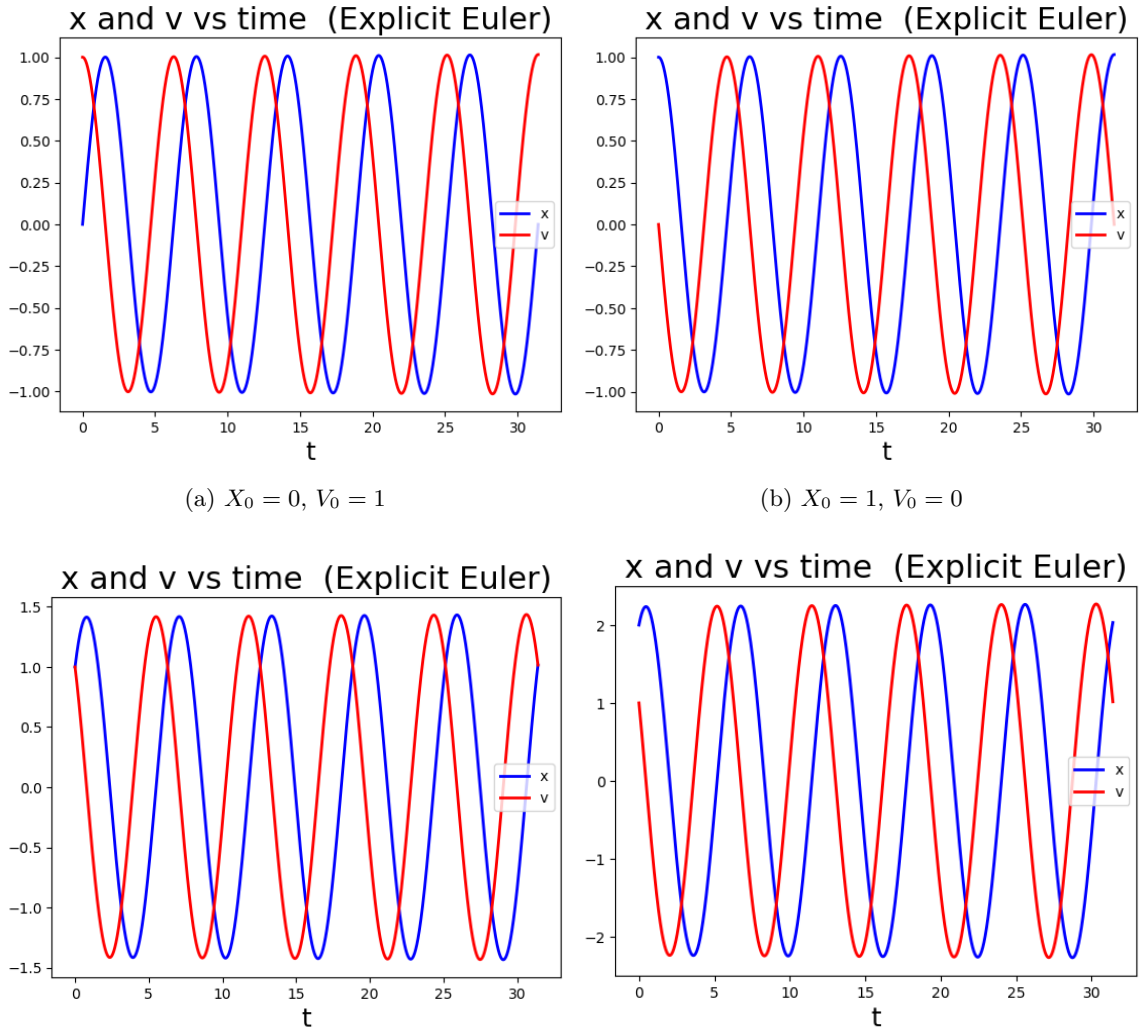
Numerical Solution of Ordinary Differential Equations **Noah Huffman**
Ph 20.3

1 Motion of a mass on a spring

First, Here are some plots of my numerical time evolutions of the system. Note that $X_0 = 0$, $V_0 = 0$ is an uninteresting initial condition because the system has not initial energy to make it dynamic.



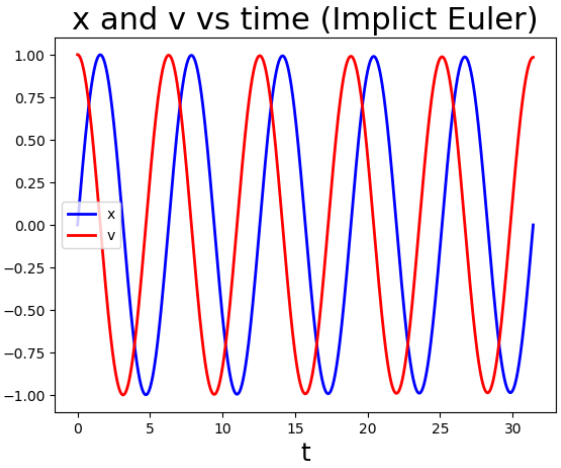
(a) $X_0 = 0$, $V_0 = 1$

(b) $X_0 = 1$, $V_0 = 0$

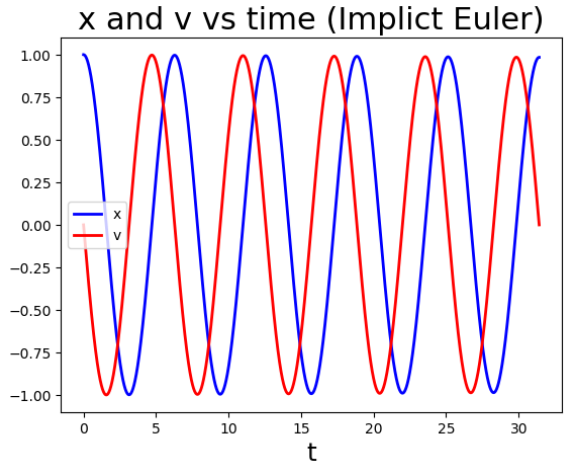
(c) $X_0 = 1$, $V_0 = 1$

(d) $X_0 = 2$, $V_0 = 1$

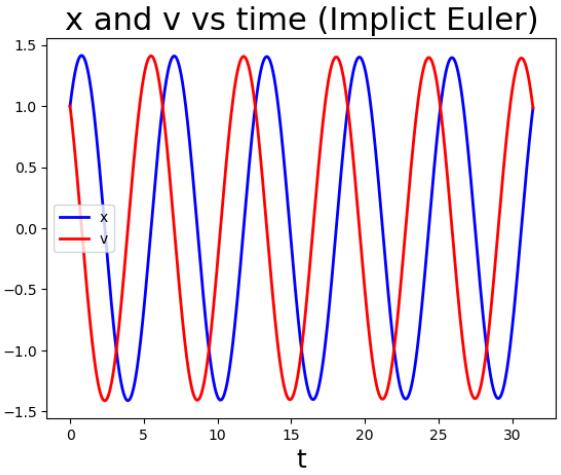
Figur 1: Explicit Euler numerical evolution of system over time for several initial conditions.



