

The stellar atmosphere simulation code *Bifrost*

Code description and validation

B. V. Gudiksen^{1,2}, M. Carlsson^{1,2}, V. H. Hansteen^{1,2}, W. Hayek³, J. Leenaarts⁴, and J. Martínez-Sykora^{1,5}

¹ Institute of Theoretical Astrophysics, University of Oslo, PO Box 1029 Blindern, 0315 Oslo, Norway
e-mail: boris@astro.uio.no

² Center of Mathematics for Applications, University of Oslo, PO Box 1053, Blindern, 0316 Oslo, Norway

³ School of Physics, University of Exeter, Stocker Road, Exeter EX4 4QL, UK

⁴ Sterrekundig Instituut, Utrecht University, Postbus 80 000, 3508 TA Utrecht, The Netherlands

⁵ Lockheed Martin Solar & Astrophysics Lab, Org. ADBS, Bldg. 252, 3251 Hanover Street Palo Alto, CA 94304, USA

Received 14 January 2011 / Accepted 27 May 2011

ABSTRACT

Context. Numerical simulations of stellar convection and photospheres have been developed to the point where detailed shapes of observed spectral lines can be explained. Stellar atmospheres are very complex, and very different physical regimes are present in the convection zone, photosphere, chromosphere, transition region and corona. To understand the details of the atmosphere it is necessary to simulate the whole atmosphere since the different layers interact strongly. These physical regimes are very diverse and it takes a highly efficient massively parallel numerical code to solve the associated equations.

Aims. The design, implementation and validation of the massively parallel numerical code *Bifrost* for simulating stellar atmospheres from the convection zone to the corona.

Methods. The code is subjected to a number of validation tests, among them the Sod shock tube test, the Orzag-Tang colliding shock test, boundary condition tests and tests of how the code treats magnetic field advection, chromospheric radiation, radiative transfer in an isothermal scattering atmosphere, hydrogen ionization and thermal conduction.

Results. *Bifrost* completes the tests with good results and shows near linear efficiency scaling to thousands of computing cores.

Key words. magnetohydrodynamics (MHD) – radiative transfer – methods: numerical – Sun: atmosphere – stars: atmospheres

1. Introduction

The development of faster computers and better algorithms has made simulations a viable experimental tool to understand a number of astrophysical problems in detail. This development was clear more than a decade ago (Miyama et al. 1999). The years that have followed have borne this out fully and there are now a number of groups that are modeling the outer layers of cool stars, including magnetic fields, to a high degree of realism (Gudiksen & Nordlund 2005; Hansteen 2004; Stein & Nordlund 2006; Hansteen et al. 2007; Schaffenberger et al. 2005; Vögler et al. 2005; Heinemann et al. 2007; Abbott 2007; Isobe et al. 2008).

Vital to this effort was the development of multi-group techniques to handle radiative transfer in the photosphere (Nordlund 1982) and for models extending into the chromosphere, scattering (Skartlien 2000). The majority of the codes mentioned above employ these multi-group methods.

As the initial exploratory phase for codes including magnetoconvection is nearing successful completion, a number of challenging problems are now being considered with some confidence.

The problems include issues such as the existence and formation of supergranulation, (Stein et al. 2009, 2010) the appearance of faculae, (Keller et al. 2004; Carlsson et al. 2004) the formation of active regions and spots (Cheung et al. 2007; Heinemann et al. 2007; Rempel et al. 2009) the flux emergence

into the chromosphere and corona (Martínez-Sykora et al. 2008, 2009a; Tortosa-Andreu & Moreno-Insertis 2009) the structure and heating of the chromosphere and corona (Abbott 2007; Carlsson et al. 2010; Hansteen et al. 2010) and acceleration of spicules (Hansteen et al. 2006; De Pontieu et al. 2007; Heggland et al. 2007; Martínez-Sykora et al. 2009b, 2010).

Recent codes solve the full radiative magnetohydrodynamic (MHD) equations with some precision. However, additional physics may have to be considered in order to solve problems inherent to the low density, low and/or high temperature conditions of the outer solar atmosphere. These effects encompass the inclusion of thermal conduction by various methods (Gudiksen & Nordlund 2005; Hansteen et al. 2007), but also Generalized Ohm's law (Arber et al. 2007), as well as non-equilibrium effects such as hydrogen ionization and molecule formation (Leenaarts et al. 2007; Wedemeyer-Böhm & Steffen 2007), are now being added to these codes.

Numerical simulations require complex algorithms to solve the physics required, but in addition, combining high spatial resolution with the large spatial scales characteristic of atmospheric phenomena requires large memory and many CPU hours computing time. The cost of high performance computing (HPC) specific hardware has driven the market for supercomputers to be focused mainly on utilizing relatively cheap off-the-shelf computer parts instead of developing specific supercomputing hardware. The cheapest performance can be gained from connecting almost standard computers through a local network. The

interconnect speed and communication software implementation is what sets one cluster apart from another. Such clusters have a distributed memory architecture, which is different from previous generations of supercomputers with specifically built hardware which were typically shared memory vector machines. The consequence is that the numerical codes used in for instance astrophysics have to be highly parallelized in order to perform well on large HPC systems using, for example, the Message Passing Interface (MPI). In the near future we expect further developments of these and other techniques such as the widespread use of adaptive mesh refinement and graphics processor units (GPUs).

In this paper we present a code, *Bifrost*¹ to solve the MHD equations in a stellar atmosphere context, specifically designed to take advantage of the environment provided by modern massively parallel machines connected through potentially slow communication channels. The main design goal has been to reduce the amount of communication required at each timestep while retaining good scaling on problems requiring up to several thousand processors.

In addition, we aim to create a flexible design in which various physical processes and extensions to the basic MHD equations are straightforward to implement without rewriting large portions of the code.

2. Basic equations and set-up

Bifrost is built on several generations of previous numerical codes, where the Oslo Stagger Code is the latest, which at their core have the same implemented method (Nordlund & Galsgaard 1995; Galsgaard & Nordlund 1996). The core of the code remains the same, but as the previous generations of codes required shared memory architectures, the need for a new massively parallel code able to run on distributed memory computers was obvious. As the core of the code was rewritten, we had the opportunity to make *Bifrost* much more modular and user friendly than the previous generations of codes.

At the core of the code is a staggered mesh explicit code that solves the standard MHD partial differential equations (PDEs) on a Cartesian grid:

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \rho \mathbf{u} \quad (1)$$

$$\frac{\partial \rho \mathbf{u}}{\partial t} = -\nabla \cdot (\rho \mathbf{u} \mathbf{u} - \tau) - \nabla P + \mathbf{J} \times \mathbf{B} + \rho \mathbf{g} \quad (2)$$

$$\mu \mathbf{J} = \nabla \times \mathbf{B} \quad (3)$$

$$\mathbf{E} = \eta \mathbf{J} - \mathbf{u} \times \mathbf{B} \quad (4)$$

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E} \quad (5)$$

$$\frac{\partial e}{\partial t} = -\nabla \cdot e \mathbf{u} - P \nabla \cdot \mathbf{u} + Q \quad (6)$$

where ρ , \mathbf{u} , e , \mathbf{B} are the density, the velocity vector, the internal energy per unit volume and the magnetic flux density vector respectively. τ , P , \mathbf{J} , \mathbf{g} , μ , \mathbf{E} and η are the stress tensor, the gas pressure, the electric current density vector, the gravitational acceleration, the vacuum permeability, the electric field vector and the magnetic diffusivity respectively. The quantity Q can contain a number of terms, depending on the individual experiment. It could for instance contain a term from the chosen Equation Of

State (EOS), a term containing the effect of the Spitzer thermal conductivity, a term from radiative transfer, etc. The EOS needed to close this set of equation can be anything from a simple ideal gas EOS to a complex EOS including detailed microphysics (see Sect. 7).

3. The method

The basic operator in the code is the 6th order differential operator. The derivative of the function f in the grid point location $+1/2$ has the form

$$\begin{aligned} \frac{\partial f_{+1/2}}{\partial x} = & a_x [f_{+1} - f_0] + \\ & b_x [f_{+2} - f_{-1}] + \\ & c_x [f_{+3} - f_{-2}] \end{aligned} \quad (7)$$

where the subscript is the gridpoint number and a_x , b_x , c_x depend on the grid point distance. As the derivative operator produces results that lie half way between grid points, the variables are not co-located in space, but placed on staggered grids such that some variables are placed at cell centers, some on cell faces and some at cell corners. By carefully choosing which variables to place in each of these locations, it is possible to minimize the number of interpolations needed in order to realign computed values in space. Nevertheless, it is not possible to escape interpolations, and when it is needed the interpolation procedures used are 5th order and look very similar to the derivative operators:

$$\begin{aligned} f_{+1/2} = & a [f_0 + f_{+1}] + \\ & b [f_{-1} + f_{+2}] + \\ & c [f_{-2} + f_{+3}] \end{aligned} \quad (8)$$

where a , b and c are constants.

The computational grid can be stretched in one direction at a time, meaning that $d\mathbf{x}$ is not constant in the computational volume. The stretching of the grid is done by a simple Jacobian transformation after a derivative operator is used. Stretching in more than one direction cannot be accomplished in this simple manner without using a number of interpolations which would decrease precision, and at the moment it is not possible to employ an adaptive mesh refinement scheme in the code, but due to the modularity of the code this can be implemented at a later stage.

3.1. Diffusion

All numerical codes are diffusive in nature, just due to the discrete nature of the algorithms used in solving the equations and because of the inaccurate machines used to solve them on. This is even true for implicit codes that do not contain any direct diffusive terms in their equations, that said, the diffusion in implicit codes is significantly smaller than for explicitly diffusive codes of the same order. *Bifrost* is an explicit code and it is therefore necessary to include diffusive terms in Eq. (1)–(6) in order to maintain stability. *Bifrost* employs a diffusive operator which is split in two major parts: a small global diffusive term and a location-specific diffusion term (sometimes dubbed “hyper diffusion”). The diffusive operator in one dimension is of the form

$$\frac{\partial f}{\partial t} = \dots + \frac{\partial}{\partial x} \left[\nu \, dx \left(\nu_1 C_{\text{fast}} + \nu_2 |\mathbf{u}| + \nu_3 \, dx \, \nabla_x^1 u_x \right) \frac{\partial f}{\partial x} Q \left(\frac{\partial f}{\partial x} \right) \right] \quad (9)$$

¹ In Norse mythology, *Bifrost* (pron. bee-frost) is the name of the rainbow bridge from Midgard (the realm of man) to Asgard (the realm of the gods), build from fire, water and air.

where

$$Q(g) = \frac{\frac{\partial}{\partial x} \left(\frac{\partial g}{\partial x} \right)}{|g| + \frac{1}{q} \frac{\partial}{\partial x} \left(\frac{\partial g}{\partial x} \right)} \quad (10)$$

and ∇_x^1 is the first order gradient in the x direction. The splitting of the diffusive terms into local and global components makes it possible to run the code with a global diffusivity that is at least a factor 10 less than if the global term were the only one implemented in the code. The splitting also makes it impossible to provide a single Reynolds number, Magnetic Reynolds number or any other dimensionless number that includes the diffusion constant, since diffusion is not constant in space or time when running *Bifrost*. Therefore it is only possible to provide a range of the above mentioned numbers, of which the smaller, global, value for the diffusion would be correct for most of the simulation volume most of the time (unless, for instance, simulating supersonic turbulence), while the higher number of the range for the diffusion would be valid in locations characterized by very large values of the gradients of a certain variable. Thus, in principle, the code can be run at much greater values of the relevant Reynolds numbers in most of the computational domain than would be otherwise possible using a single diffusion coefficient.

4. Modules

Bifrost has been created with a high degree of modularity. The code has a basic skeleton which connects to a number of modules. Any of these modules can contain a simple procedure or method, include a number of sub-modules or be left empty. For example, the timestepping procedure can be swapped between a Runge-Kutta scheme and a Hyman scheme by changing only a single line in an input file to the compiler, plus a recompilation. Most interestingly, the code can include any number of modules that provide new physics, or boundary conditions, in the same simple way.

As there can be several implemented modules handling the same job in the code (timestepping, EOS, radiative transfer etc.) this section contains the most important of these, which are presented either with a short description if the modules are standard numerical schemes, or in more detail if they are specific to *Bifrost*.

5. Timestepping

Timestepping can be handled by two different procedures: A third-order Runge-Kutta method or a third-order Hyman method. Both of these procedures produce nearly identical results. The Runge-Kutta method is able to handle a longer timestep, while being more computationally intensive, so in terms of CPU time their effectiveness is almost the same.

5.1. Third-order Runge-Kutta timestepping scheme

The implemented 3rd order Runge-Kutta scheme splits the timestep into three sub-steps. In order to take one timestep all the partial differential equations are solved three times but it is not necessary to save more than two results at a time in the same timestep, making this scheme $2N$ in memory requirements (where N is the amount of memory needed to run through the partial differential equations once). For large simulations that can be a considerable memory saving feature. The advantage

in this method is that the intermediate timestep results can be used to extrapolate the result of the total timestep further in time than that possible by using three separate time steps, while at the same time having a high order precision. The Runge-Kutta method is defined by assuming the change in the variable f can be written

$$\frac{\partial f}{\partial t} = F(t, f(t)). \quad (11)$$

The change during one timestep Δt of the variable f is then given by

$$f(t + \Delta t) = f(t) + \frac{\Delta t}{6} (k_1 + 4k_2 + k_3), \quad (12)$$

where

$$k_1 = F(t, f(t)), \quad (13)$$

$$k_2 = F\left(t + \frac{1}{2}\Delta t, f(t) + \frac{1}{2}k_1\Delta t\right) \quad \text{and} \quad (14)$$

$$k_3 = F(t + \Delta t, f(t) - k_1\Delta t + 2k_2\Delta t). \quad (15)$$

5.2. Third-order Hyman timestepping scheme

The third-order Hyman predictor-corrector scheme is described by Hyman (1979). It is an iterative multistep method employing a leap-frog scheme to attain 3rd order accuracy in time. The method is quite simple and can be described by assuming that the differential equations can be written in the following form:

$$\frac{\partial f}{\partial t} = F(t, f(t)). \quad (16)$$

The Hyman method's first step is to find a second order predictive solution to Eq. (16) by using the formula:

$$f_{n+1}^{(2)} = (1 - r^2)f_n + r^2 f_{n-1} + \Delta t(1 + r)F_n \quad (17)$$

where the superscript is the order of the term, the subscript is the timestep number and $F(t, f_n(t))$ has been written F_n for simplicity and

$$r \equiv (\Delta t_{n+1}/\Delta t_n) \quad (18)$$

with Δt being the timestep. Then the PDE's are solved again and finally a corrector is applied given by

$$F_{n+1}^{(3)} = \left[(2 - r)(1 + r)^2 f_n + r^3 f_{n-1} + \Delta t(1 + r)^2 F_n + \Delta t(1 + r)F_{n+1}^{(2)} \right] / (2 + 3r). \quad (19)$$

6. Boundary conditions

Bifrost is able to make the computational box periodic in one, two or three dimensions. Non-periodic boundaries imply that a boundary condition must be imposed on that boundary. *Bifrost* implements non-periodic boundary conditions by padding the computational domain with ghost zones in the relevant direction and filling them according to the boundary condition chosen. Several standard boundary conditions are implemented including symmetric, antisymmetric as well as extrapolated boundary values that fill the ghost zones before the MHD PDEs are computed. Experiment-specific boundary conditions such as constant entropy flux for stellar convection simulations and characteristic boundary conditions can also be used by adhering to a simple format for boundary calls. In the latter case of characteristic boundaries the conservative MHD PDEs are replaced by the characteristic PDEs at the boundary zones before the variables in these zones are updated in the usual manner.

