The Investigation of the Effects of a Higher Order Derivative on Newton's Laws of Motion Using the Pais-Uhlenbeck Oscillator

Jakob Broman
jakob.broman@hotmail.com

under the direction of
Dhrubaditya Mitra
and Bidya Binay Karak
at
Astrophysics and Astrobiology
Nordic Institute for Theoretical Physics

Research Academy for Young Scientists
July 10, 2014

Abstract

Newton's laws of motion can describe an oscillator using the second derivative of position. Further on, it has been suggested in quantum gravity that including a higher order derivative would improve the approximation of an oscillator. Therefore this study aims to investigate the effects of implementing a fourth order derivative in the equations describing oscillators. In the study, differential equations describing harmonic oscillators and the Pais-Uhlenbeck oscillator were solved numerically using the Runge-Kutta fourth order method. The solutions to these were then plotted as lissajous figures and compared. A significant difference between the solutions was obtained, especially for longer periods of time. Whether this is a result of Newton's laws or the higher order derivative theory being somewhat incorrect is impossible to determine without further studies using experimental methods.

Contents

1 Introduction

In Newton's law, $F = ma$, a second order derivative of position is used to define the acceleration. However it is suggested that implementing a higher order derivative (HOD) in the law would improve it. Therefore this paper aims to investigate how a HOD affects Newton's laws of motion.

It is suggested that including a HOD, in this case a fourth order, could answer many questions in quantum gravity. However implementing it in the theory has proven to be problematic. A theory of HOD at a quantum mechanical level contains negative energies which makes it unstable at both classical and quantum mechanical level [1]. Although mentioned, knowledge of negative energies are not necessary for the understanding of this study.

Gravity at a cosmological level is described by Einstein's equations for the gravitational field and general relativity. As described by Stelle, K. S. [2], these theories would also improve if a HOD was involved. The inclusion of a fourth order derivative might also make Newton's laws of motion even more accurate.

Although it might be applicable in other fields, this study will mainly focus on the effects of the implementation of a HOD on Newton's laws of motion, especially for their simplicity and their common use. It is also known that if Newton's laws are somewhat wrong it would probably be with a tiny amount. Otherwise it would be possible to see these faults in our everyday life, because these laws are applicable to almost everything in the world. For example they are used to describe the motion of a car, a falling object and a pendulum.

Why the implementation of a HOD would get the effects mentioned above is a complicated mathematical and physical problem, and it would not be suitable to reflect upon it in this paper. Nevertheless it is still interesting to investigate how it would actually affect the solution of the equations. The models being used to see the effects on is the harmonic oscillator and the Pais-Uhlenbeck (PU) oscillator.

1.1 Harmonic Oscillators

A harmonic oscillator is a system that undergoes repetitive harmonic motion. An example for this is a pendulum. When the object is displaced from its equilibrium it experiences a restoring force according to $F = -kx$, where k is a spring constant and x is the displacement of the object depending on time. The motion of a harmonic oscillator is sinusoidal. Combining Newton's second law of motion, $F = ma = m\frac{\mathrm{d}^2x}{\mathrm{d}t^2}$, with the formula above, $-kx = m\ddot{x}$ is obtained, in which m is the mass of the object and $\ddot{x} = \frac{\mathrm{d}^2x}{\mathrm{d}t^2}$. Rewritten and with a different notation for coefficients the equation for a harmonic oscillator looks like the following,

$$\ddot{x} + \omega^2 x = 0, \tag{1}$$

where ω is the angular frequency of the harmonic motion. This equation was first derived for oscillators at a classical mechanic level.

1.2 The Pais-Uhlenbeck Oscillator

The PU oscillator is an oscillator which describes harmonic motion on a quantum mechanical level and was first mentioned by A. Pais and G. E. Uhlenbeck in [3]. It is the simplest version of a model for a high-derivative theory [4]. The Pais-Uhlenbeck oscillator can be described as the following,

$$x^{(4)} + \eta\ddot{x} + \varphi x = 0, \tag{2}$$

where η and φ are two real constants, determined by a number of factors, and $x^{(n)} = \frac{\mathrm{d}^n x}{\mathrm{d}t^n}$. The PU oscillator combines two quantum harmonic oscillators, and can therefore be seen as a system of two coupled oscillators. The system of the two coupled quantum harmonic oscillators is shown below

$$\begin{cases} \ddot{x} + \mu_1 x - \rho_1 y = 0 \\ \ddot{y} + \mu_2 y - \rho_2 x = 0 \end{cases} . \tag{3}$$

In Equation 3, x and y are the time-dependent positions of a particle, whereas μ_1, μ_2, ρ_1 and ρ_2 are real constants just like the ones used in Equation 2.

1.3 The Concept and Solving of Differential Equations

A differential equation is defined as an equation which includes terms of derivatives. The order of a differential equation is determined by the highest derivative involved. For example if a differential equation has a term of the second derivative, and no higher term, it is a second order differential equation. The solution of a differential equation is a function, for example the differential equation $\ddot{x} + \omega^2 x = 0$ has the solution

$$x(t) = c_1 \sin(t\omega) + c_2 \cos(t\omega),$$

where c_1 and c_2 are two constants determined by initial values of the differential equation.

Differential equations are most often difficult to solve, some of them are not even possible to solve using analytic methods. Instead one may have to use a computer algorithm to solve them numerically. A simple method for solving first order differential equations is the Euler method. This however, is not very accurate, and therefore uncommonly used by scientists and mathematicians. Another method for solving differential equations is the Runge-Kutta method, which is often used and considered as an adequate method for solving ordinary differential equations [5].

2 Method

2.1 Solving Differential Equations using the Runge-Kutta Method

The Runge-Kutta fourth order method was used to approximate the solution of the differential equations. The python code can be found in Appendix A. A simplification of a general numerical method can be described as follows. Starting from an initial value given by the equations and the slope of the line at that point the computer can calculate the next point. When the next point then is calculated, the slope at that point is approximated and therefore also the next point coming afterwards can be calculated. This is then repeated for as long as needed. The Runge-Kutta fourth order method is significantly better since it uses the initial values to calculate four different slopes, and then use a weighted average to compute the next point [5].

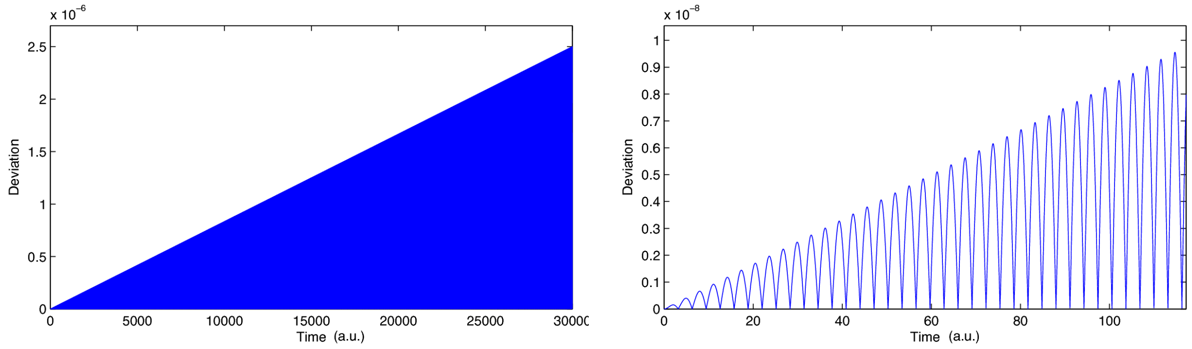
The distance between two points calculated is called the step-size. If a large step-size is used then the result will be poorly approximated. If a small step-size is used instead the solution will be much more accurate. However using a small step-size will take a long time to compute and hence it is important to find a step-size that is accurate enough without taking too long to compute.

In this study the step-size was determined using a simple reference differential equation, similar to the ones in subsection 1.1. The solution of this was then approximated using different step-sizes to determine which would be the most suitable. The largest deviations compared to the analytical solution to the differential equation after $t = 100$ a.u.[1] can be seen in table 1. It is crucial to notice that the deviation increased over time, and therefore t was increased until $t = 30000$ a.u., the longest time used in this study, to see what the largest deviation in the study will be. How the deviation changes over time can be seen in Fig. 1(a). In Fig. 1(b) a zoom in of this is shown, in which it is possible to see that the deviation follows an oscialltion. The largest deviation using a step-size of 0.01 should thus be $2.5 \cdot 10^{-6}$. Since the largest deviation for the step-size of 0.01 was very small even for longer times, this was chosen as the step-size used in the study.

Table 1: The largest deviation for variable step-sizes after $t = 100$ a.u.

Step-size	Largest deviation
1	0.5950
0.1	$8.248 \cdot 10^{-5}$
0.01	$8.248 \cdot 10^{-9}$
0.001	$1.305 \cdot 10^{-12}$

[1] a.u. – arbitrary unit. All units are arbitrary if nothing else is mentioned.



(a) Change in deviation until $t = 30000$ a.u.

(b) Fig. 1(a) in scale 1:250, showing how the deviation follows an oscillation over time.

Figure 1: Showing how the deviation changes over time using a step-size of 0.01.

2.2 Investigating the Differences between a Harmonic Oscillator and the PU Oscillator

As seen in the introduction the equations describing harmonic osillators are similar to the ones describing the PU oscillator, especially when the latter one is written as a system of two differential equations. The system of differential equations that was looked upon in this study is defined below

$$\begin{cases} \ddot{x} + \omega_1^2 x - \epsilon y = 0 \\ \ddot{y} + \omega_2^2 y - \epsilon x = 0 \end{cases}, \tag{4}$$

where ω_1 and ω_2 are angular frequencies and not necessarily the same, while ϵ is a small real constant. If $\epsilon = 0$ the system above is simply a system of two harmonic oscillators, while if $\epsilon \neq 0$ the system will describe a PU oscillator.

The Runge-Kutta fourth order method can only solve differential equations of the first order, thus it was necessary to break Equation 4 into four new first order differential equations. The result obtained was the following

$$\begin{cases} \dot{v}_x + \omega_1^2 x - \epsilon y = 0 \\ v_x = \frac{\mathrm{d}x}{\mathrm{d}t} \\ \dot{v}_y + \omega_1^2 y - \epsilon x = 0 \\ v_y = \frac{\mathrm{d}y}{\mathrm{d}t} \end{cases} , \tag{5}$$

in which $\dot{v}_x = \frac{\mathrm{d}v_x}{\mathrm{d}t}$ and $\dot{v}_y = \frac{\mathrm{d}v_y}{\mathrm{d}t}$.

To be able to compare the harmonic oscillator with the PU oscillator, variable values of ϵ has been used while the remaining values have been the same, to see how the solution changed. A small value of ϵ was considered the most appropriate for this study since it would impact the solution less than a larger one. The reason why this was considered as a good adoption is that Newton's laws are close to correct and hence if there is an error it should be small. Depending on how the solution using a small ϵ differed from the one using $\epsilon = 0$, the time was set to an amount which would allow the difference to be identified quite obviously. In all cases the smaller the ϵ, the longer the time.

Initial values $x(0)$, $\dot{x}(0)$, $y(0)$ and $\dot{y}(0)$ were then chosen randomly, as well as the angular frequencies ω_1 and ω_2. Then the system of differential equations (5) was solved using different values of $x(0)$, $\dot{x}(0)$, $y(0)$, $\dot{y}(0)$, ω_1 and ω_2, while having $\epsilon = 0$. These solutions were then plotted as lissajous figures[2]. The most interesting once were chosen to further on be solved using $\epsilon \neq 0$. Interesting means in this case that they had simple lissajous figures and the pattern of them was easy to identify and hence it was also easy to see any changes. The initial values found to be most suitable for this study can be seen in table 2. The solutions were then approximated again, using $\epsilon \neq 0$ and varying

[2] A lissajous figure plots two functions of time in one plot, with $x(t)$ at the x-axis and $y(t)$ at the y-axis. In this study both $x(t)$ and $y(t)$ are time-dependent positions. Most often when using lissajous figures, $x(t) = A\sin(at + \delta)$ and $y(t) = B\sin(bt)$. The ratio $a : b$ will determine the appearance of the figure. If the $a : b$ ratio is rational a closed lissajous figure is obtained, which will not happen when the ratio is irrational. A closed lissajous figure is one which follows the same pattern over and over again, while one which is not closed will continue to change In this study the $a : b$ ratio corresponds to the frequency ratio $\omega_1 : \omega_2$.

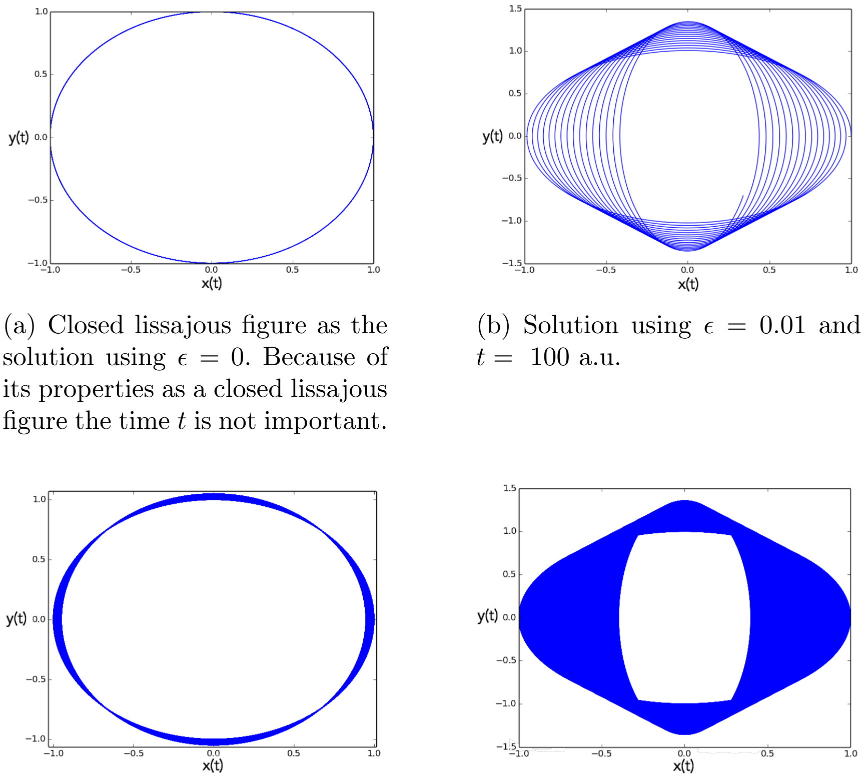
Table 2: The input values for Equation 5 used in this study.

	$x(0)$	$\dot{x}(0)$	$y(0)$	$\dot{y}(0)$	ω_1	ω_2
Lissajous figure 1:	1	0	0	1	1	1
Lissajous figure 2:	1	0	0	1	1	$\sqrt{2}$
Lissajous figure 3:	1	0	0	1	1	3

the time. All solutions were then plotted as lissajous figures and compared to the ones in which $\epsilon = 0$.

3 Results

The solutions for Equation 5 using $\omega_1 = \omega_2 = 1$, can be found in Fig. 2. When using $\epsilon = 0$



(a) Closed lissajous figure as the solution using $\epsilon = 0$. Because of its properties as a closed lissajous figure the time t is not important.

(b) Solution using $\epsilon = 0.01$ and $t =$ 100 a.u.

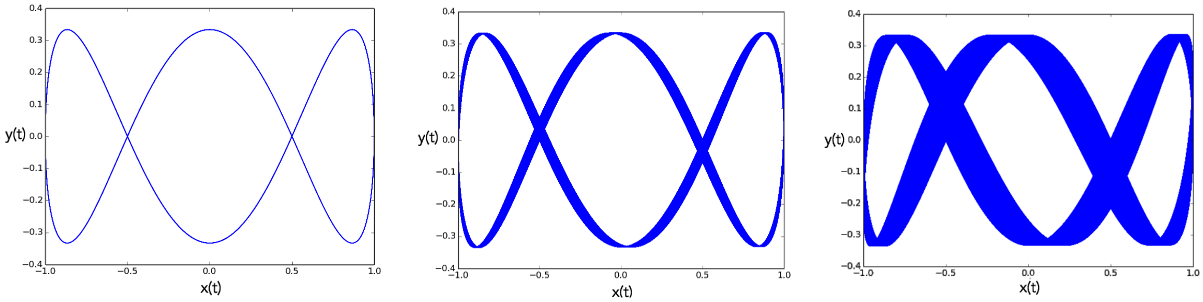
(c) Solution using $\epsilon = 0.00001$ and $t =$ 10000 a.u.

(d) Solution using $\epsilon = 0.00001$ and $t =$ 30000 a.u.

Figure 2: Showing the different solutions of the differential equation system shown in Equation 5 using variable values of ϵ and t. In all solutions $\omega_1 = \omega_2 = 1$. In the Fig. 2(c) and 2(d), a zoom in would show multiple lines laying besides each other creating the impression of a thicker line.

the solution is a closed lissajous figure, altough that is not the case for the solutions using small values of ϵ. There is a large difference in the solution using $\epsilon = 0.01$ (see Fig. 2(b)), compared to the one that is described by a harmonic oscillator (see Fig. 2(a)). There is also a large difference for the solutions using an even smaller ϵ, in this case $\epsilon = 0.00001$ (see Fig. 2(c) and 2(d)) but a much longer time is needed to notice the difference.

In Fig. 3 the solutions for Equation 5, using $\omega_1 = 1$ and $\omega_2 = 3$, are shown. For $\epsilon = 0$ a closed lissajous figure is obtained (see Fig. 3(a)), just like when using $\omega_1 = \omega_2 = 1$. When using $\epsilon = 0.01$ (see Fig. 3(b)) a difference can be seen, altough it is less clear than the one in Fig. 2 for the same value of ϵ. However for a longer time the difference is very clear here as well (see Fig. 3(c)). In both Fig. 2 and Fig. 3 the difference can be seen as a "thicker" line at small values of t, while at longer times the figures start to change more overall.



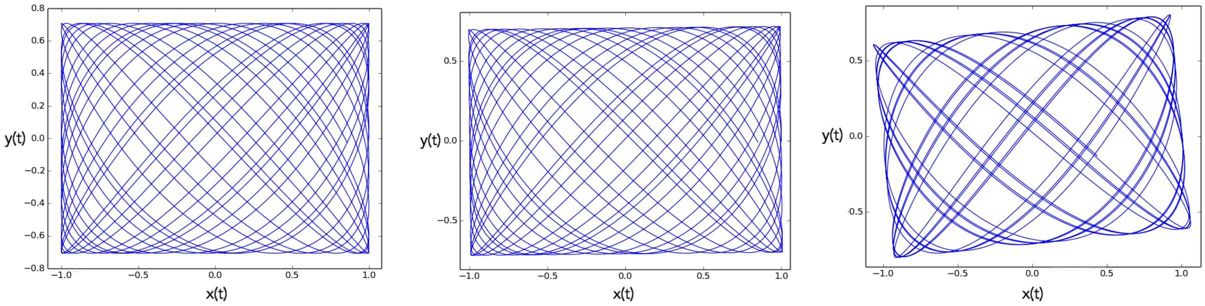
(a) Closed lissajous figure as the solution using $\epsilon = 0$. Because of its properties as a closed lissajous figure the time t is not important.

(b) Solution using $\epsilon = 0.01$ and $t =$ 10000 a.u.

(c) Solution using $\epsilon = 0.01$ and $t =$ 22500 a.u.

Figure 3: Showing the different solutions of the differential equation system shown in Equation 5 using variable values of ϵ and t. In all solutions $\omega_1 = 1$ and $\omega_2 = 3$. In Fig. 3(a) and 3(b), a zoom in would show multiple lines laying besides each other creating the impression of a thicker line.

The solutions for Equation 5 using $\omega_1 = 1$ and $\omega_2 = \sqrt{2}$, can be found in Fig. 4. A closed lissajous figure is not obtained when using $\epsilon = 0$ (see Fig. 4(a)), in contrast to the previous solutions. However it is still possible to spot a difference in the solutions when using $\epsilon \neq 0$. With $\epsilon = 0.01$ the lissajous figure tilts sligthly (see Fig. 4(b)). For a larger value of ϵ, in this case $\epsilon = 0.1$ the difference is much easier to spot (see Fig. 4(c)), and it makes the solution change overall as well apart from the tilting.



(a) Open lissajous figure as the solution using $\epsilon = 0$ and $t =$ 100 a.u.

(b) Solution using $\epsilon = 0.01$ and $t =$ 100 a.u.

(c) Solution using $\epsilon = 0.1$ and $t =$ 100 a.u.

Figure 4: Showing the different solutions of the differential equation system shown in Equation 5 using variable values of ϵ and t. In all solutions $\omega_1 = 1$ and $\omega_2 = \sqrt{2}$.

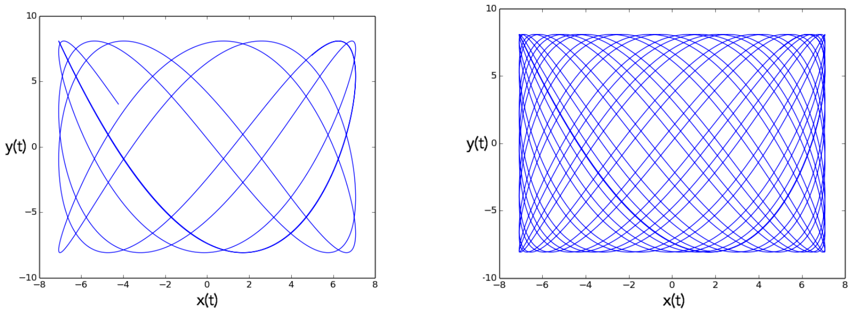
4 Discussion

As seen in the results, there is a significant difference when $\epsilon = 0$ compared to when $\epsilon \neq 0$, for all input values. When the frequency ratio ($\omega_1 : \omega_2$) was 1:1 a very distinct difference occured between the solution of Newton's equation and the PU oscillator. The difference was also easy to find at lower units of time or smaller values of ϵ.

For the frequency ratio 1:3 a longer time was needed than for the ratio 1:1. When using the ratio 1:$\sqrt{2}$ it was easy to spot the difference at small units of time, though a fairly large value of ϵ was needed.

When the frequency ratio was irrational, 1:$\sqrt{2}$, and $\epsilon = 0$ the plot would fill up close to every point in the rectangle created, but $\epsilon \neq 0$ especially $\epsilon = 0.1$ seems to change this behaviour even at small times, which is very interesting.

For the lissajous figure which frequency ratio was irrational it was shown to be very important to have the time remaining the same when varying ϵ, because when the frequency ratio was irrational a closed lissajous figure was not the solution. Therefore if t were varied the solution would have changed dramatically even if all other parametres remained the same, as can be seen in Fig. 5.



(a) Solution using $t = 30$ a.u.

(b) Solution using $t = 100$ a.u.

Figure 5: Showing how the solution changes over time when the frequency ratio is irrational while all other parameters remain the same.

For all frequency ratios, the solution of the PU oscillator is most often similar to the system described by Newton's law. In this case similar means it follows the same curvature but with a slight deflection which makes the lines go next to each other and on the zoom out of the pictures, makes it look like a thicker line. However when it is run for a longer time the difference changes overall. This is beneficial for the theory of HODs since it is known that Newton's laws would only be inaccurate with a tiny amount, at least for short times. It is hard to set up an experiment whihc can run for long periods of time without having other factors influencing it, and therefore the accuracy of Newton's laws for describing a harmonic oscillator for longer times is quite unknown.

For all the solutions in this study, the initial values $x(0)$, $\dot{x}(0)$, $y(0)$ and $\dot{y}(0)$ were the same. Although this makes the study narrower the results are still reliable, since these initial values are all possible to find in reality. However studying the differences for other initial values would be an interesting topic for further investigation. Additionally fairly large values of ϵ has been used to reduce the computation time. In reality it is plausible that the correction to Newton's laws should be described using much smaller values of ϵ.

To determine whether the harmonic oscillator or the PU oscillator describes an oscillation most accurate is not possible using this study alone. From this study it is possible to see the difference between the two ways to describe oscillation, however, which is the most correct in reality has to be determined experimentally. This study brings knowledge about which initial values are the most interesting to investigate further and for how long time it is needed to run an experiment to see the differences in reality. Hence it will be easier to develop an experiment, using the initial values used in this study, that might answer the question if Newton's laws are correct or not. As an extension to this study, an analytical understanding to the problem should also be constructed using perturbation series, in which a solution can be approximated using an analytical method, which might for example bring knowledge about how the angular frequencies affect the solution.

Although the idea of including a HOD into the description of an oscillator appeared in the field of quantum gravity it is interesting as a dynamical system by its own as well and might be applied to other fields in the long run.

In conclusion, there are differences between the solutions to the harmonic oscillator and the Pais-Uhlenbeck oscillator, which can be explained either by Newton's laws or the theory of implementing a higher order derivative being incorrect. Further studies, experimental in particular, are necessary to determine this.

5 Acknowledgments

I would like to thank my mentors Dhrubaditya Mitra and Bidya Binay Karak who have been truly helpful. I would also like to thank the Nordic Institute for Theoretical Physics for their hospitality and kindness. Thanks to my group members Toomas Liiv and Ottilia Andersson. A great thanks goes to Rays – for excellence and their partners, especially Teknikföretagen, KPS and Volvo for giving me this incredible opportunity. A personal thank you to Agnes Nordquist, my counsellor, as well as Mariam Andersson and Dennis Alp. Last but not least a huge thank you to Philip Frick for making this whole experience a memory for life.

References

[1] Pavšič, M. *Stable Self-Interacting Pais-Uhlenbeck Oscillator*, Mod. Phys. Lett. A, **28**(36), 2013.

[2] Stelle, K. S. *Classical Gravity with Higher Derivatives* , General Relativity and Gravitation, **9**(4), 1978.

[3] Pais, A., Uhlenbeck, G. E. *On Field Theories with Non-Localized Action*, Physical Review, **79**(1), 1950.

[4] Mostafazadeh, A. *A Hamiltonian Formulation of the Pais-Uhlenbeck Oscillator that Yields a Stable and Unitary Quantum System*, Phys. Lett. A, **375**(93–98), 2010.

[5] Atkinson, K., Han, W., Stewart, D. *Numerical Solution of Ordinary Differential Equations*, John Wiley & Sons, Inc. New Jersey, 2009.

A Python Code

Main file, odeN:

```
#Solve N first-order differential equation of the form:
#      dy/dt = f(y,t) + (stochastic noise)
#------------------------
""" Evolves in time N differential equation of the
  form :   dy/dt = f(y,t) + (stochastic noise)"""
# First import the needed modules
# also module for plotting
# odeN.py
from pylab import *
from matplotlib import *
import numpy as np
import timestep
import time
import sys
import equ
import os
import technical
model='none';
#-----------------
#------ Input parameters ------------
model='pais_uhlenbeck';
#-----------------
# step-size
dt=0.01;
# time
TMAX=100;
#---------------
ldiag2file=1
lparam2file=0
#----------------
def main():
  global snapcounter, l_storesnap, wdtstoresnap, dtstoresnap, stridestoresnap
  modtype,deterministic,stochastic=equ.select_model(model);
  if modtype == 'complex':
    equ.iniconf_complex();
  elif modtype == 'real':
    equ.iniconf(lparam2file);
  else:
    print 'modtype not recognised'
    print 'modtype = ',modtype;
    technical.end_program();
  y0=equ.y0;
  y=equ.y;
  f=equ.f;
#--------------
#   Check args for timestep to start on;
  t=0.0
  y=y0;
# Initial condition set
  print 'Initial time: t = ' + repr(t)
  print 'Initial guess of timestep: dt = ' + repr(dt)
  print 'Integrating upto time TMAX = ' + repr(TMAX)
# plot the initial condition
#clf()
#  plot([t],[0],'o')
#  show()
#-----------
# Integrate in time
  if (os.path.exists('diag_' + model)):
    print "Diagnostic file exists, renaming!"
    os.rename('diag_' + model, 'diag' + model + "_" + time.strftime("%Y-%m-%d %H:%M"))
  counter=0
  diagnostic_file = open('diag_' + model, 'wb')
  while (t < TMAX) :
    equ.diagnostic(y,t,counter,diagnostic_file,ldiag2file);
    # savestate:
```

```
    if deterministic ==1:
       y,y0 = timestep.rk4(y,t,dt,counter)
    if stochastic == 1:
       y=timestep.euler_marayuma(y,y0,t,dt,counter)
    t= t+dt
    counter= counter+1
    #    plot([t],[(energy-energy0)/energy0],'bo')
    #    plot([t], y[0], 'bo')
#-----------------------
if __name__ == '__main__':
  main();
else:
  print 'Importing ode solver'
  #---------------------
```

Module pais_uhlenbeck:

```
from pylab import *
import numpy as np
import noise as noise
from mpl_toolkits.mplot3d import Axes3D
# ---------------------
# omega_1 squared
omegax2=1;
# omega_2 squared
omegay2=1;
# epsilon
eps=0.0;
Nparticle=1;
dim=2;
ddim = 2*dim
pstride=ddim;
N=ddim;
f=np.zeros(N);
y0=np.zeros(N);
# ---------------------
# differential equations to solve
def model(y,t,dt,istep):
# y[0:2] is position, y[3:5] is momentum
    for ip in range(0,Nparticle):
        xx=y[0];
        vx=y[1];
        yy=y[2];
        vy=y[3];
        f[0]=vx;
        f[1] = eps*yy-omegax2*xx;
        f[2]=vy;
        f[3] = eps*xx-omegay2*yy;
    #print f
    return f;
# ---------------------
def dimension():
    return N;
# ---------------------
# initial values
def iniconf():
    y0[0]=1;
    y0[1]=0;
    y0[2]=0;
    y0[3]=1
    return y0;
# ---------------------
def diagnostic(y,t,counter,fname,ldiag2file):
    xx=y[0];
    vx=y[1];
    yy=y[2];
    vy=y[3];
    print t,xx,vx,yy,vy
# ---------------------
```

Module equ:

```
# This module contains the equation, initial condition etc.
#--------------------
# mod equ.py
from pylab import *
import numpy as np
import technical as technical
#----------------------
def iniconf(lparam2file=False):
  global N;
  global y,y0,f,eta1,eta2
  N=M.dimension();
  print 'Model variables are real'
  y0=np.zeros(N)
  y=np.zeros(N)
  f=np.zeros(N)
#---------------------
  if lparam2file:
    y0=M.iniconf(lparam2file);
  else:
    y0=M.iniconf();
  #  y0[0]=1.
  #y0[3]=1.
  return None
#---------------------
def iniconf_complex():
  global N;
  global y,y0,f,eta1,eta2;
  N=M.dimension();
  print 'Model variables are complex'
  y0=zeros(N,complex);
  y=zeros(N,complex);
  f=zeros(N,complex);
  #  eta1=zeros(N,complex);
  #eta2=zeros(N,complex);
#---------------------
  y0=M.iniconf();
  return None
#---------------------
def RHS(y,t,dt,istep):
  f = M.model(y,t,dt,istep);
  return f
#-----------------------
def SRHS(y0,t,dt,istep):
  f = M.Smodel(y,t,dt,istep)
  return f
#-----------------------
def diagnostic(y,t,counter, diagnostic_file,ldiag2file):
  M.diagnostic(y,t,counter, diagnostic_file,ldiag2file);
#-----------------------
def select_model(modelname):
  global M;
  deterministic=0;
  stochastic=0;
  if modelname == 'default':
    print 'no default model set';
    technical.end_program();
  elif modelname == 'pais_uhlenbeck':
    import pais_uhlenbeck as M;
    modtype='real';
    deterministic=1;
    stochastic=0;
  else:
    print modelname,'\t',':Model not found';
    technical.end_program();
  print 'Solving for',modelname
  return modtype,deterministic,stochastic;
#----------------------
```

Module timestep:

```
# Contains the time-stepping routines
#-----------------
# mod timestep.py
from pylab import *
import equ
# reload the moduel to make sure
# that new changes are incorporated
reload(equ)
# ==================================
# Integration schemes start below:
# ==================================
def rk4(y0,t,dt,counter):
  if counter == 1 :
    print " Integrating by Runge-Kutta 4th order scheme"
  istep=1;
  k1=dt*equ.RHS(y0,t,dt,istep)
  yhalf=y0+k1/2.
  thalf=t+dt/2.
  istep=2;
  k2=dt*equ.RHS(yhalf,thalf,dt,istep)
  yhalf=y0+k2/2.
  thalf=t+dt/2.
  istep=3;
  k3=dt*equ.RHS(yhalf,thalf,dt,istep)
  yhalf=y0+k3
  thalf=t+dt
  istep=4;
  k4=dt*equ.RHS(yhalf,thalf,dt,istep)
  y=y0+k1/6.+k2/3.+k3/3.+k4/6.
  return y,y0
#-------------------
```

The rest of the modules can be found at `https://code.google.com/p/pyoden/`.