Computational Physics 2020/21 Projects: Submission Guidelines

You only need to complete ONE of Projects 1–4.

Some points to consider when preparing your project and submitting:

- Submission deadline is **Monday, December 21st 2020** at **12:00 (noon)**.
- The word limit for the Project is **2500 words** (excluding figure captions, appendices, references etc.)

  Report-writing guidelines

  - Your report should include the following elements; an abstract, brief descriptions of the aims of the investigation and numerical methods used to achieve this, details of how you validated your code, a summary of the results, a discussion of the results, and a conclusion. (If there are several distinct & substantial investigations, then split/arrange the results and discussion at your discretion.)
  - With the "Results" block of marks there are marks allocated for each of the tasks in the script. Therefore it is prudent to ensure that you cover these in the report and give 'evidence' of the 'outcome' of each task. (See 'Marking Scheme' below, for more details.)
  - Relate your numerical results to the physics of the system under investigation.
  - Ensure the report is readable, the meaning of the text is unambiguous and the document is coherent in its entirety with good linkage and logical flow. This is not a coding project with a bit of explanatory text to be submitted alongside it; rather, it is a full physics project where the exploratory studies happen inside a computer, and the report is a full write-up describing this.
  - Make sure figures are numbered and referred to in the main text. Make sure axes are labelled, a legend is present if needed, fonts are not tiny, etc. In particular, the caption must be meaningful; not "A graph of voltage versus current", but "The dependence of the voltage on the current, where the curve is the expectation from the analytical solution, and the points are the results of the numerical calculations. The behaviour of the different algorithms starts to diverge from about X μA, and it can be seen that the XXX algorithm is the most stable of the four that were implemented." Ensure plots are of an adequate size to be clear when the document is printed.
  - For those who have completed all the tasks listed in the project: additional work that makes use of computational techniques from the lectures that are not used in the main project tasks will be considered as further evidence for good understanding and presentation.
- Make use of the feedback you received on the Assignment.
- Departmental rules on late submission – **zero marks after 24 hours, pass mark-only up to 24 hours** – will be applied unless formal mitigating circumstances are lodged. Strict timekeeping will be applied in determining if a submission is late.
- **Report Submission** – The project report must be in PDF format and submitted via the relevant assignment link on Blackboard Learn. For instance, if you choose Project 3 you would submit under

  `Computational Physics (2020-21)`
  `> Course Content`
  `> Projects`
  `> Project 3`
  `> Computational Physics (2020-21) -- Project 3 -- Report Submission`

  This component of your submission will be checked for plagiarism via **TurnItIn**.
- **Source code Submission** – Your source code must also be uploaded via the relevant assignment link, e.g. for Project 3

  `> Computational Physics (2020-21) -- Project 3 -- Source Code Submission`

  on Blackboard Learn (in the same folder as above). This component of your submission will be also be checked for plagiarism by a code similarity analysis system.

The source code can be `.py`, `.cpp`, `.c`, `.h`, `.m`, etc., source files. "Notebook"-style files (e.g., Jupyter notebook `.ipynb` files) are also acceptable. Multiple files must be combined into a single ZIP file before uploading. Its name should include your surname & initials, e.g., `Other_AN_Project?.zip` where `?` is `1`, `2`, `3` or `4`. All files required for your code to compile (e.g. including header files in `C`) must be present. It is advisable to include a README text file explaining how to generate results presented in your report with your code(s). It must be made trivial for your marker to be able to run your code in the appropriate manner—no "comment out line 15, and uncomment line 27, then compile…" etc.

The idea is that the marker should not have to figure anything out, or have to follow onerous instructions to be able to run the code and reproduce your results; you do the work for them. This is not simply for the benefit of the markers; in real life—in physics and elsewhere—packaging your code and documenting it in a way that makes sure there is no guesswork involved in using it is common practice, and you should be acquiring this skill also.

- We suggest using LaTeX to write your report, if possible. This will improve the presentation, look, organisation, and figure and equation handling. PDF output comes as a standard feature in LaTeX.

  If you use Microsoft Word, please do not drag and drop Excel graphs directly into your Word documents. This will make the file too large and it may not be printed to PDF correctly. Instead save or export the plot into a PDF file or other image standard and import it into your Word document. We do strongly recommend that you try to move your workflow away from Excel and Word though; you will find technical report writing is a lot easier if you switch to some other plotting software and use LaTeX.

Marking Scheme

- Organisation of report – 10%.
- Presentation – 10%.
- Quality of programming – 20%.
- Results and understanding – 60%.

The 'Results' are outputs from the tasks that you are prompted to do in the project script. These can be: implementation & testing of a numerical method, numerical results (values, plots, tables) obtained from your investigation, analytical work you are asked to do, etc.

Your project will be **first and second marked** by the team of demonstrators and we aim to get back to you with a preliminary mark with feedback by Monday Feb 1, 2021. The actual overall course marks will be finalised in the summer at the same time as all other exams.

Trouble submitting with Blackboard Learn?

- Submissions can be made any number of times up to the deadline, and the most recent one will count. Do not leave it until the last minute! If you do let yourself fall into the situation that you are having problems submitting a file as the deadline approaches, please email us a copy of both the source code and report, so that you can prove that they were ready on time.
- **Blackboard Learn help pages** – Guidance on fixing problems;
  https://www.imperial.ac.uk/admin-services/ict/self-service/teaching-learning/blackboard/
- **O/S + web-browser compatibility?** – On the help pages above it says:
  "Blackboard Learn works with a variety of web browsers. If your browser does not work with Blackboard, update it or try another one."
- **Java & pop-up blockers?** – On the help pages above it says:
  "Make sure you have a recent version of Java and that popups are not disabled in your browser."

Plagiarism

All the work that you submit for assessment – the text in the report, the code, the results and plots – must be your own. Any help from your colleagues or others must be clearly acknowledged in your submitted work. Occurrences of plagiarism are taken very seriously and can have serious consequences. See the Departmental Policy on Plagiarism at https://www.imperial.ac.uk/natural-sciences/departments/physics/students/current-students/undergraduates/lecture-courses/plagiarism/ for more information.

As mentioned above, both the report and code will be checked by plagiarism detection systems. Note that coursework submissions from previous years are in the databases of these systems, so do not risk copying from previous cohorts!

Unfortunately, a small minority of students have been caught out in recent years. Hopefully this will not happen this year.

Mitigating Circumstances

The Departmental policy on Mitigating Circumstances is at https://www.imperial.ac.uk/physics/students/current-students/student-welfare-and-wellbeing/mitigating-circumstances/.

In particular since Computational Physics coursework is classed as a **major piece of assessed coursework**, the following applies;

- *"If you have extenuating/major mitigating circumstances that could affect your examinations or major pieces of assessed coursework/projects, please complete a 'Request for Mitigation Form'. Please note, you are expected to have contacted the Senior Tutor in the first instance, reporting your mitigating circumstances and to schedule an appointment."*

Project 1: A log-likelihood fit for extracting neutrino oscillation parameters

Key Topic: Functional Minimisation

1 Background

Neutrinos are one of the fundamental particles in the Standard Model of Particle Physics. They are the neutral partners to the charged leptons (electrons, e, muons, μ, and taus, τ) and come in three different 'flavours' - ν_e, ν_μ and ν_τ. When neutrinos interact they create their associated charged lepton (*i.e.* a ν_e will produced an electron), allowing us to identify the flavour of the neutrino.

In the Standard Model neutrinos are massless particles. However, at the turn of the millennium the Super-Kamiokande and SNO experiments observed deficits in the expected rate of neutrino interactions from the atmosphere and the Sun. This was the discovery of neutrino oscillations, a quantum mechanical phenomenon that causes neutrinos to oscillate between the flavours as they travel. Neutrino oscillations are only possible if neutrinos have non-zero masses, and is therefore the first observation of physics beyond the Standard Model.

