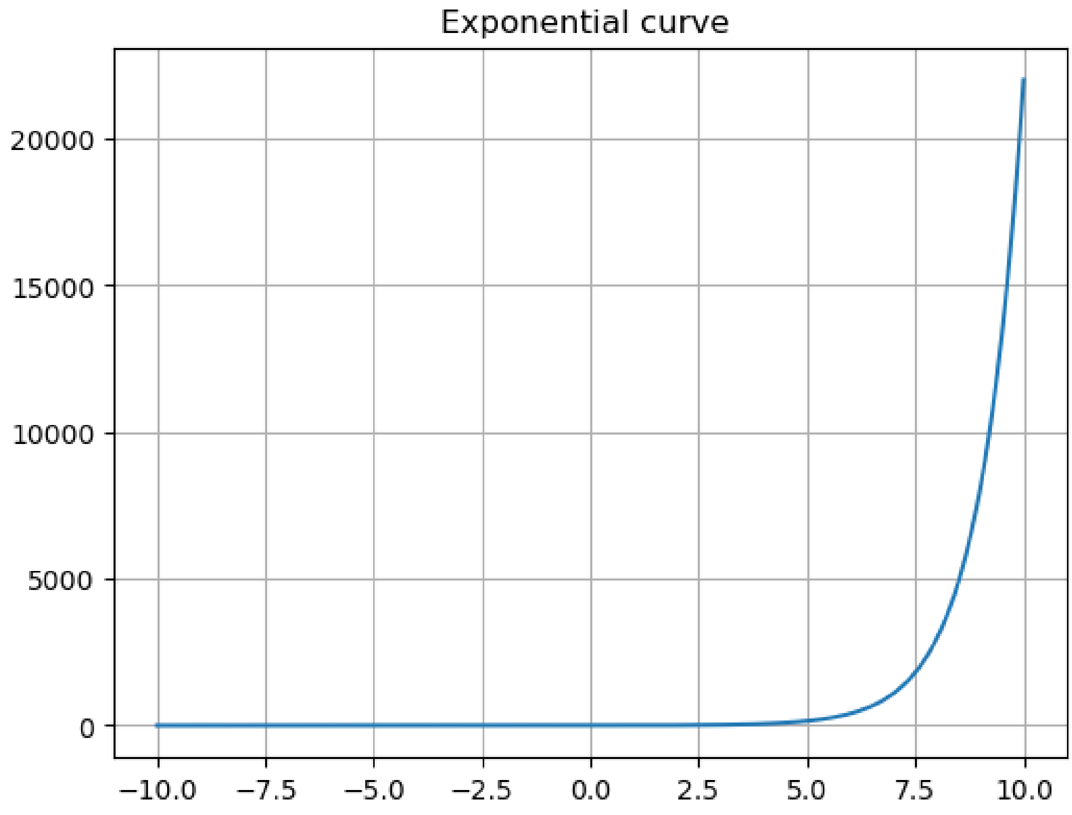
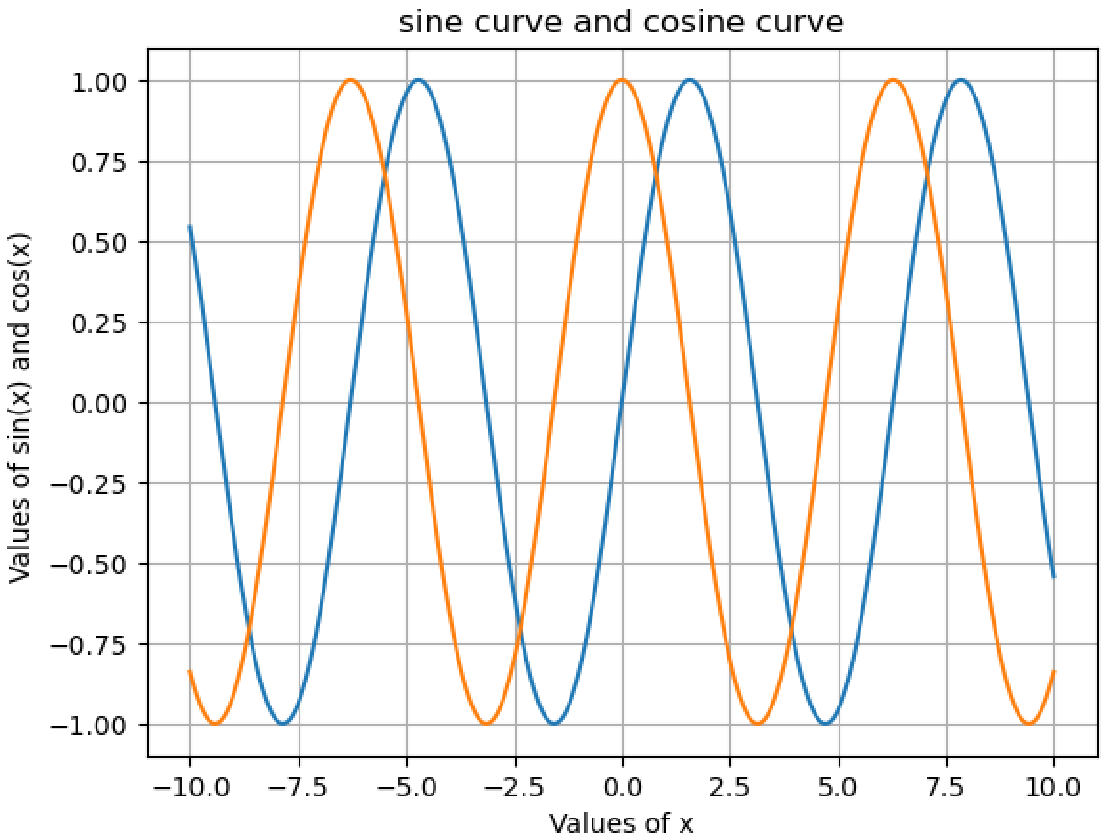
```
[ ]: #1st sem 2023-2024 (23MATX11)
```

