Ph20 - Assignment 4

Yovan Badal

12/07/2017

This assignment and report has been generated by a Makefile using a single make command, as per the requirements of Assignment 4. The Makefile includes pre-requisites and dependencies, such that only the affected files are updated by the make comment when the dependencies are modified. We first reproduce the configuration file, Makefile, source code, and command-line output. Then we provide a reproduction of Assignment 3 with the corresponding plots, both generated by the Makefile.

Configuration file

```
# Main script for ODE assignment
ODE_SOLVER=euler.py
ODE_EXE=python $(ODE_SOLVER)
SETTINGS= 1 0 0.01 20
ERR_BEHAVIOR_SETTINGS= 0.01 4
PDF_MAKER=pdflatex
```

Makefile

```
include config.mk

# Generate plots with parameters set in config.mk.

PNG_FILES=$(wildcard *.png)

all : $(PNG_FILES) Assignment_4.pdf

Assignment_4.pdf : Assignment_4.tex $(PNG_FILES)
$(PDF_MAKER) $<

$(PNG_FILES) : $(ODE_SOLVER)  plot_instruction_set.txt \
err_instruction_set.txt
while read -r file; do \
$(ODE_EXE) "$$file" $(SETTINGS); \
done <plot_instruction_set.txt
while read -r errfile; do \
$(ODE_EXE) "$$errfile" $(ERR_BEHAVIOR_SETTINGS); \
done <err_instruction_set.txt
```

