Linear Differential Equations ¶

In [13]:

```
import numpy as np
import sympy as sy
import sympy, math
x=sy.Symbol('x')
F=x**2-4*x+4
coeff=[1,-4,4]
r=np.roots(coeff)
print(r)
```

```
[ 2.  2.]
```

Example 01. Solve y'' + y' - 2y = 0

In [14]:

```
print("Solution to the given linear differential equation is given by: ")
c1 = sympy.Symbol('c1')
c2 = sympy.Symbol('c2')
x = sympy.Symbol('x')
m = np.poly([0])
f = m**2+m-2
coeff = [1,1-2]
r = np.roots(coeff)
y = c1*sympy.exp(r[0]*x)+c2*sympy.exp(r[0]*x)
print("y = ",y)
```

```
Solution to the given linear differential equation is given by:
y =  c1*exp(1.0*x) + c2*exp(1.0*x)
```

Example 02. Solve y'' + 6y' + 9y = 0

In [3]:

```
print("Solution to the given linear differential equation is given by: ")
c1 = sympy.Symbol('c1')
c2 = sympy.Symbol('c2')
x = sympy.Symbol('x')
m = np.poly([0])
f = m**2+6*m+9;
r=np.roots([1,6,9])
y=(c1+x*c2)*sy.exp(r[0]*x)
print(r)
print("y = ",y)
```

```
Solution to the given linear differential equation is given by:
[-3. +3.72529030e-08j -3. -3.72529030e-08j]
y =  (c1 + c2*x)*exp(x*(-3.0 + 3.72529029846191e-8*I))
```

Example 03. Solve $y^{iv} + 6y' + 9y = 0$

In [1]:

```
c1 = sympy.Symbol('c1')
c2 = sympy.Symbol('c2')
c3 = sympy.Symbol('c3')
c4 = sympy.Symbol('c4')
x = sympy.Symbol('x')
m = np.poly([0])
f = m**4+4;
r=np.roots([1,0,0,0,4])
print('roots are: ')
print(r)
print('Solution is: ')
y=c1*sympy.exp(r[0].real*x+c2.sympy.exp(r[1].real*x)+c3*sympy.exp(r[2].real*x))+c4*sympy.ex
```

```
  File "<ipython-input-1-b47aa1eae01c>", line 12
    y=c1*sympy.exp(r[0].real*x+c2.sympy.exp(r[1].real*x)+c3*sympy.exp(r[2].r
eal*x))+c4*sympy.exp(r[3].real*x))

^
SyntaxError: invalid syntax
```

Linear Differential equations with constant coefficients

Q1: Solve the differential equation $\frac{d^2}{dx^2} - 5f = 0$

In [16]:

```
#to print neatly
from sympy.interactive import printing
printing.init_printing(use_latex=True)
import sympy as sp
from sympy import *
```

In [17]:

```
x=Symbol('x')
x                     #defining symbol using symbol
```

Out[17]:

x

In [18]:

```
from sympy import *
f = Function('f')(x)                        #defining f as a function of x
diffeq = Eq(f.diff(x,x)-5*f,0)              #define the differential equation
diffeq                                #to print the differential equation
```

Out[18]:

$$-5f(x) + \frac{d^2}{dx^2}f(x) = 0$$

In [19]:

```
dsolve(diffeq,f)            #solving the differential equation
```

Out[19]:

$$f(x) = C_1 e^{-\sqrt{5}x} + C_2 e^{\sqrt{5}x}$$

Q2: Solve the differential equation $\frac{d^2y}{dx^2} + 5\frac{dy}{dx} + 6 = 0$

In [20]:

```
x = Symbol('x')
y = Function('y')(x)                 #defining y as a function of x
diffeq = Eq(y.diff(x,x)+5*y.diff(x)+6*y,0)     #define the #define the differential equatic
print('The differential equation is: ')
print(diff)
print('The solution is: ')
dsolve(diffeq,y)
```

```
The differential equation is:
<function diff at 0x000002719CE4E268>
The solution is:
```

Out[20]:

$$y(x) = \left(C_1 + C_2 e^{-x}\right) e^{-2x}$$

In [21]:

```
x = Symbol('x')
y = Function('y')(x)                 #defining y as a function of x
diffeq = Eq(y.diff(x,x,x)+y,0)     #define the differential equation
print('The differential equation is: ')
print(diff)
print('The solution is: ')
dsolve(diffeq,y)
```

```
The differential equation is:
<function diff at 0x000002719CE4E268>
The solution is:
```

Out[21]:

$$y(x) = C_3 e^{-x} + \left(C_1 \sin\left(\frac{\sqrt{3}x}{2}\right) + C_2 \cos\left(\frac{\sqrt{3}x}{2}\right)\right)\sqrt{e^x}$$

