Ph20 Homework 4

Simon Ricci

November 12, 2017

1 Version Control Log

```
commit 21c880c307fc42e6c6f3e389b09f401f088d9f1c
Author: sricci9 <33235604+sricci9@users.noreply.github.com>
Date:   Mon Oct 30 13:41:08 2017 -0700

    Add files via upload

commit 51f168459a26b23002de5da467d4a8ddeaa84404
Author: sricci <simonkricci@gmail.com>
Date:   Mon Oct 30 13:31:48 2017 -0700

    Added comment for assignment 4

commit 53ec162c2835359cba627d731532789c9fc6efea
Author: sricci <simonkricci@gmail.com>
Date:   Mon Oct 30 13:28:47 2017 -0700

    Original code
```

2 Makefile

```
# define a variable to check if any files not up to date
.PHONY : all

all : allPhaseSpace.png energy.png energyWithErr.png\
ErrorOsc_x0_1.0_v0_0.0_h_0.003.png EulerOsc_x0_1.0_v0_0.0_h_0.003.png\
expPhaseSpace.png impEnergy.png impEnergyWithErr.png\
impErrorOsc_x0_1.0_v0_0.0_h_0.003.png impEulerOsc_x0_1.0_v0_0.0_h_0.003.png\
impPhaseSpace%png manyErrors.png symEnergy.png symEnergyWithErr.png\
symPhaseSpace.png log.txt Homework4.pdf

allPhaseSpace%png energy%png energyWithErr%png\
ErrorOsc_x0_1.0_v0_0.0_h_0.003%png EulerOsc_x0_1.0_v0_0.0_h_0.003%png\
expPhaseSpace%png impEnergy%png impEnergyWithErr%png\
impErrorOsc_x0_1.0_v0_0.0_h_0.003%png impEulerOsc_x0_1.0_v0_0.0_h_0.003%png\
```

```
impPhaseSpace%png manyErrors%png symEnergy%png symEnergyWithErr%png\
symPhaseSpace%png :
python HarmOsc.py

# Make git log
log.txt :
git log > $@

# Make the LATEX file
Homework4.pdf : Homework4.tex
pdflatex $^
```

3 Source Code

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
plt.rc('figure', figsize=[5,5])

# Added changes for purpose of Assignment 4

def expEuler(x0, v0, h):
    '''Given initial conditions, runs the explicit Euler method and retu
    x values, v values, t values, and the h used.'''
    num = int(10 * np.pi / h)
    tVals = np.linspace(0, 10 * np.pi, num=num)
    xVals = np.zeros(num)
    vVals = np.zeros(num)
    xVals[0] = x0
    vVals[0] = v0
    for i in range(1,num):
        xVals[i] = xVals[i-1] + h * vVals[i-1]
        vVals[i] = vVals[i-1] - h * xVals[i-1]
    return h, xVals, vVals, tVals

def impEuler(x0, v0, h):
```

```
    '''Given initial conditions, runs the implicit Euler method and retu
    x values, v values, t values, and the h used.'''
    num = int(10 * np.pi / h)
    tVals = np.linspace(0, 10 * np.pi, num=num)
    xVals = np.zeros(num)
    vVals = np.zeros(num)
    xVals[0] = x0
    vVals[0] = v0
    for i in range(1,num):
        xVals[i] = (xVals[i-1] + h * vVals[i-1]) * 1/(1 + h**2)
        vVals[i] = (vVals[i-1] - h * xVals[i-1]) * 1/(1 + h**2)
    return h, xVals, vVals, tVals

def symEuler(x0, v0, h):
    '''Given initial conditions, runs the symplectic Euler method and
    returns x values, v values, t values, and the h used.'''
    num = int(10 * np.pi / h)
    tVals = np.linspace(0, 10 * np.pi, num=num)
    xVals = np.zeros(num)
    vVals = np.zeros(num)
    xVals[0] = x0
    vVals[0] = v0
    for i in range(1,num):
        xVals[i] = (xVals[i-1] + h * vVals[i-1])
        vVals[i] = (vVals[i-1] - h * xVals[i])
    return h, xVals, vVals, tVals

def eulerPlot(h, xVals, vVals, tVals, impExp):
    '''Plots the Euler plot for a given list of xVals,
    yVals, and tVals, and a given h value, and a string of
    either 'imp' or 'exp' that determines whether it's
    implicit or explicit Euler.'''
    plt.plot(tVals, xVals, color='red', label='x(t)')
    plt.plot(tVals, vVals, color='blue', label='v(t)')
    plt.legend()
    plt.xlabel('t')
    fileString = 'EulerOsc_x0_'+str(xVals[0])+'_v0_'+str(vVals[0])+\
    '_h_'+str(round(h,3))+'.png'
```

```
    assert impExp in ['imp', 'exp', 'sym']
    if impExp == 'imp': fileString = 'imp' + fileString
    plt.savefig(fileString)
    plt.clf()

def eulerError(impExp):
    '''Plots the error between the Euler solution ('imp' or 'exp') and
    the analytic solution to a simple harmonic oscillator with initial
    conditions x0 = 1 and v0 = 0.'''
    assert impExp in ['imp', 'exp']
    if impExp == 'exp': h, eeX, eeV, tVals = expEuler(1, 0, np.pi/1000)
    else: h, eeX, eeV, tVals = impEuler(1, 0, np.pi/1000)
    tVals2 = np.copy(tVals)
    analX = np.cos(tVals)
    analV = -np.sin(tVals)
    errX = analX - eeX
    errV = analV - eeV
    plt.plot(tVals, errX, color='red', label='x error')
    plt.plot(tVals, errV, color='blue', label='v error')
    plt.legend()
    plt.xlabel('t')
    fileString = 'ErrorOsc_x0_1.0_v0_0.0_h_'+str(round(h,3))+'.png'
    if impExp == 'imp': fileString = 'imp' + fileString
    plt.savefig(fileString)
    plt.clf()

def manyErrorPlots(h0):
    '''Shows the max value vs. h for the error plots
    of (1, 0.5, 0.25, 0.125, 0.0625) times h0'''
    hMults = np.array([1, 0.5, 0.25, 0.125, 0.0625])
    maxErr = np.zeros(5)
    for ind, mult in np.ndenumerate(hMults):
        Eu = expEuler(1, 0, mult * h0)
        eeX = Eu[1]
        tVals = Eu[3]
        analX = np.cos(tVals)
        errX = analX - eeX
        maxErr[ind] = np.amax(errX)
```

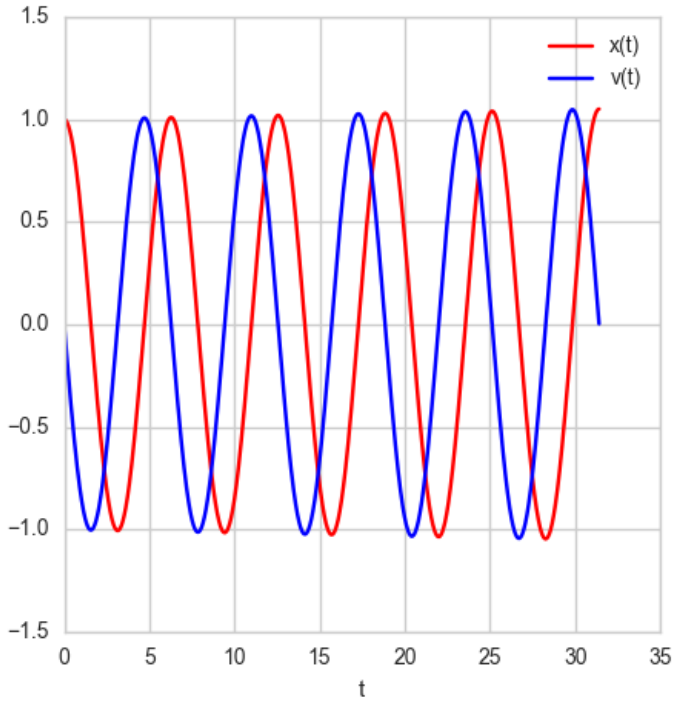
```
    plt.scatter(hMults, maxErr)
    plt.savefig('manyErrors.png')
    plt.clf()

def Energy(xVals, vVals, tVals, impExp):
    '''Plots the energy x**2 + v**2 over t.
    Requires whether it is 'imp', 'exp', or 'sym' to save the file'''
    energy = xVals ** 2 + vVals ** 2
    plt.plot(tVals, energy, color='blue', label='energy')
    assert impExp in ['imp', 'exp', 'sym']
    if impExp == 'imp': plt.savefig('impEnergy.png')
    elif impExp == 'sym': plt.savefig('symEnergy.png')
    else: plt.savefig('energy.png')
    plt.clf()
    tVals2 = np.copy(tVals)
    plt.plot(tVals, energy, color='blue', label='energy')
    plt.plot(tVals, np.cos(tVals2) - xVals + 1, color='red', label='x(t)
    if impExp == 'imp': plt.savefig('impEnergyWithErr.png')
    elif impExp == 'sym': plt.savefig('symEnergyWithErr.png')
    else: plt.savefig('energyWithErr.png')
    plt.clf()

def phaseSpace(xVals, vVals, tVals, impExp):
    '''Plots the phase-space of the trajectory produced by a given set
    x values and v values. Compares to an analytic solution for the give
    t values. Uses impExp to name a file ('imp', 'exp', or 'sym').'''
    tVals2 = np.copy(tVals)
    analX = np.cos(tVals)
    analV = -np.sin(tVals2)
    plt.plot(xVals, vVals, color='red', label=impExp + ' approx')
    plt.plot(analX, analV, color='blue', label='analytic')
    plt.legend()
    plt.xlabel('x')
    plt.ylabel('v')
    plt.xlim(-1.8, 1.8)
    plt.ylim(-1.8, 1.8)
    plt.savefig(impExp + 'PhaseSpace.png')
    plt.clf()
```

```
def allPhaseSpace(x0, v0, h):
    '''Creates a plot of phase space for all of the explicit, implicit,
    and symplectic approximations for a given set of initial conditions
    h, expX, expV, tVals = expEuler(x0, v0, h)
    h, impX, impV, tVals = impEuler(x0, v0, h)
    h, symX, symV, tVals = symEuler(x0, v0, h)
    plt.plot(expX, expV, color='red', label='exp approx')
    plt.plot(impX, impV, color='blue', label='imp approx')
    plt.plot(symX, symV, color='green', label='sym approx')
    plt.legend()
    plt.xlabel('x')
    plt.ylabel('v')
    plt.xlim(-1.8, 1.8)
    plt.ylim(-1.8, 1.8)
    plt.savefig('allPhaseSpace.png')
    plt.clf()

def figGen():
    '''Generates the figures for the Latex document accompanying the co
    Calling this function followed by compiling the Latex document will
    give the pdf report.'''
    expEuler1 = expEuler(1, 0, np.pi/1000)
    eulerPlot(expEuler1[0], expEuler1[1], expEuler1[2], expEuler1[3], 'e
    eulerError('exp')
    manyErrorPlots(np.pi/1000)
    Energy(expEuler1[1], expEuler1[2], expEuler1[3], 'exp')
    impEuler1 = impEuler(1, 0, np.pi/1000)
    eulerPlot(impEuler1[0], impEuler1[1], impEuler1[2], impEuler1[3], 'i
    eulerError('imp')
    Energy(impEuler1[1], impEuler1[2], impEuler1[3], 'imp')
    expEuler2 = expEuler(1, 0, np.pi/100)
    impEuler2 = impEuler(1, 0, np.pi/100)
    phaseSpace(expEuler2[1], expEuler2[2], expEuler2[3], 'exp')
    phaseSpace(impEuler2[1], impEuler2[2], impEuler2[3], 'imp')
    symEuler1 = symEuler(1, 0, np.pi/100)
    symEuler2 = symEuler(1, 0, np.pi/25)
    phaseSpace(symEuler2[1], symEuler2[2], symEuler2[3], 'sym')
```

```
allPhaseSpace(1, 0, np.pi/100)
Energy(symEuler1[1], symEuler1[2], symEuler1[3], 'sym')

# Execute figGen to get all of the figures
figGen()
```

4 Solution to Simple Harmonic Oscillator by Explicit Euler Method

We use the explicit Euler Method equations $x_{i+1} = x_i + hv_i$ and $v_{i+1} = v_i - hx_i$ and iterate through them with $h = \frac{\pi}{1000}$. Our initial conditions are set to $x(0) = 1$ and $v(0) = 0$.



5 Analytic Solution to Simple Harmonic Oscillator

We will derive the equation of motion for a simple harmonic oscillator with initial position $x(0) = 1$ and initial velocity $\dot{x}(0) = 0$. The differential form of the equation is

$$\ddot{x} = \frac{-k}{m}x$$

Since we are investigating an oscillator, we will guess a solution to be of the form $x(t) = acos(\omega t + \phi)$ where a, ϕ are constants representing the amplitude and phase shift of oscillation, respectively, and ω is the frequency of oscillation. Then $\ddot{x}(t) = -a\omega^2 cos(\omega t + \phi)$ so our differential equation becomes

$$-a\omega^2 cos(\omega t + \phi) = \frac{-k}{m}acos(\omega t + \phi)$$

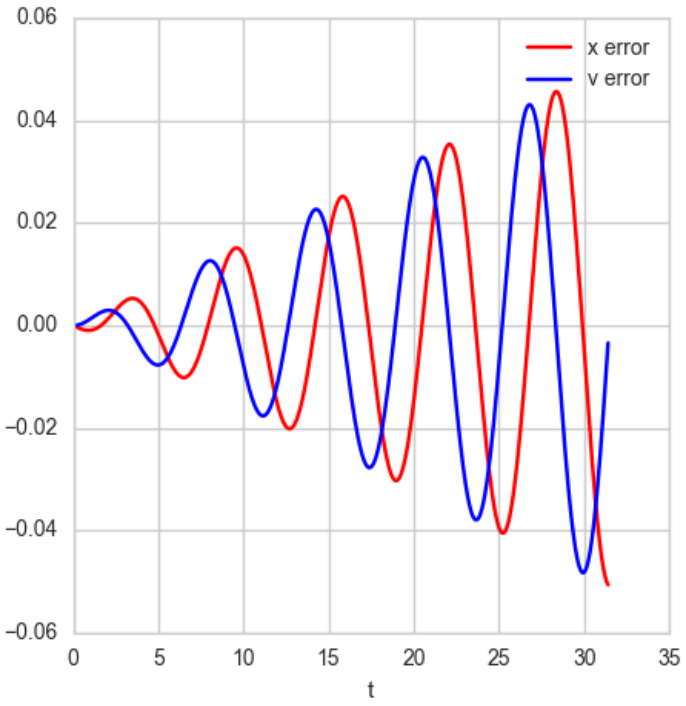
We can divide out the common terms on both sides, leaving $\omega^2 = \frac{k}{m}$ or $\omega = \sqrt{\frac{k}{m}}$. Thus we have revised our solution to $x(t) = acos(\sqrt{\frac{k}{m}}t + \phi)$.

Now we can use our initial conditions $x(0) = 1$ and $\dot{x}(0) = 0$. Using our solution, we see $x(0) = acos(\phi) = 1$. Differentiating $x(t)$ yields $\dot{x}(t) = -a\sqrt{\frac{k}{m}}sin(\sqrt{\frac{k}{m}}t + \phi)$ so $\dot{x}(0) = -a\sqrt{\frac{k}{m}}sin(\phi) = 0$. Since a, k, m, are all constants and are nonzero in nontrivial cases, then $sin(\phi) = 0$ so $\phi = 0$. Then if $\phi = 0$ our solution for $x(0)$ becomes $x(0) = a = 1$ so $a = 1$. Thus our final solution is

$$x(t) = cos(\sqrt{\frac{k}{m}}t)$$

Additionally, if $\frac{k}{m} = 1$ as given in the problem, the solution is $x(t) = cos(t)$ and $v(t) = -sin(t)$.

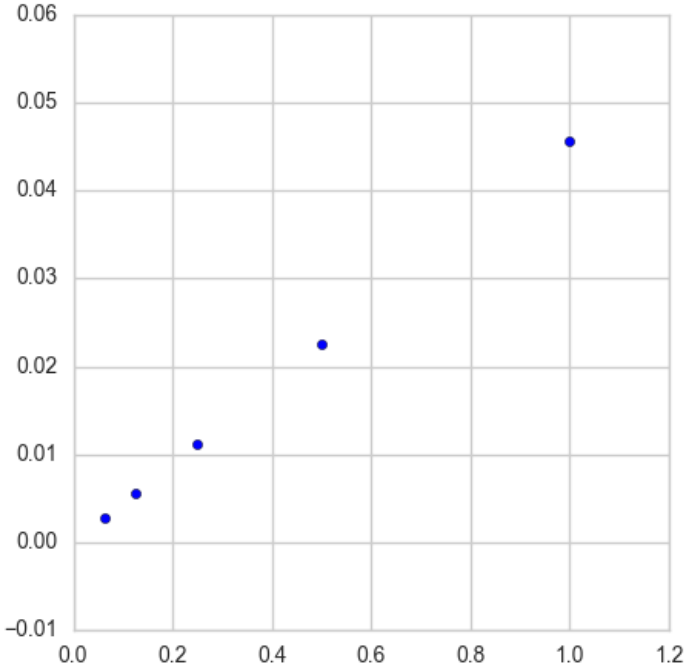
We plot the global error $x_{analytic}(t_i) - x_i$ and $v_{analytic}(t_i) - v_i$.



We can see the amplitude of the error increases linearly as t increases, but the error is relatively small on this range of values. The fact that we chose a small $h = \frac{\pi}{1000}$ means that our approximation is quite accurate.

6 Relationship Between Error and h

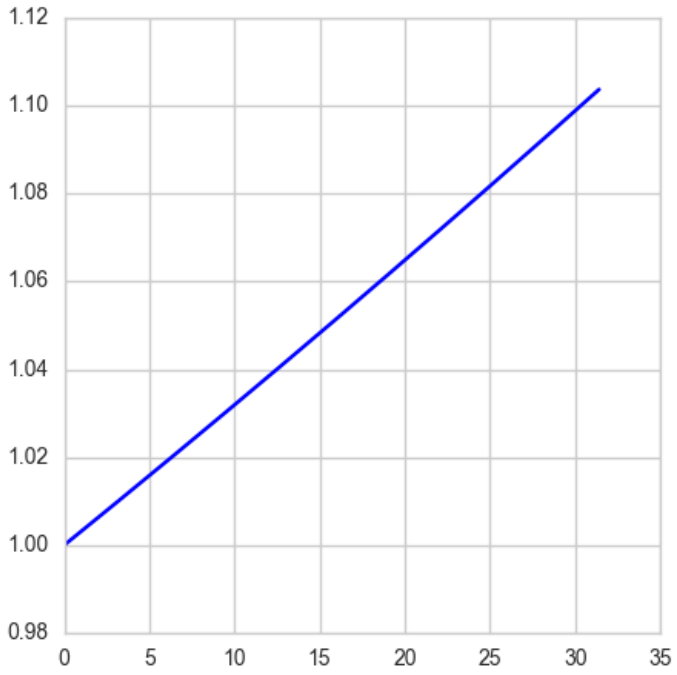
We plot the error between the analytic and explicit Euler solutions to the simple harmonic oscillator for different values of h. For the values of h, we have chosen $h_0 = \frac{\pi}{1000}$ and then divide that by 2 until we get to $\frac{1}{16}h_0$. This means every increase in h is twofold.

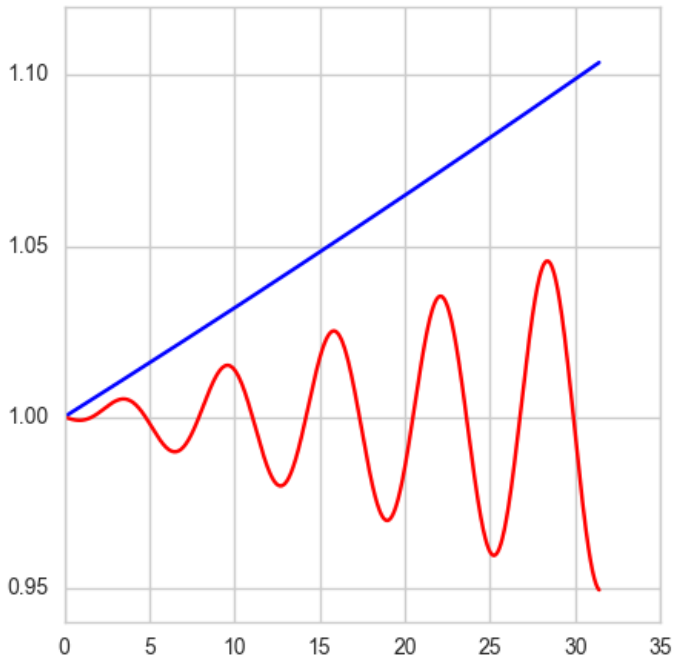


We see that the relationship between the maximum amplitude of the error on the interval $[0, 10\pi]$ and the value of h is linear. They are directly proportional.

7 Energy of the Approximation Over Time

In a simple harmonic oscillator, the total energy is supposed to be conserved. We check the legitimacy of our approximation by looking at how the energy changes with time in our approximation. We plot the simple form of energy $(x^2 + v^2) - 1$ against time, and also show the top portion of the error in $x(t)$ to compare the growth of the energy with the growth in amplitude of error. The reason we subtract the 1 is to see the error and the energy on the same scale.