Eqn. 1 is the first-order approximation to the 'survival probability' of a muon neutrino of energy E (GeV) as it travels a distance L (km). This is the probability that the muon neutrino will be observed as a muon neutrino and will not have oscillated into a tau neutrino.

$$P(\nu_\mu \longrightarrow \nu_\mu) = 1 - \sin^2(2\theta_{23}) \sin^2\left(\frac{1.267\Delta m^2_{23} L}{E}\right). \tag{1}$$

Here θ_{23} is the 'mixing angle', the parameter that determines the amplitude of the neutrino oscillation probability, and Δm^2_{23} (eV^2) is the difference between the squared masses of the two neutrinos, which determines the frequency of the oscillations. Long-baseline neutrino experiments measure the rate of muon neutrino events as a function of energy (at a fixed distance L) in order to measure these two oscillation parameters.

Neutrinos only interact through the weak nuclear force, which as the name might suggest is exceedingly weak. To put this in context, the neutrino interaction length is most easily measured in light years, compared to millimetres or centimetres for photons. Neutrino experiments therefore use extremely intense sources of neutrinos and extremely large detectors to gather data. One such experiment is T2K, which uses a 500 kW neutrino beam and a 40,000 tonne detector to measure neutrino oscillations. Even so, the weak interaction strength means that T2K is a statistically limited experiment, with $O(100)$ observed neutrino events. Given the low statistics, the number of events is best represented by a Poisson distribution, which naturally leads to the use of a *Negative Log Likelihood* (NLL) fit to extract the oscillation parameters.

2 Negative Log Likelihood fits

Let us consider a probability density function $\mathcal{P}$. In an NLL fit we evaluate the *likelihood* of a given measurement m to come from $\mathcal{P}$ by simply evaluating $\mathcal{P}(m)$. The combined likelihood of n independent measurements $\mathbf{m}$ is given as:

$$\mathcal{L} = \prod_{i=1}^{n} \mathcal{P}(m_i). \tag{2}$$

If we now consider an ensemble of probability density functions $\mathcal{P}(\mathbf{u})$, where $\mathbf{u}$ is the set of unknown parameters we want to estimate, we can calculate the likelihood as a function of $\mathbf{u}$:

$$\mathcal{L}(\mathbf{u}) = \prod_{i=1}^{n} \mathcal{P}(\mathbf{u}; m_i). \tag{3}$$

The value $\mathbf{u}_m$ where $\mathcal{L}(\mathbf{u})$ takes the maximal value will represent the best fit between the data (the set of $\mathbf{m}$) and the probability density function, and we can take the $\mathbf{u}_m$ as estimators of the true values of $\mathbf{u}$. Instead of finding the maximum of Eqn. (3), we will find the minimum of the *Negative Log Likelihood* (NLL)

$$\mathrm{NLL}(\mathbf{u}) = -\ln\left(\prod_{i=1}^{n} \mathcal{P}(\mathbf{u}; m_i)\right) = -\sum_{i=1}^{n} \ln\left(\mathcal{P}(\mathbf{u}; m_i)\right). \tag{4}$$

where $\ln(x)$ means the natural log of x.

2.1 NLL fits for Poisson distributed variables

In High Energy Physics experiments, the number of entries in each bin of a data histogram can be treated as a discrete measurement whose probability follows the Poisson distribution. For one bin, the probability density function for a Poisson-distributed integer number of entries m is

$$\mathcal{P}(m) = \frac{\lambda^m e^{-\lambda}}{m!}. \tag{5}$$

where λ is the expected average number. For many bins, m_i denotes the observed number of neutrino events in bin i. Simulations are used to predict the expected number of events λ_i in bin i. This prediction will depend on the set of unknown parameters, $\mathbf{u}$, giving $\lambda_i(\mathbf{u})$.

The general NLL formula above for the Poisson case becomes

$$\text{NLL}(\mathbf{u}) = -\ln\left(\prod_{i=1}^{n} \frac{\lambda_i^{m_i}(\mathbf{u})\, e^{-\lambda_i(\mathbf{u})}}{m_i!}\right) = \sum_{i=1}^{n}\left[\lambda_i(\mathbf{u}) - m_i + m_i \ln\left(\frac{m_i}{\lambda_i(\mathbf{u})}\right)\right]. \tag{6}$$

where Stirling's approximation $\ln(m!) \approx m\ln(m) - m$ has been used.

3 Project

The aim of the project is to create an NLL fit to extract the neutrino oscillation parameters of Eqn. 1 from a set of simulated T2K data. We can apply Eqn. 6 to our situation with $\mathbf{m}$ being measurements of the muon neutrino event numbers, binned as a function of energy, and $\mathbf{u} = (\theta_{23}, \Delta m^2_{23})$.

3.1 The data

Create a function to read the experimental data and the unoscillated event rate prediction from the supplied data file. Personalised data files can be downloaded from:

`https://www.hep.ph.ic.ac.uk/~ms2609/CompPhys/neutrino_data/username.txt`,

where you should replace `username` with your college username (xyz123, for example). Please note that copying and pasting this text from the PDF to a browser will not work, due to different text encodings - please type the link out in your browser!

Each data file contains the observed number of muon neutrino events from 0–10 GeV in energy, with each new line representing a single 'bin' of energy. There are 200 energy bins in total, so, for example, the first number in the file therefore represents the number of events observed with an energy between 0 GeV and 0.05 GeV.

Create a few histograms of the data to make sure you understand what you are looking at.

3.2 Fit function

Code the oscillation probability $P(\nu_\mu \longrightarrow \nu_\mu)$ using Eqn. 1, and create some plots as a function of energy, E, to make sure you understand the effect of changing the oscillation parameters. Use the following initial values for the variables: $\theta_{23} = \frac{\pi}{4}$, $\Delta m^2_{23} = 2.4 \times 10^{-3}$ and $L = 295$.

The data file contains both the data and the simulated event number prediction assuming the muon neutrinos do not oscillate. Code a function to apply the oscillation probability from Eqn. 1 to the simulated event rate to produce $\lambda_i(\mathbf{u})$, the oscillated event rate prediction.

Comparing the plots to the data by eye should give you a rough estimate of the oscillation parameters.

3.3 Likelihood function

Write a function to calculate the negative log likelihood, Eqn. 6. Creating a graph of the NLL as a function of θ_{23} should allow you to find the approximate position of the minimum.

3.4 Minimise

Code a parabolic minimiser as presented in the lectures. The interval you search for the minimum in should be based on your rough estimate of the value for θ_{23}. Test the minimiser on the oscillation probability function (or equivalent validation function) first to make sure it works as expected. You will need to develop a criterion for when the method has converged. Now go on to find the value of θ_{23} that minimises the NLL.

3.5 Find accuracy of fit result

The accuracy of the fit results will depend on the available statistics of the input. In general for an NLL fit we can estimate the error by looking at the shape of the NLL function around its minimum. The values θ_{23}^{+} and θ_{23}^{-} where the NLL is changed by 0.5 (i.e. in absolute units, not a factor 2) compared to the value at the minimum correspond to a shift of one standard deviation in the positive and negative direction. You can also estimate the error from the curvature of the last parabolic estimate. Do this and compare to the result from the estimate based on the change in the NLL. Comment on the relative merit of the two methods.

Create a method to automatically find the standard deviation. The error can be estimated from the scans of the function close to the minimum if you cannot get an automatic method to work. You should at this stage be able to quote a measurement of θ_{23} with an error based on the complete dataset. Comment on how you would expect the error on this measurement to change as θ_{23} approaches $\frac{\pi}{4}$.

4 Two-dimensional minimisation

4.1 The univariate method

As you know from Sec. 3.2, the oscillation probability depends upon both θ_{23} and Δm^2_{23}. Changes in Δm^2_{23} could be confused with changes to θ_{23}, so in practice a two-dimensional fit is used to extract values of both parameters.

Implement the 'Univariate' multi-dimensional fitting method discussed in the lectures to find the best fit values of both θ_{23} and Δm^2_{23}. Consider the order in which you should minimise the variables to get the best result.

4.2 Simultaneous minimisation

A more correct way to approach this problem is to simultaneously minimise the NLL with respect to both θ_{23} and Δm^2_{23}. Develop your own simultaneous two-dimensional minimiser and validate it using an appropriate function. Use this to extract the values of θ_{23} and Δm^2_{23} from your dataset, including their uncertainty. Comment on any difference between these results and what you measured previously.

5 Neutrino interaction cross-section

The neutrino interaction cross-section governs the expected rate of neutrino events - if the cross-section increases then the number of observed events per neutrino should increase as well. So far we have assumed that the neutrino interaction cross-section is constant as a function of neutrino energy. In reality the cross-section increases approximately linearly with energy, and the rate of increase is not well known.

- Code up a new function to predict $\lambda_i(\mathbf{u})$, including the effect of the cross-section, so that the expected number of events is scaled in proportion to the neutrino energy. Since the cross-section is not well known it should be included as another free parameter in your NLL minimiser. The new predicted number of events in a histogram bin should look something like:

$$\lambda_i^{\text{new}}(\mathbf{u}) = \lambda_i^{\text{old}}(\mathbf{u}) \cdot \alpha \cdot E_\nu \tag{7}$$

  where α is the free parameter.

- Update your minimiser to handle this new parameter and then produce an estimate of θ_{23}, Δm^2_{23} and the rate of increase of the cross-section with neutrino energy.

- Comment on your results.

Project 2: Solving quantum systems numerically

Key Topic: Numerical integration and Monte Carlo methods

1 Introduction

Consider a time-dependent 1D quantum-mechanical system, consisting of the wavefunction for a single particle $\psi(x,t)$. Now imagine designing an experiment to measure the position of this particle at a particular time. From quantum mechanics, you know that the probability of measuring the particle's position to be somewhere between x_1 and x_2 is given by the square of the absolute value of the wavefunction $|\psi|^2$, integrated between x_1 and x_2. Many wavefunctions cannot be integrated analytically, so there is not a closed-form expression for the probability of finding the particle in the range $x \in [x_1, x_2]$. In this assignment, you will calculate this probability numerically. This is much more powerful than the analytical approach, as it allows you to obtain a solution regardless of the form of the wavefunction.

2 Newton-Coates integration

a) Write a general routine to perform numerical integration on a user-supplied 1D function using the extended trapezoidal rule. Here 'user-supplied' means that you should design your integration function so that one of its arguments is a pointer (or equivalent depending on your coding language) to another function, which will be the integrand. This other function should take a single argument (the value of the integration variable) and return the value of the integrand at that value of the integration variable. Design your program so that it automatically refines the result to a user-specified relative accuracy ϵ.

b) Write another routine to do the same as a), but this time using the extended version of Simpson's Rule. Note the trick that relates this to evaluations of the Trapezoidal Rule, discussed in class and in Numerical Recipes!

c) Consider a particle in a simple harmonic oscillator potential. The ground state time-independent wavefunction is

$$\psi_0(x) = \left(\frac{m\omega}{\hbar\pi}\right)^{1/4} e^{-m\omega x^2/2\hbar}, \tag{1}$$

where ω is the natural angular frequency of the oscillator. Use both your integrators from a) and b) to compute the probability of finding the particle between $x_1 = 0$ and $x_2 = 2\sqrt{\hbar/m\omega}$, i.e.

$$P \equiv \int_0^{2\sqrt{\hbar/m\omega}} |\psi_0(x)|^2 \, \mathrm{d}x, \tag{2}$$

to a relative accuracy of $\epsilon = 10^{-6}$. Compare the number of function evaluations required in each case.

3 Monte Carlo integration

Write a Monte Carlo integration routine that can integrate a user-supplied function using importance sampling, and automatically refines its result to a user-specified relative accuracy ϵ. Perform the integral Eq. 2 twice more, this time using your Monte Carlo integrator with

a) A flat sampling PDF (i.e. no importance sampling)

b) A sampling PDF that drops off from a maximum at x_1 to a (non-zero) minimum at x_2, e.g.

$$\mathrm{PDF}(x) = Ax + B, \tag{3}$$

choosing A such that the PDF does not go to zero (or negative) at x_2. Note that you are free to hardcode the probability distributions for importance sampling directly into your integrator.

Compare the number of function evaluations required in a) and b) to obtain different relative accuracies (say $\epsilon = 10^{-3}$, $\epsilon = 10^{-4}$, $\epsilon = 10^{-5}$ and $\epsilon = 10^{-6}$). If you have trouble achieving one or more of these accuracies, discuss why. Give estimates for how many more steps you expect your algorithm would take to reach this accuracy, based on its performance so far and the results you found in Q1.

Try adding the option to perform adaptive importance sampling, and then compare the performance to your results in b).

4 Higher numbers of dimensions

Now consider a particle in 3D space, for which the wavefunction is $\Psi(x, y, z)$. Generalise your code from both methods (Newton-Coates and Monte Carlo) to perform the probability integral for such a wavefunction in all three dimensions.

a) Consider the particle in a simple harmonic oscillator potential. The overall ground state is the product of three 1D ground state functions

$$\Psi_0(x, y, z) = \psi_0(x)\psi_0(y)\psi_0(z) \tag{4}$$

Integrate the resulting probability over a cube from 0 to $2\sqrt{\hbar/m\omega}$ in all three dimensions using both methods. Comment on your result and the relative execution time of the methods.

b) The first excited state of the 1D oscillator is

$$\psi_1(x) = \left(\frac{m\omega}{\hbar\pi}\right)^{1/4} \sqrt{\frac{2m\omega}{\hbar}}\, x\, e^{-m\omega x^2/2\hbar}, \tag{5}$$

In 3D, a state with one unit of orbital angular momentum can be formed by combining the 1D states according to

$$\Psi_1(x) = [\psi_1(x)\psi_0(y) + i\psi_0(x)\psi_1(y)]\psi_0(z) \tag{6}$$

Perform an integration over the same cube volume to find the resulting probability.

Project 3: The Dynamics of Solitons

Key Topic: PDE Initial Value Problem

1 Background

A soliton is a single pulse that can propagate without change of shape. The earliest observations of solitons were of single water wave pulses in canals and rivers; the Severn Bore is a well-known example. We will be concerned with soliton solutions to the Korteweg de Vries (KdeV) partial differential equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{\partial^3 u}{\partial x^3} = 0, \tag{1}$$

where x and t have already been rescaled into dimensionless units. The equation has exact solutions of the form

$$u = 12\alpha^2 \operatorname{sech}^2(\alpha(x - 4\alpha^2 t)). \tag{2}$$

where α is a parameter. Equation (1) differs from our usual wave equation in two important aspects:

- The second term is non-linear, giving a wave speed that depends upon the amplitude of the disturbance, u; this is typical of shock waves.
- The third term introduces dispersive broadening of the waveform.

Whilst both the non-linearity and dispersion on their own cause distortion of the waveform, when combined in the KdeV equation the two effects cancel each other out exactly to allow solutions of the form (2) to propagate without change of shape. In this project you will write code to integrate the KdeV equation forward in time and use it to study the dynamics of single solitons, collisions between solitons, wave breaking and other non-linear wave phenomena.

2 Discretisation

The KdeV equation can be discretised using centre difference approximations for the spatial derivatives and a forward difference for the time derivative, as follows:

$$u_i^{j+1} = u_i^j - \frac{1}{4}\frac{\Delta t}{h}\left[\left(u_{i+1}^j\right)^2 - \left(u_{i-1}^j\right)^2\right] - \frac{1}{2}\frac{\Delta t}{h^3}\left[u_{i+2}^j - 2u_{i+1}^j + 2u_{i-1}^j - u_{i-2}^j\right]. \tag{3}$$

Show that this is consistent with the differential equation. Analyse the stability of the difference scheme in the limit of small u and show analytically that it is unstable. Choose an alternative method for the time discretisation; various methods will do, but a second-order (or higher) Runge-Kutta method is suitable. You do not need to prove analytically that this modified method is stable.

3 Dynamics

Write a program to propagate the pulse according to your discretisation of the KdeV equation. It will be convenient to use periodic boundary conditions, provided that the total spatial extent is large compared to the width of the pulse[1]. Now validate your implementation of the KdeV solver:

- Generate solitons with different values of α and confirm that they are propagated for at least a few transits across the spatial domain without change of shape.
- Observe how the speed of the soliton is related to its height.
- Investigate how the values of h and Δt needed for stability depend on the soliton parameter α.

[1] One way to do this is with the modulus, so use $(i \pm 1) \bmod N$ when calculating the index of the neighbour. Here N is the number of spatial points.

3.1 Collisions of solitons

You will have noticed that taller solitons move faster than smaller ones. Therefore a fast soliton should catch up and "collide" with a slower one moving in the same direction. To simulate this, set up two independent solitons of different speed and well-separated initial position. Study how the solitons interact, paying particular attention to the shape of the pulse during and after the collision. Since the waves are non-linear we do not expect height to be conserved (no linear superposition). However, you should find that another property of the wave is conserved during the collision. Study these effects for solitons of similar and very different initial speeds.

3.2 Wave breaking

Solutions of the form (2) may be considered as normal modes of the KdeV equation. Here you will study how the KdeV equation evolves a different initial waveform, which is not of the normal mode form. Try an initial waveform such as the positive part of a sine wave. You should find that the wave breaks up into a train of solitons of different sizes. An example of a sine wave breaking is shown in [1]. In the units of the differential equation (1), you should find that you need a long wave period relative to the amplitude of the initial sine wave in order to observe wave breaking.

3.3 Shock waves

The KdeV equation has stable solutions of the form (2) because of the balance of non-linear and dispersive terms. If we remove the final term in equation (1), we are left with a wave equation for a form of shock wave. Try this with your code and see what happens to the shape of the waveform (2) as it progresses. You will find the solution becomes unstable at some point; this may be rectified by introducing a diffusive term (of the form $D\,\partial^2 u/\partial x^2$ where D is the diffusion coefficient) into the new PDE to dampen the waves.

References

[1] N. J. Zabusky and M. D. Kruskal, Phys. Rev. Lett. 15, 240-243 (1965).

Project 4: Heat dissipation in microprocessors

Key Topic: PDE solving with Neumann boundary conditions

1 Introduction

An Intel Core i7 microprocessor working at 3.4 GHz produces around 95 W of thermal power. This is an enormous amount of heat that, if not extracted fast enough, will increase the temperature of the device, disrupt its normal working conditions and, eventually, damage it - maybe even melt it. These processors should not work (at least not for too long) above 60-80°C!

Much research worldwide is devoted to making the transistors in the microprocessors smaller, developing novel semiconductor materials and device architectures to waste less energy, and implementing smart ON/OFF switching sequences and sleeping routines for the unused parts of the device. However, heat - a lot of it - will still be produced.

In this project, you will study how heat is dissipated from a microprocessor, what temperature it can reach and how it can be reduced using appropriate heat dissipation structures.

2 The maths

The general heat transport equation in steady state (independent of time) relates the flux of thermal power $\vec{\phi} = -k\vec{\nabla}T$ (where $T(x, y, z)$ is the temperature) to the heat generated/consumed at a given point in space q:

$$\vec{\nabla} \cdot \vec{\phi} = q(\vec{r}) \tag{1}$$

In a homogeneous medium, i.e. in which k is a constant, this equation is an elliptic PDE that relates the temperature and the thermal power fluxes and that takes the form of Poisson's equation:

$$-k\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2}\right) = q(x, y, z) \tag{2}$$

where k is the thermal conductivity of the material (W/(m K)) and q is the power density (W/m^3) generated at any given position.

At the surfaces, heat dissipates into the ambient surroundings, assumed to be at $T_a = 20°$C, at a rate given by (W/m^2):

$$\phi_s = h(T_{surf} - T_a) \tag{3}$$

where h is the heat transfer coefficient of the interface (W/(m^{2}K)), with a value that depends on the dissipation mechanism. In this project you will consider two options: natural convection and forced convection. In the case of natural convection, the conductance depends on the temperature of the surface approximately as:

$$h_{natural} = 1.31(T_{surf} - T_a)^{1/3} \tag{4}$$

For the forced convection, h only depends on the wind speed v (m/s) as:

$$h_{forced} = 11.4 + 5.7v \tag{5}$$

What do the above equations look like in natural units?

2.1 Challenges

While Poisson's equation is relatively easy to solve, the challenge in this problem is designing the mesh and implementing the boundary conditions. As you can see in Eq. 3, the boundary conditions in this problem are given for the derivative, i.e. we have Neumann boundary conditions, and the surface is not rectangular as you will see in the next section. The latter should not be a problem, as you can always simulate a rectangular space but only include those points where you need the solution in the solving algorithm (e.g. any of the relaxation methods). You should choose a suitable convergence criterion and justify that choice based on the geometry of the problem.

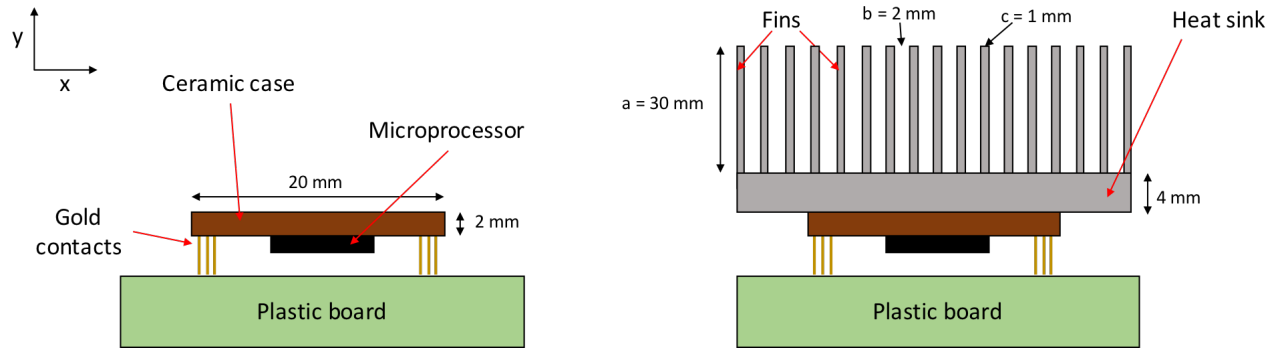

Figure 1: (a) Microprocessor with the ceramic case. (b) Including the heat sink with fins to improve heat dissipation.

.

3 No heat sink

Consider that you have a microprocessor with a width of w_m=14 mm, a thickness of t_m=1 mm and which is very long, such that you can ignore any length (i.e. z) dependence. The microprocessor is attached to a ceramic case that protects it and gives mechanical support. As shown in Fig. 1a, this case has a width of $w_c = 20$ mm and a thickness of $t_c = 2$ mm. This microprocessor, working at top speed, produces 0.5 W/mm^3 of thermal power.

Use any of the methods described in the lectures to solve Eq. 1 in two dimensions (ignore the z component) and obtain the average operating temperature of the microchip assuming only natural convection. Consider only the microprocessor and the ceramic case surrounded by air and ignore the contacts and the plastic board.

Is the resulting temperature acceptable as normal operating conditions? What would you expect if there were no convection whatsoever?

4 With heat sink

Normally, microprocessors have a heat sink attached to them that increases the area dissipating the heat and helps to reduce the temperature. Add a heat sink made of aluminium to your device (Fig. 1b), still using natural convection. The figure shows a fin separation of $b = 2$ mm, a fin thickness of $c = 1$ mm and a fin height of $a = 30$ mm, mounted on a 4 mm thick base. Study how the temperature of the microprocessor changes as you vary the number of fins (while keeping b and c constant). Try also changing the height a and separation b (with a minimum for b of 1 mm) of the fins, and discuss what works best. Can you reach a sensible working temperature (< 100°C)? Keep in mind that this should fit inside a computer, so the heat sink cannot be infinitely large!

5 With heat sink and forced convection

In many cases, even the heat sink is not enough when the microprocessor is working at top speed and some extra help extracting heat is needed. For this purpose, most microprocessors include a fan adding the circulation of air around the fins (or in the most extreme cases, a liquid coolant). To simulate this situation, add forced convection (at all boundaries, for simplicity) using a wind speed $v = 20$ m/s. How does this affect the microprocessor temperature? What is the minimum size (width and height) of the heat sink that permits the microprocessor to work at 80°C?

Appendix

Description	Material	Conductivity k (W/(m K))
Microchip	Silicon	150
Ceramic package	Copper Tungsten	230
Heat sink	Aluminium	250