PHY321: Introduction to Classical Mechanics and plans for Spring 2020

Morten Hjorth-Jensen[1,2]

Scott Pratt[1]

Carl Schmidt[3]

[1]Department of Physics and Astronomy and National Superconducting Cyclotron Laboratory, Michigan State University, USA
[2]Department of Physics, University of Oslo, Norway
[3]Department of Physics and Astronomy, Michigan State University, USA

Dec 16, 2020

Introduction

Classical mechanics is a topic which has been taught intensively over several centuries. It is, with its many variants and ways of presenting the educational material, normally the first **real** physics course many of us meet and it lays the foundation for further physics studies. Many of the equations and ways of reasoning about the underlying laws of motion and pertinent forces, shape our approaches and understanding of the scientific method and discourse, as well as the way we develop our insights and deeper understanding about physical systems.

There is a wealth of well-tested (from both a physics point of view and a pedagogical standpoint) exercises and problems which can be solved analytically. However, many of these problems represent idealized and less realistic situations. The large majority of these problems are solved by paper and pencil and are traditionally aimed at what we normally refer to as continuous models from which we may find an analytical solution. As a consequence, when teaching mechanics, it implies that we can seldomly venture beyond an idealized case in order to develop our understandings and insights about the underlying forces and laws of motion.

On the other hand, numerical algorithms call for approximate discrete models and much of the development of methods for continuous models are nowadays being replaced by methods for discrete models in science and industry, simply because **much larger classes of problems can be addressed** with discrete models, often by simpler and more generic methodologies.

As we will see below, when properly scaling the equations at hand, discrete models open up for more advanced abstractions and the possibility to study real life systems, with the added bonus that we can explore and deepen our basic understanding of various physical systems

Analytical solutions are as important as before. In addition, such solutions provide us with invaluable benchmarks and tests for our discrete models. Such benchmarks, as we will see below, allow us to discuss possible sources of errors and their behaviors. And finally, since most of our models are based on various algorithms from numerical mathematics, we have a unique oppotunity to gain a deeper understanding of the mathematical approaches we are using.

With computing and data science as important elements in essentially all aspects of a modern society, we could then try to define Computing as **solving scientific problems using all possible tools, including symbolic computing, computers and numerical algorithms, and analytical paper and pencil solutions**. Computing provides us with the tools to develope our own understanding of the scientific method by enhancing algorithmic thinking.

The way we will teach this course reflects this definition of computing. The course contains both classical paper and pencil exercises as well as computational projects and exercises. The hope is that this will allow you to explore the physics of systems governed by the degrees of freedom of classical mechanics at a deeper level, and that these insights about the scientific method will help you to develop a better understanding of how the underlying forces and equations of motion and how they impact a given system. Furthermore, by introducing various numerical methods via computational projects and exercises, we aim at developing your competences and skills about these topics.

These competences will enable you to

- understand how algorithms are used to solve mathematical problems,
- derive, verify, and implement algorithms,
- understand what can go wrong with algorithms,
- use these algorithms to construct reproducible scientific outcomes and to engage in science in ethical ways, and
- think algorithmically for the purposes of gaining deeper insights about scientific problems.

All these elements are central for maturing and gaining a better understanding of the modern scientific process *per se*.

The power of the scientific method lies in identifying a given problem as a special case of an abstract class of problems, identifying general solution methods for this class of problems, and applying a general method to the specific problem (applying means, in the case of computing, calculations by pen and paper, symbolic computing, or numerical computing by ready-made and/or self-written software). This generic view on problems and methods is particularly

important for understanding how to apply available, generic software to solve a particular problem.

However, verification of algorithms and understanding their limitations requires much of the classical knowledge about continuous models.

A well-known examples to illustrate many of the above concepts

Before we venture into a reminder on Python and mechanics relevant applications, let us briefly outline some of the abovementioned topics using an example many of you may have seen before in for example CMSE201. A simple algorithm for integration is the Trapezoidal rule. Integration of a function $f(x)$ by the Trapezoidal Rule is given by following algorithm for an interval $x \in [a, b]$

$$\int_a^b (f(x)dx = \frac{1}{2}\left[f(a) + 2f(a+h) + \cdots + 2f(b-h) + f(b)\right] + O(h^2),$$

where h is the so-called stepsize defined by the number of integration points N as $h = (b-a)/(n)$. Python offers an extremely versatile programming environment, allowing for the inclusion of analytical studies in a numerical program. Here we show an example code with the **trapezoidal rule**. We use also **SymPy** to evaluate the exact value of the integral and compute the absolute error with respect to the numerically evaluated one of the integral $\int_0^1 dx x^2 = 1/3$. The following code for the trapezoidal rule allows you to plot the relative error by comparing with the exact result. By increasing to 10^8 points one arrives at a region where numerical errors start to accumulate.

```
from math import log10
import numpy as np
from sympy import Symbol, integrate
import matplotlib.pyplot as plt
# function for the trapezoidal rule
def Trapez(a,b,f,n):
   h = (b-a)/float(n)
   s = 0
   x = a
   for i in range(1,n,1):
       x = x+h
       s = s+ f(x)
   s = 0.5*(f(a)+f(b)) +s
   return h*s
#  function to compute pi
def function(x):
    return x*x
# define integration limits
a = 0.0;  b = 1.0;
# find result from sympy
# define x as a symbol to be used by sympy
x = Symbol('x')
exact = integrate(function(x), (x, a, b))
# set up the arrays for plotting the relative error
n = np.zeros(9); y = np.zeros(9);
# find the relative error as function of integration points
```

```
for i in range(1, 8, 1):
    npts = 10**i
    result = Trapez(a,b,function,npts)
    RelativeError = abs((exact-result)/exact)
    n[i] = log10(npts); y[i] = log10(RelativeError);
plt.plot(n,y, 'ro')
plt.xlabel('n')
plt.ylabel('Relative error')
plt.show()
```

This example shows the potential of combining numerical algorithms with symbolic calculations, allowing us to

- Validate and verify their algorithms.
- Including concepts like unit testing, one has the possibility to test and test several or all parts of the code.
- Validation and verification are then included *naturally* and one can develop a better attitude to what is meant with an ethically sound scientific approach.
- The above example allows the student to also test the mathematical error of the algorithm for the trapezoidal rule by changing the number of integration points. The students get **trained from day one to think error analysis**.
- With a Jupyter notebook you can keep exploring similar examples and turn them in as your own notebooks.

In this process we can easily bake in

1. How to structure a code in terms of functions
2. How to make a module
3. How to read input data flexibly from the command line
4. How to create graphical/web user interfaces
5. How to write unit tests (test functions or doctests)
6. How to refactor code in terms of classes (instead of functions only)
7. How to conduct and automate large-scale numerical experiments
8. How to write scientific reports in various formats (LaTeX, HTML)

The conventions and techniques outlined here will save you a lot of time when you incrementally extend software over time from simpler to more complicated problems. In particular, you will benefit from many good habits:

1. New code is added in a modular fashion to a library (modules)

2. Programs are run through convenient user interfaces
3. It takes one quick command to let all your code undergo heavy testing
4. Tedious manual work with running programs is automated,
5. Your scientific investigations are reproducible, scientific reports with top quality typesetting are produced both for paper and electronic devices.

Space, Time, Motion, Reference Frames and Reminder on vectors and other mathematical quantities

Our studies will start with the motion of different types of objects such as a falling ball, a runner, a bicycle etc etc. It means that an object's position in space varies with time. In order to study such systems we need to define

- choice of origin
- choice of the direction of the axes
- choice of positive direction (left-handed or right-handed system of reference)
- choice of units and dimensions

These choices lead to some important questions such as

- is the physics of a system independent of the origin of the axes?
- is the physics independent of the directions of the axes, that is are there privileged axes?
- is the physics independent of the orientation of system?
- is the physics independent of the scale of the length?

Dimension, units and labels. Throughout this course we will use the standardized SI units. The standard unit for length is thus one meter 1m, for mass one kilogram 1kg, for time one second 1s, for force one Newton 1kgm/s^2 and for energy 1 Joule $1\text{kgm}^2\text{s}^{-2}$.

We will use the following notations for various variables (vectors are always boldfaced in these lecture notes):

- position $\boldsymbol{r}$, in one dimention we will normally just use x,
- mass m,
- time t,

- velocity $\boldsymbol{v}$ or just v in one dimension,
- acceleration $\boldsymbol{a}$ or just a in one dimension,
- momentum $\boldsymbol{p}$ or just p in one dimension,
- kinetic energy K,
- potential energy V and
- frequency ω.

More variables will be defined as we need them.

It is also important to keep track of dimensionalities. Don't mix this up with a chosen unit for a given variable. We mark the dimensionality in these lectures as $[a]$, where a is the quantity we are interested in. Thus

- $[\boldsymbol{r}]$ = length
- $[m]$ = mass
- $[K]$ = energy
- $[t]$ = time
- $[\boldsymbol{v}]$ = length over time
- $[\boldsymbol{a}]$ = length over time squared
- $[\boldsymbol{p}]$ = mass times length over time
- $[\omega]$ = 1/time

Elements of Vector Algebra

Note: This section is under revision

In these lectures we will use boldfaced lower-case letters to label a vector. A vector $\boldsymbol{a}$ in three dimensions is thus defined as

$$\boldsymbol{a} = (a_x, a_y, a_z),$$

and using the unit vectors in a cartesian system we have

$$\boldsymbol{a} = a_x\boldsymbol{e}_x + a_y\boldsymbol{e}_y + a_z\boldsymbol{e}_z,$$

where the unit vectors have magnitude $|\boldsymbol{e}_i| = 1$ with $i = x, y, z$.

Using the fact that multiplication of reals is distributive we can show that

$$\boldsymbol{a}(\boldsymbol{b} + \boldsymbol{c}) = \boldsymbol{ab} + \boldsymbol{ac},$$

Similarly we can also show that (using product rule for differentiating reals)

$$\frac{d}{dt}(\boldsymbol{ab}) = \boldsymbol{a}\frac{d\boldsymbol{b}}{dt} + \boldsymbol{b}\frac{d\boldsymbol{a}}{dt}.$$

We can repeat these operations for the cross products and show that they are distribuitive

$$\boldsymbol{a} \times (\boldsymbol{b} + \boldsymbol{c}) = \boldsymbol{a} \times \boldsymbol{b} + \boldsymbol{a} \times \boldsymbol{c}.$$

We have also that

$$\frac{d}{dt}(\boldsymbol{a} \times \boldsymbol{b}) = \boldsymbol{a} \times \frac{d\boldsymbol{b}}{dt} + \boldsymbol{b} \times \frac{d\boldsymbol{a}}{dt}.$$

The rotation of a three-dimensional vector $\boldsymbol{a} = (a_x, a_y, a_z)$ in the xy plane around an angle ϕ results in a new vector $\boldsymbol{b} = (b_x, b_y, b_z)$. This operation can be expressed in terms of linear algebra as a matrix (the rotation matrix) multiplied with a vector. We can write this as

$$\begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}.$$

We can write this in a more compact form as $\boldsymbol{b} = \boldsymbol{Ra}$, where the rotation matrix is defined as

$$\boldsymbol{R} = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Falling baseball in one dimension

We anticipate the mathematical model to come and assume that we have a model for the motion of a falling baseball without air resistance. Our system (the baseball) is at an initial height y_0 (which we will specify in the program below) at the initial time $t_0 = 0$. In our program example here we will plot the position in steps of Δt up to a final time t_f. The mathematical formula for the position $y(t)$ as function of time t is

$$y(t) = y_0 - \frac{1}{2}gt^2,$$

where $g = 9.80665 = 0.980655 \times 10^1 \mathrm{m/s}^2$ is a constant representing the standard acceleration due to gravity. We have here adopted the conventional standard value. This does not take into account other effects, such as buoyancy or drag. Furthermore, we stop when the ball hits the ground, which takes place at

$$y(t) = 0 = y_0 - \frac{1}{2}gt^2,$$

which gives us a final time $t_f = \sqrt{2y_0/g}$.

As of now we simply assume that we know the formula for the falling object. Afterwards, we will derive it.

Our Python Encounter

We start with preparing folders for storing our calculations, figures and if needed, specific data files we use as input or output files.

```
# Common imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os

# Where to save the figures and data files
PROJECT_ROOT_DIR = "Results"
FIGURE_ID = "Results/FigureFiles"
DATA_ID = "DataFiles/"

if not os.path.exists(PROJECT_ROOT_DIR):
    os.mkdir(PROJECT_ROOT_DIR)

if not os.path.exists(FIGURE_ID):
    os.makedirs(FIGURE_ID)

if not os.path.exists(DATA_ID):
    os.makedirs(DATA_ID)

def image_path(fig_id):
    return os.path.join(FIGURE_ID, fig_id)

def data_path(dat_id):
    return os.path.join(DATA_ID, dat_id)

def save_fig(fig_id):
    plt.savefig(image_path(fig_id) + ".png", format='png')

#in case we have an input file we wish to read in
#infile = open(data_path("MassEval2016.dat"),'r')
```

You could also define a function for making our plots. You can obviously avoid this and simply set up various **matplotlib** commands every time you need them. You may however find it convenient to collect all such commands in one function and simply call this function.

```
from pylab import plt, mpl
plt.style.use('seaborn')
mpl.rcParams['font.family'] = 'serif'

def MakePlot(x,y, styles, labels, axlabels):
    plt.figure(figsize=(10,6))
    for i in range(len(x)):
        plt.plot(x[i], y[i], styles[i], label = labels[i])
        plt.xlabel(axlabels[0])
        plt.ylabel(axlabels[1])
    plt.legend(loc=0)
```

Thereafter we start setting up the code for the falling object.

```
%matplotlib inline
import matplotlib.patches as mpatches
```

```
g = 9.80655 #m/s^2
y_0 = 10.0 # initial position in meters
DeltaT = 0.1  # time step
# final time when y = 0, t = sqrt(2*10/g)
tfinal = np.sqrt(2.0*y_0/g)
#set up arrays
t = np.arange(0,tfinal,DeltaT)
y =y_0 -g*.5*t**2
# Then make a nice printout in table form using Pandas
import pandas as pd
from IPython.display import display
data = {'t[s]': t,
        'y[m]': y
        }
RawData = pd.DataFrame(data)
display(RawData)
plt.style.use('ggplot')
plt.figure(figsize=(8,8))
plt.scatter(t, y, color = 'b')
blue_patch = mpatches.Patch(color = 'b', label = 'Height y as function of  time t')
plt.legend(handles=[blue_patch])
plt.xlabel("t[s]")
plt.ylabel("y[m]")
save_fig("FallingBaseball")
plt.show()
```

Here we used **pandas** (see below) to systemize the output of the position as function of time.

Average quantities

We define now the average velocity as

$$\overline{v}(t) = \frac{y(t+\Delta t) - y(t)}{\Delta t}.$$

In the code we have set the time step Δt to a given value. We could define it in terms of the number of points n as

$$\Delta t = \frac{t_{\mathrm{final}-}t_{\mathrm{initial}}}{n+1}.$$

Since we have discretized the variables, we introduce the counter i and let $y(t) \to y(t_i) = y_i$ and $t \to t_i$ with $i = 0, 1, \ldots, n$. This gives us the following shorthand notations that we will use for the rest of this course. We define

$$y_i = y(t_i), \quad i = 0, 1, 2, \ldots, n.$$

This applies to other variables which depend on say time. Examples are the velocities, accelerations, momenta etc. Furthermore we use the shorthand

$$y_{i\pm 1} = y(t_i \pm \Delta t), \ i = 0, 1, 2, \ldots, n.$$

Compact equations

We can then rewrite in a more compact form the average velocity as

$$\overline{v}_i = \frac{y_{i+1} - y_i}{\Delta t}.$$

The velocity is defined as the change in position per unit time. In the limit $\Delta t \to 0$ this defines the instantaneous velocity, which is nothing but the slope of the position at a time t. We have thus

$$v(t) = \frac{dy}{dt} = \lim_{\Delta t \to 0} \frac{y(t + \Delta t) - y(t)}{\Delta t}.$$

Similarly, we can define the average acceleration as the change in velocity per unit time as

$$\overline{a}_i = \frac{v_{i+1} - v_i}{\Delta t},$$

resulting in the instantaneous acceleration

$$a(t) = \frac{dv}{dt} = \lim_{\Delta t \to 0} \frac{v(t + \Delta t) - v(t)}{\Delta t}.$$

A note on notations: When writing for example the velocity as $v(t)$ we are then referring to the continuous and instantaneous value. A subscript like v_i refers always to the discretized values.

A differential equation

We can rewrite the instantaneous acceleration as

$$a(t) = \frac{dv}{dt} = \frac{d}{dt}\frac{dy}{dt} = \frac{d^2y}{dt^2}.$$

This forms the starting point for our definition of forces later. It is a famous second-order differential equation. If the acceleration is constant we can now recover the formula for the falling ball we started with. The acceleration can depend on the position and the velocity. To be more formal we should then write the above differential equation as

$$\frac{d^2y}{dt^2} = a(t, y(t), \frac{dy}{dt}).$$

With given initial conditions for $y(t_0)$ and $v(t_0)$ we can then integrate the above equation and find the velocities and positions at a given time t.

If we multiply with mass, we have one of the famous expressions for Newton's second law,

$$F(y, v, t) = m\frac{d^2y}{dt^2} = ma(t, y(t), \frac{dy}{dt}),$$

where F is the force acting on an object with mass m. We see that it also has the right dimension, mass times length divided by time squared. We will come back to this soon.

Integrating our equations

Formally we can then, starting with the acceleration (suppose we have measured it, how could we do that?) compute say the height of a building. To see this we perform the following integrations from an initial time t_0 to a given time t

$$\int_{t_0}^{t} dta(t) = \int_{t_0}^{t} dt\frac{dv}{dt} = v(t) - v(t_0),$$

or as

$$v(t) = v(t_0) + \int_{t_0}^{t} dta(t).$$

When we know the velocity as function of time, we can find the position as function of time starting from the defintion of velocity as the derivative with respect to time, that is we have

$$\int_{t_0}^{t} dtv(t) = \int_{t_0}^{t} dt\frac{dy}{dt} = y(t) - y(t_0),$$

or as

$$y(t) = y(t_0) + \int_{t_0}^{t} dtv(t).$$

These equations define what is called the integration method for finding the position and the velocity as functions of time. There is no loss of generality if we extend these equations to more than one spatial dimension.

Constant acceleration case, the velocity

Let us compute the velocity using the constant value for the acceleration given by $-g$. We have

$$v(t) = v(t_0) + \int_{t_0}^{t} dta(t) = v(t_0) + \int_{t_0}^{t} dt(-g).$$

Using our initial time as $t_0 = 0$s and setting the initial velocity $v(t_0) = v_0 = 0$m/s we get when integrating

$$v(t) = -gt.$$

The more general case is

$$v(t) = v_0 - g(t - t_0).$$

We can then integrate the velocity and obtain the final formula for the position as function of time through

$$y(t) = y(t_0) + \int_{t_0}^{t} dtv(t) = y_0 + \int_{t_0}^{t} dtv(t) = y_0 + \int_{t_0}^{t} dt(-gt),$$

With $y_0 = 10$m and $t_0 = 0$s, we obtain the equation we started with

$$y(t) = 10 - \frac{1}{2}gt^2.$$

Computing the averages

After this mathematical background we are now ready to compute the mean velocity using our data.

```
# Now we can compute the mean velocity using our data
# We define first an array Vaverage
n = np.size(t)
Vaverage = np.zeros(n)
for i in range(1,n-1):
    Vaverage[i] = (y[i+1]-y[i])/DeltaT
# Now we can compute the mean accelearatio using our data
# We define first an array Aaverage
n = np.size(t)
Aaverage = np.zeros(n)
Aaverage[0] = -g
for i in range(1,n-1):
    Aaverage[i] = (Vaverage[i+1]-Vaverage[i])/DeltaT
data = {'t[s]': t,
        'y[m]': y,
        'v[m/s]': Vaverage,
        'a[m/s^2]': Aaverage
        }
NewData = pd.DataFrame(data)
display(NewData[0:n-2])
```

Note that we don't print the last values!

Including Air Resistance in our model

In our discussions till now of the falling baseball, we have ignored air resistance and simply assumed that our system is only influenced by the gravitational force. We will postpone the derivation of air resistance till later, after our discussion of Newton's laws and forces.

For our discussions here it suffices to state that the accelerations is now modified to

$$\boldsymbol{a}(t) = -g + D\boldsymbol{v}(t)|v(t)|,$$

where $|v(t)|$ is the absolute value of the velocity and D is a constant which pertains to the specific object we are studying. Since we are dealing with motion in one dimension, we can simplify the above to

$$a(t) = -g + Dv^2(t).$$

We can rewrite this as a differential equation

$$a(t) = \frac{dv}{dt} = \frac{d^2y}{dt^2} = -g + Dv^2(t).$$

Using the integral equations discussed above we can integrate twice and obtain first the velocity as function of time and thereafter the position as function of time.

For this particular case, we can actually obtain an analytical solution for the velocity and for the position. Here we will first compute the solutions analytically, thereafter we will derive Euler's method for solving these differential equations numerically.

Analytical solutions

For simplicity let us just write $v(t)$ as v. We have

$$\frac{dv}{dt} = -g + Dv^2(t).$$

We can solve this using the technique of separation of variables. We isolate on the left all terms that involve v and on the right all terms that involve time. We get then

$$\frac{dv}{g - Dv^2(t)} = -dt,$$

We scale now the equation to the left by introducing a constant $v_T = \sqrt{g/D}$. This constant has dimension length/time. Can you show this?

Next we integrate the left-hand side (lhs) from $v_0 = 0$ m/s to v and the right-hand side (rhs) from $t_0 = 0$ to t and obtain

$$\int_0^v \frac{dv}{g - Dv^2(t)} = \frac{v_T}{g}\text{arctanh}(\frac{v}{v_T}) = -\int_0^t dt = -t.$$

We can reorganize these equations as

$$v_T\text{arctanh}(\frac{v}{v_T}) = -gt,$$

which gives us v as function of time

$$v(t) = v_T \tanh -(\frac{gt}{v_T}).$$

Finding the final height

With the velocity we can then find the height $y(t)$ by integrating yet another time, that is

$$y(t) = y(t_0) + \int_{t_0}^t dtv(t) = \int_0^t dt[v_T \tanh -(\frac{gt}{v_T})].$$

This integral is a little bit trickier but we can look it up in a table over known integrals and we get

$$y(t) = y(t_0) - \frac{v_T^2}{g}\log\left[\cosh\left(\frac{gt}{v_T}\right)\right].$$

Alternatively we could have used the symbolic Python package **Sympy** (example will be inserted later).

In most cases however, we need to revert to numerical solutions.

Our first attempt at solving differential equations

Here we will try the simplest possible approach to solving the second-order differential equation

$$a(t) = \frac{d^2y}{dt^2} = -g + Dv^2(t).$$

We rewrite it as two coupled first-order equations (this is a standard approach)

$$\frac{dy}{dt} = v(t),$$

with initial condition $y(t_0) = y_0$ and

$$a(t) = \frac{dv}{dt} = -g + Dv^2(t),$$

with initial condition $v(t_0) = v_0$.

Many of the algorithms for solving differential equations start with simple Taylor equations. If we now Taylor expand y and v around a value $t + \Delta t$ we have

$$y(t + \Delta t) = y(t) + \Delta t\frac{dy}{dt} + \frac{\Delta t^2}{2!}\frac{d^2y}{dt^2} + O(\Delta t^3),$$

and

$$v(t + \Delta t) = v(t) + \Delta t\frac{dv}{dt} + \frac{\Delta t^2}{2!}\frac{d^2v}{dt^2} + O(\Delta t^3).$$

Using the fact that $dy/dt = v$ and $dv/dt = a$ and keeping only terms up to Δt we have

$$y(t + \Delta t) = y(t) + \Delta tv(t) + O(\Delta t^2),$$

and

$$v(t + \Delta t) = v(t) + \Delta ta(t) + O(\Delta t^2).$$

Discretizing our equations

Using our discretized versions of the equations with for example $y_i = y(t_i)$ and $y_{i\pm1} = y(t_i + \Delta t)$, we can rewrite the above equations as (and truncating at Δt)

$$y_{i+1} = y_i + \Delta tv_i,$$

and

$$v_{i+1} = v_i + \Delta ta_i.$$

These are the famous Euler equations (forward Euler).

To solve these equations numerically we start at a time t_0 and simply integrate up these equations to a final time t_f, The step size Δt is an input parameter in our code. You can define it directly in the code below as

```
DeltaT = 0.1
```

With a given final time **tfinal** we can then find the number of integration points via the **ceil** function included in the **math** package of Python as

```
#define final time, assuming that initial time is zero
from math import ceil
tfinal = 0.5
n = ceil(tfinal/DeltaT)
print(n)
```

The **ceil** function returns the smallest integer not less than the input in say

```
x = 21.15
print(ceil(x))
```

which in the case here is 22.

```
x = 21.75
print(ceil(x))
```

which also yields 22. The **floor** function in the **math** package is used to return the closest integer value which is less than or equal to the specified expression or value. Compare the previous result to the usage of **floor**

```
from math import floor
x = 21.75
print(floor(x))
```

Alternatively, we can define ourselves the number of integration(mesh) points. In this case we could have

```
n = 10
tinitial = 0.0
tfinal = 0.5
DeltaT = (tfinal-tinitial)/(n)
print(DeltaT)
```

Since we will set up one-dimensional arrays that contain the values of various variables like time, position, velocity, acceleration etc, we need to know the value of n, the number of data points (or integration or mesh points). With n we can initialize a given array by setting all elelements to zero, as done here

```
# define array a
a = np.zeros(n)
print(a)
```

Code for implementing Euler's method

In the code here we implement this simple Eurler scheme choosing a value for $D = 0.0245$ m/s.

```
# Common imports
import numpy as np
import pandas as pd
from math import *
import matplotlib.pyplot as plt
import os
```

```
# Where to save the figures and data files
PROJECT_ROOT_DIR = "Results"
FIGURE_ID = "Results/FigureFiles"
DATA_ID = "DataFiles/"

if not os.path.exists(PROJECT_ROOT_DIR):
    os.mkdir(PROJECT_ROOT_DIR)

if not os.path.exists(FIGURE_ID):
    os.makedirs(FIGURE_ID)

if not os.path.exists(DATA_ID):
    os.makedirs(DATA_ID)

def image_path(fig_id):
    return os.path.join(FIGURE_ID, fig_id)

def data_path(dat_id):
    return os.path.join(DATA_ID, dat_id)

def save_fig(fig_id):
    plt.savefig(image_path(fig_id) + ".png", format='png')


g = 9.80655 #m/s^2
D = 0.00245 #m/s
DeltaT = 0.1
#set up arrays
tfinal = 0.5
n = ceil(tfinal/DeltaT)
# define scaling constant vT
vT = sqrt(g/D)
# set up arrays for t, a, v, and y and we can compare our results with analytical ones
t = np.zeros(n)
a = np.zeros(n)
v = np.zeros(n)
y = np.zeros(n)
yanalytic = np.zeros(n)
# Initial conditions
v[0] = 0.0  #m/s
y[0] = 10.0 #m
yanalytic[0] = y[0]
# Start integrating using Euler's method
for i in range(n-1):
    # expression for acceleration
    a[i] = -g + D*v[i]*v[i]
    # update velocity and position
    y[i+1] = y[i] + DeltaT*v[i]
    v[i+1] = v[i] + DeltaT*a[i]
    # update time to next time step and compute analytical answer
    t[i+1] = t[i] + DeltaT
    yanalytic[i+1] = y[0]-(vT*vT/g)*log(cosh(g*t[i+1]/vT))
    if ( y[i+1] < 0.0):
        break
a[n-1] = -g + D*v[n-1]*v[n-1]
data = {'t[s]': t,
        'y[m]': y-yanalytic,
        'v[m/s]': v,
        'a[m/s^2]': a
        }
NewData = pd.DataFrame(data)
```

```
display(NewData)
#finally we plot the data
fig, axs = plt.subplots(3, 1)
axs[0].plot(t, y, t, yanalytic)
axs[0].set_xlim(0, tfinal)
axs[0].set_ylabel('y and exact')
axs[1].plot(t, v)
axs[1].set_ylabel('v[m/s]')
axs[2].plot(t, a)
axs[2].set_xlabel('time[s]')
axs[2].set_ylabel('a[m/s^2]')
fig.tight_layout()
save_fig("EulerIntegration")
plt.show()
```

Try different values for Δt and study the difference between the exact solution and the numerical solution.

Simple extension, the Euler-Cromer method

The Euler-Cromer method is a simple variant of the standard Euler method. We use the newly updated velocity v_{i+1} as an input to the new position, that is, instead of

$$y_{i+1} = y_i + \Delta t v_i,$$

and

$$v_{i+1} = v_i + \Delta t a_i,$$

we use now the newly calculate for v_{i+1} as input to y_{i+1}, that is we compute first

$$v_{i+1} = v_i + \Delta t a_i,$$

and then

$$y_{i+1} = y_i + \Delta t v_{i+1},$$

Implementing the Euler-Cromer method yields a simple change to the previous code. We only need to change the following line in the loop over time steps

```
for i in range(n-1):
    # more codes in between here
    v[i+1] = v[i] + DeltaT*a[i]
    y[i+1] = y[i] + DeltaT*v[i+1]
    # more code
```

Python practicalities, Software and needed installations

We will make extensive use of Python as programming language and its myriad of available libraries. You will find Jupyter notebooks invaluable in your work.

If you have Python installed (we strongly recommend Python3) and you feel pretty familiar with installing different packages, we recommend that you install the following Python packages via **pip** as

1. pip install numpy scipy matplotlib ipython scikit-learn mglearn sympy pandas pillow

For Python3, replace **pip** with **pip3**.

For OSX users we recommend, after having installed Xcode, to install **brew**. Brew allows for a seamless installation of additional software via for example

1. brew install python3

For Linux users, with its variety of distributions like for example the widely popular Ubuntu distribution, you can use **pip** as well and simply install Python as

1. sudo apt-get install python3 (or python for pyhton2.7)

etc etc.

Python installers

If you don't want to perform these operations separately and venture into the hassle of exploring how to set up dependencies and paths, we recommend two widely used distrubutions which set up all relevant dependencies for Python, namely

- Anaconda,

which is an open source distribution of the Python and R programming languages for large-scale data processing, predictive analytics, and scientific computing, that aims to simplify package management and deployment. Package versions are managed by the package management system **conda**.

- Enthought canopy

is a Python distribution for scientific and analytic computing distribution and analysis environment, available for free and under a commercial license.

Furthermore, Google's Colab is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. Try it out!

Useful Python libraries

Here we list several useful Python libraries we strongly recommend (if you use anaconda many of these are already there)

- NumPy is a highly popular library for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays
- The pandas library provides high-performance, easy-to-use data structures and data analysis tools

- Xarray is a Python package that makes working with labelled multi-dimensional arrays simple, efficient, and fun!
- Scipy (pronounced "Sigh Pie") is a Python-based ecosystem of open-source software for mathematics, science, and engineering.
- Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.
- Autograd can automatically differentiate native Python and Numpy code. It can handle a large subset of Python's features, including loops, ifs, recursion and closures, and it can even take derivatives of derivatives of derivatives
- SymPy is a Python library for symbolic mathematics.
- scikit-learn has simple and efficient tools for machine learning, data mining and data analysis
- TensorFlow is a Python library for fast numerical computing created and released by Google
- Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano
- And many more such as pytorch, Theano etc

Your jupyter notebook can easily be converted into a nicely rendered **PDF** file or a Latex file for further processing. For example, convert to latex as

```
pycod jupyter nbconvert filename.ipynb --to latex
```

And to add more versatility, the Python package SymPy is a Python library for symbolic mathematics. It aims to become a full-featured computer algebra system (CAS) and is entirely written in Python.

Numpy examples and Important Matrix and vector handling packages

There are several central software libraries for linear algebra and eigenvalue problems. Several of the more popular ones have been wrapped into ofter software packages like those from the widely used text **Numerical Recipes**. The original source codes in many of the available packages are often taken from the widely used software package LAPACK, which follows two other popular packages developed in the 1970s, namely EISPACK and LINPACK. We describe them shortly here.

- LINPACK: package for linear equations and least square problems.

- LAPACK:package for solving symmetric, unsymmetric and generalized eigenvalue problems. From LAPACK's website http://www.netlib.org it is possible to download for free all source codes from this library. Both C/C++ and Fortran versions are available.
- BLAS (I, II and III): (Basic Linear Algebra Subprograms) are routines that provide standard building blocks for performing basic vector and matrix operations. Blas I is vector operations, II vector-matrix operations and III matrix-matrix operations. Highly parallelized and efficient codes, all available for download from http://www.netlib.org.

Basic Matrix Features

Matrix properties reminder.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \qquad \mathbf{I} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The inverse of a matrix is defined by

$$\mathbf{A}^{-1} \cdot \mathbf{A} = I$$

Relations	Name	matrix elements
$A = A^T$	symmetric	$a_{ij} = a_{ji}$
$A = \left(A^T\right)^{-1}$	real orthogonal	$\sum_k a_{ik}a_{jk} = \sum_k a_{ki}a_{kj} = \delta_{ij}$
$A = A^*$	real matrix	$a_{ij} = a^*_{ij}$
$A = A^\dagger$	hermitian	$a_{ij} = a^*_{ji}$
$A = \left(A^\dagger\right)^{-1}$	unitary	$\sum_k a_{ik}a^*_{jk} = \sum_k a^*_{ki}a_{kj} = \delta_{ij}$

Some famous Matrices.

- Diagonal if $a_{ij} = 0$ for $i \neq j$
- Upper triangular if $a_{ij} = 0$ for $i > j$
- Lower triangular if $a_{ij} = 0$ for $i < j$
- Upper Hessenberg if $a_{ij} = 0$ for $i > j + 1$
- Lower Hessenberg if $a_{ij} = 0$ for $i < j + 1$
- Tridiagonal if $a_{ij} = 0$ for $|i - j| > 1$
- Lower banded with bandwidth p: $a_{ij} = 0$ for $i > j + p$
- Upper banded with bandwidth p: $a_{ij} = 0$ for $i < j + p$
- Banded, block upper triangular, block lower triangular....

More Basic Matrix Features.

Some Equivalent Statements. For an $N \times N$ matrix $\mathbf{A}$ the following properties are all equivalent

- If the inverse of $\mathbf{A}$ exists, $\mathbf{A}$ is nonsingular.
- The equation $\mathbf{Ax} = 0$ implies $\mathbf{x} = 0$.
- The rows of $\mathbf{A}$ form a basis of R^N.
- The columns of $\mathbf{A}$ form a basis of R^N.
- $\mathbf{A}$ is a product of elementary matrices.
- 0 is not eigenvalue of $\mathbf{A}$.

Numpy and arrays

Numpy provides an easy way to handle arrays in Python. The standard way to import this library is as

```
import numpy as np
```

Here follows a simple example where we set up an array of ten elements, all determined by random numbers drawn according to the normal distribution,

```
n = 10
x = np.random.normal(size=n)
print(x)
```

We defined a vector x with $n = 10$ elements with its values given by the Normal distribution $N(0, 1)$. Another alternative is to declare a vector as follows

```
import numpy as np
x = np.array([1, 2, 3])
print(x)
```

Here we have defined a vector with three elements, with $x_0 = 1$, $x_1 = 2$ and $x_2 = 3$. Note that both Python and C++ start numbering array elements from 0 and on. This means that a vector with n elements has a sequence of entities $x_0, x_1, x_2, \ldots, x_{n-1}$. We could also let (recommended) Numpy to compute the logarithms of a specific array as

```
import numpy as np
x = np.log(np.array([4, 7, 8]))
print(x)
```

In the last example we used Numpy's unary function *np.log*. This function is highly tuned to compute array elements since the code is vectorized and does not require looping. We normaly recommend that you use the Numpy intrinsic functions instead of the corresponding **log** function from Python's **math** module. The looping is done explicitely by the **np.log** function. The alternative, and slower way to compute the logarithms of a vector would be to write

```
import numpy as np
from math import log
x = np.array([4, 7, 8])
for i in range(0, len(x)):
    x[i] = log(x[i])
print(x)
```

We note that our code is much longer already and we need to import the **log** function from the **math** module. The attentive reader will also notice that the output is $[1, 1, 2]$. Python interprets automagically our numbers as integers (like the **automatic** keyword in C++). To change this we could define our array elements to be double precision numbers as

```
import numpy as np
x = np.log(np.array([4, 7, 8], dtype = np.float64))
print(x)
```

or simply write them as double precision numbers (Python uses 64 bits as default for floating point type variables), that is

```
import numpy as np
x = np.log(np.array([4.0, 7.0, 8.0])
print(x)
```

To check the number of bytes (remember that one byte contains eight bits for double precision variables), you can use simple use the **itemsize** functionality (the array x is actually an object which inherits the functionalities defined in Numpy) as

```
import numpy as np
x = np.log(np.array([4.0, 7.0, 8.0])
print(x.itemsize)
```

Matrices in Python

Having defined vectors, we are now ready to try out matrices. We can define a 3×3 real matrix $\hat{A}$ as (recall that we user lowercase letters for vectors and uppercase letters for matrices)

```
import numpy as np
A = np.log(np.array([ [4.0, 7.0, 8.0], [3.0, 10.0, 11.0], [4.0, 5.0, 7.0] ]))
print(A)
```

If we use the **shape** function we would get $(3, 3)$ as output, that is verifying that our matrix is a 3×3 matrix. We can slice the matrix and print for example the first column (Python organized matrix elements in a row-major order, see below) as

```
import numpy as np
A = np.log(np.array([ [4.0, 7.0, 8.0], [3.0, 10.0, 11.0], [4.0, 5.0, 7.0] ]))
# print the first column, row-major order and elements start with 0
print(A[:,0])
```

We can continue this was by printing out other columns or rows. The example here prints out the second column

```
import numpy as np
A = np.log(np.array([ [4.0, 7.0, 8.0], [3.0, 10.0, 11.0], [4.0, 5.0, 7.0] ]))
# print the first column, row-major order and elements start with 0
print(A[1,:])
```

Numpy contains many other functionalities that allow us to slice, subdivide etc etc arrays. We strongly recommend that you look up the Numpy website for more details. Useful functions when defining a matrix are the **np.zeros** function which declares a matrix of a given dimension and sets all elements to zero

```
import numpy as np
n = 10
# define a matrix of dimension 10 x 10 and set all elements to zero
A = np.zeros( (n, n) )
print(A)
```

or initializing all elements to

```
import numpy as np
n = 10
# define a matrix of dimension 10 x 10 and set all elements to one
A = np.ones( (n, n) )
print(A)
```

or as unitarily distributed random numbers (see the material on random number generators in the statistics part)

```
import numpy as np
n = 10
# define a matrix of dimension 10 x 10 and set all elements to random numbers with x \in [0, 1]
A = np.random.rand(n, n)
print(A)
```

Meet the Pandas

Another useful Python package is pandas, which is an open source library providing high-performance, easy-to-use data structures and data analysis tools

for Python. **pandas** stands for panel data, a term borrowed from econometrics and is an efficient library for data analysis with an emphasis on tabular data. **pandas** has two major classes, the **DataFrame** class with two-dimensional data objects and tabular data organized in columns and the class **Series** with a focus on one-dimensional data objects. Both classes allow you to index data easily as we will see in the examples below. **pandas** allows you also to perform mathematical operations on the data, spanning from simple reshapings of vectors and matrices to statistical operations.

The following simple example shows how we can, in an easy way make tables of our data. Here we define a data set which includes names, place of birth and date of birth, and displays the data in an easy to read way. We will see repeated use of **pandas**, in particular in connection with classification of data.

```
import pandas as pd
from IPython.display import display
data = {'First Name': ["Frodo", "Bilbo", "Aragorn II", "Samwise"],
        'Last Name': ["Baggins", "Baggins","Elessar","Gamgee"],
        'Place of birth': ["Shire", "Shire", "Eriador", "Shire"],
        'Date of Birth T.A.': [2968, 2890, 2931, 2980]
        }
data_pandas = pd.DataFrame(data)
display(data_pandas)
```

In the above we have imported **pandas** with the shorthand **pd**, the latter has become the standard way we import **pandas**. We make then a list of various variables and reorganize the above lists into a **DataFrame** and then print out a neat table with specific column labels as *Name*, *place of birth* and *date of birth*. Displaying these results, we see that the indices are given by the default numbers from zero to three. **pandas** is extremely flexible and we can easily change the above indices by defining a new type of indexing as

```
data_pandas = pd.DataFrame(data,index=['Frodo','Bilbo','Aragorn','Sam'])
display(data_pandas)
```

Thereafter we display the content of the row which begins with the index **Aragorn**

```
display(data_pandas.loc['Aragorn'])
```

We can easily append data to this, for example

```
new_hobbit = {'First Name': ["Peregrin"],
              'Last Name': ["Took"],
              'Place of birth': ["Shire"],
              'Date of Birth T.A.': [2990]
              }
data_pandas=data_pandas.append(pd.DataFrame(new_hobbit, index=['Pippin']))
display(data_pandas)
```

Here are other examples where we use the **DataFrame** functionality to handle arrays, now with more interesting features for us, namely numbers. We set up a matrix of dimensionality 10×5 and compute the mean value and standard deviation of each column. Similarly, we can perform mathematial operations like squaring the matrix elements and many other operations.

```
import numpy as np
import pandas as pd
from IPython.display import display
np.random.seed(100)
# setting up a 10 x 5 matrix
rows = 10
cols = 5
a = np.random.randn(rows,cols)
df = pd.DataFrame(a)
display(df)
print(df.mean())
print(df.std())
display(df**2)
```

Thereafter we can select specific columns only and plot final results

```
df.columns = ['First', 'Second', 'Third', 'Fourth', 'Fifth']
df.index = np.arange(10)

display(df)
print(df['Second'].mean() )
print(df.info())
print(df.describe())

from pylab import plt, mpl
plt.style.use('seaborn')
mpl.rcParams['font.family'] = 'serif'

df.cumsum().plot(lw=2.0, figsize=(10,6))
plt.show()


df.plot.bar(figsize=(10,6), rot=15)
plt.show()
```

We can produce a 4×4 matrix

```
b = np.arange(16).reshape((4,4))
print(b)
df1 = pd.DataFrame(b)
print(df1)
```

and many other operations.

The **Series** class is another important class included in **pandas**. You can view it as a specialization of **DataFrame** but where we have just a single column of data. It shares many of the same features as $_{D}ataFrame.Aswith$**DataFrame**, $mostoperationsarevectorized, ac$

Basic Steps of Scientific Investigations

An overarching aim in this course is to give you a deeper understanding of the scientific method. The problems we study will all involve cases where we can apply classical mechanics. In our previous material we already assumed that we had a model for the motion of an object. Alternatively we could have data from experiment (like Usain Bolt's 100m world record run in 2008). Or we could have performed ourselves an experiment and we want to understand which forces are

at play and whether these forces can be understood in terms of fundamental forces.

Our first step consists in identifying the problem. What we sketch here may include a mix of experiment and theoretical simulations, or just experiment or only theory.

Identifying our System

Here we can ask questions like

1. What kind of object is moving
2. What kind of data do we have
3. How do we measure position, velocity, acceleration etc
4. Which initial conditions influence our system
5. Other aspects which allow us to identify the system

Defining a Model

With our eventual data and observations we would now like to develop a model for the system. In the end we want obviously to be able to understand which forces are at play and how they influence our specific system. That is, can we extract some deeper insights about a system?

We need then to

1. Find the forces that act on our system
2. Introduce models for the forces
3. Identify the equations which can govern the system (Newton's second law for example)
4. More elements we deem important for defining our model

Solving the Equations

With the model at hand, we can then solve the equations. In classical mechanics we normally end up with solving sets of coupled ordinary differential equations or partial differential equations.

1. Using Newton's second law we have equations of the type $\boldsymbol{F} = m\boldsymbol{a} = md\boldsymbol{v}/dt$
2. We need to define the initial conditions (typically the initial velocity and position as functions of time) and/or initial conditions and boundary conditions

3. The solution of the equations give us then the position, the velocity and other time-dependent quantities which may specify the motion of a given object.

We are not yet done. With our lovely solvers, we need to start thinking.

Now it is time to ask the big questions. What do our results mean? Can we give a simple interpretation in terms of fundamental laws? What do our results mean? Are they correct? Thus, typical questions we may ask are

1. Are our results for say $\boldsymbol{r}(t)$ valid? Do we trust what we did? Can you validate and verify the correctness of your results?

2. Evaluate the answers and their implications

3. Compare with experimental data if possible. Does our model make sense?

4. and obviously many other questions.

The analysis stage feeds back to the first stage. It may happen that the data we had were not good enough, there could be large statistical uncertainties. We may need to collect more data or perhaps we did a sloppy job in identifying the degrees of freedom.

All these steps are essential elements in a scientific enquiry. Hopefully, through a mix of numerical simulations, analytical calculations and experiments we may gain a deeper insight about the physics of a specific system.

Let us now remind ourselves of Newton's laws, since these are the laws of motion we will study in this course.

Newton's Laws

When analyzing a physical system we normally start with distinguishing between the object we are studying (we will label this in more general terms as our **system**) and how this system interacts with the environment (which often means everything else!)

In our investigations we will thus analyze a specific physics problem in terms of the system and the environment. In doing so we need to identify the forces that act on the system and assume that the forces acting on the system must have a source, an identifiable cause in the environment.

A force acting on for example a falling object must be related to an interaction with something in the environment. This also means that we do not consider internal forces. The latter are forces between one part of the object and another part. In this course we will mainly focus on external forces.

Forces are either contact forces or long-range forces.

Contact forces, as evident from the name, are forces that occur at the contact between the system and the environment. Well-known long-range forces are the gravitional force and the electromagnetic force.

Setting up a model for forces acting on an object

In order to set up the forces which act on an object, the following steps may be useful

1. Divide the problem into system and environment.
2. Draw a figure of the object and everything in contact with the object.
3. Draw a closed curve around the system.
4. Find contact points—these are the points where contact forces may act.
5. Give names and symbols to all the contact forces.
6. Identify the long-range forces.
7. Make a drawing of the object. Draw the forces as arrows, vectors, starting from where the force is acting. The direction of the vector(s) indicates the (positive) direction of the force. Try to make the length of the arrow indicate the relative magnitude of the forces.
8. Draw in the axes of the coordinate system. It is often convenient to make one axis parallel to the direction of motion. When you choose the direction of the axis you also choose the positive direction for the axis.

Newton's Laws, the Second one first

Newton's second law of motion: The force $\boldsymbol{F}$ on an object of inertial mass m is related to the acceleration a of the object through

$$\boldsymbol{F} = m\boldsymbol{a},$$

where $\boldsymbol{a}$ is the acceleration.

Newton's laws of motion are laws of nature that have been found by experimental investigations and have been shown to hold up to continued experimental investigations. Newton's laws are valid over a wide range of length- and time-scales. We use Newton's laws of motion to describe everything from the motion of atoms to the motion of galaxies.

The second law is a vector equation with the acceleration having the same direction as the force. The acceleration is proportional to the force via the mass m of the system under study.

Newton's second law introduces a new property of an object, the so-called inertial mass m. We determine the inertial mass of an object by measuring the acceleration for a given applied force.

Then the First Law

What happens if the net external force on a body is zero? Applying Newton's second law, we find:

$$\boldsymbol{F} = 0 = m\boldsymbol{a},$$

which gives using the definition of the acceleration

$$\boldsymbol{a} = \frac{d\boldsymbol{v}}{dt} = 0.$$

The acceleration is zero, which means that the velocity of the object is constant. This is often referred to as Newton's first law. An object in a state of uniform motion tends to remain in that state unless an external force changes its state of motion. Why do we need a separate law for this? Is it not simply a special case of Newton's second law? Yes, Newton's first law can be deduced from the second law as we have illustrated. However, the first law is often used for a different purpose: Newton's First Law tells us about the limit of applicability of Newton's Second law. Newton's Second law can only be used in reference systems where the First law is obeyed. But is not the First law always valid? No! The First law is only valid in reference systems that are not accelerated. If you observe the motion of a ball from an accelerating car, the ball will appear to accelerate even if there are no forces acting on it. We call systems that are not accelerating inertial systems, and Newton's first law is often called the law of inertia. Newton's first and second laws of motion are only valid in inertial systems.

A system is an inertial system if it is not accelerated. It means that the reference system must not be accelerating linearly or rotating. Unfortunately, this means that most systems we know are not really inertial systems. For example, the surface of the Earth is clearly not an inertial system, because the Earth is rotating. The Earth is also not an inertial system, because it ismoving in a curved path around the Sun. However, even if the surface of the Earth is not strictly an inertial system, it may be considered to be approximately an inertial system for many laboratory-size experiments.

And finally the Third Law

If there is a force from object A on object B, there is also a force from object B on object A. This fundamental principle of interactions is called Newton's third law. We do not know of any force that do not obey this law: All forces appear in pairs. Newton's third law is usually formulated as: For every action there is an equal and opposite reaction.

Motion of a Single Object

Here we consider the motion of a single particle moving under the influence of some set of forces. We will consider some problems where the force does not depend on the position. In that case Newton's law $m\dot{\boldsymbol{v}} = \boldsymbol{F}(\boldsymbol{v})$ is a first-order

differential equation and one solves for $\boldsymbol{v}(t)$, then moves on to integrate $\boldsymbol{v}$ to get the position. In essentially all of these cases we cna find an analytical solution.

Air Resistance in One Dimension

Air resistance tends to scale as the square of the velocity. This is in contrast to many problems chosen for textbooks, where it is linear in the velocity. The choice of a linear dependence is motivated by mathematical simplicity (it keeps the differential equation linear) rather than by physics. One can see that the force should be quadratic in velocity by considering the momentum imparted on the air molecules. If an object sweeps through a volume dV of air in time dt, the momentum imparted on the air is

$$dP = \rho_m dV v, \tag{1}$$

where v is the velocity of the object and ρ_m is the mass density of the air. If the molecules bounce back as opposed to stop you would double the size of the term. The opposite value of the momentum is imparted onto the object itself. Geometrically, the differential volume is

$$dV = Avdt, \tag{2}$$

where A is the cross-sectional area and vdt is the distance the object moved in time dt.

Resulting Acceleration

Plugging this into the expression above,

$$\frac{dP}{dt} = -\rho_m A v^2. \tag{3}$$

This is the force felt by the particle, and is opposite to its direction of motion. Now, because air doesn't stop when it hits an object, but flows around the best it can, the actual force is reduced by a dimensionless factor c_W, called the drag coefficient.

$$F_{\text{drag}} = -c_W \rho_m A v^2, \tag{4}$$

and the acceleration is

$$\frac{dv}{dt} = -\frac{c_W \rho_m A}{m} v^2. \tag{5}$$

For a particle with initial velocity v_0, one can separate the dt to one side of the equation, and move everything with vs to the other side. We did this in our discussion of simple motion and will not repeat it here.

On more general terms, for many systems, e.g. an automobile, there are multiple sources of resistance. In addition to wind resistance, where the force is

proportional to v^2, there are dissipative effects of the tires on the pavement, and in the axel and drive train. These other forces can have components that scale proportional to v, and components that are independent of v. Those independent of v, e.g. the usual $f = \mu_K N$ frictional force you consider in your first Physics courses, only set in once the object is actually moving. As speeds become higher, the v^2 components begin to dominate relative to the others. For automobiles at freeway speeds, the v^2 terms are largely responsible for the loss of efficiency. To travel a distance L at fixed speed v, the energy/work required to overcome the dissipative forces are fL, which for a force of the form $f = \alpha v^n$ becomes

$$W = \int dx\ f = \alpha v^n L. \tag{6}$$

For $n = 0$ the work is independent of speed, but for the wind resistance, where $n = 2$, slowing down is essential if one wishes to reduce fuel consumption. It is also important to consider that engines are designed to be most efficient at a chosen range of power output. Thus, some cars will get better mileage at higher speeds (They perform better at 50 mph than at 5 mph) despite the considerations mentioned above.

Going Ballistic, Projectile Motion or a Softer Approach, Falling Raindrops

As an example of Newton's Laws we consider projectile motion (or a falling raindrop or a ball we throw up in the air) with a drag force. Even though air resistance is largely proportional to the square of the velocity, we will consider the drag force to be linear to the velocity, $\boldsymbol{F} = -m\gamma\boldsymbol{v}$, for the purposes of this exercise. The acceleration for a projectile moving upwards, $\boldsymbol{a} = \boldsymbol{F}/m$, becomes

$$\begin{aligned} \frac{dv_x}{dt} &= -\gamma v_x, \\ \frac{dv_y}{dt} &= -\gamma v_y - g, \end{aligned} \tag{7}$$

and γ has dimensions of inverse time.

If you on the other hand have a falling raindrop, how do these equations change? See for example Figure 2.1 in Taylor. Let us stay with a ball which is thrown up in the air at $t = 0$.

Ways of solving these equations

We will go over two different ways to solve this equation. The first by direct integration, and the second as a differential equation. To do this by direct integration, one simply multiplies both sides of the equations above by dt, then divide by the appropriate factors so that the vs are all on one side of the equation and the dt is on the other. For the x motion one finds an easily integrable equation,

$$
\begin{aligned}
\frac{dv_x}{v_x} &= -\gamma dt, \qquad (8)\\
\int_{v_{0x}}^{v_x} \frac{dv_x}{v_x} &= -\gamma \int_0^t dt,\\
\ln\left(\frac{v_x}{v_{0x}}\right) &= -\gamma t,\\
v_x(t) &= v_{0x} e^{-\gamma t}.
\end{aligned}
$$

This is very much the result you would have written down by inspection. For the y-component of the velocity,

$$
\begin{aligned}
\frac{dv_y}{v_y + g/\gamma} &= -\gamma dt \qquad (9)\\
\ln\left(\frac{v_y + g/\gamma}{v_{0y} - g/\gamma}\right) &= -\gamma t_f,\\
v_{fy} &= -\frac{g}{\gamma} + \left(v_{0y} + \frac{g}{\gamma}\right) e^{-\gamma t}.
\end{aligned}
$$

Whereas v_x starts at some value and decays exponentially to zero, v_y decays exponentially to the terminal velocity, $v_t = -g/\gamma$.

Solving as differential equations

Although this direct integration is simpler than the method we invoke below, the method below will come in useful for some slightly more difficult differential equations in the future. The differential equation for v_x is straight-forward to solve. Because it is first order there is one arbitrary constant, A, and by inspection the solution is

$$v_x = Ae^{-\gamma t}. \qquad (10)$$

The arbitrary constants for equations of motion are usually determined by the initial conditions, or more generally boundary conditions. By inspection $A = v_{0x}$, the initial x component of the velocity.

Differential Equations, contn

The differential equation for v_y is a bit more complicated due to the presence of g. Differential equations where all the terms are linearly proportional to a function, in this case v_y, or to derivatives of the function, e.g., v_y, dv_y/dt, $d^2v_y/dt^2 \cdots$, are called linear differential equations. If there are terms proportional to v^2, as would happen if the drag force were proportional to the square of the velocity, the differential equation is not longer linear. Because this expression has only one

derivative in v it is a first-order linear differential equation. If a term were added proportional to d^2v/dt^2 it would be a second-order differential equation. In this case we have a term completely independent of v, the gravitational acceleration g, and the usual strategy is to first rewrite the equation with all the linear terms on one side of the equal sign,

$$\frac{dv_y}{dt} + \gamma v_y = -g. \tag{11}$$

Splitting into two parts

Now, the solution to the equation can be broken into two parts. Because this is a first-order differential equation we know that there will be one arbitrary constant. Physically, the arbitrary constant will be determined by setting the initial velocity, though it could be determined by setting the velocity at any given time. Like most differential equations, solutions are not "solved". Instead, one guesses at a form, then shows the guess is correct. For these types of equations, one first tries to find a single solution, i.e. one with no arbitrary constants. This is called the *particular* solution, $y_p(t)$, though it should really be called "a" particular solution because there are an infinite number of such solutions. One then finds a solution to the *homogenous* equation, which is the equation with zero on the right-hand side,

$$\frac{dv_{y,h}}{dt} + \gamma v_{y,h} = 0. \tag{12}$$

Homogenous solutions will have arbitrary constants.

The particular solution will solve the same equation as the original general equation

$$\frac{dv_{y,p}}{dt} + \gamma v_{y,p} = -g. \tag{13}$$

However, we don't need find one with arbitrary constants. Hence, it is called a **particular** solution.

The sum of the two,

$$v_y = v_{y,p} + v_{y,h}, \tag{14}$$

is a solution of the total equation because of the linear nature of the differential equation. One has now found a *general* solution encompassing all solutions, because it both satisfies the general equation (like the particular solution), and has an arbitrary constant that can be adjusted to fit any initial condition (like the homogneous solution). If the equation were not linear, e.g if there were a term such as v_y^2 or $v_y\dot{v}_y$, this technique would not work.

More details

Returning to the example above, the homogenous solution is the same as that for v_x, because there was no gravitational acceleration in that case,

$$v_{y,h} = Be^{-\gamma t}. \tag{15}$$

In this case a particular solution is one with constant velocity,

$$v_{y,p} = -g/\gamma. \tag{16}$$

Note that this is the terminal velocity of a particle falling from a great height. The general solution is thus,

$$v_y = Be^{-\gamma t} - g/\gamma, \tag{17}$$

and one can find B from the initial velocity,

$$v_{0y} = B - g/\gamma, \quad B = v_{0y} + g/\gamma. \tag{18}$$

Plugging in the expression for B gives the y motion given the initial velocity,

$$v_y = (v_{0y} + g/\gamma)e^{-\gamma t} - g/\gamma. \tag{19}$$

It is easy to see that this solution has $v_y = v_{0y}$ when $t = 0$ and $v_y = -g/\gamma$ when $t \to \infty$.

One can also integrate the two equations to find the coordinates x and y as functions of t,

$$\begin{aligned} x &= \int_0^t dt'\ v_{0x}(t') = \frac{v_{0x}}{\gamma}\left(1 - e^{-\gamma t}\right), \\ y &= \int_0^t dt'\ v_{0y}(t') = -\frac{gt}{\gamma} + \frac{v_{0y} + g/\gamma}{\gamma}\left(1 - e^{-\gamma t}\right). \end{aligned} \tag{20}$$

If the question was to find the position at a time t, we would be finished. However, the more common goal in a projectile equation problem is to find the range, i.e. the distance x at which y returns to zero. For the case without a drag force this was much simpler. The solution for the y coordinate would have been $y = v_{0y}t - gt^2/2$. One would solve for t to make $y = 0$, which would be $t = 2v_{0y}/g$, then plug that value for t into $x = v_{0x}t$ to find $x = 2v_{0x}v_{0y}/g = v_0 \sin(2\theta_0)/g$. One follows the same steps here, except that the expression for $y(t)$ is more complicated. Searching for the time where $y = 0$, and we get

$$0 = -\frac{gt}{\gamma} + \frac{v_{0y} + g/\gamma}{\gamma}\left(1 - e^{-\gamma t}\right). \tag{21}$$

This cannot be inverted into a simple expression $t = \cdots$. Such expressions are known as "transcendental equations", and are not the rare instance, but are the norm. In the days before computers, one might plot the right-hand side of

the above graphically as a function of time, then find the point where it crosses zero.

Now, the most common way to solve for an equation of the above type would be to apply Newton's method numerically. This involves the following algorithm for finding solutions of some equation $F(t) = 0$.

1. First guess a value for the time, $t_{\rm guess}$.
2. Calculate F and its derivative, $F(t_{\rm guess})$ and $F'(t_{\rm guess})$.
3. Unless you guessed perfectly, $F \neq 0$, and assuming that $\Delta F \approx F' \Delta t$, one would choose
4. $\Delta t = -F(t_{\rm guess})/F'(t_{\rm guess})$.
5. Now repeat step 1, but with $t_{\rm guess} \rightarrow t_{\rm guess} + \Delta t$.

If the $F(t)$ were perfectly linear in t, one would find t in one step. Instead, one typically finds a value of t that is closer to the final answer than $t_{\rm guess}$. One breaks the loop once one finds F within some acceptable tolerance of zero. A program to do this will be added shortly.

Motion in a Magnetic Field

Another example of a velocity-dependent force is magnetism,

$$\begin{aligned} \boldsymbol{F} &= q\boldsymbol{v} \times \boldsymbol{B}, \\ F_i &= q \sum_{jk} \epsilon_{ijk} v_j B_k. \end{aligned} \tag{22}$$

For a uniform field in the z direction $\boldsymbol{B} = B\hat{z}$, the force can only have x and y components,

$$\begin{aligned} F_x &= qBv_y \\ F_y &= -qBv_x. \end{aligned} \tag{23}$$

The differential equations are

$$\begin{aligned} \dot{v}_x &= \omega_c v_y, \omega_c = qB/m \\ \dot{v}_y &= -\omega_c v_x. \end{aligned} \tag{24}$$

One can solve the equations by taking time derivatives of either equation, then substituting into the other equation,

$$\begin{aligned} \ddot{v}_x = \omega_c \dot{v_y} = -\omega_c^2 v_x, \\ \ddot{v}_y &= -\omega_c \dot{v}_x = -\omega_c v_y. \end{aligned} \tag{25}$$

The solution to these equations can be seen by inspection,

$$\begin{aligned} v_x &= A\sin(\omega_c t + \phi), \\ v_y &= A\cos(\omega_c t + \phi). \end{aligned} \tag{26}$$

One can integrate the equations to find the positions as a function of time,

$$\begin{aligned} x - x_0 &= \int_{x_0}^{x} dx = \int_0^t dt v(t) \\ &= \frac{-A}{\omega_c}\cos(\omega_c t + \phi), \\ y - y_0 &= \frac{A}{\omega_c}\sin(\omega_c t + \phi). \end{aligned} \tag{27}$$

The trajectory is a circle centered at x_0, y_0 with amplitude A rotating in the clockwise direction.

The equations of motion for the z motion are

$$\dot{v_z} = 0, \tag{28}$$

which leads to

$$z - z_0 = V_z t. \tag{29}$$

Added onto the circle, the motion is helical.

Note that the kinetic energy,

$$T = \frac{1}{2}m(v_x^2 + v_y^2 + v_z^2) = \frac{1}{2}m(\omega_c^2 A^2 + V_z^2), \tag{30}$$

is constant. This is because the force is perpendicular to the velocity, so that in any differential time element dt the work done on the particle $\boldsymbol{F}\cdot dr = dt\boldsymbol{F}\cdot v = 0$.

One should think about the implications of a velocity dependent force. Suppose one had a constant magnetic field in deep space. If a particle came through with velocity v_0, it would undergo cyclotron motion with radius $R = v_0/\omega_c$. However, if it were still its motion would remain fixed. Now, suppose an observer looked at the particle in one reference frame where the particle was moving, then changed their velocity so that the particle's velocity appeared to be zero. The motion would change from circular to fixed. Is this possible?

The solution to the puzzle above relies on understanding relativity. Imagine that the first observer believes $\boldsymbol{B} \neq 0$ and that the electric field $\boldsymbol{E} = 0$. If the observer then changes reference frames by accelerating to a velocity $\boldsymbol{v}$, in the new frame $\boldsymbol{B}$ and $\boldsymbol{E}$ both change. If the observer moved to the frame where the charge, originally moving with a small velocity v, is now at rest, the new electric field is indeed $\boldsymbol{v}\times\boldsymbol{B}$, which then leads to the same acceleration as one had before. If the velocity is not small compared to the speed of light, additional γ factors come into play, $\gamma = 1/\sqrt{1-(v/c)^2}$. Relativistic motion will not be considered in this course.

Sliding Block tied to a Wall

Another classical case is that of simple harmonic oscillations, here represented by a block sliding on a horizontal frictionless surface. The block is tied to a wall with a spring. If the spring is not compressed or stretched too far, the force on the block at a given position x is

$$F = -kx.$$

The negative sign means that the force acts to restore the object to an equilibrium position. Newton's equation of motion for this idealized system is then

$$m\frac{d^2x}{dt^2} = -kx,$$

or we could rephrase it as

$$\frac{d^2x}{dt^2} = -\frac{k}{m}x = -\omega_0^2 x,$$

with the angular frequency $\omega_0^2 = k/m$.

The above differential equation has the advantage that it can be solved analytically with solutions on the form

$$x(t) = Acos(\omega_0 t + \nu),$$

where A is the amplitude and ν the phase constant. This provides in turn an important test for the numerical solution and the development of a program for more complicated cases which cannot be solved analytically.

With the position $x(t)$ and the velocity $v(t) = dx/dt$ we can reformulate Newton's equation in the following way

$$\frac{dx(t)}{dt} = v(t),$$

and

$$\frac{dv(t)}{dt} = -\omega_0^2 x(t).$$

We are now going to solve these equations using first the standard forward Euler method. Later we will try to improve upon this.

Before proceeding however, it is important to note that in addition to the exact solution, we have at least two further tests which can be used to check our solution.

Since functions like *cos* are periodic with a period 2π, then the solution $x(t)$ has also to be periodic. This means that

$$x(t + T) = x(t),$$

with T the period defined as

$$T = \frac{2\pi}{\omega_0} = \frac{2\pi}{\sqrt{k/m}}.$$

Observe that T depends only on k/m and not on the amplitude of the solution.
In addition to the periodicity test, the total energy has also to be conserved. Suppose we choose the initial conditions

$$x(t=0) = 1 \text{ m} \qquad v(t=0) = 0 \text{ m/s},$$

meaning that block is at rest at $t = 0$ but with a potential energy

$$E_0 = \frac{1}{2}kx(t=0)^2 = \frac{1}{2}k.$$

The total energy at any time t has however to be conserved, meaning that our solution has to fulfil the condition

$$E_0 = \frac{1}{2}kx(t)^2 + \frac{1}{2}mv(t)^2.$$

We will derive this equation in our discussion on energy conservation.
An algorithm which implements these equations is included below.

- Choose the initial position and speed, with the most common choice $v(t=0) = 0$ and some fixed value for the position.
- Choose the method you wish to employ in solving the problem.
- Subdivide the time interval $[t_i, t_f]$ into a grid with step size

$$h = \frac{t_f - t_i}{N},$$

where N is the number of mesh points.

- Calculate now the total energy given by

$$E_0 = \frac{1}{2}kx(t=0)^2 = \frac{1}{2}k.$$

- Choose ODE solver to obtain x_{i+1} and v_{i+1} starting from the previous values x_i and v_i.
- When we have computed $x(v)_{i+1}$ we upgrade $t_{i+1} = t_i + h$.
- This iterative process continues till we reach the maximum time t_f.
- The results are checked against the exact solution. Furthermore, one has to check the stability of the numerical solution against the chosen number of mesh points N.

The following python program (code will be added shortly)

```
#
# This program solves Newtons equation for a block sliding on
# an horizontal frictionless surface.
# The block is tied to the wall with a spring, so N's eq takes the form:
#
#  m d^2x/dt^2 = - kx
#
# In order to make the solution dimless, we set k/m = 1.
# This results in two coupled diff. eq's that may be written as:
#
#  dx/dt = v
#  dv/dt = -x
#
# The user has to specify the initial velocity and position,
# and the number of steps. The time interval is fixed to
# t \in [0, 4\pi) (two periods)
#
```

The classical pendulum and scaling the equations

The angular equation of motion of the pendulum is given by Newton's equation and with no external force it reads

$$ml\frac{d^2\theta}{dt^2} + mgsin(\theta) = 0, \tag{31}$$

with an angular velocity and acceleration given by

$$v = l\frac{d\theta}{dt}, \tag{32}$$

and

$$a = l\frac{d^2\theta}{dt^2}. \tag{33}$$

More on the Pendulum

We do however expect that the motion will gradually come to an end due a viscous drag torque acting on the pendulum. In the presence of the drag, the above equation becomes

$$ml\frac{d^2\theta}{dt^2} + \nu\frac{d\theta}{dt} + mgsin(\theta) = 0, \tag{34}$$

where ν is now a positive constant parameterizing the viscosity of the medium in question. In order to maintain the motion against viscosity, it is necessary to add some external driving force. We choose here a periodic driving force. The last equation becomes then

$$ml\frac{d^2\theta}{dt^2} + \nu\frac{d\theta}{dt} + mgsin(\theta) = Asin(\omega t), \tag{35}$$

with A and ω two constants representing the amplitude and the angular frequency respectively. The latter is called the driving frequency.

More on the Pendulum

We define

$$\omega_0 = \sqrt{g/l},$$

the so-called natural frequency and the new dimensionless quantities

$$\hat{t} = \omega_0 t,$$

with the dimensionless driving frequency

$$\hat{\omega} = \frac{\omega}{\omega_0},$$

and introducing the quantity Q, called the *quality factor*,

$$Q = \frac{mg}{\omega_0 \nu},$$

and the dimensionless amplitude

$$\hat{A} = \frac{A}{mg}$$

We have

$$\frac{d^2\theta}{d\hat{t}^2} + \frac{1}{Q}\frac{d\theta}{d\hat{t}} + sin(\theta) = \hat{A}cos(\hat{\omega}\hat{t}).$$

This equation can in turn be recast in terms of two coupled first-order differential equations as follows

$$\frac{d\theta}{d\hat{t}} = \hat{v},$$

and

$$\frac{d\hat{v}}{d\hat{t}} = -\frac{\hat{v}}{Q} - sin(\theta) + \hat{A}cos(\hat{\omega}\hat{t}).$$

These are the equations to be solved. The factor Q represents the number of oscillations of the undriven system that must occur before its energy is significantly reduced due to the viscous drag. The amplitude $\hat{A}$ is measured in units of the maximum possible gravitational torque while $\hat{\omega}$ is the angular frequency of the external torque measured in units of the pendulum's natural frequency.

Work, Energy, Momentum and Conservation laws

Energy conservation is most convenient as a strategy for addressing problems where time does not appear. For example, a particle goes from position x_0 with speed v_0, to position x_f; what is its new speed? However, it can also be applied to problems where time does appear, such as in solving for the trajectory $x(t)$, or equivalently $t(x)$.

More material to be added here.

Energy Conservation

Energy is conserved in the case where the potential energy, $V(\boldsymbol{r})$, depends only on position, and not on time. The force is determined by V,

$$\boldsymbol{F}(\boldsymbol{r}) = -\nabla V(\boldsymbol{r}). \tag{36}$$

The net energy, $E = V + K$ where K is the kinetic energy, is then conserved,

$$\begin{aligned} \frac{d}{dt}(K+V) &= \frac{d}{dt}\left(\frac{m}{2}(v_x^2+v_y^2+v_z^2) + V(\boldsymbol{r})\right) \\ &= m\left(v_x\frac{dv_x}{dt} + v_y\frac{dv_y}{dt} + v_z\frac{dv_z}{dt}\right) + \partial_x V\frac{dx}{dt} + \partial_y V\frac{dy}{dt} + \partial_z V\frac{dz}{dt} \\ &= v_xF_x + v_yF_y + v_zF_z - F_xv_x - F_yv_y - F_zv_z = 0. \end{aligned} \tag{37}$$

The same proof can be written more compactly with vector notation,

$$\begin{aligned} \frac{d}{dt}\left(\frac{m}{2}v^2 + V(\boldsymbol{r})\right) &= m\boldsymbol{v}\cdot\dot{\boldsymbol{v}} + \nabla V(\boldsymbol{r})\cdot\dot{\boldsymbol{r}} \\ &= \boldsymbol{v}\cdot\boldsymbol{F} - \boldsymbol{F}\cdot\boldsymbol{v} = 0. \end{aligned} \tag{38}$$

Inverting the expression for kinetic energy,

$$v = \sqrt{2K/m} = \sqrt{2(E-V)/m}, \tag{39}$$

allows one to solve for the one-dimensional trajectory $x(t)$, by finding $t(x)$,

$$t = \int_{x_0}^{x}\frac{dx'}{v(x')} = \int_{x_0}^{x}\frac{dx'}{\sqrt{2(E-V(x'))/m}}. \tag{40}$$

Note this would be much more difficult in higher dimensions, because you would have to determine which points, x, y, z, the particles might reach in the trajectory, whereas in one dimension you can typically tell by simply seeing whether the kinetic energy is positive at every point between the old position and the new position.

Consider a simple harmonic oscillator potential, $V(x) = kx^2/2$, with a particle emitted from $x = 0$ with velocity v_0. Solve for the trajectory $t(x)$,

$$\begin{aligned} t &= \int_0^x \frac{dx'}{\sqrt{2(E-kx^2/2)/m}} \\ &= \sqrt{m/k}\int_0^x \frac{dx'}{\sqrt{x_{\rm max}^2 - x'^2}}, \quad x_{\rm max}^2 = 2E/k. \end{aligned} \tag{41}$$

Here $E = mv_0^2/2$ and $x_{\rm max}$ is defined as the maximum displacement before the particle turns around. This integral is done by the substitution $\sin\theta = x/x_{\rm max}$.

$$\begin{aligned} (k/m)^{1/2}t &= \sin^{-1}(x/x_{\rm max}), \\ x &= x_{\rm max}\sin\omega t, \quad \omega = \sqrt{k/m}. \end{aligned} \tag{42}$$

Conservation of Momentum

Newton's third law which we met earlier states that **For every action there is an equal and opposite reaction**, is more accurately stated as **If two bodies exert forces on each other, these forces are equal in magnitude and opposite in direction**.

This means that for two bodies i and j, if the force on i due to j is called $\boldsymbol{F}_{ij}$, then

$$\boldsymbol{F}_{ij} = -\boldsymbol{F}_{ji}. \tag{43}$$

Newton's second law, $\boldsymbol{F} = m\boldsymbol{a}$, can be written for a particle i as

$$\boldsymbol{F}_i = \sum_{j \neq i} \boldsymbol{F}_{ij} = m_i \boldsymbol{a}_i, \tag{44}$$

where $\boldsymbol{F}_i$ (a single subscript) denotes the net force acting on i. Because the mass of i is fixed, one can see that

$$\boldsymbol{F}_i = \frac{d}{dt} m_i \boldsymbol{v}_i = \sum_{j \neq i} \boldsymbol{F}_{ij}. \tag{45}$$

Now, one can sum over all the particles and obtain

$$\begin{aligned} \frac{d}{dt} \sum_i m_i v_i &= \sum_{ij, i \neq j} \boldsymbol{F}_{ij} \\ &= 0. \end{aligned} \tag{46}$$

The last step made use of the fact that for every term ij, there is an equivalent term ji with opposite force. Because the momentum is defined as $m\boldsymbol{v}$, for a system of particles,

$$\frac{d}{dt} \sum_i m_i \boldsymbol{v}_i = 0, \quad \text{for isolated particles.} \tag{47}$$

By "isolated" one means that the only force acting on any particle i are those originating from other particles in the sum, i.e. "no external" forces. Thus, Newton's third law leads to the conservation of total momentum,

$$\begin{aligned} \boldsymbol{P} &= \sum_i m_i \boldsymbol{v}_i, \\ \frac{d}{dt} \boldsymbol{P} &= 0. \end{aligned} \tag{48}$$

Consider the rocket of mass M moving with velocity v. After a brief instant, the velocity of the rocket is $v + \Delta v$ and the mass is $M - \Delta M$. Momentum conservation gives

$$\begin{aligned} Mv &= (M-\Delta M)(v+\Delta v)+\Delta M(v-v_e) \\ 0 &= -\Delta M v + M\Delta v + \Delta M(v-v_e), \\ 0 &= M\Delta v - \Delta M v_e. \end{aligned}$$

In the second step we ignored the term $\Delta M \Delta v$ because it is doubly small. The last equation gives

$$\begin{aligned} \Delta v &= \frac{v_e}{M}\Delta M, \\ \frac{dv}{dt} &= \frac{v_e}{M}\frac{dM}{dt}. \end{aligned} \tag{49}$$

Integrating the expression with lower limits $v_0 = 0$ and M_0, one finds

$$\begin{aligned} v &= v_e \int_{M_0}^{M} \frac{dM'}{M'} \\ v &= -v_e \ln(M/M_0) \\ &= -v_e \ln[(M_0 - \alpha t)/M_0]. \end{aligned}$$

Because the total momentum of an isolated system is constant, one can also quickly see that the center of mass of an isolated system is also constant. The center of mass is the average position of a set of masses weighted by the mass,

$$\bar{x} = \frac{\sum_i m_i x_i}{\sum_i m_i}. \tag{50}$$

The rate of change of $\bar{x}$ is

$$\dot{\bar{x}} = \frac{1}{M}\sum_i m_i \dot{x}_i = \frac{1}{M} P_x. \tag{51}$$

Thus if the total momentum is constant the center of mass moves at a constant velocity, and if the total momentum is zero the center of mass is fixed.

Conservation of Angular Momentum

Consider a case where the force always points radially,

$$\boldsymbol{F}(\boldsymbol{r}) = F(r)\hat{r}, \tag{52}$$

where $\hat{r}$ is a unit vector pointing outward from the origin. The angular momentum is defined as

$$\boldsymbol{L} = \boldsymbol{r} \times \boldsymbol{p} = m\boldsymbol{r} \times \boldsymbol{v}. \tag{53}$$

The rate of change of the angular momentum is

$$\begin{aligned} \frac{d\boldsymbol{L}}{dt} &= m\boldsymbol{v}\times\boldsymbol{v} + m\boldsymbol{r}\times\dot{\boldsymbol{v}} \\ &= m\boldsymbol{v}\times\boldsymbol{v} + \boldsymbol{r}\times\boldsymbol{F} = 0. \end{aligned} \tag{54}$$

The first term is zero because $\boldsymbol{v}$ is parallel to itself, and the second term is zero because $\boldsymbol{F}$ is parallel to $\boldsymbol{r}$.

As an aside, one can see from the Levi-Civita symbol that the cross product of a vector with itself is zero. Here, we consider a vector

$$\begin{aligned} \boldsymbol{V} &= \boldsymbol{A}\times\boldsymbol{A}, \\ V_i &= (\boldsymbol{A}\times\boldsymbol{A})_i = \sum_{jk}\epsilon_{ijk}A_jA_k. \end{aligned} \tag{55}$$

For any term i, there are two contributions. For example, for i denoting the x direction, either j denotes the y direction and k denotes the z direction, or vice versa, so

$$V_1 = \epsilon_{123}A_2A_3 + \epsilon_{132}A_3A_2. \tag{56}$$

This is zero by the antisymmetry of ϵ under permutations.

If the force is not radial, $\boldsymbol{r}\times\boldsymbol{F}\neq 0$ as above, and angular momentum is no longer conserved,

$$\frac{d\boldsymbol{L}}{dt} = \boldsymbol{r}\times\boldsymbol{F} \equiv \boldsymbol{\tau}, \tag{57}$$

where $\boldsymbol{\tau}$ is the torque.

For a system of isolated particles, one can write

$$\begin{aligned} \frac{d}{dt}\sum_i \boldsymbol{L}_i &= \sum_{i\neq j}\boldsymbol{r}_i\times\boldsymbol{F}_{ij} \\ &= \frac{1}{2}\sum_{i\neq j}\boldsymbol{r}_i\times\boldsymbol{F}_{ij} + \boldsymbol{r}_j\times\boldsymbol{F}_{ji} \\ &= \frac{1}{2}\sum_{i\neq j}(\boldsymbol{r}_i-\boldsymbol{r}_j)\times\boldsymbol{F}_{ij} = 0, \end{aligned} \tag{58}$$

where the last step used Newton's third law, $\boldsymbol{F}_{ij} = -\boldsymbol{F}_{ji}$. If the forces between the particles are radial, i.e. $\boldsymbol{F}_{ij} \mid\mid (\boldsymbol{r}_i - \boldsymbol{r}_j)$, then each term in the sum is zero and the net angular momentum is fixed. Otherwise, you could imagine an isolated system that would start spinning spontaneously.

One can write the torque about a given axis, which we will denote as $\hat{z}$, in polar coordinates, where

$$x = r\sin\theta\cos\phi, \quad y = r\sin\theta\cos\phi, \quad z = r\cos\theta, \tag{59}$$

to find the z component of the torque,

$$\begin{aligned} \tau_z &= xF_y - yF_x \\ &= -r\sin\theta\left\{\cos\phi\partial_y - \sin\phi\partial_x\right\}V(x,y,z). \end{aligned} \tag{60}$$

One can use the chain rule to write the partial derivative w.r.t. ϕ (keeping r and θ fixed),

$$\begin{aligned} \partial_\phi &= \frac{\partial x}{\partial\phi}\partial_x + \frac{\partial_y}{\partial\phi}\partial_y + \frac{\partial z}{\partial\phi}\partial_z \\ &= -r\sin\theta\sin\phi\partial_x + \sin\theta\cos\phi\partial_y. \end{aligned} \tag{61}$$

Combining the two equations,

$$\tau_z = -\partial_\phi V(r,\theta,\phi). \tag{62}$$

Thus, if the potential is independent of the azimuthal angle ϕ, there is no torque about the z axis and L_z is conserved.

Symmetries and Conservation Laws

When we derived the conservation of energy, we assumed that the potential depended only on position, not on time. If it depended explicitly on time, one can quickly see that the energy would have changed at a rate $\partial_t V(x,y,z,t)$. Note that if there is no explicit dependence on time, i.e. $V(x,y,z)$, the potential energy can depend on time through the variations of x, y, z with time. However, that variation does not lead to energy non-conservation. Further, we just saw that if a potential does not depend on the azimuthal angle about some axis, ϕ, that the angular momentum about that axis is conserved.

Now, we relate momentum conservation to translational invariance. Considering a system of particles with positions, $\boldsymbol{r}_i$, if one changed the coordinate system by a translation by a differential distance $\boldsymbol{\epsilon}$, the net potential would change by

$$\begin{aligned} \delta V(\boldsymbol{r}_1, \boldsymbol{r}_2 \cdots) &= \sum_i \boldsymbol{\epsilon}\cdot\nabla_i V(\boldsymbol{r}_1, \boldsymbol{r}_2, \cdots) \\ &= -\sum_i \boldsymbol{\epsilon}\cdot\boldsymbol{F}_i \\ &= -\frac{d}{dt}\sum_i \boldsymbol{\epsilon}\cdot\boldsymbol{p}_i. \end{aligned} \tag{63}$$

Thus, if the potential is unchanged by a translation of the coordinate system, the total momentum is conserved. If the potential is translationally invariant in a given direction, defined by a unit vector, $\hat{\epsilon}$ in the $\boldsymbol{\epsilon}$ direction, one can see that

$$\hat{\epsilon} \cdot \nabla_i V(\boldsymbol{r}_i) \quad = \quad 0. \tag{64}$$

The component of the total momentum along that axis is conserved. This is rather obvious for a single particle. If $V(\boldsymbol{r})$ does not depend on some coordinate x, then the force in the x direction is $F_x = -\partial_x V = 0$, and momentum along the x direction is constant.

We showed how the total momentum of an isolated system of particle was conserved, even if the particles feel internal forces in all directions. In that case the potential energy could be written

$$V = \sum_{i,j \leq i} V_{ij}(\boldsymbol{r}_i - \boldsymbol{r}_j). \tag{65}$$

In this case, a translation leads to $\boldsymbol{r}_i \rightarrow \boldsymbol{r}_i + \boldsymbol{\epsilon}$, with the translation equally affecting the coordinates of each particle. Because the potential depends only on the relative coordinates, δV is manifestly zero. If one were to go through the exercise of calculating δV for small $\boldsymbol{\epsilon}$, one would find that the term $\nabla_i V(\boldsymbol{r}_i - \boldsymbol{r}_j)$ would be canceled by the term $\nabla_j V(\boldsymbol{r}_i - \boldsymbol{r}_j)$.

The relation between symmetries of the potential and conserved quantities (also called constants of motion) is one of the most profound concepts one should gain from this course. It plays a critical role in all fields of physics. This is especially true in quantum mechanics, where a quantity A is conserved if its operator commutes with the Hamiltonian. For example if the momentum operator $-i\hbar\partial_x$ commutes with the Hamiltonian, momentum is conserved, and clearly this operator commutes if the Hamiltonian (which represents the total energy, not just the potential) does not depend on x. Also in quantum mechanics the angular momentum operator is $L_z = -i\hbar\partial_\phi$. In fact, if the potential is unchanged by rotations about some axis, angular momentum about that axis is conserved. We return to this concept, from a more formal perspective, later in the course when Lagrangian mechanics is presented.

Bulding a code for the Earth-Sun system

We will now venture into a study of a system which is energy conserving. The aim is to see if we (since it is not possible to solve the general equations analytically) we can develop stable numerical algorithms whose results we can trust!

We solve the equations of motion numerically. We will also compute quantities like the energy numerically.

We start with a simpler case first, the Earth-Sun system in two dimensions only. The gravitational force F_G on the earth from the sun is

$$\boldsymbol{F}_G = -\frac{GM_\odot M_E}{r^3}\boldsymbol{r},$$

where G is the gravitational constant,

$$M_E = 6 \times 10^{24}\text{Kg},$$

the mass of Earth,

$$M_\odot = 2 \times 10^{30}\text{Kg},$$

the mass of the Sun and

$$r = 1.5 \times 10^{11}\text{m},$$

is the distance between Earth and the Sun. The latter defines what we call an astronomical unit **AU**. From Newton's second law we have then for the x direction

$$\frac{d^2x}{dt^2} = -\frac{F_x}{M_E},$$

and

$$\frac{d^2y}{dt^2} = -\frac{F_y}{M_E},$$

for the y direction.

Here we will use that $x = r\cos(\theta)$, $y = r\sin(\theta)$ and

$$r = \sqrt{x^2 + y^2}.$$

We can rewrite

$$F_x = -\frac{GM_\odot M_E}{r^2}\cos(\theta) = -\frac{GM_\odot M_E}{r^3}x,$$

and

$$F_y = -\frac{GM_\odot M_E}{r^2}\sin(\theta) = -\frac{GM_\odot M_E}{r^3}y,$$

for the y direction.

We can rewrite these two equations

$$F_x = -\frac{GM_\odot M_E}{r^2}\cos(\theta) = -\frac{GM_\odot M_E}{r^3}x,$$

and

$$F_y = -\frac{GM_\odot M_E}{r^2}\sin(\theta) = -\frac{GM_\odot M_E}{r^3}y,$$

as four first-order coupled differential equations

$$\frac{dv_x}{dt} = -\frac{GM_\odot}{r^3}x,$$

$$\frac{dx}{dt} = v_x,$$

$$\frac{dv_y}{dt} = -\frac{GM_\odot}{r^3}y,$$

$$\frac{dy}{dt} = v_y.$$

Building a code for the solar system, final coupled equations

The four coupled differential equations

$$\frac{dv_x}{dt} = -\frac{GM_\odot}{r^3}x,$$

$$\frac{dx}{dt} = v_x,$$

$$\frac{dv_y}{dt} = -\frac{GM_\odot}{r^3}y,$$

$$\frac{dy}{dt} = v_y,$$

can be turned into dimensionless equations or we can introduce astronomical units with 1 AU $= 1.5 \times 10^{11}$.

Using the equations from circular motion (with $r = 1$AU)

$$\frac{M_E v^2}{r} = F = \frac{GM_\odot M_E}{r^2},$$

we have

$$GM_\odot = v^2 r,$$

and using that the velocity of Earth (assuming circular motion) is $v = 2\pi r/\text{yr} = 2\pi\text{AU/yr}$, we have

$$GM_\odot = v^2 r = 4\pi^2 \frac{(\text{AU})^3}{\text{yr}^2}.$$

Building a code for the solar system, discretized equations

The four coupled differential equations can then be discretized using Euler's method as (with step length h)

$$v_{x,i+1} = v_{x,i} - h\frac{4\pi^2}{r_i^3}x_i,$$

$$x_{i+1} = x_i + hv_{x,i},$$

$$v_{y,i+1} = v_{y,i} - h\frac{4\pi^2}{r_i^3}y_i,$$

$$y_{i+1} = y_i + hv_{y,i},$$

Code Example with Euler's Method

The code here implements Euler's method for the Earth-Sun system using a more compact way of representing the vectors. Alternatively, you could have spelled out all the variables v_x, v_y, x and y as one-dimensional arrays.

```
# Common imports
import numpy as np
import pandas as pd
from math import *
import matplotlib.pyplot as plt
import os

# Where to save the figures and data files
PROJECT_ROOT_DIR = "Results"
FIGURE_ID = "Results/FigureFiles"
DATA_ID = "DataFiles/"

if not os.path.exists(PROJECT_ROOT_DIR):
    os.mkdir(PROJECT_ROOT_DIR)

if not os.path.exists(FIGURE_ID):
    os.makedirs(FIGURE_ID)

if not os.path.exists(DATA_ID):
    os.makedirs(DATA_ID)

def image_path(fig_id):
    return os.path.join(FIGURE_ID, fig_id)

def data_path(dat_id):
    return os.path.join(DATA_ID, dat_id)

def save_fig(fig_id):
    plt.savefig(image_path(fig_id) + ".png", format='png')


DeltaT = 0.001
#set up arrays
tfinal = 10 # in years
n = ceil(tfinal/DeltaT)
# set up arrays for t, a, v, and x
t = np.zeros(n)
v = np.zeros((n,2))
r = np.zeros((n,2))
# Initial conditions as compact 2-dimensional arrays
r0 = np.array([1.0,0.0])
v0 = np.array([0.0,2*pi])
r[0] = r0
v[0] = v0
Fourpi2 = 4*pi*pi
# Start integrating using Euler's method
for i in range(n-1):
    # Set up the acceleration
    # Here you could have defined your own function for this
    rabs = sqrt(sum(r[i]*r[i]))
    a =  -Fourpi2*r[i]/(rabs**3)
    # update velocity, time and position using Euler's forward method
    v[i+1] = v[i] + DeltaT*a
    r[i+1] = r[i] + DeltaT*v[i]
```

```
    t[i+1] = t[i] + DeltaT
# Plot position as function of time
fig, ax = plt.subplots()
#ax.set_xlim(0, tfinal)
ax.set_ylabel('x[m]')
ax.set_xlabel('y[m]')
ax.plot(r[:,0], r[:,1])
fig.tight_layout()
save_fig("EarthSunEuler")
plt.show()
```

Problems with Euler's Method

We notice here that Euler's method doesn't give a stable orbit. It means that we cannot trust Euler's method. In a deeper way, as we will see in homework 5, Euler's method does not conserve energy. It is an example of an integrator which is not symplectic.

Here we present thus two methods, which with simple changes allow us to avoid these pitfalls. The simplest possible extension is the so-called Euler-Cromer method. The changes we need to make to our code are indeed marginal here. We need simply to replace

```
r[i+1] = r[i] + DeltaT*v[i]
```

in the above code with the velocity at the new time t_{i+1}

```
r[i+1] = r[i] + DeltaT*v[i+1]
```

By this simple caveat we get stable orbits. Below we derive the Euler-Cromer method as well as one of the most utlized algorithms for sovling the above type of problems, the so-called Velocity-Verlet method.

Deriving the Euler-Cromer Method

Let us repeat Euler's method. We have a differential equation

$$y'(t_i) = f(t_i, y_i) \tag{66}$$

and if we truncate at the first derivative, we have from the Taylor expansion

$$y_{i+1} = y(t_i) + (\Delta t) f(t_i, y_i) + O(\Delta t^2), \tag{67}$$

which when complemented with $t_{i+1} = t_i + \Delta t$ forms the algorithm for the well-known Euler method. Note that at every step we make an approximation error of the order of $O(\Delta t^2)$, however the total error is the sum over all steps $N = (b-a)/(\Delta t)$ for $t \in [a, b]$, yielding thus a global error which goes like $NO(\Delta t^2) \approx O(\Delta t)$.

To make Euler's method more precise we can obviously decrease Δt (increase N), but this can lead to loss of numerical precision. Euler's method is not recommended for precision calculation, although it is handy to use in order to get a first view on how a solution may look like.

Euler's method is asymmetric in time, since it uses information about the derivative at the beginning of the time interval. This means that we evaluate the position at y_1 using the velocity at v_0. A simple variation is to determine x_{n+1} using the velocity at v_{n+1}, that is (in a slightly more generalized form)

$$y_{n+1} = y_n + v_{n+1} + O(\Delta t^2) \tag{68}$$

and

$$v_{n+1} = v_n + (\Delta t)a_n + O(\Delta t^2). \tag{69}$$

The acceleration a_n is a function of $a_n(y_n, v_n, t_n)$ and needs to be evaluated as well. This is the Euler-Cromer method.

Exercise: go back to the above code with Euler's method and add the Euler-Cromer method.

Deriving the Velocity-Verlet Method

Let us stay with x (position) and v (velocity) as the quantities we are interested in.

We have the Taylor expansion for the position given by

$$x_{i+1} = x_i + (\Delta t)v_i + \frac{(\Delta t)^2}{2}a_i + O((\Delta t)^3).$$

The corresponding expansion for the velocity is

$$v_{i+1} = v_i + (\Delta t)a_i + \frac{(\Delta t)^2}{2}v_i^{(2)} + O((\Delta t)^3).$$

Via Newton's second law we have normally an analytical expression for the derivative of the velocity, namely

$$a_i = \frac{d^2x}{dt^2}|_i = \frac{dv}{dt}|_i = \frac{F(x_i, v_i, t_i)}{m}.$$

If we add to this the corresponding expansion for the derivative of the velocity

$$v_{i+1}^{(1)} = a_{i+1} = a_i + (\Delta t)v_i^{(2)} + O((\Delta t)^2) = a_i + (\Delta t)v_i^{(2)} + O((\Delta t)^2),$$

and retain only terms up to the second derivative of the velocity since our error goes as $O(h^3)$, we have

$$(\Delta t)v_i^{(2)} \approx a_{i+1} - a_i.$$

We can then rewrite the Taylor expansion for the velocity as

$$v_{i+1} = v_i + \frac{(\Delta t)}{2}\left(a_{i+1} + a_i\right) + O((\Delta t)^3).$$

The velocity Verlet method

Our final equations for the position and the velocity become then

$$x_{i+1} = x_i + (\Delta t)v_i + \frac{(\Delta t)^2}{2}a_i + O((\Delta t)^3),$$

and

$$v_{i+1} = v_i + \frac{(\Delta t)}{2}(a_{i+1} + a_i) + O((\Delta t)^3).$$

Note well that the term a_{i+1} depends on the position at x_{i+1}. This means that you need to calculate the position at the updated time t_{i+1} before the computing the next velocity. Note also that the derivative of the velocity at the time t_i used in the updating of the position can be reused in the calculation of the velocity update as well.

Adding the Velocity-Verlet Method

We can now easily add the Verlet method to our original code as

```
DeltaT = 0.01
#set up arrays
tfinal = 10
n = ceil(tfinal/DeltaT)
# set up arrays for t, a, v, and x
t = np.zeros(n)
v = np.zeros((n,2))
r = np.zeros((n,2))
# Initial conditions as compact 2-dimensional arrays
r0 = np.array([1.0,0.0])
v0 = np.array([0.0,2*pi])
r[0] = r0
v[0] = v0
Fourpi2 = 4*pi*pi
# Start integrating using the Velocity-Verlet  method
for i in range(n-1):
    # Set up forces, air resistance FD, note now that we need the norm of the vecto
    # Here you could have defined your own function for this
    rabs = sqrt(sum(r[i]*r[i]))
    a =  -Fourpi2*r[i]/(rabs**3)
    # update velocity, time and position using the Velocity-Verlet method
    r[i+1] = r[i] + DeltaT*v[i]+0.5*(DeltaT**2)*a
    rabs = sqrt(sum(r[i+1]*r[i+1]))
    anew = -4*(pi**2)*r[i+1]/(rabs**3)
    v[i+1] = v[i] + 0.5*DeltaT*(a+anew)
    t[i+1] = t[i] + DeltaT
# Plot position as function of time
fig, ax = plt.subplots()
ax.set_ylabel('x[m]')
ax.set_xlabel('y[m]')
ax.plot(r[:,0], r[:,1])
fig.tight_layout()
save_fig("EarthSunVV")
plt.show()
```

You can easily generalize the calculation of the forces by defining a function which takes in as input the various variables. We leave this as a challenge to you.

Studying Energy Conservation

In order to study the conservation of energy, we will need to perform a numerical integration, unless we can integrate analytically. Here we present the Trapezoidal rule as a the simplest possible approximation.

Numerical Integration

It is also useful to consider methods to integrate numerically. Let us consider the following case. We have classical electron which moves in the x-direction along a surface. The force from the surface is

$$\boldsymbol{F}(x) = -F_0 \sin\left(\frac{2\pi x}{b}\right)\boldsymbol{e}_x.$$

The constant b represents the distance between atoms at the surface of the material, F_0 is a constant and x is the position of the electron. Using the work-energy theorem we can find the work W done when moving an electron from a position x_0 to a final position x through the integral

$$W = -\int_{x_0}^{x} \boldsymbol{F}(x')dx' = \int_{x_0}^{x} F_0 \sin\left(\frac{2\pi x'}{b}\right)dx',$$

which results in

$$W = \frac{F_0 b}{2\pi}\left[\cos\left(\frac{2\pi x}{b}\right) - \cos\left(\frac{2\pi x_0}{b}\right)\right].$$

Numerical Integration

There are several numerical algorithms for finding an integral numerically. The more familiar ones like the rectangular rule or the trapezoidal rule have simple geometric interpretations.

Let us look at the mathematical details of what are called equal-step methods, also known as Newton-Cotes quadrature.

Newton-Cotes Quadrature or equal-step methods

The integral

$$I = \int_a^b f(x)dx \tag{70}$$

has a very simple meaning. The integral is the area enscribed by the function $f(x)$ starting from $x = a$ to $x = b$. It is subdivided in several smaller areas whose evaluation is to be approximated by different techniques. The areas under the curve can for example be approximated by rectangular boxes or trapezoids.

Basic philosophy of equal-step methods

In considering equal step methods, our basic approach is that of approximating a function $f(x)$ with a polynomial of at most degree $N-1$, given N integration points. If our polynomial is of degree 1, the function will be approximated with $f(x) \approx a_0 + a_1 x$.

The algorithm for these integration methods is rather simple, and the number of approximations perhaps unlimited!

- Choose a step size $h = (b-a)/N$ where N is the number of steps and a and b the lower and upper limits of integration.
- With a given step length we rewrite the integral as

$$\int_a^b f(x)dx = \int_a^{a+h} f(x)dx + \int_{a+h}^{a+2h} f(x)dx + \ldots \int_{b-h}^{b} f(x)dx.$$

- The strategy then is to find a reliable polynomial approximation for $f(x)$ in the various intervals. Choosing a given approximation for $f(x)$, we obtain a specific approximation to the integral.
- With this approximation to $f(x)$ we perform the integration by computing the integrals over all subintervals.

One possible strategy then is to find a reliable polynomial expansion for $f(x)$ in the smaller subintervals. Consider for example evaluating

$$\int_a^{a+2h} f(x)dx,$$

which we rewrite as

$$\int_a^{a+2h} f(x)dx = \int_{x_0-h}^{x_0+h} f(x)dx. \tag{71}$$

We have chosen a midpoint x_0 and have defined $x_0 = a + h$.

The rectangle method

A very simple approach is the so-called midpoint or rectangle method. In this case the integration area is split in a given number of rectangles with length h and height given by the mid-point value of the function. This gives the following simple rule for approximating an integral

$$I = \int_a^b f(x)dx \approx h\sum_{i=1}^{N} f(x_{i-1/2}), \tag{72}$$

where $f(x_{i-1/2})$ is the midpoint value of f for a given rectangle. We will discuss its truncation error below. It is easy to implement this algorithm, as shown below

The correct mathematical expression for the local error for the rectangular rule $R_i(h)$ for element i is

$$\int_{-h}^{h} f(x)dx - R_i(h) = -\frac{h^3}{24}f^{(2)}(\xi),$$

and the global error reads

$$\int_{a}^{b} f(x)dx - R_h(f) = -\frac{b-a}{24}h^2 f^{(2)}(\xi),$$

where R_h is the result obtained with rectangular rule and $\xi \in [a, b]$.

We go back to our simple example above and set $F_0 = b = 1$ and choose $x_0 = 0$ and $x = 1/2$, and have

$$W = \frac{1}{\pi}.$$

The code here computes the integral using the rectangle rule and $n = 100$ integration points we have a relative error of 10^{-5}.

```
from math import sin, pi
import numpy as np
from sympy import Symbol, integrate
# function for the Rectangular rule
def Rectangular(a,b,f,n):
   h = (b-a)/float(n)
   s = 0
   for i in range(0,n,1):
       x = (i+0.5)*h
       s = s+ f(x)
   return h*s
# function to integrate
def function(x):
    return sin(2*pi*x)
# define integration limits and integration points
a = 0.0; b = 0.5;
n = 100
Exact = 1./pi
print("Relative error= ", abs( (Rectangular(a,b,function,n)-Exact)/Exact))
```

The trapezoidal rule

The other integral gives

$$\int_{x_0-h}^{x_0} f(x)dx = \frac{h}{2}\left(f(x_0) + f(x_0-h)\right) + O(h^3),$$

and adding up we obtain

$$\int_{x_0-h}^{x_0+h} f(x)dx = \frac{h}{2}\left(f(x_0+h) + 2f(x_0) + f(x_0-h)\right) + O(h^3), \qquad (73)$$

which is the well-known trapezoidal rule. Concerning the error in the approximation made, $O(h^3) = O((b-a)^3/N^3)$, you should note that this is the local

error. Since we are splitting the integral from a to b in N pieces, we will have to perform approximately N such operations.

This means that the *global error* goes like $\approx O(h^2)$. The trapezoidal reads then

$$I = \int_a^b f(x)dx = h\left(f(a)/2 + f(a+h) + f(a+2h) + \cdots + f(b-h) + f_b/2\right), \tag{74}$$

with a global error which goes like $O(h^2)$.

Hereafter we use the shorthand notations $f_{-h} = f(x_0 - h)$, $f_0 = f(x_0)$ and $f_h = f(x_0 + h)$.

The correct mathematical expression for the local error for the trapezoidal rule is

$$\int_a^b f(x)dx - \frac{b-a}{2}\left[f(a) + f(b)\right] = -\frac{h^3}{12}f^{(2)}(\xi),$$

and the global error reads

$$\int_a^b f(x)dx - T_h(f) = -\frac{b-a}{12}h^2 f^{(2)}(\xi),$$

where T_h is the trapezoidal result and $\xi \in [a, b]$.

Algorithm for the trapezoidal rule

The trapezoidal rule is easy to implement numerically through the following simple algorithm

- Choose the number of mesh points and fix the step length.
- calculate $f(a)$ and $f(b)$ and multiply with $h/2$.
- Perform a loop over $n = 1$ to $n - 1$ ($f(a)$ and $f(b)$ are known) and sum up the terms $f(a + h) + f(a + 2h) + f(a + 3h) + \cdots + f(b - h)$. Each step in the loop corresponds to a given value $a + nh$.
- Multiply the final result by h and add $hf(a)/2$ and $hf(b)/2$.

We use the same function and integrate now using the trapoezoidal rule.

```
import numpy as np
from sympy import Symbol, integrate
# function for the trapezoidal rule
def Trapez(a,b,f,n):
   h = (b-a)/float(n)
   s = 0
   x = a
   for i in range(1,n,1):
       x = x+h
       s = s+ f(x)
   s = 0.5*(f(a)+f(b)) +s
   return h*s
```

```
# function to integrate
def function(x):
    return sin(2*pi*x)
# define integration limits and integration points
a = 0.0; b = 0.5;
n = 100
Exact = 1./pi
print("Relative error= ", abs( (Trapez(a,b,function,n)-Exact)/Exact))
```

Simpsons' rule

Instead of using the above first-order polynomials approximations for f, we attempt at using a second-order polynomials. In this case we need three points in order to define a second-order polynomial approximation

$$f(x) \approx P_2(x) = a_0 + a_1 x + a_2 x^2.$$

Using again Lagrange's interpolation formula we have

$$P_2(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}y_2 + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}y_1 + \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}y_0.$$

Inserting this formula in the integral of Eq. (71) we obtain

$$\int_{-h}^{+h} f(x)dx = \frac{h}{3}\left(f_h + 4f_0 + f_{-h}\right) + O(h^5),$$

which is Simpson's rule.

Note that the improved accuracy in the evaluation of the derivatives gives a better error approximation, $O(h^5)$ vs. $O(h^3)$. But this is again the *local error approximation*. Using Simpson's rule we can easily compute the integral of Eq. (70) to be

$$I = \int_a^b f(x)dx = \frac{h}{3}\left(f(a) + 4f(a+h) + 2f(a+2h) + \cdots + 4f(b-h) + f_b\right), \tag{75}$$

with a global error which goes like $O(h^4)$.

More formal expressions for the local and global errors are for the local error

$$\int_a^b f(x)dx - \frac{b-a}{6}\left[f(a) + 4f((a+b)/2) + f(b)\right] = -\frac{h^5}{90}f^{(4)}(\xi),$$

and for the global error

$$\int_a^b f(x)dx - S_h(f) = -\frac{b-a}{180}h^4 f^{(4)}(\xi).$$

with $\xi \in [a,b]$ and S_h the results obtained with Simpson's method.

Algorithm for Simpson's rule

The method can easily be implemented numerically through the following simple algorithm

- Choose the number of mesh points and fix the step.
- calculate $f(a)$ and $f(b)$
- Perform a loop over $n = 1$ to $n - 1$ ($f(a)$ and $f(b)$ are known) and sum up the terms $4f(a + h) + 2f(a + 2h) + 4f(a + 3h) + \cdots + 4f(b - h)$. Each step in the loop corresponds to a given value $a + nh$. Odd values of n give 4 as factor while even values yield 2 as factor.
- Multiply the final result by $\frac{h}{3}$.

Code example

```
from math import sin, pi
import numpy as np
from sympy import Symbol, integrate
# function for the trapezoidal rule
def Simpson(a,b,f,n):
   h = (b-a)/float(n)
   sum = f(a)/float(2);
   for i in range(1,n):
       sum = sum + f(a+i*h)*(3+(-1)**(i+1))
   sum = sum + f(b)/float(2)
   return sum*h/3.0
# function to integrate
def function(x):
    return sin(2*pi*x)
# define integration limits and integration points
a = 0.0; b = 0.5;
n = 100
Exact = 1./pi
print("Relative error= ", abs( (Simpson(a,b,function,n)-Exact)/Exact))
```

We see that Simpson's rule gives a much better estimation of the relative error with the same amount of points as we had for the Rectangle rule and the Trapezoidal rule.

Harmonic Oscillations

The harmonic oscillator is omnipresent in physics. Although you may think of this as being related to springs, it, or an equivalent mathematical representation, appears in just about any problem where a mode is sitting near its potential energy minimum. At that point, $\partial_x V(x) = 0$, and the first non-zero term (aside from a constant) in the potential energy is that of a harmonic oscillator. In a solid, sound modes (phonons) are built on a picture of coupled harmonic oscillators, and in relativistic field theory the fundamental interactions are also

built on coupled oscillators positioned infinitesimally close to one another in space. The phenomena of a resonance of an oscillator driven at a fixed frequency plays out repeatedly in atomic, nuclear and high-energy physics, when quantum mechanically the evolution of a state oscillates according to e^{-iEt} and exciting discrete quantum states has very similar mathematics as exciting discrete states of an oscillator.

The potential energy for a single particle as a function of its position x can be written as a Taylor expansion about some point x_0

$$V(x) = V(x_0) + (x - x_0)\, \partial_x V(x)\big|_{x_0} + \frac{1}{2}(x - x_0)^2\, \partial_x^2 V(x)\big|_{x_0} + \frac{1}{3!}\, \partial_x^3 V(x)\big|_{x_0} + \cdots \tag{76}$$

If the position x_0 is at the minimum of the resonance, the first two non-zero terms of the potential are

$$\begin{aligned} V(x) &\approx V(x_0) + \frac{1}{2}(x - x_0)^2\, \partial_x^2 V(x)\big|_{x_0}\,, \qquad (77)\\ &= V(x_0) + \frac{1}{2}k(x - x_0)^2, \quad k \equiv \partial_x^2 V(x)\big|_{x_0}\,, \\ F &= -\partial_x V(x) = -k(x - x_0). \end{aligned}$$

Put into Newton's 2nd law (assuming $x_0 = 0$),

$$\begin{aligned} m\ddot{x} &= -kx, \qquad (78)\\ x &= A\cos(\omega_0 t - \phi), \quad \omega_0 = \sqrt{k/m}. \qquad (79) \end{aligned}$$

Here A and ϕ are arbitrary. Equivalently, one could have written this as $A\cos(\omega_0 t) + B\sin(\omega_0 t)$, or as the real part of $Ae^{i\omega_0 t}$. In this last case A could be an arbitrary complex constant. Thus, there are 2 arbitrary constants (either A and B or A and ϕ, or the real and imaginary part of one complex constant. This is the expectation for a second order differential equation, and also agrees with the physical expectation that if you know a particle's initial velocity and position you should be able to define its future motion, and that those two arbitrary conditions should translate to two arbitrary constants.

A key feature of harmonic motion is that the system repeats itself after a time $T = 1/f$, where f is the frequency, and $\omega = 2\pi f$ is the angular frequency. The period of the motion is independent of the amplitude. However, this independence is only exact when one can neglect higher terms of the potential, $x^3, x^4 \cdots$. Once can neglect these terms for sufficiently small amplitudes, and for larger amplitudes the motion is no longer purely sinusoidal, and even though the motion repeats itself, the time for repeating the motion is no longer independent of the amplitude.

One can also calculate the velocity and the kinetic energy as a function of time,

$$\begin{aligned} \dot{x} &= -\omega_0 A \sin(\omega_0 t - \phi), \\ K &= \frac{1}{2} m \dot{x}^2 = \frac{m\omega_0^2 A^2}{2} \sin^2(\omega_0 t - \phi), \\ &= \frac{k}{2} A^2 \sin^2(\omega_0 t - \phi). \end{aligned} \tag{80}$$

The total energy is then

$$E = K + V = \frac{1}{2} m \dot{x}^2 + \frac{1}{2} k x^2 = \frac{1}{2} k A^2. \tag{81}$$

The total energy then goes as the square of the amplitude.

A pendulum is an example of a harmonic oscillator. By expanding the kinetic and potential energies for small angles find the frequency for a pendulum of length L with all the mass m centered at the end by writing the eq.s of motion in the form of a harmonic oscillator.

The potential energy and kinetic energies are (for x being the displacement)

$$\begin{aligned} V &= mgL(1 - \cos\theta) \approx mgL \frac{x^2}{2L^2}, \\ K &= \frac{1}{2} m L^2 \dot{\theta}^2 \approx \frac{m}{2} \dot{x}^2. \end{aligned}$$

For small x Newton's 2nd law becomes

$$m\ddot{x} = -\frac{mg}{L} x,$$

and the spring constant would appear to be $k = mg/L$, which makes the frequency equal to $\omega_0 = \sqrt{g/L}$. Note that the frequency is independent of the mass.

Damped Oscillators

We consider only the case where the damping force is proportional to the velocity. This is counter to dragging friction, where the force is proportional in strength to the normal force and independent of velocity, and is also inconsistent with wind resistance, where the magnitude of the drag force is proportional the square of the velocity. Rolling resistance does seem to be mainly proportional to the velocity. However, the main motivation for considering damping forces proportional to the velocity is that the math is more friendly. This is because the differential equation is linear, i.e. each term is of order x, $\dot{x}$, $\ddot{x} \cdots$, or even terms with no mention of x, and there are no terms such as x^2 or $x\ddot{x}$. The equations of motion for a spring with damping force $-b\dot{x}$ are

$$m\ddot{x} + b\dot{x} + kx = 0. \tag{82}$$

Just to make the solution a bit less messy, we rewrite this equation as

$$\ddot{x} + 2\beta\dot{x} + \omega_0^2 x = 0, \qquad \beta \equiv b/2m,\ \omega_0 \equiv \sqrt{k/m}. \tag{83}$$

Both β and ω have dimensions of inverse time. To find solutions (see appendix C in the text) you must make an educated guess at the form of the solution. To do this, first realize that the solution will need an arbitrary normalization A because the equation is linear. Secondly, realize that if the form is

$$x = Ae^{rt} \tag{84}$$

that each derivative simply brings out an extra power of r. This means that the Ae^{rt} factors out and one can simply solve for an equation for r. Plugging this form into Eq. (83),

$$r^2 + 2\beta r + \omega_0^2 = 0. \tag{85}$$

Because this is a quadratic equation there will be two solutions,

$$r = -\beta \pm \sqrt{\beta^2 - \omega_0^2}. \tag{86}$$

We refer to the two solutions as r_1 and r_2 corresponding to the + and − roots. As expected, there should be two arbitrary constants involved in the solution,

$$x = A_1 e^{r_1 t} + A_2 e^{r_2 t}, \tag{87}$$

where the coefficients A_1 and A_2 are determined by initial conditions.

The roots listed above, $\sqrt{\omega_0^2 - \beta_0^2}$, will be imaginary if the damping is small and $\beta < \omega_0$. In that case, r is complex and the factor ert will have some oscillatory behavior. If the roots are real, there will only be exponentially decaying solutions. There are three cases:

Underdamped: $\beta < \omega_0$.

$$\begin{aligned} x &= A_1 e^{-\beta t} e^{i\omega' t} + A_2 e^{-\beta t} e^{-i\omega' t}, \quad \omega' \equiv \sqrt{\omega_0^2 - \beta^2} \\ &= (A_1 + A_2) e^{-\beta t} \cos\omega' t + i(A_1 - A_2) e^{-\beta t} \sin\omega' t. \end{aligned} \tag{88}$$

Here we have made use of the identity $e^{i\omega' t} = \cos\omega' t + i \sin\omega' t$. Because the constants are arbitrary, and because the real and imaginary parts are both solutions individually, we can simply consider the real part of the solution alone:

$$\begin{aligned} x &= B_1 e^{-\beta t} \cos\omega' t + B_2 e^{-\beta t} \sin\omega' t, \\ \omega' &\equiv \sqrt{\omega_0^2 - \beta^2}. \end{aligned} \tag{89}$$

Critical dampling: $\beta = \omega_0$**.** In this case the two terms involving r_1 and r_2 are identical because $\omega' = 0$. Because we need to arbitrary constants, there needs to be another solution. This is found by simply guessing, or by taking the limit of $\omega' \to 0$ from the underdamped solution. The solution is then

$$x = Ae^{-\beta t} + Bte^{-\beta t}. \tag{90}$$

The critically damped solution is interesting because the solution approaches zero quickly, but does not oscillate. For a problem with zero initial velocity, the solution never crosses zero. This is a good choice for designing shock absorbers or swinging doors.

Overdamped: $\beta > \omega_0$**.**

$$x = A_1 \exp -(\beta + \sqrt{\beta^2 - \omega_0^2})t + A_2 \exp -(\beta - \sqrt{\beta^2 - \omega_0^2})t \tag{91}$$

This solution will also never pass the origin more than once, and then only if the initial velocity is strong and initially toward zero.

Given b, m and ω_0, find $x(t)$ for a particle whose initial position is $x = 0$ and has initial velocity v_0 (assuming an underdamped solution).

The solution is of the form,

$$\begin{aligned} x &= e^{-\beta t}\left[A_1 \cos(\omega' t) + A_2 \sin \omega' t\right], \\ \dot{x} &= -\beta x + \omega' e^{-\beta t}\left[-A_1 \sin \omega' t + A_2 \cos \omega' t\right]. \\ \omega' &\equiv \sqrt{\omega_0^2 - \beta^2}, \quad \beta \equiv b/2m. \end{aligned}$$

From the initial conditions, $A_1 = 0$ because $x(0) = 0$ and $\omega' A_2 = v_0$. So

$$x = \frac{v_0}{\omega'} e^{-\beta t} \sin \omega' t.$$

Here we study first the case without additional friction term and scale our equation in terms of a dimensionless time τ.

Let us remind ourselves about the differential equation we want to solve (the general case with damping due to friction)

$$m\frac{d^2x}{dt^2} + b\frac{dx}{dt} + kx(t) = 0.$$

We divide by m and introduce $\omega_0^2 = \sqrt{k/m}$ and obtain

$$\frac{d^2x}{dt^2} + \frac{b}{m}\frac{dx}{dt} + \omega_0^2 x(t) = 0.$$

Thereafter we introduce a dimensionless time $\tau = t\omega_0$ (check that the dimensionality is correct) and rewrite our equation as

$$\frac{d^2x}{d\tau^2}+\frac{b}{m\omega_0}\frac{dx}{d\tau}+x(\tau)=0,$$

which gives us

$$\frac{d^2x}{d\tau^2}+\frac{b}{m\omega_0}\frac{dx}{d\tau}+x(\tau)=0.$$

We then define $\gamma = b/(2m\omega_0)$ and rewrite our equations as

$$\frac{d^2x}{d\tau^2}+2\gamma\frac{dx}{d\tau}+x(\tau)=0.$$

This is the equation we will code below. The first version employs the Euler-Cromer method.

```
# Common imports
import numpy as np
import pandas as pd
from math import *
import matplotlib.pyplot as plt
import os

# Where to save the figures and data files
PROJECT_ROOT_DIR = "Results"
FIGURE_ID = "Results/FigureFiles"
DATA_ID = "DataFiles/"

if not os.path.exists(PROJECT_ROOT_DIR):
    os.mkdir(PROJECT_ROOT_DIR)

if not os.path.exists(FIGURE_ID):
    os.makedirs(FIGURE_ID)

if not os.path.exists(DATA_ID):
    os.makedirs(DATA_ID)

def image_path(fig_id):
    return os.path.join(FIGURE_ID, fig_id)

def data_path(dat_id):
    return os.path.join(DATA_ID, dat_id)

def save_fig(fig_id):
    plt.savefig(image_path(fig_id) + ".png", format='png')


from pylab import plt, mpl
plt.style.use('seaborn')
mpl.rcParams['font.family'] = 'serif'

DeltaT = 0.001
#set up arrays
tfinal = 20 # in years
n = ceil(tfinal/DeltaT)
# set up arrays for t, v, and x
t = np.zeros(n)
v = np.zeros(n)
x = np.zeros(n)
```

```
# Initial conditions as simple one-dimensional arrays of time
x0 =  1.0
v0 = 0.0
x[0] = x0
v[0] = v0
gamma = 0.0
# Start integrating using Euler-Cromer's method
for i in range(n-1):
    # Set up the acceleration
    # Here you could have defined your own function for this
    a =  -2*gamma*v[i]-x[i]
    # update velocity, time and position
    v[i+1] = v[i] + DeltaT*a
    x[i+1] = x[i] + DeltaT*v[i+1]
    t[i+1] = t[i] + DeltaT
# Plot position as function of time
fig, ax = plt.subplots()
#ax.set_xlim(0, tfinal)
ax.set_ylabel('x[m]')
ax.set_xlabel('t[s]')
ax.plot(t, x)
fig.tight_layout()
save_fig("BlockEulerCromer")
plt.show()
```

When setting up the value of γ we see that for $\gamma = 0$ we get the simple oscillatory motion with no damping. Choosing $\gamma < 1$ leads to the classical underdamped case with oscillatory motion, but where the motion comes to an end.

Choosing $\gamma = 1$ leads to what normally is called critical damping and $\gamma > 1$ leads to critical overdamping. Try it out and try also to change the initial position and velocity. Setting $\gamma = 1$ yields a situation, as discussed above, where the solution approaches quickly zero and does not oscillate. With zero initial velocity it will never cross zero.

Sinusoidally Driven Oscillators

Here, we consider the force

$$F = -kx - b\dot{x} + F_0 \cos \omega t, \tag{92}$$

which leads to the differential equation

$$\ddot{x} + 2\beta\dot{x} + \omega_0^2 x = (F_0/m) \cos \omega t. \tag{93}$$

Consider a single solution with no arbitrary constants, which we will call a *particular solution*, $x_p(t)$. It should be emphasized that this is **A** particular solution, because there exists an infinite number of such solutions because the general solution should have two arbitrary constants. Now consider solutions to the same equation without the driving term, which include two arbitrary constants. These are called either *homogenous solutions* or *complementary solutions*, and were given in the previous section, e.g. Eq. (89) for the underdamped case. The homogenous solution already incorporates the two arbitrary constants, so any

sum of a homogenous solution and a particular solution will represent the *general solution* of the equation. The general solution incorporates the two arbitrary constants A and B to accommodate the two initial conditions. One could have picked a different particular solution, i.e. the original particular solution plus any homogenous solution with the arbitrary constants A_p and B_p chosen at will. When one adds in the homogenous solution, which has adjustable constants with arbitrary constants A' and B', to the new particular solution, one can get the same general solution by simply adjusting the new constants such that $A' + A_p = A$ and $B' + B_p = B$. Thus, the choice of A_p and B_p are irrelevant, and when choosing the particular solution it is best to make the simplest choice possible.

To find a particular solution, one first guesses at the form,

$$x_p(t) = D\cos(\omega t - \delta), \tag{94}$$

and rewrite the differential equation as

$$D\left\{-\omega^2\cos(\omega t-\delta) - 2\beta\omega\sin(\omega t-\delta) + \omega_0^2\cos(\omega t-\delta)\right\} = \frac{F_0}{m}\cos(\omega t). \tag{95}$$

One can now use angle addition formulas to get

$$\begin{aligned} D\left\{(-\omega^2\cos\delta + 2\beta\omega\sin\delta + \omega_0^2\cos\delta)\cos(\omega t)\right. & \\ \left.+(-\omega^2\sin\delta - 2\beta\omega\cos\delta + \omega_0^2\sin\delta)\sin(\omega t)\right\} &= \frac{F_0}{m}\cos(\omega t). \end{aligned} \tag{96}$$

Both the cos and sin terms need to equate if the expression is to hold at all times. Thus, this becomes two equations

$$\begin{aligned} D\left\{-\omega^2\cos\delta + 2\beta\omega\sin\delta + \omega_0^2\cos\delta\right\} &= \frac{F_0}{m} \\ -\omega^2\sin\delta - 2\beta\omega\cos\delta + \omega_0^2\sin\delta &= 0. \end{aligned} \tag{97}$$

After dividing by $\cos\delta$, the lower expression leads to

$$\tan\delta = \frac{2\beta\omega}{\omega_0^2 - \omega^2}. \tag{98}$$

Using the identities $\tan^2 + 1 = \csc^2$ and $\sin^2 + \cos^2 = 1$, one can also express $\sin\delta$ and $\cos\delta$,

$$\begin{aligned} \sin\delta &= \frac{2\beta\omega}{\sqrt{(\omega_0^2-\omega^2)^2 + 4\omega^2\beta^2}}, \\ \cos\delta &= \frac{(\omega_0^2-\omega^2)}{\sqrt{(\omega_0^2-\omega^2)^2 + 4\omega^2\beta^2}} \end{aligned} \tag{99}$$

Inserting the expressions for $\cos\delta$ and $\sin\delta$ into the expression for D,

$$D = \frac{F_0/m}{\sqrt{(\omega_0^2 - \omega^2)^2 + 4\omega^2\beta^2}}. \tag{100}$$

For a given initial condition, e.g. initial displacement and velocity, one must add the homogenous solution then solve for the two arbitrary constants. However, because the homogenous solutions decay with time as $e^{-\beta t}$, the particular solution is all that remains at large times, and is therefore the steady state solution. Because the arbitrary constants are all in the homogenous solution, all memory of the initial conditions are lost at large times, $t >> 1/\beta$.

The amplitude of the motion, D, is linearly proportional to the driving force (F_0/m), but also depends on the driving frequency ω. For small β the maximum will occur at $\omega = \omega_0$. This is referred to as a resonance. In the limit $\beta \to 0$ the amplitude at resonance approaches infinity.

Alternative Derivation for Driven Oscillators

Here, we derive the same expressions as in Equations (94) and (100) but express the driving forces as

$$F(t) \quad = \quad F_0 e^{i\omega t}, \tag{101}$$

rather than as $F_0\cos\omega t$. The real part of F is the same as before. For the differential equation,

$$\ddot{x} + 2\beta\dot{x} + \omega_0^2 x \quad = \quad \frac{F_0}{m} e^{i\omega t}, \tag{102}$$

one can treat $x(t)$ as an imaginary function. Because the operations d^2/dt^2 and d/dt are real and thus do not mix the real and imaginary parts of $x(t)$, Eq. (102) is effectively 2 equations. Because $e^{\omega t} = \cos\omega t + i\sin\omega t$, the real part of the solution for $x(t)$ gives the solution for a driving force $F_0\cos\omega t$, and the imaginary part of x corresponds to the case where the driving force is $F_0\sin\omega t$. It is rather easy to solve for the complex x in this case, and by taking the real part of the solution, one finds the answer for the $\cos\omega t$ driving force.

We assume a simple form for the particular solution

$$x_p = De^{i\omega t}, \tag{103}$$

where D is a complex constant.

From Eq. (102) one inserts the form for x_p above to get

$$D\left\{-\omega^2 + 2i\beta\omega + \omega_0^2\right\} e^{i\omega t} = (F_0/m)e^{i\omega t}, \tag{104}$$

$$D = \frac{F_0/m}{(\omega_0^2 - \omega^2) + 2i\beta\omega}.$$

The norm and phase for $D = |D|e^{-i\delta}$ can be read by inspection,

$$|D| = \frac{F_0/m}{\sqrt{(\omega_0^2 - \omega^2)^2 + 4\beta^2\omega^2}}, \qquad \tan\delta = \frac{2\beta\omega}{\omega_0^2 - \omega^2}. \tag{105}$$

This is the same expression for δ as before. One then finds $x_p(t)$,

$$\begin{aligned} x_p(t) &= \Re \frac{(F_0/m)e^{i\omega t - i\delta}}{\sqrt{(\omega_0^2 - \omega^2)^2 + 4\beta^2\omega^2}} \\ &= \frac{(F_0/m)\cos(\omega t - \delta)}{\sqrt{(\omega_0^2 - \omega^2)^2 + 4\beta^2\omega^2}}. \end{aligned} \tag{106}$$

This is the same answer as before. If one wished to solve for the case where $F(t) = F_0 \sin\omega t$, the imaginary part of the solution would work

$$\begin{aligned} x_p(t) &= \Im \frac{(F_0/m)e^{i\omega t - i\delta}}{\sqrt{(\omega_0^2 - \omega^2)^2 + 4\beta^2\omega^2}} \\ &= \frac{(F_0/m)\sin(\omega t - \delta)}{\sqrt{(\omega_0^2 - \omega^2)^2 + 4\beta^2\omega^2}}. \end{aligned} \tag{107}$$

Consider the damped and driven harmonic oscillator worked out above. Given F_0, m, β and ω_0, solve for the complete solution $x(t)$ for the case where $F = F_0 \sin\omega t$ with initial conditions $x(t=0) = 0$ and $v(t=0) = 0$. Assume the underdamped case.

The general solution including the arbitrary constants includes both the homogenous and particular solutions,

$$x(t) = \frac{F_0}{m}\frac{\sin(\omega t - \delta)}{\sqrt{(\omega_0^2 - \omega^2)^2 + 4\beta^2\omega^2}} + A\cos\omega' t e^{-\beta t} + B\sin\omega' t e^{-\beta t}.$$

The quantities δ and ω' are given earlier in the section, $\omega' = \sqrt{\omega_0^2 - \beta^2}, \delta = \tan^{-1}(2\beta\omega/(\omega_0^2 - \omega^2)$. Here, solving the problem means finding the arbitrary constants A and B. Satisfying the initial conditions for the initial position and velocity:

$$\begin{aligned} x(t=0) = 0 &= -\eta\sin\delta + A, \\ v(t=0) = 0 &= \omega\eta\cos\delta - \beta A + \omega' B, \\ \eta &\equiv \frac{F_0}{m}\frac{1}{\sqrt{(\omega_0^2 - \omega^2)^2 + 4\beta^2\omega^2}}. \end{aligned}$$

The problem is now reduced to 2 equations and 2 unknowns, A and B. The solution is

$$A = \eta\sin\delta, \quad B = \frac{-\omega\eta\cos\delta + \beta\eta\sin\delta}{\omega'}. \tag{108}$$

Resonance Widths; the Q factor

From the previous two sections, the particular solution for a driving force, $F = F_0 \cos \omega t$, is

$$\begin{aligned} x_p(t) &= \frac{F_0/m}{\sqrt{(\omega_0^2 - \omega^2)^2 + 4\omega^2\beta^2}} \cos(\omega_t - \delta), \\ \delta &= \tan^{-1}\left(\frac{2\beta\omega}{\omega_0^2 - \omega^2}\right). \end{aligned} \tag{109}$$

If one fixes the driving frequency ω and adjusts the fundamental frequency $\omega_0 = \sqrt{k/m}$, the maximum amplitude occurs when $\omega_0 = \omega$ because that is when the term from the denominator $(\omega_0^2 - \omega^2)^2 + 4\omega^2\beta^2$ is at a minimum. This is akin to dialing into a radio station. However, if one fixes ω_0 and adjusts the driving frequency one minimize with respect to ω, e.g. set

$$\frac{d}{d\omega}\left[(\omega_0^2 - \omega^2)^2 + 4\omega^2\beta^2\right] = 0, \tag{110}$$

and one finds that the maximum amplitude occurs when $\omega = \sqrt{\omega_0^2 - 2\beta^2}$. If β is small relative to ω_0, one can simply state that the maximum amplitude is

$$x_{\max} \approx \frac{F_0}{2m\beta\omega_0}. \tag{111}$$

$$\frac{4\omega^2\beta^2}{(\omega_0^2 - \omega^2)^2 + 4\omega^2\beta^2} = \frac{1}{2}. \tag{112}$$

For small damping this occurs when $\omega = \omega_0 \pm \beta$, so the $FWHM \approx 2\beta$. For the purposes of tuning to a specific frequency, one wants the width to be as small as possible. The ratio of ω_0 to $FWHM$ is known as the *quality* factor, or Q factor,

$$Q \equiv \frac{\omega_0}{2\beta}. \tag{113}$$

Numerical Studies of Driven Oscillations

Solving the problem of driven oscillations numerically gives us much more flexibility to study different types of driving forces. We can reuse our earlier code by simply adding a driving force. If we stay in the x-direction only this can be easily done by adding a term $F_{\text{ext}}(x,t)$. Note that we have kept it rather general here, allowing for both a spatial and a temporal dependence.

Before we dive into the code, we need to briefly remind ourselves about the equations we started with for the case with damping, namely

$$m\frac{d^2x}{dt^2} + b\frac{dx}{dt} + kx(t) = 0,$$

with no external force applied to the system.

Let us now for simplicty assume that our external force is given by

$$F_{\mathrm{ext}}(t) = F_0 \cos(\omega t),$$

where F_0 is a constant (what is its dimension?) and ω is the frequency of the applied external driving force. **Small question:** would you expect energy to be conserved now?

Introducing the external force into our lovely differential equation and dividing by m and introducing $\omega_0^2 = \sqrt{k/m}$ we have

$$\frac{d^2x}{dt^2} + \frac{b}{m}\frac{dx}{dt} + \omega_0^2 x(t) = \frac{F_0}{m}\cos(\omega t),$$

Thereafter we introduce a dimensionless time $\tau = t\omega_0$ and a dimensionless frequency $\tilde{\omega} = \omega/\omega_0$. We have then

$$\frac{d^2x}{d\tau^2} + \frac{b}{m\omega_0}\frac{dx}{d\tau} + x(\tau) = \frac{F_0}{m\omega_0^2}\cos(\tilde{\omega}\tau),$$

Introducing a new amplitude $\tilde{F} = F_0/(m\omega_0^2)$ (check dimensionality again) we have

$$\frac{d^2x}{d\tau^2} + \frac{b}{m\omega_0}\frac{dx}{d\tau} + x(\tau) = \tilde{F}\cos(\tilde{\omega}\tau).$$

Our final step, as we did in the case of various types of damping, is to define $\gamma = b/(2m\omega_0)$ and rewrite our equations as

$$\frac{d^2x}{d\tau^2} + 2\gamma\frac{dx}{d\tau} + x(\tau) = \tilde{F}\cos(\tilde{\omega}\tau).$$

This is the equation we will code below using the Euler-Cromer method.

```
DeltaT = 0.001
#set up arrays
tfinal = 20 # in years
n = ceil(tfinal/DeltaT)
# set up arrays for t, v, and x
t = np.zeros(n)
v = np.zeros(n)
x = np.zeros(n)
# Initial conditions as one-dimensional arrays of time
x0 =  1.0
v0 = 0.0
x[0] = x0
v[0] = v0
gamma = 0.2
Omegatilde = 0.5
Ftilde = 1.0
# Start integrating using Euler-Cromer's method
for i in range(n-1):
    # Set up the acceleration
    # Here you could have defined your own function for this
    a =  -2*gamma*v[i]-x[i]+Ftilde*cos(t[i]*Omegatilde)
```

```
    # update velocity, time and position
    v[i+1] = v[i] + DeltaT*a
    x[i+1] = x[i] + DeltaT*v[i+1]
    t[i+1] = t[i] + DeltaT
# Plot position as function of time
fig, ax = plt.subplots()
ax.set_ylabel('x[m]')
ax.set_xlabel('t[s]')
ax.plot(t, x)
fig.tight_layout()
save_fig("ForcedBlockEulerCromer")
plt.show()
```

In the above example we have focused on the Euler-Cromer method. This method has a local truncation error which is proportional to Δt^2 and thereby a global error which is proportional to Δt. We can improve this by using the Runge-Kutta family of methods. The widely popular Runge-Kutta to fourth order or just **RK4** has indeed a much better truncation error. The RK4 method has a global error which is proportional to Δt.

Let us revisit this method and see how we can implement it for the above example.

Differential Equations, Runge-Kutta methods

Runge-Kutta (RK) methods are based on Taylor expansion formulae, but yield in general better algorithms for solutions of an ordinary differential equation. The basic philosophy is that it provides an intermediate step in the computation of y_{i+1}.

To see this, consider first the following definitions

$$\frac{dy}{dt} = f(t, y), \tag{114}$$

and

$$y(t) = \int f(t, y)dt, \tag{115}$$

and

$$y_{i+1} = y_i + \int_{t_i}^{t_{i+1}} f(t, y)dt. \tag{116}$$

To demonstrate the philosophy behind RK methods, let us consider the second-order RK method, RK2. The first approximation consists in Taylor expanding $f(t, y)$ around the center of the integration interval t_i to t_{i+1}, that is, at $t_i + h/2$, h being the step. Using the midpoint formula for an integral, defining $y(t_i + h/2) = y_{i+1/2}$ and $t_i + h/2 = t_{i+1/2}$, we obtain

$$\int_{t_i}^{t_{i+1}} f(t, y)dt \approx hf(t_{i+1/2}, y_{i+1/2}) + O(h^3). \tag{117}$$

This means in turn that we have

$$y_{i+1} = y_i + hf(t_{i+1/2}, y_{i+1/2}) + O(h^3). \tag{118}$$

However, we do not know the value of $y_{i+1/2}$. Here comes thus the next approximation, namely, we use Euler's method to approximate $y_{i+1/2}$. We have then

$$y_{(i+1/2)} = y_i + \frac{h}{2}\frac{dy}{dt} = y(t_i) + \frac{h}{2}f(t_i, y_i). \tag{119}$$

This means that we can define the following algorithm for the second-order Runge-Kutta method, RK2.

$$k_1 = hf(t_i, y_i), \tag{120}$$

$$k_2 = hf(t_{i+1/2}, y_i + k_1/2), \tag{121}$$

with the final value

$$y_{i+i} \approx y_i + k_2 + O(h^3). \tag{122}$$

The difference between the previous one-step methods is that we now need an intermediate step in our evaluation, namely $t_i + h/2 = t_{(i+1/2)}$ where we evaluate the derivative f. This involves more operations, but the gain is a better stability in the solution.

The fourth-order Runge-Kutta, RK4, has the following algorithm

$$k_1 = hf(t_i, y_i) \quad k_2 = hf(t_i + h/2, y_i + k_1/2)$$

$$k_3 = hf(t_i + h/2, y_i + k_2/2) \quad k_4 = hf(t_i + h, y_i + k_3)$$

with the final result

$$y_{i+1} = y_i + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right).$$

Thus, the algorithm consists in first calculating k_1 with t_i, y_1 and f as inputs. Thereafter, we increase the step size by $h/2$ and calculate k_2, then k_3 and finally k_4. The global error goes as $O(h^4)$.

However, at this stage, if we keep adding different methods in our main program, the code will quickly become messy and ugly. Before we proceed thus, we will now introduce functions that enbody the various methods for solving differential equations. This means that we can separate out these methods in own functions and files (and later as classes and more generic functions) and simply call them when needed. Similarly, we could easily encapsulate various forces or other quantities of interest in terms of functions. To see this, let us bring up the code we developed above for the simple sliding block, but now only with the simple forward Euler method. We introduce two functions, one for the simple Euler method and one for the force.

Note that here the forward Euler method does not know the specific force function to be called. It receives just an input the name. We can easily change the force by adding another function.

```
def ForwardEuler(v,x,t,n,Force):
    for i in range(n-1):
        v[i+1] = v[i] + DeltaT*Force(v[i],x[i],t[i])
        x[i+1] = x[i] + DeltaT*v[i]
        t[i+1] = t[i] + DeltaT
```

```
def SpringForce(v,x,t):
#   note here that we have divided by mass and we return the acceleration
    return  -2*gamma*v-x+Ftilde*cos(t*Omegatilde)
```

It is easy to add a new method like the Euler-Cromer

```
def ForwardEulerCromer(v,x,t,n,Force):
    for i in range(n-1):
        a = Force(v[i],x[i],t[i])
        v[i+1] = v[i] + DeltaT*a
        x[i+1] = x[i] + DeltaT*v[i+1]
        t[i+1] = t[i] + DeltaT
```

and the Velocity Verlet method (be careful with time-dependence here, it is not an ideal method for non-conservative forces))

```
def VelocityVerlet(v,x,t,n,Force):
    for i in range(n-1):
        a = Force(v[i],x[i],t[i])
        x[i+1] = x[i] + DeltaT*v[i]+0.5*a
        anew = Force(v[i],x[i+1],t[i+1])
        v[i+1] = v[i] + 0.5*DeltaT*(a+anew)
        t[i+1] = t[i] + DeltaT
```

Finally, we can now add the Runge-Kutta2 method via a new function

```
def RK2(v,x,t,n,Force):
    for i in range(n-1):
# Setting up k1
        k1x = DeltaT*v[i]
        k1v = DeltaT*Force(v[i],x[i],t[i])
# Setting up k2
        vv = v[i]+k1v*0.5
        xx = x[i]+k1x*0.5
        k2x = DeltaT*vv
        k2v = DeltaT*Force(vv,xx,t[i]+DeltaT*0.5)
# Final result
        x[i+1] = x[i]+k2x
        v[i+1] = v[i]+k2v
        t[i+1] = t[i]+DeltaT
```

Finally, we can now add the Runge-Kutta2 method via a new function

```
def RK4(v,x,t,n,Force):
    for i in range(n-1):
# Setting up k1
        k1x = DeltaT*v[i]
        k1v = DeltaT*Force(v[i],x[i],t[i])
# Setting up k2
        vv = v[i]+k1v*0.5
        xx = x[i]+k1x*0.5
        k2x = DeltaT*vv
        k2v = DeltaT*Force(vv,xx,t[i]+DeltaT*0.5)
# Setting up k3
        vv = v[i]+k2v*0.5
        xx = x[i]+k2x*0.5
        k3x = DeltaT*vv
        k3v = DeltaT*Force(vv,xx,t[i]+DeltaT*0.5)
# Setting up k4
        vv = v[i]+k3v
```

```
        xx = x[i]+k3x
        k4x = DeltaT*vv
        k4v = DeltaT*Force(vv,xx,t[i]+DeltaT)
# Final result
        x[i+1] = x[i]+(k1x+2*k2x+2*k3x+k4x)/6.
        v[i+1] = v[i]+(k1v+2*k2v+2*k3v+k4v)/6.
        t[i+1] = t[i] + DeltaT
```

The Runge-Kutta family of methods are particularly useful when we have a time-dependent acceleration. If we have forces which depend only the spatial degrees of freedom (no velocity and/or time-dependence), then energy conserving methods like the Velocity Verlet or the Euler-Cromer method are preferred. As soon as we introduce an explicit time-dependence and/or add dissipitave forces like friction or air resistance, then methods like the family of Runge-Kutta methods are well suited for this. The code below uses the Runge-Kutta4 methods.

```
DeltaT = 0.001
#set up arrays
tfinal = 20 # in years
n = ceil(tfinal/DeltaT)
# set up arrays for t, v, and x
t = np.zeros(n)
v = np.zeros(n)
x = np.zeros(n)
# Initial conditions (can change to more than one dim)
x0 =  1.0
v0 = 0.0
x[0] = x0
v[0] = v0
gamma = 0.2
Omegatilde = 0.5
Ftilde = 1.0
# Start integrating using Euler's method
# Note that we define the force function as a SpringForce
RK4(v,x,t,n,SpringForce)

# Plot position as function of time
fig, ax = plt.subplots()
ax.set_ylabel('x[m]')
ax.set_xlabel('t[s]')
ax.plot(t, x)
fig.tight_layout()
save_fig("ForcedBlockRK4")
plt.show()
```

Principle of Superposition and Periodic Forces (Fourier Transforms)

If one has several driving forces, $F(t) = \sum_n F_n(t)$, one can find the particular solution to each F_n, $x_{pn}(t)$, and the particular solution for the entire driving force is

$$x_p(t) = \sum_n x_{pn}(t). \tag{123}$$

This is known as the principal of superposition. It only applies when the homogenous equation is linear. If there were an anharmonic term such as x^3 in the homogenous equation, then when one summed various solutions, $x = (\sum_n x_n)^2$, one would get cross terms. Superposition is especially useful when $F(t)$ can be written as a sum of sinusoidal terms, because the solutions for each sinusoidal (sine or cosine) term is analytic, as we saw above.

Driving forces are often periodic, even when they are not sinusoidal. Periodicity implies that for some time τ

$$F(t+\tau) = F(t). \tag{124}$$

One example of a non-sinusoidal periodic force is a square wave. Many components in electric circuits are non-linear, e.g. diodes, which makes many wave forms non-sinusoidal even when the circuits are being driven by purely sinusoidal sources.

The code here shows a typical example of such a square wave generated using the functionality included in the **scipy** Python package. We have used a period of $\tau = 0.2$.

```
import numpy as np
import math
from scipy import signal
import matplotlib.pyplot as plt

# number of points
n = 500
# start and final times
t0 = 0.0
tn = 1.0
# Period
t = np.linspace(t0, tn, n, endpoint=False)
SqrSignal = np.zeros(n)
SqrSignal = 1.0+signal.square(2*np.pi*5*t)
plt.plot(t, SqrSignal)
plt.ylim(-0.5, 2.5)
plt.show()
```

For the sinusoidal example studied in the previous subsections the period is $\tau = 2\pi/\omega$. However, higher harmonics can also satisfy the periodicity requirement. In general, any force that satisfies the periodicity requirement can be expressed as a sum over harmonics,

$$F(t) = \frac{f_0}{2} + \sum_{n>0} f_n \cos(2n\pi t/\tau) + g_n \sin(2n\pi t/\tau). \tag{125}$$

From the previous subsection, one can write down the answer for $x_{pn}(t)$, by substituting f_n/m or g_n/m for F_0/m into Eq.s (106) or (107) respectively. By writing each factor $2n\pi t/\tau$ as $n\omega t$, with $\omega \equiv 2\pi/\tau$,

$$F(t) = \frac{f_0}{2} + \sum_{n>0} f_n \cos(n\omega t) + g_n \sin(n\omega t). \tag{126}$$

The solutions for $x(t)$ then come from replacing ω with $n\omega$ for each term in the particular solution in Equations (94) and (100),

$$\begin{aligned} x_p(t) &= \frac{f_0}{2k} + \sum_{n>0} \alpha_n \cos(n\omega t - \delta_n) + \beta_n \sin(n\omega t - \delta_n), \qquad (127) \\ \alpha_n &= \frac{f_n/m}{\sqrt{((n\omega)^2 - \omega_0^2) + 4\beta^2 n^2 \omega^2}}, \\ \beta_n &= \frac{g_n/m}{\sqrt{((n\omega)^2 - \omega_0^2) + 4\beta^2 n^2 \omega^2}}, \\ \delta_n &= \tan^{-1}\left(\frac{2\beta n\omega}{\omega_0^2 - n^2\omega^2}\right). \end{aligned}$$

Because the forces have been applied for a long time, any non-zero damping eliminates the homogenous parts of the solution, so one need only consider the particular solution for each n.

The problem will considered solved if one can find expressions for the coefficients f_n and g_n, even though the solutions are expressed as an infinite sum. The coefficients can be extracted from the function $F(t)$ by

$$\begin{aligned} f_n &= \frac{2}{\tau}\int_{-\tau/2}^{\tau/2} dt\ F(t)\cos(2n\pi t/\tau), \qquad (128) \\ g_n &= \frac{2}{\tau}\int_{-\tau/2}^{\tau/2} dt\ F(t)\sin(2n\pi t/\tau). \end{aligned}$$

To check the consistency of these expressions and to verify Eq. (128), one can insert the expansion of $F(t)$ in Eq. (126) into the expression for the coefficients in Eq. (128) and see whether

$$f_n \quad =? \quad \frac{2}{\tau}\int_{-\tau/2}^{\tau/2} dt\ \left\{\frac{f_0}{2} + \sum_{m>0} f_m \cos(m\omega t) + g_m \sin(m\omega t)\right\} \cos(n\omega t) \qquad (129)$$

Immediately, one can throw away all the terms with g_m because they convolute an even and an odd function. The term with $f_0/2$ disappears because $\cos(n\omega t)$ is equally positive and negative over the interval and will integrate to zero. For all the terms $f_m \cos(m\omega t)$ appearing in the sum, one can use angle addition formulas to see that $\cos(m\omega t)\cos(n\omega t) = (1/2)(\cos[(m+n)\omega t] + \cos[(m-n)\omega t]$. This will integrate to zero unless $m = n$. In that case the $m = n$ term gives

$$\int_{-\tau/2}^{\tau/2} dt\ \cos^2(m\omega t) = \frac{\tau}{2}, \qquad (130)$$

and

$$\begin{aligned} f_n &=? \quad \frac{2}{\tau}\int_{-\tau/2}^{\tau/2} dt\ f_n/2 \\ &= \quad f_n\ \checkmark. \end{aligned} \tag{131}$$

The same method can be used to check for the consistency of g_n.
Consider the driving force:

$$F(t) = At/\tau, \quad -\tau/2 < t < \tau/2, \quad F(t+\tau) = F(t). \tag{132}$$

Find the Fourier coefficients f_n and g_n for all n using Eq. (128).
Only the odd coefficients enter by symmetry, i.e. $f_n = 0$. One can find g_n integrating by parts,

$$\begin{aligned} g_n &= \frac{2}{\tau}\int_{-\tau/2}^{\tau/2} dt\ \sin(n\omega t)\frac{At}{\tau} \\ u &= t,\ dv = \sin(n\omega t)dt,\ v = -\cos(n\omega t)/(n\omega), \\ g_n &= \frac{-2A}{n\omega\tau^2}\int_{-\tau/2}^{\tau/2} dt\ \cos(n\omega t) + 2A\frac{-t\cos(n\omega t)}{n\omega\tau^2}\Big|_{-\tau/2}^{\tau/2}. \end{aligned} \tag{133}$$

The first term is zero because $\cos(n\omega t)$ will be equally positive and negative over the interval. Using the fact that $\omega\tau = 2\pi$,

$$\begin{aligned} g_n &= -\frac{2A}{2n\pi}\cos(n\omega\tau/2) \\ &= -\frac{A}{n\pi}\cos(n\pi) \\ &= \frac{A}{n\pi}(-1)^{n+1}. \end{aligned} \tag{134}$$

Fourier Series

More text will come here, chpater 5.7-5.8 of Taylor are discussed during the lectures. The code here uses the Fourier series discussed in chapter 5.7 for a square wave signal. The equations for the coefficients are are discussed in Taylor section 5.7, see Example 5.4. The code here visualizes the various approximations given by Fourier series compared with a square wave with period $T = 0.2$, witth 0.1 and max value $F = 2$. We see that when we increase the number of components in the Fourier series, the Fourier series approximation gets closes and closes to the square wave signal.

```
import numpy as np
import math
from scipy import signal
import matplotlib.pyplot as plt
```

```
# number of points
n = 500
# start and final times
t0 = 0.0
tn = 1.0
# Period
T =0.2
# Max value of square signal
Fmax= 2.0
# Width of signal
Width = 0.1
t = np.linspace(t0, tn, n, endpoint=False)
SqrSignal = np.zeros(n)
FourierSeriesSignal = np.zeros(n)
SqrSignal = 1.0+signal.square(2*np.pi*5*t+np.pi*Width/T)
a0 = Fmax*Width/T
FourierSeriesSignal = a0
Factor = 2.0*Fmax/np.pi
for i in range(1,500):
    FourierSeriesSignal += Factor/(i)*np.sin(np.pi*i*Width/T)*np.cos(i*t*2*np.pi/T)
plt.plot(t, SqrSignal)
plt.plot(t, FourierSeriesSignal)
plt.ylim(-0.5, 2.5)
plt.show()
```

Solving differential equations with Fouries series

The material here was discussed during the lecture of February 19 and 21. It is also covered by Taylor in section 5.8.

Consider a particle at rest in the bottom of an underdamped harmonic oscillator, that then feels a sudden impulse, or change in momentum, $I = F\Delta t$ at $t = 0$. This increases the velocity immediately by an amount $v_0 = I/m$ while not changing the position. One can then solve the trajectory by solving Eq. (89) with initial conditions $v_0 = I/m$ and $x_0 = 0$. This gives

$$x(t) = \frac{I}{m\omega'} e^{-\beta t} \sin \omega' t, \quad t > 0. \tag{135}$$

Here, $\omega' = \sqrt{\omega_0^2 - \beta^2}$. For an impulse I_i that occurs at time t_i the trajectory would be

$$x(t) = \frac{I_i}{m\omega'} e^{-\beta(t-t_i)} \sin[\omega'(t-t_i)]\Theta(t-t_i), \tag{136}$$

where $\Theta(t - t_i)$ is a step function, i.e. $\Theta(x)$ is zero for $x < 0$ and unity for $x > 0$. If there were several impulses linear superposition tells us that we can sum over each contribution,

$$x(t) = \sum_i \frac{I_i}{m\omega'} e^{-\beta(t-t_i)} \sin[\omega'(t-t_i)]\Theta(t-t_i) \tag{137}$$

Now one can consider a series of impulses at times separated by Δt, where each impulse is given by $F_i \Delta t$. The sum above now becomes an integral,

$$
\begin{aligned}
x(t) &= \int_{-\infty}^{\infty} dt'\ F(t') \frac{e^{-\beta(t-t')} \sin[\omega'(t-t')]}{m\omega'} \Theta(t-t') \qquad (138)\\
&= \int_{-\infty}^{\infty} dt'\ F(t') G(t-t'),\\
G(\Delta t) &= \frac{e^{-\beta \Delta t} \sin[\omega' \Delta t]}{m\omega'} \Theta(\Delta t)
\end{aligned}
$$

The quantity $e^{-\beta(t-t')} \sin[\omega'(t-t')]/m\omega'\Theta(t-t')$ is called a Green's function, $G(t-t')$. It describes the response at t due to a force applied at a time t', and is a function of $t-t'$. The step function ensures that the response does not occur before the force is applied. One should remember that the form for G would change if the oscillator were either critically- or over-damped.

When performing the integral in Eq. (138) one can use angle addition formulas to factor out the part with the t' dependence in the integrand,

$$
\begin{aligned}
x(t) &= \frac{1}{m\omega'} e^{-\beta t} \left[I_c(t) \sin(\omega' t) - I_s(t) \cos(\omega' t) \right], \qquad (139)\\
I_c(t) &\equiv \int_{-\infty}^{t} dt'\ F(t') e^{\beta t'} \cos(\omega' t'),\\
I_s(t) &\equiv \int_{-\infty}^{t} dt'\ F(t') e^{\beta t'} \sin(\omega' t').
\end{aligned}
$$

If the time t is beyond any time at which the force acts, $F(t' > t) = 0$, the coefficients I_c and I_s become independent of t.

Consider an undamped oscillator ($\beta \to 0$), with characteristic frequency ω_0 and mass m, that is at rest until it feels a force described by a Gaussian form,

$$
F(t) = F_0 \exp\left\{ \frac{-t^2}{2\tau^2} \right\}.
$$

For large times ($t >> \tau$), where the force has died off, find $x(t)$.
Solve for the coefficients I_c and I_s in Eq. (139). Because the Gaussian is an even function, $I_s = 0$, and one need only solve for I_c,

$$
\begin{aligned}
I_c &= F_0 \int_{-\infty}^{\infty} dt'\ e^{-t'^2/(2\tau^2)} \cos(\omega_0 t')\\
&= \Re F_0 \int_{-\infty}^{\infty} dt'\ e^{-t'^2/(2\tau^2)} e^{i\omega_0 t'}\\
&= \Re F_0 \int_{-\infty}^{\infty} dt'\ e^{-(t'-i\omega_0\tau^2)^2/(2\tau^2)} e^{-\omega_0^2\tau^2/2}\\
&= F_0 \tau \sqrt{2\pi} e^{-\omega_0^2\tau^2/2}.
\end{aligned}
$$

The third step involved completing the square, and the final step used the fact that the integral

$$\int_{-\infty}^{\infty} dx\ e^{-x^2/2} \quad = \quad \sqrt{2\pi}.$$

To see that this integral is true, consider the square of the integral, which you can change to polar coordinates,

$$\begin{aligned} I &= \int_{-\infty}^{\infty} dx\ e^{-x^2/2} \\ I^2 &= \int_{-\infty}^{\infty} dxdy\ e^{-(x^2+y^2)/2} \\ &= 2\pi \int_0^{\infty} rdr\ e^{-r^2/2} \\ &= 2\pi. \end{aligned}$$

Finally, the expression for x from Eq. (139) is

$$x(t >> \tau) \quad = \quad \frac{F_0\tau}{m\omega_0}\sqrt{2\pi}e^{-\omega_0^2\tau^2/2}\sin(\omega_0 t).$$

The classical pendulum and scaling the equations

Let us end our discussion of oscillations with another classical case, the pendulum.

The angular equation of motion of the pendulum is given by Newton's equation and with no external force it reads

$$ml\frac{d^2\theta}{dt^2} + mgsin(\theta) = 0, \tag{140}$$

with an angular velocity and acceleration given by

$$v = l\frac{d\theta}{dt}, \tag{141}$$

and

$$a = l\frac{d^2\theta}{dt^2}. \tag{142}$$

We do however expect that the motion will gradually come to an end due a viscous drag torque acting on the pendulum. In the presence of the drag, the above equation becomes

$$ml\frac{d^2\theta}{dt^2} + \nu\frac{d\theta}{dt} + mgsin(\theta) = 0, \tag{143}$$

where ν is now a positive constant parameterizing the viscosity of the medium in question. In order to maintain the motion against viscosity, it is necessary to add some external driving force. We choose here a periodic driving force. The last equation becomes then

$$ml\frac{d^2\theta}{dt^2} + \nu\frac{d\theta}{dt} + mgsin(\theta) = Asin(\omega t), \tag{144}$$

with A and ω two constants representing the amplitude and the angular frequency respectively. The latter is called the driving frequency.

We define

$$\omega_0 = \sqrt{g/l},$$

the so-called natural frequency and the new dimensionless quantities

$$\hat{t} = \omega_0 t,$$

with the dimensionless driving frequency

$$\hat{\omega} = \frac{\omega}{\omega_0},$$

and introducing the quantity Q, called the *quality factor*,

$$Q = \frac{mg}{\omega_0 \nu},$$

and the dimensionless amplitude

$$\hat{A} = \frac{A}{mg}$$

We have

$$\frac{d^2\theta}{d\hat{t}^2} + \frac{1}{Q}\frac{d\theta}{d\hat{t}} + sin(\theta) = \hat{A}cos(\hat{\omega}\hat{t}).$$

This equation can in turn be recast in terms of two coupled first-order differential equations as follows

$$\frac{d\theta}{d\hat{t}} = \hat{v},$$

and

$$\frac{d\hat{v}}{d\hat{t}} = -\frac{\hat{v}}{Q} - sin(\theta) + \hat{A}cos(\hat{\omega}\hat{t}).$$

These are the equations to be solved. The factor Q represents the number of oscillations of the undriven system that must occur before its energy is significantly reduced due to the viscous drag. The amplitude $\hat{A}$ is measured in units of the maximum possible gravitational torque while $\hat{\omega}$ is the angular frequency of the external torque measured in units of the pendulum's natural frequency.

Gravity and Central Forces

The gravitational potential energy and forces involving two masses a and b are

$$\begin{aligned} U_{ab} &= -\frac{Gm_a m_b}{|\boldsymbol{r}_a - \boldsymbol{r}_b|}, \\ F_{ba} &= -\frac{Gm_a m_b}{|\boldsymbol{r}_a - \boldsymbol{r}_b|^2}\hat{r}_{ab}, \\ \hat{r}_{ab} &= \frac{\boldsymbol{r}_b - \boldsymbol{r}_a}{|\boldsymbol{r}_a - \boldsymbol{r}_b|}. \end{aligned} \tag{145}$$

Here $G = 6.67 \times 10^{-11}$ Nm^2/kg^2, and F_{ba} is the force on b due to a. By inspection, one can see that the force on b due to a and the force on a due to b are equal and opposite. The net potential energy for a large number of masses would be

$$U = \sum_{a<b} U_{ab} = \frac{1}{2}\sum_{a\neq b} U_{ab}. \tag{146}$$

Relative and Center of Mass Motion

Thus far, we have considered the trajectory as if the force is centered around a fixed point. For two bodies interacting only with one another, both masses circulate around the center of mass. One might think that solutions would become more complex when both particles move, but we will see here that the problem can be reduced to one with a single body moving according to a fixed force by expressing the trajectories for $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ into the center-of-mass coordinate $\boldsymbol{R}_{\rm cm}$ and the relative coordinate $\boldsymbol{r}$,

$$\begin{aligned} \boldsymbol{R}_{\rm cm} &\equiv \frac{m_1\boldsymbol{r}_1 + m_2\boldsymbol{r}_2}{m_1 + m_2}, \\ \boldsymbol{r} &\equiv \boldsymbol{r}_1 - \boldsymbol{r_2}. \end{aligned} \tag{147}$$

Here, we assume the two particles interact only with one another, so $\boldsymbol{F}_{12} = -\boldsymbol{F}_{21}$ (where $\boldsymbol{F}_{ij}$ is the force on i due to j. The equations of motion then become

$$\begin{aligned} \ddot{\boldsymbol{R}}_{\rm cm} &= \frac{1}{m_1 + m_2}\left\{m_1\ddot{\boldsymbol{r}}_1 + m_2\ddot{\boldsymbol{r}}_2\right\} \qquad (148) \\ &= \frac{1}{m_1 + m_2}\left\{\boldsymbol{F}_{12} + \boldsymbol{F}_{21}\right\} = 0. \\ \ddot{\boldsymbol{r}} &= \ddot{\boldsymbol{r}}_1 - \ddot{\boldsymbol{r}}_2 = \left(\frac{\boldsymbol{F}_{12}}{m_1} - \frac{\boldsymbol{F}_{21}}{m_2}\right) \qquad (149) \\ &= \left(\frac{1}{m_1} + \frac{1}{m_2}\right)\boldsymbol{F}_{12}. \end{aligned}$$

The first expression simply states that the center of mass coordinate $\boldsymbol{R}_{\rm cm}$ moves at a fixed velocity. The second expression can be rewritten in terms of the reduced mass μ.

$$\begin{aligned} \mu\ddot{\boldsymbol{r}} &= \boldsymbol{F}_{12}, & (150)\\ \frac{1}{\mu} &= \frac{1}{m_1}+\frac{1}{m_2}, \quad \mu=\frac{m_1 m_2}{m_1+m_2}. & (151) \end{aligned}$$

Thus, one can treat the trajectory as a one-body problem where the reduced mass is μ, and a second trivial problem for the center of mass. The reduced mass is especially convenient when one is considering gravitational problems because then

$$\begin{aligned} \mu\ddot{r} &= -\frac{Gm_1m_2}{r^2}\hat{r} & (152)\\ &= -\frac{GM\mu}{r^2}\hat{r}, \quad M\equiv m_1+m_2. \end{aligned}$$

For the gravitational problem, the reduced mass then falls out and the trajectory depends only on the total mass M.

The kinetic energy and momenta also have analogues in center-of-mass coordinates. The total and relative momenta are

$$\begin{aligned} \boldsymbol{P} &\equiv \boldsymbol{p}_1+\boldsymbol{p}_2 = M\dot{\boldsymbol{R}}_{\rm cm}, & (153)\\ \boldsymbol{q} &\equiv \mu\dot{\boldsymbol{r}}. \end{aligned}$$

With these definitions, a little algebra shows that the kinetic energy becomes

$$\begin{aligned} T &= \frac{1}{2}m_1|\boldsymbol{v}_1|^2+\frac{1}{2}m_2|\boldsymbol{v}_2|^2 & (154)\\ &= \frac{1}{2}M|\dot{\boldsymbol{R}}_{\rm cm}|^2+\frac{1}{2}\mu|\dot{\boldsymbol{r}}|^2\\ &= \frac{P^2}{2M}+\frac{q^2}{2\mu}. \end{aligned}$$

The standard strategy is to transform into the center of mass frame, then treat the problem as one of a single particle of mass μ undergoing a force $\boldsymbol{F}_{12}$. Scattering angles can also be expressed in this frame, then transformed into the lab frame. In practice, one sees examples in the literature where $d\sigma/d\Omega$ expressed in both the "center-of-mass" and in the "laboratory" frame.

Deriving Elliptical Orbits

Kepler's laws state that a gravitational orbit should be an ellipse with the source of the gravitational field at one focus. Deriving this is surprisingly messy. To do

this, we first use angular momentum conservation to transform the equations of motion so that it is in terms of r and θ instead of r and t. The overall strategy is to

1. Find equations of motion for r and t with no angle (θ) mentioned, i.e. $d^2r/dt^2 = \cdots$. Angular momentum conservation will be used, and the equation will involve the angular momentum L.

2. Use angular momentum conservation to find an expression for $\dot{\theta}$ in terms of r.

3. Use the chain rule to convert the equations of motions for r, an expression involving $r, \dot{r}$ and $\ddot{r}$, to one involving $r, dr/d\theta$ and $d^2r/d\theta^2$. This is quitecomplicated because the expressions will also involve a substitution $u = 1/r$ so that one finds an expression in terms of u and θ.

4. Once $u(\theta)$ is found, you need to show that this can be converted to the familiar form for an ellipse.

The equations of motion give

$$\begin{aligned} \frac{d}{dt}r^2 &= \frac{d}{dt}(x^2+y^2) = 2x\dot{x} + 2y\dot{y} = 2r\dot{r}, \\ \dot{r} &= \frac{x}{r}\dot{x} + \frac{y}{r}\dot{y}, \\ \ddot{r} &= \frac{x}{r}\ddot{x} + \frac{y}{r}\ddot{y} + \frac{\dot{x}^2+\dot{y}^2}{r} - \frac{\dot{r}^2}{r}. \end{aligned} \tag{155}$$

Recognizing that the numerator of the third term is the velocity squared, and that it can be written in polar coordinates,

$$v^2 = \dot{x}^2 + \dot{y}^2 = \dot{r}^2 + r^2\dot{\theta}^2, \tag{156}$$

one can write $\ddot{r}$ as

$$\begin{aligned} \ddot{r} &= \frac{F_x\cos\theta + F_y\sin\theta}{m} + \frac{\dot{r}^2 + r^2\dot{\theta}^2}{r} - \frac{\dot{r}^2}{r} \\ &= \frac{F}{m} + \frac{r^2\dot{\theta}^2}{r} \\ m\ddot{r} &= F + \frac{L^2}{mr^3}. \end{aligned} \tag{157}$$

This derivation used the fact that the force was radial, $F = F_r = F_x\cos\theta + F_y\sin\theta$, and that angular momentum is $L = mrv_\theta = mr^2\dot{\theta}$. The term $L^2/mr^3 = mv^2/r$ behaves like an additional force. Sometimes this is referred to as a centrifugal force, but it is not a force. Instead, it is the consequence of considering the motion in a rotating (and therefore accelerating) frame.

Now, we switch to the particular case of an attractive inverse square force, $F = -\alpha/r^2$, and show that the trajectory, $r(\theta)$, is an ellipse. To do this we transform derivatives w.r.t. time to derivatives w.r.t. θ using the chain rule combined with angular momentum conservation, $\dot{\theta} = L/mr^2$.

$$\begin{aligned}
\dot{r} &= \frac{dr}{d\theta}\dot{\theta} = \frac{dr}{d\theta}\frac{L}{mr^2}, \\
\ddot{r} &= \frac{d^2r}{d\theta^2}\dot{\theta}^2 + \frac{dr}{d\theta}\left(\frac{d}{dr}\frac{L}{mr^2}\right)\dot{r} \\
&= \frac{d^2r}{d\theta^2}\left(\frac{L}{mr^2}\right)^2 - 2\frac{dr}{d\theta}\frac{L}{mr^3}\dot{r} \\
&= \frac{d^2r}{d\theta^2}\left(\frac{L}{mr^2}\right)^2 - \frac{2}{r}\left(\frac{dr}{d\theta}\right)^2\left(\frac{L}{mr^2}\right)^2
\end{aligned} \tag{158}$$

Equating the two expressions for $\ddot{r}$ in Eq.s (157) and (158) eliminates all the derivatives w.r.t. time, and provides a differential equation with only derivatives w.r.t. θ,

$$\frac{d^2r}{d\theta^2}\left(\frac{L}{mr^2}\right)^2 - \frac{2}{r}\left(\frac{dr}{d\theta}\right)^2\left(\frac{L}{mr^2}\right)^2 = \frac{F}{m} + \frac{L^2}{m^2r^3}, \tag{159}$$

that when solved yields the trajectory, i.e. $r(\theta)$. Up to this point the expressions work for any radial force, not just forces that fall as $1/r^2$.

The trick to simplifying this differential equation for the inverse square problems is to make a substitution, $u \equiv 1/r$, and rewrite the differential equation for $u(\theta)$.

$$\begin{aligned}
r &= 1/u, \\
\frac{dr}{d\theta} &= -\frac{1}{u^2}\frac{du}{d\theta}, \\
\frac{d^2r}{d\theta^2} &= \frac{2}{u^3}\left(\frac{du}{d\theta}\right)^2 - \frac{1}{u^2}\frac{d^2u}{d\theta^2}.
\end{aligned} \tag{160}$$

Plugging these expressions into Eq. (159) gives an expression in terms of u, $du/d\theta$, and $d^2u/d\theta^2$. After some tedious algebra,

$$\frac{d^2u}{d\theta^2} = -u - \frac{Fm}{L^2u^2}. \tag{161}$$

For the attractive inverse square law force, $F = -\alpha u^2$,

$$\frac{d^2u}{d\theta^2} = -u + \frac{m\alpha}{L^2}. \tag{162}$$

The solution has two arbitrary constants, A and θ_0,

$$
\begin{aligned}
u &= \frac{m\alpha}{L^2} + A\cos(\theta - \theta_0), \qquad (163)\\
r &= \frac{1}{(m\alpha/L^2) + A\cos(\theta - \theta_0)}.
\end{aligned}
$$

The radius will be at a minimum when $\theta = \theta_0$ and at a maximum when $\theta = \theta_0 + \pi$. The constant A is related to the eccentricity of the orbit. When $A = 0$ the radius is a constant $r = L^2/(m\alpha)$, and the motion is circular. If one solved the expression $mv^2/r = -\alpha/r^2$ for a circular orbit, using the substitution $v = L/(mr)$, one would reproduce the expression $r = L^2/(m\alpha)$.

The form describing the elliptical trajectory in Eq. (163) can be identified as an ellipse with one focus being the center of the ellipse by considering the definition of an ellipse as being the points such that the sum of the two distances between the two foci are a constant. Making that distance $2D$, the distance between the two foci as $2a$, and putting one focus at the origin,

$$
\begin{aligned}
2D &= r + \sqrt{(r\cos\theta - 2a)^2 + r^2\sin^2\theta}, \qquad (164)\\
4D^2 + r^2 - 4Dr &= r^2 + 4a^2 - 4ar\cos\theta,\\
r &= \frac{D^2 - a^2}{D + a\cos\theta} = \frac{1}{D/(D^2 - a^2) - a\cos\theta/(D^2 - a^2)}.
\end{aligned}
$$

By inspection, this is the same form as Eq. (163) with $D/(D^2 - a^2) = m\alpha/L^2$ and $a/(D^2 - a^2) = A$.

Let us remind ourselves about what an ellipse is before we proceed.

```
import numpy as np
from matplotlib import pyplot as plt
from math import pi

u=1.      #x-position of the center
v=0.5     #y-position of the center
a=2.      #radius on the x-axis
b=1.5     #radius on the y-axis

t = np.linspace(0, 2*pi, 100)
plt.plot( u+a*np.cos(t) , v+b*np.sin(t) )
plt.grid(color='lightgray',linestyle='--')
plt.show()
```

Effective or Centrifugal Potential

The total energy of a particle is

$$
\begin{aligned}
E &= U(r) + \frac{1}{2}mv_\theta^2 + \frac{1}{2}m\dot{r}^2 \qquad (165)\\
&= U(r) + \frac{1}{2}mr^2\dot{\theta}^2 + \frac{1}{2}m\dot{r}^2\\
&= U(r) + \frac{L^2}{2mr^2} + \frac{1}{2}m\dot{r}^2.
\end{aligned}
$$

The second term then contributes to the energy like an additional repulsive potential. The term is sometimes referred to as the "centrifugal" potential, even though it is actually the kinetic energy of the angular motion. Combined with $U(r)$, it is sometimes referred to as the "effective" potential,

$$U_{\rm eff}(r) = U(r) + \frac{L^2}{2mr^2}. \tag{166}$$

Note that if one treats the effective potential like a real potential, one would expect to be able to generate an effective force,

$$\begin{aligned} F_{\rm eff} &= -\frac{d}{dr}U(r) - \frac{d}{dr}\frac{L^2}{2mr^2} \\ &= F(r) + \frac{L^2}{mr^3} = F(r) + m\frac{v_\perp^2}{r}, \end{aligned} \tag{167}$$

which is indeed matches the form for $m\ddot{r}$ in Eq. (157), which included the **centrifugal** force.

The following code plots this effective potential for a simple choice of parameters, with a standard gravitational potential $-\alpha/r$. Here we have chosen $L = m = \alpha = 1$.

```
# Common imports
import numpy as np
from math import *
import matplotlib.pyplot as plt

Deltax = 0.01
#set up arrays
xinitial = 0.3
xfinal = 5.0
alpha = 1.0   # spring constant
m = 1.0   # mass, you can change these
AngMom = 1.0  #  The angular momentum
n = ceil((xfinal-xinitial)/Deltax)
x = np.zeros(n)
for i in range(n):
    x[i] = xinitial+i*Deltax
V = np.zeros(n)
V = -alpha/x+0.5*AngMom*AngMom/(m*x*x)
# Plot potential
fig, ax = plt.subplots()
ax.set_xlabel('r[m]')
ax.set_ylabel('V[J]')
ax.plot(x, V)
fig.tight_layout()
plt.show()
```

Gravitational force example. Using the above parameters, we can now study the evolution of the system using for example the velocity Verlet method. This is done in the code here for an initial radius equal to the minimum of the potential well. We seen then that the radius is always the same and corresponds to a circle (the radius is always constant).

```
# Common imports
import numpy as np
import pandas as pd
from math import *
import matplotlib.pyplot as plt
import os

# Where to save the figures and data files
PROJECT_ROOT_DIR = "Results"
FIGURE_ID = "Results/FigureFiles"
DATA_ID = "DataFiles/"

if not os.path.exists(PROJECT_ROOT_DIR):
    os.mkdir(PROJECT_ROOT_DIR)

if not os.path.exists(FIGURE_ID):
    os.makedirs(FIGURE_ID)

if not os.path.exists(DATA_ID):
    os.makedirs(DATA_ID)

def image_path(fig_id):
    return os.path.join(FIGURE_ID, fig_id)

def data_path(dat_id):
    return os.path.join(DATA_ID, dat_id)

def save_fig(fig_id):
    plt.savefig(image_path(fig_id) + ".png", format='png')


# Simple Gravitational Force   -alpha/r

DeltaT = 0.01
#set up arrays
tfinal = 100.0
n = ceil(tfinal/DeltaT)
# set up arrays for t, v and r
t = np.zeros(n)
v = np.zeros(n)
r = np.zeros(n)
# Constants of the model, setting all variables to one for simplicity
alpha = 1.0
AngMom = 1.0  #  The angular momentum
m = 1.0  # scale mass to one
c1 = AngMom*AngMom/(m*m)
c2 = AngMom*AngMom/m
rmin = (AngMom*AngMom/m/alpha)
# Initial conditions
r0 = rmin
v0 = 0.0
r[0] = r0
v[0] = v0
# Start integrating using the Velocity-Verlet  method
for i in range(n-1):
    # Set up acceleration
    a = -alpha/(r[i]**2)+c1/(r[i]**3)
    # update velocity, time and position using the Velocity-Verlet method
    r[i+1] = r[i] + DeltaT*v[i]+0.5*(DeltaT**2)*a
    anew = -alpha/(r[i+1]**2)+c1/(r[i+1]**3)
    v[i+1] = v[i] + 0.5*DeltaT*(a+anew)
```

```
    t[i+1] = t[i] + DeltaT
    # Plot position as function of time
fig, ax = plt.subplots(2,1)
ax[0].set_xlabel('time')
ax[0].set_ylabel('radius')
ax[0].plot(t,r)
ax[1].set_xlabel('time')
ax[1].set_ylabel('Velocity')
ax[1].plot(t,v)
save_fig("RadialGVV")
plt.show()
```

Changing the value of the initial position to a value where the energy is positive, leads to an increasing radius with time, a so-called unbound orbit. Choosing on the other hand an initial radius that corresponds to a negative energy and different from the minimum value leads to a radius that oscillates back and forth between two values.

Harmonic Oscillator in two dimensions. Consider a particle of mass m in a 2-dimensional harmonic oscillator with potential

$$U = \frac{1}{2}kr^2 = \frac{1}{2}k(x^2 + y^2).$$

If the orbit has angular momentum L, we can find the radius and angular velocity of the circular orbit as well as the b) the angular frequency of small radial perturbations.

We consider the effective potential. The radius of a circular orbit is at the minimum of the potential (where the effective force is zero). The potential is plotted here with the parameters $k = m = 0.1$ and $L = 1.0$.

```
# Common imports
import numpy as np
from math import *
import matplotlib.pyplot as plt

Deltax = 0.01
#set up arrays
xinitial = 1.0
xfinal = 5.0
k = 0.1   # spring constant
m = 0.1   # mass, you can change these
AngMom = 1.0  #  The angular momentum
n = ceil((xfinal-xinitial)/Deltax)
x = np.zeros(n)
for i in range(n):
    x[i] = xinitial+i*Deltax
V = np.zeros(n)
V = 0.5*k*x*x+0.5*AngMom*AngMom/(m*x*x)
# Plot potential
fig, ax = plt.subplots()
ax.set_xlabel('r[m]')
ax.set_ylabel('V[J]')
ax.plot(x, V)
fig.tight_layout()
plt.show()
```

$$U_{\text{eff}} = \frac{1}{2}kr^2 + \frac{L^2}{2mr^2}$$

The effective potential looks like that of a harmonic oscillator for large r, but for small r, the centrifugal potential repels the particle from the origin. The combination of the two potentials has a minimum for at some radius $r_{\min}$.

$$\begin{aligned} 0 &= kr_{\min} - \frac{L^2}{mr_{\min}^3}, \\ r_{\min} &= \left(\frac{L^2}{mk}\right)^{1/4}, \\ \dot{\theta} &= \frac{L}{mr_{\min}^2} = \sqrt{k/m}. \end{aligned}$$

For particles at $r_{\min}$ with $\dot{r} = 0$, the particle does not accelerate and r stays constant, i.e. a circular orbit. The radius of the circular orbit can be adjusted by changing the angular momentum L.

For the above parameters this minimum is at $r_{\min} = 1$.

Now consider small vibrations about $r_{\min}$. The effective spring constant is the curvature of the effective potential.

$$\begin{aligned} k_{\text{eff}} &= \frac{d^2}{dr^2}U_{\text{eff}}(r)\bigg|_{r=r_{\min}} = k + \frac{3L^2}{mr_{\min}^4} \\ &= 4k, \\ \omega &= \sqrt{k_{\text{eff}}/m} = 2\sqrt{k/m} = 2\dot{\theta}. \end{aligned}$$

Here, the second step used the result of the last step from part (a). Because the radius oscillates with twice the angular frequency, the orbit has two places where r reaches a minimum in one cycle. This differs from the inverse-square force where there is one minimum in an orbit. One can show that the orbit for the harmonic oscillator is also elliptical, but in this case the center of the potential is at the center of the ellipse, not at one of the foci.

The solution is also simple to write down exactly in Cartesian coordinates. The x and y equations of motion separate,

$$\begin{aligned} \ddot{x} &= -kx, \\ \ddot{y} &= -ky. \end{aligned}$$

So the general solution can be expressed as

$$\begin{aligned} x &= A\cos\omega_0 t + B\sin\omega_0 t, \\ y &= C\cos\omega_0 t + D\sin\omega_0 t. \end{aligned}$$

The code here finds the solution for x and y using the code we developed in homework 4.

```
DeltaT = 0.01
#set up arrays
tfinal = 10.0
n = ceil(tfinal/DeltaT)
# set up arrays
t = np.zeros(n)
v = np.zeros((n,2))
r = np.zeros((n,2))
radius = np.zeros(n)
# Constants of the model
k = 0.1   # spring constant
m = 0.1   # mass, you can change these
omega02 = sqrt(k/m)  # Frequency
AngMom = 1.0  #  The angular momentum
rmin = (AngMom*AngMom/k/m)**0.25
# Initial conditions as compact 2-dimensional arrays
#x0 =rmin*0.5; y0 = sqrt(rmin*rmin-x0*x0)
x0 = 1.0; y0= 1.0
r0 = np.array([x0,y0])
v0 = np.array([0.0,0.0])
r[0] = r0
v[0] = v0
# Start integrating using the Velocity-Verlet  method
for i in range(n-1):
    # Set up the acceleration
    a =  -r[i]*omega02
    # update velocity, time and position using the Velocity-Verlet method
    r[i+1] = r[i] + DeltaT*v[i]+0.5*(DeltaT**2)*a
    anew = -r[i+1]*omega02
    v[i+1] = v[i] + 0.5*DeltaT*(a+anew)
    t[i+1] = t[i] + DeltaT
# Plot position as function of time
radius = np.sqrt(r[:,0]**2+r[:,1]**2)
fig, ax = plt.subplots(3,1)
ax[0].set_xlabel('time')
ax[0].set_ylabel('radius squared')
ax[0].plot(t,r[:,0]**2+r[:,1]**2)
ax[1].set_xlabel('time')
ax[1].set_ylabel('x position')
ax[1].plot(t,r[:,0])
ax[2].set_xlabel('time')
ax[2].set_ylabel('y position')
ax[2].plot(t,r[:,1])

fig.tight_layout()
save_fig("2DimHOVV")
plt.show()
```

With some work using double angle formulas, one can calculate

$$
\begin{aligned}
r^2 &= x^2 + y^2 \\
&= (A^2 + C^2)\cos^2(\omega_0 t) + (B^2 + D^2)\sin^2\omega_0 t + (AB + CD)\cos(\omega_0 t)\sin(\omega_0 t) \\
&= \alpha + \beta\cos 2\omega_0 t + \gamma \sin 2\omega_0 t, \\
\alpha &= \frac{A^2 + B^2 + C^2 + D^2}{2}, \quad \beta = \frac{A^2 - B^2 + C^2 - D^2}{2}, \quad \gamma = AB + CD, \\
r^2 &= \alpha + (\beta^2 + \gamma^2)^{1/2}\cos(2\omega_0 t - \delta), \quad \delta = \arctan(\gamma/\beta),
\end{aligned}
$$

and see that radius oscillates with frequency $2\omega_0$. The factor of two comes because the oscillation $x = A\cos\omega_0 t$ has two maxima for x^2, one at $t = 0$ and one a half period later.

The following code shows first how we can solve this problem using the radial degrees of freedom only.

```
DeltaT = 0.01
#set up arrays
tfinal = 10.0
n = ceil(tfinal/DeltaT)
# set up arrays for t, v and r
t = np.zeros(n)
v = np.zeros(n)
r = np.zeros(n)
E = np.zeros(n)
# Constants of the model
AngMom = 1.0  #  The angular momentum
m = 0.1
k = 0.1
omega02 = k/m
c1 = AngMom*AngMom/(m*m)
c2 = AngMom*AngMom/m
rmin = (AngMom*AngMom/k/m)**0.25
# Initial conditions
r0 = rmin
v0 = 0.0
r[0] = r0
v[0] = v0
E[0] = 0.5*m*v0*v0+0.5*k*r0*r0+0.5*c2/(r0*r0)
# Start integrating using the Velocity-Verlet  method
for i in range(n-1):
    # Set up acceleration
    a = -r[i]*omega02+c1/(r[i]**3)
    # update velocity, time and position using the Velocity-Verlet method
    r[i+1] = r[i] + DeltaT*v[i]+0.5*(DeltaT**2)*a
    anew = -r[i+1]*omega02+c1/(r[i+1]**3)
    v[i+1] = v[i] + 0.5*DeltaT*(a+anew)
    t[i+1] = t[i] + DeltaT
    E[i+1] = 0.5*m*v[i+1]*v[i+1]+0.5*k*r[i+1]*r[i+1]+0.5*c2/(r[i+1]*r[i+1])
    # Plot position as function of time
fig, ax = plt.subplots(2,1)
ax[0].set_xlabel('time')
ax[0].set_ylabel('radius')
ax[0].plot(t,r)
ax[1].set_xlabel('time')
ax[1].set_ylabel('Energy')
ax[1].plot(t,E)
```

```
save_fig("RadialHOVV")
plt.show()
```

Stability of Orbits

The effective force can be extracted from the effective potential, U_{eff}. Beginning from the equations of motion, Eq. (155), for r,

$$\begin{aligned} m\ddot{r} &= F + \frac{L^2}{mr^3} \\ &= F_{\text{eff}} \\ &= -\partial_r U_{\text{eff}}, \\ F_{\text{eff}} &= -\partial_r \left[U(r) + (L^2/2mr^2)\right]. \end{aligned} \tag{168}$$

For a circular orbit, the radius must be fixed as a function of time, so one must be at a maximum or a minimum of the effective potential. However, if one is at a maximum of the effective potential the radius will be unstable. For the attractive Coulomb force the effective potential will be dominated by the $-\alpha/r$ term for large r because the centrifugal part falls off more quickly, $\sim 1/r^2$. At low r the centrifugal piece wins and the effective potential is repulsive. Thus, the potential must have a minimum somewhere with negative potential. The circular orbits are then stable to perturbation.

The effective potential is sketched for two cases, a $1/r$ attractive potential and a $1/r^3$ attractive potential. The $1/r$ case has a stable minimum, whereas the circular orbit in the $1/r^3$ case is unstable.

If one considers a potential that falls as $1/r^3$, the situation is reversed and the point where $\partial_r U$ disappears will be a local maximum rather than a local minimum. **Fig to come here with code**

The repulsive centrifugal piece dominates at large r and the attractive Coulomb piece wins out at small r. The circular orbit is then at a maximum of the effective potential and the orbits are unstable. It is the clear that for potentials that fall as r^n, that one must have $n > -2$ for the orbits to be stable.

Consider a potential $U(r) = \beta r$. For a particle of mass m with angular momentum L, find the angular frequency of a circular orbit. Then find the angular frequency for small radial perturbations.

For the circular orbit you search for the position r_{min} where the effective potential is minimized,

$$
\begin{aligned}
\partial_r \left\{ \beta r + \frac{L^2}{2mr^2} \right\} &= 0, \\
\beta &= \frac{L^2}{mr_{\rm min}^3}, \\
r_{\rm min} &= \left(\frac{L^2}{\beta m} \right)^{1/3}, \\
\dot{\theta} &= \frac{L}{mr_{\rm min}^2} = \frac{\beta^{2/3}}{(mL)^{1/3}}
\end{aligned}
$$

Now, we can find the angular frequency of small perturbations about the circular orbit. To do this we find the effective spring constant for the effective potential,

$$
\begin{aligned}
k_{\rm eff} &= \partial_r^2 \, U_{\rm eff}|_{r_{\rm min}} \\
&= \frac{3L^2}{mr_{\rm min}^4}, \\
\omega &= \sqrt{\frac{k_{\rm eff}}{m}} \\
&= \frac{\beta^{2/3}}{(mL)^{1/3}} \sqrt{3}.
\end{aligned}
$$

If the two frequencies, $\dot{\theta}$ and ω, differ by an integer factor, the orbit's trajectory will repeat itself each time around. This is the case for the inverse-square force, $\omega = \dot{\theta}$, and for the harmonic oscillator, $\omega = 2\dot{\theta}$. In this case, $\omega = \sqrt{3}\dot{\theta}$, and the angles at which the maxima and minima occur change with each orbit.

Code example with gravitional force. The code example here is meant to illustrate how we can make a plot of the final orbit. We solve the equations in polar coordinates (the example here uses the minimum of the potential as initial value) and then we transform back to cartesian coordinates and plot x versus y. We see that we get a perfect circle when we place ourselves at the minimum of the potential energy, as expected.

```
# Simple Gravitational Force   -alpha/r

DeltaT = 0.01
#set up arrays
tfinal = 8.0
n = ceil(tfinal/DeltaT)
# set up arrays for t, v and r
t = np.zeros(n)
v = np.zeros(n)
r = np.zeros(n)
phi = np.zeros(n)
```

```
x = np.zeros(n)
y = np.zeros(n)
# Constants of the model, setting all variables to one for simplicity
alpha = 1.0
AngMom = 1.0  #  The angular momentum
m = 1.0  # scale mass to one
c1 = AngMom*AngMom/(m*m)
c2 = AngMom*AngMom/m
rmin = (AngMom*AngMom/m/alpha)
# Initial conditions, place yourself at the potential min
r0 = rmin
v0 = 0.0  # starts at rest
r[0] = r0
v[0] = v0
phi[0] = 0.0
# Start integrating using the Velocity-Verlet  method
for i in range(n-1):
    # Set up acceleration
    a = -alpha/(r[i]**2)+c1/(r[i]**3)
    # update velocity, time and position using the Velocity-Verlet method
    r[i+1] = r[i] + DeltaT*v[i]+0.5*(DeltaT**2)*a
    anew = -alpha/(r[i+1]**2)+c1/(r[i+1]**3)
    v[i+1] = v[i] + 0.5*DeltaT*(a+anew)
    t[i+1] = t[i] + DeltaT
    phi[i+1] = t[i+1]*c2/(r0**2)
# Find cartesian coordinates for easy plot
x = r*np.cos(phi)
y = r*np.sin(phi)
fig, ax = plt.subplots(3,1)
ax[0].set_xlabel('time')
ax[0].set_ylabel('radius')
ax[0].plot(t,r)
ax[1].set_xlabel('time')
ax[1].set_ylabel('Angle $\cos{\phi}$')
ax[1].plot(t,np.cos(phi))
ax[2].set_ylabel('y')
ax[2].set_xlabel('x')
ax[2].plot(x,y)

save_fig("Phasespace")
plt.show()
```

Try to change the initial value for r and see what kind of orbits you get. In order to test different energies, it can be useful to look at the plot of the effective potential discussed above.

However, for orbits different from a circle the above code would need modifications in order to allow us to display say an ellipse. For the latter, it is much easier to run our code in cartesian coordinates, as done here. In this code we test also energy conservation and see that it is conserved to numerical precision. The code here is a simple extension of the code we developed for homework 4.

```
# Common imports
import numpy as np
import pandas as pd
from math import *
import matplotlib.pyplot as plt

DeltaT = 0.01
```

```
#set up arrays
tfinal = 10.0
n = ceil(tfinal/DeltaT)
# set up arrays
t = np.zeros(n)
v = np.zeros((n,2))
r = np.zeros((n,2))
E = np.zeros(n)
# Constants of the model
m = 1.0   # mass, you can change these
alpha = 1.0
# Initial conditions as compact 2-dimensional arrays
x0 = 0.5; y0= 0.
r0 = np.array([x0,y0])
v0 = np.array([0.0,1.0])
r[0] = r0
v[0] = v0
rabs = sqrt(sum(r[0]*r[0]))
E[0] = 0.5*m*(v[0,0]**2+v[0,1]**2)-alpha/rabs
# Start integrating using the Velocity-Verlet  method
for i in range(n-1):
    # Set up the acceleration
    rabs = sqrt(sum(r[i]*r[i]))
    a =  -alpha*r[i]/(rabs**3)
    # update velocity, time and position using the Velocity-Verlet method
    r[i+1] = r[i] + DeltaT*v[i]+0.5*(DeltaT**2)*a
    rabs = sqrt(sum(r[i+1]*r[i+1]))
    anew = -alpha*r[i+1]/(rabs**3)
    v[i+1] = v[i] + 0.5*DeltaT*(a+anew)
    E[i+1] = 0.5*m*(v[i+1,0]**2+v[i+1,1]**2)-alpha/rabs
    t[i+1] = t[i] + DeltaT
# Plot position as function of time
fig, ax = plt.subplots(3,1)
ax[0].set_ylabel('y')
ax[0].set_xlabel('x')
ax[0].plot(r[:,0],r[:,1])
ax[1].set_xlabel('time')
ax[1].set_ylabel('y position')
ax[1].plot(t,r[:,0])
ax[2].set_xlabel('time')
ax[2].set_ylabel('y position')
ax[2].plot(t,r[:,1])

fig.tight_layout()
save_fig("2DimGravity")
plt.show()
print(E)
```

Scattering and Cross Sections

Scattering experiments don't measure entire trajectories. For elastic collisions, they measure the distribution of final scattering angles at best. Most experiments use targets thin enough so that the number of scatterings is typically zero or one. The cross section, σ, describes the cross-sectional area for particles to scatter with an individual target atom or nucleus. Cross section measurements form the basis for MANY fields of physics. BThe cross section, and the differential cross section, encapsulates everything measurable for a collision where all that

is measured is the final state, e.g. the outgoing particle had momentum $\boldsymbol{p}_f$. y studying cross sections, one can infer information about the potential interaction between the two particles. Inferring, or constraining, the potential from the cross section is a classic *inverse* problem. Collisions are either elastic or inelastic. Elastic collisions are those for which the two bodies are in the same internal state before and after the collision. If the collision excites one of the participants into a higher state, or transforms the particles into different species, or creates additional particles, the collision is inelastic. Here, we consider only elastic collisions.

For Coulomb forces, the cross section is infinite because the range of the Coulomb force is infinite, but for interactions such as the strong interaction in nuclear or particle physics, there is no long-range force and cross-sections are finite. Even for Coulomb forces, the part of the cross section that corresponds to a specific scattering angle, $d\sigma/d\Omega$, which is a function of the scattering angle θ_s is still finite.

If a particle travels through a thin target, the chance the particle scatters is $P_{\text{scatt}} = \sigma dN/dA$, where dN/dA is the number of scattering centers per area the particle encounters. If the density of the target is ρ particles per volume, and if the thickness of the target is t, the areal density (number of target scatterers per area) is $dN/dA = \rho t$. Because one wishes to quantify the collisions independently of the target, experimentalists measure scattering probabilities, then divide by the areal density to obtain cross-sections,

$$\sigma = \frac{P_{\text{scatt}}}{dN/dA}. \tag{169}$$

Instead of merely stating that a particle collided, one can measure the probability the particle scattered by a given angle. The scattering angle θ_s is defined so that at zero the particle is unscattered and at $\theta_s = \pi$ the particle is scattered directly backward. Scattering angles are often described in the center-of-mass frame, but that is a detail we will neglect for this first discussion, where we will consider the scattering of particles moving classically under the influence of fixed potentials $U(\boldsymbol{r})$. Because the distribution of scattering angles can be measured, one expresses the differential cross section,

$$\frac{d^2\sigma}{d\cos\theta_s \; d\phi}. \tag{170}$$

Usually, the literature expresses differential cross sections as

$$d\sigma/d\Omega = \frac{d\sigma}{d\cos\theta d\phi} = \frac{1}{2\pi}\frac{d\sigma}{d\cos\theta}, \tag{171}$$

where the last equivalency is true when the scattering does not depend on the azimuthal angle ϕ, as is the case for spherically symmetric potentials.

The differential solid angle $d\Omega$ can be thought of as the area subtended by a measurement, dA_d, divided by r^2, where r is the distance to the detector,

$$dA_d = r^2 d\Omega. \tag{172}$$

With this definition $d\sigma/d\Omega$ is independent of the distance from which one places the detector, or the size of the detector (as long as it is small).

Differential scattering cross sections are calculated by assuming a random distribution of impact parameters b. These represent the distance in the xy plane for particles moving in the z direction relative to the scattering center. An impact parameter $b = 0$ refers to being aimed directly at the target's center. The impact parameter describes the transverse distance from the $z = 0$ axis for the trajectory when it is still far away from the scattering center and has not yet passed it. The differential cross section can be expressed in terms of the impact parameter,

$$d\sigma = 2\pi b db, \tag{173}$$

which is the area of a thin ring of radius b and thickness db. In classical physics, one can calculate the trajectory given the incoming kinetic energy E and the impact parameter if one knows the mass and potential. From the trajectory, one then finds the scattering angle $\theta_s(b)$. The differential cross section is then

$$\frac{d\sigma}{d\Omega} = \frac{1}{2\pi}\frac{d\sigma}{d\cos\theta_s} = b\frac{db}{d\cos\theta_s} = \frac{b}{(d/db)\cos\theta_s(b)}. \tag{174}$$

Typically, one would calculate $\cos\theta_s$ and $(d/db)\cos\theta_s$ as functions of b. This is sufficient to plot the differential cross section as a function of θ_s.

The total cross section is

$$\sigma_{\rm tot} = \int d\Omega \frac{d\sigma}{d\Omega} = 2\pi \int d\cos\theta_s \ \frac{d\sigma}{d\Omega}. \tag{175}$$

Even if the total cross section is infinite, e.g. Coulomb forces, one can still have a finite differential cross section as we will see later on.

An asteroid of mass m and kinetic energy E approaches a planet of radius R and mass M. What is the cross section for the asteroid to impact the planet?

Solution. Calculate the maximum impact parameter, $b_{\rm max}$, for which the asteroid will hit the planet. The total cross section for impact is $\sigma_{\rm impact} = \pi b_{\rm max}^2$. The maximum cross-section can be found with the help of angular momentum conservation. The asteroid's incoming momentum is $p_0 = \sqrt{2mE}$ and the angular momentum is $L = p_0 b$. If the asteroid just grazes the planet, it is moving with zero radial kinetic energy at impact. Combining energy and angular momentum conservation and having p_f refer to the momentum of the asteroid at a distance R,

$$\begin{aligned} \frac{p_f^2}{2m} - \frac{GMm}{R} &= E, \\ p_f R &= p_0 b_{\rm max}, \end{aligned}$$

allows one to solve for $b_{\max}$,

$$\begin{aligned} b_{\max} &= R\frac{p_f}{p_0} \\ &= R\frac{\sqrt{2m(E+GMm/R)}}{\sqrt{2mE}} \\ \sigma_{\text{impact}} &= \pi R^2 \frac{E+GMm/R}{E}. \end{aligned}$$

Rutherford Scattering

This refers to the calculation of $d\sigma/d\Omega$ due to an inverse square force, $F_{12} = \pm\alpha/r^2$ for repulsive/attractive interaction. Rutherford compared the scattering of α particles (^{4}He nuclei) off of a nucleus and found the scattering angle at which the formula began to fail. This corresponded to the impact parameter for which the trajectories would strike the nucleus. This provided the first measure of the size of the atomic nucleus. At the time, the distribution of the positive charge (the protons) was considered to be just as spread out amongst the atomic volume as the electrons. After Rutherford's experiment, it was clear that the radius of the nucleus tended to be roughly 4 orders of magnitude smaller than that of the atom, which is less than the size of a football relative to Spartan Stadium.

The incoming and outgoing angles of the trajectory are at $\pm\theta'$. They are related to the scattering angle by $2\theta' = \pi + \theta_s$.

In order to calculate differential cross section, we must find how the impact parameter is related to the scattering angle. This requires analysis of the trajectory. We consider our previous expression for the trajectory where we derived the elliptic form for the trajectory, Eq. (163). For that case we considered an attractive force with the particle's energy being negative, i.e. it was bound. However, the same form will work for positive energy, and repulsive forces can be considered by simple flipping the sign of α. For positive energies, the trajectories will be hyperbolas, rather than ellipses, with the asymptotes of the trajectories representing the directions of the incoming and outgoing tracks. Rewriting Eq. (163),

$$r = \frac{1}{\frac{m\alpha}{L^2} + A\cos\theta}. \tag{176}$$

Once A is large enough, which will happen when the energy is positive, the denominator will become negative for a range of θ. This is because the scattered particle will never reach certain angles. The asymptotic angles θ' are those for which the denominator goes to zero,

$$\cos\theta' = -\frac{m\alpha}{AL^2}. \tag{177}$$

The trajectory's point of closest approach is at $\theta = 0$ and the two angles θ', which have this value of $\cos\theta'$, are the angles of the incoming and outgoing

particles. From Fig (**to come**), one can see that the scattering angle θ_s is given by,

$$\begin{aligned} 2\theta' - \pi &= \theta_s, \quad \theta' = \frac{\pi}{2} + \frac{\theta_s}{2}, \\ \sin(\theta_s/2) &= -\cos\theta' \\ &= \frac{m\alpha}{AL^2}. \end{aligned} \tag{178}$$

Now that we have θ_s in terms of m, α, L and A, we wish to re-express L and A in terms of the impact parameter b and the energy E. This will set us up to calculate the differential cross section, which requires knowing $db/d\theta_s$. It is easy to write the angular momentum as

$$L^2 = p_0^2 b^2 = 2mEb^2. \tag{179}$$

Finding A is more complicated. To accomplish this we realize that the point of closest approach occurs at $\theta = 0$, so from Eq. (176)

$$\begin{aligned} \frac{1}{r_{\min}} &= \frac{m\alpha}{L^2} + A, \\ A &= \frac{1}{r_{\min}} - \frac{m\alpha}{L^2}. \end{aligned} \tag{180}$$

Next, $r_{\min}$ can be found in terms of the energy because at the point of closest approach the kinetic energy is due purely to the motion perpendicular to $\hat{r}$ and

$$E = -\frac{\alpha}{r_{\min}} + \frac{L^2}{2mr_{\min}^2}. \tag{181}$$

One can solve the quadratic equation for $1/r_{\min}$,

$$\frac{1}{r_{\min}} = \frac{m\alpha}{L^2} + \sqrt{(m\alpha/L^2)^2 + 2mE/L^2}. \tag{182}$$

We can plug the expression for $r_{\min}$ into the expression for A, Eq. (180),

$$A = \sqrt{(m\alpha/L^2)^2 + 2mE/L^2} = \sqrt{(\alpha^2/(4E^2b^4) + 1/b^2} \tag{183}$$

Finally, we insert the expression for A into that for the scattering angle, Eq. (178),

$$\begin{aligned} \sin(\theta_s/2) &= \frac{m\alpha}{AL^2} \\ &= \frac{a}{\sqrt{a^2 + b^2}}, \quad a \equiv \frac{\alpha}{2E} \end{aligned} \tag{184}$$

The differential cross section can now be found by differentiating the expression for θ_s with b,

$$\begin{aligned}
\frac{1}{2}\cos(\theta_s/2)d\theta_s &= \frac{ab\ db}{(a^2+b^2)^{3/2}} = \frac{bdb}{a^2}\sin^3(\theta_s/2), \qquad (185)\\
d\sigma &= 2\pi b db = \frac{\pi a^2}{\sin^3(\theta_s/2)}\cos(\theta_s/2)d\theta_s\\
&= \frac{\pi a^2}{2\sin^4(\theta_s/2)}\sin\theta_s d\theta_s\\
\frac{d\sigma}{d\cos\theta_s} &= \frac{\pi a^2}{2\sin^4(\theta_s/2)},\\
\frac{d\sigma}{d\Omega} &= \frac{a^2}{4\sin^4(\theta_s/2)}.
\end{aligned}$$

where $a = \alpha/2E$. This the Rutherford formula for the differential cross section. It diverges as $\theta_s \to 0$ because scatterings with arbitrarily large impact parameters still scatter to arbitrarily small scattering angles. The expression for $d\sigma/d\Omega$ is the same whether the interaction is positive or negative.

Consider a particle of mass m and charge z with kinetic energy E (Let it be the center-of-mass energy) incident on a heavy nucleus of mass M and charge Z and radius R. Find the angle at which the Rutherford scattering formula breaks down.

Solution. Let $\alpha = Zze^2/(4\pi\epsilon_0)$. The scattering angle in Eq. (184) is

$$\sin(\theta_s/2) = \frac{a}{\sqrt{a^2+b^2}}, \quad a \equiv \frac{\alpha}{2E}.$$

The impact parameter b for which the point of closest approach equals R can be found by using angular momentum conservation,

$$\begin{aligned}
p_0 b &= b\sqrt{2mE} = Rp_f = R\sqrt{2m(E-\alpha/R)},\\
b &= R\frac{\sqrt{2m(E-\alpha/R)}}{\sqrt{2mE}}\\
&= R\sqrt{1-\frac{\alpha}{ER}}.
\end{aligned}$$

Putting these together

$$\theta_s = 2\sin^{-1}\left\{\frac{a}{\sqrt{a^2+R^2(1-\alpha/(RE))}}\right\}, \quad a = \frac{\alpha}{2E}.$$

It was from this departure of the experimentally measured $d\sigma/d\Omega$ from the Rutherford formula that allowed Rutherford to infer the radius of the gold nucleus, R.

Just like electrodynamics, one can define "fields", which for a small additional mass m are the force per mass and the additional potential energy per mass. The *gravitational field* related to the force has dimensions of force per mass, or acceleration, and can be labeled $\boldsymbol{g}(\boldsymbol{r})$. The potential energy per mass has dimensions of energy per mass. This is analogous to the electromagnetic potential, which is the potential energy per charge, and the electric field which is the force per charge.

Because the field $\boldsymbol{g}$ obeys the same inverse square law for a point mass as the electric field does for a point charge, the gravitational field also satisfies a version of Gauss's law,

$$\oint d\boldsymbol{A} \cdot \boldsymbol{g} = -4\pi G M_{\text{inside}}. \tag{186}$$

Here, M_{inside} is the net mass inside a closed area.

Gauss's law can be understood by considering a nozzle that sprays paint in all directions uniformly from a point source. Let B be the number of gallons per minute of paint leaving the nozzle. If the nozzle is at the center of a sphere of radius r, the paint per square meter per minute that is deposited on some part of the sphere is

$$F(r) \quad = \quad \frac{B}{4\pi r^2}. \tag{187}$$

Now, let F also be assigned a direction, so that it becomes a vector pointing along the direction of the flying paint. For any surface that surrounds the nozzle, not necessarily a sphere, one can state that

$$\oint \boldsymbol{dA} \cdot \boldsymbol{F} \quad = \quad B, \tag{188}$$

regardless of the shape of the surface. This follows because the rate at which paint is deposited on the surface should equal the rate at which it leaves the nozzle. The dot product ensures that only the component of $\boldsymbol{F}$ into the surface contributes to the deposition of paint. Similarly, if $\boldsymbol{F}$ is any radial inverse-square forces, that falls as $B/(4\pi r^2)$, then one can apply Eq. (188). For gravitational fields, $B/(4\pi)$ is replaced by GM, and one quickly "derives" Gauss's law for gravity, Eq. (186).

Consider Earth to have its mass M uniformly distributed in a sphere of radius R. Find the magnitude of the gravitational acceleration as a function of the radius r in terms of the acceleration of gravity at the surface $g(R)$. Assume $r < R$, i.e. you are inside the surface.

Solution: Take the ratio of Eq. (186) for two radii, R and $r < R$,

$$\begin{aligned}\frac{4\pi r^2 g(r)}{4\pi R^2 g(R)} &= \frac{4\pi G M_{\text{inside r}}}{4\pi G M_{\text{inside R}}} \\ &= \frac{r^3}{R^3} \\ g(r) &= g(R)\frac{r}{R}\ .\end{aligned}$$

The potential energy per mass is similar conceptually to the voltage, or electric potential energy per charge, that was studied in electromagnetism, if $V \equiv U/m$, $\boldsymbol{g} = -\nabla V$.

Tidal Forces

Consider a spherical planet of radius r a distance D from another body of mass M. The magnitude of the force due to M on an small object of mass δm on surface of the planet can be calculated by performing a Taylor expansion about the center of the spherical planet.

$$F = -\frac{GM\delta m}{D^2} + 2\frac{GM\delta m}{D^3}\Delta D + \cdots \tag{189}$$

If the z direction points toward the large object, ΔD can be referred to as z. In the accelerating frame of an observer at the center of the planet,

$$\delta m\frac{d^2 z}{dt^2} = F - \delta m a' + \text{other forces acting on } \delta m, \tag{190}$$

where a' is the acceleration of the observer. Because $\delta m a'$ equals the gravitational force on δm if it were located at the planet's center, one can write

$$m\frac{d^2 z}{dt^2} = 2\frac{GM\delta m}{D^3}z + \text{other forces acting on } \delta m. \tag{191}$$

Here the other forces could represent the forces acting on δm from the spherical planet such as the gravitational force or the contact force with the surface. If θ is the angle w.r.t. the z axis, the effective force acting on δm is

$$F_{\text{eff}} \approx 2\frac{GM\delta m}{D^3}r\cos\theta\hat{z} + \text{other forces acting on } \delta m. \tag{192}$$

This first force is the "tidal" force. It pulls objects outward from the center of the object. If the object were covered with water, it would distort the objects shape so that the shape would be elliptical, stretched out along the axis pointing toward the large mass M. The force is always along (either parallel or antiparallel to) the $\hat{z}$ direction.

Consider the Earth to be a sphere of radius R covered with water, with the gravitational acceleration at the surface noted by g. Now assume that a distant body provides an additional constant gravitational acceleration $\boldsymbol{a}$ pointed along

the z axis. Find the distortion of the radius as a function of θ. Ignore planetary rotation and assume $a << g$.

Solution: Because Earth would then accelerate with a, the field a would seem invisible in the accelerating frame. A tidal force would only appear if a depended on position, i.e. $\nabla \boldsymbol{a} \neq 0$.

Now consider that the field is no longer constant, but that instead $a = -kz$ with $|kR| << g$.

Solution: The surface of the planet needs to be at constant potential (if the planet is not accelerating). The force per mass, $-kz$ is like a spring, and the potential per mass is $kz^2/2$. Otherwise water would move to a point of lower potential. Thus, the potential energy for a sample mass δm is

$$V(R) + \delta m g h(\theta) - \frac{\delta m}{2} k r^2 \cos^2\theta = \text{Constant}$$

$$V(R) + \delta m g h(\theta) - \frac{\delta m}{2} k R^2 \cos^2\theta - \delta m k R h(\theta) \cos^2\theta - \frac{\delta m}{2} k h^2(\theta) \cos^2\theta = \text{Constant}.$$

Here, the potential due to the external field is $(1/2)kz^2$ so that $-\nabla U = -kz$. One now needs to solve for $h(\theta)$. Absorbing all the constant terms from both sides of the equation into one constant C, and because both h and kR are small, we can through away terms of order h^2 or kRh. This gives

$$\begin{aligned} g h(\theta) - \frac{1}{2} k R^2 \cos^2\theta &= C, \\ h(\theta) &= \frac{C}{g} + \frac{1}{2g} k R^2 \cos^2\theta, \\ h(\theta) &= \frac{1}{2g} k R^2 (\cos^2\theta - 1/3). \end{aligned}$$

The term with the factor of $1/3$ replaced the constant and was chosen so that the average height of the water would be zero.

The Sun's mass is 27×10^6 the Moon's mass, but the Sun is 390 times further away from Earth as the Sun. What is ratio of the tidal force of the Sun to that of the Moon.

Solution: The gravitational force due to an object M a distance D away goes as M/D^2, but the tidal force is only the difference of that force over a distance R,

$$F_{\text{tidal}} \propto \frac{M}{D^3} R.$$

Therefore the ratio of force is

$$\begin{aligned} \frac{F_{\text{Sun's tidal force}}}{F_{\text{Moon's tidal force}}} &= \frac{M_{\text{sun}}/D^3_{\text{sun}}}{M_{\text{moon}}/D^3_{\text{moon}}} \\ &= \frac{27 \times 10^6}{390^3} = 0.46. \end{aligned}$$

The Moon more strongly affects tides than the Sun.

Variational Calculus and Lagrangian Formalism

The calculus of variations involves problems where the quantity to be minimized or maximized is an integral.

The usual minimization problem one faces involves taking a function $\mathcal{L}(x)$, then finding the single value x for which $\mathcal{L}$ is either a maximum or minimum. In multivariate calculus one also learns to solve problems where you minimize for multiple variables, $\mathcal{L}(x_1, x_2, \cdots x_n)$, and finding the points $(x_1 \cdots y_n)$ in an n-dimensional space that maximize or minimize the function. Here, we consider what seems to be a much more ambitious problem. Imagine you have a function $\mathcal{L}(x(t), \dot{x}(t), t)$, and you wish to find the extrema for an infinite number of values of x, i.e. x at each point t. The function $\mathcal{L}$ will not only depend on x at each point t, but also on the slope at each point, plus an additional dependence on t. Note we are NOT finding an optimum value of t, we are finding the set of optimum values of x at each point t, or equivalently, finding the function $x(t)$.

One treats the function $x(t)$ as being unknown while minimizing the action

$$S = \int_{t_1}^{t_2} dt\ \mathcal{L}(x(t), \dot{x}(t), t).$$

Thus, we are minimizing S with respect to an infinite number of values of $x(t_i)$ at points t_i. As an additional criteria, we will assume that $x(t_1)$ and $x(t_2)$ are fixed, and that that we will only consider variations of x between the boundaries. The dependence on the derivative, $\dot{x} = dx/dt$, is crucial because otherwise the solution would involve simply finding the one value of x that minimized $\mathcal{L}$, and $x(t)$ would equal a constant if there were no explicit t dependence. Furthermore, x wouldn't need to be continuous at the boundary.

In the general case we have an integral of the type

$$S[q] = \int_{t_1}^{t_2} \mathcal{L}(q(t), \dot{q}(t), t)dt,$$

where S is the quantity which is sought minimized or maximized. The problem is that although $\mathcal{L}$ is a function of the general variables $q(t), \dot{q}(t), t$ (note our change of variables), the exact dependence of q on t is not known. This means again that even though the integral has fixed limits t_1 and t_2, the path of integration is not known. In our case the unknown quantities are the positions and general velocities of a given number of objects and we wish to choose an integration path which makes the functional $S[q]$ stationary. This means that we want to find minima, or maxima or saddle points. In physics we search normally for minima. Our task is therefore to find the minimum of $S[q]$ so that its variation δS is zero subject to specific constraints. The constraints can be treated via the technique of Lagrangian multipliers as we will see below.

We assume the existence of an optimum path, that is a path for which $S[q]$ is stationary. There are infinitely many such paths. The difference between two paths δq is called the variation of q.

We call the variation $\eta(t)$ and it is scaled by a factor α. The function $\eta(t)$ is arbitrary except for

$$\eta(t_1) = \eta(t_2) = 0,$$

and we assume that we can model the change in q as

$$q(t, \alpha) = q(t) + \alpha\eta(t),$$

and

$$\delta q = q(t, \alpha) - q(t, 0) = \alpha\eta(t).$$

We choose $q(t, \alpha = 0)$ as the unkonwn path that will minimize S. The value $q(t, \alpha \neq 0)$ describes a neighbouring path.

We have

$$S[q(\alpha)] = \int_{t_1}^{t_2} \mathcal{L}(q(t, \alpha), \dot{q}(t, \alpha), t)dt.$$

The condition for an extreme of

$$S[q(\alpha)] = \int_{t_1}^{t_2} \mathcal{L}(q(t, \alpha), \dot{q}(t, \alpha), t)dt,$$

is

$$\left[\frac{\partial S[q(\alpha)]}{\partial t}\right]_{\alpha=0} = 0.$$

The α dependence is contained in $q(t, \alpha)$ and $\dot{q}(t, \alpha)$ meaning that

$$\left[\frac{\partial E[q(\alpha)]}{\partial \alpha}\right] = \int_{t_1}^{t_2} \left(\frac{\partial \updownarrow}{\partial q}\frac{\partial q}{\partial \alpha} + \frac{\partial \mathcal{L}}{\partial \dot{q}}\frac{\partial \dot{q}}{\partial \alpha}\right) dt.$$

We have defined

$$\frac{\partial q(x, \alpha)}{\partial \alpha} = \eta(x)$$

and thereby

$$\frac{\partial \dot{q}(t, \alpha)}{\partial \alpha} = \frac{d(\eta(t))}{dt}.$$

Using

$$\frac{\partial q(t, \alpha)}{\partial \alpha} = \eta(t),$$

and

$$\frac{\partial \dot{q}(t,\alpha)}{\partial \alpha} = \frac{d(\eta(t))}{dt},$$

in the integral gives

$$\left[\frac{\partial S[q(\alpha)]}{\partial \alpha}\right] = \int_{t_1}^{t_2} \left(\frac{\partial \mathcal{L}}{\partial q}\eta(t) + \frac{\partial \mathcal{L}}{\partial \dot{q}}\frac{d(\eta(t))}{dt}\right) dt.$$

Integrating the second term by parts

$$\int_{t_1}^{t_2} \frac{\partial \mathcal{L}}{\partial \dot{q}}\frac{d(\eta(t))}{dt}dt = \eta(t)\frac{\partial \mathcal{L}}{\partial \dot{q}}|_{t_1}^{t_2} - \int_a^b \eta(t)\frac{d}{dx}\frac{\partial \mathcal{L}}{\partial \dot{q}}dt,$$

and since the first term dissappears due to $\eta(a) = \eta(b) = 0$, we obtain

$$\left[\frac{\partial S[q(\alpha)]}{\partial \alpha}\right] = \int_{t_1}^{t_2} \left(\frac{\partial \mathcal{L}}{\partial q} - \frac{d}{dx}\frac{\partial \mathcal{L}}{\partial \dot{q}}\right)\eta(t)dt = 0,$$

which can also be written as

$$\left[\frac{\partial S[q(\alpha)]}{\partial \alpha}\right]_{\alpha=0} = \int_{t_1}^{t_2} \left(\frac{\partial \mathcal{L}}{\partial q} - \frac{d}{dx}\frac{\partial \mathcal{L}}{\partial \dot{q}}\right)\delta q(t)dt = \delta S = 0.$$

The condition for a stationary value is thus a partial differential equation

$$\frac{\partial \mathcal{L}}{\partial q} - \frac{d}{dx}\frac{\partial \mathcal{L}}{\partial \dot{q}} = 0,$$

known as the **Euler-Lagrange** equation.

Constrained motion

Sometimes an auxiliary constraint is added to the problem (beyond fixing the end poits y_1 and y_2). Just ahead, we will work on the example of a hanging chain. The shape of the curve minimizes the potential energy, under the constraint of a fixed length of chain. Before presenting such an example we first review the method of Lagrange multipliers as a method for finding minima or maxima under constraints.

Imagine a function $f(x_1, x_2 \cdots x_n)$ for which you wish to find the minima. Additionally, you are given a constraint

$$C(x_1 \cdots x_n) = 0 \tag{193}$$

The usual condition for a a minimum is

$$\frac{\partial f}{\partial x_i} = 0, \quad \text{or } \nabla f = 0. \tag{194}$$

which would be n equations for the n variables. The gradient of a scalar is a vector, so you should think of ∇ as $\boldsymbol{\nabla}$. However, the solution will likely not satisfy the constraint, i.e. the point at which $f(x_1 \cdots x_n)$ has an extrema, may not be a point where $C(x_1 \cdots x_n) = 0$.

A necessary condition for the solution is that

$$\nabla f \cdot \boldsymbol{\epsilon} = 0, \tag{195}$$

for any infinitesimal vector $\boldsymbol{\epsilon}$ if $\boldsymbol{\epsilon}$ satisfies the condition

$$\delta C = \nabla C \cdot \boldsymbol{\epsilon} = 0. \tag{196}$$

That is to say if I take a small step in a direction that doesn't change the constraint, then f must not change if it is an extrema. Not changing the constraint implies the step is orthogonal to ∇C. As there are n dimensions of x, the vector ∇C defines one direction, and $\boldsymbol{\epsilon}$ can be in any of the $n-1$ directions orthogonal to ∇C. If $\nabla f \cdot \boldsymbol{\epsilon} = 0$ for ANY of the $n-1$ directions of $\boldsymbol{\epsilon}$ orthogonal to ∇C, then

$$\nabla f \mid\mid \nabla C. \tag{197}$$

Because the two vectors are parallel you can say there must exist some constant λ such that

$$\nabla(f - \lambda C) = 0. \tag{198}$$

Here, λ is known as a Lagrange multiplier. Satisfying this equation is a necessary, but not a sufficient condition. One could add a constant to the constraint and the gradient would not change. One must find the correct value of λ that satisfies the constraint $C = 0$, rather than $C =$ some other constant. The strategy is then to solve thw above equation then adjust λ until one finds the $x_1 \cdots x_n$ that gives $C(x_1 \cdots x_n) = 0$.

The method of Lagrange multipliers is counter-intuitive to one's intuition to use the constraint to reduce the dimensionality of the problem. Normally, minimizing a function of n variables, leads to n equations and n unknowns. A constraint could be used, by substitution, to replace the n variables with $n-1$ variables. Instead, we add an unknown parameter, λ, and change the equation to $n+1$ equations with $n+1$ unknowns, with the extra unknown being the Lagrange multiplier λ. Often, it is rather easy to solve for $x_1 \cdots x_n$. Then one is left with the usually difficult problem of finding λ, often requiring the solution of a transcendental equation.

Let us try to formalize this. We consider a function of three independent variables $f(x, y, z)$. For the function f to be an extreme we have

$$df = 0.$$

A necessary and sufficient condition is

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial y} = \frac{\partial f}{\partial z} = 0,$$

due to

$$df = \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy + \frac{\partial f}{\partial z}dz.$$

In physical problems the variables x, y, z are often subject to constraints (in our case q and the orthogonality constraint) so that they are no longer all independent. It is possible at least in principle to use each constraint to eliminate one variable and to proceed with a new and smaller set of independent varables.

The use of so-called Lagrangian multipliers is an alternative technique when the elimination of of variables is incovenient or undesirable. Assume that we have an equation of constraint on the variables x, y, z

$$\phi(x, y, z) = 0,$$

resulting in

$$d\phi = \frac{\partial \phi}{\partial x}dx + \frac{\partial \phi}{\partial y}dy + \frac{\partial \phi}{\partial z}dz = 0.$$

Now we cannot set anymore

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial y} = \frac{\partial f}{\partial z} = 0,$$

if $df = 0$ is wanted because there are now only two independent variables! Assume x and y are the independent variables. Then dz is no longer arbitrary.

However, we can add to

$$df = \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy + \frac{\partial f}{\partial z}dz,$$

a multiplum of $d\phi$, viz. $\lambda d\phi$, resulting in

$$df + \lambda d\phi = (\frac{\partial f}{\partial z} + \lambda\frac{\partial \phi}{\partial x})dx + (\frac{\partial f}{\partial y} + \lambda\frac{\partial \phi}{\partial y})dy + (\frac{\partial f}{\partial z} + \lambda\frac{\partial \phi}{\partial z})dz = 0.$$

Our multiplier is chosen so that

$$\frac{\partial f}{\partial z} + \lambda\frac{\partial \phi}{\partial z} = 0.$$

However, we took dx and dy as to be arbitrary and thus we must have

$$\frac{\partial f}{\partial x} + \lambda\frac{\partial \phi}{\partial x} = 0,$$

and

$$\frac{\partial f}{\partial y} + \lambda \frac{\partial \phi}{\partial y} = 0.$$

When all these equations are satisfied, $df = 0$. We have four unknowns, x, y, z and λ. Actually we want only x, y, z, λ need not to be determined, it is therefore often called Lagrange's undetermined multiplier. If we have a set of constraints ϕ_k we have the equations

$$\frac{\partial f}{\partial x_i} + \sum_k \lambda_k \frac{\partial \phi_k}{\partial x_i} = 0.$$

Example: brachiostone. Consider a particle constrained to move along a path (like a bead moving without friction on a wire) and you need to design a path from $x = y = 0$ to some final point x_f, y_f. Assume there is a constant force in the x direction, $F_x = mg$. Design the path so that the time the bead travels is a minimum.

The net time is

$$T = \int \frac{d\ell}{v} = \int_0^{x_f} dx \ \frac{\sqrt{1 + y'^2}}{\sqrt{2gx}} = \text{minimum}.$$

Here we made use of the fact that $d\ell = \sqrt{dx^2 + dy^2}$ and that the velocity is determined by $KE = mv^2/2 = mgx$. The Euler equations can be applied if you first define the function as

$$f(y, y'; x) \quad = \quad \frac{\sqrt{1 + y'^2}}{\sqrt{x}}.$$

The equations are then

$$\frac{d}{dx}\frac{\partial f}{\partial y'} \quad = \quad 0.$$

The simplification ensued from f not having any dependence on y. This yields the differential equation

$$\frac{y'}{x^{1/2}(1 + y'^2)^{1/2}} \quad = \quad (2a)^{-1/2}, \tag{199}$$

because $\partial f/\partial y'$ must be a constant, which with some foresight we label $(2a)^{-1/2}$. One can now solve for y',

$$
\begin{aligned}
(y')^2 &= 2ax(1+y'^2) \\
y' &= \sqrt{\frac{x}{2a-x}}, \\
y(t) &= \int_0^x dx' \, \frac{\sqrt{x'}dx'}{\sqrt{2a-x'}} = \int_0^x dx' \, \frac{x'dx'}{\sqrt{2ax'-x'^2}} \\
&= \frac{1}{2}\int_0^x \frac{(2x'-2a)dx'}{(2ax'-x'^2)^{1/2}} + a\int_0^x \frac{dx'}{\sqrt{2ax'-x'^2}} \\
&= \frac{-1}{2}\int_0^{2ax-x^2} \frac{du}{\sqrt{u}} + a\int_0^x \frac{dx'}{\sqrt{a^2-(x'-a)^2}} \\
&= -\sqrt{2ax-x^2} + a\cos^{-1}(1-x/a).
\end{aligned}
$$

This turns out to be the equation for a *cycloid* or a *brachiostone*. If you rolled a wheel of radius a down the y axis and followed a point on the rim, it would trace out a cycloid. Here, the constant a must be chosen to match the boundary condition, $y_2 = y(x_2)$. You can see the textbook for more details, plus you get a chance to work with cycloids in the exercises at the end of this chapter.

Maximizing a Function. As an example of using Lagrange multipliers for a standard optimization formula we attempt to maximize the following function,

$$F(x_1 \cdots x_n) = -\sum_{i=1}^{n} x_i \ln(x_i),$$

with respect to the n variables x_i. With no constraints, each x_i would maximize the function for

$$
\begin{aligned}
\frac{d}{dx_j}\left[-\sum_i x_i \ln(x_i)\right] &= 0 \\
-\ln(x_j) - 1 &= 0, \quad x_j = e^{-1}.
\end{aligned}
$$

Now, we repeat the problem but with two constraints,

$$\sum_i x_i = 1 \ , \qquad \sum_i x_i \epsilon_i = E.$$

Here, ϵ_i and E are fixed constants. We go forward by finding the extrema for

$$G(x_1 \cdots x_n) = F - \alpha \sum_i x_i - \beta \sum_i \epsilon_i x_i = \sum_i \left\{-x_i \ln(x_i) - \alpha x_i - \beta \epsilon_i x_i\right\}.$$

There are two Lagrange multipliers, α and β, corresponding to the two constraints. One then solves for the extrema

$$\begin{aligned}\frac{d}{dx_j}G &= 0 \\ &= -\ln(x_j) - 1 - \alpha - \beta\epsilon_j, \\ x_j &= \exp\left\{-1-\alpha-\beta\epsilon_j\right\}.\end{aligned}$$

For any given α and β this provides a solution for constraining $\sum_i x_i$ and $\sum_i \epsilon_i x_i$ to some values, just not the values of unity and E that you wish. One would then have to search for the correct values by adjusting α and β until the constraint are actually matched by solving a transcendental equation. Although this can be complicated, it is certainly less expensive than searching over all N values of x_i. This particular example corresponds to maximizing the entropy for a system, $S = -\sum_i x_i \ln(x_i)$, where x_i is the probability of the system being in a particular discrete level i that has energy ϵ_i. One wishes to maximize the entropy subject to the constraints that the probabilities sum to unity and the average energy has some given value. The result that $x_i \sim e^{-\beta\epsilon_i}$ demonstrates the origin of the Boltzmann factor, with the inverse temperature $\beta = 1/T$.

Lagrange multipliers also assist with the Euler-Lagrange equation. If one breaks an interval $x_1 < x < x_2$ into a large number $n \to \infty$ points separated by dx, the Euler-Lagrange equation involves finding the n values y_i at each point so that $\sum_i dx f\left\{y_i, y_i' = (y_{i+1} - y_{i-1})/(2dx)\right\}$ is maximized for some given function f. If an additional auxiliary constraint is added, also some function of the n values y_i, one can use the method of Lagrange multipliers. In the constraint can also be written as some function of $C(y_i, y_i')$, then one simply adds a term $\lambda C(y, y')$ to the function f and uses the Euler-Lagrange equation to find the extrema of.

$$J = \int_{x_1}^{x_2} dx\ f\left\{y(t), y'(t), x\right\} - \lambda C\left\{y(t), y'(t), x\right\}, \tag{200}$$

the one difference being that

$$f\left\{y(t), y'(t), x\right\} \to f\left\{y(t), y'(t), x\right\} - \lambda C\left\{y(t), y'(t), x\right\} \tag{201}$$

Example. Consider a chain of length L and mass per unit length κ that hangs from point $x = 0, y = 0$ to point x_f, y_f. The shape must minimize the potential energy. Find general expressions for the shape in terms of three constants which must be chosen to match $y(0) = 0, y(x_f) = y_f$ and the fixed length. Equivalently, one finds the function $y(t)$ that provides an extrema for the integral,

One must minimize

$$\int d\ell\ \kappa g y - \lambda \int d\ell = \int_0^{x_f} dx\ \sqrt{1+y'^2}\kappa g y - \lambda \int_0^{x_f} dx \sqrt{1+y'^2}.$$

Here λ is the Lagrange multiplier associated with constraining the length of the chain. The constrained length L appears nowhere in the expression. Instead, one solves for form of the answer, then adjusts λ to give the correct length. For the purposes of the Euler-Lagrange minimization one considers the function

$$f(y, y'; x) \quad = \quad \kappa g y \sqrt{1 + y'^2} - \lambda \sqrt{1 + y'^2}. \tag{202}$$

Because λ is an unknown constant and because minimizing a function multiplied by a constant is the same as minimizing the function, we can equivlently minimize the integral using the function

$$\begin{aligned} \tilde{f}(y, y'; x) &= y\sqrt{1 + y'^2} - \tilde{\lambda}\sqrt{1 + y'^2}, \\ \tilde{\lambda} &\equiv \frac{\lambda}{\kappa g}. \end{aligned} \tag{203}$$

The Euler-Lagrange equations then become

$$\frac{d}{dx}\left\{ \frac{y'}{\sqrt{1 + y'^2}} y - \tilde{\lambda} \frac{y'}{\sqrt{1 + y'^2}} \right\} \quad = \quad \sqrt{1 + y'^2}.$$

Here, we will guess at the form of the solution,

$$y' \quad = \quad \sinh[(x - x_0)/a], \quad y = a \cosh[(x - x_0)/a] + y_0.$$

Plugging into the Euler-Lagange equations,

$$\begin{aligned} \frac{d}{dx}\left\{ (a\cosh[(x - x_0)/a] + y_0) \frac{\sinh[(x - x_0)/a]}{\cosh[(x - x_0)/a]} - \tilde{\lambda} \frac{\sinh[(x - x_0)/a]}{\cosh[(x - x_0)/a]} \right\} &= \cosh[(x - x_0)/a], \\ \frac{d}{dx}\left\{ (y_0 - \tilde{\lambda}) \tanh[(x - x_0)/a] \right\} &= 0. \end{aligned}$$

This solution works if $y_0 = \tilde{\lambda}$. So the general form of the solution is

$$y = \tilde{\lambda} + a \cosh[(x - x_0)/a].$$

One must find $\tilde{\lambda}$, x_0 and a to satisfy three conditions, $y(x = 0) = 0$, $y(x = x_f) = y_f$ and that the length is L. For a hanging chain a is positive. A solution with negative a would represent a maximum of the potential energy. A remarkable property of the solution is that once you define the length and the end-point positions y_1 and y_2, the solution does not depend on κ or g. Thus, the shape of the chain would be the same if you took it to the moon. These solutions are known as catenaries.

Lagrangians

Lagrangians represent a powerful method for solving problems that would be nearly impossible by direct application of Newton's third law, $\boldsymbol{F} = m\boldsymbol{a}$. The method works well for problems where a system is well described by a few *generalized coordinates*. A generalized coordinate might be the angle describing the position of a pendulum. This one angle takes the place of using x and y to describe the position of the pendulum, then applying a clumsy constraint.

The Lagrangian equations of motion can be derived from a principle of least action, where the action S is defined as

$$S = \int dt\ L(q, \dot{q}, t), \tag{204}$$

where q is some coordinate that describes the orientation of a system and the Lagrangian L is defined as

$$L = T - U, \tag{205}$$

the difference of the kinetic and potential energies. Minimizing the action through the Euler-Lagrange equations gives the Lagrangian equations of motion,

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} = \frac{\partial L}{\partial q}. \tag{206}$$

We begin with two simple examples, neither of which gains from the Lagrangian approach.

Consider a particle of mass m connected to a spring with stiffness k. Derive the Lagrangian equations of motion.

$$\begin{aligned} L &= \frac{1}{2}m\dot{x}^2 - \frac{1}{2}kx^2, \\ \frac{d}{dt}\frac{\partial L}{\partial \dot{x}} &= \frac{\partial L}{\partial x}, \\ m\ddot{x} &= -kx. \end{aligned}$$

Derive the Lagrangian equations of motion for a pendulum of mass m and length ℓ.

$$\begin{aligned} L &= \frac{m}{2}\ell^2\dot{\theta}^2 - mg\ell(1 - \cos\theta), \\ \frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} &= \frac{\partial L}{\partial \theta}, \\ m\ell^2\ddot{\theta} &= -mg\ell\sin\theta, \\ \ddot{\theta} &= -\frac{g}{\ell}\sin\theta, \\ \ddot{\theta} &\approx -\frac{g}{\ell}\theta. \end{aligned}$$

Proving Lagrange's Equations of Motion from Newton's Laws. Lagrange's equations of motion can only be applied for the following conditions:

- The potential energy is a function of the generalized coordinates q_i, but not of $\dot{q}_i$.
- The relation between the original coordinates $x, y, z \cdots$ and the generalized coordinates does not depend on $\dot{q}_i$, e.g. $x(q,t)$ not $x(q,\dot{q},t)$.
- Any constraints used to reduce the number of degrees of freedom are functions of $\boldsymbol{q}$, but not of $\dot{\boldsymbol{q}}$.
- The motion is not dissipative (no damping or friction).

Going forward with the proof, consider $x_i(q_1, q_2 \cdots, t)$ and look at the l.h.s. of Lagrange's equations of motion.

$$\begin{aligned}
\frac{\partial T}{\partial \dot{q}_j} &= \sum_i \frac{\partial T}{\partial \dot{x}_i}\frac{\partial \dot{x}_i}{\partial \dot{q}_j} + \sum_i \frac{\partial T}{\partial x_i}\frac{\partial x_i}{\partial \dot{q}_j} \\
&= \sum_i m\dot{x}_i \frac{\partial \dot{x}_i}{\partial \dot{q}_j} \\
&= \sum_i m\dot{x}_i \frac{(\delta x_i/\delta t)|_{\text{fixed } q_{j'\neq j}}}{\delta q_j/\delta t} \\
&= \sum_i m\dot{x}_i \frac{\delta x_i|_{\text{fixed } q_{j'\neq j}}}{\delta q_j} \\
&= \sum_i m\dot{x}_i \frac{\partial x_i}{\partial q_j}.
\end{aligned} \tag{207}$$

In the first line we used the fact that T does not depend on x. Continuing with taking the derivative of U,

$$-\frac{\partial U}{\partial \dot{q}_j} = -\sum_i \frac{\partial U}{\partial x_i}\frac{\partial x_i}{\partial \dot{q}_j} = 0. \tag{208}$$

In the first line above we used the fact that U does not depend on $\dot{x}$ then we used the second condition that x does not depend on $\dot{q}$. Adding the two pieces together, then taking the derivative w.r.t. time,

$$\frac{d}{dt}\frac{\partial}{\partial \dot{q}}(T-U) = \sum_i m\ddot{x}_i \frac{\partial x_i}{\partial q_j} + \sum_i m\dot{x}_i \frac{\partial \dot{x}_i}{\partial q_j}.$$

Now, we consider the r.h.s. of Lagrange's equations. Because the kinetic energy depends only on $\dot{x}$ and not x, and because the potential depends on x but not $\dot{x}$,

$$\begin{aligned}\frac{\partial}{\partial q_j}(T-U) &= \sum_i \frac{\partial T}{\partial \dot{x}_i}\frac{\partial \dot{x}_i}{\partial q_j} - \sum_i \frac{\partial U}{\partial x_i}\frac{\partial x_i}{\partial q_j} \\ &= \sum_i m\dot{x}_i \frac{\partial \dot{x}_i}{\partial q_j} - \sum_i \frac{\partial U}{\partial x_i}\frac{\partial x_i}{\partial q_j}\end{aligned} \tag{209}$$

Using the fact that $m\ddot{x}_i = -(\partial/\partial x_i)U$, one can see that the bottom expressions above are identical,

$$\frac{d}{dt}\frac{\partial}{\partial \dot{q}_i}(T-U) = \frac{\partial}{\partial q_i}(T-U). \tag{210}$$

Lagrangian Examples. Two examples are presented here. In the first, there are two generalized coordinates, but the two equations of motion can be reduced to one through conservation laws (angular momentum in this case). In the second, there is a time-dependent constraint.

Consider a cone of half angle α standing on its tip at the origin. The surface of the cone is defined as

$$r = \sqrt{x^2+y^2} = z\tan\alpha.$$

Find the equations of motion for a particle of mass m moving along the surface under the influence of a constant gravitational force, $-mg\hat{z}$. For generalized coordinates use the azimuthal angle ϕ and r.

The kinetic energy is

$$\begin{aligned}T &= \frac{1}{2}mr^2\dot{\theta}^2 + \frac{1}{2}m(\dot{r}^2+\dot{z}^2) \\ &= \frac{1}{2}mr^2\dot{\theta}^2 + \frac{1}{2}m\dot{r}^2\left(1+\cot^2\alpha\right) \\ &= \frac{1}{2}mr^2\dot{\theta}^2 + \frac{1}{2}m\dot{r}^2\csc^2\alpha.\end{aligned}$$

The potential energy is

$$U = mgr\cot\alpha,$$

so Lagrange's equations give

$$\begin{aligned}\frac{d}{dt}\left(mr^2\dot{\theta}\right) &= 0, \\ \frac{d}{dt}\left(m\csc^2\alpha\dot{r}\right) &= mr\dot{\theta}^2 - mg\cot\alpha, \\ \ddot{r} &= r\dot{\theta}^2\sin^2\alpha - g\cos\alpha\sin\alpha\end{aligned}$$

The first equation is a statement of the conservation of angular momentum with $L = mr^2\dot{\theta}$, so the second equation can also be expressed as

$$\ddot{r} = \frac{L^2 \sin^2\alpha}{m^2 r^3} - g\sin\alpha\cos\alpha.$$

A bead slides along a wire bent in the shape of a parabola,

$$z = \frac{1}{2}kr^2, \quad r^2 = x^2 + y^2.$$

Also, the parabolic wire is rotating about the z axis with angular velocity ω. Derive the equations of motion. Are there any stable configurations?

Using the fact that

$$\dot{z} = \dot{r}\frac{\partial z}{\partial r} = kr\dot{r},$$

the kinetic and potential energies are

$$\begin{aligned} T &= \frac{1}{2}m\left(\dot{r}^2 + \dot{z}^2 + r^2\omega^2\right) \\ &= \frac{1}{2}m\left(\dot{r}^2 + (kr\dot{r})^2 + r^2\omega^2\right), \\ U &= mgkr^2/2. \end{aligned}$$

The equations of motion are then

$$\begin{aligned} \frac{d}{dt}\left\{m\dot{r}(1+k^2r^2)\right\} &= -mgkr + mk^2\dot{r}^2 r + m\omega^2 r, \\ \ddot{r} &= \frac{-gkr + \omega^2 r - k^2\dot{r}^2 r}{1+k^2r^2} \end{aligned}$$

For a stable configuration, there needs to be a solution with $\dot{r} = 0$ and $\ddot{r} = 0$. This can only happen at $r = 0$, and then for the acceleration to be inward for small deviations of r one needs to have $gk > \omega^2$. If $\omega^2 > gk$ the bead will move outward indefinitely.

Small Vibrations and Normal Modes

Two examples are provided for solving for normal modes. These are solutions with multiple generalized coordinates, where the motion is that of simple harmonic motion. However, the motion is only simple for a particular set of coordinates q_1 and q_2,

$$\begin{aligned} q_1 &= A\cos(\omega_1 t), \\ q_2 &= B\cos(\omega_2 t), \end{aligned} \tag{211}$$

while it is not necessarily simple in other coordinates. For example if $x = q_1 + q_2$, and $y = q_1 - q_2$, the x and y motions will contain mixtures of multiple frequencies. For many problems, or in the limit of small vibrations about a minimum, there is some coordinate system where the motion is simple. These are normal modes. Characterizing the normal modes involves finding the frequencies, ω_i, and the coordinate system where the motion is simple for each coordinate. This involves finding the direction, or the linear combination of x_i that form the coordinates q_i in which the motion is that of a single oscillator in each coordinate.

For a first example, we consider a system of springs, where we write the Lagrangian, then find the normal modes. For the second example, a double pendulum is considered. In this case, one must first make a small angle expansion before finding the modes. In principle, problems could have the same number of normal modes a degrees of freedom. For example, a system of 7 particles moving in three dimensions has 21 degrees of freedom. However, some of the degrees of freedom do not have oscillatory behavior. For example, for a rigid body in free space, the angles describing the orientation evolve, but do not oscillate. Also, the center-of-mass coordinates of a system of particles isolated from outside particles moves at constant velocity. One can also describe these as normal modes, but acknowledge that their characteristic frequency is zero, as there are no restoring forces.

Consider two springs, whose relaxed lengths are ℓ, connected to three masses as depicted in the figure here. Describe the two normal modes of the motion. We can write the Lagrangian as

$$\mathcal{L} = \frac{m}{2}\dot{x}_1^2 + m\dot{x}_2^2 + \frac{m}{2}\dot{x}_3^2 - \frac{k}{2}(x_2 - x_1 - \ell)^2 - \frac{k}{2}(x_3 - x_2 - \ell)^2.$$

There are three coordinates, thus there are three equations of motion,

$$\begin{aligned} m\ddot{x}_1 &= -k(x_1 - x_2 + \ell) \\ 2m\ddot{x}_2 &= -k(x_2 - x_1 - \ell) - k(x_2 - x_3 + \ell) \\ &= -k(2x_2 - x_1 - x_3) \\ m\ddot{x}_3 &= -k(x_3 - x_2 + \ell). \end{aligned}$$

This is a bit complicated because the center-of-mass motion does not easily separate from the three equations. Instead, choose the following coordinates,

$$\begin{aligned} X &= \frac{x_1 + 2x_2 + x_3}{4}, \\ q_1 &= x_1 - x_2 + \ell, \\ q_3 &= x_3 - x_2 - \ell. \end{aligned}$$

In these coordinates the potential energy only involves two coordinates,

$$U = \frac{k}{2}(q_1^2 + q_3^2).$$

To express the kinetic energy express x_1, x_2 and x_3 in terms of X, q_1 and q_3,

$$\begin{aligned} x_1 &= (3q_1 - q_3 - 4\ell + 4X)/4, \\ x_2 &= (4X - q_1 - q_3)/4, \\ x_3 &= (3q_3 - q_1 + 4\ell + 4X)/4. \end{aligned}$$

The kinetic energy and Lagrangian are them

$$\begin{aligned} T &= \frac{m}{2}\frac{1}{16}(3\dot{q}_1 - \dot{q}_3 + 4\dot{X})^2 + m\frac{1}{16}(4\dot{X} - \dot{q}_1 - \dot{q}_3)^2 + \frac{m}{2}\frac{1}{16}(3\dot{q}_3 - \dot{q}_1 + 4\dot{X})^2 \\ &= \frac{3m}{8}(\dot{q}_1^2 + \dot{q}_3^2) - \frac{m}{4}\dot{q}_1\dot{q}_3 + 2m\dot{X}^2, \\ \mathcal{L} &= \frac{3m}{8}(\dot{q}_1^2 + \dot{q}_3^2) - \frac{m}{4}\dot{q}_1\dot{q}_3 + 2m\dot{X}^2 - \frac{k}{2}q_1^2 - \frac{k}{2}q_3^2. \end{aligned}$$

The three equations of motion are then,

$$\begin{aligned} \frac{3}{4}m\ddot{q}_1 - \frac{1}{4}m\ddot{q}_3 &= -kq_1, \\ \frac{3}{4}m\ddot{q}_3 - \frac{1}{4}m\ddot{q}_1 &= -kq_3, \\ 4M\ddot{X} &= 0. \end{aligned}$$

The last equation simply states that the center-of-mass velocity is fixed. One could obtain the same result by summing the equations of motion for x_1, $2x_2$ and x_3 above. The second two equations are more complicated. To solve them, we assume a form

$$\begin{aligned} q_1 &= Ae^{i\omega t}, \\ q_3 &= Be^{i\omega t}, \end{aligned}$$

Because this is a linear equation, we can multiply the solution by a constant and it will still be a solution. Thus, we can set $B = 1$, then solve for A, effectively solving for A/B. Putting this guess into the equations of motion,

$$\begin{aligned} -\frac{3}{4}\frac{A}{B}\omega^2 + \frac{1}{4}\omega^2 &= -\omega_0^2\frac{A}{B}, \\ -\frac{3}{4}\omega^2 + \frac{1}{4}\frac{A}{B}\omega^2 &= -\omega_0^2. \end{aligned}$$

This is two equations and two unknowns, ω^2 and A/B. Substituting for A/B gives a quadratic equation,

$$\begin{aligned} \omega^4 - 3\omega_0^2\omega^2 + 2\omega_0^4 &= 0, \\ \omega_0^2 &\equiv k/m. \end{aligned}$$

The two solutions are

$$\begin{aligned} (1)\ \omega &= \omega_0, \quad A = -B, \\ (2)\ \omega &= \omega_0\sqrt{2}, \quad A = B. \end{aligned}$$

The first solution corresponds to the two outer masses moving in opposite directions, in sync, with the middle mass fixed. The second solution has both outer masses moving in the same direction, but with the center mass moving opposite. These two solutions are referred to as normal modes, and are characterized by their frequency and by the linear combinations of coordinates that oscillate together. In general, the solution is a linear combination of normal modes, which usually results in a chaotic looking motion. However, once the solution is expressed in terms of the normal modes, each of which oscillates independently in a simple manner, one can better understand the motion. Further, the frequencies of these modes represent the natural resonant frequencies of the system. This is important in the construction of many structures, such as bridges or vehicles.

Consider a double pendulum confined to the $x - y$ plane, where y is vertical. A mass m is connected to the ceiling with a massless string of length ℓ. A second mass m hangs from the first mass with an identical massless string of the same length. Using θ_1 and θ_2 to describe the orientations of the strings relative to the vertical axis, find the Lagrangian and derive the equations of motion, both for arbitrary angles and in the small-angle approximation. Finally, express the equations of motion in the limit of small oscillations.

The kinetic and potential energies are:

$$\begin{aligned} T &= \frac{1}{2}m\ell^2\dot{\theta}_1^2 + \frac{1}{2}m\left\{(\ell\dot{\theta}_1\cos\theta_1 + \ell\dot{\theta}_2\cos\theta_2)^2 + (\ell\dot{\theta}_1\sin\theta_1 + \ell\dot{\theta}_2\sin\theta_2)^2\right\} \\ &= \frac{1}{2}m\ell^2\left\{2\dot{\theta}_1^2 + \dot{\theta}_2^2 + 2\dot{\theta}_1\dot{\theta}_2\cos(\theta_1 - \theta_2)\right\}, \\ U &= mg\ell(1 - \cos\theta_1) + mg\left[\ell(1 - \cos\theta_1) + \ell(1 - \cos\theta_2)\right] \\ &= mg\ell(3 - 2\cos\theta_1 - \cos\theta_2) \end{aligned}$$

Lagrange's equations for θ_1 lead to

$$\begin{aligned} m\ell^2\frac{d}{dt}\left\{2\dot{\theta}_1 + \dot{\theta}_2\cos(\theta_1 - \theta_2)\right\} &= -m\ell^2\dot{\theta}_1\dot{\theta}_2\sin(\theta_1 - \theta_2) - 2mg\ell\sin\theta_1, \\ 2\ddot{\theta}_1 + \ddot{\theta}_2\cos(\theta_1 - \theta_2) + \dot{\theta}_2^2\sin(\theta_1 - \theta_2) &= -2\omega_0^2\sin\theta_1, \\ \omega_0^2 &\equiv g/\ell, \end{aligned}$$

and the equations for θ_2 are

$$\begin{aligned} m\ell^2 \frac{d}{dt}\left\{\dot{\theta}_2 + \dot{\theta}_1 \cos(\theta_1 - \theta_2)\right\} &= m\ell^2 \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_1 - \theta_2) - mg\ell \sin\theta_2, \\ \ddot{\theta}_2 + \ddot{\theta}_1 \cos(\theta_1 - \theta_2) &= -\omega_0^2 \sin\theta_2. \end{aligned}$$

For small oscillations, one can only consider terms linear in θ_1 and θ_2 or their derivatives,

$$\begin{aligned} 2\ddot{\theta}_1 + \ddot{\theta}_2 &= -2\omega_0^2 \theta_1, \\ \ddot{\theta}_1 + \ddot{\theta}_2 &= -\omega_0^2 \theta_2. \end{aligned} \tag{212}$$

To find the solutions, assume they are of the form $\theta_1 = Ae^{i\omega t}, \theta_2 = Be^{i\omega t}$. Solve for ω and A/B, noting that B is arbitrary.

Plug in the desired form and find

$$\begin{aligned} e^{i\omega t}(-2\omega^2 A - \omega^2 B) &= e^{i\omega t}(-2\omega_0^2 A), \\ e^{i\omega t}(-\omega^2 A - \omega^2 B) &= e^{i\omega t}(-\omega_0^2 B). \end{aligned}$$

We can treat B as arbitrary and set it to unity. When we find A, it is the same as A/B for arbitrary B. This gives the equations

$$\begin{aligned} 2\omega^2 A + \omega^2 &= 2\omega_0^2 A, \\ \omega^2 A + \omega^2 &= \omega_0^2. \end{aligned}$$

This is two equations and two unknowns. Solving them leads to a quadratic equation with solutions

$$\begin{aligned} A/B &= \pm\frac{1}{\sqrt{2}}, \\ \omega^2 &= \frac{\omega_0^2}{1 \pm 1/\sqrt{2}}. \end{aligned}$$

Again, these two solutions are the normal modes, and the general solution is a sum of the two solutions, with two arbitrary constants. For the angles θ_1 and θ_2 are:

$$\begin{aligned} \theta_1 &= \frac{A_+}{\sqrt{2}} e^{i\omega_+ t},\ \theta_2 = A_+ e^{i\omega_+ t}, \\ \theta_1 &= \frac{-A_-}{\sqrt{2}} e^{i\omega_- t},\ \theta_2 = A_- e^{i\omega_- t}, \\ \omega_\pm &= \omega_0 \sqrt{\frac{1}{1 \pm 1/\sqrt{2}}}. \end{aligned}$$

One can also express the solution in vector notation, with the vectors having arbitrary amplitudes A_+ and A_-,

$$\begin{aligned}\theta_+ &= \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 1 \end{pmatrix} A_+ e^{i\omega_+ t}, \\ \theta_- &= \begin{pmatrix} \frac{-1}{\sqrt{2}} \\ 1 \end{pmatrix} A_- e^{i\omega_- t}.\end{aligned}$$

Here, the upper/lower components of the vector describe θ_1/θ_2 respectively.

These problems can be treated as linear algebra exercises. Linear algebra is not used in this course, but nonetheless we describe how this works for the curious student. In the limit of small vibrations, the equations of motion can be expressed in the form,

$$M\ddot{q} = -Kq,$$

a form that looks like the spring equation. However, q is an $n-$dimensional vector and M and k are $n \times n$ matrices. In the double pendulum example, the dimensionality is 2 and the q refers to the θ_1 and θ_2, and the matrices for M and K can be read off (add ref).

$$M = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}, \qquad K = \begin{pmatrix} 2\omega_0^2 & 0 \\ 0 & \omega_0^2 \end{pmatrix}.$$

Multiplying both sides of the equation by the inverse matrix M^{-1},

$$\ddot{q} = -\left(M^{-1}K\right) q.$$

Here,

$$\begin{aligned} M^{-1} &= \begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix}, \\ M^{-1}K &= \begin{pmatrix} 2 & -1 \\ -2 & 2 \end{pmatrix} \omega_0^2.\end{aligned}$$

One can find a transformation, basically a rotation, that transforms to a frame where $M^{-1}K$ is diagonal. In this coordinate system the diagonal components of $M^{-1}K$ represent the squared frequencies of the normal modes,

$$M^{-1}K \to -\begin{pmatrix} \omega_+^2 & 0 \\ 0 & \omega_-^2 \end{pmatrix},$$

and are known as "eigen" frequencies. The corresponding unit vectors,

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ and } \begin{pmatrix} 0 \\ 1 \end{pmatrix} \text{ in the new coordinate system,}$$

can be rotated back into the original frame, and become the solutions for the normal modes. These are then called "eigenvectors", which are the same as the normal modes. Finding the eigenfrequencies is performed by realizing that the determinant of a matrix is unchanged by the rotation between coordinate systems. Writing the equations of motion as an eigenvalue problem,

$$[A - \lambda_i \mathbb{1}]\, u_i \;=\; 0, \quad A \equiv M^{-1}K,\ \lambda_i \equiv \omega_i^2. \tag{213}$$

In the coordinate system where $M^{-1}K$ is diagonal, and the forms for u_i are simple this requires that in that system, the diagonal elements of $M^{-1}K$ are the eigenvalues, ω_i^2. For each ω_i^2, the determinant $|A - \lambda_i \mathbb{1}|$ must vanish. This is then true in any coordinate system,

$$\det\left[A - \lambda \mathbb{1}\right] \;=\; 0, \tag{214}$$

which for a 2×2 matrix becomes

$$\begin{vmatrix} A_{11} - \lambda & A_{12} \\ A_{21} & A_{22} - \lambda \end{vmatrix} \;=\; 0, \tag{215}$$

$$A_{11}A_{22} - \lambda A_{11} - \lambda A_{22} + \lambda^2 - A_{21}A_{12} \;=\; 0. \tag{216}$$

One can solve a quadratic equation for λ, which gives two eigenvalues corresponding to ω_+^2 and ω_-^2 found above. Choosing one of the eigenvalues, one can insert one of the eigenvalues λ_i into the eigenvalue problem and solve for u_i, then choose the other eigenvalue and solve for the other corresponding vector.

If this were a 3-dimensional set of equations, the determinant would include terms like λ^3 and would become a cubic equation with three eigenvalues. One would then solve for three eigenvectors. If one has a system with dimensionality $n > 2$, one usually resorts to solving the problem numerically due to the messiness of the algebra. The main programming languages all have packages which readily diagonalize matrices and find eigenvectors and eigenvalues.

Conservation Laws

Energy is conserved only when the Lagrangian has no explicit dependence on time, i.e. $L(q, \dot{q})$, not $L(q, \dot{q}, t)$. To show this, we first define the Hamiltonian,

$$H \;=\; \sum_i \left(\dot{q}_i \frac{\partial L}{\partial \dot{q}_i} \right) - L. \tag{217}$$

After showing that H is conserved, i.e. $(d/dt)H = 0$, we then show that H can be identified with the total energy, $H = T + V$.

One can see that H is conserved by applying first using the chain rule for $(d/dt)H$, then applying Lagrange's equations,

$$\begin{aligned} \frac{d}{dt}H &= \sum_i \left\{ \ddot{q}_i \frac{\partial L}{\partial \dot{q}_i} + \dot{q}_i \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial \dot{q}_i}\ddot{q}_i - \frac{\partial L}{\partial q_i}\dot{q}_i \right\} \\ &= \sum_i \left\{ \ddot{q}_i \frac{\partial L}{\partial \dot{q}_i} + \dot{q}_i \frac{\partial L}{\partial q_i} - \frac{\partial L}{\partial \dot{q}_i}\ddot{q}_i - \frac{\partial L}{\partial q_i}\dot{q}_i \right\} \\ &= 0. \end{aligned} \tag{218}$$

These steps assumed that L had no explicit time dependence, i.e. L is a function of q and $\dot{q}$, but not of t.

Next, we show that L can be identified with the energy. Because V does not depend on $\dot{q}$,

$$H = \sum_i \frac{\partial T}{\partial \dot{q}_i}\dot{q}_i - T + V. \tag{219}$$

If the kinetic energy has a purely quadratic form in terms of $\dot{q}$,

$$T = \sum_{ij} A_{ij}(q)\dot{q}_i\dot{q}_j, \tag{220}$$

the Hamiltonian becomes

$$\begin{aligned} H &= \sum_{ij} 2A_{ij}(q)\dot{q}_i\dot{q}_j - \sum_{ij} A_{ij}(q)\dot{q}_i\dot{q}_j + V \\ &= T + V. \end{aligned} \tag{221}$$

The proof that H equals the energy hinged on the fact that the kinetic energy was quadratic in $\dot{q}$. This can be attributed to time-reversal symmetry. Because the Cartesian coordinates x_i do not depend on $\dot{q}_i$ or on time, $\dot{x}_i = (\partial x_i/\partial q_j)\dot{q}_j$. Thus, the kinetic energy, $T = m\dot{x}_i^2/2$, should be proportional to two powers of $\dot{q}$, which validates the assumption above.

Here, energy conservation is predicated on the Lagrangian not having an explicit time dependence. Without an explicit time dependence the equations of motion are unchanged if one translates a fixed amount in time because the physics does not depend on when the clock starts. In contrast, the absolute time becomes relevant if there is an explicit time dependence. In fact, conservation laws can usually be associated with symmetries. In this case the translation symmetry in time leads to energy conservation.

For another example of how symmetry leads to conservation laws, consider a Lagrangian for a particle of mass m moving in a two-dimensional plane where

the generalized coordinates are the radius r and the angle θ. The kinetic energy would be

$$T = \frac{1}{2} m \left\{ \dot{r}^2 + r^2 \dot{\theta}^2 \right\}, \tag{222}$$

and if the potential energy $V(r)$ depends only on the radius r and not on the angle, Lagrange's equations become

$$\begin{aligned} \frac{d}{dt}(m\dot{r}) &= -\frac{\partial V}{\partial r} + m\dot{\theta}^2 r, \\ \frac{d}{dt}(mr^2\dot{\theta}) &= 0. \end{aligned} \tag{223}$$

The second equation implies that $mr^2\dot{\theta}$ is a constant. Indeed, it is the angular momentum which is conserved for a radial force. Here, the conservation of angular momentum is associated with the independence of the physics to changes in θ, or in other words, rotational invariance. Once one knows the fact that $L = mr^2\dot{\theta}$ is conserved, it can be inserted into the equations of motion for $\dot{r}$,

$$m\ddot{r} = -\frac{\partial V}{\partial r} + \frac{L^2}{mr^3}. \tag{224}$$

This is related to Emmy Noether's theorem

Simply stated, if the Lagrangian L is independent of q_i, one can see that the quantity $\partial L/\partial \dot{q}_i$ is conserved,

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} = 0. \tag{225}$$

Another easy example is in Cartesian coordinates where the potential depends only on x and y but not on z. In that case, there is a translational symmetry. From the last equation, this translates into conservation of the momentum in the z direction.

Consider a pair of particles of mass m_1 and m_2 where the potential is of the form

$$U(\boldsymbol{r}_1, \boldsymbol{r}_2) = V_a(|m_1\boldsymbol{r}_1 + m_2\boldsymbol{r}_2|/(m_1 + m_2)) + V_b(|\boldsymbol{r}_1 - \boldsymbol{r}_2|).$$

Using symmetry arguments alone, are there any conserved components of the momentum? or the angular momentum??

There is no translational invariance, hence there are no conserved components of the momentum. However, there is rotational invariance about any axis that goes through the origin. Hence, there is angular momentum conservation in all three directions. Symmetry arguments are great ways to recognize the existence of conserved quantities, but actually expressing them in terms of coordinates can be tricky. For instance, you may need to write the Lagrangian in terms of angles.