Name: John Paolo Acosta

Class: EE 104 Fall 2021

About this code:

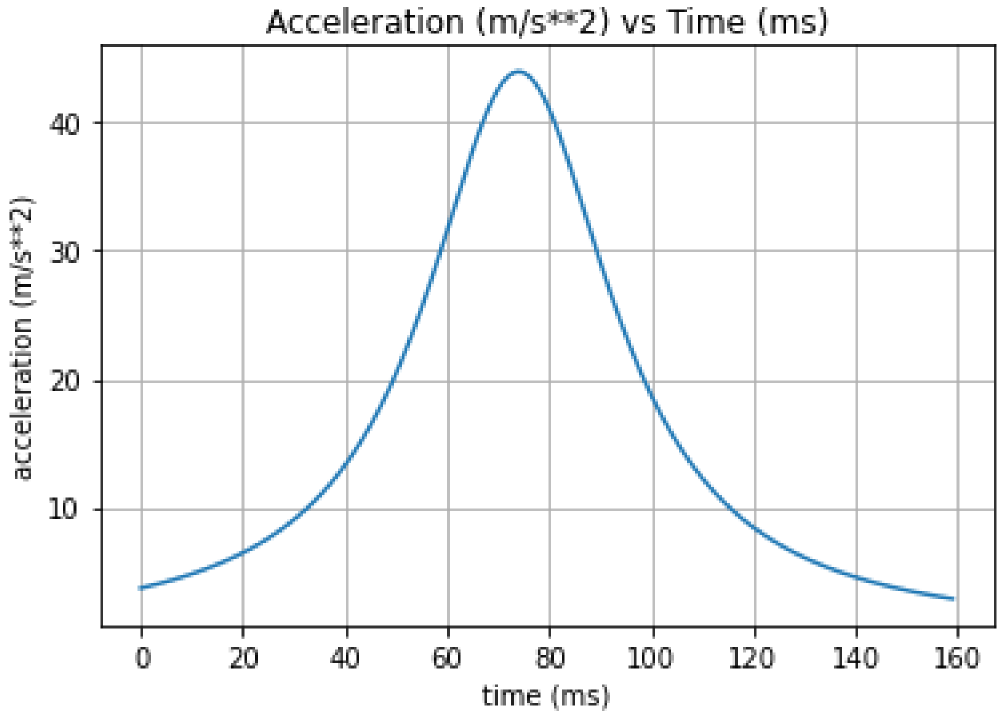
Below are 4 different codes: HIC, Electron Force, RLC, and ER modeling. The HIC model uses an integration model to display the HIC output based on the deceleration. The Electron force uses integration to calculate the electron force. The RLC uses a 2nd order or higher circuit and determines the damping by plotting based on voltage and current. The ER models a hospital with an ERU and an IRU room to help hospital determine the amount of help they need to run the hospital for COVID 19.

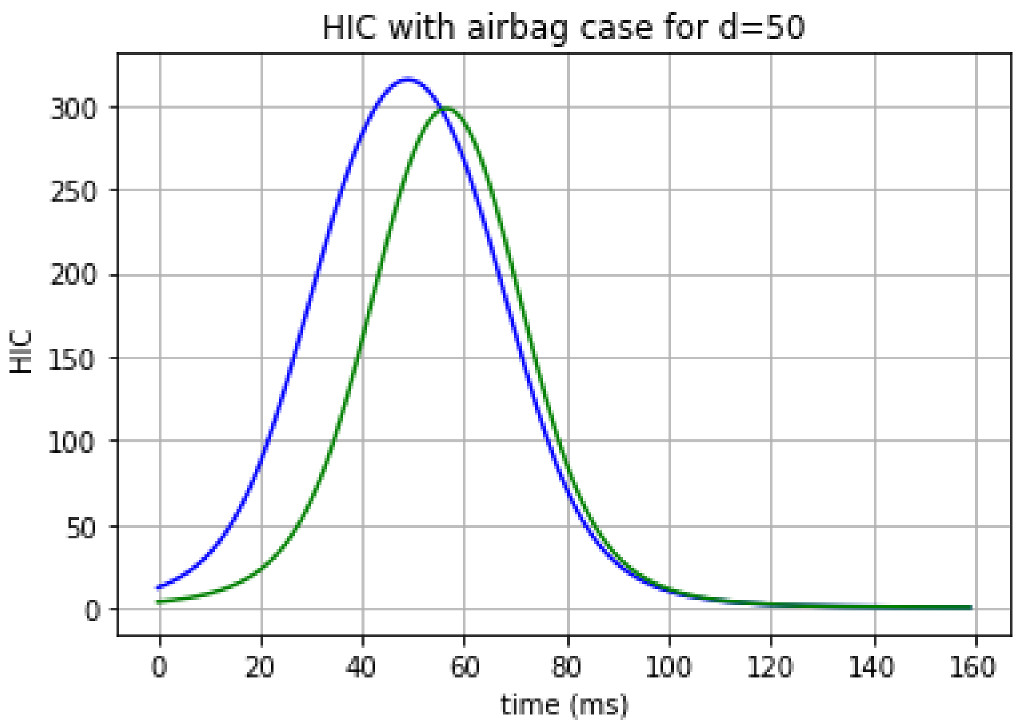
What does this code do:

This code below defines the deacceleration function with an airbag for car crashes. The HIC is then defined by taking the integral if the deacceleration function over time. The plots model the acceleration and the HIC with airbag.

```
#creates acceleration function
def acceleration(t):
    return (22000/((t-74)**2 + 500))

#creates HIC function
def hic(d, t):
    return (d*pow(1/d*integ.quad(acceleration,t,t+d)[0], 2.5)/1000)
```

```python
#plots the HIC for when d = 50 and d = 35
htd50 = [hic(50,t) for t in time]
htd35 = [hic(35,t) for t in time]
```





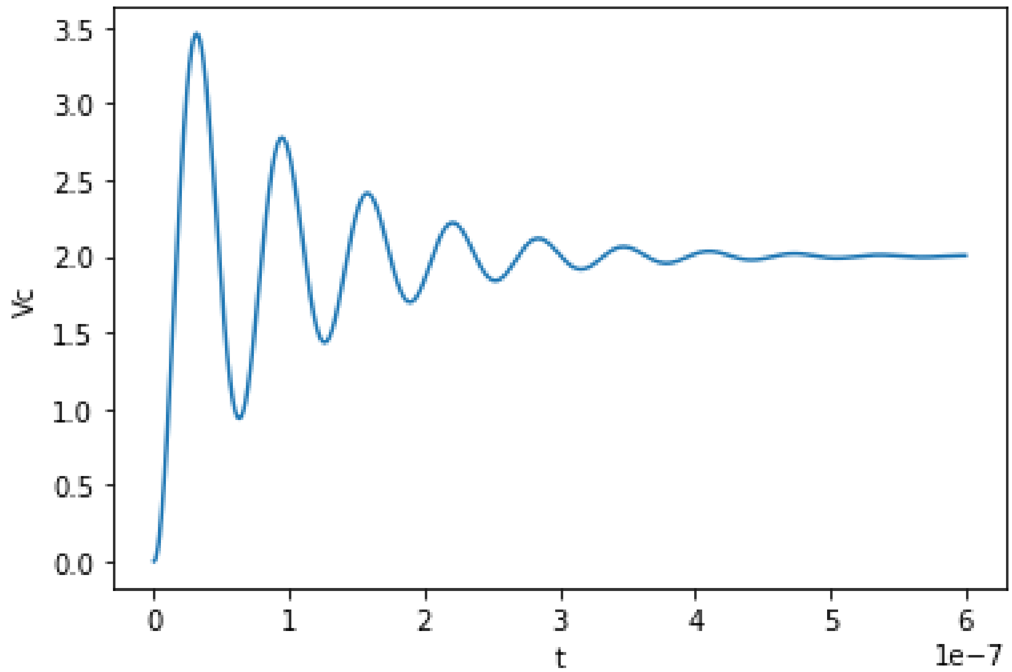
The electron force is defined below and then applied to the work function. The limits a and b are randomized from 1 to 10 in picometers.

```
#defines electron force function
def electronforce(x):
    k = 9*10**(9)
    q = 1.6*10**(-19)
    return (k*q*q/(x**2))

#defines work function
def work(a,b):
    return (integ.quad(electronforce,a,b)[0])

#generates random number between 1 and 10 for a and b in picometers
a = randrange(1, 10) * 10 ** (-12)
b = randrange(1, 10) * 10 ** (-12)
```

```
In [4]: runfile('C:/Users/siopa/Documents/SJSU/Fall 2021/EE 104/Labs/Lab5/
Numerical_Integration_Electron_Force.py', wdir='C:/Users/siopa/Documents/SJSU/Fall 2021/EE
104/Labs/Lab5')
Work from a to b:
2.0480000000000002e-17

In [5]: runfile('C:/Users/siopa/Documents/SJSU/Fall 2021/EE 104/Labs/Lab5/
Numerical_Integration_Electron_Force.py', wdir='C:/Users/siopa/Documents/SJSU/Fall 2021/EE
104/Labs/Lab5')
Work from a to b:
1.152e-17
```

The RLC circuit is defined in the code below. The circuit is then integrated over time and the plot is then developed in the graph below.