Q6. Solve (D^2 + 4)y = cos3x

In [22]:

```
x = Symbol('x')
y = Function('y')(x)                #defining y as a function of x
diffeq = Eq(y.diff(x,x)+4*y,cos(3*x))     #define the differential equation
print('The differential equation is: ')
print(diff)
print('The solution is: ')
dsolve(diffeq,y)
```

```
The differential equation is:
<function diff at 0x000002719CE4E268>
The solution is:
```

Out[22]:

$$y(x) = C_1 \sin(2x) + C_2 \cos(2x) - \frac{1}{5}\cos(3x)$$

Q7: Solve (D^2 +4D + 4)y = e^(-3x)

In [23]:

```
diffeq = Eq(y.diff(x,x)+4*y.diff(x)+4*y,exp(-3*x))     #define the differential equation
print('The differential equation is: ')
print(diff)
print('The solution is: ')
dsolve(diffeq,y)
```

```
The differential equation is:
<function diff at 0x000002719CE4E268>
The solution is:
```

Out[23]:

$$y(x) = \left(C_1 + C_2 x + e^{-x}\right) e^{-2x}$$

Q8. Solve(D^2 -2D +2)y = e^xcos(2x)

In [26]:

```
diffeq = Eq(y.diff(x,x)-2*y.diff(x)+2*y,exp(x)*cos(2*x))     #define the differential equat
print('The differential equation is: ')
print(diff)
print('The solution is: ')
dsolve(diffeq,y)
```

```
The differential equation is:
<function diff at 0x000002719CE4E268>
The solution is:
```

Out[26]:

$$y(x) = \left(C_1 \sin(x) + C_2 \cos(x) - \frac{1}{3}\cos(2x)\right) e^x$$

1. Solve $\frac{dy}{dx} = x - y, y_0 = 1$

In [27]:

```
x = sy.Symbol('x')
f = sp.Function('f')(x)
diffe = Eq(f.diff(x)-x+f,0)
diffe
dsolve(diffe,f)
```

Out[27]:

$$f(x) = (C_1 + (x - 1) e^x) e^{-x}$$

Linear Differential equations using ODEINT

1. Model: Fuction name that returns derivative values at requested y and t values as dy_dt = model(y,t)

2. y0: Initial conditions of the differential states

3. t: Time points at which the solution should be reported. Additional internal points are often calculated to maintain accuracy of the solution but are not reported.

Solve $\frac{dy}{dx} = x - y, y_0 = 1$

In [6]:

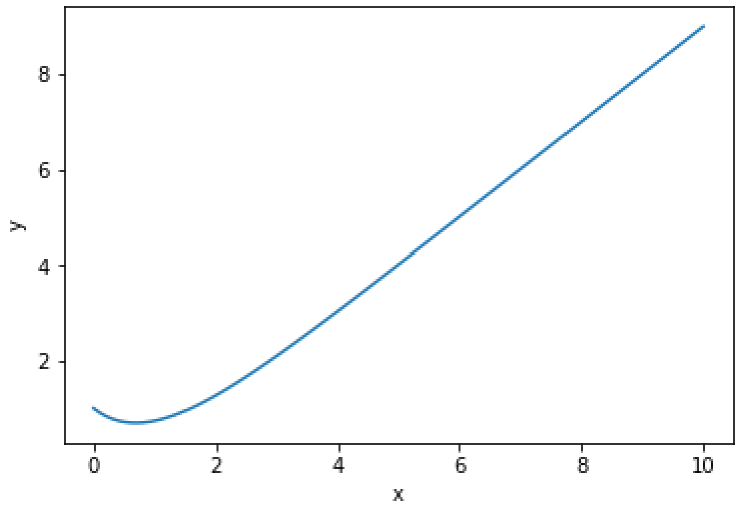
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from scipy.integrate import odeint
```

In [8]:

```
#Define a function that calculates the derivative
def dy_dx(y,x):
    return x-y
xs = np.linspace(0,10,100)
y0 = 1.0 #the initial condition
ys = odeint(dy_dx,y0,xs)
```

In [9]:

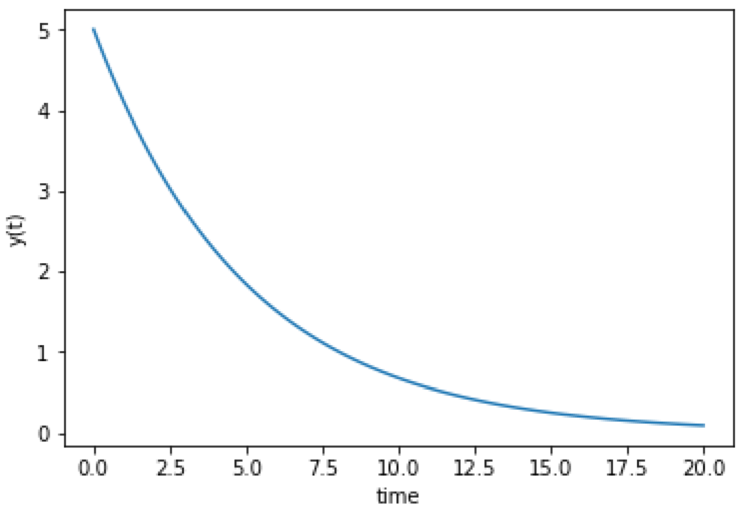
```
#plot results
plt.plot(xs,ys)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
print(ys)
```



Solve $\frac{dy}{dx} = -ky(t)$ with parameter k = 0.1,0.2,0.5 and the initial condition $y_0 = 5$

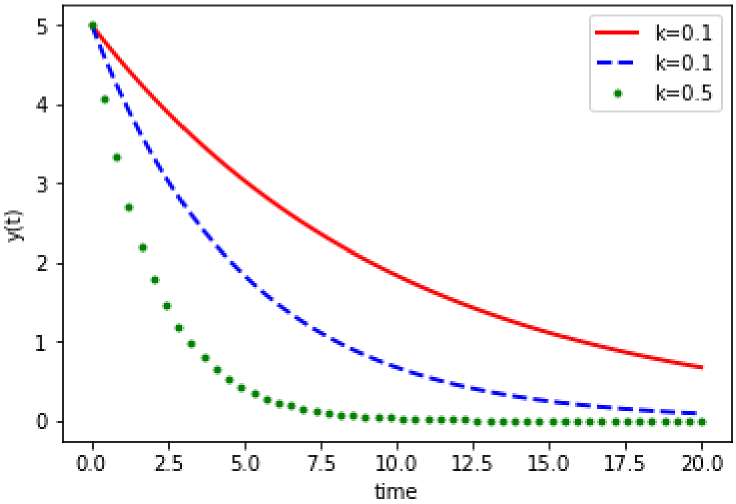
In [12]:

```
#function that returns dy/dt
def model(y,t):
    k=0.2
    dydt = -k*y
    return dydt

# initial condition
y0=5

# time points
t = np.linspace(0,20,100)

#solve ODE
y = odeint(model,y0,t)

#plot results
plt.plot(t,y)
plt.xlabel('time')
plt.ylabel('y(t)')
plt.show()
print(y)
```



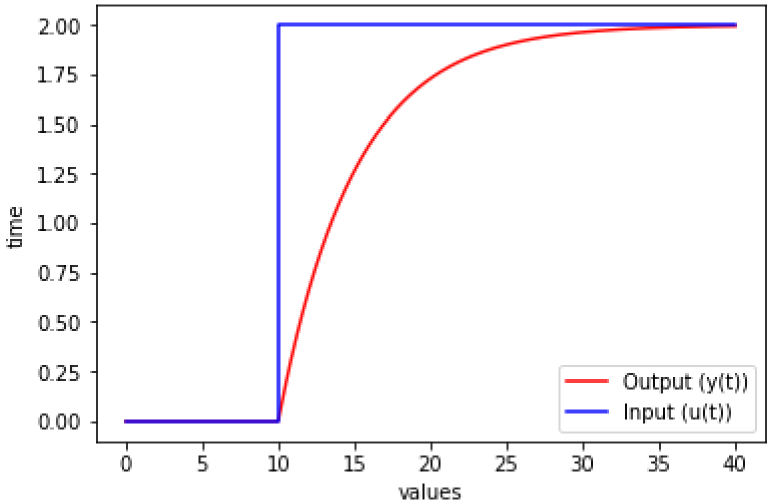
In [9]:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from scipy.integrate import odeint

#function that returns dy/dt
def model(y,t,k):
    dydt = -k*y
    return dydt

# initial condition
y0=5

# time points
t = np.linspace(0,20)

#solve ODE
k = 0.1
y1 = odeint(model,y0,t,args=(k,))
k = 0.2
y2 = odeint(model,y0,t,args=(k,))
k = 0.5
y3 = odeint(model,y0,t,args=(k,))


#plot results
plt.plot(t,y1,'r-',linewidth=2,label='k=0.1')
plt.plot(t,y2,'b--',linewidth=2,label='k=0.1')
plt.plot(t,y3,'g.',linewidth=2,label='k=0.5')
plt.xlabel('time')
plt.ylabel('y(t)')
plt.legend()
plt.show()
```



In [14]:

```
#function that returns dy/dt
def model(y,t):
    if t<10.0:
        u=0
    else:
        u=2
    dydt = (-y + u)/5.0
    return dydt

# initial condition
y0 = 0

# time points
t = np.linspace(0,40,1000)

#solve ODE
y = odeint(model,y0,t)

#plot results
plt.plot(t,y,'r-',label='Output (y(t))')
plt.plot([0,10,10,40],[0,0,2,2],'b-',label='Input (u(t))')
plt.xlabel('values')
plt.ylabel('time')
plt.legend(loc='best')
plt.show()
```



In [ ]: