PHY321: Harmonic Oscillations, Damping, Resonances and time-dependent Forces

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

Feb 26, 2022

Aims and Overarching Motivation

Monday, February 28. Damped oscillations. Analytical and numerical solutions **Reading suggestion**: Taylor sections 5.4-5.5.

Wednesday, March 2. Driven oscillations and resonances with examples. Discussion of first midterm.

Reading suggestion: Taylor sections 5.5-5.6.

Friday, March 4. Work on first midterm. The spring break!

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

$$m\ddot{x} + b\dot{x} + kx = 0. \tag{1}$$

Harmonic Oscillator, Damping

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

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

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

$$x = Ae^{rt} \tag{3}$$

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. (2),

$$r^2 + 2\beta r + \omega_0^2 = 0. \tag{4}$$

Harmonic Oscillator, Solutions of Damped Motion

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

$$r = -\beta \pm \sqrt{\beta^2 - \omega_0^2}. \tag{5}$$

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,

$$x = A_1 e^{r_1 t} + A_2 e^{r_2 t}, \tag{6}$$

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 $\exp(rt)$ will have some oscillatory behavior. If the roots are real, there will only be exponentially decaying solutions. There are three cases:

Underdamped: $\beta < \omega_0$

$$\begin{aligned} x &= A_1 e^{-\beta t} e^{i\omega' t} + A_2 e^{-\beta t} e^{-i\omega' t}, \quad \omega' \equiv \sqrt{\omega_0^2 - \beta^2} \\ &= (A_1 + A_2) e^{-\beta t} \cos\omega' t + i(A_1 - A_2) e^{-\beta t} \sin\omega' t. \end{aligned} \tag{7}$$

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:

$$\begin{aligned} x &= B_1 e^{-\beta t}\cos\omega' t + B_2 e^{-\beta t}\sin\omega' t, \\ \omega' &\equiv \sqrt{\omega_0^2 - \beta^2}. \end{aligned} \tag{8}$$

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

$$x = Ae^{-\beta t} + Bte^{-\beta t}. \tag{9}$$

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: $\beta > \omega_0$

$$x = A_1 \exp -(\beta + \sqrt{\beta^2 - \omega_0^2})t + A_2 \exp -(\beta - \sqrt{\beta^2 - \omega_0^2})t \tag{10}$$

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

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

Harmonic Oscillator, Solutions

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 above, e.g. Eq. (8) 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.

Harmonic Oscillator, Particular Solution

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

$$x_p(t) = D\cos(\omega t - \delta), \tag{11}$$

and rewrite the differential equation as

$$D\left\{-\omega^2\cos(\omega t-\delta) - 2\beta\omega\sin(\omega t-\delta) + \omega_0^2\cos(\omega t-\delta)\right\} = \frac{F_0}{m}\cos(\omega t). \tag{12}$$

One can now use angle addition formulas to get

$$\begin{aligned} D\left\{(-\omega^2\cos\delta + 2\beta\omega\sin\delta + \omega_0^2\cos\delta)\cos(\omega t)\right. & \\ \left.+(-\omega^2\sin\delta - 2\beta\omega\cos\delta + \omega_0^2\sin\delta)\sin(\omega t)\right\} &= \frac{F_0}{m}\cos(\omega t). \end{aligned} \tag{13}$$

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

$$\begin{aligned} D\left\{-\omega^2\cos\delta + 2\beta\omega\sin\delta + \omega_0^2\cos\delta\right\} &= \frac{F_0}{m} \\ -\omega^2\sin\delta - 2\beta\omega\cos\delta + \omega_0^2\sin\delta &= 0. \end{aligned} \tag{14}$$

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

$$\tan\delta = \frac{2\beta\omega}{\omega_0^2 - \omega^2}. \tag{15}$$

Solving with Driven Oscillations

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

$$\begin{aligned} \sin\delta &= \frac{2\beta\omega}{\sqrt{(\omega_0^2-\omega^2)^2+4\omega^2\beta^2}}, \\ \cos\delta &= \frac{(\omega_0^2-\omega^2)}{\sqrt{(\omega_0^2-\omega^2)^2+4\omega^2\beta^2}} \end{aligned} \tag{16}$$

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

$$D = \frac{F_0/m}{\sqrt{(\omega_0^2-\omega^2)^2+4\omega^2\beta^2}}. \tag{17}$$

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 (11) and (17) but express the driving forces as

$$F(t) = F_0 e^{i\omega t}, \tag{18}$$

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

$$\ddot{x} + 2\beta\dot{x} + \omega_0^2 x = \frac{F_0}{m}e^{i\omega t}, \tag{19}$$

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. (19) 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

$$x_p = De^{i\omega t}, \tag{20}$$

where D is a complex constant.
From Eq. (19) one inserts the form for x_p above to get

$$\begin{aligned} D\left\{-\omega^2 + 2i\beta\omega + \omega_0^2\right\} e^{i\omega t} &= (F_0/m)e^{i\omega t}, \\ D &= \frac{F_0/m}{(\omega_0^2 - \omega^2) + 2i\beta\omega}. \end{aligned} \tag{21}$$

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

$$|D| = \frac{F_0/m}{\sqrt{(\omega_0^2 - \omega^2)^2 + 4\beta^2\omega^2}}, \qquad \tan\delta = \frac{2\beta\omega}{\omega_0^2 - \omega^2}. \tag{22}$$

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

$$\begin{aligned} x_p(t) &= \Re \frac{(F_0/m)e^{i\omega t - i\delta}}{\sqrt{(\omega_0^2 - \omega^2)^2 + 4\beta^2\omega^2}} \\ &= \frac{(F_0/m)\cos(\omega t - \delta)}{\sqrt{(\omega_0^2 - \omega^2)^2 + 4\beta^2\omega^2}}. \end{aligned} \tag{23}$$

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

$$\begin{aligned} x_p(t) &= \Im \frac{(F_0/m)e^{i\omega t - i\delta}}{\sqrt{(\omega_0^2 - \omega^2)^2 + 4\beta^2\omega^2}} \\ &= \frac{(F_0/m)\sin(\omega t - \delta)}{\sqrt{(\omega_0^2 - \omega^2)^2 + 4\beta^2\omega^2}}. \end{aligned} \tag{24}$$

Damped and Driven Oscillator

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, $\omega' = \sqrt{\omega_0^2 - \beta^2}$, $\delta = \tan^{-1}(2\beta\omega/(\omega_0^2 - \omega^2)$. 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

$$A = \eta\sin\delta, \quad B = \frac{-\omega\eta\cos\delta + \beta\eta\sin\delta}{\omega'}. \tag{25}$$

Resonance Widths; the Q factor

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

$$\begin{aligned} x_p(t) &= \frac{F_0/m}{\sqrt{(\omega_0^2-\omega^2)^2 + 4\omega^2\beta^2}}\cos(\omega_t - \delta), \\ \delta &= \tan^{-1}\left(\frac{2\beta\omega}{\omega_0^2-\omega^2}\right). \end{aligned} \tag{26}$$

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

$$\frac{d}{d\omega}\left[(\omega_0^2-\omega^2)^2 + 4\omega^2\beta^2\right] = 0, \tag{27}$$

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

$$x_{\max} \approx \frac{F_0}{2m\beta\omega_0}. \tag{28}$$

$$\frac{4\omega^2\beta^2}{(\omega_0^2-\omega^2)^2 + 4\omega^2\beta^2} = \frac{1}{2}. \tag{29}$$

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,

$$Q \equiv \frac{\omega_0}{2\beta}. \tag{30}$$

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

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

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.

```
# 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 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

$$\frac{dy}{dt} = f(t,y), \tag{31}$$

and

$$y(t) = \int f(t,y)dt, \tag{32}$$

and

$$y_{i+1} = y_i + \int_{t_i}^{t_{i+1}} f(t,y)dt. \tag{33}$$

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

$$\int_{t_i}^{t_{i+1}} f(t,y)dt \approx hf(t_{i+1/2}, y_{i+1/2}) + O(h^3). \tag{34}$$

This means in turn that we have

$$y_{i+1} = y_i + hf(t_{i+1/2}, y_{i+1/2}) + O(h^3). \tag{35}$$

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

$$y_{(i+1/2)} = y_i + \frac{h}{2}\frac{dy}{dt} = y(t_i) + \frac{h}{2}f(t_i, y_i). \tag{36}$$

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

$$k_1 = hf(t_i, y_i), \tag{37}$$

$$k_2 = hf(t_{i+1/2}, y_i + k_1/2), \tag{38}$$

with the final value

$$y_{i+i} \approx y_i + k_2 + O(h^3). \tag{39}$$

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*DeltaT*DeltaT
        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 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 (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

$$x_p(t) = \sum_n x_{pn}(t). \tag{40}$$

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

$$F(t+\tau) = F(t). \tag{41}$$

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 above 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,

$$F(t) = \frac{f_0}{2} + \sum_{n>0} f_n \cos(2n\pi t/\tau) + g_n \sin(2n\pi t/\tau). \tag{42}$$

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 (23) or (24) respectively. By writing each factor $2n\pi t/\tau$ as $n\omega t$, with $\omega \equiv 2\pi/\tau$,

$$F(t) = \frac{f_0}{2} + \sum_{n>0} f_n \cos(n\omega t) + g_n \sin(n\omega t). \tag{43}$$

The solutions for $x(t)$ then come from replacing ω with $n\omega$ for each term in the particular solution in Equations (11) and (17),

$$\begin{aligned} x_p(t) &= \frac{f_0}{2k} + \sum_{n>0} \alpha_n \cos(n\omega t - \delta_n) + \beta_n \sin(n\omega t - \delta_n), \qquad (44) \\ \alpha_n &= \frac{f_n/m}{\sqrt{((n\omega)^2 - \omega_0^2) + 4\beta^2 n^2 \omega^2}}, \\ \beta_n &= \frac{g_n/m}{\sqrt{((n\omega)^2 - \omega_0^2) + 4\beta^2 n^2 \omega^2}}, \\ \delta_n &= \tan^{-1}\left(\frac{2\beta n\omega}{\omega_0^2 - n^2\omega^2}\right). \end{aligned}$$

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

$$\begin{aligned} f_n &= \frac{2}{\tau}\int_{-\tau/2}^{\tau/2} dt\ F(t)\cos(2n\pi t/\tau), \qquad (45) \\ g_n &= \frac{2}{\tau}\int_{-\tau/2}^{\tau/2} dt\ F(t)\sin(2n\pi t/\tau). \end{aligned}$$

To check the consistency of these expressions and to verify Eq. (45), one can insert the expansion of $F(t)$ in Eq. (43) into the expression for the coefficients in Eq. (45) and see whether

$$f_n \quad =? \quad \frac{2}{\tau}\int_{-\tau/2}^{\tau/2} dt\ \left\{\frac{f_0}{2} + \sum_{m>0} f_m \cos(m\omega t) + g_m \sin(m\omega t)\right\} \cos(n\omega t). \qquad (46)$$

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

$$\int_{-\tau/2}^{\tau/2} dt\ \cos^2(m\omega t) = \frac{\tau}{2}, \qquad (47)$$

and

$$\begin{aligned} f_n &=? \quad \frac{2}{\tau}\int_{-\tau/2}^{\tau/2} dt\ f_n/2 \\ &= \quad f_n\ \checkmark. \end{aligned} \tag{48}$$

The same method can be used to check for the consistency of g_n.
Consider the driving force:

$$F(t) = At/\tau, \quad -\tau/2 < t < \tau/2, \quad F(t+\tau) = F(t). \tag{49}$$

Find the Fourier coefficients f_n and g_n for all n using Eq. (45).
Only the odd coefficients enter by symmetry, i.e. $f_n = 0$. One can find g_n integrating by parts,

$$\begin{aligned} g_n &= \frac{2}{\tau}\int_{-\tau/2}^{\tau/2} dt\ \sin(n\omega t)\frac{At}{\tau} \\ u &= t,\ dv = \sin(n\omega t)dt,\ v = -\cos(n\omega t)/(n\omega), \\ g_n &= \frac{-2A}{n\omega\tau^2}\int_{-\tau/2}^{\tau/2} dt\ \cos(n\omega t) + 2A\frac{-t\cos(n\omega t)}{n\omega\tau^2}\Big|_{-\tau/2}^{\tau/2}. \end{aligned} \tag{50}$$

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

$$\begin{aligned} g_n &= -\frac{2A}{2n\pi}\cos(n\omega\tau/2) \\ &= -\frac{A}{n\pi}\cos(n\pi) \\ &= \frac{A}{n\pi}(-1)^{n+1}. \end{aligned} \tag{51}$$

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()
```