HKBK COLLEGE OF ENGINEERING

LAB MANUAL

MATHEMATICS I FOR CSE STREAM (BMATS101)

COMPLIED BY

DR. D. UMADEVI, PROF. SNEHA

Department of Engineering Mathematics,

HKBK College of Engineering, Bengaluru.

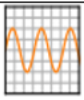
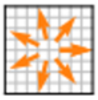
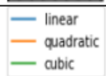
Department of Engineering Mathematics
List of Faculties in the Department

1.	Dr. C.S NAGABHUSHANA	Professor & HOD
2.	Prof. UMME SALMA	Assistant Professor
3.	Dr. D. UMADEVI	Associate Professor
4.	Prof. SHARMADA.U	Assistant Professor
5.	Prof. SNEHA.S	Assistant Professor
6.	Prof. LAKSHMI.S	Assistant Professor
7.	Prof. AZRA BEGUM	Assistant Professor
8.	Prof. ROOPASHREE	Assistant Professor
9.	Prof. ARIF ALI	Assistant Professor
10.	Prof. ISHRATH	Assistant Professor
11.	Prof. NARESH	Assistant Professor
12.	Prof. ARADHANA C.K.	Assistant Professor
13.	Prof. RASHMI	Assistant Professor
14.	Prof. JAGADEESH	Assistant Professor

Programs for Mathematics I CSE Stream Lab

Table of Contents

LAB EXPERIMENT 1: 2D-Plots of Cartesian and Polar Curves

Functions with syntax	Description
plot (*x*, *y*)	plots y versus x as lines /markers 
scatter (*x*, *y*)	scatter plot of y versus x 
quiver ([*x*, *y*], *u*, *v*, [*c*])	plots the vectors where [*x*, *y*] define the arrow locations, *u*, *v* define the arrow directions, and *c* optionally sets the color. 
pie (*data*)	draws a pie chart where *data* represents the array of data values to be plotted 
legend ( )	places a legend that describes the elements of the graph. 
xlabel ('*x-title*')	creates the label for x-axis
ylabel ('*y-title*')	creates the label for y-axis
title ('*title name*')	creates a title for the plotting
show ( )	used to display all figures
grid ( )	creates gridlines in the graph
polar (*theta, r, 'c'*)	traces the polar curve for the polar coordinates (*theta, r*) in '*c*' color.

The above functions are from matplotlib.pyplot library

Functions with syntax	Description
arange (*start, stop, step*)	returns an array that begins with the *start* value and evenly spaced elements of *step* size as per the interval. The interval mentioned is half-opened i.e. [*start, stop*)
linspace (*start, stop, num=50*)	returns an array of **evenly spaced values** within the specified interval [*start, stop*]. It is similar to **arange( )** function but instead of a *step*, it uses a sample number of 50.

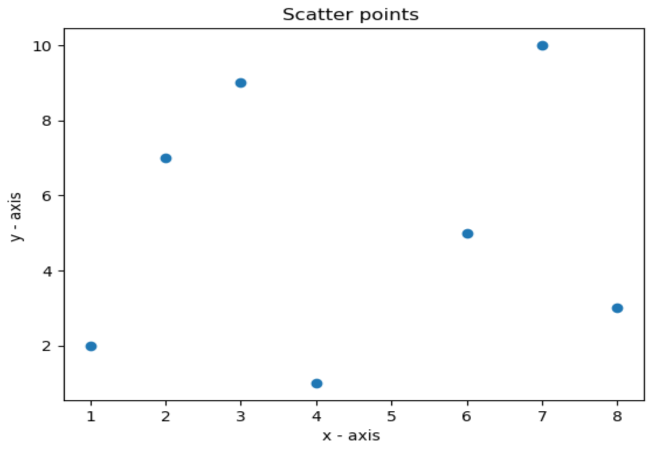
The above functions are from the numpy library

1.1. Plotting Of Cartesian Curves

1. Write a code for Plotting points (Scattered plot) (1,2), (2,7), (3,9), (4,1), (6,5), (7,10), (8,3).

```
from matplotlib.pyplot import *
x = [1, 2, 3, 4, 6, 7, 8]
y = [2, 7, 9, 1, 5, 10, 3]
scatter(x, y) # plotting the points
xlabel('x - axis') # naming the x axis
ylabel ('y - axis') # naming the y axis
title ('Scatter points ') # giving a title to my graph
show()
```

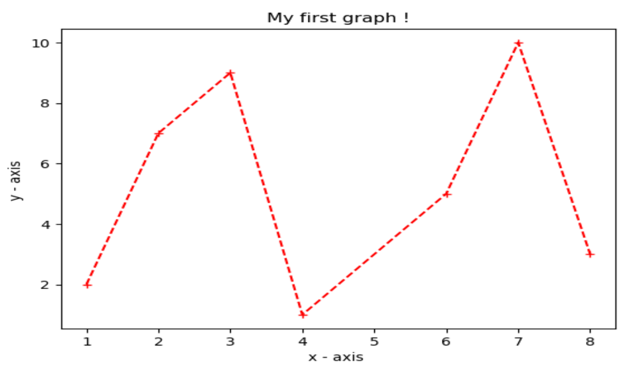
Output:



2. Write a code to plot a line (Line plot) passing through the points (1, 2), (2, 7), (3, 9), (4, 1), (6, 5), (7, 10), (8, 3).

```
from matplotlib.pyplot import *
x = [1, 2, 3, 4, 6, 7, 8]
y = [2, 7, 9, 1, 5, 10, 3]
plot(x , y , 'r+--') # plotting the points
xlabel('x - axis ') # naming the x axis
ylabel('y - axis ') # naming the y axis
title('My first graph !') # giving a title to my graph
show ()
```

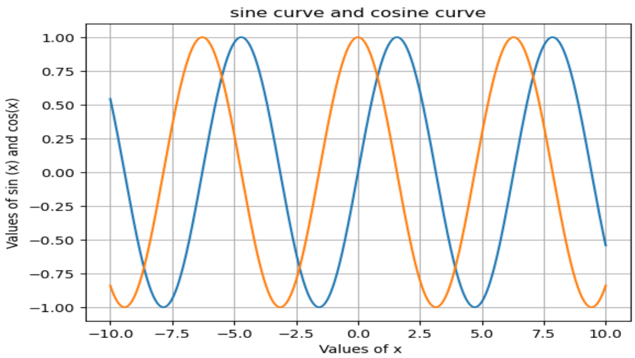
Output:



3. Write a code for plotting Sine and Cosine curves.

```
from numpy import *
from matplotlib. pyplot import *
x= arange(-10, 10, 0.001)
y1=sin(x)
y2=cos(x)
plot(x, y1, x, y2)
title("sine curve and cosine curve")
xlabel("Values of x")
ylabel("Values of sin (x) and cos(x)")
grid()
show()
```

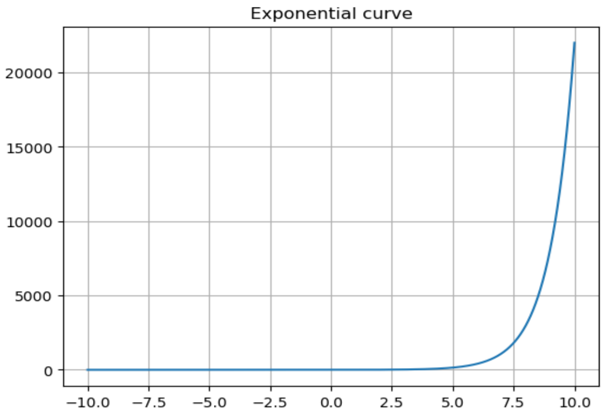
Output:



4. Write a code for plotting Exponential curve

```
from numpy import *
from matplotlib.pyplot import *
x = arange(-10, 10, 0.001)
y = exp(x)
plot(x, y)
title("Exponential curve")
grid()
show()
```

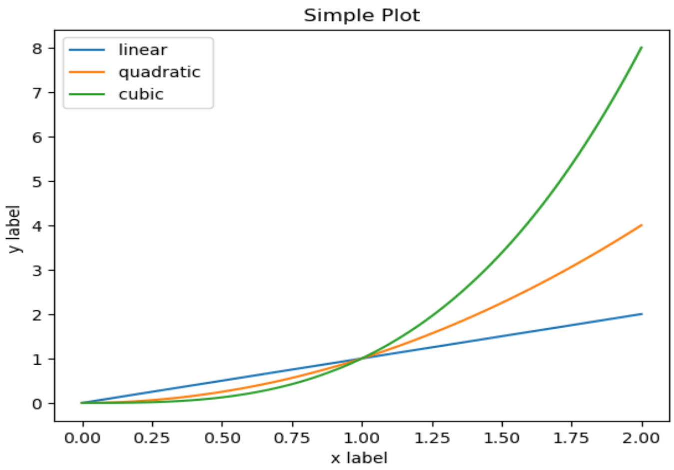
Output:



5. Write a code for plotting linear, quadratic and cubic curves

```
from matplotlib.pyplot import *
from numpy import *
x = linspace(0, 2, 100)
plot(x, x, label='linear')
plot(x, x**2, label='quadratic')
plot(x, x**3, label='cubic')
xlabel('x label')
ylabel('y label')
title("Simple Plot")
legend()
show()
```

Output:



1.2. Implicit Functions

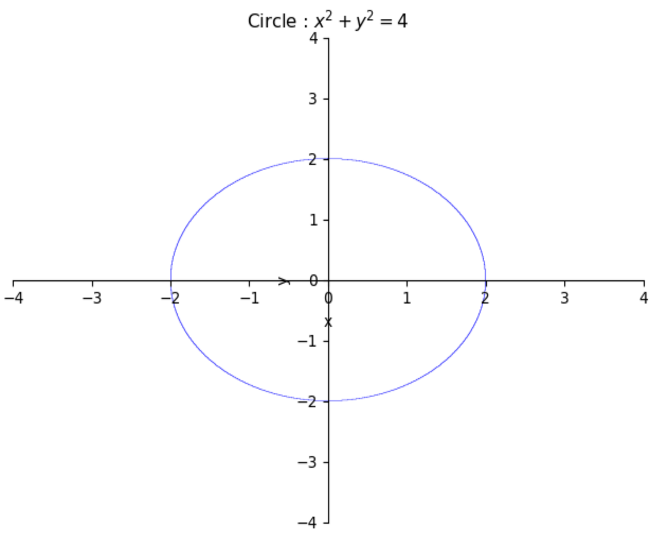
Functions with Syntax	Description
plot_implicit (*expr, x_var, y_var, title=`title name'*)	plots the equations or inequalities (*expr*=0), with symbol *x_var* to plot on x-axis or tuple giving symbol and range as (*x_var*, xmin, xmax) and symbol *y_var* to plot on y-axis or tuple giving symbol and range as (y_*var*, ymin, ymax) with 'title name'
Eq (*LHS, RHS*)	sets up an equation LHS = RHS

