... we can use the ***matrix*** function.

Example 4.1. *Create a row vector* $u = [1, 2, 3]$

```
>>> from sympy import *
>>> Matrix([[1,2,3]])
Matrix([[1, 2, 3]])
```

Example 4.2. *Create a column vector* $v = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$

```
>>> from sympy import *
>>> Matrix([[1],[2],[3]])
Matrix([
[1],
[2],
[3]])
```

Example 4.3. *For vectors* $u = \begin{bmatrix} 2 \\ 5 \\ -3 \end{bmatrix}$ *and* $v = \begin{bmatrix} 1 \\ 0 \\ -2 \end{bmatrix}$ *find,*

a) $u + v$

b) $u - v$

c) $3.u$

d) $2.u + 3.v$

```
>>> from sympy import *
>>> u=Matrix([[2],[5],[-3]])
>>> v=Matrix([[1],[0],[-2]])
>>> u+v
Matrix([
[ 3],
[ 5],
[-5]])
>>> u-v
Matrix([
[ 1],
[ 5],
[-1]])
>>> 3*u
Matrix([
[ 6],
[15],
[-9]])
>>> 2*u+3*v
Matrix([
[  7],
[ 10],
[-12]])
```

more common are given below :

1. To create an identity matrix, use **eye**. eye(n) will create an $n \times n$ identity matrix.

```
>>> from sympy import *
>>> eye(3)
Matrix([
[1, 0, 0],
[0, 1, 0],
[0, 0, 1]])
>>> eye(4)
Matrix([
[1, 0, 0, 0],
[0, 1, 0, 0],
[0, 0, 1, 0],
[0, 0, 0, 1]])
```

2. To create a matrix of all zeros, use zeros. **zeros(n, m)** creates an $n \times m$ matrix of 0s.

```
>>> from sympy import *
>>> zeros(2,3)
Matrix([
[0, 0, 0],
```

```
>>> A*v
Matrix([
[ 29],
[ 65],
[101]])
>>> A**3
Matrix([
[ 468,  576,  684],
[1062, 1305, 1548],
[1656, 2034, 2412]])
```

Note: If we try v*A the it will gives error as Matrix size mismatch: (3,1) .* (3,3)

Inverse: Let A be any non singular matrix then there exits a matrix B such that

$$AB = BA = Identity\ matrix.$$

Matrix B is called inverse of matrix A. It is denoted by A^{-1}.

In python we can find inverse by **matrix.inv()** function.

For example

```
>>> from sympy import *
>>> # Consider matrices A and B,
>>> A=Matrix([[2,1,1],[1,2,1],[1,1,2]])
>>> B=Matrix([[1,1,1],[0,1,1],[0,0,1]])
>>> A.inv()
Matrix([
[ 3/4, -1/4, -1/4],
[-1/4,  3/4, -1/4],
[-1/4, -1/4,  3/4]])
>>> B.inv()
Matrix([
[1, -1,  0],
[0,  1, -1],
[0,  0,  1]])
```

4.3 Accessing Rows and Columns, Deleting and Inserting Rows and Columns

To get an individual row or column of a matrix, use **row** or **col**. For example, **M.row(0)** will get the first row. **M.col(-1)** will get the last column.

```
>>> from sympy import *
>>> # Consider matrices A and B,
>>> A=Matrix([[2,1,1],[1,2,1],[1,1,2]])
>>> B=Matrix([[1,1,1],[0,1,1],[0,0,1]])
>>> A.row(0)
Matrix([[2, 1, 1]])
>>> B.row(2)
Matrix([[0, 0, 1]])
>>> A.col(-1)
Matrix([
[1],
[1],
[2]])
```

We can add or delete particular row in matrix. To delete a row or column, use **row_del** or **col_del**. These operations will modify the Matrix in place.

```
>>> from sympy import *
>>> # Consider matrix A
>>> A=Matrix([[2,1,1],[1,2,1],[1,1,2]])
>>> A.row_del(1)
>>> A
Matrix([
[2, 1, 1],
[1, 1, 2]])
>>> A.col_del(-1)
>>> A
Matrix([
[2, 1],
[1, 1]])
```

