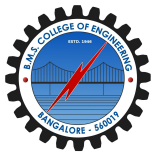


BMS College of Engineering,
Bangalore-19

Department of Mathematics

LAB MANUAL

First Semester

Engineering Mathematics using Python

January 19, 2023

General Instructions:

1. Students need to maintain record of every lab and to be submitted to the concerned faculty for evaluation in the next lab.
2. Students need to go through the lab manual and study the sample programmes and need to complete the given task in the same lab session and to show the outputs to the concerned faculty.
3. This course is an integrated course, so that there is no SEM End Exams for lab. But lab questions will be included in SEE.
4. Any damage / loss of PC components will be viewed seriously.
5. No separate practice labs will be provided.

Method of Evaluation:

1. **Lab Record:** for submitting a record with complete write up on time – 5 Marks (for each lab)
2. **During Lab Session:** For Execution of at least 2 programmes from exercise problems with proper solutions (for each lab).
3, **Lab CIE:** Two CIE's to be conducted each for 10 marks. (First CIE after 5 labs and second CIE at the end of 10th lab).
4. **Eligibility:** Minimum 40%(8/20) of marks to be maintained to take up Theory SEE.

LAB	MARKS		FINAL
CIE 1	10	Average of CIE 1 and CIE 2	10
CIE 2	10		
RECORD	5 X 10	50	10
EXECUTION IN LAB	5 X 10	50	
		TOTAL	**20**

SYLLABUS

SL. No.	Topic
Session-1	Introduction to Python: Installation of packages and Modules, Variables, Lists, Tuples, Strings and Dictionaries.
Session-2	Control statements and Looping statements, Introduction to Numpy, Sympy and Matplotlib.
LAB-1	2D plots for Cartesian and polar curves.
LAB-2	Finding angle between polar curves, curvature and radius of curvature of a given curve.
LAB-3	Finding partial derivatives and Jacobian of multivariable functions.
LAB-4	Applications to Maxima and Minima of two variables.
LAB-5	Solving the system of linear equations using Gauss-Elimination and Gauss -Seidel Method.
LAB-6	Compute eigenvalues and eigenvectors and find the largest and smallest eigenvalue by Rayleigh power method.
LAB-7	Solving the first and second order differential equations with initial/boundary conditions and visualizing their solutions.
LAB-8 (MECH)	Solution of a differential equation of oscillations of a spring with different loads.
LAB-9 (CIVIL)	Solution of a differential equation of deflection of a beam with different loads.
LAB-10 (EEE)	Solving the differential equations of electrical circuits – RC, LR and LCR.
LAB-11 (CSE)	Finding GCD using Euclid's Algorithm and solving linear Congruence.

Contents

Introduction to Python:

Please refer to the following "Link"

Installation of Python:

Please refer to the following "Link"

Basics of Python:

Please refer to the following "Link"

Programming Structures in Python:

Please refer to the following "Link"

LAB: 1

Plotting Cartesian and Polar Curves

Objectives:
In this lab students will learn

1. to plot the given cartesian equation and identify shape of the curves.
2. to plot polar curves equation and identify shape of the curves.
3. to plot the given implicit functions and identify the curves.

1.1 Syntax used in this Lab:

plot(x, y): plot x and y using default line style and color

1.1.1 Customizations:

- plot(x, y, 'bo'): plot x and y using blue circle markers
- plot(y): plot y using x as index array 0..N-1
- plot(y, 'r+'): ditto, but with red plusses
- plot(x, y, 'go–', linewidth=2, markersize=12)
- plot(x,y,color='green',marker='o',linestyle='dashed',linewidth=2, markersize=12)

1.1.2 Plotting points(Scattered plot):

scatter(x-axis-data,y-axis-data,s=None,c=None,marker=None,cmap=None, vmin=None,vmax=None,alpha=None,linewidths=None,edgecolors=None)

1.1.3 Implicit Function

plot_ implicit(expr,x-var=None,y-var=None,adaptive=True,depth=0, points=300,line-color='blue',show=True,**kwargs)

1.1.4 Parametric Function:

1. Plotting a single parametric curve with a range
   plot_parametric((expr_ x, expr_ y), range)
2. Plotting multiple parametric curves with the same range
   plot_parametric((expr_ x, expr_ y), ..., range)
3. Plotting multiple parametric curves with different ranges
   plot_parametric((expr_ x, expr_ y, range), ...)

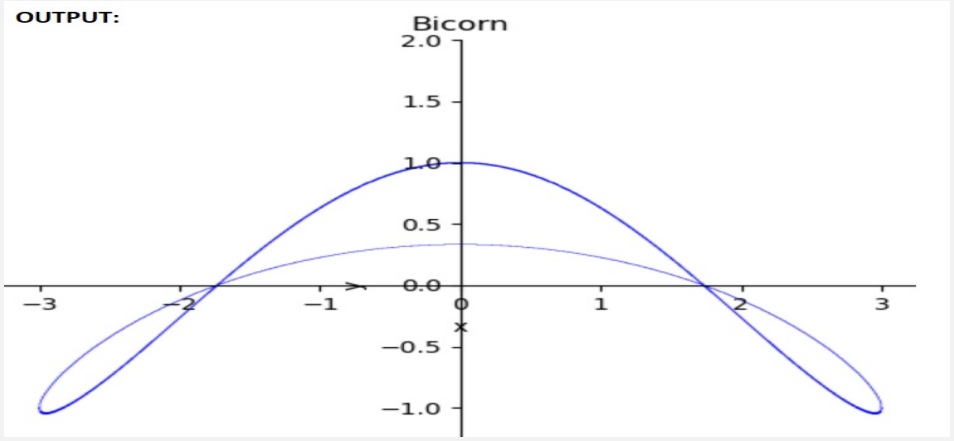
1.2 Sample Programs:

1.2.1 Implicit Functions:

Example-1: Write a Program to plot the implicit function $y^2(a^2 - x^2) = (x^2 + 2ay - a)^2$ with a=2

```
from sympy import plot_implicit, symbols, Eq
x, y = symbols('x y')
a=2
A=Eq(y**2*(a**2-x**2), (x**2+2*a*y-a)**2)
print("Output:")
p2 = plot_implicit(A,(x, -4, 4), (y, -2, 2),title= 'Bicorn')
```

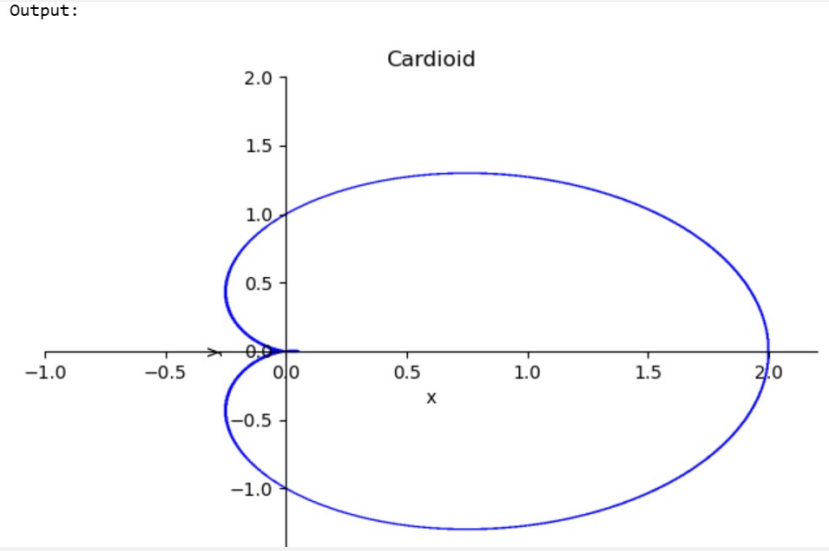
OUTPUT:



Example-2: Write a Program to plot the implicit function $(x^2 + y^2 - 2ax)^2 = 4a^2(x^2 + y^2)$ with $a = 0.5$.

```
from sympy import plot_implicit, symbols, Eq
x, y = symbols('x y')
a=0.5
print("Output:")
A=Eq((x**2+y**2-2*a*x)**2,4*a**2*(x**2+y**2))
p2 = plot_implicit(A,(x, -1, 2.2), (y, -2, 2),title= 'Cardioid') # a=0.5
```

```
Output:
```

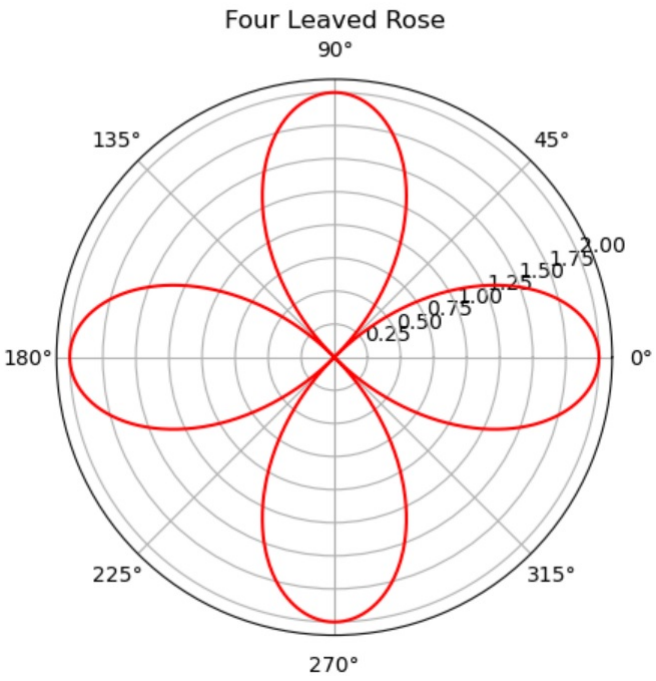


1.2.2 Polar Curves:

Example-3: Write a Program to plot the polar curve four leaved Rose: $r = 2|cos2\theta|$

```
from pylab import *
theta=linspace(0,2*pi,1000)
r=2*abs(cos(2*theta))
polar(theta,r,'r')
title("Four Leaved Rose")
print("Output:")
show()
```

Output:

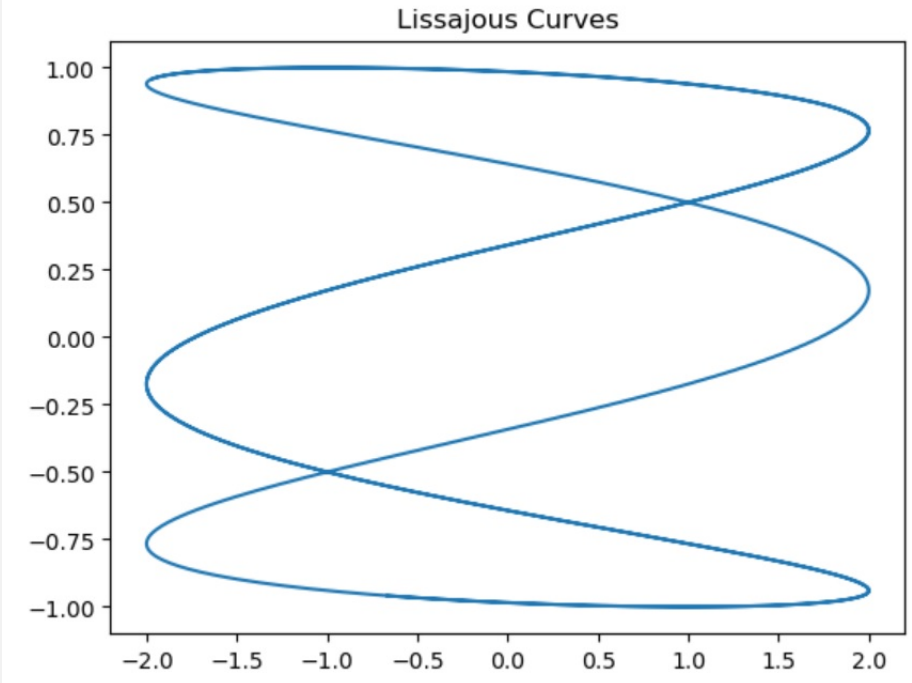


1.2.3 Parametric Curves:

Example-4: Write a Program to plot the parametric curve(Lissajous Curves) $x = asin(n\theta + c); y = bsin(\theta)$

```
def Lissajous(a,b,c,n):# User defined function "Lissajous"
  x = [] #create the list of x coordinates
  y = [] #create the list of y coordinates
  print("Output:")
  for theta in np.linspace(-5, 5, 1000):#loop over a vector theta
    x.append(a*(np.sin(n*theta+c)))#creat the list x w.r.t. theta
    y.append(b*(np.sin(theta))) #creat the list y w.r.t. theta
  plt.plot(x,y)  #plot using matplotlib.piplot
  plt.title("Lissajous Curves")
  plt.show()  #show the plot
Lissajous(2,1,np.pi/3,3) #call the function
```