6.1. Characteristic boundary conditions

The characteristic boundary conditions aim at transferring disturbances through the boundaries without any or at least with minimal reflection, a problem that can plague standard boundary conditions that simply use symmetric or extrapolated values. In this case the 8 MHD equations are written in terms of the characteristics and split into horizontal and vertical components in the following manner.

With U' defining a vector that contains the conserved MHD variables we may write the equations as

$$\frac{\partial U'}{\partial t} + \sum_{k=1}^m \frac{\partial F^k}{\partial x_k} = D', \quad (20)$$

where F contains the fluxes, D' the source terms, and $m = 3$ for a 3D problem. These conservation equations can be transformed using linear algebra into “primitive”, wave-like equations for a corresponding set of 8 primitive variables U ,

$$\frac{\partial U}{\partial t} + \sum_{k=1}^m A^k \frac{\partial U}{\partial x_k} = D, \quad (21)$$

where the three A^k are 8×8 matrices. The choice of primitive variables U is not unique, and in principle could be taken to be the conserved variables. The goal of this procedure is then to arrive at equations that resemble simple advection equations where specifying boundary conditions is straightforward: extrapolation based on one-sided derivatives for outgoing characteristics and on the basis of exterior data, such as no incident waves, for incoming characteristics. By combining all the flux divergence terms except those in the direction perpendicular to the boundary, now called z , with the source term (forming a new source term C) we can write characteristic equations for the z direction in the sought after form

$$S^{-1} \frac{\partial U}{\partial t} + \Lambda S^{-1} \frac{\partial U}{\partial z} = S^{-1} C, \quad (22)$$

as shown by e.g. Thompson (1987). Here, the rows i of the matrix S^{-1} are given by the left eigenvectors l_i^T , and Λ is the diagonal matrix formed by the eigenvalues λ_i of the matrix A^k belonging to the z direction. We now define a vector d containing the z derivatives of the characteristic equations as

$$d \equiv \Lambda S^{-1} \frac{\partial U}{\partial z}.$$

Having isolated the various characteristic wave modes propagating in the z direction we left-multiply Eq. (22) by S in order to write the MHD equations in primitive form in terms of d :

$$\frac{\partial U}{\partial t} + Sd = C \quad (23)$$

where the primitive variables U are comprised of the variables ρ , u , e and B . The d vector containing the z derivatives $d_1 - d_8$ constitute the information that is flowing along the characteristics; outflowing characteristics are defined by the interior solution, inflowing characteristics by the requirement that the incoming wave be constant in time. Expressions for Eq. (23) in primitive form and for the characteristic derivatives d can be found in Appendix A.

7. Equation of state

The code provides several different EOS modules, which can be chosen according to the experiment one wants to perform. The EOS modules provide the temperature and pressure and their thermodynamic derivatives as a function of mass density and internal energy per mass. There are currently three different modules.

The first implements an ideal gas EOS, suited for testing and idealized experiments.

The second module implements an EOS based on tables generated with the Uppsala Opacity Package (Gustafsson et al. 1975). It assumes local thermodynamic equilibrium (LTE) for atomic level populations and instantaneous molecular dissociation equilibria. This package is required when running with full radiative transfer (see Sect. 8.3), as it also provides the opacity, thermal emission and scattering probability for the radiation bins. The tables are generated with a separate program; different tables can be generated to account for different stellar spectral type, chemical composition and number of radiation bins.

The third package computes the gas temperature, gas pressure and electron density explicitly based on the non-equilibrium ionization of hydrogen in the solar atmosphere. This package can only be used for simulations of the solar atmosphere, as it depends on a number of parameters that vary with stellar spectral type. These parameters have so far only been determined for the sun. More details on non-equilibrium hydrogen ionization are given in Sect. 9.

8. Radiative transfer

Full 3D radiative transfer is computationally very costly. The properties of radiation are very different from the strictly local problem of MHD, since radiation can couple thermodynamic properties over arbitrarily long distances. For a code that relies on the local properties of MHD to parallelize, this makes radiative transfer costly not just in shear computations needed to solve the problem, but also because the results of the computation must be communicated to all nodes. As a result, the modules handling radiative transfer in *Bifrost* have employed assumptions which simplify the calculations to some extent. At the moment three modules handle radiative transfer depending on the problem at hand and they are often combined.

The electron number density is often needed in the radiative transfer modules (e.g. for the calculation of collisional excitation rates and opacities). The electron number density computation depends on which EOS package is used: if hydrogen non-equilibrium ionization is calculated, the electron number density comes from the simultaneous solution of the hydrogen rate equations, the energy equation and the charge conservation equation (see Sect. 9 for details). For the EOS package based on tables generated with the Uppsala Opacity Package, the electron density comes from solving the Saha equation for all species. If the electron density is not provided by the EOS package but is needed (e.g., for the chromospheric radiation approximation, see Sect. 8.2), it is computed using the Saha relation for hydrogen, but setting a floor to the ionization degree of 10^{-4} to account for the easily ionized metals in the solar atmosphere.

8.1. Optically thin radiative transfer

In the outer solar atmosphere (and many outer stellar atmospheres) radiative losses can be simply treated assuming that the atmosphere is optically thin. In the sun this is true for most

lines from the upper chromosphere/lower transition region up to the corona. In this case (and assuming ionization equilibrium only dependent on temperature) the radiative transfer problem reduces to a radiative loss which only depends on density and temperature of the form (note that positive Q corresponds to heating):

$$Q_{\text{thin}} = -n_{\text{H}} n_{\text{e}} f(T) \exp(-P/P_0) \quad (24)$$

where n_{H} and n_{e} are the number densities of hydrogen and electrons respectively, $f(T)$ is a function of the temperature and the term $\exp(-P/P_0)$ provides a cutoff where $P > P_0$. The radiation stems mainly from the resonance lines of multiply ionized elements such as carbon, oxygen, and iron and the function $f(T)$ can be pre-computed assuming ionization equilibrium. This relation is valid from roughly 2×10^4 K and up. In *Bifrost* the function $f(T)$ is computed by using ionization and recombination rates given by [Arnaud & Rothenflug \(1985\)](#) and [Shull & van Steenberg \(1982\)](#) and collisional excitation rates given by the HAO-DIAPER atom data package ([Judge & Meisner 1994](#)) including lines from He, C, O, Ne and Fe. The optically thin losses from hydrogen lines are normally calculated in the chromospheric approximate radiation part, see Sect. 8.2, but may instead be included here in the cases where a calibration of the chromospheric radiative losses is lacking. The total hydrogen number density is derived from the plasma density ρ assuming solar abundances. We set P_0 to a typical value of the mid-chromospheric pressure and the term $\exp(-P/P_0)$ then makes sure that there are no optically thin contributions calculated in the deep convection zone where the temperature is also above 2×10^4 K.

8.2. Chromospheric radiation approximation

Chromospheric radiative losses are dominated by a small number of strong lines from hydrogen, calcium and magnesium ([Vernazza et al. 1981](#)). The source functions and opacities of these lines are very much out of local equilibrium and the optical depth is significant; neither the optically thin approach of Sect. 8.1 nor the full radiative transfer with coherent scattering and LTE opacities (see Sect. 8.3) give good results. We approximate the radiative loss in these lines with the formulae

$$Q_{[\text{H,Ca,Mg}]} = -C(T)_{[\text{H,Ca,Mg}]} n_{\text{e}} \rho \phi_{[\text{H,Ca,Mg}]}(m_{\text{c}}) \quad (25)$$

where $C(T)n_{\text{e}}\rho$ gives the total collisional excitation rate (in energy per volume per unit time), $\phi(m_{\text{c}})$ gives the probability that this energy escapes from the atmosphere and m_{c} is the column mass. The function $\phi(m_{\text{c}})$ and the temperature-dependent coefficient $C(T)$ are determined for each element from detailed 1D radiative transfer computations with the RADYN code (see, e.g., [Carlsson & Stein 1995](#)) and 2D computations with Multi3D ([Leenaarts & Carlsson 2009](#)). These functions include hydrogen lines and the Lyman continuum and all lines and continua from Ca II and Mg II. The method is described in detail in [Carlsson & Leenaarts \(in prep.\)](#).

Half of the UV radiation lost from the corona in optically thin lines (see Sect. 8.1) goes towards the sun and most of that is eventually absorbed in the chromosphere, mostly in the Lyman continuum and the He I continuum. This radiative heating is modeled through the representative bound-free absorption cross-section σ of He I at 25 nm:

$$Q_{\text{He}} = \sigma_{\text{He,25 nm}} n_{\text{He I}} \exp(-\tau_{\text{He,25 nm}}) J_{\text{thin}} \quad (26)$$

with J_{thin} the angle-averaged radiation field caused by Q_{thin} (see [Carlsson & Leenaarts, in prep.](#)).

8.3. Full radiative transfer

Full radiative transfer computations are required when a simulation includes the convection zone beneath the photosphere, covering optically thick regions, optically thin regions, and the transition between the two regimes. A simplified treatment using, e.g., Newtonian cooling or the diffusion approximation, cannot provide sufficiently realistic radiative heating and cooling rates in this boundary layer.

Owing to the very short time scales of photon interaction and propagation in a convective stellar atmosphere, it is possible to solve radiative transfer as a time-independent problem, resulting in the expression

$$\hat{\mathbf{n}} \cdot \nabla I_{\lambda}(\mathbf{x}, \hat{\mathbf{n}}) = -\chi_{\lambda}(\mathbf{x}) I_{\lambda}(\mathbf{x}, \hat{\mathbf{n}}) + j_{\lambda}(\mathbf{x}, \hat{\mathbf{n}}), \quad (27)$$

where $I_{\lambda}(\mathbf{x}, \hat{\mathbf{n}})$ is the monochromatic specific intensity at spatial point \mathbf{x} for a beam in direction $\hat{\mathbf{n}}$ at wavelength λ , χ_{λ} is the gas opacity, and j_{λ} is the emissivity. The two material constants χ_{λ} and j_{λ} depend both on the thermodynamic state of the gas and on the radiation field, and are highly wavelength-dependent in the presence of spectral lines and other atomic and molecular transitions. Velocity fields, such as convective motions in the atmosphere, lead to an additional coupling between ray directions and wavelengths through Doppler shifts. The complexity of taking full account of the underlying physical mechanisms is computationally prohibitive. We therefore assume a static medium and LTE for the gas opacity, which thus depends only on the local gas temperature and the gas density, and which can therefore be precomputed and tabulated. [Skartlien \(2000\)](#) and [Hayek et al. \(2010\)](#) showed the importance of photon scattering in simulations of the higher solar atmosphere; we therefore include a coherent scattering contribution in the gas emission, which requires an iterative solution of the radiative transfer equation to obtain a consistent radiation field.

The vast number of spectral lines encountered in a stellar atmosphere requires, in principle, the solution of a large number of radiative transfer problems. This is currently too demanding for realistic 3D radiation-hydrodynamical calculations. We therefore approximate the opacity spectrum by substituting the monochromatic χ_{λ} with a small number of mean opacities ([Nordlund 1982](#); [Skartlien 2000](#)) and solving wavelength-integrated radiative transfer.

The radiation field encountered in cool stellar atmospheres does not contribute significantly to the momentum balance. We consider only a radiative heating rate $Q_{\text{rad},i}$ for every mean opacity (index i), given by the first moment of Eq. (27):

$$Q_{\text{rad},i} = -\nabla \cdot \mathbf{F}_i = 4\pi\chi_i(J_i - S_i), \quad (28)$$

where \mathbf{F}_i is the radiative energy flux, J_i is the mean intensity and $S_i \equiv j_i/\chi_i$ is the source function. The solver computes the radiation field using the short characteristics technique in every spatial subdomain and iterates the solution until convergence. The radiative heating rates are then added to the energy equation (Eq. (6)). If the hydrodynamical timesteps are sufficiently short that changes in the radiative energy flux are small, full radiative transfer only needs to be computed in a fraction of the hydrodynamical timesteps. A detailed description of the numerical methods and the parallelization techniques used for the full radiative transfer module in *Bifrost* can be found in [Hayek et al. \(2010\)](#).

9. Hydrogen ionization

The ionization of hydrogen in the solar chromosphere does not obey LTE or statistical equilibrium ([Carlsson & Stein 2002](#)).

Proper modeling of this ionization requires that non-equilibrium effects be taken into account.

The hydrogen ionization module solves the time-dependent rate equations for the atomic hydrogen level populations n_i

$$\frac{\partial n_i}{\partial t} + \nabla \cdot (n_i \mathbf{u}) = \sum_{j, j \neq i} n_j P_{ji} - n_i \sum_{j, j \neq i} P_{ij} \quad (29)$$

together with an equation for time-dependent H₂ molecule formation and equations for energy and charge conservation. Here P_{ij} is the rate coefficient for the transition from level i to j and n_i is the number of energy levels in the hydrogen atom (normally set to 6 in *Bifrost*). Solution of this system of equations yields the gas temperature, electron density and the atomic and molecular hydrogen populations (n_{H_2}). The radiative transition rate coefficients in the rate equations are prescribed as determined by [Sollum \(1999\)](#). This removes the effect of the global coupling of the radiation and makes the problem computationally tractable, but still computationally demanding.

The gas pressure is computed as

$$P = kT \left(n_e + n_{\text{H}_2} + \sum_{i=1}^n n_i + n_o \right), \quad (30)$$

with n_o the number density of all other atoms and molecules that are not, or do not contain, hydrogen.

Full non-LTE radiation tables as functions of n_e and T can be used to replace the radiation tables as functions of e and ρ , which assume hydrogen ionization based on Saha ionization equilibrium.

The method is described in detail in [Leenaarts et al. \(2007\)](#), the additional equation for time-dependent H₂ formation is given in Appendix B.

10. Thermal conduction

As the plasma temperature rises towards one million degrees in the tenuous transition region and corona, thermal conduction becomes one of the major terms in the energy equation, and modeling of this term is vital if this portion of the atmosphere is to be simulated with any fidelity. The implemented form of the thermal conduction takes the form ([Spitzer 1956](#)):

$$\mathbf{F}_c = -\kappa_0 T^{5/2} \nabla_{\parallel} T \quad (31)$$

where the gradient of T is taken only along the magnetic field (∇_{\parallel}) and κ_0 is the thermal conduction coefficient. The conduction across the field is significantly smaller under the conditions present in the solar atmosphere and is smaller than the numerical diffusion, so it is ignored. Since thermal conduction is described by a second order diffusion operator, this introduces several difficulties: the Courant condition for a diffusive operator such as that scales with the grid size Δx^2 instead of with Δx for the magnetohydrodynamic operators. This severely limits the time step Δt the code can be stably run at. We have implemented two solutions to this problem.

In the first, the thermal flux is calculated and if the divergence of the thermal flux sets severe restraints on the timestep, it is throttled back by locally lowering the thermal conduction coefficient κ_0 . This method is only acceptable when the number of points where the conduction has to be throttled back is very small and the results must be analyzed carefully.

A second solution is to proceed by operator splitting, such that the operator advancing the variables in time is $L = L_{\text{hydro}} + L_{\text{conduction}}$. The conductive part of the energy equation is handled by discretizing

$$\frac{\partial e}{\partial t} = -\nabla \cdot \mathbf{F}_c = -\nabla_{\parallel} \cdot \left[\kappa_0 T^{5/2} \nabla_{\parallel} T \right] \quad (32)$$

and solving the resulting problem implicitly.

