PRediction Of Geospace Radiation Environment and Solar wind parameterS

Work Package 2
Propagation of the Solar Wind from the Sun to L1

Deliverable 2.1
Conversion of SWIFT to spherical geometry

T. Arber, K. Bennett

December 31st, 2015

This project has received funding from the *European Unions Horizon 2020 research and innovation programme* under grant agreement No 637302.

Document Change Record

Issue	Date	Author	Details
1.0	24 November	T. Arber	Initial skeleton
2.0	21 December	T. Arber	Final submission

Contents

Summary

Space weather forecasts require reliable knowledge of the IMF Bz field at L1 and this is required before they are measured there *in situ*. Codes developed at the University of Michigan will take magnetogram observations and use these to drive MHD simulations, time-accurate, out to around 30 solar radii. These coronal simulations can then be used to drive a fast, spherical geometry inner-heliospheric MHD code (SWIFT) to give predictions at L1. This report covers the decisions made, and tests performed, in developing this spherical geometry MHD code. This covers the conversion of the existing Cartesian code *Lare3d* to spherical geometry by a mixture of area-volume and finite difference techniques as well as detailed testing of the shock viscosity. This report will be sufficient for a computational MHD researcher, who has a copy of the freely available *Lare3d* code, to re-engineer SWIFT. The SWIFT code is now released under version control. The release version solves MHD in spherical geometry using edge-based shock viscosity.

1 Introduction

This report details the work done, and decisions made, in developing the spherical geometry version of the SWIFT Lagrangian-remap code. This is a crucial step towards predicting solar wind values at L1 based on GONG observations. The University of Michigan is developing the AWSoM code [8] to take GONG data and use this to drive a coronal simulation out to 20-30 solar radii. At this radius output from AWSoM will be coupled to SWIFT which then completes the simulation from 20-30 radii out to L1. The coupling of the codes will be the subject of D2.2 in month 20 of the project. The full simulation suite, from GONG data to L1, will then be documented in D2.3 at the end of the project. WP2 will be coupled to all other work packages providing forecasts of L1 variables for all of the other toolsets. This coupling is the aim of WP7. The final toolchain will forecast space weather using GONG observations of the solar magnetic field coupled to predictive simulations.

2 Overview of D2.1 within WP2 activity

The overall aim of WP2 is to predict solar wind properties at L1 based on GONG data. The AWSoM code will complete this task from the solar surface up to 20-30 solar radii. After this the SWIFT code will take output from AWSoM and simulate the MHD evolution out to L1. This deliverable reports on the design and implementation of the spherical geometry code SWIFT. Specifically how the existing Cartesian MHD code *Lare3d* [1] was converted to spherical geometry. This section of the report summaries the initial state of *Lare3d* and outlines the changes that were made to convert to spherical geometry. The details of these changes are then summarised in later sections.

In S.I. units the standard ideal MHD equations are

$$\begin{aligned}
\frac{\mathrm{D}\rho}{\mathrm{D}t} &= -\rho\nabla\cdot\mathbf{v} && (1)\\
\frac{\mathrm{D}\mathbf{v}}{Dt} &= \frac{1}{\rho\mu_0}(\nabla\times\mathbf{B})\times\mathbf{B} - \frac{1}{\rho}\nabla P && (2)\\
\frac{\mathrm{D}\mathbf{B}}{\mathrm{D}t} &= (\mathbf{B}\cdot\nabla)\mathbf{v} - \mathbf{B}(\nabla\cdot\mathbf{v}) && (3)\\
\frac{\mathrm{D}\epsilon}{\mathrm{D}t} &= -\frac{P}{\rho}\nabla\cdot\mathbf{v} && (4)
\end{aligned}$$

Where ϵ is the specific internal energy density and γ is the ratio of specific heats. Here $\mathrm{D}/\mathrm{D}t$ is the advective derivative and all other terms have their usual meaning. Definitions for converting between ϵ and the more familiar pressure and temperature are

$$\begin{aligned}
P &= \frac{\rho k_B T}{\mu_m}\\
\epsilon &= \frac{P}{\rho(\gamma-1)} = \frac{k_B T}{\mu_m(\gamma-1)}
\end{aligned}$$

where μ_m is the reduced mass, i.e. the average mass of all particles in the plasma. Hence $\mu_m = m_p$ for neutral hydrogen atoms (m_p is the proton mass) and $\mu_m = 0.5m_p$ for fully ionised hydrogen.

The *Lare3d* code solves the MHD equations by first taking a Lagrangian step, i.e. solving equations (1-4), and then conservatively remapping the variables back onto the original Cartesian mesh. The advantages of this approach are that all of the physics is in the fully 3-dimensional Lagrangian step with limiters required for accurate shock solutions, i.e. TVD limiters, confined to the purely geometric remap step. This means that additional physics such as separate electron and ion temperatures or thermal conduction are easy to implement in the computationally efficient 3D Lagrangian phase. This report focuses purely on the conversion of the core *Lare3d* to spherical geometry so does not discuss the two-temperature model or thermal conduction further although these will be needed later for an accurate solar wind model. The SWIFT code will solve for the full 2π azimuthal angle, for radii from 20 or 30 solar radii out to L1 and for poloidal angles ± 60 degrees from the equator.

The Lagrangian step in *Lare3d* is roughly 1000 lines of code and the remap steps are an additional 3000 lines. All of these were implemented specifically for a regular Cartesian grid and hence the majority of the code had to be converted to spherical geometry for SWIFT. There were three key high-level decisions for this conversion.

1. **Shock viscosity**: The accurate treatment of shocks in a Lagrangian step is best treated through a compatible shock viscosity [3, 1]. This ensure the correct entropy jump across shocks and conserves energy. When moving such schemes to multi-dimensions there are however several options. These include tensor mimetic viscosity [2], edge viscosities [5] and sub-zonal pressures [4]. *Lare3d* uses mimetic tensor viscosity but this was only thoroughly tested in Cartesian geometry and is computationally expensive. For SWIFT to provide forecasts of space weather it needs to be fast and so the choice of shock viscosity was crucial. It was determined that edge-viscosity was the best choice for symmetry and speed.

2. **Area-volume approach**: It is possible to convert a Cartesian finite-difference scheme to spherical geometry by hard coding the spherical version of the differential equations in finite difference form. This is the simplest approach but has several drawbacks. Firstly the code can then only work in spherical geometry so can only

be compared to other spherical codes. This reduces the robustness of the testing and debugging. It is often important to know, when adding new physics modules, that these work in the simpler Cartesian geometry first. To accommodate this it was decided to implement the changes to spherical geometry in area-volume formalism where possible (the remap step). Compiler flags change metrics allowing the code to run in Cartesian, cylindrical or spherical geometry without a computational overhead. This was more work than a simple finite difference swap but leads to a more flexible, testable and sustainable code.

3. **Nodal forces**: Moving to spherical geometry introduces geometric factors into the forces on grid nodal points in full area-volume schemes. These are relatively easy to handle in Euler's equations but the complications for MHD in a Lagrangian, moving control volume, while solved are computationally expensive. The forces, both pressure gradient and magnetic forces, are therefore handled as finite difference with metrics handling the geometry changes. This is the only component of SWIFT which is not area-volume based.

The details of the implementation, and test cases where appropriate, for each of these elements of the spherical geometry implementation are detailed below.

2.1 Choice of shock viscosity

Testing of the shock viscosities for SWIFT was performed in the *Odin* code. This is a 2-dimensional (r,z) or (x,y) geometry arbitrary Lagrangian Eulerian (ALE) code. This code has been extensively tested for shocks generated in inertial confinement fusion experiments. The *Odin* code can work with an arbitrary grid in either geometry and is far more flexible than either *Lare3d* or SWIFT. However this flexibility leads to a slower code. A single full Lagrangian-remap step in *Odin* can be three times slower than *Lare3d* or SWIFT. Nonetheless it is an ideal testbed for assessing shock viscosities in cylindrical and spherical geometry as it already had mimetic tensor and scalar viscosities implemented along with the option to use sub-zonal pressures. Edge viscosity was added to Odin which was then run in planar, cylindrical and spherical test cases. It was found that the mimetic tensor viscosity was the most robust choice for arbitrarily distorted ALE grids but when grids were frequently remapped back to the original grid, suppressing large grid distortion, edge viscosity was as accurate as tensor viscosity and for spherical shock tests maintained symmetry better. Edge viscosity is only used in ALE grids in conjunction with sub-zonal pressures which are required to suppress hourglass modes. As hourglass modes cannot grow in a Lagrangian-remap code the conclusion for these tests was that the best option for SWIFT was an edge based shock viscosity using the scheme in [5]. This is as good as the mimetic tensor viscosity in *Lare3d* but quicker. As a result of this work for SWIFT the viscosity in *Lare3d* will now also be changed to edge viscosity.

Lare3d's use of a mimetic tensor viscosity provided all the components of the tensor required for a Cauchy Lagrangian update of the magnetic field. Hence the magnetic field for the Lagrangian step was updated through

$$B_j = \frac{\partial x_j}{\partial X_i} \frac{B_i^0}{\Delta} \tag{5}$$

where $x_j(t)$ are the locations of the nodes during the Lagrangian phase and $X_i = x_i(t = 0)$. The Jacobian for the Lagrangian transformation is Δ and $B_i^0 = B_i(t = 0)$. When SWIFT moves over to edge viscosity the tensor components $\partial x_j / \partial X_i$ are no longer pre-calculated in the viscosity and as a result the magnetic field updated is now treated through

$$\frac{DB_i}{Dt} = \int v_i \mathbf{B}.\mathbf{dS} \tag{6}$$

2.2 Area-volume Lagrangian remap in spherical geometry

SWIFT uses a staggered grid with velocities defined at cell corners (nodes), scalars defined at cell centres (zones) and magnetic field components defined on cell faces (areas). Thus magnetic field components are not all defined at the same location. Once SWIFT has chosen the location of the nodes it calculates the metric factors $(h_1, h_2, h_3) = (1, r, r \sin\theta)$

for the (r, θ, ϕ) spherical system at the zone centroid, cell faces and on those edges where it will be needed for finite differencing, see next section. Cell face areas are then defined from these metrics, e.g. $A_r = h_2 h_3 d\theta d\phi$ and the zone volume from the centroid defined metrics through $V = h_1 h_2 h_3 dr d\theta d\phi$. Changing SWIFT to work in Cartesian or cylindrical for code comparisons, prototyping new physics modules or debugging simple requires redefining the h metrics.

After a Lagrangian step the nodes have moved and the grid is deformed. The variables on this deformed grid are then conservatively remapped onto the original grid, as described in [1], using a mixture of control volumes for the density, mass coordinates for the velocities and specific internal energy and areas for the magnetic fluxes. However as *Lare3d* was coded for purely Cartesian the remap distances, for the density remap for example, used only the linear distance in the direction of the remap. These overlap distances had to be replaced by overlap volumes and swept areas in SWIFT. When $(h_1, h_2, h_3) = (1, 1, 1)$ this reverts to *Lare3d* and for $(h_1, h_2, h_3) = (1, r, r \sin\theta)$ gives spherical geometry.

2.3 Treating forces via finite difference

The control volume/area scheme used for the geometry of the remap step could be applied to the calculation of forces in the Lagrangian step, which uses a second order predictor-corrector scheme. The mass in a zone is constant during the Lagrangian step so only volume changes are needed to find the density after the grid moves. The specific energy update is not handled via equation (4) but via a compatible update based on the nodal forces [3]. All that is needed then is the control volume, Lagrangian average, of the momentum equation. Integrating equation (3), but without the $\mathbf{j} \times \mathbf{B}$ term, over the moving Lagrangian zone using the Reynolds transport theorem gives

$$M \frac{D\tilde{u}}{Dt} = -\int P \mathbf{dS} + \hat{r} \int \frac{2P}{r} dV.$$

where $\tilde{u}$ is the mass averaged velocity $\tilde{u} = \int \rho u dV/M$. The last term arrises from noting that $\int \hat{a}.\nabla P dV = \int P\hat{a}.dS - \int P\nabla.\hat{a}dV$ and applying Gauss's theorem. While this last term can be easily included in SWIFT the equivalent for the MHD force $j \times B$ on a Lagrangian grid is far less straightforward [6]. Tests have shown that as each full step of SWIFT begins with the same regular grid it is far more computationally efficient, and just as accurate, to simply evaluate j and B at the node and find $j \times B$ directly. The same approach is therefore applied to the pressure gradient term in SWIFT. This does require the gradient, divergence and curls be evaluated on a spherical grid. Since the h factors are defined in the initial conditions, and do not change, these are for the gradient of a scalar f

$$\nabla f = \left(\frac{1}{h_1}\frac{\partial}{\partial x_1}f, \frac{1}{h_2}\frac{\partial}{\partial x_2}f, \frac{1}{h_3}\frac{\partial}{\partial x_3}f\right) \tag{7}$$

the divergence of vector field $A = (a_1, a_2, a_3)$

$$\nabla.A = \frac{1}{h_1h_2h_3}\left(\frac{\partial}{\partial x_1}h_2h_3a_1, \frac{\partial}{\partial x_2}h_1h_3a_2, \frac{\partial}{\partial x_3}h_1h_2a_3\right) \tag{8}$$

and for the curl

$$\begin{aligned}\nabla \times A \quad = \quad & \left\{\frac{1}{h_2h_3}\left(\frac{\partial}{\partial x_2}h_3a_3 - \frac{\partial}{\partial x_3}h_2a_2\right),\right. && (9)\\ & \frac{1}{h_1h_3}\left(\frac{\partial}{\partial x_3}h_1a_1 - \frac{\partial}{\partial x_1}h_3a_3\right), && (10)\\ & \left.\frac{1}{h_1h_2}\left(\frac{\partial}{\partial x_1}h_2a_2 - \frac{\partial}{\partial x_2}h_1a_1\right)\right\} && (11)\end{aligned}$$

Finite differencing of these are also used to give the current density j from the magnetic field B.

2.4 Spherical test results

To check that SWIFT does correctly maintain spherical symmetry and can correctly solve MHD in this geometry a selection of idealised test cases are presented. These include simple Gaussian pulses, which when symmetric about $r = 0$, provide a good test of symmetry preservation. The calculated MHD waves speeds in each direction also verify the MHD solver is correct. Beyond these tests a proxy solar wind test is presented. This

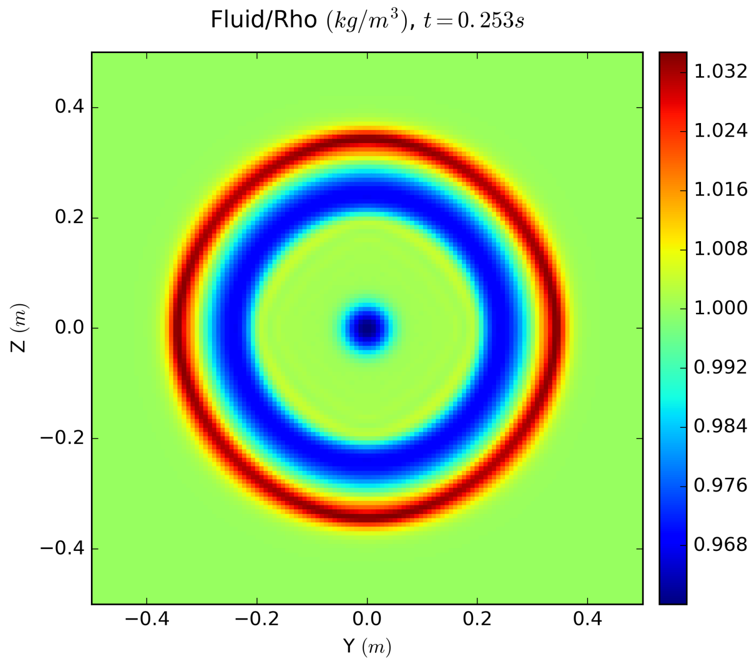

Figure 1: Normalised mass density for Gaussian pulse in Cartesian geometry using a 128^3 grid.

is not a realistic solar wind, which requires careful setup and then continuous driving from AWSoM or similar to establish a real solution, but contains all of the computational elements. These are the inflow/outflow boundaries, mass and magnetic flux injection, a wind roughly matched to a Parker wind and model spiral magnetic field.

The Gaussian pulse test is setup with a uniform density and temperature and zero velocity. A normalised mass density $\rho = 1$ was used with a a perturbation $\rho_1 = \exp(-(r - r_0)^2/\sigma^2)$ where r_0 was varied but $\sigma = 2 \times 10^{-3}$. Figure 1 shows the result for a Cartesian grid and figure 2 for the equivalent resolution spherical simulation when $r_0 = 0$. Thus SWIFT reproduces *Lare3d* results in Cartesian and gives improved symmetry for spherical geometry as expected.

The Gaussian test was repeated with the initial density enhancement offset from $r = 0$. The result in figure 3 shows that spherical symmetry is still preserved. Grid level effects can now be seen and are more pronounced at larger radii where the underlying resolution is worse. Note this is only roughly one quarter of the resolution SWIFT is expected to use

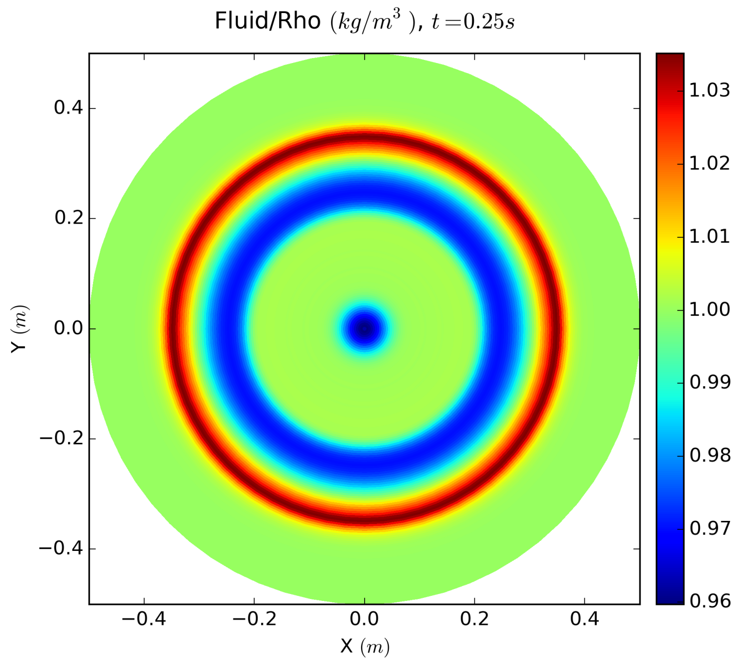

Figure 2: Normalised mass density for Gaussian pulse in spherical geometry an (64, 64, 128) grid in (r, θ, ϕ).

in space weather prediction. Also these figures are cell averaged values to clearly show the underlying numerical solution. Results from ENLIL for example use contour plots which smooth most of these grid level effects.

As a final test we setup a model solar wind. While artificial this is representative of the numerical problems expected when driven by real data from AWSoM. An analytic Parker wind is initialised with

$$u^2 - \ln(u^2) = 4\ln r' + 4\frac{4}{r'} + 3 \tag{12}$$

Here u is the radial velocity normalised to the sound speed $c_s = 2k_BT/m_p$ and the temperature is taken as uniform at 1 MK. The radius is normalised to the critical radius $r' = r/r_c$ where $r_c = GM_\odot/2c_s^2$. The mass density and pressure are then given by $\rho = C/r_c^2 c_s$ and $P = 2\rho k_b T/m_p$, with C chosen so that $\rho = 6 \times 10^6 m_p$ kg/m^3 at 1A.U. The solution domain runs from 0.1 to 1.1 A.U., poloidal angles $\pi/6$ to $\pi - \pi/6$ and a full 2π in toroidal/azimuthal angle. The resolution is (128, 32, 64) grid in (r, θ, ϕ) and

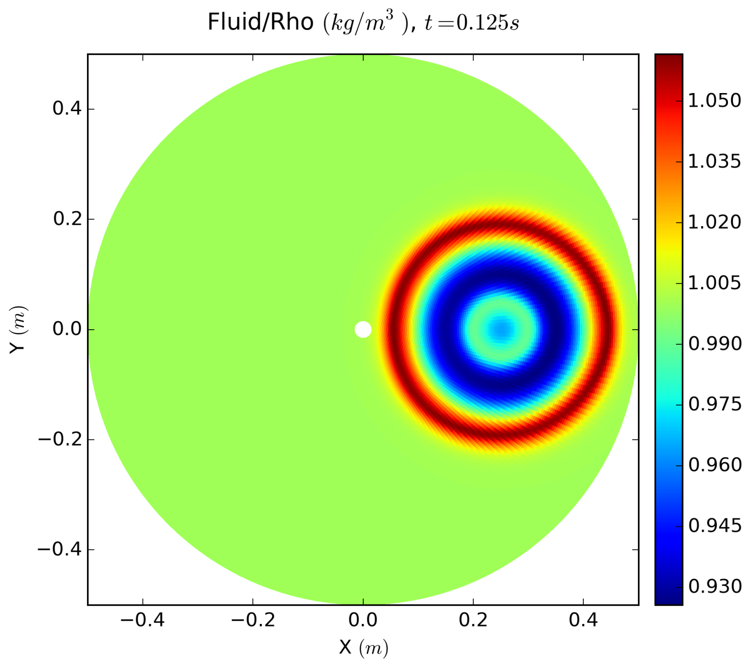

Figure 3: Normalised mass density for Gaussian pulse in spherical geometry with the pulse offset in the radial coordinate. This used the same resolution as in figure 2

the simulation is run for two solar rotations. An initial radial magnetic field is imposed with $B_r = B_0/r^2$ and $B_0 = 9$ nT so that solar rotation will impose a spiral magnetic field. Finally an artificial disturbance is created by injecting a mass density 100 times background on the inner radial boundary for angles $\phi = v_\phi t \pm 0.05\pi$ where v_ϕ is the imposed rotation speed at 0.1 A.U. from solar rotation. For all other angles the injected density is that from the analytic model above.

Plotted in figure 4 is the magnitude of the B_x component of the magnetic field $B_x = B_r \cos\phi$, scaled by $(r/au)^2$, in the ecliptic plane. The same field is plotted in figure 5 in the $\phi = \pi$ plane, i.e. the slice in (x, z) through $y = 0$ for $x < 0$. The same slices but for number density are shown in figures 6 and 7. These number densities are also scaled by $(r/au)^2$.

These tests show correct treatment of inflow/outflow boundaries in r, no issues withthe slip boundaries at $\theta = \pm 60^o$ and correct spherical symmetry. The model Parker solar wind behaves as expected and the code has been tuned and optimised for spherical geometry

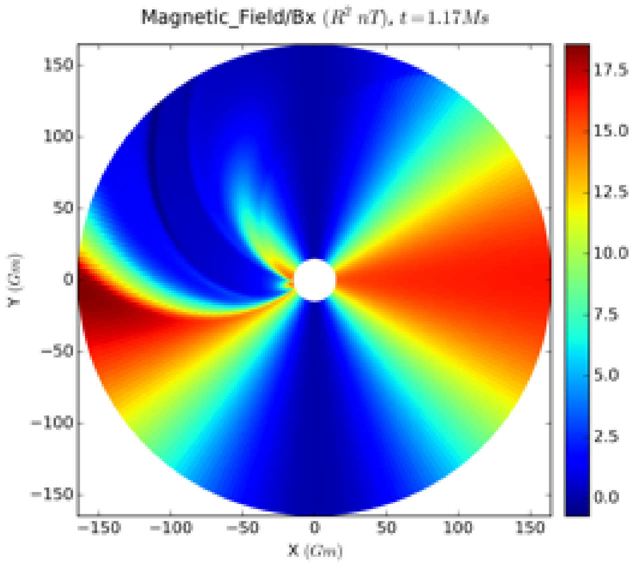

Figure 4: Magnitude of B_x in the ecliptic plane after half a solar rotation.

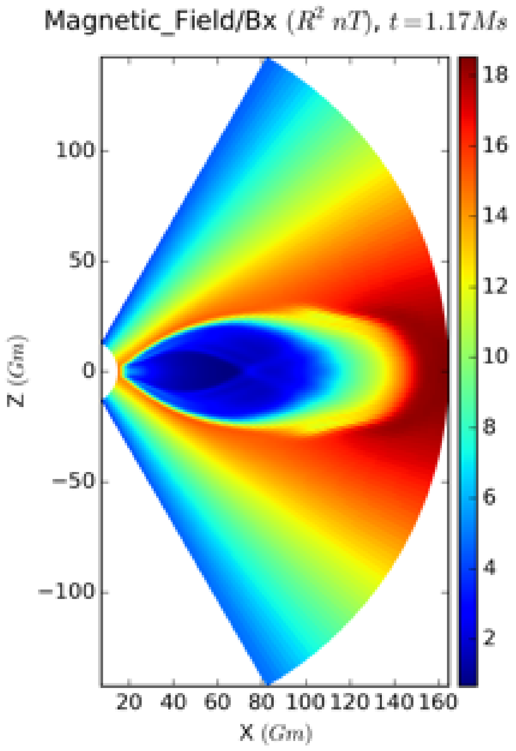

Figure 5: Magnitude of B_x in the $\phi = \pi$ plane after half a solar rotation.

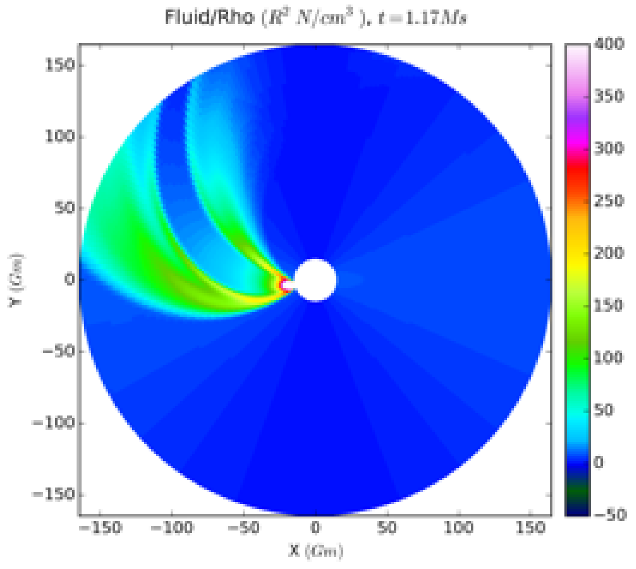

Figure 6: Number density in the ecliptic plane after half a solar rotation.

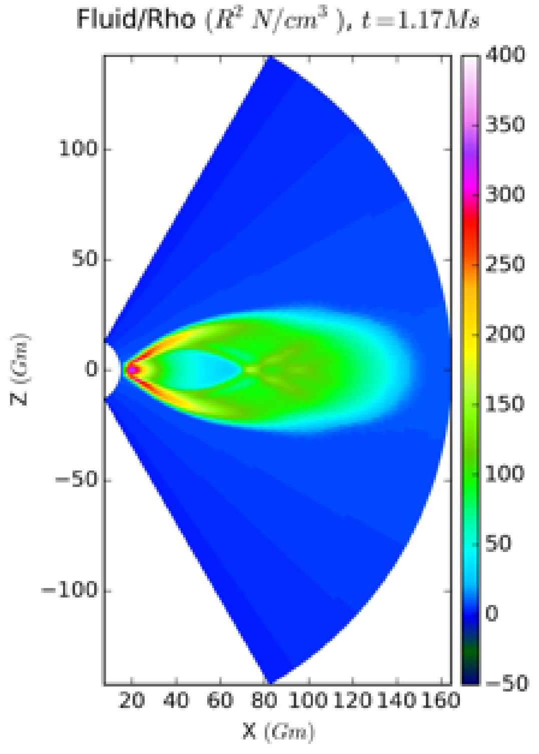

Figure 7: Number density in the $\phi = \pi$ plane after half a solar rotation.

ready for coupling to the AWSoM code.

3 Issues and future development

The 3D spherical geometry MHD code SWIFT is now released under source control from a Warwick University based *gitlab* server (https://cfsa-pmw.warwick.ac.uk). This will be coupled with the AWSoM code from the University of Michigan by month 20 of the PROGRESS project. Components of SWIFT which will be needed for a realistic solar wind model independently the implementation of spherical geometry detailed above are:

1. Thermal conduction routines from *Lare3d* which are based on super-stepping schemes [7] will be moved over to SWIFT. These have been developed and tested in *Lare3d* which was used instead of SWIFT as the development took place in parallel with the conversion of SWIFT to spherical geometry and needed a stable platform for this development prior to the completion of spherical SWIFT.

2. The two-temperature model within *Odin* needs implementing in SWIFT. This will allow shock heating only of the ions and thermal conduction only on the electrons. This has been in *Odin* for over a year and was not part of the work undertaken for this project.

3. Full testing of the feature complete SWIFT, i.e. current release plus two-temperatures and thermal conduction, prior to coupling to AWSoM.

4. The AWSoM coupling may have a problem if AWSoM tries to inject a magnetic field with an associated poloidal flux through the poloidal boundaries in SWIFT. The issue is that AWSoM runs over the whole sphere. SWIFT can do this but it is quicker if only those poloidal angles which affect results at L1 are included. SWIFT with a reduced poloidal range then has a slip boundary along those new poloidal boundaries which are absent from the driver AWSoM code. The injection of poloidal flux on these boundaries from AWSoM would need to be suppressed as

the resulting computational open boundaries would now no longer be a well posed mathematical problem.

5. Currently SWIFT relies on flat MPI domain decomposition. Changes to optimal machine architectures over the next two years, for example the Intel KNL processor or IBM Power plus nVidea, may require either multi-threading or CUDA. This decision will not be made until a clearer picture of available hardware emerges.

4 Conclusions

The Cartesian geometry *Lare3d* code has been converted to the spherical geometry MHD code SWIFT. This code has been tested for symmetry, shocks and MHD model solar-winds. Any researcher experienced in computational MHD can use this report, along with a copy of the *Lare3d* code and manual, to reproduce the SWIFT code base. The key decisions were the choice of viscosity, which routines were handed in area-volume control differencing and which through standard finite differences, albeit with a choice of three orthogonal metrics. The SWIFT code is now available for any of the PROGRESS project team to use via a Warwick maintained *gitlab* repository.

References

[1] T D Arber, A W Longbottom, C L Gerrard, and A M Milne. A Staggered Grid, Lagrangian–Eulerian Remap Code for 3-D MHD Simulations. *Journal of Computational Physics*, 171(1):151–181, 2001.

[2] J C Campbell and Mikhail J Shashkov. A Tensor Artificial Viscosity Using a Mimetic Finite Difference Algorithm. *Journal of Computational Physics*, 172(2):739–765, 2001.

[3] E J Caramana, D E Burton, Mikhail J Shashkov, and P P Whalen. The construction of compatible hydrodynamics algorithms utilizing conservation of total energy. *Journal of Computational Physics*, 146(1):227–262, October 1998.

[4] E J Caramana and Mikhail J Shashkov. elimination of artifical grid distortion by sub-zonal pressures. *Journal of Computational Physics*, 142, 1998.

[5] E J Caramana, Mikhail J Shashkov, and P P Whalen. Formulations of articificial viscosity in multi-dimensional shock codes. *Journal of Computational Physics*, 144, 1998.

[6] I J D Craig and A D Sneyd. A dynamic relaxation technique. *Astrophysical Journal*, 311, 1986.

[7] Chad D Meyer, Dinshaw S Balsara, and Tariq D Aslam. A stabilized Runge–Kutta–Legendre method for explicit super-time-stepping of parabolic and mixed equations. *Journal of Computational Physics*, 257:594–626, 2014.

[8] B van der Holst, Igor V Sokolov, X Meng, Meng Jin, Iv W B Manchester, G Tóth, and Tamas I Gombosi. ALFVÉN WAVE SOLAR MODEL (AWSoM): CORONAL HEATING. *Astrophysical Journal*, 782(2):81, 2014.