Python Assignment No. 8

Ganga Meghanath

EE15B025

Objective :

- Symbolic Algebra
- Analysis of Circuits using Laplace Transforms

Code :

```
import sympy as sym
import pylab as p
import numpy as np
import scipy.signal as sp
import matplotlib.pyplot as mp

s=sym.symbols('s')

def lowpass(R1,R2,C1,C2,G,n,Vi) :
        A=sym.Matrix([[0,0,1,-1/n],[-1/(1+s*R2*C2),1,0,0],\
                [0,-G,G,1],[-1/R1-1/R2-s*C1,1/R2,0,s*C1]])
        b=sym.Matrix([[0],[0],[0],[-1/R1]])
        V=A.inv()*b
        Vo=Vi*(-1)*V[3]
        return(A,b,Vo)

def highpass(R1,R3,C1,C2,G,n,Vi) :
        A=sym.Matrix([[C1*s+1/R1+C2*s,-C2*s,0,-1/R1],\
                [-C2*s,1/R3+C2*s,0,0],[0,0,1/(n-1)+1,-1/(n-1)],[0,1,-1,1/G]])
        b=sym.Matrix([[C1*s],[0],[0],[0]])
        V=A.inv()*b
        Vo=Vi*V[0]
        return(A,b,Vo)
```

```
def impulse_response(Vo):        #Compute and plot impulse response
        w=p.logspace(0,8,801)
        ss=1j*w
        hf=sym.lambdify(s,Vo,'numpy')
        v=hf(ss)
        p.loglog(w,abs(v),lw=2)
        p.grid(True)
        p.title('Impulse Response')
        p.xlabel('w (log scale)')
        p.ylabel('log|H(jw)|')
        p.show()

def output(Vo,a,b,n):   #compute and plot time domain response
        numerator,denominator=sym.simplify(Vo).as_numer_denom()
        try :
                num,den=sym.Poly(numerator).all_coeffs(),\
                        sym.Poly(denominator).all_coeffs()
        except :
                num,den=(numerator),sym.Poly(denominator).all_coeffs()
        H=sp.lti(np.array(num,float),np.array(den,float))
        t,x=sp.impulse(H,None,np.linspace(a,b,n))
        mp.plot(t,x)
        mp.title("Output")
        mp.xlabel('Time')
        mp.ylabel(r'$v_o$(t)')
        mp.show()

def damped_sine(Vo,a,w0):        #For question 4
        Vi=w0/((s+a)**2 + w0**2)
        Vo2=Vo*Vi
        output(Vo2,0,0.6,10001)

#Question 1
A,b,Vo=lowpass(1000.0,1000.0,1e-6,1e-6,1000.0,2,1)
impulse_response(Vo)
Vi=1/s
Vo=Vo*(-Vi)
output(Vo,0,0.01,1001)

#Question 2 and 3
A,b,Vo=highpass(1000.0,1000.0,1e-6,1e-6,1000.0,2,1)
impulse_response(Vo)
Vi=2000*np.pi/(s**2 + (2000*np.pi)**2) + s/(s**2 + (2e6*np.pi)**2)
Vo1=Vo*Vi
output(Vo1,0,0.01,100001)
output(Vo1,0,0.00009,10001)
```

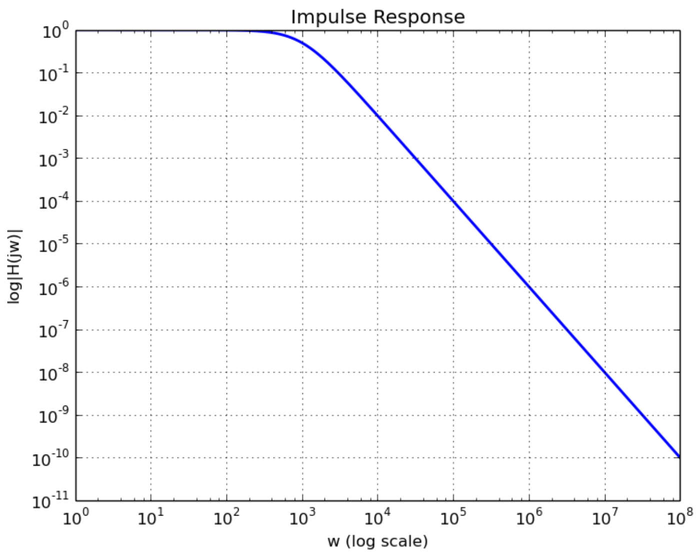
```
#Question 4
damped_sine(Vo,10,800)
damped_sine(Vo,20,500)

#Question 5
Vi=1/s
Vo3=Vo*Vi
output(Vo3,0,0.05,1001)
```

