Ph 20: Assignment #4

Maya Fuller

Due: May 14, 2018

MakeFile

```
#Generate completed assignment PDF
.PHONY : pdf
pdf : set4.tex *.png vc.log
pdflatex $<

#Generate version control log
vc.log :
git log > $@

# Generate plots
%.png : ode.py
python $< 0 0.8 100000 0.001 # Change python code command line arguments


.PHONY : clean
clean :
rm -f *.png
rm set4.pdf
rm set4.aux
rm set4.log
rm set4.out
rm vc.log
```

Version Control

```
commit 5ebedee8ad0f727dd03b435d1dbe22ed0385417d
Author: Maya F <38202996+m-fuller@users.noreply.github.com>
Date:   Mon May 14 03:19:04 2018 -0700

    First use of MakeFile
```

```
commit cc314677dc2ae025772552f680f7f84f961347c8
Author: Maya F <38202996+m-fuller@users.noreply.github.com>
Date:   Mon May 14 02:59:21 2018 -0700

    Add assignment 4

commit 06dab4a6d599f8fb41b5691f42704023793b09c6
Author: Maya F <38202996+m-fuller@users.noreply.github.com>
Date:   Sat May 12 17:56:46 2018 -0700

    Add originally submitted files for assignment 3

commit 90fc462d144fb480d554dea7c583ae720cc29b25
Author: Maya F <38202996+m-fuller@users.noreply.github.com>
Date:   Sat May 12 17:23:38 2018 -0700

    Discuss concerns about Mars' climate for Mummy

commit 65175befdbfe7a0367655d58f1bba6fd3e638b7d
Author: Maya F <38202996+m-fuller@users.noreply.github.com>
Date:   Sat May 12 17:07:55 2018 -0700

    Add concerns about effects of Mars' moons on Wolfman

commit 12b684ce931b951875433a68e5a630410f9a6ed0
Author: Maya F <38202996+m-fuller@users.noreply.github.com>
Date:   Sat May 12 16:57:00 2018 -0700

    Start notes on Mars as a base
```

Source Code

```
import sys
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.gridspec import GridSpec

# Parts 1 and 2 Variables

x0=float(sys.argv[1]) #initial position
v0=float(sys.argv[2]) #initial velocity
t0=0
N=int(sys.argv[3]) # number of time steps
h0=float(sys.argv[4]) # length of time steps in seconds
```

```
phix=np.arcsin(x0)-t0
phiv=np.arccos(v0)-t0

numHVal=5
xErrorMax=np.zeros(numHVal)
xErrorMaxIm=np.zeros(numHVal)

# Assignment Part 1

def xExVal(xi,vi,h):
    return xi+h*vi
def vExVal(xi,vi,h):
    return vi-h*xi
def xImVal(xi,vi,h):
    return (xi+h*vi)/(1+h**2)
def vImVal(xi,vi,h):
    return (vi-h*xi)/(1+h**2)

def iterate(h):
    t=np.zeros(N)
    t[0]=t0
    xEx=np.zeros(N)
    xEx[0]=x0
    vEx=np.zeros(N)
    vEx[0]=v0
    xIm=np.zeros(N)
    xIm[0]=x0
    vIm=np.zeros(N)
    vIm[0]=v0
    for i in range(1,N):
        xEx[i]=xExVal(xEx[i-1],vEx[i-1],h)
        vEx[i]=vExVal(xEx[i-1],vEx[i-1],h)
        t[i]=t0+i*h
        xIm[i]=xImVal(xIm[i-1],vIm[i-1],h)
        vIm[i]=vImVal(xIm[i-1],vIm[i-1],h)
    return (t, xEx, vEx, xIm, vIm)

def vstimePlots(t,xEx,vEx):
    phix=np.arcsin(x0)-t0
    phiv=np.arccos(v0)-t0
    plt.figure(figsize=(14,10))
    plt.title('Explicit Euler Method')
    plt.subplot(221)
    plt.plot(t,xEx,'r-')
    plt.xlabel('time (t)')
```

```
    plt.ylabel('x(t)')
    plt.subplot(222)
    plt.plot(t,vEx,'r-')
    plt.xlabel('time (t)')
    plt.ylabel('v(t)')
    plt.subplot(223)
    plt.plot(t,np.sin(t+phix)-xEx,'m-')
    plt.xlabel('time (t)')
    plt.ylabel(r'$x_{analytic}(t)-x(t)$')
    plt.subplot(224)
    plt.plot(t,np.cos(t+phiv)-vEx,'m-')
    plt.xlabel('time (t)')
    plt.ylabel(r'$v_{analytic}(t)-v(t)$')
    plt.savefig('vstimePlots.png')

def energyPlot(t,xEx,vEx):
    plt.figure(figsize=(14,10))
    plt.title('Explicit Euler Method')
    plt.subplot(311)
    plt.plot(t,np.power(xEx,2)+np.power(vEx,2),'r-')
    plt.xlabel('time (t)')
    plt.ylabel('total energy (E)')
    plt.subplot(312)
    plt.plot(t,(np.power(np.sin(t+phix),2)+np.power(np.cos(t+phiv),2)),'b-')
    plt.xlabel('time (t)')
    plt.ylabel(r'$E_{analytic}(t)$')
    plt.subplot(313)
    plt.plot(t,(np.power(np.sin(t+phix),2)+np.power(np.cos(t+phiv),2)) - (np.power(xEx,2)+
    plt.xlabel('time (t)')
    plt.ylabel(r'$E_{analytic}(t)-E(t)$')
    plt.savefig('energyPlot.png')

def errorPlot():
    plt.figure(figsize=(14,10))
    plt.title('Explicit Euler Method')
    for i in range(numHVal):
        itFuncs=iterate(h0/(2**i))
        xErrorMax[i]=np.amax(np.sin(itFuncs[0]+phix)-itFuncs[1])
    plt.plot(h0/(2**(np.arange(numHVal))),xErrorMax,'mo-')
    plt.xlabel('h')
    plt.ylabel(r'$max(x_{analytic}(t)-x(t))$')
    plt.savefig('errorPlot.png')

def implicitPlot(t,xIm,vIm):
    phix=np.arcsin(x0)-t0
```