Source Code

```
import sys
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rcParams

rcParams.update({'figure.autolayout': True})

def euler_plot(x_0, v_0, h, t):
    N = int(t/h)
    x_arr = np.zeros(N + 1)
    v_arr = np.zeros(N + 1)
    t_arr = np.arange(N + 1, dtype=float)

    t_arr *= float(h)

    x_arr[0] = x_0
    v_arr[0] = v_0

    for i in range(len(t_arr) - 1):
        x_arr[i + 1] = float(x_arr[i]) + float(h)*float(v_arr[i])
        v_arr[i + 1] = float(v_arr[i]) - float(h)*float(x_arr[i])

    plt.figure(1)
    plt.subplot(211)
    plt.xlabel('t')
```

```
    plt.ylabel('x')
    plt.plot(t_arr, x_arr)

    plt.subplot(212)
    plt.xlabel('t')
    plt.ylabel('v')
    plt.plot(t_arr, v_arr)
    plt.savefig('euler_plot.png')
    plt.close()


def euler_err(x_0, v_0, h, t):
    N = int(t/h)
    x_arr = np.zeros(N + 1)
    v_arr = np.zeros(N + 1)
    t_arr = np.arange(N + 1, dtype=float)

    t_arr *= float(h)

    x_arr[0] = 1
    v_arr[0] = 0

    for i in range(len(t_arr) - 1):
        x_arr[i + 1] = float(x_arr[i]) + float(h)*float(v_arr[i])
        v_arr[i + 1] = float(v_arr[i]) - float(h)*float(x_arr[i])

    x_err = np.zeros(N + 1)
    v_err = np.zeros(N + 1)

    for i in range(len(t_arr)):
        x_err[i] = np.cos(t_arr[i]) - x_arr[i]
        v_err[i] = -np.sin(t_arr[i]) - v_arr[i]

    plt.figure(1)
    plt.subplot(211)
    plt.xlabel('t')
    plt.ylabel('$x_{analytic}(t_i) - x_i$')
    plt.plot(t_arr, x_err)

    plt.subplot(212)
    plt.xlabel('t')
    plt.ylabel('$v_{analytic}(t_i) - v_i$')
    plt.plot(t_arr, v_err)
    plt.savefig('euler_err.png')
    plt.close()


def maxabs(a, axis=None):
    """Return slice of a, keeping only those values that \
            are furthest away
    from 0 along axis"""
    maxa = np.amax(a)
    mina = np.amin(a)
```

```
    if abs(maxa) > abs(mina):
        out = maxa

    else:
        out = mina

    return out


def max_x_err(h):
    """Maximum x error when solving up to t=20"""
    t = 20
    N = int(t/h)
    x_arr = np.zeros(N + 1)
    v_arr = np.zeros(N + 1)
    t_arr = np.arange(N + 1, dtype=float)

    t_arr *= float(h)

    x_arr[0] = 1
    v_arr[0] = 0

    for i in range(len(t_arr) - 1):
        x_arr[i + 1] = float(x_arr[i]) + float(h)*float(v_arr[i])
        v_arr[i + 1] = float(v_arr[i]) - float(h)*float(x_arr[i])

    x_err = np.zeros(N + 1)

    for i in range(len(t_arr)):
        x_err[i] = np.cos(t_arr[i]) - x_arr[i]

    return maxabs(x_err)


def err_behavior(h0, k):
    "Plotting max x error for h0/2^j; j = 0,1,2,...,k \
            when solving up to t=20"""
    err_array = np.zeros(k+1)
    h_array = np.zeros(k+1)

    for i in range(k):
        err_array[i] = max_x_err(float(h0/(2**i)))
        h_array[i] = float(h0/(2**i))

    plt.xlabel('h')
    plt.ylabel('$max[x_{analytic}(t_i) - x_i]$')
    plt.plot(h_array, err_array)
    plt.savefig('err_behavior.png')
    plt.close()


def euler_energy(x_0, v_0, h, t):
```

```
    N = int(t/h)
    x_arr = np.zeros(N + 1)
    v_arr = np.zeros(N + 1)
    t_arr = np.arange(N + 1, dtype=float)

    t_arr *= float(h)

    x_arr[0] = x_0
    v_arr[0] = v_0

    for i in range(len(t_arr) - 1):
        x_arr[i + 1] = float(x_arr[i]) + float(h)*float(v_arr[i])
        v_arr[i + 1] = float(v_arr[i]) - float(h)*float(x_arr[i])

    energy_arr = x_arr**2 + v_arr**2

    plt.xlabel('t')
    plt.ylabel('E')
    plt.plot(t_arr, energy_arr)
    plt.savefig('euler_energy.png')
    plt.close()


def euler_implicit_plot(x_0, v_0, h, t):
    N = int(t/h)
    x_arr = np.zeros(N + 1)
    v_arr = np.zeros(N + 1)
    t_arr = np.arange(N + 1, dtype=float)

    t_arr *= float(h)

    x_arr[0] = x_0
    v_arr[0] = v_0

    for i in range(len(t_arr) - 1):
        x_arr[i + 1] = float(1/(h**2 + 1))*(float(x_arr[i]) \
                + float(h)*float(v_arr[i]))
        v_arr[i + 1] = float(1/(h**2 + 1))*(float(v_arr[i]) \
                - float(h)*float(x_arr[i]))

    plt.figure(1)
    plt.subplot(211)
    plt.xlabel('t')
    plt.ylabel('x')
    plt.plot(t_arr, x_arr)

    plt.subplot(212)
    plt.xlabel('t')
    plt.ylabel('v')
    plt.plot(t_arr, v_arr)
    plt.savefig('euler_implicit_plot.png')
    plt.close()
```

```
def euler_implicit_err(x0, v0, h, t):
    N = int(t/h)
    x_arr = np.zeros(N + 1)
    v_arr = np.zeros(N + 1)
    t_arr = np.arange(N + 1, dtype=float)

    t_arr *= float(h)

    x_arr[0] = 1
    v_arr[0] = 0

    for i in range(len(t_arr) - 1):
        x_arr[i + 1] = float(1/(h**2 + 1))*(float(x_arr[i]) \
                + float(h)*float(v_arr[i]))
        v_arr[i + 1] = float(1/(h**2 + 1))*(float(v_arr[i]) \
                - float(h)*float(x_arr[i]))

    x_err = np.zeros(N + 1)
    v_err = np.zeros(N + 1)

    for i in range(len(t_arr)):
        x_err[i] = np.cos(t_arr[i]) - x_arr[i]
        v_err[i] = -np.sin(t_arr[i]) - v_arr[i]

    plt.figure(1)
    plt.subplot(211)
    plt.xlabel('t')
    plt.ylabel('$x_{analytic}(t_i) - x_i$')
    plt.plot(t_arr, x_err)

    plt.subplot(212)
    plt.xlabel('t')
    plt.ylabel('$v_{analytic}(t_i) - v_i$')
    plt.plot(t_arr, v_err)
    plt.savefig('euler_implicit_err.png')
    plt.close()


def implicit_max_x_err(h):
    """Maximum x error when solving up to t=20"""
    t = 20
    N = int(t/h)
    x_arr = np.zeros(N + 1)
    v_arr = np.zeros(N + 1)
    t_arr = np.arange(N + 1, dtype=float)

    t_arr *= float(h)

    x_arr[0] = 1
    v_arr[0] = 0

    for i in range(len(t_arr) - 1):
```

```
        x_arr[i + 1] = float(1/(h**2 + 1))*(float(x_arr[i]) \
                + float(h)*float(v_arr[i]))
        v_arr[i + 1] = float(1/(h**2 + 1))*(float(v_arr[i]) \
                - float(h)*float(x_arr[i]))

    x_err = np.zeros(N + 1)

    for i in range(len(t_arr)):
        x_err[i] = np.cos(t_arr[i]) - x_arr[i]

    return maxabs(x_err)


def implicit_err_behavior(h0, k):
    "Plotting max x error for h0/2^j; j = 0,1,2,...,k \
            when solving up to t=20"""
    err_array = np.zeros(k+1)
    h_array = np.zeros(k+1)

    for i in range(k):
        err_array[i] = implicit_max_x_err(float(h0/(2**i)))
        h_array[i] = float(h0/(2**i))

    plt.xlabel('h')
    plt.ylabel('$max[x_{analytic}(t_i) - x_i]$')
    plt.plot(h_array, err_array)
    plt.savefig('implicit_err_behavior.png')
    plt.close()


def implicit_euler_energy(x_0, v_0, h, t):
    N = int(t/h)
    x_arr = np.zeros(N + 1)
    v_arr = np.zeros(N + 1)
    t_arr = np.arange(N + 1, dtype=float)

    t_arr *= float(h)

    x_arr[0] = x_0
    v_arr[0] = v_0

    for i in range(len(t_arr) - 1):
        x_arr[i + 1] = float(1/(h**2 + 1))*(float(x_arr[i]) \
                + float(h)*float(v_arr[i]))
        v_arr[i + 1] = float(1/(h**2 + 1))*(float(v_arr[i]) \
                - float(h)*float(x_arr[i]))

    energy_arr = x_arr**2 + v_arr**2

    plt.xlabel('t')
    plt.ylabel('E')
    plt.plot(t_arr, energy_arr)
    plt.savefig('implicit_euler_energy.png')
```

```
    plt.close()


def euler_phase(x_0, v_0, h, t):
    N = int(t/h)
    x_arr = np.zeros(N + 1)
    v_arr = np.zeros(N + 1)

    x_arr[0] = x_0
    v_arr[0] = v_0

    for i in range(len(x_arr) - 1):
        x_arr[i + 1] = float(x_arr[i]) + float(h)*float(v_arr[i])
        v_arr[i + 1] = float(v_arr[i]) - float(h)*float(x_arr[i])

    plt.xlabel('x')
    plt.ylabel('v')
    plt.plot(x_arr, v_arr)
    plt.axes().set_aspect('equal')
    plt.savefig('euler_phase.png')
    plt.close()


def implicit_euler_phase(x_0, v_0, h, t):
    N = int(t/h)
    x_arr = np.zeros(N + 1)
    v_arr = np.zeros(N + 1)

    x_arr[0] = x_0
    v_arr[0] = v_0

    for i in range(len(x_arr) - 1):
        x_arr[i + 1] = float(1/(h**2 + 1))*(float(x_arr[i]) \
                + float(h)*float(v_arr[i]))
        v_arr[i + 1] = float(1/(h**2 + 1))*(float(v_arr[i]) \
                - float(h)*float(x_arr[i]))

    plt.xlabel('x')
    plt.ylabel('v')
    plt.plot(x_arr, v_arr)
    plt.axes().set_aspect('equal')
    plt.savefig('implicit_euler_phase.png')
    plt.close()


def symplectic_euler_phase(x_0, v_0, h, t):
    N = int(t/h)
    x_arr = np.zeros(N + 1)
    v_arr = np.zeros(N + 1)

    x_arr[0] = x_0
    v_arr[0] = v_0
```

```
    for i in range(len(x_arr) - 1):
        x_arr[i + 1] = float(x_arr[i]) + float(h)*float(v_arr[i])
        v_arr[i + 1] = float(v_arr[i]) - float(h)*float(x_arr[i+1])

    plt.xlabel('x')
    plt.ylabel('v')
    plt.plot(x_arr, v_arr)
    plt.axes().set_aspect('equal')
    plt.savefig('symplectic_euler_phase.png')
    plt.close()


def analytic_phase(x_0, v_0, h, t):
    N = int(t/h)
    t_arr = np.arange(N + 1, dtype=float)

    t_arr *= float(h)

    x_arr = np.cos(t_arr)
    v_arr = -np.sin(t_arr)

    plt.xlabel('x')
    plt.ylabel('v')
    plt.plot(x_arr, v_arr)
    plt.axes().set_aspect('equal')
    plt.savefig('analytic_phase.png')
    plt.close()


def symplectic_euler_energy(x_0, v_0, h, t):
    N = int(t/h)
    x_arr = np.zeros(N + 1)
    v_arr = np.zeros(N + 1)
    t_arr = np.arange(N + 1, dtype=float)

    x_arr[0] = x_0
    v_arr[0] = v_0
    t_arr *= float(h)

    for i in range(len(x_arr) - 1):
        x_arr[i + 1] = float(x_arr[i]) + float(h)*float(v_arr[i])
        v_arr[i + 1] = float(v_arr[i]) - float(h)*float(x_arr[i+1])

    energy_arr = x_arr**2 + v_arr**2

    plt.xlabel('t')
    plt.ylabel('E')
    plt.plot(t_arr, energy_arr)
    plt.savefig('symplectic_euler_energy.png')
    plt.close()


def main():
```

```
    if not (sys.argv[1] == 'err_behavior' or \
            sys.argv[1] == 'implicit_err_behavior'):
        eval('%(a)s(%(b)s,%(c)s,%(d)s,%(e)s)' % {"a": sys.argv[1], \
                "b": sys.argv[2], "c": sys.argv[3], "d": sys.argv[4], \
                "e": sys.argv[5]})
    else:
        eval('%(a)s(%(b)s,%(c)s)' % {"a": sys.argv[1], \
                "b": sys.argv[2], "c": sys.argv[3]})


main()
```

Command-line Output

```
pdflatex Assignment_4.tex
This is pdfTeX, Version 3.14159265-2.6-1.40.17 (TeX Live 2016) (preloaded format=pdflatex)
 restricted \write18 enabled.
entering extended mode
(./Assignment_4.tex
LaTeX2e <2016/03/31>
Babel <3.9r> and hyphenation patterns for 3 language(s) loaded.
(/usr/share/texlive/texmf-dist/tex/latex/base/article.cls
Document Class: article 2014/09/29 v1.4h Standard LaTeX document class
(/usr/share/texlive/texmf-dist/tex/latex/base/size11.clo))
(/usr/share/texlive/texmf-dist/tex/latex/amsmath/amsmath.sty
For additional information on amsmath, use the ‘?’ option.
(/usr/share/texlive/texmf-dist/tex/latex/amsmath/amstext.sty
(/usr/share/texlive/texmf-dist/tex/latex/amsmath/amsgen.sty))
(/usr/share/texlive/texmf-dist/tex/latex/amsmath/amsbsy.sty)
(/usr/share/texlive/texmf-dist/tex/latex/amsmath/amsopn.sty))
(/usr/share/texlive/texmf-dist/tex/latex/amsfonts/amssymb.sty
(/usr/share/texlive/texmf-dist/tex/latex/amsfonts/amsfonts.sty))
(/usr/share/texlive/texmf-dist/tex/latex/amscls/amsthm.sty)
(/usr/share/texlive/texmf-dist/tex/latex/physics/physics.sty
(/usr/share/texlive/texmf-dist/tex/latex/l3packages/xparse/xparse.sty
(/usr/share/texlive/texmf-dist/tex/latex/l3kernel/expl3.sty
(/usr/share/texlive/texmf-dist/tex/latex/l3kernel/expl3-code.tex)
(/usr/share/texlive/texmf-dist/tex/latex/l3kernel/l3pdfmode.def))))
(/usr/share/texlive/texmf-dist/tex/latex/graphics/graphicx.sty
(/usr/share/texlive/texmf-dist/tex/latex/graphics/keyval.sty)
(/usr/share/texlive/texmf-dist/tex/latex/graphics/graphics.sty
(/usr/share/texlive/texmf-dist/tex/latex/graphics/trig.sty)
(/usr/share/texlive/texmf-dist/tex/latex/graphics-cfg/graphics.cfg)
(/usr/share/texlive/texmf-dist/tex/latex/graphics-def/pdftex.def
(/usr/share/texlive/texmf-dist/tex/generic/oberdiek/infwarerr.sty)
(/usr/share/texlive/texmf-dist/tex/generic/oberdiek/ltxcmds.sty))))
(/usr/share/texlive/texmf-dist/tex/latex/titling/titling.sty)
(/usr/share/texlive/texmf-dist/tex/latex/tools/verbatim.sty)
(/usr/share/texlive/texmf-dist/tex/latex/geometry/geometry.sty
(/usr/share/texlive/texmf-dist/tex/generic/oberdiek/ifpdf.sty)
(/usr/share/texlive/texmf-dist/tex/generic/oberdiek/ifvtex.sty)
```

```
(/usr/share/texlive/texmf-dist/tex/generic/ifxetex/ifxetex.sty))
(./Assignment_4.aux)
(/usr/share/texlive/texmf-dist/tex/context/base/mkii/supp-pdf.mkii
[Loading MPS to PDF converter (version 2006.09.02).]
) (/usr/share/texlive/texmf-dist/tex/generic/oberdiek/pdftexcmds.sty
(/usr/share/texlive/texmf-dist/tex/generic/oberdiek/ifluatex.sty))
(/usr/share/texlive/texmf-dist/tex/latex/oberdiek/epstopdf-base.sty
(/usr/share/texlive/texmf-dist/tex/latex/oberdiek/grfext.sty
(/usr/share/texlive/texmf-dist/tex/generic/oberdiek/kvdefinekeys.sty))
(/usr/share/texlive/texmf-dist/tex/latex/oberdiek/kvoptions.sty
(/usr/share/texlive/texmf-dist/tex/generic/oberdiek/kvsetkeys.sty
(/usr/share/texlive/texmf-dist/tex/generic/oberdiek/etexcmds.sty)))
(/usr/share/texlive/texmf-dist/tex/latex/latexconfig/epstopdf-sys.cfg))
*geometry* driver: auto-detecting
*geometry* detected driver: pdftex
(/usr/share/texlive/texmf-dist/tex/latex/amsfonts/umsa.fd)
(/usr/share/texlive/texmf-dist/tex/latex/amsfonts/umsb.fd) [1{/usr/share/texliv
e/texmf-dist/fonts/map/pdftex/updmap/pdftex.map}] [2] [3] [4] [5] [6] [7]
[8] [9] [10] <euler_plot.png, id=40, 462.528pt x 346.896pt>
<use euler_plot.png> [11 <./euler_plot.png>]
<euler_err.png, id=47, 462.528pt x 346.896pt> <use euler_err.png> [12 <./euler_
err.png>] <err_behavior.png, id=54, 462.528pt x 346.896pt>
<use err_behavior.png> [13 <./err_behavior.png>]
<euler_energy.png, id=60, 462.528pt x 346.896pt> <use euler_energy.png>
[14 <./euler_energy.png>]
<implicit_err_behavior.png, id=67, 462.528pt x 346.896pt>
<use implicit_err_behavior.png> [15 <./implicit_err_behavior.png>]
<implicit_euler_energy.png, id=72, 462.528pt x 346.896pt>
<use implicit_euler_energy.png> [16 <./implicit_euler_energy.png>]
<analytic_phase.png, id=77, 462.528pt x 346.896pt> <use analytic_phase.png>
[17 <./analytic_phase.png>] <euler_phase.png, id=82, 462.528pt x 346.896pt>
<use euler_phase.png> [18 <./euler_phase.png>]
<implicit_euler_phase.png, id=87, 462.528pt x 346.896pt>
<use implicit_euler_phase.png>
Underfull \hbox (badness 10000) in paragraph at lines 140--141

[19 <./implicit_euler_phase.png>]
<symplectic_euler_phase.png, id=93, 462.528pt x 346.896pt>
<use symplectic_euler_phase.png> [20 <./symplectic_euler_phase.png>]
Underfull \hbox (badness 10000) in paragraph at lines 153--154

<symplectic_euler_energy.png, id=98, 462.528pt x 346.896pt>
<use symplectic_euler_energy.png> [21 <./symplectic_euler_energy.png>]
(./Assignment_4.aux)

LaTeX Warning: Label(s) may have changed. Rerun to get cross-references right.

 )
(see the transcript file for additional information)</usr/share/texlive/texmf-d
ist/fonts/type1/public/amsfonts/cm/cmbx12.pfb></usr/share/texlive/texmf-dist/fo
nts/type1/public/amsfonts/cm/cmmi10.pfb></usr/share/texlive/texmf-dist/fonts/ty
pe1/public/amsfonts/cm/cmmi8.pfb></usr/share/texlive/texmf-dist/fonts/type1/pub
lic/amsfonts/cm/cmr10.pfb></usr/share/texlive/texmf-dist/fonts/type1/public/ams
```

```
fonts/cm/cmr12.pfb></usr/share/texlive/texmf-dist/fonts/type1/public/amsfonts/c
m/cmr17.pfb></usr/share/texlive/texmf-dist/fonts/type1/public/amsfonts/cm/cmr8.
pfb></usr/share/texlive/texmf-dist/fonts/type1/public/amsfonts/cm/cmsy10.pfb></
usr/share/texlive/texmf-dist/fonts/type1/public/amsfonts/cm/cmsy8.pfb></usr/sha
re/texlive/texmf-dist/fonts/type1/public/amsfonts/cm/cmti10.pfb></usr/share/tex
live/texmf-dist/fonts/type1/public/amsfonts/cm/cmtt10.pfb>
Output written on Assignment_4.pdf (21 pages, 409339 bytes).
Transcript written on Assignment_4.log.
```

