```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Mar 18 15:14:07 2020

COIS 2310H Assignment 4 Question 4 (Module)

lyapunov removed for Assignment 5

@author: davidstothers
"""

import numpy as np


global g
global N
global tmax

g=9.81

tmax = 1800.0
N = int((tmax*100)+1)

def pendulum_EC(Lengthec,theta0ec=1.0,omega0ec=0.0,forceqec=0.0,\
                drivefreqec=1.0,driveampec=0.0):
    """
    Uses the Euler-Cromer method to approximate the motion of a pendulum,
    to about 0.08 radians of accuracy.
    Arguments: (theta0 = 1.0, omega0 = 0.0, forceq=0.0, drivefreq=1.0,
                driveamp=0.0)
    Returns arrays for time, position, and speed.
    """
    # Error currently sits at about 0.08 radians, or about 4.58 degrees.
    # It is not sensitive to dt.
    # I cannot find any ways to optimize the code further.

    print("Simulating EC pendulum...")
    print("drivefreq = "+str(drivefreqec))
    t=np.linspace(0,tmax,N,dtype=float)
    dt=1.0*(t[1]-t[0])

    theta=np.arange(0,N,dtype=float)
    omega=np.arange(0,N,dtype=float)

    theta[0]=theta0ec
    omega[0]=omega0ec

    for i in np.arange(0,N-1):
        omega[i+1] = 1.0*(omega[i]+(anet(Lengthec,theta[i],omega[i],forceqec,\
```

```python
                                                    drivefreqec,driveampec,t[i]))*dt)
        theta[i+1] = 1.0*(theta[i]+omega[i+1]*dt)

        if(theta[i+1]>np.pi):
            theta[i+1]-=2.0*np.pi
        elif(theta[i+1]<-1.0*np.pi):
            theta[i+1]+=2.0*np.pi

    print("Simulation done!")
    return t[:],theta[:],omega[:]

def pendulum_verlet (Lengthv,theta0v = 1.0,omega0v=0.0,forceqv=1.0,\
                     drivefreqv=1.0,driveampv=1.0):
    """
    Uses the Verlet method to simulate a pendulum of the given length.

    Arguments:
        theta0v: The initial angle of the pendulum. Defaults to 1 radian.
        omega0v: The initial speed of the pendulum. Defaults to 0 rad/s.
        forceqv: The damping constant. Defaults to 1.
        drivefreqv: The driving force frequency. Defaults to 1 Hz.
        driveampv: The driving force amplitude. Defaults to 1.

    """
    print("Simulating Verlet pendulum...")
    print("drivefreq = "+str(drivefreqv))
    global g
    global N
    global tmax
    tv = np.linspace(0,tmax,num=N,dtype=float)
    dtv = tv[1]-tv[0]

    thetav=np.zeros((N),dtype=float)
    omegav=np.zeros((N),dtype=float)

    thetav[0] = theta0v
    thetav[1] = thetav[0]
    omegav[0] = omega0v

    #The matching issues to EC have been isolated to the driving frequency.
    for i in np.arange(1,N-1):
        thetav[i+1] = 2.0*thetav[i]
        thetav[i+1]-= thetav[i-1]*(1.0-(forceqv*dtv)/2.0)
        thetav[i+1]+= (driveampv*np.sin(drivefreqv*tv[i])-np.sin(thetav[i])\
                      *g/Lengthv)*dtv**2
        thetav[i+1]/= 1.0+(forceqv*dtv)/2.0

        omegav[i]=(thetav[i+1]-thetav[i-1])/(2*dtv)

    mthetav=map_theta(thetav)
```

```
    print("Simulation done!")
    return tv[:],mthetav[:],omegav[:]

def strobe_theta (theta,time,driveperiod):
    dt=(time[1]-time[0])/2
    strobe = np.logical_and(np.abs(np.remainder(time[:], driveperiod)) < dt, \
                            time[:]>10.0*driveperiod)

    sth = np.zeros(np.shape(theta))
    sth = theta[strobe]

    return sth[:]

def anet(L,theta,omega,q,dfreq,damp,t):
    Fdamp = -1.0*q*omega
    Fg = -1.0*np.sin(1.0*theta)*g/L
    Fdrive = damp*np.sin(dfreq*t)

    Fnet = 1.0*(Fg+Fdamp+Fdrive)
    return Fnet

def map_theta(theta):
    for i in np.arange(theta.size):
        while (theta[i]>np.pi):
            theta[i]-=2*np.pi
        while (theta[i]<-1.0*np.pi):
            theta[i]+=2*np.pi

    return theta
```