Solving Second Order Ordinary Differential Equations with Python

Chris Emery

December 1, 2016

Abstract

This report focuses on solving second order differential equations using the Python programming language. For this specific case, we will be solving a second order ODE represented by a series RLC circuit. First, we will show the derivation of the equation used in python and then describe/explain the code used.

1 The Derivation

To solve a second order ODE for a series RLC ciruit like the one below we will apply mesh analysis. Doing so will result in the equation

$$V_s = V_c + V_L + V_R \tag{1}$$

Applying the concept of ODE yeilds the following equation

$$ax'' + bx' + c = 0 \tag{2}$$

Applying the above equation to our circuit and measuring the output at the capacitor, we will take the derivitive at C in terms of voltage.

we now plug this information back into equation (2) to obtain:

$$L\frac{d^2V}{dt^2} + R\frac{dV}{dt} + VC = V_s \tag{3}$$

if we let $x = VC$, then $x' = C\frac{dV}{dt}$ and $x'' = C\frac{d^2V}{dt^2}$
then equation (3) becomes:

$$LCx" + RCx' + x = 0 \tag{4}$$

Solving for X" yeilds:

$$x'' = \frac{V_s - VC - RCx'}{LC} \tag{5}$$

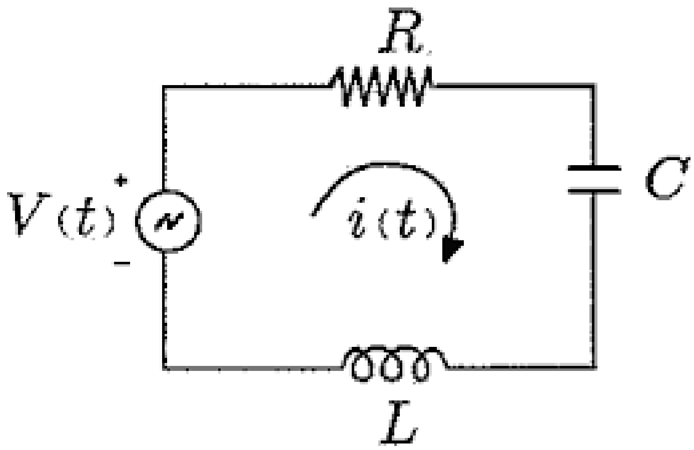


Figure 1: Series RLC circuit

In order to format this equation so that we can use it in our python code we will let

$$x" = s'$$

The final equation used in the Python program becomes:

$$s' = \frac{V_s - VC - RCs}{LC} \tag{6}$$

2 The Code

2.1 Solving Second Order ODE using values read from a file and output written to a file

```
from scipy import integrate
import numpy as np
import sys, csv, os
import matplotlib as plot
from pylab import *
from ctypes._endian import BigEndianStructure, LittleEndianStructure
import ctypes._endian



f=sys.argv[1]
```

```
with open(f,'r') as i:
    data = i.readlines()


file = open('Sup1.txt','w')






Vs = float(data[3])  # voltageSource
R = float(data[9])
L = float(data[7])
C = float(data[5])

file.write("The response of the circuit is: \n")

if float(C>((4*L)/(R**2))):
    file.write('Overdamped')
elif float(C==4*L/R**2):
    file.write('Critically damped')
else:
    file.write('UnderDamped')

def rlc(A,s):
    Vc,s=A   #prepares equation for array used in res

    res=np.array([s,(Vs-Vc-(R*C*s))/(L*C)]) #ODE in terms of voltage observed at the Capacit
    return res

time=np.linspace(0.0,0.5e-6,1000) #array of time intervals

vc,s = integrate.odeint(rlc,[0.0,0.0],time).T  #(function,initial values,time)
f.close()



i=1.0e-9*s #current
file.write(str(i))
figure('Voltage vs Time')
plot(time,vc)
xlabel('t')
ylabel('Vc')
savefig('Voltage vs Time.png')
```

```
show()
file.close()
```

2.2 Inputing your own file:

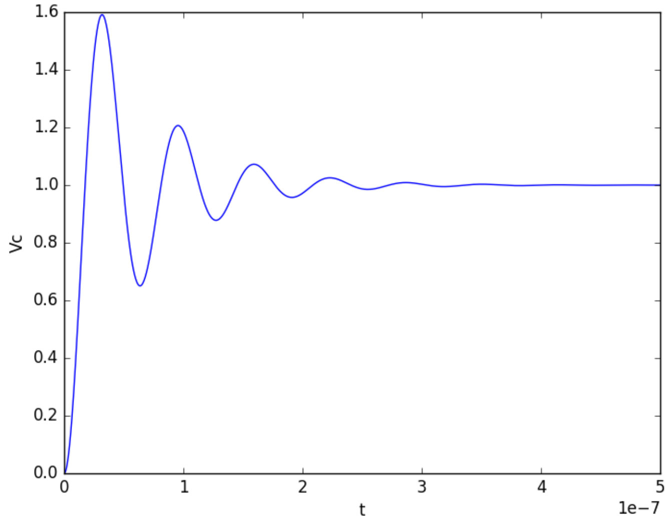
When entering your own input file to be used in the program, please ensure your code is in the format like the example below

```
#upload a file in the same format as the below example:
'''
To solve a second order RLC circuit, provide the following values:

Initial Voltage source (Vs):
1.0
Capacitor (C-in Faradads):
1.0e-9
Inductor (L-in Henrys):
100.00e-9
Resistance (R-in Ohms):
3.3
'''
#to ensure accuracy in the code, you can directly copy and paste the above template
#into your own file and change values as necessary.
```

2.3 Using hard coded files:

```
from scipy import integrate
import numpy as np
import os, sys
import matplotlib as plot
from pylab import *



f=open('Sup.txt','r')
file = open('Sup1.txt','w')

data = f.readlines()
Vs = float(data[3])  # voltageSource
R = float(data[9])
L = float(data[7])
C = float(data[5])

file.write("The response of the circuit is: \n")
```

```
if float(C>((4*L)/(R**2))):
    file.write('Overdamped\n')
elif float(C==4*L/R**2):
    file.write('Critically damped\n')
else:
    file.write('UnderDamped\n')

def rlc(A,t):
    Vc,s=A   #prepares equation for array used in res

    res=np.array([s,(Vs-Vc-(R*C*s))/(L*C)]) #ODE in terms of voltage observed at the Capacit

    return res

time=np.linspace(0.0,0.5e-6,1000) #array of time intervals

vc,s = integrate.odeint(rlc,[0.0,0.0],time).T  #(function,initial values,time)

i=1.0e-9*s #current

figure('Voltage vs Time')
file.write('Current Data: \n')
file.write(str(i))
plot(time,vc)
xlabel('t')
ylabel('Vc')
savefig('Voltage vs Time(user input).png')
show()
```



vs Time.png

Figure 2: Underdamped Circuit response

3 Conclusion

Both codes used above were able to successfully complete the objective of this report. We were able to analyze a given RLC ciruit by applying second order differential equations in python. We showed that this can be done utilizing a variety of different user input options (real time user input and read/write using files)

4 References

Reed,Dwight "Solving Second Order Differential Equations" dwightreid.com March 6,2013. November 2016

Alexander, Charles K. and Sadiku, Matthew N.O.Fundamentals of Electric Circuits, fifth edition.

Krummel, Michelle. LaTeX Tutuorial. youtube.com November 2016