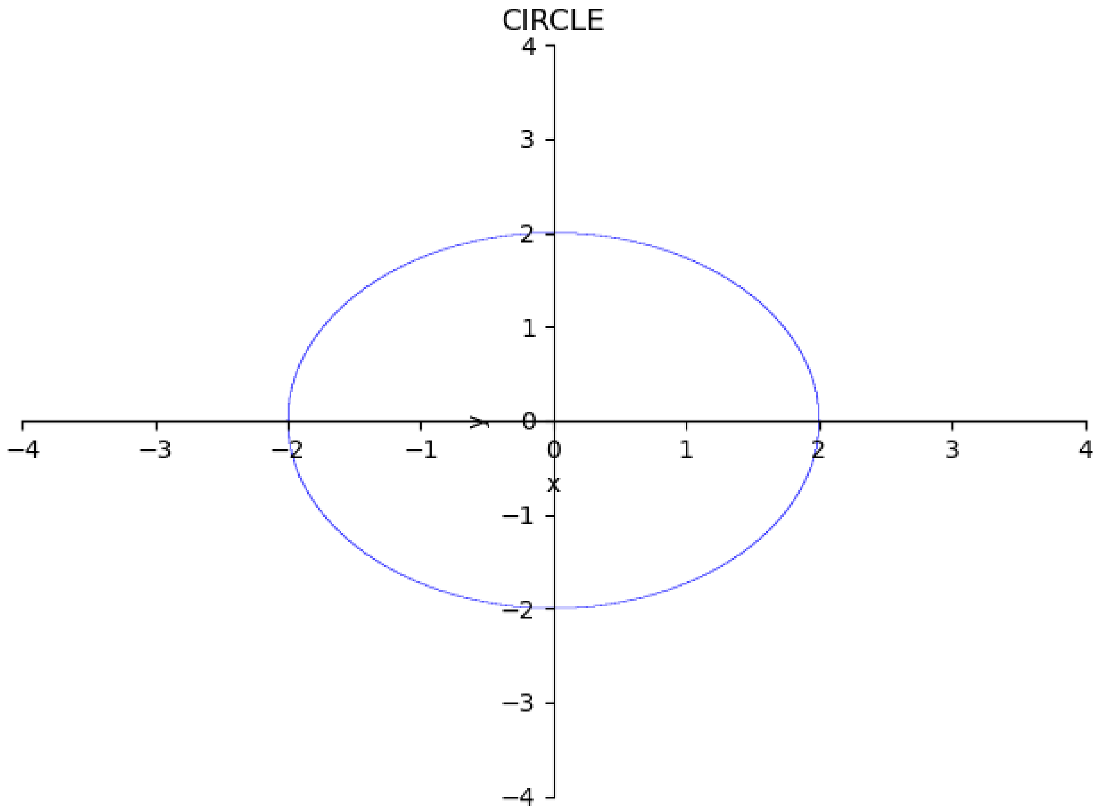
```
[53]: #Lab 1: Write the Python code and Execute the program to find Exponential curve, y = e^x.
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(-10, 10, 0.001) # x takes the values between -10 and 10 with a step l
y = np.exp(x) # Exponential function
plt.plot(x,y) # plotting the points
plt.title("Exponential curve ") # giving a title to the graph
plt.grid() # displaying the grid
plt.show() # shows the plot
```