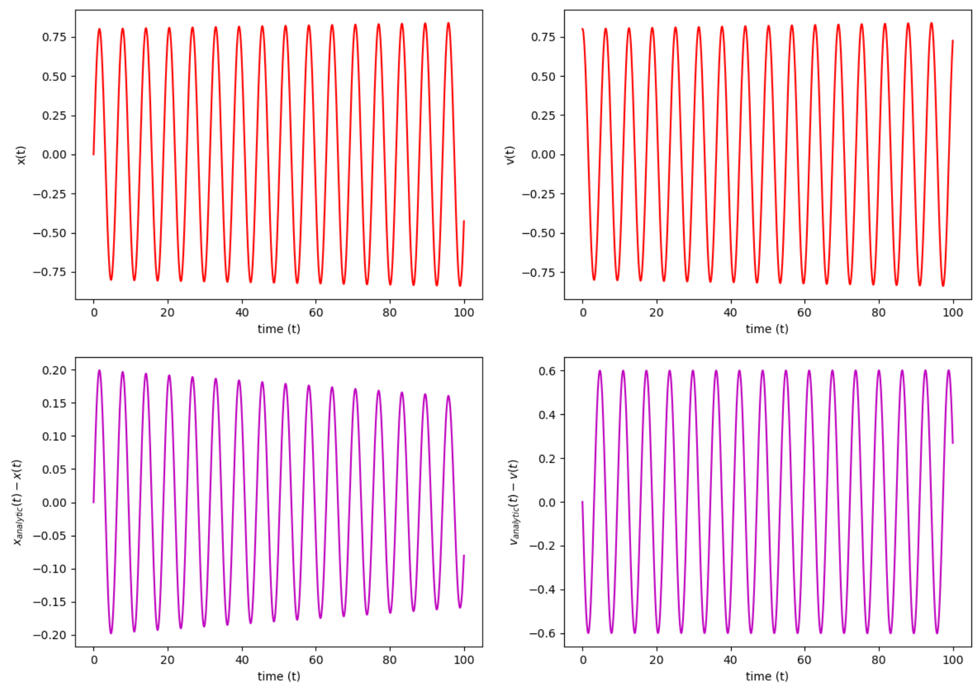
```
    phiv=np.arccos(v0)-t0
    plt.figure(figsize=(14,10))
    plt.title('Implicit Euler Method')
    plt.subplot(221)
    plt.plot(t,xIm,'r-')
    plt.xlabel('time (t)')
    plt.ylabel('x(t)')
    plt.subplot(222)
    plt.plot(t,vIm,'r-')
    plt.xlabel('time (t)')
    plt.ylabel('v(t)')
    plt.subplot(223)
    plt.plot(t,np.sin(t+phix)-xIm,'m-')
    plt.xlabel('time (t)')
    plt.ylabel(r'$x_{analytic}(t)-x(t)$')
    plt.subplot(224)
    plt.plot(t,np.cos(t+phiv)-vIm,'m-')
    plt.xlabel('time (t)')
    plt.ylabel(r'$y_{analytic}(t)-y(t)$')
    plt.savefig('implicitPlot.png')

def energyImplicitPlot(t,xIm,vIm):
    plt.figure(figsize=(14,10))
    plt.title('Implicit Euler Method')
    plt.subplot(311)
    plt.plot(t,np.power(xIm,2)+np.power(vIm,2),'r-')
    plt.xlabel('time (t)')
    plt.ylabel('total energy (E)')
    plt.subplot(312)
    plt.plot(t,(np.power(np.sin(t+phix),2)+np.power(np.cos(t+phiv),2)),'b-')
    plt.xlabel('time (t)')
    plt.ylabel(r'$E_{analytic}(t)$')
    plt.subplot(313)
    plt.plot(t,(np.power(np.sin(t+phix),2)+np.power(np.cos(t+phiv),2)) - (np.power(xIm,2)+n
    plt.xlabel('time (t)')
    plt.ylabel(r'$E_{analytic}(t)-E(t)$')
    plt.savefig('energyImplicitPlot.png')

def errorImplicitPlot():
    plt.figure(figsize=(14,10))
    plt.title('Implicit Euler Method')
    for i in range(numHVal):
        itFuncs=iterate(h0/(2**i))
        xErrorMaxIm[i]=np.amax(np.sin(itFuncs[0]+phix)-itFuncs[3])
    plt.plot(h0/(2**(np.arange(numHVal))),xErrorMaxIm,'mo-')
```

```
    plt.xlabel('h')
    plt.ylabel(r'$max(x_{analytic}(t)-x(t))$')
    plt.savefig('errorImplicitPlot.png')

# Assignment Part 2
def iterateSym(h):
    xSym=np.zeros(N)
    xSym[0]=x0
    vSym=np.zeros(N)
    vSym[0]=v0
    for i in range(1,N):
        xSym[i]=xExVal(xSym[i-1],vSym[i-1],h)
        vSym[i]=vExVal(xSym[i],vSym[i-1],h)
    return (xSym, vSym)

def phaseSpace(xEx,vEx,xIm,vIm,xSym,vSym):
    gs=GridSpec(14,14)
    plt.figure(figsize=(14,10))
    plt.subplot(gs[0:4,0:4])
    plt.plot(xEx,vEx,'r-',linewidth=0.2)
    plt.xlabel(r'$x_{explicit}(t)$')
    plt.ylabel(r'$v_{explicit}(t)$')
    plt.subplot(gs[0:4,5:9])
    plt.plot(xSym,vSym,'b-',linewidth=0.2)
    plt.xlabel(r'$x_{symplectic}(t)$')
    plt.ylabel(r'$v_{symplectic}(t)$')
    plt.subplot(gs[0:4,10:14])
    plt.plot(xIm,vIm,'r-',linewidth=0.2)
    plt.xlabel(r'$x_{implicit}(t)$')
    plt.ylabel(r'$v_{implicit}(t)$')
    plt.subplot(gs[6:14,0:6])
    plt.plot(xSym-xEx,vSym-vEx,'m-',linewidth=1)
    plt.xlabel(r'$x_{symplectic}(t)-x_{explicit}(t)$')
    plt.ylabel(r'$v_{symplectic}(t)-v_{explicit}(t)$')
    plt.subplot(gs[6:14,8:14])
    plt.plot(xSym-xIm,vSym-vIm,'m-',linewidth=1)
    plt.xlabel(r'$x_{symplectic}(t)-x_{implicit}(t)$')
    plt.ylabel(r'$v_{symplectic}(t)-v_{implicit}(t)$')
    plt.savefig('phaseSpace.png')

def energySymplecticPlot(t,xSym,vSym):
    plt.figure(figsize=(14,10))
    plt.title('Symplectic Euler Method')
    plt.plot(t,np.power(xSym,2)+np.power(vSym,2),'r-')
    plt.xlabel('time (t)')
```

```
    plt.ylabel('total energy (E)')
    plt.savefig('energySymplecticPlot.png')

def vSymPlot(t,vSym):
    phiv=np.arccos(v0)-t0
    plt.figure(figsize=(14,10))
    plt.plot(t,vSym,'r-',label='Symplectic')
    plt.plot(t,np.cos(t+phiv),'b-',label='Exact')
    plt.xlabel('time(t)')
    plt.ylabel('v(t)')
    plt.legend()
    plt.savefig('vSymPlot.png')

#Parts 1 and 2 Plotting

iterFunc=iterate(h0)
vstimePlots(iterFunc[0],iterFunc[1],iterFunc[2])
energyPlot(iterFunc[0],iterFunc[1],iterFunc[2])
errorPlot()
iterFunc=iterate(h0)
implicitPlot(iterFunc[0],iterFunc[3],iterFunc[4])
energyImplicitPlot(iterFunc[0],iterFunc[3],iterFunc[4])
errorImplicitPlot()
iterFunc=iterate(h0)
iterSymFunc=iterateSym(h0)
phaseSpace(iterFunc[1],iterFunc[2],iterFunc[3],iterFunc[4],iterSymFunc[0],iterSymFunc[1])
energySymplecticPlot(iterFunc[0],iterSymFunc[0],iterSymFunc[1])
vSymPlot(iterFunc[0],iterSymFunc[1])
```

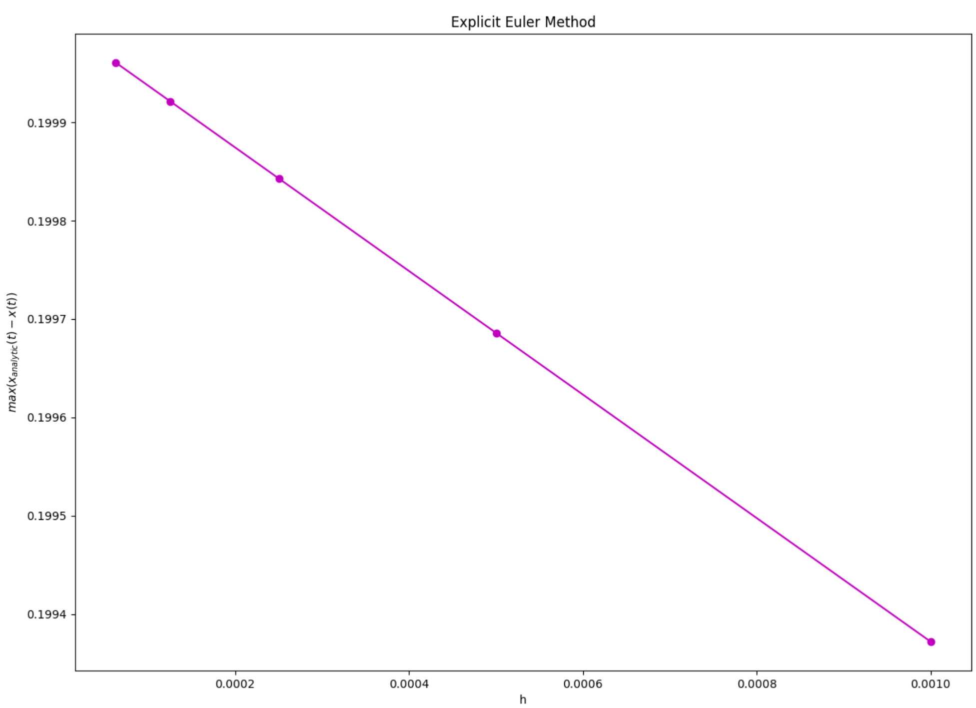
Part 1

Problem 1 and 2



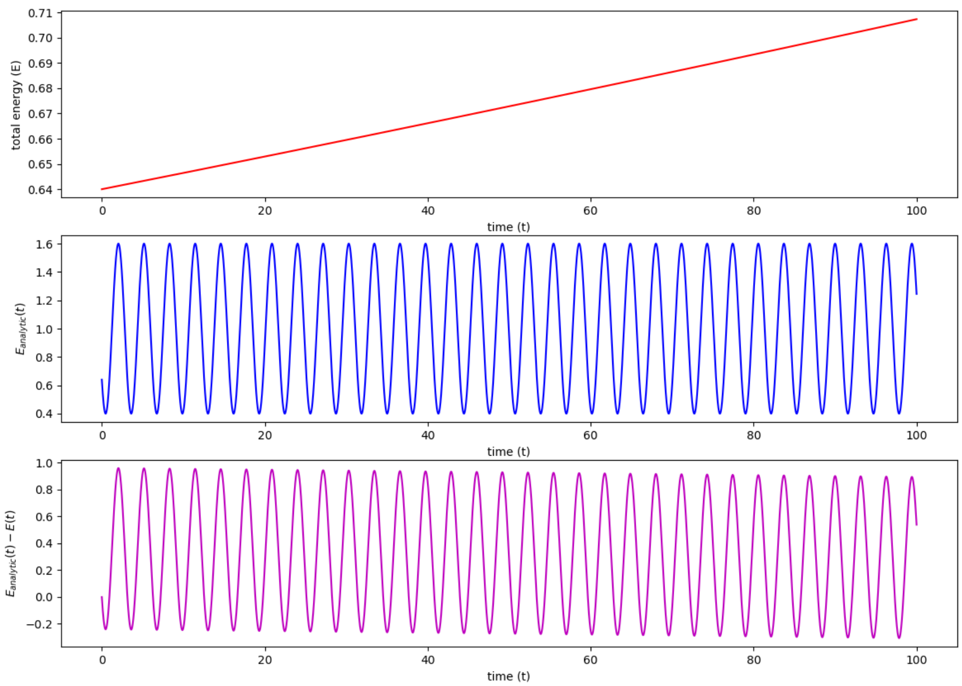
x(t)
v(t)
time (t)
$x_{analytic}(t) - x(t)$
$v_{analytic}(t) - v(t)$

Problem 3

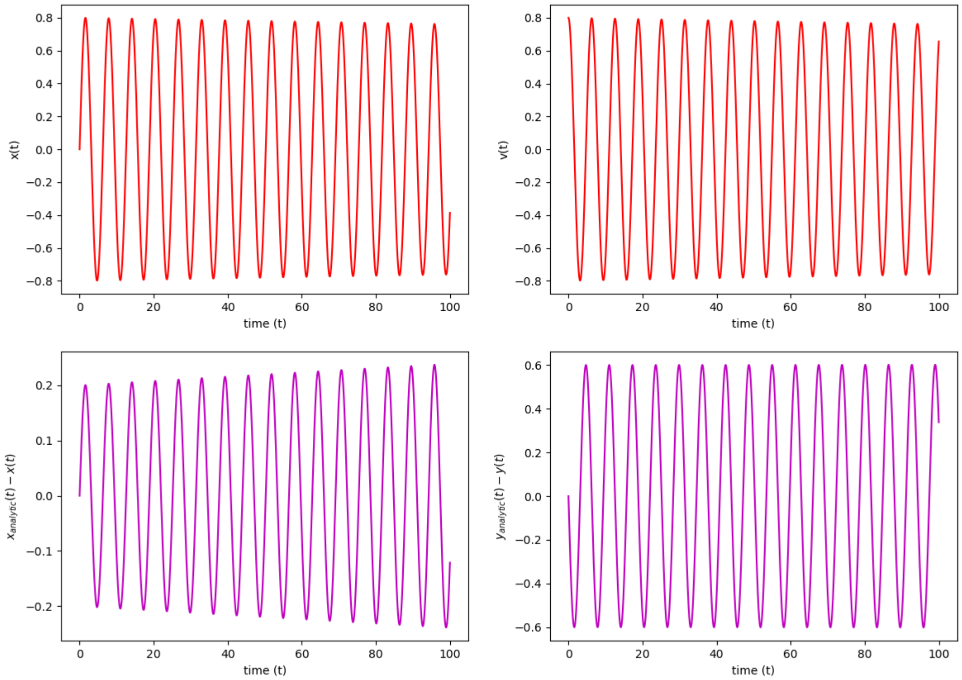


Explicit Euler Method
$max(x_{analytic}(t) - x(t))$
h

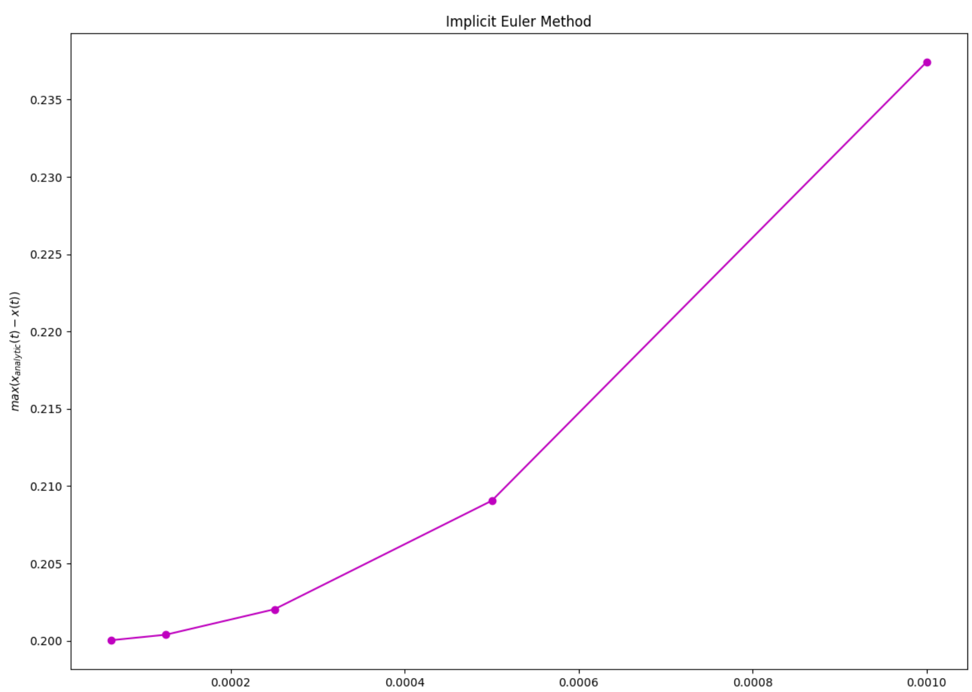
Problem 4



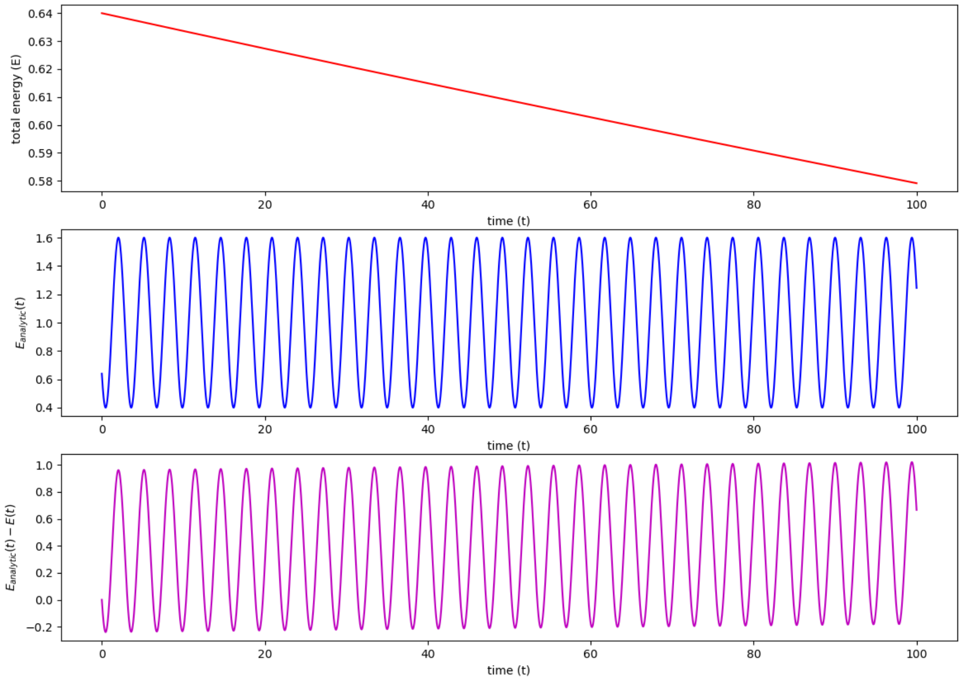
Problem 5



x(t)
v(t)
time (t)
$x_{analytic}(t) - x(t)$
$y_{analytic}(t) - y(t)$

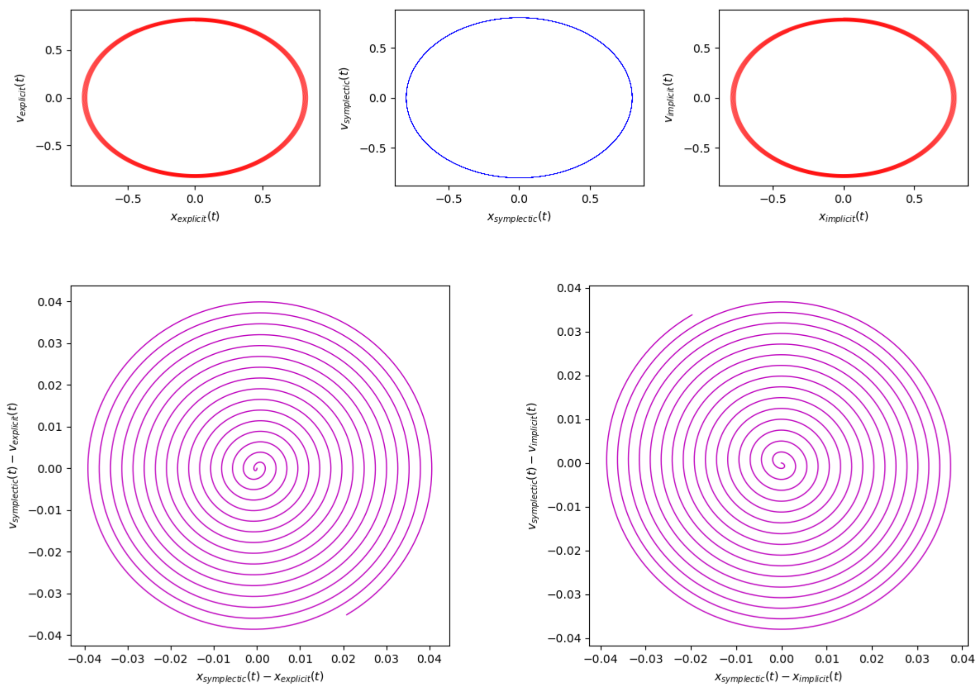


Implicit Euler Method
$max(x_{analytic}(t) - x(t))$
h

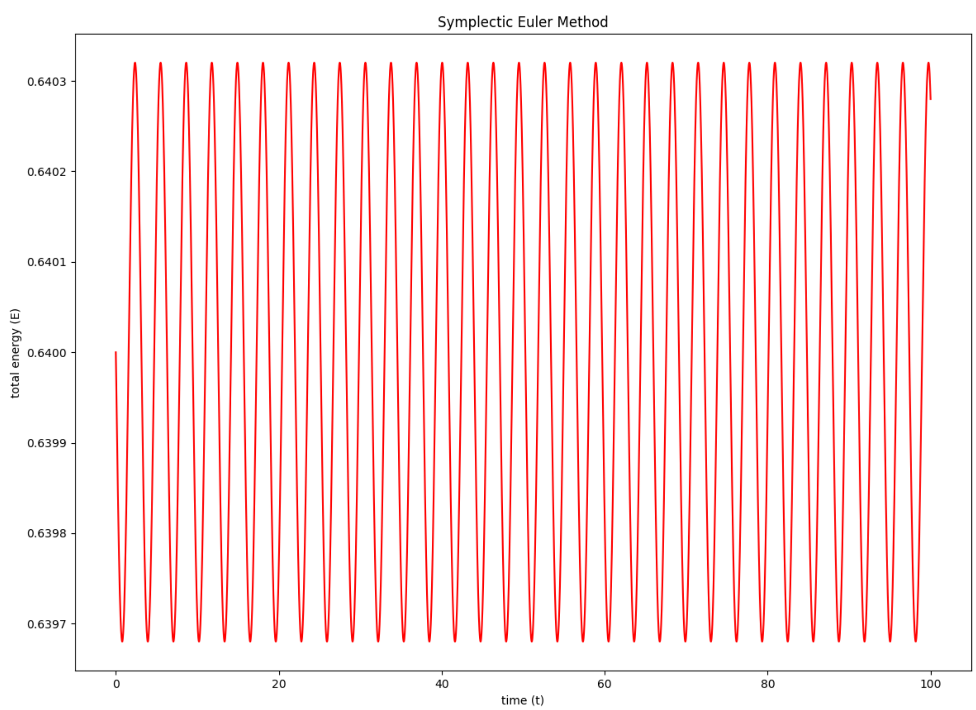


Part 2

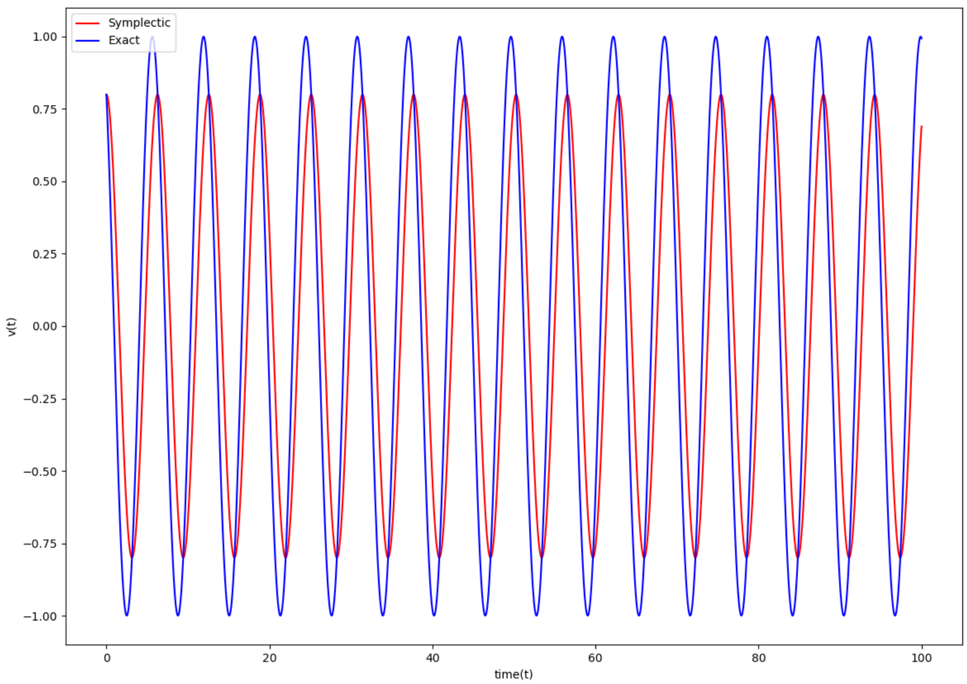
Problem 1 and 2


$v_{explicit}(t)$
$x_{explicit}(t)$
$v_{symplectic}(t)$
$x_{symplectic}(t)$
$v_{implicit}(t)$
$x_{implicit}(t)$
$v_{symplectic}(t) - v_{explicit}(t)$
$x_{symplectic}(t) - x_{explicit}(t)$
$v_{symplectic}(t) - v_{implicit}(t)$
$x_{symplectic}(t) - x_{implicit}(t)$

Problem 3


Symplectic Euler Method
total energy (E)
time (t)

Problem 4



Symplectic
Exact
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
v(t)
0
20
40
60
80
100
time(t)