A Fusion Proton Diagnostic for Low Field Tokamaks

Alexander Lampson

Submitted for the degree of Master of Philosophy

UNIVERSITY OF YORK
PHYSICS
SEPTEMBER 2012

Abstract

The use of a pinhole-type detector to image the high-energy protons produced by interactions of fast particles within the plasma of low field tokamaks such as the Mega Amp Spherical Tokamak is explored. The minimum number of detector elements in a pinhole-type detector needed to produce accurate images of the plasma was found to be 144, giving a 5cm by 5cm resolution of the plasma in the R, Z plane. The image is recreated via a combination of singular value decomposition and maximum entropy methods. The proton production rate is a function of the spatial distribution of fast particles, and so images of the proton production distribution can be used to directly diagnose the fast particle distribution within the plasma. These fast particles are mostly produced by neutral beam injection, and so the methods presented here can be used to more accurately control the neutral beam injection energy to ensure that the heating and current drive is applied to the desired regions of the plasma.

As the orbits of the fusion born protons are influenced by the magnetic field within the plasma, a transformation matrix is created to link the detector image with the intial proton distribution within the plasma. To do this, large numbers of test particles are needed. The calculation of the orbits of the test particles is performed using CUDA GPUs which can calculate the orbits of up to 100,000 particles per second. This takes advantage of the embarrassingly parallel nature of the problem, which is ideally suited to calculation on the GPU.

Contents

List of Figures

List of Tables

Acknowledgements

"I should have been a plumber." - Albert Einstein

I would like to thank my parents, for their unceasing and unfailing support. My siblings have also helped me to the best of their ability, all throughout my life. The assistance of my family has been constant, and I would like to thank them all for their efforts to make me a better person.

The physics department at York, and now the York Plasma Institute, have worked tirelessly to ensure that all the support needed was provided. I would like to thank Dr. Vann for all his hard work in getting me as far as I have, as well as all the other members of the Plasma Group. A special mention must be made of Dr. Dudson, who always has time for a chat, and can shed light on any problem. My co-workers in office C10, and now in the YPI, Dr. Myers, Dr. Buxton, and Dr. (to be) Dickinson have provided a fine example of how it should be done. Thank you for your company over the last 5 years. Thank you as well to all the other members of the PhD office in the YPI, for helping to keep me sane in difficult moments.

York has been a marvellous city, with many things to recommend it. I would like to thank Browns, for feeding me so well over the last 4 years, and Hercules the tiny horse, for bringing a little joy into my life with his simple existence. Finally I would like to thank the east coast main line, for managing to get me to London semi-reliably, during those times when York couldn't provide what was needed.

Last, but certainly not least, I would like to thank Dr. M. Antonik, for her love, support, and kindness over the last four years. May our future be as good as the past has been.

Declaration

I declare that this thesis is the result of my own research except where acknowledged in the text. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature	:	
Name	:	
Date	:	

Chapter 1

Introduction

1.1 Energy Production and Consumption

Global estimated energy usage in 2011 was approximately 54 PW hrs, of which approximately 87% was provided by fossil fuels [8]. This includes energy used for heating, cooking, electricity, and transport. At current rates of consumption, the proven reserves of oil will last for 54.2 years; the proven reserves of natural gas for 63.6 years; and proven reserves of coal for 112 years. By considering only proven reserves, the use of new technologies, such as fracking to produce shale gas, is discounted. These are therefore very pessimistic predictions. World energy consumption is increasing, and although the proportion of energy provided by fossil fuels is decreasing, as can be seen in Fig. (1.1), the total usage of fossil fuels is increasing. Unless more efficient fission power stations are put into operation, the supply of uranium will also be depleted in 83 years [22]. The remaining supply of fossil fuels, and uranium, is summarised in Table (1.1).

The increasing demand for energy, and the limited supply of traditional fuels, leads to a serious problem. If we are to maintain the level of energy useage at 2011 levels, and assuming that all energy requirements can be met by electricity, we require a worldwide generating capacity of approximately 6 TW, which must be independent of fossil fuel production within 100 years. If we are to replace this capacity evenly over 200 years, this

Fuel	Proven Reserves	Consumption	Time till Depletion [Years]
Oil	234000 million tonnes	4100 million tonnes in 2011	57
Gas	208 trillion cubic metres	3.2 trillion cubic metres in 2011	65
Uranium	5327 thousand tonnes	64 thousand tonnes in 2010	83
Coal	860 trillion tonnes	7.7 trillion tonnes in 2011	112

Table 1.1: The reserves, and consumption for fuels providing over 90% of the worlds energy supply [8]. The reserves listed are dependent on the cost of extraction: as the value of the fuel increases, previously uneconomic reserves can be utilised.

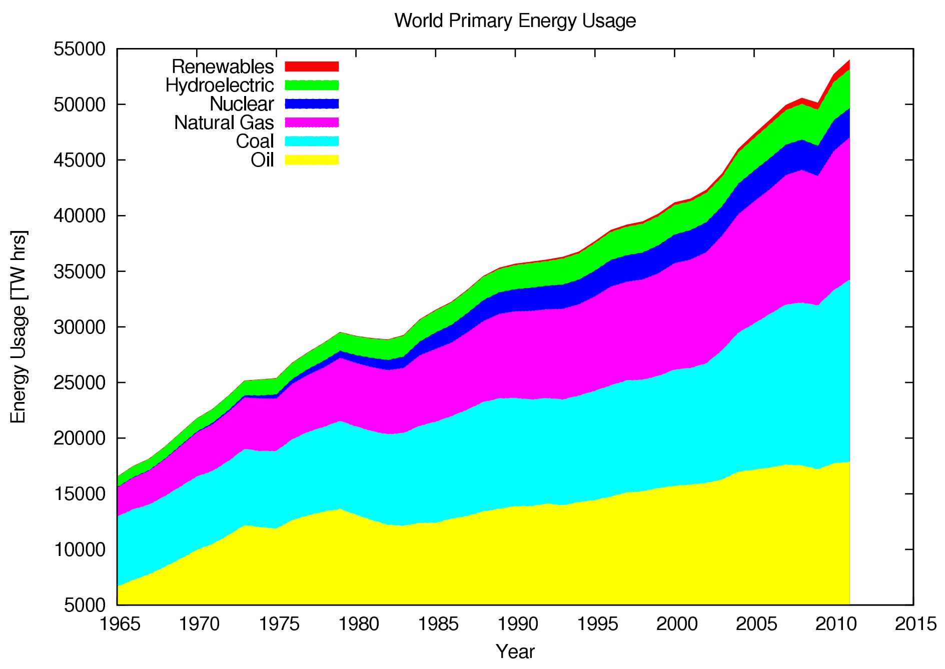


Figure 1.1: The approximate total world energy useage, by fuel type.

requires the construction of 30 GW of fossil fuel free generating capacity per year. This assumes that the fossil fuel consumption requirements will drop as the new generating capacity is added, giving approximately 200 years till depletion of fossil fuel reserves.

1.1.1 Available technological solutions in Britain

Given the increasing global energy demand shown in Fig. (1.1), and limits to supply of traditional fuels shown in Table (1.1), how can Britain, and the rest of the world, ensure continuity of energy supply? In addition to these constraints, we should consider the potential impact of solutions on climate change, and the security of supply. There are five main groups of existing renewable electricity generating technologies, being wind, wave and tidal, hydroelectric, biomass, and solar. Each of these has particular advantages and disadvantages on local and global scales. Generating schemes for each technology are divided into large scale schemes, being those with a capacity greater than 5MW, and small scale schemes.

Electricity generation and transmission must match demand exactly. If power cuts due to lack of generating capacity are to be avoided, the historic rule of thumb is that generating capacity must exceed peak demand by 20% [37]. This applied to situations with relatively small numbers of large independent generating stations. Fossil fuel and nuclear stations have approximately 5% chance of being unable to generate at times of peak capacity, and are largely uncorrelated. Renewable generating schemes have greater intermittency than traditional power stations. This means that a much larger installed generating capacity is needed to ensure continuity of supply. This is especially true if a reduction in generating

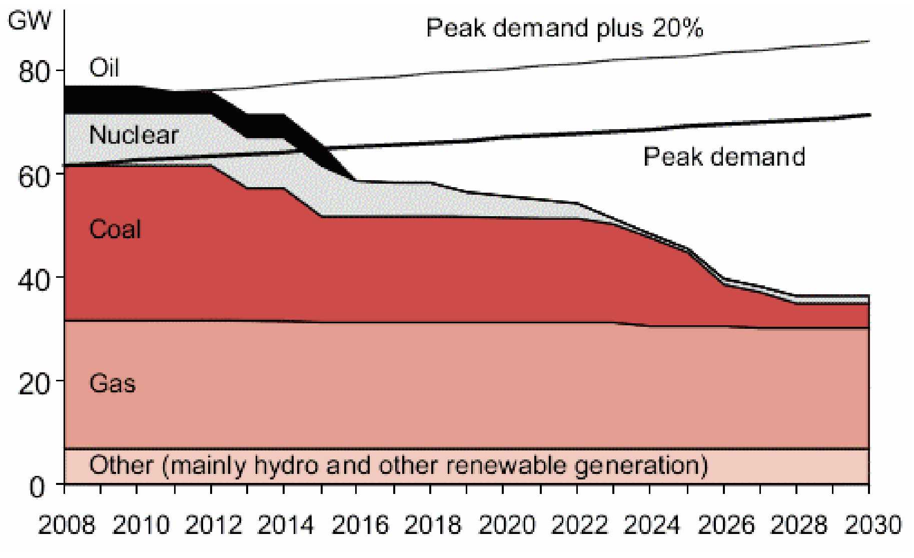


Figure 1.2: Predicted electricity demand and generation capacity, showing forecast closures due to the age of nuclear power stations and the impact of the European Large Combustion Plant directive [37].

capacity in one or more of the schemes is correlated, for example days with low wind speeds can correlate with a reduction in the amount of rain, decreasing the effectiveness of both wind and hydro generating schemes. Britain must replace ageing nuclear and coal power stations to maintain supply, as shown in Fig. (1.2).

Wind

Generation of electricity from both onshore and offshore turbines suffers from intermittency. The load factor is a measure of how much electricity will be produced by a given generator. For the UK, the 2011 load factor for offshore wind turbines was 36%, and for onshore wind turbines was 27% [35]. The load factor varies from year to year, from a minimum of 24% in 2003 to a maximum of 36% in 2011 for offshore wind, and a minimum of 22% in 2010 to a maximum of 28% in 2002 for onshore wind. These load factors are for locations with strong steady winds. For areas with low, or variable wind speeds, such as urban areas, load factors are likely to be as low as 1 or 2%. The UK currently has 4.6GW of installed onshore wind capacity, and 2.35 GW of offshore wind capacity. The variability of wind power is a concern for the UK, as wind speeds can be correlated over length scales greater than the total area of the UK [38]. This means that there must be additional generating capacity to cover at least the base-load power requirements, independent of the weather conditions. This additional capacity can come from energy storage schemes, such as pumped water or compressed air schemes, but at considerable additional cost. There is evidence that the wind speeds are positively correlated with the peak energy demands [43], but not strongly enough to be reliable. Wind generation schemes, with their inherent short-term variability in output, place large stresses on electricity transmission networks such as the National Grid, and require considerable extra generating capacity. This problem is magnified as a greater proportion of the total electricity generating capacity consists of wind power, making wind unsuitable for more than, at most, 20% of total electricity mix [40].

Wave and Tide

There are two types of tidal generators. Tidal barrages trap water at high tide, and release it at low tide to power turbines, and tidal stream devices use fast-flowing tidal streams to power turbines. For wave power, three schemes are in use. The first utilises an oscillating water column in a hollow cylinder to drive changes in air pressure, the second uses a system of articulated cylinders which move with the waves, again compressing air. The third absorbs energy from the waves by using the movement of a floating platform to induce rotation in gyroscopes.

There are currently no wave generating schemes in the UK operating to supply the national grid. Although there is considerable potential wave power [11], this energy generating scheme suffers from similar intermittency problems to wind power, and the size of waves, and power available, is strongly correlated with the wind speed. This makes wave power suitable only for a small portion of the total UK electricity mix.

Tidal power has smaller potential capacity, being reliant on particular geographic conditions. There are several potential sites, such as the Severn estuary, which could produce up to 1.8GW of electricity [34]. Tidal generators suffer from intermittency problems associated with high and low tides, as well as changes in efficiency at spring and neap tides. These are very predictable, lessening the impact on the National Grid. However, the expense and large environmental impacts, combined with the inherent intermittency of tidal movements, mean that there are currently no plans to construct large scale energy generating tidal barrages.

Hydroelectric

Hydroelectric schemes have been in operation for many years, and are a mature technology. However, large-scale schemes all consist of dams and reservoirs in mountainous regions, and there are very few suitable locations that have not already been utilised in the UK. Smaller schemes are generally river based, and suffer similar but more predictable intermittency to wind generators. The lack of suitable locations for hydroelectric power also apply to using pumped storage of water as a method of generator smoothing, as suitable locations for this are also lacking [36].

Biomass

There are three sources of biomass in general consideration in the UK. The first is wood from sustainable forestry practices, which may be felled directly or consist of excess from sawmills. This is usually burnt to provide small scale heating and electricity generation. Specially grown energy crops, such as sugar beat or oilseed rape, form the second source of biomass. These are generally used to create transport bio fuels; bio ethanol for sugar crops, or bio diesel for oil crops. This is relatively inefficient, yielding approximately four tonnes of useful biomass per hectare per year for sugar beet, and one tonne per hectare per year for oilseed rape [50]. Finally, gas produced by the decomposition of organic materials is currently the largest source of biomass generation in the UK [35]. Biomass generation is not intermittent, and so can be easily utilised by the National Grid. However, most sources of biomass, such as gas from landfill sites, is currently being utilised, and the use of farmland to grow biomass crops for electricity generation may be unfavourable if the resultant increase in food imports requires larger energy expenditure than is gained from the grown biomass.

Solar

Solar generation, through solar photovoltaics or solar furnaces, is increasing worldwide. However, the UK is unsuited for solar generation, as peak demand is during the winter months, and the amount of cloud cover reduces the load factor of solar installations. The UK has approximately 1GW of installed PV capacity [35], producing 252 GW hrs per annum. This gives a load factor of approximately 3%. This may underestimate the load factor, as not all of the installed capacity has been operational for the entire year. Typical load factors for solar PV in the UK are around 10%. In addition to the low load factor, solar energy in the UK is very intermittent, and therefore unsuitable as more than a few percent of the total electricity mix.

The chief problem with the renewable electricity generating techniques that are currently in use is intermittency. Although this can be ameliorated through smart grids, and a large over-provision of electricity generating capacity, an ideal solution would need to have low intermittency, low carbon emissions, and a low environmental impact. In addition, in the UK, the security of supply is a potential issue. Already, as local production of fossil fuels decreases whilst demand increases, the UK is increasingly dependent on imported gas and coal. To solve this problem, within the constraints, the UK must turn to as yet immature technologies. Chiefly, nuclear power, both fusion and fission, has the potential to have low intermittency and low carbon emissions, whilst a cost-effective energy storage technology would greatly reduce the problem of intermittency.

Fission

Considering nuclear fission, whilst current uranium reserves will be depleted in approximately 80 years at current rates of consumption, current fission power stations operate a ”once-through” fuelling system. Fast breeder reactor designs can significantly improve the utilisation of the uranium, whilst reducing the radioactivity of the waste products. The waste material radioactivity is mostly due to relatively short-lived actinide species. Fast breeder reactors can be split into two types: fast neutron reactors use the neutrons produced in one reaction to induce fission in other actinides, whilst thorium fuel cycle reactors follow a decay chain with many fewer heavy actinides. However, very few fast breeder reactors have been built, and most designs are untested.

The main alternative under consideration is nuclear fusion. This has the potential to provide non-polluting, waste free, and dependable electricity generation whilst in operation. It can ensure security of supply, as fuel can be gathered direct from seawater, and it is not intermittent. It is a technology that is still under development, and will require considerable time and research funding to bring it to maturity.

1.2 Fusion

Fusion describes the process by which light elements combine into heavier elements. This requires that the nuclei of the elements can get into such proximity that the strong nuclear force, which attracts nucleons such as protons, is greater than the electrostatic force, which will repels similarly charged particles. To achieve this proximity in the nuclei of the elements undergoing fusion requires considerable energy input to the nuclei. The energies involved are sufficient to cause the fuel to enter a plasma state.

Plasmas are a form of high energy gas, in which the electrons have dissociated from the ions. This makes a plasma a quasi-neutral gas comprised of charged particles, with approximately equal numbers of electrons and protons. The most effective way of combining bulk quantities of hydrogen is by heating the hydrogen gas until it becomes a plasma.

To create a fusion plasma the plasma, usually deuterium (D) and tritium (T), needs to be heated, and kept hot, for long enough to undergo fusion reactions. There are several schemes for heating and confining the plasma, chiefly magnetic confinement, usually in tokamaks, and inertial confinement, in which lasers are used to heat the plasma, which is confined by the inertia of the particles.

As can be seen from Fig. 2.1 the temperatures needed to produce fusion reactions at useful rates are high. In most research reactors deuterium only plasmas are used. This is due to the similarity of the physics of deuterium only plasmas to deuterium-tritium plasmas, without the higher levels of radioactivity of the D-T plasma, and the expense of acquiring the tritium.

The main reaction under consideration for the fusion plasmas is

$$ {}^{3}_{1}\mathrm{T} + {}^{2}_{1}\mathrm{D} \rightarrow {}^{4}_{2}\mathrm{He} + n \tag{1.1} $$

The likelihood of this reaction peaks at a lower temperature than other hydrogen reactions, so fusion occurs more readily at lower temperatures. This is less of an engineering challenge than trying to sustain higher temperatures.

1.2.1 Fusion Triple Product

The confinement time, τ_E, is the time taken for a plasma to lose energy to the surroundings.

$$ \tau_E = \frac{W}{P_{\text{loss}}} \tag{1.2} $$

where W is the energy content, and P_{loss} is the rate of energy loss. In a fusion reactor, the plasma must be always be above a critical temperature. The plasma must therefore be heated at the same rate that energy is lost for steady state operation. This heating may be from fusion reactions within the plasma, or from external heating.

For D-T reactions, assuming a plasma consisting solely of equal numbers of deuterium and tritium ions the rate of fusion reactions per unit volume per unit time is given by

$$ f = n_D n_T < \sigma v > = \frac{1}{4} n_e^2 < \sigma v > \tag{1.3} $$

where v is the relative velocity of the parent particles, and $< \sigma v >$, the reaction cross-section at a velocity v, is averaged over the Maxwellian velocity distribution at temperature T. This lets us find the rate of fusion heating, assuming that all the fusion produced neutrons are lost, whilst all the charged products are retained. The energy of the α-particles, $E_{ch} = 3.5$ MeV. If we require that all the heating comes from the charged fusion products, we can find the minimum rate of fusion reactions required to sustain the temperature of the plasma

$$ \begin{aligned} f E_{ch} &\geq P_{\text{loss}} \\ \frac{1}{4} n_e^2 < \sigma v > E_{ch} &\geq \frac{3 n_e k_B T}{\tau_e} \\ n_e \tau_e &\geq \frac{12}{E_{ch}} \frac{k_B T}{< \sigma v >} \end{aligned} \tag{1.4} $$

If we assume that the plasma has a maximum attainable pressure, we find that the fusion power density, n_{FP} is given by

$$n_{\text{FP}} \propto \frac{p^2 < \sigma v >}{T^2} \tag{1.5}$$

This tells us that the maximum fusion power for any given device is obtained when $< \sigma v > /T^2$ is a maximum. If we now combine Eqns. (1.4) and (1.5), we can find the fusion triple product

$$n_e T \tau_E \geq \frac{12 k_B}{E_{ch}} \frac{T^2}{< \sigma v >} \tag{1.6}$$

This gives a simple characterisation of a fusion plasma for a range of devices, allowing comparisons between plasmas produced via the various confinement schemes.

1.2.2 Plasma Devices

There are several schemes for devising fusion plasma devices. These broadly fall into four categories, being magnetic confinement, inertial confinement, electrostatic confinement, and mechanical confinement. Magnetic confinement relies upon strong magnetic fields to confine the plasma for sufficient time to allow fusion to occur. Inertial confinement uses the inertia of a rapidly created plasma to confine the plasma. Electrostatic confinement uses a series of electric grids to constrain the plasma. Mechanical confinement creates a plasma, and then rapidly mechanically compresses it to provide the required conditions for fusion. Of these techniques, magnetic confinement has given the highest plasma temperature, and ITER, a tokamak currently under construction, is expected to produce ten times as much energy as is used to create and maintain the plasma.

1.2.3 Tokamak Devices

A tokamak is a toroidal magnetic containment chamber. The word tokamak derives from the Russian acronym for toroidal magnetic confinement chamber. The toroidal geometry of these devices allows a closed magnetic field to be formed. Ideally, the plasma would be held in a sphere. This gives the lowest surface area to volume, which increases the efficiency of the device by reducing the area through which heat can be lost. However, to confine the particles of the plasma, a closed magnetic field is needed. By considering the magnetic field lines as being similar to hairs on a ball, it can shown that it is impossible to create a closed magnetic field that covers the surface of a sphere. It is possible to create closed magnetic field lines on the surface of a torus, and the increase in surface area to volume ratio is not sufficient to preclude heating the plasma to the temperature required for fusion reactions.

Tokamaks are generally classed into small aspect ratio, or spherical tokamaks, and large aspect ratio devices. The aspect ratio considered here is the ratio between the major

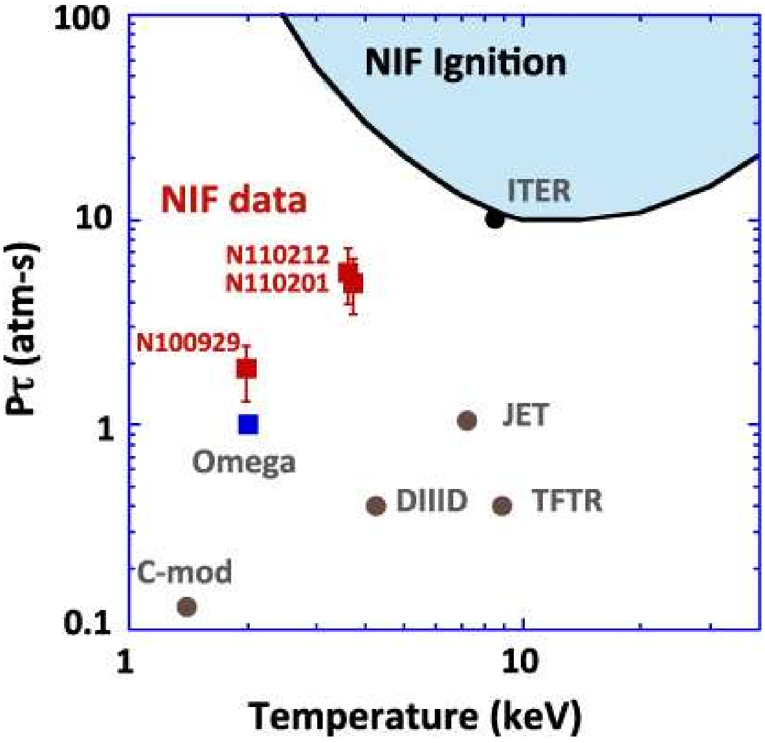


Figure 1.3: The generalised Lawson confinement parameter, averaged pressure times confinement times, versus the ion temperature for a range of tokamak and laser plasmas. The light blue region shows the criteria for ignition [15].

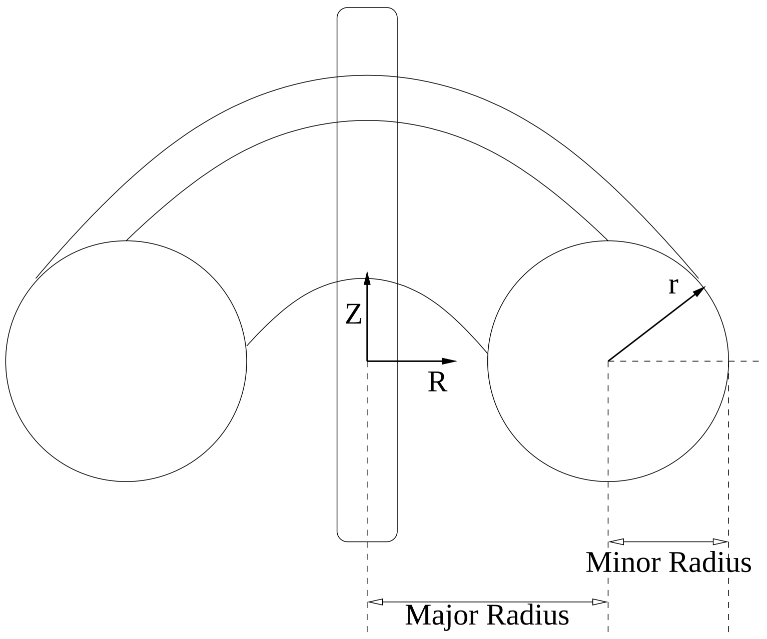


Figure 1.4: Tokamak Cross-Section

and minor radii, as shown in Fig. (1.4). The major radius is the distance from the axis through the gap to the centre of the ring of the torus, whilst the minor radius is the radius of the ring of the torus. Large aspect ratio tokamaks therefore look like inner tubes, or ring donuts, whilst small aspect ratio devices appear more similar to cored apples. The Joint European Torus (JET), and ITER are examples of large aspect ratio tokamaks with aspect ratios of approximately 3, whilst the Mega-Amp Spherical Tokamak (MAST) is a spherical tokamak with an aspect ratio of approximately 1.3. One of the advantages of spherical tokamaks is that the magnetic field needed to confine a similar volume of plasma is lower. Due to this lower magnetic field, the products of fusion reactions are not as well confined, and will generally escape within one Larmor orbit, as shown in section (1.3.3). This is undesirable, as the possibility of heating using these fast particles is lost, however it allows the detection of these particles to be used as a diagnostic.

A MAST plasma typically has major radius $R \approx 0.85$m, a minor radius $r \approx 0.65$m, plasma current $I_p < 1.3$MA, and a toroidal magnetic field $B_t = 0.3 - 0.6$T [48]. It is one of the best diagnosed tokamaks currently utilised. This is in part due to the separation of the outboard edge of the plasma and the vacuum vessel wall, which is not found in many other tokamaks. This allows diagnostic equipment to approach the plasma, and to view beyond the plasma edge.

1.2.4 Tokamak Plasma Heating

The gas within a tokamak must be heated before it will become a plasma, and that heat must be maintained sufficiently long to allow fusion events to occur. There are several schemes in use for tokamak plasma heating, chiefly radio frequency heating, ohmic heating, neutral beam injection, and compressive heating.

Compressive Heating

By utilising the “frozen-in” condition of MHD plasmas, we can move the plasma by moving the magnetic fields. In particular, by increasing the toroidal or poloidal magnetic field the major or minor radius of the plasma can be reduced. However, the potential gains from this are limited, as the plasma will diffuse across the field lines, and the amount of compression that can be achieved is limited. In addition, once the plasma has been compressed, the heating cannot be reapplied without allowing the magnetic field to relax and the plasma to cool. As such, this is not a suitable technique for heating steady state plasmas.

Ohmic Heating

A toroidal electric current is induced in the plasma. The tokamak and the plasma can now be considered as a transformer, where the plasma ring forms a one-turn secondary winding, whilst the primary winding is situated in the central column. The pulse length for a plasma discharge is limited, in part, by the maximum current that can be driven in the primary winding. The effective heating of the plasma is dependent on the resistivity of the plasma. The Ohmic power is $I_p^2\mathcal{R}$, where I_p is the plasma current, and $\mathcal{R}$ is the plasma resistivity. As the temperature of the plasma increases, the resistivity of the plasma decreases [45], limiting the effectiveness of the heating. This method of heating is commonly used to provide start-up heating, as well as providing heating throughout the life of the plasma[25]. For designs such as spherical tokamaks with very thin central columns, this can be complicated to arrange [51].

Radio Frequency Heating

High-frequency electromagnetic waves are created outside the plasma. If these waves have frequencies close to the characteristic frequencies of the plasma, they will pass energy to particles within the plasma. This is analogous to Landau damping [52]. By ensuring that the phase velocity of the wave is greater than the average velocity of the thermal particles, energy is passed from the wave to plasma [29]. Due to the difference in gyration frequencies of electrons and ions, radio frequency (RF) heating is usually categorised as ion cyclotron resonance heating or electron cyclotron resonance heating. Ion cyclotron heating will utilise resonant frequencies either at the ion cyclotron frequency, or at a harmonic, as dictated by the requirements of the plasma being heated. These frequencies range between $\sim 20 - 60$Mhz. Electron cyclotron heating requires much higher frequencies, in the order of 100GHz. By careful consideration of the frequency used, and the polarisation and direction of the radio waves, the heating can be localised to certain areas within the plasma. This exploits the fact the the energy exchange between the particles and the wave only happens when the wave phase velocity is comparable with the particle velocities. RF heating has been used in many tokamaks [52].

Neutral Beam Heating

A beam of high energy neutral hydrogen or deuterium atoms is injected into the plasma. The neutral particles will be unaffected by the magnetic field. After collisions with electrons and ions in the bulk of the plasma, the neutral particles will become ionised, and constrained by the magnetic fields. The resulting fast ions increase the temperature of the plasma. By fast ions, we mean ions with energies greater than two standard deviations above the mean thermal ion temperature, if the thermal ion temperature has a Maxwellian

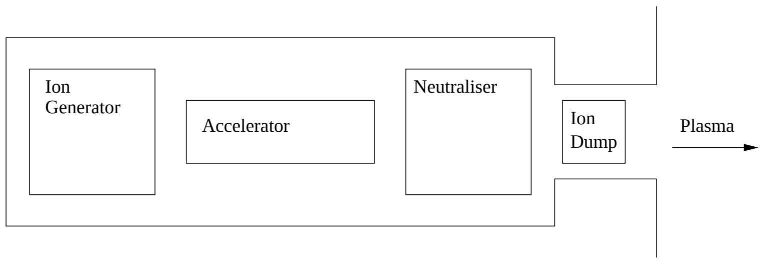


Figure 1.5: Scheme of a Neutral Beam Generator

distribution. The temperature of fast ions can be much greater than two standard deviations above the average temperature of the bulk plasma.

The neutral beam must be of sufficient energy that the particles penetrate to the core of the plasma before undergoing ionisation, but not so energetic that the beam passes completely through the plasma without collisions. This limits both the lower and upper bounds of the energy of the beam.

In order to produce neutral beams of sufficient energy, we first create hydrogen ions. These are then accelerated via a series of electric fields, and then the resultant high energy ions are neutralised before being injected into the plasma. An example of a neutral beam generator is shown in Fig. (1.5).

The injection of fast neutral hydrogen atoms into the plasma serves several purposes. The fast ions produced in the plasma will heat the plasma, and act as replacement ions for those lost to the wall of the device. By carefully tuning the energy of the fast atoms, the region of the plasma in which the atoms are ionised can be chosen. The speed of rotation of the plasma can also be influenced if the atoms are injected co-rotationally or counter-rotationally [47]. This requires several neutral beams being injected into the same plasma.

The interaction of the beam consists of four main physical processes. First, the injected atoms are ionised by collisions with the plasma. This occurs initially in interactions with the electrons, and then with the ions [16]. The three processes following the slowing down of the beam atoms are that the fast ions will drift in the magnetic field, that the fast ions will scatter from plasma ions and electrons, slowing the fast ions, and that there are charge-exchange collisions of the fast ions with background neutral atoms.

The fine-tuning of the injection energy of the neutral beam is critical for the success of this technique. If the energy of the particles is too high not all of the beam will be ionised in the plasma. This is known as ‘shine-through’. If the energy is too low, the atoms will not reach the centre of the plasma before being ionised. This gives unfavourable particle and power deposition profiles, with most of the energy deposited in the edge plasma. This reduces the efficiency of the heating of the plasma, as these fast ions will diffuse out of the plasma much faster. In addition, if the neutral beam injection is being used to provide toroidal rotation in an attempt to stabilise the plasma, or a toroidal current to maintain the quality of the poloidal field, an incorrectly adjusted beam energy may drastically reduce the efficiency and lifetime of the plasma shot.

MAST has a high power neutral beam injection system, providing 5MW, and due to be upgraded to 7.5MW at stage 1 of the MAST upgrade, and then to 12.5MW at stage 2 [3]. These beams provide super Alfvénic fast particles. These fast particles are expected to drive a number of instabilities, such as Alfvén eigenmodes (AE). It is extremely important to future reactors that fast particle driven instabilities are well understood, as a burning plasma will create a large population of fast particles in the core of the plasma. Measuring the fast particle population is therefore very important in understanding the physics of the fast ions, and the behaviour of burning plasmas.

1.3 Super-Thermal Ions

After the beam particles are ionised, the deuterium ions are still in a non-collisional regime. There is a period of "slowing-down" time during which they undergo interactions with the thermal ions. The 40keV particles injected into a typical MAST plasma will thermalise after ~ 65ms.

After the particles have slowed to a collisional regime, they will still have more energy than the thermal ions. This energy will be lost to the bulk plasma, providing the heating. The slowing down time allows the particles to spread from the ionisation site through the plasma following the magnetic field lines. The fusion interactions are therefore assumed to happen uniformly toroidally. This means that the steady state proton production profile can be modelled purely in the poloidal, or R, Z plane.

1.3.1 Collisionallity

The collision frequency of a plasma is the rate at which particles within the plasma are scattered by other particles. It is defined as being the inverse of the time taken for the particle trajectory to be changed by 90° by collisions. For fast particles in low-field tokamaks, the collision frequency is given by [52]

$$\nu_c = N\sigma_c(v)v \tag{1.7}$$

where N is the number density per unit volume of the plasma, and $\sigma_c(v)$ is the cross section of a particle with velocity v. For fast protons, the cross section for Coulomb scattering can be approximated as

$$\sigma_c(E) = \frac{5}{16}Z^2\frac{e^4}{\varepsilon^2} \tag{1.8}$$

where $\sigma_c(E)$ is the cross section in terms of energy, e is the electron charge, Z is the charge number, and ε is the particle energy in the centre of mass system. We substitute equation 1.8 into equation 1.7 to find the collision frequency. As we can see, the higher the energy of the particle, the lower the collisional frequency. If the collisional frequency is less than the transit frequency $\omega = v_T/l$ where v_T is the thermal speed, and l is the length scale under consideration then we say that the plasma is in the non-collisional regime. Fusion born protons have very high energies for particles in a tokamak plasma, and can therefore be modelled as collisionless during the time it takes them to leave the plasma.

1.3.2 Proton Energies

Fusion produced protons will have two initial energy components. The energy resulting from the fusion reaction, found in the centre of mass frame, and the energy of the centre of mass frame. The fusion energy will result in an initial velocity of the proton that is dependent on the velocity of the centre of mass frame. The initial fusion energy of a proton resulting from a D-D reaction is approximately 3MeV. If we assume that the reaction is between a thermal ion and a beam ion, the thermal ion will have an energy, in MAST, of approximately 1 keV, whilst the beam ion will have an energy of at most 40 keV, given the capability of the MAST neutral beam devices before the upgrade in 2011. This gives the centre of mass frame an energy of at most 41 keV. The velocity distribution resulting from these energies is discussed further in section 2.5. We can find the theoretical energy distribution at the detector using techniques found in [9], and compare this to our calculations.

1.3.3 Charged Particle Orbits

Charged particles feel a force given by the Lorentz equation due to a magnetic field, which, in the absence of an electric field, gives the following equation of motion:

$$m\frac{d\mathbf{v}}{dt} = Ze(\mathbf{v} \times \mathbf{B}(\mathbf{x})) \tag{1.9}$$

where m is the mass of the particle, $\mathbf{v}$ is the velocity, Ze is the charge, and $\mathbf{B}(\mathbf{x})$ is the magnetic field at position $\mathbf{x}$.

The velocity can be split into components perpendicular ($\mathbf{v}_\perp$) and parallel ($\mathbf{v}_\parallel$) to the magnetic field. If we consider the parallel component of Eq. 1.9 we find that $\mathbf{v}_\parallel$ is a constant for a constant magnetic field. Taking the perpendicular component gives

$$\frac{m}{Ze}\frac{d\mathbf{v}}{dt} = \mathbf{v}_\perp \times \mathbf{B}(\mathbf{x}) \tag{1.10}$$

Taking the scalar product of 1.10 with $\mathbf{v}_\perp$ gives $\mathbf{v}_\perp \cdot \dot{\mathbf{v}}_\perp = \mathbf{0}$, i.e. that $v_\perp^2$ is a constant.

Using 1.10 again, we find that

$$\left(\frac{m}{Ze}\right)^2 \ddot{\mathbf{v}}_\perp = \frac{m}{Ze}(\dot{\mathbf{v}}_\perp \times \mathbf{B}) = -\mathbf{B^2}\mathbf{v}_\perp \tag{1.11}$$

We can then define a quantity, ω_c, the Larmor frequency, where

$$\omega_c = \frac{ZeB}{m} \tag{1.12}$$

If we now move to a coordinate system x, y, z, where the z-axis is in the direction of the magnetic field, we can define the velocity as

$$\begin{aligned} v_x &= v_\perp \cos(\omega_c t + \phi) \\ v_y &= -v_\perp \sin(\omega_c t + \phi) \end{aligned} \tag{1.13}$$

(1.14)

We can see that the particle will therefore follow a helical path around a line known as the guiding centre. The radius of this path, being the distance from the guiding centre to the edge of the Larmor orbit, is

$$r_L = \frac{v_\perp}{\omega_C} \tag{1.15}$$

Species	Energy	Velocity (ms^{-1})	Larmor Radius (m)
Thermal Electron	1 keV	1.88×10^7	2.13×10^{-4}
Thermal Ion	1 keV	3.10×10^5	1.29×10^{-2}
Beam Ion	40 keV	1.96×10^6	8.17×10^{-2}
Fusion Proton	3.02 MeV	2.41×10^7	0.5

Table 1.2: The energy, and Larmor radius, of species of particles found in the plasma under consideration. The Larmor radius is calculated assuming a constant magnetic field of 0.5 Tesla, and that $v_\perp = v$.

1.4 Tokamak Diagnostics

There are two classes of diagnostic devices on MAST that gather data on the fast particle population. The fast particles will undergo fusion reactions, giving off charged and neutral particles, and excite plasma instabilities that will perturb the magnetic field. The two classes of diagnostic devices are therefore those that detect the particles given off, and those that detect the change in the magnetic field. The perturbations in the magnetic field can be detected outside the plasma, using either the large number of Mirnov coils, or the toroidal Alfvén eigenmodes (TAE) antenna installed on MAST. The neutral particles given off are neutrons, and can be detected, either through neutron counters, or neutron energy spectrum analysers to provide information about the fusion rate and the energy distribution of the fast particle populations. Similarly, detection of the charged particles could provide a measurement of the spatial distribution of the fast particles, and this will be the topic of this thesis. We will now provide a brief review of each technique.

1.4.1 Neutral Particle Diagnostics

There are a number of neutral particle diagnostics currently installed in MAST. These can be split into three main types. There are 5 neutron counters, consisting of two fission chambers, a ^{3}He proportional counter, an activation foil, and a BF_3 proportional counter [46]. These record the total neutron production throughout each plasma shot. There is a compact neutral particle analyser, which determines the energy distribution of the fast ions by measuring the energy distribution of the energetic neutral atoms resulting from charge exchange processes involving the fast ions [48]. Finally, there is the neutron camera, which records the neutron count along 4 collimated sight lines.

The neutron camera installed in MAST consists of four liquid scintillator detectors, each viewing the plasma through a 90cm collimator [13]. The detectors are shielded from stray neutrons by 90cm of high density pure polyethylene, and from the magnetic field by two soft iron boxes and a layer of μ-metal. The polyethylene is a source of 2.23 MeV γ rays, and so the detectors are additionally shielded with lead. The signals are sampled at 250 MHz, and pulse shaped discrimination is used to identify γ ray and neutron events. The neutron count rate along each sight line can then be recorded with a 2 ms time resolution. This is sufficiently fast to study the effects of AEs on the fast particle population.

1.4.2 Magnetic Diagnostics

MAST has a very large number of magnetic diagnostics, with more than 600 mirnov coils recording the magnetic field data. Energetic particle modes, such as AEs, can produce electromagnetic oscillations with frequencies ranging from 80-300kHz [30]. The effects of these oscillations on the magnetic field can be detected using the TAE antennas. MAST has 12 TAE antennas, positioned with six above and six below the midplane on the outboard side of the plasma, and evenly spaced around the toroidal extent of the device. These antenna have the ability to actively probe the plasma at frequencies up to 500kHz. This allows the measurement of stable AE modes [18]. The presence of AE modes can lead to transport of energetic particles, and a high resolution array of Mirnov coils is used to identify the toroidal mode number of the AE modes [21].

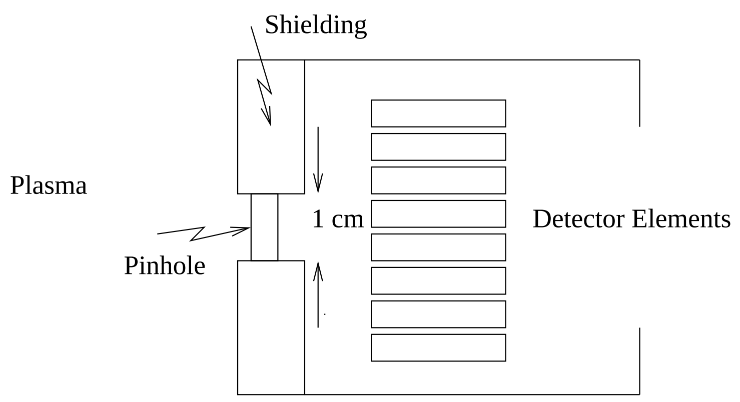


Figure 1.6: A possible schematic for a pinhole type detector. Only the probe head is shown.

1.4.3 Charged Particle Diagnostics

There are two main forms of diagnostic devices observing charged particles. The first measures the energy spectrum of the fusion products, and from this one infers the ion energy spectra of fast ions [44]. This is not dependent on modelling the orbits of the fusion products. It can also be used to measure the central ion temperature [7].

The second measures the orbit of the protons, and can, ideally, localise the initial position of the orbit. This allows the temperature gradient, and the fast particle distribution to be measured.

1.4.4 Proton Detector Design

Pinhole and Detector Plate

This design consists of an array of detector elements positioned behind a pinhole, as shown in Fig. (1.6). This requires only a single small probe into the device, and operates in a manner similar to traditional cameras. The resolution of the device is limited by the expense and size of the detector elements, as well as the required size of the pinhole. The size of the detector elements is limited by the space available behind the pinhole, as placing the array of detector elements too far from the pinhole will result in additional orbiting behaviour within the detector, invalidating the assumption that these particles are travelling in straight lines between the pinhole and the detector elements. The pinhole must be small enough to ensure that the incident angle of the particles can be accurately recorded, and large enough that sufficient particles will enter the device to ensure that measurements can be taken on timescales relevant to the phenomena being considered. The detecting elements must be seperated such that cross talk noise is reduced, and the required size of the detector elements make this more challenging to engineer. In addition, there may be parts of the plasma that will be invisible to the detector, if the detector is in the shadow of magnetic coils.

Detector Elements in Collimating Tubes

By splitting the detector into a number of individual well-collimated sensors, each covering distinct orbits, the complication of making the detector sufficiently small so as to fit behind a pinhole is removed. The fast particle profile of the plasma can then be inferred from the data recorded [5]. The total particle production along each orbit is recorded, and by ensuring that the orbits overlap the number of protons produced in each part of the plasma may be inferred. However, the orbits must be well collimated to ensure that the data collected is sufficiently localised that positions of overlap of the orbits can be identified. This limits the number of overlap positions, and the accuracy of the data recorded. A four channel system based on this design will be installed on MAST for the 2013 campaign [39]. This will consist of 4 detectors, considering orbits passing through the core of the plasma.

1.4.5 Proton Detector Materials

The detector will almost certainly be a surface barrier detector, using a layer of doped material as a reverse biased diode. As charged particles traverse the thin layer of doped material, they will induce ionisation currents which can be measured. The amplifiers and measurement equipment associated with detecting these currents will need to be sufficiently far from the plasma that they are not affected by the temperature or radiation, but close enough to the detector that the levels of noise added by the electromagnetic interference present within the tokamak will not swamp the signal.

Silicon Detectors

These have been used to detect fusion born protons on a range of tokamak devices [20]. The advantage of silicon detectors is that they are relatively inexpensive, and, having been in use for a number of years, well understood. However, these detectors require low operating temperature (< 50°C) and relatively few high energy particles ($< 10^9$ protons/cm^2) to prevent damage to the detector [49].

Natural Diamond Detectors

By replacing the silicon layer in the detector with a natural diamond layer, the range of operating conditions can be considerably widened [27]. The diamond detectors will perform at up to $200°$ in a magnetic field of up to 3T [2]. This allows the detector to be much closer to the plasma, giving a higher proton flux. However, these detectors are less common, more expensive, and less well understood than the silicon detectors.

1.4.6 Etendue Problem

The etendue of an optical system is described as being a product of an area A, and the solid angle Ω_S where the solid angle is defined by the properties of the collimating optics of the system. The intensity of a system can be found by measuring the power per unit frequency crossing the etendue. The larger the etendue, the more light gathered by the system. In classical optical systems, the etendue of the system will remain constant. The power detected by the optics is

$$I A \Omega_s = \int j A \Omega_s ds \tag{1.16}$$

where s is the distance along a ray trajectory, I is the intensity, and j is the emissivity

If we try to apply this system to the fusion-born protons leaving the plasma we find that the etendue is no longer constant. This is due to the curved paths the protons take out of the plasma. We must therefore find the etendu of each point in turn using a Monte Carlo method.

Chapter 2

Proton Production Rate

2.1 Deuterium Plasma Reactions

Most tokamak devices, for almost all plasmas, operate with deuterium only plasmas. The use of tritium/deuterium plasmas has been extremely limited, as it leads to radioactive activation of the dust, walls, and interior of the device. The use of deuterium only plasmas allows the investigation of plasmas similar to those that will be used in reactors without the radioactive activation of the device. The injection of fast deuterium into the plasma is used to heat the plasma, and to drive current. It is expected that these fast atoms will be ionised, and will react with thermal ions and other fast ions in the core. The two reactions that produce the most easily detectable products in deuterium plasmas are:

$$\begin{aligned} {}^2_1\mathrm{D} + {}^2_1\mathrm{D} &\rightarrow {}^3_1\mathrm{T} + \mathrm{p}^+ & (2.1)\\ {}^2_1\mathrm{D} + {}^2_1\mathrm{D} &\rightarrow {}^3_2\mathrm{He} + \mathrm{n}^0 & (2.2)\end{aligned}$$

These reactions are equally probable [52], but are still comparatively rare, with the reactivities shown in Fig. 2.1 [6].

2.2 Reaction Probabilities

The reaction rate per unit volume, $\mathcal{R}$:

$$\mathcal{R} = \int\int \sigma(v')v' f_1(v_1) f_2(v_2) d^3v_1 d^3v_2 \tag{2.3}$$

The likelihood of a reaction is a product of two parts:

1. The Reaction Cross-section, $\sigma(v')$.

2. The Probability of Particle Interaction, $v' f_1(v_1) f_2(v_2)$.

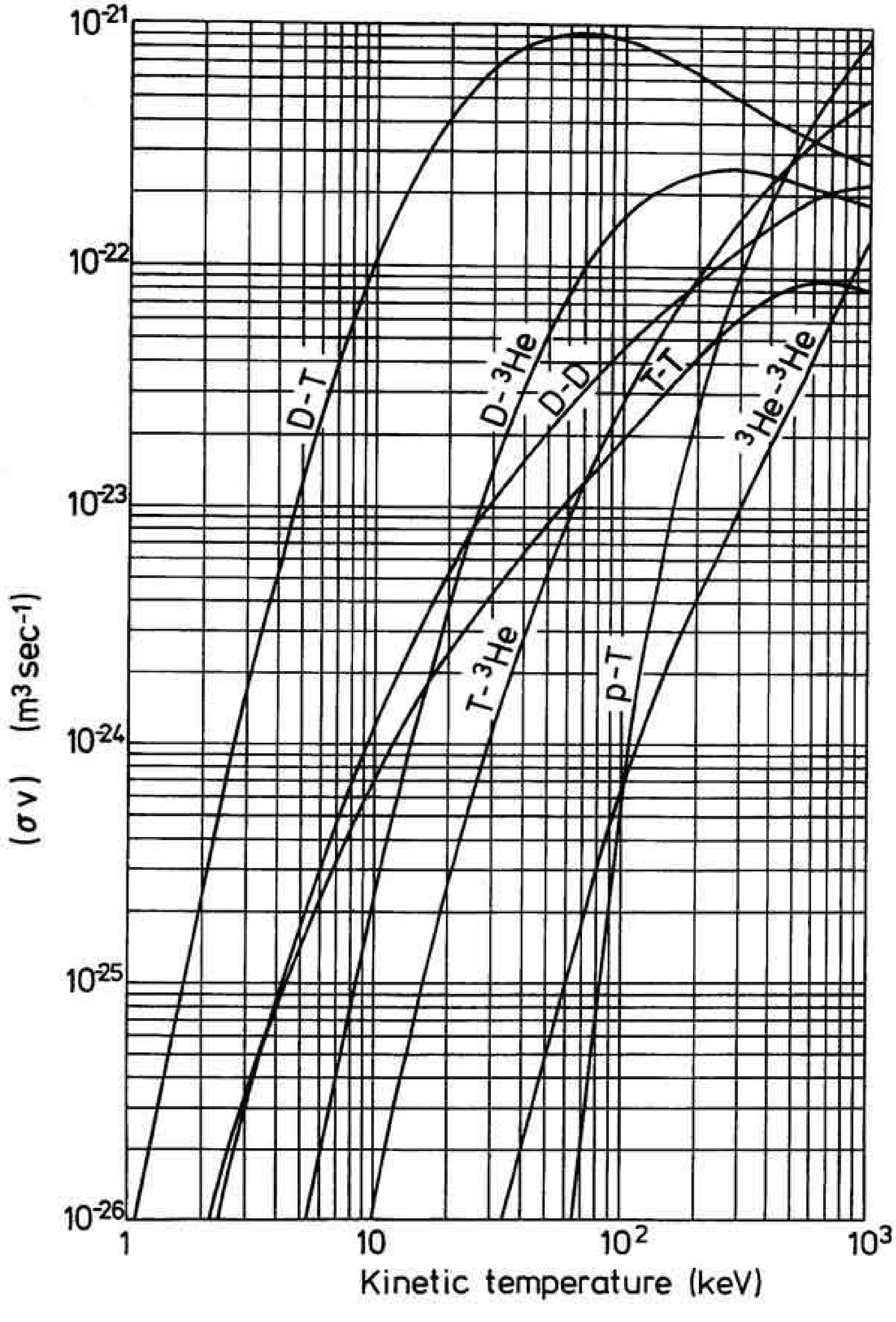


Figure 2.1: The probability of fusion reactions is dependent on the temperature of the constituent ions [23]. The deuterium/tritium interaction has the highest peak at lower temperatures, and is therefore the fuel of choice for reactors. The deuterium/deuterium reaction rate is lower so a deuterium only plasma at the same energy will produce a much lower number of fusion products.

where $v' = v_2 - v_1$.

A population of particles will have a certain proportion of interactions, and each interaction will have a certain chance of creating a fusion reaction.

2.2.1 Reaction Cross-section

The reaction cross-section, σ, gives the likelihood of an interaction between two deuterium ions resulting in one of the reactions given in section 2.1.

$$\sigma \propto \frac{1}{v'^2} \exp(-2G) \tag{2.4}$$

where v' is the relative velocity of the interacting particles $v_1 - v_2$, and G is the Gamow factor, governing the likelihood of the reaction overcoming the Coulomb barrier, where

$$G \approx \frac{e^2}{4\pi\epsilon_0} \frac{\pi Z^2}{\hbar v'} \tag{2.5}$$

where e is the charge on an electron, ϵ_0 is the permittivity of free space, Z is the number of protons in each of the reacting species, here simplified as the reaction involves only one species, $\hbar = h/2\pi$ where h is Planck's constant, and v' is the relative velocity of the two particles.

Thus, for a reaction between two deuterium ions:

$$\sigma(v') \propto \frac{1}{v'^2} \exp\left(\frac{e^2}{4\epsilon_0} \frac{1}{\hbar v'}\right) \tag{2.6}$$

2.2.2 Comparison of Reaction Cross Sections

The reaction cross section, σ, has been parameterised by Miley, Towner, and Ivich [32], Bosch and Hale [6], and Li, Wei, and Liu [28]. Here, these parameterisations are given, and compared for a sample MAST plasma. The accuracy of the parameterisation can be found by comparison to the measured neutron rate, as similar numbers of both protons and neutrons are created in a typical plasma.

Parameterisation via Miley [32]

$$\sigma(g) = 10^{-24} \left(\frac{\frac{A_2}{(A_4 - A_3 g)^2 + 1}}{g \left(\exp\left\{ \frac{A_1}{\sqrt{g}} \right\} - 1 \right)} \right) \tag{2.7}$$

where $A_1 = 46.097$, $A_2 = 372$, $A_3 = 4.63 \times 10^{-4}$, and $A_4 = 1.220$.

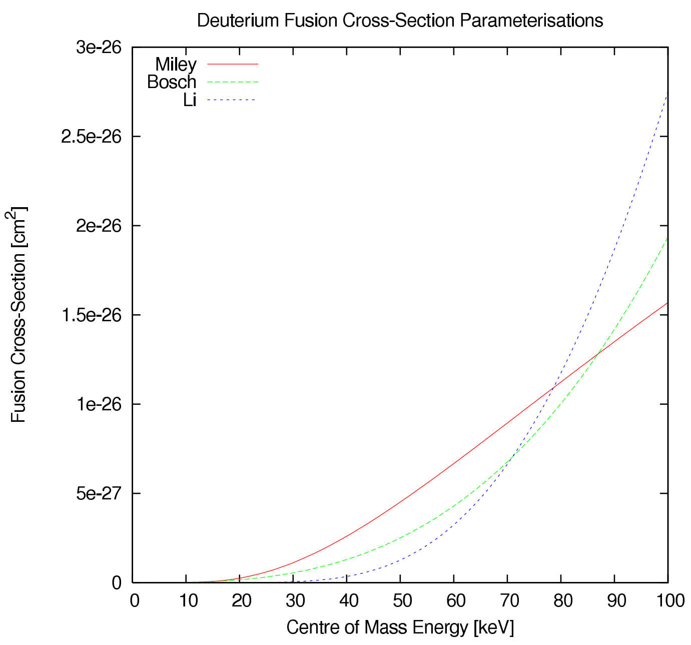


Figure 2.2: Comparison of the Fusion Cross-Sections for the Three Parameterisation Schemes

Parameterisation via Bosch and Hale [6]

$$\sigma(g) = \frac{A1 + g(A2 + g(A3 + g(A4 + gA5)))}{g \exp\left(\frac{B_G}{\sqrt{g}}\right)} \tag{2.8}$$

$$B_G = \pi \alpha Z_1 Z_2 \sqrt{2 m_r c^2} \tag{2.9}$$

where B_G is the Gamow constant, α is the fine structure constant, $m_r c^2$ is the reduced mass of the particle in keV, $A1 = 5.3701 \times 10^4$, $A2 = 3.3027 \times 10^2$, $A3 = -1.2706 \times 10^{-2}$, $A4 = 1.4987 \times 10^{-6}$, and $A5 = 1.8181 \times 10^{-10}$.

Paramterisation via Li, Wei, and Liu [28]

$$\sigma(g) = \frac{\pi}{\frac{2\mu}{\hbar^2} g_{\text{lab}} \left(\frac{M_b}{M_a + M_b}\right)} \frac{1}{\theta^2} \frac{(-4C_3)}{(C_1 + C_2 g_{\text{lab}})^2 + \left(C_3 - \left(\frac{1}{\theta^2}\right)\right)^2} \tag{2.10}$$

where μ is the reduced mass, $\hbar$ is the Planck constant divided by 2π, $1/\theta^2$ is the Gamow penetration factor

$$\theta^2 = \frac{1}{2\pi} \left(\exp\left[\frac{2\pi}{k a_c}\right] - 1 \right) \tag{2.11}$$

where a_c is the length of the Coulomb unit, $a_c = \hbar^2 / \mu z_1 z_2 e^2$, z_1 and z_2 are the charge numbers of the colliding nuclei, e is the electron charge, $k = \sqrt{2\mu g/\hbar^2}$, $C_1 = -60.2641$, $C_2 = 0.05066$, $C_3 = -54.9932$.

Parameterisation	Total Number of Protons Produced
Miley	2.6×10^{12}
Bosch	1.5×10^{13}
Li	6.8×10^{13}

Table 2.1: Total Proton Production in a Sample Plasma, based on MAST shot 18808.

Interaction Rate

We assume that the velocity distribution, $f(v)$, of the plasma is Maxwellian:

$$f(v) = n\left(\frac{m}{2\pi T}\right)^{\frac{3}{2}} \exp\left(-\frac{mv^2}{2T}\right) \tag{2.12}$$

where n is the number density, m is the mass of the particles, T is the temperature and v is the velocity.

2.2.3 Reaction Rate

Combining the velocity distribution and the reaction cross-section, we can find the reaction rate per unit volume, $\mathcal{R}$:

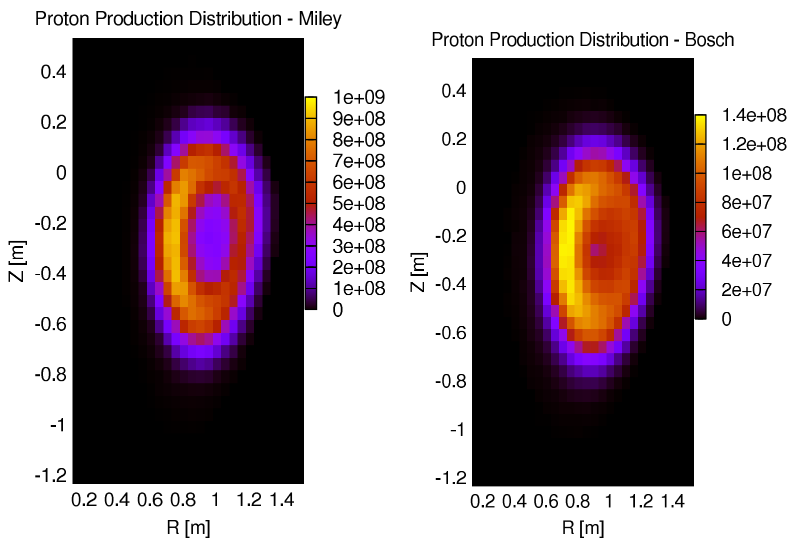
$$\mathcal{R} = \int\int \sigma(v')v' f_1(v_1) f_2(v_2) d^3v_1 d^3v_2 \tag{2.13}$$

Substituting in for $\sigma(v')$ from equation 2.6, and for $f(v)$ from equation 2.12 we obtain

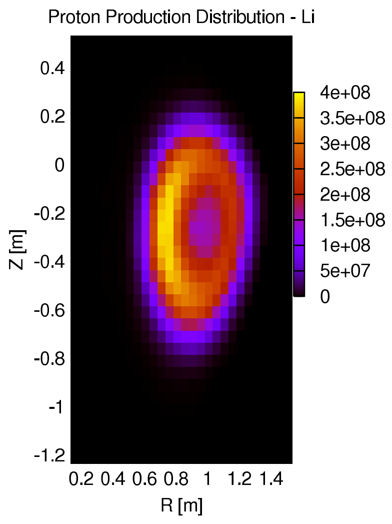
$$\begin{aligned}
\mathcal{R} &= n_1\left(\frac{m_1}{2\pi T}\right)^{\frac{3}{2}} n_2\left(\frac{m_2}{2\pi T}\right)^{\frac{3}{2}} \int\int \frac{1}{v'} \exp\left\{-\frac{2e^2}{4\epsilon_0}\frac{1}{\hbar v'}\right\} \exp\left\{-\frac{m_1 v_1^2}{2T}\right\} \\
&\quad \exp\left\{-\frac{m_2 v_2^2}{2T}\right\} d^3v_1 d^3v_2 && (2.14) \\
&= n_1 n_2 \frac{(m_1 m_2)^{\frac{3}{2}}}{(2\pi T)^3} \int\int \frac{1}{v'} \exp\left\{-\frac{2e^2}{4\epsilon_0}\frac{1}{\hbar v'}\right\} \exp\left\{-\frac{m_1 v_1^2}{2T}\right\} \\
&\quad \exp\left\{-\frac{m_2 v_2^2}{2T}\right\} d^3v_1 d^3v_2 && (2.15)
\end{aligned}$$

2.3 Total Proton Production per Parameterisation

We can use each reaction rate parameterisation in our calculation of the proton production per unit area of the plasma. The results can be seen in Fig. 2.3. The total number of protons produced in the sample plasma for each parameterisation is shown in Table 2.1. From the neutron yield monitors the measured yield is $\sim 10^{13}$ s^{-1} [46]. The average of our three parameterisations is 2.9×10^{13}. We shall therefore use the Miley parameterisation, as this gives the closest result to the average of our three parameterisations.



(a) Miley Parameterisation (b) Bosch Parameterisation



(c) Li Parameterisation

Figure 2.3: The rate of proton production for each parameterisation, with 5cm grid spacing. The Miley parameterisation gives a total number of protons nearest to the average, and to the measured neutron yields.

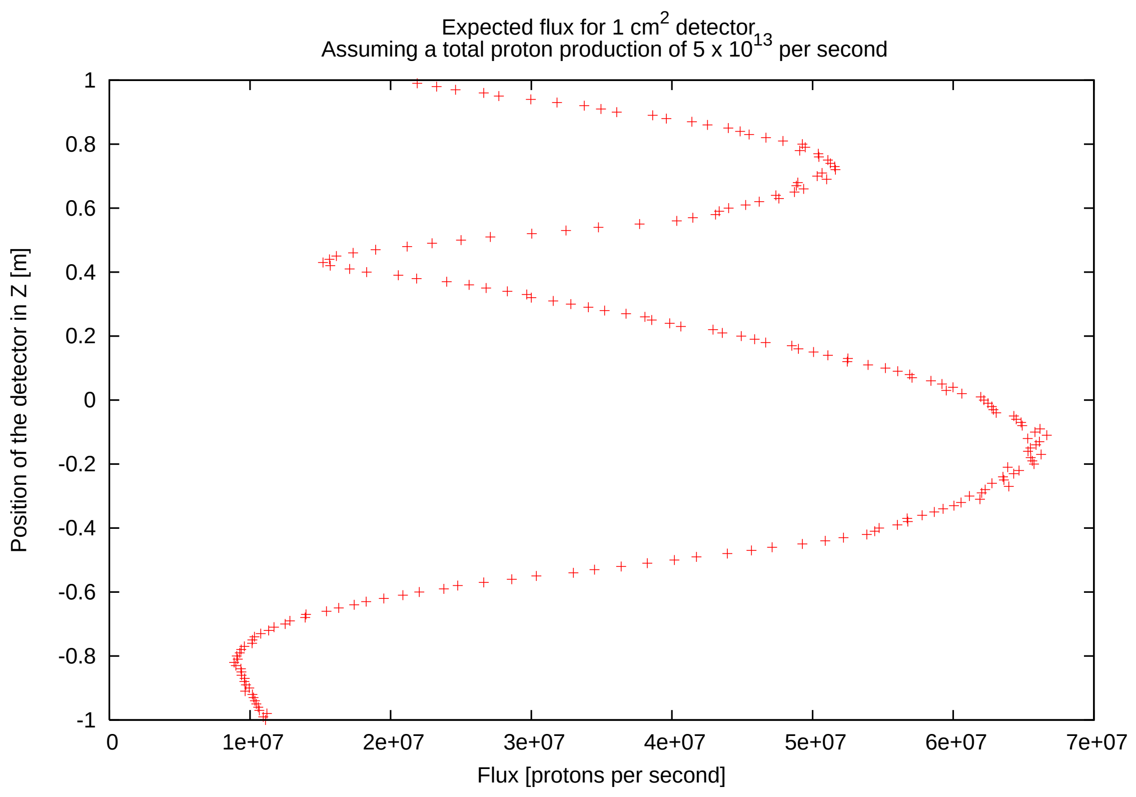


Figure 2.4: The proton flux at the vessel wall. The shadows of the magnetic coils can be seen.

2.4 Proton Flux

Having found the proton production per unit area, it is possible to find the proton flux per unit area of the wall. This is shown in Fig. (2.4). If we assume that the plasma is toroidally symmetric, we can find the ideal location for the detector to receive the maximum flux of protons. This should also correspond to having the widest possible view of the plasma.

2.5 Particle Velocities

The initial velocity of the proton is determined by the energy released in the fusion event, and the velocity of the centre of mass of the parent particles. From equation 2.1 we know that the proton will have an energy of 3.02 MeV. In addition, the beam ions, which are the highest contributor to the reaction rate, have an energy of approximately 40 keV before the MAST upgrade, and the thermal ions have approximately 1 keV. We can divide the fusion events into three types: Beam-Beam fusion, Beam-Thermal fusion, and Thermal-Thermal fusion. For a typical MAST plasma, $\sim$ 80% of the fusion products seen are produced by beam-thermal ion interactions, and $\sim$ 20% from beam-beam interactions. The contribution to the initial speed of the proton from the energy released in the fusion reaction is 2.41×10^7 ms^{-1}. The velocity of the centre of mass of the parent particles is approximately 4×10^5 ms^{-1}. This gives a maximum initial velocity of 2.45×10^7 ms^{-1}.

2.5.1 Beam Ion Relaxation Time

The slowing down time of the injected fast particles can be estimated from the measured neutral flux after the beam has been shut off. The relaxation time to a Maxwellian distribution is ~ 65 ms [48]. If we consider a typical MAST plasma, where within the central region the electron temperature and density are assumed to be constant, and the effective charge is 1, as measured by the Thompson scattering and bremsstrahlung imaging diagnostics. If we consider the passing particles, and assume that $n_i = n_e$, we can find the energy loss rate of the primary beam ion energy, E_b

$$\frac{dE_b}{dt} = -\frac{2^{\frac{1}{2}} n_e Z_b^2 e^4 m_e^{\frac{1}{2}} \ln \Lambda_e}{6\pi^{\frac{3}{2}} \epsilon_0^2 m_b T_e^{\frac{3}{2}}} E_b - \frac{2^{\frac{1}{2}} n_i Z_i^2 Z_b^2 e^4 m_b^{\frac{1}{2}} \ln \Lambda_i}{8\pi^{\frac{3}{2}} \epsilon_0^2 m_i E_b^{\frac{1}{2}}} \tag{2.16}$$

This is simplified to

$$\frac{dE_b}{dt} = -\frac{2E_b}{\tau_{ei}} \left[1 + \left(\frac{E_c}{E_b} \right)^{\frac{3}{2}} \right] \tag{2.17}$$

where τ_{ei} is the electron ion collision time, and E_c is the critical beam energy. This is defined as being the fast ion energy at which ions and electrons receive equal power transfer, and is given by

$$E_c = m_b T_e \left(\frac{3\pi^{\frac{1}{2}} Z_i^2 \ln \Lambda_i}{4 m_e^{\frac{1}{2}} m_i \ln \Lambda_e} \right)^{\frac{2}{3}} \tag{2.18}$$

For a typical MAST deuterium plasma, the critical energy is ≈ 20keV.

We can estimate the energy of the beam ions some time t after injection by taking the integral of Eq. (2.17)

$$E_b(t) = E_0 \left[e^{\frac{-3t}{\tau_{ei}}} - \left(\frac{E_c}{E_0} \right)^{\frac{3}{2}} \left(1 - e^{\frac{3t}{\tau_{ei}}} \right) \right]^{\frac{2}{3}} \tag{2.19}$$

where E_0 is the initial beam energy. This is limited to the finding the time for the beam ions to slow to energy of the thermal ions. This allows us to find the characteristic slowing down time, τ_s for the beam ions to fully thermalise

$$\tau_s = \frac{\tau_{ei}}{3} \ln \left[\frac{1 + \left(\frac{E_c}{E_0} \right)^{\frac{3}{2}}}{\left(\frac{T_e}{E_0} \right)^{\frac{3}{2}} + \left(\frac{E_c}{E_0} \right)^{\frac{3}{2}}} \right] \tag{2.20}$$

This time can be compared to the relaxation time required for the fast particle pop-

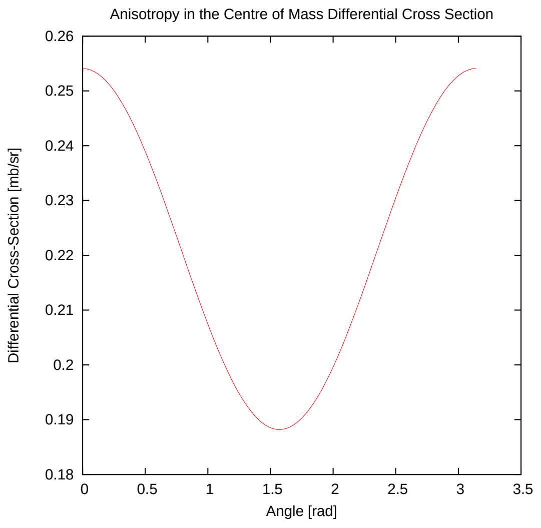


Figure 2.5: Anistropy in the Differential Cross-Section for the $^2\mathrm{H}(d,p)^3\mathrm{H}$ Reaction.

ulation to dissipate after the neutral beams are switched off, and there is a clear fit [48]. The slowing down time is therefore well described by classical theory, and is due predominantly to Coulomb collisions and charge exchange phenomena.

There is evidence that the velocity distribution of fusion products is not isotropic, but dependent on the velocities of the parent particles [10]. The particles are assumed to travel along magnetic field lines, as the effects of the gyro-orbits will average out and the effects of particle drifts are neglected, and so the velocity distribution can be implemented with the parent particles being considered as following the magnetic field lines. We can therefore find the particle velocity distribution as a function of the magnetic field angle. The protons are assumed to be created in fusion events between a beam ion and a thermalised plasma ion, where the energy of the beam is 40 keV, and the energy of the thermal ion is 1 keV. Coupled with the assumption that the particles will be following the magnetic field lines, as it assumed that any effect of the gyro-orbit will average out over large numbers of fusion events, the initial velocity distribution as a function of magnetic field angle will show a level of anisotropy. The anisotropy in the differential cross-section is shown in Fig. 2.5.1.

2.6 LOCUST-GPU

LOCUST-GPU is a full-orbit code, including electron drag, thermal ion scattering, and diffusion [1]. This provides an accurate model of the fast particle distribution due to the neutral beams injected into the plasma. This distribution could be used as the expert knowledge in Bayesian probability calculations, allowing the calculation of the entropy of our plasma image with reference to the LOCUST-GPU results. The distribution of fast particle energy is extremely useful to the calculation of the initial energies of the fusion born protons, which in turn allows the expected energy of protons at the detector to be calculated.

Chapter 3

Orbit Code and the GPU

3.1 Introduction to CUDA

Compute Unified Device Architecture (CUDA) is a computer architecture designed to allow easy parallelisation of computing tasks developed by nVidia [33]. The main method of using CUDA devices is the CUDA for C language. The term CUDA has therefore spread to refer to the programming language as well as the class of devices. CUDA includes extensions to C that allow control of the graphics processing unit (GPU). This allows parallel code to be run using the computing resources of the GPU. The GPU is, in general, a single instruction multiple data device, optimised for vector calculations.

There are several advantages to using the GPU. The problem of tracking independent particles along their orbits requires limited instructions, needs only limited access to data stored in the host device, and can utilise hardware optimised for vector problems. General Purpose Graphical Processing Units (GPGPUs) are ideally suited to problems of this type. In particular, we take advantage of the hardware accelerated two dimensional interpolation usually used for graphics texture interpolation. This allows us to interpolate the magnetic field at a low computational cost. However, the limited amount of memory available on the device does restrict the size of the problem that can be considered, as well as restricting the possibility of interaction between the particles under consideration.

The two devices used for this work are an nVidia GeForce GTX 260 and a GeForce GT 640. These devices have a large number of CUDA cores (see Tab. (3.1)), each of which is an independent computing unit. Each core consists of a limited amount of registry memory, an integer processor, a floating point processor, and a input/output bus connecting the core to the rest of the streaming multiprocessor. These cores can access a range of memory types on the device. The cards are compared in Table (3.1).

When programming these devices, the program must be split first into two parts. The kernel is the code which will run on the CUDA device, whilst the CPU code is used to initialise the code, and pass data to and from the CUDA device. The kernel code is broadly

Device	GTX 260	GT 640
Architecture	Tesla	Fermi
CUDA cores	192	384
Clock Rate [MHz]	576	900
Device Memory (DDR3) [MB]	896	2048
Memory Bandwidth [GB/s]	111.9	28.5
Maximum Performance [GFLOPS]	715	691
Streaming Multiprocessors	1	2
Orbits calculated per second	3560	47770

Table 3.1: Comparison of GTX 260 and GT 640 CUDA devices

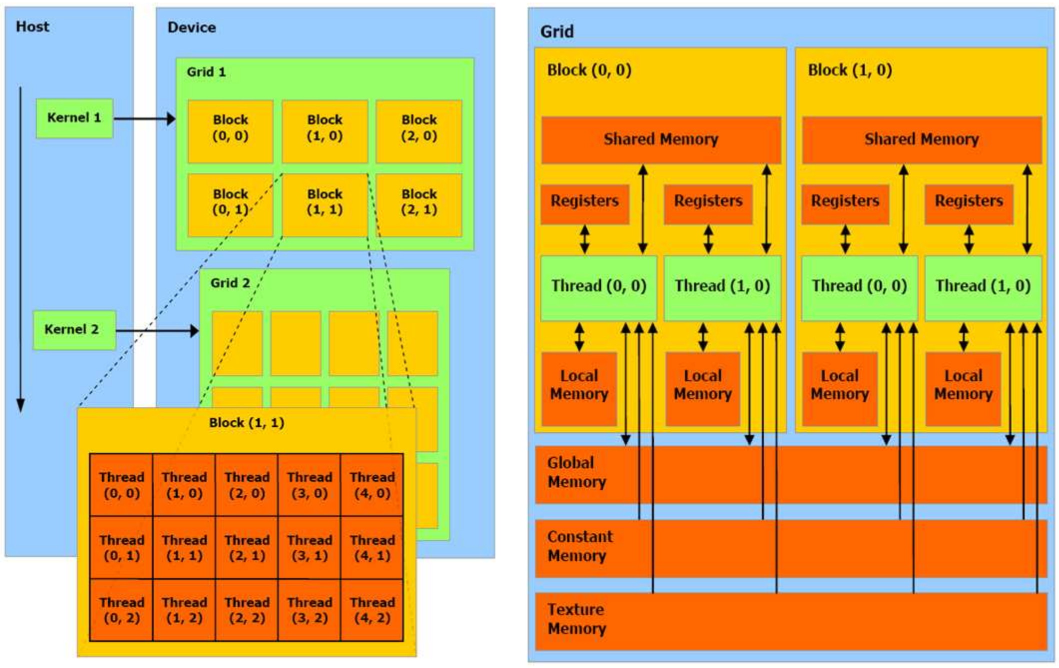


Figure 3.1: Heirarchy of the CUDA device, showing the sequence of code operation on the left, and the architecture of the device memory on the right [33].

comparable to a function in C. These kernels will be executed in threads, similar to POSIX threads, with the threads formed into groups called blocks. These threads will be executed in parallel within blocks, and for devices with multiple streaming multiprocessors, the blocks can be executed in parallel. Each thread will run on a single CUDA core, and each block on a single streaming multiprocessor. The threads are further bundled into warps, with 32 threads per warp. Each warp is run simultaneously, and the same instruction set is used across all the threads within a warp. To ensure that the device is used to full capacity, it is common to have many more threads per block than there are CUDA cores on the device. The threads may depend upon code running in other threads within the same block, but the blocks must be able to run independantly, and may be run in any order. The blocks are collected into grids, with one grid called per invocation of the kernel. The hierarchy of the code execution and memory on the device is shown in Fig. (3.1).

The memory in a CUDA program is broadly split into four types. Initially, we divide

Memory Type	Speed [GBs^{-1}]
Coalesced	100
Misaligned Contiguous	50
Random Scatter/Gather	10

Table 3.2: Memory bandwidth for various types of read within the CUDA device.

the memory accessed by the program into host memory, attached to the CPU, and device memory, on the CUDA card. On the card, each CUDA core will have access to per-thread local memory registers in the core itself. The threads bundled into a block have access to memory shared within a block, but not between blocks. Finally, there is the global memory on the device. This is accessible by any thread within a grid. Accessing the global memory has about 400-600 clock cycles of latency, as opposed to 10 cycles for register and 20 cycles for shared memory. This latency can be hidden by ensuring that reads are contiguous, especially if the read is contiguous for a warp. For current Tesla devices, the memory bandwidth within the device can be seen in Table (3.2).

There are two further sub-types of memory on the device. The constant memory, which is accessable by all threads, is nearly as fast as the register memory. However, it can only be updated from the host, and is read-only from the device. More importantly for our purposes is the texture memory. This is also faster than the global memory, but can only be updated from the CPU. The texture memory is optimised for 2D locality, and spatially cached. This is designed to store 2D arrays, and any memory lookup to the texture memory will read not only the single value requested, but also an area of memory surrounding it. This memory will be stored in a local cache, so any further reads to this area within a warp will be much faster. It allows non-integer memory array fetching, using either bilinear interpolation or nearest point lookup. This interpolation is hardware accelerated, and is as fast as a simple memory lookup.

The CUDA devices are, in general, optimised for integer and 32-bit floating point operation. These operations will be performed considerably more efficiently than 64-bit floating point operations. The GT 640 can perform 192 32-bit floating point addition or multiplication operations per clock cycle, but only 8 64-bit floating point addition or multiplication operations. If the algorithm used requires greater than 32-bit accuracy it is very difficult to achieve performances approaching theoretical peak performance from the CUDA devices. The theoretical peak GPU performance has been increasing faster than the CPU performance over the last 10 years, as shown in Fig. (3.2). The cost of GPUs is driven largely by the gaming market, although devices designed purely for high performance computing have also been released. Modern games have come to rely upon extremely realistic graphics to help provide immersion. These graphics require ever more powerful GPUs to render them, and the cost of the design and manufacture of the GPU is spread across the entire gaming market, lowering the cost for high performance devices.

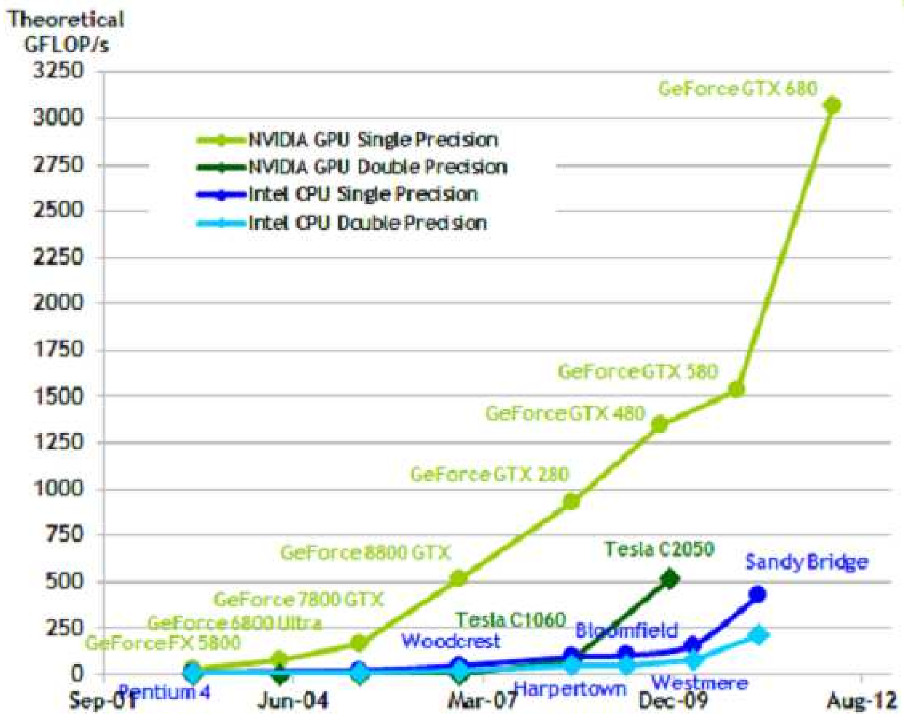


Figure 3.2: The theoretical peak performance for GPU and CPU over the last decade. The GPU peak performance has outgrown the CPU performance, although accessing the peak performance will be restricted to problems that are suited to the GPU architecture [33].

3.1.1 CUDA Optimisation

CUDA optimisation falls into two main categories. We must optimise algorithms, and the memory access. There is considerable overlap between these categories, as we can hide memory access latency by careful choice of calculation order. The chief method of calculation optimisation consists of ensuring that all threads within a warp have a similar runtime. If one thread in a warp takes much longer to finish, the others will wait, rather than having their CUDA cores reassigned to new warps. This is known as warp divergence.

The second part of optimisation is ensuring that the code is not limited by the memory latency or bandwidth. The memory bandwidth between the host and the device, shown in Table (3.1), limits the speed at which data can be passed between the host and the device. However, the kernel launch is asynchronous to the host code. We can start passing data to the kernel, hide the latency by performing further calculations on the host, start the kernel, perform more calculations on the host as the kernel runs, and the data is passed back. Providing the algorithm allows this additional calculation to be run on the host, this can hide the latency of the kernel. For the code running on the device, careful use of all the memory types can allow the latency of memory lookups to be minimised. In addition, caching of texture memory lookups, combined with warps running similar data, can ensure that the memory bottlenecks are avoided on the device.

3.2 Orbit Following Code

The orbit-following code CUEBIT is based on an algorithm created by Dr. McClements [31]. It is an energy conserving solution to the Lorentz equation in the absence of an electric field.

$$\begin{aligned} \frac{d\mathbf{x}}{dt} &= \mathbf{v} && (3.1) \\ m\frac{d\mathbf{v}}{dt} &= Ze\mathbf{v} \times \mathbf{B}(\mathbf{x}) && (3.2) \end{aligned}$$

where m is the mass of the proton, Ze is the charge, $\mathbf{v}$ is the velocity and $\mathbf{B}(\mathbf{x})$ is the magnetic field at position $\mathbf{x}$.

The Eqns. (3.2), (3.1) can be approximated using the finite difference system

$$\begin{aligned} m\frac{\mathbf{v^{(n+1)}} - \mathbf{v^{(n)}}}{\delta t} &= e\left(\frac{\mathbf{v^{(n+1)}} + \mathbf{v^{(n)}}}{2}\right) \times \mathbf{B}\left(\frac{\mathbf{x^{(n+1)}} + \mathbf{x^{(n)}}}{\mathbf{2}}\right) && (3.3) \\ \frac{\mathbf{x^{(n+1)}} - \mathbf{x^{(n)}}}{\delta t} &= \frac{\mathbf{v^{(n+1)}} + \mathbf{v^{(n)}}}{2} && (3.4) \end{aligned}$$

where the superscript n denotes the timestep under consideration. This ensures that the energy of the particle is conserved, which can be shown by taking the scalar product of $\mathbf{v^{(n+1)}} + \mathbf{v^{(n)}}$ with Eq. 3.3, giving $(v^{(n+1)})^2 = (v^{(n)})^2$.

Using Cartesian coordinates, and replacing the $(\mathbf{x^{(n+1)}} + \mathbf{x^{(n)}})/\mathbf{2}$ with $\mathbf{x^{(n)}}$, we can obtain a first approximation of the solution to Eq. 3.3 as a 3×3 matrix, which can then be solved. This gives an estimate of $\mathbf{v^{(n+1)}}$, which can be used to refine the estimate of $\mathbf{x^{(n+1)}}$ using Eq. 3.4, which in turn is used to gain a more accurate estimate of $\mathbf{v^{(n+1)}}$, and so on. It is found that an acceptable level of accuracy can be obtained by repeating this process three times, reducing the computational overhead of testing for convergence.

3.2.1 4th-Order Runge-Kutta Technique

In order to solve equation 3.2 as efficiently as possible to allow larger numbers of particles to be tracked, improving the overall statistics, the scheme used by Dr. McClements, [31] was replaced with a 4th order Runge-Kutta scheme [26]. The general method for this scheme is given as

$$\begin{aligned} \frac{dy}{dt} &= f(t, y) && (3.5) \\ y(t^{(0)}) &= y^{(0)} && (3.6) \end{aligned}$$

where f is some function of t and y, $t^{(0)}$ is the start point of the system, and $y^{(0)}$ is the value of y when $t = 0$. The numerical approximation of this is then given by

$$y^{(n+1)} = y^{(n)} + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4) \quad (3.7)$$

$$t^{(n+1)} = t^{(n)} + h \quad (3.8)$$

where $y^{(n+1)}$ is the approximation of $y(t^{(n+1)})$, n denotes the timestep, h the time interval, and

$$k_1 = f(t^{(n)}, y^{(n)}) \quad (3.9)$$

$$k_2 = f(t^{(n)} + \frac{1}{2}h, y^{(n)} + \frac{1}{2}hk_1) \quad (3.10)$$

$$k_3 = f(t^{(n)} + \frac{1}{2}h, y^{(n)} + \frac{1}{2}hk_2) \quad (3.11)$$

$$k_4 = f(t^{(n)} + h, y^{(n)} + hk_3) \quad (3.12)$$

This has a total accumulated error of order h^4.

3.2.2 CUEBIT++

The serial code utilising the CUEBIT method to follow the orbit of particles until the particles leave the device, known as CUEBIT++ was initially used to follow sample particle orbits. This ensured that the energy of the particles was conserved, but was computationally more intense that the Runge-Kutta technique outlined in Section (3.2.1). Using the GTX 260, the CUEBIT++ code takes 257 minutes and 29 seconds to calculate the orbits of 45,793,280 particles, a rate of 2964 particles a second. The Runge-Kutta code takes 225 minutes and 20 seconds to calculate the orbits of 48,128,000 particles, a rate of 3560 particles per second.

3.2.3 Back Code

We can also take the image at the detector plate, and use this as the basis for particles accelerated back into the plasma. This requires that we know the initial speed of the particle, which is assumed to be unchanged from the birth speed of the particles, and the arrival angles of the particles, which are found from the detector image.

3.3 Particle Creation and Detection

Having split the plasma into pixels, as shown in section 5.1.1, we are in a position to start the process of finding the transformation matrix M. The transformation matrix relates the birth positions of the protons to the detector image of the protons. Particles are created in equal numbers in each small volume of the plasma corresponding, when projected into the R, Z plane, to pixels within the last closed flux surface. Within each pixel, the particles are given a random position in the R, Z plane, as well as a random toroidal angle. The fusion born protons have an initial velocity distribution as calculated in Sec. 2.5. The particles are created in batches of 4096, as this allows the most efficient use of the available memory on the CUDA device. By using 4096 particles, with one particle per thread, we ensure that all the CUDA cores are continually in use. We must pass only the initial positions and velocities of the particles to the CUDA device, and pass the penultimate and ultimate positions of the particles as they hit the walls of the device back to the detector. This allows us to minimise the amount of data we need to pass to and from the device.

Having found the positions of the particle before and after it has hit the wall, we perform a simple linear interpolation between these two points to find the point of intersection with the wall. If this point of intersection coincides with the position of the detector, it can be recorded as a successful point. This requires that the stepsize be sufficiently small that the curve between the points can be estimated as a straight line. Here, we can use the assumed toroidal symmetry of the tokamak and magnetic field to allow us to minimise the numbers of particles we need to simulate. If we assume that the detector is a ring around the device, limited in the Z-direction, we can find all the particles that hit this detector, and then rotate the initial position of the particles by the same angle as the angle the detector would have to be rotated to localise it to the original angular extent. If the angular extent of the detector is ψ, this allows us to simulate $2\pi/\psi$ fewer particles to get the same number of particles hitting the detector.

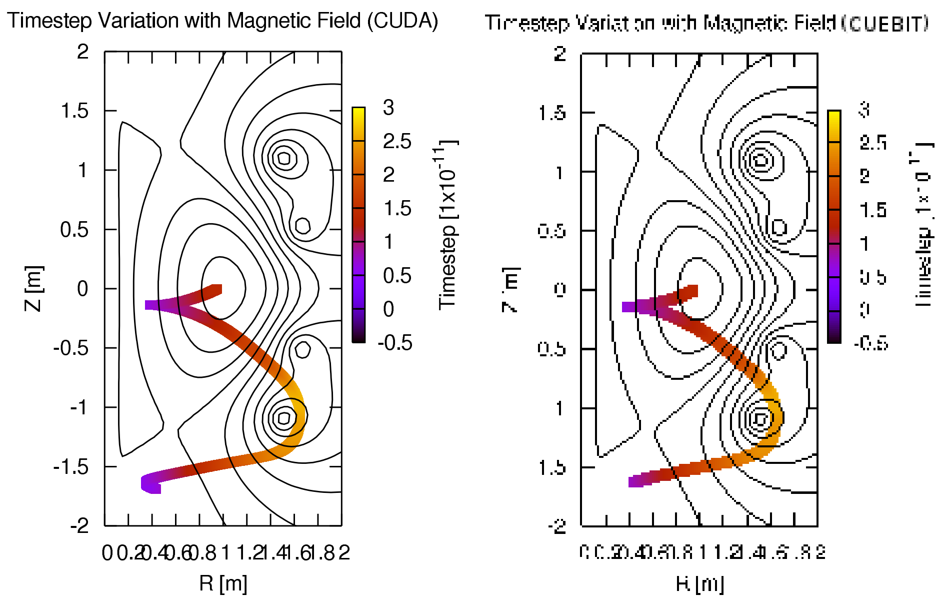
3.4 Results

3.4.1 Benchmarking CUDA Code Against Serial Code

The CUEBIT++ code was rewritten to take advantage of the CUDA compute capabilities of an nVidia GTX 260, and an nVidia GT 640.

Speed of codes

The CUDA code can assign initial positions and velocities to, track the orbits of, and test for interaction with detector of 3,560 particles a second for the GTX 260, and 47,770 particles per second for the GT 640. For a plasma split into 5cm by 5cm pixels, at least 8.5×10^8 particles are needed to ensure an accurate matrix, as shown in Section (4.1.2), which would take a total run time of approximately 297 minutes to complete. The serial code, run purely on the CPU, can track the orbits of 67 particles per second, and would



(a) CUDA/Runge-Kutta Code (b) Serial/CUEBIT code

Figure 3.3: Comparison of the changes in timestep for CUDA and CUEBIT codes over a single orbit. The timestep increases with the magnetic field, as it is a proportion of Larmor period. This is most evident near the central column.

take 211,000 minutes to calculate the matrix. The CUDA code, running on the GT640, is approximately 700 times faster than the serial code running on a 2.27GHz Intel Xeon E5520 CPU.

Accuracy of results

The energy of a charged particle in a magnetic field is independent of the time of measurement. We can therefore compare the energy of the particles at the detector to the initial energy of the particles. If the energy is constant, or nearly so, we can be confident that the code is conserving energy. Taking protons born with an energy of 3.1225 MeV, which have anisotropic birth velocities as calculated in Section (2.5), we can find the energies of the protons at the detector. This is shown in Fig. (3.4). The detector is slightly biased towards particles with greater energy, which is a result of the anisotropic initial distribution. The energy of the particle is conserved, and if we ignore the contribution to the energy of the parents centre of mass, and consider only the fusion energy, the energy of the particle at the detector is unchanged.

We can compare the accuracy of the CUDA code, particularly the hardware accelerated interpolation, in a number of ways. In figure 3.3 we can see that both CUDA and C++ codes have the same timestep, as is expected.

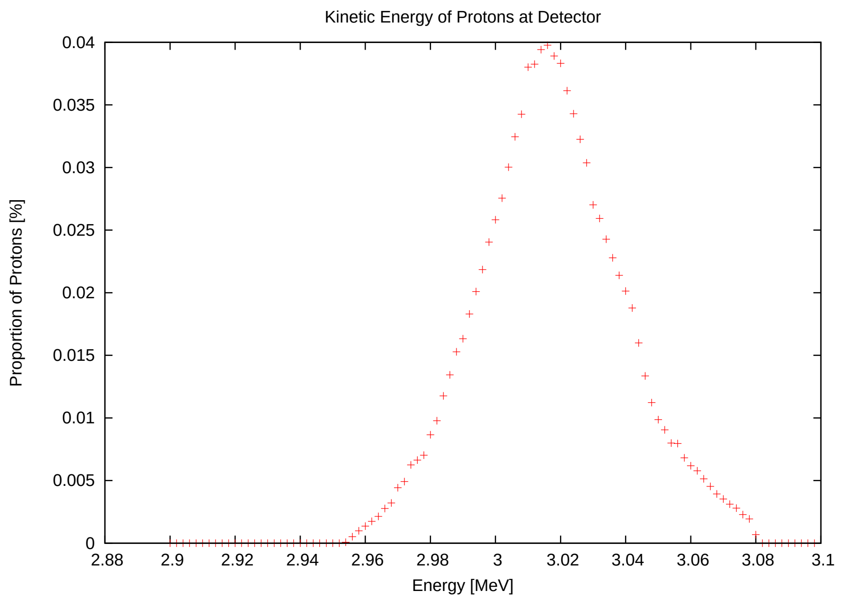


Figure 3.4: The kinetic energy of particles incident on the detector. The peak is at 3MeV, as expected. The greater spread towards the higher energies is due to there being some areas of the plasma from which only protons with higher energies are able to reach the detector.

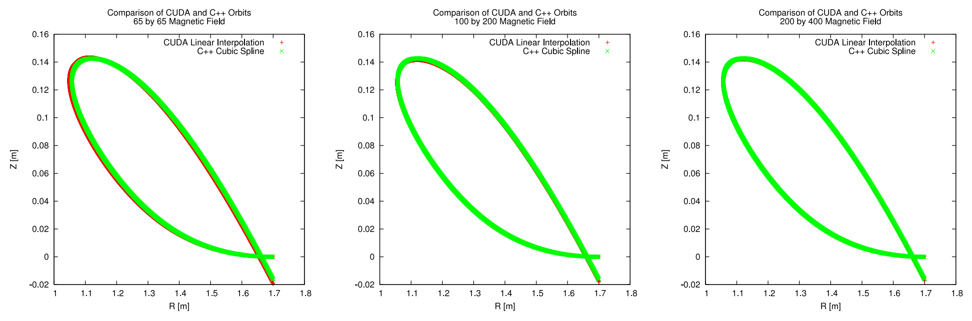


Figure 3.5: Comparison of CUEBIT on both CUDA and C++, showing Orbits for different Magnetic Field Gridsizes, being, from left to right, 65 by 65, 100 by 200, and 200 by 400. This shows the difference between the linear interpolation, as performed on the GPGPU, and the bicubic spline, as performed on the CPU.

3.4.2 Interpolation of the Magnetic Field

The magnetic field is taken from EFIT data. This gives the magnetic field as a 65 by 65 grid in R, Z. We assume that the magnetic field will be toroidally symmetric. The hardware accelerated linear interpolation of the CUDA device is compared to the bicubic spline interpolation of the C++ code. The bicubic spline technique is then used to resample the magnetic field. Example orbits can be seen in Figs. (3.5(a), 3.5(b), 3.5(c)). The accuracy of the orbit increases with the increasing number of points in the magnetic field grid. The increase in accuracy must be counterbalanced by the increased memory usage of the larger magnetic grid on the CUDA device.

Timestep Larmor Fraction	Final Position X [m]	Final Position Y [m]	Final Position Z [m]	No. of Points
0.1	1.2674	1.3828	0.1592	3
0.01	1.1681	1.2808	-0.02683	39
0.001	1.1487	1.2539	-0.01622	393
0.0001	1.1488	1.2535	-0.01636	3938
0.00001	1.1486	1.2533	-0.016235	39383
0.000001	1.1486	1.2533	-0.016228	393835

Table 3.3: C++ Change in final postion of a sample orbit with change in timestep. The orbit is sufficiently accurate when the timestep is is 0.0001 of the Larmor period.

3.4.3 Orbit Tracking Timestep

The stepsize in the orbit tracking is taken as a proportion of the period of the Larmor orbit. The Larmor period, L is given by

$$L = \frac{2\pi m}{q|B|} \tag{3.13}$$

As the timestep, as a proportion of the Larmor period, is decreased, the accuracy of the orbit, and the time needed to calculate the orbit, is increased. We want, therefore, to choose the timestep such that the orbit is calculated as fast as possible, to the given levels of accuracy. The table 3.3 shows the change in accuracy as the timestep size is decreased, showing the change in the final position of the orbit as the number of points in the orbit is increased. Each point in the orbit takes the same amount of time to calculate. As the detector pinhole is 1cm by 1cm, the inaccuracy of the final position of the orbit should be no more than 1mm. We therefore choose the timestep to be 0.0001 of the Larmor period.

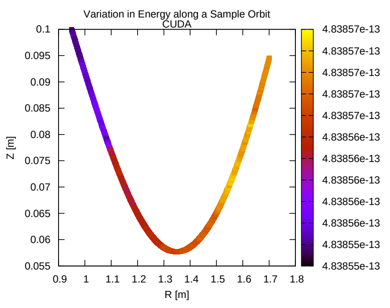
3.4.4 Constants of Motion

We can test the accuracy of the codes by considering the constants of motion for particles moving through the plasma. We are expecting these to have some variation due to the limits of numerical accuracy, but excessive variation is to be avoided at all costs. We shall consider the energy of the particle, and the toroidal canonical momentum.

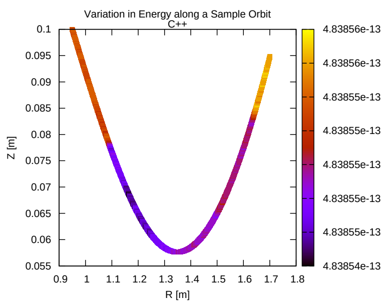
Energy

The energy of the particle along the orbit is expected to be constant. As we can see from Fig. (3.6) the variation in energy along a particle orbit is extremely small, which is consistent with this being a constant of motion to the limits of numerical accuracy. The Lagrangian of the particle is given by

$$L = \frac{1}{2} m\mathbf{v} \cdot \mathbf{v} - \mathbf{q}\mathbf{v} \cdot \mathbf{A} \tag{3.14}$$



(a) CUDA code, colour denotes energy in J



(b) C++ code, colour denotes energy in J

Figure 3.6: The energy of the particle at each timestep in the orbit is shown in colour. The variation in the energy along the orbit is extremely small, so this may be taken to be a constant of motion. This demonstrates that both calculating schemes conserve the energy.

The agreement between the CUDA and C++ codes is extremely good. The total difference between the energies recorded by each code is shown in Fig. (3.7). This is also consistent with the energy being a constant of the motion.

Canonical Toroidal Momentum

The Lagrangian of the proton can be expressed as

$$L = \frac{1}{2}mr^2\dot{\phi}^2 - qr\dot{\phi}A_\phi \tag{3.15}$$

where m is the mass of the proton, r is the distance from the centre of the tokamak, ϕ is the toroidal angle, q is the charge, and A_ϕ is the toroidal component of the vector potential. As the plasma we are considering is toroidally symmetric

$$\frac{\partial A_\phi}{\partial \dot{\phi}} = 0 \tag{3.16}$$

The canonical toroidal momentum is given by [52]

$$\begin{aligned} p_\phi &= \frac{\partial L}{\partial \dot{\phi}} && (3.17)\\ &= mr^2\dot{\phi} - qrA_\phi && (3.18)\\ &= mrv_\phi - q\Psi && (3.19) \end{aligned}$$

The canonical toroidal momentum along a sample orbit for RK4 and CUEBIT codes is shown in Fig. 3.8.

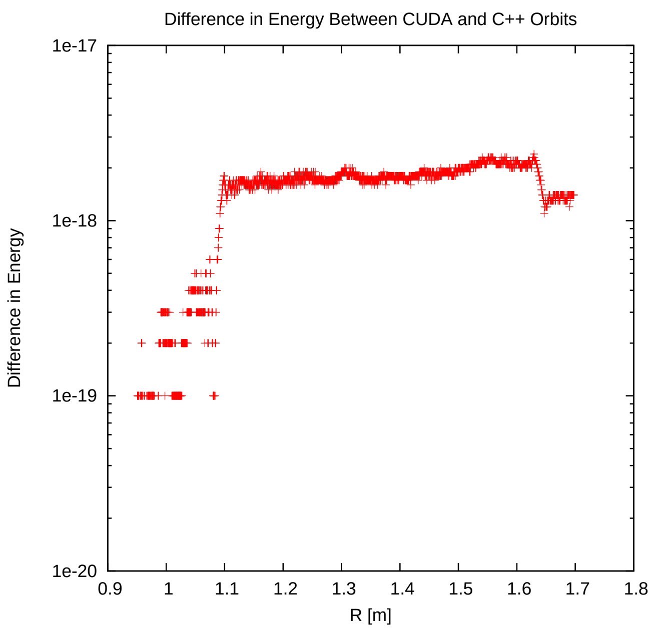


Figure 3.7: The difference in energy recorded at each timestep between the C++ and CUDA codes. This shows that the difference in the energies calculated at each timestep for the C++ and CUDA codes is of the order of numerical error for the CUDA device.

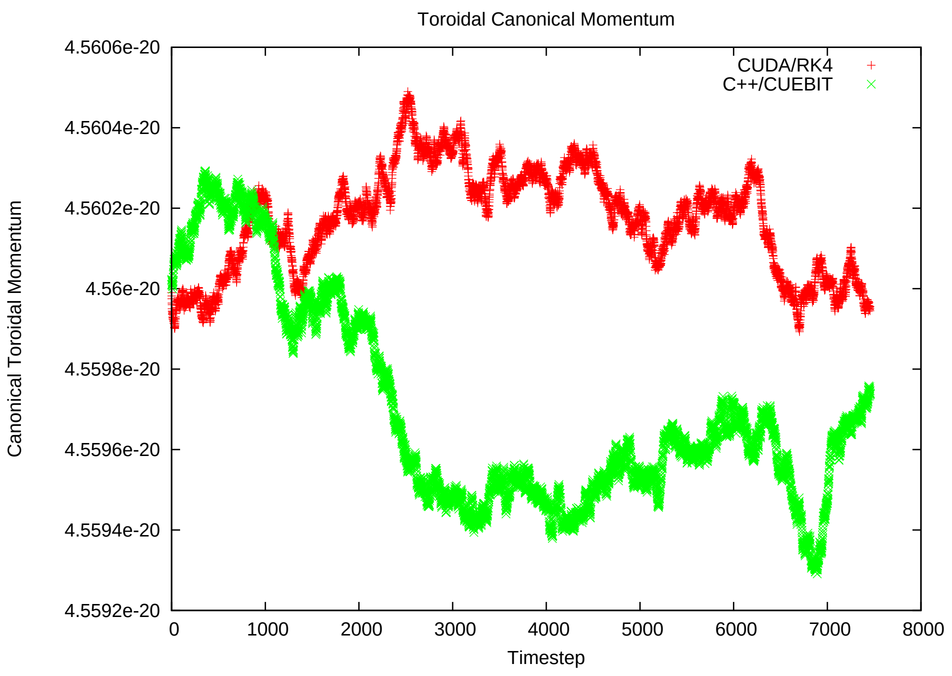


Figure 3.8: The canonical toroidal momentum along a single orbit. The variation along an orbit is small, and is due to numerical error.

Chapter 4

Constructing the Transformation Matrix

The transformation matrix is a matrix that relates a given distribution of the birthplaces of fusion born protons within the plasma to the positions of incidence of the fusion born protons at the detector elements. The rows of the matrix correspond to pixel subdivisions of the R, Z plane of the plasma, and the columns of the matrix correspond to the detector elements.

4.1 Brute Force Matrix Construction

The transformation matrix can be constructed by creating large numbers of particles in each grid cell in the plasma. These particles are then tracked along their orbits. Those that hit the detector have their arrival angle at the detector recorded, along with the initial position. These angles and positions are then used to construct the transformation matrix. This matrix will be non-square, with the number of columns given by the number of grid points in the plasma, and the number of rows given by the number of detector elements. The size of the matrix can be truncated by considering only points within the last closed flux surface, and only elements of the detector which register a signal. For a typical MAST plasma, with the plasma divided into 5cm by 5cm cells, and the detector divided into $15°$ by $15°$ cells, the transformation matrix will have ~ 70 rows and ~ 430 columns with non-zero elements. Each element of the matrix corresponds to the probability that a particle from the region of the plasma corresponding to that column will intersect with the detector within the range of incident angles corresponding to that row. As can be seen in Fig. (4.1) the matrix will typically be rectangular, with many more columns than rows. As the number of elements in the detector is increased, so too can the number of pixels into which the plasma is divided, such that the transformation matrix will always be rectangular.

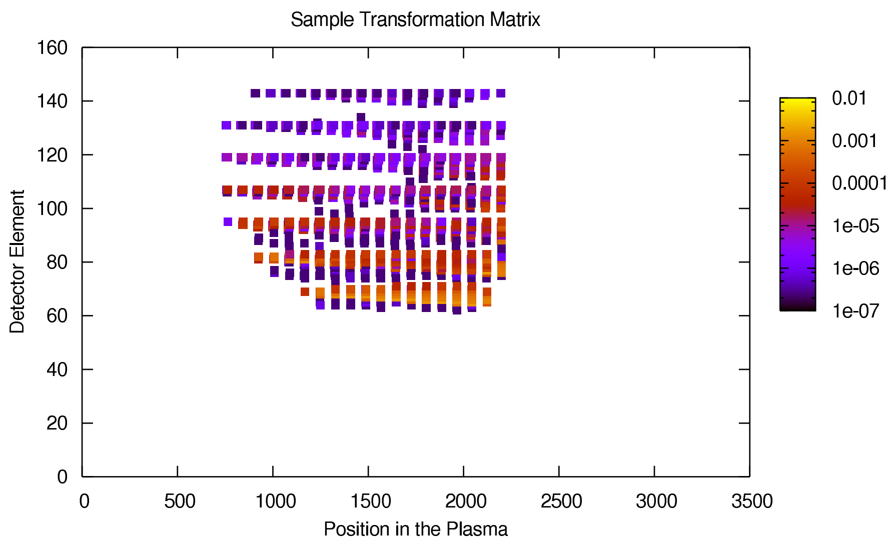


Figure 4.1: A sample matrix, for a 5cm by 5cm plasma grid and 15deg by 15deg detector elements. The plasma grid and detector element number is chosen using a Z-order curve, which rasters throught the 2D image line by line to produce a 1D vector of the pixel intensities.

4.1.1 Random Number Generation

If the initial velocity of the particles is random, within the distribution calculated in Section (2.5), we must ensure that sufficiently many particles are used such that the matrix will have converged onto a given solution. The randomness of the initial distribution must be of sufficient quality that the convergence of the solution is not over-estimated. To ensure this, the CUDA library random number generator was not used, as this is a flawed implementation of the xorwow algorithm [41]. Instead, the required random numbers were generated on the host, and passed to the device. To ensure that the initial velocity distribution is consistent with the anisotropic behaviour seen in deuterium deuterium reactions [10] a simple accept-reject algorithm is used. To do this, a random number is generated for each initial angle up to a maximum given by the maximum value in Fig. (2.5.1). If the random number chosen is lower than the corresponding height of the curve at that angle, the angle is accepted. If the random number is higher, the angle is rejected. The parent particles are assumed to be moving along the magnetic field lines, as any velocity component due to gyro-orbit motion will, for large numbers of particles, cancel out. Although each particle will have a component of velocity perpendicular to the guiding centre orbit, the average velocity for a large number of particles will be along the guiding centre. It can therefore be assumed that the orbits of the parent particles will be parallel to the magnetic field.

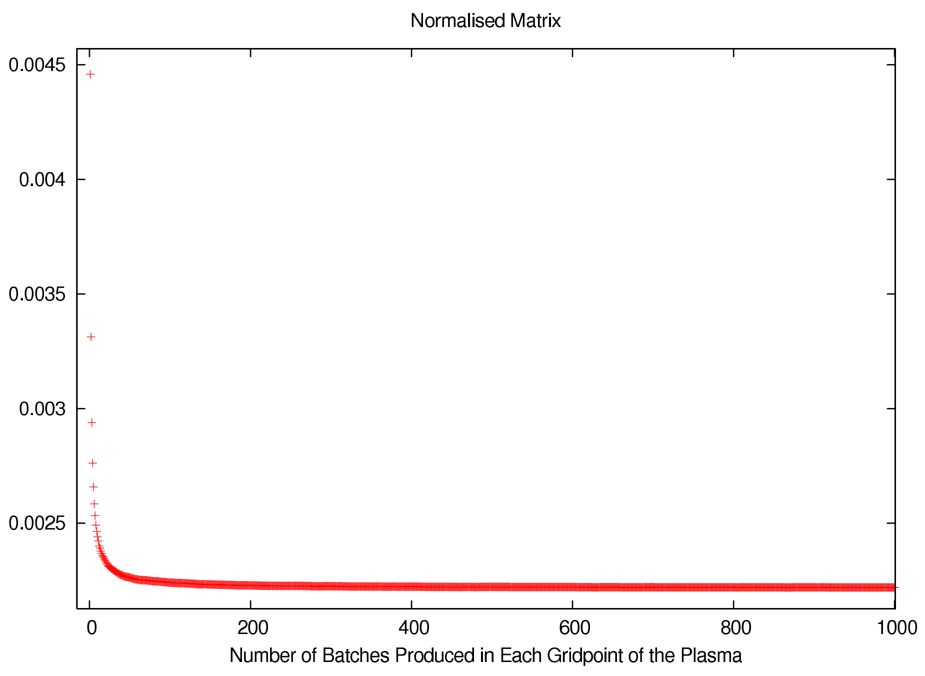


Figure 4.2: The normalised matrix value, as given in Eq. (4.1), converges after approximately 400 batches of particles are created in each grid cell of the plasma.

4.1.2 Accuracy of the Matrix

As the number of particles simulated increases, the matrix will converge towards the state that would be experienced with an infinite number of particles. To demonstrate that a sufficient number of particles are being simulated, it is important to be able to characterise the matrix, such that we can easily see the changes to it. The normalised matrix is

$$||\mathbf{M}|| = \sqrt{\sum_{ij} \left(\frac{M_{ij}}{l} \right)^2} \tag{4.1}$$

where l is the number of points in the matrix. The normalised matrix is then calculated for increasing numbers of particles. To ensure the optimal use of the CUDA card, these particles are produced in batches of 4096 particles. The matrix will converge after approximately 400 batches of particles, or 1.6×10^6 particles, are produced at each point, as shown in Fig. (4.2).

Figure (4.2) shows the normalised value of the matrix. The accuracy of this standard can found by considering the change in the normalised value of the matrix under two circumstances. First, we consider the difference between the normalised matrix calculated using a limited number of particles, and the normalised matrix calculated using a very large number of particles, to be as close to infinitely many particles as can be reasonably simulated, $||\delta\mathbf{M}||$

$$||\delta\mathbf{M}^k|| = \frac{||\mathbf{M}^k - \mathbf{M}^\infty||}{||\mathbf{M}^\infty||} \tag{4.2}$$

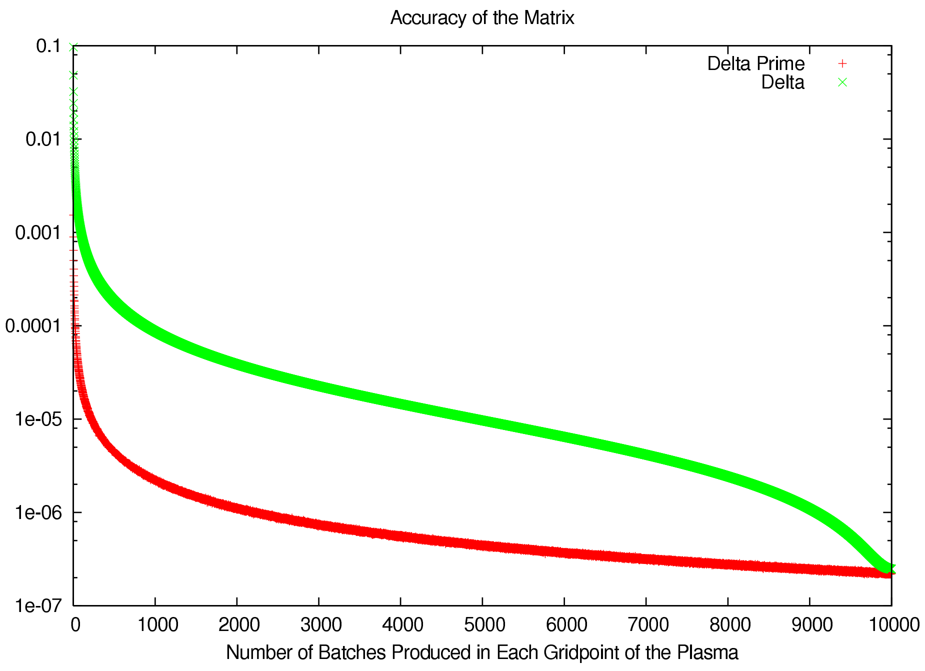


Figure 4.3: The normalised matrix is compared to both the previous matrix, and the matrix for a large number of particles as the number of particles simulated is increased. This shows that the convergence is not as fast as concluded from Fig. (4.2), and that up to 1000 batches are needed per grid cell to ensure that the change in the matrix is less than 10^{-4} from the matrix with a large number of points. Delta refers to the result from Eq. (4.2), whilst Delta prime refers to the result from Eq. (4.3).

where the superscript k denotes the number of particles created at each point in the plasma. Simulating infinitely many particles is beyond the scope of this research, and so 10,000 batches of 4096 particles each was chosen as a number that was both larger than any likely to be used, but still able to be computed in a reasonable amount of time.

Secondly, the difference between $\mathbf{M}^k$ and $\mathbf{M}^{k+1}$ is calculated, to show the speed of convergence.

$$||\delta'\mathbf{M}|| = \frac{||\mathbf{M}^k - \mathbf{M}^{k+1}||}{||\mathbf{M}^{k+1}||} \tag{4.3}$$

The change in the matrix as the number of batches is increased, and the comparison to the matrix at a very large number of particles, show that convergence is not as fast as expected from Fig. (4.2), and that whilst 200 batches per point is enough to ensure an error of less than 1% in the matrix, further gains take considerably larger numbers of particles. Batches of 1000 sets of 4096 particles are therefore used as these give a good accuracy in the matrix, whilst not being excessively onerous to compute.

4.2 Particles Moving Backwards in Time

An alternative approach to constructing the transformation matrix can come from finding the solid angle subtended by the detector elements at each point in the plasma. This solid angle, as a proportion of the whole, corresponds to the probability that a particle created in a particular area will hit the detector. Once this has been found, we can construct the transformation matrix in a similar way to the brute force method. We start by producing a large number of particles at the detector, and track the particles as they move backwards in time. At each grid point in the volume of the plasma, the velocity of any particle which passes through that grid point is recorded. Using these velocities, we can track particles forwards through the plasma, allowing the solid angle subtended by the detector to be calculated.

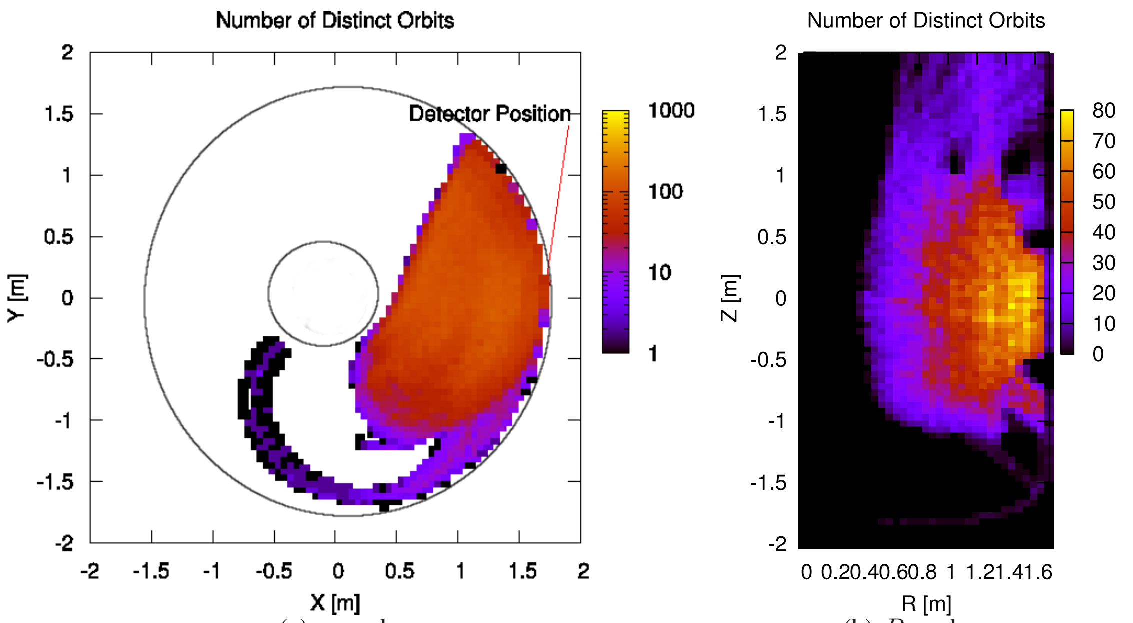
4.2.1 Solid Angle

The solid angle subtended by the detector is the region of velocity space at the point of particle creation that produces orbits which intersect the detector. The initial velocity of a fusion born proton is dependent on the velocities of its parent particles. The velocity of the centre of mass of the parent particles can be up to 80 keV, whilst that resulting from the fusion event is 3.02 MeV. The maximum inaccuracy in the initial velocity of these particles is 2.6%. It is expected that the particles will slow from injection velocities before undergoing fusion events, so the average inaccuracy will be less than this. To simplify this problem, we assume that the parents are moving parallel to the magnetic field, whilst the component of velocity from the fusion event is isotropically distributed. We are assuming that the anisotropy in the velocity of fusion products, as found by Brown and Jarmie [10], is small enough to be ignored here.

4.2.2 From the Detector to the Plasma

For each point in the plasma an initial velocity that gives an orbit which intersects with the detector must be found, or found to not exist. The easiest way to find this is to launch particles backwards in time from the detector and track their progress through the plasma. For each region of the plasma the particle passes through, we may therefore record the corresponding velocity. Although the proton production in the plasma is continuous, we divide it into pixels for ease of calculation. Each pixel intensity is considered as the average of the pixel production rate across that region. We assume that the magnetic field is toroidally symmetric, and divide the plasma into regions in x, y, and z.

Let us start with a pinhole design for the detector. The pinhole is assumed to be 1cm tall in the Z-direction, and have an angular extent of 1.47×10^{-3}radians in the ϕ-direction,



(a) x, y plane (b) R, z plane

Figure 4.4: Number of Distinct Orbits found from Backwards Particles, projected into the (X, Y) and (R, Z) planes

or approximately 1cm in the (x, y)-plane at 1.7 m from the origin. Behind the pinhole we place an array of detecting elements, and so can limit our orbits to only those that will intersect with the detecting element under consideration.

The orbits of the backwards particles are found by reversing time. At each grid point, we will find a number of particles will pass through that point. By comparing their velocities, we can find the number of distinct orbits that pass through both that point and the detector. The magnitude of the velocity is assumed to be constant, with any variation arising from numerical inaccuracy is assumed to be small. For each grid point in the plasma, any orbit with angles within 0.05 radians of a previous orbit is not counted, to prevent double counting. The distinct orbits can be seen in Fig. (4.4).

4.2.3 Ensuring that the Orbits Match

The initial points of the backwards orbits of the particles may be displaced from the final points of the forwards orbits by up to half the grid cell size. If the backwards orbit is started with the same initial velocity as recorded in the forwards orbit, but at a different position in space, the orbits will not exactly match. We may therefore have to make a small initial adjustment to the angle of the velocity, to ensure that the backwards orbit will hit the detector. Let the initial guess at the correct angles by denoted by (θ_1, ϕ_1), the position of the detector be (Z_d, ψ_d), the required changed to hit the detector be $(\delta\theta, \delta\phi)$, and the distance from the point at which the forward orbit hits the wall to the detector be $(\delta Z_d, \delta\psi_d)$. We can link all these together via a matrix equation

$$\begin{pmatrix} \delta Z_d \\ \delta \psi_d \end{pmatrix} = \begin{pmatrix} \frac{\partial Z_d}{\partial \theta} & \frac{\partial Z_d}{\partial \phi} \\ \frac{\partial \psi_d}{\partial \theta} & \frac{\partial \psi_d}{\partial \phi} \end{pmatrix} \begin{pmatrix} \delta \theta \\ \delta \phi \end{pmatrix} \tag{4.4}$$

We would like to know $\delta\theta$ and $\delta\phi$. To do this, we can simply take the inverse of the matrix. We approximate the gradients of Z_d and ψ_d by producing two new orbits, changing θ_1 to θ_2 in the first, and ϕ_1 to ϕ_2 in the second. The distance from the intersection of the orbit with the wall to the detector is given by (Z_1, ψ_1) in the first case, and (Z_2, ψ_2) in the second. We can therefore approximate as follows

$$\frac{\partial Z_d}{\partial \theta} = \frac{Z_2 - Z_1}{\theta_2 - \theta_1} \tag{4.5}$$

and similarly for the rest of the elements of the matrix.

We can then iterate through this process to find the initial velocity of an orbit which hits the detector from the region of interest in the plasma.

Newton-Raphson Method in Two Dimensions

We have a problem that is equivalent to a root finding problem in two dimensions, with two variables. Let the orbit tracking be a function, f, with independent variables θ, ϕ, and dependant variables Z, ψ, being the distance from the point of intersection with the wall of the device to the detector. If the change is small, we can expand $\mathbf{f}$ using Taylor series

$$f_i(\mathbf{x} + \delta\mathbf{x}) = f_i(\mathbf{x}) + \sum_{j=1}^{2} \frac{\partial f_i}{\partial x_j} \delta x_j + O(\delta\mathbf{x}^2) \tag{4.6}$$

where $i = 1, 2$, $f_1 = Z$, $f_2 = \psi$, $x_1 = \theta$, $x_2 = \phi$.

We can form a Jacobean matrix, $\mathbf{J}$ via

$$J_{ij} \equiv \frac{\partial f_i}{\partial x_j} \tag{4.7}$$

The desired endpoint is that the orbit should intersect with the detector, i.e. that $\mathbf{f}(\mathbf{x} + \delta\mathbf{x}) = \mathbf{0}$. Our Taylor expansion is only valid near to $\mathbf{x}$, so $\delta\mathbf{x}$ must be small. We can therefore neglect terms of order $\delta\mathbf{x}^2$ and higher. This gives

$$\delta\mathbf{x} = -\mathbf{J}^{-1}\mathbf{f} \tag{4.8}$$

This problem is not efficiently stated for calculation on CUDA devices. The overheads of parallelisation on CUDA devices is such that for calculation of small numbers of orbits it is more efficient to run such calculations on the host device. The orbit-finding code is therefore run on the host device, before the corrected orbit is passed to the CUDA device for calculation of the solid angle.

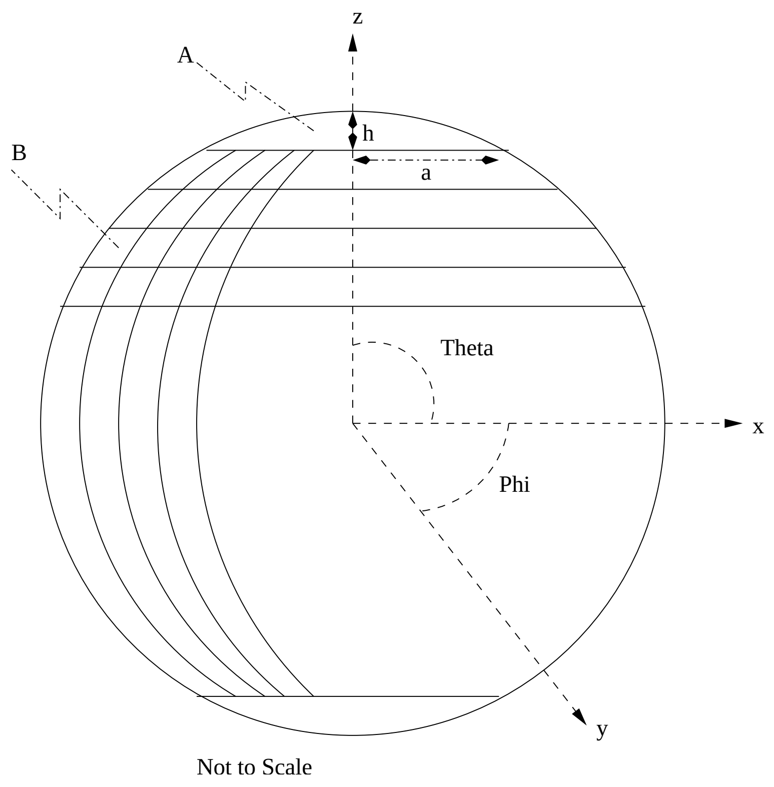


Figure 4.5: Schematic for the division of velocity space into regions of equal surface area.

4.2.4 Dividing the Velocity Space

We can consider the initial velocity space of fusion born proton to be a sphere, with the change in initial velocity from co- and counter-rotating ion collisions included later as influences to the orbit, rather than distortions to the sphere. The direction of the velocity can be described in terms of the angles θ and ϕ, where $0 \leq \theta < \pi$ and $0 \leq \phi < 2\pi$.

We would like to convert the velocity sphere into a number of regions, each with equal surface area. This allows us to find the approximate solid angle of the detector without having to find all the edges of the solid angle subtended by the detector, a computational saving.

To convert a sphere into regions of equal surface area, we start by deciding to split the sphere into regions of equal angle $\delta\theta$. This splits the sphere into rings, each ring subtending the same angle in θ. Each ring can then be divided into a number of regions in ϕ, each region having the same surface area as the uppermost ring. A schematic for this is shown in Fig. (4.5). The intention is that the regions labelled A and B should have the same surface area. This is analogous to placing a large number of evenly spaced points over the surface of a sphere, which is a hard problem. If the surface area of the regions is allowed to vary, this becomes a much easier problem. As a result, these regions have similar, but not exactly the same, surface areas.

We can find the surface area of the region labelled A in Fig. (4.5) because it forms a spherical cap. The surface area of a spherical cap can be calculated via

$$S_A = 2\pi r h \tag{4.9}$$

where S_A is the surface area of the region A, r is the radius of the sphere, and h is the height of the spherical cap. The height of the spherical cap is related to the radius of the sphere via the radius of the base of the cap, a, where

$$r = \frac{a^2 + h^2}{2h} \tag{4.10}$$

We can find a by comparison to our chosen pixel size in θ

$$a = r\sin(\delta\theta) \tag{4.11}$$

Assuming that the velocity has been normalised, and substituting equations (4.11) and (4.10) into equation (4.9) gives a formula for the surface area of the cap

$$S_A = 2\pi\left(1 + \sqrt{1 - \sin(\delta\theta)}\right) \tag{4.12}$$

We now have the problem of how to divide further rings in ϕ to ensure that the regions are of equal surface area. For each ring, the surface area of the whole ring can be calculated by increasing $\delta\theta$ in equation (4.12). The area of the previous cap can be taken from this area, giving the area of the ring. We know that each region should be approximately the same area as the first cap, and so we can simply divide the area of the ring by the area of the first cap. We then round this number to the nearest integer, which gives the change in ϕ needed to retain approximately equal surface areas for each region. We can therefore characterise each ring in terms of the angle θ.

To find the extent of the detector in velocity space, we can then rotate the subdivided sphere such that the initial angle, $(\theta, \phi) = (0, 0)$, and test the velocity of each region in turn. The initial region, as formed by the first spherical cap, will always contain a velocity giving an orbit that intersects with the detector, and so gives the minimum solid angle subtended by the detector. The size of $\delta\theta$ should therefore be chosen to ensure that the change in subtended solid angle along the orbit is significant. Along the length of an orbit, the angle subtended by the detector will range from large, when the distance along the orbit to the detector is small, to small, when the distance along the orbit to the detector is large. This can be compared to the chance that a particle born in a particular position along the orbit will hit the detector. Close to the detector, the chances of hitting the detector will be higher than if the particle is born far from the detector. However, too small a $\delta\theta$ will result in greater computational cost. The required accuracy must therefore be balanced against the time required for the computation. The full extent of the detector can be found by testing the velocity for each region in a ring, stopping when no region of the ring gives an orbit which intersects with the detector. A more efficient method of performing this calculation is shown in Section (4.2.5).

The rotation of the velocity sphere such that the axis of rotational symmetry, $i.e.$ the z-axis of the unrotated sphere, is aligned to the angle of the velocity of the orbit known to intersect with the detector, is done by transforming the coordinates of the velocity space from spherical polar to Cartesian coordinates, performing the rotation, and then converting back to spherical polar coordinates. We start with the angles of the known orbit, θ and ϕ, and the angles of the region of the divided sphere, $\delta\theta$ and $\delta\phi$. We are trying to find the Cartesian coordinates that correspond to the angles associated with a particular region of the divided sphere, which is itself aligned to some particular angle. First we convert the region's angles to Cartesian coordinates

$$\begin{aligned} x &= \cos(\delta\phi)\sin(\delta\theta) \\ y &= \sin(\delta\phi)\sin(\delta\theta) \\ z &= \cos(\delta\theta) \end{aligned}$$

The radius of the sphere is assumed to be unitary, and is therefore neglected. Secondly, we rotate these angle around the y axis by θ, to get the declination

$$\begin{aligned} x' &= z\sin(\theta) + x\cos(\theta) \\ y' &= y \\ z' &= z\cos(\theta) - x\sin(\theta) \end{aligned}$$

Thirdly, we rotate about the z axis by ϕ

$$\begin{aligned} x'' &= x\prime\cos(\phi) - y\prime\sin(\phi) \\ &= (\cos(\delta\theta)\sin(\theta) + \cos(\delta\phi)\sin(\delta\theta)\cos(\theta))\cos(\phi) - \sin(\delta\phi)\sin(\delta\theta)\sin(\phi) \\ y'' &= x\prime\sin(\phi) + y\prime\cos(\phi) \\ &= (\cos(\delta\theta)\sin(\theta) + \cos(\delta\phi)\sin(\delta\theta)\cos(\theta))\sin(\phi) + \sin(\delta\phi)\sin(\delta\theta)\sin(\phi) \\ z'' &= z\prime \\ &= \cos(\delta\theta)\cos(\theta) - \cos(\delta\phi)\sin(\delta\theta)\sin(\theta) \end{aligned}$$

Finally, these are converted back into spherical polar coordinates, to be used as the initial velocity of the proton in the test orbit.

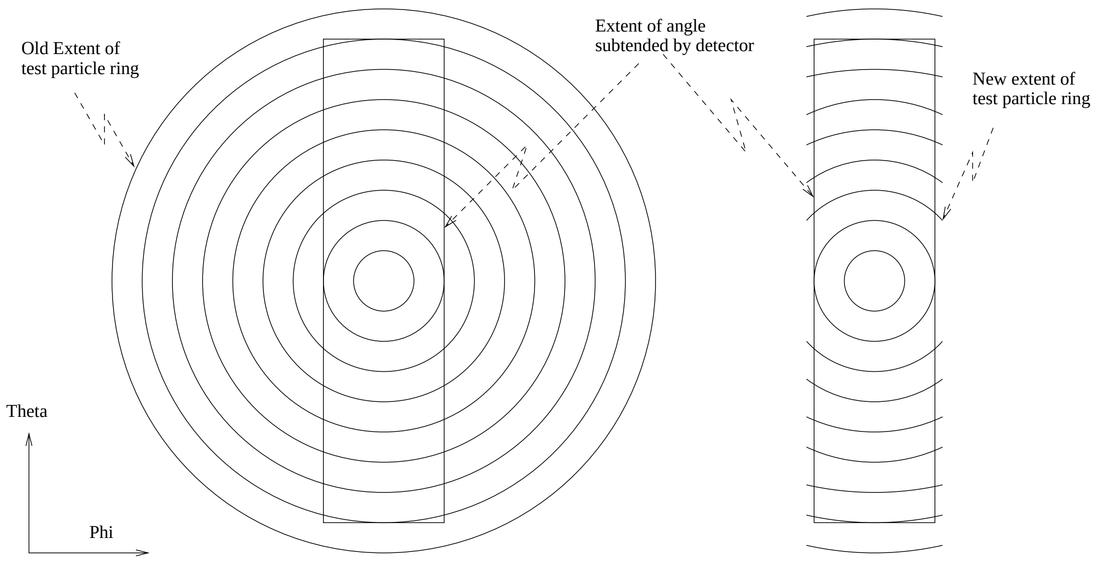


Figure 4.6: The reduction in the required computation to find the solid angle subtended by the detector is demonstrated by the reduced lengths of the test particle rings shown to the right of the image, as compared to the full rings on the left.

4.2.5 Refinements to the Constant Surface Area Method

The angle subtended by the detector tends to be long and thin, by which I mean that the extent in θ_{velocity} is much greater than the extent in ϕ_{velocity}. It quickly becomes inefficient to calculate test orbits for all the points in each circle of $\delta\theta$. We would therefore like to only produce test orbits for the area slightly overlapping the detector.

First, the edges of the detector must be found. In any given ring, there will be either two or four edges. Knowing the edges in one ring allows us to limit our test orbits in the next ring to only those angles that are similar to the successful orbits of the previous ring. Finding the edges is done first by calculating all the orbits for an early ring. Taking each orbit in turn, we can find the angles which correspond to the edge of the detector. For each subsequent orbit, the probable extent of the detector is known, and so it is possible, by checking an extra orbit to either side of the previous extent, find the new extent, as shown in Fig. (4.6). This is far more computationally efficient than calculating the orbits for the complete ring. In most cases, the extent of the detector can be found by considering the four edge angles, $\phi_1, \phi_2, \phi_3,$ andϕ_4, where

$$0 \leq \phi_1 \leq \phi_2 \leq \phi_3 \leq \phi_4 \leq 2\pi \tag{4.13}$$

The detector extent in ϕ for this ring is therefore given by

$$\phi_D = (\phi_2 - \phi_1) + (\phi_4 - \phi_3) \tag{4.14}$$

There is a complication that arises if the angle subtended by the detector includes $\phi = 0$, in which case

$$\Omega = ((2\pi - \phi_4) + \phi_1) + (\phi_3 - \phi_2) \tag{4.15}$$

4.2.6 Variable Surface Area

The solid angles subtended by the detector at different positions along the length of a single orbit can cover several orders of magnitude. The solid angle subtended by the detector at the point just before the orbit intesects with the wall will be large, whilst the solid angle subtended at the far end of the orbit will be small. If we divide the surface area of the velocity space sphere into equal parts, we must make those parts small enough that variation can be seen in adjacent points along the length of the orbit, even when the total solid angles subtended at each of those points is small. If we are to also test every region of solid angle both within the area subtended by the detector, and adjacent to the edge of that area, this can take considerable amounts of computation, especially if the area subtended by the detector is large. We must therefore have a scheme which will distinguish between the smallest surface areas subtended by the detector, but also be fast to compute for the largest solid areas. This can be done by abandoning the fixed size for the pixels of solid angle. Instead, we have a fixed number of pixels per ring. This means that the rings closest to the top of our velocity sphere have pixels that are small, and so we can distinguish between the solid angles of detectors far along the orbits, which are likely to be small. As the solid angle subtended increases, the pixel size will increase, so the amount of calculation needed for larger detectors is reduced. This also allows easier optimisation of the CUDA code, as each ring can be calculated as a single kernel, as opposed to trying to calculate the optimum number of orbits per kernel for each ring. This is the scheme currently used.

4.2.7 Jacobian Calculation of Solid Angle

An alternative approach to calculating the solid angle is to find the Jacobian matrix, $\mathbf{J}$ of an orbit which hits the detector.

$$\mathbf{J} = \begin{pmatrix} \frac{\partial Z}{\partial \theta} & \frac{\partial Z}{\partial \phi} \\ \frac{\partial \psi}{\partial \theta} & \frac{\partial \psi}{\partial \phi} \end{pmatrix} \tag{4.16}$$

This, coupled with the size of the detector, allows us to find the change in angle corresponding to a change in the position of intersection with the wall

$$\delta Z \, \delta\psi = |\mathbf{J}| \delta\theta \, \delta\phi \tag{4.17}$$

where δZ and $\delta\psi$ are the dimensions of the detector. The solid angle, Ω is therefore

$$\Omega = \delta\theta\,\delta\phi \sin\theta_0 = \frac{\delta Z \delta\psi}{|\mathbf{J}| \sin\theta_0} \tag{4.18}$$

where θ_0 is the initial angle of the orbit which intersects the detector. This assumes a linear relationship between the initial angle and the final position at the wall. This is not true in all cases, and so this technique is not used.

4.3 Image Resolution

The fusion-born particles will be created in a continuum in space. We choose a length scale over which to pixelate the plasma. This length scale is chosen to maximise the useful information we can extract from the detector image, whilst limiting the errors introduced. Ideally, the detector would also be a continuum, allowing us to directly compare detector image to plasma. However, economic and physical constraints on the detector force us to consider areas of the detector as single elements. We must therefore consider the length scale of events in the plasma that may affect the proton production distribution. For a typical MAST plasma, the Alfvén velocity is of the order of 10^6 ms^{-1}. The Alfvén eigenmodes will typically have corresponding wavelengths up to $\sim$ 20m [17], of which perturbative AEs will have wavelengths of between 2 and 20m. The toroidal modes with mode numbers n=1-3 have a radial width which is comparable to the drift orbit of the fast ions when these ions have slowed to the Alfvén speed [42]. These modes have a radial width of $\sim$ 5cm, and are a major contributor to the fast particle transport. Being able to image the plasma on 5cm lengthscales would therefore be useful. Over longer timescales, chirping, sawtooth, and fishbone modes lead to fast particle loss with features again having $\sim$ 5cm lengthscales [12]. It was decided to consider detectors in terms of the extent of each pixel in the θ and ϕ directions. The detectors record the incoming protons in terms of their incident angle. Detectors were considered that have a range of resolutions, from $6\,\mathrm{deg}$ to $45\,\mathrm{deg}$. As the length scale of the plasma image is increased, the inversion of the associated transformation matrix will move from underspecified to over specified. For MAST shot 18808, assuming, for the sake of simplicity, that the plasma is toroidally symmetric, the plasma will have 516 5cm by 5cm regions within the LCFS. Although this assumption can be relaxed, the numerical calculations will then take considerably longer. A detector with a $6\,\mathrm{deg} \times 6\,\mathrm{deg}$ pixel size will have 900 pixels, making this an over-specified problem, although not all the detector pixels will record data, and not all the points in the plasma will contribute points that hit the detector. The results of this inversion are shown in Fig. (4.7(a)). This reconstruction is indistinguishable from the known distribution, which is to be expected for an over specified problem. We can therefore move to smaller grid sized in the plasma, allowing more information to be extracted, until the image starts to degrade, as shown in Figs. (4.7(b)), (4.7(c)).

4.4 Detector Design and Calibration

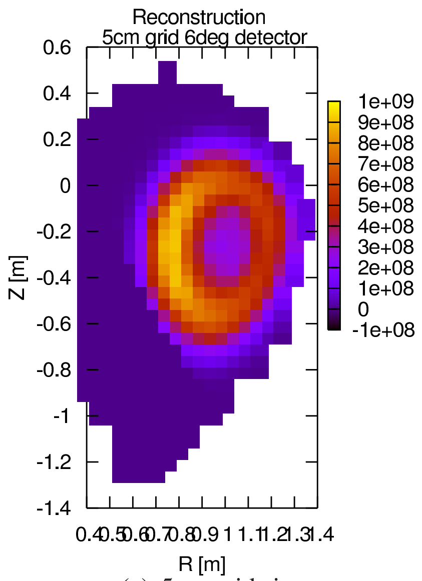
The detector is based on a 1cm by 1cm square pinhole, with an array of detector elements mounted 1cm behind it. The detector will be positioned at the midplane, at $R = 1.7$ m, corresponding to a position mounted on the MAST reciprocating arm.

The expected proton flux can be calibrated by comparison to the neutron flux in the device. The total proton production will be comparable to the total neutron production, typically $\sim 10^{13}$ s^{-1} [46]. The production of neutrons along particular lines of sight is also known, due to measurements by the neutron camera [13]. This allows comparison to both the total expected flux and the flux at particular locations within the plasma.

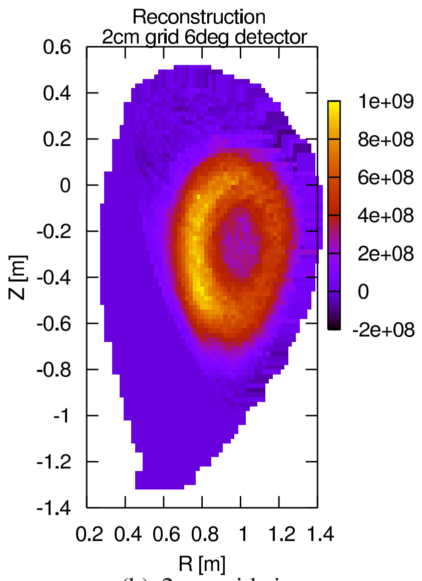
We can compare the brute force and backwards calculations by checking that the probability of a particle hitting this detector from a particular point in the plasma is consistent, shown in Fig. (4.8). This means that we calculate the matrix $\mathbb{M}$ via both methods, and chack that the outputs are similar. Neither of these codes creates the detector image. The two are currently not directly comparable, as the back tracking code doesn't include the energy of the centre of mass of the particles, but they do show sufficient similarities to allow some confidence in the results.

4.5 Accuracy of Orbit Tracking Code

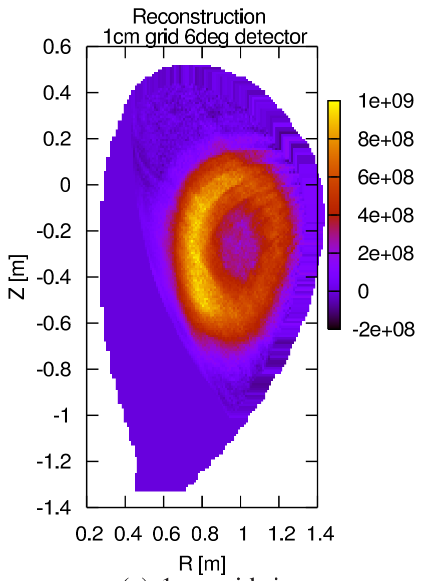
There must be a balance between the speed of calculation of the code, and the accuracy of the results obtained. As we desire to calculate the orbits as quickly as possible, we must therefore decide what is the minimum accuracy required to give reasonable results. The relationship between accuracy and computational time is generally an inverse one, in as much as increased accuracy will require longer times to calculate. The level of accuracy needed will depend on the exact device under consideration, for example the dimensions of the detector will determine the level of accuracy needed in the calculation of the final point of the orbit. There are two main parameters governing the level of accuracy in the calculation. The first is the time step used in the orbit, given in terms of the fraction of the time taken to complete a single Larmor orbit. The second parameter is the size of the magnetic field grid. This is particularly important in the CUDA code, as the CUDA devices offer a hardware-accelerated bilinear interpolation of magnetic field data. This allows interpolation at the same speed as a simple matrix look up on the CUDA device, but is limited to linear interpolation only. As a result of this, the more points there are in the magnetic field data, the more accurate the CUDA code will be. However, as there is only limited memory on the device, larger magnetic field matrices will take space that



(a) 5cm grid size

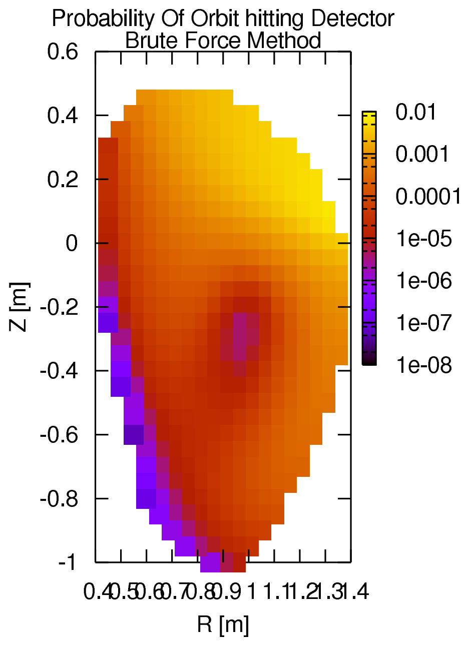


(b) 2cm grid size

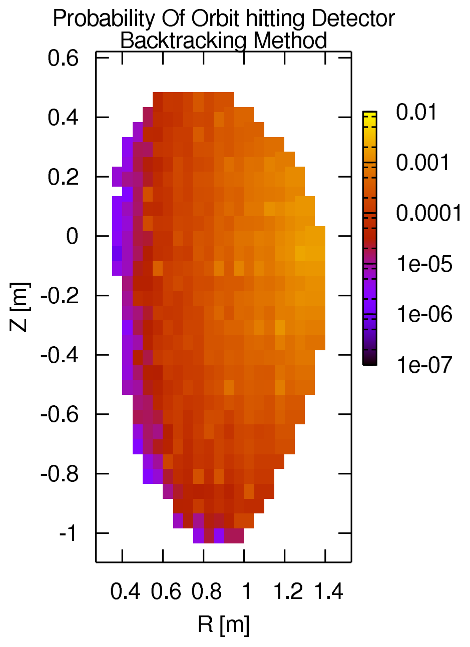


(c) 1cm grid size

Figure 4.7: Reconstruction of the proton production distribution for a variety of grid sizes, with each detector pixel corresponding to incident angles in a 6° by 6° range. This shows the decrease in quality of reconstructions as the number points in the plasma increases.



(a) Brute Force Method



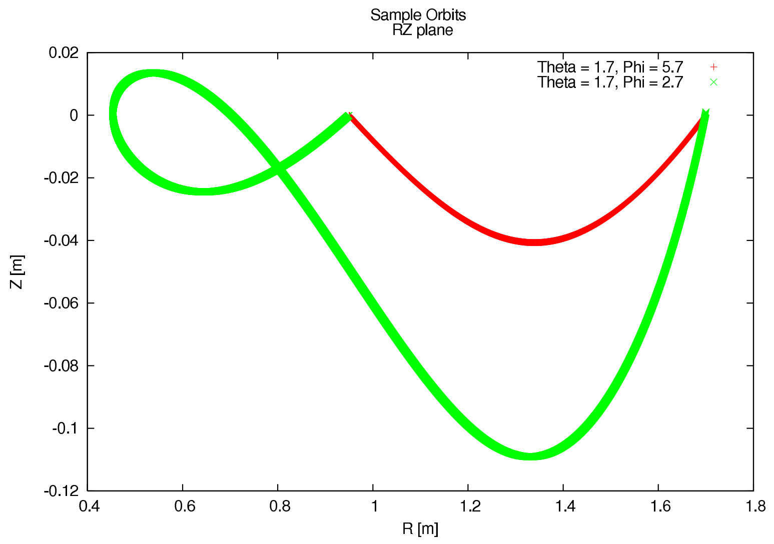
(b) Backwards Orbit Tracking

Figure 4.8: The probability of a particle from each point in the plasma hitting the detector. The two codes are not directly comparable, as the backwards code doesn't include the energy of the centre of mass of the parent particles.

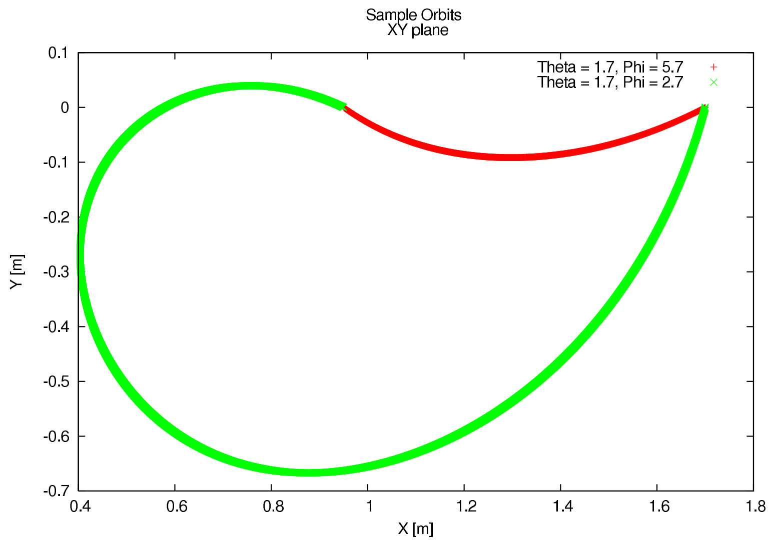
would otherwise be used to hold particle position calculations. The fewer particles that can be run in each CUDA kernel, the longer the code will take to calculate equivalent numbers of particle orbits.

We must now choose the required accuracy of our system. The detector design under consideration has a 1cm square pinhole, centred at $X = 1.7$, $Y = 0.0$, $Z = 0.0$, but following the curve of the wall. We therefore choose to have a minimum accuracy in the position of the intersection of the orbit with the wall of the device of $\approx 2.5\text{mm}$. In addition, we consider detectors with between 3600 and 25 elements, corresponding to an angular resolution of $3\,\text{deg}$ to $35\,\text{deg}$. Our system should therefore have a minimum accuracy of $2\,\text{deg}$ in incident angle of orbit to detector.

The length of the orbit will also determine the accuracy of the result. A particle which is lost to the plasma on its first orbit will be considerably more accurately simulated than one which experienced multiple orbits before it hits the wall. We must therefore consider the accuracy as a function of orbit length. The test initial positions and orbits have been chosen to demonstrate the accuracy for orbits that are at least half the length of a Larmor orbit, as well as demonstrating the accuracy for both first orbit losses and multiple orbit losses. The test particles are created with initial positions and angles as shown in Tab. (4.1). Sample orbits can be seen in Fig. (4.9).



(a) In RZ plane



(b) In XY plane, i.e. by considering the projection of the orbit from above.

Figure 4.9: Sample orbits used to test the accuracy of the codes, showing a first orbit loss in red and a multiple orbit loss in green.

X [m]	Y [m]	Z [m]	θ [rad]	ϕ [rad]
0.75	0.0	0.0	1.6470	3.088791
0.95	0.0	0.0	1.70895	5.7202
0.95	0.0	0.0	1.6957	2.74475

Table 4.1: The initial positions and angles of the sample orbits used in calculating the accuracy of the orbit following code.

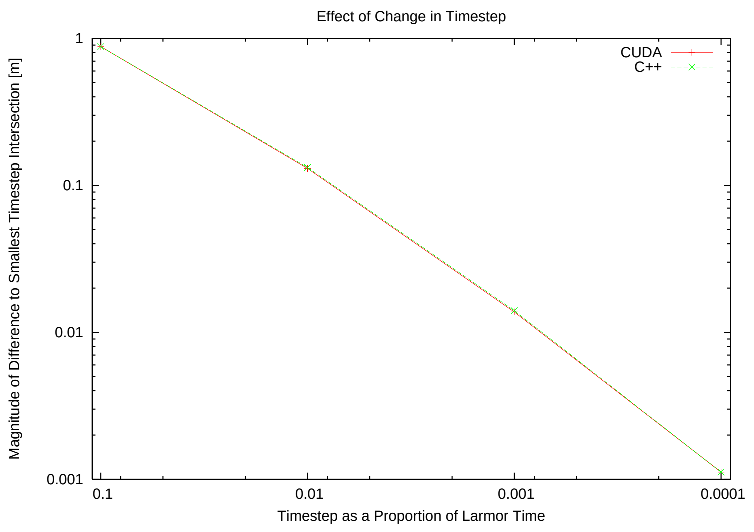


Figure 4.10: The change in position of the point of intersection of the orbit with the wall at various time steps. To achieve the desired accuracy of 2.5 mm the timestep can be no larger than $0.0001\times$Larmor time.

4.5.1 Time step

The change in the accuracy of the final position for both CUDA and C++ codes is shown in Fig. (4.10). If we are to achieve the desired accuracy of 2.5mm at the point of intersection, we must use a time step that is no larger than $0.0001\times$ Larmor time. If we consider the change in the intersection angle as a result of changing the time step, as shown in Fig. (4.11), we can see that to achieve the desired accuracy of 2° the time step can be as large as $0.009\times$Larmor time. However, as the time step need for accuracy in the position is smaller, we will use that time step.

4.6 Magnetic Field Grid Size

We would like to know the smallest magnetic field data grid that will give satisfactory accuracy. We can compare the results of the CUDA orbits with the C++ orbits to find the accuracy of the magnetic interpolation. The different sizes of the magnetic fields are generated by using a C++ code to resample the 65×65 data file provided by EFIT from shot 18808 on MAST. As such, the bicubic spline interpolation is assumed to be perfectly accurate, at least in comparison to the linear interpolation.

It can be seen that certain sizes of magnetic field give more accurate results than would be expected from the number of points in the magnetic field grid. This is probably due to certain sizes fitting more easily into the memory of the device. CUDA, to use the device memory more efficiently, aligns the data in memory with the size of the memory by padding the data. This may affect the hardware accelerated interpolation, making certain sizes of magnetic field matrix more efficient and accurate. This effect can be seen in

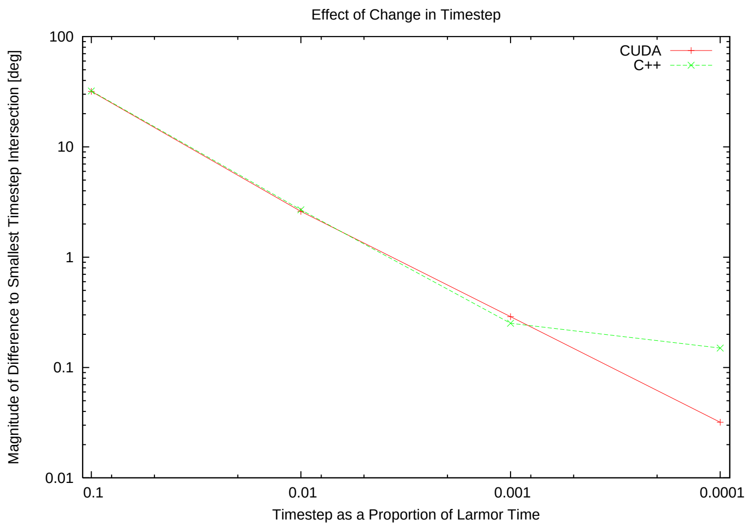


Figure 4.11: The change in angle of intersection of the orbit with the wall at various time steps. The orbit is the sample orbit from $X = 0.75, Y = 0.0, Z = 0.0$. To achieve the desired accuracy of $2°$ the timestep must be no larger than $0.009\times$ Larmor time.

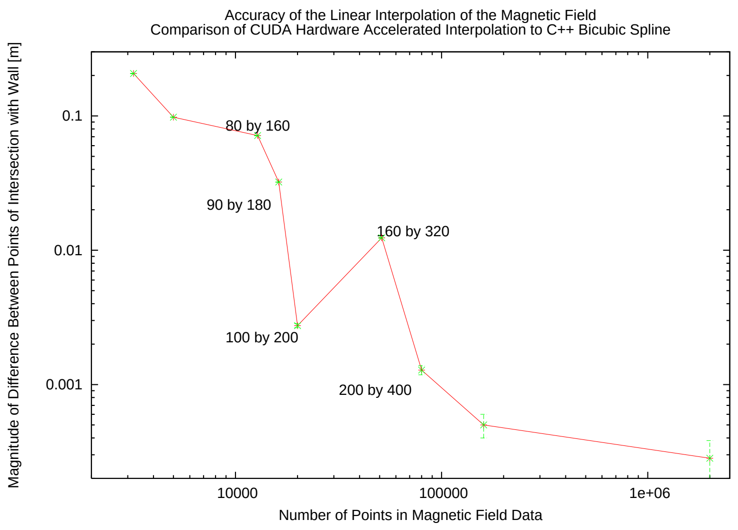


Figure 4.12: The accuracy of the orbit as a function of the number of points in the magnetic field. The decrease in accuracy in point 160 by 320 is due to the hardware interpolation of the magnetic field being more accurate if there is less padding in the array sizes.

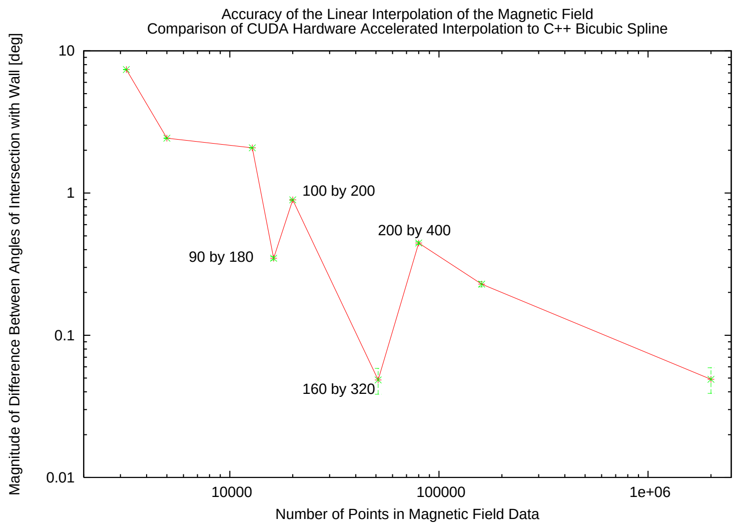


Figure 4.13: The accuracy of the intersection angle of the orbits with the wall as a function of the number of points in the magnetic field.

Fig. (4.12). From this, we can see that to achieve the desire accuracy, with a maximum displacement in final position of 2.5mm, the magnetic field should be 100×200. The magnetic field is otherwise assumed to be toroidally symmetric.

The accuracy of the intersection angle required can be achieved by the 50×100 magnetic field. This data set, however, does not have the required accuracy in the position. We therefore use the 100×200 magnetic field data, as this allows us to achieve the desired accuracy in both the position of the intersection with the wall, and the angle of intersection with the wall.

4.7 Effect of Displacement of the Initial Position of the Orbit on the Point of Intersection with the Wall

A further problem, particularly with the backwards/forwards orbits approach, is that by considering orbits that pass through regions of space, rather than particular points, the initial point of the forwards orbit will be displaced from the point in the backwards orbit giving the velocity used for the forward orbit. It is important to choose a grid size that will minimise this inaccuracy, whilst still allowing computation of the translation matrix in reasonable time. Here, it is easier to consider C++ and CUDA codes separately, as it is not expected that one will be more accurate than the other.

As can be seen in Figs. (4.14), (4.16), even a change in initial position of as little as 1cm will lead to an inaccuracy greater than our desired minimum accuracy of 2.5mm in position and 3 degrees in incident angle. This dependency of the orbit on initial position shows how the backwards/forwards orbit method, which should be considerably more accurate than the forward method alone, may suffer. The requirement that a successful

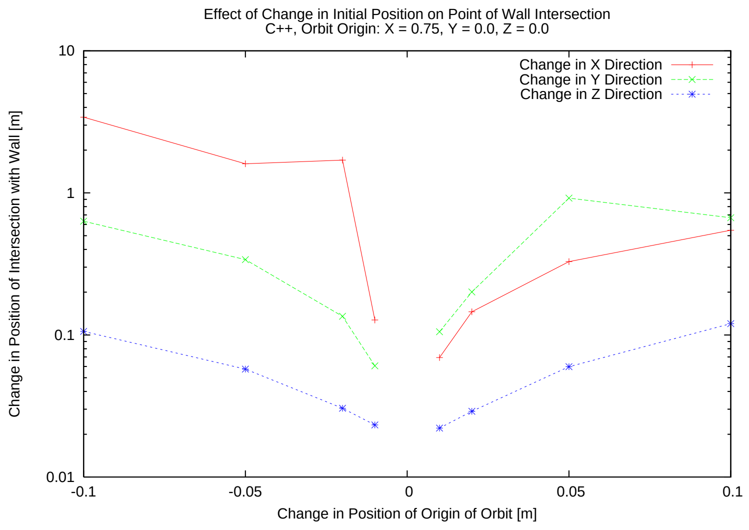


Figure 4.14: The change in the position of the intersection of the sample orbits due to a change in the initial position of the orbit. The initial velocity of the orbit is unchanged.

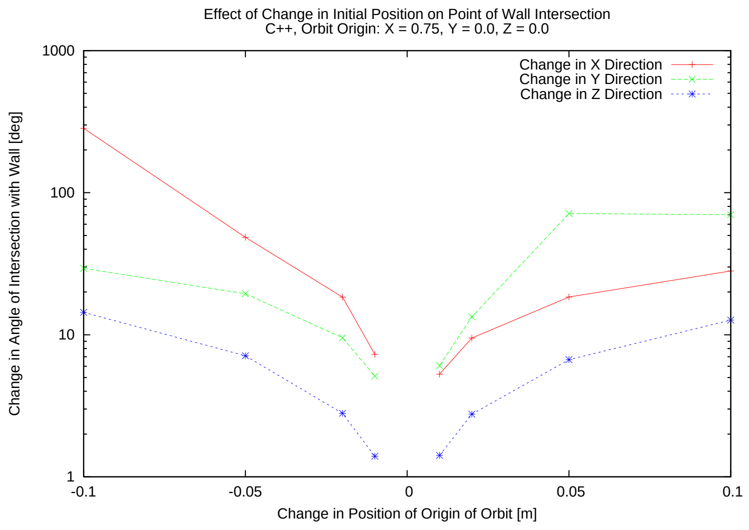


Figure 4.15: The change in angle of intersection of the sample orbits with the wall due to a change in the initial position of the orbit, as calculated using C++ code. The initial velocity of the orbits is unchanged.

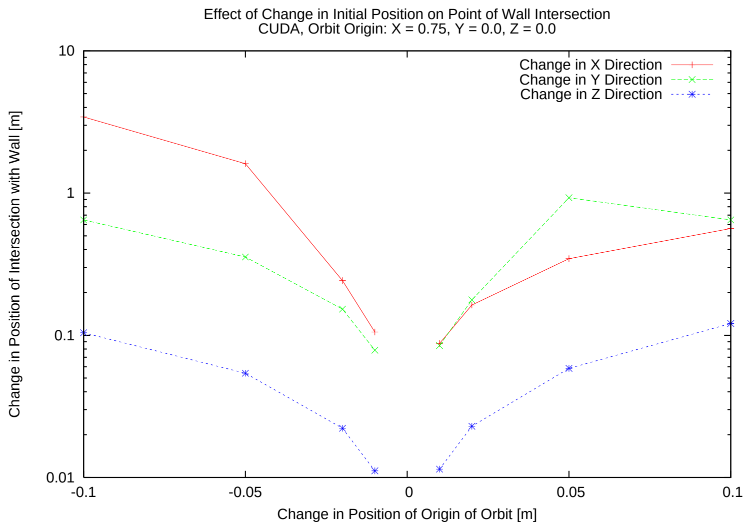


Figure 4.16: The change in position of the intersection of the sample orbit with the wall due to a change in the initial position of the orbit as calculated using the CUDA code. The initial velocity of the orbit is unchanged.

forward orbit be found, even with the initial guess of a successful backwards orbit, adds an additional computational constraint to the code, and one which is not amenable to CUDA parallelisation, unlike the orbit tracking.

The inaccuracy is just as apparent in the CUDA orbit tracking code, as can be seen in Fig. (4.16). We can see that the changes in the position and angle of intersection with the wall are more dependant on the displacement in the X, Y plane than in the Z direction. This may allow us to consider a grid with longer grid cells in the Z direction than in the X, Y directions.

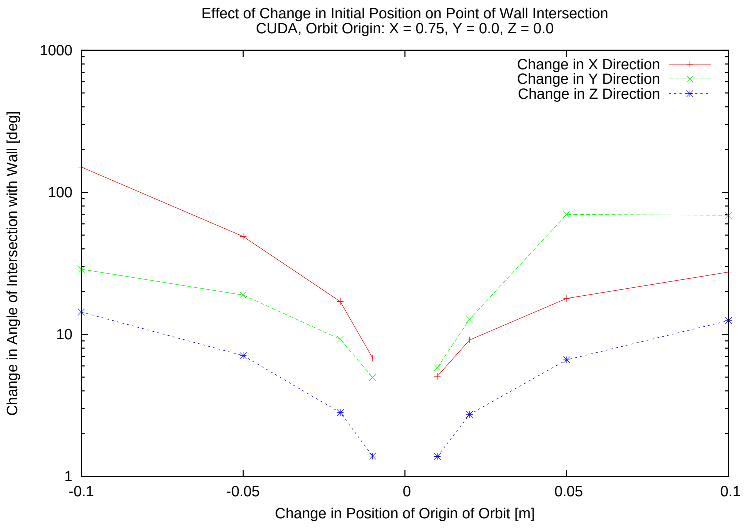


Figure 4.17: The change in the angle of intersection of the sample orbit with the wall due to a change in the initial position of the orbit, as calculated using the CUDA code. The initial velocity of the orbit is unchanged.

Chapter 5

Image Reconstruction

The proton production distribution in the plasma can be found by finding the inverse of the transformation matrix, and applying this to the measured detector image

$$\mathbf{y} = \mathbf{M}\mathbf{x} \tag{5.1}$$

$$\mathbf{x} = \mathbf{M}^{-1}\mathbf{y} \tag{5.2}$$

Where $\mathbf{y}$ is the detector image, reduced to a vector via a z-curve method, and $\mathbf{x}$ is the projection of the fast particle distribution in the (R, Z) plane, also reduced to a vector via a z-curve method. To find $\mathbf{x}$ from $\mathbf{y}$ requires that the matrix be invertible. As the matrix is rectangular, a pseudo-inverse is found instead. This is done using Singular Value Decomposition (SVD). The image found from Eq. (5.2) will probably not be exactly the proton distribution. To ensure that the reconstruction has the minimum of assumptions associated with it, the entropy of the reconstructed image is maximised, subject to the constraint that the image produced by applying Eq. (5.1) be consistent with the image recorded by the detector.

5.1 Reduced Transformation Matrix

The matrix produced in Chapter (4), an example of which is shown in Fig. (4.1), shows that there exist areas of the plasma which do not contribute particles which hit the detector. In addition, there are detector elements which do not record any particle impacts. These areas of the plasma and detector elements can be removed from the matrix, leaving a reduced matrix which still contains all the available information about the plasma. An example of the reduced matrix is shown in Fig. (5.1).

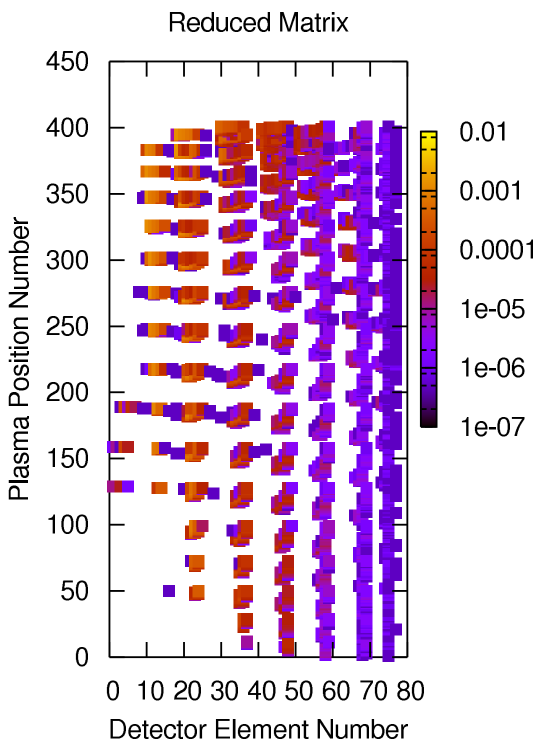


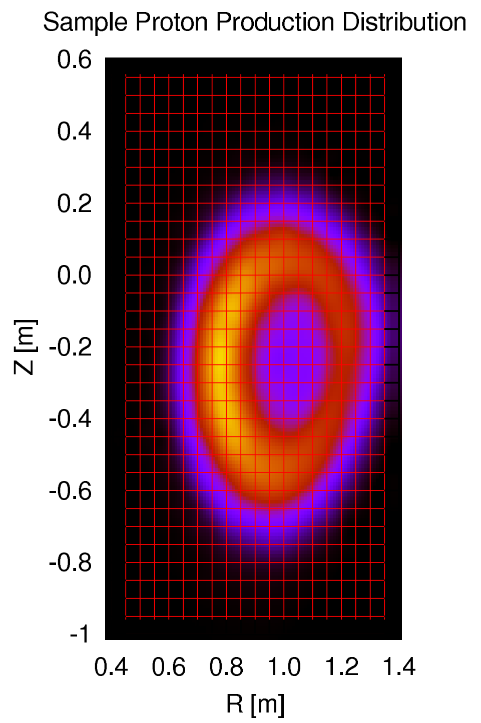
Figure 5.1: The reduced form of the transformation matrix, where only the rows and columns with non-zero elements are retained.

5.1.1 Plasma Image

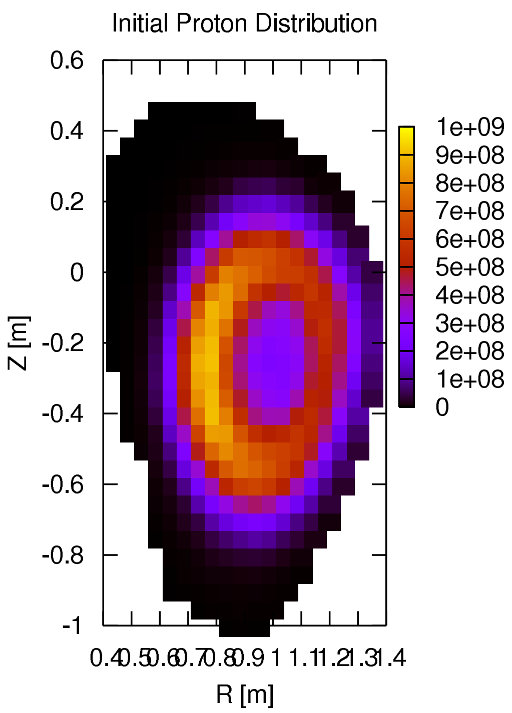
Although the proton production in the plasma is continuous, we divide it into pixels for calculation. Each pixel intensity is considered as the average of the proton production rate across that region. We assume that the plasma is toroidally symmetric for computational simplicity, and divide it into regions in the poloidal, or R, Z-plane, as shown in Fig. (5.2). The total image of the plasma can then be described as a one dimensional vector of the intensities, $\mathbf{x}$. Each component of $\mathbf{x}$ can then be associated with the R, Z-coordinate of the pixel. When dealing with the reduced matrix, the pixels which do not contribute to the detector are removed, leaving only those containing orbits that coincide with the detector. The plasma image may be reconstructed by recording which rows of the matrix have been retained, and the associated position of those rows in the R, Z-plane

5.1.2 Detector Image

The image at the detector is recorded as as series of pixel intensities, $\mathbf{y}$. The incident angle of the protons determines which detector element records their arrival, creating an image in angles that are broadly equivalent to toroidal and poloidal directions of the tokamak, as shown in Fig. (5.3). The two dimensional image that is constructed from the incoming protons is then converted to a one dimensional vector. This vector is then reduced to the length of the columns of the reduced matrix.

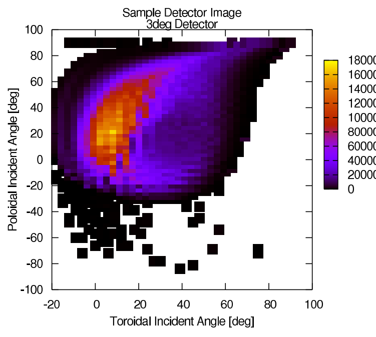


(a) Before pixelation

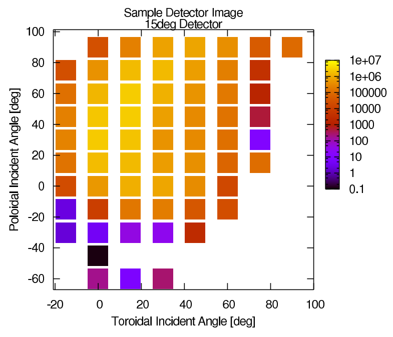


(b) After pixelation

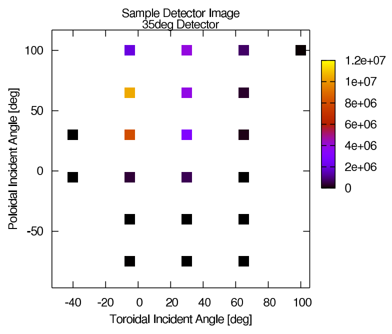
Figure 5.2: Example proton production distribution



(a) 60 by 60 Detector Elements



(b) 12 by 12 Detector Elements



(c) 5 by 5 Detector Elements

Figure 5.3: Detector Images from Sample Proton Distribution shown in Fig. 5.2. This detectors have a 1cm^2 pinhole, and are positioned at R=1.7, Z=0 within the device.

5.2 Singular Value Decomposition

We may combine the proton distribution, $\mathbf{x}$ and the detector image $\mathbf{y}$ with the the transformation matrix calculated using the CUDA code, we have a simple expression for the plasma image

$$\mathbf{M}\mathbf{x} = \mathbf{y} \tag{5.3}$$

where $\mathbf{M}$ is the transformation matrix, $\mathbf{x}$ is the proton production distribution in the plasma, and $\mathbf{y}$ is the detector image.

If $\mathbf{M}$ was directly invertable, we could then simply find the proton production distribution via

$$\mathbf{x} = \mathbf{M}^{-1}\mathbf{y} \tag{5.4}$$

However, we will always divide the plasma into the greatest number of pixels possible, as shown in Section (5.1.1). This ensures that there will be more points within the last closed flux surface than in the detector image, i.e. that there are more elements in $\mathbf{x}$ than $\mathbf{y}$. This gives us an under-specified problem. Because we have chosen $\mathbf{x}$ and $\mathbf{y}$ in this way, the transformation matrix $\mathbf{M}$ will not be square. As such, we cannot find an inverse of the matrix such that all the information is retained. We must therefore calculate the Moore-Penrose pseudo-inverse of $\mathbf{M}$ to extract as many clues as to the real proton production distribution as possible. To calculate this pseudo-inverse, we have chosen to use the Singular Value Decomposition (SVD) technique.

The SVD algorithm being used is an iterative QR algorithm, from the ALGLIB library [4].

5.2.1 QR algorithm

The QR algorithm for the singular value decomposition of a rectangular matrix, $\mathbf{M}$, has two phases [14]. Initially, we compute orthogonal matrices $\mathbf{U}_1$ and $\mathbf{V}_1$ such that

$$\mathbf{B} = \mathbf{U}_1^T\mathbf{M}\mathbf{V}_1 \tag{5.5}$$

where $\mathbf{B}$ has non-zero elements on the diagonal and first super-diagonal only. This is done via Householder transformations of the matrix $\mathbf{M}$. We then compute two further orthogonal matrices, $\mathbf{U}_2$ and $\mathbf{V}_2$, again via Householder transformations of $\mathbf{B}$, such that

$$\mathbf{W} = \mathbf{U}_2^T\mathbf{B}\mathbf{V}_2 \tag{5.6}$$

where $\mathbf{W}$ is diagonal and non-negative. The diagonal elements, w_i, of $\mathbf{W}$ are the singular values of $\mathbf{M}$. Furthermore, we can find the right singular vectors via

$$\mathbf{U} = \mathbf{U}_1\mathbf{U}_2 \tag{5.7}$$

and the left singular vectors via

$$\mathbf{V} = \mathbf{V}_1\mathbf{V}_2 \tag{5.8}$$

where the columns of $\mathbf{U}$ and $\mathbf{V}$ are the right and left singular vectors, respectively.

From our SVD decomposition of the transformation $\mathbf{M}$ we therefore get the three matrices

$$\underbrace{\mathbf{M}}_{a\times b} = \underbrace{\mathbf{U}}_{a\times a}\underbrace{\mathbf{W}}_{a\times b}\underbrace{\mathbf{V}^{\mathbf{T}}}_{b\times b} \tag{5.9}$$

where $\mathbf{U}$ and $\mathbf{V}$ are both unitary matrices, such that

$$\mathbf{U}\mathbf{U}^{\mathbf{T}} = \mathbf{U}^{\mathbf{T}}\mathbf{U} = \mathbf{U}^{-1}\mathbf{U} = \mathbf{I} \tag{5.10}$$

and similarly for $\mathbf{V}$. $\mathbf{W}$ is directly invertable, being diagonal. From this, we can construct the pseudo-inverse

$$\underbrace{\mathbf{M}^{-\mathbf{1}}}_{b\times a} = \underbrace{\mathbf{V}}_{b\times b}\underbrace{\mathbf{W}^{-1}}_{b\times a}\underbrace{\mathbf{U}^{\mathbf{T}}}_{a\times a} \tag{5.11}$$

We must pay careful attention to the inverting of the diagonal matrix, $\mathbf{W}$. If there are errors in the small elements of $\mathbf{W}$, these can have a disproportionate impact on the accuracy of $\mathbf{W}^{-\mathbf{1}}$, and the impact of these errors increases as we approach the limits of numerical accuracy for the machine. We must therefore take the decision to truncate the non-zero elements of $\mathbf{W}$ at some value that prevents these numerical errors. All elements of $\mathbf{W}$ with values smaller than some w_{min} are then set to zero. w_{min} is therefore chosen to be the limit of numerical accuracy of the machine running the code. This may also, if w_{min} is set too high, lose useful information in the inversion. In addition, if, as in the case currently under consideration, a is larger than b, we will be unable to recover all the information in $\mathbf{x}$ when we take the pseudo-inverse of $\mathbf{M}$. We must therefore make a series of assumptions about the nature of $\mathbf{x}$ to maximise the utility of our image.

We can safely assume that there will be no point in the plasma that will have a negative production rate of protons. The energy of the fusion produced protons is great enough that the plasma can be considered to be non-collisional, as shown in section 1.3.1, so any absorption of protons is likely to be small compared to the total production. We therefore assume that $\mathbf{x}$ cannot be negative at any point. A more important assumption is that the proton production distribution will be smoothly varying.

For noisy data, there will be many possible proton production distributions that satisfy Eq. (5.3). By maximising the entropy of the test image, whilst ensuring that the detector data is unchanged, we can select the test image that requires the fewest assumptions to reproduce. The entropy of the image, as calculated in Section (5.3), is a measure of the smoothness of this distribution, where distributions with higher entropies will be smoother.

5.2.2 Accuracy

The accuracy of the singular value decomposition is almost entirely governed by the decision of the minimum eigenvalue to consider. For underspecified problems, we will ordinarily be able to find the number of eigenvalues we expect, and utilise them all. However, very small eigenvalues, which are close to the limits of the accuracy of the computing device, will still need to be ignored. The cut-off used is

$$w_{\text{min}} = 1 \times 10^{-15} \tag{5.12}$$

The value used is an absolute value, chosen such that the potential rounding error in calculation using 32-bit floating point numbers is eliminated.

5.3 Maximum Entropy

5.3.1 Theory

The theory behind the maximum entropy interpretation of data sets is usually presented by considering the problem of monkeys throwing balls into boxes. If we have a team of monkeys tossing N identical balls into M boxes, the number of balls in the j^{th} box, n_j, or the intensity of the j^{th} pixel is given by

$$n_j = N f_j \tag{5.13}$$

where f_j is the probability that a ball will be thrown into box j. We can then say that the number of ways any given combination of balls in boxes can occur is

$$g = \frac{N!}{\Pi_1^M n_j!} \tag{5.14}$$

We can then define the entropy of the image as

$$\begin{aligned}
S &\equiv \ln g & (5.15)\\
&= \ln\left(\frac{N!}{\Pi_1^M n_j!}\right) & (5.16)\\
&= \ln(N!) - \ln(\Pi_1^M n_j!) & (5.17)\\
&= \ln(N!) - \sum_1^M (\ln(n_j!)) & (5.18)
\end{aligned}$$

at this point we apply Stirling's approximation

$$\begin{aligned}
S &\approx N\ln(N) - N - \sum_1^M (n_j \ln(n_j) - n_j) & (5.19)\\
&= N\ln(N) - \sum_1^M (n_j \ln(n_j)) \text{ As } \textstyle\sum_1^M n_j = N & (5.20)\\
&= N\ln(N) - \sum_1^M (N f_j \ln(N f_j)) & (5.21)\\
&= N\ln(N) - \sum_1^M (N f_j \ln(N) + N f_j \ln(f_j)) & (5.22)\\
&= N\ln(N) - N\ln(N) - \sum_1^M (N f_j \ln(f_j)) \text{ As } \textstyle\sum_1^M f_j = 1 & (5.23)\\
&= -N \sum_1^M (f_j \ln(f_j)) & (5.24)
\end{aligned}$$

Our efforts to maximise this entropy should generate the image that is maximally ignorant. By this, we mean that the selected distribution is one that makes the least claim of being informed beyond the stated prior data. In the above example, there is no prior data, and so the entropy will be maximised by a uniform, or flat, distribution. We have no data about the likelihood of a monkey throwing a ball into a particular box, and so should consider all boxes equally likely to be the recipient of a ball.

5.3.2 Implementation

The entropy of an image with M pixels is defined [19] as

$$S = \sum_{i=1}^{M} -p_i \log(p_i) \tag{5.25}$$

where p_i is the probability that a particular pixel has a particular proton production rate, $p_i = x_i n_i / x_t ot$. Here x_i is the proton production rate per unit area of the plasma,

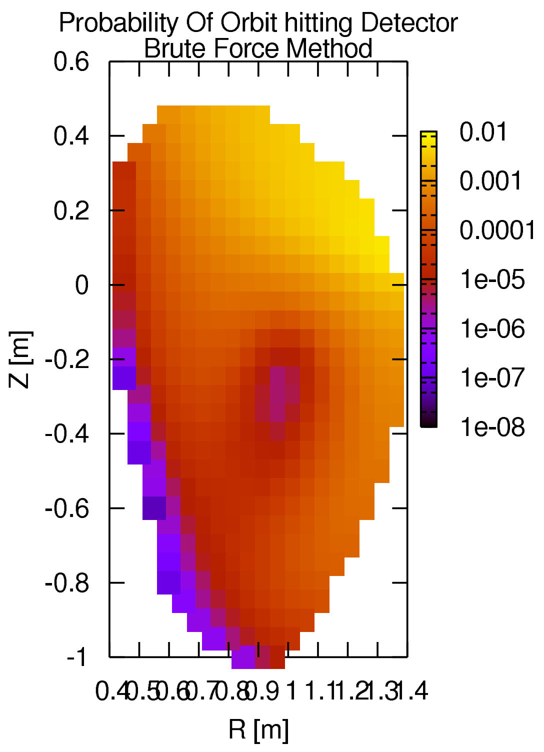


Figure 5.4: The probability of a particle born in each position in R, Z hitting the detector. The large variation in probabilities should be noted.

considered in the R, Z plane, and $x_{\text{tot}} = \sum_i x_i$. $p_i = x_i/N$. It is important that $\sum_i^N p_i = 1$, and that $0 < p_i \leq 1$ for all p_i. For the problem under consideration we are using

$$S = \sum_{i=1}^{M} \begin{cases} -\frac{x_i n_i}{N} \log\left(\frac{x_i n_i}{N} + \epsilon\right) & \text{if } \frac{x_i ni}{N} \geq 0 \\ -\frac{x_i n_i}{N} \log(\epsilon) & \text{if } \frac{x_i ni}{N} < 0 \end{cases} \tag{5.26}$$

where x_i is the proton production rate in that area of the plasma, n_i is the corresponding probability that a proton from that region of the plasma will hit the detector, shown in Fig. (5.3.2), N is the total number of particles that are recorded at the detector, and ϵ is a small positive constant to ensure that, should x_i be negative, although this is unphysical in a system without particle sinks, we can still calculate the entropy of the image. If we differentiate equation 5.25 with respect to p, we get

$$\frac{\partial S}{\partial p_i} = -\log(p_i) - \frac{p_i}{p_i} \tag{5.27}$$

$$= -\log(p_i) - 1 \tag{5.28}$$

whilst if we perform the same operation on $S = \sum_i^N -p_i \log(p + \epsilon)$

$$\begin{aligned}\frac{\partial S}{\partial p_i} &= -\log(p_i+\epsilon) - \frac{p_i}{p_i} && (5.29)\\ &= -\log(p_i+\epsilon) - 1 && (5.30)\end{aligned}$$

So we can see that these functions are both continuous, allowing us to add the ϵ, which allows consideration of points where $x_i = 0$.

The large variation in the probabilities of the particles hitting the detector leads to potential problems when calculating the entropy of the image. The positions with a large probability of hitting the detector have a comparatively larger influence on the resulting entropy. However, as these areas which contribute most particles to the detector image, all else being equal, this means that biasing the entropy calculation in favour of these areas can be seen as beneficial. The areas with low probability of hitting the detector can vary by much larger amounts, and still have the same total impact on the entropy as the high probability areas. This can lead to a much lower influence on the results from these areas, which in turn means that the variation in these areas is less well suppressed by the entropy maximisation.

5.4 Constructing the Code

We now have most of the basic building blocks of an analysis code that will allow us to get the maximum information from each of our detector images. This code is independent of the method used to construct the matrix, $\mathbf{M}$. We shall now outline the code, as it is expected to be run. The first part of the code is making sure that we have the data we need, and we shall therefore read in the detector image, $\mathbf{y}$, the transformation matrix, $\mathbf{M}$, and the corresponding list of pixel positions within the last closed flux surface which will allow us to output the results in the R, Z plane.

The first step is to use SVD to create a pseudo-inverse of $\mathbf{M}$, as outlined in Section (5.2). We must choose $w_{\min}$ such that we preserve as much information as possible without introducing numerical errors. Typically, a value of 10^{-15} is used. This gives an initial guess at the proton production distribution, $\mathbf{x}_0$, as well as the matrix $\mathbf{V}$, which can be split into orthogonal vectors spanning the basis of $\mathbf{M}$.

We then find those vectors in $\mathbf{V}$ that are in the kernel of $\mathbf{M}$, and we are expecting to find R of these, such that

$$\mathbf{M}\mathbf{v}_j = 0 \tag{5.31}$$

where $1 \leq j \leq R$. We can then add a combination of the vectors $\mathbf{v}_j$ to $\mathbf{x}_0$ to create a proton distribution that will comply with our assumptions, whilst still satisfying the condition that $\mathrm{M}\mathbf{x} = \mathbf{y}$

$$\mathbf{x} = \mathbf{x}_0 + \sum_j \lambda_j \mathbf{v}_j \tag{5.32}$$

where λ_j is the magnitude of the contribution of each $\mathbf{v}_j$. We now have two conditions to satisfy, namely that all the points in $\mathbf{x}$ are positive, and that the entropy of the resultant image is maximised. We have a simple method of changing $\mathbf{x}$ without changing our image in $\mathbf{y}$, via choosing the values of λ_j. We have chosen a combination of unconstrained and constrained optimisation algorithms to produce the desired result.

5.5 Non-linear Conjugate Gradient Optimisation

Let the function we are trying to minimise be $f(x_{(i)})$. We find the gradient of this function, such that

$$d_{(0)} = r_{(0)} = -f'(x_{(0)}) \tag{5.33}$$

We then have to find the values of $\alpha_{(i)}$, where $\alpha_{(i)}$ are scalars, that minimises

$$f(x_{(i)} + \alpha_{(i)} d_{(i)}) \tag{5.34}$$

by ensuring that the gradient is orthogonal to the search direction. We can then iterate

$$\begin{aligned} x_{(i+1)} &= x_{(i)} + \alpha_{(i)} d_{(i)} && (5.35)\\ r_{(i+1)} &= -f'(x_{(i+1)}) && (5.36)\\ \beta_{(i+1)} &= \frac{r^T_{(i+1)} r_{(i+1)}}{r^T_{(i)} r_{(i)}} && (5.37)\\ d_{(i+1)} &= r_{(i+1)} + \beta_{(i+1)} d_{(i)} && (5.38) \end{aligned}$$

The stopping conditions are either the difference between the solutions of the current iteration and the previous iteration is small enough, or some maximum number of iterations is reached.

5.6 Unconstrained Optimisation

The first part of the optimisation process is to choose the values of λ_j such that all the points in $\mathbf{x}$ are non-negative. This is done using an unconstrained conjugate gradient method. The function to be minimised is the absolute sum of the negative elements of

$\mathbf{x}$, where $\mathbf{x}$ is calculated as in Eq. (5.32). The values of λ_j are varied, according to the ALGLIB CG-optimisation method. The gradient of the function is calculated numerically.

5.7 Constrained Optimisation

The next part of the optimisation is to maximise the entropy whilst ensuring that all the points of $\mathbf{x}$ remain non-negative. We therefore have a linear constraint, and a pair of boundary constraints governing the size of λ_j. To simplify these constraints, we will introduce a variable, $\mathbf{w}$, such that

$$\mathbf{w} = \sum_j \lambda_j \mathbf{v_j} \tag{5.39}$$

We must now find the direction of $\mathbf{w}$ which is in the direction of the steepest descent of the entropy, as defined in equation 5.26. We must therefore find the maximum of $\mathbf{w} \cdot \nabla S$, subject to the linear constraint that $\mathbf{w} >= -\mathbf{x}$, and the boundary constraints that $-\lambda_{\text{max}} <= \lambda_j <= \lambda_{\text{max}}$. This is done using the LP-SOLVE library, based upon the revised simplex method. Once we have found the step direction we can choose the step size. To do this, we introduce a variable governing the step length, μ, such that

$$\mathbf{x}^i = \mathbf{x}^{i-1} + \mu \mathbf{w} \tag{5.40}$$

We can use a simple 1d optimisation scheme to choose μ such that we maximise the entropy of $\mathbf{x}^i$. This is done by starting from some small, fixed value of μ, μ_0. This is added to the current best guess of μ. If the entropy of the resulting $\mathbf{x}$ is increased, we double the step size. If the entropy is less than before, we reset the step size to μ_0.

At the end of this process, we will have selected from the class of possible $\mathbf{x} = \mathbf{M^{-1}y}$, the $\mathbf{x}$ that satisfies the constraint that all the elements of $\mathbf{x}$ are non-negative, and that has the highest entropy.

5.8 An Alternative Iterative Entropy Maximisation Scheme

The inversion of the transformation matrix can be computationally intensive for large matrices. This is especially true for systems with few detector elements, which will have transformation matrices which are extremely non-square. Ideally, a maximum entropy scheme would not require the inversion of this matrix. However, this is reasonably quick for the matrices under consideration, where most ofthe detectors have less than 300 elements, and the plasma is split into 5cm $\times$ 5cm regions.

We shall therefore apply Bayesian probability theory to the problem. We can write the posterior probability function as [24]

$$p(\mathbf{x}|\mathbf{y},\sigma,I) = \frac{p_p(\mathbf{x}|I)\cdot p_l(\mathbf{y}|\mathbf{x},\sigma,I)}{p_e(\mathbf{y}|I)} \tag{5.41}$$

where σ is the error, and I is the background expert knowledge, $p_p(\mathbf{x}|I)$ is the prior probability density function, $p_l(\mathbf{y}|\mathbf{x},\sigma,I)$ is a likelihood probability density function, and $p_e(\mathbf{y}|I)$ is a normalisation factor of the posterior probability function.

We should like to choose the prior probability density function such that the entropy of $\mathbf{x}$ is maximised. This function is therefore

$$p_p(\mathbf{x}|I) = \left(\frac{\alpha}{2\pi}\right)^{\frac{N}{2}} \exp(\alpha S) \tag{5.42}$$

where α is a regularisation coefficient, which can be viewed as a competition factor between the entropy and the χ^2 of the measured data. S is the entropy, as found in Eq. (5.25).

The likelihood probability density function gives the probability that the detector image is given by the Eq. (5.3). The reconstructed detector image is compared to the measured detector image via

$$p_l(\mathbf{y}|\mathbf{x},\sigma,I) = \frac{1}{\Pi_{k=1}^{N}\sqrt{2\pi\sigma_k}}\exp(-\frac{1}{2}F) \tag{5.43}$$

where

$$F = \sum_{k=1}^{M}\left(\frac{y_k - \sum_{i=1}^{N} M_{ij}x_i}{\sigma_k}\right) \tag{5.44}$$

which is a modified χ^2 cost function, and $p_e(\mathbf{y}|I)$ is a normalisation factor of the posterior probability function.

The entropy is a convex function and F is a convex ellipsoid, which means that there must be a global maximum of the function

$$Q(\mathbf{x},\mathbf{y}) = \left(\frac{S}{S_{\text{initial}}}\right) - \frac{1}{2}\left(\frac{F}{F_{\text{initial}}}\right) \tag{5.45}$$

where the initial values of S and F are the values for the first guess of the iteration. The values of the pixels can then be updated via

$$x_i^{(n+1)} = x_i^{(n)}\left(1 + \alpha^{(n)}\frac{\partial S}{\partial x_i} - \beta^{(n)}\frac{\partial F}{\partial x_i}\right) \tag{5.46}$$

where α and β are arbitrarily chosen, and their size reduced each iteration to ensure that the effect of overshooting the minimum is minimised. The method of choosing $\alpha^{(n)}, \beta^{(n)}$ is described in Section (5.8.2). The problem remains of whether there should be a correction factor relating S to F. The simplest case omits this factor, whilst using the initial values

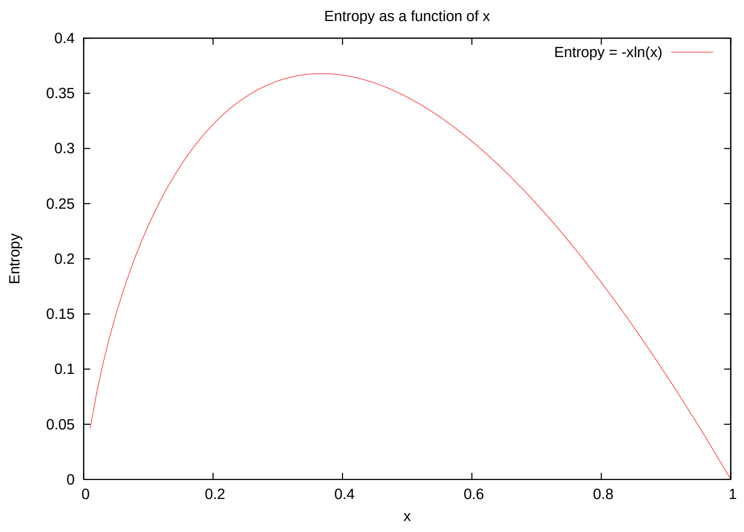


Figure 5.5: The entropy of a particular pixel, demonstrating the convex shape of the function when attempting to maximise the entropy. x is the probability of the pixel having a particular value.

of S and F prevents problems arising from one being much larger than the other.

5.8.1 Maximum Entropy as a Convex Function

The entropy of a particular point, as defined in Eq. (5.25), forms a convex curve, as shown in Fig. (5.5). The maximum entropy that can be achieved for a large ensemble of points will therefore be when all the points are equal, known as the ”grey” solution, because it corresponds to an image with all the intensities at some low value. Image inversion via maximum entropy will therefore tend to underestimate peaks, and overestimate troughs.

Similarly, minimising the F value of the data will lead to underestimation of peaks, as this is also a convex function, as shown in Fig. (5.6). As both of the conditions we are trying to meet are convex functions, we know that there will exist some global maximum. This is complicated by the introduction of errors in data. In this instance, we would not expect to maximise the entropy subject to the constraint that the F is zero, but rather that it is small.

5.8.2 An Iterative Minimum Finder for Non-Differentiable Convex Functions

We can exploit the convex shape of F and the entropy to easily find the values of α and β that will minimise the F and maximise the entropy. Initially, we are more interested in minimising the F value of our test distribution of initial proton positions. We will therefore set α to zero, and attempt to minimise the F value of

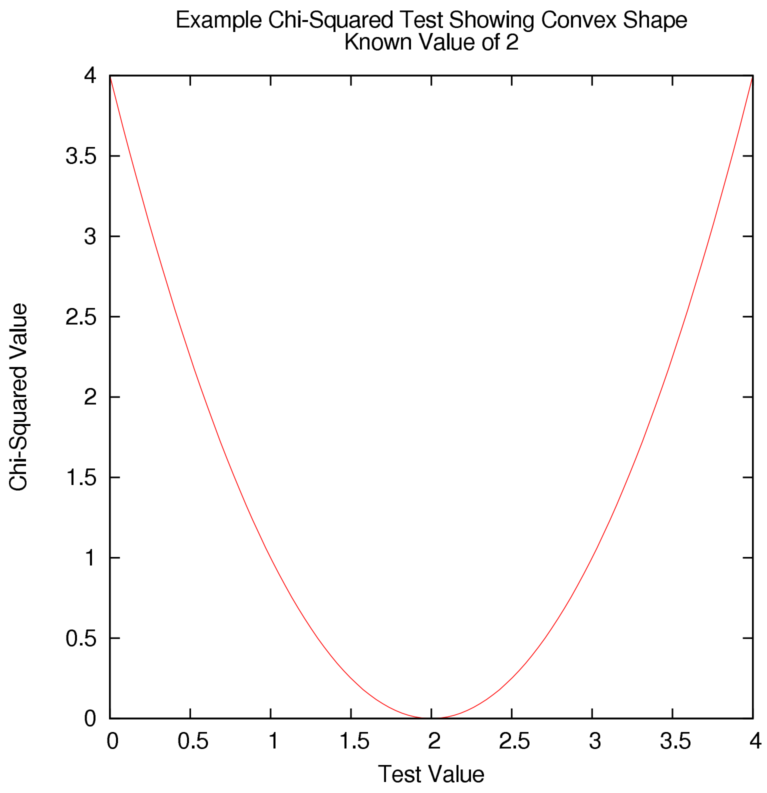


Figure 5.6: An example of the F values around a known value of 2. When minimising F of an ensemble, this is a convex function.

$$p_i^{(n+1)} = p_i^{(n)} \left(-\beta \frac{\partial F}{\partial p_i^{(n)}} \right) \tag{5.47}$$

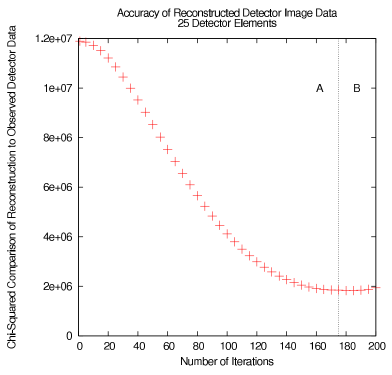
as calculated using Eq. (5.44). Our previous iteration will give us a value of F when β is zero. If we then find the value of F for a very small β, we can approximate a tangent to the curve. The gradient of the tangent will be negative if our β is smaller than the value, β_0 which gives a minimum F, and similarly if the tangent is positive, β is larger than β_0. After the initial tangent has been found, the next β can be found by finding the intersection of the tangent with $F = 0$. We can record the points of the closest approach from either side of β_0, and the tangents to the curve at both of these points. The next β to be tested is found by finding the point of intersection of the two tangents. The ending point of the scheme can be chosen as being when the difference between β_- and β_+ is less than some cut-off value. This scheme requires the calculation of F twice for each test beta, and will converge more rapidly than pattern search methods.

5.8.3 Removing Unphysical Proton Production Values

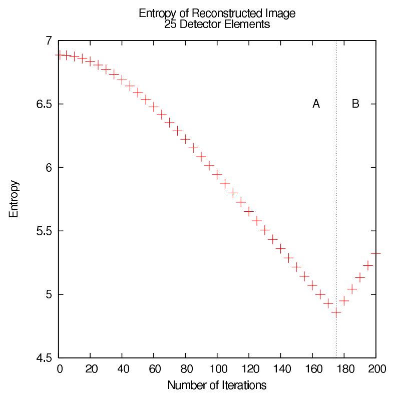
If the initial values for the pixels are chosen by multiplying the detector image by the pseudoinverse of the transformation matrix, as found by SVD, the initial χ^2 test will be close to zero. Using this as a starting point means that the algorithm converges extremely quickly. However, the values found by multiplying the pseudoinverse with the detector image may be negative, which is unphysical. These values are therefore set to zero. This will stop the reconstructed image being a solution to Eq. (5.1). The iterative scheme outlined above can be used to find the closest solution to this problem, by setting α to zero in Eq. (5.46). This iterates towards a solution with the minimum difference between the observed and reconstructed detector images. Once this has reached a minimum, α can be increased, and the iterative scheme to maximise the entropy, subject to the cost function in Eq. (5.45), used to find a reconstructed image of the plasma. The region marked ”A” in Fig. (5.7) is the region in which α is set to zero, whilst the region ”B” shows results for non-zero α and β. The values of S_{initial} and χ^2_{initial} are set as the values of S and χ^2 at the point of transition from region ”A” to ”B”. The cost function is only calculated in region ”B”, and is shown in Fig. (5.7(c)). The whole system converges in 200 iterations, with 175 iterations needed to find the minimum in χ^2. The χ^2 value found was calculated as

$$\chi^2 = (\mathbf{d} - \mathbf{M}\mathbf{x})^2 \tag{5.48}$$

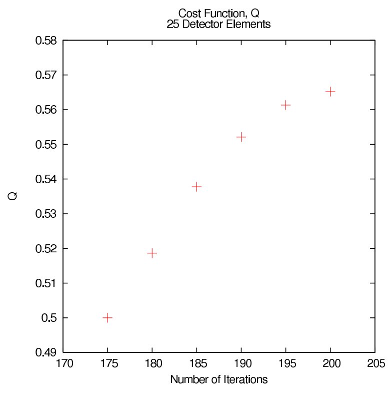
The final error should be considered against the total number of particles at the detector, which is $\sim 2 \times 10^7$, over 25 pixels. The final error of 2×10^6 in the detector image is the square of the absolute error, which is small in comparison to the total number of particles at the detector.



(a) χ^2 values at 5 iteration intervals.



(b) Entropy values at 5 iteration intervals



(c) Cost function values at 5 iteration intervals

Figure 5.7: The values of the χ^2 comparison of the reconstructed data and the observed data, the entropy of the reconstruction, and the value of the cost function are shown as the scheme iterates. In the region labelled "A", the value of α is set to zero, allowing a minimum of the χ^2 to be found. In the region labelled "B", the full iterative scheme is applied.

Chapter 6

Results

6.1 Simple Image Inversion

The simplest approach to reconstructing the proton production distribution is to find the inverse of the transformation matrix, giving

$$\mathbf{y} = \mathbf{M}\mathbf{x} \tag{6.1}$$

$$\mathbf{x} = \mathbf{M^{-1}}\mathbf{y} \tag{6.2}$$

Where $\mathbf{y}$ is the detector image, $\mathbf{M}$ is the transformation matrix mapping the initial spatial distribution of the protons onto the detector image, and $\mathbf{x}$ is the initial spatial distribution of the protons. The transformation matrix will not be directly invertable, and so the pseudo-inverse is found using Singular Value Decompostion (SVD), as described in Chapter (5). This has been done for a variety of test cases. Each test case has the same magnetic field, initial proton distribution, and detector position and size. The number of imaging elements in the detector is varied from 3600 to 36, the first being a very overspecified case, and the last very underspecified. The accuracy of the reconstruction is tested by a modified χ^2 test. The unmodified test

$$\chi^2 = \sum_i \frac{(O_i - E_i)^2}{E_i} \tag{6.3}$$

where O_i is the observed value, and E_i is the expected value, is unsuitable for this problem as there are regions where the expected values of proton distribution are zero. In addition, this test will overestimate the inaccuracy in regions where the expected value is small, and underestimate it in regions where the expected value is large. If we are concerned with measuring the fast particle population as a diagnostic for the neutral beam heating, it is preferable to weight the measure of accuracy in favour of the largest values

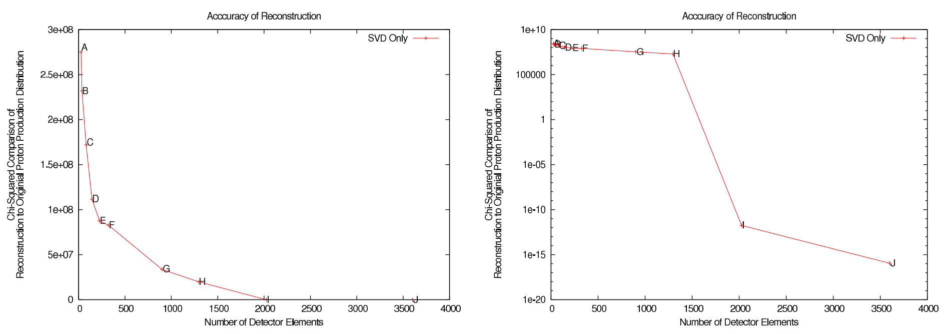


Figure 6.1: The accuracy of the reconstruction increases as the number of elements in the detector increases. The large jump in accuracy between points H and I is due to the problem changing from underspecified to overspecified between these points.

of the proton distribution rather than the smallest. The modified test used is

$$\chi^2 = \frac{\sum_i (O_i - E_i)^2}{\sum_i E_i} \tag{6.4}$$

The results can be seen in Fig. (6.1(a)). The large decrease in accuracy between the underspecified and overspecified problems can be seen as the number of imaging elements in the detector is increased, shown in Fig. (6.1(b)). The reconstructions for the labelled points can be compared to the initial distribution in Fig. (6.2). This shows that the general features of the initial distribution, namely a ring of greater proton production around a slight dip at the core of the plasma, are present in even quite under-specified problems. However, as the problem is increasingly underspecified, the SVD inversion will introduce regions of negative particle production. This is unphysical, as the particles produced are very unlikely to experience collisions sufficient to reduce their energy to the energy of the general population of the plasma in the time required to escape the plasma. There are no particle sinks within the plasma, only particle sources, and so the appearance of areas of negative particle production in the reconstruction must be an error.

6.2 Recovering the Plasma Image by Maximising the Entropy

As the SVD only reconstruction produces regions of negative particle production, and due to the expectation that the particle production distribution is expected to be smoothly varying, it is possible to apply certain preconditions to the data used for the maximum entropy reconstruction. The first is to replace any negative values with zeros in the reconstructed particle distribution. The effect of this on the accuracy of the reconstruction can be seen in Fig. (6.3). This will stop our distribution being a solution to Eq. (6.1) in addition to smoothing, and removing the unphysical particle sinks. We can then return our distribu-

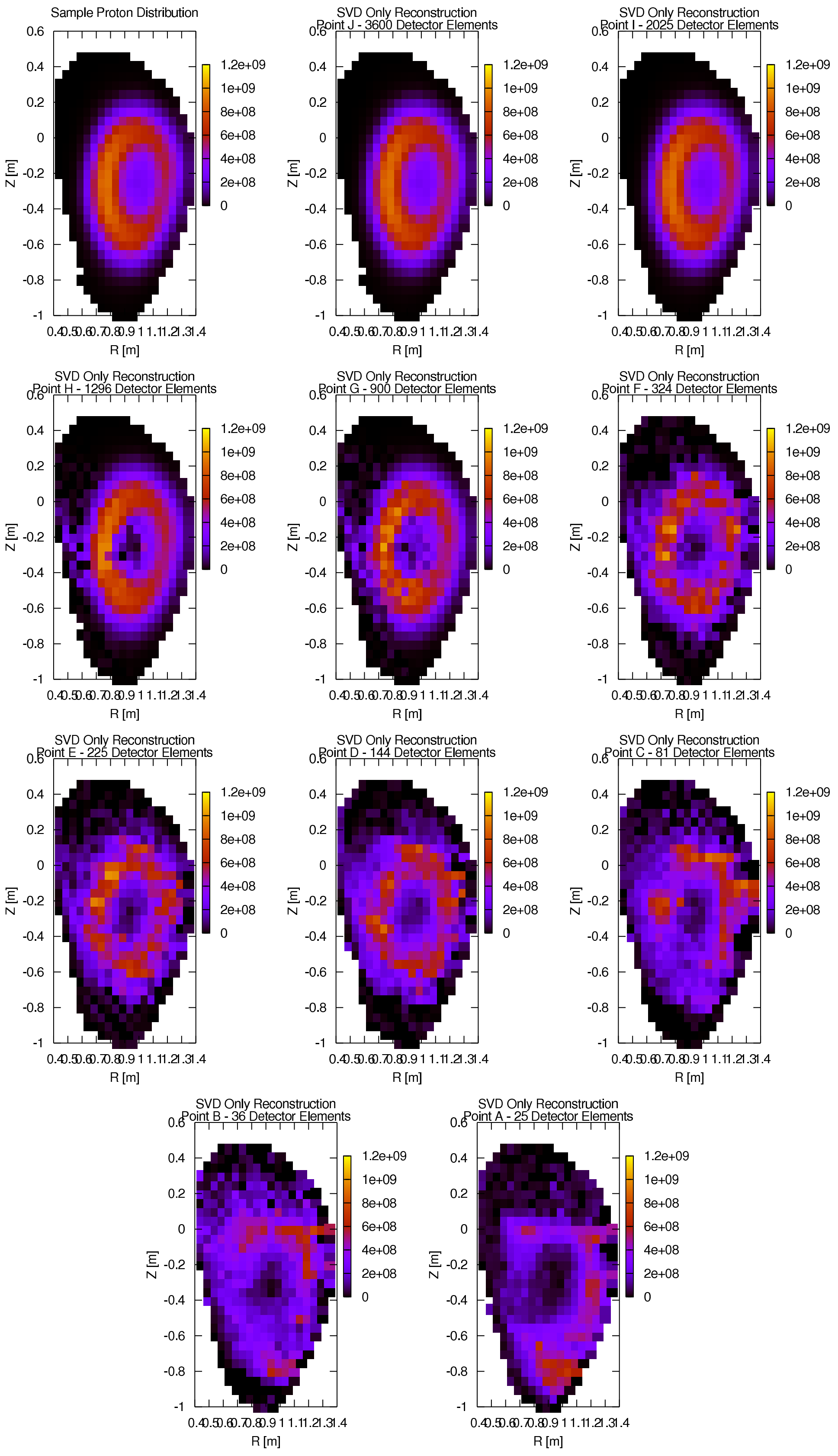


Figure 6.2: Example reconstructions using only the pseudo-inverse of the matrix, as calculated via SVD. Any negative values are not shown, to allow easier comparison to the maximum entropy reconstructions.

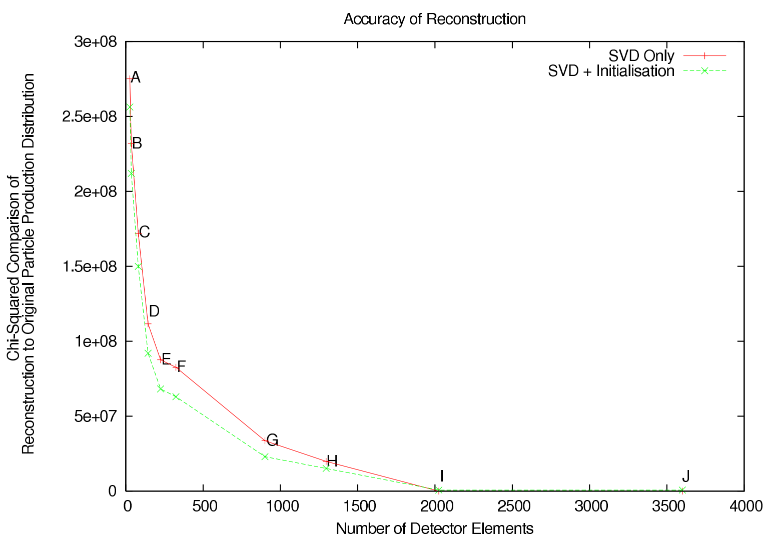


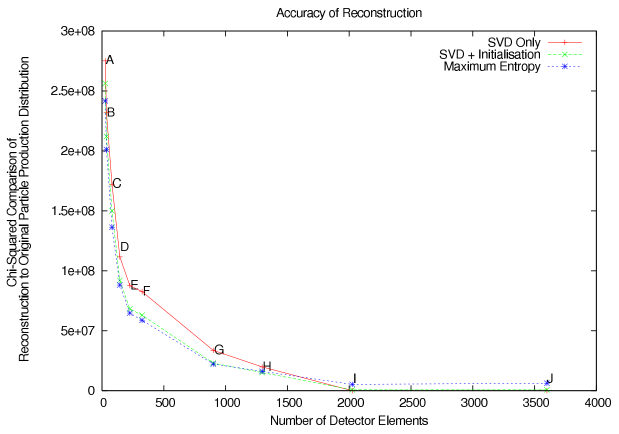
Figure 6.3: The negative values of the reconstructions are removed, and smoothing applied. This is then used as the initial guess for the Maximum Entropy iterative scheme.

tion to being a near solution of Eq. (6.1) by following the iterative scheme in Section (5.8), but using only the $\partial F/\partial x_i$ terms in updating the pixel values. Once the agreement with the observed data has converged, as tested by the modified χ^2 test comparison to the observed data, the entropy can be maximised whilst the increase in F is kept as low as possible. The example solutions from the full SVD and maximum entropy processes can be seen in Fig. (6.5). As can be seen in Fig. (6.4(a)), this technique fails on those examples where the the problem is overspecified, as it needlessly smooths already sufficiently smooth results. However, for detectors with only small numbers of elements compared to the number of pixels in the plasma, this increases the accuracy of the reconstruction, as shown in Fig. (6.4(b)). It is therefore possible, for this particular case, to find the minimum number of elements needed by the detector to reproduce the proton production distribution. First we must decide upon a reasonable level of accuracy in the reconstruction. From Fig. (6.5), we can see that a reasonable reconstruction is achieved at point D, which has a dectector containing 12×12 elements. Detectors with fewer elements no longer reliably reproduce the ring feature of the known distribution. This reconstruction also has an accuracy, as measured by the modified χ^2 test, just under 1×10^8. This can therefore be used as the cut-off of acceptable reconstruction.

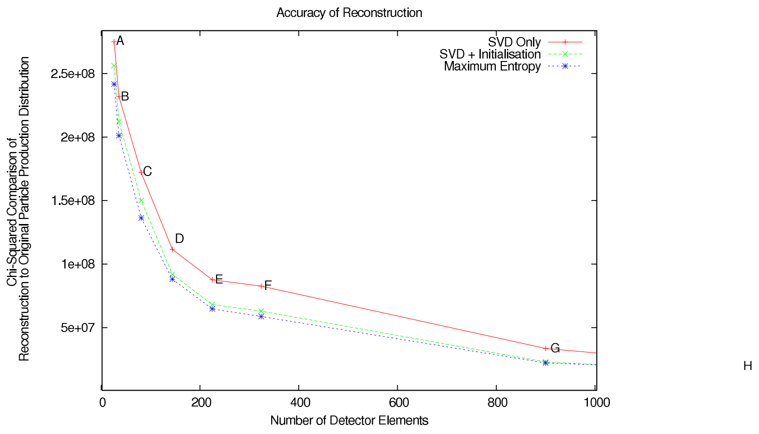
6.3 Displacing Initial Proton Distribution

The sensitivity of the detector to the initial proton distribution can be tested by shifting the initial distribution in either the R or the Z direction. An example of a shifted proton distribution, and the reconstruction of this distribution, are shown in Fig. (6.6)

The change in accuracy of the reconstructions of point D due to displacements in both



(a) Maximum Entropy technique fails for the over-specified problem



(b) Maximum Entropy technique successfully improves the reconstruction for underspecified problem

Figure 6.4: The Maximum Entropy scheme produces more accurate reconstructions than the only SVD reconstruction for the underspecified problem, but fails for the over-specified problem.

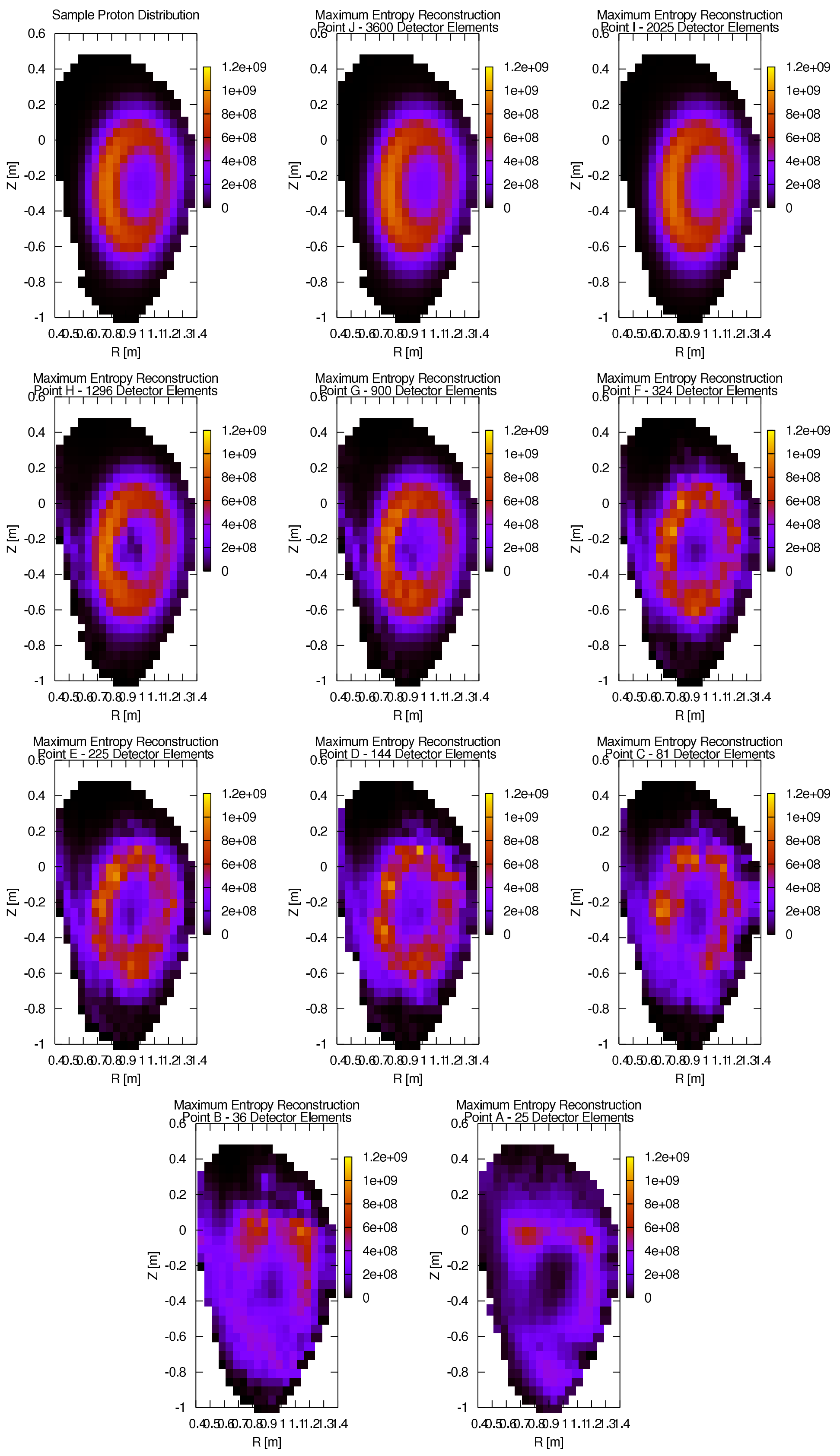
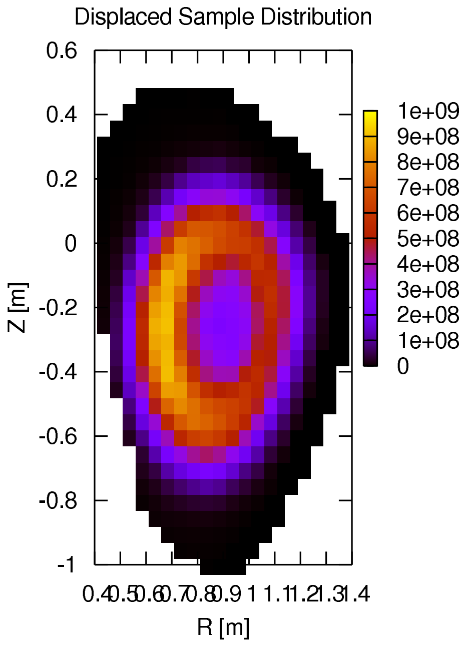
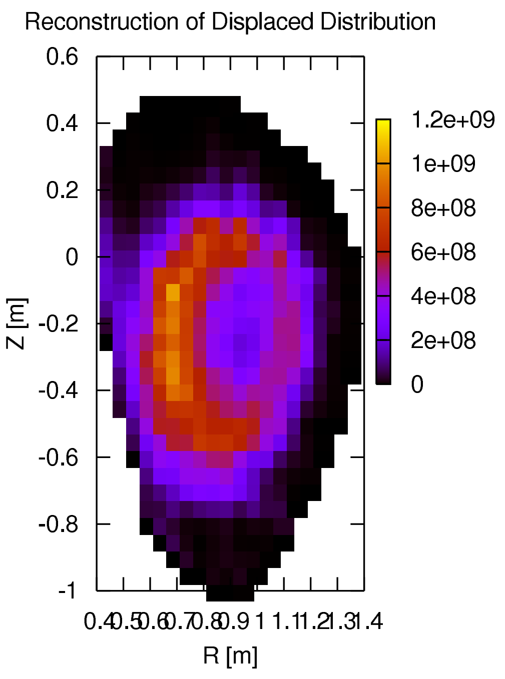


Figure 6.5: Example reconstructions using a combination of the SVD pseudo-inverse and the iterative Maximum Entropy scheme.



(a) Known Distribution



(b) Reconstruction

Figure 6.6: The displacement giving the greatest increase in accuracy. The proton distribution is moved 10cm towards the central column in the R, Z plane. This shows that the detector position is more important to the quality of the reconstruction than previously considered.

the R and Z directions is shown in Fig. (6.7). This shows that the midplane is the correct position for the plasma, but that the detector may be positioned too close to the plasma. The change in accuracy as the proton distribution is moved away from the plasma implies that either the choice of last closed flux surface as a cut-off for the production of protons for the construction of the matrix was too close to the centre of the plasma, or that the detector produces better results slightly further from the plasma, where the field of view is improved, or some combination of the two. For a detector positioned at $R = 1.7$, the plasma doesn't fill the field of view, as shown in Fig. (5.3), making it likely that the detector position is important in the accuracy of the reconstruction. However, if the detector is moved away from the plasma the proton flux is reduced, and there is more potential for multi-orbit losses to decrease the sharpness of the image, increasing the time needed to collect the data for a reconstruction.

6.4 Limits of Accuracy

6.4.1 Noise

The image at the detector, and, to a lesser extent, the plasma in the RZ-plane are both noisy, in as much as that they are both not completely smooth functions. One source of this noise is the discrete nature of the particles, which means that we cannot have a

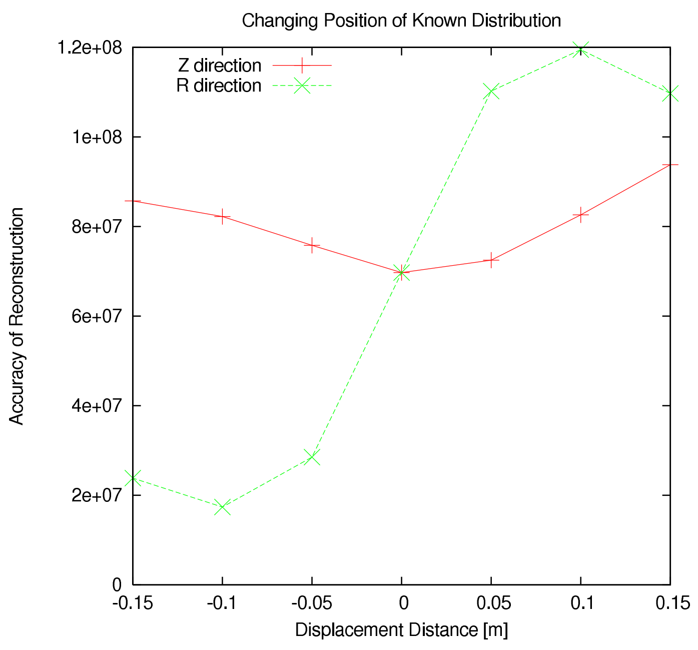


Figure 6.7: The proton distribution was displaced in both the R and Z directions by up to 15cm. The accuracy of the reconstruction is calculated via the modified χ^2 test.

continuum of results across detector images. This can be minimised by considering larger numbers of particles, as was explained in the Chapter 4, such that the effect of individual particles is small compared to the effect of all the particles. However, this can only reduce the noise. If we consider the points at the edge of the detector image, where very few particles are likely to strike, there will only ever be a small number of particles, no matter how many are simulated. In addition, for the inversion methods mentioned in Chapter (5), the greatest effect of small changes in the initial distribution of protons is seen at the edges of the detector image, the area we are most likely to see the noise from the discrete nature of the particles. The problem of calculating how long the detector will need to gather sufficient particles to reduce this noise to acceptable levels is part of the future work that needs to be done on this problem.

We may broadly divide the rest of the noise into two main types. First, we will consider false positives. False positives may be considered as those occasions upon which the detector has registered a signal, but the source of the signal was not a fusion born proton. There are various possible sources for this sort of noise, including other species of charged particles, fast neutrals, and particles which have come from the interaction of the plasma with the detector itself. The false positive rate is expected to be broadly similar across the detector as these events result from things like the interaction of the detector with charged particles diffused from the plasma, which whould strike the detector approximately equally in all detector elements, and that it is assumed that it can be expressed as a percentage of the maximum recorded signal. Secondly, there are false negatives. These arise when a fusion born proton has hit the detector, but the detector has not recorded this

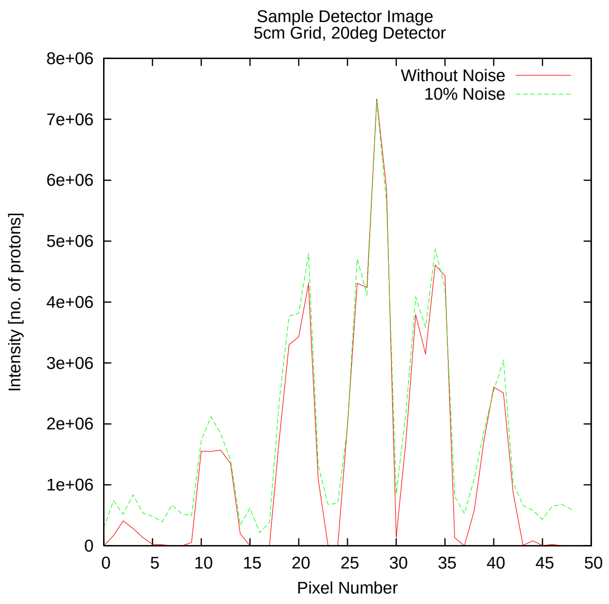


Figure 6.8: Effect of 10% noise on the intensities at each element of the detector

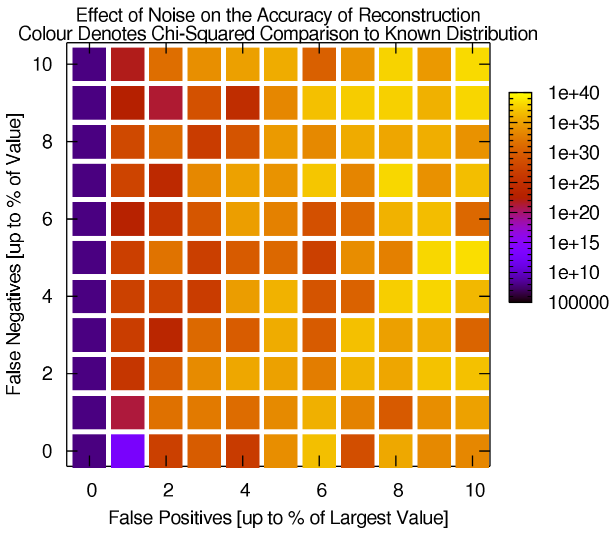


Figure 6.9: Effects of varying levels of noise on the accuracy of the results. These results show that the reconstruction is very sensitive to false positives, but not so sensitive to false negatives. The results are gathered from comparing reconstructions with added noise to those without, via a χ^2 comparison.

event. It this instance, these are modelled by reducing the signal at each detector element by an amount proportional to that signal. In both cases, the variation in the signal is assumed to be random, up to a maximum of the value given. Fig. (6.8) shows the impact on the detector intensities of a false positive and false negative rate of up to 10%. The noise is assumed to be a uniform distribution, up to the size defined by the percentage rate used in each instance. The effecs of the noise on the reconstruction can be seen in Fig. 6.9, where the accuracy of the results is calculated using the same modified χ^2 test as other results. The impact of the false positives is much greater than that of the false negatives, but the problem of false positivies is a simpler one to solve, by using a thicker screen in front of the detector. This helps to eliminate the false positives arising from particles that are not fusion born protons, at the cost of increasing the false negative rate. This will trap low energy charged particles, such as those that diffuse across the separatrix, whilst still allowing a proportion of the fast fusion products to pass. Other types of noise, such as the noise generated by cross talk between detector elements is not considered here, as a more advanced concept of the detector is required before progress can be made in this area.

6.5 Proton Production Averaged over Ψ

The proton production rate of the plasma is often given as a function of Ψ, which forms a convenient substitute for the minor radius, especially for more triangular plasmas, such as those in MAST. The Ψ of the sample magnetic field is shown in Fig. (6.10). The assumption that the fast particles are evenly spread over flux surfaces allows us to find the average proton production at intervals of Ψ. This assumption is made so that the results can be compared to results from [5], where the same assumption is made.

The average proton production rate for both the sample distribution taken from shot 18808, and the reconstructed distribution, is shown in Fig. (6.11). The reconstruction is based on the 144 element detector, point D in Fig. (6.4(a)). The broadening of the curve of the reconstruction, and lowering of the peak, are expected from the maximum entropy technique. This is due to the pressure towards uniformity exerted by attempting to maximise the entropy. Most of the features are successfully reconstructed, as is the total area under the curve. This corresponds the the total production of protons per unit time.

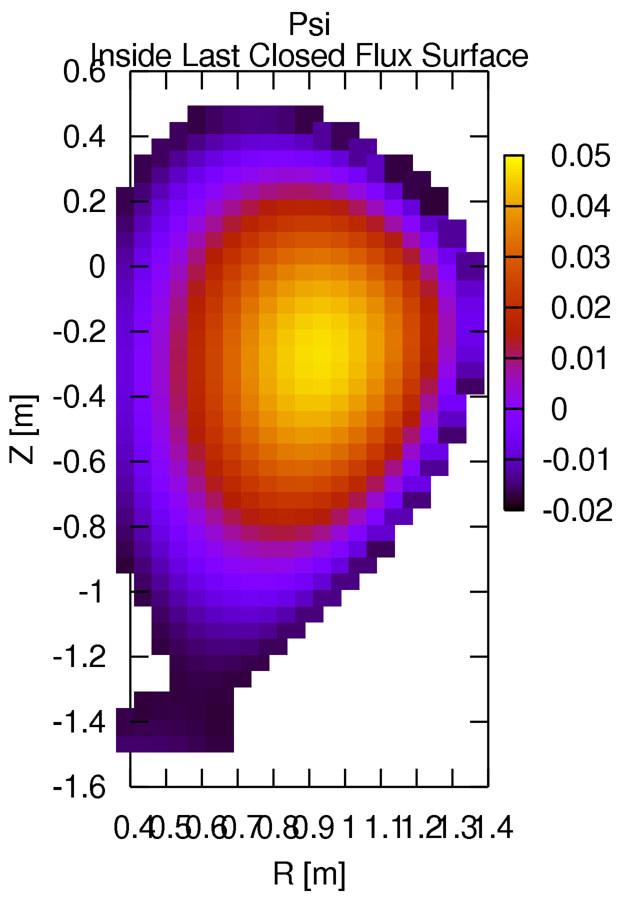


Figure 6.10: Ψ for the sample distribution. Areas of equal Ψ are assumed to be on the same flux surface, and so the proton production rate can be averaged over this area.

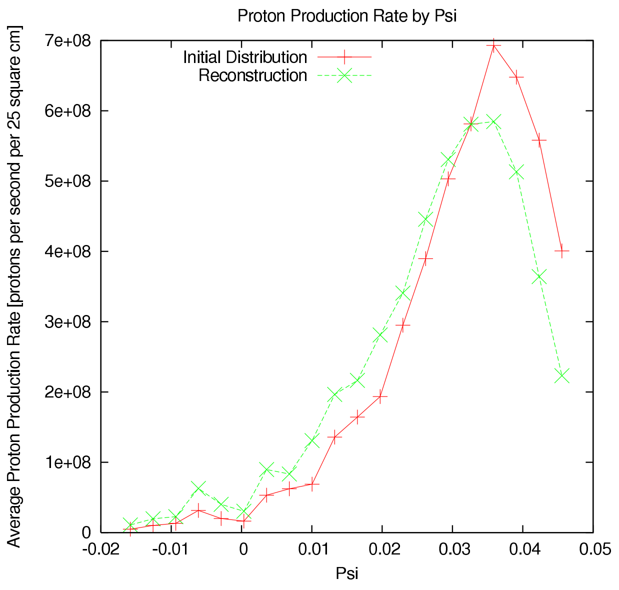


Figure 6.11: The proton production rate at intervals of Ψ. The lowered peak, and broadening of the curve is expected from a maximum entropy reconstruction. The position of the reconstruced peak coincides with that of the sample distribution.

Chapter 7

Conclusions

A set of codes has been developed to simulate and interpret the data recorded by a notional proton detecting diagnostic on a low-field tokamak, specifically the Mega-Amp Spherical Tokamak (MAST), Culham. This diagnostic is designed to measure the fast particle distribution of a deuterium plasma, in particular the spatial distribution of the fast particles injected into the plasma by the neutral beam injection system. These fast particles are used to heat the plasma, and provide part of the plasma current. The use of the neutral beam injection system is vital to producing the quality of the plasma needed for the research undertaken at MAST. The proposed diagnostic will be able to directly image the fast particle distribution, using the protons produced by the reaction of the fast particles with both the bulk plasma and other fast ions. The rate at which these protons are expected to be produced within the plasma was calculated, and a method which allowed the signals produced by the detector to be interpreted in terms of the fast particle distribution was developed.

The reaction cross of the deuterium-deuterium reaction was calculated using a variety of parameterisations [32, 6, 28]. The expected total proton production rate was determined for a sample MAST plasma, taken from shot 18808, and compared to the total neutron emission rate. This was used to determine parameterisation used, as a deuterium reaction is equally likely to produce a neutron as a proton. The initial velocity distribution of the fusion born protons was calculated. This is calculated in two parts. The first the velocity of the particle resulting from the fusion event, and the second is that resulting from the combined velocities of the parent particles. The rotation of the plasma is neglected. The anisotropy of the velocity distribution caused by the velocity of the parents was accounted for, as was the anisotropy inherant to products of the fusion reaction [10].

A code was developed to utilise a general purpose graphical processing unit (GPGPU), using CUDA language and architecture, in tracking the orbits of fusion born protons through the device. These protons are non-collisional, and have sufficient energy to be lost to the plasma within a few Larmor orbits. The code takes advantage of the embarassingly parallelisability of the problem to calculate the Larmor orbits of up to 100,000 protons per second using an nVidia GT 640. This can be achieved due to the non-interacting nature of the protons, and efficient use of the hardware. A 4th-Order Runge-Kutta algorithm was used to solve the Lorentz equation, being found to be faster than the CUEBIT algorithm for this problem, whilst retaining similar accuracy.

The orbit-following code was used to allow the creation of a transformation matrix linking the initial proton distribution in the plasma to the image recorded by the detector. This was done using both a brute force method, relying on creating a large number of particles at each point in the plasma and tracking them forwards in time, and a method that calculated the etendu of each point in the plasma by tracking the particles backwards in time from the detector. The first method is faster for simple problems, where assumptions such as the toroidal symmetry of the plasma allow considerable gains in accuracy without having to calculate greater numbers of particles. The second method, of intially tracking the particles backwards in time, is faster for problems where these simplifying assumptions cannot be made, and a full 3-d model of the plasma must be created. The use of CUDA and the GPU allowed large numbers to particles to be simulated in short times, allowing the necessary number of particles for these matrices to be calculated using a desktop machine. The detector used was a simple pinhole detector, with an array of detecting elements behind the aperture. The problem of finding the best position and design for the detector was not considered.

Finally, a code was developed to recreate the initial proton distribution from the detector image. This relied upon a mix of singular value decomposition of the transformation matrix, allowing an inverse to be found, and maximising the entropy of the candidate reconstruction to ensure that the reconstruction is the most uniform possible that is consistent with the data. This should produce a reconstruction in which any departure from uniformity is essential for the reconstruction to fit the data recorded. However, questions remain over the best method for calculating the entropy of the image, and the best algorithm to use whilst maximising the entropy. The algorithm presented in this thesis weights each point in the plasma by the likelihood that that point will contribute particles to the image captured at the detector. The intention was to ensure that the influence of points that only contribute fewer particles should be reduced, which should decrease the error in those points contributing the most particles. If the proton distribution is such that the areas with the highest levels of proton production are also areas which are less likely to contribute points to the detector, this can lead to distortion in the reconstrution. The method by which the detector image of the reconstruction is made consistent with the measured image deserves further consideration. The current method enforces positive values in the reconstruction at the start of the process, whilst other techniques can ensure that there are no negative values by assigning large costs to these values.

This work shows that if detectors can be made with at least 144 detector elements, reasonable recreations of the fusion proton distribution can be made in the R, Z-plane. This would allow the fast ion distribution to be diagnosed on a 5 by 5cm grid, which will allow more accurate diagnosis of the neutral beam injection system.

7.1 Future Work

There is considerable scope for future work arising from this project. This can be split into two main parts. First, there is the design and position of the detector. Very little work has been done in finding the most effective position and design of the detector. In particular, it would be beneficial to model the proton detector currently being installed in MAST. Secondly, the maximum entropy scheme currently in use may not be the most efficient and effective method of reconstructing the fusion proton distribution.

The future work on the detector may be split into two parts, being the design of the detector itself, and the position of the detector. These two parts will be somewhat interconnected, as the ideal position for one design may not be the ideal location for another. In addition, the possibility of having smaller, more widely spaced detectors, such as those in W. U. Boeglin et. al. (2010) [5]. This would also provide an alternative method of interpreting the results from the proton detectors currently being installed in MAST.

Alternative maximum entropy methods would also center on two points. Firstly, the choice of probabilities of the pixels of the reconstruction should be examined. Secondly, it would be beneficial to be able to select the algorithm used to maximise the entropy that best fits the use case. It may be that using an alternative measure of the information of the image, for example Fisher information, would allow for faster and more accurate reconstructions.

Finally, there must be a consideration of the cost of the detector. As the number of detector elements required grows, so will the cost of the device. In any project, a final decision must be made on the basis of cost vs. reward. Will the resulting data be worth the money required to gather it? An important piece of future work will therefore be to consider the cost of any proposed detector.

Appendix A

Appendix 1: Orbit Following Code

A.1 CUDA RK4 Orbit Code

The 4th Order Runge-Kutta code used on the CUDA devices.

```
  __global__ void testKernal(float4 *vel_D, float4 *pos_D, float4
{

  //define constants
  const float q_over_m = 1. * ELECTRONIC_CHARGE / PROTON_MASS;

  const float Timestep_Larmor_Fraction = 0.0001;

  //loop counters
  float time = 0.0;

  //iteration counter
  int nits = 0;

  //bfield container
  float4 B0 = make_float4(0.0, 0.0, 0.0, 0.0);
  float4 posOld = make_float4(0.0, 0.0, 0.0, 0.0);
  float4 posNew = make_float4(0.0, 0.0, 0.0, 0.0);
  float4 velOld = make_float4(0.0, 0.0, 0.0, 0.0);
  float4 velNew = make_float4(0.0, 0.0, 0.0, 0.0);
  float R = 0.0;
  float4 B_base = make_float4(0.0, 0.0, 0.0, 0.0);
  float phi = 0.0;
  float4 k1 = make_float4(0.0, 0.0, 0.0, 0.0);
  float4 v1 = make_float4(0.0, 0.0, 0.0, 0.0);
  float4 pos1 = make_float4(0.0, 0.0, 0.0, 0.0);
  float R1 = 0.0;
  float4 B_base1 = make_float4(0.0, 0.0, 0.0, 0.0);
  float4 B1 = make_float4(0.0, 0.0, 0.0, 0.0);
  float4 k2 = make_float4(0.0, 0.0, 0.0, 0.0);
  float4 v2 = make_float4(0.0, 0.0, 0.0, 0.0);
  float4 pos2 = make_float4(0.0, 0.0, 0.0, 0.0);
  float R2 = 0.0;
  float4 B_base2 = make_float4(0.0, 0.0, 0.0, 0.0);
  float4 B2 = make_float4(0.0, 0.0, 0.0, 0.0);
  float4 k3 = make_float4(0.0, 0.0, 0.0, 0.0);
  float4 v3 = make_float4(0.0, 0.0, 0.0, 0.0);
  float4 pos3 = make_float4(0.0, 0.0, 0.0, 0.0);
  float R3 = 0.0;
  float4 B_base3 = make_float4(0.0, 0.0, 0.0, 0.0);
  float4 B3 = make_float4(0.0, 0.0, 0.0, 0.0);
  float4 k4 = make_float4(0.0, 0.0, 0.0, 0.0);
  float RR = 0.0;

  int threadNum = (blockDim.x*blockIdx.x) + threadIdx.x;
  float4 pos = pos_D[threadNum];
  posNew_out[threadNum] = make_float4(0.0, 0.0, 0.0, 0.0);
  posOld_out[threadNum] = make_float4(0.0, 0.0, 0.0, 0.0);

  //Calculate the magnetic pitch angle
  float Rinit = sqrt(pos.x*pos.x+pos.y*pos.y);
  float4 pete = tex2D(tex_B, (((Rinit)/2.0)), (((pos.z+2.0)/4.0)))
  float jim = sqrt((pete.x*pete.x)+(pete.y*pete.y)+(pete.z*pete.z)

  //Gives Magnetic Pitch Angle
  float4 Bnorm = make_float4((pete.x/jim), (pete.y/jim), (pete.z/j

  float4 anang2 = vel_D[threadNum];

  float birthvel = sqrt((2.*3020000.0*1.60217e-19)/1.67262158e-27)
//   float initialvel = sqrt((2.*39000.0*1.60217e-19)/1.67262158e-
  float initialvel = 0.0;
```

```
    float4 vel = make_float4((anang2.x*birthvel)+(initialvel*Bnorm.x)

    float larmor_period = (TWO_PI / q_over_m) / jim;
    float dt = Timestep_Larmor_Fraction * larmor_period;

    //loop for lifetime of the particle
    while((nits<nits_max) and (time< END_TIME))
    {
      //loop counter
      nits++;

      //needed for calculation of larmor_period
      pete = tex2D(tex_B, ((((sqrt(pos.x*pos.x + pos.y*pos.y))/2.0)))
      jim = sqrt((pete.x*pete.x)+(pete.y*pete.y)+(pete.z*pete.z));

      //Give larmor period and timestep, ensures accuracy
      larmor_period = TWO_PI / q_over_m / jim;
      dt = Timestep_Larmor_Fraction * larmor_period;

//      if ((sqrt(vel.x*vel.x+vel.y*vel.y+vel.z*vel.z)*dt) > 0.01)
//        dt = 0.01/sqrt(vel.x*vel.x+vel.y*vel.y+vel.z*vel.z);
//      }

      //divides positions into new and old
      posOld = make_float4(pos.x, pos.y, pos.z, pos.w);
      posNew = make_float4(pos.x, pos.y, pos.z, pos.w);

      //divides velocities into new and old
      velOld = make_float4(vel.x, vel.y, vel.z, vel.w);
      velNew = make_float4(vel.x, vel.y, vel.z, vel.w);

      //interpolated B-field
      R = sqrt(pos.x*pos.x + pos.y*pos.y);
      B_base = tex2D(tex_B, (((R)/2.0)), (((pos.z+2.0)/4.0)));
      phi = atan2(pos.y, pos.x);
      B0 = make_float4((B_base.x*cos(phi)) - (B_base.y*sin(phi)), (B_

      //Lorentz force positions
    pos = make_float4(posOld.x,  posOld.y, posOld.z, posOld.w);
    //lorentz force velocities
    vel = make_float4(velOld.x, velOld.y, velOld.z, velOld.w);

    //Runge-Kutta method
    //K1 term
    k1 = make_float4(q_over_m*(vel.y*B0.z-B0.y*vel.z), q_over_m*(ve
    //Data from K1 term
    v1 = make_float4(vel.x + k1.x*0.5*dt, vel.y+k1.y*0.5*dt, vel.z-
    pos1 = make_float4(pos.x+(v1.x*0.5*dt), pos.y+(v1.y*0.5*dt), po
    R1 = sqrt(pos1.x*pos1.x+pos1.y*pos1.y);
    B_base1 = tex2D(tex_B, (((R1)/2.0)), (((pos1.z+2.0)/4.0)));
    B1 = make_float4(B_base1.x*cos(phi) - B_base1.y*sin(phi), B_bas
    //K2 term
    k2 = make_float4(q_over_m*((v1.y*B1.z)-(v1.z*B1.y)), q_over_m*
    //data from K2 term
    v2 = make_float4(vel.x + k2.x*0.5*dt, vel.y+k2.y*0.5*dt, vel.z-
    pos2 = make_float4(pos.x+(v2.x*0.5*dt), pos.y+(v2.y*0.5*dt), po
    R2 = sqrt(pos2.x*pos2.x+pos2.y*pos2.y);
    B_base2 = tex2D(tex_B, (((R2)/2.0)), (((pos2.z+2.0)/4.0)));
    B2 = make_float4(B_base2.x*cos(phi) - B_base2.y*sin(phi), B_bas
    //K3 term
    k3 = make_float4(q_over_m*((v2.y*B2.z)-(v2.z*B2.y)), q_over_m*
    //data from K3 term
    v3 = make_float4(vel.x + k3.x*dt, vel.y+k3.y*dt, vel.z+k3.z*dt,
    pos3 = make_float4(pos.x+(v3.x*dt), pos.y+(v3.y*dt), pos.z+(v3
    R3 = sqrt(pos3.x*pos3.x+pos3.y*pos3.y);
    B_base3 = tex2D(tex_B, (((R3)/2.0)), (((pos3.z+2.0)/4.0)));
    B3 = make_float4(B_base3.x*cos(phi) - B_base3.y*sin(phi), B_bas
    //K4 term
    k4 = make_float4(q_over_m*((v3.y*B3.z)-(v3.z*B3.y)), q_over_m*

    //final acceleration
    velNew = make_float4(velOld.x+((1./6.)*dt*(k1.x+(2.0*k2.x)+(2.0


//    Calculate new position
    posNew = make_float4(posOld.x +  (dt * velNew.x),posOld.y + (dt
        //replace positions and velocities
        pos = posNew;
        vel = velNew;

        //Used to calculate if the particle has left the device
        RR = sqrt((pos.x*pos.x)+(pos.y*pos.y));

        float phi = atan2(pos.y, pos.x);
        float v_phi = (vel.x*-sin(phi))+(vel.y*cos(phi));
        float torCanMom = ((PROTON_MASS*v_phi*RR)) - (ELECTRONIC_CHARGE

        track_pos[nits-1] = make_float4(pos.x, pos.y, pos.z, torCanMom)

        //increment time counter
        time += dt;

        //stops calculation if outside vessel or hits coil
        if((RR > 1.7) or (fabs(pos.z)>2.0) or (RR < 0.22) or ((RR > 1.4
        {

          posNew_out[threadNum] = make_float4(posNew.x, posNew.y, posNe
          posOld_out[threadNum] = make_float4(posOld.x, posOld.y, posOl

          break;
        }
      }
    }
```

A.2 CUDA CUEBIT Code

The CUEBIT algorithm used on the CUDA devices.

```
__global__ void cuebitKernal(float4 *vel_D, float4 *pos_D, float4 *
{

  //define constants
  const float q_over_m = 1. * ELECTRONIC_CHARGE / PROTON_MASS;
const float Timestep_Larmor_Fraction = 0.0001;

//loop counters
float time = 0.0;

//iteration counter
int nits = 0;

//bfield container
float4 B0 = make_float4(0.0, 0.0, 0.0, 0.0);
float4 posOld = make_float4(0.0, 0.0, 0.0, 0.0);
float4 posNew = make_float4(0.0, 0.0, 0.0, 0.0);
float4 velOld = make_float4(0.0, 0.0, 0.0, 0.0);
float4 velNew = make_float4(0.0, 0.0, 0.0, 0.0);
float R = 0.0;
float4 B_base = make_float4(0.0, 0.0, 0.0, 0.0);
float phi = 0.0;
float RR = 0.0;

int threadNum = (blockDim.x*blockIdx.x) + threadIdx.x;
float4 pos = pos_D[threadNum];
posNew_out[threadNum] = make_float4(0.0, 0.0, 0.0, 0.0);
posOld_out[threadNum] = make_float4(0.0, 0.0, 0.0, 0.0);

//Calculate the magnetic pitch angle
float Rinit = sqrt(pos.x*pos.x+pos.y*pos.y);
float4 pete = tex2D(tex_B, (((Rinit)/2.0)), (((pos.z+2.0)/4.0)))
float jim = sqrt((pete.x*pete.x)+(pete.y*pete.y)+(pete.z*pete.z)

//Gives Magnetic Pitch Angle
float4 Bnorm = make_float4((pete.x/jim), (pete.y/jim), (pete.z/j

float4 anang2 = vel_D[threadNum];

float4 ang =  make_float4(anang2.x+Bnorm.x, anang2.y+Bnorm.y, ana
float4 ang2 = make_float4(ang.x/(sqrt((ang.x*ang.x)+(ang.y*ang.y)

float birthvel = sqrt((2.*3020000.0*1.60217e-19)/1.67262158e-27)
```

```
//   float initialvel = sqrt((2.*39000.0*1.60217e-19)/1.67262158e-2
  float initialvel = 0.0;
  float4 vel = make_float4((anang2.x*birthvel)+(initialvel*Bnorm.x)

  float larmor_period = TWO_PI / q_over_m / jim;
  float dt = Timestep_Larmor_Fraction * larmor_period;

  //loop for lifetime of the particle
  while ((nits<nits_max) and (time < END_TIME)) {
    //loop counter
    nits++;

    //needed for calculation of larmor_period
    pete = tex2D(tex_B, ((((sqrt(pos.x*pos.x + pos.y*pos.y))/2.0)))
    jim = sqrt((pete.x*pete.x)+(pete.y*pete.y)+(pete.z*pete.z));

    //Give larmor period and timestep, ensures accuracy
    larmor_period = TWO_PI / q_over_m / jim;
    dt = Timestep_Larmor_Fraction * larmor_period;

    if ((sqrt(vel.x*vel.x+vel.y*vel.y+vel.z*vel.z)*dt) > 0.01) {
      dt = 0.01/sqrt(vel.x*vel.x+vel.y*vel.y+vel.z*vel.z);
    }

    //divides positions into new and old
    posOld = make_float4(pos.x, pos.y, pos.z, pos.w);
    posNew = make_float4(pos.x, pos.y, pos.z, pos.w);

    //divides velocities into new and old
    velOld = make_float4(vel.x, vel.y, vel.z, vel.w);
    velNew = make_float4(vel.x, vel.y, vel.z, vel.w);

    //interpolated B-field
    R = sqrt(pos.x*pos.x + pos.y*pos.y);
    B_base = tex2D(tex_B, ((R/2.0)), (((pos.z+2.0)/4.0)));
    phi = atan2(pos.y, pos.x);
    B0 = make_float4(B_base.x*cos(phi) - B_base.y*sin(phi), B_base
for (int ITERATIONS = 0; ITERATIONS < 4; ITERATIONS++) {
  pos.x = 0.5*(posOld.x+posNew.x);
  pos.y = 0.5*(posOld.y+posNew.y);
  pos.z = 0.5*(posOld.z+posNew.z);
  pos.w = 0.5*(posOld.w+posNew.w);
  vel.x = 0.5*(velOld.x+velNew.x);
  vel.y = 0.5*(velOld.y+velNew.y);
  vel.z = 0.5*(velOld.z+velNew.z);
  vel.w = 0.5*(velOld.w+velNew.w);
//interpolated B-field
  R = sqrt(pos.x*pos.x + pos.y*pos.y);
  B_base = tex2D(tex_B, ((R/2.0)), (((pos.z+2.0)/4.0)));
  phi = atan2(pos.y, pos.x);
  B0 = make_float4(B_base.x*cos(phi) - B_base.y*sin(phi), B_bas
  double lambda = 0.5 * q_over_m * dt;
  double ax = lambda * B0.x; // corresponding to \alpha_x
  double ay = lambda * B0.y; // corresponding to \alpha_y
  double az = lambda * B0.z; // corresponding to \alpha_z
  double M11 = 1.0+ax*ax-ay*ay-az*az;
  double M12 = 2.0*(ax*ay+az);
  double M13 = 2.0*(az*ax-ay);
  double M21 = 2.0*(ax*ay-az);
  double M22 = 1.0-ax*ax+ay*ay-az*az;
  double M23 = 2.0*(ay*az+ax);
  double M31 = 2.0*(az*ax+ay);
  double M32 = 2.0*(az*ay-ax);
  double M33 = 1.0-ax*ax-ay*ay+az*az;
  double aa = 1.0/(1.0+ax*ax+ay*ay+az*az); // pre-multiplier
  velNew.x = aa * (M11*velOld.x + M12*velOld.y + M13*velOld.z)
  velNew.y = aa * (M21*velOld.x + M22*velOld.y + M23*velOld.z)
  velNew.z = aa * (M31*velOld.x + M32*velOld.y + M33*velOld.z)
  posNew.x = posOld.x + 0.5 * dt * (velOld.x + velNew.x);
  posNew.y = posOld.y + 0.5 * dt * (velOld.y + velNew.y);
  posNew.z = posOld.z + 0.5 * dt * (velOld.z + velNew.z);
}

//replace positions and velocities
pos = posNew;
```

```
      vel = velNew;

      //Used to calculate if the particle has left the device
      RR = sqrt((pos.x*pos.x)+(pos.y*pos.y));



      float phi = atan2(pos.y, pos.x);
      float v_phi = (vel.x*-sin(phi))+(vel.y*cos(phi));
      float torCanMom = ((PROTON_MASS*v_phi*RR)) - (ELECTRONIC_CHARG

      track_pos[nits-1] = make_float4(pos.x, pos.y, pos.z, torCanMom
      //increment time counter
      time += dt;

      //stops calculation if outside vessel or hits coil
      if((RR > 1.7) or (fabs(pos.z)>2.0) or (RR < 0.22) or ((RR > 1.
      {
        posNew_out[threadNum] = make_float4(posNew.x, posNew.y, posNe
        posOld_out[threadNum] = make_float4(posOld.x, posOld.y, posOl

        break;
      }
    }
  }
```

Appendix B

Appendix 2: Maximum Entropy Finder

This is the method used to find the maximum entropy, whilst ensuring that the detector data is consistent with the observed data.

```
while ((success < 2))  {

  fred++;
  S = 0.0;

  scaled_sum = 0.0;
  for (int i = 0; i < Plasma_Length; i++) {
    if (gamma[i] > 0.0) {
      if (pixels[i] > 0.0) {
        pixels_scaled[i] = ((pixels[i]*gamma[i]))/(data_sum);
      }
      else {
        pixels_scaled[i] = 0e0;
      }
    }
    else {
      pixels_scaled[i] = 0.0;
    }
    scaled_sum += pixels_scaled[i];
  }

  for (int i = 0; i < Plasma_Length; i++) {
    if ((pixels_scaled[i] > 0.) && (pixels_scaled[i] < 1.)) {
      S += -1.0*(pixels_scaled[i]*log(pixels_scaled[i]));
    }
```

```
}
for (int i = 0; i < Detector_Length; i++) {
  Fake_data[i] = 0.0;
}

for (int i = 0; i < Detector_Length; i++) {
  for (int j = 0; j < Plasma_Length; j++) {
    Fake_data[i] += Matrix[i][j]*pixels[j];
  }
}

Fold = F;
F = 0.0;
for (int i = 0; i < Detector_Length; i++) {
  F += (((data[i]) - (Fake_data[i]))*((data[i]) - (Fake_data[i
}
Fnew = F;

Qold = Q;
if ((Fold <= Fnew) && (fred > 2)) {
  change2 = true;
  for (int i = 0; i < Plasma_Length; i++) {
  }
}
if ((change2 == true) && (change3 == false)) {
  change3 = true;
  Finitial = F;
  Sinitial = S;
}

if (change2 == true) {
  Q = ((S/Sinitial) - ((F/Finitial)));
  flength = 1;
}
else {
  Q = -F/2.;
  alpha = 0.0;
  flength = 1;
```

```
    }
    Qnew = Q;
    Fold = F;

    if ((Qold>=Qnew) && (change2 == true)) {
      success++;
    }

    for (int i = 0; i < Plasma_Length; i++) {
      f2[i] = 0.0;
    }

    for (int i = 0; i < Plasma_Length; i++) {
      for (int j = 0; j < Detector_Length; j++) {
        f2[i] += -(2./data_sum)*MTransp[i][j]*(data[j] - Fake_data
      }
    }

    for (int i = 0; i < Plasma_Length; i++) {
      if ((pixels_scaled[i] > 0.) && (pixels_scaled[i] < 1.)) {
        s[i] = -(1.+log(pixels_scaled[i]));
      }
      else {
        s[i] = 0.0;
      }
    }

    beta_left = 0.0;
    beta_right = 10.0;
    beta_test = 0.0;
    line_left = make_double2(0.0, 0.0);
    line_right = make_double2(0.0, 0.0);
    int check = 0;
//     Beta Minimum finder
    while ((fabs(beta_left-beta_right) > 1e-6) && (check < 100)) {
      check++;
      //Find beta_1, F_1, beta_2, F_2
      if (beta_test == 0.0) {
```

```
    beta_1 = 0.0;
    F_1 = Fold;
    point_1 = make_double2(beta_1, F_1);
  }
  else {
    beta_1 = beta_test - change;
    for (int i = 0; i < Plasma_Length; i++) {
      pixels_test[i] = pixels[i]*(1. - (beta_1*f2[i]));
      if (pixels_test[i] < 0.0) {
      }
    }
    for (int i = 0; i < Detector_Length; i++) {
      Fake_data[i] = 0.0;
    }
    for (int i = 0; i < Detector_Length; i++) {
      for (int j = 0; j < Plasma_Length; j++) {
        if (pixels_test[j] > 0.0) {
          Fake_data[i] += Matrix[i][j]*pixels_test[j];
        }
      }
    }
    F_1 = 0.0;
    for (int i = 0; i < Detector_Length; i++) {
      F_1 += ((data[i] - Fake_data[i])*(data[i] - Fake_data[i]
    }
    point_1 = make_double2(beta_1, F_1);
  }

  beta_2 = beta_test + change;
  for (int i = 0; i < Plasma_Length; i++) {
    pixels_test[i] = pixels[i]*(1. - (beta_2*f2[i]));
  }
  for (int i = 0; i < Detector_Length; i++) {
    Fake_data[i] = 0.0;
  }
  for (int i = 0; i < Detector_Length; i++) {
    for (int j = 0; j < Plasma_Length; j++) {
      if (pixels_test[j] > 0.0) {
          Fake_data[i] += Matrix[i][j]*pixels_test[j];
        }
      }
    }
    F_2 = 0.0;
    for (int i = 0; i < Detector_Length; i++) {
      F_2 += ((data[i] - Fake_data[i])*(data[i] - Fake_data[i]))/
    }

    point_2 = make_double2(beta_2, F_2);

    line_new = EqOfLine(point_1, point_2);

    if (line_new.x < 0.0) {
      line_left = line_new;
    }
    else if (line_new.x > 0.0) {
      line_right = line_new;
    }
    else {
      break;
    }

    beta_test = (line_right.y - line_left.y)/(line_left.x - line_
    if (beta_test < 0.0) {
      beta_test = 0.0;
    }
    if (line_new.x < 0.0) {
      beta_left = beta_test;
    }
    else {
      beta_right = beta_test;
    }
  }

  if (change2 == true) {
    alpha_left = 0.0;
    alpha_right = 1.0;
```

```
alpha_test = 0.0;
double q_test[10];
for (int i = 0; i < 10; i++) {
  q_test[i] = 0.0;
}
double q_max = 0.;
int q_high = 0;
check = 0;
while ((fabs(alpha_left-alpha_right) > 1e-7) && (check < 100)
  check++;

  for (int k = 0; k < 10; k++) {
    alpha_test = (((alpha_right-alpha_left)/10.)*double(k)) +
    for (int i = 0; i < Plasma_Length; i++) {
      pixels_test[i] = pixels[i]*(1. + (alpha_test*s[i]) - (b
      if (pixels_test[i] < 0.0) {
      }
    }
    for (int i = 0; i < Detector_Length; i++) {
      Fake_data[i] = 0.0;
    }
    for (int i = 0; i < Detector_Length; i++) {
      for (int j = 0; j < Plasma_Length; j++) {
        if (pixels_test[j] > 0.0) {
          Fake_data[i] += Matrix[i][j]*pixels_test[j];
        }
      }
    }
    F_1 = 0.0;
    for (int i = 0; i < Detector_Length; i++) {
      F_1 += ((data[i] - Fake_data[i])*(data[i] - Fake_data[i
    }

    pixel_sum = 0.0;
    for (int i = 0; i < Plasma_Length; i++) {
      pixel_sum += pixels_test[i]*gamma[i];
    }
```

```
      for (int i = 0; i < Plasma_Length; i++) {
        if (gamma[i] > 0.0) {
          pixels_scaled[i] = ((pixels_test[i]*gamma[i]))/(data_
        }
        else {
          pixels_scaled[i] = 0.0;
        }
      }

      S_1 = 0;
      for (int i = 0; i < Plasma_Length; i++) {
        if ((pixels_scaled[i] > 0.) && (pixels_scaled[i] < 1.)
          S_1 += -1.0*(pixels_scaled[i]*log(pixels_scaled[i]))
        }
      }
      q_test[k] = ((S_1/Sinitial)-(F_1/(Finitial)));
    }

    for (int i = 0; i < 10; i++) {
      if (q_test[i] > q_max) {
        q_max = q_test[i];
        q_high = i;
      }
    }

    alpha_left = (((alpha_right-alpha_left)/10.)*double(q_high
    if (alpha_left < 0.) {
      alpha_left = 0.0;
    }
    alpha_right = (((alpha_right-alpha_left)/10.)*double(q_hig
    alpha_test = (((alpha_right-alpha_left)/10.)*double(q_high
    alpha = alpha_test;
    beta = beta_test;
  }
}
else {
  alpha = 0.0;
  beta = beta_test;
  }
  for (int i = 0; i < Plasma_Length; i++) {
    pixels[i] = pixels[i]*(1. + (alpha*(s[i])) - ((beta*f2[i])));
    if (pixels[i] < 0.0) {
      pixels[i] = 0.0;
    }
  }
}
```

Definitions

$\mathbf{x}$	Plasma image pixel intensities
$\mathbf{y}$	Detector image pixel intensities
$\mathbf{M}$	Transformation matrix
$\mathbf{B}$	Magnetic field
Ψ	Poloidal magnetic flux
R	Major radius of tokamak
Z	Height of tokamak
θ	Toroidal Angle
ϕ	Poloidal Angle
n	Density
σ	Reaction cross-section
$\mathbf{v}$	Velocity
m	Mass
Ze	Charge
G	Gamow factor
t	Time
E	Energy
p	Momentum
T	Temperature

Bibliography

[1] RJ Akers, E. Verwichte, TJ Martin, SD Pinches, and R. Lake, *Gpgpu monte carlo calculation of gyro-phase resolved fast ion and n-state resolved neutral deuterium distributions.*, Proceedings of the 39th EPS Conference on Plasma Physics, July 2012.

[2] A. Alekseyev and G. Martin, *The application of natural diamond detectors to 3 MeV proton diagnostic at TORE SUPRA*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **417** (1998), no. 2-3, 400 – 404.

[3] Thomas R. Barrett, Chris Jones, Peter Blatchford, Brendan Smith, Roy McAdams, and Nick Woods, *Engineering design of the double neutral beam injection system for mast upgrade*, Fusion Engineering and Design **86** (2011), no. 68, 789 – 792, Proceedings of the 26th Symposium of Fusion Technology.

[4] S. Bochkanov and V. Bystritsky, *Alglib*, 2012.

[5] W. U. Boeglin, R. Valenzuela Perez, and D. S. Darrow, *Concept of a charged fusion product diagnostic for nstx*, Review of Scientific Instruments **81** (2010), no. 10, 10D301.

[6] H.-S. Bosch and G.M. Hale, *Improved formulas for fusion cross-sections and thermal reactivities*, Nuclear Fusion **32** (1992), no. 4, 611.

[7] Hans-Stephan Bosch, *Diagnostics with charged fusion products in ASDEX*, Review of Scientific Instruments **61** (1990), no. 6, 1699 –1707.

[8] BP, *Statistical review of world energy 2012*, Tech. report, 2012.

[9] L.S. Brown, D.L. Preston, and R.L. Singleton Jr, *Charged particle motion in a plasma: Electron-ion energy partition*, Arxiv preprint arXiv:1106.1463 (2011).

[10] Ronald E. Brown and Nelson Jarmie, *Differential cross sections at low energies for* $^2h(d,p)^3h$ *and* $^2h(d,n)^3he$, Phys. Rev. C **41** (1990), no. 4, 1391–1400.

[11] Climate Change Capital, *The path to power: Stage 4*, Tech. report, 2006.

[12] M. Cecconello, S. Sangaroon, M. Turnyanskiy, S. Conroy, I. Wodniak, R.J. Akers, G. Ericsson, and the MAST Team, *Observation of fast ion behaviour with a neutron emission profile monitor in mast*, Nuclear Fusion **52** (2012), no. 9, 094015.

[13] M. Cecconello, M. Turnyanskiy, S. Conroy, G. Ericsson, E. Ronchi, S. Sangaroon, R. Akers, I. Fitzgerald, A. Cullen, and M. Weiszflog, *A neutron camera system for MAST*, Review of Scientific Instruments **81** (2010), no. 10, 10D315.

[14] James Demmel and W. Kahan, *Accurate singular values of bidiagonal matrices*, SIAM J. Sci. Stat. Comput **11** (1990), 873–912.

[15] Siegfried H Glenzer, Brian K Spears, M John Edwards, Ethan T Alger, Richard L Berger, Darren L Bleuel, David K Bradley, Joseph A Caggiano, Debra A Callahan, Carlos Castro, Daniel T Casey, Christine Choate, Daniel S Clark, Charles J Cerjan, Gilbert W Collins, Eduard L Dewald, Jean-Michel G Di Nicola, Pascale Di Nicola, Laurent Divol, Shamasundar N Dixit, Tilo Dppner, Rebecca Dylla-Spears, Elizabeth G Dzenitis, James E Fair, Lars Johan Anders Frenje, M Gatu Johnson, E Giraldez, Vladimir Glebov, Steven M Glenn, Steven W Haan, Bruce A Hammel, Stephen P Hatchett II, Christopher A Haynam, Robert F Heeter, Glenn M Heestand, Hans W Herrmann, Damien G Hicks, Dean M Holunga, Jeffrey B Horner, Haibo Huang, Nobuhiko Izumi, Ogden S Jones, Daniel H Kalantar, Joseph D Kilkenny, Robert K Kirkwood, John L Kline, James P Knauer, Bernard Kozioziemski, Andrea L Kritcher, Jeremy J Kroll, George A Kyrala, Kai N LaFortune, Otto L Landen, Douglas W Larson, Ramon J Leeper, Sebastien Le Pape, John D Lindl, Tammy Ma, Andrew J Mackinnon, Andrew G MacPhee, Evan Mapoles, Patrick W McKenty, Nathan B Meezan, Pierre Michel, Jose L Milovich, John D Moody, Alastair S Moore, Mike Moran, Kari Ann Moreno, David H Munro, Bryan R Nathan, Abbas Nikroo, Richard E Olson, Charles D Orth, Arthur Pak, Pravesh K Patel, Tom Parham, Richard Petrasso, Joseph E Ralph, Hans Rinderknecht, Sean P Regan, Harry F Robey, J Steven Ross, Jay D Salmonson, Craig Sangster, Jim Sater, Marilyn B Schneider, F H Sguin, Michael J Shaw, Milton J Shoup, Paul T Springer, Wolfgang Stoeffl, Larry J Suter, Cliff Avery Thomas, Richard P J Town, Curtis Walters, Stephen V Weber, Paul J Wegner, Clay Widmayer, Pamela K Whitman, Klaus Widmann, Douglas C Wilson, Bruno M Van Wonterghem, Brian J MacGowan, L Jeff Atherton, and Edward I Moses, *First implosion experiments with cryogenic thermonuclear fuel on the national ignition facility*, Plasma Physics and Controlled Fusion **54** (2012), no. 4, 045013.

[16] R.J Goldston, D.C McCune, H.H Towner, S.L Davis, R.J Hawryluk, and G.L Schmidt, *New techniques for calculating heat and particle source rates due to neu-*

tral beam injection in axisymmetric tokamaks, Journal of Computational Physics **43** (1981), no. 1, 61 – 78.

[17] M.P. Gryaznevich and S.E. Sharapov, *Perturbative and non-perturbative modes in start and mast*, Nuclear Fusion **46** (2006), no. 10, S942.

[18] M.P. Gryaznevich, S.E. Sharapov, M. Lilley, S.D. Pinches, A.R. Field, D. Howell, D. Keeling, R. Martin, H. Meyer, H. Smith, R. Vann, P. Denner, E. Verwichte, and the MAST Team, *Recent experiments on alfvén eigenmodes in MAST*, Nuclear Fusion **48** (2008), no. 8, 084003.

[19] S.F. Gull and J. Skilling, *Maximum entropy method in image processing*, Communications, Radar and Signal Processing, IEE Proceedings F **131** (1984), no. 6, 646–659.

[20] W. W. Heidbrink and J. D. Strachan, *Tokamak ion temperature and poloidal field diagnostics using 3-MeV protons*, Review of Scientific Instruments **56** (1985), no. 4, 501–518.

[21] MJ Hole, LC Appel, and R. Martin, *A high resolution mirnov array for the mega ampere spherical tokamak*, Review of Scientific Instruments **80** (2009), no. 12, 123507–123507.

[22] OECD/International Atomic Energy Agency (IAEA), *Uranium 2011*, (2011).

[23] G.W.C. Kaye, T.H. Laby, and National Physical Laboratory (Great Britain), *Kaye & laby tables of physical & chemical constants*, National Physical Laboratory, 1995.

[24] J. Kim and W. Choe, *Fast singular value decomposition combined maximum entropy method for plasma tomography*, Review of Scientific Instruments **77** (2006), no. 2.

[25] Heinz Knoepfel, *Tokamak start-up : problems and scenarios related to the transient phases of a thermonuclear fusion reactor*, Plenum Press, 1986.

[26] A. G. Korn and T. M. Korn, *Mathematical handbook for scientists and engineers*, Dover, 2000.

[27] A.V. Krasilnikov, V.N. Amosov, and Yu.A. Kaschuck, *Natural diamond detector as a high energy particle spectrometer*, Nuclear Science, IEEE Transactions on **45** (1998), no. 3, 385 –389.

[28] X.Z. Li, Q.M. Wei, and B. Liu, *A new simple formula for fusion cross-sections of light nuclei*, Nuclear Fusion **48** (2008), 125003.

[29] A. G. Litvak, *High-frequency plasma heating*, American Institute of Physics, 1992.

[30] R Martin, MP Gryaznevich, SE Sharapov, TC Hender, R Gaffka, D Howell, T Wade, S Warder, and the MAST team, *Tae antenna for alfvén mode studies on mast*, Proceedings of the 39th EPS Conference on Plasma Physics, July 2007.

[31] K. G. McClements and A. Thyagaraja, *An energy-conserving scheme for solving the lorentz force equation*, Plasma Physics Note 2002/2.1, 2002.

[32] G.H. Miley, H. Towner, and N. Ivich, *Fusion cross sections and reactivities*, Tech. report, Illinois Univ., Urbana (USA), 1974.

[33] NVIDIA, *NVIDIA CUDA Programming Guide 4.2*, 2012.

[34] Department of Energy and Climate Change, *Severn tidal power feasability study*, Tech. report, 2010.

[35] ______, *Digest of united kingdom energy statistics*, Tech. report, 2012.

[36] The Institute of Engineering and Technology, *Hydroelectric power factfile*, Tech. report, 2007.

[37] House of Lords Economic Affairs Committee, *The economics of renewable energy*, Tech. report, 2008.

[38] James Oswald, Mike Raine, and Hezlin Ashraf-Ball, *Will british weather provide reliable electricity?*, Energy Policy **36** (2008), no. 8, 3212 – 3225.

[39] R Perez and W Boeglin, *A charged fusion product diagnostic for mast - design review*, Tech. report, 2012.

[40] National Grid plc, *National electricity transmission system (nets) seven year statement*, Tech. report, 2011.

[41] Mutsuo Saito and Makoto Matsumoto, *A deviation of CURAND: standard pseudo-random number generator in CUDA for GPGPU*.

[42] S. E. Sharapov, *Fast particle driven alfvn eigenmodes in tokamaks*, Fusion Science and Technology **57** (2010), no. 2T, 156 – 163, Proceedings of the Ninth Carolus Magnus Summer School on Plasma and Fusion Energy Physics.

[43] Graham Sinden, *Characteristics of the uk wind resource: Long-term patterns and relationship to electricity demand*, Energy Policy **35** (2007), no. 1, 112–127.

[44] D. R. Slaughter, *Fusion-product spectral linewidth in d-d, d-t, and [sup 3]he-d plasmas as an ion speed-distribution diagnostic*, Review of Scientific Instruments **60** (1989), no. 4, 552–561.

[45] L. Spitzer, *Physics of fully ionized gases*, Interscience, 1992.

[46] Keith Stammers and M.J. Loughlin, *The calibration of the MAST neutron yield monitors*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **562** (2006), no. 1, 521 – 530.

[47] S. Suckewer, H.P. Eubank, R.J. Goldston, J. McEnerney, N.R. Sauthoff, and H.H. Towner, *Toroidal plasma rotation in the plt tokamak with neutral-beam injection*, Nuclear Fusion **21** (1981), no. 10, 1301.

[48] MR Tournianski, RJ Akers, PG Carolan, and DL Keeling, *Anisotropic fast neutral particle spectra in the mast spherical tokamak*, Plasma physics and controlled fusion **47** (2005), 671.

[49] S. Väyrynen et al., *Irradiation of silicon particle detectors with MeV-protons*, (2009).

[50] Piero Venturi and Gianpietro Venturi, *Analysis of energy comparison for crops in european agricultural systems*, Biomass and Bioenergy **25** (2003), no. 3, 235 – 255.

[51] G.M. Voss, S. Davis, A. Dnestrovskij, A. Kirk, P.J. Knight, M. Loughlin, M.H. O'Brien, D. Sychugov, A. Tabasso, and H.R. Wilson, *Conceptual design of a component test facility based on the spherical tokamak*, Fusion Engineering and Design **83** (2008), no. 10-12, 1648 – 1653, Proceedings of the Eight International Symposium of Fusion Nuclear Technology - ISFNT-8 SI.

[52] John. Wesson and J. W. Connor, *Tokamaks / john wesson ; with contributions from j.w. connor ... [et al.]*, Clarendon Press, Oxford ; New York :, 1987 (English).