We discretize the $L_{\text{conduction}}$ operator using the Crank-Nicholson (or “ θ ”) method and thus write

$$(e^{n+1} - e^*)/dt = \theta \nabla_{\parallel} \cdot \mathbf{F}_c^{n+1} + (1 - \theta) \nabla_{\parallel} \cdot \mathbf{F}_c^n \quad (33)$$

where the quantities with superscript n are computed before the hydrodynamic timestep, the quantities with superscript $*$ are computed after the hydrodynamic timestep and the quantities with superscript $n + 1$ are the temperatures deduced implicitly. The variable θ is set to a value between 0.5 and 1. The implicit part of the problem is in our case computed using a multi-grid solver (a concise introduction to multi-grid methods may be found in chapter 19 of [Press et al. 1992](#)).

In implementing the multi-grid solver, we use the same domain decomposition as in the magnetohydrodynamic part of the code. The small scale residuals in Eq. (33) are smoothed using a few Gauss-Seidel sweeps at high resolution, before the resulting temporary solution is injected onto a coarser grid on which smoothing is again performed. The process continues by Gauss-Seidel smoothing sweeps on steadily coarser grids until the size of the problem on individual processors is small enough to be communicated and stored on each and every processor. At this point the partial solutions are spread to all processors which continue the Gauss-Seidel smoothing, now on the global problem, on steadily coarser grids. Finally, when the coarsest grid size is reached, the solution is driven to convergence by performing a great number of Gauss-Seidel sweeps. Subsequently, the coarser solutions are corrected and then prolonged on successively finer grids, first globally, but after a certain grid size is reached the temporary solution is spread and the problem is again solved locally, using the same domain decomposition as in the magnetohydrodynamic part of the code. After each prolongation, the solution is smoothed using a few Gauss-Seidel sweeps. The entire cycle can be repeated as many times as desired, until a converged solution is found.

11. Tests and validation

Bifrost has been extensively tested in order to validate the results it provides. When possible, the tests are taken from the literature to validate the code and allow comparison with previous work. These tests were selected for their simplicity, their utility and their challenge to the different terms in the algorithms.

11.1. The Sod shock tube test

The Sod shock tube test ([Sod 1978](#)) has become a standard test in computational HD and MHD. It consists of a one-dimensional flow discontinuity problem which provides a good test of a compressible code’s ability to capture shocks and contact discontinuities within a small number of zones, and to produce the correct density profile in a rarefaction wave. The test can also be used to check if the code is able to satisfy the Rankine-Hugoniot shock jump conditions, since this test has an analytical solution.

The code is set to run a 1D problem and using the initial conditions for the Sod problem. The fluid is initially at rest on

either side of a density and pressure jump. The density and pressure jumps are chosen so that all three types of flow discontinuity (shock, contact, and rarefaction) develop. To the left, respectively right side of the interface we have:

$$\rho_R = 1, \quad (34)$$

$$\rho_L = 0.125, \quad (35)$$

$$P_R = 1/\gamma, \quad (36)$$

$$P_L = 0.125/\gamma. \quad (37)$$

The ratio of specific heats is chosen to be $\gamma = 5/3$ on both sides of the interface and with a uniform magnetic field perpendicular to the 1D axis of the domain ($B_z = 1$). The units are normalized, with the density and pressure in units of the density and pressure on the left hand side of jump and the velocity in units of the sound speed. The length is in unit of the size of the domain and the time in units of the time required to cross the domain at the speed of sound.

We have run the simulation at different resolutions and with different values of the numerical diffusion coefficient in an effort to find the minimal diffusion the code can be run with without developing numerical instabilities. From this test and the following ‘‘field advection’’ and Orzag-Tang tests we find that the diffusion parameters should be greater than $\nu_1 > 0.02$, $\nu_2 > 0.2$ and $\nu_3 > 0.2$ to ensure stability. Other parameters, such as the quench parameter, have also been varied in testing the code, in order to find the best values for capturing the Sod shocks properly.

Figure 1 shows the density and pressure profiles at time 0.193. The solid line is our numerical solution and the dashed line is the analytic solution at the same instant in time. It is clear that when running with the numerical diffusion coefficients large enough to avoid post shock numerical instabilities, but low enough not to lose the sharp shock profiles, the code solves the Sod shock correctly.

11.2. The magnetic field advection test

This is a multidimensional convection test, which serves to test the conservation of the various MHD quantities such as the density, momentum, and magnetic field with advection. Moreover, multidimensional MHD problems present a special challenge to the conservation of the divergence of the magnetic field (Tóth & Odstrcil 1996). This can only occur if the scheme preserves $\nabla \cdot \mathbf{B} = 0$.

This advection test is based on a test described previously by DeVore (1991). The test involves advecting a cylindrical current distribution, which forms a tube of magnetic field, diagonally across the grid. Any arbitrary advection angle can be chosen and the tube can have any orientation. For the 3D results shown here, the problem domain is given by $0 \leq x_1 \leq 1$; $-1/(2 \cos(30^\circ)) \leq x_2 \leq 1/(2 \cos(30^\circ))$, $0 \leq x_3 \leq 1$, and the flow is inclined at 60 degrees to the x_2 axis in the same plane as x_1 ; where x_1 , x_2 and x_3 could be either x , y or z . The loop is oriented in the x_3 direction. This geometry ensures the flow does not cross the grid along a diagonal, so the fluxes in x_1 , x_2 and x_3 -directions will be different.

The flow velocity is set to 1.0, with direction $u_1 = \sin(60^\circ)$ and $u_2 = \cos(60^\circ)$. We have run many tests, changing u_x , u_y and u_z between these u_1 and u_2 values and have checked that there is no dependence on any direction. The density is 1, pressure is $1/\gamma$, and the gas constant is $\gamma = 5/3$. Periodic boundary conditions are used everywhere.

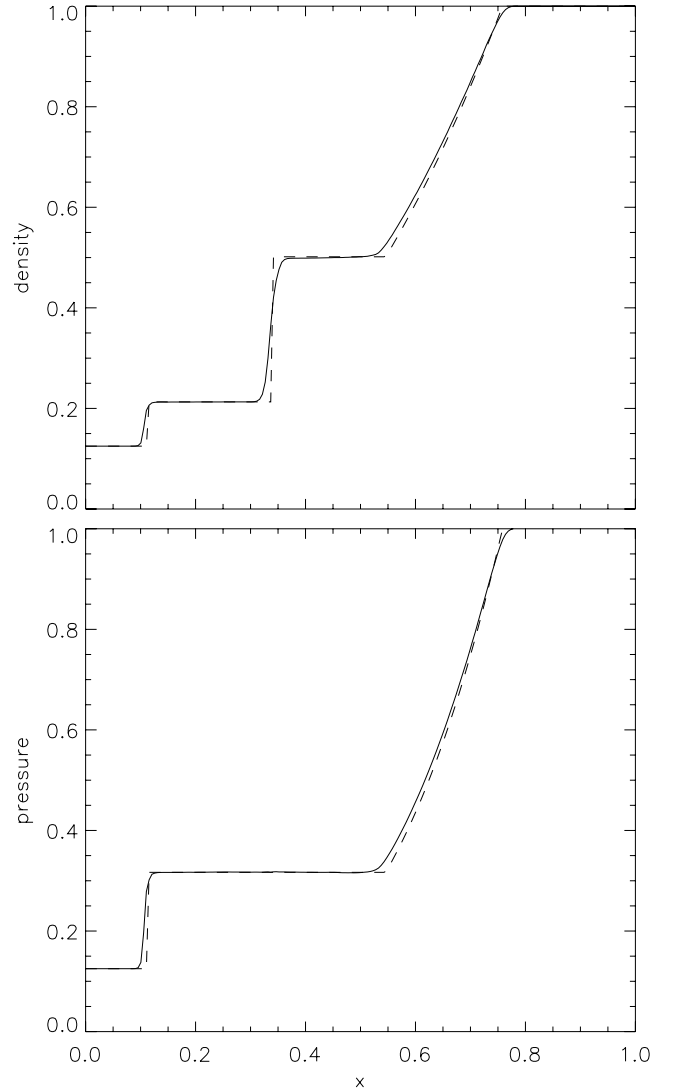


Fig. 1. Density profile (*top panel*) and pressure divided by the maximum of the pressure profile (*bottom panel*) at time 0.193. The solid line is the numerical result and the dashed line is the analytical solution.

The magnetic field is initialized using a vector potential, to make sure that the magnetic field is initially divergence free. The potential in this example is given by:

$$A_3 = 0.009e^{-(5r)^6}$$

where r is the distance from the box centre in the $x - y$ plane, and the two other components of the vector potential are set to zero. The magnetic vector potential provides the magnetic field vector according to $\mathbf{B} = \nabla \times \mathbf{A}$, and leaves a narrow cylinder of magnetic field and the divergence of the magnetic field is zero as calculated by *Bifrost*.

The code has solved this test with different setups of the direction of the initial velocity and orientation of the loop. We have also checked that the test performs successfully with varying diffusion and quench parameters. The increase in $\nabla \cdot \mathbf{B}$ with time lies within the numerical error. This means that every 10^6 time-step the cumulative error of the numerical errors makes it necessary to perform a cleaning of the $\nabla \cdot \mathbf{B}$.

In Fig. 2, the magnetic cylinder shows small alterations in shape due mainly to the rectangular grid used, and because the width of the magnetic ring initially was chosen to be at the limit

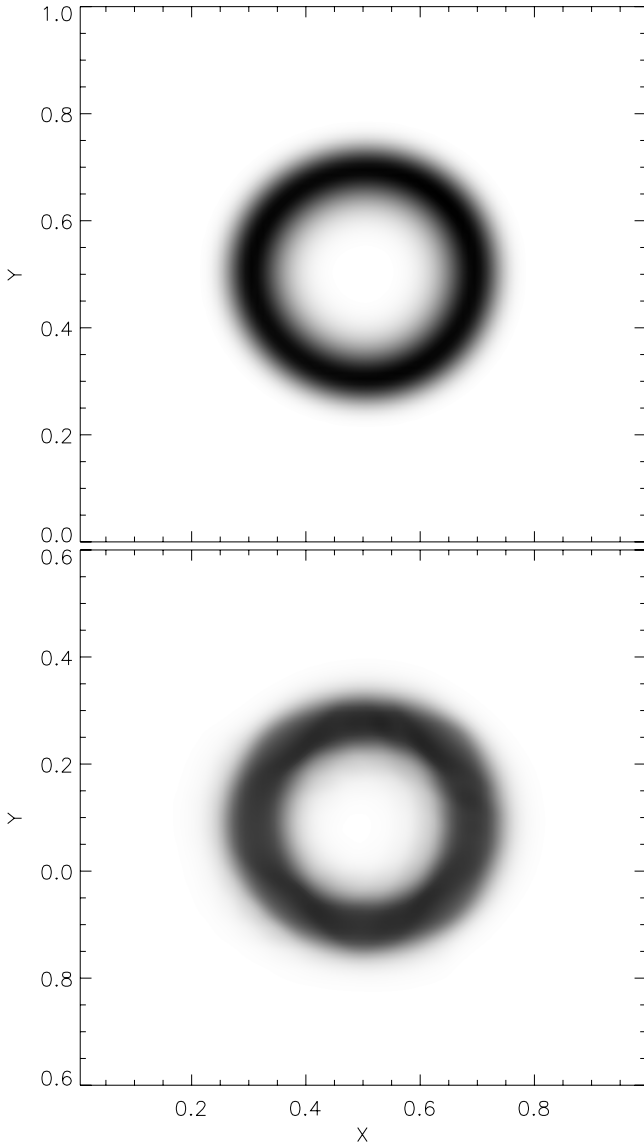


Fig. 2. The magnitude of the magnetic field shown at $t = 0$ (*top*) and at $t = 1.155$ (*bottom*) when the cylinder has moved one box length along the x -axis, and roughly half way along the y -axis. The lower contour plot has been centered to make a comparison easier.

of the resolution capability of the code, which exaggerates the effect of the grid.

11.3. The Orszag-Tang test

The problem was first studied by Orszag & Tang (1979). Since then solutions to the problem have been extensively compared in numerical MHD simulations. This provides a good way to compare codes.

The Orszag-Tang (O-T) vortex is a model problem for testing the transition to supersonic 2D MHD turbulence. Thus, the problem tests how robust the code is at handling the formation of MHD shocks and shock-shock interactions. The problem can also provide some quantitative estimates of how significant magnetic monopoles affect the numerical solutions, testing the $\nabla \cdot \mathbf{B} = 0$ condition. Finally, as mentioned, the problem is a very common test of numerical MHD codes in two dimensions, and has been used in many previous studies.

The set up is the following: the domain is 2D and goes from $0 \leq x \leq 1$ and $0 \leq y \leq 1$. The boundary conditions are periodic. The density $\rho = 1$, the pressure $P = 1/\gamma$ and $\gamma = 5/3$ everywhere. Note that this choice gives a sound speed of $c_s = \sqrt{\gamma P/\rho} = 1$. The initial velocities are periodic with:

$$u_x = -\sin(2\pi y), \quad (38)$$

$$u_y = \sin(2\pi x). \quad (39)$$

The magnetic field is initialized using a periodic vector potential defined at zone corners:

$$A_z = B_0 \left[\frac{\cos(4\pi x)}{4\pi} + \frac{\cos(2\pi y)}{2\pi} \right]. \quad (40)$$

with $B_0 = \sqrt{1/(4\pi)}$. Face-centered magnetic fields are computed using $\mathbf{B} = \nabla \times \mathbf{A}_z$ to guarantee $\nabla \cdot \mathbf{B} = 0$ initially. This gives:

$$B_x = -B_0 \sin(2\pi y), \quad (41)$$

$$B_y = B_0 \sin(4\pi x). \quad (42)$$

We have run a number of simulations with different grid resolution, diffusion and quench parameters. We observed good evolution of the shocks in all runs when we used diffusion parameters set to $\nu_1 > 0.02$, $\nu_2 > 0.2$ and $\nu_3 > 0.2$ in agreement with the previous tests. However, the shocks will be smoother when increasing the diffusion or when decreasing the resolution. Figure 3 is created to be directly compared with Fig. 3 of Ryu et al. (1998) who use a upwind, total variation diminishing code, and even though there are small differences it is hard to tell which code is superior in spite of the two codes being fundamentally different.

11.4. Test of chromospheric radiation approximation

There are no analytical tests that can be used to test our recipes of the chromospheric radiation described in Sect. 8.2. The best we can do is to make a comparison with a simulation where the approximated processes have been calculated in detail. For this purpose we use 1D radiation hydrodynamic simulations calculated with the RADYN code, see e.g., Carlsson & Stein (1995, 2002). This code solves the one-dimensional equations of mass, momentum, energy, and charge conservation together with the non-LTE radiative transfer and population rate equations, implicitly on an adaptive mesh.

Since we have used such a simulation to determine the free parameters in our recipes, we use a simulation with a different velocity field for the test. Figure 4 shows the comparison of the radiative cooling for one timestep in the RADYN simulation with the cooling calculated with the *Bifrost* recipes. The timestep shown has a strong wave that is close to steepening into a shock at $\lg(m_c) = -4.2$ (height of 1.2 Mm). The maximum temperature at the wave crest is 7000 K. Above the wave the temperature rises rapidly into the corona. At $\lg(m_c) = -5.5$ the temperature is 9000 K. The *Bifrost* approximations come close to describing the cooling in both hydrogen and calcium except that the cooling is overestimated in hydrogen just below the transition region (left part of the figure). Inspection of the cooling as function of time at a given height shows that the recipe for hydrogen typically overestimates the cooling in the hot phases but does not include heating in the cool phases with the integral over time being close to the RADYN results. For further tests of the chromospheric radiation approximations see Carlsson & Leenaarts (in prep.).

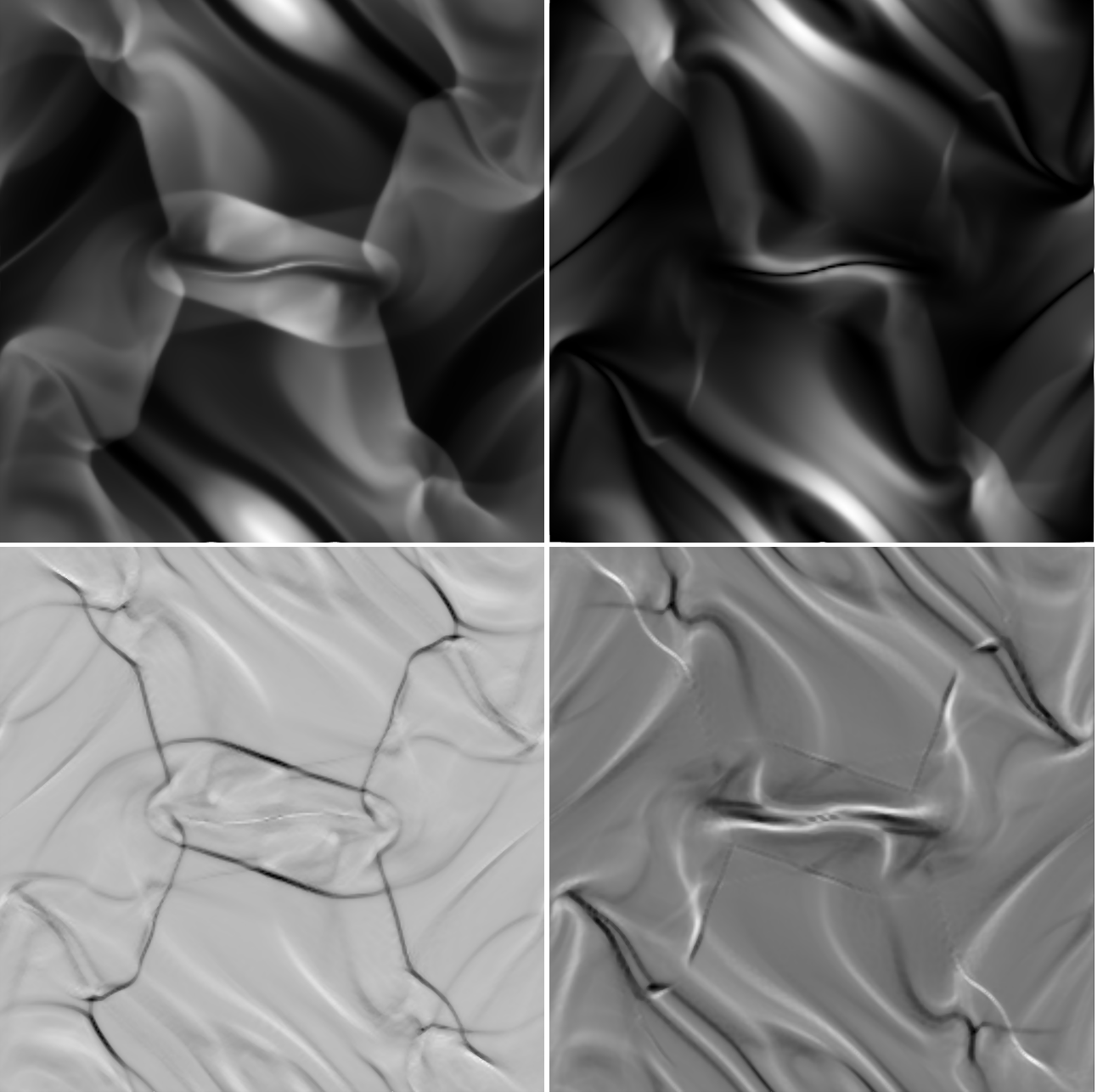


Fig. 3. The result of the Orszag-Tang test on a 256×256 grid, showing gas pressure (*upper left*), magnetic pressure (*upper right*), divergence of the velocity $\nabla \cdot \mathbf{u}$ (*lower left*) and the rotation of the velocity $(\nabla \times \mathbf{u})_z$ (*lower right*). This figure can be directly compared with Fig. 3 of [Ryu et al. \(1998\)](#).

11.5. Full radiative transfer test in an isothermal scattering atmosphere

The radiative transfer equation with coherent scattering has analytical solutions in the case of a static 1D plane-parallel isothermal atmosphere if the photon destruction probability ϵ is constant at all depths (e.g., [Rybicki & Lightman 1979](#)). The anisotropy of the radiation field needs to be restricted to linear dependence on the cosine of the zenith angle, $\mu = \cos \theta$, making the Eddington approximation exact and reducing the problem to solving a second-order equation for the mean intensity J . This setup is also known as the “two-stream” approximation. As second-order Gauss-Legendre quadrature yields an exact representation of integrals over a linear polynomial, a numerical result for J becomes directly comparable to an analytical solution (cf. [Trujillo Bueno & Fabiani Bendicho 1995](#)).

We set the radiative transfer solver to reproduce the mean radiation field

$$J_{\text{an}}(\tau) = \left[1 - \frac{e^{-\sqrt{3}\epsilon\tau}}{1 + \sqrt{\epsilon}} \right] B, \quad (43)$$

with the source function

$$S(\tau) = \left[1 - (1 - \sqrt{\epsilon}) e^{-\sqrt{3}\epsilon\tau} \right] B, \quad (44)$$

where τ is the optical depth in the atmosphere, ϵ is the photon destruction probability and B is the Planck function. The full radiative transfer module of the *Bifrost* code operates on a static 3D atmosphere with a resolution of $50 \times 50 \times 120$ grid points and with a horizontally homogeneous isothermal stratification; other

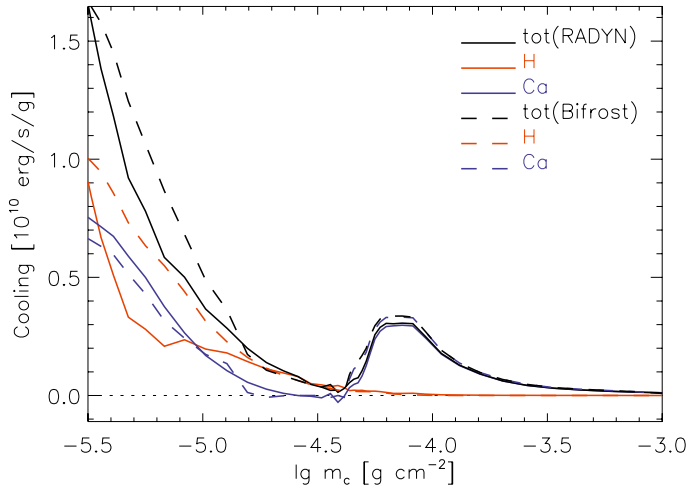


Fig. 4. Radiative cooling as function of column mass for a detailed 1D simulation with the RADYN code (solid) and with the *Bifrost* recipes for chromospheric radiation (dashed). Total cooling (black) and the contributions from hydrogen lines and the Lyman continuum (red) and lines from Ca II (blue).

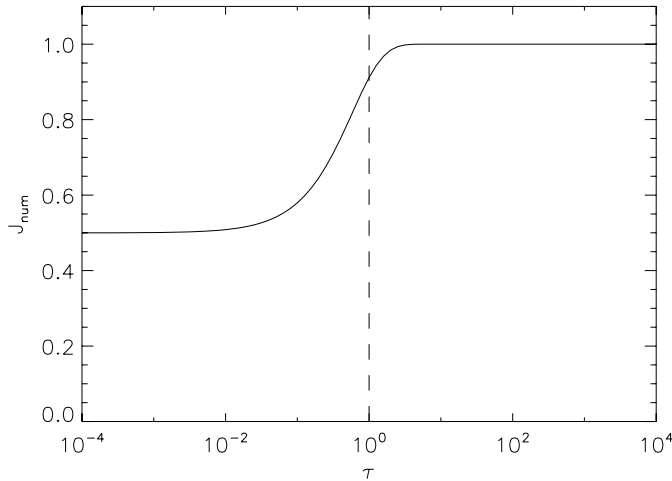


Fig. 5. Numerical solution for the mean intensity J in LTE ($\epsilon = 1.0$) as a function of optical depth τ . The dashed line at $\tau = 1$ marks the optical surface.

physics modules are not used during the computation. The interpolation algorithms needed for radiative transfer in 3D simulations are validated separately with the searchlight test described in Hayek et al. (2010). Optical depths are preset on a grid that is equidistant in $\log_{10} \tau$ with 10 grid points per decade. Specific intensities are computed at $\mu = \pm 1/\sqrt{3}$ and arbitrary azimuth angles ϕ . The radiative transfer code uses double precision arithmetic to handle strong scattering at large optical depths, which appears due to the constancy of ϵ in the atmosphere. The Planck function is set to $B = 1.0$ in arbitrary units for all computations; it is also used as a first-guess source function for the solver.

In the LTE case ($\epsilon = 1$), the solver delivers $I^+(\tau) = 1.0$ for outgoing intensities at $\mu = 1/\sqrt{3}$ and $I^-(\tau) = 1.0 - e^{-\sqrt{3}\tau}$ for ingoing intensities at $\mu = -1/\sqrt{3}$ (see Fig. 5). The numerical solution is equivalent to the analytical solution given by Eq. (43), as the Gauss-Seidel solver uses the formal solution of the radiative transfer equation, which leads to identical expressions. In the scattering case, photon destruction probabilities assume

values between $\epsilon = 10^{-1}$ (moderate scattering) and $\epsilon = 10^{-6}$ (strong scattering), decreasing by factors of 10. The left column of Fig. 6 shows the numerical results for J . The mean intensity near the surface decreases for smaller ϵ due to outward photon losses, and the thermalization depth $\tau_{\text{therm}} = 1/\sqrt{\epsilon}$, where the radiation field is completely thermalized ($J \approx B$), moves deeper into the atmosphere (dot-dashed line). At the smallest optical depths, the numerical solution delivers $J_{\text{num}}(\tau = 10^{-4}) \approx \sqrt{\epsilon}$ for small ϵ , as expected from Eq. (43).

The center column of Fig. 6 shows the optical depth-dependence of the total error of the numerical solution as the relative deviation between J_{num} and J_{an} ,

$$\Delta J_{\text{rel}}(\tau) = \frac{J_{\text{num}}(\tau) - J_{\text{an}}(\tau)}{J_{\text{an}}(\tau)}. \quad (45)$$

At large optical depths $\tau \gg 1$, radiative transfer is local and the total error depends on the source function gradient through the discretization error of the logarithmic optical depth grid. Using Eq. (44), one obtains the variation ΔS of the source function between adjacent grid points with constant spacing $\Delta \log \tau$,

$$\begin{aligned} \Delta S &\approx \tau \frac{dS}{d\tau} \Delta \log \tau \\ &= (1 - \sqrt{\epsilon}) B \left(\sqrt{3\epsilon} \tau e^{-\sqrt{3\epsilon}\tau} \right) \Delta \log \tau \\ &= (1 - \sqrt{\epsilon}) B \left(\sqrt{3} (\tau/\tau_{\text{therm}}) e^{-\sqrt{3}\tau/\tau_{\text{therm}}} \right) \Delta \log \tau. \end{aligned} \quad (46)$$

In the LTE case, $\Delta S = 0$ everywhere in the atmosphere since $\epsilon = 1.0$ and $S = B = 1.0$, independent of $\Delta \log \tau$, and the accuracy of the numerical solution does not depend on the grid resolution. For $\epsilon < 1.0$ and at optical depths $\tau \gtrsim \tau_{\text{therm}}$, the radiation field is entirely thermalized ($J_{\text{num}} \approx B = 1.0$), and ΔJ_{rel} vanishes since $\Delta S \rightarrow 0.0$. The source function starts to decrease quickly through the decreasing mean intensity near τ_{therm} due to outward photon losses, causing a sharp increase in the error of the numerical solution. ΔJ_{rel} peaks in the translucent zone as ΔS is largest between $1.0 < \tau < \tau_{\text{therm}}$. The magnitude of the peak grows with decreasing ϵ through the $(1 - \sqrt{\epsilon})$ factor in Eq. (46); an upper limit is reached with $\epsilon \rightarrow 0.0$. It also follows from Eq. (46) that $\max(\Delta J_{\text{rel}})$ scales with the grid resolution. Further up in the atmosphere, the error decreases again through the finer grid spacing. At optical depths $\tau \ll \tau_{\text{therm}}$, the local discretization error becomes negligible due to the vanishing $\Delta S \rightarrow 0.0$. However, as I^+ and I^- decouple from the local source function and the radiation field becomes anisotropic, the error becomes independent from the source function gradient. ΔJ_{rel} is dominated by the propagated error of outgoing radiation I^+ from deeper layers and is therefore constant.

The right column of Fig. 6 shows the convergence speed of the numerical result to the analytical solution, measured through the maximum relative deviation $\max(\Delta J_{\text{rel}})$ in the atmosphere (see Eq. (45)). Dashed lines represent computations where Gauss-Seidel corrections for the source function were applied only during upsweeps, while solid lines show the results when both upsweeps and downsweeps were corrected, which increases convergence speeds by about a factor of two (see Trujillo Bueno & Fabiani Bendicho 1995). Convergence is significantly slower when the thermalization depth moves to deep, very optically thick layers for small photon destruction probabilities, requiring about 250 iterations in the strongest scattering case. We find a numerical error of $\max(\Delta J_{\text{rel}}) \approx 3.7 \times 10^{-3}$ at $\epsilon = 10^{-6}$, similar to the error quoted in Trujillo Bueno & Fabiani Bendicho (1995).

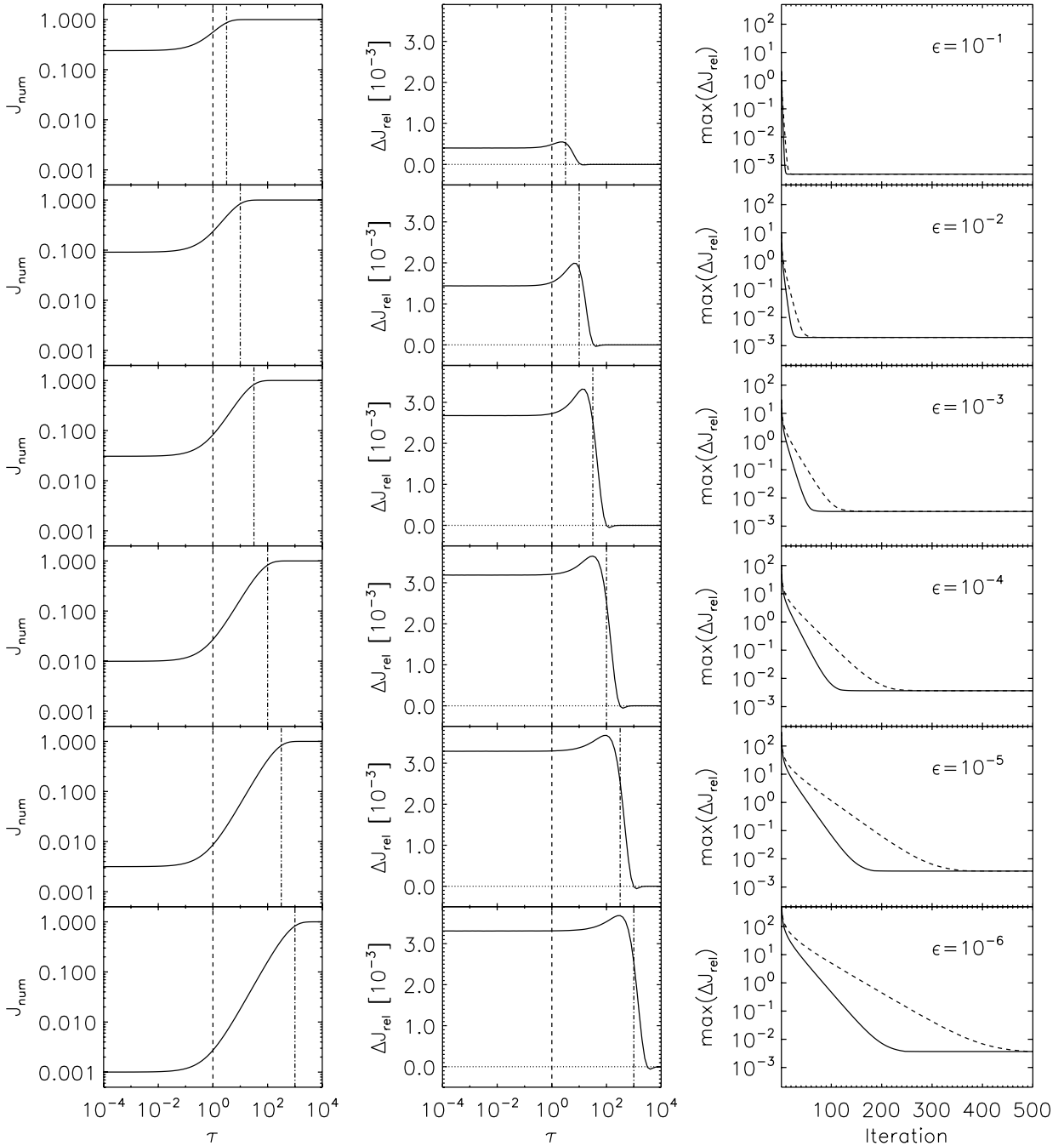


Fig. 6. Numerical solution for the mean intensity J_{num} as a function of optical depth τ (left column), relative deviation ΔJ_{rel} from the analytical solution as a function of optical depth τ (center column), and maximum relative deviation $\max(\Delta J_{\text{rel}})$ as a function of iteration count (right column). The photon destruction probability ϵ ranges from 10^{-1} (top row) to 10^{-6} (bottom row). The dashed and dot-dashed lines in the left column and center column mark the optical surface at $\tau = 1$ and the thermalization depth τ_{therm} . The dotted line in the center column indicates zero deviation. Dashed lines in the right column show the convergence speed for Gauss-Seidel corrections applied only during upsweeps, solid lines show the convergence speed for corrections applied during both upsweeps and downsweeps.

11.6. Hydrogen ionization test

Non-equilibrium hydrogen ionization generally produces atomic level populations that are neither in statistical equilibrium nor in Saha-Boltzmann equilibrium (LTE). However, in the case of simulations of the solar atmosphere spanning from the upper convection zone to the lower corona, two limiting cases are produced: near the convection zone boundary the level populations obey LTE statistics; near the coronal boundary the populations obey statistical equilibrium because of the fast transition rates

there. Both limiting cases are reproduced in statistical equilibrium non-LTE radiative transfer codes. We will therefore compare the results from a *Bifrost* run with those of a statistical equilibrium code *in those two limits*. In the intermediate regime the statistical equilibrium is not valid and there will be large differences between the two codes.

We took a snapshot from a 2D simulation of the solar atmosphere that included the hydrogen ionization module (HION). This simulation had a grid size of 512×325 points, spanning

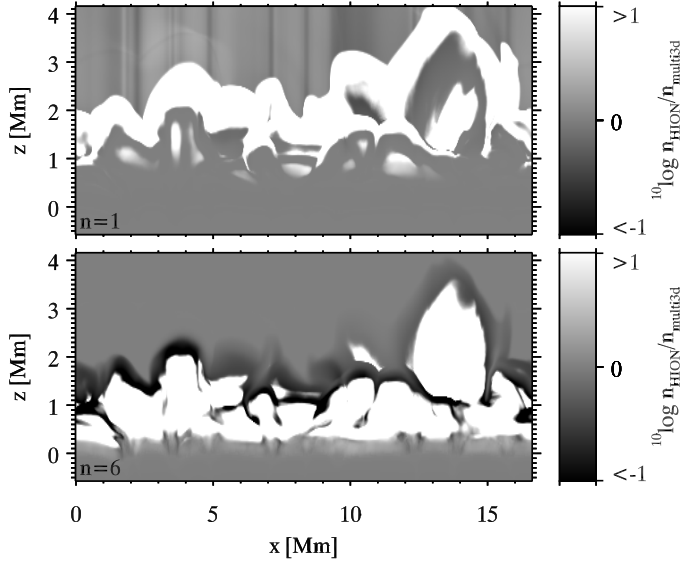


Fig. 7. Ratio of the hydrogen level populations computed with the hydrogen ionization module to the populations from a detailed computation assuming statistical equilibrium. *Upper panel:* atomic hydrogen ground level; *lower panel:* proton density.

from 1.5 Mm below the photosphere to 14 Mm above it. It included a weak magnetic field (average unsigned flux density in the photosphere of 0.3 G), thermal conduction and full, chromospheric and optically thin radiative transfer. The convection zone boundary was open, the coronal boundary used the methods of characteristics.

We then computed the statistical equilibrium values of the hydrogen level populations from this snapshot using the radiative transfer code Multi3D treating each column as a 1D plane-parallel atmosphere. As input we took the snapshot geometry, the mass density, the electron density and the temperature. We set the velocity field to zero. As model atom we used a 5-level plus continuum hydrogen atom with all radiative transitions from the ground level put in radiative equilibrium, as is assumed in the HION module. We treated all remaining lines assuming complete redistribution.

Figure 7 shows a comparison of the level populations obtained from the *Bifrost* and the Multi3D computation. Both the $n = 1$ and the proton density are equal in the convection zone (below $z = 0$ Mm), showing that *Bifrost* correctly reproduces LTE populations there. The proton densities in the corona are also equal for *Bifrost* and Multi3D. The $n = 1$ populations in the corona are not identical.

However, the differences are small, the populations are always within a factor of two of each other. The exact value of the coronal population density depends on the radiation field, and the simple HION assumption of a coronal radiation field that is constant in space does not capture the small-scale variations present in the Multi3D computation as indicated by the vertical stripes in the upper panel of Fig. 7. This striping is caused by variations in the atmospheric quantities down in the photosphere and chromosphere where the coronal radiation field is set.

11.7. Boundary condition test

The full 3D MHD equations allow a wide variety of wave modes that cross the boundaries at different angles depending on their origin and on the topology and strength of the magnetic field. A comprehensive test of all these modes falls outside the scope of

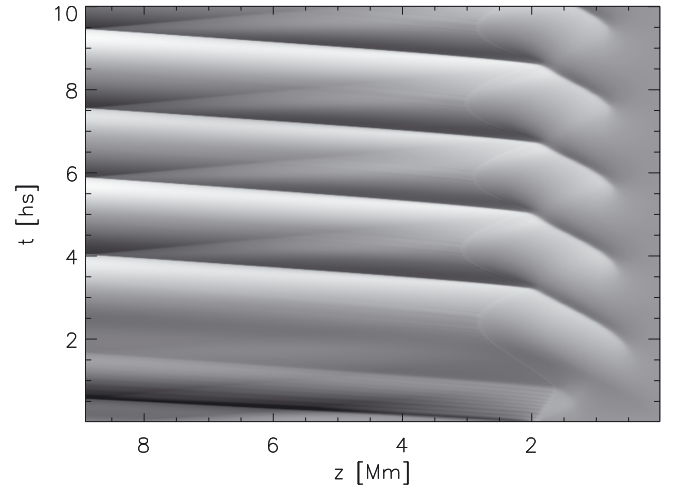


Fig. 8. Vertical velocity u_z in 1D test model. The color scale is set to span $\pm 30 \text{ km s}^{-1}$ with black representing upflow and white downflow. The chromosphere adjusts its structure, which initially is not quite in hydrostatic equilibrium, by emitting acoustic waves at the cut-off frequency. These waves are initially strong enough to form shocks, as here during the first 1000 s of the simulation.

this paper, but we will present a couple of tests to show the range of boundary condition behavior we have observed. The examples shown contain the full set of physics that *Bifrost* supports and thus are meant to represent the “typical” production run the code is meant for, though with simpler geometries. Both of the tests shown were at the same time used to test the thermal conduction module as described in Sect. 11.8.

The first test is a 1D model containing a photosphere, chromosphere, and corona that has a vertical extent $z = 9$ Mm, which is discretized on a grid containing 256 points with $\Delta z = 35$ km throughout the computational domain. The model has solar gravity and a weak (0.1 G) vertical magnetic field. Radiative losses are included through the optically thin and chromospheric approximations, but the full radiative transfer module is turned off. Thermal conduction is included and the corona is kept heated by maintaining the temperature at the upper boundary at 1.1 MK. The initial model is not perfectly in hydrostatic nor energetic equilibrium and the atmosphere responds by launching a set of acoustic disturbances at roughly the acoustic cut off frequency of 3 min. These are initially of high amplitude, forming shock waves, but at later times damping out to much lower amplitudes and forming linear waves. In Fig. 8 we plot the vertical velocity u_z as a function of height and time. Evidently, acoustic waves originate in the lower to middle chromosphere near $z \approx 500$ km and propagate upward steepening into shocks. At roughly 2 Mm they encounter the transition region before entering the corona proper.

It is clear that while the boundary seems fairly well behaved, the strongest shocks do lead to some reflection, which we have measured to be of order 5% or less in terms of reflected energy. There may be several reasons for such reflection: one is that high amplitude waves seem in general more difficult to transmit through a boundary. In addition, setting a hot plate as a temperature boundary condition will give some reflection as the temperature is forced to a given value.

The second experiment uses the same background atmosphere as in the 1D test described above, but we have expanded to span 16.5 Mm in the horizontal direction forming a 2D atmosphere that contains a magnetic field of greater complexity: this magnetic field is formed by inserting positive and negative

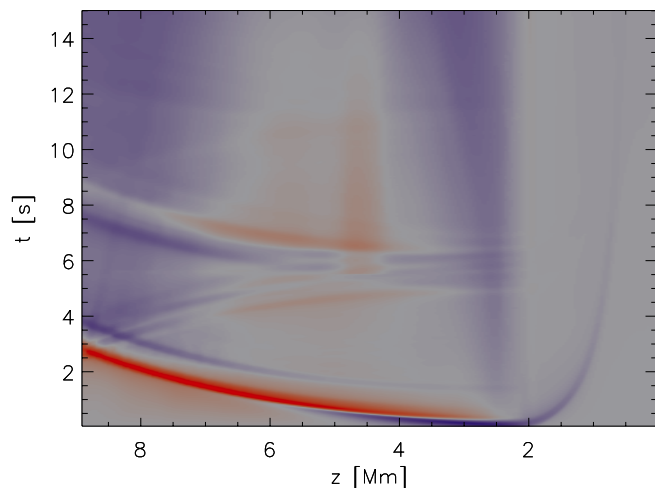


Fig. 9. Vertical velocity u_z in 2D test model, red is downflow and blue upflow with a color scale set to $\pm 50 \text{ km s}^{-1}$, as a function of time and height at the position $x = 5 \text{ Mm}$ in our 2D test model (see text). The disturbance clearly visible is a fast mode wave that originates due to a slight imbalance in the Lorentz force in the transition region in the initial state. The wave propagates fairly cleanly through the upper boundary, with a reflection of some 5%.

magnetic polarities of $\pm 1000 \text{ G}$ strength at the bottom boundary, spanning 1 Mm and centered at $x = 2.5 \text{ Mm}$ and $x = 13.5 \text{ Mm}$ respectively, and then computing the potential field that arises from this distribution (see Fig. 11). Note that the field is quite strong and that plasma beta is less than one in most of the modeled domain. Again there is a slight hydrostatic imbalance in the initial state and a transient wave is generated, in this case in the form of a fast mode wave originating close to the transition region. The Alfvén speed is quite high, some 9000 km s^{-1} near the transition region, falling to 1000 km s^{-1} at the upper boundary, since much of the field has closed at lower heights in the atmosphere. In comparison, the speed of sound lies in the range 100 km s^{-1} to 160 km s^{-1} in the corona. Thus, the generated fast mode, traveling essentially at the Alfvén speed, propagates to the upper boundary very quickly, using only 3 s to cover the 7 Mm from the transition region to the upper boundary. The transient wave’s amplitude on leaving the upper boundary depends on location but lies between -70 km s^{-1} and 150 km s^{-1} . In Fig. 9 we plot the vertical velocity as a function of height and time at the representative horizontal position $x = 5 \text{ Mm}$. As in the previous example we do find a certain amount of reflection, but again find it to be energetically unimportant, at less than 5% of the energy contained in the original wave at all horizontal locations. Note that the reflected wave is re-reflected off the transition region and that this second wave seems very nicely transmitted through the upper boundary.

Setting boundary conditions for the type of models discussed here is a compromise between long term stability and the best possible transmission of outwardly propagating modes. In the examples presented above we have attempted to show the results where stability is the most important factor, which is attained by limiting the velocity amplitude allowed in the ghost zones, as would be typical for a production run that aims to model the outer solar atmosphere for a duration of order several thousand seconds. This safety factor increases the amount of reflection seen, a compromise that allows us to run simulations for long periods of time.

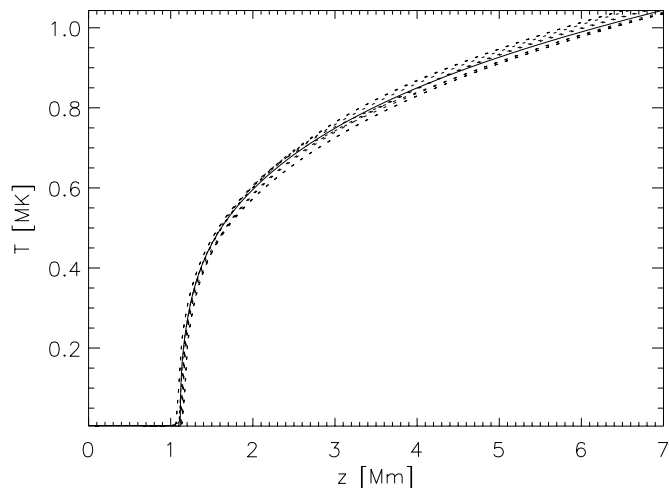


Fig. 10. Comparison of high resolution 1D model (solid line $\Delta z \geq 80 \text{ m}$) with *Bifrost* test run with $\Delta z = 35 \text{ km}$ (dashed lines). The models are set up to have a temperature maximum of 1.1 MK at the upper boundary such that conduction dominates the energetics of the atmosphere in the corona and transition region. The results from the *Bifrost* run are taken from the same model as used in Fig. 8, but at much later times ($t > 8000 \text{ s}$) when acoustic perturbations are largely damped and the amplitude of transition region motion is less than 300 km .

11.8. Thermal conduction test

In order to test the thermal conduction module we present the temperature structure that arises in the two examples presented above in Sect. 11.7. In the first we consider a 1D model with vertical magnetic field of the upper photosphere, chromosphere and corona in which the chromosphere is slowly relaxing by shedding acoustic waves at the cut-off frequency. The upper boundary temperature is set to 1.1 MK and conduction is the dominant term in the energy balance in the corona and transition region, down to temperatures of roughly 10 kK occurring at a height of $z = 1.2 \text{ Mm}$. After 3 h solar time the amplitude of the acoustic waves being generated is much reduced, and the location of the transition region moves by less than 300 km during a full wave period. In Fig. 10 we plot the temperature as a function of height for several timesteps during a wave period 3 h solar time after the experiment began. Also plotted is the temperature profile that arises from a much higher resolution 1D model computed with the TTRANZ code (Hansteen 1993) with the same upper boundary temperature. The latter model uses an adaptive grid (Dorfi & Drury 1987) that concentrates grid points in regions of strong gradients; in the present model the grid size is of order 80 m in the lower transition region. The temperature profiles in the two models are quite similar even though the grid spacing is much larger in the *Bifrost* run. Of course, not all aspects of the relevant physics can be reproduced with $\Delta z = 35 \text{ km}$ in the high temperature gradient environment of the lower transition region. We find sudden large temperature changes in transition region grid points as the location of the transition region changes due to the passage of acoustic waves. These temperature changes become sources of (higher frequency) acoustic waves with amplitudes of order some 100 m/s that can be discerned on close inspection of Fig. 8. This artifact first disappears when the grid size is set to $\Delta z \leq 15 \text{ km}$ (for typical coronal temperatures $\lesssim 2 \text{ MK}$).

In the second example we consider the case of rapid heating of coronal plasma in a magnetized atmosphere and follow how thermal conduction leads the deposited energy along the

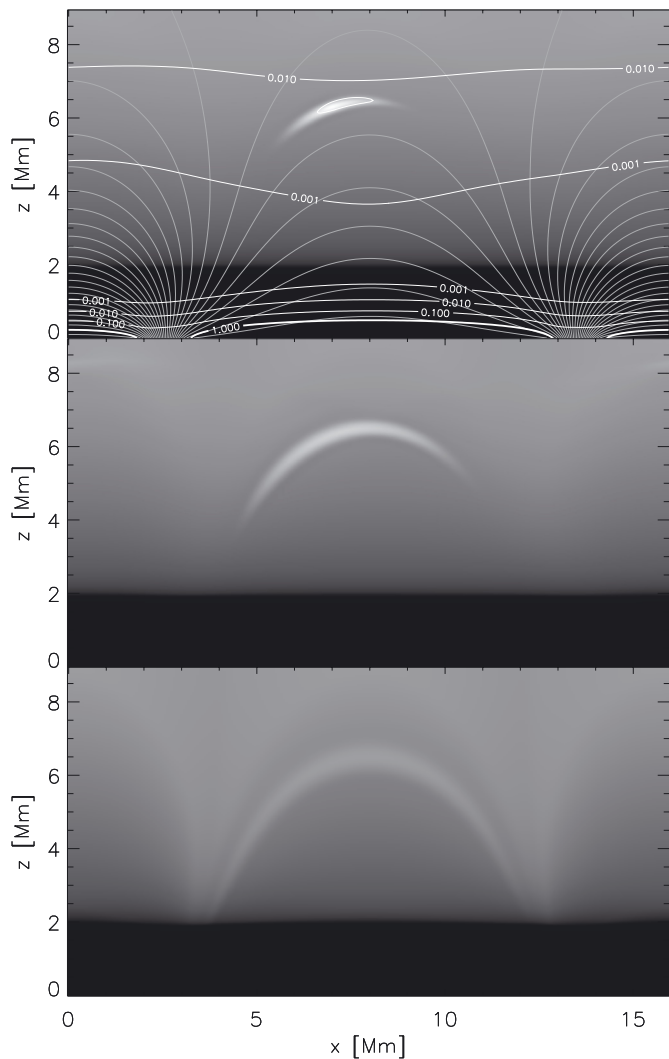


Fig. 11. Time evolution of 2D model which is heated for 1 s at position $x = 7$ Mm, $z = 6.3$ Mm with 50 J/m^3 over a region spanning $100 \times 100 \text{ km}^2$. The panels show the temperature at $t = 0.6$ s (*top*) when the maximum temperature is 2.25 MK, at $t = 1.8$ s (*middle*) when the maximum temperature is 1.63 MK, and at $t = 10$ s (*bottom*) where the maximum temperature is 1.16 MK. The temperature increases rapidly in the heated region, reaching 2.4 MK at 1 s, and plasma is heated by thermal conduction along the field as the plasma cools. Note that the upper boundary is set to have zero temperature gradient, so the entire atmosphere cools as well. Magnetic field lines are indicated with thin grey contours and contours of constant β are shown with white numbered lines.

magnetic field lines. We use the same atmosphere as described in the 2D case given in the boundary condition test above. During the first second of the model run we deposit 50 J/m^3 over a region spanning $100 \times 100 \text{ km}^2$ (an amount of energy equivalent to a large nano-flare) at location $x = 7$ Mm, $z = 6.3$ Mm, after which we allow the atmosphere to cool. At the upper boundary we set a zero temperature gradient boundary condition.

Figure 11 shows the temporal evolution of this model with snapshots taken at 0.6 s, 1.8 s, and 10 s. The deposited energy rapidly heats the coronal plasma, originally at 1 MK, to 2.4 MK at $t = 1$ s. The heat is efficiently conducted away from the site of energy deposition and has already after 0.6 s increased temperatures in a region several thousand kilometers along the magnetic field. After energy deposition ends the maximum temperature

in the heated region decreases while the region itself continues to expand along the magnetic field towards the transition region and chromosphere. At $t = 10$ s the heated region has cooled quite a bit, but is still hotter than the ambient atmosphere and has spread out to form a loop-like structure. Note that the ambient atmosphere is also cooling, the portions of the atmosphere connected via the magnetic field to the upper boundary cooling least rapidly. The width of the heated region is fairly constant, but some spreading perpendicular to the magnetic field is evident in the last frame shown at $t = 10$ s.

12. Parallelization

Bifrost was written to be massively parallel. It employs the Message Passing Interface (MPI), because it is very well developed and exists on almost all super computers. There are a number of other options for parallelization, including OpenMP and the `thread()` mechanism included in the different variations of the C programming language, but we found MPI to suit our needs the best.

MHD on a regular grid is trivial to parallelize by splitting the computational grid into subdomains and distribute one subdomain to each node. In the case of *Bifrost*, computing the derivatives or interpolating the variables uses a stencil of 6 grid points. The two/three gridpoints nearest the edge of a subdomain, then relies on data that is outside the subdomain belonging to the local node, but instead belongs to the neighboring subdomain. The way this data is acquired most efficiently, depends on the problem. The two possible solutions are to communicate the needed data to neighboring nodes every time it is needed, and the other is to supply each node with a number of “ghost cells” around its allocated subdomain so that the whole stencil used in for instance a derivative operator is present at the node that needs it. The extreme solution would be for each node to have a very large number of ghost cells, in the most extreme case the complete computational grid, and therefore never need to communicate, but that would make the code non-parallel. Choosing to include ghost cells around each sub domain makes it necessary to do more computations, since the ghost cells are copies of cells belonging to neighboring nodes, but makes it possible to do less communication, as the ghost cells can be filled with the correct values from the neighboring nodes less often. The number of ghost cells chosen is therefore influenced by the relative importance of the communication speed, the computation speed and the numerical scheme of the code.

We have chosen to keep the communication between nodes to a minimum at the expense of doing more computations. Five ghost cells makes it possible to do both a derivative and an interpolation along the same coordinate direction without having to communicate with neighboring nodes. That choice was made, because it allows us to get good performance even on machines with relatively slow internode communication speeds. It is very hard to quantify exactly when this choice is wise and when not, because so many parameters enter the problem. For instance, a global minimum operation scales with the number of nodes used and the communication speed, a simple neighbor communication will depend both on the communication time, the physical setup of the nodes and the switches that connect them etc. But it is worth noting that typical communication times in large systems are of millisecond scale, while the frequency of the cpus are in the gigahertz range. So from this very simple argument, it should be possible for a CPU to do roughly 10^3 computations (multiplications, additions), in the time it takes to do one communication.

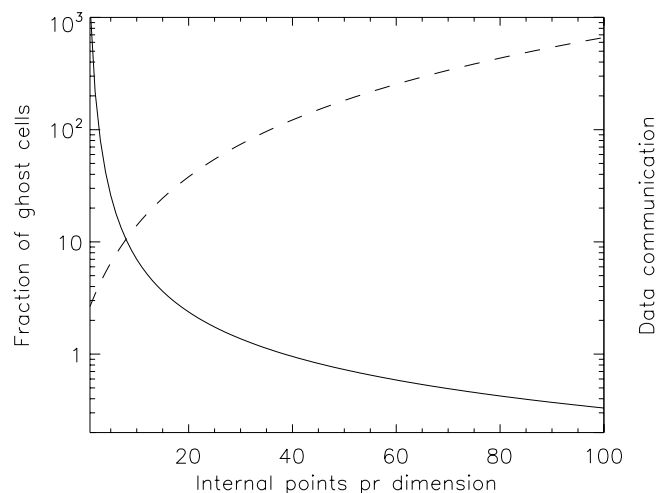


Fig. 12. The relative number of computed gridpoints (solid) and the total data needed to be communicated (dashed) as a function of the number of internal grid points per dimension for a 3D cubic sub domain.

Figure 12 shows how the data communicated increases with the number of internal gridpoints per node, and how the relative increase in computation decreases with the number of internal grid points. Both of the curves in Fig. 12 do not take into account communication speed or computation speed, so both curves can be shifted up and down the y -axis when applying them to a specific system. Communication can be split into initialization and actual communication of the data. In general the initialization takes very long compared with the actual transmission of the data, so there is a large offset in the communication time, but the MHD part of *Bifrost* only communicates with neighboring subdomains, so this offset depends on the dimensionality of the problem, and not on the number of cpus or internal points per subdomain. Since *Bifrost* was developed to handle as large computations as possible, the number of internal grid points will be rather high, so the rapid divergence between the two curves makes it plausible that doing the extra computations is the correct choice.

Scaling can be measured in two different ways, called strong and weak scaling. Strong scaling uses a set problem size and then timing measurements are made for different number of nodes, while weak scaling uses a set problem size per node, which means that when the number of nodes increases, the problem size also increases. Strong scaling is mainly a test of the communication overhead. If the communication takes up a constant amount of time, it should take a relatively larger and larger part of the run time as the number of cpus goes up. Weak scaling gives a measure of how well the code handles larger and larger problem sizes and number of cpus without being influenced by the raw communication time.

Both strong and weak scaling tests of *Bifrost* have been performed on a number of computer architectures. We will here report on the results from a Cray XT4, with each node containing one quad-core AMD Opteron 2.3 GHz cpu and with a proprietary Cray Seastar2 interconnect, made available to us by the Partnership for Advanced Computing in Europe (PRACE), and from a Silicon Graphics ICE system, with each node containing two quad-core Intel Nehalem-EP 2.93 GHz cpus with Infiniband interconnect, located at NASA Advanced Supercomputing Division.

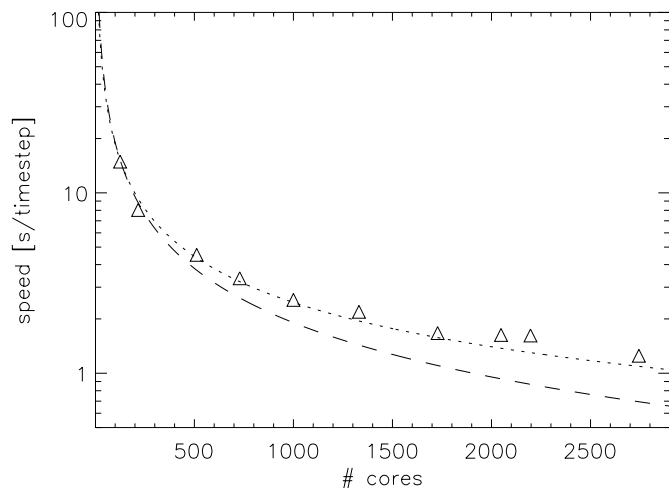


Fig. 13. Scaling when running a pure MHD case with 500^3 gridpoints on a Cray XT4 system with different number of cores (triangles), the theoretical scaling curve (dashed) and the theoretical scaling curve taking into account ghost cells (dotted).

Strong scaling tests were run with just the simplest configuration: pure MHD on an uneven staggered mesh, with an ideal EOS, without radiation, conduction or any other advanced physics or boundary conditions. The timing is performed on the Cray XT4 in such a way that the number of internal points per dimension is 30 or more, so the lower end of Fig. 12 is never reached. Such a test will show if there are communication bottlenecks in the code which will significantly slow the code down when running on a large number of nodes and when there is too large a penalty due to the use of ghost cells. Figure 13 shows the scaling results for a 500^3 gridpoint run, and as the number of cores increases the relative amount of grid cells that are ghost cells increases, and consequently the code uses more and more time calculating the ghost cells compared to the internal cells. If the relative increase in ghost cells is taken into account, *Bifrost* scales very well. Figure 13 also shows that if the number of cells per dimension for each core becomes less than 50, then efficiency of the code has dropped by about 35% compared to perfect scaling, and consequently, each core should not get a computational sub domain which is smaller than 50 grid points on a side.

The weak scaling tests were performed with a production setup of a solar simulation extending from the convection zone 2.5 Mm below the photosphere to the corona 14.5 Mm above the photosphere. The horizontal extent was 24×24 Mm². When the full radiation module was switched on in some of the weak scaling tests, scattering was included and the radiation field was described with 26 rays. The bottom boundary was transparent, the top boundary used the method of characteristics and horizontal boundaries were periodic.

In theory *Bifrost* should be able to handle weak scaling very well, when only modules that use local data are used. If modules like full radiative transfer and Spitzer conductivity are included, the scaling is expected to drop below the theoretically best scaling, because these modules use non-local data and provide non-local results. When including modules using non-local data, the drop in efficiency with larger number of cpus depends on the choice of, and implementation of, the solver as well as communication time. For our tests the problem size was adjusted such that each core had a subdomain of $64 \times 64 \times 64$ internal grid-points.

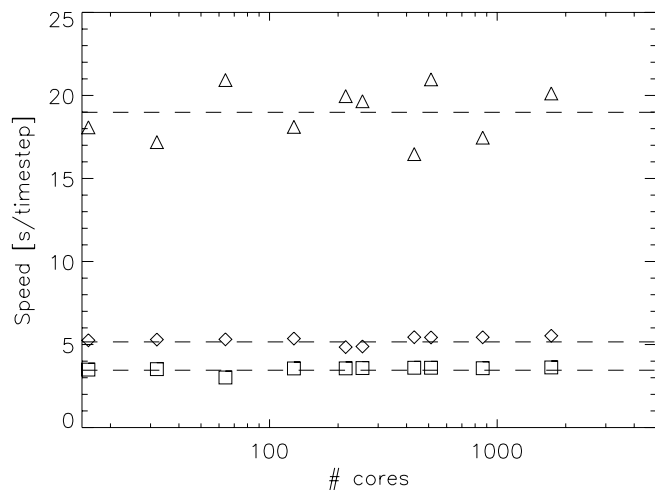


Fig. 14. Weak scaling results for *Bifrost* on a Cray XT4 system when running MHD with a realistic EOS and chromospheric radiation (squares), MHD with a realistic EOS, chromospheric radiation and Spitzer heat conduction (diamonds) and MHD with a realistic EOS, chromospheric radiation, Spitzer heat conduction and full radiative transport (triangles). Dashed lines show the average timing on the runs up to 256 cores.

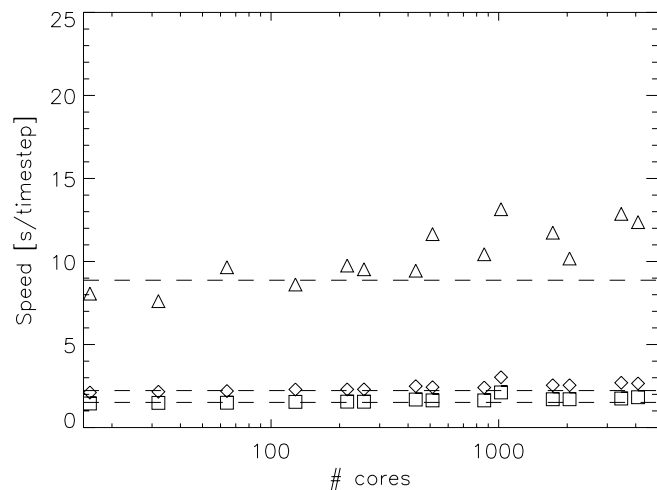


Fig. 15. Weak scaling results for *Bifrost* on a Silicon Graphics ICE system when running MHD with a realistic EOS and chromospheric radiation (squares), MHD with a realistic EOS, chromospheric radiation and Spitzer heat conduction (diamonds) and MHD with a realistic EOS, chromospheric radiation, Spitzer heat conduction and full radiative transport (triangles). Dashed lines show the average timing on the runs up to 256 cores.

The weak scaling results for *Bifrost* on the Cray XT4 are provided in Fig. 14 and show very little dependence on the number of cores for each type of experiment. For the full radiative transfer case it is important to note that the number of iterations for convergence of the scattering radiative transfer problem depends on the change from the previous timestep, the aspect ratio between spacing in the vertical and horizontal directions and the resolution. The large fluctuation in the timings for the full radiative transfer case seen in Fig. 14 is completely caused by the variations in the number of iterations per timestep for the different cases and when this is taken into account the time per timestep does not increase with increased number of cores. Figure 15 shows the same experiments but run on the Silicon Graphics ICE system with Infiniband interconnect. There is again little dependence on the number of cores when the trend of increasing number of iterations with increased resolution for the radiation is taken into account.

Figures 14, 15 also show that both the Spitzer heat conduction and full radiative transfer take up a large fraction of the time. Spitzer heat conduction increases the time by almost 50%, while full radiative transfer increases the computing time by more than a factor of 5. The large computational effort needed for the full radiative transfer is caused by the scattering iterations – typically 3–15 iterations are needed for convergence within one timestep. In practical problems, the full radiative transfer does not have as large a penalty on the timing as it might seem, since the radiative transfer module only needs to be run in a fraction of the timesteps for chromospheric/coronal problems where scattering is important. In a typical solar simulation with 32 km horizontal resolution (the 1728 core points in Figs. 14, 15) the timestep set by the Courant-Friedrich-Levy condition and the Alfvén speed in the corona is 3 ms while the radiation is updated only every 300 ms which means that the radiative transfer increases the computing time only by 2%. For photospheric problems, the radiative timescale is comparable with the hydrodynamic one and the radiative transfer module needs to be called for every timestep. On the other hand, there is no need to iterate

when scattering is unimportant. For such simulations the radiative transfer increases the computing time by about 60% compared with the pure MHD case.

13. Conclusion

The development of numerical methods and computer power has made numerical simulations of stellar atmospheres highly relevant. It is now possible to create observational predictions from advanced realistic numerical simulations, and observations can therefore partially validate the results of the numerical simulations. If such predictions are made and confirmed by observations it is likely that other predictions from the numerical simulations are also correct, making it possible to get much more information about the stellar atmospheres than would be possible through observations alone. To provide as good simulation results as possible, it is necessary to have a highly efficient and parallel numerical code. The tendency for modern supercomputers to be distributed memory systems, makes it extremely important that numerical codes are highly parallel. For pure MHD that is not difficult to attain because MHD is a local process, but it becomes much more complicated for non-local processes or numerical solvers.

The numerical MHD code *Bifrost* has been created to meet the requirements of present supercomputers. It is developed by a group of researchers, post docs and Ph.D. students and provides a simple interface to include further developments in boundary conditions and physical regimes that are not included in the simplified core of the code. Several extension modules are already provided and several more are under development. Results using *Bifrost* have already been published (Hayek et al. 2010; Carlsson et al. 2010; Hansteen et al. 2010; Leenaarts et al. 2011, in press). The pure MHD module of *Bifrost* has been extensively tested through standard tests and has performed well. There are at the moment eight individual modules that have been finished and tested, and several more are under development and testing. The finished modules include a number of equations of state,

radiation transport and thermal conduction and tests of them have been presented and all produce results according to expectations.

Bifrost has been tested for scaling performance. Both strong scaling and weak scaling results are very good on the two systems we have tested on. We would ideally have liked to test on a system where the interconnect is slower than on the Silicon Graphics ICE and Cray XT4 systems to get further knowledge about the bottlenecks a slow interconnect would present for *Bifrost*. The hardware communication architecture plays a role on the scaling behavior of *Bifrost*, but these would most likely be more severe if we had made a choice of doing more communication. Since *Bifrost* is primarily designed to run large simulations, using a large number of cores, we believe we have made the correct prioritization in choosing more computations over communication.

The very good scaling performance and the modules already developed will make it possible to simulate the whole solar atmosphere from the top of the convection zone to the corona with a degree of realism that has not been attained before. It has become more and more clear that the solar atmosphere cannot be split into the traditional separate layers, the photosphere, chromosphere, transition region and corona. The solar atmosphere is one large connected system, and it is necessary to include the whole atmosphere to attain credible results. The consequence is that the code used for such a simulation will have to include the special physics important in each layer, making the numerical code much more complex than a numerical code designed to deal with just one of the layers. *Bifrost* is uniquely qualified for that task.

The relative ease of creating new setups for simulations, inclusion of special physics and boundary conditions, make it possible to use *Bifrost* for detailed solar atmosphere simulations and for stellar atmospheres in general. Several investigations using *Bifrost* have been done or are under way, several of these including PhD students who have been able to use this “state of the art” numerical code with relatively little instruction. There are a number of modules being developed, including a module that can follow the ionization states of heavy elements and a module that introduces a modified Ohm’s law. There is a trend towards computer systems using the large raw floating point performance of Grapical Processing Units (GPUs), and a new parallelization module for such systems is also under development.

The very good scaling performance of *Bifrost* makes it possible to make simulations of stellar atmospheres with a very large resolution while still encompassing a large enough volume to make the simulation realistic, and include a large number of physical effects. The results can be used to predict observational effects, which might earlier have lead to wrong diagnostics of the physical parameters in the solar atmosphere.

Acknowledgements. We would like to thank the referee for very useful comments, that helped us make improvements. We would like to thank Robert F. Stein, Andrew McMurry and Colin Rosenthal for working out the formalism for the characteristic boundary conditions. This research was supported by the Research Council of Norway through the grant “Solar Atmospheric Modelling” and through grants of computing time from the Programme for Supercomputing and through grants SMD-07-0434, SMD-08-0743, SMD-09-1128, SMD-09-1336, and SMD-10-1622 from the High End Computing (HEC) division of NASA. The authors wish to thank PRACE for opportunity to run experiments on HPC centers in Europe. J.L. acknowledges financial support from the European Commission through the SOLAIRE Network (MTRN-CT-2006-035484) and from the Netherlands Organization for Scientific Research (NWO).

Appendix A: Characteristic boundary conditions

The boundary equations in terms of the primitive variables $(\rho, \mathbf{u}, e, \mathbf{B})$ can be written in the following form

$$\frac{\partial \rho}{\partial t} = -\frac{1}{c_s^2} \left[\frac{1}{\rho} \left(\frac{\partial P}{\partial e} \right)_\rho d_2 + \frac{\rho \alpha_+}{2} (d_5 + d_6) + \frac{\rho \alpha_-}{2} (d_7 + d_8) \right] - (\mathbf{u}_H \cdot \nabla_H) \rho - \rho \nabla_H \cdot \mathbf{u}_H \quad (\text{A.1})$$

$$\frac{\partial u_x}{\partial t} = -\frac{s_z}{2} \left[R_y (-d_3 + d_4) + \frac{c_- \alpha_-}{c_s^2} R_x (-d_5 + d_6) + \frac{c_+ \alpha_+}{c_s^2} R_x (d_7 - d_8) \right] - \frac{1}{\rho} \frac{\partial \left(P + \frac{B^2}{2\mu_0} \right)}{\partial x} - (\mathbf{u}_H \cdot \nabla_H) u_x + \frac{1}{\mu_0 \rho} (\mathbf{B}_H \cdot \nabla_H) B_x \quad (\text{A.2})$$

$$\frac{\partial u_y}{\partial t} = -\frac{s_z}{2} \left[R_x (d_3 - d_4) + \frac{c_- \alpha_-}{c_s^2} R_y (-d_5 + d_6) + \frac{c_+ \alpha_+}{c_s^2} R_y (d_7 - d_8) \right] - \frac{1}{\rho} \frac{\partial \left(P + \frac{B^2}{2\mu_0} \right)}{\partial y} - (\mathbf{u}_H \cdot \nabla_H) u_y + \frac{1}{\mu_0 \rho} (\mathbf{B}_H \cdot \nabla_H) B_y \quad (\text{A.3})$$

$$\frac{\partial u_z}{\partial t} = -\frac{1}{2c_s^2} [c_+ \alpha_+ (d_5 - d_6) + c_- \alpha_- (d_7 + d_8)] - (\mathbf{u}_H \cdot \nabla_H) u_z + g + \frac{1}{\mu_0 \rho} (\mathbf{B}_H \cdot \nabla_H) B_z \quad (\text{A.4})$$

$$\frac{\partial e}{\partial t} = -\frac{1}{c_s^2} \left[\frac{1}{\rho} \left(\frac{\partial P}{\partial \rho} \right)_e + \frac{1}{2} (e + P) \alpha_+ (d_5 + \alpha_+ d_6) + \frac{1}{2} (e + P) \alpha_- (d_7 + d_8) \right] - \nabla_H \cdot (e \mathbf{u}_H) - P \nabla_H \cdot \mathbf{u}_H + Q \quad (\text{A.5})$$

$$\frac{\partial B_x}{\partial t} = -\frac{\sqrt{\mu_0 \rho}}{2} \left[R_y (d_3 + d_4) + \frac{\alpha_-}{c_s} R_x (d_5 + d_6) - \frac{\alpha_+}{c_s} R_x (d_7 + d_8) \right] - (\mathbf{u}_H \cdot \nabla_H) B_x + B_x (\nabla_H \cdot \mathbf{u}_H) + (\mathbf{B}_H \cdot \nabla_H) u_x \quad (\text{A.6})$$

$$\frac{\partial B_y}{\partial t} = -\frac{\sqrt{\mu_0 \rho}}{2} \left[-R_x (d_3 + d_4) + \frac{\alpha_-}{c_s} R_y (d_5 + d_6) - \frac{\alpha_+}{c_s} R_y (d_7 + d_8) \right] - (\mathbf{u}_H \cdot \nabla_H) B_y + B_y (\nabla_H \cdot \mathbf{u}_H) + (\mathbf{B}_H \cdot \nabla_H) u_y \quad (\text{A.7})$$

$$\frac{\partial B_z}{\partial t} = -(\mathbf{u}_H \cdot \nabla_H) B_z - B_z (\nabla_H \cdot \mathbf{u}_H) + (\mathbf{B}_H \cdot \nabla_H) u_z \quad (\text{A.8})$$

where we have defined the quantities

$$s_z = \text{sign}(B_z) \\ R_x = \frac{B_x}{B_h} \quad R_y = \frac{B_y}{B_h} \\ \alpha_+^2 = \frac{c_s^2 - c_-^2}{c_+^2 - c_-^2} \quad \alpha_-^2 = \frac{c_+^2 - c_s^2}{c_+^2 - c_-^2}$$

using the velocities

$$c_s^2 = \left(\frac{\partial P}{\partial \rho} \right)_e + \frac{e + P}{\rho} \left(\frac{\partial P}{\partial e} \right)_\rho$$

$$c_a^2 = \frac{B^2}{\mu_0 \rho}$$

$$c_z^2 = \frac{B_z^2}{\mu_0 \rho}$$

$$c_+^2 = \frac{c_a^2 + c_s^2}{2} + \sqrt{\left(\frac{c_a^2 + c_s^2}{2} \right)^2 - c_z^2 c_s^2}$$

$$c_-^2 = \frac{c_a^2 + c_s^2}{2} - \sqrt{\left(\frac{c_a^2 + c_s^2}{2} \right)^2 - c_z^2 c_s^2}.$$

Note the unorthodox ordering of the characteristics: by convention these are usually ordered by the amplitude of the characteristic speeds λ_i , this was not known by us at the time these equations were derived. Therefore, the characteristic z derivatives \mathbf{d} are given by

$$d_1 = u_z \frac{\partial B_z}{\partial z} \quad (\text{A.9})$$

$$d_2 = u_z(e + P) \frac{\partial \rho}{\partial z} - \rho \frac{\partial e}{\partial z} \quad (\text{A.10})$$

$$d_3 = (u_z + c_z) \left(-s_z R_y \frac{\partial u_x}{\partial z} + s_z R_x \frac{\partial u_y}{\partial z} \right. \quad (\text{A.11})$$

$$\left. + \frac{R_y}{\sqrt{\mu_0 \rho}} \frac{\partial B_x}{\partial z} - \frac{R_x}{\sqrt{\mu_0 \rho}} \frac{\partial B_y}{\partial z} \right) \quad (\text{A.12})$$

$$d_4 = (u_z - c_z) \left(s_z R_y \frac{\partial u_x}{\partial z} - s_z R_x \frac{\partial u_y}{\partial z} \right. \quad (\text{A.13})$$

$$\left. + \frac{R_y}{\sqrt{\mu_0 \rho}} \frac{\partial B_x}{\partial z} - \frac{R_x}{\sqrt{\mu_0 \rho}} \frac{\partial B_y}{\partial z} \right) \quad (\text{A.14})$$

$$d_5 = (u_z + c_+) \left(\frac{\alpha_+}{\rho} \frac{\partial P}{\partial z} - s_z R_x c_- \alpha_- \frac{\partial u_x}{\partial z} - s_z R_y c_- \alpha_- \frac{\partial u_y}{\partial z} \right. \quad (\text{A.15})$$

$$\left. + c_+ \alpha_+ \frac{\partial u_z}{\partial z} + \frac{R_x c_s \alpha_-}{\sqrt{\mu_0 \rho}} \frac{\partial B_x}{\partial z} + \frac{R_y c_s \alpha_-}{\sqrt{\mu_0 \rho}} \frac{\partial B_y}{\partial z} \right)$$

$$d_6 = (u_z - c_+) \left(\frac{\alpha_+}{\rho} \frac{\partial P}{\partial z} + s_z R_x c_- \alpha_- \frac{\partial u_x}{\partial z} + s_z R_y c_- \alpha_- \frac{\partial u_y}{\partial z} \right. \quad (\text{A.16})$$

$$\left. - c_+ \alpha_+ \frac{\partial u_z}{\partial z} + \frac{R_x c_s \alpha_-}{\sqrt{\mu_0 \rho}} \frac{\partial B_x}{\partial z} + \frac{R_y c_s \alpha_-}{\sqrt{\mu_0 \rho}} \frac{\partial B_y}{\partial z} \right)$$

$$d_7 = (u_z + c_-) \left(\frac{\alpha_-}{\rho} \frac{\partial P}{\partial z} + s_z R_x c_+ \alpha_+ \frac{\partial u_x}{\partial z} + s_z R_y c_+ \alpha_+ \frac{\partial u_y}{\partial z} \right. \quad (\text{A.17})$$

$$\left. + c_- \alpha_- \frac{\partial u_z}{\partial z} - \frac{R_x c_s \alpha_+}{\sqrt{\mu_0 \rho}} \frac{\partial B_x}{\partial z} - \frac{R_y c_s \alpha_+}{\sqrt{\mu_0 \rho}} \frac{\partial B_y}{\partial z} \right)$$

$$d_8 = (u_z - c_-) \left(\frac{\alpha_-}{\rho} \frac{\partial P}{\partial z} - s_z R_x c_+ \alpha_+ \frac{\partial u_x}{\partial z} - s_z R_y c_+ \alpha_+ \frac{\partial u_y}{\partial z} \right. \quad (\text{A.18})$$

$$\left. - c_- \alpha_- \frac{\partial u_z}{\partial z} - \frac{R_x c_s \alpha_+}{\sqrt{\mu_0 \rho}} \frac{\partial B_x}{\partial z} - \frac{R_y c_s \alpha_+}{\sqrt{\mu_0 \rho}} \frac{\partial B_y}{\partial z} \right).$$

Along inflowing characteristics the characteristic derivatives \mathbf{d} are changed to provide transmitting boundaries by requiring that the incoming characteristics remaining constant. Thus, we require that the boundary conditions for the incoming characteristics must satisfy the static case where $\mathbf{u} = 0$ and $\frac{\partial \mathbf{u}}{\partial t} = 0$, so that

$$d_i = (\mathbf{S}^{-1} \mathbf{C}^{(u=0)})_i. \quad (\text{A.19})$$

The components of $\mathbf{C}^{(u=0)}$ are 0 for the magnetic field equation and the density equation. For the other equations:

$$C_e^{(u=0)} = Q$$

$$C_{u_x}^{(u=0)} = -\frac{1}{\rho} \frac{\partial}{\partial x} \left(P + \frac{B^2}{2\mu_0} \right) + \frac{1}{\mu_0 \rho} (\mathbf{B}_h \cdot \nabla_h) B_x$$

$$C_{u_y}^{(u=0)} = -\frac{1}{\rho} \frac{\partial}{\partial y} \left(P + \frac{B^2}{2\mu_0} \right) + \frac{1}{\mu_0 \rho} (\mathbf{B}_h \cdot \nabla_h) B_y$$

$$C_{u_z}^{(u=0)} = g + \frac{1}{\mu_0 \rho} (\mathbf{B}_h \cdot \nabla_h) B_z.$$

Assuming $Q = 0$, the non-zero incoming characteristic derivative vector \mathbf{d} can be calculated to be:

$$d_3^i = \frac{s_z}{\rho} \left[-(\mathbf{R}_h \times \nabla_h) \left(P + \frac{B_z^2}{2\mu_0} \right) + \frac{B_H}{\mu_0} (\nabla_h \times \mathbf{B}_h) \right] \quad (\text{A.20})$$

$$d_4^i = -d_3^i \quad (\text{A.21})$$

$$d_5^i = c_+ \alpha_+ \left(g + \frac{1}{\mu_0 \rho} (\mathbf{B}_h \cdot \nabla_h) B_z \right) \quad (\text{A.22})$$

$$+ c_- \alpha_- \frac{s_z}{\rho} (\mathbf{R}_h \cdot \nabla_h) \left(P + \frac{B_z^2}{2\mu_0} \right) \quad (\text{A.23})$$

$$d_6^i = -d_5^i \quad (\text{A.24})$$

$$d_7^i = c_- \alpha_- \left(g + \frac{1}{\mu_0 \rho} (\mathbf{B}_h \cdot \nabla_h) B_z \right) \quad (\text{A.25})$$

$$- c_+ \alpha_+ \frac{s_z}{\rho} (\mathbf{R}_h \cdot \nabla_h) \left(P + \frac{B_z^2}{2\mu_0} \right) \quad (\text{A.26})$$

$$d_8^i = -d_7^i. \quad (\text{A.27})$$

Then either d_i or d_i^i is chosen depending on the sign of λ_i at the boundary; i.e. whether or not the characteristic is in- or out-flowing.

Appendix B: Time-dependent H₂ formation

The module that computes non-equilibrium hydrogen ionization has been extended to include a ninth equation for time-dependent H₂ formation. It is given by:

$$F_9 = \frac{n_{\text{H}_2}}{n_{\text{H}_2}^0} - \frac{\Delta t}{n_{\text{H}_2}^0} (C_{3\text{H}} n_1^3 - C_{\text{H}_2\text{H}} n_{\text{H}_2} n_1) - 1 = 0, \quad (\text{B.1})$$

with $n_{\text{H}_2}^0$ the H₂ population of the previous timestep, Δt the timestep and n_1 the population of atomic hydrogen in the ground state. The rate coefficients are given by

$$C_{\text{H}_2\text{H}} = \alpha \left(\frac{T}{300 \text{ K}} \right)^\beta e^{-\gamma/T}, \quad (\text{B.2})$$

$$C_{3\text{H}} = \frac{C_{\text{H}_2\text{H}}}{K(T)}. \quad (\text{B.3})$$

The values for α , β and γ are taken from the UMIST database (Woodall et al. 2007, www.udfa.net); $K(T)$ is the chemical equilibrium constant taken from Tsuji (1973). The derivatives of the functional F_9 with respect to the dependent variables are

$$\frac{\partial F_9}{\partial T} = \frac{\Delta t}{n_{\text{H}_2}^0} \left(\frac{\partial C_{3\text{H}}}{\partial T} n_1^3 - \frac{\partial C_{\text{H}_2\text{H}}}{\partial T} n_{\text{H}_2} n_1 \right), \quad (\text{B.4})$$

$$\frac{\partial F_9}{\partial n_1} = -\frac{\Delta t}{n_{\text{H}_2}^0} (3C_{3\text{H}} n_1^2 - C_{\text{H}_2\text{H}} n_{\text{H}_2}), \quad (\text{B.5})$$

$$\frac{\partial F_9}{\partial n_{\text{H}_2}} = \frac{1}{n_{\text{H}_2}^0} - \frac{\Delta t}{n_{\text{H}_2}^0} C_{\text{H}_2\text{H}} n_1. \quad (\text{B.6})$$

The rate equation for n_1 from Leenaarts et al. (2007) is modified to include source and sink terms due to H_2 :

$$F_3 = \frac{n_1}{n_1^0} - \frac{\Delta t}{n_1^0} \left(\sum_{j=1}^6 n_j P_{j1} + C_{H_2H} n_{H_2} n_1 - C_{3H} n_1^3 \right) - 1 = 0, \quad (\text{B.7})$$

with n_1^0 the ground state hydrogen population from the previous timestep. The derivatives of this equation and equations expressing energy conservation and hydrogen nucleus conservation (see Leenaarts et al. 2007) are modified correspondingly.

References

- Abbett, W. P. 2007, *ApJ*, 665, 1469
- Arber, T. D., Haynes, M., & Leake, J. E. 2007, *ApJ*, 666, 541
- Arnaud, M., & Rothenflug, R. 1985, *A&AS*, 60, 425
- Carlsson, M., & Stein, R. F. 1995, *ApJ*, 440, L29
- Carlsson, M., & Stein, R. F. 2002, *ApJ*, 572, 626
- Carlsson, M., Stein, R. F., Nordlund, Å., & Scharmer, G. B. 2004, *ApJ*, 610, L137
- Carlsson, M., Hansteen, V. H., & Gudiksen, B. V. 2010, *Mem. Soc. Astron. It.*, 81, 582
- Cheung, M. C. M., Schüssler, M., & Moreno-Insertis, F. 2007, *A&A*, 467, 703
- De Pontieu, B., Hansteen, V. H., Rouppe van der Voort, L., van Noort, M., & Carlsson, M. 2007, *ApJ*, 655, 624
- DeVore, C. R. 1991, *J. Comp. Phys.*, 92, 142
- Dorfi, E. A., & Drury, L. O. 1987, *J. Comp. Phys.*, 69, 175
- Galsgaard, K., & Nordlund, Å. 1996, *J. Geophys. Res.*, 101, 13445
- Gudiksen, B. V., & Nordlund, Å. 2005, *ApJ*, 618, 1020
- Gustafsson, B., Bell, R. A., Eriksson, K., & Nordlund, Å. 1975, *A&A*, 42, 407
- Hansteen, V. 1993, *ApJ*, 402, 741
- Hansteen, V. H. 2004, in *Multi-Wavelength Investigations of Solar Activity*, ed. A. V. Stepanov, E. E. Benevolenskaya, & A. G. Kosovichev (Cambridge: Cambridge University Press), IAU Symp., 223, 385
- Hansteen, V. H., De Pontieu, B., Rouppe van der Voort, L., van Noort, M., & Carlsson, M. 2006, *ApJ*, 647, L73
- Hansteen, V. H., Carlsson, M., & Gudiksen, B. 2007, in *The Physics of Chromospheric Plasmas*, ed. P. Heinzel, I. Dorotovič, & R. J. Rutten, ASP Conf. Ser., 368, 107
- Hansteen, V. H., Hara, H., De Pontieu, B., & Carlsson, M. 2010, *ApJ*, 718, 1070
- Hayek, W., Asplund, M., Carlsson, M., et al. 2010, *A&A*, 517, A49
- Hegglund, L., De Pontieu, B., & Hansteen, V. H. 2007, *ApJ*, 666, 1277
- Heinemann, T., Nordlund, Å., Scharmer, G. B., & Spruit, H. C. 2007, *ApJ*, 669, 1390
- Hyman, J. M. 1979, in *Proc. third IMACS international symposium on computer methods for partial differential equations*, ed. R. Vichnevetsky, & R. Stepleman, International Association for Mathematics and Computers in Simulation, Dept. of Computer Science, Rutgers University, New Brunswick, N.J. 08903 USA: IMACS, 3, 313
- Isobe, H., Proctor, M. R. E., & Weiss, N. O. 2008, *ApJ*, 679, L57
- Judge, P. G., & Meisner, R. W. 1994, in *Solar Dynamic Phenomena and Solar Wind Consequences, the Third SOHO Workshop*, ed. J. J. Hunt, ESA SP, 373, 67
- Keller, C. U., Schüssler, M., Vögler, A., & Zakharov, V. 2004, *ApJ*, 607, L59
- Leenaarts, J., & Carlsson, M. 2009, in *ASP Conf. Ser.* 415, ed. B. Lites, M. Cheung, T. Magara, J. Mariska, & K. Reeves, 87
- Leenaarts, J., Carlsson, M., Hansteen, V., & Rutten, R. J. 2007, *A&A*, 473, 625
- Leenaarts, J., Carlsson, M., Hansteen, V., & Gudiksen, B. 2011, *A&A*, 530, A124
- Martínez-Sykora, J., Hansteen, V., & Carlsson, M. 2008, *ApJ*, 679, 871
- Martínez-Sykora, J., Hansteen, V., & Carlsson, M. 2009a, *ApJ*, 702, 129
- Martínez-Sykora, J., Hansteen, V., DePontieu, B., & Carlsson, M. 2009b, *ApJ*, 701, 1569
- Martínez-Sykora, J., Hansteen, V., & Moreno-Insertis, F. 2011, *ApJ*, accepted [arXiv:1011.4703]
- Miyama, S. M., Tomisaka, K., & Hanawa, T. 1999, *Astrophys. Space Sci. Lib.*, 240, Numerical Astrophysics (Dordrecht: Springer)
- Nordlund, A. 1982, *A&A*, 107, 1
- Nordlund, Å., & Galsgaard, K. 1995, *A 3D MHD Code for Parallel Computers*, Tech. Rep., Astronomical Observatory, Copenhagen University
- Orszag, S. A., & Tang, C.-M. 1979, *J. Fluid Mech. Dig. Arch.*, 90, 129
- Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. 1992, *Numerical recipes in FORTRAN, The art of scientific computing* (Cambridge: Cambridge University Press)
- Rempel, M., Schüssler, M., Cameron, R. H., & Knölker, M. 2009, *Science*, 325, 171
- Rybicki, G. B., & Lightman, A. P. 1979, *Radiative processes in astrophysics* (New York: Wiley-Interscience)
- Ryu, D., Miniati, F., Jones, T. W., & Frank, A. 1998, *ApJ*, 509, 244
- Schaffnerberger, W., Wedemeyer-Böhm, S., Steiner, O., & Freytag, B. 2005, in *Chromospheric and Coronal Magnetic Fields*, ed. D. E. Innes, A. Lagg, & S. A. Solanki, ESA Spec. Pub., 596,
- Shull, J. M., & van Steenberg, M. 1982, *ApJS*, 48, 95
- Skartlien, R. 2000, *ApJ*, 536, 465
- Sod, G. A. 1978, *J. Comp. Phys.*, 27, 1
- Sollum, E. 1999, Master's thesis, University of Oslo
- Spitzer, L. 1956, *Physics of Fully Ionized Gases* (New York: Interscience Publishers)
- Stein, R. F., & Nordlund, Å. 2006, *ApJ*, 642, 1246
- Stein, R. F., Nordlund, Å., Georgoviani, D., Benson, D., & Schaffnerberger, W. 2009, in *ASP Conf. Ser.* 416, ed. M. Dikpati, T. Arentoft, I. González Hernández, C. Lindsey, & F. Hill, 421
- Stein, R. F., Lagerfjord, A., Nordlund, A., & Georgobiani, D. 2010, in *Am. Astron. Soc. Meeting Abstracts*, 216, 211.03
- Thompson, K. W. 1987, *J. Comp. Phys.*, 68, 1
- Tortosa-Andreu, A., & Moreno-Insertis, F. 2009, *A&A*, 507, 949
- Tóth, T., & Odstrčil, D. 1996, *J. Comp. Phys.*, 128, 82
- Trujillo Bueno, J., & Fabiani Bendicho, P. 1995, *ApJ*, 455, 646
- Tsuji, T. 1973, *A&A*, 23, 411
- Vernazza, J. E., Avrett, E. H., & Loeser, R. 1981, *ApJS*, 45, 635
- Vögler, A., Shelyag, S., Schüssler, M., et al. 2005, *A&A*, 429, 335
- Wedemeyer-Böhm, S., & Steffen, M. 2007, *A&A*, 462, L31
- Woodall, J., Agúndez, M., Markwick-Kemper, A. J., & Millar, T. J. 2007, *A&A*, 466, 1197