Question 1

```
def lowpass(R1,R2,C1,C2,G,n,Vi) :
        A=sym.Matrix([[0,0,1,-1/n],[-1/(1+s*R2*C2),1,0,0],\
                [0,-G,G,1],[-1/R1-1/R2-s*C1,1/R2,0,s*C1]])
        b=sym.Matrix([[0],[0],[0],[-1/R1]])
        V=A.inv()*b
        Vo=Vi*(-1)*V[3]
        return(A,b,Vo)

def impulse_response(Vo):        #Compute and plot impulse response
        w=p.logspace(0,8,801)
        ss=1j*w
        hf=sym.lambdify(s,Vo,'numpy')
        v=hf(ss)
        p.loglog(w,abs(v),lw=2)
        p.grid(True)
        p.title('Impulse Response')
        p.xlabel('w (log scale)')
        p.ylabel('log|H(jw)|')
        p.show()

def output(Vo,a,b,n):    #compute and plot time domain response
        numerator,denominator=sym.simplify(Vo).as_numer_denom()
        try :
                num,den=sym.Poly(numerator).all_coeffs(),sym.Poly(denominator
        except :
                num,den=(numerator),sym.Poly(denominator).all_coeffs()
        H=sp.lti(np.array(num,float),np.array(den,float))
        t,x=sp.impulse(H,None,np.linspace(a,b,n))
        mp.plot(t,x)
        mp.title("Output")
        mp.xlabel('Time')
```

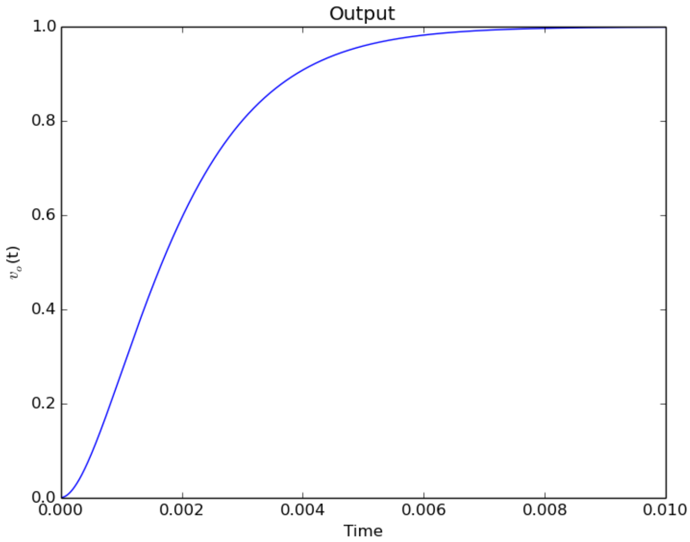
```
        mp.ylabel(r'$v_o$(t)')
        mp.show()

#Question 1
A,b,Vo=lowpass(1000.0,1000.0,1e-6,1e-6,1000.0,2,1)
impulse_response(Vo)
Vi=1/s
Vo=Vo*(-Vi)
output(Vo,0,0.01,1001)
```

This gives us a low pass filter. This means that all signal frequencies below the filter bandwidth will be allowed to pass through the filter.



