UK Atomic Energy Authority

UKAEA-STEP-PR(23)09

R. W. Eardley, A. J. Dubas, A. Davis

On scalable liquid-metal MHD solvers for fusion breeder blanket multiphysics applications

On scalable liquid-metal MHD solvers for fusion breeder blanket multiphysics applications

R. W. Eardley, A. J. Dubas, A. Davis

This is a preprint of a paper submitted for publication in
Plasma Physics and Controlled Fusion

On scalable liquid-metal MHD solvers for fusion breeder blanket multiphysics applications

R W Eardley, A J Dubas and A Davis

UK Atomic Energy Authority, Culham Science Centre, Abingdon, UK

E-mail: `rupert.eardley@ukaea.uk`

Abstract. While substantial research effort has been made recently in the development of computational liquid-metal magnetohydrodynamics (MHD) solvers, this has typically been confined to closed-source and commercial codes. This work aimed to investigate some open-source alternatives. Two OpenFOAM-based MHD solvers, `mhdFoam` and `epotFoam`, were found to show strong scaling profiles typical of fluid dynamics codes, while weak scaling was impeded by an increase in iterations per timestep with increasing resolution. Both were found to accurately solve the Shercliff and Hunt flow problems for Hartmann numbers from 20 to 1000, except for `mhdFoam` which failed in the Hunt flow $Ha = 1000$ case. An inductionless MHD solver was implemented in the Proteus MOOSE application as a proof of concept, using two methods referred to as the kernel method and material method. The material method was found to converge with a wider range of preconditioners than the kernel method, however the kernel method was found to be significantly more accurate. Future work will aim to build on these studies, exploring more advanced OpenFOAM MHD solvers as well as improving the Proteus MHD solver.

Keywords: liquid-metal magnetohydrodynamics, OpenFOAM, MOOSE

Submitted to: *Plasma Phys. Control. Fusion*

1. Introduction

A key step in accelerating the development of magnetic confinement fusion (MCF) power plants is developing the capability to study and design tokamak components *in silico*, reducing the financial costs of prototyping and testing through predictive modelling [1]. Digital replicas and twins of these components, potentially even extending to full plant simulations, pose a significant challenge, requiring carefully validated multiphysics software that can make efficient use of upcoming exascale high-performance computing (HPC) systems. These multiphysics packages must therefore be scalable (to maximise the size of problems that can be studied), as well as portable (such that they function on the architectures of future HPC resources), in addition to being capable of accurately simulating the physics involved.

For MCF devices to become commercially viable, a key component will be the breeder blanket used to generate tritium required for the fusion reaction [2]. Some designs of these blankets use a liquid metal as the breeding material, the flow of which is governed by magnetohydrodynamics (MHD) due to the interaction with the strong magnetic fields confining the plasma. These components, and the complex liquid-metal flows within them, have been the target of significant research effort in recent years, with many studies aiming to improve liquid-metal magnetohydrodynamics (LM-MHD) simulation capabilities [3, 4]. These efforts have typically been focused on closed-source research codes and commercial solvers, which have shown promising results [5]. However, the closed-source nature of the research codes makes access challenging, while prohibitive licensing and digital rights management of commercial solvers restricts portability.

In order to capture all the key physics involved in breeder blanket designs with liquid-metal flows in multiphysics simulations, LM-MHD solvers must be integrated into multiphysics packages. The capability already exists to study some of the physics involved, such as in the AURORA code which couples MOOSE and OpenMC for neutron transport and thermo-mechanical analysis [6]. MOOSE is an open-source, parallel finite element method (FEM) library which simplifies coupling between MOOSE-based applications as well as external codes through its MultiApp system [7], and is reported to be highly scalable [8]. AURORA is part of a wider suite of open-source tools under development at `https://github.com/aurora-multiphysics`, covering fluid dynamics (Proteus), electromagnetism (Apollo and Hephaestus), tritium transport (Achlys) and more. Adding LM-MHD capability to this collection will enable more detailed fusion breeder blanket simulations, and fill a gap in the current capability of the suite. This could be achieved by introducing a new application to couple in an external open-source solver, or by adding new physics to one of the existing solvers such as Proteus using the inbuilt FEM solver capabilities of MOOSE.

To simulate conducting fluids in magnetic fields, several effects must be considered in addition to capturing the fundamental coupling between fluid dynamics and electromagnetism in the bulk of the fluid. The behaviour at the walls must be resolved, particularly in terms of electromagnetic coupling to the walls (and beyond) and boundary layers that become extremely narrow for high magnetic flux densities [3]. Furthermore, flows may be turbulent, and so this behaviour must either be resolved through direct numerical simulation (DNS) or modelled using methods such as large-eddy simulation (LES) [9]. Thermal effects, such as buoyancy and temperature dependence of material properties, must also be considered [3].

This work aims to investigate existing open-source LM-MHD solvers, as well as introduce a new proof-of-concept implementation using MOOSE. An overview of relevant LM-MHD and parallel scaling theory is provided in section 2. Section 3 details a study of two OpenFOAM-based solvers, `mhdFoam` and `epotFoam`, evaluating them in terms of both parallel scaling and validation against analytic solutions. Section 4 describes an early implementation of a new LM-MHD solver in the Proteus MOOSE application, before drawing overall conclusions in section 5.

2. Theory

2.1. Liquid-metal MHD

The set of partial differential equations (PDEs) governing liquid metals can be derived by combining the incompressible Navier-Stokes equations with Maxwell's equations, as detailed in [10]. For a single fluid, with constant fluid properties (density ρ, kinematic viscosity ν, magnetic permeability μ and conductivity σ), the resulting equations are

$$\rho\frac{\partial \boldsymbol{u}}{\partial t} + \rho(\boldsymbol{u}\cdot\nabla)\boldsymbol{u} - \rho\nu\nabla^2\boldsymbol{u} + \nabla p = \boldsymbol{J}\times\boldsymbol{B} \tag{1a}$$

$$\nabla\cdot\boldsymbol{u} = 0 \tag{1b}$$

$$\frac{\partial \boldsymbol{B}}{\partial t} + \frac{1}{\mu}\nabla\times\left(\frac{1}{\sigma}\nabla\times\boldsymbol{B}\right) = \nabla\times(\boldsymbol{u}\times\boldsymbol{B}) \tag{1c}$$

$$\nabla\cdot\boldsymbol{B} = 0 \tag{1d}$$

$$\boldsymbol{J} = \frac{1}{\mu}(\nabla\times\boldsymbol{B}) \tag{1e}$$

where $\boldsymbol{u}$ is flow velocity, p is pressure, $\boldsymbol{B}$ is magnetic flux density and $\boldsymbol{J}$ is current density. These equations describe momentum conservation

$(1a)$, incompressibility $(1b)$, magnetic induction $(1c)$, the solenoidal nature of magnetic fields $(1d)$ and Ampère's law neglecting electric displacement currents $(1e)$. Current density $\boldsymbol{J}$ can be eliminated by substituting $(1e)$ into the Lorentz force term in $(1a)$.

Details of the mechanism can be found in texts such as [11], however it can be briefly summarised as follows. The motion of a conducting fluid relative to an imposed magnetic field $\boldsymbol{B}_0$ induces current density $\boldsymbol{J} = \sigma(\boldsymbol{E} + \boldsymbol{u} \times \boldsymbol{B})$ in the fluid. These currents induce a magnetic field $\boldsymbol{b}$, such that the field lines of the total field $\boldsymbol{B} = \boldsymbol{B}_0 + \boldsymbol{b}$ are dragged with the fluid flow. A Lorentz force $\boldsymbol{f}_{\mathrm{EM}} = \boldsymbol{J} \times \boldsymbol{B}$ acts on the fluid, opposing flow perpendicular to B-field lines. The resulting system forms a strongly coupled feedback loop. Joule heating provides an additional source of energy dissipation, acting on a timescale

$$\tau = \frac{\rho}{\sigma B^2} \tag{2}$$

known as the magnetic damping time [11].

Several key dimensionless groups can be derived from these equations. As with classical fluid dynamics, the Reynolds number

$$Re = \frac{UL}{\nu} \tag{3}$$

describes the ratio of inertial to viscous forces [12]. In the absence of MHD effects, flow typically becomes turbulent for high Re. The Hartmann number

$$Ha = BL\sqrt{\frac{\sigma}{\rho\nu}} \tag{4}$$

describes the strength of magnetic forces relative to viscous forces, where B is the magnetic flux density scale [4]. The magnetic Reynolds number is analogous to the Reynolds number

$$R_m = \mu\sigma UL \tag{5}$$

but instead describes the ratio between magnetic induction and magnetic diffusion [4]. If $R_m \ll 1$, magnetic induction is negligible and the induced magnetic field is negligible, such that $\boldsymbol{B}$ can be treated as a constant imposed field, $\boldsymbol{B} \approx \boldsymbol{B}_0$. This is typically the regime of fusion blanket applications [4]. In the $R_m \ll 1$ limit the inductionless (or quasi-static) approximation can be taken [13], resulting in a simplified set of equations

$$\rho\frac{\partial \boldsymbol{u}}{\partial t} + \rho(\boldsymbol{u}\cdot\nabla)\boldsymbol{u} - \rho\nu\nabla^2\boldsymbol{u} + \nabla p = \boldsymbol{J} \times \boldsymbol{B}_0 \tag{6a}$$

$$\nabla \cdot \boldsymbol{u} = 0 \tag{6b}$$

$$\nabla^2\phi = \nabla \cdot (\boldsymbol{u} \times \boldsymbol{B}_0) \tag{6c}$$

$$\boldsymbol{J} = \sigma(-\nabla\phi + \boldsymbol{u} \times \boldsymbol{B}_0) \tag{6d}$$

where ϕ is the scalar electric potential. These equations are less strongly coupled due to the lack of feedback in the magnetic field, with the Poission equation $(6c)$ being relatively simple to solve. The induced current $\boldsymbol{J}$ can be eliminated by substituting $(6d)$ into the Lorentz force term of $(6a)$.

