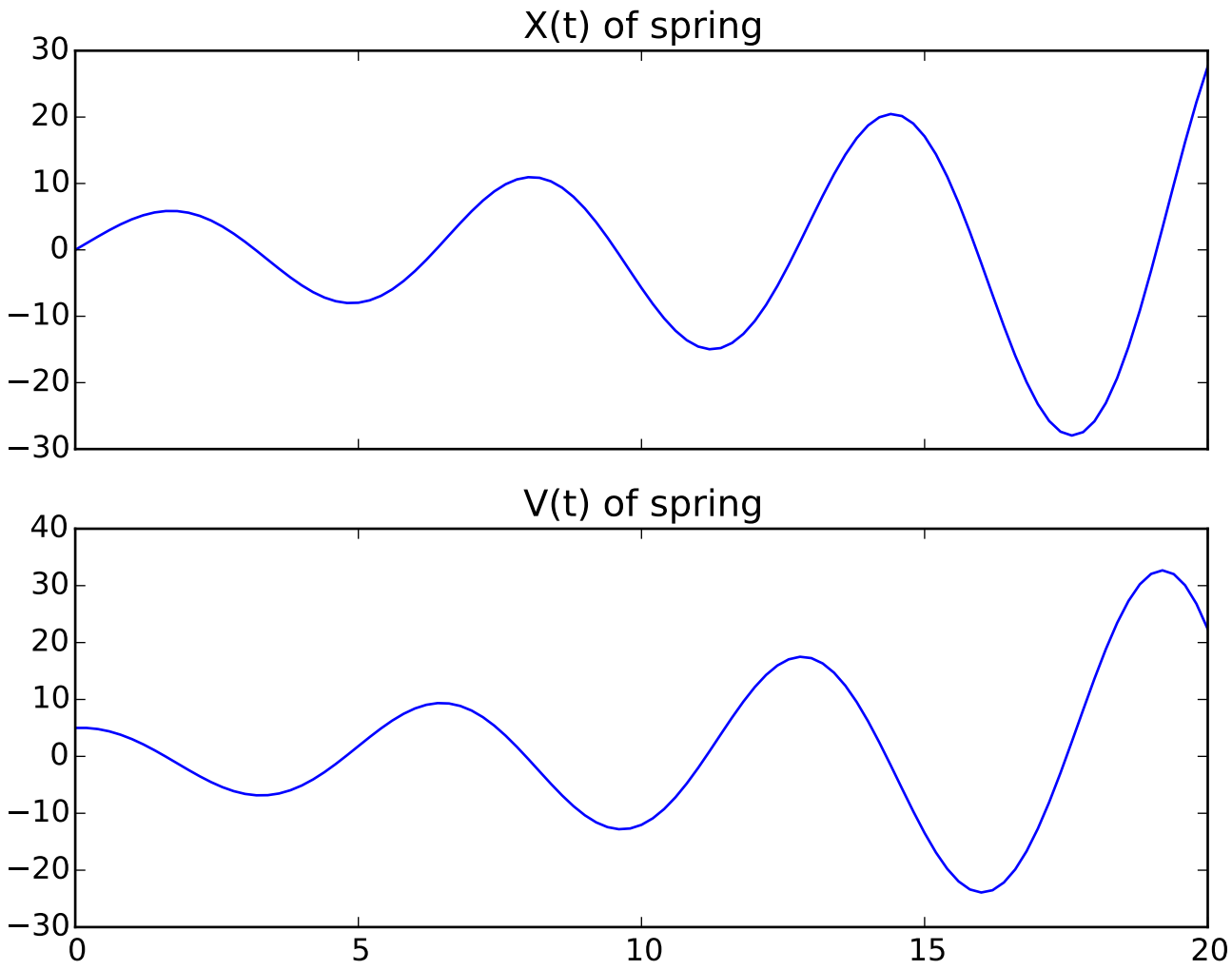
Rita Sonka
Ph 20
homework set 3
10/11/2016
Week 4 makefile Compilation: 10/25/2016

========== WEEK 3 STUFF ==========

Problem 1.
See attached code as well. The values input to this one are $x_0 = 0$, $v_0 = 5$, h = .2, N = 100 (so 20 units of time).

Figure 1: Explicit Euler



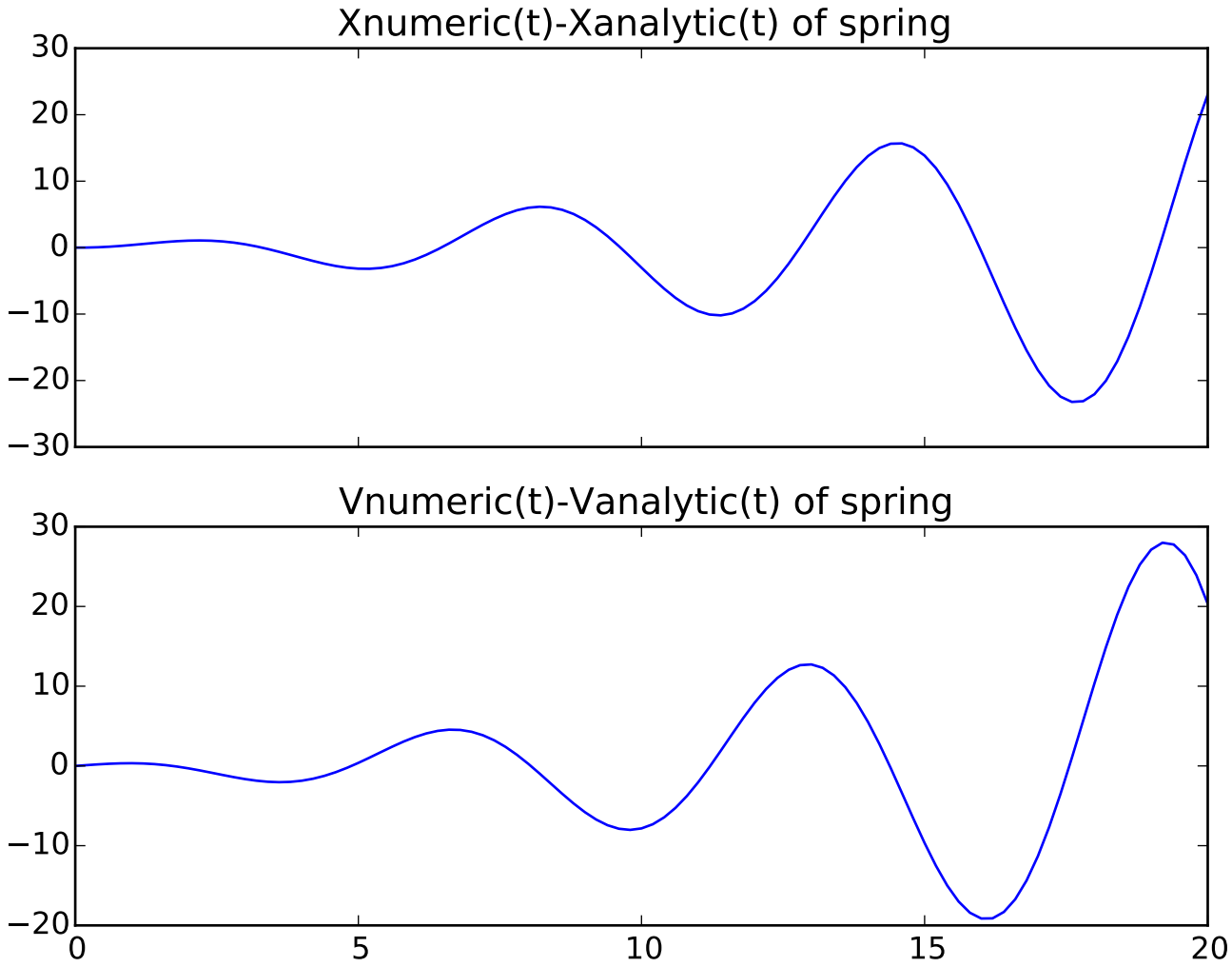
Problem 2.
I have done this a bunch of times and I'm sure my grader has too, so I'm going to skip a lot of steps.