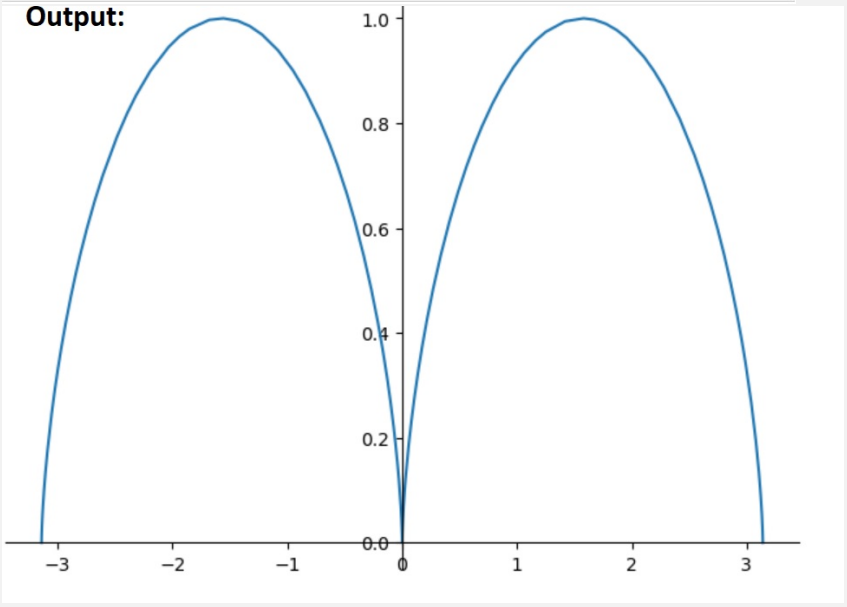
```
Output:
```



Example-5: Write a Program to plot the parametric curve Cardioid $x = a(t - sin(t))$ and $y = a(1 - cos(t))$ with $a = 0.5$.

```
from sympy import *
from sympy.plotting import plot_parametric
t,a=Symbol('t'),0.5
x = a*(t - sin(t))
y = a*(1 - cos(t))
plot_parametric(x, y, (t, -2*pi, 2*pi))
show()
```

Output:



1.3 Activity-1:Cartesian Curves:

Exercise: Write the program to Plot the following curves:

1. Catenary :$y = acosh(x/a)$
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. Circle $x^2 + y^2 = a$
5. Astroid: $x^{2/3} + y^{2/3} = a^{2/3}, a > 0$
6. Strophoid: $y^2(a - x) = x^2(a + x), a > 0$
7. Cissiod: $y^2(a - x) = x^3, a > 0$
8. Lemniscate: $a^2y^2 = x^2(a^2 - x^2)$
9. Folium of Descartes: $x^3 + y^3 = 3axy$

1.4 Activity-2:Polar Curves:

Exercise: Write the program to Plot the following curves:

1. Spiral of Archimedes: $r = a + b\theta$
2. Limacon: $r = a + bcos\theta$
3. Cardioid: $r = a + acos(\theta)$
4. Ellipse: $r = \frac{ab}{\sqrt{asin^2(\theta)+bcos^2(\theta)}}$
5. Circle $r = a$ for $a = 1, 2, 3$
6. Lituus :$r^2 = \frac{a^2}{\theta}$
7. Cardioid: $r = a - asin(\theta)$
8. Cochleoid:$r = \frac{asin(\theta)}{\theta}$

1.5 Activity-3:Parametric Curves:

Exercise: Write the program to Plot the following curves:

1. Circle: $x = acos(\theta); y = asin(\theta)$
2. Folium of Descartes:$x(t) = \frac{3at}{1+t^3}$;$y(t) = \frac{3at^2}{1+t^3}$
3. Circle: $x = 5cos(\theta)$; $y = 5sin(\theta)$
4. Involute of a Circle:$x = a(cost + tsint)$;$y(t) = a(sint - tcost)$
5. Cardioid $x = a(2cos(t) - cos(2t))$;$y = a(2sin(t) - sin(2t))$
6. Nephroid:$x(t) = a(3cost - cos(3t))$;$y(t) = a(3sint - sin(3t))$
7. Tractrix: $x = \frac{1}{cosht}$;$y = t - tanht$

Observations:

LAB: 2

Angle between polar curves and Radius of curvature

Objectives:
In this lab students will learn

1. to find the derivative of given function using **sympy**.
2. to find angle between two polar curves.
3. to find radius of curvature of cartesian and polar curves.

2.1 Angle between radius vector and tangent:

The angle between radius vector and tangent of a polar curve $r = f(\theta)$ is given by $tan\phi = r\frac{d\theta}{dr}$.

2.1.1 Angle between two polar curves:

If ϕ_1 and ϕ_2 are angle between radius vector and tangent of two polar curves then $\alpha = |\phi_1 - \phi_2|$ is the angle between two curves at the point of intersection.

2.2 Radius of curvature

Formula to calculate Radius of curvature in cartesian form is $\rho = \frac{(1+y_1^2)^{3/2}}{y_2}$

Formula to calculate Radius of curvature in polar form is $\rho = \frac{(r^2+r_1^2)^{3/2}}{r^2+2r_1^2-rr_2}$

The Radius of curvature of Parametric curves is $\rho = \frac{(x'^2+y'^2)^{\frac{3}{2}}}{y''x'-x''y'}$, where $x' = \frac{dx}{dt}, x'' = \frac{d^2x}{dt^2}, y' = \frac{dy}{dt}, y'' = \frac{d^2y}{dt^2}$.

2.3 Syntax Used in this Lab:

- **diff(function,variable):**-Returns the derivative of **function** w.r.t the **variable**. Here both expression and variable are symbols(it is called from **sympy**.
- **idiff(Equation,variable_1,variable_2,order):**-Return the derivative of implicit **Equation** of two variables w.r.t. first variable of given **order**.
- **subs():** math_expression.subs(variable, substitute)-Returns value of the **expression** by substituting the **variable=substitute**.
- **Derivative(expression, reference variable):**-Returns the derivative of the **expression** w.r.t to **reference variable**. Here both expression and variables are symbols(it is called from **sympy**.
- **doit(x):**-Returns the evaluated object
- **simplify():** simplify(expression)-Returns the simplified form of the **expression**.
- **display(expression):**-It is an output statement displays the mathematical**expression**.

2.4 Sample Programs:

2.4.1 Angle between Polar Curves:

Example-1:Find the angle between the curves $r = a(1 - cost)$ and $r = b(1 - sint)$ with $a = 3$ and $b = 4$.

```
from sympy import *
import math as mt
r,t =symbols('r,t')
a,b=3,4
r1,r2=a*(1-cos(t)),b*(1-sin(t));
dr1,dr2=diff(r1,t),diff(r2,t)
t1,t2=r1/dr1,r2/dr2
q=solve(r1-r2,t)
w1=t1.subs({t:float(q[1])})
w2=t2.subs({t:float(q[1])})
y1=atan(w1)
y2=atan(w2)
w=abs(y1-y2)
print('Angle between curves in radians is %0.3f'%(w))
print("Angle in degrees",mt.degrees(w))
```

```
Angle between curves in radians is 2.356
Angle in degrees 135.0
```

2.4.2 Radius of curvature in Cartesian curves:

Example-2:(Explicit function) Find the radius of curvature for $y = 4sin(x) - sin(2x)$ at $x = \pi/2$.

```
from sympy import *
x,y=symbols('x,y')
Y=4*sin(x)-sin(2*x)
Y1=diff(Y,x)
dY=Y1.subs(x,pi/2)
Y2=diff(Y1,x)
DY=Y2.subs(x,pi/2)
rho=(1+dY**2)**1.5/(DY);
print("The radius of curvature is")
display(simplify(abs(rho)))
```

Output:

```
The radius of curvature is
```

2.79508497187474

2.4.3 Radius of curvature in Cartesian curves:

Example-3:(Implicit function) Find the radius of curvature for $x^3+y^3=3axy$ at $\left[\frac{3a}{2},\frac{3a}{2}\right]$.

```
from sympy import *
x,y,a=symbols('x,y,a')
Y=x**3+y**3-3*a*x*y
P=[3*a/2,3*a/2]
Y1=idiff(Y,y,x)
dY=lambdify((x,y),Y1)
dY=simplify(dY(P[0],P[1]))
X1=idiff(Y,x,y)
dX=lambdify((x,y),X1)
dX=simplify(dX(P[0],P[1]))
if dX==0:
    X2=idiff(Y,x,y,2)
    rho=simplify((1+dX**2)**1.5/(X2));
else:
    Y2=idiff(Y,y,x,2)
    rho=simplify((1+dY**2)**1.5/(Y2));
rho1=lambdify((x,y),rho)
rho2=rho1(P[0],P[1])
print("The radius of curvature is:")
display(abs(rho2))
```

Output:

```
The radius of curvature is:
```

$$0.265165042944954\,|a|$$

2.4.4 Radius of Curvature of Polar Curves:

Example-4: Find the radius of curvature for $r = acos(nt)$ at $t = \pi/2$ and $n = 1$.

```
from sympy import *
t,r,a,n=symbols('t r a n')
r=a*cos(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)
rho1=rho1.subs(n,1)
print("The radius of curvature is")
display(simplify(rho1))
```

```
The radius of curvature is
```

$$\frac{\left(a^2\right)^{1.5}}{2a^2}$$

2.4.5 Radius of Curvature of Parametric Curves:

Example-5:Find the radius of curvature for $x = a^{\frac{3}{2}}cos^{\frac{3}{2}}(t)$ and $y = a^{\frac{3}{2}}sin^{\frac{3}{2}}(t)$ at $t = \pi/2$ and $n = 1$.

```
from sympy import *
from sympy.abc import rho, x,y,r,K,t,a,b,c,alpha
y=(a*sin(t))**(3/2)
x=(a*cos(t))**(3/2)
X=simplify(Derivative(y,t).doit())
Y=simplify(Derivative(x,t).doit())
dydx=X/Y
Z=Derivative(dydx,t).doit()
W=Derivative(x,t).doit()
rho=simplify((1+dydx**2)**1.5/(Z/W))
print('Radius of curvature is')
display(ratsimp(rho))
t1=pi/4
r1=1;
rho1=rho.subs(t,t1);
rho2=rho1.subs(a,r1);
display('Radius of curvature at r=1 and t=pi/4 is',simplify(rho2));
curvature=1/rho2;
print('\n\n Curvature at (1,pi/4) is',float(curvature))
```