The Hartmann and Reynolds numbers can be combined to form the Stuart number (or magnetic interaction parameter)

$$N = \frac{Ha^2}{Re} = \frac{\sigma B^2 L}{\rho U} \tag{7}$$

which provides information about the turbulent behaviour of the fluid [14]. In the inductionless approximation, turbulent behaviour is classical for $N \ll 1$, becomes anisotropic and intermittent for $N \sim 1$ with turbulent structures stretched across the magnetic field lines, and becomes fully laminar with complex structure for $N \gg 1$. This work considers only the case of $N \ll 1$ with low Re, such that flow is laminar.

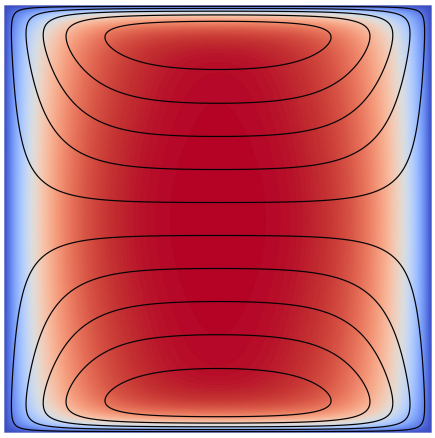

Figure 1. Shercliff flow for $Ha = 20$. The colour map shows velocity, while the contours show the magnetic field and also correspond to induced current field lines.

While liquid-metal flows are often complex, analytic solutions have been obtained for some simpler cases of laminar flow in homogeneous magnetic fields. The solution for laminar conducting channel flow between two parallel walls was obtained by Hartmann [15]. This was later extended to the cases of laminar flow in a square duct, with Shercliff deriving the solution for the case of perfectly insulating walls [16], and Hunt studying the more general case of mixed wall conductivities including arbitrary wall conductivity [17]. In all these cases the magnetic field is applied perpendicular to the flow. Figure 1 shows the cross-section of the Shercliff solution. The opposing pair of walls perpendicular to $\boldsymbol{B}$ are known as the Hartmann walls and the remaining walls parallel to $\boldsymbol{B}$ are the side walls. The core flow is seen to be inviscid. Velocity varies rapidly in the Hartmann layers (at the Hartmann walls) and the side layers (at the side walls). The

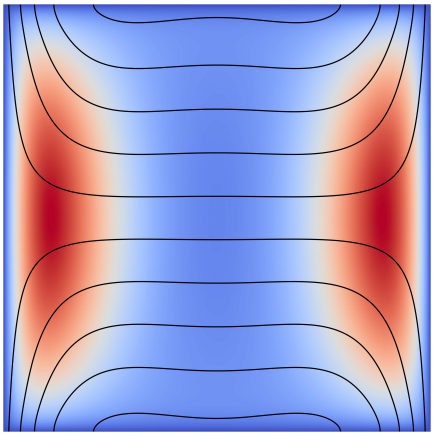

Figure 2. Hunt flow for $Ha = 20$. The colour map shows velocity, while the contours show the magnetic field and also correspond to induced current field lines.

Hartmann layer thickness δ varies strongly with Ha as $\delta \propto Ha^{-1}$, while the side layers have a weaker dependence of $\delta \propto Ha^{-\frac{1}{2}}$ [13]. Similar behaviour is observed for the Hunt case, shown in figure 2 for the subcase of perfectly conducting Hartmann walls and perfectly insulating side walls. Here instead velocity is concentrated into jets along the side wall. As Ha increases, the velocity in the core region drops asymptotically to zero and the Hartmann and side layers decrease in thickness as with Shercliff flow, while reverse flow jets can form between the side wall jet and core regions.

A key challenge in simulating liquid-metal MHD flows is resolving the Hartmann and side layers. This becomes a significant challenge as Ha approaches $Ha \sim 10^4$, as is common in fusion [4, 18], as the Hartmann layers in particular become vanishingly small.

2.2. Parallel scaling

A critical aspect of simulations targeting the exascale is parallel scaling. This can be considered in terms of two key properties. Strong scaling is the relationship between the time required to perform a fixed amount of work and the amount of computational resources used. Weak scaling in contrast is the relationship between the time required for an increasing amount of work with proportionally increasing computational resources. To quantify these scaling relations, first consider the rate R at which a computer does work

$$R = \frac{W}{t} \tag{8}$$

where W is a measure of the amount of work done by the computer (such as the total number of degrees of freedom (DOFs)) and t is the computation time [19]. As computational resources are increased, the relative speedup S_N can then be considered

$$S_N = \frac{R_N}{R_{\text{ref}}} = \frac{W_N}{W_{\text{ref}}} \frac{t_{\text{ref}}}{t_N} \tag{9}$$

where $R_{\text{ref}} = \frac{W_{\text{ref}}}{t_{\text{ref}}}$ is the rate at which work W_{ref} is conducted on a base number of cores N_{ref} in time t_{ref}, and $R_N = \frac{W_N}{t_N}$ is the rate at which work W_N is conducted on N cores in time t_N, where $N > N_{\text{ref}}$.

For strong scaling work is constant, $W_N = W_{\text{ref}}$, so the strong scaling speedup is

$$S_N^{\text{strong}} = \frac{t_{\text{ref}}}{t_N} \tag{10}$$

while for weak scaling the work increases proportionally with computational resources, $\frac{W_N}{N} = \frac{W_{\text{ref}}}{N_{\text{ref}}}$, giving the weak scaling speedup as

$$S_N^{\text{weak}} = \frac{N}{N_{\text{ref}}} \frac{t_{\text{ref}}}{t_N} \tag{11}$$

which is sometimes referred to as scaled speedup [20].

Parallel efficiency η_N can be defined as the ratio of the total resource use time per amount of work for on N_{ref} cores to that on N cores,

$$\eta_N = \frac{t_{\text{ref}} N_{\text{ref}} / W_{\text{ref}}}{t_N N / W_N} \tag{12}$$

which for strong scaling gives $\eta_N = \frac{t_{\text{ref}} N_{\text{ref}}}{t_N N}$, while for weak scaling $\eta_N = \frac{t_{\text{ref}}}{t_N}$. In the case of strong scaling, a threshold number of cells per core can be considered, below which increasing the number of resources is inefficient. If n is the number of cells per core, the threshold value of n below which efficiency drops below 80% is defined as $n_{0.8}$.

Ideal scaling is the case in which $\eta_N = 1 \forall N$. For strong scaling this would mean $S_N^{\text{strong}} = N/N_{\text{ref}}$, however in reality this is typically not achievable for most codes due to a finite fraction of the code running in serial, with S_N^{strong} asymptotically approaching a maximum theoretical speedup given by Amdahl's law [21, 20]. For weak scaling, the ideal case (in which scaled speedup $S_N^{\text{weak}} = N/N_{\text{ref}}$) is often unattainable as explained by Gustafson's law [20], which predicts a linear relationship $S_N^{\text{weak}} \propto N/N_{\text{ref}}$ with a gradient less than 1 due to a non-zero serial fraction of a given code.

3. Liquid-Metal MHD in OpenFOAM

OpenFOAM is a freely distributed open-source finite volume method (FVM) software package, primarily designed for computational fluid dynamics (CFD) [22]. LM-MHD capability for OpenFOAM has been developed within OpenFOAM itself (the `mhdFoam` solver) as well as through more advanced closed-source academic solvers implemented in OpenFOAM, such as the work of Mistrangelo and Bühler [23]. While the usefulness of closed-source codes is limited to those

with access, there are some freely available open-source OpenFOAM solvers that may extend similar capabilities to the wider community.

3.1. Open-source OpenFOAM MHD solvers

The `mhdFoam` solver, distributed with OpenFOAM, solves a variation of the full induction formulation (1) for $\boldsymbol{u}$, p and $\boldsymbol{B}$ for incompressible conducting fluids. To ensure the magnetic face flux field is divergence free in accordance with Maxwell's equations, a fictitious magnetic flux pressure p_B is introduced. The pressure implicit with splitting of operators (PISO) algorithm [24] is used for the coupling between pressure and velocity to effectively apply $\nabla\cdot\boldsymbol{u} = 0$, and a comparable (but simplified) method referred to as the B-PISO loop is applied by analogy for $\nabla \cdot \boldsymbol{B} = 0$, with the magnetic induction equation taking the place of the momentum predictor [25].

An inductionless MHD solver has been implemented in the open-source `epotFoam` solver presented by Tassone [25] based on the works of Dousset [26] and Mas de les Valls [27], solving equations based on the electric potential inductionless formulation (6) for $\boldsymbol{u}$, p, and ϕ, while $\boldsymbol{B}$ is assumed to be unchanged by the flow. Again a PISO loop is used for pressure-velocity coupling, however the fluid and electromagnetic solves are coupled only by the electric potential solve consuming the velocity $\boldsymbol{u}$ computed earlier in the timestep and outputting the Lorentz force to be input into the momentum equation solve in the next timestep [25].

For the studies presented here, OpenFOAM 9 from the OpenFOAM Foundation was used for both `mhdFoam` and `epotFoam`. This choice was made due to this being the latest version of OpenFOAM with which `epotFoam` was found to be compatible.

