# Physics Part II: Computational Physics Exercises

This project contains my solutions for the 3 exercises in the computational physics course, these were all attempted in `python3` with a wide usage of the `numpy` and `-rscipy` -r packages, a standard adopted by the scientific computing community.

## Exercise 1

The first exercise was on numerical integration, exploring the use of a standard Gaussian method and a Monte-Carlo method. The implementation of the Gaussian method was straightforward using `scipy.integrate.quad`, and was used to solve the *Fresnel* integrals. However, this technique scales poorly for higher dimensions which is why a Monte-Carlo method was used for the case of an 8-dimensional integral.

The `python` code for this exercise is found in `/ex1` and is separated into `numInt.py` and `-rfresnel.py` -r. Relevant figures are printed as PDF documents using `-rmatplotlib.pyplot` -r.

## Exercise 2

The second exercise involved solving ODEs numerically for the specific case of a normal rigid pendulum. While analytic solutions are easily found in the small angle approximation, like most problems, a closed form solution cannot be found for the general case. Hence numerical methods are employed to tackle the problem at hand, and for this exercise, the *LSODA* method built into `scipy` was tested alongside an explicitly coded *Runge-Kuta* method.

With these tools, the behaviour of the pendulum was tested with various damping and driving forces. One particularly interesting case was the chaotic motion performed by the pendulum under large driving forces which made the system very sensitive to initial conditions. To illustrate this, an animation of 2 pendulums with a 0.05% difference in initial displacement was made using `matplotlib.animate` and outputted as an MP4 file.

The `python` code for this exercise is found in `/ex2` and is separated into `ODE.py` and `-renergyPlot.py` -r. Relevant figures are printed as PDF documents using `-rmatplotlib.pyplot` -r.

## Exercise 3

The final exercise is an investigation into the use of a Fast Fourier Transform which is used to find the diffraction pattern of a various apertures. The Fourier

transform serves the same purpose as the aperture integral if the frequency is scaled appropriately.

The problem was separated into two tasks, firstly, a function was written to produce a `numpy` array with complex numbers that represent the desired aperture. This is then passed into a standard FFT function in the `numpy` package which is given the aforementioned aperture function as input. The result is then scaled appropriately and plotted.

The `python` code for this exercise is found in `/ex3` and is contained in `FFT.py`. Relevant figures are printed as PDF documents using `-rmatplotlib.pyplot` -r.