1.12
1.10
1.08
1.06
1.04
1.02
1.00
0.98
0
5
10
15
20
25
30
35



We can see from the top figure that the energy does increase linearly with time. Looking at the bottom figure, the amplitude of the error also increases linearly. However, the energy increases at twice the rate of the error.

8 The Implicit Euler Method

The implicit Euler method is given by the matrix equation

$$\begin{pmatrix} 1 & -h \\ h & 1 \end{pmatrix} \times \begin{pmatrix} x_{i+1} \\ v_{i+1} \end{pmatrix} = \begin{pmatrix} x_i \\ v_i \end{pmatrix}$$

We would like to solve this equation for expressions of x_{i+1} and v_{i+1} in terms of x_i and v_i, respectively. We do this by inverting our two-by-two matrix.

$$\begin{pmatrix} 1 & -h \\ h & 1 \end{pmatrix}^{-1} = \frac{1}{1+h^2} \times \begin{pmatrix} 1 & h \\ -h & 1 \end{pmatrix}$$

Our matrix equation is then

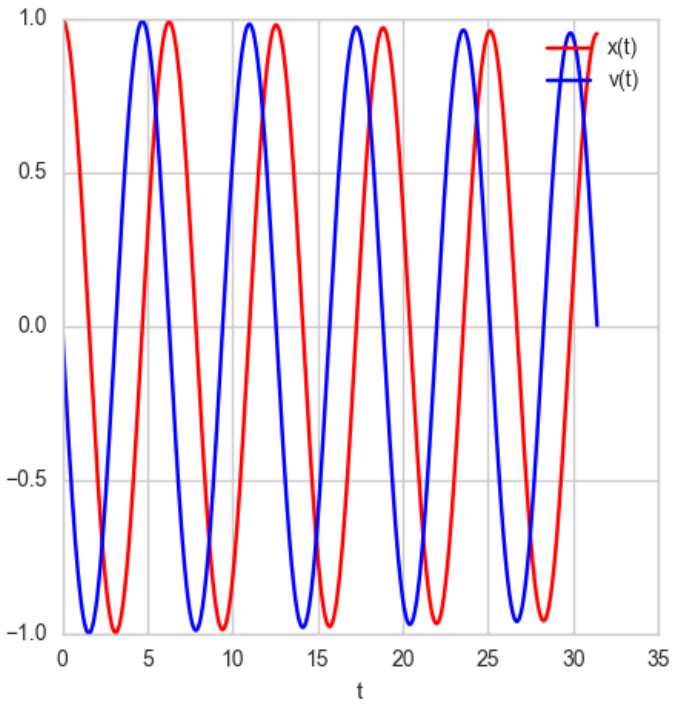
$$\begin{pmatrix} x_{i+1} \\ v_{i+1} \end{pmatrix} = \frac{1}{1+h^2} \times \begin{pmatrix} 1 & h \\ -h & 1 \end{pmatrix} \times \begin{pmatrix} x_i \\ v_i \end{pmatrix}$$

which reduces to

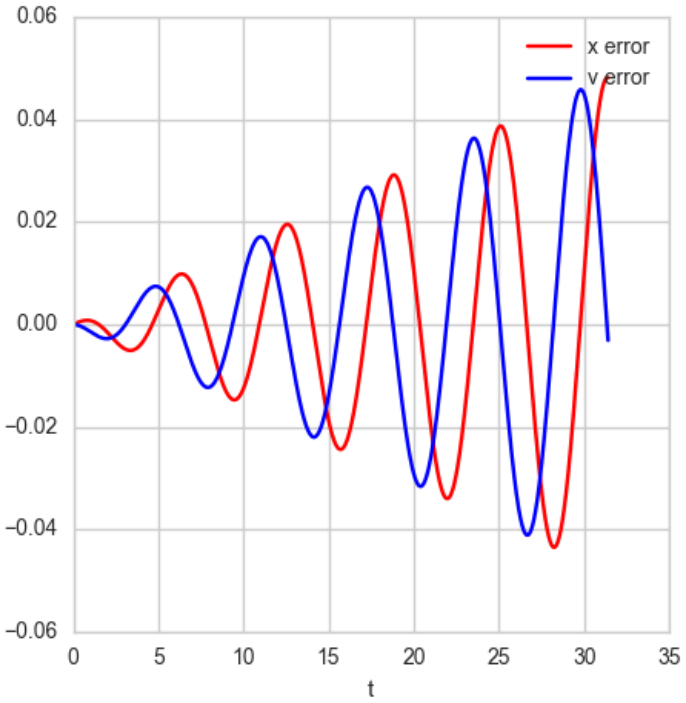
$$\begin{pmatrix} x_{i+1} \\ v_{i+1} \end{pmatrix} = \frac{1}{1+h^2} \times \begin{pmatrix} x_i + hv_i \\ -hx_i + x_i \end{pmatrix}$$

9 Comparison of the Implicit and Explicit Euler Methods

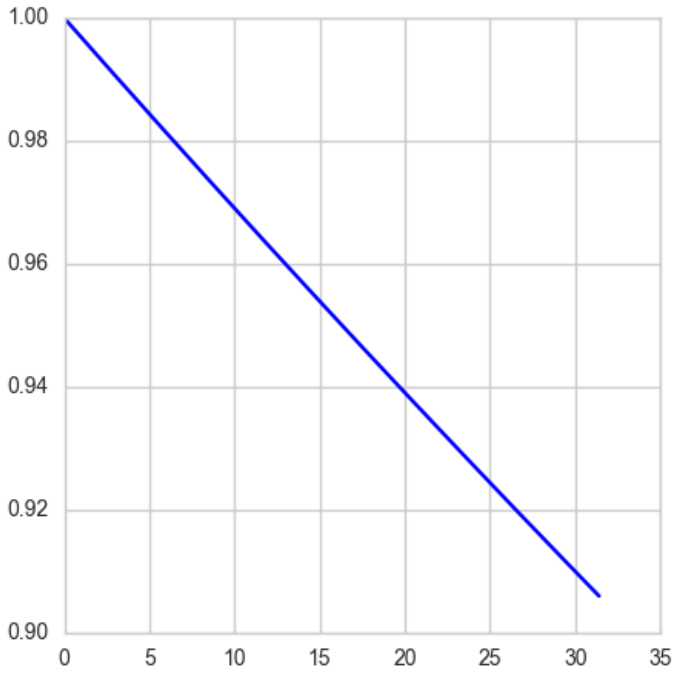
Here we will look at graphics for the implicit Euler method calculated with $h = \frac{\pi}{1000}$. All of the figures are comparable to those shown for the explicit Euler method.

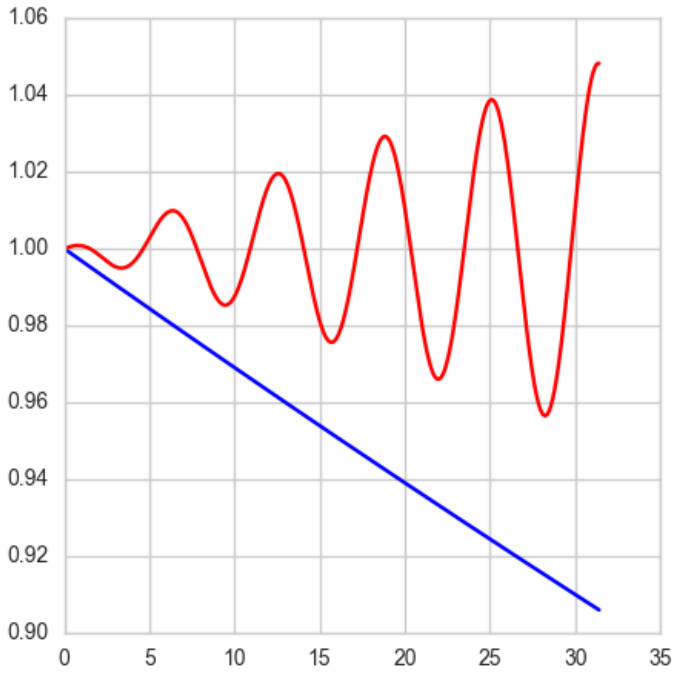


Here we see that the explicit and implicit Euler methods each give sinusoids, but while the explicit Euler method shows an increasing amplitude, the implicit Euler method's amplitude decreases over time.



When we analyze the error from the implicit Euler method, we see that the magnitude is identical to that of the explicit method. This means the implicit and explicit methods have the same rate of increase in error, and that one just is decreasing away from the actual solution and one is increasing away from it.

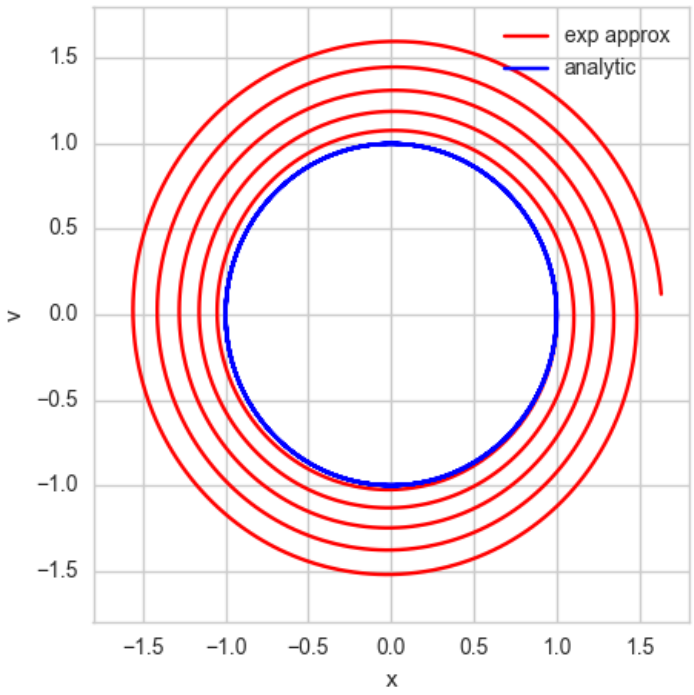

1.00
0.98
0.96
0.94
0.92
0.90
0
5
10
15
20
25
30
35

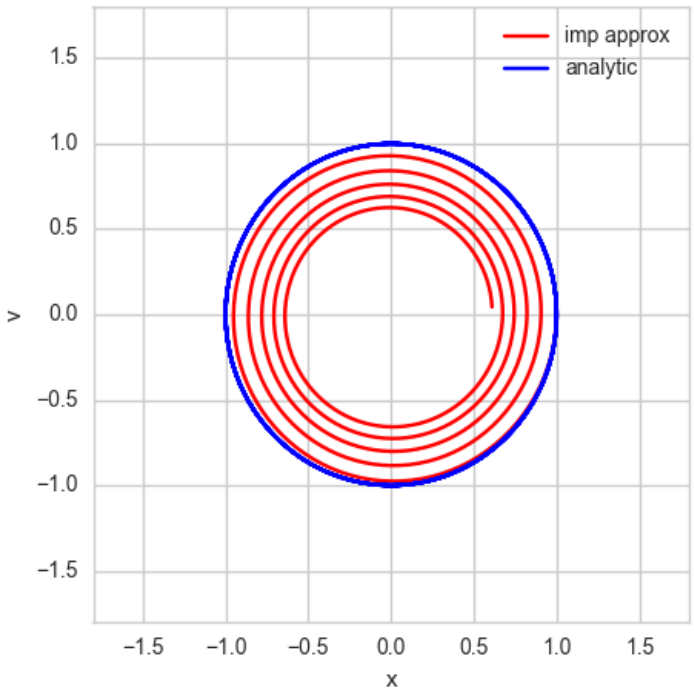


Now we can confirm the belief that the implicit and explicit solutions give mirror image approximations centered around the actual solution. We see above that the energy decreases linearly, and decreases twice as fast as the rate of change of the amplitude of the error, just as the explicit solution did. Thus the implicit and explicit solutions are equally effective, and neither is perfect.

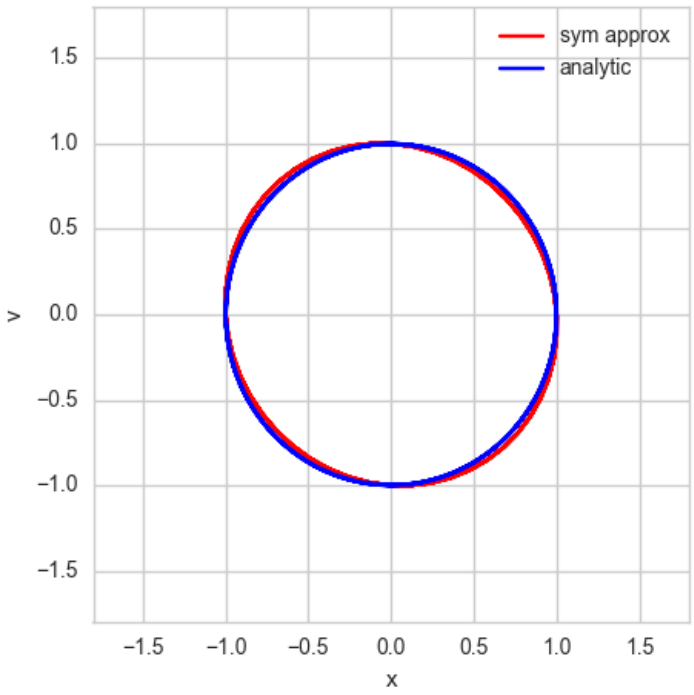
10 Symplectic Euler Method

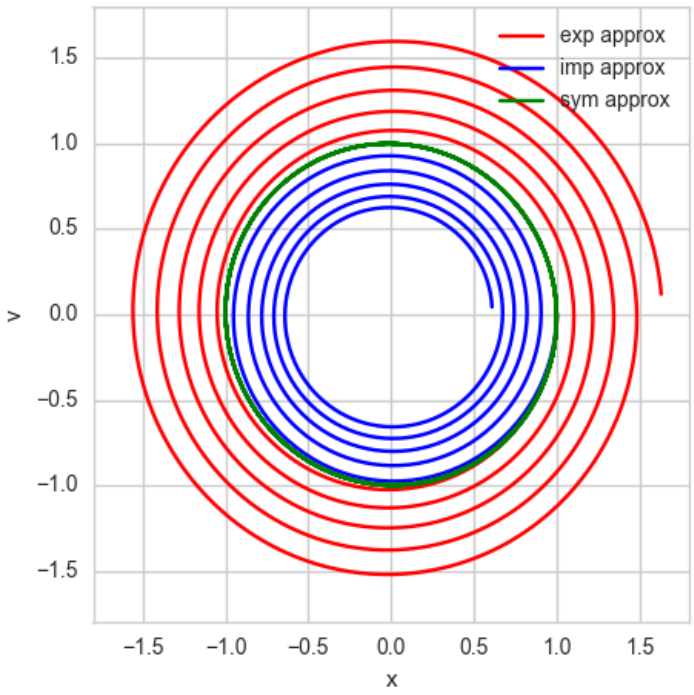
We will first look at the phase space plots of x vs. v for the explicit and implicit Euler methods. In these plots, we have set $h = \frac{\pi}{100}$.


exp approx
analytic
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
v
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
x

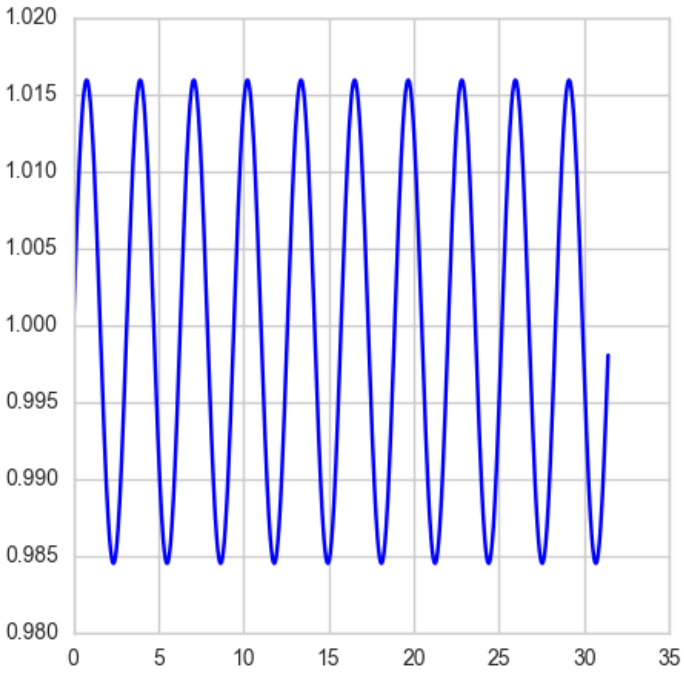


We see that for the top figure, the plots spirals out, and for the bottom figure, the plot spirals in. The blue circle is the analytic solution, so we can see how the approximations spiral away from the analytic solution as t increases.


sym approx
analytic
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
v
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
x



Here we have added in the symplectic approximation, which is a middle ground between the implicit and explicit methods. We notice that in the top figure, where we have decreased h to $\frac{\pi}{25}$, the symplectic approximation and the analytic solution are nearly indistinguishable in phase space. However, we can see that the symplectic approximation actually fluctuates slightly out and in relative to the analytic solution. In some areas, the two lines overlap perfectly, but elsewhere we can see both colors next to each other. This means the symplectic approximation does not perfectly approximate the analytic solution but instead oscillates around it. In the bottom figure, we remove the analytic solution and put all of the approximations together. Additionally, we return the h for the symplectic approximation to $\frac{\pi}{100}$. Here we can see the difference between the approximations, and see how well the symplectic approximation preserves the analytic solution's phase space geometry compared to the other two.



Now we can see that the symplectic approximation has fluctuating energy. The energy typically is centered around $x^2 + v^2 = 1$. The fluctuating energy explains the minute oscillations we see in the phase space figure for the symplectic approximation.