APPLIED PROGRAMMING LAB
(Week - 7)
Laplace Transform

Shubhangi Ghosh
EE15B129
Department of Electrical Engineering

March 15, 2017

1 INTRODUCTION:

This report presents the use of certain polynomial functions in the numpy library and also explores various functions of signals toolbox and their usefulness in computing various aspects of the system transfer function and impulse response.

1.1 Polynomial Functions:

In this code, computation of Fourier coefficients are illustrated for the functions:

S.No.	Function	Description
1.	poly1d	Defines polynomials upon passing coefficients as an array
2.	polyadd	Adds two polynomials
3.	polymul	Multiplies two polynomials and generates an array of
		coefficients of resulting polynomial.

1.2 Signals Toolbox:

This is imported as 'import scipy.signal as sp'.

S.No.	Function	Description
1.	lti	H=sp.lti([1,2,1],[1,2,1,1])
		Defines a transfer function on specifying numerator and denominator polynomials.
		w,S,phi=H.bode()
		w -> Frequency
		S -> Magnitude
		phi -> Phase
2.	impulse	t,x=sp.impulse(H,None, linspace(0,10,101))
		Computes impulse response of a transfer function given initial conditions.
		The initial conditions are given as values of each of the time domain derivatives,
		in order starting from lowest derivative to highest, in an array.
3.	lsim	t,y,svec=sp.lsim(H,u,t)
		This simulates y=u*h

2 BRIEF CODE EXPLANATION:

2.1 Functions Used In Code:

1. $f(t, coeff)$:
   Computes $cos(\alpha t)e^{-0.05t}$ given time vector t, and time coefficient α.

2. $H(coeff)$:
   Computes transfer function, $\frac{1}{1+\alpha^2}$ given time coefficient α.

3. $impulse\_res(decay, color)$:
   Computes impulse response of the transfer function, $\frac{s+\omega}{((s+\omega)^2+\alpha^2)(s^2+\alpha^2)}$, where ω = decay(given), $\alpha = 1.5s^{-1}$,
   in this case, and plots $w.r.t.$ time in specified colour.

- First, we find transfer function using lti from signal toolbox.
- Then we find impulse response from transfer function using impulse from signal toolbox.

2.2 Problem Statements:

2.2.1 Problem 1.

1. We find the time response of a spring satisfying

$$x + 2.25\ddot{x} = f(t). \tag{1}$$

2. .

$$f(t) = cos(\alpha t)e^{-\omega t} \tag{2}$$

3. Using (1) and Laplace transform properties,$X(s) = \frac{F(s)}{s^2+2.25}$.
4. We know that $F(s) = \frac{s+\omega}{((s+\omega)^2+\alpha^2)(s^2+\alpha^2)}$.
5. We find $x(t)$ using the function $impulse\_res(decay, color)$.
6. We do this for values $\alpha = 1.5s^{-1}$and $\omega = 0.50s^{-1}$and $\omega = 0.05s^{-1}$.

2.2.2 Problem 2.

1. We do the same as the previous problem except that we keep ω fixed and vary α from 1.4 to 1.6 in steps of 0.05.
2. This time, we find the transfer function of the system and use convolution properties using $lsim$, to determine output signal.

2.2.3 Problem 3.(Coupled Springs)

The coupled spring differential equtions are:

$$\ddot{x} + (x - y) = 0 \tag{3}$$

$$\ddot{y} + 2(y - x) = 0 \tag{4}$$

On solving them in Laplace Domain, with the initial condition $x(0) = 1$, all higher derivatives being 0,

- The Laplace domain equations are:

$$s^2X(s) - sx(0) + X(s) - Y(s) = 0 \tag{5}$$

$$s^2Y(s) + 2(Y(s) - X(s)) = 0 \tag{6}$$

- We substitute $Y(s)$ from (5) in (6) to obtain,

$$X(s) = \frac{s^2+2}{s^3+3s}$$

$$Y(s) = \frac{2}{s^3+3s}$$

- We plugin these transfer functions in the *impulse* function, to obtain impulse response.

2.2.4 Problem 5.(Bode Plot of RLC Circuit)

1. The transfer function of the given RLC Circuit is:

$$H(s) = \frac{1}{s^2LC + sL/R + 1}$$

2. We substitute values of $R, L\, and\, C$ and plot magnitude and phase response of transfer function using $H.Bode()$.

2.2.5 Output of RLC Circuit By Given Input

1. We know the time-domain input and the frequency domain transfer function of the system.
2. We thus use lsim to find time-domain output by convolution.
3. The initial part is the transient response of the system. Two natural frequencies occur at the roots of the polynomial, $s^2LC + sL/R + 1$, at $-\frac{L}{R} \pm \sqrt{\frac{L^2}{R^2} - 4 * LC}$.
4. Since these poles are in the Left Half Plane, they die down eventually, giving rise to a stable system.
5. What remains is the steady state response which is a sinusoid, since the input is sinusoidal.

2.3 CODE:

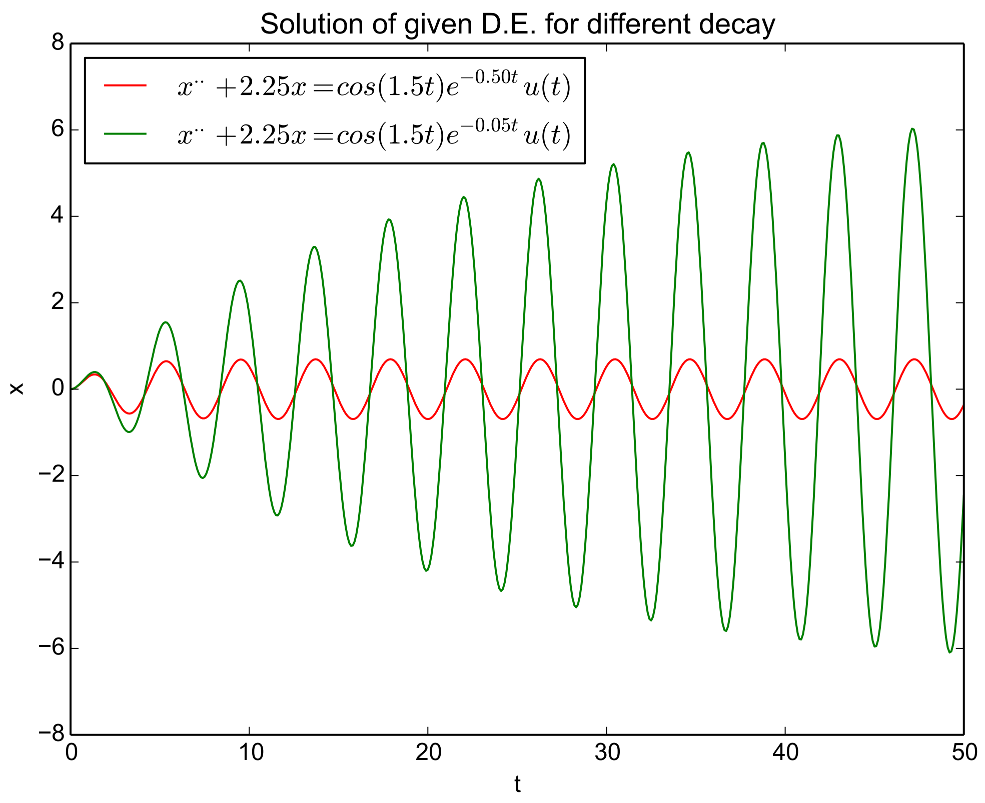
```
#Importing header files
import numpy as np
from scipy import *
from matplotlib.pyplot import *
from scipy.integrate import quad
from math import *
from pylab import *
import mpl_toolkits.mplot3d.axes3d as p3
import sys
import scipy.signal as sp
# computes f(t) as mentioned in question for different time coefficients
def f(t,coeff):
return cos(coeff*t)*exp(-0.05*t)
# computes frequency response given numerator and denominator polynomials
def H(coeff):
```

```
Num = np.poly1d([1.0])
Den = np.poly1d([1.0,0.0,coeff**2])
H = sp.lti(Num,Den)
return H
# computes impulse response given numerator and denominator polynomials
of transfer function
def impulse_res(decay,color):
Num = np.poly1d([1.0,decay])
Den_1 = np.polyadd(np.polymul([1.0,decay],[1.0,decay]),[2.25])
Den_2 = np.poly1d([1.0,0.0,2.25])
Den = np.polymul(Den_1,Den_2)
H = sp.lti(Num,Den) #transfer function(frequency response)
#using impulse function to compute impulse response from transfer function
t,x = sp.impulse(H,None,linspace(0,50,501))
plot(t,x,color)
return H
#plotting solution of a given differential equation for diffrent valus of decay paramet
title('Solution of given D.E. for different decay')
xlabel('t')
X = impulse_res(0.5,'r')
X = impulse_res(0.05,'g')
legend((r'$x^{..}+2.25x=cos(1.5t)e^{-0.50t}u(t)$','$x^{..}+2.25x=cos(1.5t)e^{-0.50t}u(t
savefig('de_decay.pdf',format='pdf')
show()
#plotting solution of a given differential equation for diffrent values of time coeffic
title(r'Solution of $x^{..}+2.25x=cos(\alpha*t)e^{-0.50t}u(t)$')
xlabel('t')
for i in arange(1.40,1.65,0.05):
t=linspace(0,50,501)
t,y,svec=sp.lsim(H(i),f(t,i),t)
plot(t,y,label=r"$\alpha$=%.2f" %i)
legend(loc='best')
savefig('de_time.pdf',format='pdf')
show()
#solution to differential equation representing coupled springs
#x
title(r'Solution to coupled equations: $x^{..}+(x-y)=0,y^{..}+2(y-x)=0$')
xlabel('t')
Num = np.poly1d([1.0,0.0,2.0])
Den = np.poly1d([1.0,0.0,3.0,0.0])
X = sp.lti(Num,Den)
t,x = sp.impulse(X,None,linspace(0,20,501)) #initial condition vector added
plot(t,x,'r')
#y
Num = np.poly1d([2.0])
Den = np.poly1d([1.0,0.0,3.0,0.0])
X = sp.lti(Num,Den)
t,y = sp.impulse(X,None,linspace(0,20,501))
plot(t,y,'g')
legend(('x','y'),loc='best')
savefig('impulse.pdf',format='pdf')
show()
#RLC circuit
```

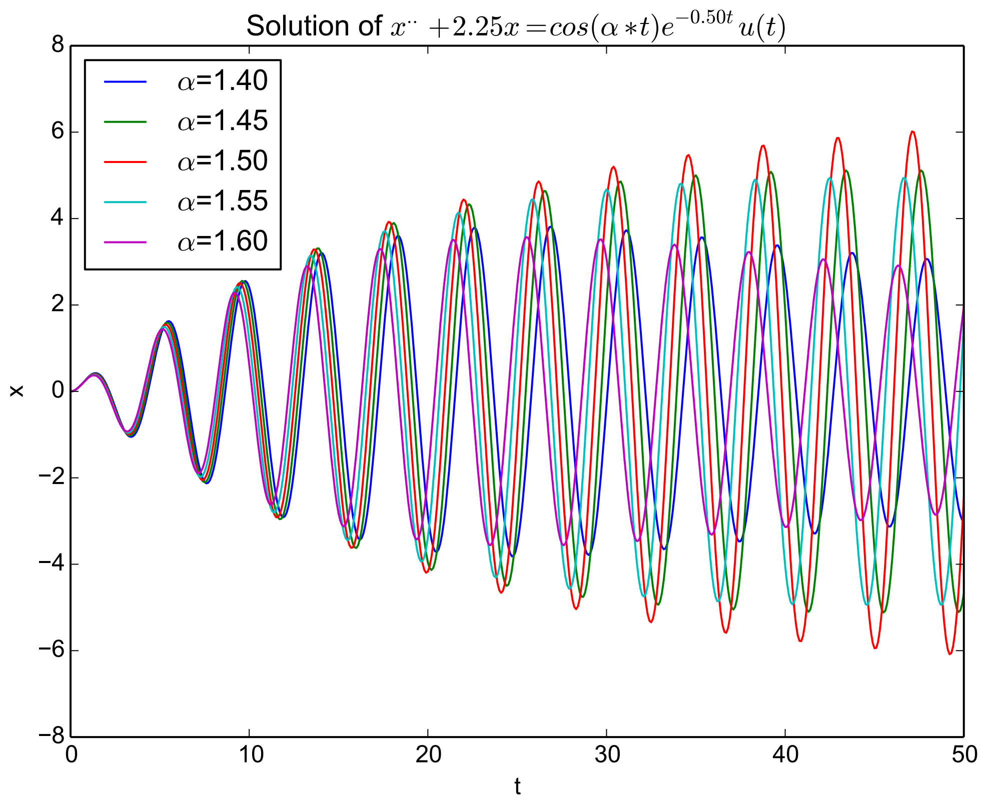
```
#Bode plot for transfer function of an RLC circuit
title('Bode plot of RLC filter')
Num = np.poly1d([1.0])
Den = np.poly1d([10.0**(-12),10.0**(-4),1.0])
H = sp.lti(Num,Den)
w,S,phi=H.bode()
#magnitude response
subplot(2,1,1)
xlabel(r'$\omega$')
ylabel(r'$\|H(e^{j\omega})\|$ in dB')
semilogx(w,S)
#frequency response
subplot(2,1,2)
xlabel(r'$\omega$')
ylabel(r'$ang(H(e^{j\omega}))$ in degrees')
semilogx(w,phi)
savefig('bode.pdf',format='pdf')
show()
#RLC filter output for a given input
title(r'RLC filter output for $v_i(t)=cos(10^{3}t)-cos(10^{6}t)$')
t=arange(0,0.03,10**-6)
v_i = np.cos(1000.0*t)-np.cos((10.0**6)*t)
t,y,svec=sp.lsim(H,v_i,t)
xlabel(r'$t$')
ylabel(r'$x$')
plot(t,y)
savefig('rlc.pdf',format='pdf')
show()
#Transient of RLC filter output for a given input
title(r'RLC filter output transient for $v_i(t)=cos(10^{3}t)-cos(10^{6}t)$')
t=arange(0,30.0*10**(-6),10**-6)
v_i = np.cos(1000.0*t)-np.cos((10.0**6)*t)
t,y,svec=sp.lsim(H,v_i,t)
xlabel(r'$t$')
ylabel(r'$x$')
plot(t,y)
savefig('rlc_trans.pdf',format='pdf')
show()
```

2.4 PLOTS:

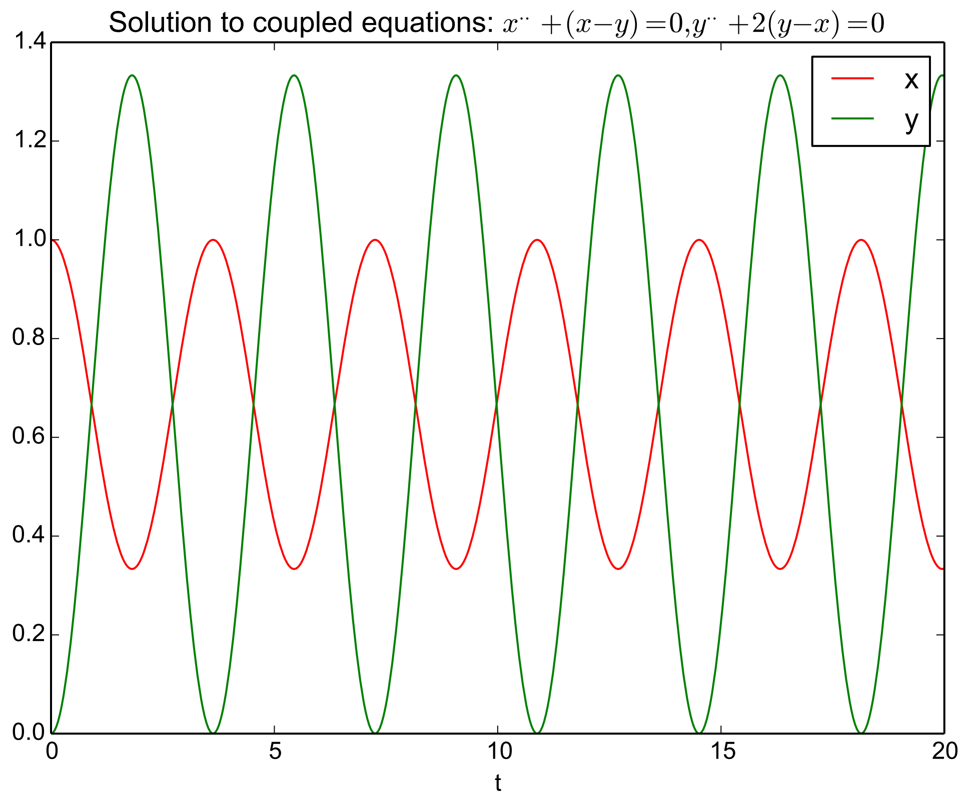
2.4.1 Plot of Solution for different decay parameters, ω