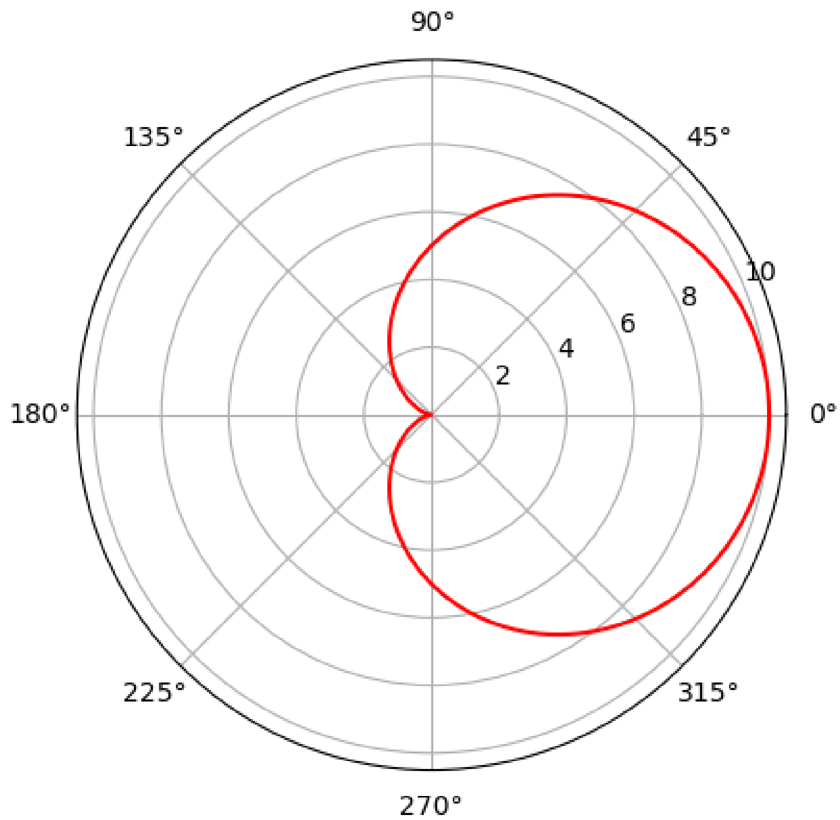
[54]:
```
#Sine and Cosine curves
x = np.arange(-10, 10, 0.001)
y1 = np.sin(x)
y2=np.cos(x)
plt.plot(x,y1,x,y2) # plotting sin and cos function together with same values␣
 ↪of x
plt.title("sine curve and cosine curve")
plt.xlabel("Values of x")
plt.ylabel("Values of sin(x) and cos(x) ")
plt.grid()
plt.show()
```



[59]:
```
#Circle: x2 + y2 = 4
from sympy import plot_implicit, symbols, Eq
x,y=symbols('x,y')
p1= plot_implicit(Eq (x**2+y**2,4),(x,-4,4),(y,-4,4),title='CIRCLE')
```



[60]:
```
#Cardiod: r = 5(1 + cos )
from pylab import *
theta=linspace(0,2*np.pi,1000)
r1=5+5*cos(theta)
polar(theta,r1,'r')
show()
```



[1]:
```
#Lab 2:
#Find the angle between the curves r = 4 (1 + cos t) and r = 5 (1 - cos t).
from sympy import *
r,t =symbols('r,t')
r1=4*(1+cos(t));
r2=5*(1-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 %0.4f'%float(w))
```

```
Angle between curves in radians is 1.5708
```

[14]:
```
#Find the radius of curvature for r = 4 ( 1 + cos t) at t = /
from sympy import *
t=symbols('t')
r=symbols('r')
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 %3.4f units'%rho1)
```

```
The radius of curvature is 3.7712 units
```