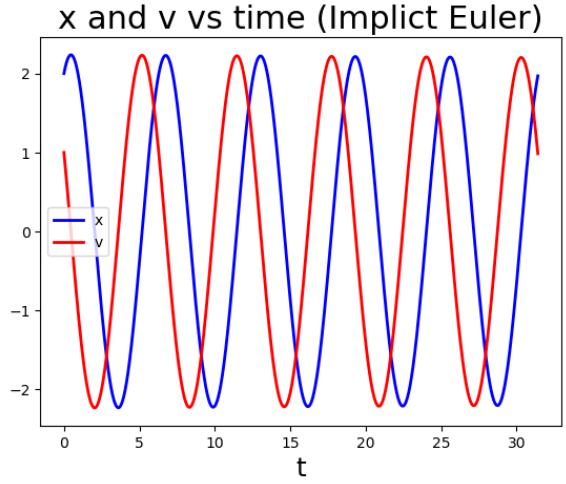
(a) $X_0 = 0$, $V_0 = 1$



(b) $X_0 = 1$, $V_0 = 0$



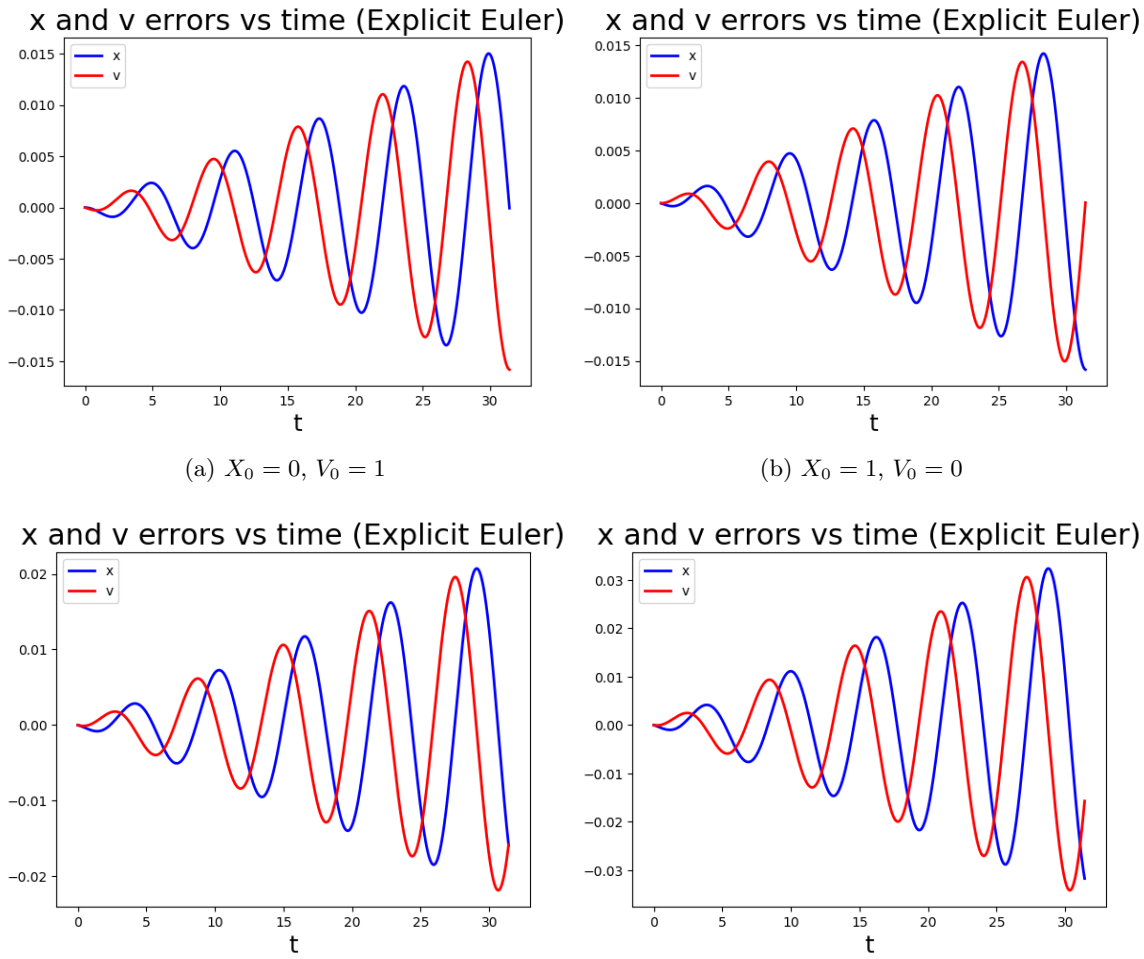
(c) $X_0 = 1$, $V_0 = 1$



(d) $X_0 = 2$, $V_0 = 1$

Figur 2: Implicit Euler numerical evolution of system over time for same initial conditions.

2 Global errors



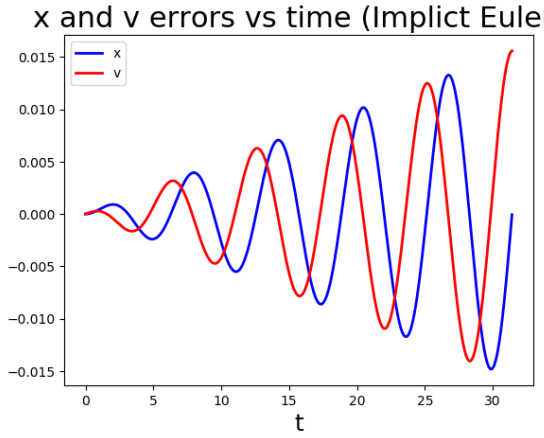
(a) $X_0 = 0$, $V_0 = 1$

(b) $X_0 = 1$, $V_0 = 0$

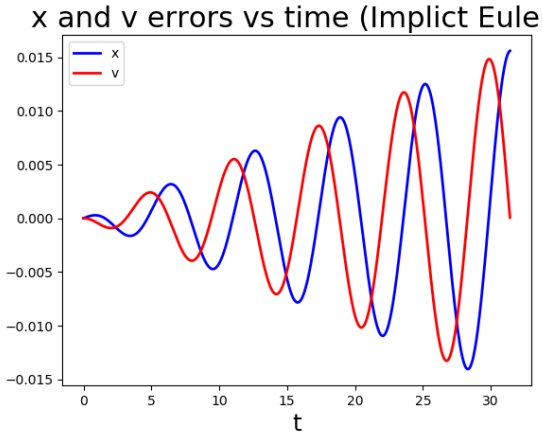
(c) $X_0 = 1$, $V_0 = 1$

(d) $X_0 = 2$, $V_0 = 1$

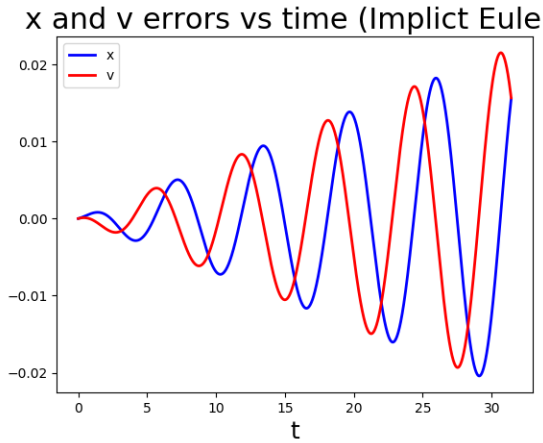
Figur 3: Explicit Euler global errors. Regardless of the choice of initial conditions, the errors grow over time.



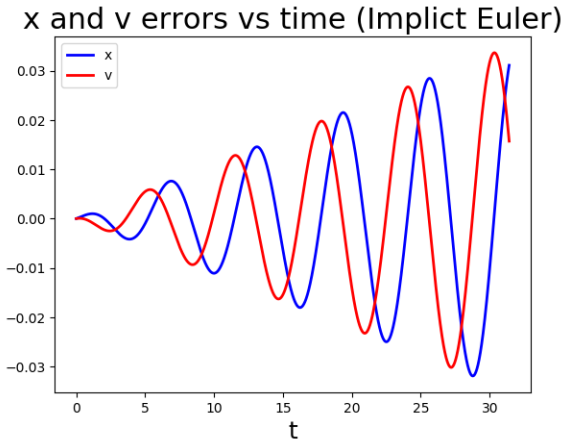
(a) $X_0 = 0$, $V_0 = 1$



(b) $X_0 = 1$, $V_0 = 0$



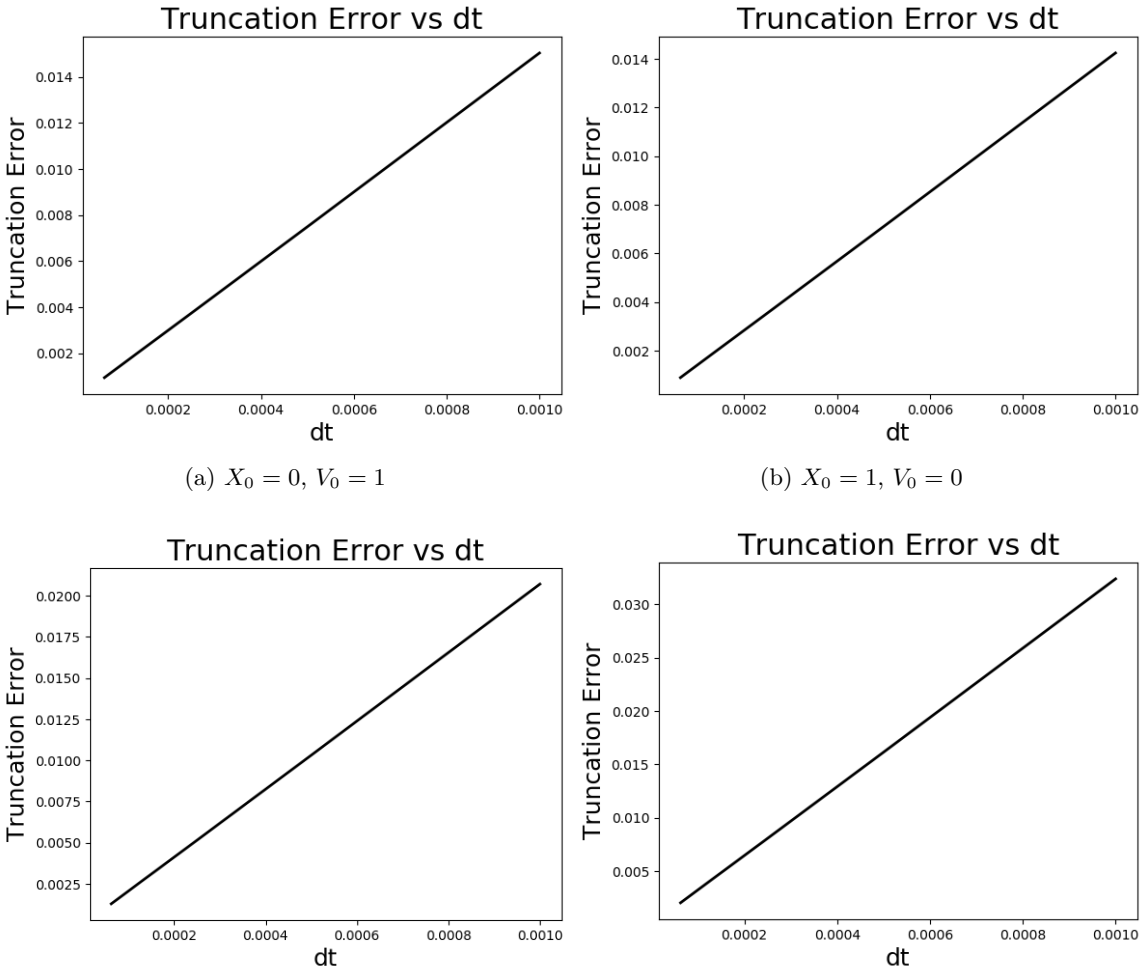
(c) $X_0 = 1$, $V_0 = 1$



(d) $X_0 = 2$, $V_0 = 1$

Figur 4: Implicit Euler global errors. Notice how they are just the negative of the Explicit ones. This makes sense since we are essentially going "backwards"with this method compared to the explicit method (i.e. using using new values to compute old ones rather than vice versa).

3 Truncation Error



(a) $X_0 = 0$, $V_0 = 1$

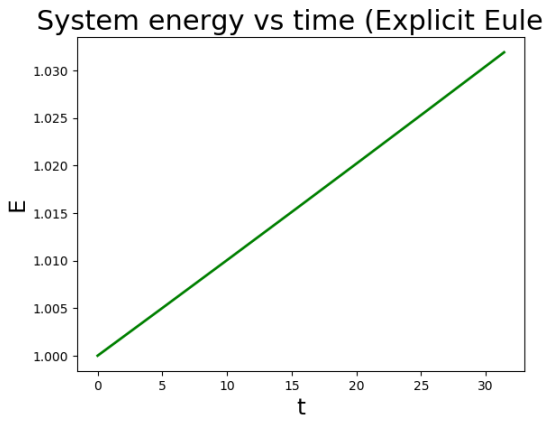
(b) $X_0 = 1$, $V_0 = 0$

(c) $X_0 = 1$, $V_0 = 1$

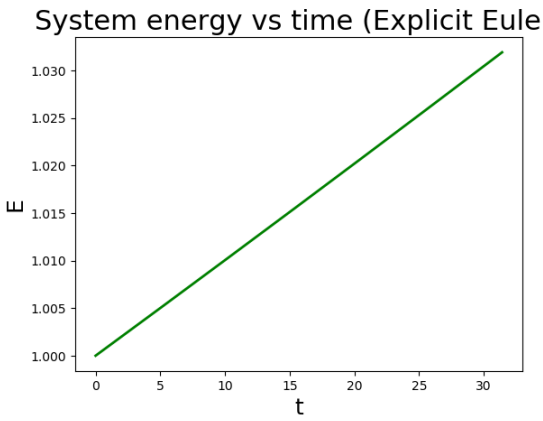
(d) $X_0 = 2$, $V_0 = 1$

Figur 5: Truncation error versus step size. For relatively small step sizes, the truncation error is proportional to the step size regardless of initial conditions. It is also independent of the Euler method we choose. This linear relationship between truncation error and step size is also positive as expected, implying that larger steps lead to more error.

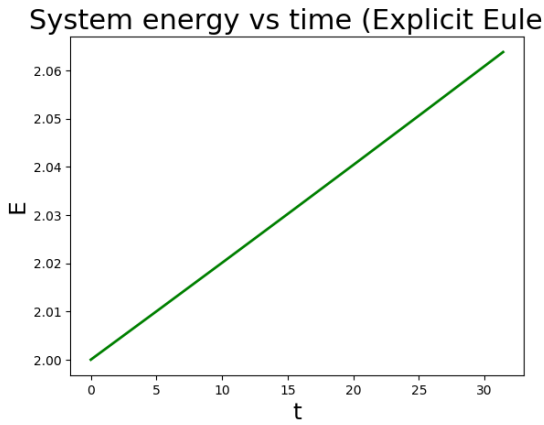
4 Total Energy



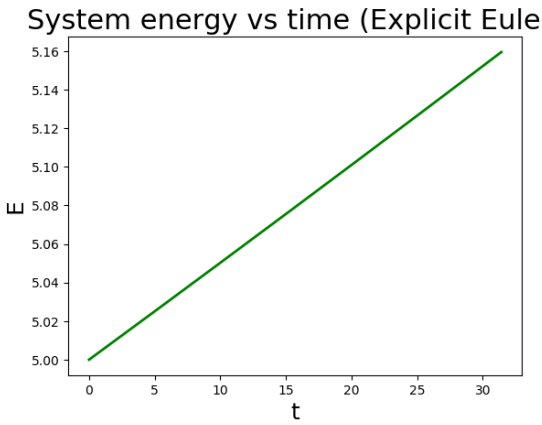
(a) $X_0 = 0$, $V_0 = 1$



(b) $X_0 = 1$, $V_0 = 0$

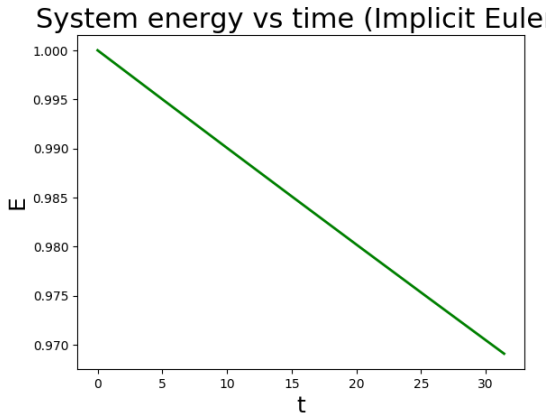


(c) $X_0 = 1$, $V_0 = 1$

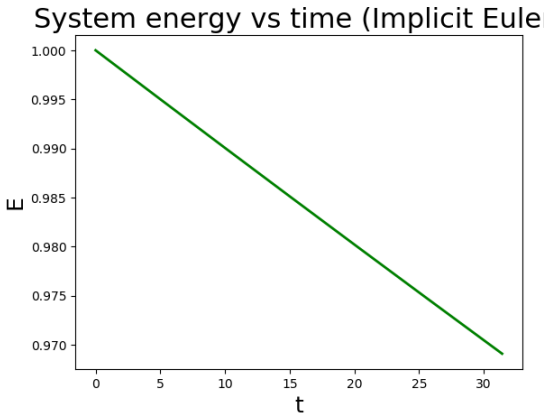


(d) $X_0 = 2$, $V_0 = 1$

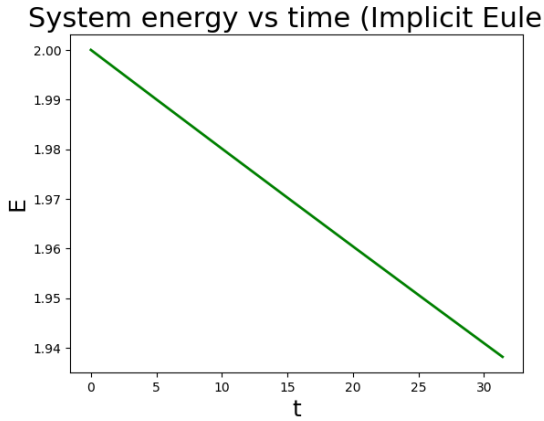
Figur 6: Total system energy given by the numerical explicit Euler method.



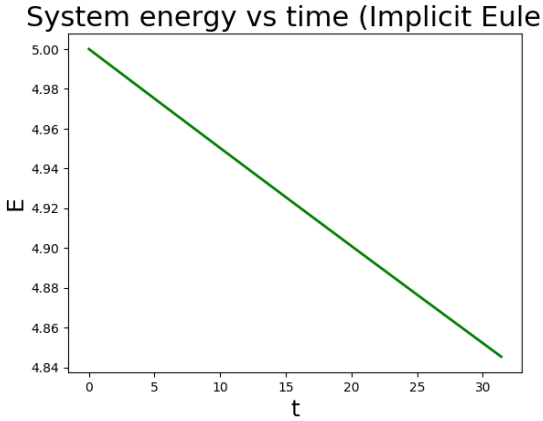
(a) $X_0 = 0$, $V_0 = 1$



(b) $X_0 = 1$, $V_0 = 0$



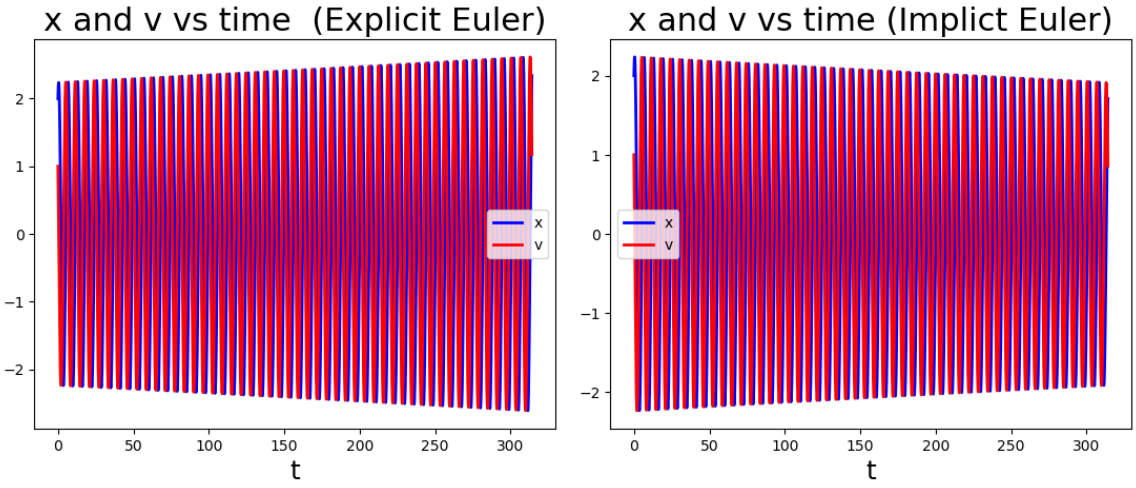
(c) $X_0 = 1$, $V_0 = 1$



(d) $X_0 = 2$, $V_0 = 1$

Figur 7: Total system energy given by the numerical implicit Euler method.

Notice how in both cases, the system energy increases over time when it should remain constant. This is because we are adding the squares of position and velocity, which is comparable to adding their amplitudes within a scale factor. However, as the global error plots show, the amplitude of our numerical solutions is increasing/decreasing (Explicit/Implicit) relative to that of the analytic answers, whose amplitudes remain constant. These changing amplitude of our estimates is why our energy grows or shrinks over time.

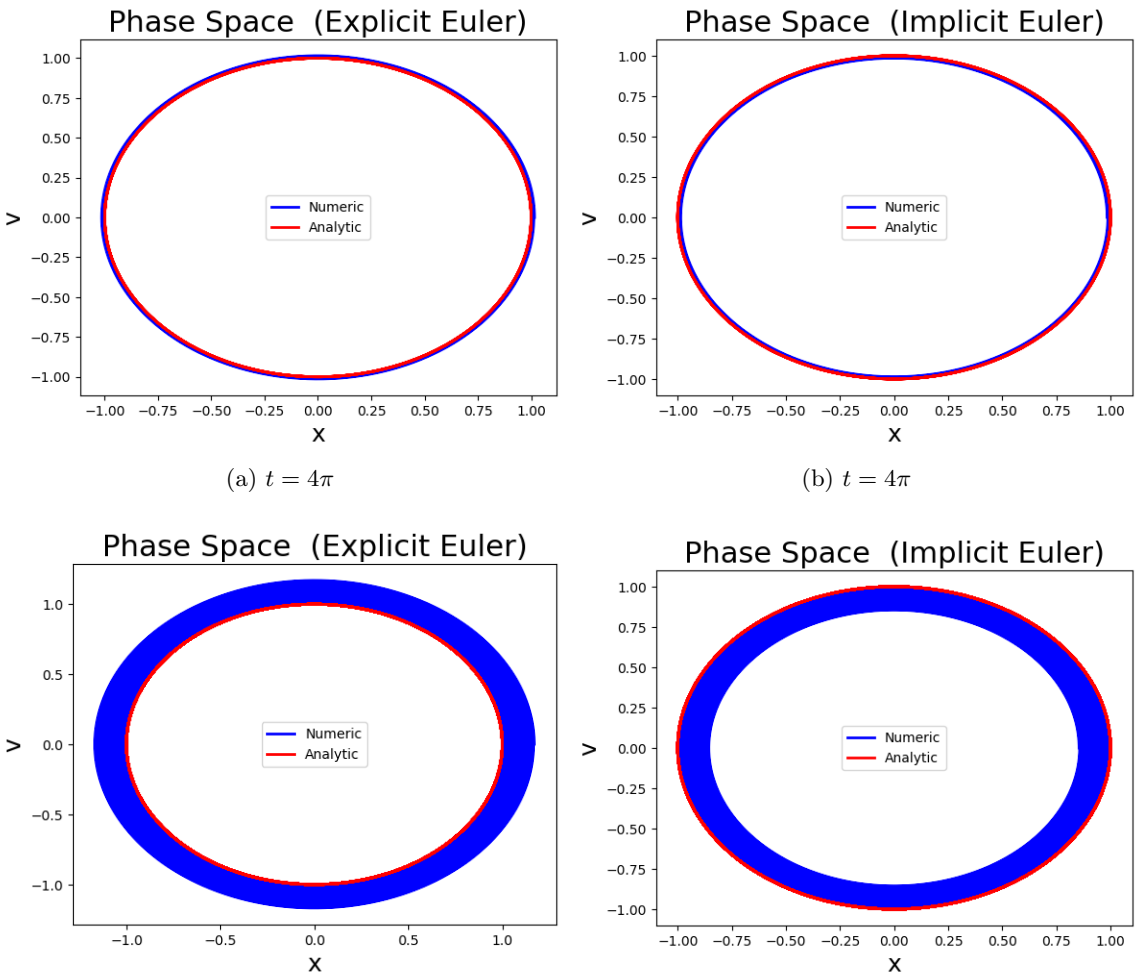


(a) $X_0 = 2$, $V_0 = 1$

(b) $X_0 = 2$, $V_0 = 1$

Figur 8: After many cycles, the explicit solution grows while the implicit one shrinks.

5 Phase Space and Symplectic Integration



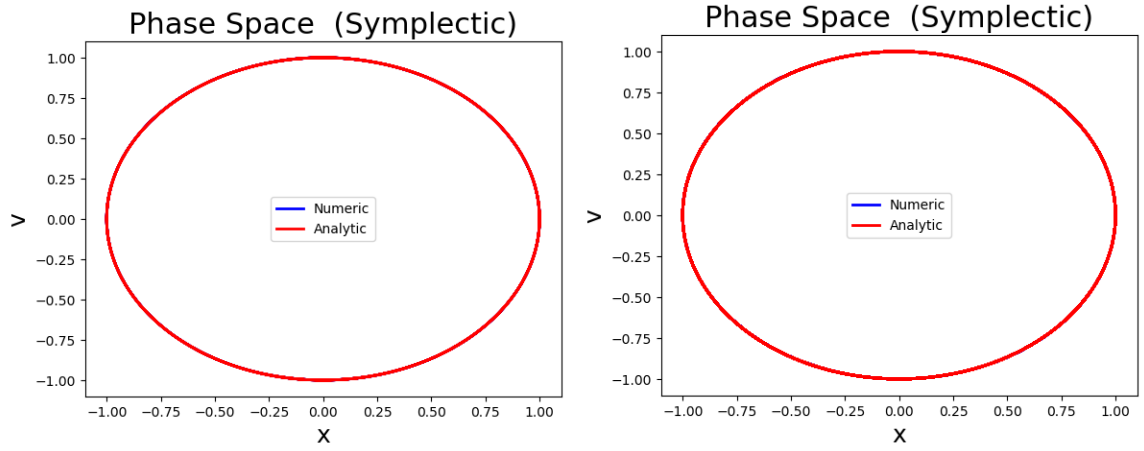
(a) $t = 4\pi$

(b) $t = 4\pi$

(c) $t = 100\pi$

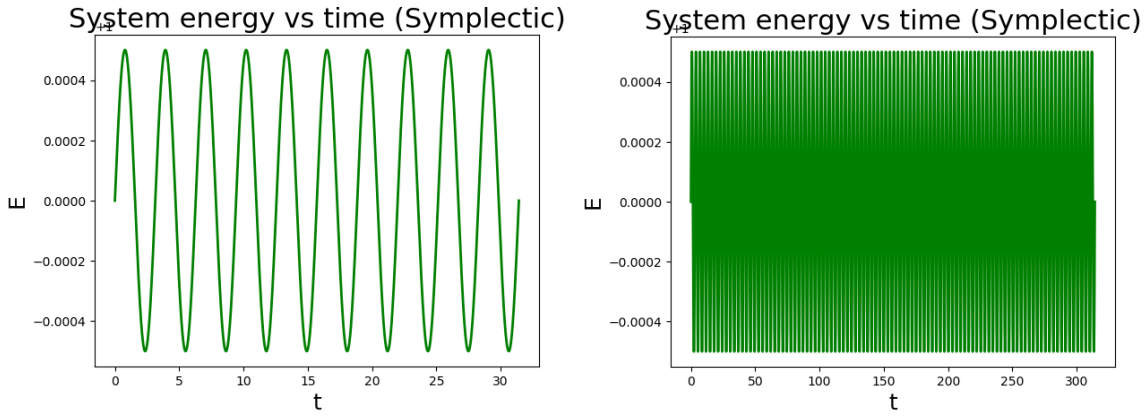
(d) $t = 100\pi$

Figur 9: Phase space plots for both the Explicit and Implicit Euler methods for initial conditions $x_0 = 1$, $v_0 = 0$. As time goes on, error in the system energy calculation does not preserve area in phase space. The area for the Explicit curve increase while the area of the Implicit curve decreases, corresponding to increasing and decreasing system energy respectively.



(a) $t = 4\pi$ (b) $t = 100\pi$

Figur 10: Phase space plots for both the Symplectic methods for initial conditions $x_0 = 1$, $v_0 = 0$. Regardless of the length of time, this scheme preserves area in phase space.



(a) $t = 4\pi$ (b) $t = 100\pi$

Figur 11: Plots of offset total system energy over time for the symplectic method. This numeric solution oscillates over time, but on average the energy is constant, reflecting how the radius of the circle in phase space is constant around one cycle.

6 pythoncode

```
# coding: utf-8

# In[23]:


#!/usr/bin/python
import sys
import math
import numpy as np
```

```
import matplotlib.pyplot as plt
import sympy as sp
import mpmath
import argparse

#Note Use argparse next time

def Explicit_Euler(Xi,Vi,t0,tf,dt):
    N = int(round((tf-t0)/dt))
    t = np.linspace(t0,tf,N+1)
    x = np.zeros(len(t))
    v = np.zeros(len(t))
    v[0] = Vi
    x[0] = Xi

    for k in range(0,N):
        x[k+1] = x[k] + dt*v[k]
        v[k+1] = v[k] - dt*x[k]
    # Also Defines analytic functions given initial conditions
    xa = Xi*np.cos(t)+Vi*np.sin(t)
    va = Vi*np.cos(t)-Xi*np.sin(t)
    return Xi,Vi,x,v,t,dt,xa,va

# This function evaluates t,x,and v in a dt range around the set value
def Evaluate(value):
    index = np.where(np.logical_and(t>=value-dt, t<=value+dt))
    print("t("+repr(value)+"+-dt)=%s" % t[index])
    print("x("+repr(value)+"+-dt)=%s" % x[index])
    print("v("+repr(value)+"+-dt)=%s" % v[index])
    return t[index],x[index],v[index]

def Implicit_Euler(Xi,Vi,t0,tf,dt):
    N = int(round((tf-t0)/dt))
    t = np.linspace(t0,tf,N+1)
    x = np.zeros(len(t))
    v = np.zeros(len(t))
    v[0] = Vi
    x[0] = Xi

    # This uses numpy's linear algebra solver to solve equation (9)
    for i in range(0,N):
        a = np.array([[1,-dt],[dt,1]])
        b = np.array([x[i],v[i]])
        [x[i+1],v[i+1]] = np.linalg.solve(a,b)

    # Also Defines analytic functions given initial conditions
    xa = Xi*np.cos(t)+Vi*np.sin(t)
    va = Vi*np.cos(t)-Xi*np.sin(t)
    return Xi,Vi,x,v,t,dt,xa,va
```

```
def Symplectic (Xi,Vi,t0,tf,dt):
    N = int(round((tf-t0)/dt))
    t = np.linspace(t0,tf,N+1)
    x = np.zeros(len(t))
    v = np.zeros(len(t))
    v[0] = Vi
    x[0] = Xi

    for i in range(0,N):
        x[i+1] = x[i]+dt*v[i]
        v[i+1] = v[i]-dt*x[i+1]

    # Also Defines analytic functions given initial conditions
    xa = Xi*np.cos(t)+Vi*np.sin(t)
    va = Vi*np.cos(t)-Xi*np.sin(t)
    return Xi,Vi,x,v,t,dt,xa,va

# Initialize Variables
if len(sys.argv) == 6 :
Xi = float(sys.argv[1])
Vi = float(sys.argv[2])
ti = float(sys.argv[3])
tf = float(sys.argv[4])*np.pi
dt = float(sys.argv[5])
elif sys.argv[1] == 'long':
Xi = float(sys.argv[2])
Vi = float(sys.argv[3])
ti = 0
tf = 100*np.pi
dt = 0.001
elif len(sys.argv) == 3:
Xi = float(sys.argv[1])
Vi = float(sys.argv[2])
ti = 0
tf = 10*np.pi
dt = 0.001
else:
Xi = 1
Vi = 0
ti = 0
tf = 10*np.pi
dt = 0.001


length = len(sys.argv)

Xi,Vi,x,v,t,dt,xa,va = Explicit_Euler(Xi,Vi,ti,tf,dt)
te,xe,ve = Evaluate(3*np.pi)
```

```
# Question 1
if length == 1 or length == 3 or length == 6 or sys.argv[1] ==’long’ or sys.argv[1] == ’x_v_E
plt.figure()
plt.plot(t, x, linewidth=2.0, color=’blue’, label=’x’)
plt.plot(t, v, linewidth=2.0, color=’red’, label=’v’)
plt.xlabel(’t’, fontsize=18)
plt.ylabel(’’, fontsize=18)
plt.title(’x and v vs time  (Explicit Euler)’, fontsize=22)
plt.legend([’x’,’v’], loc=’best’)
if sys.argv[1] == ’long’:
filename =’long_Explicit_X0_’+repr(Xi)+’_V0_’+repr(Vi)
else:
filename = ’Explicit_X0_’+repr(Xi)+’_V0_’+repr(Vi)
filename = filename.replace(’.’,’_’)+’.png’
plt.savefig(filename,bbox_inches=’tight’)
#plt.show()

# Question 2
if length == 1 or length == 3 or length == 6 or sys.argv[1] ==’long’ or sys.argv[1] == ’x_v_e
plt.figure()
plt.plot(t, (xa-x), linewidth=2.0, color=’blue’, label=’x’)
plt.plot(t, (va-v), linewidth=2.0, color=’red’, label=’v’)
plt.xlabel(’t’, fontsize=18)
plt.ylabel(’’, fontsize=18)
plt.title(’x and v errors vs time (Explicit Euler)’, fontsize=22)
plt.legend([’x’,’v’], loc=’best’)
if sys.argv[1] == ’long’:
filename = ’long_Explicit_Errors_X0_’+repr(Xi)+’_V0_’+repr(Vi)
else:
filename = ’Explicit_Errors_X0_’+repr(Xi)+’_V0_’+repr(Vi)
filename = filename.replace(’.’,’_’)+’.png’
plt.savefig(filename,bbox_inches=’tight’)
#plt.show()

# Question 3
n = 4
h = dt*np.power(1/2,range(0,n+1))
trunc = np.zeros(len(h))

for i in range(len(h)):
    _,_,x1,_,_,_,xa1,_ = Explicit_Euler(Xi,Vi,ti,tf,h[i])
    trunc[i] = np.amax(xa1-x1)

if length == 1 or length == 3 or length == 6 or sys.argv[1] ==’long’ or sys.argv[1] == ’Trunc
plt.figure()
plt.plot(h, trunc, linewidth=2.0, color=’black’, label=’trunc’)
plt.xlabel(’dt’, fontsize=18)
```

```
plt.ylabel('Truncation Error', fontsize=18)
plt.title('Truncation Error vs dt', fontsize=22)
if sys.argv[1] == 'long':
filename = 'long_Truncation_X0_'+repr(Xi)+'_V0_'+repr(Vi)
else:
filename = 'Truncation_X0_'+repr(Xi)+'_V0_'+repr(Vi)
filename = filename.replace('.','_')+'.png'
plt.savefig(filename,bbox_inches='tight')
#plt.show()

# Question 4
E = x**(2)+v**(2)

if length == 1 or length == 3 or length == 6 or sys.argv[1] =='long' or sys.argv[1] == 'energ
plt.figure()
plt.plot(t, E, linewidth=2.0, color='green', label='E')
plt.xlabel('t', fontsize=18)
plt.ylabel('E', fontsize=18)
plt.title('System energy vs time (Explicit Euler)', fontsize=22)
if sys.argv[1] == 'long':
filename = 'long_Explicit_Energy_X0_'+repr(Xi)+'_V0_'+repr(Vi)
else:
filename = 'Explicit_Energy_X0_'+repr(Xi)+'_V0_'+repr(Vi)
filename = filename.replace('.','_')+'.png'
plt.savefig(filename,bbox_inches='tight')
#plt.show()

# Question 5
_,_,xI,vI,tI,_,xaI,vaI = Implicit_Euler(Xi,Vi,ti,tf,dt)

if length == 1 or length == 3 or length == 6 or sys.argv[1] =='long' or sys.argv[1] == 'x_v_I
plt.figure()