3.2. Scalability methods

To assess the parallel scaling properties of the `mhdFoam` and `epotFoam` solvers, the laminar Shercliff flow case was used as a benchmark [16]. For this, a 20 m $\times$ 2 m $\times$ 2 m square duct centred on $(10, 0, 0)$ was used, with flow along the x-axis and a magnetic field $\boldsymbol{B}_0 = (0, 20, 0)$ T was imposed parallel to the side walls (as opposed to the Hartmann walls) of the domain as defined in section 2.1. All constant properties (density, viscosity, permeability and conductivity) were set to unity, such that $Ha = |\boldsymbol{B}_0| = 20$. The inlet at $x = 0$ m was set with uniform inlet velocity $\boldsymbol{u}_{\text{in}} = (1, 0, 0)\,\text{m s}^{-1}$ with a `fixedValue` boundary condition (BC), resulting in laminar flow with $Re \sim 2$. The walls were set with `noSlip` conditions implementing $\boldsymbol{u} = 0$ and the outlet at $x = 20$ m was set as a `zeroGradient` velocity condition imposing zero normal gradient in the velocity over the boundary. Pressure was set to $p = 0$ Pa at the outlet using a `fixedValue` BC, with `zeroGradient` conditions on all other boundaries. In the `mhdFoam` case, the magnetic field was imposed as a constant vector `fixedValue` BC on the walls with `zeroGradient` conditions on the inlet and outlet, and the fictitious magnetic flux pressure p_B was set to zero at the inlet and outlet using `fixedValue` BCs with `zeroGradient` conditions on the walls. For `epotFoam`, instead the magnetic field was set as a constant property with `zeroGradient` conditions for electric potential ϕ on all boundaries. These electromagnetic BCs represent perfectly insulating walls. For `mhdFoam`, the magnetic field was set with an initial condition equal to the imposed field $\boldsymbol{B}_0 = (0, 20, 0)$ T, while all other variables ($\boldsymbol{u}$, p, p_B) were set to zero. For `epotFoam`, all variables ($\boldsymbol{u}$, p, ϕ) were set with initial conditions of zero.

The simulations were run in a transient solve despite the benchmark case being a steady laminar solution, due to the lack of steady-state support from these solvers. The solvers were run for 2 s simulation time to allow pressure to stabilise through the domain and allow the system to reach steady-state. Euler timestepping with second-order Gaussian integration for spatial gradients was used (`Gauss linear`), with linear interpolation used to obtain cell-surface values from cell-centred values. Laplacian schemes used the `Gauss linear corrected` setting, for unbounded second order conservative Laplacian terms. The simulation was therefore first order in time and second order in space. Orthogonal surface-normal gradient schemes were used due to the regularity of the hexahedral mesh used. For both `mhdFoam` and `epotFoam`, the velocity field equation was solved for using the symmetric Gauss-Seidel smoother method. In `mhdFoam`, the same solver was used for the magnetic flux density. With both `mhdFoam` and `epotFoam`, the pressure field was solved using the preconditioned conjugate gradient (PCG) solver with the simplified diagonal-based incomplete Cholesky (DIC) preconditioner, using 3 corrector iterations without non-orthogonal correctors in the PISO loop as well as in the B-PISO loop for `mhdFoam`. Similarly, for the electric potential solve in `epotFoam` the PCG solver was used with DIC, but without corrector iterations. PCG and DIC were also used for the p_B solve in `mhdFoam`. A `maxIter` limit of 2000 iterations was set for each solver, which was never reached.

In all cases for strong and weak scaling, the mesh was generated using OpenFOAM's `blockMesh` utility, generating block-structured hexahedral meshes. The domain was decomposed for parallelisation using the Scotch algorithm [28] distributed with OpenFOAM, using default settings. The Scotch algorithm features automated decomposition strategies such

as minimising the number of processor boundaries. Strong scaling was studied using 3 cases of the base problem at different resolutions, with the timestep adjusted according to the Courant-Friedrichs-Lewy (CFL) condition [12], as detailed in table 1. Mesh grading was used in the 80k and 640k cell cases to increase resolution smoothly between the centre of the domain and each wall. This was achieved by setting the expansion ratio E, defined in OpenFOAM as the cell size ratio between the first and last cell in a direction, with the individual cell sizes between those cells determined by a geometric progression. For each case a single grading ratio $E_{y,z} = E_y = E_z$ (specified in table 1) was set in both the y- and z-directions, while the resolution in the flow direction was uniform. A cross-section of the mesh for the 80k cell case is shown in figure 3. The 80k and 640k cell cases used $N_{\text{ref}} = 1$ core. The highest resolution 10 million cell case used a full node for the reference case, $N_{\text{ref}} = 76$ cores, due to the expected wall times exceeding limits on the HPC cluster used for $N_{\text{ref}} < 76$ at this resolution.

Table 1. Spatial and temporal discretisation in the strong scaling cases. Number of cores in the reference cases are also stated.

	Cells			Grading		
Case	x	y	z	$E_{y,z}$	$\Delta t(s)$	N_{ref}
80k	100	40	20	10	0.005	1
640k	200	80	40	5	0.0025	1
10M	500	200	100	1	0.0008	76

To study weak scaling, the reference case was set up with $(50, 20, 10)$ cells run on a single core ($N_{\text{ref}} = 1$), with a timestep of 0.008 s. Constant $n = 10000 \pm 1\%$ cells per core was maintained as N was increased by increasing the number of cells in each direction from the reference case by a factor of $\sqrt[3]{N/N_{\text{ref}}}$, manually adjusting the resolution by up to 3 cells in each direction to minimise deviation from $n = 10000$ cells per core. The timestep was decreased by the same factor in accordance with the CFL condition [12]. Because mesh grading would not be needed for the higher resolution cases, it was not used for any case in the weak scaling study in order to maintain mesh uniformity as a control variable; accuracy was therefore expected to improve with increasing number of cells (and so increasing number of cores) as the boundary layers became better resolved. The root-mean-square error (RMSE) in velocity $u_x = \boldsymbol{u} \cdot \hat{\boldsymbol{x}}$ and error in pressure drop K were also calculated based on the analytic Shercliff solution [16], with the cross-sectional velocity profile taken at $x = 13\,\text{m}$ and pressure drop measured as the average pressure gradient over the region $12.5 \leq x \leq 13.5\,\text{m}$. The RMSE was normalised by dividing by the mean absolute

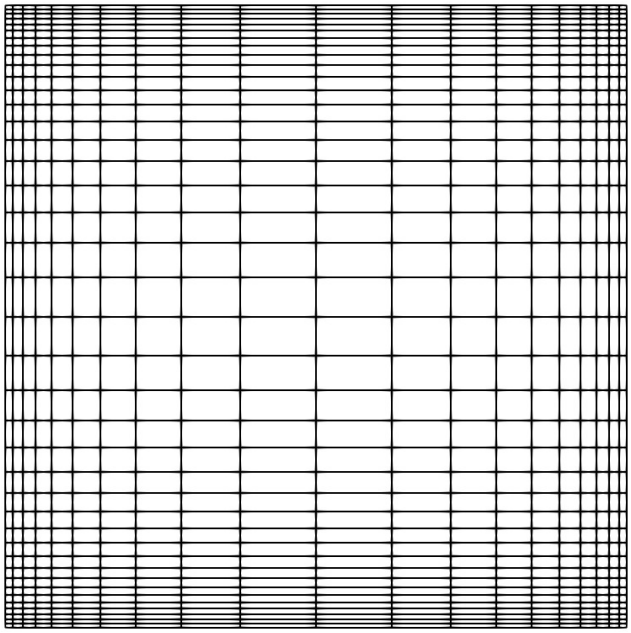

Figure 3. Transverse cross-section of the mesh for the 80k cell strong scaling case, with 40 cells in y and 20 cells in z with expansion ratio $E_{y,z} = 10$ in both directions.

analytic value of u_x over the slice, while pressure drop error was normalised by the analytic pressure drop. For the analytic solution, the Hunt solution [17] was used in the exponential form described by Ni et al. [29] with 200 Fourier iterations, which simplifies to the Shercliff solution in the case of perfectly insulating Hartmann and side walls. This form of the solution was used due to higher accuracy in numerically computing the exponential terms, rather than computing the hyperbolic functions of the original solutions.

Simulations were run by creating and meshing the geometry with `blockMesh`, decomposing it with `decomposePar` if $N > 1$, running the simulation with either `mhdFoam` or `epotFoam` and finally reconstructing the full domain with `reconstructPar` if $N > 1$. To calculate computation times, only the run-time of the simulation itself was measured, with the meshing, decomposition and recombination times ignored. Additionally, the runtime of the the first timestep was ignored due to additional processes during simulation initialisation, with the time calculated as the mean time per timestep averaged over the next 20 timesteps. This eliminated the dependence on the timestep, as the overall simulation time was expected to increase with decreasing timestep in the weak scaling case, such that the scaling assessment was based only on work per timestep. Each strong scaling or weak scaling case with each solver was repeated 3 times, with the results averaged. In the weak scaling case, this process was also performed for the total number of iterations per timestep for all the field solves in

order to understand the behaviour of the problem with increasing resolution.

Scaling behaviour was studied using the Cambridge Service for Data Driven Discovery (CSD3) HPC cluster's Ice Lake nodes, each consisting of 2 38-core Intel Xeon Platinum 8368Q CPUs for a total of 76 cores, with 256 GiB RAM per node and connected via Mellanox HDR200 Infiniband. For runs using fewer than 76 cores, a single node was reserved, while runs using more cores increased resources by adding full nodes, up to a maximum of 29 nodes ($N = 2204$). It should be noted that OpenFOAM was found to be simple to build on CSD3, indicating that it should be trivial to port to other HPC systems.