[10]:
```
#Lab 3
#Expand sin(x) as Taylor series about x=pi/2 upto 3rd degree term. Also find␣
↪sin(100°)
import numpy as np
from matplotlib import pyplot as plt
from sympy import *
x=Symbol('x')
y=sin(x)
format
x0=float(pi/2)
dy=diff(y,x)
d2y=diff(y,x,2)
d3y=diff(y,x,3)
yat=lambdify(x,y)
dyat=lambdify(x,dy)
d2yat=lambdify(x,d2y)
d3yat=lambdify(x,d3y)
y=yat(x0)+((x-x0)/1)*dyat(x0)+((x-x0)**2/2)*d2yat(x0)
+((x-x0)**3/6)*d3yat(x0)
print(simplify(y))
yat=lambdify(x,y)
print("%.3f" % yat(pi/2+10*(pi/180)))
```

```
-1.02053899928946e-17*x**3 - 0.5*x**2 + 1.5707963267949*x - 0.23370055013617
0.985
```

[1]:
```
#Find the Maclaurin series expansion of sin(x) + cos(x) upto 3rd degree term.
import numpy as np
from matplotlib import pyplot as plt
from sympy import *
x=Symbol('x')
y=sin(x)+cos(x)
format
x0=float(0)
dy=diff(y,x)
```

```
d2y=diff(y,x,2)
d3y=diff(y,x,3)
yat=lambdify(x,y)
dyat=lambdify(x,dy)
d2yat=lambdify(x,d2y)
d3yat=lambdify(x,d3y)
y=yat(x0)+((x-x0)/1)*dyat(x0)+((x-x0)**2/2)*d2yat(x0)+((x-x0)**3/6)*d3yat(x0)
print(simplify(y))
yat=lambdify(x,y)
print("%.3f" % yat(0.2))
```

```
-0.166666666666667*x**3 - 0.5*x**2 + 1.0*x + 1.0
1.179
```

[27]:
```
#Lab 4: Prove that if u =  (xcosy-ysiny) then   +   =0.
from sympy import *
x,y=symbols('x,y')
u=exp(x)*(x*cos(y)-y*sin(y))
display(u)
dux=diff(u, x)
duy=diff(u, y)
uxx=diff(dux, x)
uyy=diff(duy,y)
w=uxx+uyy
w1=simplify(w)
print('Ans',float(w1))
```

$(x\cos(y) - y\sin(y))\, e^{x}$

```
Ans 0.0
```

[2]:
```
#If  =        ,  =        ,  =  sin         ( , , ) / ( , , )
from sympy import *
from sympy.abc import rho,phi ,theta
X=rho*cos(phi)*sin(theta);
Y=rho*cos(phi)*cos(theta);
Z=rho*sin(phi);
dx=Derivative(X,rho).doit()
dy=Derivative(Y,rho).doit()
dz=Derivative(Z,rho).doit()
dx1=Derivative(X,phi).doit()
dy1=Derivative(Y,phi).doit()
dz1=Derivative(Z,phi).doit()
dx2=Derivative(X,theta).doit()
dy2=Derivative(Y,theta).doit()
dz2=Derivative(Z,theta).doit()
J=Matrix([[dx,dy,dz],[dx1,dy1,dz1],[dx2,dy2,dz2]])
print('The Jacobian matrix is')
display(J)
```

```
print('J=')
display(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=

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

[13]:

```
#Evaluate    →    ()/
from sympy import Limit, Symbol, exp, sin
x=Symbol('x')
L=Limit((sin(x))/x,x,0).doit()
display(L)
```

1

[15]:

```
#Prove that   →∞( + ) = e
from sympy import *
from math import inf
x=Symbol('x')
L=limit((1+1/x)**x, x, inf).doit()
display(L)
```

e

[3]:

```
#Lab 5: The temperature oa a body drops from 100c to 75c in 10 mins where the␣
↪surrounding
#is at the temperature 20c.What will be the temperature of the body after half␣
↪an hour?
from sympy import *
import numpy as np
from matplotlib import pyplot as plt
k= Symbol ('k')
T= Symbol ('T')
T0=20 # surrounding temp
T1=100 # inital temp
# one reading t=10 minute temp is 75 degree
t=10
T2=75
c=T1-T0 # to calculate c at time =0 minutes
k=(1/t)*log((c)/(T2-T0))
print('k=',k)
t1=30
T=Function('T')(t1)
T=T0+((T1-T0)*exp(-k*t1))# solution
```