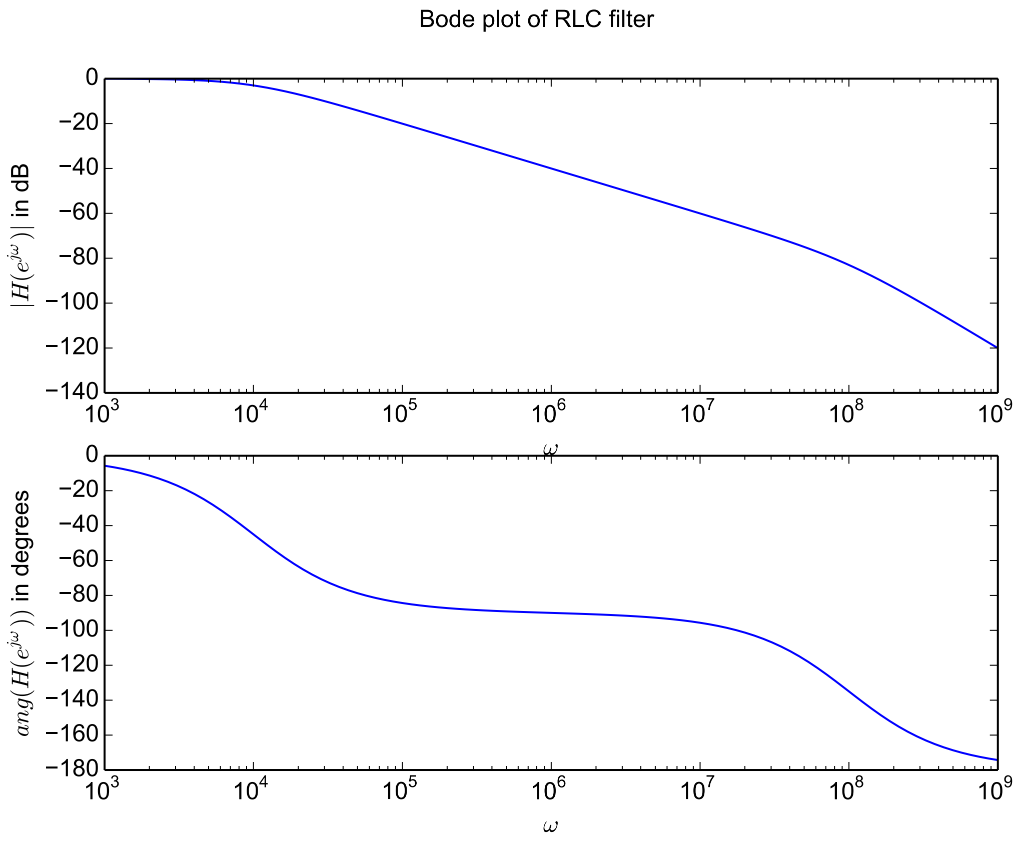
2.4.2 Plot of Solution for different time coefficients, α



