UK Atomic Energy Authority

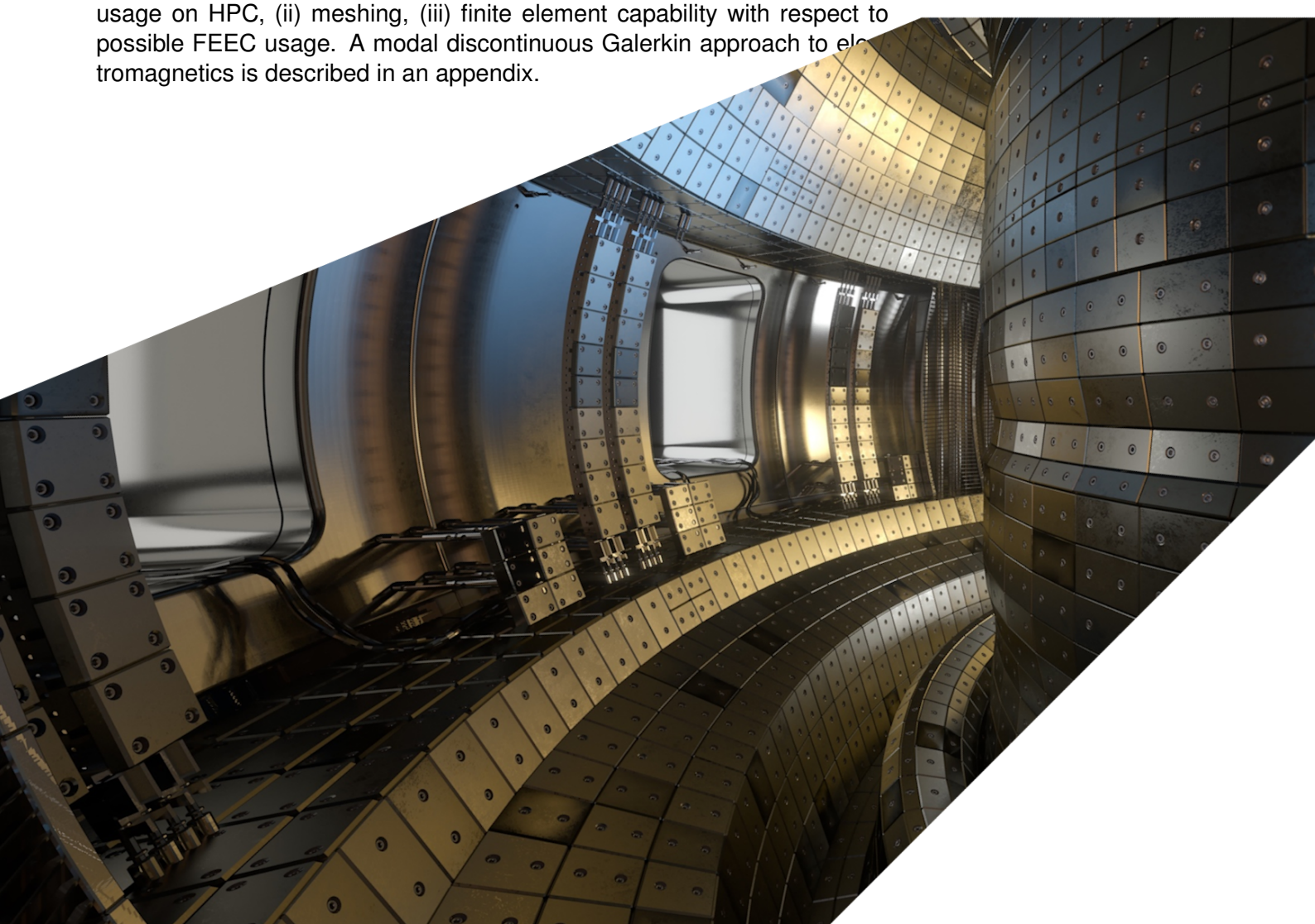
ExCALIBUR

Management of external research. Spectral Elements and Referent Model Procurement

M6c.1

Abstract

The report describes work for ExCALIBUR project NEPTUNE at Milestone M6c.1. It collates technical material used to inform the preparation of Calls Contract T/AW085/22 and Contract T/AW086/22. The material is better presented with that relating the latter contract appearing first, so the report can examine the challenges for computational physics before the specifics of spectral finite elements. Regarding T/AW086/22 (“Advanced referent model procurement”), there is a refresher of the time and spatial scales involved in (i) classical fluid dynamics, (ii) neutral species from the kinetic standpoint, and (iii) tokamak plasma. Drawbacks of three different variational approaches to fluid dynamics are also described. Regarding T/AW085/22 (“Spectral element procurement”), in order to clarify the scope of future work, attention is given to accurate description of what is currently available within the code, specifically (i) the current status with respect to usage on HPC, (ii) meshing, (iii) finite element capability with respect to possible FEEC usage. A modal discontinuous Galerkin approach to electromagnetics is described in an appendix.



UKAEA REFERENCE AND APPROVAL SHEET

	Client Reference:	
	UKAEA Reference:	CD/EXCALIBUR-FMS/0067
	Issue:	1.00
	Date:	October 9, 2022
Project Name: ExCALIBUR Fusion Modelling System		

	Name and Department	Signature	Date
Prepared By:	Wayne Arter Ed Threlfall Joseph Parker Will Saunders BD	N/A N/A N/A N/A	October 9, 2022 October 9, 2022 October 9, 2022 October 9, 2022
Reviewed By:	Wayne Arter Project Technical Lead	W. Arter	October 9, 2022

1 Introduction

This report provides a literature review designed to inform the production of Calls T/AW085/22 “Spectral element procurement” and T/AW086/22 “Advanced referent model procurement”. Ideally the physical model of the plasma should be derived with a view to efficient numerical implementation in general and at Exascale in particular. Such treatment is a key part of the role of the computational physicist, see Section 2. In practice, there is a need to demonstrate as rapidly as possible that the drift kinetic model proposed by Parra et al [1] is feasible. This has meant use of a numerical approach with which the Oxford group is comfortable, namely a Chebyshev spectral finite element implementation in Julia rather than the spectral/hp elements used in Nektar++, where developments are proceeding almost independently.

Generally the Parra et al model with Oxford’s numerics have performed successfully, to the extent that the Advanced referent model procurement needs comparatively little direct input. However, a hopefully minor numerical difficulty has been encountered, see Section 2.2 The report continues by highlighting typical challenges presented by the key physics to be modelled, starting with an appreciation of neutral particles in Section 3.1, then moving onto issues raised by the need to model tokamak plasma in Section 3.2 from plasma core to wall, notably the sizes of the electric field. Section 3.3 examines whether the model equations might be better expressed as a variational principle, but concludes that this unlikely to be helpful. These latter two sections play a largely confirmatory role for Parra et al’s approach.

Section 4 discusses material relevant to the spectral element procurement. The main consideration is the production of a $2d3v$ proxyapp, involving libraries of spectral elements including meshing capabilities. The work is summarized in Section 5.

2 Challenges for Computational Physics

2.1 General

Knowledge of the smallest length and timescales that could be important in a model is in general critical for a reliable determination of its properties. Typically the model is a nonlinear partial differential equation and the properties that are sought, are its solutions. It is known, see eg. ref [2, 3] that failure to represent the smallest scales adequately can lead to catastrophic code failure, and that even when for example numerical dissipation is added to suppress these failures, properties such as wave propagation speed will be subject to large errors unless the wavelength is adequately represented by the discretisation. Almost regardless of whether machine is Exascale or not, it is helpful to be able to work with the most economical discrete representation of a field, which might simply be a function of position or depend on velocity-space coordinates also. It will be assumed that already for example a solenoidal 2-D vector field has been reduced to a scalar representation in terms of the flux as say a function of Cartesian coordinates $f_{2D}(x, y)$.

A discrete representation implies point samples, in the simplest case merely values of a scalar field f at different points in space. To model interaction with other fields or calculate derivatives of f, it is necessary to interpolate to evaluate f at other locations. Interpolation requires knowledge about the likely behaviour of f at other points. The information available is that f relates to a solution of an advection-diffusion problem in n space dimensions and time, where n is up to 6 as Vlasov and Fokker-Planck type equations are allowed. Sources are an additional complication.

Considering the limit where diffusion vanishes, it is evident that it is not always adequate to assume that f is continuous, since information about f propagates along streamlines, eg. $f(x, y, t) = f(x-vt, y, 0)$ in the simple case of a 1-D flow v in the x-direction. In the absence of viscosity there is perfect memory of initial conditions, so the discontinuities propagate unmodified. Fortunately even a small number of collisions eventually smoothes away discontinuities, however it should already be apparent that different interpolation strategies will be optimal in different locations depending on initial conditions.

Classical fluid dynamical problems are of two sorts, loosely speaking depending on whether they are produced by mechanical action such as pumping and aerofoil motion, or by body forces such as buoyancy. The former, properly characterised by the generation of fluid vorticity at material boundaries are of limited application to NEPTUNE. In both sorts, but particularly the latter, there is a flow start-up phase in which a linearised treatment is insightful. In this phase, the optimal representation will often consist of just one or two coefficients of the most unstable eigenmodes which may have a simple trigonometric function dependence on one or more spatial coordinates, eg. refs [4, 5]. As solutions tend to (quasi-)steady-state, particularly when the viscosity is relatively small, gradients tend once more to steepen due now to non-linear effects. Converging advective flows frequently form shocks, although the physics suggests that such discontinuities will be isolated from one another.

The challenge in fluid dynamical modelling is not therefore merely to produce optimal approximations, but to maintain optimality as the flow structure evolves in some typically nonlinear way. This is one of the key challenges addressed by computational fluid dynamics CFD. Use of high order approximations adds to the complexity, see the books by Boyd [6] and more comprehensively

Karniadakis and Sherwin [7] who describe the spectral/hp element method whereby finite element size h and local order of polynomial representation p are allowed to vary in pursuit of optimality.

2.2 Numerical Problems in Work of T/NA085/20

Problems of grid-scale oscillation have been encountered when outflow boundary conditions rather than periodic conditions are imposed on solutions of the kinetic equations. The appearance of the problem may be related to use of a coordinate system in velocity space which is in translation at the drift speed relative to laboratory coordinates. Thus at the wall the boundary, ie. where the density function has to vanish, is (in the present formulation) not fixed. This leads to a situation where there is a physically very rapidly variation of density potentially causing difficulties as above. However, there are other possibilities, although the above seems best to explain why the issue arises in certain cases and not others.

3 Physics Challenges

3.1 Neutrals

For the case of neutral particle modelling, estimates of scales based on the mean free path (mfp) λ and collision time τ (or alternatively average speed $\langle v \rangle$) are frequently adequate. Indeed, taking mesh-size $h = \mathcal{O}(\lambda)$ may well be excessively conservative.

To see this, note that the viscosity D (and indeed any transport coefficient expressed as a diffusivity) is of order $\lambda \langle v \rangle$ [8, § 9] so that the mesh Reynolds (Peclet) number is

$$Re_h = \frac{Uh}{D} = \frac{U}{\langle v \rangle} \frac{h}{\lambda} \tag{1}$$

Since temperature T by definition satisfies $\frac{1}{2} m \langle v^2 \rangle = \frac{3}{2} kT$, $\langle v \rangle$ may be estimated as $\mathcal{O}(\sqrt{kT})$, ie. the sound speed, thus

$$Re_h = M_A \frac{h}{\lambda} \tag{2}$$

where M_A is the Mach number and so provided the fluid model is adequate, it is only necessary that

$$h < \lambda / M_A \tag{3}$$

Further, for neutrals, λ may be estimated in terms of the diameter of the sphere of influence d as

$$\lambda = \frac{1}{\sqrt{2\pi N d^2}} \tag{4}$$

where N is the number density and a Maxwellian distribution of particle speeds is assumed. The diameter d is defined as the average of the diameters of the particles involved in the collision.

3.2 Plasma in a tokamak

Unfortunately estimating smallest length and timescales when charged particles are present is more involved. For the case of a fully ionised plasma of a single particle species, textbooks give three timescales for equilibration, for ions to thermalise, electrons to thermalise, and the longest timescale is for ions and electrons to reach the same temperature. The smallest lengthscale that should be considered is the Debye length. These points are described very nicely in the early chapters of the textbook by Helander and Sigmar [9].

From the numerical point-of-view, models to treat species which are far from thermal equilibrium must account for the non-Maxwellian distribution of particle velocity, either by use of (super)-particle sampling or by introducing extra dimensions to represent the variation of particle density in velocity as well as position space. Plasma species closer to Maxwellian may be treated more like classical fluids but with additional terms to represent deviations from the normal distribution which have been averaged over velocity space. CFD techniques for classical fluids may be considered in these circumstances.

The greater duration of the time taken for the ions and electron to achieve a common temperature means that it is frequently appropriate to treat each species as a separate fluid with a different mean velocity and temperature. Many of the effects due to inter-species interactions produce terms resembling viscosity and thermal conduction, so that classical CFD techniques become even more appropriate, as only the energy equipartition term is without some kind of classical counterpart, and to a first approximation, these plasmas obey a pointwise Ohm's Law, ie. current density proportional to electric field expressed as $\mathbf{J} = \sigma_E \mathbf{E}$, where from [10], assuming T_e is measured in eV, then

$$\sigma_E = 0.0239 \cdot \frac{(T_e)^{3/2}}{Z\Lambda} \ \ m^2 s^{-1} \tag{5}$$

However, the two-fluid equations are stiff in several senses [11]. Not only is it the case that the momentum equations for $\mathbf{u}_e$ and $\mathbf{u}_i$ have timescales in the ratio $m_i/m_e \approx 1836$ even for Hydrogen ions, but also the electrostatic forces are relatively enormous. The latter may be seen from Coulomb's Law if the electrostatic potential is introduced so that $\mathbf{E} = -\nabla\phi$, then the ratio

$$\frac{\rho_c}{\epsilon_o \nabla \cdot \mathbf{E}} = \frac{L_n^2 \sum_s q_s n_s}{\epsilon_o \Delta\phi_o} \tag{6}$$

where $\Delta\phi_o$ is a typical fluctuation level in the potential, assumed to have a length-scale L_n comparable to the width of the SOL. Writing $\sum_s q_s n_s = e n_o \Delta n$, Equation (6) can be reexpressed as

$$\frac{\rho_c}{\epsilon_o \nabla \cdot \mathbf{E}} = \frac{n_o e^2}{\epsilon_o m_e} \frac{L_n^2 \Delta n}{(e\Delta\phi_o/m_e)} = \omega_{pe}^2 \frac{L_n^2 \Delta n}{(\frac{1}{2} v_o^2)}. \tag{7}$$

The quantity ω_{pe} is the plasma frequency, which for a reference density $n_o = 10^{20} m^{-3}$ is $5.6 \times 10^{11} s^{-1}$. If the speed v_o is identified with the electron sound speed v_{the} then $\omega_{pe}/v_{the} = 1/\lambda_D$, where λ_D is known as the Debye length, and since clearly $\lambda_D << 1$, it follows since all three ratios in Equation (7) must be unity, that tokamak plasmas are forced by the electric field to be quasi-neutral, meaning $\Delta n << 1$.

The consistent treatment of the electric field is a major issue for models which seek to model significant regions of the core plasma. This is most graphically seen when the Braginskii electrical

conductivity is evaluated for parameters representative of a reactor plasma core, viz. $T_e = 25\,\text{keV}$, $n = 10^20\,\text{m}^3$, eg. by using the SMARDDA-MISC software [12], whence an estimate $\sigma_E = 5 \times 10^9\,\Omega^{-1}\text{m}^{-1}$. Assuming currents of order several MA in a cross-sectional area of order square metres, ie. current densities of $10^6\,\text{MAm}^{-2}$, implies an electric field of order $10^{-3}\,\text{Vm}^{-1}$. Thanks to the strong temperature dependence in Equation (5), the electrical conductivity is much lower when the plasma temperature falls towards the values of $10\,\text{eV}$ found in the edge regions, so the central electric field is tiny compared to the experimentally measured electric fields in the SOL of order $10^{+3}\,\text{Vm}^{-1}$.

3.3 Variational principles

Finite element equations are easily expressed if there is a variational principle governing the system dynamics. For non-dissipative discrete physical systems, this is usually the case, so that there are variational principles available for the corresponding continuum models such as the Vlasov equation and inviscid fluid dynamics. In the continuum case, where there is bulk flow, there are a number of possibilities, Larsson [13, § 1] recognises three, namely

1. A straightforward generalisation of the Lagrangian from the discrete system
2. A 'constrained' Lagrangian approach, whereby mass and energy conservation are imposed as constraints on the variation.
3. A Lagrangian approach whereby only mass and energy variations consistent with the flow perturbations are allowed.

Possibility 1 leads to a variation conducted in coordinates moving with the flow (Lagrangian coordinates), rather than the more convenient coordinates fixed in the laboratory frame (Eulerian coordinates). Introducing Eulerian coordinates via the constraints, Possibility 2 gives rise to the 'Lin' variables which many find somewhat mysterious, but in any event their introduction doubles the number of variables in the system to be solved for, ruling out the approach on grounds of cost. Possibility 3 is typically approached using the machinery of Lie derivatives which is found excessively mathematical by many, thereby putting them off. However, P3 appears less formidable when it is understood that this machinery was introduced by Lie precisely to deal with flow problems, and in any event as well as Larsson [13], there are textbooks [14, 15] which provide translation to more familiar vector notation. Simply stated and unsurprisingly in the light of their manner of introduction, allowed variations are constrained to take the form of Lie derivatives [16, § 7].

Possibility 3 may be made explicit by introducing the Lagrangian density ℓ which is a functional of a vector field $\mathbf{v}$ and a scalar field s, giving an action

$$L = \int \ell(\mathbf{v}, s) d\mathbf{x} dt \tag{8}$$

The resulting variational principle $\delta L = 0$ is

$$\delta L = \int \left(\frac{\delta \ell}{\delta \mathbf{v}} \cdot \delta \mathbf{v} + \frac{\delta \ell}{\delta s} \delta s \right) d\mathbf{x} dt = 0 \tag{9}$$

This must hold for variations $\boldsymbol{\eta}(\mathbf{r},t)$ in position $\mathbf{r}(t)$ of the fluid elements, which are arbitrary except that they must vanish at the endpoints of the time interval. Suppose the perturbations emerge as parameter ϵ increases from zero, as

$$\frac{\partial \mathbf{r}}{\partial \epsilon} = \boldsymbol{\eta}(\mathbf{r},t,\epsilon) \tag{10}$$

$$\frac{\partial \mathbf{r}}{\partial t} = \mathbf{v}(\mathbf{r},t,\epsilon) \tag{11}$$

Assuming $\mathbf{r}(t,\epsilon)$ and using Equations(10) and (11), the velocity variations are found to be constrained by

$$\delta\mathbf{v} = \frac{\partial \boldsymbol{\eta}}{\partial t} + \mathcal{L}_{\mathbf{v}}(\boldsymbol{\eta}), \tag{12}$$

and correspondingly scalar variations by

$$\delta s = -\mathcal{L}_{\boldsymbol{\eta}}(s) \tag{13}$$

where the Lie derivative of vector $\mathbf{w}$ is

$$\mathcal{L}_{\mathbf{v}}(\mathbf{w}) = -\mathbf{v}\cdot\nabla\mathbf{w} + \mathbf{w}\cdot\nabla\mathbf{v} \quad = \quad [\mathbf{v},\mathbf{w}] \quad \text{(Lie bracket)} \tag{14}$$

and the Lie scalar derivative is

$$\mathcal{L}_{\boldsymbol{\eta}}(s) = \boldsymbol{\eta}\cdot\nabla s \tag{15}$$

A point to note is that should variation involve changes to a pseudsoscalar such as magnetic field $\mathbf{B}$ or a pseudoscalar such as density ρ, then different definitions of Lie derivative apply, viz.

$$\mathcal{L}^2_{\boldsymbol{\eta}}(\mathbf{B}) = +\boldsymbol{\eta}\nabla\cdot\mathbf{B} - \nabla\times(\boldsymbol{\eta}\times\mathbf{B}) \tag{16}$$

$$\mathcal{L}^3_{\boldsymbol{\eta}}(\rho) = \nabla\cdot(\boldsymbol{\eta}\rho) \tag{17}$$

There is unfortunately the restriction to models of magnetic field and thermodynamic evolution which are non-dissipative. Although for example, Salmon [17] points out that dissipative effects can also be treated within a variational framework, albeit at the expense of solving for additional field variables, it is unclear how this might be used here, except as a guide as to the effects might best be discretised.

However, it seems the above variational principle provides little useful guidance as to how the non-dissipative effects might be discretised. The natural approach to the discretisation of Equation (9) would be to assume that the variations are specified by weighted sums of elements of a basis function space say $\boldsymbol{\phi}_{pqr}$, so that Equation (9) yields a set of equations, one for each member of the function space. The problem arises from the need to have a Lagrangian density $\ell = \frac{1}{2}\rho\mathbf{v}^2$ to model compressible flow. Evidently, the variational equation contains separate terms corresponding to variation of $\mathbf{v}$ and ρ respectively, namely

$$\delta L = \int \left(\rho\mathbf{v}\cdot\delta\mathbf{v} + \frac{1}{2}\mathbf{v}^2\delta\rho\right) d\mathbf{x}dt = 0 \tag{18}$$

Substituting for $\mathbf{v}$ using Equation (12) and $\delta\rho = -\mathcal{L}^3_{\boldsymbol{\eta}}(\rho)$ (Equation (17)) it is evident that the two separate terms in Equation (18) have to partially cancel analytically in order to give the expected $\rho\mathbf{v}\cdot\nabla\mathbf{v}$ term. Thus unless care is taken, the natural approach to discretisation may give rise to

additional spurious terms. If the standard analytic manipulations are pursued, the result is the usual ideal fluid evolution equations weighted by η integrated over time and space. It is interesting that the implied time weight function has to vanishes at the end-points of the range of integration, but really nothing useful for discretisation appears to arise from variational approaches, and it seems preferable to continue to discretise the hydrodynamic equations directly.

4 Modelling with spectral elements

This section contains material relevant to the ongoing re-shaping of *Nektar++* into a FEM library designed to suit NEPTUNE requirements, particularly regarding interfacing with UKAEA-developed particle codes. In order to clarify the scope of future work, some attention is given to accurate description of what is currently available within the code.

4.1 Refactoring and extension of *Nektar++*

Efforts to prepare *Nektar++* for the Exascale are vital for NEPTUNE purposes and currently constitute an effort to make the code compatible, in terms of performance and scalability, with large GPU-based systems (there is clearly strong developer motivation for this modification as well, given that the full DNS CFD simulation of a realistic system of industrial interest is often an exascale problem). Much of the current *Nektar++* code base was written using a coarse-grained MPI parallelism paradigm without support for vectorized arithmetic. GPU support is presently limited to specific code kernels and as such involves significant data transfer workload on and off the GPU and also run-time reconfiguration of data structures. A unifying strategy for an efficient GPU implementation involves separation of the data storage from the operators; hence, the current `ExpList` class is intended to transition to a pure storage object, called `FieldStorage` (NB. more recently renamed to just `Field`), which will be device-aware and capable of supporting a range of data layouts (and which may ultimately use a SYCL buffer). In addition, core operators will support CPUs (x86 and ARM, along with AVX2 / AVX512 and equivalents) and GPUs. There are many other details in the ongoing development plan eg. ensuring data locality in discontinuous Galerkin implementations, avoiding retaining in-memory unnecessary objects such as pre-static-condensation version of system matrices, and avoiding the use of shared pointers for small objects (which wastes execution time). These refactoring plans are in tandem with a migration to C++17.

The *Nektar++* Python interface is also being upgraded in scope from a workflow automation utility to a more general interface, leveraging the suitability of Python as a 'glue' language. Functionality will include the programmatic specification of simulations (bypassing the current requirement for an XML session file) and wrapping sufficient API functionality to facilitate solver integration with other software (note the ubiquity of Python in user interfaces) and also enable the construction of new solvers within Python in a DSL-like manner that avoids the need to recompile the library to generate a solver (ie. the 'outer loop' can be Python) or implement a novel operator.

The use of implicit time-stepping techniques has been noted as an area of great relevance for NEPTUNE. It has been noted that some *Nektar++* incompressible Navier-Stokes simulations are constrained by stability to use a time step that may be up to two orders of magnitude below that

needed to resolve the physics of interest - this is a problem that becomes worse the smaller the element size in the mesh. Recent developments in implementing a fully-implicit scheme (based on [18]) were presented at the recent *Nektar++* Workshop 2022 and showed promising initial results with an order of magnitude increase in stable time step size. Note that the current scheme is mixed with the diffusion treated implicitly and the advection explicitly in an IMEX time-advancing scheme; the explicit advection means that the code is subject to a maximum stable time step size from the CFL criterion. This work has the potential to speed up the simulations of convective heat transfer reported in [19] and may apply more generally to future NEPTUNE simulations.

4.2 Meshing

A number of additions have been made to the 2-D meshing capabilities of *NekMesh* under the programme of work in grant T/NA078/20. These include the ability to perform r- and h-refinement, or the generation of a quadrilateral-meshed 'boundary layer' region, local to a particular CAD-exact curve bounding the mesh in order to achieve a higher numerical resolution in the neighbourhood of a particular magnetic field surface. The magnetic field surfaces in a tokamak are three-dimensional so it will be necessary to provide similar capabilities in three-dimensional meshes, initially in the case of axisymmetric geometries.

Another meshing issue, which arises from the need to integrate *Nektar++* with a particle-in-cell code, is the need to generate non-physical 'halo' (aka 'guard') cells surrounding the sub-meshes that are generated when a large computational mesh is distributed over potentially a large number of MPI ranks. These 'halo' cells are necessary for keeping track of particles transitioning between MPI ranks. It is envisaged that this functionality will be added to *NekMesh*.

A final note concerns the higher-dimensional representation of non-equilibrium continuum fields, which is an alternate choice to a particle-based scheme. This involves meshes in up to three additional dimensions (velocity-space). It has become clear that the meshing of velocity space is non-trivial because of the existence of eg. outflow boundary conditions in the Parra et al formulation [1] and so it is not possible in all cases to mesh velocity space with a uniform grid.

4.3 Incorporation of finite element exterior calculus (FEEC) in *Nektar++*

The use of techniques from finite element exterior calculus (for which see [20]) is seen as one of the main pathways to an effective code that couples particles to an electromagnetic field. The necessary element types (here called 'conforming') and mathematical framework (for example, the weak mixed formulation of a partial differential equation) are not yet implemented within *Nektar++*. (Note, however, that the developers have considered the appropriateness of the numerical schemes where necessary eg. in the incompressible Navier-Stokes solver it is possible to specify a Taylor-Hood scheme known to give a stable discretization, in which the velocity components are discretized with a polynomial space one order higher than that used for the pressure.)

The classes of finite element currently implemented in *Nektar++* are listed below (and see p.50 onward of [7]), in which the nomenclature is as in a *Nektar++* session file. The bases divide into two types: *modal*, and *nodal*. The elements are generically implemented by mapping the physical

element to a reference element, which is typically defined on the coordinate interval $[-1, 1]$ per dimension. All elements store scalar degrees of freedom and there is currently no model for vector or tensor quantities within the core *Nektar++* libraries.

- `MODIFIED`: a modified orthogonal polynomial basis, with only two modes per dimension being non-zero on the element boundaries. Aside from this criterion, which reduces the density of the coupling between finite elements, the basis functions are not localized to particular nodes. The basis is hierarchical, meaning that increasing order $p-1$ to p requires only the addition of order-p modes rather than the reassignment of the entire basis and that lower-order computations are a subset of a given order. The basis functions in one dimension are, for order p and $\xi \in [-1, 1]$,

$$\psi_n(\xi) = \begin{cases} \frac{1-\xi}{2} & (n=0), \\ \left(\frac{1-\xi}{2}\right)\left(\frac{1-\xi}{2}\right) P_{n-1}^{1,1}(\xi), n = 1, \ldots, p-1 \\ \frac{1+\xi}{2} & (n=p). \end{cases} \tag{19}$$

  The $P_m^{\alpha,\beta}(\xi)$ denotes the order-m Jacobi polynomial (the case $\alpha = \beta = 0$ corresponds to the familiar Legendre polynomials and the $\alpha = \beta = 1$ case used here to their derivatives). This choice maintains a high degree of orthogonality, for example the elemental mass matrix has a mostly tri-diagonal structure and the Laplacian matrix has, except for the boundary modes, a diagonal structure.

- `GLL_LAGRANGE`: an interpolatory and therefore nodal basis in which a Gauss-Lobatto-Legendre (GLL) quadrature rule with $p+2$ points is used, giving an exact integration of the mass matrix, which is full.

- `GLL_LAGRANGE_SEM`: an interpolatory and therefore nodal basis in which a reduced GLL quadrature rule with $p+1$ points is used; the quadrature points are collocated with the Lagrange nodes. Without lumping, this results in a diagonal mass matrix with entries that are the quadrature weights. This is the 'classical' spectral element method.

Both GLL schemes are nodal bases in which the Lagrange polynomials are defined with points at the zeros of the Gauss-Lobatto polynomials. The only difference lies in the quadrature rule: a GLL quadrature with $p+1$ points is exact only up to and including polynomials of order $2p-1$ whereas the mass matrix, being bilinear in the basis functions, contains integrals of polynomials of order $2p$. Section 2.3 of ref [7] indicates that the error in the inexact diagonal lumped mass matrix is of the same order as the approximation error of the expansion.

The above elements are implemented in continuous Galerkin on 2-D quadrilaterals and 3-D hexahedra and also, with some exceptions, simplices (2-D triangles, 3-D tetrahedra); discontinuous Galerkin is available with the `MODIFIED` basis for all shape types but only quads / hexes for the nodal bases [21]. In addition, it is possible to specify different polynomial orders in different dimensions for tensor product elements as well as different point sets for the locations of pointwise field values.

Some of the important element types used in FEEC store vectorial degrees of freedom of various types (eg. the $H(\mathrm{div})$-conforming elements have degrees of freedom corresponding to the vector

field component normal to the element boundaries, and the $H(\mathrm{curl})$-conforming elements store components tangent to element edges). *Nektar++* currently represents vectors on a scalar-per-Cartesian component basis and so it seems that the vectorial element classes may bring additional benefit in terms of the amount of data stored (viz. store the component along an edge, rather than three independent Cartesian components). It is not known to the author whether eg. the reduced quadrature technique of classical spectral elements can be applied to the finite element types used in FEEC.

4.3.1 Transformation from reference element

One particular issue in implementing conforming discretizations (eg. the Raviart-Thomas elements [22], which discretize $H(\mathrm{div})$) in *Nektar++* lies with the strategy of mapping quantities from a reference element onto the general finite element in the mesh. This technique enables the precomputation of quadratures in the reference space (ie. once) in order to construct element matrices which are then mapped to each of the 'physical' finite elements. A lucid presentation is found in [23] in which is discussed assembly of $H(\mathrm{div})$ and $H(\mathrm{curl})$ finite elements, in the context of the FEniCS project [24].

For scalar quantities the transformation law applicable to the evaluation of an integral over a finite element is trivial:

$$\mathcal{F}^{\mathrm{scalar}}(\Phi) = \Phi \circ F^{-1}. \tag{20}$$

the content of which may be illustrated as follows. The mapping F of coordinates, for a simplex in two dimensions, where the target triangle has vertex coordinates (x_i, y_i) for $i = 1, 2, 3$, in the affine case is given by

$$x = r(x_2 - x_1) + s(x_3 - x_1), \tag{21}$$
$$y = r(y_2 - y_1) + s(y_3 - y_1) \tag{22}$$

Thus another way of regarding Equation (20) is that on the left, is the scalar function Φ as a function of 'real' space coordinates (x, y) and on the right as a function of local coordinates (r, s), so that $\mathcal{F}^{\mathrm{scalar}}(\Phi) = \Phi(x(r, s), y(r, s))$.

The Jacobian matrix of F is simply

$$J = \begin{pmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{pmatrix}. \tag{23}$$

It is apparent that the above transformation law does not in general preserve the continuity of normal or tangential traces that would be necessary for a continuous mapping of vector divergences or vector curls, otherwise respectively, $H(\mathrm{div}, \Omega_0) \to H(\mathrm{div}, \Omega)$ or $H(\mathrm{curl}, \Omega_0) \to H(\mathrm{curl}, \Omega)$ (where Ω_0 denotes the reference element and Ω the physical element). The normal trace continuity is preserved by the *contravariant Piola mapping*

$$\mathcal{F}^{\mathrm{div}}(\Phi) = \frac{1}{\det(J)} J\Phi \circ F^{-1}, \tag{24}$$

while the tangent trace continuity is preserved by the *covariant Piola mapping*

$$\mathcal{F}^{\mathrm{curl}}(\Phi) = J^{-T}\Phi \circ F^{-1}. \tag{25}$$

Note that in two dimensions, the contravariant and covariant Piola transforms are equivalent in the case of a conformal, orientation-preserving map, which is easily shown using the Cauchy-Riemann equations to equate components in the following:

$$\frac{1}{\det(J)}J = \frac{1}{\det(J)}\begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{pmatrix} \tag{26}$$

and

$$J^{-T} = \frac{1}{\det(J)}\begin{pmatrix} \frac{\partial v}{\partial y} & -\frac{\partial v}{\partial x} \\ -\frac{\partial u}{\partial y} & \frac{\partial u}{\partial x} \end{pmatrix}. \tag{27}$$

It would be necessary to write implementations of the transformations (24), 25 in order to provide *Nektar++* implementations of the relevant elements. An initial version could support a restricted range of operators, avoiding some of the complexity of a flexible form language implementation as in [23].

Note that the transformations to reference element are more involved for the non-straight-sided elements that are likely to occur when using meshes that conform to the first wall or to magnetic flux surfaces, though the results that the Piola transformations preserve the stated trace continuities still hold for smooth nondegenerate mappings where the Jacobian matrix is invertible.

4.3.2 Sum-factorization

The sum-factorization or tensor product technique was re-discovered by Orszag [25] and is considered to be the key to the efficiency of spectral methods. It relies on the expansions having a tensor product decomposition. A description can be found on [7, p.179], in which the central formula is

$$U_{ij} = \sum_{r=1}^{p}\sum_{s=1}^{p} f_{rs}h_{ir}h_{js} \equiv \sum_{r=1}^{p} h_{ir}\left(\sum_{s=1}^{p} f_{rs}h_{js}\right), \tag{28}$$

the key observation being that the bracket on the in the right-hand-side expression is independent of i and so the bracket is computed once for each choice of r, j and then *re.used* for each i. This reduces the $\mathcal{O}(p^4)$ complexity of the double sum and the evaluations over i, j to two $\mathcal{O}(p^3)$ summations. The double sum in (28) is appropriate to 2-D reductions but recursive application of the same principle leads to further efficiencies in higher dimensions; generally it is possible to reduce the algorithmic complexity of operations such as the above from p^{2D} to p^{D+1} where D is the number of space dimensions. Note there must also be a tensor product structure in the

computed quantity ie. it is labelled by i and j and there is no gain, for example, for the repeated evaluation of a tensor-product field at a set of $\mathcal{O}(p)$ arbitrary points (in this case there is a δ_{ij} on both sides of the equation and the sums are already $\mathcal{O}(p^3)$).

There are a number of deployments of this tactic within *Nektar++*, for example the backward transformation which corresponds to evaluation of the field at quadrature points starting from the coefficients of the expansion basis (ie. a change of basis) and is called as `BwdTrans` in the API - see implementations for various element types eg. `Nektar::Collections::BwdTrans_SumFac_Tri`, `Nektar::Collections::BwdTrans_SumFac_Hex` (see [26]). The inner product is another example (see eg. `Nektar::Collections::IProductWRTBase_SumFac_Quad`).

It is also obvious that the technique applies to the interpolation of field data onto a rectilinear grid, perhaps for producing a visual display of the spatial field or a representation suitable for data processing eg. spatial Fourier transform. The example in Appendix B shows a similar sort of efficiency gain.

It will be essential to leverage this technique, wherever applicable, in any new code. The paper [27] contains description of sum factorization on cuboid cells for $H(\mathrm{div})$ and $H(\mathrm{curl})$ conforming elements within the *Firedrake* framework [28] . The potential applicability of sum factorization for conforming element types on simplices is currently less clear; an initial implementation could be restricted to regular meshes and slab geometries in NEPTUNE.

5 Summary

Spectral element models implemented in the *Nektar++* framework continue to be central to future plans for NEPTUNE. There is clear evidence that the *Nektar++* developers have designs to ensure that their code can continue to prosper in a GPU-dominated exascale era and also to improve the user experience regarding adding new solvers and interfacing with other codes by means of the Python interface. Material in this report has been written with the aim of indicating potential issues in implementing finite element exterior calculus techniques in *Nektar++* with reference to publications by the developers of FEniCS / *Firedrake*. The extent to which these methods synchronize with the existing efficiency optimizations of *Nektar++* (chiefly, sum factorization) remains to be fully explored.

As far as the overall project NEPTUNE is concerned, resolving the numerical difficulties encountered in the work of Grant T/NA085/20 ought to be the priority objective. There is a lack of detail at present to how their root cause might be identified and the difficulties overcome, but assuming the root cause to be the discontinuous nature of the distribution function at the effective wall, then there are range of subgridding techniques, most notably artificial viscosities such as SVV, 'Spectral Vanishing Viscosity' (Tadmor, as referenced in the textbook by Karniadakis and Sherwin [7]) to mollify the Gibbs' phenomenon. However, other causes may necessitate the use of ideas from moving boundary numerics, mappings, and/or immersing boundaries.

Acknowledgement

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

References

[1] J. T. Parker and W. Arter. Technical report on Physics model selection. Technical Report CD/EXCALIBUR-FMS/0058-M2.8.2, UKAEA, 2021. `https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/ukaea_reports/CD-EXCALIBUR-FMS0058-M2.8.2.pdf`.

[2] J.W. Eastwood and W. Arter. Interpretation of disruptions in tokamak simulations. *Physical Review Letters*, 57:2528–2531, 1986. `http://dx.doi.org/10.1103/PhysRevLett.57.2528`.

[3] J.W. Eastwood and W. Arter. Spurious behaviour of numerically computed fluid flow. *I.M.A. Journal of Numerical Analysis*, 7:205–222, 1987.

[4] D.J. Acheson. Local analysis of thermal and magnetic instabilities in a rapidly rotating fluid. *Geophysical & Astrophysical Fluid Dynamics*, 27(1-2):123–136, 1983.

[5] W.O. Criminale, T.L. Jackson, and R.D. Joslin. *Theory and Computation in Hydrodynamic Stability*. Cambridge University Press, 2003.

[6] J.P. Boyd. *Chebyshev and Fourier Spectral Methods*. Dover Publications, Mineola, NY, 2001. http://www-personal.umich.edu/ jpboyd/BOOK_Spectral2000.html.

[7] 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.

[8] Linda E Reichl. *A Modern Course in Statistical Physics 4th Ed.* John Wiley & Sons, 2016.

[9] P. Helander and D.J. Sigmar. *Collisional transport in magnetized plasmas*. Cambridge University Press, 2005.

[10] W. Arter. Equations for EXCALIBUR/NEPTUNE Proxyapps. Technical Report CD/EXCALIBUR-FMS/0021-1.20-M1.2.1, UKAEA, 2021. https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/ukaea_reports/CD-EXCALIBUR-FMS0021-1.01-M1.2.1.pdf.

[11] W. Arter. Numerical simulation of magnetic fusion plasmas. *Reports on Progress in Physics*, 58:1–59, 1995. http://dx.doi.org/10.1088/0034-4885/58/1/001.

[12] Code generation QPROG style, example of Braginskii plasma transport coefficients. https://github.com/wayne-arter/smardda-misc, 2017. Accessed: December 2020.

[13] J. Larsson. A practical form of Lagrange–Hamilton theory for ideal fluids and plasmas. *Journal of Plasma Physics*, 69(3):211–252, 2003.

[14] D. Lovelock and H. Rund. *Tensors, Differential Forms and Variational Principles*. Dover, New York, 1988.

[15] G. Webb. *Magnetohydrodynamics and Fluid Dynamics: Action Principles and Conservation Laws*. Springer, 2018.

[16] D.D. Holm, J.E. Marsden, and T.S. Ratiu. The Euler Poincare Equations and Semidirect Products with Applications to Continuum Theories. *Advances in Mathematics*, 137:1–81, 1998.

[17] R. Salmon. Hamiltonian fluid mechanics. *Annual Review of Fluid Mechanics*, 20:225–256, 1988.

[18] Dong S. and Shen J. An unconditionally stable rotational velocity-correction scheme for incompressible flows. *Journal of Computational Physics* **229** *(2010), 7013-7029*, 2010.

[19] E. Threlfall and W. Arter. Finite Element Models: Complementary Activities I. Technical Report CD/EXCALIBUR-FMS/0051-M6.1, UKAEA, 2021. https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/ukaea_reports/CD-EXCALIBUR-FMS0051-M6.1.pdf.

[20] D.N. Arnold. *Finite Element Exterior Calculus*. CBMS-NSF regional conference series in applied mathematics **93**, SIAM, 2018.

[21] Moxey D. Private communication, 2022.

[22] Raviart-Thomas finite elements. https://en.wikipedia.org/wiki/Raviart-Thomas_basis_functions. Accessed: September 2022.

[23] Rognes M.E., Kirby R.C., and Logg A. Efficient assembly of H(div) and H(curl) conforming finite elements. *SIAM Journal on Scientific Computing 31, no. 6 (2010): 4130-4151*, 2010.

[24] FEniCS Project. https://fenicsproject.org/. Accessed: September 2022.

[25] Orszag S.A. Spectral methods for problems in complex geometries. *Journal of Computational Physics* **37***, 70*, 1980.

[26] Nektar++ documentation: BdwTrans.cpp file reference. https://doc.nektar.info/doxygen/4.4.1/_bwd_trans_8cpp.html. Accessed: October 2022.

[27] Homolya M., Kirby R.C., and Ham D.A. Exposing and exploiting structure: optimal code generation for high-order finite element methods. *submitted to ACM Transactions on Mathematical Software*, 2017.

[28] Florian Rathgeber, David A. Ham, Lawrence Mitchell, Michael Lange, Fabio Luporini, Andrew T. T. Mcrae, Gheorghe-Teodor Bercea, Graham R. Markall, and Paul H. J. Kelly. *Firedrake*: automating the finite element method by composing abstractions. *ACM Trans. Math. Softw., 43(3):24:124:27, 2016. URL: http://arxiv.org/abs/1501.01809, arXiv:1501.01809, doi:10.1145/2998441.*, 2015.

[29] Busch K., König M., and Niegemann J. Discontinuous Galerkin methods in nanophotonics. *Laser Photonics Rev. 5, No. 6, 773809 (2011) / DOI 10.1002/lpor.201000045*, 2011.

[30] Niegemann J., Diehl R., and Busch K. Efficient low-storage Runge-Kutta schemes with optimized stability regions. *Journal of Computational Physics* **231** *Issue 2 (2012), 364-372*, 2012.

A Extracts from External Grants for NEPTUNE Y4-6

A.1 Spectral element procurement

Spectral elements will form the core of the NEPTUNE software, and thus any usage thereof must perform well at the Exascale. Moreover, the elements need to have non-planar surfaces in order to represent more efficiently and accurately the first wall of the tokamak. The requirement for close coupling with a high-dimensional representation of the plasma, such as either eg. by particles or by gyro-averaged kinetic equations, represents a further challenge. A specific NEPTUNE objective which must be supported as a priority is the production of a $2d3v$ proxyapp. to be helped by completing Proxyapp 2-6 (described under number 6 in FM-WP2) following the plan set out in a report to be produced under contract T/NA083/20 or otherwise.

There will therefore be an ongoing need throughout the remainder of the project, to support usage and development of a library of spectral elements, such as Nektar++. Software development will be necessary in order to accommodate efficient interaction with the high-dimensional representations,

and likely be required in order to ensure efficient operation on new architectures, indeed for long term code stability, it would be desirable to refactor the library to minimise dependencies on other libraries. There will be a similar ongoing need to support usage and development of a mesher geared to the production of non-planar high order finite elements, such as Nekmesh.

The outputs will therefore be the core of an efficient spectral element mesher and a library or package of libraries of spectral elements tuned for use at Exascale and the special needs of project NEPTUNE, as represented by higher layer components corresponding to respectively a developer DSL (in SYCL/C++) and an end-user DSL at the level of Python or Julia. The supporting Nektar++ library/ies must be demonstrated in particular to enable efficient solution for Proxyapp 2-6, for the $2d3v$ proxyapp, and for 3-D elliptic problems at Exascale. Proxyapps, to include Proxyapp 3-2, should conform to emerging high standards for the production of NEPTUNE software.

A.2 Advanced Referent Model Procurement

The objective is to produce a gyro-averaged theoretical time dependent model of plasma suitable for project NEPTUNE, meaning that it is of actionable accuracy in the outer pedestal, separatrix region and scrape-off layers of a tokamak plasma. Building on the work of Grant T/NA085/20 or otherwise, such a generally 5-D model (3-D in space, 2-D in velocity space) is likely to have the property of reducing to a fluid-like (5-moment, only spatial dependence) model in regions of higher collisionality. Such a model is expected to require special treatments for the separatrix and for the wall, in order to account for realistic magnetic equilibria and ensure accurate modelling of the geometry. Non-axisymmetric electromagnetic field effects, and coupling between different plasma species in addition to neutral components should also be considered.

The output will be theoretical models that include in the aforementioned effects, wherein the derivation of suitable collision operator will be of especial importance. The practicality of these models should be demonstrated by suitable proxyapps, conforming to the emerging high NEPTUNE standard where appropriate.

B Note on a computationally-efficient discontinuous Galerkin method

In this section a simple example of a *modal* time-evolution code for Maxwell's equations in two dimensions is exhibited as an example showing the types of structure that can be exploited for numerical optimizations. The implementation is a time-explicit upwinded discontinuous Galerkin scheme; for more details, albeit with a focus on nodal methods, see [29]. The method shows reduction in algorithmic complexity (over a naive implementation) equivalent to that in the description of sum-factorization in the main text.

For simplicity, assume a uniform grid of square elements. The mapping between the reference elements and the physical element is a trivial rescaling. The hierarchical basis functions per dimension are the Legendre polynomials $\psi_n(\xi) = P_n^{0,0}(\xi)$ on the interval $\xi \in [-1, 1]$ and the normalization of the two-dimensional tensor product modes is such that

$$\int_{-1}^{1}\int_{-1}^{1}\psi_m(x,y)\psi_n(x,y)d^2x = \delta_{mn}. \tag{29}$$

This means that the element mass matrix is explicitly a multiple of the identity matrix.

Other matrices required for the DG scheme are the element stiffness matrices

$$(S_x)_{mn} = \int_{-1}^{1}\int_{-1}^{1}\psi_m(x,y)\frac{\partial}{\partial x}\psi_n(x,y)d^2x, \tag{30}$$

$$(S_y)_{mn} = \int_{-1}^{1}\int_{-1}^{1}\psi_m(x,y)\frac{\partial}{\partial y}\psi_n(x,y)d^2x, \tag{31}$$

and the element face mass matrices are, for example

$$(F_{y+})_{mn} = \int_{-1}^{1}\int_{-1}^{1}\psi_m(x,y)\psi_n(x,y)\delta(y-1)d^2x. \tag{32}$$

These need to be evaluated on the appropriate edge (ie. subspace of codimension one) ie. at a fixed value of one of the coordinates as indicated by the delta function.

B.1 One-dimensional implementation

Maxwell's equations in vacuo are, in natural units,

$$\dot{E} = \nabla \times H, \tag{33}$$

$$\dot{H} = -\nabla \times E. \tag{34}$$

Specializing to one dimension (x), these become

$$\dot{E}_y = -\partial_x H_z, \tag{35}$$

$$\dot{H}_z = -\partial_x E_y. \tag{36}$$

The computational mesh is the interval $[0,1]$ divided into N equal elements with a periodic condition at the interval ends. The basis is Legendre function per element and so the two lowest modes are, on element labelled by n where $n = 0, 1, ..., N-1$,

$$\psi_0^n(x) = \sqrt{N} \tag{37}$$

$$\psi_1^n(x) = \sqrt{12N^3}\left(x - \frac{n+\frac{1}{2}}{N}\right). \tag{38}$$

These are normalized such that $\int_n \psi_a^n(x)\psi_b^n(x)dx = \delta_{ab}$ where $\int_n \equiv \int_{\frac{n}{N}}^{\frac{n+1}{N}}$ ie. to give an identity local (and global) mass matrix.

The element stiffness matrix has one non-zero component

$$(S_x)_{01} \equiv \int_n \psi_0^n(x)\frac{\partial}{\partial x}\psi_1^n(x)dx = \sqrt{12}N. \tag{39}$$

The edge mass matrices are computed as (integration over point is point evaluation)

$$\begin{aligned}(F_{x-}) &= \psi_a^n(x)\psi_b^n(x)|_{x=\frac{n}{N}}, &(40)\\ (F_{x+}) &= \psi_a^n(x)\psi_b^n(x)|_{x=\frac{n+1}{N}}. &(41)\end{aligned}$$

Hence

$$\begin{aligned}(F_{x-})_{00} &= N, &(42)\\ (F_{x-})_{01} &= -\sqrt{3}N, &(43)\\ (F_{x-})_{10} &= -\sqrt{3}N, &(44)\\ (F_{x-})_{11} &= 3N. &(45)\\ & &(46)\end{aligned}$$

and

$$\begin{aligned}(F_{x+})_{00} &= N, &(47)\\ (F_{x+})_{01} &= \sqrt{3}N, &(48)\\ (F_{x+})_{10} &= \sqrt{3}N, &(49)\\ (F_{x+})_{11} &= 3N. &(50)\\ & &(51)\end{aligned}$$

The initial condition is a cosine standing wave

$$\begin{aligned}E_y &= 0, &(52)\\ H_z &= \cos 2\pi x. &(53)\end{aligned}$$

The required initial data are overlap integrals between this initial condition and the individual modes.

The upwind numerical fluxes to be applied to $\dot{E}_y$ and $\dot{H}_z$ respectively are

$$\frac{(\Delta E - n(n\cdot\Delta E) + n\times\Delta H}{2}, \tag{54}$$

$$\frac{(\Delta H - n(n\cdot\Delta H) - n\times\Delta E}{2}. \tag{55}$$

The left-hand-side fluxes evaluate to

$$\frac{\Delta E_y + \Delta H_z}{2}, \tag{56}$$

$$\frac{\Delta H_z + \Delta E_y}{2}, \tag{57}$$

and the right-hand-side fluxes

$$\frac{\Delta E_y - \Delta H_z}{2}, \tag{58}$$

$$\frac{\Delta H_z - \Delta E_y}{2}. \tag{59}$$

The field differences eg. ΔE_y are always 'outside minus inside' with respect to the element in question.

If the system is zeroth-order, the situation is very simple. There is no stiffness matrix and the semi-discrete equations take the form

$$\begin{aligned} \dot{E}_n &= \frac{N}{2}\left(E_{n+1} + E_{n-1} - 2E_n + H_{n-1} - H_{n+1}\right), \\ \dot{H}_n &= \frac{N}{2}\left(H_{n+1} + H_{n-1} - 2H_n + E_{n-1} - E_{n+1}\right). \end{aligned} \tag{60}$$

where N appears where it is more usual to see $1/h$. The first 3 terms may be recognised as the simplest finite difference approximation to a diffusion term $h\partial^2/\partial x^2$. This is an example of a finite-volume method (which here corresponds to zero-order DG; note that higher-order extensions of finite-volume methods, eg. the Godunov scheme, do exist).

The 1-D code can be run with a choice of time stepper (a 14-stage 5^{th} LSRK scheme noted for its good stability properties is used [30]). Outputs from the code can be seen in Fig.1. Note that it is expected that an upwind DG code is numerically dissipative, hence the loss of amplitude seen in the figure (the dissipation diminishes spectrally as the order is increased).

B.2 Two-dimensional implementation

Computation of the relevant matrices makes clear the efficiency improvements of this scheme over eg. a nodal scheme in which all the matrices are full. Note that basis in 2-D falls into the

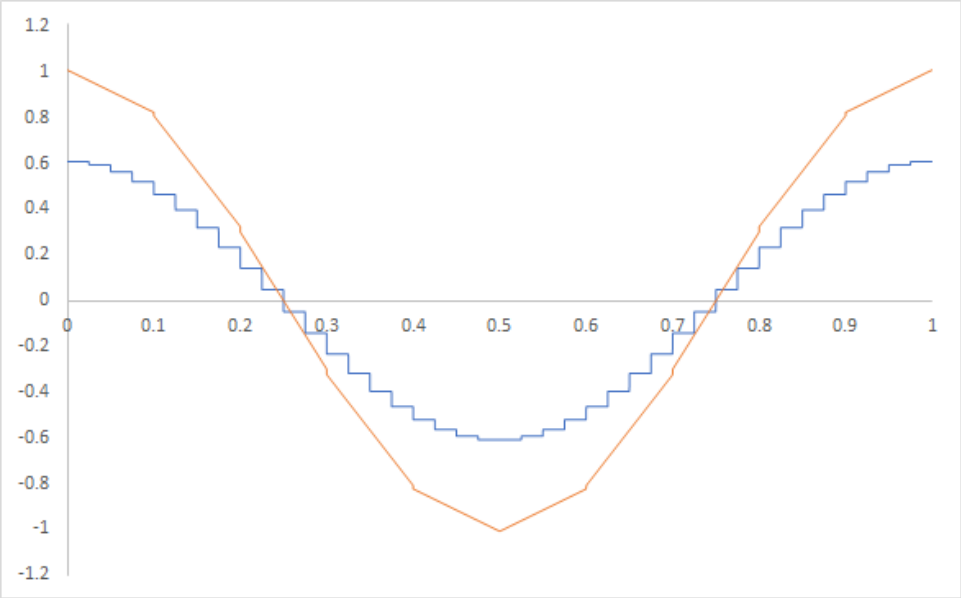


Figure 1: Output of the magnetic field from the one-dimensional DG code after one optical cycle, after which the analytic solution has returned to the cosine initial condition. This is seen to be approximately true for the numerical solutions, however, the 40-element zero-order code output (blue curve) has decayed in amplitude due to the dissipative nature of the upwinded scheme. The ten-element first-order code (orange curve) produces a much better answer. Note the evident field discontinuities at element boundaries in both examples.

familiar triangular binomial pattern (with the number of modes needed for order-p being the pth triangular number); tiers in this triangle contain modes of equivalent aggregate polynomial order. This is exhibited in Table 1 which shows the ten modes needed for $p = 3$ in 2-D; all are defined on $(x, y) \in [-1, 1]^2$. The equivalent basis in 3-D has a 'trinomial' structure (and the total number of modes is a tetrahedal number).

mode number i	tier	mode num x	mode num y	mode ψ_i
0	0	0	0	$\frac{1}{2}$
1	1	1	0	$\frac{\sqrt{3}}{2}x$
2	1	0	1	$\frac{\sqrt{3}}{2}y$
3	2	2	0	$\frac{\sqrt{5}}{4}(3x^2-1)$
4	2	1	1	$\frac{3}{2}xy$
5	2	0	2	$\frac{\sqrt{5}}{4}(3y^2-1)$
6	3	3	0	$\frac{\sqrt{7}}{4}(5x^3-3x)$
7	3	2	1	$\frac{\sqrt{15}}{4}(3x^2-1)y$
8	3	1	2	$\frac{\sqrt{15}}{4}x(3y^2-1)$
9	3	0	3	$\frac{\sqrt{7}}{4}(5y^3-3y)$

Table 1: Table showing the lowest ten modes in 2-D.

The non-zero entries of the x-stiffness matrix, defined as

$$(S_x)_{ij} \equiv \int_{-1}^{1} dx \int_{-1}^{1} dy \, \left(\psi_i \, \frac{\partial \psi_j}{\partial x} \right) \tag{61}$$

are

$$\begin{aligned} (S_x)_{01} &= \sqrt{3} \\ (S_x)_{06} &= \sqrt{7} \\ (S_x)_{13} &= \sqrt{15} \\ (S_x)_{24} &= \sqrt{3} \\ (S_x)_{36} &= \sqrt{35} \\ (S_x)_{47} &= \sqrt{15} \\ (S_x)_{58} &= \sqrt{3} \end{aligned} \tag{62}$$

(63)

The matrix is sparse (7 non-zero entries out of a possible 100) and the entries are simply square roots of odd numbers. Note that the higher-order modes function effectively as finite-volume as there is no coupling between them from the stiffness matrices.

The edge mass matrices show a conspicuous pattern that allows a computational efficiency gain (Fig.2). Modes associated to a particular polynomial order repeat the structure present at lower orders, up to a multiplier, while adding one new row structure. The 'repeated' rows summed against

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	1	0	3	0	0	5	0	0	0	7	0	0	0	0	9	0	0	0	0	0	11
1	0	1	0	0	3	0	0	0	5	0	0	0	0	7	0	0	0	0	0	9	0
2	3	0	9	0	0	15	0	0	0	21	0	0	0	0	27	0	0	0	0	0	33
3	0	0	0	1	0	0	0	3	0	0	0	0	5	0	0	0	0	0	7	0	0
4	0	3	0	0	9	0	0	0	15	0	0	0	0	21	0	0	0	0	0	27	0
5	5	0	15	0	0	25	0	0	0	35	0	0	0	0	45	0	0	0	0	0	55
6	0	0	0	0	0	0	1	0	0	0	0	3	0	0	0	0	0	5	0	0	0
7	0	0	0	3	0	0	0	9	0	0	0	0	15	0	0	0	0	0	21	0	0
8	0	5	0	0	15	0	0	0	25	0	0	0	0	35	0	0	0	0	0	45	0
9	7	0	21	0	0	35	0	0	0	49	0	0	0	0	63	0	0	0	0	0	77
10	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	3	0	0	0	0
11	0	0	0	0	0	0	3	0	0	0	0	9	0	0	0	0	0	15	0	0	0
12	0	0	0	5	0	0	0	15	0	0	0	0	25	0	0	0	0	0	35	0	0
13	0	7	0	0	21	0	0	0	35	0	0	0	0	49	0	0	0	0	0	63	0
14	9	0	27	0	0	45	0	0	0	63	0	0	0	0	81	0	0	0	0	0	99
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	9	0	0	0	0
17	0	0	0	0	0	0	5	0	0	0	0	15	0	0	0	0	0	25	0	0	0
18	0	0	0	7	0	0	0	21	0	0	0	0	35	0	0	0	0	0	49	0	0
19	0	9	0	0	27	0	0	0	45	0	0	0	0	63	0	0	0	0	0	81	0
20	11	0	33	0	0	55	0	0	0	77	0	0	0	0	99	0	0	0	0	0	121

Figure 2: Repeating pattern in edge mass matrix for the top edge of a quadrilateral element where the index i labels the row and j the column. Numbers shown here are the actual matrix entries multiplied by two and then squared, in order to make the underlying structure clear; the shading colour indicates rows in which the squared matrix components are related to the 'first' row of a particular colour by multiplication by an odd integer. Note that the tiers (modes of common polynomial order) have columns where the non-zero components correspond to odd entries of the $1, 3, 5, ...$ times tables.

the vector of field differences are computed simply by rescaling previously-computed sums. Possibly the identity $\frac{dP_n(x)}{dx} = \frac{n}{x^2-1}\left(xP_n(x) - P_{n-1}(x)\right)$ relating the derivative of a Legendre polynomial to two Legendre polynomials, plays a role.

Note that in *nodal* DG, the numerical fluxes involve only the nodes that are localized at the element boundary; however, the full mass matrix means involving *all* modes in the flux calculation.

B.2.1 Concluding remarks

The efficiencies shown here are yet more pronounced in three dimensions, as observed earlier in the discussion on sum-factorization.

If it is necessary to include non-rectilinear features in the mesh, it is possible to mesh using a mixture of simplices near boundaries and rectilinear elements in bulk regions. An example mesh is shown in Fig.3. In this case, the strategy design pattern could be used to enable the run-time selection of algorithm to compute the numerical flux depending on the types of elements involved (discussion of numerical stability questions associated to such processes is omitted here).

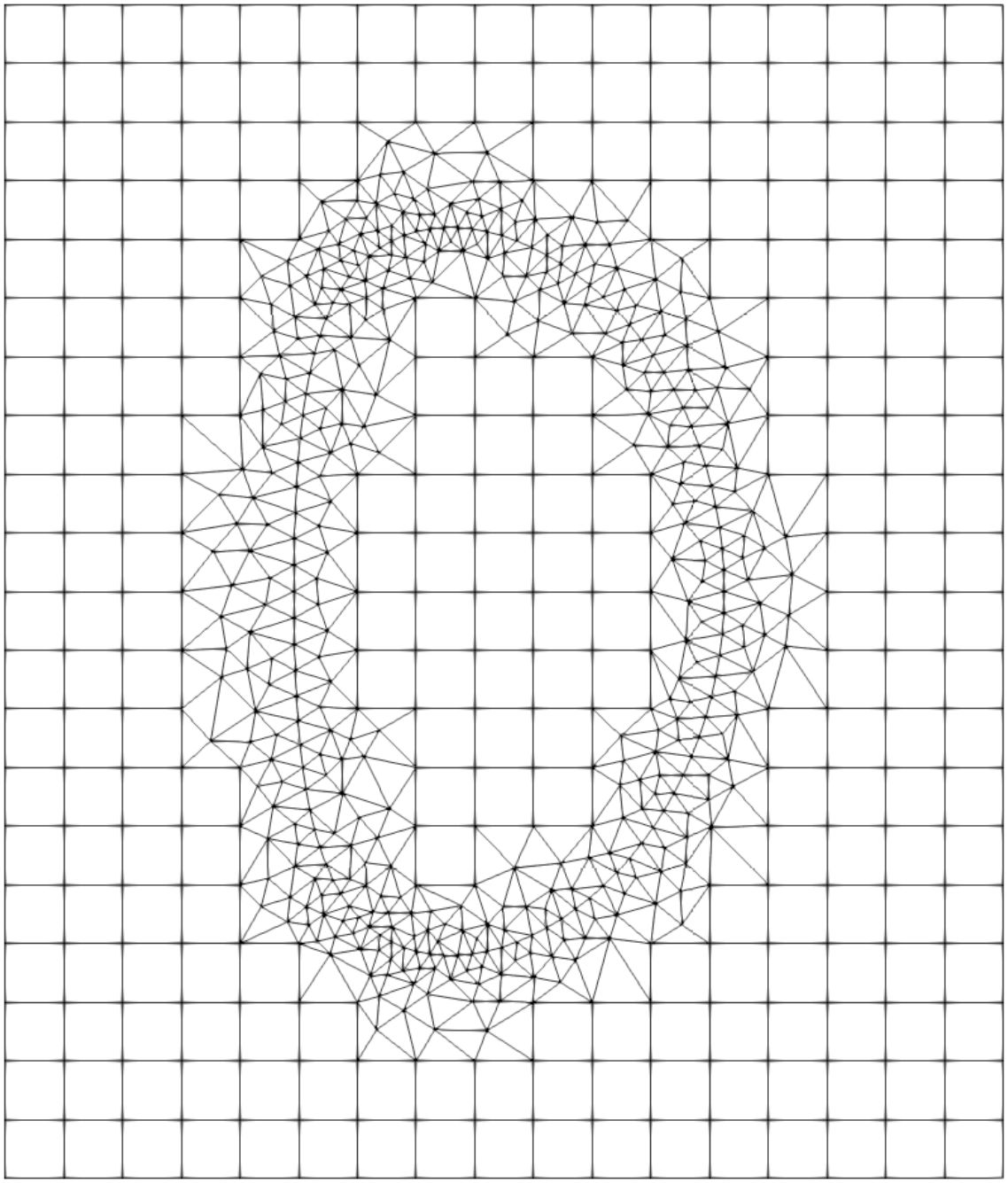


Figure 3: Example mesh using regular quadrilaterals in bulk regions and constrained Delaunay triangulation local to non-axis-aligned geometry, in this case a D-shaped closed curve intended as proxy for a half-cross-section of a tokamak first wall.