Step response of the filter ,
ie, Output of the highpass filter when the input is :

$$v_i(t) = u_0(t)$$

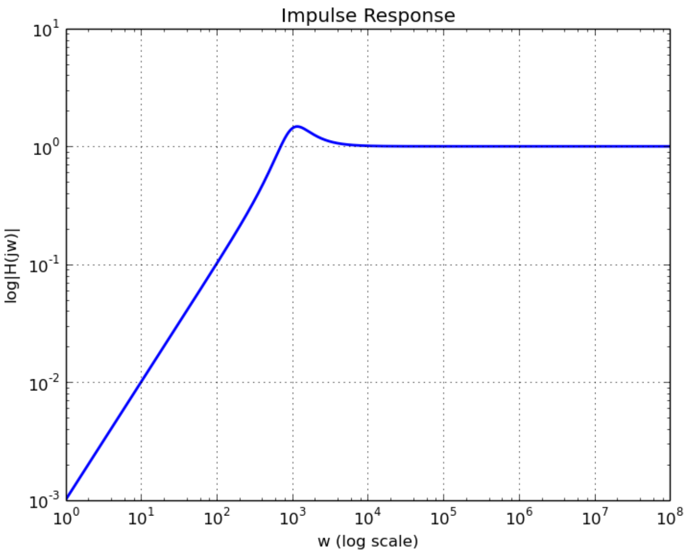


Question 2 and 3

```
def highpass(R1,R3,C1,C2,G,n,Vi) :
        A=sym.Matrix([[C1*s+1/R1+C2*s,-C2*s,0,-1/R1],\
                [-C2*s,1/R3+C2*s,0,0],[0,0,1/(n-1)+1,-1/(n-1)],[0,1,-1,1/G]])
        b=sym.Matrix([[C1*s],[0],[0],[0]])
        V=A.inv()*b
        Vo=Vi*V[0]
        return(A,b,Vo)

def impulse_response(Vo):       #Compute and plot impulse response
        w=p.logspace(0,8,801)
        ss=1j*w
        hf=sym.lambdify(s,Vo,'numpy')
        v=hf(ss)
        p.loglog(w,abs(v),lw=2)
        p.grid(True)
        p.title('Impulse Response')
        p.xlabel('w (log scale)')
        p.ylabel('log|H(jw)|')
        p.show()

def output(Vo,a,b,n):    #compute and plot time domain response
        numerator,denominator=sym.simplify(Vo).as_numer_denom()
```

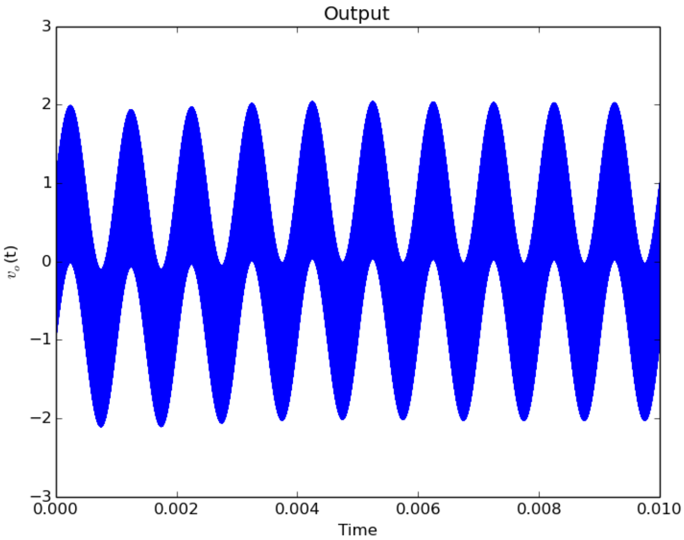
```
        try :
                num, den=sym.Poly(numerator).all_coeffs(),sym.Poly(denominator
        except :
                num, den=(numerator),sym.Poly(denominator).all_coeffs()
        H=sp.lti(np.array(num, float),np.array(den, float))
        t,x=sp.impulse(H,None,np.linspace(a,b,n))
        mp.plot(t,x)
        mp.title("Output")
        mp.xlabel('Time')
        mp.ylabel(r'$v_o$(t)')
        mp.show()

#Question 2 and 3
A,b,Vo=highpass(1000.0,1000.0,1e-6,1e-6,1000.0,2,1)
impulse_response(Vo)
Vi=2000*np.pi/(s**2 + (2000*np.pi)**2) + s/(s**2 + (2e6*np.pi)**2)
Vo1=Vo*Vi
output(Vo1,0,0.01,100001)
output(Vo1,0,0.00009,10001)
```

