Finding Eigenvalues and Eigenvectors Manually

To find the eigenvalues we first form the characteristic polynomial, which is the determinant of the matrix I m, where m is a square
matrix, is an indeterminate, and I is the identity matrix of the same dimensions as m:

In[121]:=
```
Clear[m];
```

In[122]:=
```
m = {{2, -1, 0}, {-1, 2, 0}, {0, 0, 3}};
```

In[123]:=
```
c = Det[λ IdentityMatrix [3] - m]
```

Out[123]= $-9 + 15\,\lambda - 7\,\lambda^2 + \lambda^3$

Roots of characteristic polynomial:

In[124]:=
```
Solve[c == 0, λ]
```

Out[124]= $\{\{\lambda \to 1\}, \{\lambda \to 3\}, \{\lambda \to 3\}\}$

There are two eigenvalues λ=1 and λ=3. The eigenvalue 3 is reported twice because it occurs twice as a root of the characteristic polynomial c.

In[125]:=
```
Factor[c]
```

Out[125]= $(-3 + \lambda)^2\,(-1 + \lambda)$

In[126]:=
```
NullSpace[1 * IdentityMatrix [3] - m]
```

Out[126]= {{1, 1, 0}}

In[128]:=
```
NullSpace[3 * IdentityMatrix [3] - m]
```

Out[128]= {{0, 0, 1}, {-1, 1, 0}}

In[129]:=
```
Eigensystem [m]
```

Out[129]= {{3, 3, 1}, {{0, 0, 1}, {-1, 1, 0}, {1, 1, 0}}}