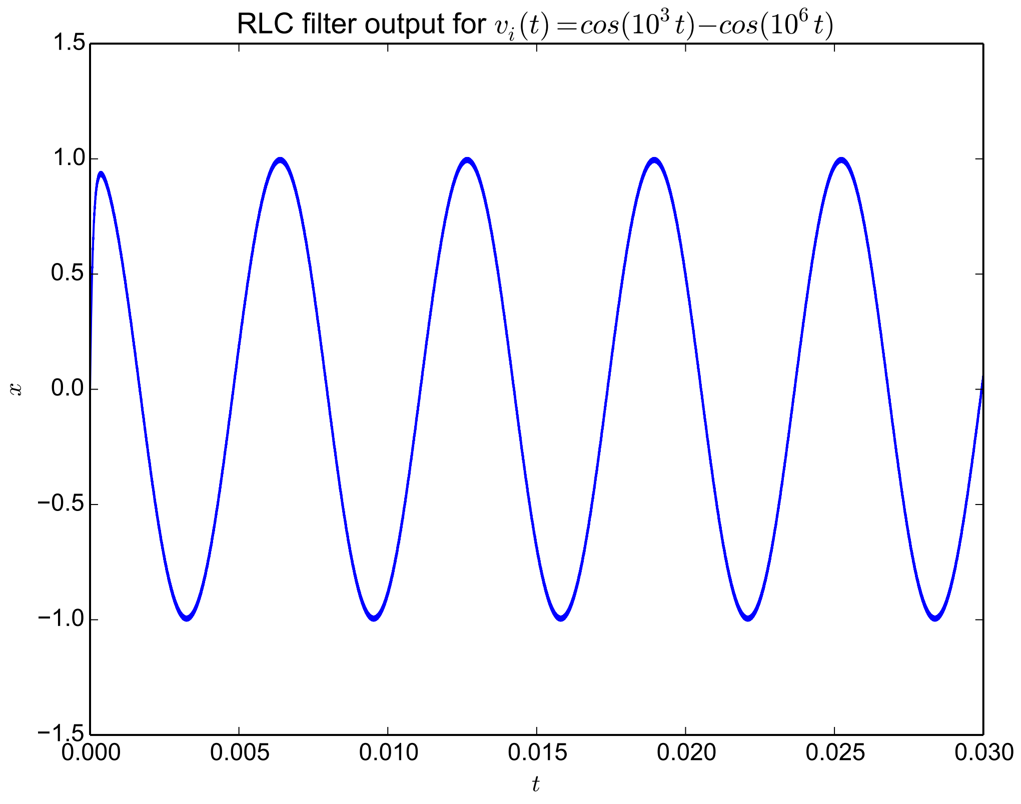
2.4.3 Coupled Springs



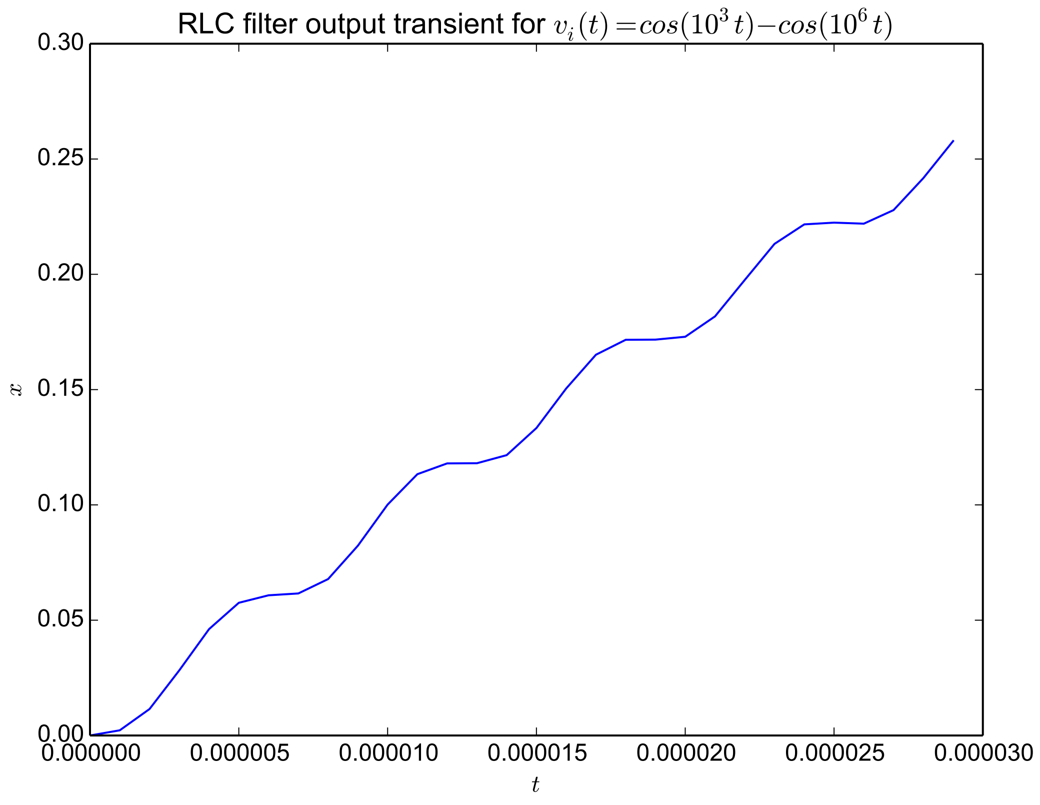
2.4.4 Bode Plot of Magnitude and Phase Response of RLC Circuit



2.4.5 Steady State Response of an RLC circuit



2.4.6 Transient Response of an RLC circuit


RLC filter output transient for $v_i(t) = cos(10^3 t) - cos(10^6 t)$
x
t