UK Atomic Energy Authority

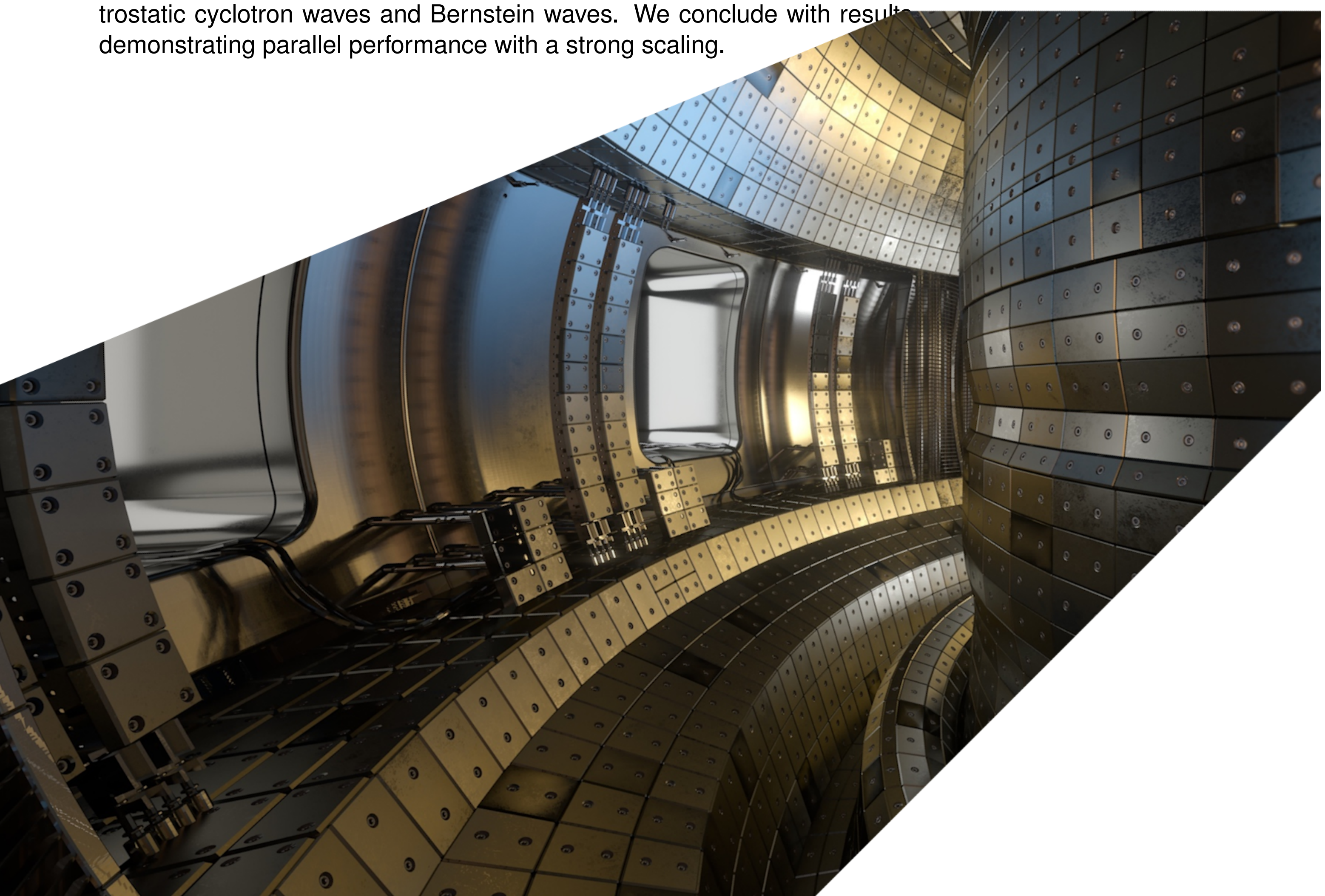
ExCALIBUR

1-D and 2-D Particle Models

M4c.1 Version 1.00

Abstract

The report describes work for ExCALIBUR project NEPTUNE at Milestone M4c.1. It describes 1-D and 2-D particle models with up to 3 velocity-space dimensions specifically the algorithms, and the methodology applied to assess the correctness and validity of our implementations. We developed of a SYCL-based particle framework, *NESO-Particles*, to represent particle data and motion on unstructured meshes, consistent with a parallelisation strategy based on domain decomposition and utilising MPI. Particle data is projected onto 2-D finite element representations within the Nektar++ software ecosystem, thereby completing a two-way coupling between Nektar++ and our in-house developed *NESO-Particles* library. First we set out the details of the Particle-In-Cell (PIC) algorithm needed for NEPTUNE, where the 'cells' correspond to finite elements. Thereafter we demonstrate the ability of this approach to reproduce known physical results, namely, (1) growth rates of the two-stream instability, (2) the helical particle trajectories expected in a uniform applied magnetic field, and in the plasma context, (3) dispersion relations for warm Langmuir waves, electrostatic cyclotron waves and Bernstein waves. We conclude with results demonstrating parallel performance with a strong scaling.



UKAEA REFERENCE AND APPROVAL SHEET

	Client Reference:	
	UKAEA Reference:	CD/EXCALIBUR-FMS/0070
	Issue:	1.00
	Date:	December 23, 2022
Project Name: ExCALIBUR Fusion Modelling System		

	Name and Department	Signature	Date
Prepared By:	James Cook Will Saunders Wayne Arter BD	N/A N/A N/A	December 23, 2022 December 23, 2022 December 23, 2022
Reviewed By:	Wayne Arter Project Technical Lead	W. Arter	December 23, 2022

1 Introduction

1.1 Electrostatic Particle-In-Cell

Particle-In-Cell (PIC) is a simulation approach where N computational particles represent a species of interest in a simulation domain $\Omega \subset \mathbb{R}^d$. Each particle i holds a position $\vec{r}_i \in \Omega$ and carries a velocity $\vec{v}_i \in \mathbb{R}^{d_v}$. We consider the case where $d \leq d_v$, ie. the velocity of the particle may be a point in a higher dimensional space that the position component. For example we consider the case where there are 2 spatial dimensions and 3 velocity dimensions in a, so called, "$2d3v$" simulation. In this report we assume that the first d dimensions of the d_v velocity space are commensurate with the simulation domain Ω, ie. if Ω has two dimensions labelled "x" and "y" then the first two velocity components are in the x and y directions respectively.

Each computational particle carries a charge q_i. We consider a system where charged particles interact only though an electrostatic interaction and assume that there are no magnetic fields induced by particle motion. There may exist a background magnetic field independent of the particle system that influences the particle motion. Although the computational particles carry a non-zero charge they will not interact directly through a pairwise Coulomb potential. Instead the Electric potential ϕ is approximated on a computational mesh that covers the simulation domain Ω. Such meshes are typically formed of "cells" which motivates the name of the method.

In this report the computational mesh is a collection of triangles and quadrilaterals on which a Finite Element Method (FEM) is constructed. This unstructured approach is highly uncommon among existing PIC implementations but allows the computational domain Ω to approximate a physical system with more complex, and realistic, geometry. In each cell in the mesh, functions such as the electric potential ϕ and charge density ρ are represented by an order p polynomial. Typically, existing PIC implementations only represent quantities at the nodes of the computational mesh resulting in a $p = 0$ representation. Our implementation uses the *Nektar++* library[1, 2] as a FEM implementation. We refer the reader to the *Nektar++* documentation and supporting material for details of the FEM method.

1.2 Particle Force Computation

If we assume, for the moment, that there is zero magnetic field then the force on particle i is given by

$$F(\vec{r}_i, t) = q_i E(\vec{r}_i(t)) \tag{1}$$

where

$$E(\vec{r}) = -\nabla\phi(\vec{r}) \tag{2}$$

is the electric field. In our implementation we compute the field E by first computing the electric potential ϕ then evaluating the derivative to obtain E. The potential ϕ is given as the solution of a Poisson equation

$$\Delta\phi(\vec{r}) = -\rho(\vec{r}) \tag{3}$$

where ρ is the charge density given by

$$\rho(\vec{r}) = \sum_i q_i \delta(\vec{r} - \vec{r}_i). \tag{4}$$

Note that in Equation (4) the LHS is a quantity with a mesh based representation and the RHS is a quantity stored on a collection of particles. In section 1.3 we describe exactly how this change of representations is performed in our implementation. As described earlier, we use *Nektar++* as a FEM implementation to solve Equation (3) in rectangular domains with periodic boundary conditions. With periodic boundary conditions the potential ϕ is defined up to a constant. Although this constant does not affect the dynamics of the particle system it does affect the potential energy of the system and this potential energy is a useful diagnostic quantity.

In this report we shall enforce that

$$\int_\Omega \rho d\vec{x} = 0 \tag{5}$$

though the addition of a constant offset over the entire domain. We shall then choose a constant offset to the computed electric potential such that

$$\int_\Omega \phi d\vec{x} = 0. \tag{6}$$

This approach computes a physically realistic and unique solution to Equation (3).

1.3 Charge Deposition

In PIC literature "charge deposition" refers to the process of converting a particle representation of charge density into a mesh based representation of charge density. In our implementation the mesh representation is a scalar valued function that is a member of a finite element function space. We consider Continuous Galerkin function spaces of polynomial order p. In this FEM formulation a scalar valued function u defined over Ω is represented by a set of coefficients α_j such that

$$u = \sum_{j=1} \alpha_j \psi_j \tag{7}$$

where ψ_j is the j^{th} basis function. These basis functions are chosen, by the FEM implementation, to span the desired function space. The output of the deposition method we describe is independent of the exact basis functions used by the underlying FEM implementation.

To deposit a scalar valued quantity we apply an L2 Galerkin projection to approximate the "true" function $\hat{u}$ by a finite element function u. In this report the original function is the charge density described by particle data as

$$\hat{\rho}(\vec{r}) = \sum_i q_i \delta(\vec{r} - \vec{r}_i) \tag{8}$$

and we seek a function ρ such that

$$\rho(\vec{r}) = \sum_{j=1} \alpha_j \psi_j \tag{9}$$

and

$$\langle \rho - \hat{\rho}, \psi_j \rangle = 0 \,\forall\, j \tag{10}$$

where $\langle f, g \rangle$ denotes the inner product of f and g. Equation (10) is equivalent to enforcing that the projection error is in the null space of the finite element function space and hence is "optimal" in some sense. Rearranging Equation (10) gives a linear system of equations to solve for the coefficients α_j that describe the approximation ρ,

$$M\vec{\alpha} = \vec{\Psi}, \tag{11}$$

where M is the mass matrix such that $M_{ij} = \langle \psi_i, \psi_j \rangle$ and $(\vec{\Psi})_j = \langle \hat{\rho}, \psi_j \rangle$. As our particle shape function is a Dirac delta the evaluation of the RHS of Equation (11) can be simplified as

$$(\vec{\Psi})_j = \langle \hat{\rho}, \psi_j \rangle \tag{12}$$

$$= \sum_i q_i \int_\Omega \delta(\vec{r} - \vec{r}_i)\psi_j d\vec{x} \tag{13}$$

$$= \sum_i q_i \psi_j(\vec{r}_i) \tag{14}$$

which can be evaluated without performing quadrature. Once the RHS of Equation (11) is computed, the system is solved for the coefficients $\vec{\alpha}$ by calling the FEM library methods for solving mass matrix system. These coefficients $\vec{\alpha}$ then represent the function ρ which forms the RHS of the Poisson Equation (3) after a neutralising contribution has been added.

In this report we assume that the FEM library provides methods that allow functions to be evaluated at arbitrary points in the domain, ie. the computation of $\phi(\vec{r}_i)$, $\partial\phi(\vec{r}_i)/\partial x$ and $\partial\phi(\vec{r}_i)/\partial y$ are performed by provided functions. In practice our implementation performs optimisations such as caching reference coordinates to accelerate function evaluations in comparison to naively calling the FEM library methods.

1.4 Time Integration

Various numerical methods exists to advance particle positions and velocities forward in time from an initial state. For example with a non-zero magnetic field the Boris method [3] is popular. With no magnetic field the Velocity Verlet scheme provides second order accuracy in time, is symplectic and is a three stage process as described in Algorithm 1.

```
for timestep $n = 1, \dots, n_{\max}$ do
    For all particles $i$: $\vec{v}_i \mapsto \vec{v}_i + \frac{\delta t}{2m_i}\vec{F}_i$, $\vec{r}_i \mapsto \vec{r}_i + \delta t \vec{v}_i$
    Construct $\rho$ as described in Section 1.3 and solve the Poisson Equation (3)
    For all particles $i$: $\vec{v}_i \mapsto \vec{v}_i + \frac{\delta t}{2m_i}\vec{F}_i$
end
```

Algorithm 1: Velocity Verlet integrator. A time step of size δt is used to integrate forward in time until the final time $T = n_{\max}\delta t$.

There are multiple different forms of the Boris integration method that can be applied in systems with magnetic fields. In Algorithm 2 we provide an overview of the Boris integration scheme applied in this $2d3v$ report.

```
for timestep $n = 1, \ldots, n_{\max}$ do
    Construct $\rho$ as described in Section 1.3 and solve the Poisson Equation (3)
    for For all particles $i$ do
        Set $\vec{t} = \vec{B}(\vec{r}_i)\frac{q_i \delta t}{2m_i}$
        Set $\vec{s} = \frac{2}{1+||\vec{t}||_2^2}\vec{t}$
        Set $\vec{v}_- = \vec{v} + \frac{q_i \delta t}{2m_i}\vec{E}(\vec{r}_i)$
        Set $\vec{v}' = (\vec{v}_- \times \vec{t}) + \vec{v}_-$
        Set $\vec{v}_+ = (\vec{v}' \times \vec{s}) + \vec{v}_-$
        Set $\vec{v}_i = \vec{v}_+ + \frac{q_i \delta t}{2m_i}\vec{E}(\vec{r}_i)$
        Set $\vec{r}_i = \vec{r}_i + \delta t \vec{v}_i$
    end
end
```

Algorithm 2: Boris integrator for systems with non-zero magnetic field $\vec{B}$. A time step of size δt is used to integrate forward in time until the final time $T = n_{\max}\delta t$.

2 Task Work

2.1 Implementation

As described in the introduction this implementation involves both particles and finite elements. In this implementation we realise a system where particles exist on, and are tracked over, the mesh used for the finite element method. As part of the particle tracking, individual particles are assigned to unique mesh cells when the particle positions are updated. Assigning particles to cells allows the underlying implementation to determine which MPI rank owns the particle based on its current position. Furthermore, for the charge deposition operation it is known a-priori that all particles that contribute to the degrees of freedom (DOFs) of a mesh cell are owned, and exist on, the MPI rank that owns the cell. This alignment of particle decomposition and mesh decomposition, along with the δ particle shape, greatly reduced inter-process communication for the charge deposition process.

Our particle implementation is the *NESO-Particles* (NP) [4] library which is specifically designed for representing particle data on spatially decomposed unstructured meshes. NP approaches the performance-portability challenge by implementing performance critical components in the SYCL2020 language. SYCL enables developers to write memory allocations and parallel loops in a language that is agnostic of the tool-chain which will ultimately compile the code for target hardware. For example the same source code could be compiled for an OpenMP runtime though a standard C++ compiler or for Graphics Processing Units (GPUs) through a vendor specific compiler. Note that this transferability indicates portability but does not imply performance on the end hardware. Creating performant SYCL source code is an active research area of the NEPTUNE project.

NP is also designed to be agnostic of the particular unstructured mesh particles exist on. This independence allows NP to be a generic particle library without a requirement that an end user must use a particular mesh implementation, eg. a *Nektar++* mesh. For a given mesh implementation an interface “shim” must be provided that enables NP to manage particles over that particular mesh. This shim provides the map from physical space to mesh space that determines which cell owns a particular point in space. Furthermore the shim describes the parallel decomposition of the mesh such that when a particle departs an MPI rank the destination rank can be determined. This result is a particle library that natively represents particles on a spatially decomposed *Nektar++* mesh in an efficient manner.

As previously alluded to, we offload all aspects of the FEM to the *Nektar++* library. Within *Nektar++* we construct two “fields” to store the solution and RHS of Equation (3). A *Nektar++* field is a function from the requested finite element function space. In this report our function space is constructed from a Continuous Galerkin (CG) polynomial basis of order p and typically in the presented examples $2 \leq p \leq 4$. The periodic boundary conditions are enforced by the chosen function space and we implemented a bespoke *Nektar++* “solver” to solve Equation (3) in the context of our particular forcing term.

The final implemented component we discuss is the projection operation that deposits the particle charges onto the mesh. This projection operation reduces to evaluating each basis function at the location of each particle and multiplying the result by the particle weight. As the support of a basis function is the cell it is defined over, the computation of the Ψ_j values is a cell-wise operation we make that observation that

$$\Psi_j = \sum_i q_i \psi_j(\vec{r}_i) \tag{15}$$

$$= \sum_{i \in C_j} q_i \psi_j(\vec{r}_i) \tag{16}$$

where C_j is the cell over which basis function j is defined. Hence the computation of the RHS of Equation (11) is an operation that is local to each cell and requires no MPI communication for δ shaped particles. For a CG function space, the mass-matrix solve that follows the construction of the RHS is expected to involve communication.

As NP natively stores particle on a per-cell basis it is straightforward to loop over all particles contained within each mesh cell. This cell-wise ordering can be exploited to avoid write contention when constructing the Ψ_j values. In particular this structure promotes a high arithmetic intensity, and computationally efficient, algorithm to compute the Ψ_j values for each cell. Efficient implementation of the Ψ_j values is considered future work and in this report we consider the correctness and viability of the algorithms we describe rather than the implementation efficiency.

2.2 Two-Stream Instability

The Two-Stream (TS) instability is a common test case for implementations of charged particles that interact through an electrostatic potential and, in this case, zero magnetic field. The model constructs two "streams" of identically charged particles, with an isotropic neutralising background field, that are initially travelling in opposite directions. In the limit of an infinite number of computational particles the charge density ρ is zero everywhere in the domain at the initial time $t = 0$. Hence $\phi(t = 0) = 0$ and $\vec{E}(t = 0) = 0$ and the particles experience no perturbing force from their initial velocities. The initial condition is created for N particles by applying Algorithm 3.

Note that there exists a distinction between computational particles and physical particles. The desired number of physical particles is many orders of magnitude more than the number of computational particles which can feasibly be represented on a computer. Hence each computational particle, ie. particles that exist in *NESO-Particles*, represent many physical particles which are characterised by assigning each computational particle a "weight". This weight allows conversion between the density of computational particles and the density of physical particles that the computational particles represent.

```
for For all particles $i$ do
    Sample $\vec{r}_i$ from 2-D Sobol[5] distribution scaled to fit $\Omega$.
    Set $q_i = q$
    Set $m_i = m$
    Set $w_i = w$
    Sample $\xi = \text{uniform}(0, 1)$
    if $\xi < 1/2$ then
        Set $\vec{v}_i = (v_b, 0, 0)^T$
    else
        Set $\vec{v}_i = (-v_b, 0, 0)^T$
    end
end
```

Algorithm 3: Initial condition for two-stream instability constructed with N computational particles. Computational domain $\Omega = [0, L_x] \times [0, L_y]$. Particle charge q, mass m and weight w. Particle velocities are drawn from $\{-v_b, v_b\}$ for some fixed velocity $v_b \in \mathbb{R}$.

This equilibrium is unstable as due to numerical and algorithmic inaccuracies the charge density and potential fields will not be exactly zero everywhere in the domain resulting in a net force on particles that perturbs the particle distribution away from the initial condition. Once this perturbation is initiated the potential energy of the overall system grows at an exponential rate which can be estimated from the linear dispersion relation for a plasma. This theoretical growth rate is compared with the observed growth rate in the simulation to assess implementation correctness.

To compute the theoretical growth rate we first consider the linear electrostatic plasma dispersion relation

$$1 + \Sigma_s \frac{\omega_{ps}^2}{k_\parallel} \int_{-\infty}^{\infty} \frac{\partial f_s(v_\parallel)}{\partial v_\parallel} \frac{1}{\omega - k_\parallel v_\parallel} dv_\parallel = 0 \tag{17}$$

where, in SI units:

$\omega_{ps} = \sqrt{\frac{q^2 n}{m \epsilon_0}}$ The plasma frequency of species s.

q Is the charge of the physical particle (ie. not the computational macro-particle) of species s.

m Is the mass of the physical particle species s.

n Number density of the species.

ϵ_0 Is the electric permittivity.

ω Is a complex frequency.

$v_\parallel$ Is the one-dimensional velocity coordinate ($v_\parallel = \vec{v}_x$ in our simulations).

$f_s(v_\parallel)$ The distribution function of species s and is normalised such that $\int_{-\infty}^{\infty} f_s(v_\parallel) dv_\parallel = 1$.

$k_\parallel$ The wavenumber parallel to the velocity $v_\parallel$ (and also parallel the magnetic field, which otherwise does not feature here; the problem should be considered "$1d1v$" and $\vec{B} = 0$).

Two oppositely travelling streams exhibit a velocity distribution function of $f(v_\parallel) = \delta(v_\parallel \pm v_b)$. We proceed by evaluating the integral in Equation (17) for this particular velocity distribution. Starting by integration by parts we obtain

$$1 + \Sigma_s \frac{\omega_{ps}^2}{k_\parallel} \int_{-\infty}^{\infty} \frac{\partial f_s(v_\parallel)}{\partial v_\parallel} \frac{1}{\omega - k_\parallel v_\parallel} dv_\parallel = 1 - \Sigma_s \frac{\omega_{ps}^2}{k_\parallel} \int_{-\infty}^{\infty} f_s(v_\parallel) \frac{\partial}{\partial v_\parallel} \left(\frac{1}{\omega - k_\parallel v_\parallel} \right) dv_\parallel \tag{18}$$

where observing that

$$\frac{\partial}{\partial v_\parallel} \left(\frac{1}{\omega - k_\parallel v_\parallel} \right) = \frac{k_\parallel}{(\omega - k_\parallel v_\parallel)^2} \tag{19}$$

gives

$$1 - \Sigma_s \omega_{ps}^2 \int_{-\infty}^{\infty} f_s(v_\parallel) \frac{1}{(\omega - k_\parallel v_\parallel)^2} dv_\parallel = 0. \tag{20}$$

We now substitute the general distribution function f_s for the two-stream distribution function $f(v_\parallel) = \delta(v_\parallel \pm v_b)$ to obtain

$$1 - \frac{\omega_{ps}^2}{(\omega - k_\parallel v_b)^2} - \frac{\omega_{ps}^2}{(\omega + k_\parallel v_b)^2} = 0. \tag{21}$$

We seek the fastest growing waves that satisfy Equation (21). First we create non-dimensional variables $x = \omega/\omega_{ps}$ and $u = v_b k_\parallel/\omega_{ps}$ and rewrite Equation (21) as

$$1 - \frac{1}{(x-u)^2} - \frac{1}{(x+u)^2} = 0 \tag{22}$$

which is a quartic equation with solution

$$x = \pm\sqrt{u^2 + 1 \pm \sqrt{4u^2+1}}. \tag{23}$$

For the fastest growing mode $\frac{\partial x}{\partial u} = 0$ which corresponds to $u = \frac{\sqrt{3}}{2}$ and four solutions for x, two real $x = \pm\frac{\sqrt{15}}{2}$ and two imaginary $x = \pm\frac{i}{2}$.

All these solutions share $u = \frac{\sqrt{3}}{2}$ for which $v_b k_{\parallel,max} = \frac{\sqrt{3}}{2}\omega_{ps}$. If the wavenumber of the fastest growing mode is resolved by the PIC code, the nonlinear self-consistent solution will be dominated by the dynamics of the fastest growing mode and exhibit its growth rate, $\gamma_{max} = \frac{\omega_{ps}}{2}$ (where γ is typically used in this context to represent the imaginary component of the complex frequency). Note that the unstable solutions have zero real frequency.

Employing periodic boundary conditions and tuning the parameters such that exactly M modes of the fastest growing mode fit in the box in the direction of $v_\parallel$ of length L, ($k_{\parallel,max} = M\frac{2\pi}{L}$), we arrive at:

$$\gamma_{max} = \frac{\omega_{ps}}{2} = v_b M \frac{2\pi}{L\sqrt{3}}, \omega_{ps} = v_b M \frac{4\pi}{\sqrt{3}L}, n = v_b^2 M^2 \frac{m\epsilon_0}{q^2} \frac{16\pi^2}{3L^2} \tag{24}$$

Hence if we initialise all particles as the same species (not two distinct species that differ by sign of drift speed) and randomly assign one of half of them with $+v_b$ and the other half with $-v_b$, we get the case where the particle weight is controlled by the number of particles and the total number density $n_T = 2n$. Hence particle weight $w = \frac{n_T L_x L_y}{N_{comp,part}}$ where $L_x = 1$, $L_y = 1$ and $N_{comp,part}$ are the lengths of the domain in x and y (both set to unity) and the total number of computational macro-particles used in the simulation, respectively.

In general the particle weight is

$$w = \frac{2nL_xL_y}{N_{comp,part}} = v_b^2 M^2 \frac{m\epsilon_0}{q^2}\frac{16\pi^2}{3L^2}\frac{2L_xL_y}{N_{comp,part}} \tag{25}$$

to squeeze an integral number of maximally unstable modes into the simulation domain. So for $v_b = M = m = q = \epsilon_0 = L_x = L_y = 1$

- $\gamma_{max} = \frac{\omega_{ps}}{2} = \frac{\omega_{pT}}{2\sqrt{2}} = \frac{2\pi}{\sqrt{3}}$
- $n_T = \frac{32\pi^2}{3}$
- $w = \frac{32\pi^2}{3N_{comp,part}}$
- $\omega_{pT} = \sqrt{\sum_{s=l,r} \omega_{ps}^2} = \omega_{ps}\sqrt{2}$

s the linear theory assumes that the simulation domain is 1-D, we created a 2-D rectangular mesh with a 100:1 aspect ratio between the x extent $L_x = 1$ and the y extent $L_y = 0.01$. The mesh is a structured mesh of quadrilaterals with 1 element in the y direction and 20 elements in the x direction. A CG function space is constructed with degree 4 polynomials and periodic boundary conditions. A total of $500\,000$ computational particles are initialised via Algorithm 3. The simulation is integrated forward in time via the Velocity Verlet integrator in Algorithm 1 with a non-dimensionalised time step $\delta t = 0.001$ for 3000 time steps. The implementation of this two-stream setup we describe is available as an example solver in the *NESO* framework [6].

In Figure 1 we plot the potential and kinetic energy of the particle system as a function of time. We observe that at the initial time $t = 0$ the vast majority of the energy is kinetic with little potential energy. Furthermore we observe that as time progresses energy is transferred between kinetic and potential forms as expected.

We note that the error in total energy remains bounded throughout the simulation at approximately 5 significant figures which is very acceptable for a simulation of this type. At $t = 0$ we observe that the initial condition exhibits approximately zero potential energy which indicates that the initialisation of particle positions via the Sobol method is sufficiently uniform.

In Figure 2 we plot the potential energy measured particle wise as in Figure 1 and additionally field wise by evaluating $\int_\Omega \phi^2 d\vec{x}$. These two measurements are not expected to be exactly equal, however both should grow in magnitude according to the theory we introduced earlier in this section. In Figure 2 we plot this theoretical growth rate along with the growth rate measured from the simulation. We observe good correspondence between the fitted and theoretical growth rates.

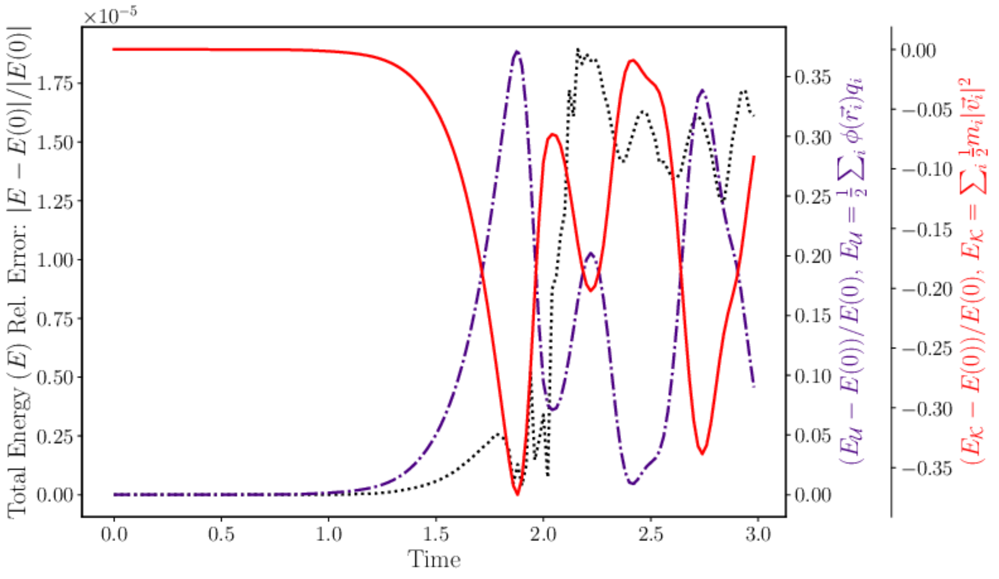


Figure 1: Variation in Potential energy $E_{\mathcal{U}}$ (plotted in dash-dot indigo), variation in kinetic energy $E_{\mathcal{K}}$ (plotted in solid red) and total energy E (plotted in dotted black) against simulation time. Potential and kinetic energy are plotted relative to the initial total energy $E(0)$.

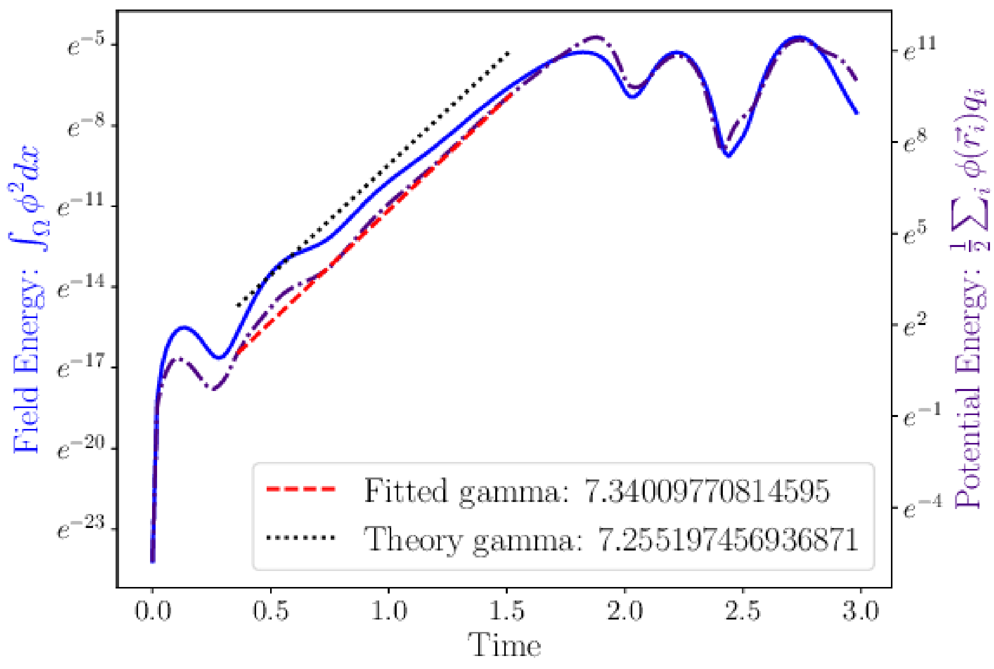


Figure 2: Potential energy measured particle-wise (plotted in dash-dot indigo) and measured field-wise (plotted in solid blue) against simulation time. Measured growth rate (plotted in dashed red) is compared with theoretical growth rate (plotted in dotted black).

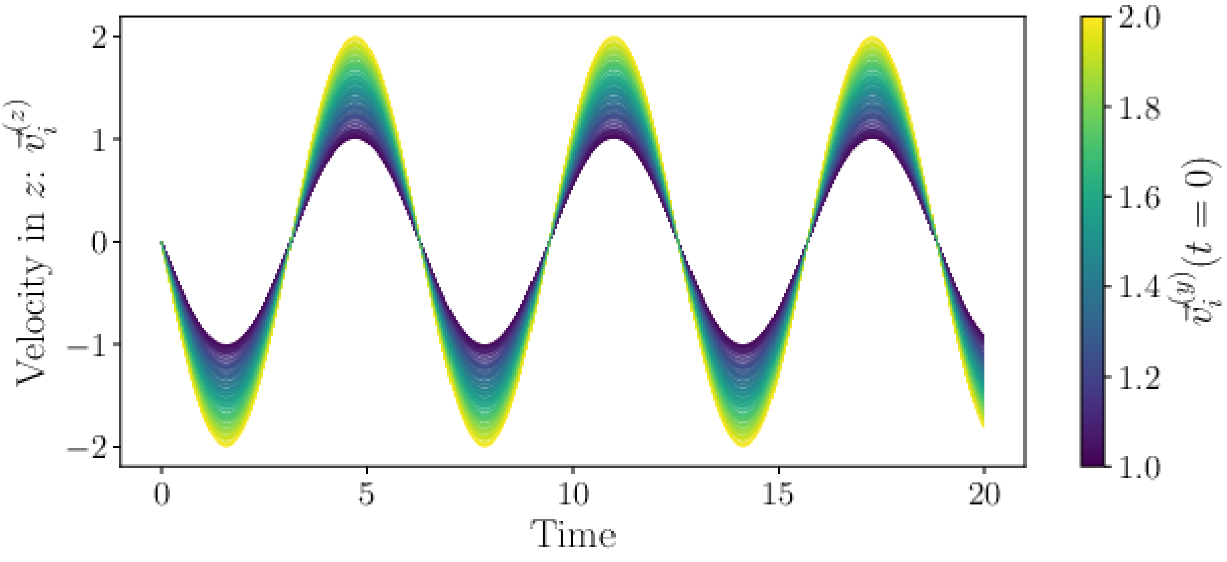


Figure 3: z-component of velocity of 200 particles plotted against simulation time. Line colour indicates the y-component of velocity the particle was initialised with.

2.3 Helical Trajectory along x-axis

To test motion in the third velocity dimension we investigated charged particle motion under the influence of only a magnetic field, ie. $\vec{E} = 0$ for this experiment. We shall set $\vec{B} = (1, 0, 0)^T$ to create circular particle orbits in the $y - x$ plane. The particle distribution is initialised by following Algorithm 4 and is integrated forward in time via the Boris integrator in Algorithm 2.

```
for For all particles $i$ do
    Sample $\vec{r}_i$ from 2-D Sobol[5] distribution scaled to fit $\Omega$.
    Set $\vec{v}_i^{(x)} = 1$
    Set $\vec{v}_i^{(y)} =$ uniform$(1, 2)$
    Set $\vec{v}_i^{(z)} = 0$
    Set $m_i = 1$
    Set $q_i = 1$
end
```

Algorithm 4: Initial condition for helical particle trajectories.

In Figure 3 we plot the z-component of velocity for 200 particles and observe the expected sinusoidal pattern of motion. In Figure 4 we plot the z-component of velocity against the y-component of velocity and observe the expected circular orbit in the zy plane.

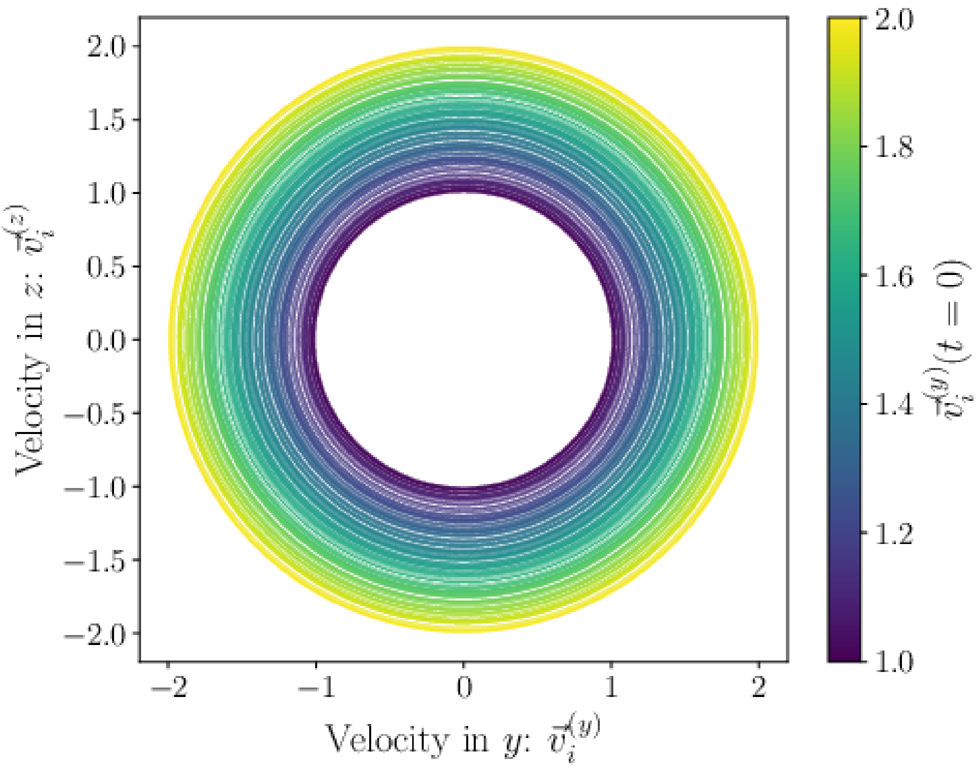


Figure 4: z-component of velocity of 200 particles plotted against y-component of velocity. Line colour indicates the y-component of velocity the particle was initialised with.

2.4 Warm Langmuir waves

The warm Langmuir wave is an electrostatic oscillation at the plasma frequency in the long wavelength limit. At short wavelengths it is damped, but rises quadratically in wavenumber for intermediate wavenumbers;

$$\omega \approx \omega_{pe}\left(1 + \frac{3}{2}\lambda_D^2 k^2\right) \tag{26}$$

where ω_{pe} is the plasma frequency, v_{th} is the thermal velocity of the particles and $\lambda_D = v_{th}/\omega_{pe}$ is the Debye length. The use of the word *warm* here comes from the origin of the quadratic term; the plasma model with knowledge of the bulk thermal velocity. The PIC code is fully kinetic where the particle velocities are drawn from a distribution characterised by the thermal velocity. Hence the *kinetic* PIC code captures the warm physics.

Figure 5 is created from a $2d3v$ implementation of the Poisson-Lorentz set of equations using Nektar++, NESO, and NESO-Particles combined to create an $2d3v$ electrostatic PIC code. This figure shows that the continuous Galerkin technology of Nektar++ works with the SYCL particle technology and indicates a huge leap forwards in capability.

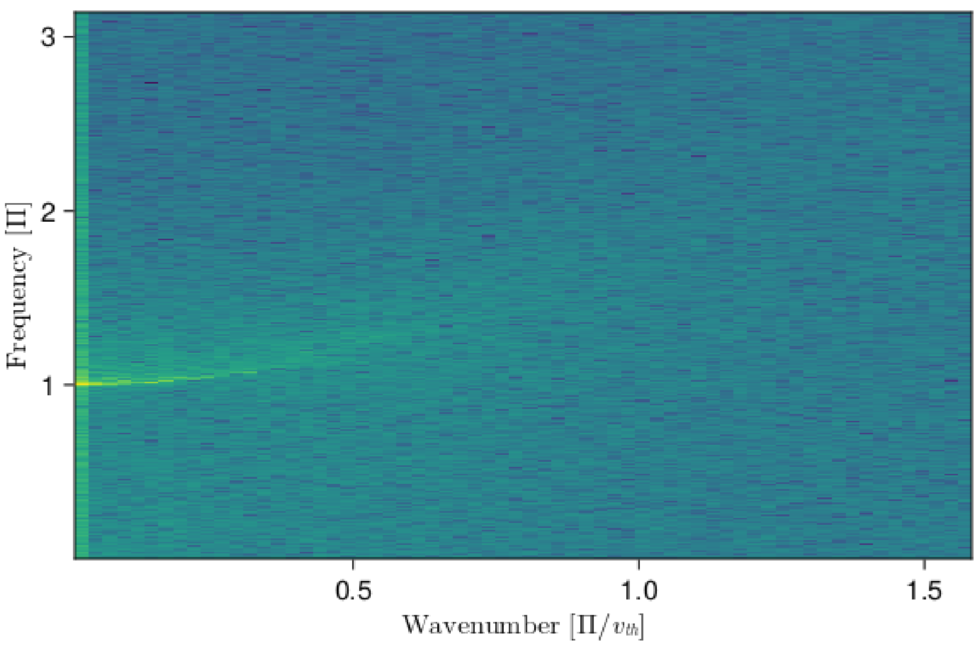


Figure 5: The power spectral density as function of frequency and wavenumber showing the warm plasma dispersion relation. The logarithm of the absolute value of the spatiotemporal Fourier transform of the electric field component parallel to the magnetic field in time and the direction parallel to the magnetic field. Yellow indicates regions of undamped normal modes and highlights the dispersion relation of the warm Langmuir wave.

2.5 Electrostatic cyclotron waves

Electrostatic cyclotron waves oscillate at harmonics of the plasma frequency. Those with harmonics greater than one are kinetic phenomena and can be visible in the Fourier transform of electric field components perpendicular to the magnetic field. Figure 6 is such an example from a $2d3v$ electrostatic PIC code written in julia.

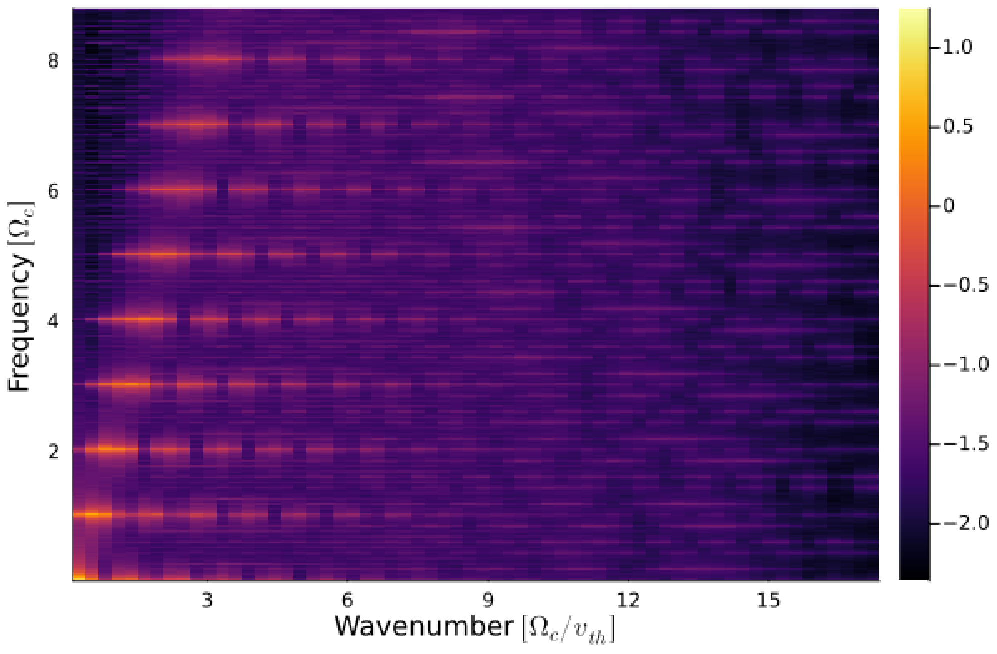


Figure 6: Horizontal stripes show electrostatic cyclotron waves from a $2d3v$ electrostatic PIC code written in julia for a single plasma species and a magnetic field in the plane. Colour indicates the logarithm of the absolute value of the spatiotemporal Fourier transform of the electric field component perpendicular to the magnetic field in time and the direction parallel to the magnetic field. Yellow indicates regions of undamped normal modes.

2.6 Bernstein Waves

Bernstein waves are an electrostatic kinetic plasma phenomenon that can be reproduced numerically with a time stationary magnetic field and a single thermal electron population, characterised by thermal velocity v_{th}, in the presence of numerical neutralisation, with periodic boundary conditions. These normal modes are supported by the interplay between the gyration of particles around the magnetic field and their thermal motion, and they propagate perpendicular to the magnetic field.

The dispersion relation of Bernstein waves differs from cyclotron harmonics, $\omega = l\Omega_e$, where Ω_e is the electron cyclotron frequency and l is the integral harmonic number, as the ratio of the plasma frequency to the cyclotron frequency, ω_{pe}/Ω_e increases. The distortion away from cyclotron harmonics is particular visible at low frequencies and low wavenumbers in the power spectral density of the supporting fields.

Random thermal noise serves to excite waves in frequency ω and wavenumber k space, but perpendicular fluctuations are everywhere damped except where $(\omega, k_\perp)$ represents that of the Bernstein mode (or hybrid wave). The noise inherent in PIC codes suffices to highlight Bernstein modes when plotting the power spectral density of the electrostatic potential; a heatmap of the logarithm of the absolute value of its spatio-temporal Fourier transform.

The dispersion arises from the generating function Bessel identity at the foundation of linear kinetic plasma theory,

$$\exp(\lambda\cos\theta) = \sum_{n=-\infty}^{\infty} I_n(\lambda)\exp(in\theta), \tag{27}$$

which describes the bunching the of particles around the gyro-orbit as a response to perturbations and where $\lambda = \frac{v_{th}^2 k_\perp^2}{2\Omega_e^2}$.

In the large $k_\perp$ limit, the linear solution (Eq. 30.1 of Ref. [7]),

$$0 = 1 - \frac{\omega_{pe}^2}{\Omega_e^2}\frac{\exp(-\lambda)}{\lambda}\sum_{n=-\infty}^{\infty}\frac{n^2 I_n(\lambda)}{\omega^2 - n^2\Omega_e^2}. \tag{28}$$

It is the sum, here, that causes zeros in the dispersion relation of Equation (28), whereby pairs of harmonics determine the location of the root of the dispersion relation, which also prohibits a Bernstein wave with $\omega < |\Omega_e|$.

Figure 7 shows electrostatic cyclotron waves and Bernstein modes in the dispersion relation created by a $1d2v$ PIC code written in julia where the spatial domain is aligned in x, the velocity components in x and y and the magnetic field aligned with z.

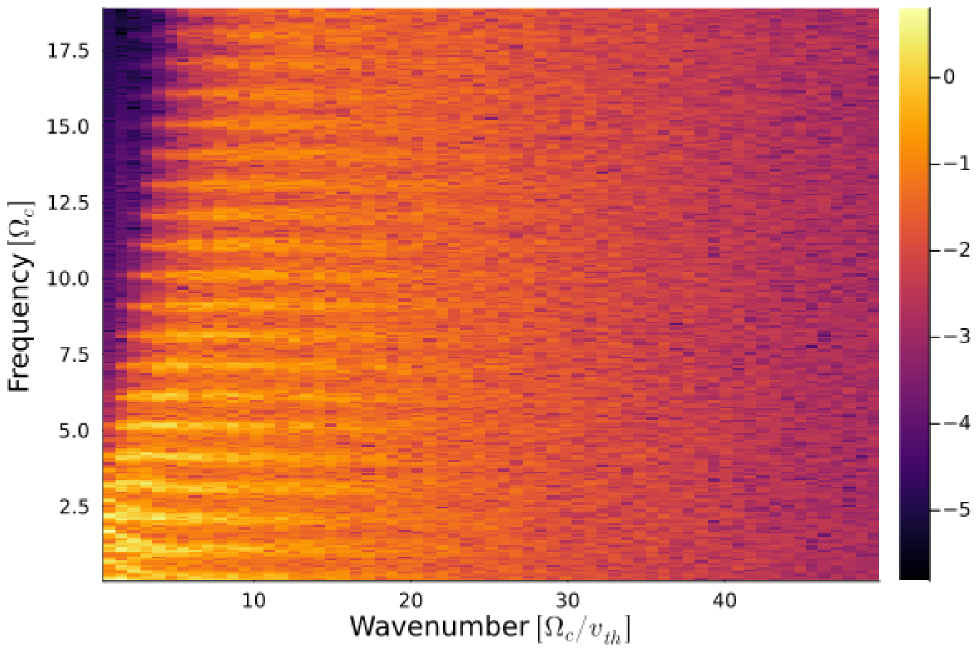


Figure 7: Horizontal stripes show electrostatic cyclotron waves mixed in with dispersive Bernstein waves from a $1d3v$ electrostatic PIC code written in julia for a single plasma species and a magnetic field pointing orthogonal to the simulation domain. Colour indicates the logarithm of the absolute value of the spatiotemporal Fourier transform of the electric field component perpendicular to the magnetic field in time and the direction parallel to the magnetic field. Yellow indicates regions of undamped normal modes.

2.7 Initial Strong Scaling Results

Although the implementation is at a very early stage we can start to assess the parallel performance characteristics. We set up a strong scaling experiment based on the two-stream problem with a modification such that domain Ω is a unit square and all other parameters are unmodified except for the number of particles. We set the number of computational particles to $3\,200\,000$ for a smaller experiment and $25\,600\,000$ for a larger experiment. A strong scaling experiment records the time to solution for a fixed problem as the amount of computational resource is increased.

We perform this experiment on the CSD3 cluster with Intel Skylake nodes consisting of 32 cores per node (Xeon Gold 6142). The complier for our implementation and all dependencies is Intel OneAPI version 2022.1.0 and we used Intel MPI version 2021.7. The SYCL platform is the OneAPI *host* target that places one computational thread per MPI rank, ie. the experiment is performed with MPI only. In Figure 8 we plot the results of the strong scaling experiment for node counts between 1 and 8.

We do not investigate higher node counts as these results are from a very preliminary version of *NESO* where we have already identified opportunities for significant performance improvements. For example, many of the *Nektar++* calls we make are to methods originally implemented as setup

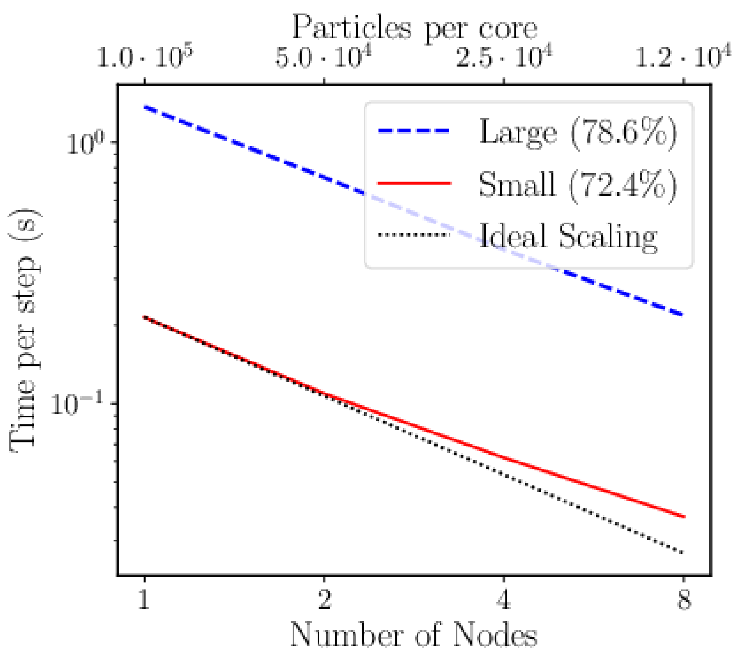


Figure 8: Time per simulation step plotted against node counts for the strong scaling experiment. Larger experiment plotted in dashed blue, Smaller experiment in solid red and ideal scaling in dotted black. Top axis indicates computational particles per core for the smaller strong scaling experiment. Number of particles per core for the larger experiment is eight times the smaller experiment. Percentages indicate the parallel efficiency at the largest node count relative to one node.

routines in the original FEM use case and hence are not optimised as much as is desirable for a function called within a main loop. Secondly we have not applied improvements which we have already identified and plan to implement in our particle transport implementation.

We observe that, for an initial implementation, Figure 8 demonstrates acceptable parallel scaling. Note that in the strong scaling limit of the smaller experiment there are only $\mathcal{O}(10\,000)$ particles per CPU core which is relatively small for a PIC code. Now that we possess a parallel PIC code we intend to continuously profile and optimise the implementation, a process for which these results should be considered an initial point of reference.

3 Summary

In this report we described the overarching models and algorithms we applied to assess the correctness and validity of our methodology and implementations. This work involved the development of a SYCL based particle framework, *NESO-Particles*, to represent particle data on unstructured meshes. This particle framework enables particle motion over these unstructured meshes in conjunction with a domain decomposition based parallelisation strategy utilising MPI. The details of exactly how particles are transferred between MPI ranks is described is the M4.3 report [8].

The M4.3 report also provides initial results from the L2 projection algorithm which is employed to convert from particle to FEM based representations. In M4.3 initial viability of this approach is explored, whilst in this report we demonstrate the ability of this approach to reproduce known physical results. Production of these results required implementation of the projection approach within the *Nektar++* ecosystem. Furthermore we demonstrated initial implementations that utilise Nektar++ as a FEM implementation coupled to our *NESO-Particles* library.

Acknowledgement

The support of the UK Meteorological Office and Strategic Priorities Fund is acknowledged.

This work was performed using resources provided by the Cambridge Service for Data Driven Discovery (CSD3) operated by the University of Cambridge Research Computing Service (www.csd3.cam.ac.uk), provided by Dell EMC and Intel using Tier-2 funding from the Engineering and Physical Sciences Research Council (capital grant EP/T022159/1), and DiRAC funding from the Science and Technology Facilities Council (www.dirac.ac.uk).

References

[1] D. Moxey et al. Nektar++ website. `https://www.nektar.info`, 2020. Accessed: June 2020.

[2] G. Karniadakis and S. Sherwin. *Spectral/hp element methods for computational fluid dynamics 2nd Ed.* Oxford University Press, 2005. `https://doi.org/10.1093/acprof:oso/9780198528692.001.0001`.

[3] J.P. Boris. Relativistic plasma simulation-optimization of a hybrid code. *Proceeding of Fourth Conference on Numerical Simulations of Plasmas*, November 1970.

[4] UKAEA. NESO-Particles. `https://github.com/ExCALIBUR-NEPTUNE/NESO-Particles`, 2022.

[5] S. Joe and F.Y. Kuo. Constructing Sobol sequences with better two-dimensional projections. *SIAM Journal on Scientific Computing*, 30(5):2635–2654, 2008.

[6] UKAEA. NESO. `https://github.com/ExCALIBUR-NEPTUNE/NESO`, 2022.

[7] Marco Brambilla. *Kinetic Theory of Plasma Waves: Homogeneous Plasmas*. Clarendon Press, Oxford, 1998.

[8] W. Saunders, J. Cook, and W. Arter. High-dimensional Models Complementary Actions 2. Technical Report CD/EXCALIBUR-FMS/0062-M4.3, UKAEA, 2022. `https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/ukaea_reports/CD-EXCALIBUR-FMS0062-M4.3.pdf`.