PHY321: Time-dependent Forces and Fourier Series, begin two-body problems

Morten Hjorth-Jensen[1,2]

[1]Department of Physics and Astronomy and Facility for Rare Ion Beams (FRIB), Michigan State University, USA
[2]Department of Physics, University of Oslo, Norway

Mar 8, 2021

Aims and Overarching Motivation

Driven oscillations and resonances with numerical examples and Fourier Series

Monday. **Reading suggestion**: Taylor sections 5.6-5.8.

Wednesday. Summary oscillations and resonances with examples and Fourier Series

Reading suggestion: Taylor chapter 5.

Friday. Begin two-body problems **Reading suggestion**: Taylor section 8.2.

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 $F_{\mathrm{ext}}(x,t)$. 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.

```python
# 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)
```

```python
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 dimensionless time
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

$$\frac{dy}{dt} = f(t, y), \tag{1}$$

and

$$y(t) = \int f(t, y)dt, \tag{2}$$

and

$$y_{i+1} = y_i + \int_{t_i}^{t_{i+1}} f(t, y)dt. \tag{3}$$

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

$$\int_{t_i}^{t_{i+1}} f(t, y)dt \approx hf(t_{i+1/2}, y_{i+1/2}) + O(h^3). \tag{4}$$

This means in turn that we have

$$y_{i+1} = y_i + hf(t_{i+1/2}, y_{i+1/2}) + O(h^3). \tag{5}$$

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

$$y_{(i+1/2)} = y_i + \frac{h}{2}\frac{dy}{dt} = y(t_i) + \frac{h}{2}f(t_i, y_i). \tag{6}$$

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

$$k_1 = hf(t_i, y_i), \tag{7}$$

$$k_2 = hf(t_{i+1/2}, y_i + k_1/2), \tag{8}$$

with the final value

$$y_{i+i} \approx y_i + k_2 + O(h^3). \tag{9}$$

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

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

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

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)
```

```python
# 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

$$x_p(t) = \sum_n x_{pn}(t). \tag{10}$$

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

$$F(t + \tau) = F(t). \tag{11}$$

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 $\tau = 0.2$.

```python
import numpy as np
import math
```

```python
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 week 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,

$$F(t) = \frac{f_0}{2} + \sum_{n>0} f_n \cos(2n\pi t/\tau) + g_n \sin(2n\pi t/\tau). \tag{12}$$

We can write down the answer for $x_{pn}(t)$, by substituting f_n/m or g_n/m for F_0/m. By writing each factor $2n\pi t/\tau$ as $n\omega t$, with $\omega \equiv 2\pi/\tau$,

$$F(t) = \frac{f_0}{2} + \sum_{n>0} f_n \cos(n\omega t) + g_n \sin(n\omega t). \tag{13}$$

The solutions for $x(t)$ then come from replacing ω with $n\omega$ for each term in the particular solution,

$$\begin{aligned}
x_p(t) &= \frac{f_0}{2k} + \sum_{n>0} \alpha_n \cos(n\omega t - \delta_n) + \beta_n \sin(n\omega t - \delta_n), \qquad (14)\\
\alpha_n &= \frac{f_n/m}{\sqrt{((n\omega)^2 - \omega_0^2) + 4\beta^2 n^2 \omega^2}},\\
\beta_n &= \frac{g_n/m}{\sqrt{((n\omega)^2 - \omega_0^2) + 4\beta^2 n^2 \omega^2}},\\
\delta_n &= \tan^{-1}\left(\frac{2\beta n\omega}{\omega_0^2 - n^2\omega^2}\right).
\end{aligned}$$

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

$$f_n = \frac{2}{\tau}\int_{-\tau/2}^{\tau/2} dt\ F(t)\cos(2n\pi t/\tau), \tag{15}$$
$$g_n = \frac{2}{\tau}\int_{-\tau/2}^{\tau/2} dt\ F(t)\sin(2n\pi t/\tau).$$

To check the consistency of these expressions and to verify Eq. (15), one can insert the expansion of $F(t)$ in Eq. (13) into the expression for the coefficients in Eq. (15) and see whether

$$f_n \ =?\ \frac{2}{\tau}\int_{-\tau/2}^{\tau/2} dt\ \left\{\frac{f_0}{2} + \sum_{m>0} f_m\cos(m\omega t) + g_m\sin(m\omega t)\right\}\cos(n\omega t). \tag{16}$$

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

$$\int_{-\tau/2}^{\tau/2} dt\ \cos^2(m\omega t) = \frac{\tau}{2}, \tag{17}$$

and

$$f_n \ =?\ \frac{2}{\tau}\int_{-\tau/2}^{\tau/2} dt\ f_n/2 \tag{18}$$
$$= f_n\ \checkmark.$$

The same method can be used to check for the consistency of g_n.

Consider the driving force:

$$F(t) = At/\tau, \quad -\tau/2 < t < \tau/2, \quad F(t+\tau) = F(t). \tag{19}$$

Find the Fourier coefficients f_n and g_n for all n using Eq. (15).

Only the odd coefficients enter by symmetry, i.e. $f_n = 0$. One can find g_n integrating by parts,

$$g_n = \frac{2}{\tau}\int_{-\tau/2}^{\tau/2} dt\ \sin(n\omega t)\frac{At}{\tau} \tag{20}$$
$$u = t,\ dv = \sin(n\omega t)dt,\ v = -\cos(n\omega t)/(n\omega),$$
$$g_n = \frac{-2A}{n\omega\tau^2}\int_{-\tau/2}^{\tau/2} dt\ \cos(n\omega t) + 2A\left.\frac{-t\cos(n\omega t)}{n\omega\tau^2}\right|_{-\tau/2}^{\tau/2}.$$

The first term is zero because $\cos(n\omega t)$ will be equally positive and negative over the interval. Using the fact that $\omega\tau = 2\pi$,

$$
\begin{aligned}
g_n &= -\frac{2A}{2n\pi}\cos(n\omega\tau/2) \\
&= -\frac{A}{n\pi}\cos(n\pi) \\
&= \frac{A}{n\pi}(-1)^{n+1}.
\end{aligned} \tag{21}
$$

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.

```python
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

Material to be added.

Response to Transient Force

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 the equations with initial conditions $v_0 = I/m$ and $x_0 = 0$. This gives

$$x(t) = \frac{I}{m\omega'} e^{-\beta t} \sin \omega' t, \quad t > 0. \tag{22}$$

Here, $\omega' = \sqrt{\omega_0^2 - \beta^2}$. For an impulse I_i that occurs at time t_i the trajectory would be

$$x(t) = \frac{I_i}{m\omega'} e^{-\beta(t-t_i)} \sin[\omega'(t-t_i)]\Theta(t-t_i), \tag{23}$$

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,

$$x(t) = \sum_i \frac{I_i}{m\omega'} e^{-\beta(t-t_i)} \sin[\omega'(t-t_i)]\Theta(t-t_i) \tag{24}$$

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,

$$\begin{aligned} x(t) &= \int_{-\infty}^{\infty} dt' \; F(t') \frac{e^{-\beta(t-t')} \sin[\omega'(t-t')]}{m\omega'} \Theta(t-t') \\ &= \int_{-\infty}^{\infty} dt' \; F(t') G(t-t'), \\ G(\Delta t) &= \frac{e^{-\beta \Delta t} \sin[\omega' \Delta t]}{m\omega'} \Theta(\Delta t) \end{aligned} \tag{25}$$

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. (25) one can use angle addition formulas to factor out the part with the t' dependence in the integrand,

$$\begin{aligned} x(t) &= \frac{1}{m\omega'} e^{-\beta t} \left[I_c(t) \sin(\omega' t) - I_s(t) \cos(\omega' t) \right], \\ I_c(t) &\equiv \int_{-\infty}^{t} dt' \; F(t') e^{\beta t'} \cos(\omega' t'), \\ I_s(t) &\equiv \int_{-\infty}^{t} dt' \; F(t') e^{\beta t'} \sin(\omega' t'). \end{aligned} \tag{26}$$

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,

$$F(t) \quad = \quad F_0 \exp\left\{\frac{-t^2}{2\tau^2}\right\}.$$

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. (26). 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. (26) 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

$$ml\frac{d^2\theta}{dt^2} + mgsin(\theta) = 0, \tag{27}$$

with an angular velocity and acceleration given by

$$v = l\frac{d\theta}{dt}, \tag{28}$$

and

$$a = l\frac{d^2\theta}{dt^2}. \tag{29}$$

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

$$ml\frac{d^2\theta}{dt^2} + \nu\frac{d\theta}{dt} + mgsin(\theta) = 0, \tag{30}$$

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

$$ml\frac{d^2\theta}{dt^2} + \nu\frac{d\theta}{dt} + mgsin(\theta) = Asin(\omega t), \tag{31}$$

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

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

More on the Pendulum

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.

Two-body Problems

The gravitational potential energy and forces involving two masses a and b are

$$\begin{aligned} V_{ab} &= -\frac{Gm_am_b}{|\boldsymbol{r}_a - \boldsymbol{r}_b|}, \\ F_{ba} &= -\frac{Gm_am_b}{|\boldsymbol{r}_a - \boldsymbol{r}_b|^2}\hat{r}_{ab}, \\ \hat{r}_{ab} &= \frac{\boldsymbol{r}_b - \boldsymbol{r}_a}{|\boldsymbol{r}_a - \boldsymbol{r}_b|}. \end{aligned} \tag{32}$$

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

$$V = \sum_{a<b} U_{ab} = \frac{1}{2}\sum_{a\neq b} V_{ab}. \tag{33}$$

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 $\boldsymbol{r}$,

$$\begin{aligned}
\boldsymbol{R}_{\rm cm} &\equiv \frac{m_1\boldsymbol{r}_1 + m_2\boldsymbol{r}_2}{m_1+m_2}, \qquad (34)\\
\boldsymbol{r} &\equiv \boldsymbol{r}_1 - \boldsymbol{r}_2.
\end{aligned}$$

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

$$\begin{aligned}
\ddot{\boldsymbol{R}}_{\rm cm} &= \frac{1}{m_1+m_2}\left\{m_1\ddot{\boldsymbol{r}}_1 + m_2\ddot{\boldsymbol{r}}_2\right\} \qquad (35)\\
&= \frac{1}{m_1+m_2}\left\{\boldsymbol{F}_{12} + \boldsymbol{F}_{21}\right\} = 0.\\
\ddot{\boldsymbol{r}} &= \ddot{\boldsymbol{r}}_1 - \ddot{\boldsymbol{r}}_2 = \left(\frac{\boldsymbol{F}_{12}}{m_1} - \frac{\boldsymbol{F}_{21}}{m_2}\right) \qquad (36)\\
&= \left(\frac{1}{m_1} + \frac{1}{m_2}\right)\boldsymbol{F}_{12}.
\end{aligned}$$

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.

$$\begin{aligned}
\mu\ddot{\boldsymbol{r}} &= \boldsymbol{F}_{12}, \qquad (37)\\
\frac{1}{\mu} &= \frac{1}{m_1} + \frac{1}{m_2}, \qquad \mu = \frac{m_1 m_2}{m_1+m_2}. \qquad (38)
\end{aligned}$$

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

$$\begin{aligned}
\mu\ddot{r} &= -\frac{Gm_1m_2}{r^2}\hat{r} \qquad (39)\\
&= -\frac{GM\mu}{r^2}\hat{r}, \quad M \equiv m_1 + m_2.
\end{aligned}$$

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

$$\begin{aligned}
\boldsymbol{P} &\equiv \boldsymbol{p}_1 + \boldsymbol{p}_2 = M\dot{\boldsymbol{R}}_{\rm cm}, \qquad (40)\\
\boldsymbol{q} &\equiv \mu\dot{\boldsymbol{r}}.
\end{aligned}$$

With these definitions, a little algebra shows that the kinetic energy becomes

$$\begin{aligned} T &= \frac{1}{2}m_1|\boldsymbol{v}_1|^2 + \frac{1}{2}m_2|\boldsymbol{v}_2|^2 \\ &= \frac{1}{2}M|\dot{\boldsymbol{R}}_{\rm cm}|^2 + \frac{1}{2}\mu|\dot{\boldsymbol{r}}|^2 \\ &= \frac{P^2}{2M} + \frac{q^2}{2\mu}. \end{aligned} \tag{41}$$

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

$$\begin{aligned} \frac{d}{dt}r^2 &= \frac{d}{dt}(x^2+y^2) = 2x\dot{x} + 2y\dot{y} = 2r\dot{r}, \\ \dot{r} &= \frac{x}{r}\dot{x} + \frac{y}{r}\dot{y}, \\ \ddot{r} &= \frac{x}{r}\ddot{x} + \frac{y}{r}\ddot{y} + \frac{\dot{x}^2+\dot{y}^2}{r} - \frac{\dot{r}^2}{r}. \end{aligned} \tag{42}$$

Recognizing that the numerator of the third term is the velocity squared, and that it can be written in polar coordinates,

$$v^2 = \dot{x}^2 + \dot{y}^2 = \dot{r}^2 + r^2\dot{\theta}^2, \tag{43}$$

one can write $\ddot{r}$ as

$$\begin{aligned}
\ddot{r} &= \frac{F_x \cos\theta + F_y \sin\theta}{m} + \frac{\dot{r}^2 + r^2\dot{\theta}^2}{r} - \frac{\dot{r}^2}{r} \\
&= \frac{F}{m} + \frac{r^2\dot{\theta}^2}{r} \\
m\ddot{r} &= F + \frac{L^2}{mr^3}.
\end{aligned} \tag{44}$$

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, $\dot{\theta} = L/mr^2$.

$$\begin{aligned}
\dot{r} &= \frac{dr}{d\theta}\dot{\theta} = \frac{dr}{d\theta}\frac{L}{mr^2}, \\
\ddot{r} &= \frac{d^2r}{d\theta^2}\dot{\theta}^2 + \frac{dr}{d\theta}\left(\frac{d}{dr}\frac{L}{mr^2}\right)\dot{r} \\
&= \frac{d^2r}{d\theta^2}\left(\frac{L}{mr^2}\right)^2 - 2\frac{dr}{d\theta}\frac{L}{mr^3}\dot{r} \\
&= \frac{d^2r}{d\theta^2}\left(\frac{L}{mr^2}\right)^2 - \frac{2}{r}\left(\frac{dr}{d\theta}\right)^2\left(\frac{L}{mr^2}\right)^2
\end{aligned} \tag{45}$$

Equating the two expressions for $\ddot{r}$ in Eq.s (44) and (45) eliminates all the derivatives w.r.t. time, and provides a differential equation with only derivatives w.r.t. θ,

$$\frac{d^2r}{d\theta^2}\left(\frac{L}{mr^2}\right)^2 - \frac{2}{r}\left(\frac{dr}{d\theta}\right)^2\left(\frac{L}{mr^2}\right)^2 = \frac{F}{m} + \frac{L^2}{m^2r^3}, \tag{46}$$

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 $u(\theta)$.

$$
\begin{aligned}
r &= 1/u, \\
\frac{dr}{d\theta} &= -\frac{1}{u^2}\frac{du}{d\theta}, \\
\frac{d^2r}{d\theta^2} &= \frac{2}{u^3}\left(\frac{du}{d\theta}\right)^2 - \frac{1}{u^2}\frac{d^2u}{d\theta^2}.
\end{aligned} \tag{47}
$$

Plugging these expressions into Eq. (46) gives an expression in terms of u, $du/d\theta$, and $d^2u/d\theta^2$. After some tedious algebra,

$$
\frac{d^2u}{d\theta^2} = -u - \frac{Fm}{L^2u^2}. \tag{48}
$$

For the attractive inverse square law force, $F = -\alpha u^2$,

$$
\frac{d^2u}{d\theta^2} = -u + \frac{m\alpha}{L^2}. \tag{49}
$$

The solution has two arbitrary constants, A and θ_0,

$$
\begin{aligned}
u &= \frac{m\alpha}{L^2} + A\cos(\theta - \theta_0), \\
r &= \frac{1}{(m\alpha/L^2) + A\cos(\theta - \theta_0)}.
\end{aligned} \tag{50}
$$

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. (50) 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,

$$
\begin{aligned}
2D &= r + \sqrt{(r\cos\theta - 2a)^2 + r^2\sin^2\theta}, \\
4D^2 + r^2 - 4Dr &= r^2 + 4a^2 - 4ar\cos\theta, \\
r &= \frac{D^2 - a^2}{D + a\cos\theta} = \frac{1}{D/(D^2 - a^2) - a\cos\theta/(D^2 - a^2)}.
\end{aligned} \tag{51}
$$

By inspection, this is the same form as Eq. (50) 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

$$
\begin{aligned}
E &= U(r) + \frac{1}{2}mv_\theta^2 + \frac{1}{2}m\dot{r}^2 \\
&= U(r) + \frac{1}{2}mr^2\dot{\theta}^2 + \frac{1}{2}m\dot{r}^2 \\
&= U(r) + \frac{L^2}{2mr^2} + \frac{1}{2}m\dot{r}^2.
\end{aligned} \tag{52}
$$

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,

$$
U_{\rm eff}(r) = U(r) + \frac{L^2}{2mr^2}. \tag{53}
$$

Note that if one treats the effective potential like a real potential, one would expect to be able to generate an effective force,

$$
\begin{aligned}
F_{\rm eff} &= -\frac{d}{dr}U(r) - \frac{d}{dr}\frac{L^2}{2mr^2} \\
&= F(r) + \frac{L^2}{mr^3} = F(r) + m\frac{v_\perp^2}{r},
\end{aligned} \tag{54}
$$

which is indeed matches the form for $m\ddot{r}$ in Eq. (44), 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
```

```python
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()
```