To insert rows or columns, use **row_insert** or **col_insert**. These operations do not operate in place.

```
>>> from sympy import *
>>> # Consider matrix A
>>> A=Matrix([[2,1,1],[1,2,1],[1,1,2]])
>>> A=A.row_insert(1, Matrix([[0,5, 4]]))
>>> A
Matrix([
[2, 1, 1],
[0, 5, 4],
[1, 2, 1],
[1, 1, 2]])
>>> A=A.col_insert(0, Matrix([[0],[5],[6],[4]]))
>>> A
Matrix([
[0, 2, 1, 1],
[5, 0, 5, 4],
[6, 1, 2, 1],
[4, 1, 1, 2]])
```

following examples:

Example 4.4. *Solve the following system of equations:*

$$3x + 2y - z = 3 \quad 2x - 2y + 4z = 6 \quad 2x - y + 2z = 9$$

```
>>> from sympy import *
>>> x, y, z = symbols("x, y, z")
>>> A = Matrix([[3, 2, -1], [2, -2, 4], [2, -1, 2]])
>>> b = Matrix([3, 6, 9])
>>> linsolve((A, b), [x, y, z])
FiniteSet((6, -11, -7))
```

Example 4.5. *Solve the following system of equations:*

$$7x + 6y - 8z = 3 \quad 7x - 2y + 2z = 0 \quad 6x - y - 2z = 9$$

```
>>> from sympy import *
>>> x, y, z = symbols("x, y, z")
>>> A = Matrix([[7, 6, -8], [7, -2,2 ], [6, -1,-2]])
```

```
>>> b = Matrix([3, 0, 9])
>>> linsolve((A, b), [x, y, z])
FiniteSet((-9/79, -276/79, -489/158))
```

Example 4.6. *Solve the following system of equations:*

$$x + 2y + 3z = 3 \quad 4x + 5y + 6z = 6 \quad 7x + 8y + 9z = 9$$

```
>>> from sympy import *
>>> x, y, z = symbols("x, y, z")
>>> A = Matrix([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
>>> b = Matrix([3, 6, 9])
>>> linsolve((A, b), [x, y, z])
FiniteSet((z - 1, 2 - 2*z, z))
```

4.5.2 Gauss Jordan Method

Here are some examples:

Example 4.7. *Solve the following system:*

$$2x + y = 5; \quad x + 2y = 7$$

```
>>> from sympy import *
>>> A = Matrix([[2, 1], [1, 2]])
>>> B = Matrix([5, 7])
>>> A.gauss_jordan_solve(B)
(Matrix([
[1],
[3]]), Matrix(0, 1, []))
```

Example 4.8. *Solve the following system:*

$$x + 2y + 3z = 3; \quad 4x + 5y + 6z = 6 \quad 7x + 8$$

```
>>> from sympy import *
>>> A = Matrix([[1, 2, 3], [4, 5, 6], [7, 8, 10]])
>>> B = Matrix([3, 6, 9])
>>> sol, params = A.gauss_jordan_solve(B)
>>> sol
Matrix([
[-1],
[ 2],
[ 0]])
>>> params
Matrix(0, 1, [])
```

4.5.3 LU- decomposition Method

... to solve system $AX = B$ usinf L

given in of **syms**. Cosider the following examples:

Example 4.9. *Solve the following sysytem using LU Decomposition method.*

$$\begin{pmatrix} 6 & 18 & 3 \\ 2 & 12 & 1 \\ 4 & 15 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 3 \\ 19 \\ 0 \end{pmatrix}$$

```
>>> # First we import sympy.
>>> from sympy import *
>>> # We define variables £ x,y,z. £
>>> from sympy.abc import x, y, z
>>> # We need to give agumented matrix AB .
>>> AB = Matrix([[6,18, 3,3], [2, 12, 1,19], [4, 15, 1,0]])
>>> solve_linear_system_LU(AB,[x,y,z])
{x: -14, y: 3, z: 11}
```

Example 4.10. *Solve the following sysytem using LU Decomposition method.*

$$\begin{pmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 7 \\ 8 \\ 6 \end{pmatrix}$$

```
>>> # First we import sympy.
>>> from sympy import *
>>> # We define variables £ x,y,z. £
>>> from sympy.abc import x, y, z
>>> # We need to give agumented matrix AB .
>>> AB = Matrix([[1,2, 2,7], [2, 1, 2,8], [2, 2, 1,6]])
>>> solve_linear_system_LU(AB,[x,y,z])
{x: 7/5, y: 2/5, z: 12/5}
```