This gives us a high pass filter. This means that all signal frequencies above the filter bandwidth will be allowed to pass through the filter.

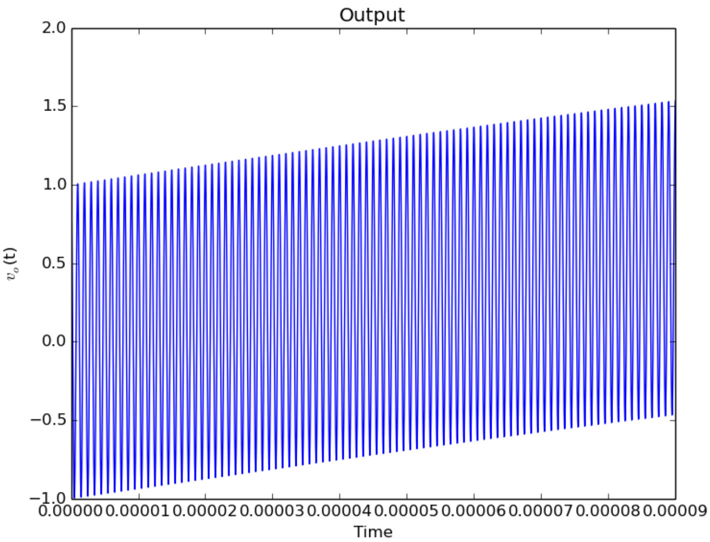


Output of the filter when the input is :

$$v_i(t) = (sin(2000\pi t) + cos(2 \times 106\pi t)) * u_0(t)$$



Zooming in :



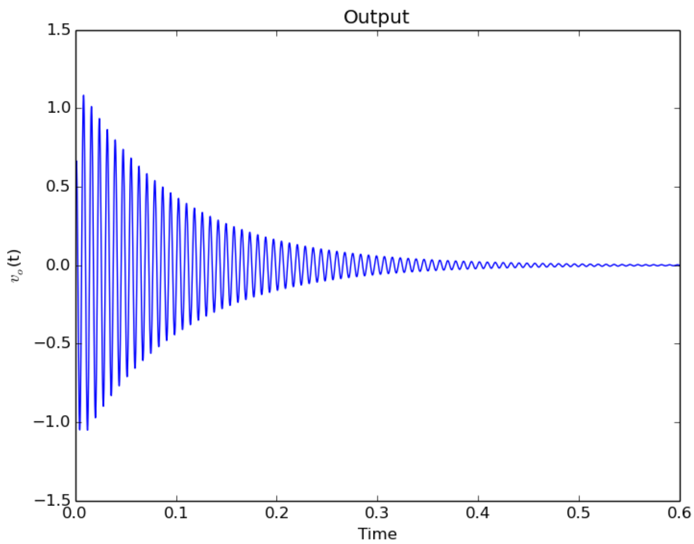
Question 4

```
def output(Vo,a,b,n):    #compute and plot time domain response
        numerator,denominator=sym.simplify(Vo).as_numer_denom()
```

```
        try :
                num,den=sym.Poly(numerator).all_coeffs(),sym.Poly(denominator
        except :
                num,den=(numerator),sym.Poly(denominator).all_coeffs()
        H=sp.lti(np.array(num,float),np.array(den,float))
        t,x=sp.impulse(H,None,np.linspace(a,b,n))
        mp.plot(t,x)
        mp.title("Output")
        mp.xlabel('Time')
        mp.ylabel(r'$v_o$(t)')
        mp.show()

def damped_sine(Vo,a,w0):          #For question 4
        Vi=w0/((s+a)**2 + w0**2)
        Vo2=Vo*Vi
        output(Vo2,0,0.6,10001)

#Question 4
damped_sine(Vo,10,800)
damped_sine(Vo,20,500)
```