```python
print(T,'°C')
```

```
k= 0.0374693449441411
45.9960937500000 °C
```

[10]:
```python
#Lab 6: Solve:  /  + y tanx -  3sec ( ) = 0
from sympy import *
x,y=symbols('x,y')
y=Function('y')(x)
y1=Derivative(y, x)
z1=dsolve (Eq (y1+y*tan(x)-y**3*sec(x),0),y)
display(z1)
```

```
[Eq(y(x), -sqrt(1/(C1 - 2*sin(x)))*cos(x)),
 Eq(y(x), sqrt(1/(C1 - 2*sin(x)))*cos(x))]
```

[12]:
```python
# Solve:  /  + 2y -  5 = 0
from sympy import *
x,y=symbols('x, y')
y=Function("y")(x)
y1=Derivative(y, x)
z1=dsolve (Eq (y1+x**2*y-x**5,0),y)
display(z1)
```

$y(x) = C_1 e^{-\frac{x^3}{3}} + x^3 - 3$

[16]:
```python
# Lab 7: Evaluate the integral   ( 2+  2)
from sympy import *
x=symbols('x')
w1=integrate((x**2+y**2),(y,0,x),(x,0,1))
print(simplify(w1))
```

```
1/3
```

[26]:
```python
#Evaluate the integral   (   )3- - 0   3- 0
from sympy import *
x= Symbol('x')
y= Symbol('y')
z= Symbol('z')
w2=integrate(integrate(integrate((x*y*z),(z,0,3-x-y)),(y,0,3-x)),(x,0,3))
print(w2)
```

```
81/80
```

[29]:
```python
# Lab 8: Evaluate the integral   -  ∞
from sympy import *
x=symbols('x')
w1=integrate(exp(-x),(x,0,float('inf')))
print(simplify(w1))
```

```
1
```

[38]:
```python
#Verify that  ( , )= ( ) ( )/ ( + )
from sympy import beta, gamma
m=5;
n=7;
m=float(m);
n=float(n);
s=beta(m,n);
t=(gamma(m)*gamma(n)/gamma(m+n));
print(s,t)
if(abs(s-t)<=0.00001):
    print('Beta and Gamma are related')
else:
    print('Given values are wrong')
```

```
0.000432900432900433 0.000432900432900433
Beta and Gamma are related
```

[40]:
```python
#Lab 9: Check whether the following system of homogenous linear equation has␣
 ↪non-trivial solution.
#x1 + 2x2 - x3 = 0,2x1 + x2 + 4x3 = 0, 3x1 + 3x2 + 4x3 = 0
import numpy as np
A=np.matrix([[1,2,-1],[2,1,4],[1,-1,5]])
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",n-r,"non - trivial solution (s)")
```

```
System has 1 non - trivial solution (s)
```

[46]:
```python
#Examine the consistency of the following system of equations and solve if␣
 ↪consistent.
#x1 + 2x2 - x3 = 1, 2x1 + x2 + 4x3 = 2, 3x1 + 3x2 + 4x3 = 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 ")
        print(np.linalg.solve(A,B))
    else:
```

```
        print("The system has infinitely many solutions ")
else:
    print("The system of equations is inconsistent")
```

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

[49]:
```
#Lab 10: Obtain the Eigenvalues and Eigenvectors for the given matrix P =[ , , ;␣
 ↪ , , ;  , , ].
import numpy as np
I=np.array([[4,3,2],[1 ,4 , 1],[3 , 10 , 4]])
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)
## To display one eigen value and correspondingeigen vector
print(" Eigen value :\n ", w[0])
print("\n Corresponding Eigen vector :", v[:,0])
```

```
 Given matrix :
 [[ 4  3  2]
 [ 1  4  1]
 [ 3 10  4]]

 Eigen values :
 [8.98205672 2.12891771 0.88902557]

 Eigen vectors :
 [[-0.49247712 -0.82039552 -0.42973429]
 [-0.26523242  0.14250681 -0.14817858]
 [-0.82892584  0.55375355  0.89071407]]
 Eigen value :
  8.982056720677651

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

[51]:
```
#Compute the numerically Largest Eigenvalue and Eigenvector of the matrix␣
 ↪[6,-2,2; -2,3,-1; 2,-1,3]
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])
```

```python
a=np.array([[6,-2,2],[-2,3,-1],[2,-1,3]])
for i in range(20):
    x=np.dot(a,x)
    lambda_1, x=normalize(x)
print('Eigenvalue:',lambda_1)
print('Eigenvector:',x)
```

```
Eigenvalue: 7.999999999989086
Eigenvector: [ 1.  -0.5  0.5]
```

[ ]: