You should submit one notebook per exercise, containing the code and results for all the tasks in that exercise.

Plotting your results

You will want to present many of your results in the form of graphs made using pyplot, which will appear below the code cell when the cell is run. **Remember to add captions for the axes and a title.** Here is a simple Python code fragment which plots a sine function with axes.

```
plt.plot(np.sin(np.linspace(-10,10,100)))
plt.xlabel("Distance along axis (m)")
plt.ylabel("Intensity (arbitrary units)")
plt.title("Diffraction pattern of a double slit");
```

Note the use of a semicolon on the last line to suppress any printed output — not essential but makes the notebook look cleaner. You can also use the "object-oriented" pyplot plotting style used in places in the lectures, which can be helpful when you are plotting multiple plots in the same figure.

It is usually a good idea to do the long-running part of any calculation in one code cell and then do the plotting in a subsequent code cell. This way, you can change the plotting code and re-run it to improve the appearance of the plot and add labels etc without re-running all of the calculations.

If you are doing very-long-running calculations you may want to save the results to a file to analyse/-plot later. You can use `numpy.savez()` function to do this.

Uploading your notebook to the TiS

You should use one notebook per exercise. When you are finished the exercise and are ready to submit it for assessment, you can use the "Download .ipynb" item under the "File" menu in Colab. You can then upload the file from your computer to the TiS.

Exercise 1: The Driven Pendulum

Goal

To explore the physics of a non-linear oscillator (a damped, driven pendulum) by accurate integration of its equation of motion.

Physics

The pendulum comprises a bob of mass m on a light rod of length l and swings in a uniform gravitational field g. If there is a resistive force equal to αv where the bob speed is v, and a driving sinusoidal couple G at frequency Ω_D, we can write

$$m\,l^2\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = -m\,g\,l\,\sin(\theta) - \alpha\,l\,\frac{\mathrm{d}\theta}{\mathrm{d}t} + G\sin(\Omega_D t) \tag{1}$$

Rearranging:

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = -\frac{g}{l}\,\sin(\theta) - q\,\frac{\mathrm{d}\theta}{\mathrm{d}t} + F\sin(\Omega_D t) \tag{2}$$

where we have defined $q \equiv \alpha/(m\,l)$ and $F \equiv G/(m\,l^2)$.

For the purposes of this exercise, take $l = g$, so the natural period for small oscillations should be 2π seconds. Also let the driving angular frequency be $\Omega = 2/3\mathrm{rad\,s^{-1}}$. This leaves q and F, and the initial conditions, to be varied. For all of these problems, we will start the pendulum from rest i.e. $\dot\theta = 0$ at $t = 0$, but we will vary the initial displacement θ_0. The parameter space to explore then is in the three values q, F and θ_0.

Tasks

Core Task 1 First re-write this second-order differential equation Equation 2 as a pair of linked first-order equations in the variables $y_0 = \theta$ and $y_1 = \omega = \dot\theta$. Now write a program that will integrate this pair of equations using a suitable algorithm, from a given starting point $\theta = \theta_0$ and $\omega = \omega_0$ at $t = 0$. I recommend using `scipy` rather than implementing your own Runge-Kutta or other technique.

Test the code by setting $q = F = 0$ and starting from $(\theta_0, \omega_0) = (0.01, 0.0)$, and plotting the solution for 10, 100, 1000…natural periods of oscillation. Overlay on your plot the expected theoretical result for small-angle oscillations — make sure they agree!

Test how well your integrator conserves energy: run the code for, say, 10,000 oscillations and plot the evolution of energy with time.

Now find how the amplitude of undriven, undamped oscillations depends affects the period. Plot a graph of the period T versus θ_0 for $0 < \theta_0 < \pi$.

> **Include in your notebook:** Source code, appropriately-scaled plots showing how well energy is conserved in at least one case and a plot of period versus amplitude, conclusions text containing a couple of sentences summarising what you managed to achieve, and the value of the period for $\theta_0 = \pi/2$

Core task 2 Now turn on some damping, say $q = 1, 5, 10$, plot some results, and check that the results make sense. Now turn on the driving force, leaving $q = 0.5$ from now on, and investigate with suitable plots the behaviour for $F = 0.5, 1.2, 1.44, 1.465$. What happens to the period of oscillation? *Note that the period of oscillation is best observed in the angular velocity rather than angular position, to avoid problems with wrap-around at $\pm\pi$.*

> **Include in your notebook:** Source code, a sentence or two in the conclusions sub-section explaining what you see in the solutions, and illustrative plots of the displacement θ and the angular velocity $\frac{\mathrm{d}\theta}{\mathrm{d}t}$ versus time.

Supplementary task 1 Investigate the sensitivity to initial conditions: compare two oscillations, one with $F = 1.2, \theta_0 = 0.2$ and one with $F = 1.2, \theta_0 = 0.20001$. Integrate for a 'long time' to see if the solutions diverge or stay the same.

> **Include in your notebook:** Source code, a sentence or two in the conclusions sub-section explaining what you see in the solutions, and illustrative plots of the behaviour.

Supplementary Task 2 Try plotting angle versus angular speed for various solutions, to compare the type of behaviour in various regimes: you can investigate chaotic behaviour using this simple code — have a look in the books or a web site for examples. There is a nice demo for example at http://www.myphysicslab.com. There's lots more physics to be explored here — experiment if you have time!

> **Include in your notebook:** Source code, a sentence or two in the conclusions sub-section explaining what you see in the solutions, and illustrative plots of the behaviour.

Hints

1. Hopefully rewriting the equation as 2 ODEs is straightforward: if not, ask!
2. I recommend that you use a "canned" ODE integrator, for example the `scipy.integrate.solve_ivp()` function, rather than writing your own.
3. You will need to write a function that evaluates the derivatives of the ODEs at a given time given the current values of the variables. You can look at the example code solving the spinning ring problem discussed in lectures. You can perhaps adapt some of that code if you get stuck.
4. Think about the time window you choose for your plots: in some cases, plotting less than the full set of data you have computed may make for clearer visualisation of what is going on.
5. Many of the ODE integrator functions in `scipy.integrate` and elsewhere are adaptive-stepping algorithms, and have options which are set using "keyword arguments" to control this

behaviour. Firstly, you can set a desired accuracy for the solution: setting a higher accuracy may make the program run more slowly because it will typically take shorter steps. Secondly, the functions usually have the capability to return an interpolated value of the solution at user-specified times, and not just at the time steps used to integrate the ODE. You can choose a time sampling to give appropriately smooth plots. It pays to experiment with these values to see if they have any effect on your results, especially when investigating "chaotic" behaviour.

6. To find the period versus amplitude relationship, a simple (and just about acceptable) way is to measure the period off a suitable plot, and do this for several values of θ_0. However, it is much better to alter your code to estimate the period directly. You can then loop over θ_0 values and plot the period versus amplitude relation. Two obvious approaches spring to mind. First, you can find when θ first goes negative. This is when the time is approximately $T/4$ where T is the period. How accurate would this result be? You could also find several zero crossings by considering when y changes sign or becomes exactly zero after a step is taken; by counting many such zero crossings and recording the time between them you can get a more accurate value for T.
7. Note that you need to think about what happens when the pendulum goes "over the top" and comes down the other side — you need to think about the 2π ambiguities involved, and what this means in terms of the "period" of an oscillation.

Exercise 2: Fraunhofer and Fresnel Diffraction

Goal

Write a program to calculate the near and far-field diffraction patterns of an arbitrary one-dimensional complex aperture using the Fast Fourier Transform technique. Test this program by using simple test apertures (a slit) for which the theoretical pattern is known. Investigate more complicated apertures for which analytical results are difficult to compute.

Physics

Plane monochromatic waves, of wavelength λ, arrive at normal incidence on an aperture, which has a complex transmittance $A(x)$. The wave is diffracted, and the pattern is observed on a screen a distance D from the aperture and parallel to it. We want to calculate the pattern when the screen is in the far-field of the aperture (Fraunhofer diffraction) and also in the near-field (Fresnel).