$$\text{Differential equation:} \quad m * \ddot{x} + k * x = 0 \tag{1}$$

$$\text{We're setting k/m} = 1 \text{ for convenience.} \quad \ddot{x} + x = 0 \tag{2}$$

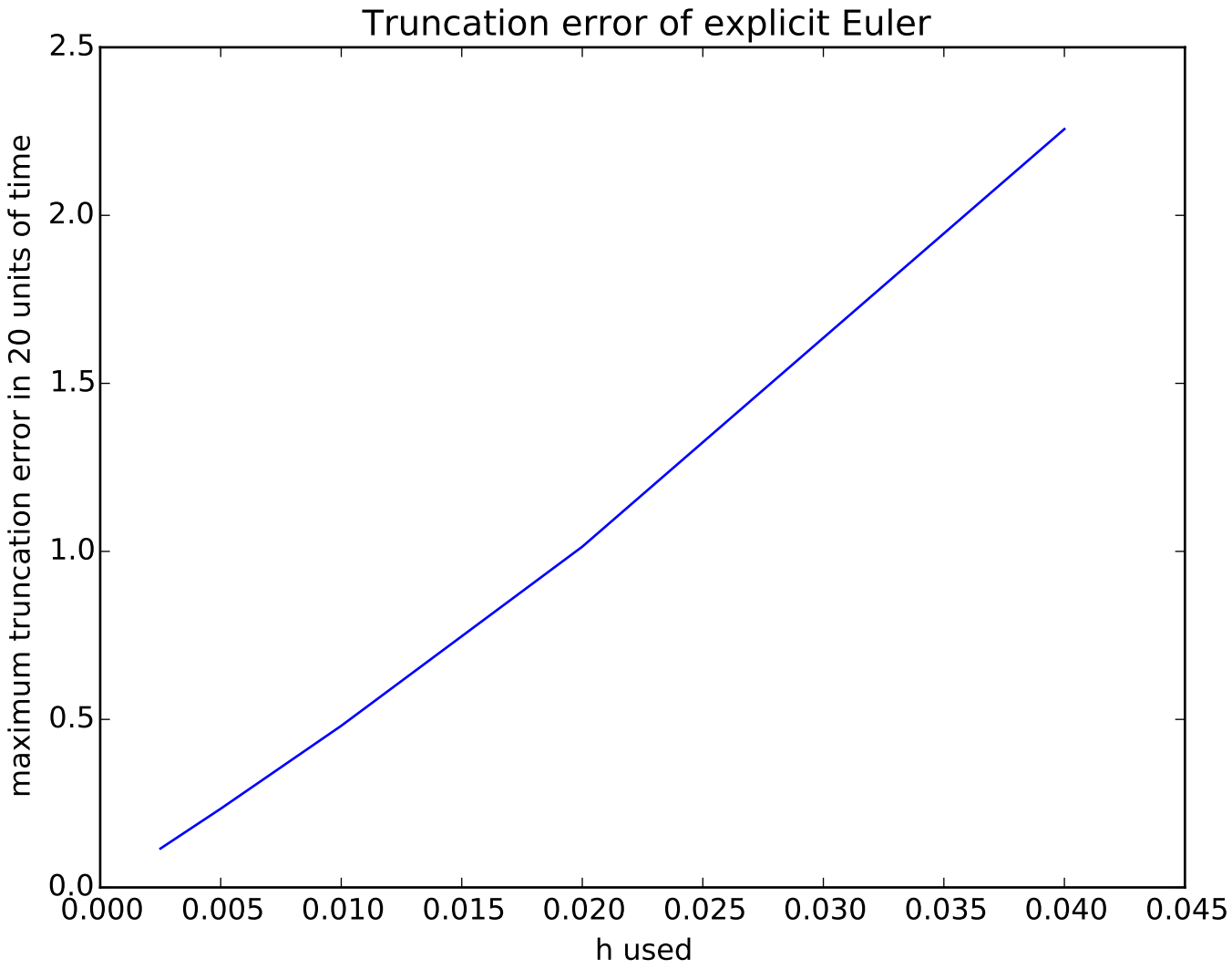
$$\text{Thus the answer is:} \quad x(t) = v_0 * Sin(t) + x_0 * Cos(t) \quad \text{where } v_0 \text{ is initial velocity and } x_0 \text{ is initial position.} \tag{3}$$

Figure 2: Explicit Euler Errors (same input as Figure 1)



Problem 3.

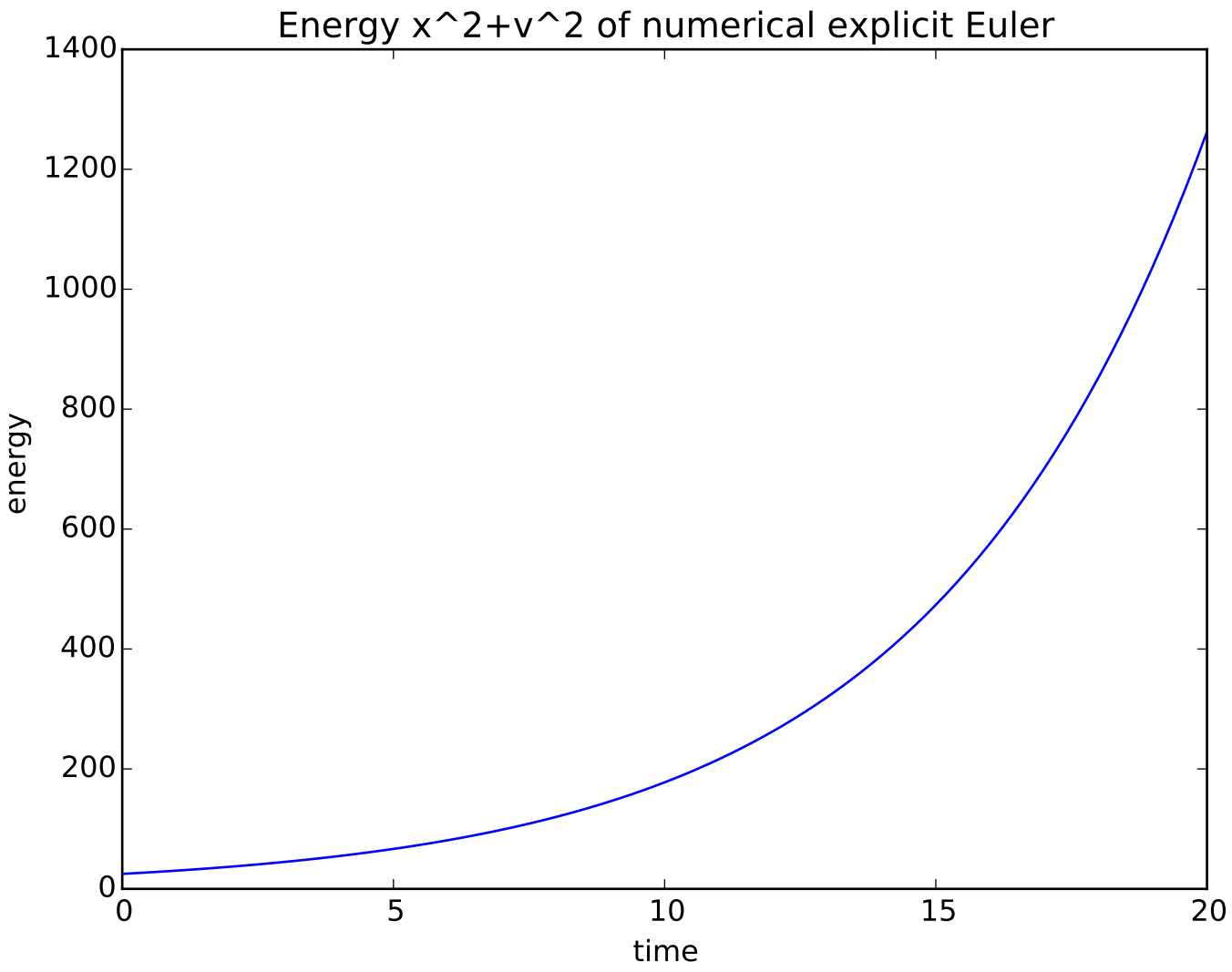
Figure 3: Explicit Euler Truncation Errors as a function of h



Pretty close to linear–and it only gets moreso as h gets smaller.

Problem 4.

Figure 4: Explicit Euler Energy (same input as Figure 1)



Just like the errors in x and v, it starts fairly linear in time but quickly becomes exponentially worse.

Problem 5.
We're starting from the implicit Euler methods:

$$x_{i+1} = x_i + h * v_{i+1} \qquad v_{i+1} = v_i - h * x_i \tag{4}$$

Now we simply solve a linear system of two equations in two unknowns.

$$v_{i+1} = v_i - h * (x_i + h * v_{i+1}) \tag{5}$$

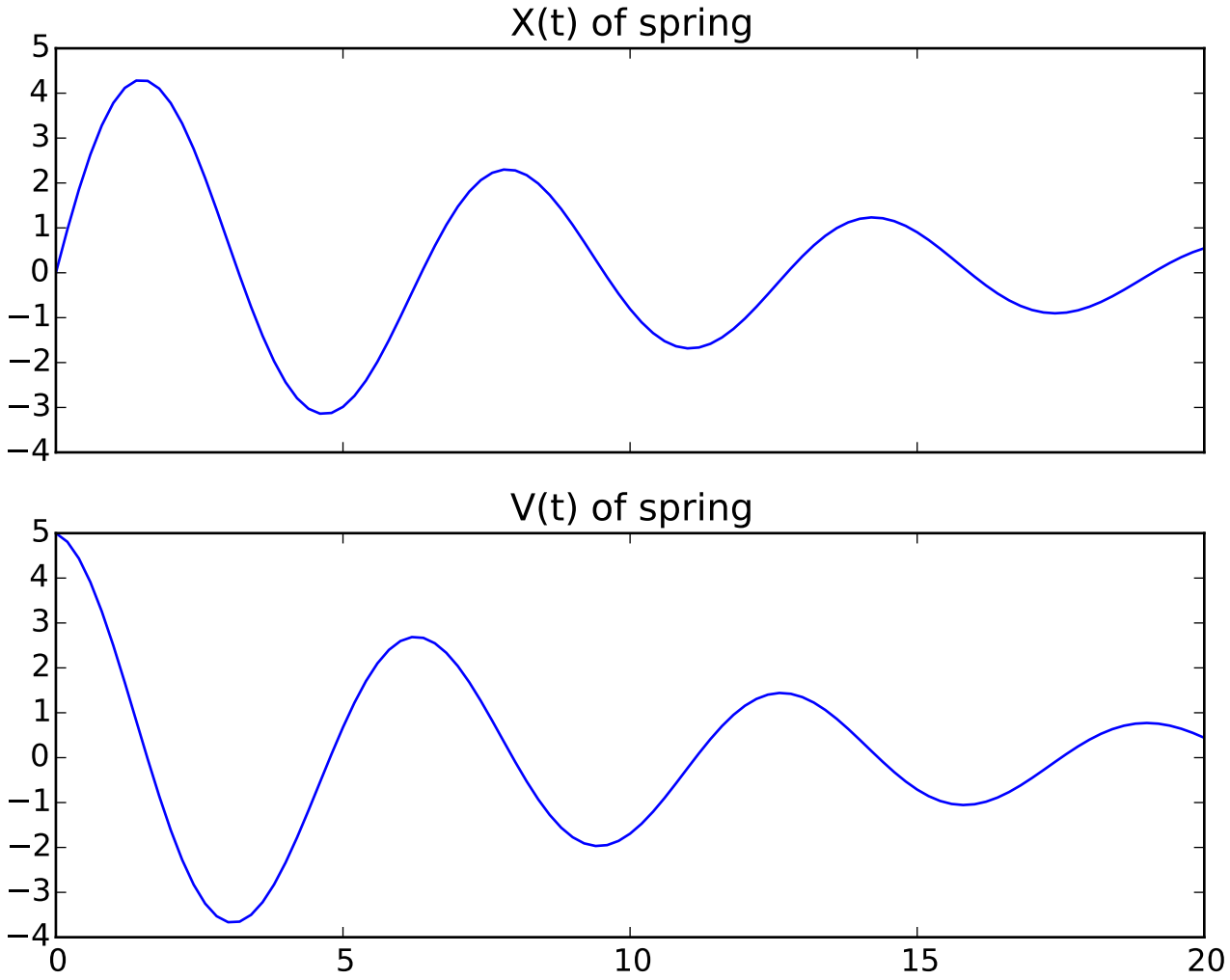
$$v_{i+1} = \frac{v_i - h * x_i}{1 + h^2} \tag{6}$$

$$x_{i+1} = x_i + h * (\frac{v_i - h * x_i}{1 + h^2}) \tag{7}$$

$$x_{i+1} = \frac{x_i + h * v_i}{1 + h^2} \tag{8}$$

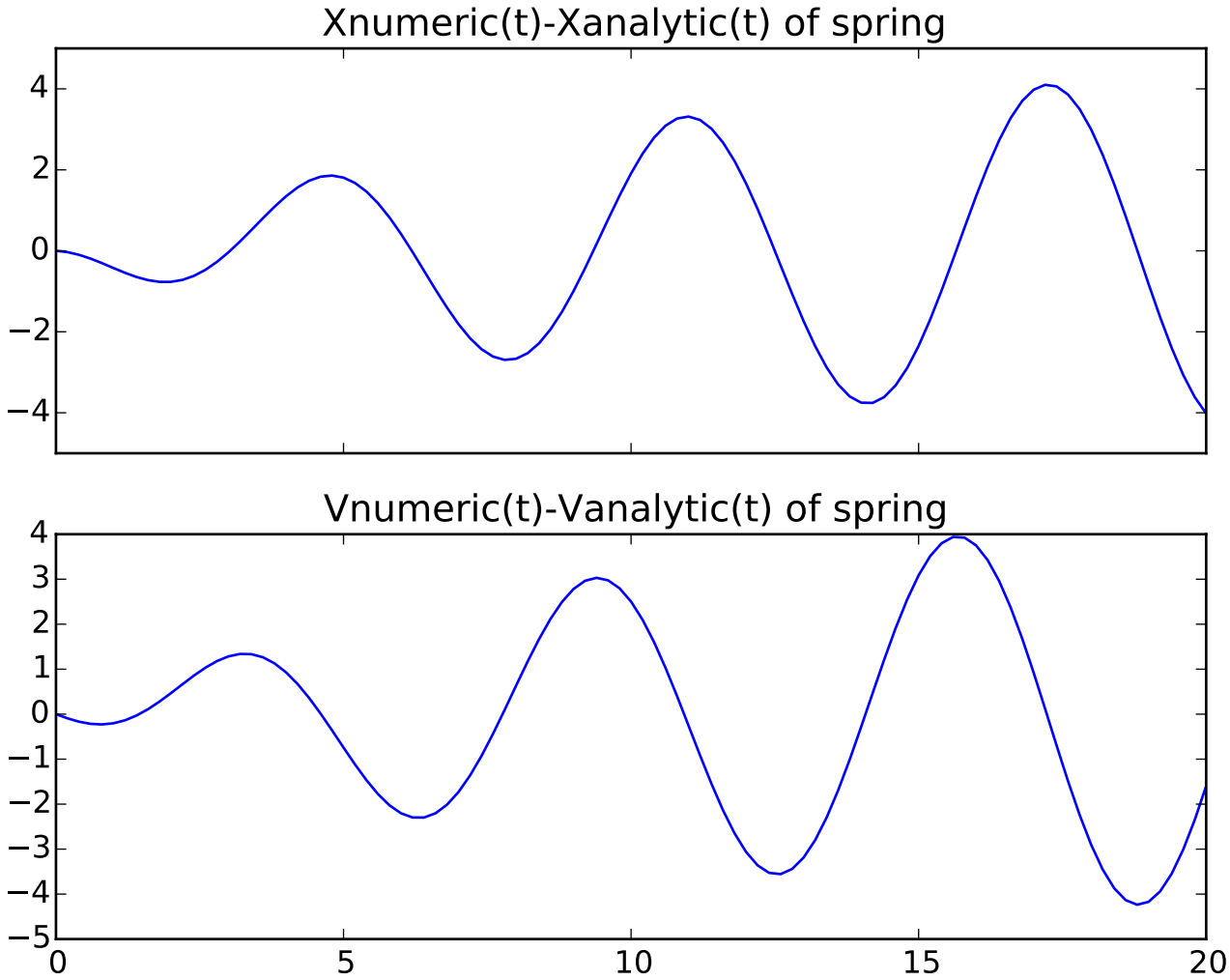
And now the results:

Figure 5: Implicit Euler (same input as Figure 1)



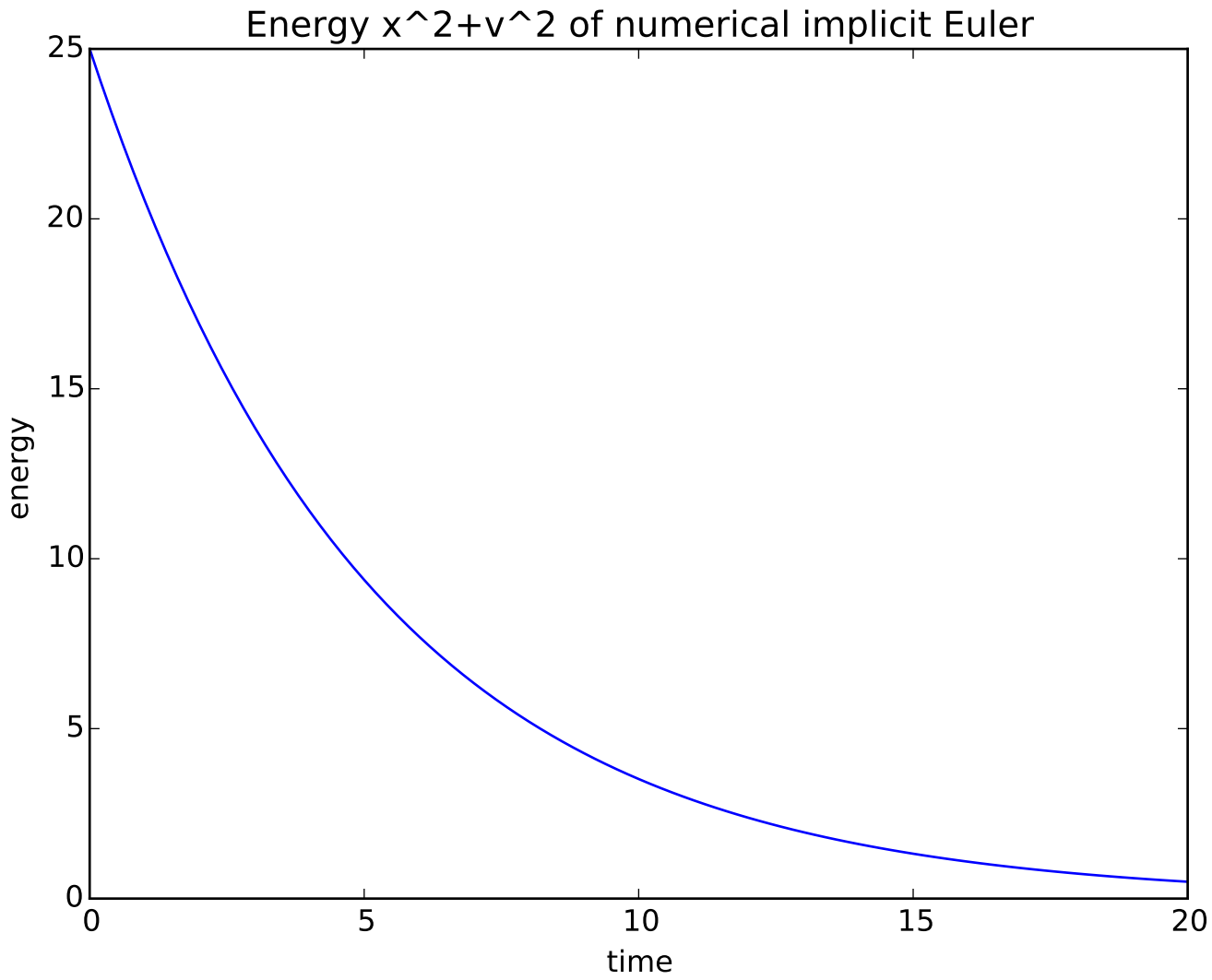
Compare to the analytic solution:

Figure 6: Implicit Euler Errors (same input as Figure 1)



We see here that Implicit Euler has the opposite effect of explicit Euler; explicit Euler "creates" energy, while implicit euler "destroys" energy, at roughly the same rate. Let's check this with the energy.

Figure 7: Implicit Euler Energy (same input as Figure 1)

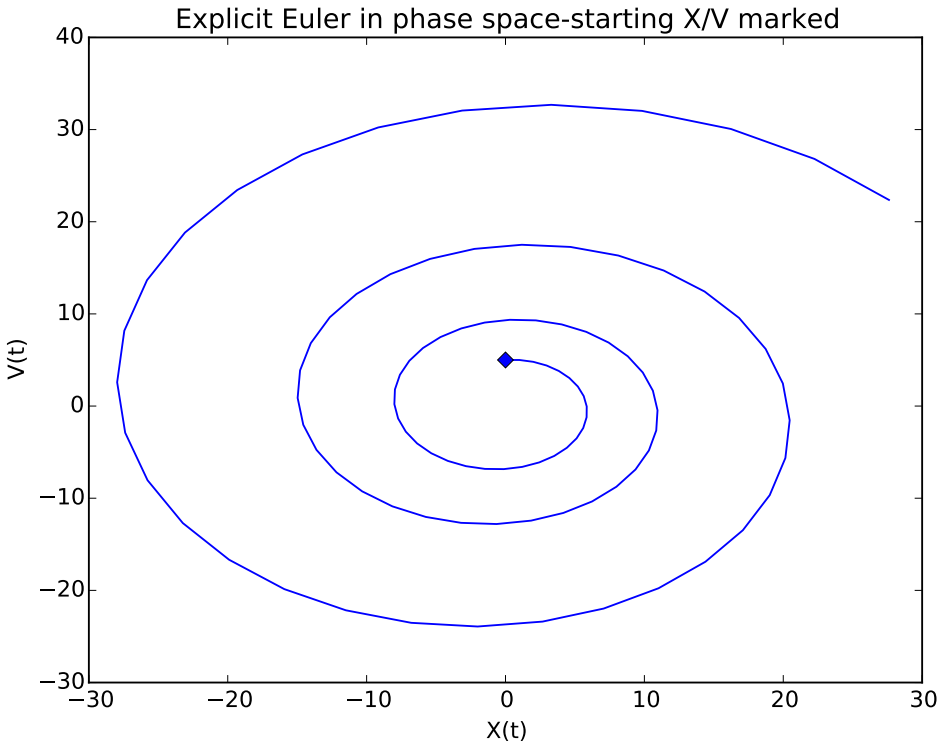


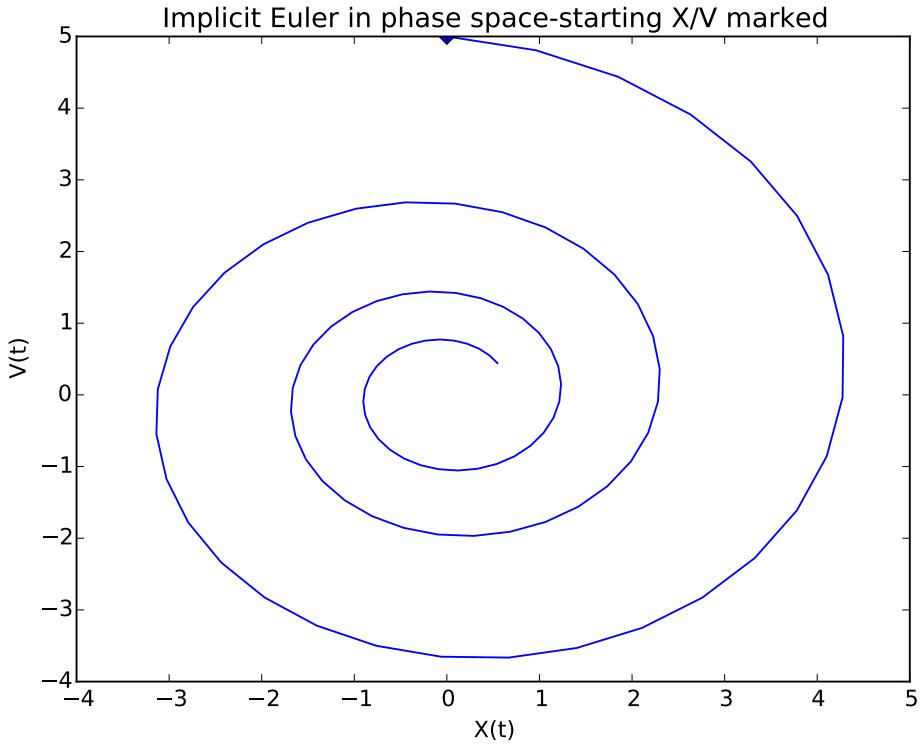
This confirms our expectation; the decrease in energy is roughly linear at the start but quickly becomes exponential.

Rita Sonka
Ph 20
homework set 3 PART 2
10/11/2016

Problem 1.

Figure 8: Explicit and Implicit Euler in Phase Space (same input as Figure 1)

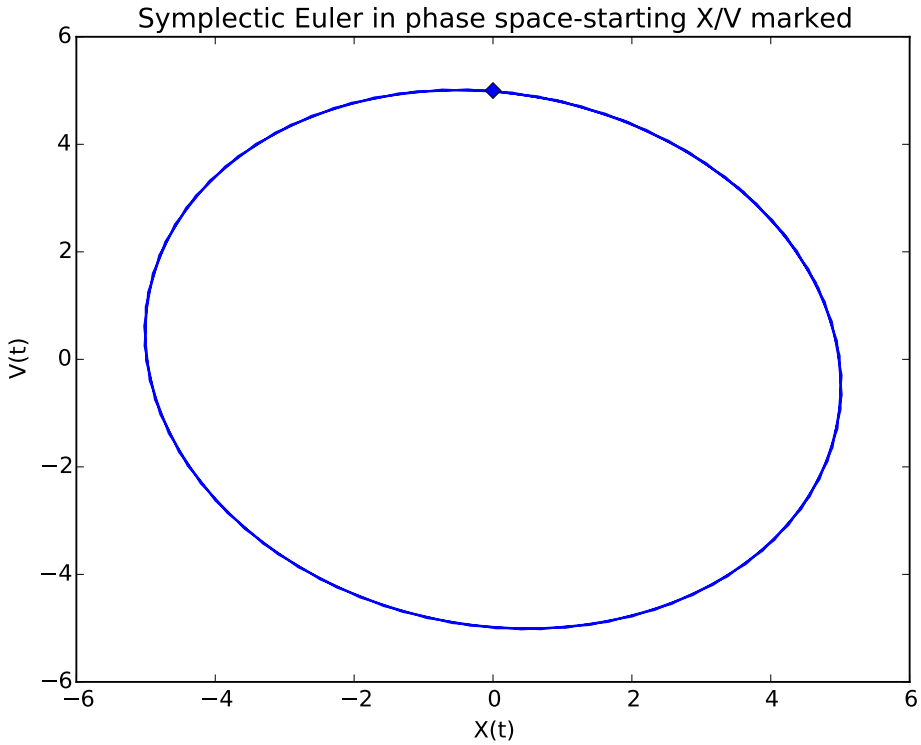




This pretty clearly demonstrates what we had already found–the explicit Euler increases in energy with time, while Implicit Euler decreases in energy with time. In phase space, this corresponds to Explicit Euler spiraling outwards and Implicit Euler spiralling inwards.

Problem 2.

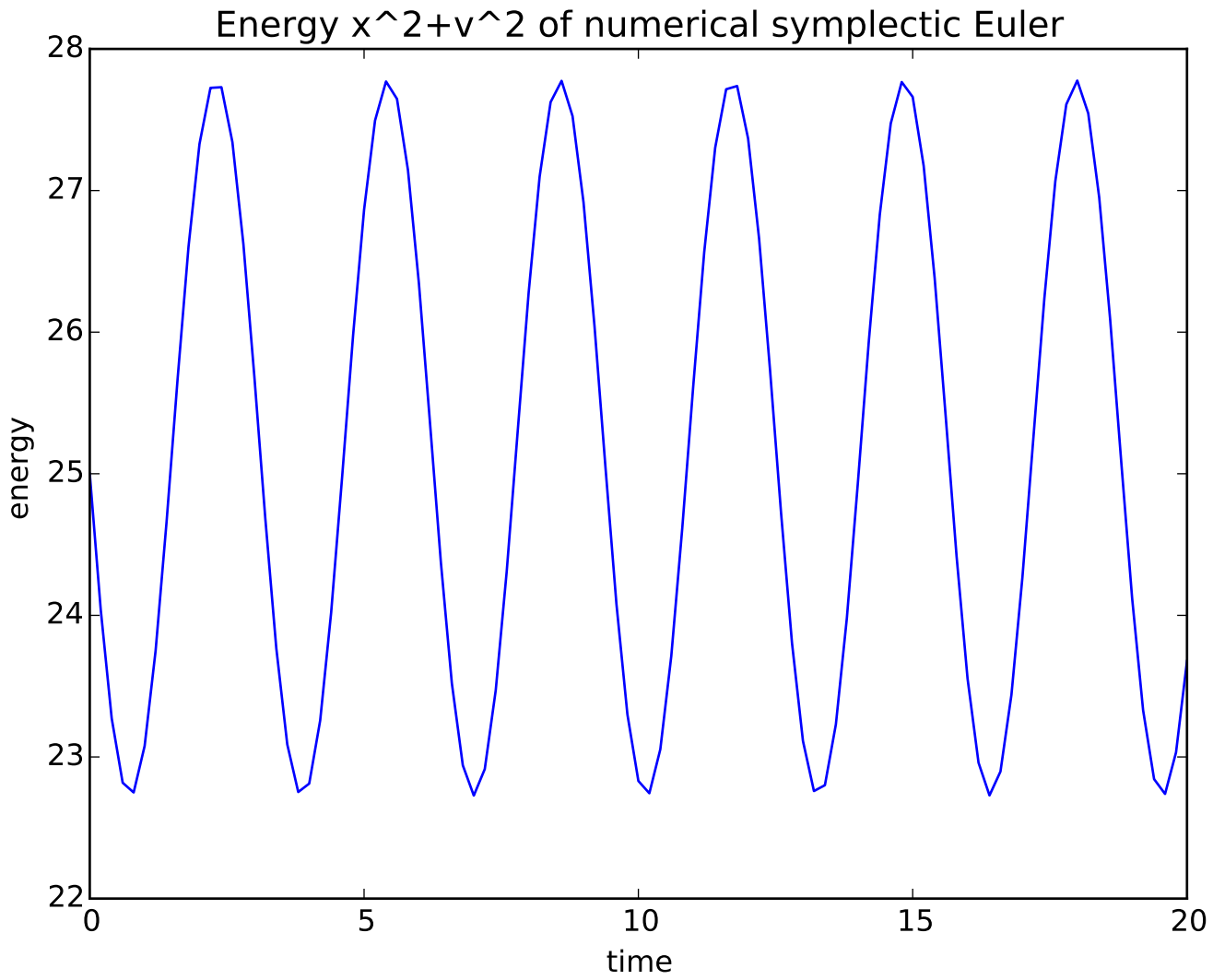
Figure 9: Symplectic Euler in Phase Space (same input as Figure 1)



Unlike Explicit or Implicit Euler, Symplectic Euler DOES conserve energy (on the average, anyway)–we see that it makes a (seemingly) closed circuit of the origin. However, this closed circuit is not a circle–for any time t, the exact values of X and V will likely be off (hard to tell exactly just from phase space).

Problem 3.

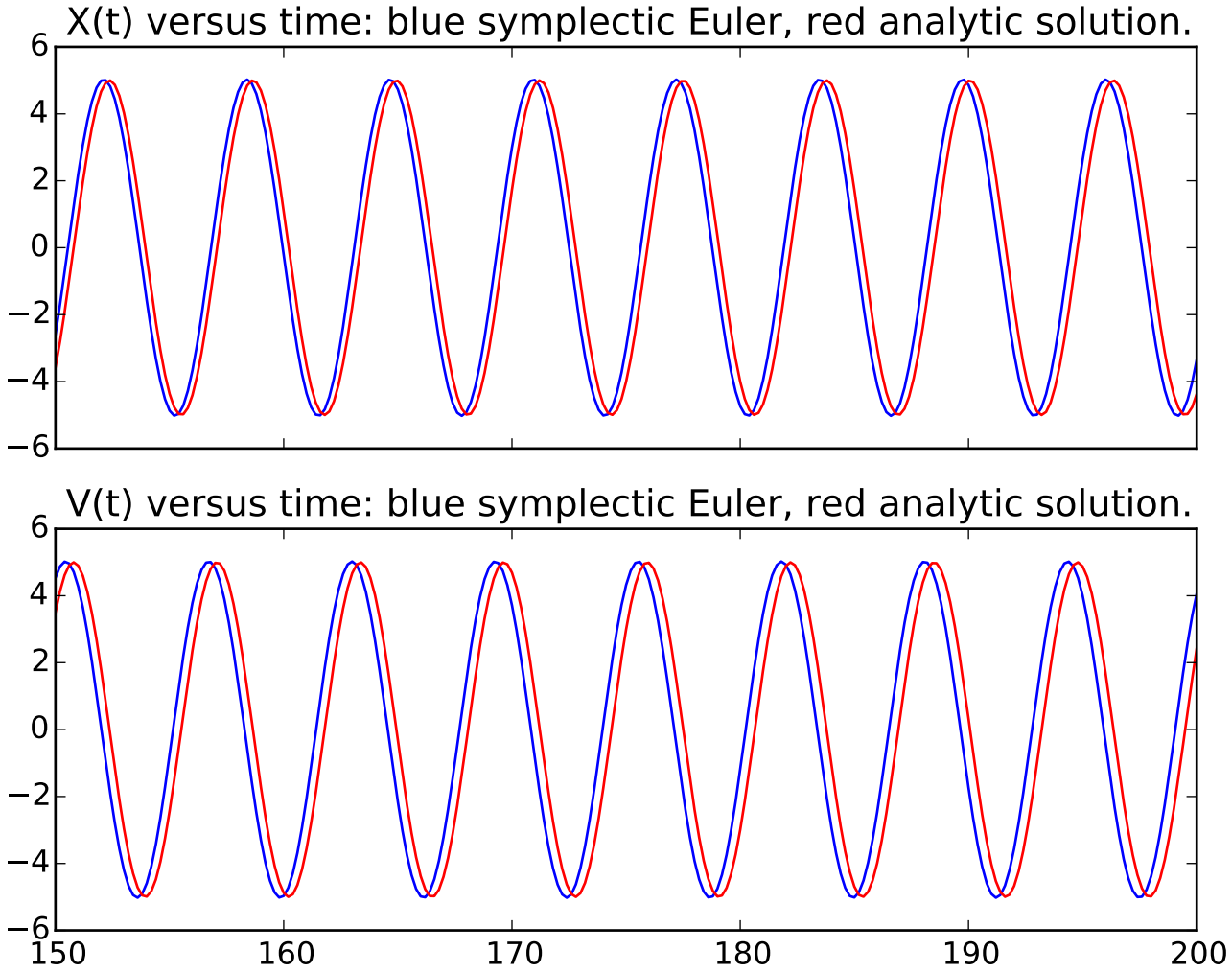
Figure 10: Symplectic Euler Energy (same input as Figure 1)



This gives a greater idea of the time evolution of the error. We see a sinusoidal variation in v^2+x^2 with time, corresponding to the ellipse we saw in phase space. So over time it will on average have the right energy, but at any one point the x or v or both is/are likely off.

Problem 4.

Figure 11: Symplectic Euler phase difference (same input as Figure 1, except N = 750-1000 instead of 0-100)



While it isn't obvious from this graph, the v(t) has actually been off by that exact amount since the very beginning, and has neither grown nor shrank. The X(t) on the other hand is clearly getting out of phase over the long term. So I guess that phase diagram would slowly turn into a very fuzzy ellipsoid if I had taken it out long enough.

========== THINGS THAT WERE INCLUDED SEPARATELY IN THE ORIGINAL WEEK 3, plus new week 4 stuff. ==========

========== Version control logs (dynamically updated via makefile):

```
commit 4cdfb2231bdd51d0c0db4d51ec8ab35704c22cb2
Author: Rita Sonka(Ph20) <rsonka@boojum.caltech.edu>
Date:   Tue Oct 25 16:37:49 2016 -0700

    Added verbatim input to Tex file.

commit d6c3fdbddf251d1f0ebefcbd19109797b4b2e05e
Author: Rita Sonka(Ph20) <rsonka@boojum.caltech.edu>
Date:   Tue Oct 25 16:12:50 2016 -0700

    Added in new files

commit ba54c1e246876b43d6739525d2b568544042532c
Author: Rita Sonka(Ph20) <rsonka@boojum.caltech.edu>
```

```
Date:   Tue Oct 25 16:11:15 2016 -0700

    Right before adding commit history to makefile

commit 128e6a6cf793962cfc66419e4bf250549c8b6d05
Author: Rita Sonka(Ph20) <rsonka@boojum.caltech.edu>
Date:   Tue Oct 25 13:47:04 2016 -0700

     Second Commit

commit f23f8d4abe46694bbb6f28c49de7e440deb4faaf
Author: Rita Sonka(Ph20) <rsonka@boojum.caltech.edu>
Date:   Tue Oct 18 17:59:19 2016 -0700

    first commit
```

========== Makefile:

```
# Makefile for Ph 20 week 3 assignment-done as part of week 4 assignment.
# Graphics Files
GF = week3Graphics/
# Graphics Generation
GG = plotGenerationScripts/


week3report.pdf : week3report.tex $(GG)* versionControlLog
pdflatex $^

$(GG)%.py :
python $@ $(GF)

.PHONY: versionControlLog
versionControlLog :
git log > versionControlLog.txt
```

========== Source Code:

```
'''Rita Sonka
10/11/2016
Ph 20
hw 3
'''
import numpy as np
import matplotlib.pyplot as plt




# Problem 1 ======================================

# Using the w=1 case....
def explicitEuler(x0, v0, h, N):
    TT = np.arange(N+1)*h
    XX = np.zeros([N+1])
    VV = np.zeros([N+1])
    XX[0] = x0
    VV[0] = v0
    for i in range(1, N+1):
        XX[i] = XX[i-1] + h*VV[i-1]
        VV[i] = VV[i-1] - h*XX[i-1]
```

```
    return (TT, XX, VV)

def plotExplicitEuler(x0, v0, h, N, saveName):
    (TT, XX, VV) = explicitEuler(x0, v0, h, N)
    #if saveName:
    plotXVT(TT, XX, VV, 'X(t) of spring', 'V(t) of spring', saveName=saveName)
    #else:
    #    plotXVT(TT, XX, VV, 'X(t) of spring', 'V(t) of spring')

def plotXVT(TT, XX, VV, xtitle, vtitle, saveName=""):
    f, axarr = plt.subplots(2, sharex=True)
    axarr[0].plot(TT, XX)
    axarr[0].set_title(xtitle)
    axarr[1].plot(TT, VV)
    axarr[1].set_title(vtitle)
    if saveName:
        plt.savefig(saveName, bbox_inches='tight')
    else:
        plt.show()
    plt.gcf().clear()


# Problem 2 ========================================

def analyticEuler(x0, v0, h, N):
    TT = np.arange(N+1)*h
    XX = x0*np.cos(TT) + v0*np.sin(TT)
    VV = - x0*np.sin(TT) + v0*np.cos(TT)
    return (TT, XX, VV)

# Using the w=1 case....
def explicitEulerErrors(x0, v0, h, N):
    (TT, XX, VV) = explicitEuler(x0, v0, h, N)
    (Ta, Xa, Va) = analyticEuler(x0, v0, h, N)
    return (TT, XX - Xa, VV - Va)


def plotExplicitEulerErrors(x0, v0, h, N, saveName):
    (TT, XX, VV) = explicitEulerErrors(x0, v0, h, N)
    plotXVT(TT, XX, VV, 'Xnumeric(t)-Xanalytic(t) of spring', 'Vnumeric(t)-Vanalytic(t) of spring', saveName=s


# Problem 3 ========================================

# It is slowly accumulating more options...
def plotXY(xX, yY, titleString, xlabel, ylabel, logLog=False, semiLogY=False, markPoints=[], saveName=""):
    # xX and yY should be numpy arrays.
    # the below bit was part of me attempting to prove the closed surface thing.
    if semiLogY:
        plt.semilogy(xX, yY, '-bD', markevery=markPoints)
    elif logLog:
        plt.loglog(xX, yY, '-bD', markevery=markPoints)
    else:
        plt.plot(xX, yY,'-bD', markevery=markPoints)
    plt.title(titleString)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    if saveName:
        plt.savefig(saveName, bbox_inches='tight')
    else:
```

```
        plt.show()
    plt.gcf().clear()

# Using the w=1 case....
def eE_truncation(x0, v0, h0, time, saveName):
    h0 = float(h0)
    HH = np.array([h0, h0/2, h0/4, h0/8, h0/16])
    truncError = np.zeros([5])
    for i in range(5):
        N = int(float(time) / HH[i])+1
        (TT, XX, VV) = explicitEulerErrors(x0, v0, HH[i], N)
        truncError[i] = max(XX)
    plotXY(HH, truncError, "Truncation error of explicit Euler", "h used", "maximum truncation error in " + st
    plt.gcf().clear()


# Problem 4 ======================================

def eE_energy(x0, v0, h, N, saveName):
    (TT, XX, VV) = explicitEuler(x0, v0, h, N)
    EE = XX*XX + VV*VV
    plotXY(TT, EE, "Energy x^2+v^2 of numerical explicit Euler", "time", " energy ", saveName=saveName)
    plt.gcf().clear()

# Problem 5 ======================================

# 5.1

# Using the w=1 case....
def implicitEuler(x0, v0, h, N):
    TT = np.arange(N+1)*h
    XX = np.zeros([N+1])
    VV = np.zeros([N+1])
    XX[0] = float(x0)
    VV[0] = float(v0)
    for i in range(1, N+1):
        XX[i] = (XX[i-1] + h*VV[i-1])/(1+h*h)
        VV[i] = (VV[i-1] - h*XX[i-1])/(1+h*h)
    return (TT, XX, VV)

def plotImplicitEuler(x0, v0, h, N, saveName):
    (TT, XX, VV) = implicitEuler(x0, v0, h, N)
    plotXVT(TT, XX, VV, 'X(t) of spring', 'V(t) of spring', saveName=saveName)

# 5.2

# Using the w=1 case....
def implicitEulerErrors(x0, v0, h, N):
    (TT, XX, VV) = implicitEuler(x0, v0, h, N)
    (Ta, Xa, Va) = analyticEuler(x0, v0, h, N)
    return (TT, XX - Xa, VV - Va)

def plotImplicitEulerErrors(x0, v0, h, N, saveName):
    (TT, XX, VV) = implicitEulerErrors(x0, v0, h, N)
    plotXVT(TT, XX, VV, 'Xnumeric(t)-Xanalytic(t) of spring', 'Vnumeric(t)-Vanalytic(t) of spring', saveName=s

# don't have to mimic 3 (yay!)

# 5.4
```

```
def iE_energy(x0, v0, h, N, saveName):
    (TT, XX, VV) = implicitEuler(x0, v0, h, N)
    EE = XX*XX + VV*VV
    plotXY(TT, EE, "Energy x^2+v^2 of numerical implicit Euler", "time", " energy ", saveName=saveName)



# PART 2 ==========================================================================



# Problem 1 ========================================

def eE_iE_phaseSpace(x0, v0, h, N, saveName1, saveName2):
    (eT, eX, eV) = explicitEuler(x0, v0, h, N)
    (iT, iX, iV) = implicitEuler(x0, v0, h, N)
    plotXY(eX, eV, "Explicit Euler in phase space-starting X/V marked", "X(t)", "V(t)", markPoints = [eX[0]],
    plotXY(iX, iV, "Implicit Euler in phase space-starting X/V marked", "X(t)", "V(t)", markPoints = [iX[0]],


# Problem 2 ========================================

def symplecticEuler(x0, v0, h, N):
    TT = np.arange(N+1)*h
    XX = np.zeros([N+1])
    VV = np.zeros([N+1])
    XX[0] = float(x0)
    VV[0] = float(v0)
    for i in range(1, N+1):
        XX[i] = XX[i-1] + h*VV[i-1]
        VV[i] = VV[i-1] - h*XX[i]
    return (TT, XX, VV)

def symp_phaseSpace(x0, v0, h, N, saveName):
    (sT, sX, sV) = symplecticEuler(x0, v0, h, N)
    plotXY(sX, sV, "Symplectic Euler in phase space-starting X/V marked", "X(t)", "V(t)", markPoints = [sX[0]]

# Problem 3 ========================================

def symp_energy(x0, v0, h, N, saveName):
    (TT, XX, VV) = symplecticEuler(x0, v0, h, N)
    EE = XX*XX + VV*VV
    plotXY(TT, EE, "Energy x^2+v^2 of numerical symplectic Euler", "time", " energy ", saveName=saveName)



# Problem 4 ========================================

def symp_lag(x0, v0, h, N, saveName):
    startN = int(float(3)/4*N)
    (TT, XX, VV) = symplecticEuler(x0, v0, h, N)
    (Ta, Xa, Va) = analyticEuler(x0, v0, h, N)
    f, axarr = plt.subplots(2, sharex=True)
    axarr[0].plot(TT[startN:], XX[startN:], 'b', Ta[startN:], Xa[startN:], 'r')
    axarr[0].set_title("X(t) versus time: blue symplectic Euler, red analytic solution.")
    axarr[1].plot(TT[startN:], VV[startN:], 'b', Ta[startN:], Va[startN:], 'r')
    axarr[1].set_title("V(t) versus time: blue symplectic Euler, red analytic solution.")
    plt.savefig(saveName, bbox_inches='tight')
    plt.gcf().clear()
```