Computer simulations are the third tool, alongside theory and experiment, that scientists can use to understand natural phenomena. For systems with many degrees of freedom such as a laboratory plasma with more than 10^{20} particles, simulations provide a means in which to model the system and develop an understanding of the physics. In the limit of numerical noise, simulations provide the user with perfect diagnostics, the ability to make measurements of desired quantities without disturbing the system. This is an invaluable asset for testing theoretical predictions. Simulations are therefore a powerful utensil that compliment experiments and theory. Deep insight into the physics of plasmas has been gained by the use of computer simulations. In 1964 Landau damping of electrostatic waves was observed in a computational plasma experiment by Dawson [1]. This phenomena had been predicted by theory but had no empirical observations to support it at the time. There have been huge increases in computational power since then and as that power has increased so has the capability of computer simulations to explore the natural world. Today a whole host of plasma simulation codes are used to study tokamak plasmas. The application of these codes range from simulating large volumes of a tokamak, providing macroscopic quantities such as plasma density and temperature to tracking individual particles in a small sample of the divertor region to determine their behaviour as they approach the plasma-surface boundary. Simulations can enhance theoretical understanding and provide measurements that would be impossible to make in a physical experiment.

In the ideal case of unlimited computing power and memory, simulation codes would model plasmas by following the trajectories of every particle in the plasma as they move due to self-consistent electric and magnetic fields. Each particle in the system would interact with every other particle via the electric field. The simulations would be advanced in infinitesimally small time steps. At each time step the force acting on each particle due to every other particle would be calculated, this force would then go into Newton's equations of motion to supply each particle with a new velocity and position. Interaction forces would then need to be re-calculated and the cycle repeated until the simulation had run for the desired time. Even with the power of modern computers it is not possible to do this. A complete simulation of a tokamak plasma would be required to follow $\approx 10^{21}$ particles for millions of time steps, calculating the force at each time step. The amount of operations required to calculate these particle-particle interactions for n particles in the system scales on the order n^2. No computer is capable of handling this many

particles. However it is still possible to capture all of the relevant physics without such stringent computing demands.

Various computational techniques have been developed to simulate plasmas and they broadly fall into one of three categories. Particle-In-Cell (PIC) codes which follow the trajectories of individual particles in the plasma, Fluid models that treat the electrons and ions as separate fluids and Hybrid models that use a combination of fluid and PIC techniques to model the plasma [2]. Each method has certain advantages and disadvantages that determine where its use is applicable. These will be discussed below.

Out of the three categories PIC codes are considered to be the most fundamental way to model a plasma. Individual electrons and ions are tracked as they move across a spatial domain responding to self consistent electric and magnetic fields generated by the electric charge of the particles. The individual particles are in fact superparticles that represent many real particles. Rather than particle-particle interactions between every pair of particles in the simulation, superparticles deposit charge at discrete grid points along the domain. Other field quantities such as the electrostatic potential and electric field are then calculated from this charge density. The field is then interpolated back to the particles from the grid points in order to generate a new velocity and position. The discretisation of the field values as well as the use of superparticles allows modelling of the plasma from first principles. The essential physics of a real plasma can be captured with far fewer particles than are present in a real experiment. However in order to reduce statistical noise in the simulations large numbers of superparticles must be followed and there are certain stability criteria that must be met. As a result PIC simulations are compute-intensive, the simulations take a long time to run and this restricts PIC simulations to studying small regions of plasma.

Fluid codes ease the computational burden by treating the ions and electrons as separate fluids rather than individual particles. The plasma is described by the density, mean velocity and mean energy of the electrons and ions. Fluid codes solve the fluid equations which arise by taking the moments of the governing kinetic equations. The fluid equations require closure conditions that must be approximated [3]. This method is applicable when the plasma is in thermal equilibrium [4]. As fluid codes are less demanding on computer resources they can be used to study the evolution of large-scale instabilities in tokamak plasmas.

Hybrid models exist that combine aspects of fluid and PIC codes. A common implementation of this technique is to treat the electrons as a fluid while

modelling the ions kinetically. In this case the electrons are a neutralising background with a Boltzmann velocity distribution. This technique increases the required time step by orders of magnitude as only the motion of the ions has to be tracked rather than the faster motion of the electrons. This also reduces the number of particles that have to be followed. As a result these models can be run in less time but are useful if the user is only interested in the ion dynamics.

It is not possible to simulate a Langmuir probe without capturing the physics of the sheath [5]. The plasma sheath cannot be correctly modelled by fluid codes as the particle distributions inside the sheath are far from being in equilibrium [6]. The exact position of the sheath boundary is not well defined either as there is a smooth transition from the pre-sheath to the sheath near surfaces. PIC codes on the other hand make no assumptions about the distribution of the particles, they allow for any distribution function in phase space. By applying the fundamental equations the PIC method is able to preserve most of the physics. An alternative method to PIC is to directly integrate the Vlasov equation, which in 1D is given by

$$\frac{\partial f_s}{\partial t} + v\frac{\partial f_s}{\partial x} + \frac{q_s E}{m_s}\frac{\partial f_s}{\partial v} = 0 \tag{1}$$

where f_s is the phase space distribution function for a given species s. Direct Vlasov codes numerically solve the Vlasov equation on a phase space grid without the use of particles. The codes can be used for the same spatial regions and time scales as PIC codes but the algorithms are computationally expensive and suffer from numerical instability [7]. The benefits are that they do not suffer from numerical noise which effects PIC simulations due to the finite number of superparticles used [8]. Complex probe designs such as the Ball-pen probe, which will be discussed in chapter X, require the tracking of individual particles in order to interpret experimental data obtained by the probes. Therefore completely kinetic PIC simulations must be carried out. The rest of this chapter will proceed to describe the general PIC method before moving on to describe each step of the algorithm in more detail.

0.1 General Method

It is not possible with modern day computers to simulate a plasma by tracking all 10^{21} particles and calculating all the particle-particle interactions for each pair of particles in the system. The PIC scheme overcomes this problem

by using single particles to represent a given number of physical particles, so called superparticles and by introducing a spatial grid as shown in figure 1. The superparticles have the same charge to mass ratio as their real particle equivalents and so follow the same trajectories as that of a real particle. For the remainder of this chapter the word particle is synonymous with superparticle. The spatial domain of the simulation is discretised into grid cells. For simplicity equal length grid cells will be assumed but this is not a necessity for the PIC scheme. Varying grid sizes are often employed for computational efficiency, in order to resolve regions where steep gradients are expected to exist, and not over-resolve other regions. At these cell boundaries, field values, namely the charge density, electrostatic potential and electric field are calculated. However the particles see a continuous domain and are free to take any position within the simulation. The charge density at each grid point is determined by the locations of the particles. The other field values are calculated from this charge density. The mechanism in which particle positions across a continuous domain are converted to charge densities on a spatial grid and correspondingly the mechanism which translates field values from grid locations back to the individual particles is known as weighting and will be discussed in section 0.1. At the beginning of an electrostatic PIC simulation the grid is loaded with a particular distribution of particles depending on the plasma parameters to be modelled. It then follows an algorithm as depicted in figure 2. Each particle deposits charge to neighbouring grid points, Poisson's equation is solved to obtain a potential, the derivative of this is used to calculate the electric field and the field is mapped back to the particles which are then accelerated and moved. Once the particles have been moved, the PIC algorithm is complete, time is advanced by one time step and the whole cycle restarts by calculation of a new charge density based on the updated particle positions. This cycle will carry on until a certain time has been reached or steady-state has been obtained. The aforementioned steps are essential to any application of the PIC method to plasma simulations. Additional steps can also be added to the cycle based on the requirements of the user. These steps can include Monte Carlo collisions between the particles, absorption and injection of particles and other boundary effects such as sputtering, secondary electron emission and specular reflection. These steps are often added to the end of the PIC cycle once all the original particles in the system have been moved.

The introduction of the grid means particles interact with each other via a charge density rather than pair to pair interactions. For a simulation with

n particles the introduction of the grid reduces the amount of calculations required per time step to the order of n rather than n^2 as in the particle-particle scheme. Splitting a physical, continuous domain up into grid cells does have implications which need to be considered in order to ensure the simulation can still produce physically accurate results. The consequences of introducing a grid on to the domain and how this can still accurately represent a plasma are discussed in section 0.8. The grid allows the equations that determine field values and particle motion to be solved using finite differencing. A scheme that takes continuous differential equations and converts them so that they can be solved on discrete grids in space and time. The first step of finite differencing is to carry out a Taylor expansion. For a function $f(x)$, a discretised form of the first differential of the equation can be obtained as follows. First carry out a Taylor expansion in the forward direction

$$f(x + \Delta x) = f(x) + \Delta x \frac{\partial f(x)}{\partial x} + \frac{\Delta x^2}{2!} \frac{\partial^2 f(x)}{\partial x^2} + \frac{\Delta x^3}{3!} \frac{\partial^3 f(x)}{\partial x^3} \tag{2}$$

Repeating the procedure in the backwards direction

$$f(x - \Delta x) = f(x) - \Delta x \frac{\partial f(x)}{\partial x} + \frac{\Delta x^2}{2!} \frac{\partial^2 f(x)}{\partial x^2} - \frac{\Delta x^3}{3!} \frac{\partial^3 f(x)}{\partial x^3} \tag{3}$$

The two equations can be combined giving

$$\frac{\partial f(x)}{\partial x} = \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x} + \frac{\Delta x^3}{3!} \frac{\partial^3 f(x)}{\partial x^3} \tag{4}$$

Combining the two equations in this way is known as central differencing. The term proportional to Δx^3 is dropped so this method is second order accurate.

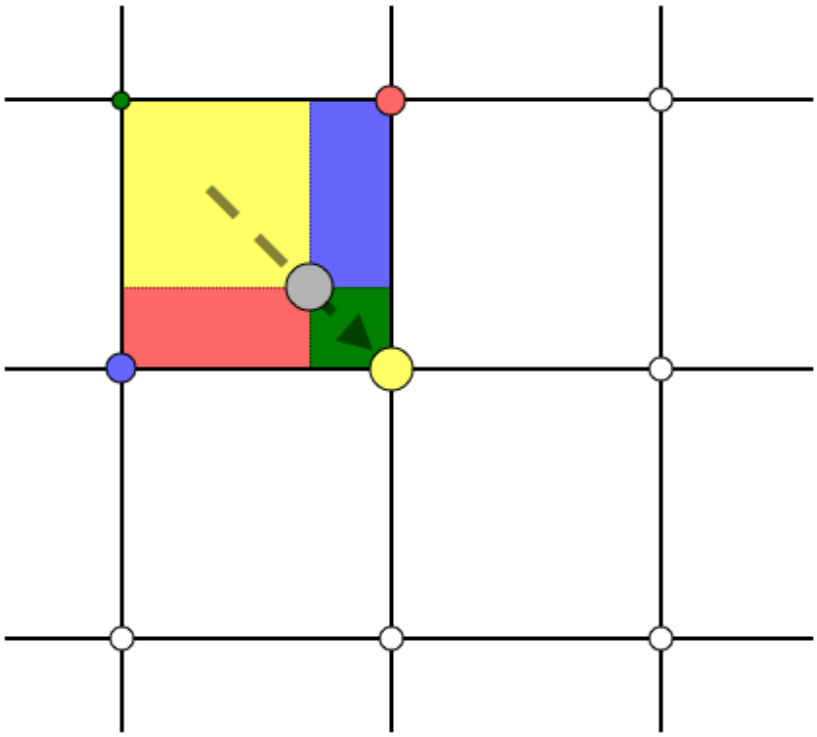

Figure 1: A representation of a two dimensional grid. The particle (grey circle) moves through the domain and deposits charge on the grid points. The area of the rectangle is proportional to the amount of charge deposited at each grid point.

Each step of the PIC cycle will now be detailed with VSim specific algorithms detailed where appropriate.

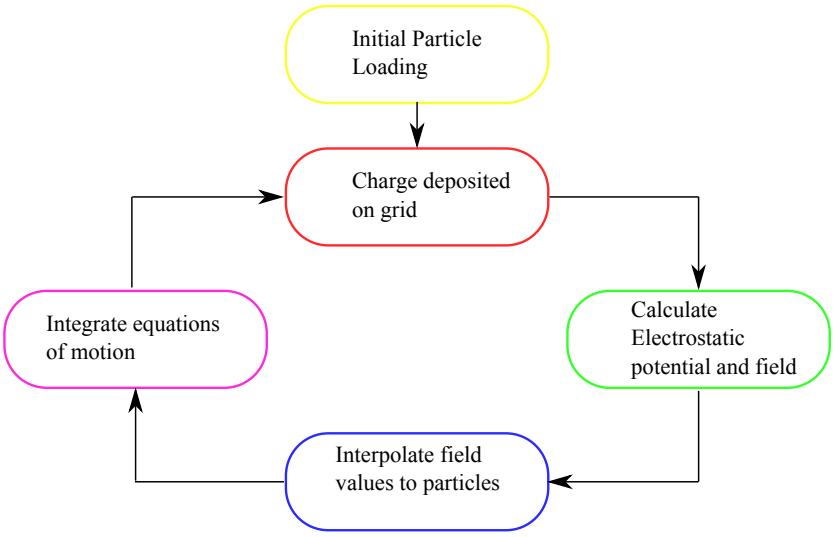

Figure 2: A flow chart of the essential steps of the PIC algorithm

Calculating the Charge Density

During this stage of the cycle, the charge density at each grid point is calculated from the individual particle locations. Weighting is the name given to the calculations which interpolate charge densities to the grid points from the continuous particle positions. Various weighting algorithms exist. The simplest weighting algorithm is to simply deposit all of a particles charge to its nearest grid point. This is the Nearest Grid Point (NGP) scheme. This is a zero order weighting method. Any particles within half a cell of a grid point are assigned to that grid point. Let Δx be the cell width, x the distance between the particle and grid point X and $W(X)$ denote the weighting at grid point X. In the NGP scheme

$$W(X) = \begin{cases} 1, & \text{if } |x| \leq \frac{\Delta x}{2} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

This is a computationally fast weighting method as it only requires one grid point look up per particle but this comes at the expense of adding noise to the simulation. As a particle moves away from its original grid point and into the region of a new grid point the grid density at the new grid point suddenly jumps up to have a value of one and the grid point it just left falls down to zero. This weighting scheme is not commonly deployed due to the

noisy transition as particles move between cells. It is possible to reduce this noise by spreading the charge of the particle over more grid points at the expense of increased computation time. First-order weighting also known as area weighting smooths the density and field fluctuations compared to NGP but is more computationally expensive as it requires two grid point look-ups for each particle. In this method, for a 1D simulation, each particle contributes charge to its nearest two grid points. The first step calculates the offset (δ) of the particle from the closest grid point to its left.

$$\delta = x_i - X_j \tag{6}$$

where x_i is the particles position and X_j the x-coordinate of the grid point. $x_i > X_j$ always. The charge assigned to the j^{th} grid point is then

$$W(j) = 1 - \delta \tag{7}$$

The charge assigned to the j+1 cell is

$$W(j+1) = \delta \tag{8}$$

Such that the total charge deposited to the grid equals the charge of the particle. This results in a much smoother contribution to the charge density as the particle propagates through the grid. This is the most commonly employed weighting scheme and is the default for VSim. Higher order weighting methods do exist such as quadratic and cubic splines, that are second order and third order accurate respectively. These schemes further smooth the non-physical noise at the expense of more computation time by increasing the number of grid points that a particle contributes its charge to. This does lead to complications at the edge of the simulations boundary where there are insufficient neighbouring grid points.

Calculating the Potential

Now the charge density is known at each grid point, Poisson's equation for electrostatics can be solved to obtain the electrostatic potential.

$$\nabla^2 \psi = -\frac{\rho}{\epsilon_0} \tag{9}$$

This can be solved numerically on a discretised grid using finite differencing. For a 1D simulation Poisson's equation can be expressed in finite difference

form

$$\frac{\partial^2\psi}{\partial x^2} = \frac{\psi(x+\Delta x) - 2\psi(x) + \psi(x-\Delta x)}{(\Delta x)^2} = -\frac{\rho_x}{\epsilon_0} \tag{10}$$

For clarity we rewrite (10) with labels based on the grid number j.

$$\frac{\partial^2\psi}{\partial x^2} = \frac{\psi_{j+1} - 2\psi_j + \psi_{j-1}}{(\Delta x)^2} = -\frac{\rho_j}{\epsilon_0} \tag{11}$$

The value of ψ at grid point j, (ψ_j), depends on the value of ψ at the two grid points either side of it (ψ_{j-1} and ψ_{j+1}), so the grid points are coupled together. In order to find the value of ψ_j at N different grid points requires the solution of N coupled linear equations. These coupled equations can be expressed in matrix form.

$$\begin{pmatrix} B_1 & C_1 & & & \\ A_2 & B_2 & C_2 & & \\ & A_3 & B_3 & C_3 & \\ & & \ddots & \ddots & \ddots \\ & & & A_N & B_N \end{pmatrix} \begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \vdots \\ \psi_N \end{pmatrix} = \begin{pmatrix} \rho_1 \\ \rho_2 \\ \rho_3 \\ \vdots \\ \rho_N \end{pmatrix} \tag{12}$$

Where $A = 1, B = -2$ and $C = 1$. A matrix like this with non-zero elements only on the diagonal and one place either side of it is known as a tri-diagonal matrix. The value of charge density (ρ_i) at each grid point is known as it was calculated in the previous step of the PIC algorithm. This matrix equation must now be solved in order to obtain the potential at each grid point. There are various numerical methods to find the solution and they can be divided into two categories: iterative methods and direct methods. VSim supports both type of solvers. For parallel simulations running on multiple cores, the iterative solver is employed. Before a solution can be found, the boundary conditions must be supplied as the grid points at the edge of the domain only have one neighbouring grid point. Two common choices for boundary conditions exist, Dirichlet boundary conditions where ψ_1 and ψ_N are set to a fixed value or Neumann boundary conditions where the gradient of the potential is fixed at the boundary. The implementation of Dirichlet boundary conditions is simple, B_1 and B_N are set equal to one, $C_1 = 0$ and $A_N = 0$. ψ_1 and ψ_N are then given the desired potential boundary values α and β respectively. The first and last matrix equations then read

$$1.\psi_1 + 0.\psi_2 = \alpha \tag{13}$$

$$0.\psi_{N-1} + 1.\psi_N = \beta \tag{14}$$

Neumann boundary conditions involve fixing the gradient of the potential(i.e. the electric field) at the edge of the domain. This could be implemented by setting $B_1 = -\frac{1}{\Delta x}$, $C_1 = \frac{1}{\Delta x}$ and $\rho_1 = \alpha$. Thus giving the first line in the matrix equation as

$$\frac{\psi_2 - \psi_1}{\Delta x} = \alpha \tag{15}$$

Once the boundary conditions have been supplied the tri-diagonal matrix equation can be solved.

Calculating the Electric Field

Once the potential is known at each grid point the electric field is easily found by calculating the gradient of the potential.

$$E = -\nabla\psi \tag{16}$$

which in one dimension becomes

$$E = -\frac{\partial\psi}{\partial x} \tag{17}$$

This can be discretised as before using finite differencing.

$$E_J = -\frac{\psi_{J+1} - \psi_J}{\Delta x} \tag{18}$$

The Particle Mover

The final step in the PIC cycle is to calculate a new position and velocity for each particle in the simulation based on the forces acting on them. In order to do this the following equations of motion must be solved

$$\vec{F} = m\frac{d\vec{v}}{dt} = q(\vec{E} + \vec{v} \times \vec{B}) \tag{19}$$

$$\vec{v} = \frac{d\vec{x}}{dt} \tag{20}$$

The particles positions and velocities can be found by integrating the differential equations (19) and (20) again using finite difference methods.

VSim uses a leap-frog scheme with a Boris advance [9] to push the particles. The leap-frog method involves offsetting the velocity by half a time step from the position. So the velocity of the particles is only known at half integer time steps while the positions are known at integer time steps. This requires the initial velocities of the particles to be moved back half a time step at the beginning of the simulation, a "de-acceleration", in order to have time centred velocities. This just requires calculating the fields as before. This method is known as Leap-frog because in order to calculate new positions requires a leap over the known velocity. The algorithm is demonstrated in figure 3.

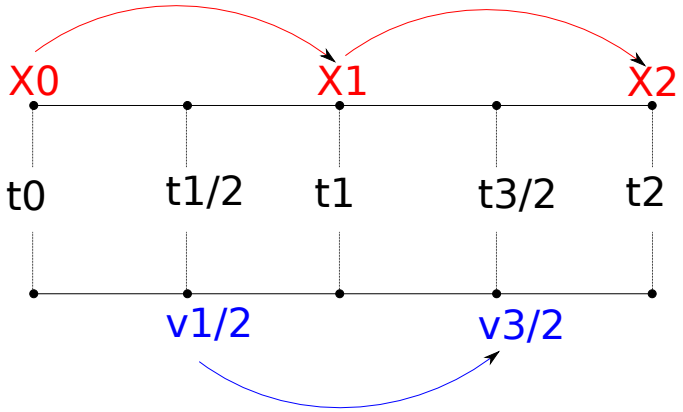

Figure 3: A graphical representation of the Leap-Frog scheme. Particle positions are known at integer time steps while velocities are known at half-integer time steps.

In discretised form the equations of motion become

$$\frac{x_{t+1} - x_t}{\Delta_t} = v_{t+1/2} \tag{21}$$

$$\frac{v_{t+1/2} - v_{t-1/2}}{\Delta t} = \frac{q}{m}\left[\vec{E}_t + \frac{(v_{t+1/2} + v_{t-1/2})}{2} \times B_t\right] \tag{22}$$

First equation 22 must be solved to get the new velocity of the particle ($v_{t+1/2}$), this is then inserted into equation 21 to obtain a new position for the particle (x_{t+1}). This is carried out for every particle in the system.

The most common implementation of the Boris scheme separates the effects of the electric and magnetic fields. Firstly half of the impulse due to the electric field is added to the particle's velocity creating an intermediate variable v^-

$$v^- = v_{t-1/2} + \frac{q}{m} E_t \frac{\Delta t}{2} \tag{23}$$

The magnetic field then acts on v^- to create a second intermediate variable v^+. The magnetic field only effects the rotation of the velocity vector not the magnitude.

$$\frac{v^+ - v^-}{\Delta t} = \frac{q}{2m}(v^+ + v^-) x B_t \tag{24}$$

Finally the second half of the electric impulse is added to v^+ to obtain the new velocity for the particle.

$$v_{t+1/2} = v^+ + \frac{q}{m} E_t \frac{\Delta t}{2} \tag{25}$$

The Boris scheme can be used to advance particles for an arbitrarily large number of time steps whilst remaining accurate and is therefore the de facto standard for particle movers [10]. The value for equation 25 is then substituted into equation 21 to obtain a new position for each particle.

Stability Conditions

PIC simulations utilise finite difference equations to find solutions to continuous differential equations on a discretised grid. The use of finite difference equations can provide accurate physical results provided certain stability conditions are met. To ensure the stability of the leap-frog particle advancing algorithm the time step (Δt) must be sufficiently small such that

$$\omega_p \Delta t < 2. \tag{26}$$

where ω_p is the plasma frequency. A time step of this size ensures stability of the algorithm but a further restraint must be placed upon the time step for it to provide accurate results [11].

$$\omega_p \Delta t < 0.2. \tag{27}$$

The second constraint determines the minimum grid spacing (Δx) that can be used.

$$\Delta x \leq \lambda_D \tag{28}$$

where λ_D is the Debye length. PIC codes can only resolve phenomena that are larger than Δx, anything smaller than this is smoothed over. PIC codes must resolve the Debye length in order to accurately capture the shielding effects. If this criteria is not met, non-physical numerical heating of the electrons will occur. The temperature of the electrons will increase until the Debye length is such that the stability criteria is met. The time step is further constrained by the criteria that no particle should be able to travel more than one grid cell in a given time step known as the Courant-Friedrich-Lewy condition [12].

$$\Delta t < \frac{\Delta x}{v_{max}} \tag{29}$$

where v_{max} is the speed of the fastest moving particle in the system. For high density, low temperature plasmas $\lambda_D \approx 10^{-6}$ while electron velocities can exceed $10^6 ms^{-1}$. This means thousands of grid cells may be necessary to simulate the $10mm^2$ tip of a Langmuir probe with time steps as small as $10^{-12}s$. Simulations with these demands can only be carried out on super-computers.

0.2 Langmuir Probe Simulations

Particle Loading and Injection

In Langmuir probe simulations, and any other plasma discharge simulation, particles are lost to the various collecting surfaces and must be replaced so that a constant density plasma can be simulated. It is therefore necessary to include an additional step into the PIC cycle in which new particles are introduced into the simulation so that the simulation can converge to a steady state. This step is referred to as particle injection. For clarity, particle loading only occurs at the very beginning of the simulation to distribute plasma throughout the domain, after the initial particles are loaded into the simulation the loading algorithm will not be used again. Particle injection takes place continuously throughout the simulation at the end of the PIC cycle once all other existing particles have been moved. It is desirable to inject particles at a rate that conserves the plasma density specified at the beginning of the simulation however it is not a necessity. The simulation will reach a steady-state density once the particle injection rate is balanced by the outflow rate of particles. The rate required to maintain a constant

density plasma can be estimated as

$$R_{constant-density} = v_{th,s}\Delta T * N/\Delta x \tag{30}$$

where $v_{th,s}$ is the thermal velocity of the species, ΔT the size of the time step, N the number of particles per cell specified at the beginning of the simulation and Δx the gridspacing. $R_{constant-density}$ gives the number of particles that must be injected at the beginning of each time step in order to preserve the specified plasma density. The rate will be different for electrons and ions as electrons move across grid cells in much less time due to their low mass, leading to a higher thermal velocity.

Langmuir probe theory is based on the assumption that the electrons and ions have a Maxwellian velocity distribution. Multiple algorithms exist to generate Maxwellian velocity distributions and these are detailed in Chapter 16 of Birdsall [11]. A velocity distribution generated with such an algorithm is shown in figure 4. The simulation will be required to run for many thousands of time steps in order to reach a steady state solution. For Langmuir probe simulations the requirements of an effective particle injection algorithm are to ensure the probe samples a constant temperature and density plasma as a Langmuir probe in a real plasma would. The algorithm must preserve the specified particle density and conserve the Maxwellian velocity distribution of particles. The former requirement is simply met by injecting particles at the rate at which they move across grid cells as given by equation 30. The latter requirement is not so easily met. PIC simulations model a small region of the plasma and use superparticles to sample the velocity distribution. As there are orders of magnitude differences between the number of superparticles followed and the true number of particles in a real plasma, losses of superparticles in a PIC simulation can dramatically change the velocity distribution over very short time scales. The problem is often enhanced by the lack of collisions in fusion-relevant PIC simulations. Due to the high densities and low temperatures of a SOL plasma, PIC simulations can only feasibly model small regions of the plasma. The dimensions of the simulation region are often smaller than any collisional mean free paths, so particles are able to move across the whole domain without experiencing a collision. Collisions drive particles towards Maxwellian distributions. Without this restoring force it is of crucial importance to sample from the correct velocity distribution during particle injection.

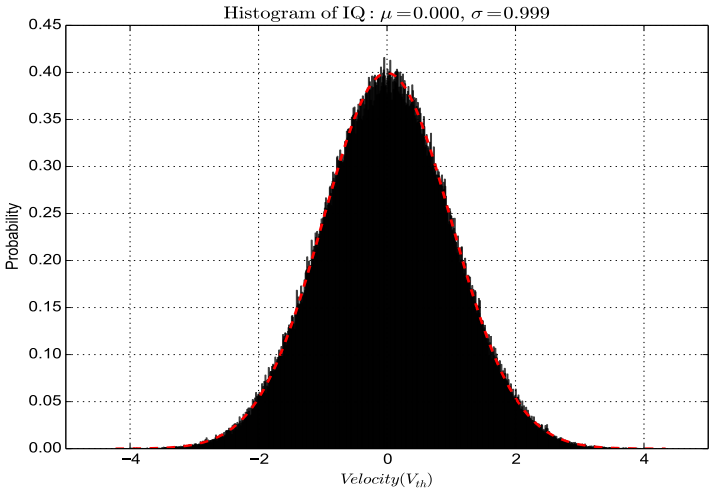

Figure 4: The velocity distribution of particles generated by the Maxwellian source function.

In early simulations, particle loading at the beginning of the simulation and particle injection throughout the simulation both sampled the same source function. For every particle a random velocity was chosen sampled from a Maxwellian distribution. However it was found in practise that this did not conserve the Maxwellian distribution. With increasing simulation time, the velocity distribution of the particles narrowed, resulting in an artificial 'cooling' of the plasma. The initial velocity distribution of particles and the distribution after many time steps is shown in figure 12. Although not anticipated the reasons for this observation are simple. Particles are initially loaded into the simulation domain with a range of velocities all sampled from a Maxwellian distribution. Provided there are enough superparticles in the domain, the distribution will be sufficiently represented by the finite number of particles. As time advances, particles exit the simulation once they reach an absorbing boundary layer and new particles enter in the injection phase of the PIC cycle. The fastest particles in the simulation are on average the first to leave as they quickly move across the domain to an absorbing surface. However if the injected particles are also sampled from the same Maxwellian distribution then it is far more likely that the injected particle will have a velocity close to zero rather than a high velocity in the tail of the distribution. As a result the fastest particles leave the simulation quickly to

be replaced by slow moving particles that reside in the simulation for a long time. This results in an under representation of the fastest particles in the simulation. The high energy tail disappears, the distribution narrows and the effective temperature of the plasma cools. Ideal probe theory assumes a constant temperature, Maxwellian plasma. Experimental measurements generally work on this assumption too. It is not possible to compare experimental data with simulation results if the simulation plasma is not at a constant temperature. A source function that conserves temperature as the simulation runs is desired. This source function must replace the particles at a rate proportional to their velocity. The fastest particles are required to be replaced more often while the slow moving particles not so much. Rather than the Maxwellian distribution that peaks at $v = 0$ the new distribution must fall to zero at this point as those particles should never leave the simulation. A method to investigate the required shape of the source function was established.

Temperature Conserving Source Function

A 1D simulation was adequate for the purposes of identifying the correct source function to use. The simulation domain is shown in figure 5.

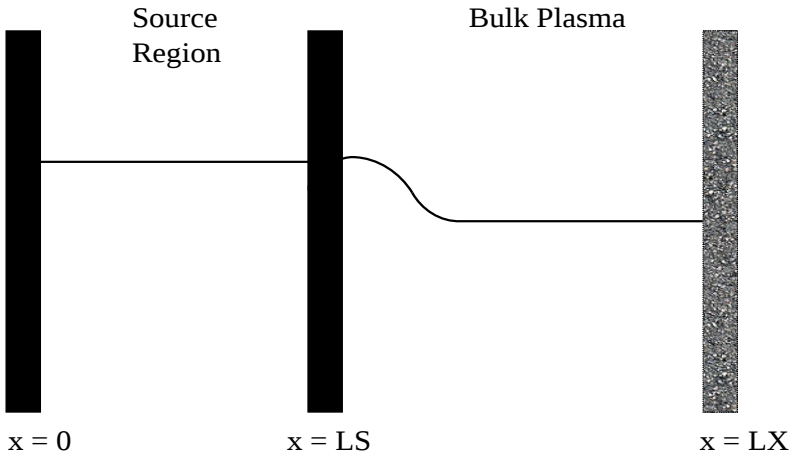

Figure 5: The simulation domain used to determine the temperature conserving source function. Particles born into the source region cannot escape the simulation. Before reflecting off the $x = LS$ plane, source particles are cloned. This clone escapes the source region and replenishes particles lost in the bulk plasma.

The model tracks a length of plasma , hundreds of Debye Length (λ_D) long and follows the motion of all ions and electrons as they move in self-consistent electric fields. The presence of a magnetic field impacts the source function so to begin with only electrostatic simulations were carried out. On one side of the domain is an absorbing surface that represents the probe. This surface can be held to any potential. Any charged particles that hit the surface are deleted from the simulation and their current recorded. On the opposite side of the domain is a source region of plasma. The purpose of this source region is to supply the bulk plasma with new particles to replenish those lost to the sides. At the beginning of the simulation, a quasi-neutral plasma with a Maxwellian distribution fills the entire domain. Any particles born into the source region ($0 \leq x \leq LS$) are trapped in the source region for the duration of the simulation. These source particles travel back and forth in the source region and are reflected at the boundries $x = 0$ and $x = LS$. Both edges of the source are held at the same potential (V_{source}) which fixes the plasma potential. As there is no potential difference between the two sides and no particles can escape, the plasma source remains at the

temperature and density specified at the beginning of the simulation. Both the source plasma and the bulk plasma maintain the Maxwellian distribution throughout the duration of the simulation as shown in figure 6.

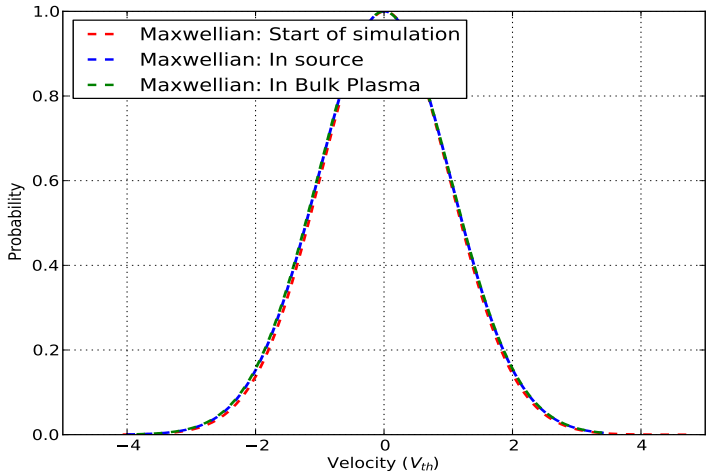

Figure 6: The velocity distribution of particles in different regions of the plasma at the end of the simulation. The Maxwellian distribution is conserved.

Any source particle striking the $x = LS$ boundary is copied before being reflected. This copy is identical to the original particle, having the same charge, mass, position and velocity but is not reflected at the source boundary. Instead the copy travels into the bulk plasma, allowing the source region to replenish the bulk plasma. By looking at the velocity distribution of particles exiting the source, it is possible to determine the form of the source function required to conserve a Maxwellian plasma. The distribution of particles leaving the source region is equivalent to the distribution of particles that would exit the simulation of a bulk plasma. It is these velocities that must be sampled in order to maintain a constant temperature plasma. The velocity distribution of particles leaving the source is shown in figure 7

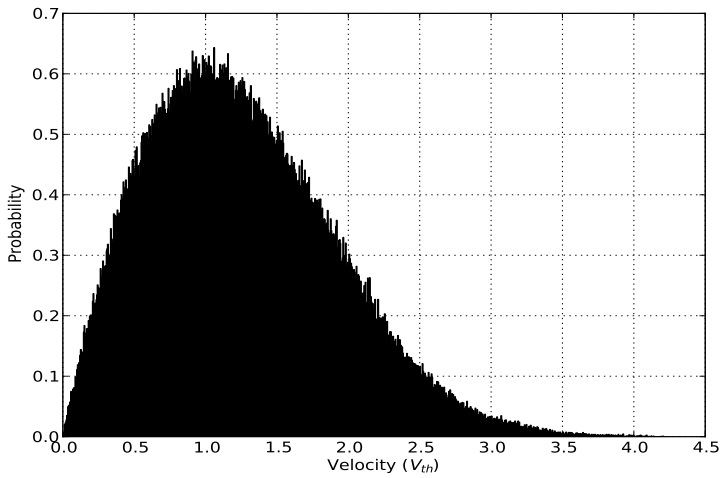

Figure 7: The velocity distribution of particles that exit the source region.

This is the Emmert source function [13] described by

$$S_(v) = \frac{m_i v}{T_s} exp\left(-\frac{m_i v^2}{2T_s}\right) \tag{31}$$

The shape of the source function generated from equation 31 is shown in figure 8.

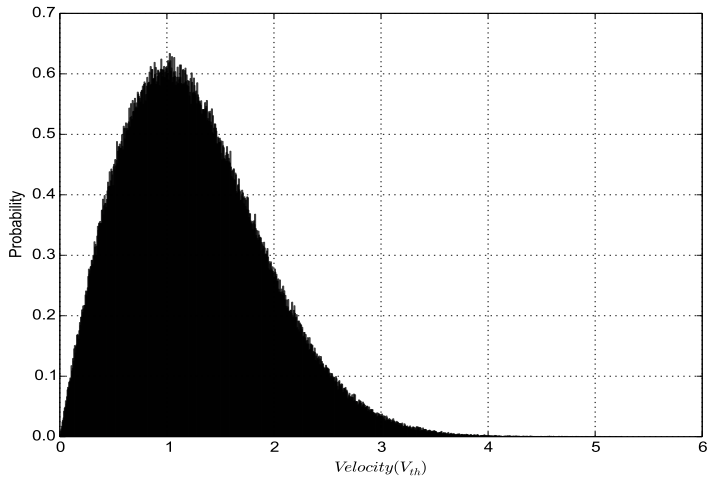

Figure 8: The Emmert source function only replenishes the high energy tail of the distribution. It is identical to the distribution of particles that escape the reflecting domain in my simulations.

As well as providing a source function, this method validated the rate of injection. It was found that the ratio of electrons exiting the source to ions exiting the source was equal to the ratio of their thermal velocities. Now a temperature conserving source function has been determined there is no need for the source region. Simulating the source region requires tracking many particles, which is feasible in 1D simulations but would take up too much computer time for higher dimensions. The simulation domain without the source function is shown in figure 9.

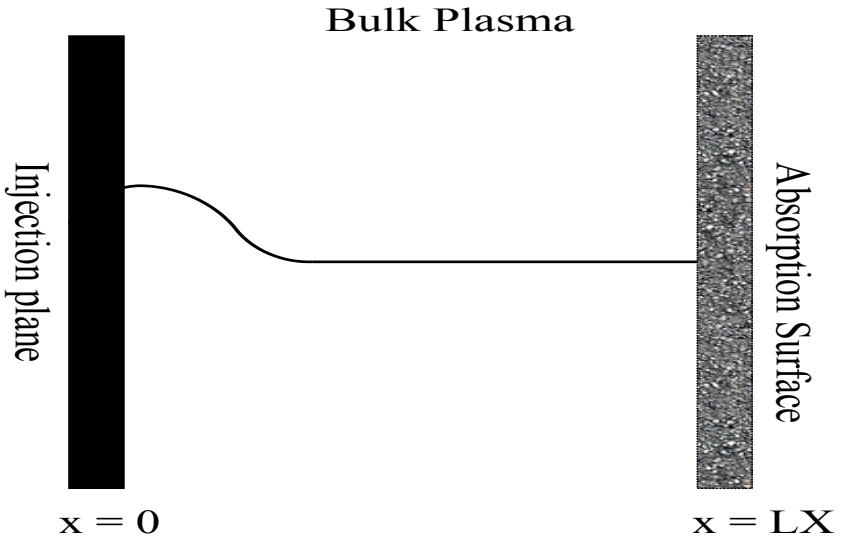

Figure 9: The domain for the 1D probe simulations used to test the source function.

A plasma source is required to replenish particles that are lost to the probe. This source can be held at any potential, I choose 0V, this sets the plasma potential. The system reaches a steady state over a few ion transit times at which point the current drained by the probe surface reaches a constant value and the plasma density stabilises.

0.3 Generating a Velocity from a Source Function

The Emmert source function is described by equation 31. The source function is a cumulative distribution function. As this distribution function is invertible it can be used to generate a velocity. The distribution is in the form of a Weibull distribution

$$W(v) = \alpha\beta v^{\beta-1} \exp -\alpha v^{\beta} \tag{32}$$

with $\beta = 2$ and $alpha = \frac{m}{2qT_s}$. By applying the fundamental transformation law of probabilities, it is possible to use a randomly generated number x, where $0 \leq x \leq 1$, from a uniform distribution and transform this into a randomly generated number belonging to the Weibull distribution. For two probability distribution functions $p(x)$ and $S(v)$ the fundamental transformation law states

$$|p(x)dx| = |dS(v)dv| \tag{33}$$

or

$$S(v) = p(x)\left|\frac{dx}{dv}\right| \tag{34}$$

as x is from a uniform distribution, $p(x)$ is constant and so

$$S(v) = \left|\frac{dx}{dv}\right| \tag{35}$$

therefore

$$x = \int_0^v S(v)dv \tag{36}$$

Integrating the Weibull distribution gives

$$x = 1 - \exp -\alpha v^{\beta} \tag{37}$$

Inverting this gives a relation for the velocity in terms of the random number x

$$v = \left[-\frac{1}{\alpha}\ln(1-x)\right]^{\frac{1}{\beta}} \tag{38}$$

Substituting in values for α and β gives

$$v = \left[\frac{-2qT}{m}\ln(1-x)\right]^{\frac{1}{2}} \tag{39}$$

0.4 Reproducing Ideal Probe Theory

In order to test the suitability of PIC codes to the study of Langmuir probe behaviour, multiple simulations were carried out to reproduce predictions made from ideal probe theory. The main equations of ideal probe theory will now be summarised before simulation results are presented. A complete description of ideal probe theory can be found in chapter two. In experiments, a probe IV curve is produced by sweeping the voltage across the probe and measuring the collected current for each voltage. The total current reaching the probe will be the sum of the electron and ion currents to the probe.

$$I_{Probe} = I_{ion} + I_{electron} \tag{40}$$

The magnitude of these currents will vary with the applied probe voltage. If the probe is biased such that it is negatively biased with respect to the

plasma potential V_{plasma} then ions will be collected at their saturated value and only the portion of the electron population with sufficient energy to overcome the negative bias will be able hit the probe. As the probe bias becomes more negative, less of the electron population can reach the probe. Eventually no electrons reach the probe and the probe only collects the ion saturation current given by

$$I_{sat}^{+} = n_s e v_B A \tag{41}$$

where n_s is the ion density at the sheath edge, e is the fundamental charge and A the collection area of the exposed probe tip. If the probe is biased sufficiently positively with respect to V_{plasma} no ions will be able to reach the probe. The current collected by the probe is then the electron saturation current given by

$$I_{sat}^{-} = \frac{1}{4} n_s e \sqrt{\frac{8KT_e}{\pi M_e}} \tag{42}$$

Provided the probe bias meets the following criteria

$$V_{probe} \leq V_{plasma} \tag{43}$$

The current to the probe will consist of the ion saturation current plus a reduced electron current

$$I_{probe} = I_{sat}^{+} + I_{sat}^{-} \exp\left(\frac{e(V_{probe} - V_{plasma})}{T_e}\right) \tag{44}$$

This can be rearranged to simplify the measurement of T_e.

$$\ln(I_e) = \frac{V_{probe}}{T_e} + \ln(I_{sat}^{+}) \tag{45}$$

Taking $V_{plasma} = 0$. By plotting the natural logarithm of the electron current against the probe bias and measuring the gradient it is possible to use probe measurements to determine the electron temperature. With a working source function that preserves plasma temperature, simulations should be able to reproduce this important prediction of probe theory. Multiple simulations were run using the domain as shown in figure 9. In each simulation a different bias voltage was applied to the probe, the simulation was run to steady state and the electron and ion current reaching the probe was recorded. The source temperature of the electrons and ions was set to 5eV. Using ideal probe theory

allowed the correct temperature to be measured by the simulated probe as shown in figure **??**.

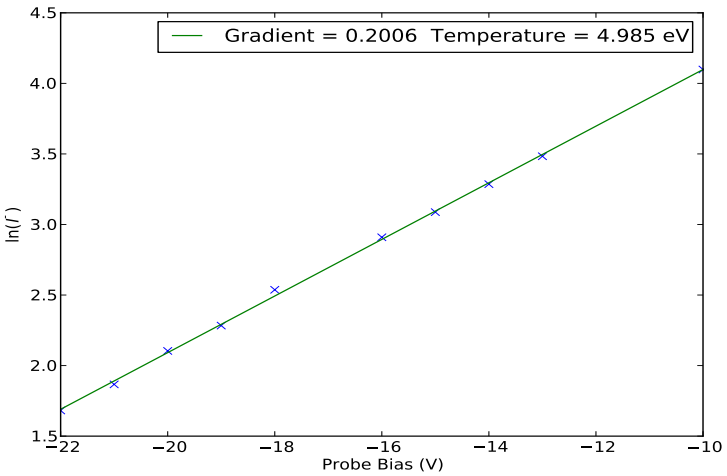

Figure 10: The log of the electron current against probe bias voltage for the Emmert source function. The gradient of the natural log plot is equivalent to the reciprocal of the electron temperature. Using the Emmert source function to inject new particles allows the probe to measure the correct source temperature.

For comparison the simulations were repeated using the Maxwellian source function rather than the temperature conserving Emmert function to inject particles. As can be seen in figure 11 the probe measures a lower electron temperature, a consequence of the narrowing of the velocity distribution. Figure 12 compares the velocity distribution of the electrons at the end of simulation when sampling from each distribution. It is clear that the correct temperature is conserved when sampling from the Emmert source function but a loss of temperature is detected when using the Maxwellian function. For the Maxwellian runs, the probe measured a temperature somewhere in between the original specified temperature and the final plasma temperature.

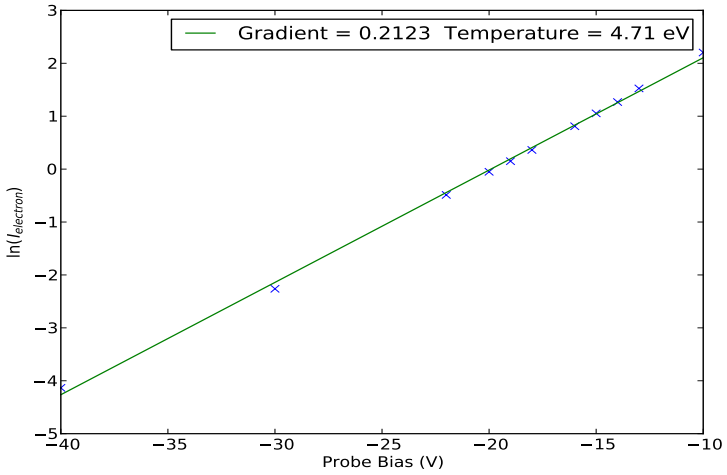

Figure 11: The log of the electron current against probe bias voltage for the Maxwellian source function.The gradient of the natural log plot is equivalent to the reciprocal of the electron temperature. Using the Maxwellian source function to inject new particles results in the probe measuring a temperature that is lower than the specified source temperature.

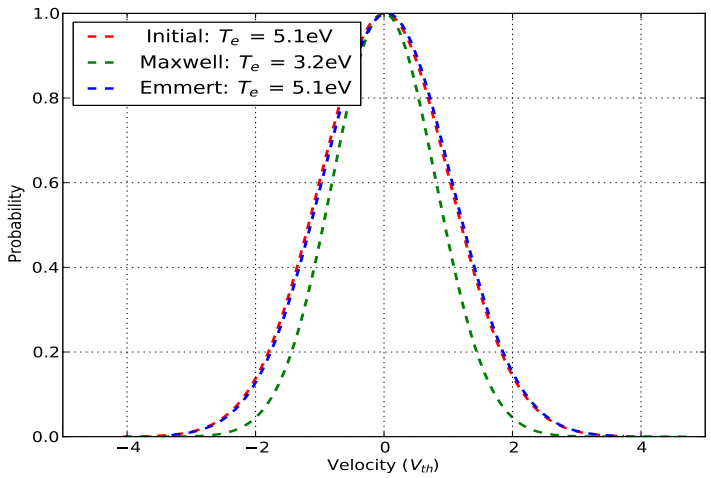

Figure 12: A comparison of the electron velocity distributions. Red- The distribution of electrons at the beginning of both simulations. Green- The electron distribution at the end of the simulation using a Maxwellian source function. Blue- The electron distribution at the end of the simulation using the Emmert source function.

The ability to reproduce ideal probe theory confirms the correct choice of source function. This was a crucial starting point. With a reliable source function, scenarios with more complicated physics can now be explored.

0.5 Source Function for Magnetised Plasmas

In order to simulate Langmuir probes in fusion plasmas a magnetic field must be added to the simulations. In the previous simulations, without a magnetic field, it was found that the Emmert source function was correct to use in the x-direction as the rate at which particles were lost to the probe was directly proportional to the v_x component of their velocity. v_y and v_z were sampled from a Maxwellian distribution. The presence of the magnetic field, which does not have to lie along one of the Cartesian axis adds an additional step to the particle injection algorithm. Particle velocities are generated relative to the magnetic coordinates, $v_{\parallel}, v_{\perp,1}$ and $v_{\perp,2}$. These velocities must then be transformed so that they lie along the coordinate axis. In a magnetised plasma the rate at which particles are lost to the wall now

depends on their parallel velocity so this velocity should be generated from the Emmert distribution. The two perpendicular velocities are again generated from Maxwellian distributions. The field orientated velocities $v_{\parallel}, v_{\perp,1}$ and $v_{\perp,2}$ are then converted to Cartesian velocities v_x, v_y, v_z by the following transformations [14].

$$v_x = v_{\parallel} b_x + v_{\perp,2} \sqrt{b_y^2 + b_z^2} \tag{46}$$

$$v_y = v_{\parallel} b_y + \frac{v_{\perp,1} b_z - v_{\perp,2} b_x b_y}{\sqrt{b_y^2 + b_z^2}} \tag{47}$$

$$v_z = v_{\parallel} b_z - \frac{v_{\perp,1} b_y + v_{\perp,2} b_x b_z}{\sqrt{b_y^2 + b_z^2}} \tag{48}$$

These transformations have been tested to ensure they generate correct parallel velocities for the particles.

0.6 Floating Wall Conditions

In simulations of Langmuir probes it is desirable to implement floating surfaces to represent a probe operated in floating mode or a surrounding divertor tile. To include floating boundary conditions in VSim requires the use of heavy particles. These are particles with the same charge as that of an ion or an electron but a mass that is sufficiently large such that the particle will not move due to the forces imparted upon it throughout the duration of the simulation. A mass of 1kg is sufficient. If a particle comes into contact with a floating surface in the simulation, the particle is deleted from the simulation. A heavy particle is then emitted at this point of absorption, so the charge of the absorbed particles build up on the wall. A charge and electrostatic potential naturally build up on the floating surface this way without having to impose a floating potential. To test this boundary condition a simulation was carried out with the simulation domain used in section 0.4. Instead of biasing the right hand side of the simulation to a set probe bias potential, the wall now absorbed particles that hit it and re-emitted heavy particles in their place. The simulation was run until the wall reached a constant potential and drained a steady current. The current and potential on the wall agree well with what is expected based on the electron temperature as shown in figure 13.

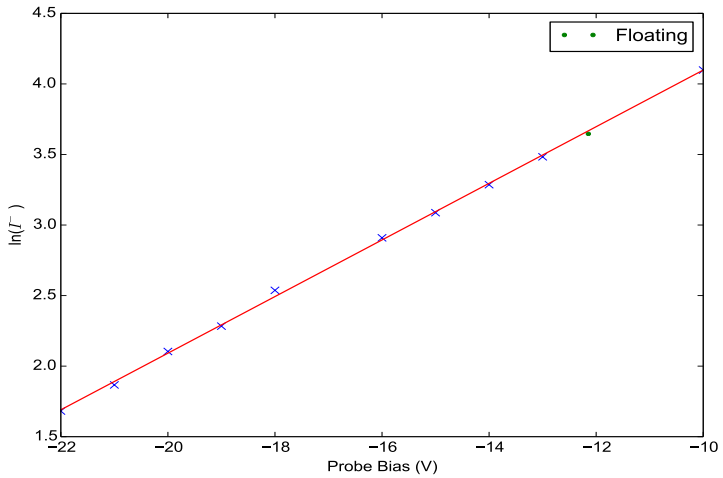

Figure 13: The green circle represents the floating potential and current drained by the probe using the heavy particle floating boundary condition.

0.7 Assumptions the model makes

In our simulations there are no collisions between the charged particles. Typical plasma parameters used in the simulations are of the order $n_e \approx 1 \times 10^{18}$ and $T_e \approx 10eV$. The simulation domain usually spans no longer than $5mm$ in either direction. The demand that the grid must resolve the Debye length means it is not feasible to simulate larger lengths than this as the required amount of grid points will be too computationally demanding. The mean-free path (λ) for both electrons and ions is calculated to far exceed the length of typical simulation domains for the plasma parameters. Using equations from Wesson [15] it is calculated that $\lambda_{electron} = 7.2cm$ and $\lambda_{ion} = 10cm$ for the stated plasma parameters. As a result particles can travel across the entire simulation domain multiple times without experiencing a collision. It has also been assumed that there are no neutrals or impurities present so the plasma consists of electrons and singly charged ions.

0.8 Consquences of the Computational Grid

Finite Sized Particles

Rather than the point sized particles of a physical plasma, particles in a PIC simulation are finite sized clouds of uniform charge. Finite sized particles are a direct result of weighting particles on to the grid. The weighting method determines the effective size and shape of the particle as viewed by grid. For the first order weighting scheme described above, a fraction of the clouds charge which is in the J^{th} cell is weighted to the X_J grid point and the rest of it goes to the X_{J+1} grid point. This gives the particle a triangular shape, so the particle is effectively a triangular cloud of uniform charge centred at x_i with a width of $2\Delta x$ as it is able to influence grid points either side of it.

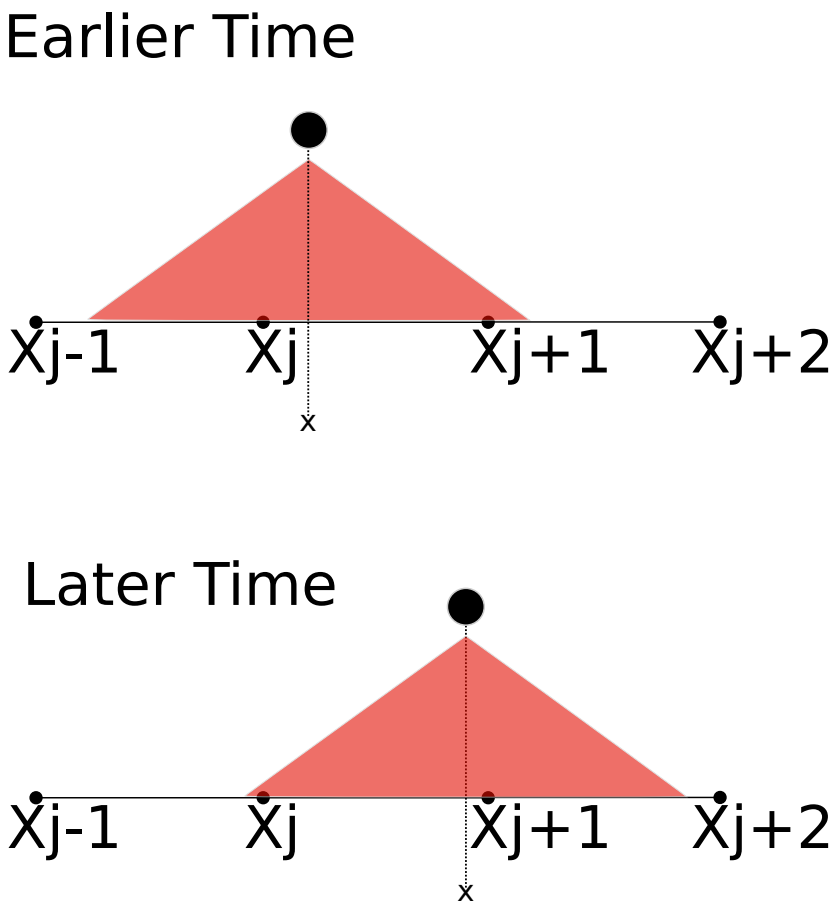

Figure 14: The effective shape of a particle at position x as seen by the grid. At the earlier time the majority of the particles charge is deposited to the X_J grid point as this is the particles nearest grid point. At a later time the particle has advanced and now deposits most of its charge to grid point X_{J+1}.

Typical plasmas are considered weakly coupled systems as they have a large number of particles in a given cube of volume λ_D^3. This means the trajectory of each particle is affected by a very large number of particles, so the trajectory is smooth as many particles contribute to the electric field. Effects from close encounters with other particles do not dominate particle motion. A strongly coupled system on the other hand is one in which there are few particles per Debye length. The trajectory of a particle is then strongly affected by a collision with another particle and the electric field is noisy and irregular. PIC codes manage to replicate the physics of real plasmas despite

using fewer particles by using finite sized particles. Finite sized particles interact more weakly during close encounters than the point size particles of a real plasma. Once finite size particles begin to overlap, the force they exert on each other decreases, reaching zero once the particles fully overlap [16]. By reducing the interaction among particles PIC codes can successfully model weakly coupled plasmas. In terms of long range interactions, particles in a PIC code only interact with particles in their own and closely neighbouring grid cells. This is physically justified as the grid cell is on the order of a Debye length. In a physical plasma the field from a localised charge is screened out at distances greater than a Debye length.

Self Force

The presence of the spatial grid can introduce non-physical effects into the simulation, an example of which is the self force. The self force is the name given to the phenomenon in which a charged particle becomes aware of its own field and so repels itself. This non-physical force is inherent in any system that uses the spatial grid to work out charge density and electric field values. Imagine a charged particle placed in between two grid points on a one dimensional grid. Let the particle reside closer to the left grid point to begin with. Using the first order weighting scheme the particle will deposit more of its charge on to the left grid point than it will on to the right. This density will then be used to work out a potential and finally a field which is then interpolated back to the particles position, not necessarily using the same weighting scheme but for now consider it is the same. The particle will feel a larger force from the grid point on its left than it will do from the grid point on its right due to the relative distances and as a result it will be repelled from the left grid point and head towards the right. Although this argument assumed a first order weighting scheme, the self force is present with any method of weighting. It is possible to prove that the self force is introduced by the addition of a spatial grid rather than from any approximations made in using finite difference methods to solve Poissons equation. The following method is adapted from the textbook 'Numerical particle-in-cell Methods: Theory and Applications' [17].

The electric potential due to a point charge with charge q at a distance r is given by

$$\psi = \frac{1}{4\pi\epsilon_0}\frac{q}{r} = \frac{Cq}{r} \tag{49}$$

where C is a constant. Now introduce a spatial grid with spacing dx. Let the particle be a distance r from the grid point on the left and so a distance $dx - r$ from the point on the right. The particle will deposit a charge ρ_1 on the left grid point and a charge ρ_2 on the right where

$$\rho_1 = q\frac{(dx - r)}{dx} \tag{50}$$

and

$$\rho_2 = q\frac{r}{dx} \tag{51}$$

The particle will feel a potential that is a combination of the potential from the two grid points. From (49)the potential at the particle is

$$\frac{Cq}{dx}\left(\frac{dx - r}{r} + \frac{r}{dx - r}\right) \tag{52}$$

The force at the particle is given by

$$F = -\frac{\partial\psi}{\partial r}q = Cq^2 dx\frac{dx - 2r}{r^2(dx - r)^2} \tag{53}$$

Despite the potential and electric field having been solved exactly without finite difference methods there still exists a self force due to the introduction of a spatial grid. This force is zero if the particle is at the midpoint of a cell ($r = \frac{dx}{2}$) and otherwise acts to repel the particle from its closest grid point.

It is possible to quantify this force in the case of a complete PIC simulation where finite difference methods are used. As before Poisson's equation is discretised

$$-\rho_J = \frac{\psi_{J+1} - 2\psi_J + \psi_{J-1}}{(dx)^2} \tag{54}$$

This equation can be solved analytically without a matrix equation.

$$\psi_J = \frac{I - J}{I}A + \frac{J}{I}B + \frac{dx^2}{I}\left[(I - J)\sum_{k=1}^{J} k\rho_k + J\sum_{k=J+1}^{I-1}(I - k)\rho_k\right] \tag{55}$$

Where I is the total number of grid cells , A is the potential at the left hand side boundary and B the potential at the right hand side. This will now be carried out for one particle. Let the particle lie between grid points $x_{\alpha-1}$ and x_α at position x. The offset (σ) is given by

$$\sigma = \frac{x - x_{\alpha-1}}{dx} \tag{56}$$

Using (55) it is possible to calculate the potential at the grid points either side of the particle.

$$\psi_\alpha = \frac{I-\alpha}{I}A + \frac{\alpha}{I}B + \frac{dx^2}{I}(I-\alpha)\left[(\alpha-1)\rho_{\alpha-1} + \alpha\rho_\alpha\right] \tag{57}$$

$$\psi_{\alpha-1} = \frac{I-\alpha+1}{I}A + \frac{\alpha-1}{I}B + \frac{dx^2}{I}(\alpha-1)\left[(I-\alpha+1)\rho_{\alpha-1} + (I-\alpha)\rho_\alpha\right] \tag{58}$$

Similar expressions can be derived for $\psi_{\alpha+1}$ and $\psi_{\alpha-2}$. These can then be used to derive the value of the electric field at the adjacent grid points.

$$E_\alpha = -\frac{\psi_{\alpha+1}-\psi_\alpha}{dx} = \frac{A-B}{Idx} - \frac{dx}{I}\left[(1-\alpha)\rho_{\alpha-1} - \alpha\rho_\alpha\right] \tag{59}$$

$$E_{\alpha-1} = -\frac{\psi_\alpha-\psi_{\alpha-1}}{dx} = \frac{A-B}{Idx} - \frac{dx}{I}\left[(1-\alpha)\rho_{\alpha-1} + (I-\alpha)\rho_\alpha\right] \tag{60}$$

Using the first order weighting method, the force at the particle E_i will be given by

$$E_i = E_{\alpha-1}(1-\sigma)+\sigma E_\alpha = \frac{A-B}{Idx} - \frac{dx}{I}\left[(1-\alpha)\rho_{\alpha-1} + (I-\alpha-I\sigma)\rho_\alpha\right] \tag{61}$$

The self force is inherent in PIC codes and always acts to repel a particle from its nearest grid point. Interestingly this equation shows the impact that boundary conditions have on the self force. The force not only depends on the location of the particle in the grid cell but also in which cell it lies in. Provided there are sufficient amount of particles the self force will be negligible.

1 References

References

[1] John M. Dawson. Thermal relaxation in a onespecies, onedimensional plasma. *Physics of Fluids*, 7(3), 1964.

[2] H C Kim, F Iza, S S Yang, M Radmilovi-Radjenovi, and J K Lee. Particle and fluid simulations of low-temperature plasma discharges: benchmarks and kinetic effects. *Journal of Physics D: Applied Physics*, 38(19):R283, 2005.

[3] E. Westerhof and J. Pratt. Closure of the single fluid magnetohydrodynamic equations in presence of electron cyclotron current drive. *Physics of Plasmas*, 21(10):102516, October 2014.

[4] P.A. Sturrock. *Plasma Physics: An Introduction to the Theory of Astrophysical, Geophysical and Laboratory Plasmas*. Stanford-Cambridge program. Cambridge University Press, 1994.

[5] V Hrubỳ and R Hrach. Selected techniques for langmuir probe modelling in low-temperature plasmas. In *WDS 2010-Proceedings of Contributed Papers. Proceedings of the 19th Annual Conference of Doctoral Students, held 1-4 June 2010, in Prague. Edited by Jana Safránková and Jirí Pavlû. Part II-Physics of Plasmas and Ionized Media (ISBN 978-80-7378-140-8) MATFYZPRESS, Prague, 2010., p. 38-41*, volume 1, pages 38–41, 2010.

[6] Janez Krek, Nikola Jeli, and Joe Duhovnik. Particle-in-cell (pic) simulations on plasmasheath boundary in collision-free plasmas with warm-ion sources. *Nuclear Engineering and Design*, 241(4):1261 – 1266, 2011. International Conference on Nuclear Energy for New Europe 2009.

[7] D. Nunn. A novel technique for the numerical simulation of hot collision-free plasma; vlasov hybrid simulation. *J. Comput. Phys.*, 108(1):180–196, September 1993.

[8] G. Snchez-Arriaga and D. Pastor-Moreno. Direct vlasov simulations of electron-attracting cylindrical langmuir probes in flowing plasmas. *Physics of Plasmas*, 21(7), 2014.

[9] J. P. Boris. Relativistic plasma simulation-optimization of a hybrid code,. *Proceedings of the Fourth Conference on Numerical Simulations of Plasmas*, 20, 1970.

[10] Hong Qin, Shuangxi Zhang, Jianyuan Xiao, Jian Liu, Yajuan Sun, and William M. Tang. Why is boris algorithm so good? *Physics of Plasmas*, 20(8), 2013.

[11] Charles K. Birdsall and A. Bruce Langdon. *Plasma physics via computer simulation*. Series in plasma physics. Taylor & Francis, New York, 2005. Originally published: New York ; London : McGraw-Hill, 1985.

[12] James Harrison. Characterisation of detached plasmas on the mast tokamak. September 2010.

[13] G A Emmert, R M Wieland, A T Mense, and J N Davidson. Electric sheath and presheath in a collisionless, finite ion temperature plasma. *Physics of Fluids*, 23(4), 1980.

[14] Michael Komm. Studies of tokamak edge plasma and its interaction with the first wall. September.

[15] John. Wesson and J. W. Connor. *Tokamaks / John Wesson ; with contributions from J.W. Connor ... [et al.].* Clarendon Press Oxford ; New York, 1987.

[16] Giovanni Lapenta. *Particle In Cell Methods With Application to Simulations in Space Weather.* Cambridge University Press, New York, NY, USA, 1992.

[17] Nikolaevich Grigorev, Yu. N. Grigoryev, V. A. Vshivkov, and M. P. Fedoruk. *Numerical "particle-in-cell" Methods: Theory and Applications.* Theory and Applications. VSP, 2002.