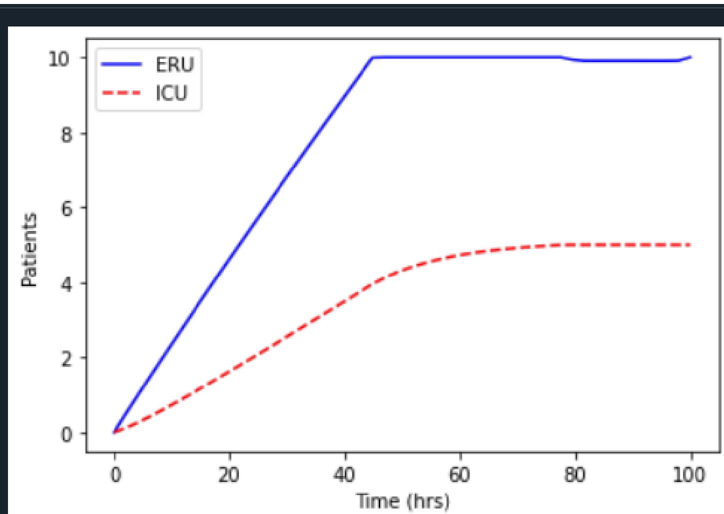
```python
def rlc(A,t):
    Vc,x=A
    V = 2.0 #voltageSource
    R = 1.0
    L = 50.0e-9 #50nH
    C = 2.0e-9 #2nF
    res=np.array([x,(V-Vc-(x*R*C))/(L*C)])
    return res
```



The code below models an ER hospital for COVID 19. The hospital has 2 rooms ERU and ICU, where the ERU diagnoses patients for the disease and the ICU takes care of the disease parents by putting them on ventilators. This creates an inflow and outflow model similar to a water tank model, and focuses on the when the hospital will have too many patients coming in. The plot below models when the nurses in the ICU have to stop letting people in and the when the ICU gets filled up over the entire day in 24 hours.

```
#Paolo Acosta
#013117104
import numpy as np
import matplotlib.pyplot as plt
from gekko import GEKKO

m = GEKKO()

# integration time points
m.time = np.linspace(0,100)

# constants
rate1 = 0.5 #rate of eru to icu
rate2 = 0.7 #rate to exit icu

nurses_eru = 20 #number of nurses in eru
nurses_icu = 10 #number of nurses in icu

ventilators = 2 #number of ventilators
clean = .5 #time to clean ventilators
use = 2 #time to clean
diagnosis = .5 #time to diagnose
Ac_eru = nurses_eru/diagnosis
Ac_icu = nurses_icu/(ventilators*(use + clean))
# inflow
qin1 = 10  # people/hour

# variables
eru = m.Var(value=0,lb=0,ub=10)
icu = m.Var(value=0,lb=0,ub=5)
overflow1 = m.Var(value=0,lb=0)
overflow2 = m.Var(value=0,lb=0)

# outflow equations
qin2 = m.Intermediate(rate1 * eru**0.5)
qout1 = m.Intermediate(qin2 + overflow1)
qout2 = m.Intermediate(rate2 * icu**0.5 + overflow2)
```

```python

# mass balance equations
m.Equation(Ac_eru*eru.dt()==qin1-qout1)
m.Equation(Ac_icu*icu.dt()==qin2-qout2)

# minimize overflow
m.Obj(overflow1+overflow2)

# set options
m.options.IMODE = 6 # dynamic optimization

# simulate differential equations
m.solve()

# plot results
plt.figure(1)
plt.plot(m.time,eru,'b-')
plt.plot(m.time,icu,'r--')
plt.xlabel('Time (hrs)')
plt.ylabel('Patients')
plt.legend(['ERU','ICU'])
plt.show()
```



*****INSTALL FOLLOWING PYTHON MODULES*****

import math as m

```
import numpy as np
import scipy.integrate as integ
import matplotlib.pyplot as plt
from random import randrange
from gekko import GEKKO
```