3.3. Scalability results and discussion

The strong scaling profiles for the 80k and 640k cases are shown in figure 4. It can be seen that `epotFoam` was faster per timestep, due to the simpler system of equations. The scaling profiles observed are typical of CFD codes, with parallel efficiency diminishing as the number of cores increased due to the increase in memory communication between the domain partitions becoming dominant. As the resolution of the problem increased, $n_{0.8}$ increased, from $n_{0.8} \sim 7.0 \times 10^3$ cells per core for the 80k cell case to $n_{0.8} \sim 2.5 \times 10^4$ for the 640k cell case. In the higher resolution 10M cell case shown in figure 5, it can be seen that scaling wass initially super-linear, demonstrating very good strong scaling for $n > 10^4$. While it can be seen that $n_{0.8} \sim 6 \times 10^3$ cells per core in this case, this cannot be directly compared to the lower resolution strong scaling profiles due to the higher value of N_{ref}.

Results of the weak scaling study are shown in figure 6. While Gustafson's law predicts a linear relationship between scaled speedup S_N^{weak} and N/N_{ref}, a non-linear relationship was instead found with the rate of increase of scaled speedup dropping off as N increased, resulting in poor weak scaling efficiency as seen in figure 6(c) This was found to be due to the number of iterations of each solver increasing as the resolution increased, as shown in figure 6(b). As a result, the amount of work in each timestep increased more than the intended change from resolution increasing proportionally with resources, slowing down the solution. This may have been due to the nature of the sparse matrix problem involved in the solve, and as such a different choice of preconditioners and solvers may have reduced this behaviour. A full study of the preconditioners and matrix problem solvers available in OpenFOAM may present a better selection for this type of MHD problem, with the potential for improved weak scaling, however a preconditioner study was beyond the scope of this study.

It can be seen from figure 6(d) that as resolution increased, so too did the accuracy of the simulation. At lower resolution, `epotFoam` was found to be unstable and failed to converge on the solution, indicating that `epotFoam` particularly struggled when the Hartmann and side layers were insufficiently resolved (with only 1 to 2 cell centres in the Hartmann layers in the $N = 1$ and $N = 2$ cases). This was less of an issue for `mhdFoam`, for which the relative errors steadily improved as resolution increased. However, the errors became large in both cases for the $N = 1216$ core case, and the simulation failed due to numerical instability before reaching the 2 s endpoint of the simulation for the cases with higher resolution. In each of these high resolution failed cases, the Courant number was seen to grow before the simulation failed, despite the CFL condition being met in the problem setup.

3.4. Validation methods

To further validate these solvers, the Shercliff (perfectly insulating walls) and Hunt flow (perfectly insulating side walls with perfectly conducting Hartmann walls) cases (as described in section 2.1) were studied up to higher Hartmann numbers ($Ha = 20$, $Ha = 100$ and $Ha = 1000$).

The Shercliff case was set up using the same methods described in section 3.2 for both `mhdFoam` and `epotFoam`. The $Ha = 20$ cases were set up exactly as described for the the 80k cell case in table 1. To set up the $Ha = 100$ and $Ha = 1000$ cases, the magnitude of the magnetic field applied parallel to the side walls was increased to $|\boldsymbol{B}_0| = 100\,\text{T}$ and $|\boldsymbol{B}_0| = 1000\,\text{T}$ respectively, and the resolution and mesh grading ratios were increased as described in table 2. The higher Ha cases used different grading ratios in the y- and z-directions to capture the stronger dependence of the Hartmann layer thickness on Ha than that of the side layers, varying this along with the resolution in order to reduce the size of the cells at the Hartmann and side walls by the same factor as the expected reduction in Harmann and side layer thicknesses. Initially the timestep for the higher Ha cases was decreased by the same factor as the change in the size of the smallest cell in y (equal to the change in Hartmann number), based on the CFL condition. However, this was found to be insufficient, with stability instead requiring the timestep to be decreased by a factor closer to the increase in Ha^2, as shown in table 2. The $Ha = 20$ and $Ha = 100$ cases were run on 8 cores, while the $Ha = 1000$ cases were run on 76 cores due to the increased resolution. Due to the increased wall time requirements from the small timestep, the end time of the $Ha = 1000$ case was reduced to 0.02 s. To ensure the solution was fully developed, the inlet velocity in the $Ha = 1000$ case

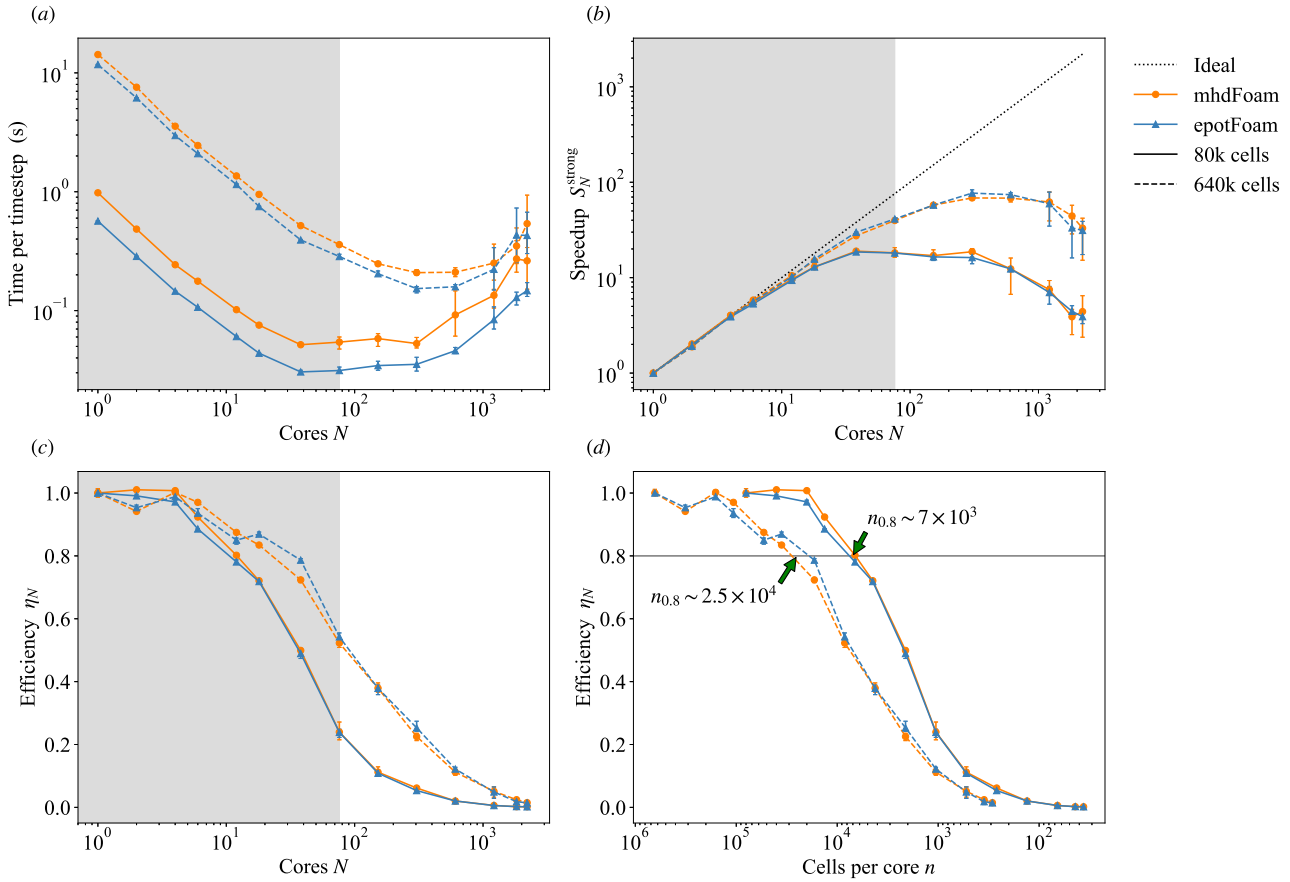

Figure 4. Strong scaling profiles for the 80k cell (solid lines) and 640k cell (dashed lines) cases, for `mhdFoam` (orange) and `epotFoam` (blue). Ideal strong scaling is shown in (*b*) by the dotted black line. The shaded region marks intra-node scaling. Approximate values of $n_{0.8}$ are marked in (*d*), with the horizontal line showing 80% efficiency.

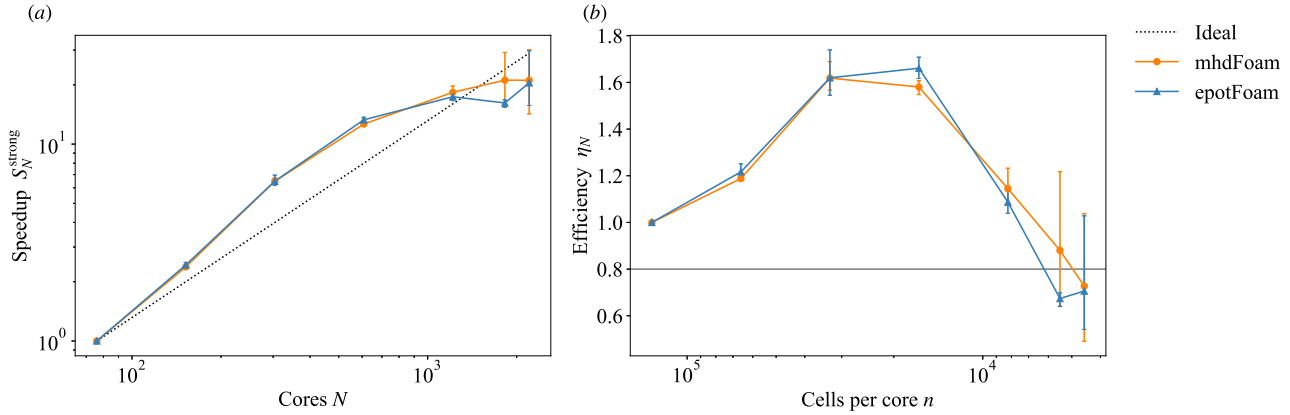

Figure 5. Inter-node strong scaling profiles for the 10M cell case, for `mhdFoam` (orange) and `epotFoam` (blue). Ideal strong scaling is shown in (*a*) by the dotted black line, where super-linear scaling can be seen between $N = N_{ref}$ and $N = 2N_{ref}$, and between $N = 2N_{ref}$ and $N = 4N_{ref}$.

was increased to $100\,\mathrm{m\,s^{-1}}$, with the flow remaining laminar at $Re \sim 200$.