plt.plot(t, xI, linewidth=2.0, color='blue', label='x')
plt.plot(t, vI, linewidth=2.0, color='red', label='v')
plt.xlabel('t', fontsize=18)
plt.ylabel('', fontsize=18)
plt.title('x and v vs time (Implict Euler)', fontsize=22)
plt.legend(['x','v'], loc='best')
if sys.argv[1] == 'long':
filename = 'long_Implicit_X0_'+repr(Xi)+'_V0_'+repr(Vi)
else:
filename = 'Implicit_X0_'+repr(Xi)+'_V0_'+repr(Vi)
filename = filename.replace('.','_')+'.png'
plt.savefig(filename,bbox_inches='tight')
#plt.show()

if length == 1 or length == 3 or length == 6 or sys.argv[1] =='long' or sys.argv[1] == 'x_v_e
plt.figure()
plt.plot(t, (xaI-xI), linewidth=2.0, color='blue', label='x')
```

```
plt.plot(t, (vaI-vI), linewidth=2.0, color='red', label='v')
plt.xlabel('t', fontsize=18)
plt.ylabel('', fontsize=18)
plt.title('x and v errors vs time (Implict Euler)', fontsize=22)
plt.legend(['x','v'], loc='best')
if sys.argv[1] == 'long':
filename = 'long_Implicit_Errors_X0_'+repr(Xi)+'_V0_'+repr(Vi)
else:
filename = 'Implicit_Errors_X0_'+repr(Xi)+'_V0_'+repr(Vi)
filename = filename.replace('.','_')+'.png'
plt.savefig(filename,bbox_inches='tight')
#plt.show()

EI = xI**(2)+vI**(2)

if length == 1 or length == 3 or length == 6 or sys.argv[1] =='long' or sys.argv[1] == 'energ
plt.figure()
plt.plot(t, EI, linewidth=2.0, color='green', label='E')
plt.xlabel('t', fontsize=18)
plt.ylabel('E', fontsize=18)
plt.title('System energy vs time (Implicit Euler)', fontsize=22)
if sys.argv[1] == 'long':
filename = 'long_Implicit_Energy_X0_'+repr(Xi)+'_V0_'+repr(Vi)
else:
filename = 'Implicit_Energy_X0_'+repr(Xi)+'_V0_'+repr(Vi)
filename = filename.replace('.','_')+'.png'
plt.savefig(filename,bbox_inches='tight')
#plt.show()

# Part 2
######################
if length == 1 or length == 3 or length == 6 or sys.argv[1] =='long' or sys.argv[1] == 'phase
plt.figure()
plt.plot(x, v, linewidth=2.0, color='blue', label='numeric')
plt.plot(xa, va, linewidth=2.0, color='red', label='analytic')
plt.xlabel('x', fontsize=18)
plt.ylabel('v', fontsize=18)
plt.title('Phase Space  (Explicit Euler)', fontsize=22)
plt.legend(['Numeric','Analytic'], loc='best')
if sys.argv[1] == 'long':
filename = 'long_Explicit_Phase_X0_'+repr(Xi)+'_V0_'+repr(Vi)
else:
filename = 'Explicit_Phase_X0_'+repr(Xi)+'_V0_'+repr(Vi)
filename = filename.replace('.','_')+'.png'
plt.savefig(filename,bbox_inches='tight')
#plt.show()

if length == 1 or length == 3 or length == 6 or sys.argv[1] =='long' or sys.argv[1] == 'phase
plt.figure()
```

```
plt.plot(xI, vI, linewidth=2.0, color='blue', label='numeric')
plt.plot(xaI, vaI, linewidth=2.0, color='red', label='analytic')
plt.xlabel('x', fontsize=18)
plt.ylabel('v', fontsize=18)
plt.title('Phase Space  (Implicit Euler)', fontsize=22)
plt.legend(['Numeric','Analytic'], loc='best')
if sys.argv[1] == 'long':
filename = 'long_Implicit_Phase_X0_'+repr(Xi)+'_V0_'+repr(Vi)
else:
filename = 'Implicit_Phase_X0_'+repr(Xi)+'_V0_'+repr(Vi)
filename = filename.replace('.','_')+'.png'
plt.savefig(filename,bbox_inches='tight')
#plt.show()

# b
_,_,xs,vs,ts,_,xas,vas = Symplectic(Xi,Vi,ti,tf,dt)

if length == 1 or length == 3 or length == 6 or sys.argv[1] =='long' or sys.argv[1] == 'phase
plt.figure()
plt.plot(xs, vs, linewidth=2.0, color='blue', label='numeric')
plt.plot(xa, va, linewidth=2.0, color='red', label='analytic')
plt.xlabel('x', fontsize=18)
plt.ylabel('v', fontsize=18)
plt.title('Phase Space  (Symplectic)', fontsize=22)
plt.legend(['Numeric','Analytic'], loc='best')
if sys.argv[1] == 'long':
filename = 'long_Symplectic_Phase_X0_'+repr(Xi)+'_V0_'+repr(Vi)
else:
filename = 'Symplectic_Phase_X0_'+repr(Xi)+'_V0_'+repr(Vi)
filename = filename.replace('.','_')+'.png'
plt.savefig(filename,bbox_inches='tight')
#plt.show()

Es = xs**(2)+vs**(2)

if length == 1 or length == 3 or length == 6 or sys.argv[1] =='long' or sys.argv[1] == 'energ
plt.figure()
plt.plot(ts, Es, linewidth=2.0, color='green', label='E')
plt.xlabel('t', fontsize=18)
plt.ylabel('E', fontsize=18)
plt.title('System energy vs time (Symplectic)', fontsize=22)
if sys.argv[1] == 'long':
filename = 'long_Symplectic_Energy_X0_'+repr(Xi)+'_V0_'+repr(Vi)
else:
filename = 'Symplectic_Energy_X0_'+repr(Xi)+'_V0_'+repr(Vi)
filename = filename.replace('.','_')+'.png'
plt.savefig(filename,bbox_inches='tight')
#plt.show()
```

```
k = 10
me = round(-len(t)/k)

if length == 1 or length == 3 or length == 6 or sys.argv[1] ==’long’ or sys.argv[1] == ’x_v_S
plt.figure()
plt.plot(ts[me:], (xas)[me:], linewidth=2.0, color=’blue’, label=’x’)
plt.plot(ts[me:], (xs)[me:], linewidth=2.0, color=’red’, label=’v’)
plt.xlabel(’t’, fontsize=18)
plt.ylabel(’’, fontsize=18)
plt.title(’x and v vs time (Symplectic)’, fontsize=22)
plt.legend([’Exact’,’Symplectic’], loc=’best’)
if sys.argv[1] == ’long’:
filename = ’long_Symplectic_X0_’+repr(Xi)+’_V0_’+repr(Vi)
else:
filename = ’Symplectic_X0_’+repr(Xi)+’_V0_’+repr(Vi)
filename = filename.replace(’.’,’_’)+’.png’
plt.savefig(filename,bbox_inches=’tight’)
#plt.show()
```

7 Git Log

commit a996d372daf98cff6a79466845c25957878e324f Author: Noah Huffman <nhuffman@caltech.edu> Date: Thu May 10 02:10:13 2018 -0700

Added a comment

commit 3e0b9f04e6986fc3f1c7d3cdca0dc7010b67dafe Author: Noah Huffman <nhuffman@caltech.edu> Date: Thu May 10 02:00:30 2018 -0700

PH 20 Assignment 3