The above functions are from the sympy library

1. Write a Code to plot the equation of the circle $x^2 + y^2 = 4$

```
from sympy import *
x, y = symbols('x y')
p1 = plot_implicit(Eq(x**2+y**2,4), (x,-4,4), (y,-4,4), title = 'Circle:
                                                           $x^2+y^2=4$')
```

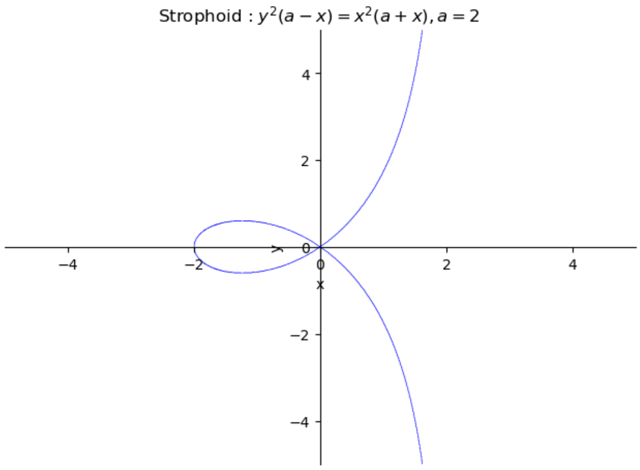
Output:



2. Write a Code to plot the equation of the Strophoid $y^2(a-x) = x^2(a+x)$ take *a = 2*

```
from sympy import *
x, y = symbols('x y')
p2 = plot_implicit(Eq((y**2)*(2-x), (x**2)*(2+x)), (x, -5, 5), (y, -5, 5),
                                                title='Strophoid: $y^2(a-x)=x^2(a+x), a=2$')
```

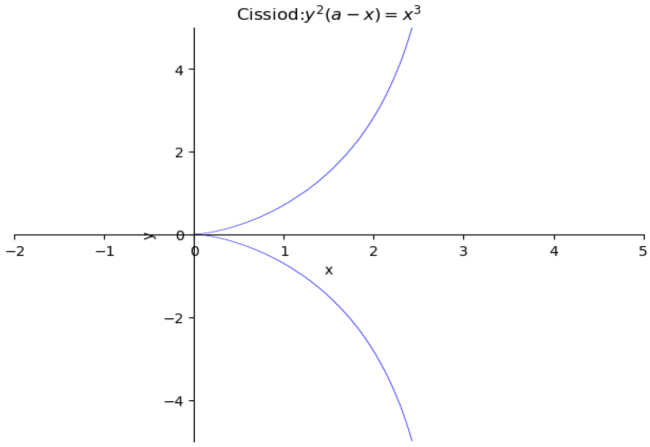
Output:



3. Write a code to plot the equation of the Cissiod: $y^2(a-x) = x^3$ take *a = 3*

```
from sympy import *
x, y = symbols('x y')
p3 = plot_implicit(Eq((y**2)*(3-x), x**3), (x,-2,5), (y,-5,5), title
                                                            = ‘Cissiod: $y^2(a-x)=x^3$')
```

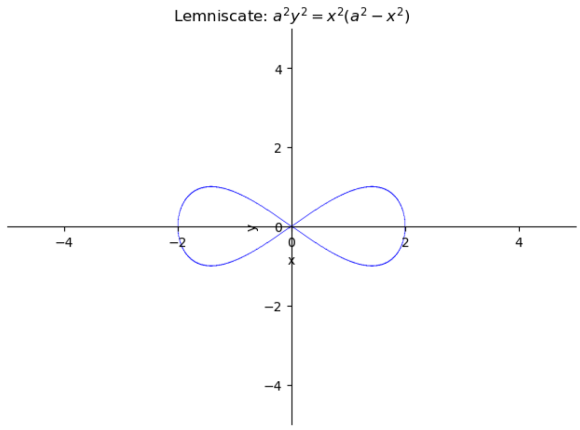
Output:



4. Write a code to plot the equation of Lemniscate: $a^2y^2 = x^2(a^2 - x^2)$ take *a = 2*

```
from sympy import *
x, y = symbols ('x y')
p4 = plot_implicit(Eq(4*(y**2), (x**2)*(4-x**2)), (x,-5,5), (y,-5,5),
                                                 title ='Lemniscate: $a^2y^2=x^2(a^2-x^2)$')
```

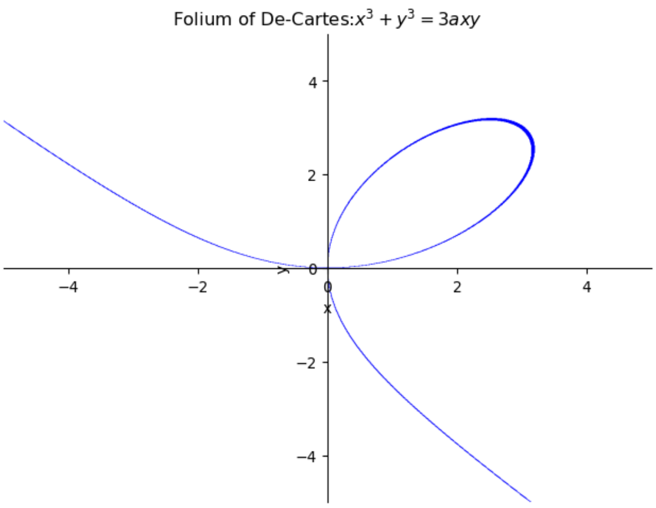
Output:



5. Write a code to plot the equation of Folium of De-Cartes: $x^3 + y^3 = 3axy$, take a=2

```
from sympy import *
x, y = symbols('x y')
p5 = plot_implicit(Eq(x**3+y**3, 3*2*x*y), (x,-5,5), (y,-5,5), title=
                                        'Folium of De-Cartes:$x^3+y^3=3axy$')
```

Output:



1.3. Polar Curves

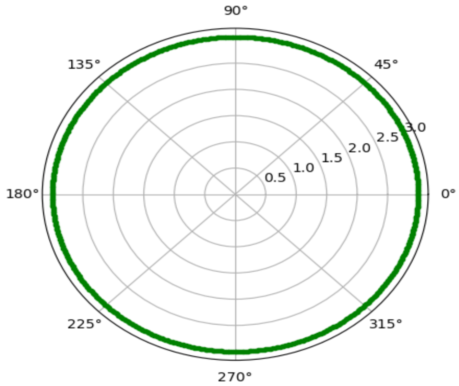
PYLAB

- **pylab** is a historic interface. The equivalent replacement is **matplotlib.pyplot**.
- ‘from pylab import *’ imports all the functions from matplotlib.pyplot, numpy, numpy.fft, numpy.linalg, and numpy.random, and some additional functions into the global namespace.

1. Write a code to plot a curve of circle in polar form take $r = 3$

```
from pylab import *
axes(projection ='polar')
r = 3
rads = arange(0, (2*pi), 0.01)
for i in rads:
    polar(i,r,'g.')
show()
```

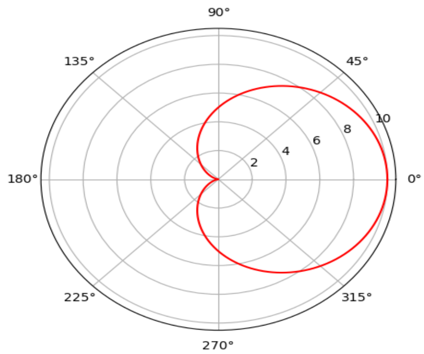
Output:



2. Write a code to plot a Cardioid: r = 5(1 + cos θ)

```
from pylab import *
theta = linspace(0, 2*pi, 1000)
r1=5+5*cos(theta)
polar(theta, r1,'r')
show()
```

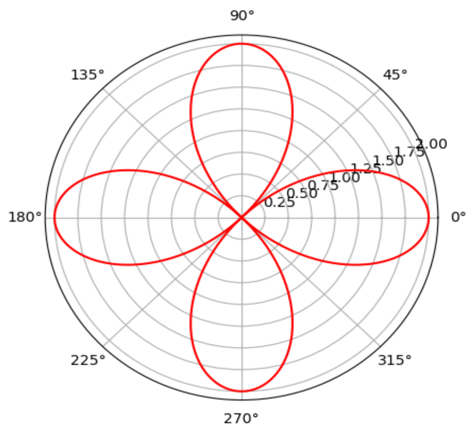
Output:



3. Write a code to plot a four leaved Rose: $r = 2|\cos 2x|$

```
from pylab import *
theta = linspace(0, 2*pi, 1000)
r = 2*abs(cos(2*theta))
polar(theta, r, 'r')
show()
```

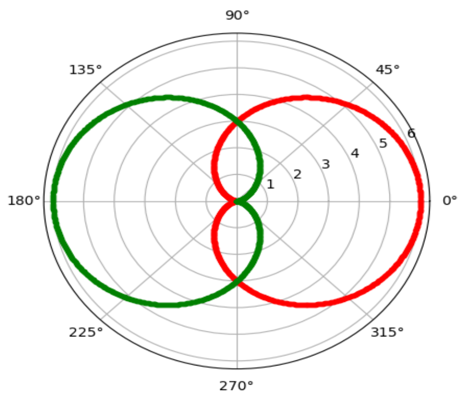
Output:



4. Write a code to plot a cardioids : $r = a + a\ \cos\theta$ **and** $r = a - a\ \cos\theta,\ \ a = 3$

```
from pylab import *
theta = linspace(0, 2*pi, 1000)
a = 3
r1 = a+a*cos(theta)
r2 = a-a*cos(theta)
polar(theta, r1,'r.', theta, r2,'g.')
show()
```

Output:

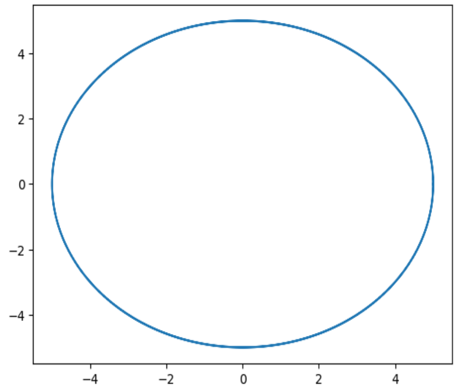


1.4. Parametric curves

1. Write a code to plot parametric equation of circle: $x = a\ \cos\theta\ , y = a\sin\theta$ **take** $a = 5$

```
from pylab import *
theta = linspace(-2*pi, 2*pi, 1000)
a = 5
x = (a*cos(theta))
y = (a*sin(theta))
plot(x, y)
show()
```

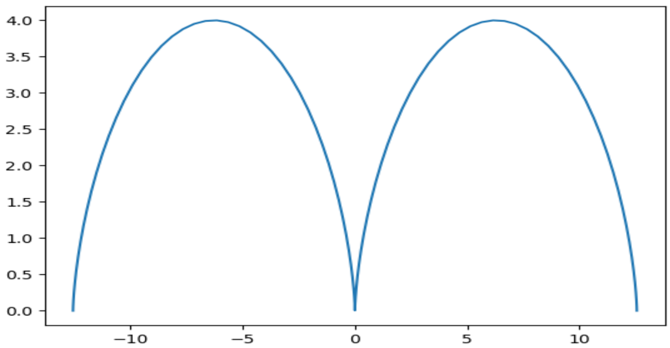
Output:



2. Write a code to plot parametric Equation of cycloid: $x = a(\theta - \sin\theta), y = a(\theta - cos\theta)$ take $a = 2$

```
from pylab import *
theta = linspace(-2*pi, 2*pi, 100)
a = 2
x = a*(theta-sin(theta))
y = a*(1-cos(theta))
plot(x,y)
show()
```

Output:



EXERCISE:

Plot the following :

1. Parabola $y^2 = 4ax$
2. Hyperbola $\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$
3. Lower half of the circle $x^2 + 2x = 4 + 4y - y^2$
4. $\cos\left(\frac{\pi x}{2}\right)$
5. $1 + \sin\left(x + \frac{\pi}{4}\right)$
6. Spiral of Archimedes: $r = a + b\theta$
7. Limacon: $r = a + b\cos\theta$

LAB Experiment 2: Finding angle between two polar curves, curvature and radius of curvature

Functions with syntax	Description
cos ( ), **sin** ( ), **tan** ( ), **asin** ( ), **acos** ( ), **atan** ( ),	trigonometric functions and inverse trigonometric functions
abs (*number*)	returns the absolute value of the number
Symbol ('*variable*')	defines a single variable
symbols ('*variable1*, *variable2*')	defines multiple variables
solve (*expression*)	solves the equation and returns the roots of the equation
diff (*expression, variable*) / **Derivative** (*expression, variable*). **doit**()	differentiates the expression w.r.t. variable
expression.**subs** (*variable, value*)	substitutes the value for the variable in the expression and returns it
simplify (*expression*)	returns a simplified mathematical expression corresponding to the input expression.
ratsimp (*expression*)	to simplify the rational function

All the above functions are from sympy library

2.1. Angle between two polar curves :

Angle between radius vector and tangent is given by $\tan\phi = r\frac{d\theta}{dr} \Rightarrow \phi = \tan^{-1}\left(r\frac{d\theta}{dr}\right)$

Angle between two polar curves at the point of intersection is $|\ \phi_1 - \phi_2\ |$

1. Find the angle between the curves $r = 4(1 + cost)$ and $r = 5(1 - cost)$.

```
from sympy import*
r,t=symbols('r,t')
r1=4+4*cos(t)
r2=5-5*cos(t)
dr1=diff(r1,t)
dr2=diff(r2,t)
t1=r1/dr1
t2=r2/dr2
q=solve(r1-r2,t)
w1=t1.subs({t:float(q[0])})
w2=t2.subs({t:float(q[0])})
y1=atan(w1)
y2=atan(w2)
w=abs(y1-y2)
print('Angle between curves in radians is %0.3f'%(w))
```

Output: `Angle between curves in radians is 1.571`

2. Finding the angle between the curves $r = 4\ cost$ and $r = 5\ sint$.

```
from sympy import*
r,t=symbols('r,t')
r1=4*cos(t)
r2=5*sin(t)
dr1=diff(r1,t)
dr2=diff(r2,t)
t1=r1/dr1
t2=r2/dr2
q=solve(r1-r2,t)
w1=t1.subs({t:float(q[0])})
w2=t2.subs({t:float(q[0])})
y1=atan(w1)
y2=atan(w2)
w=abs(y1-y2)
print('Angle between curves in radians is %0.3f'%(w))
```

Output: `Angle between curves in radians is 1.571`

2.2. Radius of curvature

Radius of curvature (Cartesian form), $\rho = \frac{(1+y_1^2)^{\frac{3}{2}}}{y_2}$

Radius of curvature (polar form), $\rho = \frac{(r^2+r_1^2)^{\frac{3}{2}}}{r^2+2r_1^2-rr_2}$

1. Find the radius of curvature, $r = 4(1 + cost)$ at $t = \frac{\pi}{2}$.

```
from sympy import*
r,t=symbols('r,t')
r=4*(1+cos(t))
r1=Derivative(r,t).doit()
r2=Derivative(r1,t).doit()
rho=(r**2+r1**2)**(1.5)/(r**2+2*r1**2-r*r2);
rho1=rho.subs(t,pi/2)
print('The radius of curvature is',(rho1))
```

Output: `The radius of curvature is 3.77123616632825`

2. Find the radius of curvature for $r = asin(nt)$ at $t = \frac{\pi}{2}$ and $n = 1$.

```
from sympy import*
r,t,a,n=symbols('r,t,a,n')
r=a*sin(n*t)
r1=Derivative(r,t).doit()
r2=Derivative(r1,t).doit()
rho=(r**2+r1**2)**(1.5)/(r**2+2*r1**2-r*r2)
rho1=rho.subs([(t,pi/2),(n,1)])
print('The radius of curvature is')
display(simplify(rho1))
```

Output: The radius of curvature is $\frac{(a^2)^{1.5}}{2a^2}$

2.3. Parametric curves

Radius of curvature (Cartesian form), $\rho = \frac{(1+y_1^2)^{\frac{3}{2}}}{y_2}$

where $y_1 = \frac{dy/dt}{dx/dt}$ and $y_2 = \frac{dy_1/dt}{dx/dt}$

1. Find radius of curvature and curvature of $x = a\cos(t)$, $y = a\sin(t)$.

```
from sympy import*
x,a,t,y=symbols('x,a,t,y')
y=a*sin(t)
x=a*cos(t)
y1=simplify(Derivative(y,t).doit())/simplify(Derivative(x,t).doit())
y2=simplify(Derivative(y1,t).doit())/simplify(Derivative(x,t).doit())
rho=simplify(1+y1**2)**(1.5)/y2
display('Radius of curvature is', ratsimp(rho))
rho1= rho.subs([(t,pi/2),(a,5)])
print('The radius of curvature at a=5, t=pi/2 is', rho1)
curvature=1/rho1
print('Curvature at (5,pi/2) is', float(curvature))
```

Output:

```
'Radius of curvature is'
```

$$-a\left(\frac{1}{\sin(t)^2}\right)^{1.5}\sin(t)^3$$

```
The radius of curvature at a=5, t=pi/2 is -5
Curvature at (5, pi/2) is -0.2
```

2. Find radius of curvature and curvature of $\boldsymbol{x = (a\cos(t))^{\frac{3}{2}}\,;\;\; y = (a\sin(t))^{\frac{3}{2}}}$

```
from sympy import*
x,a,t,y=symbols('x,a,t,y')
x=(a*cos(t))**(3/2)
y=(a*sin(t))**(3/2)
y1=simplify(Derivative(y,t).doit())/simplify(Derivative(x,t).doit())
y2=simplify(Derivative(y1,t).doit())/simplify(Derivative(x,t).doit())
rho=simplify(1+y1**2)**(1.5)/y2
display('Radius of curvature is',ratsimp(rho))
rho1=rho.subs([(t,pi/4),(a,1)])
print('the radius of curvature at a=1, t=pi/4 is %0.4f'%rho1)
curvature=1/rho1
print('curvature at (1, pi/4)is %0.3f'%float(curvature))
```

Output: `'Radius of curvature is'`

$$\frac{-3.0(\mathrm{a}\cos(t))^{3.0}\left(\frac{a\sin(t)^{3.0}}{a\cos(t)^{3}\tan(t)^{4}}+1\right)^{1.5}\sin(t)^{3}\tan(t)}{(a\sin(t))^{1.5}\cos(t)}$$

```
the radius of curvature at a=1, t=pi/4 is -2.5227
curvature at (1, pi/4) is -0.396
```

EXERCISE

1. Find the angle between radius vector and tangent to the following polar curves

   a) $r = a\theta$ and $r = \frac{a}{\theta}$

   Ans: Angle between curves in radians is 90.000

   b) $r = 2\sin(\theta)$ and $r = 2\cos(\theta)$

   Ans: Angle between curves in radians is 90.000

2. Find the radius of curvature of $r = a(1-\cos(t))$ at $t = \frac{\pi}{2}$

   Ans: $\frac{0.942809041582063(a^2)^{1.5}}{a^2}$

3. Find radius of curvature of $x = a\cos^3(t), y = a\sin^3(t)$ at $t = 0$.

   Ans: $\rho = 0.75\sqrt{3}$ and $\kappa = 0.769800$

4. Find the radius of curvature of $r = a\cos(t)$ at $t = \frac{\pi}{4}$

   Ans: $\frac{(a^2)^{1.5}}{2a^2}$

5. Find the radius of curvature of $x = a(t-\sin(t))$ and $y = a(1-\cos(t))$ at $t = \frac{\pi}{2}$.

   Ans: $\rho = 2.82842712$ and $\kappa = 0.353553$

LAB EXPERIMENT 3: Finding partial derivatives and Jacobian functions of several variables.

Functions with syntax	Description
Matrix ([[x_{11}, x_{12}], [x_{21}, x_{22}]])	creates a matrix of order 2 × 2
det (A) / Determinant(A).doit( )	returns determinant of a matrix A

All the above functions are from sympy library.

sympy.abc - module exports all latin and greek letters as Symbols.

3.1. Partial derivatives

1. Prove that mixed partial derivatives, $u_{xy} = u_{yx}$ for $u = e^x(x\cos(y) - y\sin(y))$.

```
from sympy import*
x,y=symbols('x,y')
u=exp(x)*(x*cos(y)-y*sin(y))
uxy=diff(u,x,y)
uyx=diff(u,y,x)
if uxy==uyx:
    print('Mixed partial derivatives are equal')
else:
    print('Mixed partial deivatives are not equal')
```

Output: `Mixed partial derivatives are equal`

2. Prove that if $u = e^x(x\cos(y) - y\sin(y))$, then $u_{xx} + u_{yy} = 0$.

```
from sympy import*
x,y=symbols('x,y')
u=exp(x)*(x*cos(y)-y*sin(y))
uxx=diff(u,x,x)
uyy=diff(u,y,y)
w=simplify(uxx+uyy)
print('Answer=',w)
```

Output: `Answer= 0`

3.2. Jacobians

1. If $u = \frac{xy}{z}, v = \frac{yz}{x}, w = \frac{zx}{y}$ then prove that $J = 4$.

```
from sympy import*
x,y,z=symbols('x,y,z')
u=x*y/z
v=y*z/x
w=z*x/y
J=Matrix([[diff(u,x),diff(u,y),diff(u,z)],[diff(v,x),diff(v,y),diff(v,z)],
                                            [diff(w,x),diff(w,y),diff(w,z)]])
display ('The Jacobian matrix is',J)
J1=det(J).doit()
print("The Jacobian value is",J1)
```

Output: 'The Jacobian matrix is'

$$\begin{bmatrix} \frac{y}{z} & \frac{x}{z} & -\frac{xy}{z^2} \\ -\frac{yz}{x^2} & \frac{z}{x} & \frac{y}{x} \\ \frac{z}{y} & -\frac{xz}{y^2} & \frac{x}{y} \end{bmatrix}$$

The Jacobian value is 4

2. If $u = x + 3y^2 - z^3, v = 4x^2yz, w = 2z^2 - xy$, then prove that at $(1, -1, 0)$, $J = 20$.

```
from sympy import*
x,y,z=symbols('x,y,z')
u=x+3*y**2-z**3
v=4*x**2*y*z
w=2*z**2-x*y
J=Matrix([[diff(u,x),diff(u,y),diff(u,z)],[diff(v,x),diff(v,y),diff(v,z)],
                                            [diff(w,x),diff(w,y),diff(w,z)]])
display('The Jacobian Matrix is',J)
J1=det(J).doit()
display(J1)
J2=J1.subs([(x,1),(y,-1),(z,0)])
print("The Jacobian value is",J2)
```

Output: 'The Jacobian Matrix is'

$$\begin{bmatrix} 1 & 6y & -3z^2 \\ 8xyz & 4x^2z & 4x^2y \\ -y & -x & 4z \end{bmatrix}$$

$4x^3y - 24x^2y^3 + 12x^2yz^3 + 16x^2z^2 - 192xy^2z^2$

The Jacobian value is 20

3. If $X = \rho\cos(\phi)\sin(\theta), Y = \rho\cos(\phi)\cos(\theta),\ Z = \rho\sin(\theta)$, then find $\frac{\partial(X,Y,Z)}{\partial(\rho,\phi,\theta)}$.

```
from sympy import *
from sympy.abc import *
X=rho*cos(phi)*sin(theta);
Y=rho*cos(phi)*cos(theta);
Z=rho*sin(phi);
J=Matrix([[diff(X,rho),diff(Y,rho),diff(Z,rho)],[diff(X,phi),diff(Y,phi),
                    diff(Z,phi)],[diff(X,theta),diff(Y,theta),diff(Z,theta)]])
print('The Jacobian matrix is')
display(J)
print('The Jacobian value is')
display(simplify(Determinant(J).doit()))
```

Output: The Jacobian matrix is

$$\begin{bmatrix} \sin(\theta)\cos(\phi) & \cos(\phi)\cos(\theta) & \sin(\phi) \\ -\rho\sin(\phi)\sin(\theta) & -\rho\sin(\phi)\cos(\theta) & \rho\cos(\phi) \\ \rho\cos(\phi)\cos(\theta) & -\rho\sin(\theta)\cos(\phi) & 0 \end{bmatrix}$$

The Jacobian value is

$\rho^2\cos(\emptyset)$

Exercise

1. $u = tan^{-1}\left(\frac{y}{x}\right)$. Verify that $\frac{\partial^2 u}{\partial y\,\partial x} = \frac{\partial^2 u}{\partial x\,\partial y}$

   Ans: True

2. If $u = \log\frac{(x^2+y^2)}{(x+y)}$, show that $xu_x + yu_y = 1$

   Ans: True

3. If $x = u - v, y = v - uvw$ and $z = uvw$, find Jacobian of x, y, z w.r.t. u, v, w

   Ans: uv

4. If $x = r\cos(t)$ and $y = r\sin(t)$, then find $\frac{\partial(x,y)}{\partial(r,t)}$.

   Ans: $J = r$

5. If u=$x + 3y^2 - z^3$, v=$4x^2yz$ and w=$2z^2 - xy$, find $\frac{\partial(u,v,w)}{\partial(x,y,z)}$ at (-2,-1,1) .

LAB EXPERIMENT 4: Applications of Maxima and Minima of functions of two variables, Taylor series expansion, and L' Hospital's Rule

Functions with Syntax	Description
solve (*expression*)	solves the mathematical equation (*expression* = 0) and it will return the roots of the equation
solve ([*expression1, expression2*], [*var1, var2*])	solves a system of equations (*expression1*= 0, *expression2*= 0) for the variables *var1* and *var 2*
lambdify (*var1*, *expression*)	translates *expression* into Python functions
lambdify ([*var1,var2*], *expression*)	argument to **lambdify** ( ) function is a list of variables, followed by the *expression* to be evaluated
limit (*expression, variable, value*)	returns the limit of the *expression* when the *variable* tends to the *value*
float (‘*inf*’)	the standard representation of Python infinity

All the above functions are from sympy library.

4.1. Maxima and Minima problem

1. Find the Maxima and Minima of $f(x, y) = x^2 + xy + y^2 + 3x - 3y + 4$.

```
from numpy import *
from sympy import *
x,y=symbols('x,y')
f=x**2+x*y+y**2+3*x-3*y+4
fx=diff(f,x)
fy=diff(f,y)
p=solve([fx,fy],[x,y])
print("The stationary points are",p)
A=diff(fx,x)
B=diff(fy,x)
C=diff(fy,y)
A1=A.subs(p)
B1=B.subs(p)
C1=C.subs(p)
D=A1*C1*-B1**2
print("for",p,"D is",D,"and A is",A1)
if(D>0 and A1<0):
    print("The Function attains Maxima")
    Max=f.subs(p)
    print("Maximum value is",Max)
elif(D>0 and A1>0):
    print("The Function attains Minima")
    Min=f.subs(p)
    print("Minimum value is",Min)
elif(D<0):
    print("It is a saddle point ")
elif(D==0):
    print("Further test required")
```

Output :

```
The stationary points are {x: -3, y: 3}
for {x: -3, y: 3} D is -4 and A is 2
It is a saddle point
```

4.2. Taylor Series and Maclaurin's Series Expansion

1. Expand sin(x) as Taylor series about $x = \frac{\pi}{2}$ up to 3rd degree term. Also find $sin(100°)$

```
from sympy import *
x=Symbol('x')
y=sin(x)
y1=diff(y,x)
y2=diff(y,x,2)
y3=diff(y,x,3)
yx=lambdify(x,y)
y1x=lambdify(x,y1)
y2x=lambdify(x,y2)
y3x=lambdify(x,y3)
x0=float(pi/2)
TS=yx(x0)+(x-x0)*y1x(x0)+(x-x0)**2*y2x(x0)/2+(x-x0)**3*y3x(x0)/6
print("Taylor Series expansion is")
display(simplify(TS))
t=float(100*pi/180)
print("sin(100)=",yx(t))
```

Output:

Taylor Series expansion is

$-1.02053899928946e-17\,x^3 - 0.5\,x^2 + 1.5707963267949\,x - 0.23370055013617$

sin(100)= 0.984807753012208

2. Find the Maclaurin's series expansion of $sin(x) + cos(x)$ upto 3^{rd} degree term, calculate $sin(10°) + cos(10°)$.

```
from sympy import *
x=Symbol('x')
y=sin(x)+cos(x)
y1=diff(y,x)
y2=diff(y,x,2)
y3=diff(y,x,3)
yx=lambdify(x,y)
y1x=lambdify(x,y1)
y2x=lambdify(x,y2)
y3x=lambdify(x,y3)
x0=float(pi/2)
MS=yx(0)+x*y1x(0)+x**2*y2x(0)/2+x**3*y3x(0)/6
print("Maclaurin Series expansion is")
display(simplify(MS))
m=float(10*pi/180)
print("sin(10)+cos(10)=",yx(m))
```

Output:

```
Maclaurin Series expansion is
```

-0.166666666666667 x^3 - 0.5 x^2 + 1.0 x + 1.0

```
sin(10) + cos(10)= 1.1584559306791384
```

4.3. L' Hospital Rule

1. Evaluate $\lim_{x\to 0} \frac{\sin x}{x}$

```
from sympy import *
x=Symbol('x')
l=limit((sin(x))/x,x,0)
print("limit value =", l)
```

Output: `limit value= 1`

2. Evaluate $\lim_{x\to 1} \frac{5x^4-4x^2-1}{10-x-9x^3}$

```
from sympy import *
x=Symbol('x')
l=limit((5*x**4-4*x**2-1)/(10-x-9*x**3), x, 1)
print("limit value =", l)
```

Output: `limit value= -3/7`

3. Prove that $\lim_{x\to\infty} \left(1+\frac{1}{x}\right)^x = e$

```
from sympy import *
x=Symbol('x')
l=limit((1+1/x)**x,x,float('inf'))
print("limit value =", l)
```

Output: `limit value= E`

EXERCISE

1. Find the Taylor Series expansion of $y = e^{-2x}$ at $x = 0$ upto third degree term.

   Ans: $-0.333333333333333x^3 + 0.666666666666667x^2 - 1.0x + 1.0$

2. Expand $y = xe^{-3x^2}$ as Maclaurin's series upto fifth degree term.

   Ans: $x(0.75 * x^4 - 0.75 * x^2 + 0.5)$

3. Find the Taylor Series expansion of $y = cos(x)$ at $x = \frac{\pi}{3}$.

   Ans: $0.010464x^4 + 0.00544x^3 - 0.155467x^2 - 0.1661389657x + 0.827151505$

4. Find the Maclaurin's series expansion of $y = e^{-sin^{-1}(x)}$ at $x = 0$ upto x^3 term.

   Ans: $-0.0833333333333333x^3 + 0.166666666666667x^2 - 0.5x + 1.0$

5. Evaluate $\lim_{x\to 0} \frac{2\sin x - \sin 2x}{x - \sin x}$

   Ans: 6

6. Evaluate $\lim_{x\to\infty} \sqrt{x^2 + x + 1} - \sqrt{x^2 + 1}$

   Ans: 0.5

LAB EXPERIMENT 5: Solution of First Order differential equation and plotting the solution curves.

Functions with syntax	Description
dsolve (*eq, y(x), ics={y(x$_0$): y$_0$}, hint*)	solves ordinary differential equation *eq* for function *y(x)*, using the initial condition $y(x_0) = y_0$ and using method *hint*.
Function ('y') (t)	to specify a function (for example y) of its independent variable (for example t), so that y represents y(t)
plot (*expression, range*)	plots any valid sympy *expression*. If not mentioned, *range* uses default as (-10, 10).
Y. **rhs**	extracts the right-hand side expression of equation Y

All the above functions are from sympy library.

1. Solve $\frac{dP}{dt} = r, r = 5, P(0) = 1$ and plot the solution.

```
from sympy import*
import matplotlib.pyplot as plt
t,r=symbols('t,r')
P=Function('P')
de=Eq(Derivative(P(t),t),r)
display(de)
sol=dsolve(de,P(t),ics={P(0):1})
print("The solution of given linear differential equation is:")
display(sol)
sol=sol.subs(r,5)
print("The graph of solution curve ")
display(sol)
plot(sol.rhs)
plt.show()
```

Output:

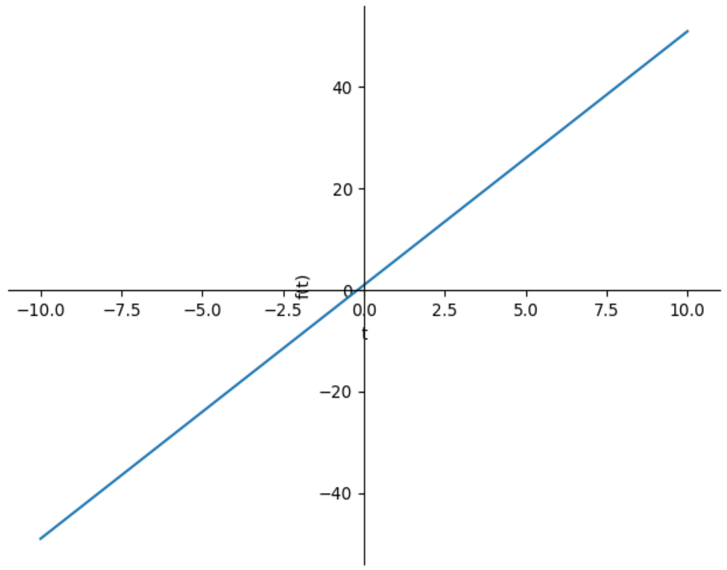
$$\frac{d}{dt}P(t) = r$$

The solution of given linear differential equation is:

$$P(t) = rt + 1$$

The graph of solution curve

$$P(t) = 5t + 1$$



2. Solve $\frac{dy}{dx} = -Ky, K = 0.3, y(0) = 5$ and plot the solution.

```
from sympy import*
import matplotlib.pyplot as plt
x,k=symbols('x, k')
y=Function('y')
de=Eq(Derivative(y(x),x),-k*y(x))
display(de)
sol=dsolve(de,y(x),ics={y(0):5})
print("The solution of given linear differential equation is:")
display(sol)
sol=sol.subs(k,0.3)
print("The graph of solution curve ")
display(sol)
plot(sol.rhs)
plt.show()
```

Output:

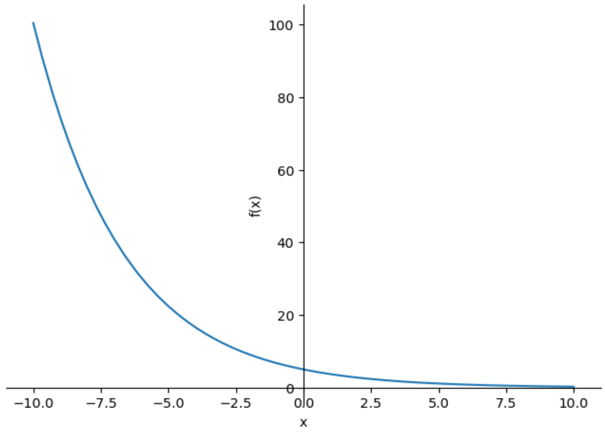
$$\frac{d}{dx}y(x) = -ky(x)$$

The solution of given linear differential equation is:

$$y(x) = 5e^{-kx}$$

The graph of solution curve

$$y(x) = 5e^{-0.3x}$$



3. Solve $x^3\frac{dy}{dx} - x^2y + y^4cosx = 0, y(\pi) = 1$ and plot the solution.

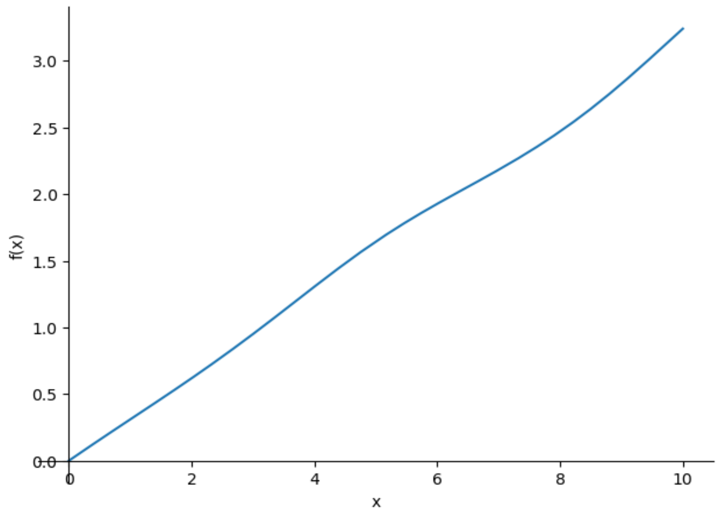
```
from sympy import*
import matplotlib.pyplot as plt
x=symbols('x')
y=Function('y')
y1=Derivative(y(x),x)
de=Eq(x**3*y1-(x**2)*y(x)+(y(x)**4)*cos(x),0)
display(de)
sol=dsolve(de,y(x),ics={y(pi):1},hint="Bernoulli")
print("The solution and graph of given Bernoulli's differential equation is")
display(sol)
plot(sol.rhs)
plt.show()
```

Output:

$$x^3\frac{d}{dx}y(x) - x^2y(x) + y^4(x)\cos(x) = 0$$

The solution and graph of given Bernoulli's differential equation is:

$$y(x) = \sqrt[3]{\frac{x^3}{3\sin(x) + \pi^3}}$$



4. Solve $\frac{dy}{dx} + ytan(x) - y^3sec(x) = 0,\, y(0) = 1$ and plot the solution.

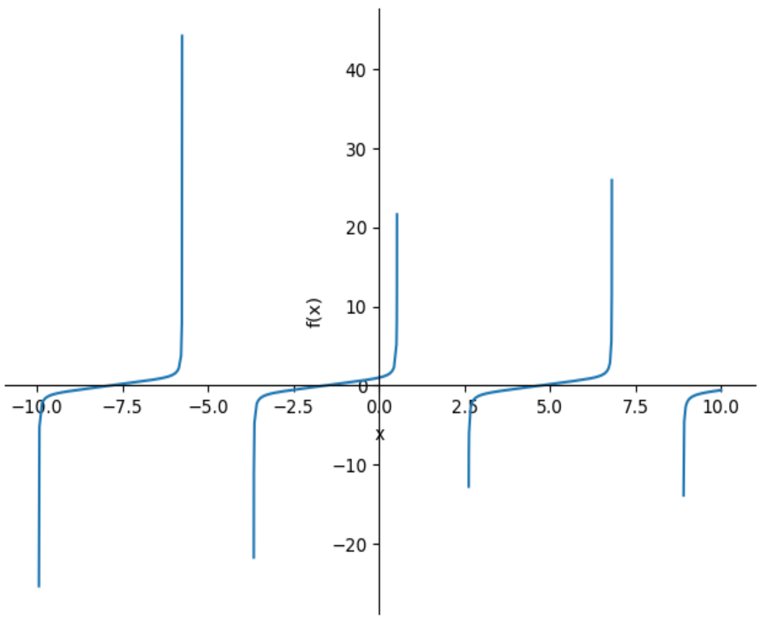
```
from sympy import*
import matplotlib.pyplot as plt
x=symbols('x')
y=Function('y')
y1=Derivative(y(x),x)
de=Eq(y1+y(x)*tan(x)-y(x)**3*sec(x),0)
display(de)
sol=dsolve(de,y(x),ics={y(0):1},hint="Bernoulli")
print("The solution and graph of given Bernoulli's differential equation is")
display(sol)
plot(sol.rhs)
plt.show()
```

Output:

$$-y^3(x)\sec(x) + y(x)\tan(x) + \frac{d}{dx}y(x) = 0$$

The solution and graph of given Bernoulli's differential equation is

$$y(x) = \sqrt{\frac{1}{1 - 2\sin(x)}}\cos(x)$$



5. Simulate $\tau\frac{dy}{dt} = -y + K_p$; $K_p = 3.0, \tau = 2.0$ and $y(0) = 1$.

```
from sympy import*
import matplotlib.pyplot as plt
T,t,K=symbols('T, t, K')
y=Function('y')
de=Eq(T*Derivative(y(t),t),-y(t)+K)
display(de)
sol=dsolve(de,y(t),ics={y(0):1})
print("The solution of given linear differential equation is:")
display(sol)
sol=sol.subs([(T,2.0),(K,3.0)])
print("The graph of solution curve ")
display(sol)
plot(sol.rhs)
plt.show()
```

Output:

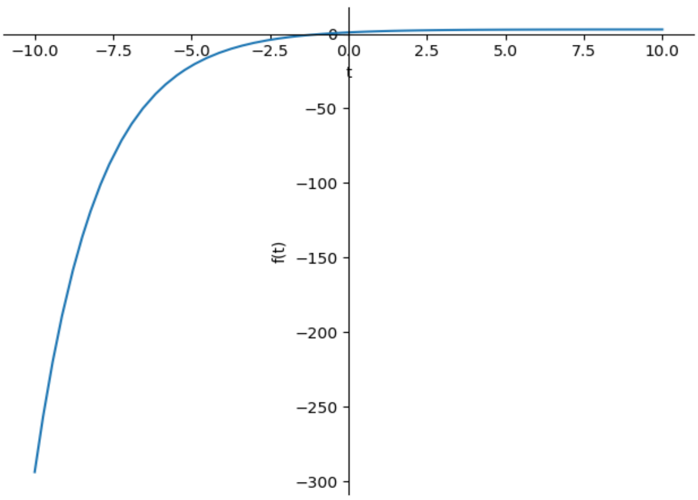
$$T\frac{d}{dt}y(t) = K - y(t)$$

The solution of given linear differential equation is:

$$y(t) = K + (1 - K)\,e^{-\frac{t}{T}}$$

The graph of solution curve

$$y(t) = 3.0 - 2.0e^{-0.5t}$$



EXERCISE

1. Solve the following differential equations and plot the solution curves:

a. $y \sin x \, dx - (1 + y^2 + \cos^2 x) \, dy = 0.$

Ans: $\frac{1}{2} y\cos 2x + \left(\frac{3}{2}\right) y + \frac{y^3}{3} = 0.$

b. $\frac{dy}{dx} = x + y$ subject to condition $y(0) = 2.$

Ans: $y = 3e^x - x - 1.$

c. $\frac{dy}{dx} = x^2$ subject to condition $y(0) = 5.$

Ans: $y = \frac{x^3}{3} + 5.$

d. $x^2 y' = y \log(y) - y'$

Ans: $y(x) = e^{C_1 tan^{-1} x}$

e. $y' - y - xe^x = 0.$

Ans: $y(x) = (C_1 + \frac{x^2}{2})e^x$

LAB EXPERIMENT 6: Finding GCD using Euclid's algorithm

Functions with syntax	Description
gcd (*f, g*)	computes Greatest Common Divisor for polynomials *f* and *g*
igcd (*a, b*)	returns the value of the greatest common divisor for non-negative integers a and b

The above functions are from the sympy library.

Euclidean Algorithm:

1. For $a > b, a = b \times q + r\ (by\ division\ algorithm)$, where $0 \leq r < b$
2. If r = 0, then GCD = b
3. If r $\neq$ 0, then assume $a = b$ & $b = r$ and repeat steps 1 to 3 until r = 0

6.1. Finding GCD of two numbers using Euclidean algorithm

1. Write a code to find the GCD of 614 and 124 by defining a new function using Euclidean Algorithm

```
def gcdab(a,b):
    r=1
    if b<a:
        a,b=b,a
    while(r>0):
        r=b%a
        print(b,"=",a ,"x", b//a,"+",r)
        b=a
        a=r
        continue
    print('GCD =',b)
gcdab(614,124)
```

Output:

```
614 = 124 x 4 + 118
124 = 118 x 1 + 6
118 = 6 x 19 + 4
6 = 4 x 1 + 2
4 = 2 x 2 + 0
GCD = 2
```

6.2. Identifying Relatively Prime Numbers

1. Write a code to check whether 163 and 512 are relatively prime

```
def rp(a,b):
    r=1
    a1,b1=a,b
    if b<a:
        a,b=b,a
    while(r>0):
        r=b%a
        b=a
        a=r
        continue
    if b==1:
        print(f' {a1} and {b1} are relatively prime')
    else:
        print(f' {a1} and {b1} are not relatively prime')
rp(163,512)
```

Output: `163 and 512 are relatively prime`

6.3. Checking divisibility

1. Write a code to check the number 8 divides the number 128.

```
def div(a,b):
    r=1
    a1,b1=a,b
    if b<a:
        a,b=b,a
    while(r>0):
        r=b%a
        b=a
        a=r
        continue
    if b==a1:
        print(f' {a1} divides {b1} ')
    else:
        print(f' {a1} doesnot divides {b1}')
div(8,128)
```

Output: `8 divides 128`

6.4. Express GCD of a, b as a linear combination of a and b.

1. Calculate GCD of 76, 13 and express GCD as 76x + 13y.

```
def glin(a, b):
    if b == 0:
        return a, 1, 0
    gcd, x1, y1 = glin(b, a % b)
    x = y1
    y = x1 - (a // b) * y1
    return gcd, x, y
a, b = 76, 13
gcd, x, y = glin (a, b)
print(f"GCD of {a} and {b} is: {gcd}")
print(f"Linear Combination: {gcd} = {a}*({x}) + {b}*({y})")
```

Output:

```
GCD of 76 and 13 is: 1
Linear Combination: 1 = 76*(6) + 13*(-35)
```

EXERCISE

1. Find the GCD of 234 and 672 using Euclidean algorithm.

   Ans: 6

2. What is the largest number that divides both 1024 and 1536?

   Ans: 512

3. Find the greatest common divisor of 6096 and 5060?

   Ans: 4

4. Prove that 1235 and 2311 are relatively prime.

LAB EXPERIMENT 7: Solving Linear congruence of the form $\boldsymbol{ax \equiv b(mod m)}$.

Procedure to solve $\boldsymbol{ax \equiv b(mod\ m)}$

(i) Find $\gcd(a, m) = d$
(ii) If d does not divide b, then the linear congruence has no solution
(iii) If d divides b, then the linear congruence has d solutions
(iv) Find an integer i from 0 to $m-1$ such that $x_0 = \frac{(m \times i + b)}{a}$ is an integer and x_0 is the initial soln.
(v) Other solutions are given by $x = x_0 + \frac{m}{d} \times t$, where $t = 0,1,2,\cdots,d-1$

1. Show that the linear congruence $\boldsymbol{6x \equiv 5(mod15)}$ has no solution

```
from sympy import*
a=int(input('enter integer a '))
b=int(input('enter integer b '))
m=int(input('enter integer m '))
d=gcd(a,m)
if(b%d!=0):
    print('The congruence has no integer solution')
else:
    for i in range(0,m):
        x0=(m*i+b)/a
        if(x0//1==x0):
            print(f'The {d} solutions are')
            for j in range (0,d):
                x=int(x0)+(m/d)*j
                print(f' x = {x}(mod {m})')
            break
```

Output:

```
enter integer a 6
enter integer b 5
enter integer m 15
the congruence has no integer solution
```

2. Find the solution of the congruence $\boldsymbol{5x \equiv 3(mod13)}$.

```
from sympy import*
a=int(input('enter integer a '))
b=int(input('enter integer b '))
m=int(input('enter integer m '))
d=gcd(a,m)
if(b%d!=0):
    print('The congruence has no integer solution')
else:
    for i in range(0,m):
        x0=(m*i+b)/a
        if(x0//1==x0):
            print(f'The {d} solutions are')
            for j in range (0,d):
                x=int(x0)+(m/d)*j
                print(f' x = {x}(mod {m})')
            break
```

Output:
```
enter integer a 5
enter integer b 3
enter integer m 13
The 1 solutions are
 x = 11(mod 13)
```

Finding inverse of a mod m is equivalent to find $ax \equiv 1(mod\ m)$.

3. Find the inverse of 5 mod 13

```
from sympy import*
a=int(input('enter integer a '))
b=int(input('enter integer b '))
m=int(input('enter integer m '))
d=gcd(a,m)
if(b%d!=0):
    print('The congruence has no integer solution')
else:
    for i in range(0,m):
        x0=(m*i+b)/a
        if(x0//1==x0):
            print(f'The {d} solutions are')
            for j in range (0,d):
                x=int(x0)+(m/d)*j
                print(f' x = {x}(mod {m})')
            break
```

Output:
```
enter integer a 5
enter integer b 1
enter integer m 13
The 1 solutions are
 x = 8(mod 13)
```

4. Find the solution of the linear congruence $28x \equiv 56(mod\ 49)$

```
from sympy import*
a=int(input('enter integer a '))
b=int(input('enter integer b '))
m=int(input('enter integer m '))
d=gcd(a,m)
if(b%d!=0):
    print('The congruence has no integer solution')
else:
    for i in range(0,m):
        x0=(m*i+b)/a
        if(x0//1==x0):
            print(f'The {d} solutions are')
            for j in range (0,d):
                x=int(x0)+(m/d)*j
                print(f' x = {x}(mod {m})')
            break
```

Output:
```
enter integer a 28
enter integer b 56
enter integer m 49
The 7 solutions are
 x = 2(mod 49)
 x = 9(mod 49)
 x = 16(mod 49)
 x = 23(mod 49)
 x = 30(mod 49)
 x = 37(mod 49)
 x = 44(mod 49)
```

EXERCISE

1. Find the solution of the congruence $12x \equiv 6(mod 23)$.

   Ans: 12

2. Find the multiplicative inverse of 3 mod 31.

   Ans: 21

3. Prove that $12x \equiv 7(mod 14)$ has no solution. Give a reason for the answer.

   Ans: Because GCD (12,14) = 2 and 2 doesnot divide 7.

LAB EXPERIMENT 8: Numerical solution of a system of equations, test for consistency and graphical representation of the solution

8.1. System of homogenous linear equations:

The linear system of equations of the form AX=0 is called the system of homogenous linear equations.

The n-tuple (0, 0, ..., 0) is a trivial solution of the system. The homogeneous system of m equations AX =0 in n unknowns has a non-trivial solution if and only if the rank of the matrix A is less than n. Further, if $\rho(A) = r < n$, then the system possesses (n-r) linearly independent solutions.

Functions with syntax		Description
sympy library	**numpy library**	
Matrix ([[*row 1*], [*row 2*], …, [*row n*]])	**matrix** ([[*row 1*], [*row 2*], …, [*row n*]])	creates a *m x n* matrix .
A.rank ( )	**linalg.matrix_rank** (*A*)	gives the rank of matrix A
shape (A)	**shape** (A)	gives the dimension of matrix A
A.shape [0], **A.shape** [1]	**A.shape** [0], **A.shape** [1]	gives the number of rows, columns in matrix A respectively
A.col_insert (*A.shape [1], B*)	**concatenate((***A,B), axis=1***)**	creates augmented matrix AB
x,y,z = symbols('*x, y, z*') **solve_linear_system** (*AB, x, y, z*)	**linalg.solve** (*A, B)*	solves the system of equations and returns the solutions of *x, y, z*

1. **Check whether the following system of homogenous linear equation has non-trivial solution.** $x_1 + 2x_2 - x_3 = 0$, $2x_1 + x_2 + 4x_3 = 0$, $3x_1 + 3x_2 + 4x_3 = 0$

```
from sympy import *
A=Matrix([[1 ,2 ,-1],[2 ,1 , 4],[3 ,3 , 4]])
B=Matrix([0,0,0])
r=A.rank()
n=A.shape[1]
print(f"The rank of the coefficient matrix is {r}")
print(f"The number of unknowns are {n}")
if (r==n):
    print("System has trivial solution")
else:
    print("System has", n-r, "non-trivial linearly independent solution(s)")
```

OUTPUT:

```
The rank of the coefficient matrix is 3
The number of unknowns are 3
System has trivial solution
```

2. **Check whether the following system of homogenous linear equation has non-trivial solution** $\boldsymbol{x_1 + 2x_2 - x_3 = 0}$ **,** $\boldsymbol{2x_1 + x_2 + 4x_3 = 0}$ **,** $\boldsymbol{x_1 - x_2 + 5x_3 = 0}$

```
from sympy import *
A=Matrix([[1,2,-1],[2,1,4],[1,-1,5]])
B=Matrix([0,0,0])
r=A.rank()
n=A.shape[1]
print(f"The rank of the coefficient matrix is {r}")
print(f"The number of unknowns are {n}")
if (r==n):
    print("System has trivial solution")
else:
    print("System has", n-r, "non-trivial linearly independent solution(s)")
```

OUTPUT:

```
The rank of the coefficient matrix is 2
The number of unknowns are 3
System has 1 non-trivial linearly independent solution(s)
```

8.2. System of Non-homogenous Linear Equations

The linear system of equations of the form AX = B is called system of non-homogenous linear equations if not all elements in B are zeros.