To set up the Hunt flow cases, the Shercliff flow cases were used as a basis, modifying only the boundary conditions in order to reflect the perfectly conducting Hartmann walls. For `mhdFoam`, the magnetic field BC on the Hartmann walls at $y = 1$ and $y = -1$ was changed from the fixed vector BC to a `zeroGradient` BC. For `epotFoam`, this required instead changing the electric potential BC on the $y = 1$

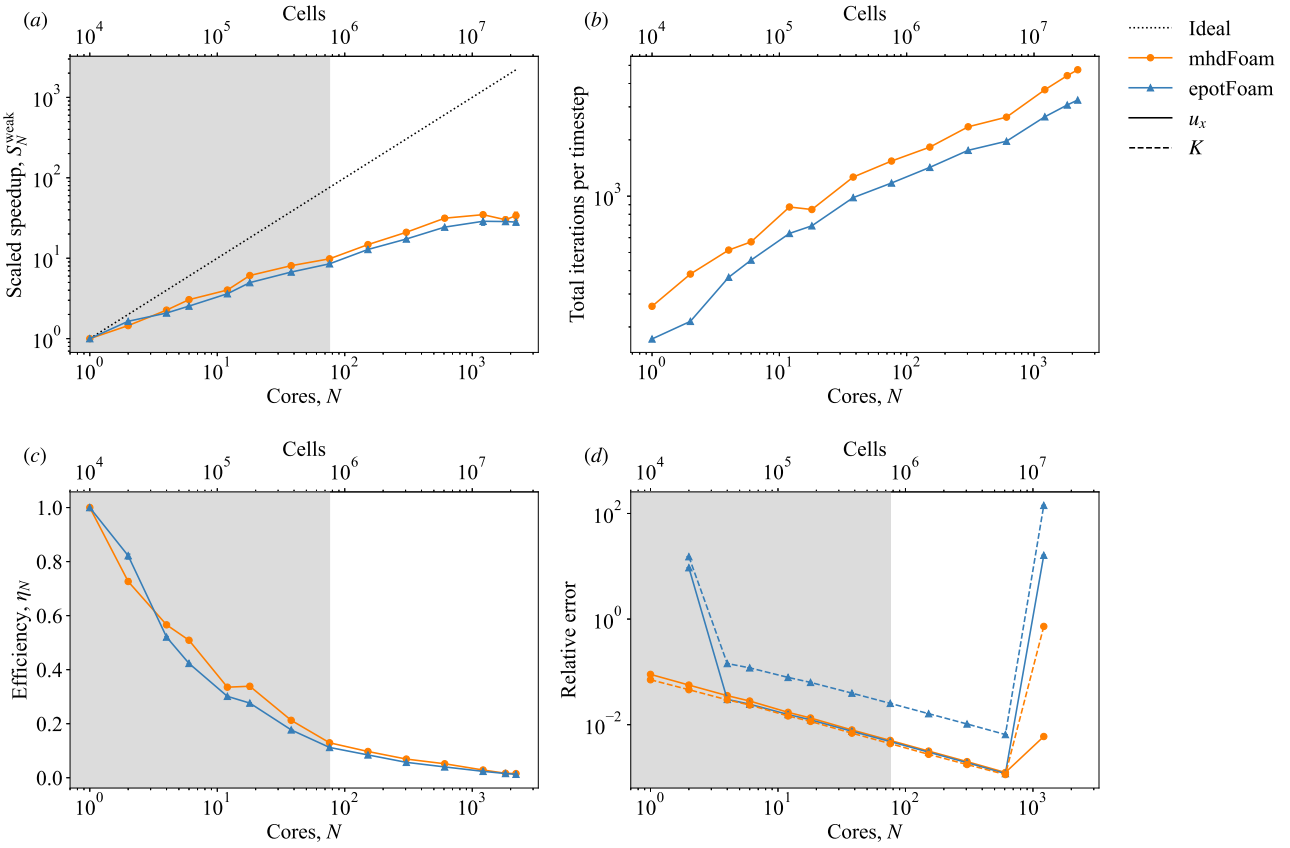

Figure 6. Weak scaling profiles for `mhdFoam` (orange) and `epotFoam` (blue). Ideal weak scaling is shown in (a) and (c) by the dotted black line. An increase in number of iterations is seen in (b), which plots the total number of iterations for all matrix solvers per timestep. RMSE in u_x and relative error in K are shown in (d) by solid and dashed lines respectively for each solver.

and $y = -1$ walls from `zeroGradient` to `fixedValue`, setting $\phi = 0$ there.

The RMSE in the velocity field and error in pressure drop at $x = 13\,\mathrm{m}$ compared to the analytic solution were calculated for each case, as described in section 3.2.

Table 2. Spatial and temporal discretisation in the $Ha = 20$, 100 and 1000 Shercliff and Hunt flow validation cases.

	Cells			Grading			
Ha	x	y	z	E_y	E_z	$\Delta t_{\mathrm{CFL}}(s)$	$\Delta t_{\mathrm{stable}}(s)$
20	100	40	20	10	10	5×10^{-3}	5×10^{-3}
100	100	60	30	50	20	1×10^{-3}	2×10^{-4}
1000	300	200	50	200	50	1×10^{-4}	1×10^{-6}

3.5. Validation results

Figure 7 shows the `mhdFoam` and `epotFoam` velocity profiles for Shercliff and Hunt flow compared to the analytic solutions, while tables 3 and 4 give the relative errors for the Shercliff and Hunt cases respectively. In most cases, both solvers were capable of accurately resolving the problem, with `epotFoam` typically calculating u_x more accurately than `mhdFoam`, but with an order of magnitude higher error in the pressure drop K; this is consistent with the results observed in figure 6, showing that this difference persisted over a wide range in both Hartmann number and resolution. For both solvers, u_x RMSE was notably higher in the Hunt flow problem. Discrepancies in $u_x(y, z = 0)$ can be seen in figure 7 near the core of the flow.

Table 3. Relative errors in cross-sectional velocity field u_x and pressure drop K at $x = 13\,\mathrm{m}$ for both `mhdFoam` and `epotFoam` for the Shercliff flow case with varying Hartmann number.

	`mhdFoam`		`epotFoam`	
Ha	u_x RMSE	K error	u_x RMSE	K error
20	0.916%	0.400%	0.792%	1.60%
100	0.567%	0.108%	0.394%	1.15%
1000	0.229%	0.0217%	0.603%	0.642%

However, `mhdFoam` was found to fail due to numerical instability in the $Ha = 1000$ Hunt flow problem after many timesteps (progressing to

