ASTRO5010 / SUPATSA
The Sun's Atmosphere

Project Guide

Version 3 – January 12, 2020

1 Aims and Learning Outcomes

The aim of these computer projects is specifically to offer initial training in solar physics research which will be useful for students interested in pursuing a career in astrophysics or theoretical astrophysics.

The intended learning outcomes are

- use the theories, principles and ideas presented in the course to solve a computational data-analysis or modelling problem, in the context of the study of the solar atmosphere
- summarise in a written report the theories and methods adopted to solve the computational project
- produce a critical assessment of the project outcomes in the context of published research in professional journals

This course provides an opportunity to develop programming skills via the computer-based project. The language of choice is Python but other programming languages are acceptable. Some resources in Python are made available on this site and on the School's JupyterHub.

Students who are not comfortable with programming should be prepared to learn Python. For specific support, contact the course coordinator as early as possible who may be able to provide advice and guidance.

2 Projects

There is a choice of four projects:

1. Analysing EIS Data
2. Analysing Microwave Data
3. A Simple Radiative Transfer Model
4. Simulating Alfvén Waves

The first two provide an introduction to solar data analysis, while the latter two projects provide an introduction to modelling physical processes relevant to the study of the solar atmosphere.

While you are encouraged to work on *all four projects*, you are required to produce only one report on just one of the four above projects. You are free to choose which project you want to be assessed on. Each project script contains a number of tasks for students to complete.

Support to complete the project work will be available throughout the Semester by emailing the course coordinator or contact any of the other lecturers (details on Moodle).

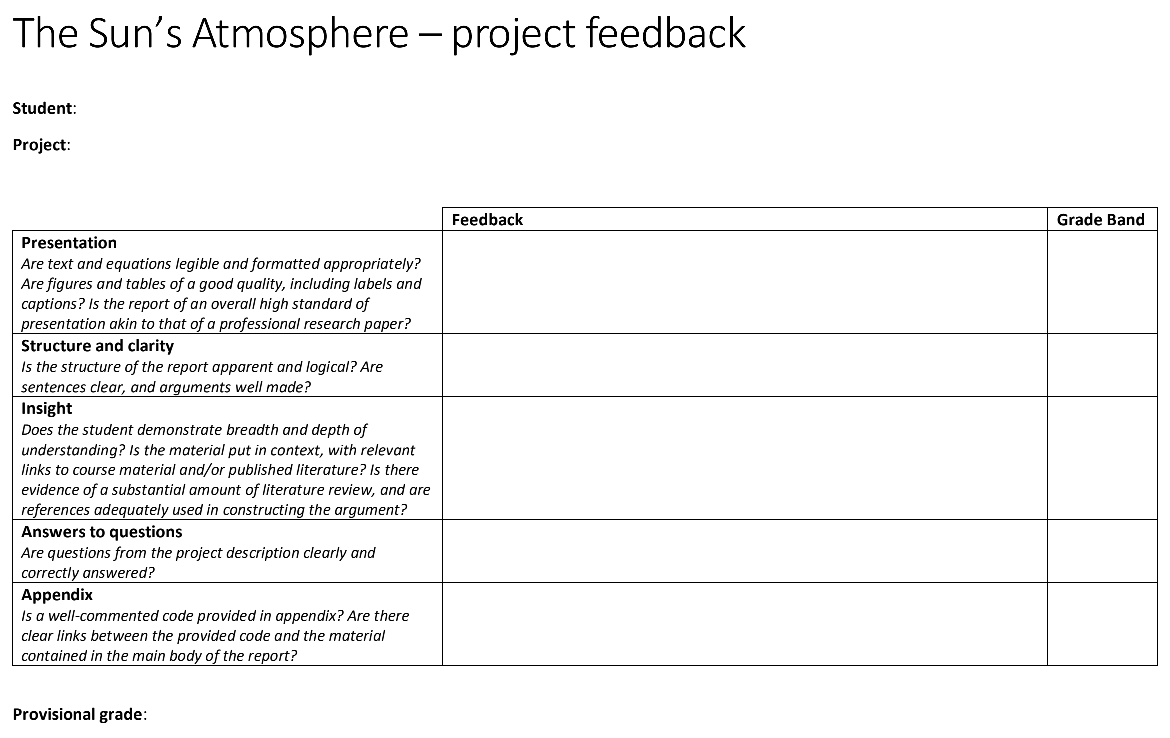
3 Assessment

Students taking ”The Sun’s Atmosphere” course are required to submit a final report on the computer-based project demonstrating understanding of issues associated to solar data analysis and their interpretation, or associated to theory and modelling of high-energy physics in the solar context. This constitutes 50% of the final grade in this course.