The non-homogeneous system of m equations AX=B in n unknowns is

- Consistent (has a solution) if and only if, $\rho(A) = \rho([A|B])$
- has unique solution if $\rho(A) = \rho([A|B]) = n$
- has infinitely many solutions, $\rho(A) < n$
- inconsistent $\rho(A) \neq \rho([A|B])$

Functions with syntax	Description
figure ( )	create a new figure, or activate an existing figure.
fig.add_subplot(*111, projection='3d'***)**	defines new axes as a typical subplot, with a 3d projection, to alert Matplotlib that we're about to throw three-dimensional data at it.
ax.plot_surface (x,*y,z,alpha=0.5)*	# surface plot is a representation of three-dimensional dataset, where X and Y are 2D array of points of x and y while Z is 2D array of heights. # alpha parameter is used to control the transparency of a plot. It takes a value between 0 (completely transparent) and 1 (completely opaque).
ax.scatter *(X, Y, Z, color='red')*	marks the point x,y,z with the specified color.

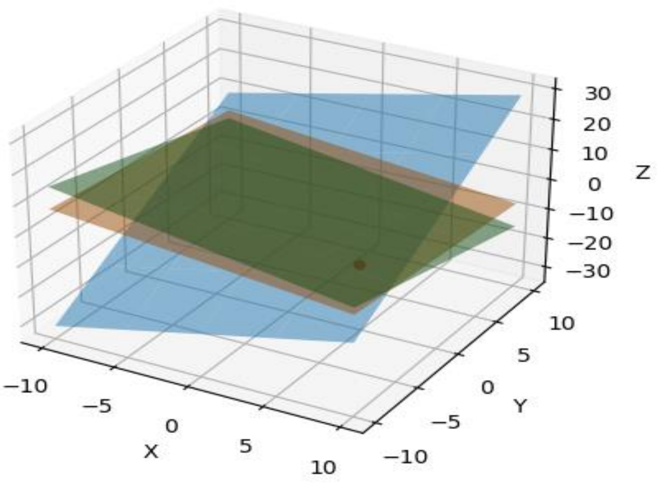
The above functions are from matplotlibrary

1. Examine the consistency of the following system of equations and solve if consistent, also plot the graphical solution. $x_1 + 2x_2 - x_3 = 0$, $2x_1 + x_2 + 4x_3 = 0$, $3x_1 + 3x_2 + 4x_3 = 0$

```
from numpy import *
from matplotlib.pyplot import *
A = matrix([[1, 2, -1], [2, 1, 4], [3,3,4]])
B = matrix([[1],[2],[1]])
AB=concatenate((A, B), axis=1)
if(linalg.matrix_rank(A)==linalg.matrix_rank(AB)):
    if(linalg.matrix_rank(A)==A.shape [1]):
        print("The system has unique solution")
    else:
        print("The system has infinitely many solutions")
    x0 = linalg.solve(A, B)
    print(x0)
    X,Y,Z= x0[0],x0[1],x0[2]
    x_lim,y_lim = linspace(-10, 10, 5),linspace(-10, 10, 5)
    x, y = meshgrid(x_lim, y_lim)
    z1,z2,z3 =(x+2*y - 1),(2-2*x-y)/4,(1-3*x-3*y)/4
    fig = figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.plot_surface(x,y,z1,alpha=0.5),ax.plot_surface(x,y,z2,alpha=0.5),
                                       ax.plot_surface(x,y,z3,alpha=0.5)
    ax.scatter(X, Y, Z, color='red')
    ax.set_xlabel('X'),ax.set_ylabel('Y'),ax.set_zlabel('Z')
    show()
else:
     print("The system of equations is inconsistent")
```

OUTPUT:

```
The system has unique solution
[[ 7.]
 [-4.]
 [-2.]]
```



2. Examine the consistency of the following system of equations and solve and plot the solution if consistent. $x_1 + 2x_2 - x_3 = 0$, $2x_1 + x_2 + 5x_3 = 0$, $3x_1 + 3x_2 + 4x_3 = 0$

```
from numpy import *
from matplotlib.pyplot import *
A = matrix([[1, 2, -1], [2, 1, 5], [3,3,4]])
B = matrix([[1],[2],[1]])
AB=concatenate((A, B), axis=1)
if(linalg.matrix_rank(A)==linalg.matrix_rank(AB)):
    if(linalg.matrix_rank(A)==A.shape [1]):
        print("The system has unique solution")
    else:
        print("The system has infinitely many solutions")
    x0 = linalg.solve(A, B)
    print(x0)
    X,Y,Z= x0[0],x0[1],x0[2]
    x_lim,y_lim = linspace(-10, 10, 5),linspace(-10, 10, 5)
    x, y = meshgrid(x_lim, y_lim)
    z1,z2,z3 =(x+2*y - 1),(2-2*x-y)/4,(1-3*x-3*y)/4
    fig = figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.plot_surface(x,y,z1,alpha=0.5),ax.plot_surface(x,y,z2,alpha=0.5),
                                    ax.plot_surface(x, y, z3, alpha=0.5)
    ax.scatter(X, Y, Z, color='red')
    ax.set_xlabel('X'),ax.set_ylabel('Y'),ax.set_zlabel('Z')
    show()
else:
     print("The system of equations is inconsistent")
```

OUTPUT:

```
The system of equations is inconsistent
```

EXERCISE

1. Find the solution of the homogenous system of equations

$$x + y + z = 0, \qquad 2x + y - 3z = 0, \qquad 4x - 2y - z = 0$$

2. Find the solution of the non-homogenous system of equations

$$25x + y + z = 27, \qquad 2x + 10y - 3z = 9, \qquad 4x - 2y - z = -10$$

3. Find the solution of the non-homogenous system of equations

$$x + y + z = 2, \qquad 2x + 2y - 2z = 4, \qquad x - 2y - z = 5$$

LAB EXPERIMENT 9: Solution of a System of Linear Equations by Gauss-Seidel Method

As we already know the *def* keyword is used to define a normal function in Python. Similarly, the *lambda* keyword is used to define an anonymous function in Python

Syntax : ***lambda*** *arguments* : *expression*

Lambda function can have any number of arguments but only one expression, which is evaluated and returned. It is efficient whenever one wants to create a function that only contains expressions in a single line of a statement.

Procedure for Gauss–Seidel Method:

Given a system of linear equations in three unknowns:

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1$$
$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2$$
$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3$$

- Gauss-Seidel can be applied only if the system of equations is diagonally dominant. That is, $|a_{11}| > |a_{12}| + |a_{13}|$, $|a_{22}| > |a_{21}| + |a_{23}|$, $|a_{33}| > |a_{31}| + |a_{32}|$
- If not diagonally dominant, then rearrange the system to satisfy the above condition. Write the system of equations as $x_1 = \frac{1}{a_{11}}[b_1 - a_{12}x_2 - a_{13}x_3]$,
$$x_2 = \frac{1}{a_{22}}[b_2 - a_{21}x_1 - a_{23}x_3]$$
$$x_3 = \frac{1}{a_{33}}[b_3 - a_{31}x_1 - a_{32}x_2]$$
- Start the iteration with $x_1 = 0, x_2 = 0, x_3 = 0$ as the initial approximation values.
- Keep substituting the recent values of x_1, x_2, x_3 in the above formula for x_1, x_2, x_3 in every iteration. This process continues until we get the solution to the desired degree of accuracy.

1. Solve the system using Gauss-Seidel method:

$$\mathbf{20x + y - 2z = 17, 3x + 20y - z = -18, 2x - 3y + 20z = 25}$$

```
f1 = lambda x,y,z:(17-y+2*z)/20
f2 = lambda x,y,z:(-18-3*x+z)/20
f3 = lambda x,y,z:(25-2*x+3*y)/20
x0, y0, z0 = 0, 0, 0
e = float(input('Enter tolerable error : '))
print('\t Iteration \t x \t y \t z\n')
for i in range (0, 25):
    x1 = f1(x0, y0, z0)
    y1 = f2(x1, y0, z0)
    z1 = f3(x1, y1, z0)
    print('\t\t %d \t %0.4f \t %0.4f \t %0.4f\n' %(i, x1, y1, z1))
    e1, e2, e3 = abs(x0-x1), abs(y0-y1), abs(z0-z1)
    x0,y0,z0 = x1,y1,z1
    if e1>e and e2>e and e3>e:
```

```
        continue
    else:
        break
print('\n Solution : x = %0.3f , y = %0.3f and z = %0.3f\n'% (x1, y1, z1))
```

OUTPUT:

```
Enter tolerable error: 0.001
       Iteration   x     y     z

            0     0.8500      -1.0275     1.0109

            1     1.0025      -0.9998     0.9998

            2     1.0000      -1.0000     1.0000

 Solution: x = 1.000, y = -1.000 and z = 1.000
```

2. Solve the system using Gauss-Seidel method:

$$x + 2y - z = 3,\ 3x - y + 2z = 1,\ 2x - 2y + 6z = 2$$

```
f1 = lambda x, y, z: (1+y-2*z)/3
f2 = lambda x, y, z: (3-x+z)/2
f3 = lambda x, y, z: (2-2*x+2*y)/6
x0, y0, z0 = 0, 0, 0
e = float(input('Enter tolerable error : '))
print('\t Iteration \t x\t y\t z\n')
for i in range(0, 25):
    x1= f1(x0, y0, z0)
    y1= f2(x1, y0, z0)
    z1= f3(x1, y1, z0)
    print('\t\t  %d \t %0.4f \t %0.4f \t %0.4f \n' %(i, x1, y1, z1))
    e1,e2,e3 = abs(x0-x1),abs(y0-y1),abs(z0-z1)
    x0,y0,z0 = x1,y1,z1
    if e1>e and e2>e and e3>e:
        continue
    else :
        break
print('\n Solution : x = %0.3f, y = %0.3f and z = %0.3f\n'%(x1, y1, z1))
```