OUTPUT:

```
Radius of curvature is
```

$$-\frac{3.0(a\cos(t))^{3.0}\left(\frac{(a\sin(t))^{3.0}}{(a\cos(t))^{3.0}\tan^4(t)}+1\right)^{1.5}\sin^2(t)\tan^2(t)}{(a\sin(t))^{1.5}}$$

```
'Radius of curvature at r=1 and t=pi/4 is'
```

$$-2.52268924576114$$

```
 Curvature at (1,pi/4) is -0.39640237166757364
```

2.5 Activity-1:

1. Find the angle between radius vector and tangent to the following polar curves and hence find the angle between them:
   a) $r = a\theta$ and $r = \frac{a}{\theta}$,
   b) $r = 2sin(\theta)$ and $r = 2cos(\theta)$
   c) $r = 2(1 + sin(\theta))$ and $r = 2(1 + cos(\theta))$

2. Find the angle between the following polar curves:
   a) $r = e^{a\theta}$ and $r = e^{-a\theta}$,
   b) $r = 4sin^2(\theta)$ and $r = 8sin(\theta)$
   c) $r^2 = 4sin(2 * \theta)$ and $r^2 = 4cos(2 * \theta)$

2.6 Activity-2:

1. Find the radius of curvature of $r = 2(1 - cos(t))$ at $t = \frac{\pi}{2}$.
2. Find the radius of curvature of $r = acos(t)$ at $t = \frac{\pi}{4}$.
3. Find the radius of curvature of $r = asin(2t)$ at $t = \frac{\pi}{4}$.
4. Find radius of curvature of $x = acos^3(t)$, $y = asin^3(t)$ at $t = 0$.
5. Find the radius of curvature of $x = a(t - sin(t))$ and$y = a(1 - cos(t))$ at $t = \pi$.
6. Find the radius of curvature of $x(t) = a(3cost - cos(3t))$;$y(t) = a(3sint - sin(3t))$at $t = \frac{\pi}{2}$.
7. Find the radius of curvature of $x^2 + y^2 = 25$ at point $(3, 4)$.
8. Find the radius of curvature of $y = ccosh\left(\frac{x}{c}\right)$ at the point $(0, c)$.
9. Find the radius of curvature of $x^2 = \frac{a(x^2+y^2)}{y}$ at the point $(-2a, 2a)$.
10. Find the radius of curvature of $y^2x = 4(2 + x)$at $* - 2, 1)$.

Observations:

LAB: 3

Partial Derivatives and Jacobian

Objectives:
In this lab will learn

1. to find partial derivatives of multi-variable functions.
2. to find the Jacobian of functions of two and three variables.

3.1 Partial derivatives:

- diff(u,x)= $\frac{\partial u}{\partial x}$
- diff(u,y)= $\frac{\partial u}{\partial y}$
- diff(u,x,x)= $\frac{\partial^2 u}{\partial x^2}$
- diff(u,x,y)=$\frac{\partial^2 u}{\partial x \partial y}$

3.2 Jacobian:

Let $x = g(u, v)$ and $y = h(u, v)$ be a transformation of the plane. Then the Jacobian of this transformation is $\mathbf{J} = \frac{\partial(x,y)}{\partial(u,v)} = \begin{vmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \end{vmatrix}$.

3.3 Syntax Used in Lab:

1. **Matrix():**
   Matrix([[row1],[row2],[row3]....[rown]])
   A 3 by 3 matrix can be defined as
   Matrix([[a11,a12,a13],[a21,a22,a23],[a31, a32 a33]])

2. **Determinant():** or **det():**
   Determinant(M)
   M: is input matrix

3.4 Sample Programs:

3.4.1 Partial Derivatives:

Example-1: Find the partial derivatives U_x and U_y and prove that mixed partial derivatives , $U_{xy} = U_{yx}$ for $U = e^x(xcos(y) - ysin(y))$.

```
from sympy import *
x,y=symbols('x,y')
U=exp(x)*(x*cos(y)-y*sin(y))
dux,duy=diff(U,x),diff(U,y)
duxy,duyx=diff(dux,y),diff(duy,x)  # or duyx=diff(u,y,x)
print("Output:")
print("Ux=",dux)
print("Uy=",duy)
if duxy==duyx:
    print('Mixed partial derivatives are equal')
else:
    print('Mixed partial derivatives are not equal')
```

```
Output:
Ux= (x*cos(y) - y*sin(y))*exp(x) + exp(x)*cos(y)
Uy= (-x*sin(y) - y*cos(y) - sin(y))*exp(x)
Mixed partial derivatives are equal
```

Example-2: If $u = tan^{-1}\left(\frac{2xy}{x^2-y^2}\right)$ then prove that $u_{xx} + u_{yy} = 0$.

```
from sympy import *
x,y=symbols('x,y')
u=atan((2*x*y)/(x**2-y**2))
#display(u)
dux=diff(u,x)
duy=diff(u,y)
uxx=diff(dux,x) # or uxx=diff(u,x,x)
uyy=diff(duy,y) # or uyy=diff(u,y,y)
w=uxx+uyy
#display(w)
w1=simplify(w)
print("Output:")
print('Uxx+Uyy=:',float(w1))
```

```
Output:
Uxx+Uyy=: 0.0
```

3.4.2 Jacobian:

Example-3: If $u = \frac{xy}{z}, v = \frac{yz}{x}, w = \frac{zx}{y}$ then prove that $J = 4$.

```
from  sympy import *
init_printing()
x,y,z=symbols('x,y,z')
u,v,w=x*y/z,y*z/x,z*x/y
dux,duy,duz=diff(u,x),diff(u,y),diff(u,z)
dvx,dvy,dvz=diff(v,x),diff(v,y),diff(v,z)
dwx,dwy,dwz=diff(w,x),diff(w,y),diff(w,z)
J=Matrix([[dux,duy,duz],[dvx,dvy,dvz],[dwx,dwy,dwz]]);
display("The Jacobian matrix is:",J)
Jac=det(J).doit()
print('J = ', Jac)
```

```
'The Jacobian matrix is:'
```

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

```
J =  4
```

Example-4: If $X = \rho cos(\phi)sin(\theta)$, $Y = \rho cos(\phi)cos(\theta)$, $Z = \rho sin(\phi)$ **then find** $\frac{\partial(X,Y,Z)}{\partial(\rho,\phi,\theta)}$.

```
from sympy.abc import rho, phi, theta
X,Y,Z=rho*cos(phi)*sin(theta),rho*cos(phi)*cos(theta),rho*sin(phi);
dx,dy=Derivative(X,rho).doit(),Derivative(Y,rho).doit()
dz=Derivative(Z,rho).doit()
dx1,dy1=Derivative(X,phi).doit(),Derivative(Y,phi).doit();
dz1=Derivative(Z,phi).doit()
dx2,dy2=Derivative(X,theta).doit(),Derivative(Y,theta).doit();
dz2=Derivative(Z,theta).doit();
J=Matrix([[dx,dy,dz],[dx1,dy1,dz1],[dx2,dy2,dz2]]);
display('The Jacobian matrix is:',J)
print("J=",simplify(Determinant(J).doit()))
```

```
'The Jacobian matrix is:'
```

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

```
J= rho**2*cos(phi)
```

3.5 Activity-1:

1. If $u = tan^{-1}(y/x)$ verify that $\frac{\partial^2 u}{\partial y \partial x} = \frac{\partial^2 u}{\partial x \partial y}$.
2. If $u = log(\frac{x^2+y^2}{x+y})$ then show that $xu_x + yu_y = 1$.
3. If $u = tan^{-1}\left(\frac{2xy}{x^2-y^2}\right)$then show that $u_{xx} + u_{yy} = 0$.
4. If $w = x^2y + y^2z + z^2x$ then show that $w_x + w_y + w_z = (x + y + z)^2$.
5. If $u = \frac{y}{z} + \frac{z}{x}$ show that $xu_x + yu_y + zu_z = 0$.

3.6 Activity-2:

1. If $x = u - v, y = v - uvw$ and $z = uvw$ find Jacobian of x, y, z w.r.t u, v, w.
2. If $x = rcos(t)$ and $y = rsin(t)$ then find the $\frac{\partial(x,y)}{\partial(r,t)}$.
3. If $x = u + v + w$, $y = uv + vw + wu$ and $z = uvw$ find $\frac{\partial(x,y,z)}{\partial(u,v,w)}$.
4. If $u = \frac{2yz}{x}$, $v = \frac{3zx}{y}$ and $w = \frac{4xy}{z}$ then find $\frac{\partial(u,v,w)}{\partial(x,y,z)}$
5. If $u = x + \frac{y^2}{x}$ and $v = \frac{y^2}{x}$ then find the $\frac{\partial(u,v)}{\partial(x,y)}$.

Observations:

LAB: 4

Extrema of function of two variables:

Objectives: In this lab students will learn to find the maxima and minima of function of two variables.

4.1 Syntax:

1. **sympy.solve(expression):**Returns the solution to a mathematical expression/polynomial.
2. **sympy.evalf():**Returns the evaluated mathematical expression.
3. **sympy.lambdify(variable, expression, library):**Converts a SymPy expression to an expression that can be numerically evaluated. lambdify acts like a lambda function, except it, converts the SymPy names to the names of the given numerical library, usually NumPy or math.

4.2 Sample Programme:

Example-1: Find the maxima and minima of $f(x,y) = x^3 + 3xy^2 - 15x^2 - 15y^2 + 72x$.

```
import sympy as sym
x,y,a=sym.symbols('x,y,a')
f=x**3+3*x*y**2-15*x**2-15*y**2+72*x
dx1 = sym.Derivative(f,x).doit()
dy1 = sym.Derivative(f,y).doit()
eq1,eq2 = sym.Eq(dx1,0),sym.Eq(dy1,0)
sol = sym.solve([eq1,eq2],(x,y))
print("The critical points are",sol)
dx2 = sym.Derivative(f,x,2).doit()
dxy = sym.Derivative(dx1,y).doit()
dy2 = sym.Derivative(f,y,2).doit()
for i,j in sol:
    A = dx2.subs({x:i,y:j}).evalf()
    B = dxy.subs({x:i,y:j}).evalf()
    C = dy2.subs({x:i,y:j}).evalf()
    dis = A*C - B*B
    if(dis>0 and A>0):
        print("The function has minimum at (",i,",",j,")")
        q1=f.subs({x:i,y:j}).evalf()
        print("Minimum value:",q1)
    elif(dis>0 and A<0):
        print("The function has maximum at (",i,",",j,")")
        q2=f.subs({x:i,y:j}).evalf()
        print("Maximum value:",q2)
    elif(dis<0):
        print("(",i,",",j,") is a saddle point")
    elif(dis==0):
        print("it needs furthur investigation.")
```

Output:

```
The critical points are [(4, 0), (5, -1), (5, 1), (6, 0)]
The function has maximum at ( 4 , 0 )
Maximum value: 112.000000000000
( 5 , -1 ) is a saddle point
( 5 , 1 ) is a saddle point
The function has minimum at ( 6 , 0 )
Minimum value: 108.000000000000
```

4.3 Activity:

1. Find the maxima and minima of $x^2 + 2y^2 - 2xy - x - y - 2$
2. Find the maxima and minima of $x^3 + y^3 - 3y - 12x + 20$
3. Find the maxima and minima of $x^2 + y^2 + 6x + 12$
4. Find the maxima and minima of $x^4 + y^4 - y^2 - x^2 + 1$
5. Find the maxima and minima of $xy + \frac{a^3}{x} + \frac{a^3}{y}$
6. Find numbers such that sum of three numbers is constant and their product is maximum.
7. Find the shortest distance from the origin to the surface xyz^2.
8. Let $T = 400xyz^2$ is temperature on the surface of sphere $x^2+y^2+z^2 = 4$. Find the point of maximum temperature.