The deadline for submitting the report on the chosen project is given on Moodle.

3.1 Marking criteria

A marking sheet providing clear information about criteria used to assess the reports is shown below.



The Sun’s Atmosphere – project feedback

Student:

Project:

	Feedback	Grade Band
Presentation *Are text and equations legible and formatted appropriately? Are figures and tables of a good quality, including labels and captions? Is the report of an overall high standard of presentation akin to that of a professional research paper?*		
Structure and clarity *Is the structure of the report apparent and logical? Are sentences clear, and arguments well made?*		
Insight *Does the student demonstrate breadth and depth of understanding? Is the material put in context, with relevant links to course material and/or published literature? Is there evidence of a substantial amount of literature review, and are references adequately used in constructing the argument?*		
Answers to questions *Are questions from the project description clearly and correctly answered?*		
Appendix *Is a well-commented code provided in appendix? Are there clear links between the provided code and the material contained in the main body of the report?*		

Provisional grade:

Categories include Presentation, Structure and clarity, Insight, Answers to questions from project guide, and Appendix.

MSc students The marker gives a primary grade for each category. An overall grade point is then awarded using the 22-pt scale.

SUPA students The marker gives a primary score for each category. An overall mark is then awarded using the 100-pt scale.

A second marker moderates these marks. In case of significant discrepancy, a third academic marker is called upon.

4 Report guidance

The report has no strict page limits. It should contain an introduction which will provide, among other things, the context of the assessment and will describe how the chosen project relates to the material discussed in the lectures. The main body of the report will describe the work done and the results obtained, using numbered sections. You should summarise the theories and methods adopted to solve the problem. There should be a discussion containing a critical assessment of the project outcomes in the context of published research in professional journals (see ADS). An appendix should also be included with a COMMENTED copy of the code used. All graphs, output, description and tables should be in the main body of text. The appendix is for commented code and algorithm description only.

It is important to interpret the results obtained, and to show evidence for any interpretations. As in a peer-reviewed article published in a research journal, try to be concise with your descriptions of what you did. They still need to be coherent (make sense) and describe what you have done to someone that does not know. However, it is equally important to include your insight and analysis.

You can use any style that you like (e.g. using article or report or another in Latex). The recommended font size is 11pt.

Your figures should have figure captions and graphs should have axis labels. You should also reference your figures and graphs in the text using their figure number (e.g. ”The result of xxx can be seen in Figure 1 and is found because yyy…).

Submission is electronically as a PDF file via Moodle. See instructions there. Marking criteria are detailed in §3.1. The report should demonstrate a firm understanding of the tasks completed and of the physical principles discussed, and interpretation should be provided for results obtained.

We hope you will have fun working on these projects!

Analysing EIS Data

NB You are expected to answer all questions in this lab explicitly in your report. You are also required to research information from scientific literature to apply in your analysis, citing all references used.

1 Introduction

The Extreme Ultraviolet Imaging Spectrometer (EIS) is an instrument on the Hinode spacecraft, launched in 2006 to a geocentric orbit for solar observation purposes. EIS records spatially resolved spectra and is used to identify spectral lines that can be used to deduce temperatures and investigate the coronal heating problem. In this lab you will use EIS spectral data to diagnose the temperature of a flare and investigate variations in line-broadening across the flare (whilst accounting for instrumental effects).

2 EIS Line Profile Analysis

2.1 Choosing Spectral Lines

There are fits data files provided with this lab that contain level-1 (preprocessed) EIS data in a format suitable for use with Python and Matlab. Extract these files to a suitable location to work on them. In Matlab these files can be loaded with the 'fitsread' function and using 'astropy.io.fits' in Python. Documentation for these, and all other library software used throughout the project can be found online. In the provided data files there are three folders containing the EIS data and errors for three different spectral lines in the flare we are investigating. You will also need to know how to get a list of file paths from the operating system (see 'dir' in Matlab and 'os.listdir' in Python).

The first task is to load the data for the spectral lines into a data structure that you can work with. In the following we will outline the recommended data structure, but you are free to use your own if you can think of a superior way of representing the information.

First find the paths for the files and then load them. Since we will be loading multiple similar sets of data it makes sense to define a function for the task. Note that python loads the data in whatever order it gets the information from the operating system, and not necessarily the correct order for the data set, hence the need to sort the data by wavelength after loading the fits files (the wavelength is stored in the fits header under the tag 'LAMBDA'). Then, once the data is sorted it is to be placed into a three-dimensional array for which the first axis is the wavelength of the observation, with the other two representing the row and column of the pixel respectively. This function should then return a fits header (from one file as they are all the same other than the wavelength information), the data array and the array of corresponding wavelengths.