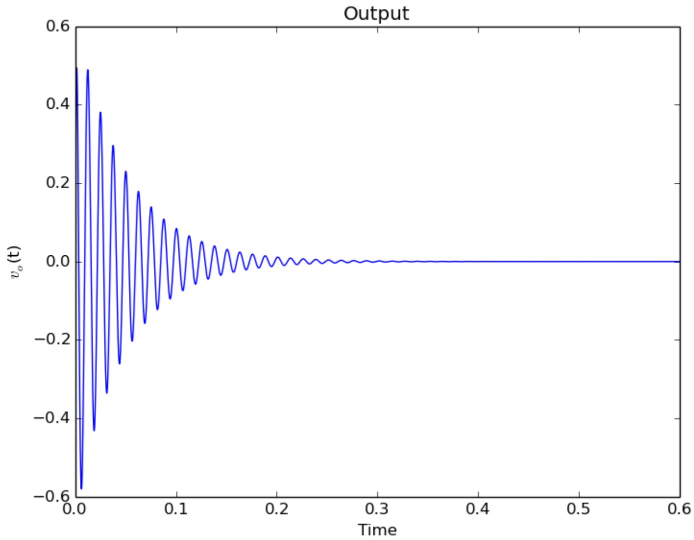
Output of the high pass filter when the input is :

$$v_i(t) = e^{-10t} sin(800t)$$



Output of the highpass filter when the input is :

$$v_i(t) = e^{-20t} sin(500t) * u_0(t)$$

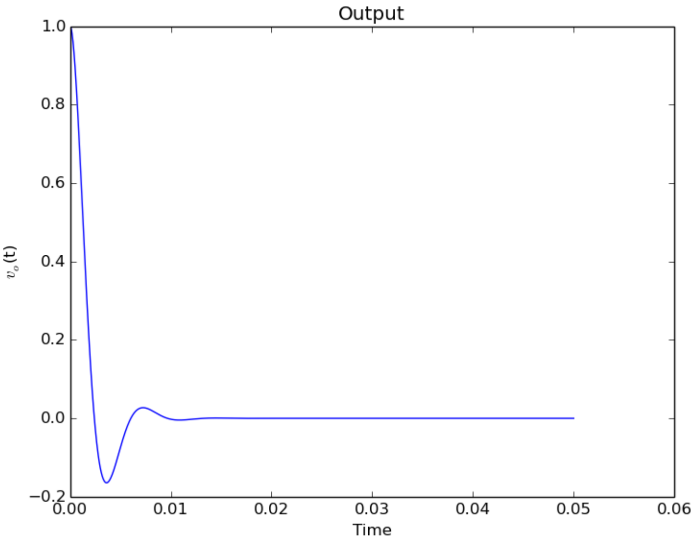


Question 5

```
def output(Vo,a,b,n):    #compute and plot time domain response
        numerator,denominator=sym.simplify(Vo).as_numer_denom()
        try :
                num,den=sym.Poly(numerator).all_coeffs(),sym.Poly(denominator
        except :
                num,den=(numerator),sym.Poly(denominator).all_coeffs()
        H=sp.lti(np.array(num,float),np.array(den,float))
        t,x=sp.impulse(H,None,np.linspace(a,b,n))
        mp.plot(t,x)
        mp.title("Output")
        mp.xlabel('Time')
        mp.ylabel(r'$v_o$(t)')
        mp.show()

#Question 5
Vi=1/s
Vo3=Vo*Vi
output(Vo3,0,0.05,1001)
```

Output of the highpass filter when the input is :

$$v_i(t) = u(t)$$


Output
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
v_o(t)
0.00
0.01
0.02
0.03
0.04
0.05
0.06
Time