OUTPUT:

```
Enter tolerable error: 0.0001
       Iteration   x     y     z

            0     0.3333      1.3333      0.6667

            1     0.3333      1.6667      0.7778

 Solution: x = 0.333, y = 1.667 and z = 0.778
```

3. Solve the system using Gauss-Seidel method:

$$10x + y + z = 12,\ x + 10y + z = 12,\ x + y + 10z = 12.$$

```
f1 = lambda x, y, z: (12-y-z)/10
f2 = lambda x, y, z: (12-x-z)/10
f3 = lambda x, y, z: (12-x-y)/10
x0, y0, z0 = 0, 0, 0
e = float(input('Enter tolerable error: '))
print('\t Iteration \t x \t y \t z \n')
for i in range (0, 25):
    x1 = f1(x0, y0, z0)
    y1 = f2(x1, y0, z0)
    z1 = f3(x1, y1, z0)
    print('\t\t %d \t %0.4f \t %0.4f \t %0.4f\n' %(i, x1, y1, z1))
    e1,e2,e3 = abs(x0-x1), abs(y0-y1), abs(z0-z1)
    x0,y0,z0 = x1,y1,z1
    if e1>e and e2>e and e3>e:
        continue
    else :
        break
print ('\n Solution: x = %0.3f, y = %0.3f and z = %0.3f\n'% (x1, y1, z1))
```

OUTPUT:

```
Enter tolerable error: 0.0001
       Iteration   x     y     z

            0     1.2000      1.0800      0.9720

            1     0.9948      1.0033      1.0002

            2     0.9996      1.0000      1.0000

            3     1.0000      1.0000      1.0000

 Solution: x = 1.000, y = 1.000 and z = 1.000
```

4. Solve the system using Gauss-Seidel method:

$$5x - y - z = -3, \qquad x - 5y + z = -9, \qquad 2x + y - 4z = -15$$

```
f1 = lambda x, y, z: (-3+y+z)/5
f2 = lambda x, y, z: (9+x+z)/5
f3 = lambda x, y, z: (15+2*x+y)/4
x0, y0, z0 = 0, 0, 0
e = float(input('Enter tolerable error: '))
print('\t Iteration \t x \t y \t z \n')
for i in range (0,  25):
    x1 = f1(x0, y0, z0)
```

```
    y1 = f2(x1, y0, z0)
    z1 = f3(x1, y1, z0)
    print('\t\t %d \t %0.4f \t %0.4f \t %0.4f\n' %(i, x1, y1, z1))
    e1,e2,e3 = abs(x0-x1),abs(y0-y1),abs(z0-z1)
    x0,y0,z0 = x1,y1,z1
    if e1>e and e2>e and e3>e:
        continue
    else :
        break
print('\n Solution: x = %0.3f, y = %0.3f and z = %0.3f\n'%(x1, y1, z1))
```

OUTPUT:

```
Enter tolerable error: 0.0001
       Iteration   x     y     z

             0     -0.6000     1.6800      3.8700

             1     0.5100      2.6760      4.6740

             2     0.8700      2.9088      4.9122

             3     0.9642      2.9753      4.9759

             4     0.9902      2.9932      4.9934

             5     0.9973      2.9982      4.9982

             6     0.9993      2.9995      4.9995

             7     0.9998      2.9999      4.9999

             8     0.9999      3.0000      5.0000

 Solution: x = 1.000, y = 3.000 and z = 5.000
```

EXERCISE

1. Check whether the following system are diagonally dominant or not
   (i) $25x + y + z = 27,\ 2x + 10y - 3z = 9, 4x - 2y - 12z = -10$
   (ii) $x + y + z = 7, 2x + y - 3z = 3, 4x - 2y - z = -1$

2. Solve the following system of equations using Gauss Seidel method
   (i) $4x + y + z = 6, 2x + 5y - 2z = 5, x - 2y - 7z = -8$
   (ii) $27x + 6y - z = 85, 6x + 15y + 2z = 72, x + y + 54z = 110$

LAB EXPERIMENT 10: Compute eigenvalues and corresponding eigenvectors. Find dominant and corresponding eigenvector by Rayleigh power method

Functions with syntax	Description
linalg.eig(*A* **)**	computes the eigenvalues and eigenvectors of a square array/matrix.
dot(*A*, *X*)	returns the dot product of vectors *A* and *X*.

The above functions are from numpy library.

10.1. Eigenvalues and Eigenvectors

Let A be a n × n matrix. λ is an eigenvalue of matrix A and **x**, a non-zero vector, is called an eigenvector if it satisfies A**x** = λ**x.** We say, **x** is an eigenvector of A corresponding to eigenvalue λ.

1. Obtain the eigen values and eigen vectors for the given matrix $\begin{pmatrix} \mathbf{4} & \mathbf{3} & \mathbf{2} \\ \mathbf{1} & \mathbf{4} & \mathbf{1} \\ \mathbf{3} & \mathbf{10} & \mathbf{4} \end{pmatrix}$

```
from numpy import *
A = matrix([[4, 3, 2], [1, 4, 1], [3, 10, 4]])
w, v = linalg.eig(A)
print("\n Eigenvalues: \n", w)
print("\n Eigenvectors: \n", v)
```

OUTPUT:

```
Eigenvalues:
[8.98205672 2.12891771 0.88902557]

Eigenvectors:
[[-0.49247712 -0.82039552 -0.42973429]
[-0.26523242  0.14250681 -0.14817858]
[-0.82892584  0.55375355  0.89071407]]
```

2. Obtain the eigen values and eigen vectors for the given matrix $\begin{pmatrix} \mathbf{1} & \mathbf{-3} & \mathbf{3} \\ \mathbf{3} & \mathbf{-5} & \mathbf{3} \\ \mathbf{6} & \mathbf{-6} & \mathbf{4} \end{pmatrix}$

```
from numpy import *
A = matrix([[1, -3, 3], [3, -5, 3], [6, -6, 4]])
w, v = linalg.eig(A)
print("\n Eigen values: \n", w)
print("\n Eigen vectors: \n", v)
```

OUTPUT:

```
Eigenvalues:
[ 4. -2. -2.]
```

```
Eigenvectors:
[[ 0.40824829 -0.40824829 -0.30502542]
 [ 0.40824829  0.40824829 -0.808424  ]
 [ 0.81649658  0.81649658 -0.50339858]]
```

10.2. Largest eigenvalue and corresponding eigenvector by Rayleigh's power method

Procedure for Rayleigh's power method:

Given a matrix A,

- we assume the initial eigenvector $X^{(0)}$ as $[1\ \ 0\ \ 0]^T$ (or) $[0\ \ 1\ \ 0]^T$ (or) $[0\ \ 0\ \ 1]^T$ (or) $[1\ \ 1\ \ 1]^T$ and find the matrix product $AX^{(0)}$
- take the largest absolute value outside from $AX^{(0)}$ as a common factor to obtain the form $AX^{(0)} = \lambda^{(1)}X^{(1)}$ (This process is called normalization)
- again, find product $AX^{(1)}$ and normalize it to put in the form $AX^{(1)} = \lambda^{(2)}X^{(2)}$
- this iterative process is continued till two consecutive iterative values of λ and X are same upto a desired degree of accuracy.
- the values so obtained are respectively the largest eigenvalue λ and the corresponding eigen vector X of the given matrix A.

1. Compute the numerically largest eigenvalue of P = $\begin{bmatrix} 6 & -2 & 2 \\ -2 & 3 & -1 \\ 2 & -1 & 3 \end{bmatrix}$ **by power method.**

```
from numpy import *
def normalize(x):
    lam = abs(x).max()
    x_n = x/lam
    return lam, x_n
x = matrix([[1], [1], [1]])
A = matrix([[6, -2, 2], [-2, 3, -1], [2, -1, 3]])
for i in range(10):
    x1 = dot(A,x)
    l, x = normalize(x1)
print('Eigenvalue:', l)
print('Eigenvector:', x)
```

OUTPUT:

```
Eigenvalue: 7.999988555930031
Eigenvector:
[[ 1.        ]
[-0.49999785]
[ 0.50000072]]
```

2. Compute the numerically largest eigenvalue of P = $\begin{bmatrix} 1 & 1 & 3 \\ 1 & 5 & 1 \\ 3 & 1 & 1 \end{bmatrix}$ by power method.

```
from numpy import *
def normalize(x):
    lam = abs(x).max()
    x_n = x/lam
    return lam, x_n
x = matrix([[1], [1], [1]])
A = matrix([[1, 1, 3], [1, 5, 1], [3, 1, 1]])
for i in range(10):
    x1 = dot(A,x)
    l, x = normalize(x1)
print('Eigenvalue:', l)
print('Eigenvector:', x)
```

OUTPUT:

```
Eigenvalue: 6.001465559355154
Eigenvector: [[0.5003663]
 [1.        ]
 [0.5003663]]
```

EXERCISE

1. Find the eigenvalues and eigenvectors of the following matrices

a. P = $\begin{bmatrix} 25 & 1 \\ 1 & 3 \end{bmatrix}$ b. P = $\begin{bmatrix} 25 & 1 & 2 \\ 1 & 3 & 0 \\ 2 & 0 & -4 \end{bmatrix}$ c. P = $\begin{bmatrix} 11 & 1 & 2 \\ 0 & 10 & 0 \\ 0 & 0 & 12 \end{bmatrix}$ d. P = $\begin{bmatrix} 3 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 12 \end{bmatrix}$

2. Find the dominant eigenvalue of the matrix P = $\begin{bmatrix} 25 & 1 & 2 \\ 1 & 3 & 0 \\ 2 & 0 & -4 \end{bmatrix}$. Take $X_0 = (1, 0, 1)^T$.

3. Find the dominant eigenvalue of the matrix P = $\begin{bmatrix} 6 & 1 & 2 \\ 1 & 10 & -1 \\ 2 & 1 & -4 \end{bmatrix}$. Take $X_0 = (1, 1, 1)^T$.

4. Find the dominant eigenvalue of the matrix P = $\begin{bmatrix} 5 & 1 & 1 \\ 1 & 3 & -1 \\ 2 & -1 & -4 \end{bmatrix}$ by power method.