Observations:

LAB: 5

Solving system of linear equation $A\overline{x} = b$

Objectives:
In this lab students will learn

1. to test for consistency of the system $A\overline{x} = b$ and represent the solution graphically.
2. to check whether the given system is diagonally dominant or not.
3. to find the solution of the system $A\overline{x} = b$ using Gauss-Elimination and Gauss-Seidel Methods.

5.1 Syntax for the commands used:

- **numpy.matrix(data, dtype = None):**Returns a matrix from an array-like object, or from a string of data. A matrix is a specialized 2-D array that retains its 2-D nature through operations.
- **numpy.linalg.matrix_rank(A):**Return rank of the array.
- **numpy.shape(A):**Returns the shape of an array.
- **sympy.Matrix():**Creates a matrix.

5.2 Solution of system of equations:

5.2.1 System of homogeneous linear equations:

The linear system of equations of the form $AX = 0$ is called system of homogeneous linear system of equations. The n-tuple $(0, 0, \ldots, 0)$ 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.

5.3 Sample Programme:

Example-1: Check whether the following system of homogeneous linear equations $x_1 + 2x_2 - x_3 = 0$, $2x_1 + x_2 + 4x_3 = 0$ and $3x_1 + 3x_2 + 4x_3 = 0$ has non-trivial solution.

```
import numpy as np
A=np.matrix([[1,2,-1],[-2,1,4],[3,-3,10]])
B=np.matrix([[0],[0],[0]])
r=np.linalg.matrix_rank(A)
n=A.shape[1]
if (r==n):
    print("System has trivial solution")
else:
    print("System has non-trivial solution")
```

Output:

```
System has trivial solution
```

5.4 Non-homogeneous system of Linear Equations:

The linear system of equations of the form $AX = B$ is called system of non-homogeneous 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, $\rho(A) = n$
- has infinitely many solutions, $\rho(A) < n$
- system is inconsistent $\rho(A) \neq \rho([A|B])$.

5.5 Sample Programme:

Example-2: Examine the consistency of the following system of equations $x_1 + 2x_2 - x_3 = 1$, $2x_1 + x_2 + 4x_3 = 2$ and $3x_1 + 3x_2 + 4x_3 = 1$.

```
A=np.matrix([[1,2,-1],[2,1,4],[3,3,4]])
B=np.matrix([[1],[2],[1]])
AB=np.concatenate((A,B), axis=1)
rA=np.linalg.matrix_rank(A)
rAB=np.linalg.matrix_rank(AB)
n=A.shape[1]
if (rA==rAB):
    if (rA==n):
        print("The system has unique solution")
    else:
        print("The system has infinitely many solutions")
else:
    print("The system of equations is inconsistent")
```

Output:

```
The system has unique solution
```

Example-2: Examine the consistency of the following system of equations $x_1 + 2x_2 - x_3 = 1$, $2x_1 + x_2 + 4x_3 = 2$ and $3x_1 + 3x_2 + 4x_3 = 1$.

```
import sympy as sp
x, y, z=sp.symbols('x y z')
A=sp.Matrix([[1,2,-1],[2,1,5],[3,3,4]])
B=sp.Matrix([[1],[2],[1]])
AB=A.col_insert(A.shape[1],B)
rA=A.rank()
rAB=AB.rank()
n=A.shape[1]
print("The coefficient matrix is")
sp.pprint(A)
print(f"The rank of the coefficient matrix is {rA}")
print("The augmented matrix is")
sp.pprint(AB)
print(f"The rank of the augmented matrix is {rAB}")
print(f"The number of unkowns are {n}")
if (rA==rAB):
    if (rA==n):
        print("The system has unique solution")
    else:
        print("The system has infinitely many solutions")
    print(sp.solve_linear_system(AB,x,y,z))
else:
    print("The system of equations is inconsistent")
```

Output:

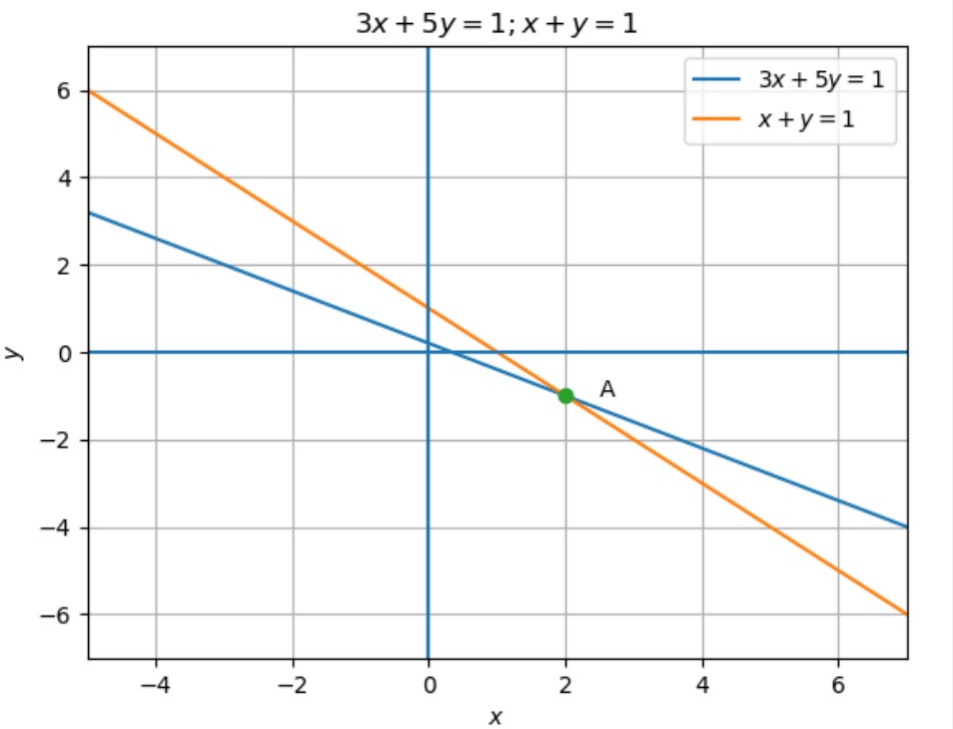
```
The coefficient matrix is
⎡1  2  -1⎤
⎢        ⎥
⎢2  1  5 ⎥
⎢        ⎥
⎣3  3  4 ⎦
The rank of the coefficient matrix is 2
The augmented matrix is
⎡1  2  -1  1⎤
⎢           ⎥
⎢2  1  5   2⎥
⎢           ⎥
⎣3  3  4   1⎦
The rank of the augmented matrix is 3
The number of unkowns are 3
```

5.6 Graphical representation of solution:

Example-4: Obtain the solution of $3x + 5y = 1, x + y = 1$ graphically.

```
from sympy import *
import numpy as np
import matplotlib.pyplot as plt
x,y=symbols('x,y')
sol=solve([3*x+5*y-1,x+y-1],[x,y])
p,q=sol[x],sol[y]
print('Point of intersection is A (', p ,',', q, ')\n')
x = np.arange(-10, 10, 0.001)
y1,y2 = (1-3*x)/5,1-x
plt.plot(x,y1,x,y2)
plt.plot(p,q,marker = 'o')
plt.annotate('A', xy=(p,q), xytext=(p+0.5, q))
plt.xlim(-5,7),plt.ylim(-7,7)
plt.axhline(y=0),plt.axvline(x=0)
plt.title("$3x+5y=1; x+y=1$")
plt.xlabel("Values of x"),plt.ylabel("Values of y ")
plt.legend(['$3x+5y=1$', '$x+y=1$'])
plt.grid(),plt.show()
```

Output:



5.7 Gauss-Seidel Method:

Gauss Seidel method is an iterative method to solve system of linear equations. The method works if the system is diagonally dominant. That is $|a_{ii}| \geq \sum_{i \neq j} |a_{ij}|$ for all i.

5.7.1 Sample Programme:

Example-1: Solve the system of equations using Gauss-Seidel method: $10x + y - 2z = 14; 3x + 10y - z = -13; 2x - y + 10z = 17$.

```
f1 = lambda x,y,z: (14-y+2*z)/10
f2 = lambda x,y,z: (-13-3*x+z)/10
f3 = lambda x,y,z: (17-2*x+y)/10
x0,y0,z0,count,condition = 0,0,0,1,True
e = float(input('Enter tolerable error: '))
print('\nCount\tx\ty\tz\n')
while condition:
    x1 = f1(x0,y0,z0)
    y1 = f2(x1,y0,z0)
    z1 = f3(x1,y1,z0)
    print('%d\t%0.4f\t%0.4f\t%0.4f\n' %(count, x1,y1,z1))
    e1 = abs(x0-x1);
    e2 = abs(y0-y1);
    e3 = abs(z0-z1);
    count += 1
    x0,y0,z0 = x1,y1,z1
    condition = e1>e and e2>e and e3>e
print('\nSolution: x=%0.3f, y=%0.3f and z = %0.3f\n'% (x1,y1,z1))
```

Output:

```
Enter tolerable error: 0.0001

Count   x       y       z

1       1.4000  -1.7200 1.2480

2       1.8216  -1.7217 1.1635

3       1.8049  -1.7251 1.1665

4       1.8058  -1.7251 1.1663


Solution: x=1.806, y=-1.725 and z = 1.166
```

Example-2: Apply Gauss-Seidel method to solve the system of equations: $20x + y - 2z = 17; 3x + 20y - z = -18; 2x - 3y + 20z = 25$.

```
from numpy import *
def seidel(a, x ,b):
    n = len(a)
    for j in range(0, n):
        d = b[j]
        for i in range(0, n):
            if(j != i):
                d=d-a[j][i] * x[i]
        x[j] = d / a[j][j]
    return x
a=array([[20.0,1.0,-2.0],[ 3.0,20.0,-1.0],[2.0,-3.0,20.0]])
x=array([[0.0],[0.0],[0.0]])
b=array([[17.0],[-18.0],[25.0]])
for i in range(0, 25):
    x = seidel(a, x, b)
print("Solution of given system is:",x)
```

Output:

```
Solution of given system is: [[ 1.][-1.][ 1.]]
```

Example-3: Check whether the given matrix is diagonally dominant or not. $10x + y + 100z = 12; x + 10y + z = 12; x + y + 10z = 12$.

```
from numpy import *
import sys
a=array([[10.0,1.0,100],[ 1.0,10.0,1.0],[1.0,1.0,10.0]])
x=array([[1.0],[0.0],[0.0]])
b=array([[12.0],[12.0],[12.0]])
for i in range(0,len(a)):
  asum=0
  for j in range(0,len(a)):
    if (i!=j):
      asum=asum+abs(a[i][j])

  if(asum<=a[i][i]):
    continue
  else:
    print("The system is not diagonally dominant")
```

Output:

```
The system is not diagonally dominant
```

5.8 Activity:

1. Find the solution of the system of homogeneous equations $x+y+z=0$, $2x+y-3z=0$ and $4x-2y-z=0$.
2. Find the solution of the system of non-homogeneous equations $25x+y+z=27$, $2x+10y-3z=9$ and $4x-2y-12z=-10$.
3. Find the solution of the system of non-homogeneous equations $x+y+z=2$, $2x+2y-2z=4$ and $x-2y-z=5$.
4. Check whether the following system of equations are consistent
   a. $x+y+z=2$, $2x+2y-2z=6$ and $x-2y-z=5$.
   b. $2x+y+z=4$, $4x+2y-2z=8$ and $4x+22y+2z=5$.
5. Check whether the following system are diagonally dominant or not
   a. $25x+y+z=27$, $2x+10y-3z=9$ and $4x-2x-12z=-10$.
   b. $x+y+z=7$, $2x+y-3z=3$ and $4x-2x-z=-1$.
6. Solve the following system of equations using Gauss-Seidel Method
   a. $4x+y+z=6$, $2x+5y-2z=5$ and $x-2x-7z=-8$.
   b. $27x+6y-z=85$, $6x+15y+2z=72$ and $x+y+54z=110$

Observations:

LAB: 6

Eigenvalues and Eigenvectors

Objectives:
In this lab students will learn

1. to find eigen values and corresponding eigen vectors of given matrix.
2. to find dominant eigenvalue and corresponding eigen vector by Rayleigh power method.

6.1 Syntax for the commands used:

- **np.linalg.eig(A):**Compute the eigenvalues and right eigenvectors of a square array
  Returns the following:
  w(. . . , M) array-The eigenvalues, each repeated according to its multiplicity. The eigenvalues are not necessarily ordered. The resulting array will be of complex type, unless the imaginary part is zero in which case it will be cast to a real type. When a is real the resulting eigenvalues will be real (0 imaginary part) or occur in conjugate pairs.
  v(. . . , M, M) array-The normalized (unit 'length') eigenvectors, such that the column $v[:, i]$ is the eigenvector corresponding to the eigenvalue $w[i]$.
- **np.linalg.eigvals(A):**Computes the eigenvalues of a non-symmetric array
- **np.array(parameter):** Creates n-dimensional-array $np.array([[1, 2, 3]])$ is a one-dimensional array and $np.array([[1, 2, 3, 6], [3, 4, 5, 8], [2, 5, 6, 1]])$ is a multi-dimensional array.

- **lambda arguments:expression:** Anonymous function or function without a name. This function can have any number of arguments but only one expression,which is evaluated and returned. They are syntactically restricted to a single expression. Example: $f = lambda\ x : x * *2 - 3 * x + 1$(Mathematically $f(x) = x^2 - 3x + 1$).

- $np.dot(vector_a, vector_b)$:Returns the dot product of vectors a and b.

6.2 Eigenvalues and Eigenvectors:

6.2.1 Sample Programme:

Example-1: Write a programme to find the eigen values and eigen vectors of the matrix $A = \begin{bmatrix} 5 & 1 & 1 \\ 1 & 5 & 1 \\ 1 & 1 & 5 \end{bmatrix}$.

```
import numpy as np
I=np.array([[5,1,1],[1,5,1],[1,1,5]])
print("\n Given matrix: \n", I)
#x=np.linalg.eigvals(I)
w,v = np.linalg.eig(I)
print("\n Eigen values: \n", w)
print("\n Eigen vectors: \n", v)
```

```
Given matrix:
[[5 1 1]
[1 5 1]
[1 1 5]]

Eigen values:
[4. 7. 4.]

Eigen vectors:
[[-0.81649658  0.57735027  0.43072735]
[ 0.40824829  0.57735027 -0.81607614]
[ 0.40824829  0.57735027  0.38534879]]
```

```
## To display one eigen value and correspondingeigen vector

print("Eigen value:\n ", w[0])
print("\n Corresponding Eigen vector :", v[:,0])
```

```
Eigen value:
  8.982056720677654

 Corresponding Eigen vector : [-0.49247712 -0.26523242 -0.82892584]
```

6.3 Rayleigh power method:

For a given Matrix A and a given initial eigen vector X_0, the power method goes as follows:

- Calculate AX_0 and take the largest number say λ_1
- Write the column vector as $AX_0 = \lambda_1 X_1$. At this stage , λ_1 is the approximate eigen value and X_1 will be the corresponding eigen vector.
- Next multiply the Matrix A with X_1 and continue the iterations.
- This method is going to give the dominant eigenvalue of the Matrix and it is given by Rayleigh Quotient $\lambda_{max} = \frac{x^T Ax}{x^T x}$ and x is the corresponding dominant eigenvector.

6.3.1 Sample Programme:

Example-2: Compute the numerically largest eigenvalue of $P = \begin{bmatrix} 25 & -2 & 2 \\ -2 & 3 & -1 \\ 2 & -1 & 0 \end{bmatrix}$ by Rayleigh power method.

```
import numpy as np
def normalize(x):
    fac = abs(x).max()
    x_n = x / x.max()
    return fac, x_n
x = np.array([1, 1,1])
a = np.array([[-25,-2,2 ],
              [-2,3,-1],[2,-1,0]])
sum1=0
for i in range(10):
    x = np.dot(a, x)
    lambda_1, x = normalize(x)
NUM1=np.dot(x,a*x)
DENOM=np.dot(x,x)
n = len(NUM1)
for i in range(n):
    sum1 = sum1 + NUM1[i]
lambda_max=sum1/DENOM
print('Eigenvalue:', lambda_max)
print('Eigenvector:', x)
```

Output:

```
Eigenvalue: -25.288792552861228
Eigenvector: [ 1.          0.06799875 -0.07639753]
```

6.4 Activity:

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 eigen value of the following matrices using power method

   a. $P = \begin{bmatrix} 25 & 1 & 2 \\ 1 & 3 & 0 \\ 2 & 0 & -4 \end{bmatrix}$, Take $X_0 = (1, 0, 1)^T$.

   b. $P = \begin{bmatrix} 6 & 1 & 2 \\ 1 & 10 & -1 \\ 2 & 1 & -4 \end{bmatrix}$, Take $X_0 = (1, 1, 1)^T$.

   c. $P = \begin{bmatrix} 5 & 1 & 1 \\ 1 & 3 & -1 \\ 2 & -1 & -4 \end{bmatrix}$, Take $X_0 = (1, 0, 0)^T$.

   d. $P = \begin{bmatrix} 4 & 1 & 2 \\ 2 & 6 & -1 \\ 2 & -1 & -4 \end{bmatrix}$, Take $X_0 = (1, 1, 1)^T$.

   e. $P = \begin{bmatrix} -2 & -2 & 2 \\ 1 & 1 & -1 \\ 2 & -1 & -4 \end{bmatrix}$,Take $X_0 = (1, 0, -1)^T$.

   f. $P = \begin{bmatrix} 0.5 & 2 & 1 \\ 1 & 10 & -1 \\ 2 & -1 & 4 \end{bmatrix}$,Take $X_0 = (-1, 0, 0)^T$.

   g. $P = \begin{bmatrix} 5 & 1 & 2 \\ 1 & 3 & -1 \\ 2 & -1 & -4 \end{bmatrix}$,Take $X_0 = (1, 0, 1)^T$.

   h. $P = \begin{bmatrix} 7 & 1 & 1 \\ 1 & 7 & 0 \\ 2 & 0 & 7 \end{bmatrix}$, Take $X_0 = (-1, -1, 0)^T$.

3. Find the least eigenvalue of the matrix using power method $P = \begin{bmatrix} 5 & 1 & 1 \\ 1 & 3 & -1 \\ 2 & -1 & -4 \end{bmatrix}$. Take $X_0 = (1, 0, 0)^T$.

Observations:

LAB: 7

Solution of Initial and Boundary value problems:

Objectives:
In this lab students will learn

1. to find the solution of first order differential equation subject to given initial condition.
2. to plot the graph of an analytical solution of the given first order ODE.

7.1 Syntax for the commands used:

1. **dsolve(eq, func=None, hint='default', simplify=True, ics=None, xi=None, eta=None, x0=0, n=6, **kwargs):**Solves ordinary differential equation and system of ordinary differential equations.
3. **linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None, axix=0):** Returns the vector from "start" and "stop" with number of points is equal to "num".

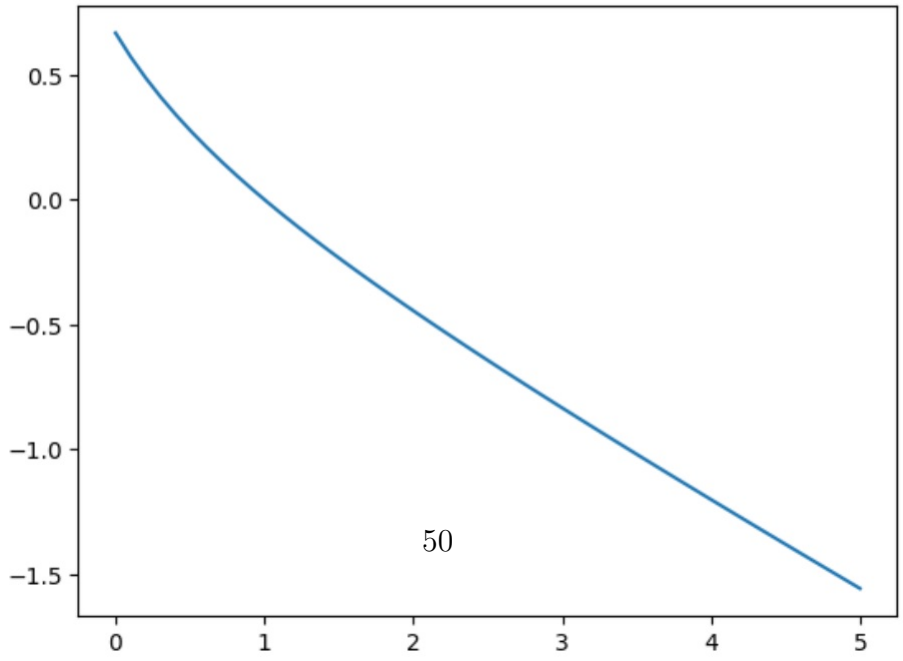
7.2 Sample Programme:

Example-1: Solve the differential equation $y' - \frac{2t}{1-t^2}y = -1$ subject to condition $y(1) = 2$.

```
#from __future__ import division
from sympy import *
from numpy import *
from matplotlib.pyplot import *
x, y, z, t = symbols('x y z t')
eq = Eq(f(t).diff(t), (2*t / (1 - t**2)) * f(t) - 1)
print('ODE class: ', classify_ode(eq)[0])
an_sol = dsolve(eq, hint='1st_linear', ics={f(1): 2})
display(an_sol)
lmbd_sol = lambdify(t, an_sol.rhs)
t_range = linspace(0,5,50)
Y=[lmbd_sol(ti) for ti in t_range]
T=array(t_range)
Z=array(Y)
plot(T,Z)
show()
```

Output: ODE class: factorable

$$f(t) = \frac{-\frac{t^3}{3} + t - \frac{2}{3}}{t^2 - 1}$$

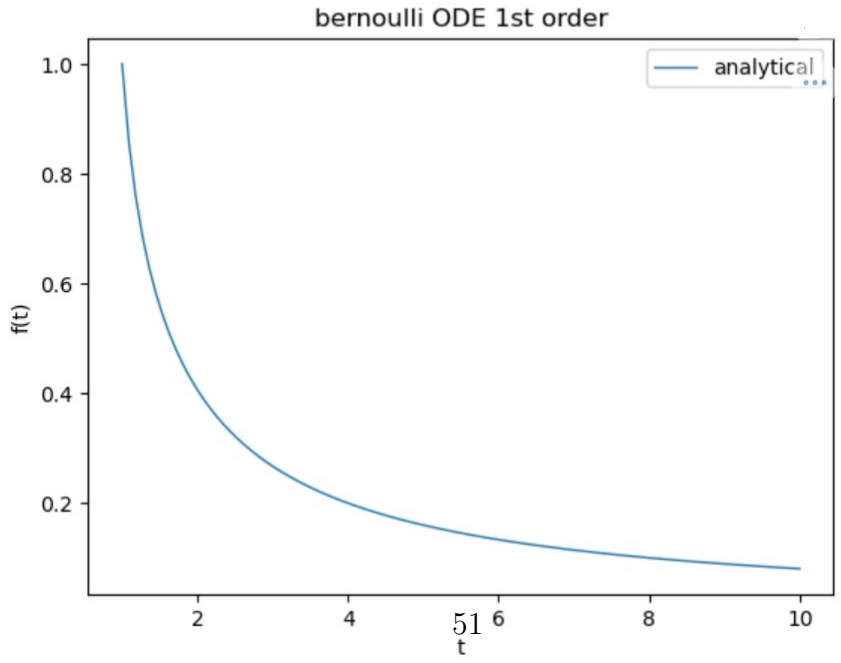


Example-2: Solve the differential equation $y' + \frac{y}{t} = -\frac{y^4}{t}$ subject to condition $y(1) = 1$.

```
import numpy as np
import matplotlib.pyplot as plt
from sympy import *
x, y, z, t = symbols('x y z t')
eq = Eq(f(t).diff(t), -(1/t) * f(t) - (1/t) * f(t)**4)
an_sol = dsolve(eq, ics={f(1): 1})
print('ODE class: ', classify_ode(eq)[0])
display(an_sol)
t_begin,t_end,t_nsamples=1,10,101
t_space = np.linspace(t_begin, t_end, t_nsamples)
lmbd_sol = lambdify(t, an_sol.rhs)
x_an_sol = lmbd_sol(t_space)
plt.figure()
plt.plot(t_space, x_an_sol, linewidth=1, label='analytical')
plt.title('bernoulli ODE 1st order')
plt.xlabel('t'),plt.ylabel('f(t)')
plt.legend(),plt.show()
```

Output: ODE class: factorable

$$f(t) = \frac{2^{\frac{2}{3}} \sqrt[3]{\frac{1}{t^3 - \frac{1}{2}}}}{2}$$

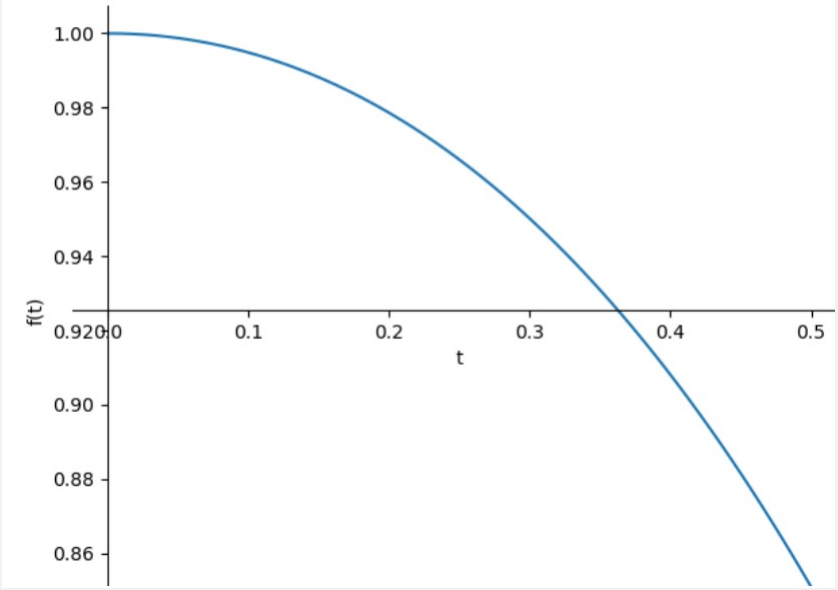


Example-3: Solve $\frac{d^2y}{dt^2} - 2\frac{dy}{dt} + y = sin(t)$ subject to conditions $y(0) = 1$ and $y'(0) = 0$..

```
from sympy import *
from matplotlib.pyplot import show
from sympy.plotting import plot
t = symbols('t')
y = Function('y')(t)
yd = y.diff(t)
ydd = y.diff(t,2)
eqn = Eq(ydd-2*yd+y,sin(t))
ysol=dsolve(eqn,y,ics={y.subs(t,0):1,yd.subs(t,0):0})
an_sol=ysol
display(an_sol)
plot(ysol.rhs,(t,0,0.5))
show()
```

Output:

$$y(t) = \left(\frac{1}{2} - \frac{t}{2}\right)e^t + \frac{\cos(t)}{2}$$



7.3 Activity:

1. Solve $ysinxdx - (1 + y^2 + cos^2x)dy = 0$.
2. Solve $\frac{dy}{dx} = x + y$ subject to condition $y(0) = 2$ and plot the graph of solution.
3. Solve $\frac{dy}{dx} = x^2$ subject to condition $y(0) = 5$ and plot the graph of solution.
4. Solve $y' - y - xe^x = 0$.
5. Solve $x^2y' = ylog(y) - y'$

Observations:

LAB: 8

Solving Higher order differential equations of Mechanical systems:

Objectives: In this lab students will learn

1. to solve the differential equation of a mass-spring-dash-pot system .
2. to represent the solution graphically.

Solution of second order ordinary differential equation and plotting the solution curve

A second order differential equation is defined as $\frac{d^2y}{dx^2} + P(x)\frac{dy}{dx} + Q(x)y = f(x)$ where $P(x)$, $Q(x)$ and $f(x)$ are functions of x.When $f(x) = 0$, the equation is called **homogeneous** second order differential equation.

8.1 Mechanical Systems:

The motion of the spring mass system is given by the differential equation $m\frac{d^2x}{dt^2} + a\frac{dx}{dt} + kx = f(t)$. where, m is the mass of a spring coil,x is the displacement of the mass from its equilibrium position, a is damping constant, k is spring constant.

1. Free and undamped motion - $a = 0, f(t) = 0$.
   Differential Equation : $m\frac{d^2x}{dt^2} + kx = 0$

2. Free and damped motion: $f(t) = 0$
   Differential Equation : $m\frac{d^2x}{dt^2} + a\frac{dx}{dt} + kx = 0$

3. Forced and damped motion:
   Differential Equation : $m\frac{d^2x}{dt^2} + a\frac{dx}{dt} + kx = f(t)$

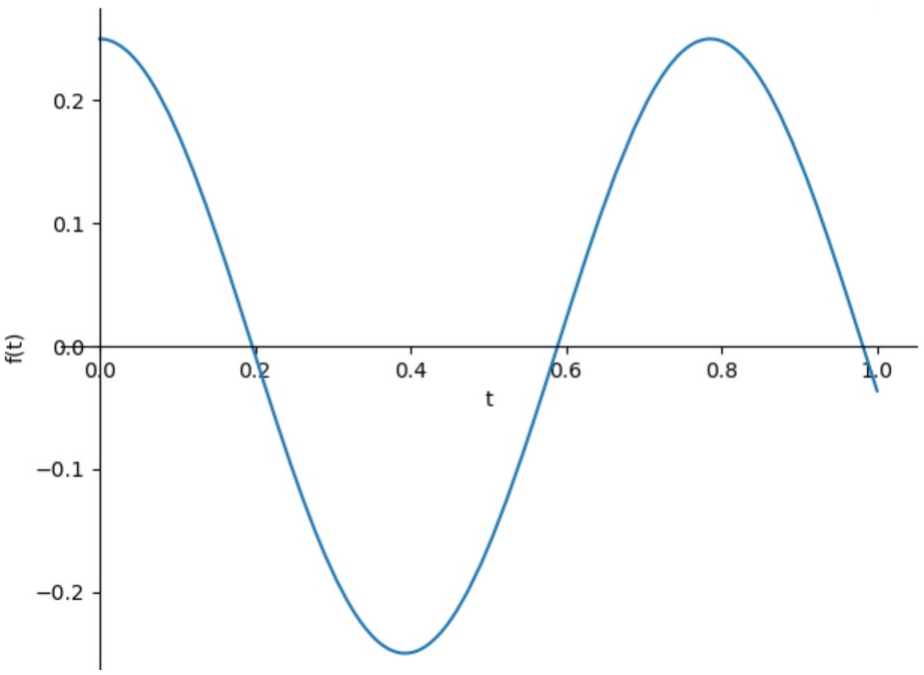
8.2 Sample Program:

Example-1: Solve the mechanical system $\frac{d^2x}{dt^2} + 64x = 0, x(0) = \frac{1}{4}, x'(0) = 1.$

```
from sympy import *
from matplotlib.pyplot import show
from sympy.plotting import plot
t = symbols('t')
x = Function('x')(t)
xd = x.diff(t)
xdd = x.diff(t,2)
eqn = Eq(xdd+64*x,0)
xsol=dsolve(eqn,x,ics={x.subs(t,0):0.25,xd.subs(t,0):0})
an_sol=xsol
display(an_sol)
plot(xsol.rhs,(t,0,1))
show()
```

Output:

$$x(t) = 0.25\cos(8t)$$

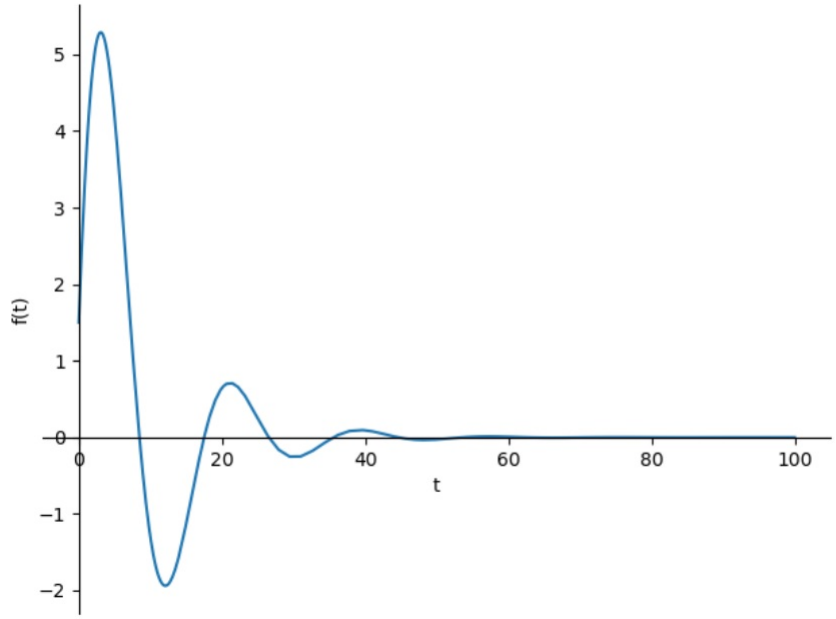


Example-2: Solve the mechanical system $9\frac{d^2x}{dt^2} + 2\frac{dx}{dt} + 1.2x = 0, x(0) = 1.5, x'(0) = 2.5$.

```
from sympy import *
from matplotlib.pyplot import show
from sympy.plotting import plot
t = symbols('t')
x = Function('x')(t)
xd = x.diff(t)
xdd = x.diff(t,2)
eqn = Eq(9*xdd+2*xd+1.2*x,0)
xsol=dsolve(eqn,x,ics={x.subs(t,0):1.5,xd.subs(t,0):2.5})
an_sol=xsol
display(an_sol)
plot(xsol.rhs,(t,0,50))
show()
```