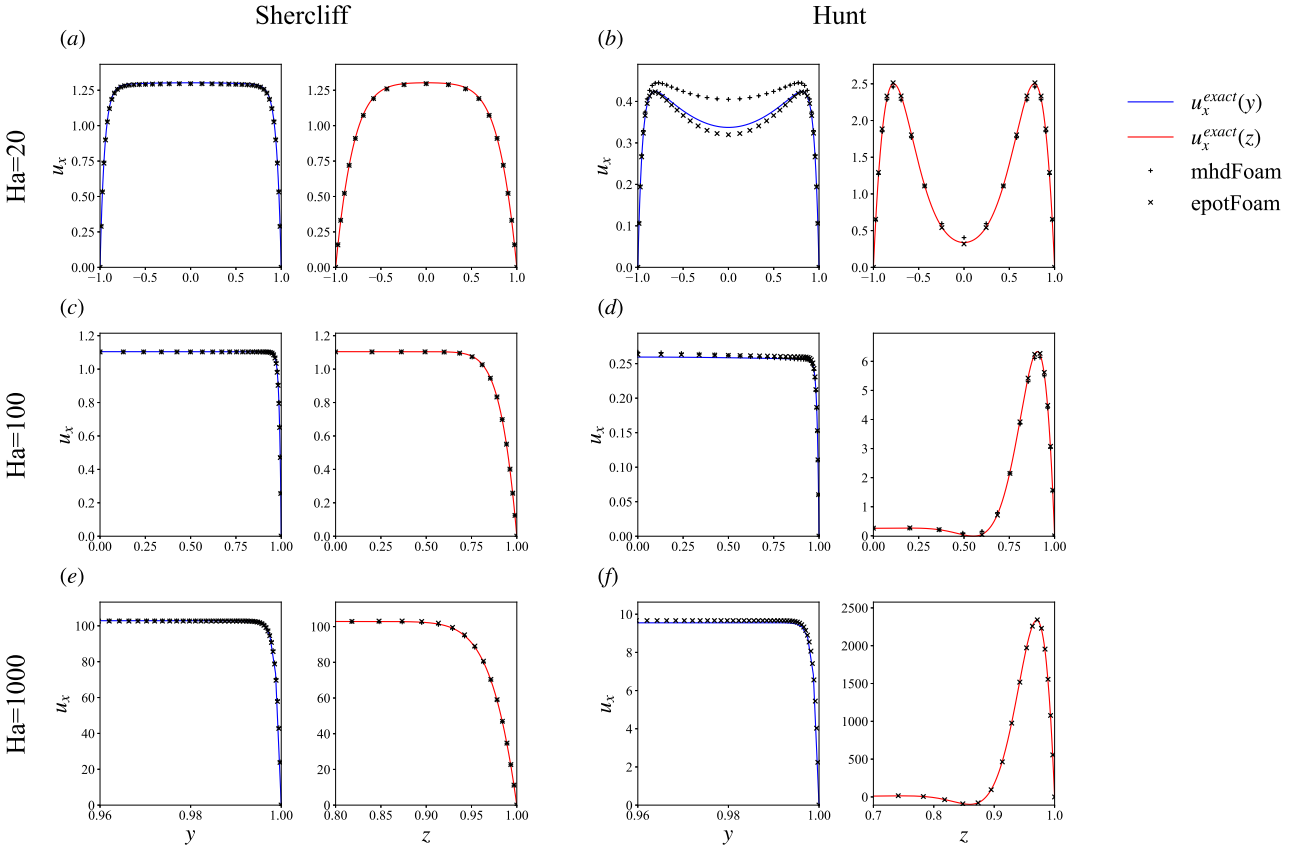

Figure 7. Comparison of velocity profiles computed with `mhdFoam` and `epotFoam` to the analytic solution for the Shercliff and Hunt problems. Plots of $u_x(y, z = 0)$ and $u_x(y = 0, z)$, measured at $x = 13\,\text{m}$ are shown in the left and right parts of each subplot respectively. For $Ha = 20$, shown in (a) and (b), the full slices are shown. For $Ha = 100$, shown in (c) and (d), and for $Ha = 1000$, shown in (e) and (f), the domain shown is restricted to the narrow boundary layers on one side of the symmetrical domain to show the ability of the solvers to simulate the strong velocity gradients.

Table 4. Relative errors in cross-sectional velocity field u_x and pressure drop K at $x = 13\,\text{m}$ for both `mhdFoam` and `epotFoam` for the Hunt flow case with varying Hartmann number.

	`mhdFoam`		`epotFoam`	
Ha	u_x RMSE	K error	u_x RMSE	K error
20	2.49%	0.211%	2.29%	1.50%
100	3.20%	0.725%	2.41%	1.18%
1000	unstable		2.32%	0.835%

$t = 0.0016\,\text{s}$ before failing), even with the timestep decreased to 1×10^{-7} (reaching $t = 0.0028\,\text{s}$). The cause of this remains to be identified.

3.6. Validation discussion

Both solvers were largely capable of simulating both problems, indicating that OpenFOAM may be a promising candidate for further development of LM-MHD solvers. However, various issues with these solvers were identified. Firstly, the cause of the failure of the `mhdFoam` in the $Ha = 1000$ Hunt flow case is unknown to the authors, and requires further investigation. Additionally, the requirement of a reduced timestep as Hartmann number is increased poses an interesting challenge. This could be an additional stability requirement for LM-MHD, however the authors were unable to find much research on this topic in the literature. The dependence appeared to be consistent with $\Delta t \propto Ha^{-2}$, which could be related to the magnetic damping time $\tau = \frac{\rho}{\sigma|\boldsymbol{B}|^2}$ which is understood to be relevant in the low R_m limit [11]. An argument is made in [30] that the timestep should be adjusted due to the addition of the Lorentz force as a source term in the momentum equation. The result is the modification of the timestep required for stability from that calculated from the CFL condition to

$$\frac{1}{\Delta t} = \frac{1}{\Delta t_{\text{CFL}}} + \sigma|\boldsymbol{B}|^2 \tag{13}$$

however this is not dimensionally consistent, differing by a factor of density; instead replacing the second term on the right hand side with τ would resolve this inconsistency. A parameter sweep of density and conductivity, and identifying the maximum timestep

at which the simulations remain stable, would inform this hypothesis, as would studying the dependence of stability on other parameters. It should be noted that this could be a dependence specific to OpenFOAM, the chosen Euler timestepping scheme, FVM or this specific regime, or it could be a fundamental limitation of LM-MHD simulations. Regardless of the cause, this poses a challenge to fusion-relevant simulations, with high Hartmann number significantly increasing simulation resources both from the increased number of timesteps required for a given time *in silico* as well as the high resolution required to resolve the narrow Hartmann and side layers, with the former imposing a limitation that cannot be countered easily without parallel-in-time methods.

4. Initial implementation of a liquid-metal MHD solver in Proteus

Proteus is an application built on the MOOSE FEM framework [31], focusing on fluid dynamics and related domains for mulitphysics coupling, which can be found at `https://github.com/aurora-multiphysics/proteus`. This work aimed to introduce a simple steady state incompressible resistive MHD solver into Proteus as a proof-of-concept, in order to investigate the viability of this approach for further work in liquid-metal simulation. For this initial implementation, the inductionless approximation (6) was used. This decision was made to minimise the numerical complexity and to test the simplest case of incompressible resistive MHD, with the inductionless approximation being valid in many typical fusion-relevant problems.

4.1. Implementation

To implement this solver, the incompressible FEM part of the MOOSE Navier-Stokes module was used as a base [32]. This includes `Kernel` objects, which compute weak form terms in PDEs and their Jacobians, for most of the terms in the continuity and momentum equations, with the exception of the Lorentz force term. Automatic differentiation (AD) was used in all kernels in this implementation to calculate Jacobians, reducing the complexity of implementing new kernels as well as improving convergence by using an accurate Jacobian, at the cost of increased memory requirements [33]. The result was an inductionless resistive magnetohydrodynamic incompressible Navier-Stokes with automatic differentiation (IRMINSAD) solver.

The required additional kernels were identified by expressing each of the terms of (6) in weak form by rearranging each equation such that the right hand side is zero, multiplying by a test function ψ and integrating over the domain Ω. Terms with time derivatives were dropped in order to solve in steady state. Current density $\boldsymbol{J}$ was eliminated by substituting $(6d)$ into the momentum equation $(6a)$. In $(6a)$ the only additional term is the Lorentz force,

$$\begin{aligned} -\boldsymbol{J}\times\boldsymbol{B}_0 &\rightarrow \int_\Omega \psi\sigma(\nabla\phi-\boldsymbol{u}\times\boldsymbol{B}_0)\times\boldsymbol{B}_0 \\ &= \int_\Omega \psi\sigma\nabla\phi\times\boldsymbol{B}_0 - \int_\Omega \psi\sigma(\boldsymbol{u}\times\boldsymbol{B}_0)\times\boldsymbol{B}_0 \end{aligned} \tag{14}$$

which can alternatively be expressed as two separate kernels for the electrostatic and velocity-dependent terms respectively. The Poisson equation for electric potential converts to weak form as

$$\begin{aligned} &\int_\Omega(\nabla\psi)\cdot(\nabla\phi) - \int_{\partial\Omega}\psi(\nabla\phi)\cdot\hat{\boldsymbol{n}} \\ &-\int_\Omega(\nabla\psi)\cdot(\boldsymbol{u}\times\boldsymbol{B}_0) + \int_{\partial\Omega}\psi(\boldsymbol{u}\times\boldsymbol{B}_0)\cdot\hat{\boldsymbol{n}} = 0 \end{aligned} \tag{15}$$

where the terms integrated over the boundaries of the domain $\partial\Omega$ are the BCs. The first term in (15) represents diffusion of electric potential and is implemented in MOOSE in the `ADDiffusion` kernel, while the third term represents the production of electric potential.

A first attempt at implementing these kernels in MOOSE, referred to here as the kernel method, utilised the MOOSE Navier-Stokes kernels `INSADMass` for the incompressibility constraint, and `INSADMomentumAdvection`, `INSADMomentumViscous` and `INSADMomentumPressure` for the momentum equation. These kernels require the `INSADMaterial Material` object. The `IRMINSADMomentumLorentzElectrostatic` and `IRMINSADMomentumLorentzFlow` kernels were added to Proteus and used to implement the separate electrostatic and velocity-dependent terms of (14) respectively. This implementation requires using second order velocity and first order pressure variables, due to the lack of stabilisation terms. The Poisson equation for electric potential (15) was implemented using the MOOSE `ADDiffusion` kernel for the $\nabla^2\phi$ term and a new `IRMINSADElectricPotentialProduction` kernel in Proteus for the $\nabla\cdot(\boldsymbol{u}\times\boldsymbol{B}_0)$ term.

An additional method took a similar approach, but instead more closely followed the design philosophy of the MOOSE incompressible Navier-Stokes implementation in which strong form residuals are calculated in a `Material` object, and then computed in weak form in `Kernel` objects. This allows the computed strong forms to be used elsewhere in the solve, such as in stabilisation terms. Adding the pressure-stabilized Petrov-Galerkin (PSPG) and streamline-upwind Petrov-Galerkin (SUPG) terms into the incompressibility constraint and momentum equation respectively provides stabilisation such that equal first order pressure and velocity variables can be used [32]. The `INSADMaterial` and `INSADTauMaterial`

MOOSE objects were duplicated in Proteus, and the Lorentz force strong form parts $\sigma\nabla\phi \times \boldsymbol{B}_0$ and $\sigma(\boldsymbol{u} \times \boldsymbol{B}_0) \times \boldsymbol{B}_0$ were added to create `IRMINSADMaterial` and `IRMINSADTauMaterial` (the latter of which inherits from the former) in which these components were added to the total strong residual of the momentum equation. This allowed these terms to be included in the residuals required for the PSPG and SUPG stabilisation terms. The kernel `IRMINSADMomentumLorentz` was added to add both Lorentz force parts to the momentum equation. This implementation is referred to as the material method. To set up a problem with this method, the same approach as the kernel method was used but instead with first order velocity and pressure variables, the `INSADMassPSPG` kernel added to the incompressibility constraint, `INSADMomentumSUPG` added to the momentum equation, the `IRMINSADMomentumLorentzElectrostatic` and `IRMINSADMomentumLorentzFlow` kernels replaced with `IRMINSADMomentumLorentz`, and with `IRMINSADTauMaterial` replacing `INSADMaterial`.

4.2. Methods

Both methods were tested against the Shercliff and Hunt problems for $Ha = 20$. The cases were set up to match the $Ha = 20$ validation cases described in section 3.4 in terms of physical properties and BCs, with the exception of the inlet which was set with a parabolic velocity profile with a mean velocity of $1\,\mathrm{m\,s^{-1}}$. As with those cases, a $100 \times 40 \times 20$ mesh was used, however mesh grading was limited to a grading ratio of 5 due to the solvers due to poor convergence with stronger grading (however, this was tested only with the `asm` preconditioner). The kernel method used 20-node hexahedra as required for the second order vector Lagrange velocity shape functions, with 8-node hexahedra used for the material method for first order velocity. Velocity was initialised with the parabolic flow profile used for the inlet. For both methods, pressure and electric potential were represented by first order Lagrange shape functions. The magnetic field was represented by a first order Lagrange `AuxVariable`, which could in principle reflect a non-uniform magnetic field or be replaced by a coupled variable, but in this case was simply calculated as uniform $\boldsymbol{B}_0 = (0, 20, 0)\,\mathrm{T}$. Each case was solved in steady state using Newton's method [33], using automatic scaling and limiting linear iterations to 100 and non-linear iterations to 1000. Each method was tested in the Shercliff and Hunt flow cases with several preconditioners: `asm`, `bjacobi`, `gamg`, `gasm`, `ilu` and `ksp`. Additionally, tests were conducted using 8 cores and 28 cores in order to gain a basic understanding of strong scaling, setting a time limit of 24 hours and 12 hours for 8 cores and 28 cores respectively. On 28 cores, both replicated and distributed mesh methods were tried, however this was not found to make a significant difference to the results. For this work, the CSD3 Cascade Lake nodes were used, each consisting of 56 CPU cores, with 192 GiB RAM per node and HDR Infiniband interconnect. Cascade Lake was used for simplicity due to incompatibilities found between certain modules required for MOOSE on the Ice Lake nodes, however this is expected to be a simple problem to fix in the future.

4.3. Results

Using the kernel method, the solution diverged for all preconditioners in the Shercliff case, except for with `ilu` for which the solve reached the time limit. This method was more successful in the Hunt case, for which `bjacobi` and `ksp` converged for both 8 and 28 cores. For this case `gasm` and `ilu` converged in the 28 core tests, however with 8 cores the solve diverged with `gasm` and ran out of time with `ilu`, indicating that with `gasm` further decomposing the problem aided convergence. The `ksp` preconditioner was found to converge fastest, solving the problem in 3.8 hours on 8 cores and 1.5 hours on 28 cores.

The material method was found to converge with a greater proportion of the preconditioners tested than the kernel method. For the Shercliff case, `asm` and `ilu` both converged for both 8 and 28 cores, while `bjacobi` and `gasm` converged for 8 cores only and all other preconditioners failed. The `asm` preconditioner converged fastest for this problem, solving in 13.2 min on 8 cores and 5.5 min on 28 cores. For Hunt flow, `asm` converged but more slowly, with `gamg` converging fastest in 13.4 min on 8 cores and 6 min on 28 cores. In this case all the other preconditioners were successful on 8 cores, while `bjacobi`, `gasm` and `ksp` failed on 28 cores.

For the preconditioners with which the simulations converged both with 8 cores and 28 cores, strong scaling was clearly demonstrated for both methods, with a 3.5 times increase in computational resources resulting in speedups of approximately 2.5.

Figure 8 shows a comparison of the Proteus material method solution for Shercliff flow to the analytic solution. Here significant deviation can be seen, with the the magnitude of the velocity profile in the Proteus solution substantially lower than that of the analytic solution. This suggests a failure to conserve mass as the mass flow rate at $x = 13\,\mathrm{m}$ in this simulation is clearly significantly lower than in the analytic solution. This is reflected in error calculations, where $\mathrm{RMSE}(u_x) = 14\%$, with a similarly high pressure drop error of $K_{\mathrm{error}} = 10\%$. The situation is similar for Hunt flow, as seen in figure 9, for which $\mathrm{RMSE}(u_x) = 18\%$ and $K_{\mathrm{error}} = 11\%$. This was

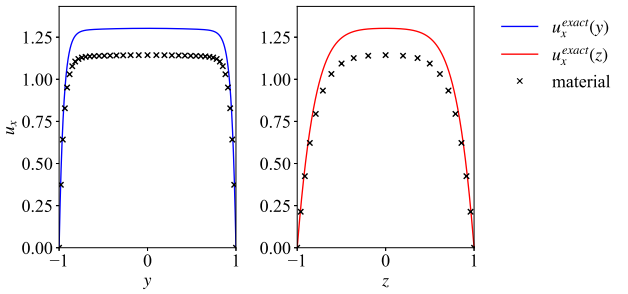

Figure 8. Proteus material method velocity profiles $u_x(y, z = 0)$ and $u_x(x = 13, y = 0, z)$ compared to the analytic solution for the Shercliff case at $Ha = 20$, measured at $x = 13\,\mathrm{m}$. The kernel method was found to not converge.

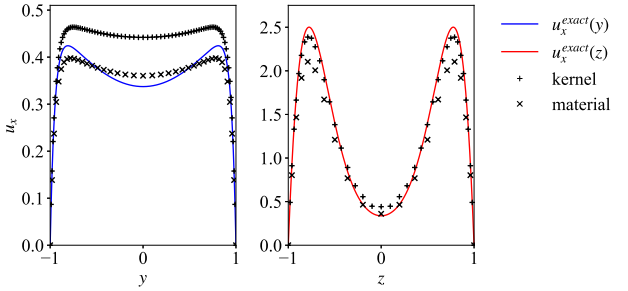

Figure 9. Proteus velocity profiles $u_x(y, z = 0)$ and $u_x(y = 0, z)$ compared to the analytic solution for the Hunt case at $Ha = 20$, measured at $x = 13\,\mathrm{m}$. Results for both the kernel and material methods are shown.

found to be the case regardless of whether a uniform velocity inlet or the parabolic inlet described in section 4.2 was used.

However, it can be seen in figure 9 that although convergence for the kernel method was slow, and was not achieved for the Shercliff case, it was found to be more accurate than the material method in the Hunt case, for which $\mathrm{RMSE}(u_x) = 4.5\%$ and $K_{\mathrm{error}} = 2.0\%$, approaching the accuracy found with `mhdFoam` and `epotFoam` (see table 4). The largest inaccuracies in velocity were seen at $z = 0\,\mathrm{m}$, where higher velocity was observed than in the analytic solution, and at the peaks of the side wall jets where the maximum velocity was lower.

These simulations were also found to take significantly longer to compute than the OpenFOAM simulations. While on 8 cores the Proteus Shercliff flow case using the material method (434805 DOFs) and the `asm` preconditioner took 13.2 min to converge to a residual of 1.3×10^{-6}, in the equivalent `epotFoam` case (80,000 cells using mesh grading with a ratio of $E = 10$, for approximately 4×10^5 DOFs) took only 10.5 s in real time to reach the 2 s simulation time end point on the same resources, around 75 times faster. It was found that when solving the Shercliff problem with the material method and `asm` preconditioner, increasing the limit on number of linear iterations per nonlinear iteration from 100 to 1000 significantly reduced the number of nonlinear iterations required, substantially accelerating the solve, however the solution was still inaccurate and the solution time was still more than an order of magnitude greater than for the OpenFOAM solvers.

4.4. Discussion

The Proteus LM-MHD implementation was found to have several disadvantages compared to the OpenFOAM solvers, including less accurate solutions, longer solve times, and an inability to increase mesh grading beyond a ratio $E \sim 8$ with the `asm` preconditioner. The kernel method was found to be more accurate than the material method, which could be due to either the increased number of degrees of freedom from the higher element order for velocity or potential inaccuracies in the material method stabilisation terms. However, this implementation aimed only to test the concept of using MOOSE for MHD problems. Future work will aim to improve the implementation, as well as consider alternative routes for introducing MHD features into MOOSE such as coupling in external solvers, in order to enable multiphysics simulations incorporating LM-MHD. Both methods implemented in Proteus were found to show reasonable strong scaling, as expected based on [8], however further insights could be gained by performing a full scaling analysis. Relaxing the limit on number of linear iterations was found to accelerate convergence for the material method in the Shercliff flow case with the `asm` preconditioner, so investigating this further for a wider range of problems and preconditioners could provide a route to reducing simulation wall times. Higher mesh grading ratios were only tested with the `asm` preconditioner, so high mesh grading should be tested with other preconditioners. Additionally, this work only considered the approach of solving for all fields in a single matrix, however splitting the problem by fields would allow for different preconditioners to be selected for each field, which could improve convergence.

5. Conclusions

This work has identified OpenFOAM to be a useful package for portable, scalable, open-source CFD, and which could be a strong candidate for developing open-source LM-MHD capability. The `mhdFoam` and `epotFoam` solvers are good examples of the full induction and inductionless formulations respectively, with both accurately solving simple LM-MHD cases. Both solvers demonstrated good strong scaling, though $n_{0.8}$ was seen to increase with increasing resolution.

Room for improvement was observed in terms of weak scaling, though this could be approached by performing a preconditioner study to identify the optimal set of preconditioners for weak scaling. However, both solvers lack features that would be required for more practical applications, such as the ability to simulate walls of arbitrary conductivity or couple in thermal effects such as buoyancy and electromagnetic heating. Furthermore, this work only validated the solvers against simple problems with analytic solutions. More complex validation cases exist, such as those outlined in [34], which include turbulent behaviour, thermal effects, and extension to $Ha \sim 10^4$. Turbulent flow could be simulated in these solvers by DNS, however for complex multiphysics simulations it would likely be beneficial to reduce resolution requirements by using turbulence modelling methods such as LES [9]. Future work aims to explore more advanced open-source LM-MHD solvers, such as those developed by Blishchik [35], which incorporate electromagnetic coupling with the walls and dynamic Smagorinsky LES turbulence modelling in addition to other features. Additionally, it would be beneficial for some simple cases to be able to solve in steady state, however this is only a benefit if behaviour is steady which is not necessarily the case for practical fusion applications.

An initial implementation of LM-MHD was added to the Proteus MOOSE application and studied, in order to test this route. This was found to require significant improvements, with poor accuracy and convergence (particularly with high mesh grading ratios) becoming evident. However, this study has identified these areas which can be improved on in future work, and may inform future research efforts on FEM LM-MHD implementations.

Overall, it was found that open-source LM-MHD solvers have the capability to become viable scalable, portable solvers for inclusion into multiphysics packages for fusion research. OpenFOAM is clearly a promising candidate, with significant existing research in the literature and some more advanced solvers, provided issues with weak scaling can be overcome. However, it is clear that this area of research has fallen behind the closed-source development of prominent solvers in the field such as those demonstrated in [5], and as such there is significant room for further open-source development in this field.

Acknowledgements

RWE would like to thank Gerasimos Politis for many insightful discussions which helped to develop my knowledge of liquid-metal MHD and to refine this work.

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).

This work has been part-funded by STEP, a UKAEA programme to design and build a prototype fusion energy plant and a path to commercial fusion.

References

[1] Davis, Andrew, Dubas, Aleksander and Otin, Ruben 2020 *EPJ Web Conf.* **245** 09001 URL https://doi.org/10.1051/epjconf/202024509001

[2] Boccaccini L, Aiello G, Aubert J, Bachmann C, Barrett T, Del Nevo A, Demange D, Forest L, Hernandez F, Norajitra P, Porempovic G, Rapisarda D, Sardain P, Utili M and Vala L 2016 *Fusion Engineering and Design* **109-111** 1199–1206 ISSN 0920-3796 proceedings of the 12th International Symposium on Fusion Nuclear Technology-12 (ISFNT-12) URL https://www.sciencedirect.com/science/article/pii/S0920379615304427

[3] Smolentsev S 2021 *Fluids* **6** ISSN 2311-5521 URL https://www.mdpi.com/2311-5521/6/3/110

[4] Smolentsev S 2022 *Fusion Science and Technology* **0** 1–23 URL https://doi.org/10.1080/15361055.2022.2116905

[5] Smolentsev S, Rhodes T, Yan Y, Tassone A, Mistrangelo C, Bühler L and Urgorri F R 2020 *Fusion Science and Technology* **76** 653–669 URL https://doi.org/10.1080/15361055.2020.1751378

[6] Brooks H and Davis A 2022 *Plasma Physics and Controlled Fusion* **65** 024002 URL https://dx.doi.org/10.1088/1361-6587/aca998

[7] Gaston D R, Permann C J, Peterson J W, Slaughter A E, Andrš D, Wang Y, Short M P, Perez D M, Tonks M R, Ortensi J, Zou L and Martineau R C 2015 *Annals of Nuclear Energy* **84** 45–54

[8] Permann C J, Gaston D R, Andrš D, Carlsen R W, Kong F, Lindsay A D, Miller J M, Peterson J W, Slaughter A E, Stogner R H and Martineau R C 2020 *SoftwareX* **11** ISSN 2352-7110 URL https://doi.org/10.1016/j.softx.2020.100430

[9] Krasnov D, Zikanov O, Schumacher J and Boeck T 2008 *Physics of Fluids* **20** 095105 URL https://doi.org/10.1063/1.2975988

[10] Gerbeau J F, Le Bris C and Lelièvre T 2006 *Mathematical Methods for the Magnetohydrodynamics of Liquid Metals* (Oxford University Press) ISBN 9780198566656 URL https://doi.org/10.1093/acprof:oso/9780198566656.001.0001

[11] Davidson P A 2001 *An Introduction to Magnetohydrodynamics* Cambridge Texts in Applied Mathematics (Cambridge University Press)

[12] Greenshields C and Weller H 2022 *Notes on Computational Fluid Dynamics: General Principles* (Reading, UK: CFD Direct Ltd)

[13] Müller U and Bühler L 2001 *Magnetofluiddynamics in Channels and Containers* (Springer Berlin, Heidelberg) ISBN 978-3-540-41253-3

[14] Zikanov O and Thess A 1998 *Journal of Fluid Mechanics* **358** 299–333

[15] Hartmann J 1937 *Mathematisk-fysiske Meddelelser* **15**

[16] Shercliff J A 1953 *Mathematical Proceedings of the Cambridge Philosophical Society* **49** 136–144

[17] Hunt J C R 1965 *Journal of Fluid Mechanics* **21**(4) 577–590 ISSN 0022-1120 URL https://www.cambridge.org/core/product/identifier/S0022112065000344/type/journal_article

[18] Mistrangelo C and Bühler L 2017 *PAMM* **17** 115–118 URL https://onlinelibrary.wiley.com/doi/abs/10.1002/pamm.201710033

[19] Moreland K and Oldfield R 2015 Formal metrics for large-scale parallel performance *High Performance Computing* ed Kunkel J M and Ludwig T (Cham: Springer International Publishing) pp 488–496 ISBN 978-3-319-20119-1

[20] Gustafson J L 1988 *Commun. ACM* **31** 532–533 ISSN 0001-0782 URL https://doi.org/10.1145/42411.42415

[21] Amdahl G M 1967 Validity of the single processor approach to achieving large scale computing capabilities *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference* AFIPS '67 (Spring) (New York, NY, USA: Association for Computing Machinery) p 483–485 ISBN 9781450378956 URL https://doi.org/10.1145/1465482.1465560

[22] Weller H G, Tabor G, Jasak H and Fureby C 1998 *Computers in Physics* **12** 620–631 URL https://aip.scitation.org/doi/abs/10.1063/1.168744

[23] Mistrangelo C and Bühler L 2011 *Fusion Science and Technology* **60** 798–803 URL https://doi.org/10.13182/FST11-A12483

[24] Issa R 1986 *Journal of Computational Physics* **62** 40–65 ISSN 0021-9991 URL https://www.sciencedirect.com/science/article/pii/0021999186900999

[25] Tassone A 2016 Magnetic induction and electric potential solvers for incompressible MHD flows *Proceedings of CFD with OpenSource Software* ed Nilsson H

[26] Dousset V 2009 *Numerical simulations of MHD flows past obstacles in a duct under externally applied magnetic field* Ph.D. thesis Coventry University

[27] Mas de les Valls E 2011 *Development of a simulation tool for MHD flows under nuclear fusion conditions* Ph.D. thesis Universitat Politècnica de Catalunya

[28] Pellegrini F 2012 Scotch and PT-Scotch Graph Partitioning Software: An Overview *Combinatorial Scientific Computing* ed Uwe Naumann O S (Chapman and Hall/CRC) pp 373–406 URL https://hal.inria.fr/hal-00770422

[29] Ni M J, Munipalli R, Huang P, Morley N B and Abdou M A 2007 *Journal of Computational Physics* **227** 205–228 ISSN 0021-9991 URL https://www.sciencedirect.com/science/article/pii/S0021999107003269

[30] Khodak A, Titus P, Brown T and Klabacha J 2018 *Fusion Engineering and Design* **137** 124–129 ISSN 0920-3796 URL https://www.sciencedirect.com/science/article/pii/S0920379618305982

[31] Lindsay A D, Gaston D R, Permann C J, Miller J M, Andrš D, Slaughter A E, Kong F, Hansel J, Carlsen R W, Icenhour C, Harbour L, Giudicelli G L, Stogner R H, German P, Badger J, Biswas S, Chapuis L, Green C, Hales J, Hu T, Jiang W, Jung Y S, Matthews C, Miao Y, Novak A, Peterson J W, Prince Z M, Rovinelli A, Schunert S, Schwen D, Spencer B W, Veeraraghavan S, Recuero A, Yushu D, Wang Y, Wilkins A and Wong C 2022 *SoftwareX* **20** 101202 ISSN 2352-7110 URL https://www.sciencedirect.com/science/article/pii/S2352711022001200

[32] Peterson J W, Lindsay A D and Kong F 2018 *Advances in Engineering Software* **119** 68–92

[33] Lindsay A, Stogner R, Gaston D, Schwen D, Matthews C, Jiang W, Aagesen L K, Carlsen R, Kong F, Slaughter A *et al.* 2021 *Nuclear Technology* 1–18 URL https://doi.org/10.1080/00295450.2020.1838877

[34] Smolentsev S, Badia S, Bhattacharyay R, Bühler L, Chen L, Huang Q, Jin H G, Krasnov D, Lee D W, de les Valls E M, Mistrangelo C, Munipalli R, Ni M J, Pashkevich D, Patel A, Pulugundla G, Satyamurthy P, Snegirev A, Sviridov V, Swain P, Zhou T and Zikanov O 2015 *Fusion Engineering and Design* **100** 65–72 ISSN 0920-3796 URL https://www.sciencedirect.com/science/article/pii/S0920379614003263

[35] Blishchik A, van der Lans M and Kenjereš S 2021 *International Journal of Heat and Fluid Flow* **90** 108800 ISSN 0142-727X URL https://www.sciencedirect.com/science/article/pii/S0142727X21000308