We now reproduce Assignment 3.

Part 1

1. We implement the explicit Euler method, and use the script to plot x and v as functions of time, for initial conditions $x(0) = 1$, $v(0) = 0$, $h = 0.01$ and t running from 0 to 20.

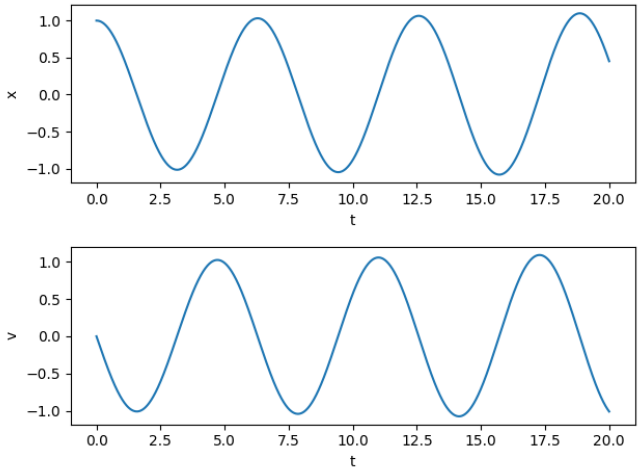

Figure 1: Plot of x and v as functions of time for initial conditions $x(0) = 1$, $v(0) = 0$, and stepsize $h = 0.01$.

2. We observe that the analytical solution to the simple harmonic oscillation equation $F = -kx$ for the intial conditions given is $x = \cos(t)$ and $v = -\sin(t)$.

We can compare our numerical solution to the analytic solution by plotting the global errors $x_{analytic}(t_i) - x_i$ and $v_{analytic}(t_i) - v_i$ against time.

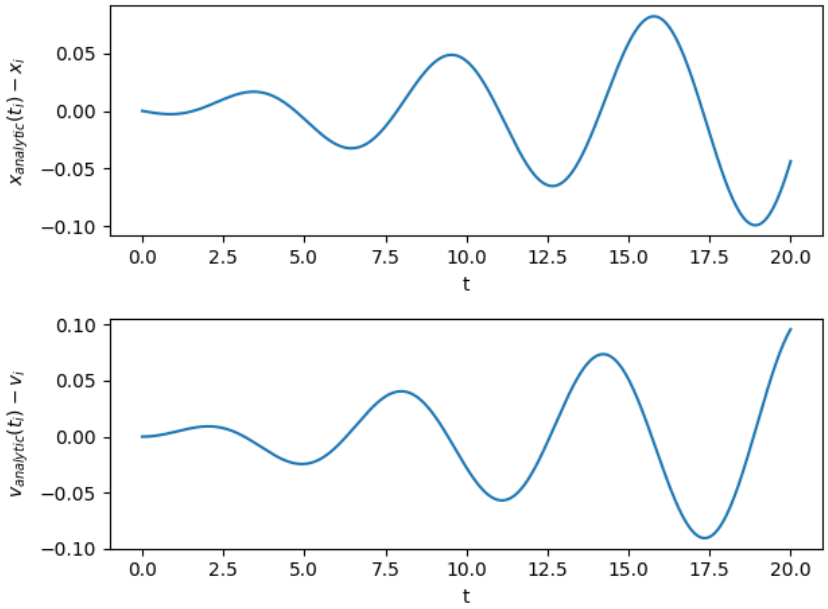

Figure 2: Plot of global errors against time for the explicit Euler method, with stepsize $h = 0.01$.

3. We now plot the maximum global error in x against h, integrating up to the same final time $t = 20$.

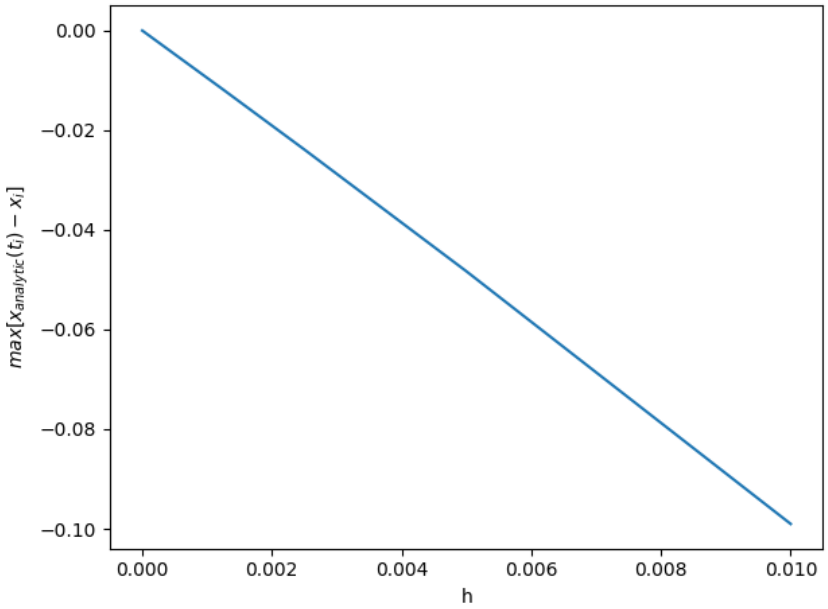

Figure 3: Plot of maximum global error in x against h for the explicit Euler method, integrating up to $t = 20$ for each h.

The plot indicates that the truncation error is proportional to h for reasonable h, where the global error decreases linearly with h.

4. We now plot the normalized total energy $E = v^2 + x^2$ as a function of time:

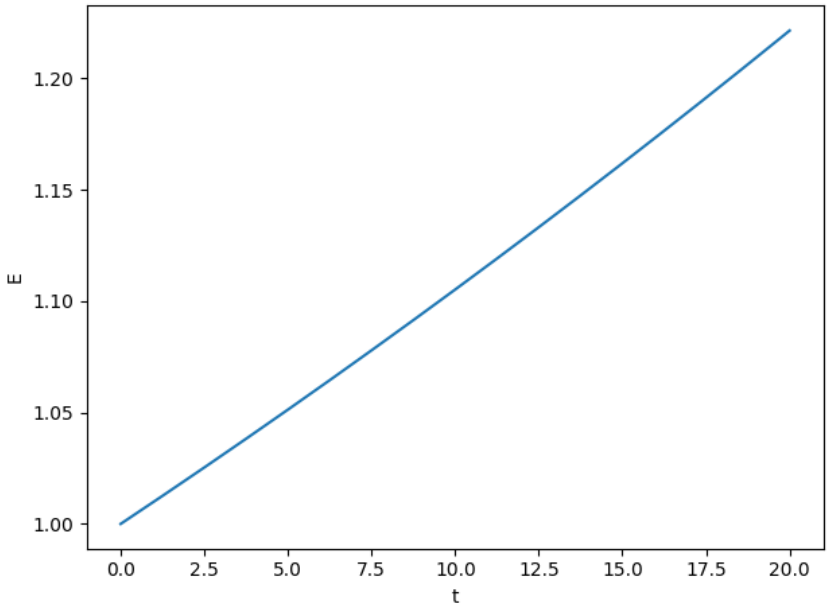

Figure 4: Normalized energy computed from the results of the explicit Euler method for the same initial conditions and with $h = 0.01$ as before.

We observe that there is a linear increase with time of the normalized energy, similar to the behavior of the global error in with respect to h.

5. We implement the implicit Euler method using the recurrence relations:

$$x_i = x_{i-1} + hv_{i-1}$$
$$v_i = -x_{i-1} + v_{i-1}$$

We then plot the global error behavior and the normalized energy as we did for our implementation of the explicit Euler method, making sure to use the same $h = 0.01$ for the normalized energy plot to draw a fair comparison.

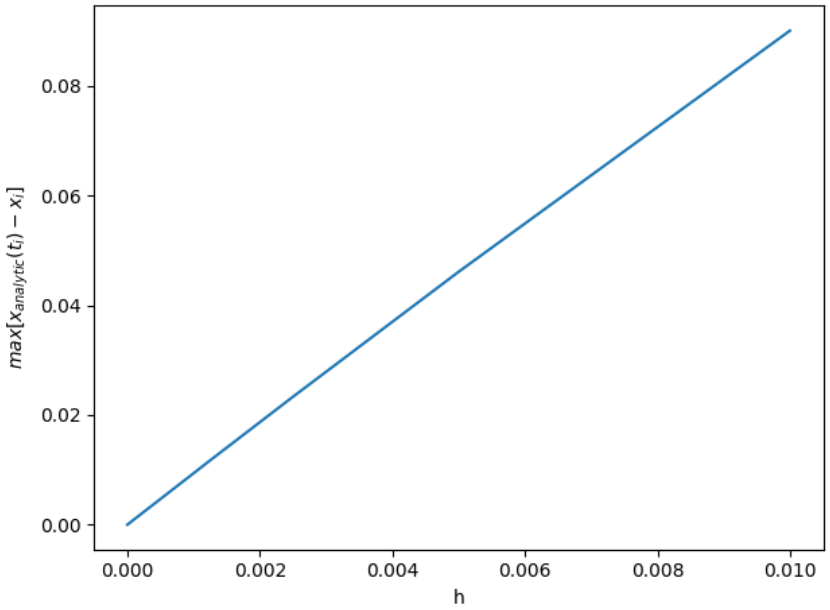

Figure 5: Plot of maximum global error in x against h for the implicit Euler method, integrating up to $t = 20$ for each h.

We observe that the global error for the implicit Euler method increases linearly with h.

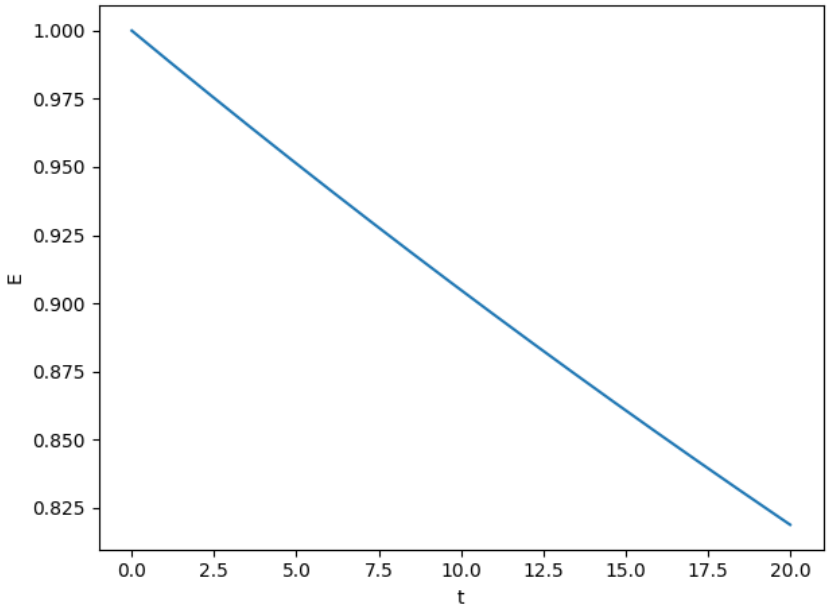

Figure 6: Normalized energy computed from the results of the implicit Euler method for the same initial conditions and with $h = 0.01$ as before.

We observe that the normalized energy for the implicit Euler method decreases linearly in time, with gradient negative that for the explicit Euler method.

Part 2

1. We plot our solutions in phase space (plotting v againsts x) setting $h = 0.01$ and integrating up to $t = 20$.

First, we do this for the analytical solution for the system:

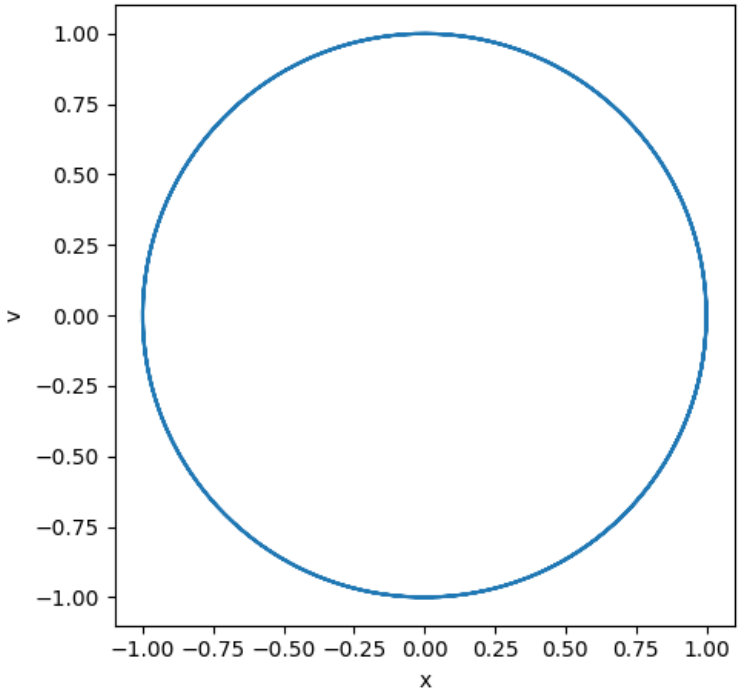

Figure 7: Plotting the analytic solution in phase space, up to $t = 20$.

This solution is exactly what we would expect, since $E = x^2 + v^2$ is conserved.

We then plot the numerical solution obtained using the explicit Euler method:

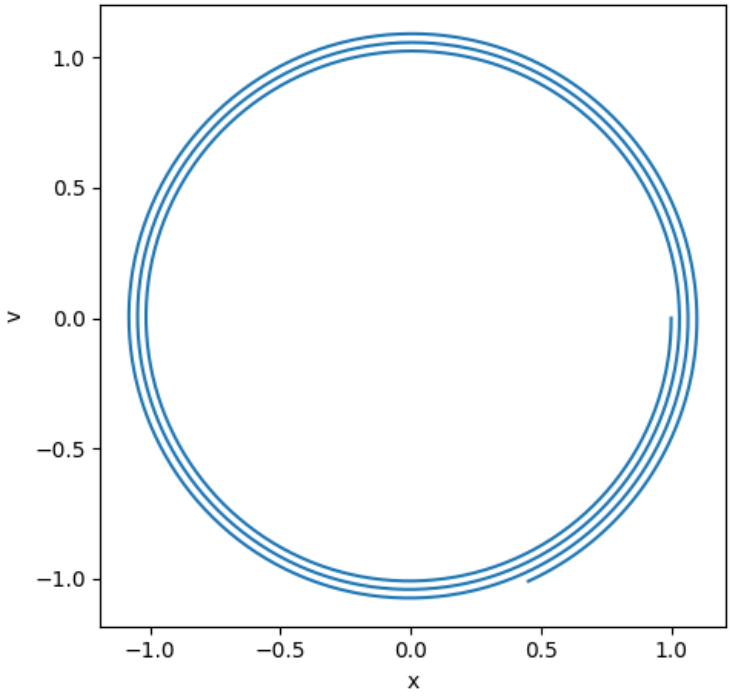

Figure 8: Plotting the numerical solution obtained using the explicit Euler method, integrating up to $t = 20$.

This is what we expect for this method, since errors are induced such that E increases linearly in time, i.e. trajectories in phase space spiral outwards.

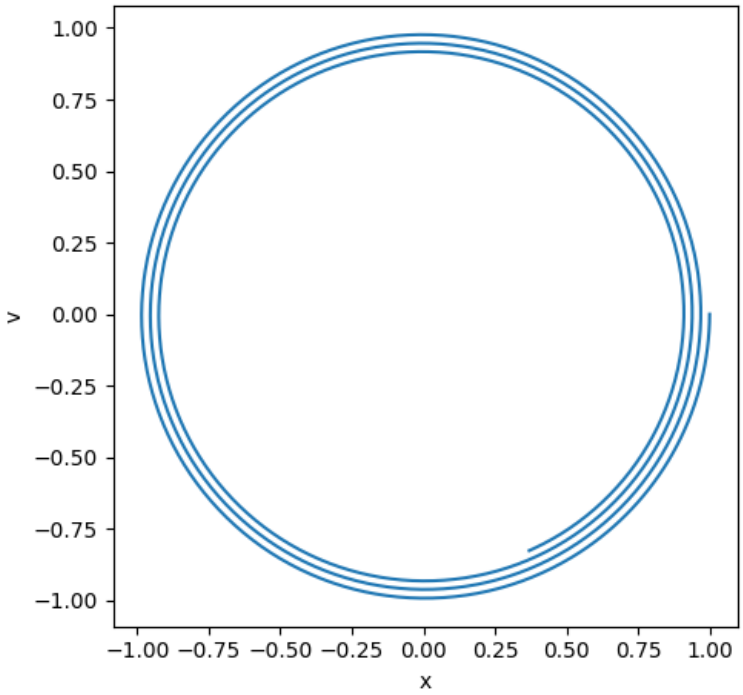

Figure 9: Plotting the numerical solution obtained using the implicit Euler method, integrating up to $t = 20$.

This is what we expect for this method, since errors are induced such that E increases linearly in time, i.e. trajectories in phase space spiral inwards.

2. We implement the symplectic Euler method, and again plot the numerical solution obained in phase space, setting $h = 0.01$ and integrating up to $t = 20$.

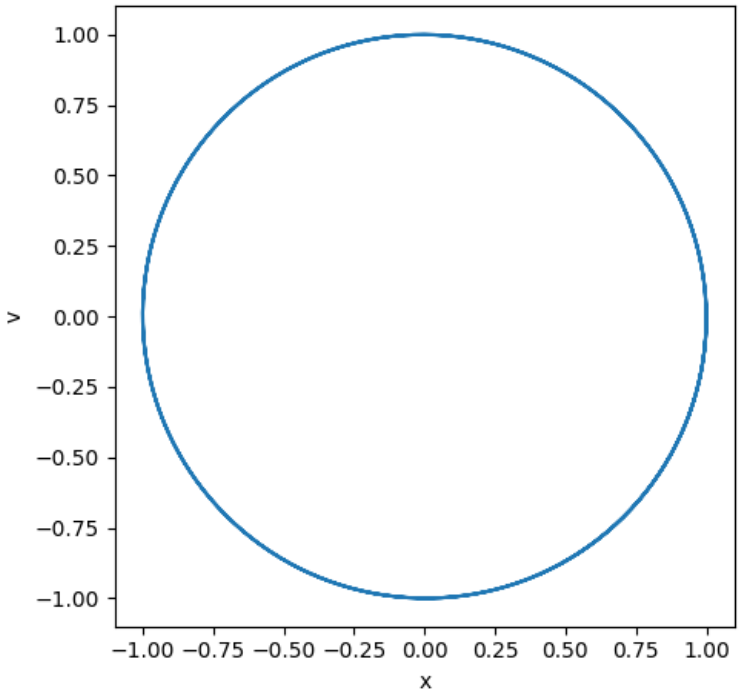

Figure 10: Plotting the numerical solution obtained using the symplectic Euler method, integrating up to $t = 20$.

This shows the behavior we expect from a symplectic method in that it conserves E, i.e. the curve in phase space is closed and bounded.

3. We now plot the evolution of $E = x^2 + v^2$ with time, using the symplectic Euler method, setting $h = 0.01$ and integrating to $t = 20$.

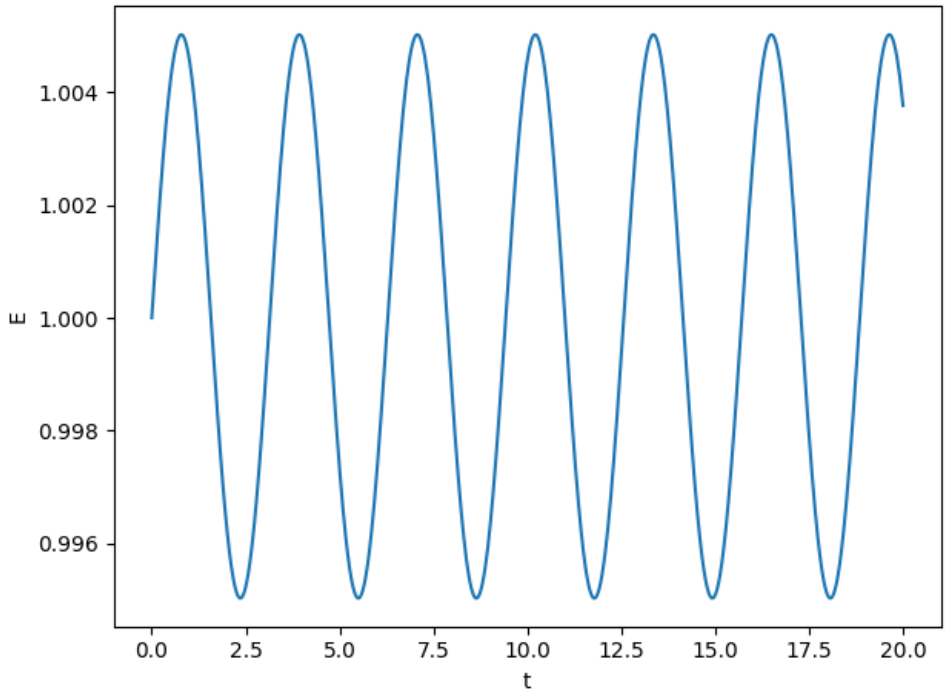

Figure 11: Normalized energy computed from the results of the symplectic Euler method for the same initial conditions and with $h = 0.01$ as for all parts above, integrating up to $t = 20$.

We first note that the analytic solution for E with respect to time is obviously the line $E = 1$ since E is conserved. We can therefore observe that the deviations from $E = 1$ time-average to zero, as we expect from a symplectic solution. This is exactly what we saw in phase space: there are slight deviations from a perfect circle but the resulting curve is nevertheless closed and encloses the same area as the analytic phase space curve, indicating that deviation time-average to zero.