The error files are loaded in *exactly* the same way as the data files since they have the same structure. These files specify the error on the measurement at each point per wavelength.

Now, to get a basic idea of the layout of the data it is good to have a look at a simple intensity map of the data. This is done by summing intensities along the wavelength axis of the data and displaying the resulting picture. With this EIS data, however, it is not quite so simple as the pixels are not square and this needs to be accounted for before we can present them in a way that accurately represents the flare. This can be solved by looking at the fits header, and setting the extent of the plot based on the centre and x and y steps (stored in the fits header).

2.2 Diagnosing the Plasma Temperatures

Plot an intensity map as described in the previous section (don't forget to insert a title detailing what is plotted, as well as labels for the axes). Whilst the emission maps tell us very little about the widths of the lines in the plasma, they do allow us to approximately diagnose the relative temperatures.

Question 2.1. By taking the strongest spectral line in each region (even by eye), suggest which parts of the flare and the background are hotter by the presence of the different lines.

2.3 Isolating Spectral Lines

After setting the plot to scale the emission maps correctly it is now labelled in terms of solar location (in arcseconds). This is not particularly useful for picking individual pixels out of the dataset, but luckily using the data stored in the fits headers it is fairly easy to translate between pixel-space and this ”solar-space”. It is recommended to write two helper functions to convert in each direction to achieve this task.

Now, we want to be able to look at the spectrum at a certain pixel of the data, this is as simple as slicing the array along the wavelength axis whilst fixing the x and y values. This value can then be plotted along with the error bars (symmetric) specified by the error files. It is sometimes helpful to indicate which pixel is being investigated on the emission map to remain aware of which region of the data set it is in.

Question 2.2. Plot a few sample line profiles for different ions / transitions and at different locations. What does that tell you about the observed region?

The following applies only to the Fe XVI spectral line, and the others cannot be used for the remainder of this lab.

Question 2.3. What are the reasons for rejecting the other spectral lines? (Hint: look at the shape of the line at various points across the flare and the raster – we expect an approximately Gaussian profile).

Whilst perusing the line shape at different points across the Fe XVI raster you may note that some points are set to an intensity of -100. This is to mark them as containing errors. These need to be corrected before the data on these pixels can be fit. A simple heuristic for this is (the complete version used in SolarSoft is better – see ‘eis_replace_missing‘): if a pixel is labelled as missing and neither of its neighbours are missing then use their average value and error. If one neighbour is also missing then just set the pixel value to the other neighbour's. If both surrounding pixels are missing then set the pixel to 0, or raise an error so that you know to pick a different pixel for the analysis.

The next task is to fit a line profile to the data. The default profile to use is a Gaussian with a constant background, although certain effects may require fitting with more complicated profiles such as a Kappa distribution (the convolution of a Gaussian and a power law). First create a function representing this mathematical function (in python this function takes an array of wavelengths and a tuple of parameters). The line does not take up the entirety of the frequency range, thus it would not be wise to use the entirety of the frequency range for fitting. Instead, use only the values within a +/-0.2 Å window of the official line value.

Fitting in Python (SciPy), Matlab and IDL all use the same core function, MINPACK, written in Fortran77 for non-linear least-squares fitting. In SciPy look for 'scipy.optimize.curve_fit'. Now this tool can be used to find the parameters for the previously defined Gaussian to several different points around the flare.

Question 2.4. Give several examples of observed spectral lines fitted with a Gaussian profile. Investigate how the line width varies at the foot-points, loop-top and several other (2+) points around the flare. Does this match what you expect? Be sure to calculate a goodness of fit here.

2.4 Non-Thermal Line Widths

An ideal spectral line, viewed through an ideal detector would be incredibly narrow. The widths here, however, are not insignificant and are due to the compounding of several effects: natural broadening and collisional broadening (insignificant here), thermal broadening (from the motion of the ions), non-thermal broadening, and instrumental broadening. All of these different effects add in quadrature to give the total effect. Instrumental broadening is a significant effect here, amounting to 0.067 Å over the flare (this value varies slightly with the Y position of the pixel). Thermal broadening can be calculated as per the usual formula. Take the plasma to have $\log_{10} T = 6.4$.

Question 2.5. Using this information, estimate the non-thermal width of the spectral line at the same points as previously around the flare. How does this width vary, and does this match what you expect? Once again, calculate a goodness of fit.

2.5 Fitting the Entire Raster

Another task that is commonly performed in spectroscopy is fitting the line for every pixel in the raster. This is interesting because it allows the integration of a "true" emission map from the fitted parameters. This task is less easy than it may appear as "background" pixels have a very low signal-to-noise ratio on this line, making it extremely difficult to fit. Some empirical heuristics will therefore be required to avoid large errors here. In the commonly used SolarSoft package, the function 'eis_auto_fit' is used for this purpose and could be the origin of some inspiration for you (this function is written in IDL and can be found by searching the internet). By using the fit across the raster and the standard result of the Gaussian integral estimate an emission map with true numerical values for each pixel.

Analysing Microwave Data

NB You are expected to answer all questions in this lab explicitly in your report. You are also required to research information from scientific literature to apply in your analysis, citing all references used.

1 Introduction

In this lab you will use data from the Nobeyama Radio Polarimeters (NoRP) and two telescopes in the Radio Solar Telescope Network (RSTN). NoRP is a set of full sun polarimeters based in Japan that has been operating at frequencies up to 80 GHz from 1984. RSTN is an array of five American operated observatories around the world that provide constant solar observations. Originally set up to monitor space weather events ahead of the Apollo missions, these observatories still provide useful scientific information at lower frequencies than Nobeyama. Using data from these observatories you will investigate solar emission during a flare event.

2 Theory

2.1 Gyrosynchrotron Emission

The data you will be using in this lab is in the microwave range (0.245 GHz – 80 GHz). In this region the principal flare emission mechanism is gyrosynchrotron (gyro) emission. This emission occurs when mildly relativistic electron spiral around the magnetic field line of a flare. Strong gyro emission is dependent on a sufficiently high electron density and can be suppressed, especially at low frequencies. This suppression is named Razin suppression and occurs when the index of refraction deviates significantly from unity. Gyro emission is also affected by free-free absorption at low temperatures and self-absorption at high electron densities in strong magnetic fields.

The analytical expression of gyro spectra is extremely complex (Ramaty 1969 for derivation). Simpler functions can however be fitted to actually observed spectra (Stahli, Gary and Hurford, 1989).

Question 2.1. Using Stahli et al (1989), find the function that can be fit to gyro spectra and how its parameters relate to the parameters of the spectrum.

2.2 Plasma Emission

There is an additional emission mechanism operating at the low-frequency range of the data set ($<$ 2 GHz) which is termed plasma emission. This emission is the production of electromagnetic waves by non-linear coupling of Langmuir waves in the plasma. This emission is not interesting for the study of gyro emission, but should be taken into account in the fitting.

3 Analysis

3.1 The event

The event that you will be investigating is an X1 class flare with its peak at 01:59:38UT 28-10-2013. A video of the event from AIA on SDO in UV can be found here and a gif of the CME as seen by LASCO on SoHO can be found here.

With this lab 3 sets of data are provided. This data is all in comma separated value (csv) format. More details about the format will be provided in the two Sections on loading data.

3.2 Loading the NoRP Data

In the folder ‘norp_event_131028‘ are the csv files associated with this event. ‘freq.csv‘ contains the list of the observed frequencies, ‘fi.csv‘ contains an array of the Stokes I parameter (total flux) for each reading of each frequency. ‘mvd.csv‘ contains a boolean array of which frequencies in ‘fi.csv‘ are valid where 1 indicates valid and 0 indicates that the data should be discarded. ‘tim.csv‘ is an array of the milliseconds since the start of the day and the number of days since 1979-1-1 which can be used to provide a consistent time-system with which to combine the instruments. The Nobeyama event data goes from 01:53:40UT – 02:11:34UT on 2013-10-28.

All of these files can be parsed easily by csv parsers, directly into a ‘numpy.array‘ in Python using ‘genfromtxt‘ and using ‘csvread‘ in Matlab. The fields in ‘mvd.csv‘ should be parsed as integers and in ‘tim.csv‘ should be parsed as (unsigned) 64-bit integers to avoid overflow.

3.3 Converting the Time Array

If the observations happen to be split across multiple days then the day and millisecond offset can become hard to work with. For this reason it is sensible to convert all of our times into a consistent, single monotonic format. The recommendation here would be milliseconds since 1979-1-1 although other forms could be used. When this form is stored as a 64-bit unsigned integer it will take almost 585 million years to overflow.