Output:

$$x(t) = (7.66651877999929 \sin(0.347832796499967t) + 1.5 \cos(0.347832796499967t))\, e^{-0.111111111111111t}$$

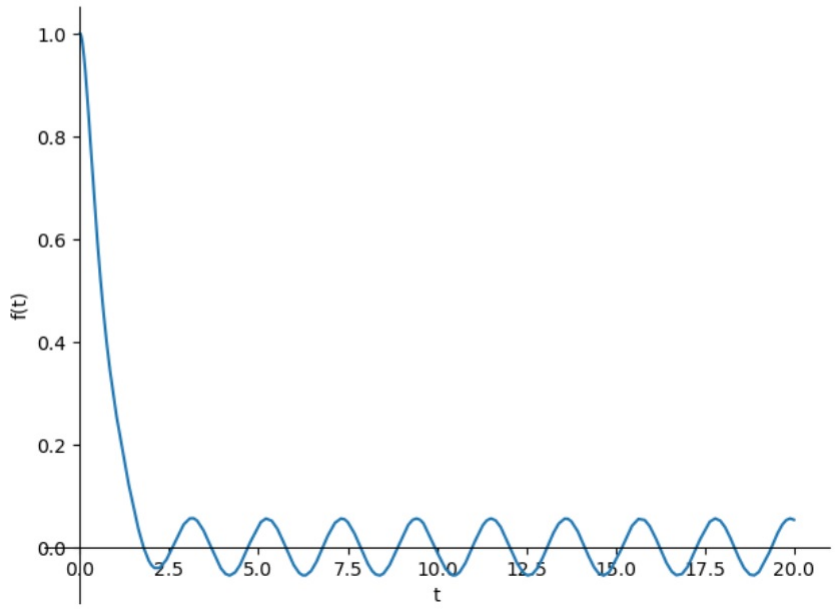


Example-3: Solve the mechanical system $\frac{d^2x}{dt^2} + 6\frac{dx}{dt} + 9x = sin(3t)$, $x(0) = 1, x'(0) = 0$.

```
from sympy import *
from matplotlib.pyplot import show
from sympy.plotting import plot
t = symbols('t')
x = Function('x')(t)
xd = x.diff(t)
xdd = x.diff(t,2)
eqn = Eq(xdd+6*xd+9*x,sin(3*t))
xsol=dsolve(eqn,x,ics={x.subs(t,0):1,xd.subs(t,0):0})
an_sol=xsol
display(an_sol)
plot(xsol.rhs,(t,0,20))
show()
```

Output:

$$x(t) = \left(\frac{19t}{6} + \frac{19}{18}\right)e^{-3t} - \frac{\cos(3t)}{18}$$



8.3 Activity-1:

1. Solve $y'' - 5y' + 6y = cos(4x)$
2. Solve $x^2y'' - 5xy' + 6y = cos(4logx)$
3. Solve $y'' + 2y' + 2y = cos(2x), y(0) = 0, y'(0) = 0$
4. $3\frac{d^2x}{dt^2} + 2\frac{dx}{dt} - 2x = cos(2x)$ with $x(0) = 0; x'(0) = 0$

8.4 Activity-2:Mechanical Systems

1. Solve $\frac{d^2x}{dt^2} + 64x = 0, x(0) = \frac{1}{4}, x'(0) = 1$
2. Solve $9\frac{d^2x}{dt^2} + 2\frac{dx}{dt} + 1.2x = 0, x(0) = 1.5, x'(0) = 2.5$

Observations:

LAB: 9

Deflection of Beams in Civil Engineering :

Objectives: In this lab students will learn

1. to solve the differential equation of a Deflection of beams.
2. to represent the solution graphically.

9.1 Deflection of Beams:

Equation of the Elastic Curve

The governing second order differential equation for the elastic curve of a beam deflection is

$$EI\frac{d^2y}{dx^2} = M$$

where EI is the flexural rigidity, M is the bending moment, and y is the deflection of the beam (+ve upwards).

Boundary Conditions

Fixed at $x = a$:

$$\text{Deflection is zero} \Rightarrow y\Big|_{x=a} = 0$$

$$\text{Slope is zero} \Rightarrow \frac{dy}{dx}\Big|_{x=a} = 0$$

Simply supported at $x = a$:

$$\text{Deflection is zero} \Rightarrow y\Big|_{x=a} = 0$$

A fourth order differential equation can also be written as

$$\boxed{EI\frac{d^4y}{dx^4} = -w}$$

where is w is the distributed load.

Here, two more boundary conditions are needed in terms of bending moment and shear force.

Boundary Conditions

Free at $x = a$:

$$\text{Bending moment is zero} \Rightarrow M = EI\frac{d^2y}{dx^2}\Big|_{x=a} = 0$$

$$\text{Shear force is zero} \Rightarrow V = EI\frac{d^3y}{dx^3}\Big|_{x=a} = 0$$

Simply supported at $x = a$:

$$\text{Bending moment is zero} \Rightarrow M = EI\frac{d^2y}{dx^2}\Big|_{x=a} = 0$$

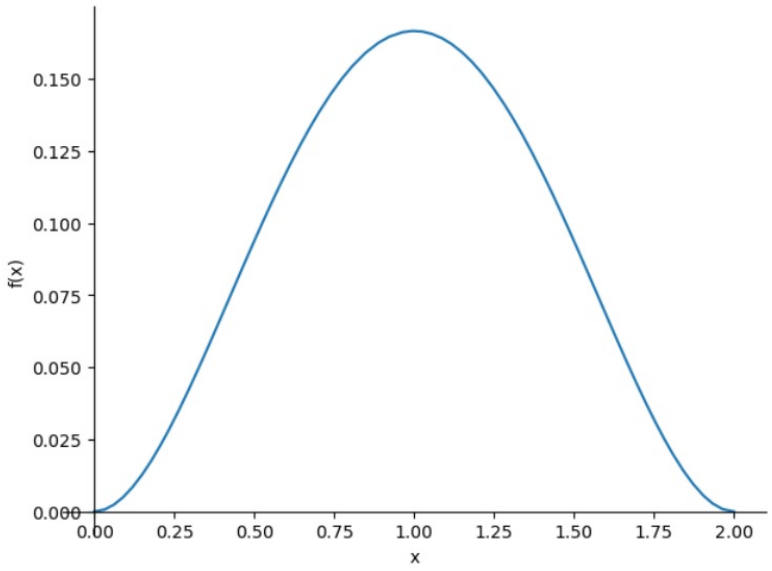
Example-1: Find the deflection of beam defined by following differential equation $\frac{d^4Q}{dt^4} = w$ subject to boundary conditions $Q(0) = 0, Q(2) = 0, Q'(0) = 0$ and $Q'(2) = 0$.

```
from sympy import *
from matplotlib.pyplot import show,plot
from sympy.plotting import plot
x = symbols('x')
Q = Function('Q')(x)
I =Function('I')(x)
Qd = Q.diff(x)
Qdd = Q.diff(x,2)
Qddd=Q.diff(x,3)
Qdddd=Q.diff(x,4)
w=4
eqn = Eq(Qdddd,w)
Qsol=dsolve(eqn,Q,ics={Q.subs(x,0):0,Qd.subs(x,0):0,Q.subs(x,2):0,Qd.subs(x,2):0})
Q_sol=simplify(Qsol)
print("Displacement of the beam is:")
display(Q_sol)
plot(Q_sol.rhs,(x,0,2))
show()
```

Output:

```
Displacement of the beam is:
```

$$Q(x) = \frac{x^2\left(x^2 - 4x + 4\right)}{6}$$



9.2 Activity-1:

1. Find the displacement in the cantilever beam which is simply supported one end at $x = 0$ and and fixed at $x = 2$.
2. Find the displacement in the beam which is simply supported at $x = 0$ and $x = 3$.
3. Find the displacement in the cantilever beam which is fixed at $x = 0$ and and freely supported $x = 4$
4. Find the displacement in the beam which is fixed at both ends $x = 0$ and $x = L = 1$.
5. Find the displacement in the beam for which bending moment is 0 at $x = 0$ and fixed at the other end $x = 4$.

Observations:

LAB: 10

Solving the differential equations of electrical circuits

Objectives: In this lab students will learn

1. to solve the differential equations of RC, LR and LRC series circuits.
2. to represent the solution of differential equation in electrical science graphically.

10.1 RLC Series Circuit:

Let resistance R, Inductance L and capacitance C are connected in series with an applied emf $E(t)$. Then according to Kirchoff's law, the sum of the voltage drops in a closed RLC circuit equals the impressed voltage. Hence
$L\frac{dI}{dt} + RI + \frac{1}{C}Q = E(t)$
This equation contains two unknowns, the current I in the circuit and the charge Q on the capacitor. However,we know that $\frac{dQ}{dt} = I$ above equation can be converted into the second order equation in Q, given by
$L\frac{d^2Q}{dt^2} + R\frac{dQ}{dt} + \frac{1}{C}Q = E(t)$

1. **Free Oscillations:** We say that an RLC circuit is in free oscillation if $E(t) = 0$ for $t > 0$, so that above equation becomes
   $L\frac{d^2Q}{dt^2} + R\frac{dQ}{dt} + \frac{1}{C}Q = 0$
2. **Forced Oscillations With Damping:** $E(t) \neq 0$ and $t > 0$, we have
   $L\frac{d^2Q}{dt^2} + R\frac{dQ}{dt} + \frac{1}{C}Q = E(t)$
   These equations must be supported with initial conditions:$Q(0) = Q_0, Q'(0) = I_0$.

10.2 Sample Programs:

Example-1: Find the current I in the RLC circuit which is defined the differential equation $L\frac{d^2Q}{dt^2} + R\frac{dQ}{dt} + \frac{1}{C}Q = 0$.

