Computational Physics

Projects

David Buscher

Version 2.8 March 2022

Contents

Introduction

You should choose one of the following projects. The aim is to develop and test *your own* `Python` (or C++) program, and use this to investigate the problem. You may (and indeed are encouraged) to use the functions of `numpy`, `scipy` and `pandas` `Python` packages where needed (or the GNU Science Library `GSL`, and the `FFTW` library for C++). You may also make use of other open-source packages, but if you do so you must make this clear in your report, using citations where appropriate.

Keep a record of the analysis you do before programming, details of the library routines you use, and the tests that you make, as well as the final commented program and results. The written up form of these, possibly together with an oral session with a head of class, will form the basis of the assessment.

The problems are to differing degrees **open-ended**: what is given below is just a starting point for each project. You should investigate the problem you have chosen as thoroughly as you can in the time allocated for this task. Bear in mind that the credit for this work is one unit of further work.

Write-up

Your report should not exceed 3000 words (excluding the abstract, captions, references and appendices — note that your code listing should be in an appendix). **Include a word count at the end of your summary.**

The following is intended as a guide to the structure of your report, and what should be included in it. You should follow the basic structure suggested in the booklet "Keeping Laboratory Notes and Writing Formal Reports" and the general advice given there on style, except as noted below.

The aim is to produce a report on your solution to one of the set problems. Background material will have been covered in lectures and in the exercises.

What you should concentrate on in your write up, therefore, are those aspects of computational physics relevant to the problem you have chosen. The physics of some of the problems may go beyond your current knowledge, and understanding the advanced ideas behind some of the problems is not part of this exercise. However, basic physical reasoning and the physics that you have already met in Part IB and this term's Part II courses will be assumed.

You should include the following:

1. A brief introduction to the problem.
2. The analysis of the computational aspects of the problem that were necessary for you to undertake its solution. This might include a brief description of relevant computational physics, choice of approach, choice of suitable routines, scaling of the problem, etc.
3. The implementation: that is, your program. The program listing of the main computational code will be in an appendix, so in the main text your description of the program should be relatively short, concentrating on your overall approach and the way you implemented the necessary algorithms to solve the problem.
4. Performance: you should include a brief section on the performance of your program, and discuss any steps taken to ensure the program ran efficiently. You might also like to include indicative run times for your code, i.e. how much CPU time was used to generate a given result.
5. Results and analysis. This should include discussion of the tests you performed to demonstrate that your program is working correctly. Your final results should also be presented together with a discussion of errors and any other computational problems. You should aim to present several clear plots which illustrate the results.
6. An appendix containing a listing of the main numerical code, together with a brief indication of what other code there is (it is helpful to give the names of the files etc). Listings of plotting programs/functions and similar ancillary code can be omitted from the appendix, but should be included in ZIP file containing code which forms part of your submission. Note that if you are coding in Google Colab, then you can export a `.py` file which is likely to be more convenient

than a `.ipynb` file for formatting as a listing in the report appendix.

Structure

A possible structure might be the following (although details are problem specific):

Abstract; Introduction; Analysis; Implementation; Results and Discussion; Conclusions.

Creating your Report: Word or LaTeX

You can create your report using any word processing package of your choice. Many of you will use MS Word or similar applications, which is fine. However, an excellent alternative is LaTeX, a popular program for creating documents that has good support for scientific work (mathematics), and for including program listings; it is widely used in scientific research. The lecture handouts and the manuals (including this one) were created using LaTeX.

If you would like to try this system, there is information on the web, including on the course web page.

Submitting your report

You should submit your report and ancilliary information on the TiS (https://www-teach.phy.cam.ac.uk/students/courses/computing-project/117). You should upload:

1. A copy of your report as a PDF file (you should not include the Word or LaTeX source). Please give the file a sensible name, something like `project-CRSID.pdf` would be a good choice.
2. A ZIP file (see https://en.wikipedia.org/wiki/ZIP_(file_format)) containing the source code of your program(s), in one or multiple files. You are asked to include a copy of the numerical code in an appendix to your report, but a copy of the plotting and ancilliary code, as well as the numerical code, should all be in your code ZIP file. If you are doing your programming on Google Colab, it would be helpful to include both the `.ipynb` format notebook and the corresponding `.py` files.
3. (Optionally) a ZIP file containing any animation files you refer to in your report. If you are uploading animations, be sure to clearly indicate in your report which specific animation is relevant to which particular scientific result — do not expect the assessors to look at any animation unless it is specifically referenced as evidence for some particular result (this is just like the convention for figures - they are only deemed relevant if explicitly referred to in the text). Animations can be in any standard video file format, e.g. mp4. Animated GIF files are acceptable for short animations, but they get large very quickly and there is an upload limit of 100MB.;

Marking

The credit for this project is one unit of further work. Approximately equal credit will be given to each of the following areas:

1. Analysis of the computational physics aspects of the problem;
2. Implementation of the algorithm, including code style, readability, quality;
3. Results, analysis of errors (if applicable), tests and discussion of the relevant computational physics;
4. Overall presentation of the report, structure, etc.

A number of candidates may be selected for Viva voce (i.e. oral) examination early in the Lent Term after submission, as a matter of routine, and therefore a summons to a Viva should not be taken to indicate that there is anything amiss. You will be asked some straightforward questions on your project work, and may be asked to elaborate on the extent of discussions you have had with other students. You may also be asked to give a live demonstration of your program. So long as you can demonstrate that your write-up is indeed your own, your answers will not alter your project marks.

Programming language

If you so wish, you may use other languages to help with scripting, graphing etc, but *the core numerical routines must be written in `Python` or `C++`*.

Collaboration

The remarks about collaboration and cheating which are contained in the course handbook, with reference to experimental work, also apply here. The project differs from the compulsory exercises in this respect. In the exercises, working together to promote your learning is encouraged. In this assessed project, you should work essentially on your own.

Discussion of the problems among students is encouraged, as this will often help your understanding. However, your program and your report should be written by you, and only results produced by your program should be presented in your report. It would be unacceptable for two students to submit near identical programs.

Any attempt to pass off the work of others as being produced by yourself will be regarded as cheating and may be referred to the examiners. When the report is handed in, you will be asked to make a declaration about the degree of collaboration that went in to the work that is being submitted.

Project A: Tidal Tails and Saturn's Rings

The aim of the problem is to develop a simple N-body simulation of interacting galaxies, and optionally develop it to investigate Saturn's rings. The problem will be simplified by considering two heavy masses and a number of light test masses. You should approach the problem in two stages.

Stage 1: creation of a single model galaxy

The model galaxy consists of a central heavy mass and five rings of test particles. Consider carefully the scaling of the problem. One possibility is to use a set of units such that the central mass has a mass of 1 unit and G=1. Arrange the test particles in rings about the central mass at radii 2, 3, 4, 5 and 6 units. An appropriate density of particles is 12, 18, 24, 30 and 36 on the five rings respectively. Since they are to be treated as test particles the only force felt by each particle is from the central mass alone. Determine appropriate velocities for the particles and set up an N-body simulation so that the particles correctly remain on circular orbits. Test the accuracy and stability of your chosen ODE integrator — testing stability over long periods is especially important if you intend to simulate Saturn's rings.

Stage 2: interaction

Stage 2 involves the introduction of a second "heavy" mass. This represents a perturbing galaxy with similar mass interacting gravitationally with the first. Determine the initial conditions of this second

galaxy so that it will pass, approximately, a distance of 9 to 12 units from the first central mass at closest approach; the orbit should be chosen to be parabolic in the first instance (i.e. $E_{\text{total}} = 0$).

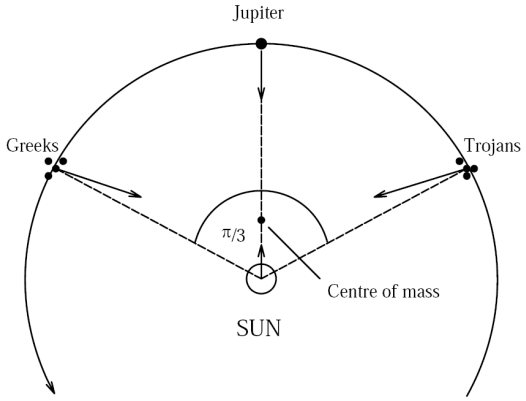
Test the motions of the two masses without the test masses. Re-introduce the test masses which now see a force due to the two "heavy" masses. Follow the evolution of the system by plotting the positions of all of the masses as a function of time (experiment with choosing the most appropriate timescale on which to plot snapshots of the system).

Use this model to explore the effect of reversing the direction of approach from clockwise to anticlockwise on the formation of tidal tails.

At this stage you can explore one of two problems:

1. Explore tidal tails in more depth, including the effects of varying the mass of the disrupting galaxy and the distance of closest approach. Try to derive as many quantitative results as possible, for example counting the fraction of test masses in different initial orbits which are disrupted in one sense or another. Increasing the number of test masses (by several orders of magnitude) and adding orbits at intermediate radii will help to increase the accuracy of these results.
2. Use the framework developed so far to study the dynamics of Saturn's rings, with the test masses representing light "boulders" in the rings. You are likely to need to have a large number of boulders in a large number of circular orbits to adequately represent the rings. Include the gravitational effects of Saturn itself (mass 5.7×10^{26} kg) when setting up the initial orbits and then add one or more moons: you can start by looking at the gravitational effect of the moon Mimas, assuming a circular orbit of radius 185,000 km and a mass of 3.7×10^{19} kg. The effects may take many thousands of orbits to build up: it is acceptable to speed things up by increasing the mass of Mimas by perhaps a factor of 100 or more.
   Many of the features seen in Saturn's rings are thought to arise in part from the effects of collisions between the boulders. These effects are time-consuming to model accurately in this framework, so it is ok to assume collisionless test particles and see which effects are still observable. If you like, you could then try to approximate the effects of collisions by adding some kind of "viscosity" to the system, for example by exchanging a chosen (small) fraction of their momentum between test particles which pass within some radius of one another. Try to develop quantitative measures of any qualitative phenomena you observe.

Project B: The Trojan asteroids



Accompanying the planet Jupiter in its orbit around the Sun are two groups of asteroids, named individually after the main protagonists in the Trojan wars. The two groups are constrained at points of stable equilibrium (the Lagrange points) preceding and trailing the planet at an angular distance of $\pi/3$ radians. The combination of the gravitational attraction of the Sun and Jupiter gives a resultant force on a Trojan asteroid towards the centre of mass of the two bodies such that the centripetal acceleration causes the asteroid to orbit with the same period as the planet.

Write a program to solve the equations of motion for this system numerically. Remember to test your program: it is important to test the accuracy and stability of your chosen ODE integrator over long periods in this problem.

Start the asteroids at positions "slightly" displaced from the Lagrange points to investigate the stability of the equilibrium positions. Plot the positions of the asteroids through a few hundred orbits, and show that they oscillate about the Lagrange points, but do not escape from the stable position.

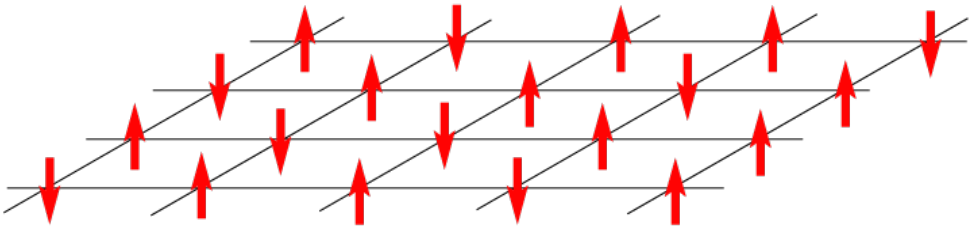
Derive quantitative measures of how far the orbits wander from the Lagrange points. Explore how the wander of the orbits varies with respect to the initial conditions in both displacement and velocity — you could try selecting random points in the initial (position, velocity) phase space and see how they evolve. Use the results from a large number of initial conditions to develop quantitative relationships where possible. Explore the effects of evolution over many orbits — do initially stable orbits become unstable?

Trojan asteroids are also present in the orbit of Mars. Run your program again for a range of other planetary masses, within one or two orders of magnitude of that of Jupiter, and derive a relationship linking the range of wander and the planetary mass.

Suitably scaled units for this problem are solar system units, where the mass of the Sun ($M_\odot$) is taken as unit mass, unit distance is the astronomical unit (AU) and unit time is one year. For such a system

the gravitational constant G is $4\pi^2$. The mass of Jupiter can be taken to be $0.001\ M_\odot$ and the average distance of Jupiter from the Sun is $5.2\ \mathrm{AU}$. The masses of the asteroids can be regarded as negligible in this system.

Project C: The Ising Model of a Ferromagnet



In this project you will use the Metropolis algorithm to investigate ferromagnetism using a simplified model of the spin-spin interactions known as the Ising model.

In the Ising model we treat the magnet as a set of spins on fixed lattice sites in two dimensions. Suppose we have an N by N lattices, for a total of N^2 sites. The energy of the system is written

$$E = -J \sum_{<ij>} s_i \, s_j - \mu \, H \sum_{i=1}^{N^2} s_i$$

where the sum $< ij >$ is over nearest neighbour pairs of atoms only. In 2-D, there are thus 4 nearest neighbour interactions. J is the exchange energy, μ the magnetic moment, and H the field. It is easiest to use *periodic boundary conditions*, where the lattice "wraps round" from the left edge to the right edge and from the top edge to the bottom edge; topologically the system is then the same as the surface of a toroid. With these boundary conditions every spin has 4 neighbours.

The basic algorithm is as follows: starting from some initial set of spin orientations, perhaps a random or uniform set, the model is evolved in time by a Monte-Carlo technique: pick a spin, and calculate the energy ΔE required to flip it. If $\Delta E < 0$, flip the spin; if $\exp(-\Delta E/(k_B T)) > p$, where p is a random number drawn from a uniform distribution in the range $[0, 1]$, then flip the spin; else do nothing.

We now repeat this step for many lattice sites. One can either step systematically through the lattice doing N^2 possible flips, or pick N^2 random points. You can think of these N^2 operations together as a single *time step*.

Now repeat many steps to evolve the spin system, sampling relevant quantities and averaging them. The physical goal is to measure the thermodynamic and magnetic properties of the system as a function of time and as a function of the temperature and magnetic field.

You will find it easiest to measure T in units of J/k_B (where J is the exchange energy). Use your model to investigate, as quantitatively and accurately as possible, the physical properties of the system.

The following are the **minimum** set of properties to investigate:

1. Set $H = 0$. Try to quantify how long (i.e. number of "steps") it takes to get from a given starting state, either completely random or all spins in the same direction, to a state which appears to be in equilibrium. Repeat this for a range of different temperatures T.
2. Quantify how the total magnetisation M fluctuates with "time" when the system is in equilibrium. The *autocovariance* of the magnetisation $M(t)$ for a time lag τ is given by

$$A(\tau) = \langle M'(t)M'(t+\tau)\rangle\,,$$

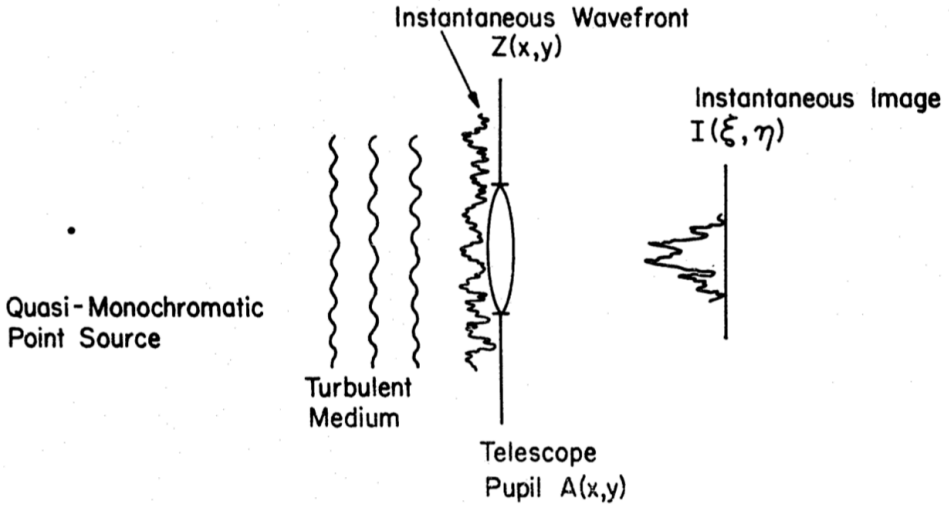
   where $\langle\rangle$ denotes averaging over a "long" time and M' is the deviation from the mean $M' = M - \langle M\rangle$. The *autocorrelation* is given by $a(\tau) = A(\tau)/A(0)$.
   Determine the "decorrelation time": the time lag τ_e over which the autcorrelation falls to $1/e$ at different temperatures, especially near the critical temperature T_c.
   Plot graphs of the decorrelation time vs the temperature for two different lattice sizes N, and consider how this impacts the averaging time needed to get accurate values in the other investigations.
3. Determine how the mean magnetisation $\langle M\rangle$ varies with temperature (it is acceptable to determine the properties of $\langle|M|\rangle$ instead.)
4. Measure the heat capacity C of the system as a function of temperature. The fluctuation-dissipation theorem that says that $C = \sigma_E^2/(k_B T^2)$ where σ_E is the standard deviation of fluctuations in the system energy. Derive a critical temperature T_C for different values of N. Compare your value of T_C with Onsager's analytical result for an infinite lattice: $T_c = 2/\ln(1+\sqrt{2})$.
5. Investigate "finite-size scaling", which suggests that the critical temperature $T_C(N)$ for a lattice size of size N varies as

$$T_C(N) = T_C(\infty) + aN^{-1/\nu}$$

   where a and ν are constants. Derive estimates of $T_C(\infty)$ from your data and compare them to Onsager's result, remembering that you need to quantify your uncertainties.
6. Investigate what happens when $H \neq 0$: in particular, examine hysteresis effects when H is cycled at different temperatures.

Project D: Optical speckles

If you look through a telescope at a point-like object (such as a star) at high mangification, you will see the image "shimmer" due the optical effects of turbulence in the Earth's atmosphere, a phenomenon known as "seeing". For large telescopes, the image breaks up into "speckles" and these appear to randomly "boil" as a function of time. This project aims to simulate the images you see through the telescope and use these simulated images to derive quantitative information about the image distortions due to seeing.



The diagram above represents the model for this scenario. The telescope is represented as an aperture followed by a perfect lens which forms an image in the telescope focal plane. When illuminated by a distant point source of light, the amplitude distribution in the focal plane is the Fraunhofer diffraction pattern, equal to the Fourier transform of the aperture function $A(x, y)$. For a perfect optical telescope with no "seeing", $A(x, y) = 1$, for $r < D/2$ and $A = 0$ otherwise, where D is the telescope diameter and $r^2 = x^2 + y^2$. The image (the intensity pattern, which is proportional to the modulus squared of the image-plane amplitude pattern) is then the well-known Airy pattern.

In reality, refractive index variations the Earth's atmosphere distort the phase of the light arriving at the telescope leading to a "corrugated" wavefront. This can be modelled as a complex aperture function $A(x, y)$ which then leads to distortions in the image and it is these distortions we will simulate.

The investigations to be performed are:

1. No atmosphere (a space-based telescope). Assuming the telescope is well-constructed (not always the case!), then the phase is zero across the aperture. Write a program to calculate the image of a point source seen through a circular telescope aperture of diameter D using a 2-dimensional FFT. A convenient dimensionless set of image plane coordinates is the angular co-

ordinate as projected onto the sky, in units of λ/D where λ is the wavelength — for $\lambda = 500$ nm and $D = 1$ m then this unit is 0.5 μrad or about 0.1 arc-second.
Quantitatively assess the accuracy of your image as a function of the density of sampling of the aperture and the amount of "zero-padding" outside the aperture. Don't forget that the Discrete Fourier Transform (which the FFT implements) is periodic in both the spatial and the frequency domains, and this can lead to image artefacts.

2. Now create a 2-D "phase screen" which models the random wavefront perturbations introduced by the atmosphere. Rather than using purely uncorrelated random perturbations from point to point, introduce correlated perturbations with a correlation length l_c.
You can generate correlated perturbations by convolving (blurring/smoothing) an uncorrelated 2-D array of unit-variance normally-distributed random numbers with a 2-D spatial Gaussian with width $\sigma = l_c$. Note that the RMS amplitude of the resulting perturbations σ_ϕ is proportional to both the height of the Gaussian and its width. If you scale the height of the Gaussian inversely with its width l_c, then this will keep the RMS wavefront perturbation σ_ϕ constant — check that this works.
You can use an FFT to do the convolution, but note that the convolution done with an FFT leads to periodic boundary conditions (i.e. the convolution wraps around one edge of the array and back to the other), so make your phase screen considerably larger than your aperture so that you effectively mask out the boundary effects.
Create a phase screen with $l_c = 0.2D$ and with RMS amplitude of $\sigma_\phi = 1.5$ radian and use this to generate the complex aperture function of a telescope looking through this set of phase perturbations. From this compute the image seen in the focal plane. Generate several independent random phase screens and plot the images produced from these.
3. Generate "long-exposure" images by averaging together a large number of these distorted images. Investigate how the long-exposure image properties vary as a function of the RMS distortion amplitude σ_ϕ in the range 0.1 radian to 10 radians while keeping l_c fixed at $0.2D$.
One particular measure to investigate is the "Strehl ratio": the peak intensity in the image compared to the peak intensity from the same aperture with no phase distortions. Compare the Strehl ratios you measure with the "extended Maréchal approximation" which states that the long-exposure Strehl ratio S falls off with σ_ϕ as

$$S \approx e^{-\sigma_\phi^2}.$$

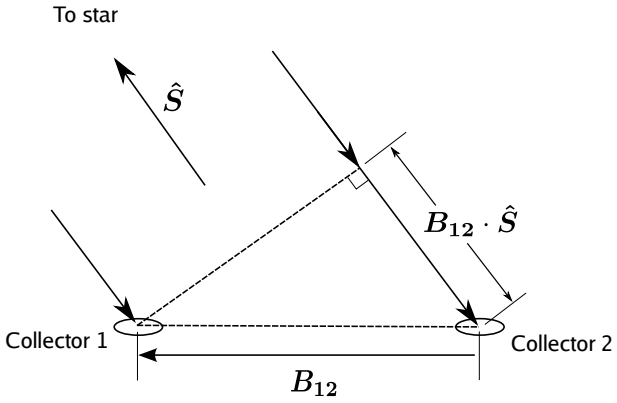
Plot 1-d cross-sections through the long-exposure images to give image profiles for different σ_ϕ and plot how the full-width at half-maximum (FWHM) changes with σ_ϕ.
4. Now set $\sigma_\phi = 4$ radians and explore the effect of varying the spatial correlation length of the perturbations l_c in the range from $l_c = 0.05D$ to $l_c = 2D$.
Investigate the effect of changing l_c on the "speckle transfer function", which is the 2-

dimensional spatial power spectrum of the instantaneous (i.e. not long-exposure, but rather individual "snapshot") image intensity, averaged over a large number of images and normalised to unity at the origin.

It is conventional to plot the 1-dimensional radial profile of the 2-D speckle transfer function on a log-log scale or on a log scale in the y-axis and linear in the spatial-frequency axis. You can compare your results qualitatively with the experimental results in Aimé et al. (1979, doi: 10.1080/713820046). Note that these authors use the term *Modulation Transfer Function* (MTF), which is the modulus of the Fourier transform of the image, normalised so that the MTF at zero spatial frequency is unity, equivalent to the square-root of the speckle transfer function.

Remember to develop error estimates, especially when working with random quantities — this is easiest to do by looking at the scatter between different results from different runs using the same set of parameters. You can represent errors in line plots by using shaded error bands.

Project E: Surveying using stars



In this project you will do "data science" on a large dataset of optical path delay observations from the 6-telescope CHARA interferometer in California (see http://www.chara.gsu.edu for a general description). You will use this data to determine the 3-dimensional locations of the telescopes and try to infer something about their movement over long timescales (California is a seismically-active zone!).

The interferometer observes the interference between light arriving at different telescopes, separated by distances of up to 330 metres, in order to make images of stars at very high angular resolution. To generate interference fringes, the optical path delays from the star to the point of interference must be matched to a few microns. The optical path difference between two telescopes is given by $\hat{S} \cdot (\vec{x}_1 - \vec{x}_2) + d_1 - d_2$ where $\hat{S}$ is the unit vector pointing towards the star, $\vec{x}_1$ and $\vec{x}_2$ are the vector locations of the telescopes with respect to some origin and d_1 and d_2 are internal delays inside the interferometer associated with the optical paths from telescopes 1 and 2 respectively to the point of interference. Note that the vector between two telescopes is known as the "baseline" $\vec{B}_{12}$.

The "internal" delays consist of (a) a continuously-variable optical path length equaliser (OPLE) whose path delay is measured in real time by a laser interferometer to sub-micron accuracy (b) a set of discrete "Pipes of Pan" (POP) delays which can be switched in to access different parts of the sky (c) an quasi-fixed component dependent on optics in the beam path from each telescope to the beam combination point (known as the "constant term"). To the above can be added random errors due to atmospheric optical path delays above each telescope, which vary on timescales of less than a second but have a maximum amplitude of order 10–100 microns, and mechanical drifts in the POP and the "constant term", which can be at the level of hundreds of microns during a given night, but up to millimetres over the longer term.

The aim of this project is to determine the positions of the telescopes as precisely as possible from a sequence of delay measurements, made when the total delays have been equalised to such an extent that interference fringes have been observed.

The data for this project are available on Google Drive in the folder https://drive.google.com/drive/

folders/1Qgd51o0bbjOYv3ECVfdropuQRsMy7ReC?usp=sharing (links to the Google Drive folder can also be be found on the course Moodle site). You can access this data from Google Colab by following any of the options for accessing files given in https://colab.research.google.com/notebooks/io.ipynb. The delay data are in CSV (comma-separated-variable) files, contain tables of delay measurements and ancilliary data.

Each row in these tables represents the delay measured on a given star with a given pair of telescopes at a given time of night. The columns are:

utc : The date and time in UTC (Universal Coordinated Time, the successor to Greenwich Mean Time) when the observation was made.

star : Identifier for the star being observed. More detailed information about any given star can be found in the catalogue `starlist.zip` in the Google Drive folder.

elevation, azimuth : The elevation and azimuth of the star, in degrees, at the instant the delay was measured. Azimuth and elevation are angular coordinates defined relative to the local north and local horizontal plane. See https://en.wikipedia.org/wiki/Horizontal_coordinate_system for more information.

tel_1, tel_2 : Identifiers for the two telescopes making up the baseline being measured.

pop_1, pop_2 : Identifiers for the respective POP setups being used. Note that the P5B1 setup on the W1 telescope is unrelated to the P5B1 setup on the W2 telescope or any other telescope.

cart_1, cart_2 : delays measured in metres on the two OPLEs (one per telescope) at the same instant as fringes were observed.

You can start out by assuming that the delay introduced by any given POP setting is unknown but stable and that the "constant term" is stable on any given night but can vary by unknown amounts from night to night.

The tasks are as follows:

1. Analyse the data from the `2019_04_07.csv` file to yield estimates for the locations of all the telescopes on that night. I recommend using the Moore-Penrose pseudo-inverse (based on Singular Value Decomposition) to solve for the positions (as well as all the other model parameters).
   You should construct a "design matrix" for the problem, noting that it is normally a good idea to use the telescope positions rather than the baseline vectors as model parameters. This will allow you to take advantage of the additional constraints arising from the fact there are 15 possible baseline vectors but we need only 5 independent vectors to represent the relative telescope locations.
   Compare your results with the results presented by ten Brummelaar et al. (2005, doi:10.1086/430729) who give the following table:

TABLE 1
AVAILABLE BASELINES

Telescopes	East (m)	North (m)	Height (m)	Baseline (m)
S2-S1	−5.748	33.581	0.644	34.076
E2-E1	−54.970	−36.246	3.077	65.917
W2-W1	105.990	−16.979	11.272	107.932
W2-E2	−139.481	−70.372	3.241	156.262
W2-S2	−63.331	165.764	−0.190	177.450
W2-S1	−69.080	199.345	0.454	210.976
W2-E1	−194.451	−106.618	6.318	221.853
E2-S2	76.149	236.135	−3.432	248.134
W1-S2	−169.322	182.743	−11.462	249.392
W1-E2	−245.471	−53.393	−8.031	251.340
W1-S1	−175.071	216.324	−10.818	278.501
E2-S1	70.401	269.717	−2.788	278.767
E1-S2	131.120	272.382	−6.508	302.368
W1-E1	−300.442	−89.639	−4.954	313.568
E1-S1	125.371	305.963	−5.865	330.705

NOTES.—These numbers are based on the results of a global baseline solution from mid-2004. The rms error of this fitting process was 1891 μm.

2. Analyse all the available nights of data in 2019 to see if there is any evidence for the telescopes moving due to seasonal effects between April and November of that year.
   You will need to have estimates for the uncertainties of your telescope position solution for each month. There are two possible to estimate the solution errors. The first is to compare solutions on two different nights in the same month and use the variation between solutions to estimate the solution uncertainty.
   The second way to derive uncertainty estimates is to use Singular Value Decomposition (SVD) of the design matrix to determine the U, w and V^T matrices. From this we can determine the uncertainty on a model parameter θ_j by using the singular values w_i and the columns of the V matrix (rows of V^T) as follows:

$$\sigma^2(\theta_j) = \sum_i \left(\frac{V_{ji}}{w_i} \right)^2 \sigma_d^2$$

   where σ_d is the RMS error on the delay measurements, assumed to be the same for all measurements in a given fit. Remember to check for particularly small values of the singular values w_i, corresponding to degeneracies, and replace $1/w_i$ with zero in these cases. Small is usually defined as around $N\epsilon$ times the maximum singular value where N is the number of data points and ϵ is the machine precision.
   You can estimate σ_d from the residuals in your data, assuming that χ^2 of the best-fit model is approximately equal to the number of degrees of freedom.
3. Now analyse the data from the `2012_all_v2.csv` file, which contains data from all the nights in 2012 to see if there is evidence for motion of the telescopes between 2012 and 2019.
   Again it is important to derive uncertainties in order to quantify whether any differences are real or just due to experimental uncertainty.

It may help in quantifying uncertainties to divide the 2012 dataset into subsets (perhaps one or two months' worth of data per subset) and compare the solutions from different subsets.
In the 2012 dataset there are typically fewer measurements per night, which leads to the possibilities in degeneracies in the data, i.e. singularities in the design matrix, particularly if you choose subsets which contain only a few nights of data.
You can find any degeneracies in a given dataset by using SVD of the design matrix and looking for particularly small singular values. The corresponding column of the V matrix can be used to understand the linear combination of model parameters which are involved in the degeneracy (see "Numerical Recipes" for more information on how to interpret and deal with singular values which are close to zero). Use this to interpret the form of the degeneracies and whether/how they affect the model parameters we are interested in, namely the telescope locations.

4. If you have time, you could also look for evidence seasonal variations in the telescope positions during 2012.

Remember to check and potentially "clean up" your data: for example you should plot residuals to look for outliers. Make sure to present visualisations of the data which back up your conclusions.