3.4 Isolating the Valid ”time-slices”

The next stage is to isolate the values in the flux array where the values for all frequencies are valid. This step is not strictly necessary but it makes some parts easier. If, however, it is not the majority of the times where most points are valid then looking at other options would be sensible. In the case of this data the vast majority of the times have valid fluxes associated with all values.

A ”mask” of the valid array indices can be created by bit-wise ”and-ing” together the columns of ‘mvd‘ and then retaining only the values where this mask is 1. This same mask must also be applied to the time array, to avoid the two desynchronising.

3.5 Subtracting the Quiet Sun

As the peak emission of large solar events is relatively short but powerful it doesn’t affect the mean of the data too much. A simple heuristic for finding the mean quiet sun counts for each frequency is to take the mean of all points that are less than the mean emission over the period.

Question 3.1. More advanced techniques can be used. How would you go about calculating the background/quiet sun flux from this data?

Once a quiet sun flux has been calculated from each frequency it must be subtracted from each value to produce an array of the emission due to the flare.

3.6 Plotting the data

The Nobeyama data can now be plotted. It is interesting to plot a section around the peak of the flare. According to Nobeyama data, the peak emission occurs at 01:59:38UT (i.e. 358s after the start of the dataset).

Question 3.2. Average each flux in a ± 30s window about this point and then plot the averaged flux against frequency on a log-log plot. Does this shape match what you would expect?

3.7 Loading the RSTN Data

Luckily the 7 frequencies of NoRP data is not all the available data. Additional data can be added from the RSTN stations in Learmonth (Australia) and Palehua (Hawaii).

Question 3.3. Why can RSTN data be added only from these two observatories and not the observatories in USA and Italy?

The RSTN data is provided in a very similar format to the NoRP data. There are two different folders for the two different observatories. These folders are named ‘apl131028‘ and ‘phf131027‘, where ‘apl‘ stands for the Learmonth data and ‘phf‘ for the Palehua data. In each folder there is a ‘flux.csv‘ containing an array of flux vs frequency in a similar (but rotated) sense to the ‘fi.csv‘ in NoRP data. There is also ‘freq.csv‘ and ‘tim.csv‘ which are in exactly the same format as their NoRP counterparts.

For both observatories all the provided points are valid, so the ’Isolating the Valid ”time-slices”’ Section need not be repeated. The quiet sun needs subtracting in the same way as the NoRP data, and then all data sets should be plotted on the same graph. The three data sets should appear fairly continuous.

3.8 Fitting the Gyrosynchrotron Emission

Now that you have a complete set of background-subtracted data it is time to use the equation you found earlier to fit the gyro emission. Fit the data using your package’s non-linear least squares fitting function (e.g. ‘curve_fit‘ for ‘scipy‘).

Question 3.4. What is the χ^2 of this fit, and the values of the high and low-frequency slopes?

3.9 Fitting the Plasma Emission

Plasma emission can have an effect on the quality of the fit to the gyro equation. At low frequencies ($<$ 2GHz) the plasma emission appears as a power law. Fit this power law and then use your fit to subtract this effect from the flux data.

4 Conclusions

Question 4.1. Now, re-fit the gyro component of these adjusted fluxes and compare the slopes and χ^2 to your previous values. Is there any change/improvement? Is the plasma emission significant? What is the frequency of maximum emission of your gyro fit? How would this vary if the plasma parameters (densities, magnetic fields, etc.) vary?

A Simple Radiative Transfer Model

NB You are expected to answer all questions in this lab explicitly in your report. You are also required to research information from scientific literature to apply in your analysis, citing all references used.

1 Introduction

In this lab you will learn to implement a simple method of numerically calculating the several radiative transfer cases for a simple homogeneous spherical gas cloud model.

First we will investigate a technique, raytracing, which can be used to analyse such problems in three-dimensional in a relatively efficient computational manner. Then we will modify this previous section to investigate the transfer of light through our spherical cloud.

2 Raytracing a Sphere

2.1 Introduction and description

In this section we will describe and implement the basic design of an *orthographic* raytracer. Raytracing is the oldest form of three-dimensional computer graphics, first investigated as early as 1968. It consists of ”firing” a number of rays from a location to serve as each pixel on the screen and seeing what each ray intersects with. In orthographic raytracing all of these rays are parallel, and rather than behaving like a small CCD which can take in a large field of view by virtue of a lens with a certain focal length, a large grid, the same size as the desired field of view is used. This grid is a grid of pixels and a ray is fired directly forward from each pixel, hence all of these rays are parallel, giving rise to the term orthographic.

In the simplest form of raytracing, a ray stops when it hits a surface and the colour/property of the surface hit (modulo lighting) determines the colour of the pixel. However this tells us very little about the cloud the light is passing through. Instead we are going to use a technique referred to as depth, or volumetric path tracing. When applying this technique, the ray traverses every object in the ”scene” and a sorted list of all of these intersections is produced. This list is used to determine the distance the path traverses through each homogenous object, and then the emission and absorption properties of the objects can be used to determine the effect on the light passing through them (the sorted list is used from furthest intersection to closest to apply this in the correct order).

From scratch you will code the simplest form of volumetric path tracer possible, a path tracer that traces the depth of a sphere traversed. The sphere is the easiest shape due to its simple mathematical description. Shapes like cuboids are not especially difficult to implement either, but an optimised version of these is a more interesting challenge. Shapes that can be described by any polynomial dependence up to order 4 are fairly readily traceable using analytic equations, above this it becomes necessary to resort to root finding techniques to solve the equations (this is also true of shapes not defined by polynomials). In many cases it would be easier to produce a triangle-based mesh and write a solver for this mesh based on its triangular faces (it is trivial to work out whether a ray is entering or leaving a mesh from the direction of the winding of the triangle traversed).

Question 2.1. As an exercise, solve the geometric problem of the intersection of a ray and sphere by hand. Derive the quadratic equation describing this problem and the three different cases described by the sign of the discriminant. How can the distance traversed by the ray inside the sphere be found?

2.2 Implementation of algorithm

Start by setting up a grid of "pixels" from which the rays will be fired, this is the ‘image‘ for this system. Now some means of describing spheres and rays is required.

If working in an object-oriented language such as Python, it is customary to describe items like rays as *objects* in which there is a Ray class, describing how rays are defined. A ray has both an origin and a direction, meaning that it needs to contain two three-dimensional vectors.

The same should be done for the sphere as it has a centre defined by a vector and a radius, defined by a number. The method that calculates its intersection with the ray can also be added to the class definition.

Here the ‘intersects‘ method computes the intersection points of a ray with the sphere and returns these as a list, using the method you have previously derived. If the determinant is negative or zero then an empty list is returned, otherwise a list of the two roots is returned.

Now, if you project a ray perpendicularly to the grid from each pixel and calculate the intersections with an instance of a sphere, you can trace and plot an image of the depth traversed by each ray.

Congratulations, you have built a volumetric ray tracer that can trace a single sphere! It can be extended to more complex/multiple shapes fairly easily, although multiple shapes becomes more complex with respect to sorting shown in the next section.

3 Radiative Transfer

The next step is to add radiative transfer to this model. This has been discussed in depth in lectures as it is a key component of modern astronomy. In lectures a set of six schematic line profiles from different cases of τ and S_ν for a homogeneous object are given. Your task is to reproduce these line profiles by modifying your simulation parameters.

First an array of frequencies over which to calculate the properties of the cloud is needed. In the example code 16 are used for easy plotting of images in a 4x4 grid. A uniform background flux, emission and absorption coefficients for the cloud should also be defined. The emission and absorption coefficients are set to be multiples of each other, to make the source function constant with frequency. You can choose a profile for these coefficients, for example a Lorentzian or a Gaussian which both model realistic shapes for spectral lines.

Since there is now 16 frequencies to be evaluated, the ‘image‘ grid must be replaced by an array of ‘images‘, one for each frequency. Now, all that is required to observe the effects of the emission and absorption is to modify the inner body of the tracing loop to observe the radiative transfer equation for homogeneous sources (not forgetting that $\tau_\nu = \int_0^{\text{dist}} \alpha_\nu ds$).

Question 3.1. Plot the images for each frequency on a 4x4 grid. Do any of the line profiles appear to match what you would expect?

It is important to also verify the shapes of the line profiles on graphs like the schematic graphs in the notes. The simplest thing to do here is to take the flux at the central pixel of the image and plot this against frequency. Since knowledge of τ is required for these graphs it can be recalculated for this pixel (if you were fancy you could store it from earlier, but since it's a fairly trivial calculation, it's not overly important).

Question 3.2. Find six different sets of parameters required to reproduce the schematic plots in the notes and verify these by looking at the images at different freqencies and the plots.

4 More General Models (Theoretical)

At this point you should have successfully modelled the various line profiles for this homogeneous object.

Question 4.1. This method can generalise to more complex scenarios. Discuss the possible pitfalls with multiple overlapping objects with different parameters. In which class of problem would this technique become useful? (Consider combining with a phenomenon that has an analytic or numerical expression for the emission and absorption at 1-dimensional points in a non-homogeneous object).

Simulating Alfvén Waves

NB You are expected to answer all questions in this lab explicitly in your report. You are also required to research information from scientific literature to apply in your analysis, citing all references used.

1 Introduction

In this lab you will use the ideal magnetohydrodynamic (MHD) equations to simulate the propagation of Alfvén waves through a plasma. These waves are a travelling oscillation of ions in the plasma for which the magnetic field serves as the restoring force. Alfvén waves are named in honour of Hannes Alfvén who earned the Nobel Prize in 1970 for his "fundamental work and discoveries in magnetohydrodynamics with fruitful applications in different parts of plasma physics". As the simplest form of plasma wave to simulate, Alfvén waves provide a good introduction to the numerical techniques applied in MHD simulations.

2 Theory

Let us embark from the equations of ideal magnetohydrodynamics:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) = 0 \tag{1}$$

$$\rho \frac{\partial \vec{u}}{\partial t} + \rho (\vec{u} \cdot \nabla) \vec{u} = -\nabla p + \vec{j} \times \vec{B} \tag{2}$$

$$\frac{\partial p}{\partial t} + (\vec{u} \cdot \nabla) p = -\gamma p \nabla \cdot \vec{u} \tag{3}$$

$$\frac{\partial \vec{B}}{\partial t} = \nabla \times \left(\vec{u} \times \vec{B} \right) , \tag{4}$$

where

$$\vec{j} = \frac{1}{\mu_0} \left(\nabla \times \vec{B} \right) . \tag{5}$$

Matters can be further simplified by assuming that there is a constant field along the z-axis such that $\vec{B} = B_0 (0, 0, 1)^T$. The flow can also be assumed to be incompressible which gives a solenoidal velocity field and via (1) implies a constant $\rho = \rho_0$.

Now, the system of equations can be linearized by substituting (5) into (2) and applying the previous simplifications to give

$$\rho_0 \frac{\partial \vec{u}}{\partial t} = \frac{1}{\mu_0} \nabla \times \vec{B}_1 \times \vec{B}_0 \tag{6}$$

Here $\vec{B}_1$ is such that

$$\frac{\partial \vec{B}_1}{\partial t} = \nabla \times \left(\vec{u} \times \vec{B}_0\right) \tag{7}$$

Assume (without loss of generality) both $\vec{u}$ and $\vec{B}_1$ as oriented along the y-axis then

$$\vec{u} = (0, u_y(z,t), 0)^T \tag{8}$$
$$\vec{B}_1 = (0, B_y(z,t), 0)^T \tag{9}$$

Question 2.1. Using the previous equations show equations (10) and (11).

$$\rho_0 \frac{\partial u_y}{\partial t} = \frac{B_0}{\mu_0} \frac{\partial B_y}{\partial z} \tag{10}$$
$$\frac{\partial B_y}{\partial t} = B_0 \frac{\partial u_y}{\partial z} \tag{11}$$

These can be combined to give a form of the 1-dimensional wave equation

$$\frac{\partial^2 u_y}{\partial t^2} = c_A^2 \frac{\partial^2 u_y}{\partial z^2} \tag{12}$$

Question 2.2. Give an expression for c_A in terms of the other constants; what does this value represent?

The general solution to this wave-equation is a pair of counter-propagating waves i.e.

$$u_y = F(z - c_A t) + G(z + c_A t) \tag{13}$$

For some functions F and G. Now, take equations (10) and (11) and apply the following change of variables

$$u = \sqrt{\rho_0} u_y$$
$$b = \frac{B_y}{\sqrt{\mu_0}}$$

Equations (10) and (11) then become

$$\frac{\partial u}{\partial t} = c_A \frac{\partial b}{\partial z} \tag{14}$$
$$\frac{\partial b}{\partial t} = c_A \frac{\partial u}{\partial z} \tag{15}$$

This is a pair of symmetric wave equations, which together form a hyperbolic partial differential equation. In this case the problem could be solved analytically from here, however this becomes tiresome when coefficients and similar change. Using computers a result can be quickly obtained for many cases as well as being able to numerically solve analytically impossible problems. The method for solving these equations will be discussed in the next Section.

3 Simulation

3.1 The Numerical Scheme

The most common technique for numerically solving partial differential equations is the finite difference scheme which behaves like the derivative from first principles but using a finite step size, rather than the infinitesimal limit. In the case of hyperbolic equations such as these it can be difficult to find accurate numeric solutions since simple first order methods such as the upwind scheme will cause dispersive errors. The scheme that will be used in this lab is the MacCormack scheme, which is still a simple scheme but has better stability and much lower error than many other methods. The MacCormack scheme works by using a prediction step and then correcting this step afterwards. In this case we let the subscript represent the position of the value, and the superscript represents the time index observed. Then the predictor step is given by

$$u_i^p = u_i^n - c\left(b_{i+1}^n - b_i^n\right) \tag{16}$$
$$b_i^p = b_i^n - c\left(u_{i+1}^n - u_i^n\right) \tag{17}$$

In this section $c = c_A \dfrac{\Delta t}{\Delta z}$ with Δt as the the timestep and Δz as the step between two grid points. The superscript p indicates the predictor step. Now the correction step is applied to provide the true value at the next timestep.

$$u_i^{n+1} = \frac{1}{2}\left[u_i^n + u_i^p - c\left(b_i^p - b_{i-1}^p\right)\right] \tag{18}$$
$$b_i^{n+1} = \frac{1}{2}\left[b_i^n + b_i^p - c\left(u_i^p - u_{i-1}^p\right)\right] \tag{19}$$

It is useful to quantify in what situations this scheme is stable. A technique called von Neumann stability analysis exists for this purpose. To begin with assume that the system is stable in space and time then the eigenmodes of the equations are of the form $u_j^n = \xi^n e^{ikj\Delta z}$ where k is a real spatial wave number, ξ is complex and i is the complex unit. The system is unstable for $|\xi| > 1$. von Neumann stability analysis is not an incredibly rigorous technique but usually works in practice (as it originates from the Fourier decomposition of the system) and it is fairly quick to calculate and apply. It is also useful for applying adaptive step-sizing in simulations where the velocity varies across the domain.

Question 3.1. Using the expressions for the two steps and the eigenmodes, find, in terms of c, the condition for which the system is stable.

3.2 Coding a Solution

Now it is time to code a solution using the method described above. To do this some additional constraints are required in the form of initial and boundary conditions. The aim is to examine the behaviour of the system when a wave is driven from the left-hand side and reflects off the right hand side. Let the driving force be $u(0,t) = \sin(2\pi t)$ for all t. Then at $t = 0$ we have $u(z,0) = 0$, for all z. The initial conditions for b can also be chosen to be $b(z,0) = 0$.

The first boundary condition is set by the driving force as $u(0,t) = \sin(2\pi t)$ for all t. This is connected to a Neumann condition on b by (14): $\frac{\partial b}{\partial z} = \dfrac{2\pi}{c_A}\cos(2\pi t)$. At the other end of the system a reflection is imposed so $u(z_{\text{end}}, t) = 0$ for all t.

As described previously, this system will be solved on a domain split into discrete points. First split the domain $[0, z_{\text{end}}]$ into $0, \ldots, N$ points. Applying the previous conditions we immediately see $u_N^p = 0$ and $b_N^p = b_{N-1}^p$ for all timesteps. As the predictor depends on point $n+1$ these two end points have to be handled separately. Every other predictor point can be handled by applying (16) and (17) for $i \in [0,\ N-1]$.

For the corrector step a similar problem arises at the opposite end of the domain: the points at $z = 0$ cannot be calculated without the -1 step. All other points for $i \in [1,\ N]$ can be immediately calculated. The boundary conditions make u_0 easy for all timesteps. b_0 is more challenging, however, as it requires the point u_{-1}^p.

Question 3.2. An expression for the second derivative: $\dfrac{\partial^2 u}{\partial t^2}(0,t)$ can be found from the boundary conditions. Combine this with (12) to obtain an expression for the second spatial derivative of u at $z = 0$.

Given that the second order central finite difference at u_n is given by $\dfrac{\partial^2 u(z,t)}{\partial z^2} = \dfrac{u_{i-1}^n - u_i^n + u_{i+1}^n}{\Delta z^2}$, combine this expression with the expression obtained for the second spatial derivative and then rearrange to find an expression for the required ghost point u_{-1}^p

Question 3.3. Using the previous conditions and ghost point, code a simulation that iterates through time plotting $u(z)$ and $b(z)$ for each timestep. It may be interesting to produce a movie of this simulation. Set up your parameters so that $c = 1$, with $c_A = 10$ and $\Delta t < \Delta z$. Describe the time evolution of the system (make sure that you allow it to run for long enough to reach a steady state). What happens when you increase and decrease c by modifying its component parameters?

4 Conclusions

Question 4.1. Starting from the results of your simulation discuss the accuracy of these results and the applicability of this type of solution to wider physics and plasma physics problems. Briefly describe how you would generalise this wave problem to higher dimensions and any changes or improvements you would make.