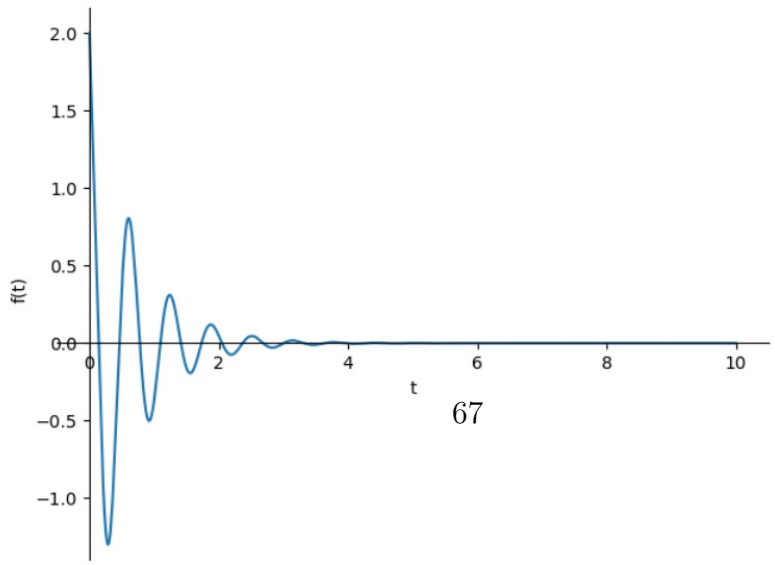
```
from sympy import *
from matplotlib.pyplot import show
from sympy.plotting import plot
t = symbols('t')
Q = Function('Q')(t)
I =Function('I')(t)
Qd = Q.diff(t)
Qdd = Q.diff(t,2)
L,R,C=1,3,0.01
eqn = Eq(L*Qdd+R*Qd+(1/C)*Q,0)
Qsol=dsolve(eqn,Q,ics={Q.subs(t,0):0,Qd.subs(t,0):2})
Q_sol=simplify(Qsol)
display(Q_sol)
I_Sol=diff(Q_sol.rhs,t)
I_sol=I_Sol
print("Currect in the circuit:")
display(I_sol)
plot(I_sol,(t,0,10))
show()
```

Output:

$$Q(t) = 0.20228869496967e^{-1.5t}\sin(9.88685996664259t)$$

```
Currect in the circuit:
```

$$-0.303433042454504e^{-1.5t}\sin(9.88685996664259t) + 2.0e^{-1.5t}\cos(9.88685996664259t)$$



Example-2: Find the current I in the RLC circuit which is defined the differential equation $L\frac{d^2Q}{dt^2} + R\frac{dQ}{dt} + \frac{1}{C}Q = E_0 cos(\omega t)$.

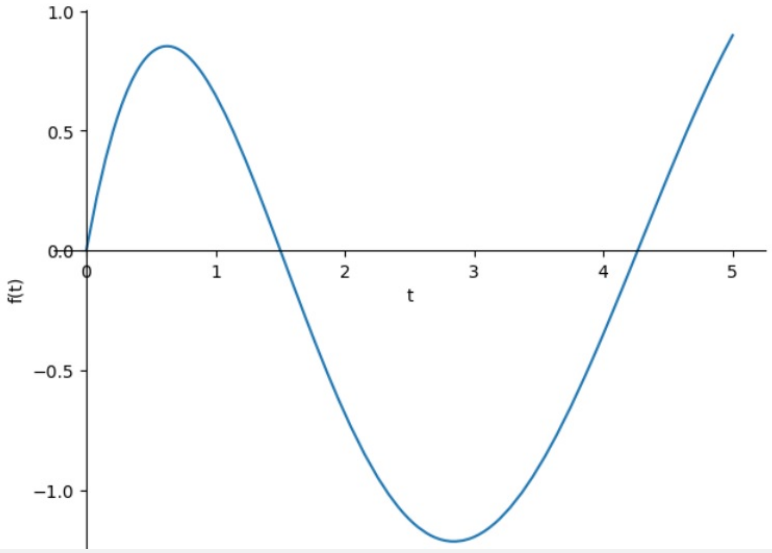
```
from sympy import *
from matplotlib.pyplot import show
from sympy.plotting import plot
t,w = symbols('t,w')
Q = Function('Q')(t)
I =Function('I')(t)
Qd = Q.diff(t)
Qdd = Q.diff(t,2)
L,R,C,w=1,2,0.5,1
eqn = Eq(L*Qdd+R*Qd+(1/C)*Q,3*cos(w*t))
Qsol=dsolve(eqn,Q,ics={Q.subs(t,0):0,Qd.subs(t,0):0})
Q_sol=simplify(Qsol)
display(Q_sol)
I_Sol=diff(Q_sol.rhs,t)
I_sol=I_Sol
print("Currect in the circuit:")
display(I_sol)
plot(I_sol,(t,0,5))
show()
```

Output:

$$Q(t) = \left((1.2\sin(t) + 0.6\cos(t))\, e^{1.0t} - 1.8\sin(1.0t) - 0.6\cos(1.0t)\right) e^{-1.0t}$$

```
Currect in the circuit:
```

$$-1.0\left((1.2\sin(t) + 0.6\cos(t))\, e^{1.0t} - 1.8\sin(1.0t) - 0.6\cos(1.0t)\right) e^{-1.0t}$$
$$+\left((-0.6\sin(t) + 1.2\cos(t))\, e^{1.0t} + 1.0\cdot(1.2\sin(t) + 0.6\cos(t))\, e^{1.0t} + 0.6\sin(1.0t) - 1.8\cos(1.0t)\right) e^{-1.0t}$$



10.3 Activity-1:

- Find the current in the RLC circuit, assuming that $E(t) = 0$ for $t > 0$
  a) $R = 3$,$L = 0.1$,$C = 0.01$, with conditions zero initial conditions.
  b) $R = 10$,$L = 1$,$C = 0.1$, with conditions zero initial conditions.
  c) $R = 3$,$L = 0.1$,$C = 0.01$, with conditions zero initial conditions.
  d) $R = 10$,$L = 1$,$C = 0.1$, with conditions zero initial conditions.
  e) $R = 10$,$L = 1$,$C = 0.1$, with conditions zero initial conditions.

10.4 Activity-2:

- Find the steady state current in the circuit described by the equation:
  a) $110I'' + 3I' + 100I = 10cos10t$
  b) $I'' + 3I' + 2I = sint$
  c) $5I'' + 10I' + 100I = -5cos2t$
  d) $I'' + 5I' + 5I = e^t$
  e) $I'' + 3I' + 10I = 2t$

Observations:

LAB: 11

Linear Congruences for Computer Science Cluster

11.1 Finding GCD and solving linear Congruence

Objectives: In this lab students will learn

1. to find the gcd of two numbers using Euclid algorithm
2. to express it as linear combination of numbers.
2. to find the solution of the linear congruence.

11.1.1 Euclidean algorithm:

This algorithm is useful to find GCD of two numbers. The algorithm is as follows:
The two numbers a and b can be assumed positive such that $a < b$. Let r_1 be the remainder when b is divided by a. Then $0 \leq r_1 < a$. That is $b = ak_1 + r_1$. Now let r_2 be the remainder when a is divided by r_1. That is $a = r_1k_2 + r_2$. Where $0 \leq r_2 < r_1$. Continue this process of dividing each divisor by the next remainder. At some stage we obtain remainder 0. The "last non-zero remainder is the GCD" of a and b. This is known as Euclid's algorithm.

Algorithm analysis:
1. Recursive process - operations are repeated till **stopping criterion** is reached.
2. The **output** of one step is used as the **input of the next step**.

Relatively Primes:
Two numbers a and b are called "relatively prime" or "co-prime" if their GCD (also known as HCF) is equal to 1. For example: 2 and 19 are relatively prime, because 1 is the largest natural number that divides both 2 and 19.

11.1.2 Sample Programme:

Example-1: Check whether 241 and 512 are relatively prime or not.

```
def gcd1(a,b):
    c=1;
    if b <a:
        t=b;
        b=a;
        a=t;
    while (c>0):
        c=b%a;
        print(a,c);
        b=a;
        a=c;
        continue
    print('GCD= ',b);
r=gcd1(241,512)
if (r==2) and (r>2):
    print("Given numbers are not relative primes")
else:
    print("Given numbers are relativcely primes")
```

Output:

```
241 30
30 1
1 0
GCD=  1
Given numbers are relativcely primes
```

Example-2: Calculate GCD of (a,b) and express it as linear combination of a and b. Calculate GCD=d of 21 and 19 , express the GCD as $21x+19y=d$.

```
from  sympy import *
a=int(input('enter the first number :'))
b=int(input('enter the second number :'))
s1,s2,t1,t2,r1,r2=1,0,0,1,a,b;
r3=(r1%r2);
q = (r1-r3)/r2;
s3=s1-s2*(q);
t3=t1-t2*q;
while (r3!=0):
    r1,r2,s1,s2,t1,t2=r2,r3,s2,s3,t2,t3;
    r3=(r1%r2);
    q = (r1-r3)/r2;
    s3=s1-s2*(q);
    t3=t1-t2*q;
print('the GCD of ',a,' and',b,'is',r2);
print('%d x %d + %d x %d = %d\n'%(a,s2,b, t2,r2));
```

Output:

```
enter the first number :21
enter the second number :19
the GCD of  21  and 19 is 1
21 x -9 + 19 x 10 = 1
```

11.2 Solving Linear Congruence:

An equation of the form $ax \equiv b(\text{mod m})$ is called linear congruence relation. For a given a,b and m, x is called the solution of the congruence. The linear congruence has a solution if and only if $gcd(a, m)$ divides b.

11.2.1 Sample Programme:

The solution of the congruence $ax \equiv 1(\text{mod } p)$ is called multiplicative inverse of $a(modp)$.
Example-1: Solve the linear congruence $21 \equiv 9mod24$.

```
import sys
import numpy as np
def ExtendedEuclidAlgo(a, b):
    if a == 0 :
        return b, 0, 1
    gcd, x1, y1 = ExtendedEuclidAlgo(b % a, a)
    x,y = y1 - (b // a) * x1,x1
    return gcd, x, y
a=int(input('Enter integer a: '));
b=int(input('Enter integer b: '));
n=int(input('Enter integer n: '));
A,B = a % n, b % n
u,v = 0,0
d, u, v = ExtendedEuclidAlgo(A, n)
if (B % d != 0):
    sys.exit("Solution doesnot exits")
else:
    x0 = (u * (B // d)) % n
    if (x0 < 0):
        x0 += n
z=np.linspace(1,d,d)
for i in range(d):
    y=(x0 + i * (n // d)) % n
    z[i]=y
print("Solutions of given linear congruence are: ", z)
```


Output:

```
Enter integer a: 21
Enter integer b: 9
Enter integer n: 24
Solutions of given linear congruence are:  [21.  5. 13.]
```

Example-2: Solve the linear congruence $4 \equiv 2 mod 6$.

```
from sympy import *
from math import *
import numpy as np
a=int(input('Enter integer a: '));
b=int(input('Enter integer b: '));
n=int(input('Enter integer n: '));
d=gcd(a,n)
if (b%d!=0):
    print('the congruence has no integer solution');
else:
    for i in range(1,n-1):
        x=(n/a)*i+(b/a)
        if(x//1==x):
            print('the solution of the congruence is ', x)
            break
x0,z=x,np.linspace(1,d,d)
for i in range(d):
    y=(x0 + i * (n // d)) % n
    z[i]=y
print("Solutions of given linear congruence are: ", z)
```

Output:

```
Enter integer a: 4
Enter integer b: 2
Enter integer n: 6
the solution of the congruence is  2.0
Solutions of given linear congruence are:  [2. 5.]
```

11.3 Activity-1:

1. Find the GCD of 234 and 672 using Euclidean algorithm.
2. What is the largest number that divides both 1024 and 1536?
3. Find the greatest common divisor of 6096 and 5060?
4. Prove that 1235 and 2311 are relatively prime.
5. Are 9797 and 7979 coprime?
6. Write a function in Python to compute the greatest common divisor of 15625 and 69375.
7. Using a Python module, find the GCD of 4096 and 6144.
8. What is the GCD of $x^4y + 3xy^2 + 5xy$ and $2x^3y + 3x^2y^3 + 5x^2y$.

11.4 Activity-2:

1. Find the solution of the congruence $12x \equiv 6(\text{ mod } 23)$
2. Find the multiplicative inverse of 3 mod 31
3. Prove that $12x \equiv 7(\text{ mod } 14)$ has no solution. Give reason for the answer.
4. Find the solution of the congruence $5x \equiv 3(\text{mod } 13)$
5. Check whether $4x \equiv 3(\text{mod } 12)$ has a solution

Observations: