

The Hydrodynamics and Radiation Code

HydRad

User Guide

CONTENTS

1. A Brief Overview of HYDRAD and its Capabilities.....	3
2. Preliminaries.....	5
2.1 System Requirements.....	5
2.2 Downloading HYDRAD.....	6
2.3 Installing HYDRAD.....	7
2.4 The HYDRAD Graphical User Interface (GUI).....	7
2.5 Testing the Installation.....	9
3. Creating a Set of Initial Conditions.....	15
3.1 Geometrical Properties.....	16
3.2 Boundary Conditions.....	17
3.3 Parameter Space for Solutions.....	19
4. Output.....	21
4.1 Runtime Information.....	21
4.1.1 Initial Conditions.....	21
4.1.2 HYDRAD.....	22
4.2 Data Files and Formats.....	24
4.2.1 Common Files.....	25
4.2.2 Initial Conditions.....	29
4.2.3 HYDRAD.....	29
5. Configuring HYDRAD's Physics Model.....	33
5.1 Heating.....	33
5.1.1 Background Heating.....	34
5.1.2 Phenomenological Heating.....	34
5.1.3 Electron Beam Heating.....	36
5.1.4 Alfven Wave Heating.....	37
5.2 Radiation.....	38
5.2.1 Power-law Radiation.....	38
5.2.2 Lookup Tables for Radiation.....	39
5.2.3 User-defined Radiation.....	39
5.2.4 Decoupling Non-equilibrium Ionization from the Radiation Calculation.....	41
5.2.5 Optically-thick Photosphere and Chromosphere.....	41
5.3 Thermal Conduction.....	43

5.3.1 Classical (Spitzer Harm) Thermal Conduction.....	43
5.3.2 Suppression of Thermal Conduction by Magnetic Turbulence.....	44
5.3.3 The Johnson & Bradshaw Modification to Thermal Conduction.....	44
5.3.4 Non-classical Thermal Conduction.....	45
5.4 Inter-species Collisions.....	46
5.5 Flux Tube Properties.....	47
 6. Configuring and Optimizing the Numerical Solver.....	48
6.1 Radiation.....	48
6.2 Atomic Processes.....	49
6.3 Thermal Conduction and Bulk Transport.....	49
6.4 Kinematic and Artificial Viscosity.....	49
6.5 Other Physics and Stability-related Parameters.....	50
 7. Configuring and Managing the Adaptive Grid.....	51
7.1 Basic Grid Properties.....	51
7.2 Adaptive Refinement Parameters.....	52
7.3 Additional Restriction Parameters.....	53

1. A Brief Overview of HYDRAD and its Capabilities

The HYDroynamics and RADiation (HYDRAD) code was originally¹ developed to model the field-aligned plasma physics of solar coronal loops subject to impulsive thermal heating, with particularly careful attention paid to the time-dependent evolution of the emitting ion species and their radiative coupling to the plasma, and dynamically capturing the small spatial scales that arise in the solar transition region. It solves the conservative form (mass, momentum, and energy density) of the hydrodynamic equations. Over the years it has evolved into a flexible and powerful code capable of modeling a broad variety of phenomena including: multi-species plasma confined to full-length, magnetic flux tubes of arbitrary geometrical and cross-section variation in the field-aligned direction²; solar flares driven by non-thermal electrons³ and Alfvén waves⁴, and the non-thermal equilibrium response of the chromosphere⁵; coronal rain formed by condensations in thermal non-equilibrium where the adaptive grid is required to fully resolve and track multiple steep transition regions⁶; and ultracold, strongly coupled laboratory plasmas^{7,8} composed of weakly-ionized strontium.

The code is written in C++ and designed to be modular in its structure, such that new capabilities (e.g. physics processes) can be added in a relatively straightforward way and handled robustly by the numerical scheme. HYDRAD is also intended to be fairly undemanding of computational resources, though its needs do depend strongly on the particular nature of each model run (e.g. physics requirements, spatial resolution). The recently

¹ Bradshaw, S. J., & Mason, H. E. "A Self-Consistent Treatment of Radiation in Coronal Loop Modeling", 2003, *Astronomy & Astrophysics*, 401, 699

² Bradshaw, S. J., & Viall, N. M. "Patterns of Activity in a Global Model of a Solar Active Region", 2016, *Astrophysical Journal*, 821, 63

³ Reep, J. W., Bradshaw, S. J., & McAteer, R. T. J. "On the Sensitivity of the GOES Flare Classification to Properties of the Electron Beam in the Thick Target Model", 2013, *Astrophysical Journal*, 778, 76

⁴ Reep, J. W., Russell, A. J. B., Tarr, L. A., & Leake, J. E. "A Hydrodynamic Model of Alfvénic Wave Heating in a Coronal Loop and Its Chromospheric Footpoints", 2018, *Astrophysical Journal*, 853, 101

⁵ Reep, J. W., Bradshaw, S. J., Crump, N. A., & Warren, H. P. "Efficient Calculation of Non-Local Thermodynamic Equilibrium Effects in Multithreaded Hydrodynamic Simulations of Solar Flares", 2019, *Astrophysical Journal*, 871, 18

⁶ Johnston, C. D., Cargill, P. J., Antolin, P., Hood, A. W., De Moortel, I., & Bradshaw, S. J. "The Effects of Numerical Resolution, Heating Timescales and Background Heating on Thermal Non-Equilibrium in Coronal Loops", 2019, *Astronomy & Astrophysics*, in press

⁷ McQuillen, P., Castro, J., Strickler, T., Bradshaw, S. J., & Killian, T. C. "Ion Holes in Ultracold Neutral Plasmas", 2013, *Physics of Plasmas*, 20, 043516

⁸ McQuillen, P., Castro, J., Bradshaw, S. J., & Killian, T. C. "Emergence of Kinetic Behavior in Streaming Ultracold Neutral Plasmas", 2015, *Physics of Plasmas*, 22, 043514

implemented non-LTE solver for a 6-level hydrogen atom in the optically-thick chromosphere necessitated parallelization of part of the code (the OpenMP standard is employed) to recover acceptable runtimes. A significant performance gain may also be obtained when solving for the time-dependent ionization state of a large number of elements coupled to the electron energy equation via the radiative loss term. Otherwise, if this functionality is not required, then HYDRAD is generally most efficiently executed in single-processor mode with multiple instances running in an “embarrassingly-parallel” exploration of a parameter space, for example. Some experimentation with the OpenMP option (under the “Files” tab in the HYDRAD GUI, Section 2.4) may be needed to determine the optimum performance settings for your hardware.

The code has been extensively deployed, tested, and used for a large number of scientific investigations on Windows PC, Mac, and Linux platforms, and found to be stable and robust. However, as with all complex software tools, there are likely to be as yet undetected and unaccounted for behavioral peculiarities on various hardware/operating system/compiler combinations. It is strongly advisable to follow the recommendations laid out in Section 2.1. Unresolvable issues can be submitted to the HYDRAD GitHub repository⁹. Submitted issues must include a clear description of the problem and other pertinent information (e.g. hardware/OS/compiler details).

⁹ <https://github.com/rice-solar-physics/hydrad>

2. Preliminaries

2.1 System Requirements

HYDRAD has been designed to run on a broad range of hardware, from basic laptops to high-end supercomputing facilities. The primary determining factor in the code's performance is processor speed, though note that this does not necessarily correlate with clock speed. Modern, efficient processors can achieve higher throughput and floating-point operations per second (FLOPS) at lower clock speeds than earlier generations. In its most basic configuration the memory foot-print of HYDRAD is almost negligible (a couple of MB) and it is therefore small enough to fit entirely into the L2 cache of most processors, which avoids additional memory paging operations and significantly reduces the execution time. However, in more sophisticated configurations (e.g. non-equilibrium ion populations radiatively coupled to the plasma) memory requirements can grow to ~ 1 GB due to the atomic data HYDRAD must store and access to make the ionization and radiation calculations.

The amount of data produced by a single run is also strongly dependent on what the code is configured to do. A basic model may output no more than 1 GB of data, but when ionization state and atomic level populations are required, in addition to all of the terms of the evolution equations, particularly at high temporal cadence, then the amount of data produced per run can grow to several hundred GB.

HYDRAD assumes the GNU g++ compiler is installed by default and the most recent version is always preferred, with access to the latest flags and switches, to build the fastest and most efficient code. Sometimes the Intel C++ compiler will build faster code on Intel hardware and HYDRAD can be configured to use alternative compilers.

The g++ compiler is available as standard on Mac and Linux platforms. It is strongly recommended that the latest version is installed for building HYDRAD. The Cygwin¹⁰ environment is recommended for providing functionality similar to a Linux distribution in Windows. Make sure that the most recent version of g++ is selected in the Cygwin download manager,

¹⁰ <https://www.cygwin.com>

which makes sure that all relevant packages and their dependencies are obtained.

HYDRAD also includes a graphical user interface (GUI) to simplify the process of configuring and executing the code (Section 2.4). The GUI is written in Java (a single .jar file) and requires the Java runtime environment¹¹ to be installed on your system. The most recent version is always recommended.

2.2 Downloading HYDRAD

HYDRAD is available from a single GitHub repository¹², managed by the Rice University Solar Physics Group, to make it as straightforward as possible to maintain a current, stable, verified, and validated version of the code, with rollback to previous versions if needed, and to respond to requests for additional features/functionality and bug fixes. HYDRAD should not be obtained from anywhere else and doing so is practically inviting trouble. If your version of the code becomes somehow corrupted then simply delete it and download a clean copy from the official repository.

The recommended way to obtain HYDRAD from GitHub is to clone the repository on your local machine. That way, you can “pull” updates down from the repository as and when they become available. You will need to install the GitHub command line tools on your machine to do this. GitHub also provides an option to download the repository as a .zip file.

To clone the HYDRAD repository, first log into the website and select the green “Clone or download” button, and then copy the URL in the text box (or choose to download the .zip file if you prefer). Next, open a terminal window and change to the directory in which you want to place the cloned HYDRAD directory, and then type “git clone <URL>” where the URL that you copied from the text box is pasted in place of <URL>.

The cloned HYDRAD directory can be updated at any time simply by entering the directory and typing “git pull”.

¹¹ <https://www.java.com>

¹² <https://github.com/rice-solar-physics/hydrad>

When you have successfully obtained HYDRAD (either by cloning or by downloading the .zip file) you will see the following top-level directory structure:

```
HYDRAD/  
  .git/  
  Forward_Model/  
  Heating_Model/  
  HYDRAD/  
  HYDRAD_GUI/  
  Initial_Conditions/  
  Kinetic_Model/  
  Radiation_Model/  
  Resources/  
  Results/  
  Visualisation/
```

There will also be some files:

```
.gitignore  
HYDRAD_GUI.jar  
LICENSE  
README.md
```

2.3 Installing HYDRAD

When you have followed the steps described in Sections 2.1 and 2.2 to place a copy of HYDRAD on your system then it is recommended that this copy is kept pristine (e.g. as a local master copy). All instances of HYDRAD (e.g. with different sets of input parameter values) should then be configured and run using a new copy of this master copy (which is always kept up-to-date). The on-disk foot-print of the HYDRAD installation (~14 MB) is negligible compared to its volume of data output and the capacity of modern storage devices and so, despite some redundancy, maintaining multiple instances/copies is not a significant use of resources.

2.4 The HYDRAD Graphical User Interface (GUI)

The functionality of HYDRAD is set-up for each run via a large number of C++ header (.h) files and configuration files, which are used to build the code

at compile time and read by the code at runtime, and they determine all aspects of what physics is included, and how the code behaves and performs. Writing these files oneself for each run, even based on a set of default templates, would be laborious and tedious, and hence a GUI has been developed which allows the user to select the required functionality in a far more simple and intuitive manner. This makes setting up and running HYDRAD quick and easy. The GUI writes all of the necessary header and configuration files, based on the user's choices, and then compiles and executes the code.

The GUI is written in Java and is contained in a single .jar file. The Java runtime environment is required to use it. Change to the HYDRAD directory and type “java -jar HYDRAD_GUI.jar” to run the GUI or double-click on the corresponding application icon. The GUI opens and reads the configuration file “HYDRAD_GUI/config/default.cfg” to populate the options in the GUI with

their default settings.

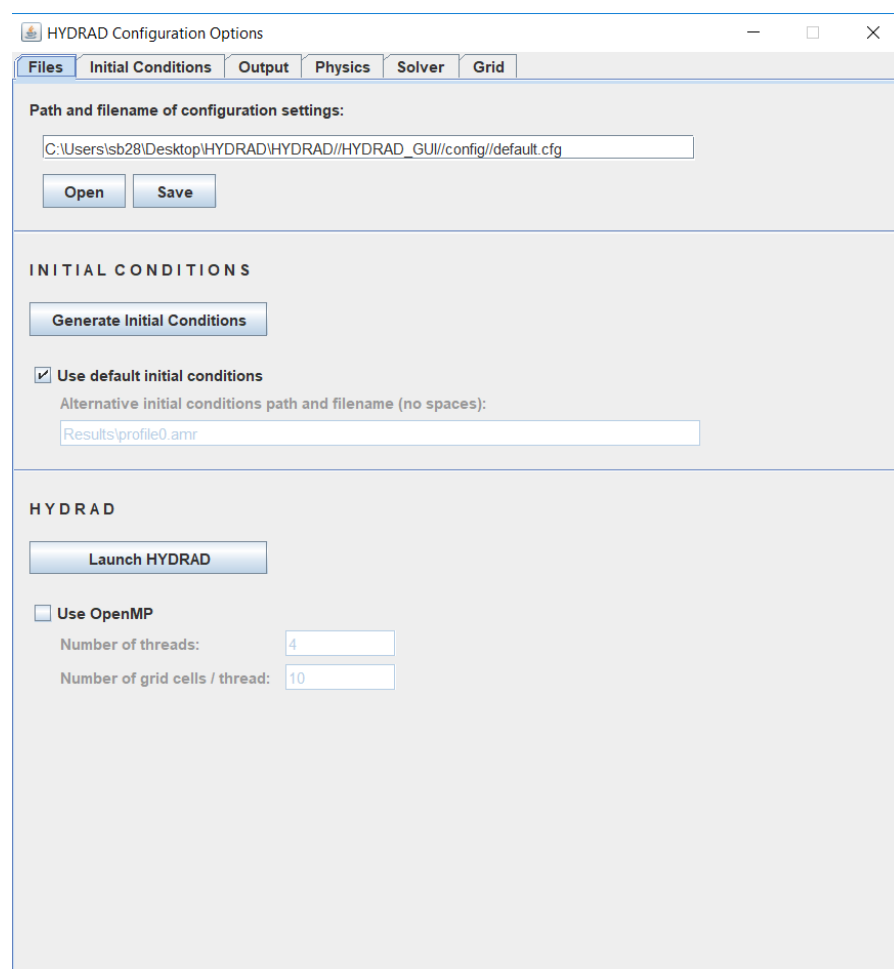


Figure 1: The HYDRAD GUI opens to the “Files” tab and presents options related to saving/restoring configuration settings, creating initial conditions, and running HYDRAD.

Figure 1 shows the appearance of the HYDRAD GUI when it is opened. The first text box shows the full path and filename of the configuration file in which the selections for each of the user-configurable options are stored. The user can save and reload multiple configurations, under different filenames, for model runs with different sets of parameter values. The GUI configuration file(s) can also be

provided as part of a help request/bug report via GitHub so that identical configurations can be set-up and investigated, and the code behaviour reproduced.

Making changes to any of the GUI options, copying the default path and directory in the text box, choosing “Save” to bring up a dialog box, and pasting the path and directory into the “File Name” box before saving the changes will overwrite “default.cfg” in the “HYDRAD_GUI/config/” directory. This means that if the GUI is then closed and reopened, the changes made by the user will have been preserved.

Choosing “Open” brings up a dialog box which allows the user to select a previously saved configuration file for the GUI.

The remaining options under the “Files” tab are discussed in Section 2.5.

2.5 Testing the Installation

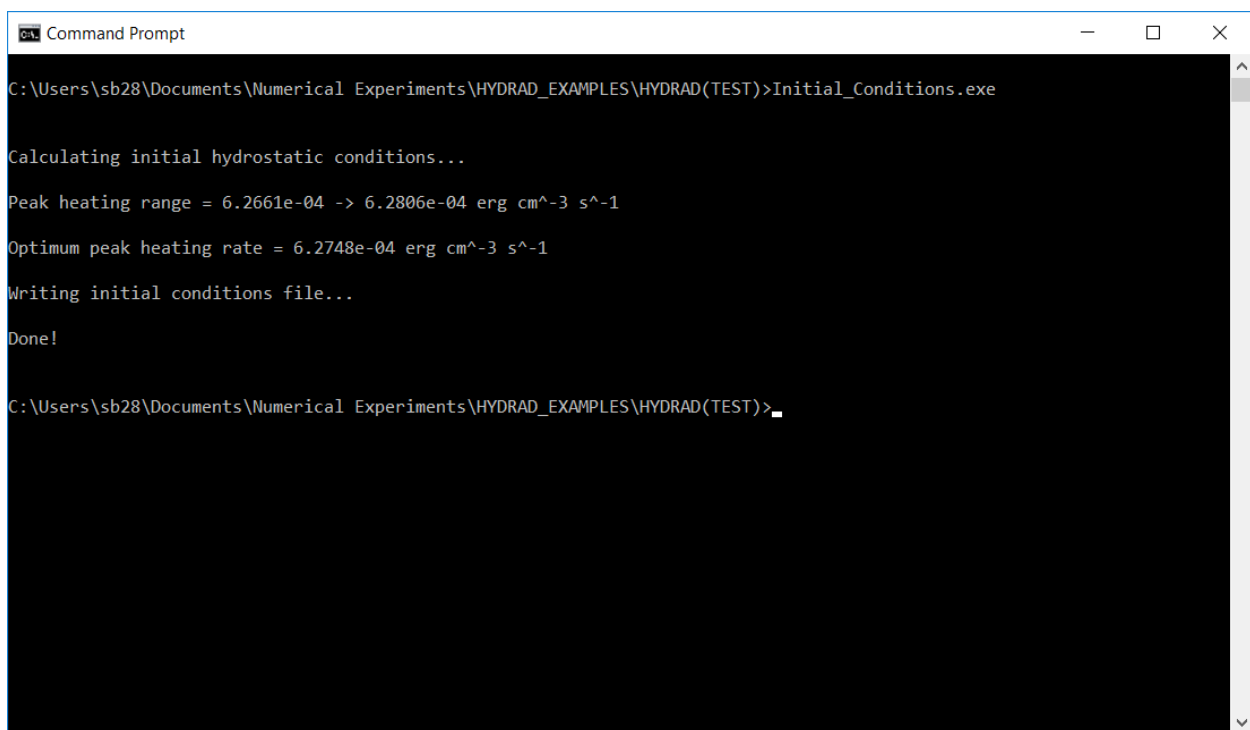
Once a pristine (master) copy of HYDRAD exists on your system make a copy of it and place it somewhere else. You can also rename it for convenience (it helps if the name is descriptive), e.g.

```
HYDRAD(TEST_RUN)/  
  .git/  
  Forward_Model/  
  Heating_Model/  
  HYDRAD/  
  HYDRAD_GUI/  
  Initial_Conditions/  
  Kinetic_Model/  
  Radiation_Model/  
  Resources/  
  Results/  
  Visualisation/  
  
  .gitignore  
  HYDRAD_GUI.jar  
  LICENSE  
  README.md
```

Run the GUI “java -jar HYDRAD_GUI.jar” or double-click on the corresponding application icon.

The default GUI settings have been chosen to create a semi-circular coronal loop of total length 60 Mm and peak temperature 1.6 MK at the apex, with two isothermal chromospheres (0.02 MK) of length 5 Mm at each end, and for HYDRAD to maintain this coronal loop in a stable hydrostatic equilibrium by applying only the “background” heating needed to create it.

First, click on “Generate Initial Conditions” under the “Files” tab. All being well, the GUI will write the necessary header and configuration files, compile, and then execute the code which calculates the initial plasma conditions in the coronal loop, subject to the parameter values chosen under the “Initial Conditions” tab, for HYDRAD to evolve.



```
Command Prompt
C:\Users\sb28\Documents\Numerical Experiments\HYDRAD_EXAMPLES\HYDRAD(TEST)>Initial_Conditions.exe

Calculating initial hydrostatic conditions...
Peak heating range = 6.2661e-04 -> 6.2806e-04 erg cm^-3 s^-1
Optimum peak heating rate = 6.2748e-04 erg cm^-3 s^-1
Writing initial conditions file...
Done!

C:\Users\sb28\Documents\Numerical Experiments\HYDRAD_EXAMPLES\HYDRAD(TEST)>
```

Figure 2: Executing *Initial_Conditions.exe* should produce the output shown here for the default GUI configuration settings used to test the HYDRAD installation.

On some systems (e.g. Windows PC) a terminal window pops-up in which this code compiles and runs. On other systems (e.g. Linux-based) the code runs in the background. The code can be forced to run in its own terminal window by first clicking on “Generate Initial Conditions” to write the

necessary header and configuration files, and then halting the background process “Initial_Conditions.exe” in a new terminal window. Change to the top-level “HYDRAD/” directory of the copy you are running and execute the script “Initial_Conditions.bat” that is also written by the GUI, which compiles and runs the code.

Figure 2 shows the expected output to the terminal window when the code which calculates the initial conditions is compiled and executed with the default GUI configuration settings.

The data files describing the initial conditions, which HYDRAD takes as input, are written into the “Initial_Conditions/profiles/” directory, e.g.

```
HYDRAD(TEST)/  
  Initial_Conditions/  
    profiles/
```

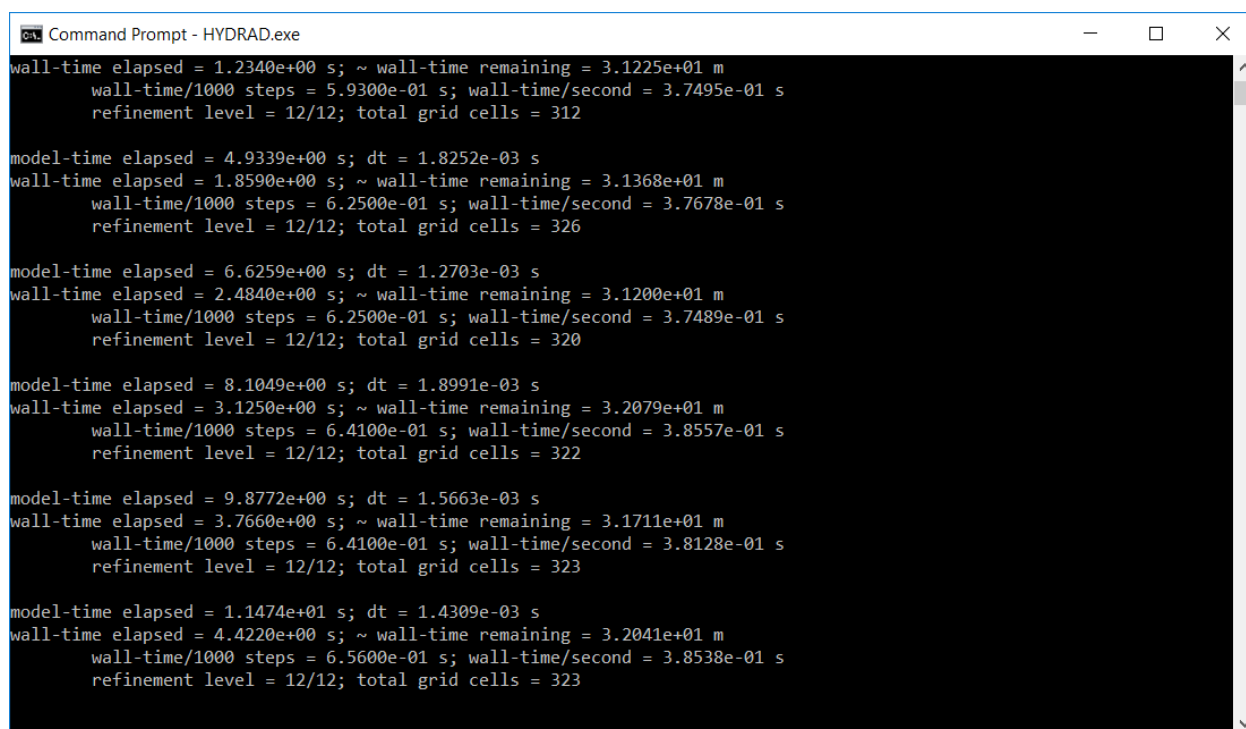
There should be several data files.

```
initial.amr  
initial.amr.gravity  
initial.amr.phy  
initial.amr.sol  
initial.amr.table  
profiles.txt
```

The format and contents of the first five files are described in Sections 4.2.1 – 4.2.2. The last file is a plain text document describing the contents of the directory.

When these files have been successfully created then click on “Launch HYDRAD” under the “Files” tab. The GUI will write the necessary header and

configuration files, compile, and then execute HYDRAD, subject to the default set of parameter values.



```

Command Prompt - HYDRAD.exe

wall-time elapsed = 1.2340e+00 s; ~ wall-time remaining = 3.1225e+01 m
wall-time/1000 steps = 5.9300e-01 s; wall-time/second = 3.7495e-01 s
refinement level = 12/12; total grid cells = 312

model-time elapsed = 4.9339e+00 s; dt = 1.8252e-03 s
wall-time elapsed = 1.8590e+00 s; ~ wall-time remaining = 3.1368e+01 m
wall-time/1000 steps = 6.2500e-01 s; wall-time/second = 3.7678e-01 s
refinement level = 12/12; total grid cells = 326

model-time elapsed = 6.6259e+00 s; dt = 1.2703e-03 s
wall-time elapsed = 2.4840e+00 s; ~ wall-time remaining = 3.1200e+01 m
wall-time/1000 steps = 6.2500e-01 s; wall-time/second = 3.7489e-01 s
refinement level = 12/12; total grid cells = 320

model-time elapsed = 8.1049e+00 s; dt = 1.8991e-03 s
wall-time elapsed = 3.1250e+00 s; ~ wall-time remaining = 3.2079e+01 m
wall-time/1000 steps = 6.4100e-01 s; wall-time/second = 3.8557e-01 s
refinement level = 12/12; total grid cells = 322

model-time elapsed = 9.8772e+00 s; dt = 1.5663e-03 s
wall-time elapsed = 3.7660e+00 s; ~ wall-time remaining = 3.1711e+01 m
wall-time/1000 steps = 6.4100e-01 s; wall-time/second = 3.8128e-01 s
refinement level = 12/12; total grid cells = 323

model-time elapsed = 1.1474e+01 s; dt = 1.4309e-03 s
wall-time elapsed = 4.4220e+00 s; ~ wall-time remaining = 3.2041e+01 m
wall-time/1000 steps = 6.5600e-01 s; wall-time/second = 3.8538e-01 s
refinement level = 12/12; total grid cells = 323

```

Figure 3: Executing HYDRAD.exe should produce the output shown here for the default GUI configuration settings used to test the HYDRAD installation.

As in the case with the code which calculates the initial conditions, on some systems a terminal window pops-up in which HYDRAD compiles and runs. On other systems HYDRAD runs in the background. HYDRAD can be forced to run in its own terminal window by first clicking on “Launch HYDRAD” to write the necessary header and configuration files, and then halting the background process “HYDRAD.exe” in a new terminal window. Change to the top-level “HYDRAD/” directory of the copy you are running and execute the script “HYDRAD.bat” that is also written by the GUI, which compiles and runs HYDRAD.

Figure 3 shows the expected output to the terminal window when HYDRAD is compiled and executed with the default GUI configuration settings. The

status information displayed in the window is explained in detail in Section 4.1.2.

The data files containing the results of the calculations are written into the “Results/” directory, e.g.

HYDRAD(TEST)/
Results/

There should be data files of the form:

profile0.amr
profile0.phy
profile1.amr
profile1.phy
.
.
profileN.amr
profileN.phy

The format and contents of these files are described in Section 4.2.1.

Provided these files describe a coronal loop of total length 60 Mm, with peak temperature at the apex of 1.6 MK and two isothermal (0.02 MK) chromospheres at each end, maintained in hydrostatic equilibrium, then HYDRAD is working correctly and is ready to be applied to specific problems.

If HYDRAD is unexpectedly interrupted during a calculation (e.g. a power-cut or mandatory installation of OS updates followed by a system reboot) then it can continue from its last set of output files to minimize wasted time. Run the GUI and uncheck the box next to “Use default initial conditions” under the “Files” tab. Enter “Results/profileN.amr”, where ‘N’ corresponds to the numbers of the last set of output files, in the text box. Click on “Launch HYDRAD”.

The “Use OpenMP” option under the “Files” tab in the GUI is most effective when solving for the time-dependent ionization state of a large number of elements coupled to the electron energy equation via the radiative loss term (Sections 5.2.3 – 5.2.4 and / or when a full, non-LTE radiative transfer calculation is performed to solve for the hydrogen level population in the

chromosphere (Section 5.2.5). The “Number of threads” text box allows the user to specify the number of hardware threads that HYDRAD divides the calculation among, and the “Number of grid cells / thread” specifies the number of cells in the grid allocated to each thread simultaneously. It will be necessary to experiment with different values for these parameters in order to find the optimum combination for a particular model configuration and hardware platform. Generally, HYDRAD’s performance is more sensitive to the number of threads chosen, up to a limit beyond which it will not scale further and performance may even decrease. Choosing the “Use OpenMP” option for less sophisticated calculations may hinder performance and HYDRAD is then best used in single-processor mode by keeping this option unchecked.

3. Creating a Set of Initial Conditions

The initial conditions are the set of mass, momentum, and energy (electron and ion) density profiles in the field-aligned direction, defined at $t = 0$ seconds, that HYDRAD evolves subject to some external driver (e.g. thermal and/or non-thermal energy deposition, an incident wave or impulse).

There are several ideal conditions that the initial conditions ought to satisfy. First of all, they should be as close to hydrostatic equilibrium ($v = 0$ cm/s and $d/dt = 0$ s⁻¹) at possible; this ensures that evolution for $t > 0$ s is due only to the external driver and dynamical behavior arising from a lack of equilibrium will not be erroneously attributed to it. Background noise, most evident in the spectrum of velocity perturbations in the absence of an external driver, is unavoidable in discretization schemes, which essentially truncate high-frequency parts of the spectrum and allow unresolved modes to grow. Nonetheless, they can be controlled through a careful choice and implementation of numerical methods, and by ensuring that all stability criteria are satisfied (Section 6). Generally, one need not be concerned with background velocity fluctuations that are much less (e.g. of order 1%) than the local speed of the relevant wave (in this case the acoustic mode). Next, the initial conditions should be energetically negligible compared with the time-integrated energy of the external driver; this ensures that the evolution for $t > 0$ is effectively independent of the initial conditions.

On occasion, it can be important to carefully tailor the initial conditions in order to make the problem tractable. For example, consider a loop of total length 200 Mm that is initially isothermal at 0.02 MK everywhere along its length: the density scale height of such a low-temperature loop is significantly shorter than the height of its apex above the solar surface; consequently, for reasonable densities at the base of the transition region, the coronal density in hydrostatic equilibrium falls to just a few particles per cubic centimeter. This has two implications: (1) the coronal plasma becomes insufficiently collisional for the fluid treatment underlying the hydrodynamic equations to be valid; and (2) the time-step associated with thermal conduction, particularly if the plasma temperature increases due to heating, becomes impractically small. Avoiding this situation means that it is often preferable to commence model runs with hydrostatic loops that are at least close to coronal conditions, particularly as they increase in length. The paradox here is that one begins with a coronal loop in order to investigate how coronal loops are created, but it is ameliorated by ensuring that the initial conditions

are energetically negligible compared with the external driver (as discussed above).

We now discuss the options provided by the GUI which allow the user to select and configure a broad range of initial conditions for HYDRAD to evolve. When configuration is complete, including the “Physics”, “Solver”, and “Grid” options (Sections 5 – 7), the initial conditions can be calculated by clicking on “Generate Initial Conditions” under the “Files” tab (Section 2.5).

3.1 Geometrical Properties

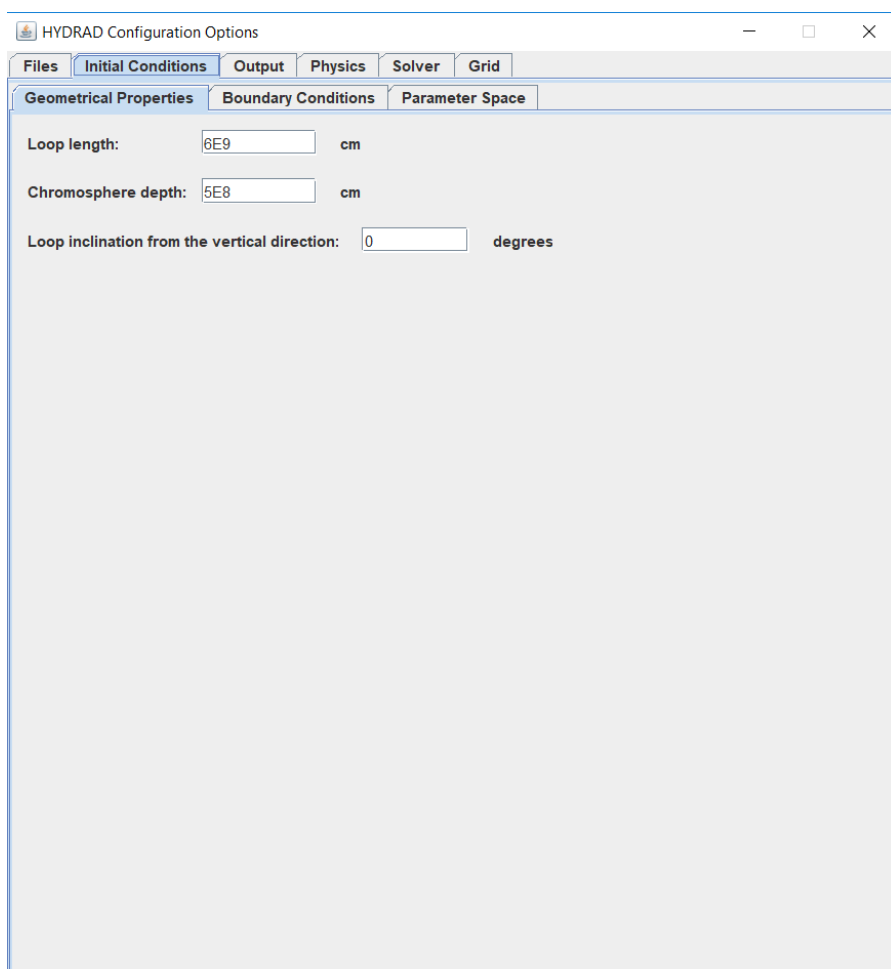


Figure 4 shows the options presented to the user for configuring the geometrical properties of the initial conditions that HYDRAD will evolve. In the example shown, the total length of the loop is 6×10^9 cm (60 Mm), which includes two 5×10^8 cm (5 Mm) chromospheres at each foot-point. The inclination of the loop from the vertical at 0° (perpendicular to the solar surface)

Figure 4: The options for configuring the geometrical properties of the initial conditions presented under the Initial Conditions tab of the HYDRAD GUI.

through 90° (parallel to the solar surface). The default geometrical shape of the loop is semi-circular, but this can be changed by instead specifying a polynomial fit to the gravitational acceleration as a function of field-aligned position (Section 5.5). The cross-section of the loop is uniform by default but,

again, this can be changed by specifying a polynomial fit to the magnetic field strength (Section 5.5).

3.2 Boundary Conditions

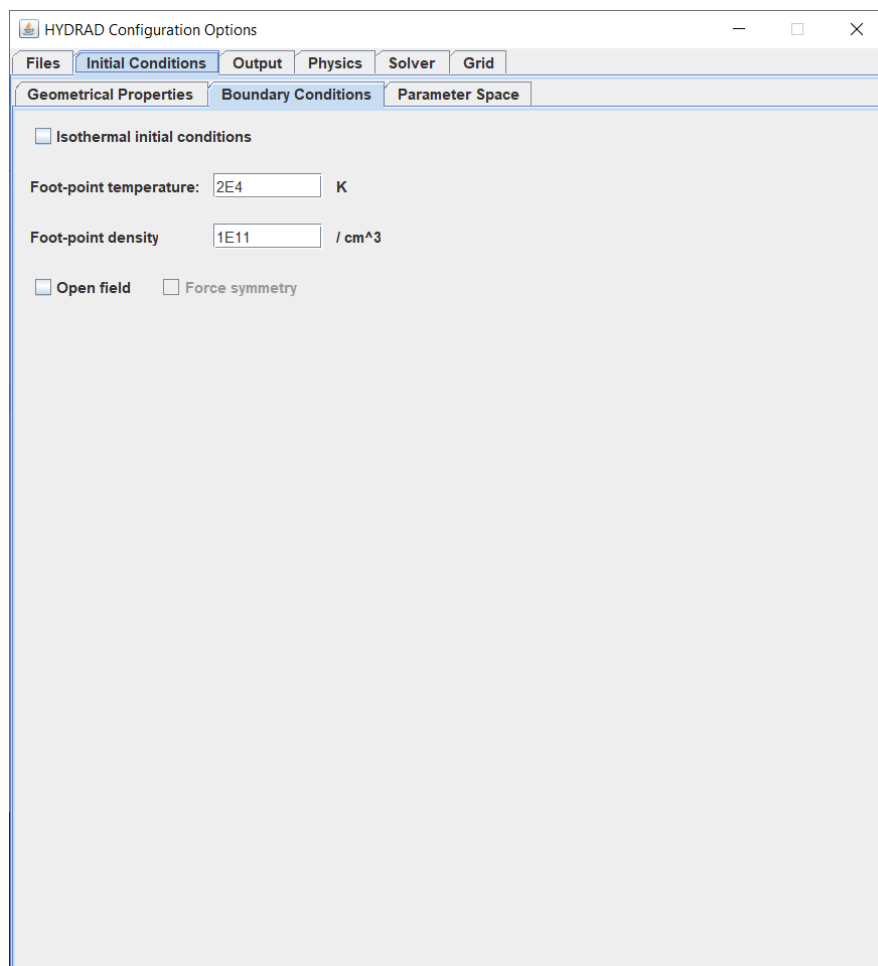


Figure 5 shows the options presented to the user for specifying the boundary conditions to use when solving to find the initial conditions. The foot-point temperature and density corresponds to the values of these quantities at the top of the chromosphere / base of the transition region. In the simplest (default) treatment of the chromosphere, the foot-point

Figure 5: The options for specifying the boundary conditions to use when solving to find the initial conditions.

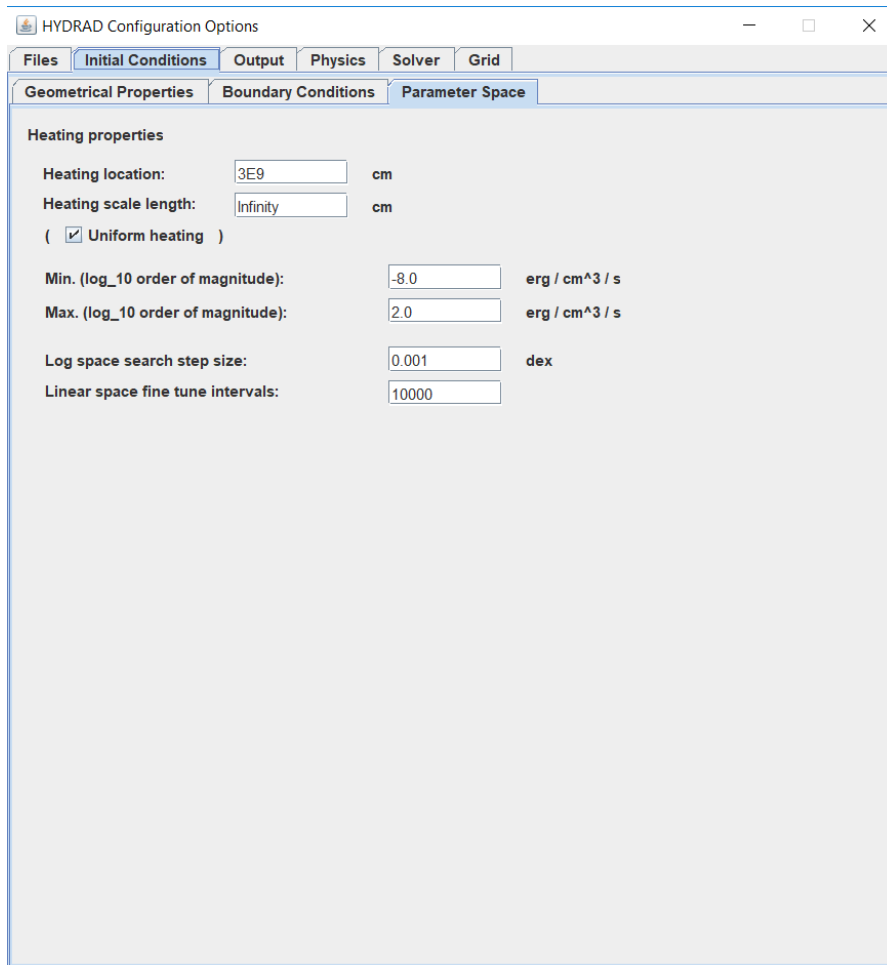
temperature is the isothermal temperature of the chromosphere that is added to each leg of the loop.

If the user wishes to conduct a numerical experiment for an open flux tube (e.g. a solar wind model) then selecting the “Open field” check box instructs the initial conditions code to calculate a hydrostatic, open field solution. As previously discussed, the properties of the flux tube with respect to the field-aligned gravitational acceleration and magnetic field strength can be user-specified as described in Section 5.5. Otherwise, the flux tube is assumed to be vertical and uniform in cross-section. The upper boundary is open, allowing mass and energy to flow through it and leave the domain. The

“Force symmetry” check box allows the open flux tube to behave as one half of a closed flux tube by enforcing a zero flux condition across the upper boundary. This makes evolving symmetrical solutions more efficient by halving the domain size.

If the “Isothermal initial conditions” box is selected then the temperature gradient is set to zero throughout the initial atmosphere. Be aware that the low temperature of the chromosphere yields a relatively short density scale height for long loops, which can lead to precipitously low number densities at coronal altitudes. In addition to issues concerning the validity of the collisional / fluid approximation in this regime, even gentle heating of such initial conditions can sharply raise the temperature and lead to extremely short time-scales associated with thermal conduction and commensurately short integration time-steps. While this option obviates the need for any “background” heating, the user is advised to experiment with care and to restrict the loop length relative to the density scale height.

3.3 Parameter Space for Solutions



The options for configuring the solver which finds the heating parameter values for the set of initial conditions satisfying the specified geometrical properties and boundary conditions are shown in Figure 6. This is the commonly referred to “background” heating, which maintains the initial conditions in hydrostatic equilibrium unless /

Figure 6: The options for configuring the solver which finds the heating parameter values for the initial conditions.

until they are driven out-of-equilibrium by (for example) an energy release or when an external force is applied to the system.

The heating location specifies the location along the loop at which the energy release is centered and the scale length is the width of the Gaussian spatial distribution of energy in the field-aligned direction. If the “Uniform heating” box is ticked then the scale length is infinite and it doesn’t matter where the heating is located, otherwise the heating location is the location of peak volumetric heating.

The “Min.” and “Max.” quantities specify the range of volumetric heating values the solver searches through in order to find the solution for the initial conditions which best satisfies the user requirements. The default range (Figure 6) is 10^{-8} to 10^2 erg cm⁻³ s⁻¹ which is sufficient for essentially all

conceivable (physical) initial conditions. The speed of the solver can be increased by narrowing this range if the solution is known to fall within it.

The “Log space search step size” box instructs the initial conditions solver to evaluate the volumetric heating values in steps of 0.001 dex by default. E.g. 10^{-8} , $10^{-7.999}$, ..., $10^{1.999}$, 10^2 10^{-8} to 10^2 erg cm⁻³ s⁻¹.

When the initial conditions solver has located an approximate solution (it doesn't necessarily search the entire range specified) then it takes much finer steps within (by default) a 0.001 dex window of that solution. The “Linear space fine tune intervals” box instructs the solver how many steps to take within this window in order to find the best solution. The speed of the solver can be increased by reducing the number of steps or the accuracy (in terms of satisfying the boundary conditions) of the solution can be increased by increasing the number of steps.

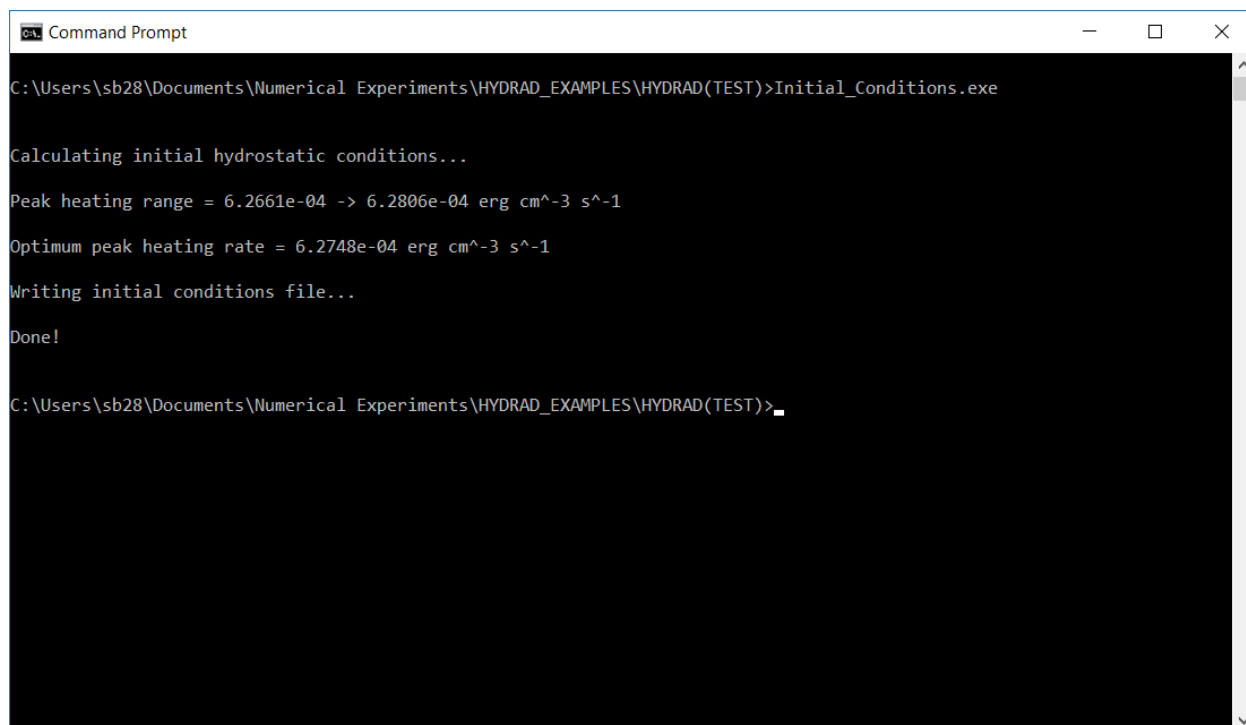
The default values have been found to be adequate in essentially all cases.

4. Output

4.1 Runtime Information

4.1.1 Initial Conditions

When the parameter values have been chosen for calculating the set of initial conditions then clicking “Generate Initial Conditions” under the “Files” GUI tab will write the necessary configuration and source code files, and run the initial conditions solver (Section 2.5).



```
Command Prompt
C:\Users\sb28\Documents\Numerical Experiments\HYDRAD_EXAMPLES\HYDRAD(TEST)>Initial_Conditions.exe

Calculating initial hydrostatic conditions...
Peak heating range = 6.2661e-04 -> 6.2806e-04 erg cm^-3 s^-1
Optimum peak heating rate = 6.2748e-04 erg cm^-3 s^-1
Writing initial conditions file...
Done!

C:\Users\sb28\Documents\Numerical Experiments\HYDRAD_EXAMPLES\HYDRAD(TEST)>
```

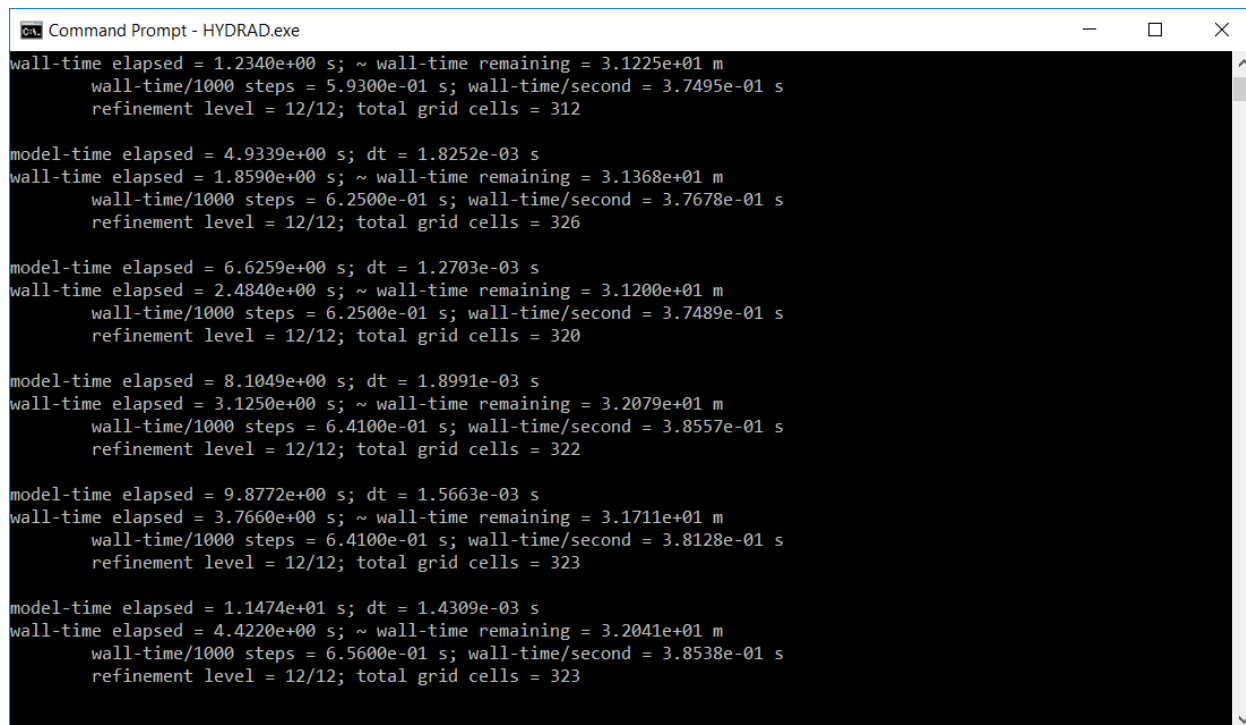
Figure 7: Clicking “Generate Initial Conditions” should produce output resembling what is shown here (see Figure 2).

The output in Figure 7 shows the region of the heating parameter space in which a solution, satisfying the conditions provided by the user, has been found. The peak volumetric heating rate (volumetric heating rate at the specified heating location) that produces a valid solution is first found to be in the range 6.2661×10^{-4} to 6.2806×10^{-4} erg cm⁻³ s⁻¹ and then, by taking small steps within this range to fine tune the solution (Section 3.3), the optimum peak volumetric heating rate, which produces the best match solution, is calculated as 6.2661×10^{-4} erg cm⁻³ s⁻¹. These heating values will change, depending on what the user-specified parameter values are.

The code outputs several data files (Section 2.5), which are described in Sections 4.2.1 – 4.2.2.

4.1.2 HYDRAD

When the parameter values for configuring HYDRAD (Sections 5 – 7) have been chosen then clicking “HYDRAD” under the “Files” GUI tab will write the necessary configuration and source files, and launch the hydrodynamics code. If the user deselects the checkbox “Use default initial conditions” then the set of initial conditions identified by the path and filename in the associated text box will be used by HYDRAD when it runs. This is useful for continuing a run from the last set of output files, for example, if it was interrupted for some reason (a power outage, OS installing updates and rebooting, etc.)



```

wall-time elapsed = 1.2340e+00 s; ~ wall-time remaining = 3.1225e+01 m
wall-time/1000 steps = 5.9300e-01 s; wall-time/second = 3.7495e-01 s
refinement level = 12/12; total grid cells = 312

model-time elapsed = 4.9339e+00 s; dt = 1.8252e-03 s
wall-time elapsed = 1.8590e+00 s; ~ wall-time remaining = 3.1368e+01 m
wall-time/1000 steps = 6.2500e-01 s; wall-time/second = 3.7678e-01 s
refinement level = 12/12; total grid cells = 326

model-time elapsed = 6.6259e+00 s; dt = 1.2703e-03 s
wall-time elapsed = 2.4840e+00 s; ~ wall-time remaining = 3.1200e+01 m
wall-time/1000 steps = 6.2500e-01 s; wall-time/second = 3.7489e-01 s
refinement level = 12/12; total grid cells = 320

model-time elapsed = 8.1049e+00 s; dt = 1.8991e-03 s
wall-time elapsed = 3.1250e+00 s; ~ wall-time remaining = 3.2079e+01 m
wall-time/1000 steps = 6.4100e-01 s; wall-time/second = 3.8557e-01 s
refinement level = 12/12; total grid cells = 322

model-time elapsed = 9.8772e+00 s; dt = 1.5663e-03 s
wall-time elapsed = 3.7660e+00 s; ~ wall-time remaining = 3.1711e+01 m
wall-time/1000 steps = 6.4100e-01 s; wall-time/second = 3.8128e-01 s
refinement level = 12/12; total grid cells = 323

model-time elapsed = 1.1474e+01 s; dt = 1.4309e-03 s
wall-time elapsed = 4.4220e+00 s; ~ wall-time remaining = 3.2041e+01 m
wall-time/1000 steps = 6.5600e-01 s; wall-time/second = 3.8538e-01 s
refinement level = 12/12; total grid cells = 323

```

Figure 8: Clicking “HYDRAD” should produce output resembling what is shown here (see Figure 3).

The output in Figure 8 shows the information HYDRAD outputs to the terminal screen as it time-steps. This allows the user to monitor the progress of the code and identify any potential problems (e.g. extremely small time-steps). The first line shows the model time (the time on the Sun, for example) that has elapsed and the current time-step (dt). HYDRAD integrates on the smallest of the time-scales associated with the physical processes included

in the calculation, to ensure that the principal time-scale of the system is always captured. Depending on computational resources available to the user, time-steps less than 10^{-7} or 10^{-8} seconds may render the calculation intractable and the user may choose to mitigate this by re-running the experiment with fewer grid cell refinement levels and / or adopting other measures (Sections 6 – 7).

The second line in Figure 8 shows the real (wall) time that has elapsed since the experiment began and an estimate for the amount of wall time remaining. This estimate becomes more accurate as time progresses and when dt remains reasonably uniform in time.

The third line gives the wall time taken to perform 1000 time-steps (the number between each terminal screen update) and the amount of wall time required to evolve the solution by 1 second of (e.g.) solar time. This provides information on the execution speed of the code and can be used to optimize / fine-tune its performance. Note that under the “Output” GUI tab the user can choose the number of time-steps between updating the terminal screen by modifying the value in the line “Show progress every N time steps” (where $N = 1000$ by default). Since writing to the screen can slow execution if done too frequently, it is necessary to find a balance between speed and the rate at which progress and performance information is required. In the case of simple calculations, when the code can time-step extremely quickly, then increasing N may result in a significant speed increase.

The final line shows the current maximum grid cell refinement level in the domain relative to the largest allowable refinement level, and the total number of grid cells comprising the domain. If the refinement level is (nearly) always maxed out during a run then it may be that some regions of the solution are under-resolved. However, the user may need to consider the trade-off between accuracy and speed. If the total number of grid cells exhibits very significant variation during a run then it may indicate a lot of small-scale spatial structure emerging in the solution; sometimes this can be physical and should therefore be allowed adequate spatial resolution, but it can also be due to the order of interpolation used to insert grid cells at increasingly high spatial resolution, which can introduce unwanted numerical artifacts. Spatially resolving this small-scale numerical noise can decrease the time-step and increase the runtime substantially. While the interpolation algorithm attempts to minimize it, when it becomes a problem the most

effective way to eliminate it is to enforce linear interpolation (upon grid cell restriction) by choosing “Use linear restriction” under the “Grid” GUI tab.

4.2 Data Files and Formats

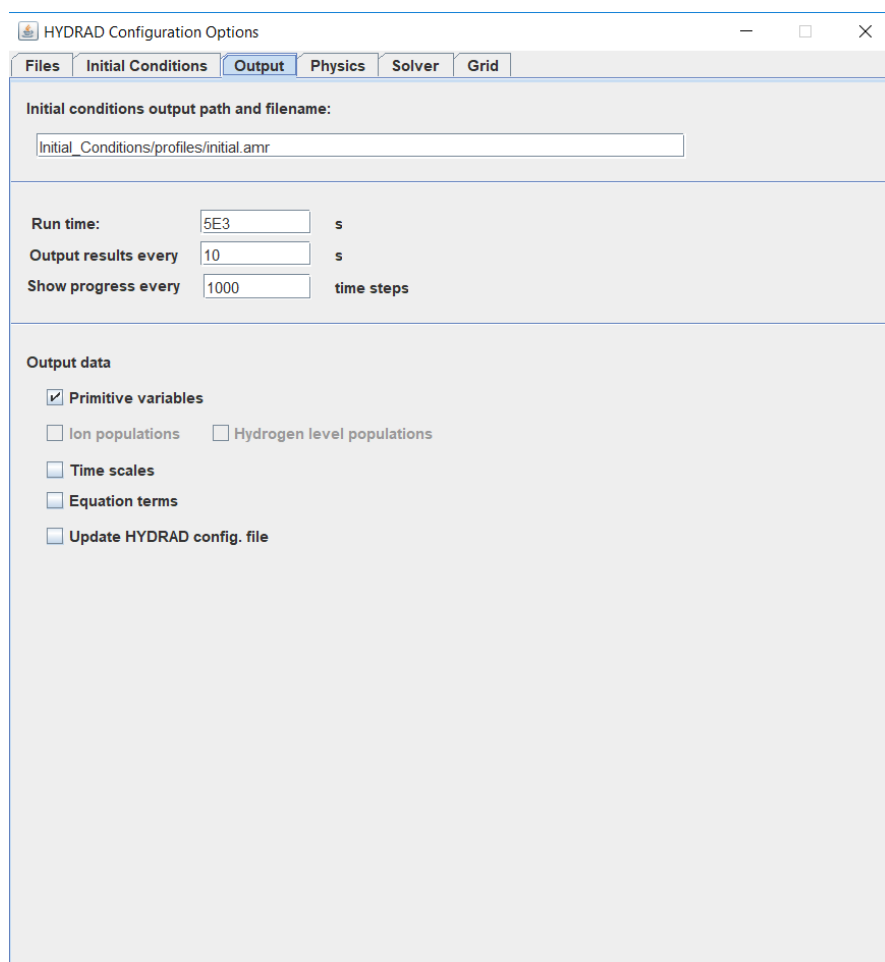


Figure 9: The options for configuring the update rate for performance / progress information, and which data products are written to the “Results/” directory.

The “Output” GUI tab (Figure 9) contains options for choosing the overall (e.g. solar) runtime of the code (“Run time”) and how frequently the evolving solution is written to data files (“Output results every”) for analysis.

Numerical experiments with fast-changing solutions (e.g. solar flares) may require output on a faster cadence (1 second or sub-second timescales) than models of (e.g.) heating in quiescent active region coronal loops, to

capture the details of their evolution.

There are also options under the “Output” GUI tab for selecting which data products HYDRAD writes at each output interval. The “.amr” files are always written, since these are used to set up the HYDRAD computational domain and allow it to be run from the most recent set of output data in case of interruption. The other files are user-selectable and some options are disabled (greyed out), depending upon how HYDRAD has been configured. These are described in Sections 4.2.1 and 4.2.3.

When the checkbox next to “Update HYDRAD config. file” is selected then the configuration file “HYDRAD/config/config.cfg” is updated at each output interval with the most recent “.amr” file, such that HYDRAD can simply be re-run from the command line (by calling HYDRAD.exe) and will start again at that point, following any interruption. This is useful when executing on time-limited High Performance Computing (HPC) clusters, where codes are halted and then automatically re-queued, by the scheduler, to reduce wait times for all users.

4.2.1 Common Files

The files common to both the initial conditions code and HYDRAD are the “.amr”, “.phy” and the files which contain the fitting coefficients for the piecewise polynomial fits to the field-aligned gravitational acceleration and the magnetic field strength. All of the files described in Sections 4.2.1 – 4.2.3 are plain ASCII text files, which can be read / written with any suitable text editor.

“.amr” Files

These are the main results files, which contain the information necessary to reconstruct the grid and the solution at the corresponding time. The file contains the conserved quantities and its format is:

Time (s)							
File #							
Length (cm)							
Total Cells							
Field-aligned Position (cm)	Cell Width (cm)	Particle Mass Density* (g cm ⁻³)	Momentum Density (g cm ⁻² s ⁻¹)	Electron Energy Density (erg cm ⁻³)	Ion Energy Density (erg cm ⁻³)	Refinement Level (RL)	Series of IDs for paired cells at each RL

* Note that when solving for an optically-thick chromosphere (Section 5.2) the “Mass Density” column is split into two columns; “Electron Mass Density” and “Particle Mass Density”, where “particle” refers to the neutral / ionized element (typically mass-adjusted hydrogen to account for the non-negligible helium abundance and its influence on the plasma bulk behaviour).

The initial conditions code writes “Initial_Conditions/profiles/initial.amr”, which contains all of the information necessary to create the domain and populate it with the solution at time zero. This file is output at very high spatial

resolution, where every grid cell can be maximally refined or close to it (Section 7.1), and so it can be quite large. This is necessary to ensure the force balance is correctly resolved and the solution guaranteed to be in steady-state, such that it can be perturbed / driven at time zero without the need for further relaxation over several sound wave crossing times. The first operation HYDRAD performs upon loading “initial.amr” into memory is to coarsen (prolong) the grid, consistent with the user conditions for restriction and prolongation (Section 7.2), such that high spatial resolution is placed only where it is needed and the grid is optimized for efficient computation. The grid then dynamically adapts as the solution evolves.

HYDRAD writes “Results/profileN.amr”, where N corresponds to the file number, at each output interval. Thus, “Results/profile0.amr” contains the grid properties and solution (conserved variables) at time zero (essentially the same as “initial.amr” but at the optimal, non-uniform spatial resolution), “profile1.amr” contains this information at the first output interval (e.g. 10 seconds), “profile2.amr” at the next output interval (e.g. 20 seconds), and so forth. These “.amr” files can be used to restart HYDRAD from the corresponding time in the event of interruption.

“.phy” Files

These files contain the physically measurable quantities (primitive variables) and its format is:

Field-aligned Position (cm)	Bulk Velocity (cm s^{-1})	Sound Speed (cm s^{-1})	Electron Number Density (cm^{-3})	Particle# Number Density (cm^{-3})	Electron Pressure (dyne cm^{-2} or erg cm^{-3})	Particle Pressure (dyne cm^{-2} or erg cm^{-3})	Electron Temperature (K)	Particle Temperature (K)	Electron Heat Flux (erg $\text{cm}^{-2} \text{s}^{-1}$)	Particle Heat Flux (erg $\text{cm}^{-2} \text{s}^{-1}$)
-----------------------------	--------------------------------------	------------------------------------	--	---	--	--	--------------------------	--------------------------	--	--

Note that “particle” refers to the neutral / ionized element (typically mass-adjusted hydrogen to account for the non-negligible helium abundance and its influence on the plasma bulk behaviour).

The initial conditions code writes “Initial_Conditions/profiles/initial.amr.phy” (at lower resolution than the corresponding “.amr” file), which the user can check before running HYDRAD to ensure the solution is satisfactory.

HYDRAD writes “Results/profileN.phy”. In principle, all of the quantities in a “.phy” file can be calculated from the quantities in the corresponding “.amr” file, with the exception of the heat fluxes (which involve a temperature

gradient, but can still be estimated if their precise values are not needed), which means they need not be output if storage and / or data transmission is an issue.

Piecewise Polynomial Fit Files

The initial conditions code and HYDRAD can solve for arbitrary geometry (defined by the field-aligned gravitational acceleration) and variable flux tube cross-section (defined by the inverse of the magnetic field strength). The field-aligned profiles of these quantities are given by piecewise polynomial fits to tabulated data, where the fitting coefficients for each fit are stored in files that are read by the codes upon execution. The paths and names of these files are specified under the “Physics/Flux Tube” GUI tab (Section 5.5). When neither checkbox is selected the defaults (semi-circular / vertical geometry, constant cross-section) are used by the initial conditions code and HYDRAD. These properties can be changed between calculating the initial conditions and launching HYDRAD.

Note: the initial conditions code is robust and generally does a good job of finding steady-state solutions, however, in cases of awkward flux tube geometries and / or unusual behavior in the cross-section, it may be preferable to solve for the defaults when calculating the initial conditions and then allow HYDRAD to find a new solution by relaxing them subject to arbitrary geometry / cross-section.

HYDRAD also includes a utility to calculate piecewise fits to tabulated data (e.g. field-aligned gravitational acceleration and magnetic field strength) and output the files containing the fitting coefficients. This can be found in the directory “Resources/Utils/generatePieceWiseFit/”. Opening the file “source/main.cpp” reveals the following line of code:

```
pMagneticField = new CPieceWiseFit( (char*)"fits/B(closed).txt", (char*)"fits/initial.amr.B" );
```

The first path/filename (“fits/B(closed).txt”) is an input file of tabulated field-aligned magnetic field strength data for a closed flux tube (there is also a table of data for an open flux tube in the “fits/” directory). The second path/filename (“fits/initial.amr.B”) is the output file that will contain the fitting coefficients for the piecewise polynomial fit to the magnetic field data and is given to the initial conditions code and HYDRAD as described above (and in Section 5.5).

Note that the tabulated data does not have to be magnetic field data; it can also be field-aligned gravitational acceleration. This piecewise fitting utility can be used to fit any tabulated data the user has to hand and the path / filenames can be changed to point to the user's own input and output files.

The format of the input file of tabulated data is:

Order of fit (6th order in the example file)
Number of sub-domains (number of regions the data is divided into, which are fitted individually; 1 in the example)
List of sub-domain boundaries (since there is only 1 sub-domain the x-range is $0 \rightarrow 1$ in the example; if there were 2 then the range might be $0 \rightarrow 0.5 \rightarrow 1$ or $0 \rightarrow 0.33 \rightarrow 0.66 \rightarrow 1$ in the case of 3 sub-domains)
Total number of tabulated data points
List of X and Y values (the X values must be normalized to 1)

The format of the file containing the piecewise fitted coefficients is:

Order of fit
Number of sub-domains
List of sub-domain boundaries
List of fitting coefficients for the first sub-domain
Min / Max values of the tabulated data in the first sub-domain
List of fitting coefficients for the second sub-domain
Min / Max values of the tabulated data in the second sub-domain
.
.

The utility can be run by executing the file "build_scripts/build_generatePieceWiseFit.bat" (or by copying the single-line entry in it to the command line) and then executing the compiled file "generatePieceWiseFit.exe".

Of course, the user is also free to write their own utilities to fit tabulated data and write fitted coefficient files.

4.2.2 Initial Conditions

The initial conditions code writes two unique files.

“.sol”

The file format is:

Peak volumetric heating rate corresponding to optimum solution

The initial conditions code writes “Initial_Conditions/profiles/initial.amr.sol”.

“.table”

This file has the same format as the tabulated data input to the piecewise polynomial fitting utility described previous.

The initial conditions code writes “Initial_Conditions/profiles/initial.amr.table” when the default geometry (for gravitational acceleration) is chosen. This can be used as a basis file for modifying the default profile, if desired, and then refitting, or simply ignored.

4.2.3 HYDRAD

HYDRAD can write several user-selectable files, containing different data products, determined by the “Output data” chosen under the “Output” GUI tab (Figure 9). The “.amr” files are always output and the “Primitive variables” are contained in the “.phy” files, described in Section 4.2.1.

“.scl”

These files contain the time scales associated with the physical processes in each grid cell and are written when the “Time scales” checkbox is selected. The file format is:

Field-aligned Position (cm)	Cell Width (cm)	Bulk Transport (s)	Electron Thermal Conduction (s)	Particle* Thermal Conduction (s)	Kinematic Viscosity (s)	Inter-species Collisions (s)	Radiation (s)	Bound ↔ Free** (s)
-----------------------------	-----------------	--------------------	---------------------------------	----------------------------------	-------------------------	------------------------------	---------------	--------------------

+ Note that “particle” refers to the neutral / ionized element (typically mass-adjusted hydrogen to account for the non-negligible helium abundance and its influence on the plasma bulk behaviour).

++ The smallest atomic time scale, which is only available and written to the “.scl” files when solving for the non-equilibrium ionization state.

“.trm”

These files contain the individual terms in each grid cell of the conservation equations solved by HYDRAD, and the time rate-of-change of each conserved quantity. They are written when the “Equation terms” checkbox is selected. The file format (starting from the second column) is:

	$s \text{ (cm)}$							
Mass Cons. [g cm ⁻³ s ⁻¹]	$\frac{\partial \rho}{\partial t}$	$-\frac{\partial}{\partial s}(\rho v)$						
		$-\frac{\partial}{\partial s}(\rho v^2)$	$-\frac{\partial}{\partial s}(P_e + P_i)$	$-\rho g$	$\frac{\partial}{\partial s}\left(\eta \frac{\partial v}{\partial s}\right)$	$\frac{\partial}{\partial s}\left(\eta_{\text{Num}} \frac{\partial v}{\partial s}\right)$		
Mom. Cons. [g cm ⁻² s ⁻¹]	$\frac{\partial}{\partial t}(\rho v)$	$-\frac{\partial}{\partial s}[(E_e + P_e)v]$	$-\frac{\partial F_{ce}}{\partial s}$	0	$\frac{k_B n}{\gamma - 1} v_{ei}(T_i - T_e)$	H	-R	$v \frac{\partial P_e}{\partial s}$
Electron Energy Cons. [erg cm ⁻³ s ⁻¹]	$\frac{\partial E_e}{\partial t}$	$-\frac{\partial}{\partial s}[(E_i + P_i)v]$	$-\frac{\partial F_{ci}}{\partial s}$	$-\rho g v$	$\frac{k_B n}{\gamma - 1} v_{ei}(T_e - T_i)$	H	0	$-v \frac{\partial P_e}{\partial s}$
Particle Energy Cons. [erg cm ⁻³ s ⁻¹]	$\frac{\partial E_i}{\partial t}$	$-\frac{\partial}{\partial s}[(E_e + P_e)v]$	$-\frac{\partial F_{ci}}{\partial s}$	$-\rho g v$	$\frac{k_B n}{\gamma - 1} v_{ei}(T_e - T_i)$	H	0	$-v \frac{\partial P_e}{\partial s}$

$\left(\frac{s\rho}{n\rho}\right)\frac{s\rho}{\rho}$	0			
$\left(\frac{s\rho}{n\rho}\right)\frac{s\rho}{\rho}$	0			

A complete description of the equation terms is available in Bradshaw & Cargill (2013)¹³.

“.ine”

These files contain the non-equilibrium ion populations, in each grid cell, of the elements selected by the user to be solved out-of-equilibrium. This output option becomes available when either a full non-equilibrium radiation calculation is chosen or the user chooses to solve for the ionization state decoupled from the conservation equations under the “Physics/Radiation” GUI tab (Sections 5.2.3 – 5.2.4). The elements to be included in the radiation calculation are listed in the file “Radiation_Model/config/elements_neq.cfg”, along with the sets of ion emissivities, element abundances, and ionization / recombination data to use (Section 5.2.5). The “elements_neq.cfg” file is written by the GUI when the “Generate Initial Conditions” and “HYDRAD” buttons under the “Files” GUI tab are clicked. The format of the “.ine” file is:

Field-aligned Position (cm)				
Atomic Number of First Element	Neutral Population	Singly Ionized Population	...	Fully Ionized Population
Atomic Number of Next Element	Neutral Population	Singly Ionized Population	...	Fully Ionized Population
.
Field-aligned Position (cm)				

Note that the ion populations are normalized to 1.0 in the “.ine” files.

The equilibrium ion populations are not output because they can trivially be obtained from tables of ionization data by interpolation as a function of electron temperature (and density when density-dependent rate data is available).

¹³ Bradshaw, S. J., & Cargill, P. J. “The Influence of Numerical Resolution on Coronal Density in Hydrodynamic Models of Impulsive Heating”, 2013, Astrophysical Journal, 770, 12

“.Hstate”

These files contain the level populations, in each grid cell, of the 6-level hydrogen atom that can be solved to more accurately calculate the electron number density in the chromosphere. The first five levels are ground and excited states, and the sixth level is the ionization continuum. This output option becomes available when the user chooses to include the 6-level hydrogen atom, with an optically-thick photosphere and chromosphere calculation, under the “Physics/Radiation” GUI tab (Section 5.2.5). The file format is:

Field-aligned Position (cm)	Ground State Population	First Excited State Population	...	Ionized State Population
Field-aligned Position (cm)

Note that the level populations are normalized to 1.0 in the “.Hstate” files.

5. Configuring HYDRAD's Physics Model

The physics capabilities of HYDRAD, available under the “Physics” GUI tab, are described in this Section. HYDRAD is designed to be a powerful numerical hydrodynamics code, incorporating all of the key field-aligned physics processes one might expect to operate in the solar atmosphere, but its extreme configurability and flexibility also allow it to be put to other purposes. If a physical system can be reduced to a problem of modeling fluid flow with a dominant degree of freedom, then HYDRAD (in some cases with minor modifications to the code) can address it. Nonetheless, here the focus will be on applications to the solar atmosphere.

5.1 Heating

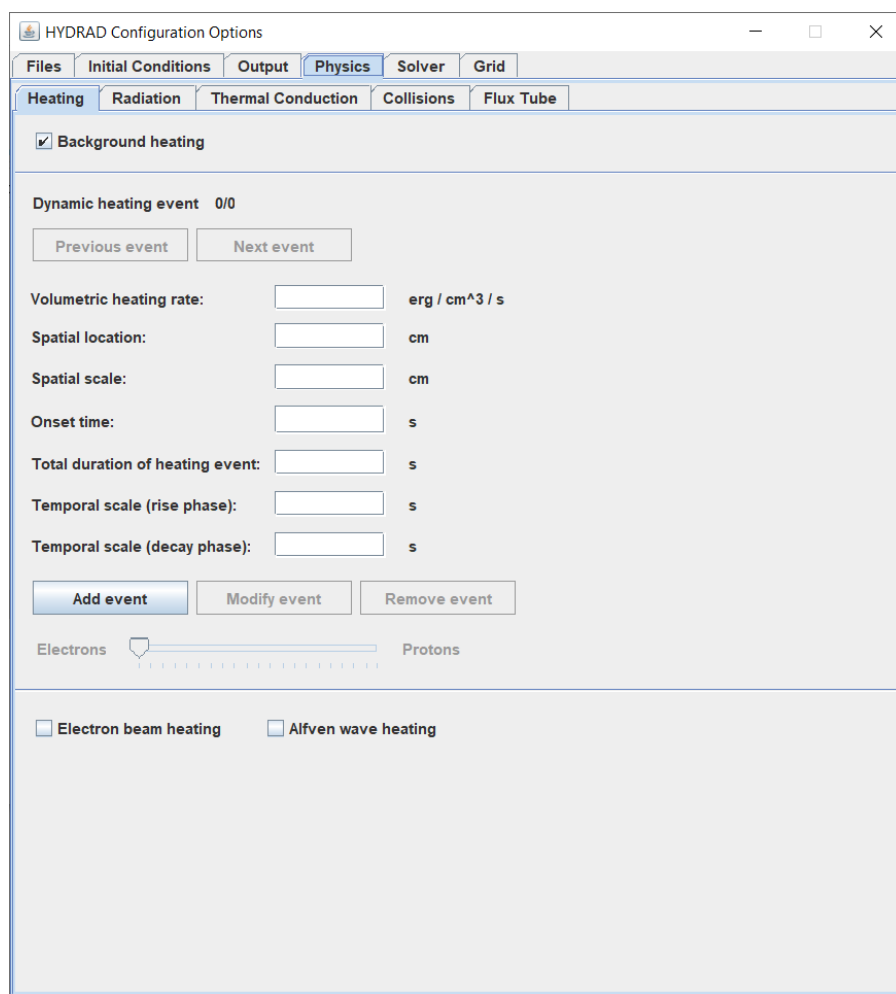


Figure 10: The options for configuring the heating properties used to emulate the driving influence of the underlying mechanisms to study the plasma response.

The coronal heating problem is one of the most well-known of the unsolved problems in astrophysics. The question of how the coronae of magnetically active stars reach temperatures several orders of magnitude greater than their surface temperatures. The scientific consensus has converged on magnetism playing a key role. Specifically, the conversion of magnetic energy

into thermal and kinetic energy, leading to plasma heating and particle acceleration. While the partition of energy among these forms is uncertain, the mechanism of energy release is believed to be magnetic reconnection and / or wave breaking (perhaps with some form of turbulent energy dissipation in each case), both of which are expected to produce a spatially localized (a few thousand km) and impulsive (a few seconds to minutes) profile of energy release. While it is not possible to account for the precise physics of these mechanisms in field-aligned models (nor, indeed, in higher-dimensional fluid-based models), their driving influence can be satisfactorily emulated through the careful choice of a phenomenological expression designed to capture the properties of the energy deposition.

5.1.1 Background Heating

Figure 10 shows the options available to the user under the “Physics/Heating” GUI tab. Leaving the “Background heating” box checked ensures that the steady heating used to calculate the initial conditions is always applied. With no other heating added, this option maintains the initial conditions in hydrostatic equilibrium and guarantees the initial conditions are ultimately recovered following impulsive heating. If this option is unchecked, with no other heating added, the plasma cools.

5.1.2 Phenomenological Heating

The “Dynamic heating event” section allows the user significant flexibility to define one or a series of impulsive heating events localized or otherwise in space and time. The phenomenological heating expression is:

$$H(s, t) = H_{\text{Max}} \exp \left[-\frac{(s - s_0)^2}{2L_H^2} \right] f(t)$$

where H_{Max} is the maximum volumetric heating rate ($\text{erg cm}^{-3} \text{ s}^{-1}$) at location s_0 , and L_H is the Gaussian (spatial) width of the event. $f(t)$ is the temporal envelope of the event.

Volumetric heating rate:	0.1
Spatial location:	3E9
Spatial scale:	2E8
Onset time:	0.0
Total duration of heating event:	60.0
Temporal scale (rise phase):	30.0

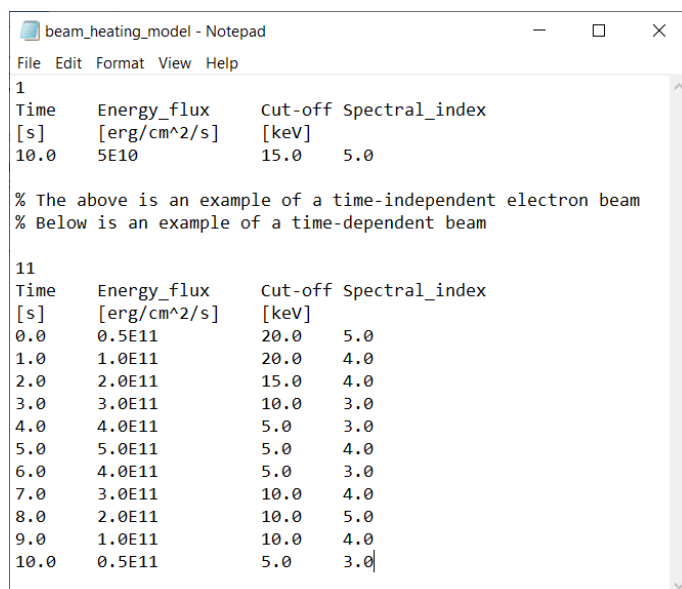
Temporal scale (decay phase):	30.0
-------------------------------	------

Table 1: *Parameter values for a localized and impulsive heating event.*

For example, a reasonably powerful impulsive heating event, localized to a couple of thousand km at the apex of a 60 Mm (full length) coronal loop and of one minute duration, with a triangular temporal profile (linear rise and decay), would be specified by the parameter values given in Table 1. Once the parameter values have been entered into the text boxes, the user clicks “Add event” and the “Dynamic heating event” counter increments. The user can then repeat this procedure to add more events. The heating events can be navigated using the “Previous event” and “Next event” buttons. Clicking “Remove event” removes the currently selected event. If a parameter value is modified then the user **MUST** click “Modify event” to save the changes. The slide bar allows the user to specify the energy partition between electrons and the dominant element (in this case hydrogen / protons): to the left-hand extreme, electrons receive all of the energy; and to the right-hand extreme, protons receive all of the energy. The selection is applied to all of the heating events.

5.1.3 Electron Beam Heating

Electron beam heating in (for example) solar flare models can be deployed in HYDRAD by selecting the “Electron beam heating” check box. The user must then modify the plain text file “Heating_Model/config/beam_heating_model.cfg” to configure the beam parameter values for their own use case.



```

1
Time      Energy_flux      Cut-off Spectral_index
[s]       [erg/cm^2/s]          [keV]
10.0      5E10                15.0    5.0

% The above is an example of a time-independent electron beam
% Below is an example of a time-dependent beam

11
Time      Energy_flux      Cut-off Spectral_index
[s]       [erg/cm^2/s]          [keV]
0.0      0.5E11            20.0    5.0
1.0      1.0E11            20.0    4.0
2.0      2.0E11            15.0    4.0
3.0      3.0E11            10.0    3.0
4.0      4.0E11            5.0     3.0
5.0      5.0E11            5.0     4.0
6.0      4.0E11            5.0     3.0
7.0      3.0E11            10.0    4.0
8.0      2.0E11            10.0    5.0
9.0      1.0E11            10.0    4.0
10.0     0.5E11            5.0     3.0

```

Figure 11: The configuration file “beam_heating_model.cfg” which specifies the parameter values for electron beam heating.

Figure 11 shows the contents and structure of the “beam_heating_model.cfg” configuration file. The beam parameters can be fixed or time-dependent during the period when the beam operates. The electrons are currently injected halfway along the flux tube and beams travel down each leg. A beam with fixed parameter values, configured to last for 10 seconds, carrying an energy flux of $(\text{erg cm}^{-2} \text{ s}^{-1})$, with a low-energy cut-off at 15 keV and a spectral index of 5, is shown in the first four lines of Figure 11.

The “1” in the first line indicates there is only one row of entries for the parameter values, marking the beam as time-independent.

To configure a time-dependent beam the first eight rows in Figure 11 can be deleted, such that the file begins with the number “11” (or however many rows of time-dependent parameter values are required), indicating there are eleven rows of time-dependent values. The beam heating code linearly interpolates between quantities at times intermediate to the entries on each row. The time-dependent example in Figure 11 shows a beam with all of its properties varying for 10 seconds, before switching off.

The energy associated with ionizing hydrogen and helium is accounted for in the electron energy balance when beam heating is active.

The physics implementation of electron beam heating is described in Reep et al. (2013)¹⁴.

5.1.4 Alfvén Wave Heating

At present the “Alfvén wave heating” option is inactive. However, a description of its implementation is given by Reep et al. (2018)¹⁵. This Section will be updated when the wave heating functionality has been integrated into the publicly available HYDRAD code.

¹⁴ Reep, J. W., Bradshaw, S. J., & McAteer, R. T. J. “On the Sensitivity of the GOES Flare Classification to Properties of the Electron Beam in the Thick Target Model”, 2013, *Astrophysical Journal*, 778, 76

¹⁵ Reep, J. W., Russell, A. J. B., Tarr, L. A., & Leake, J. E. “A Hydrodynamic Model of Alfvénic Wave Heating in a Coronal Loop and Its Chromospheric Footpoints”, 2018, *Astrophysical Journal*, 853, 101

5.2 Radiation

HYDRAD was originally conceived and designed to provide a comprehensive treatment of the optically-thin radiation emitted by the upper solar atmosphere. The main focus of this effort was to account for non-equilibrium ion populations in the radiation calculation, which necessitates solving the detailed balance equations for the ionization state of each significant radiating element and then recalculating the radiative energy losses due to these populations at each time step.

5.2.1 Power-law Radiation

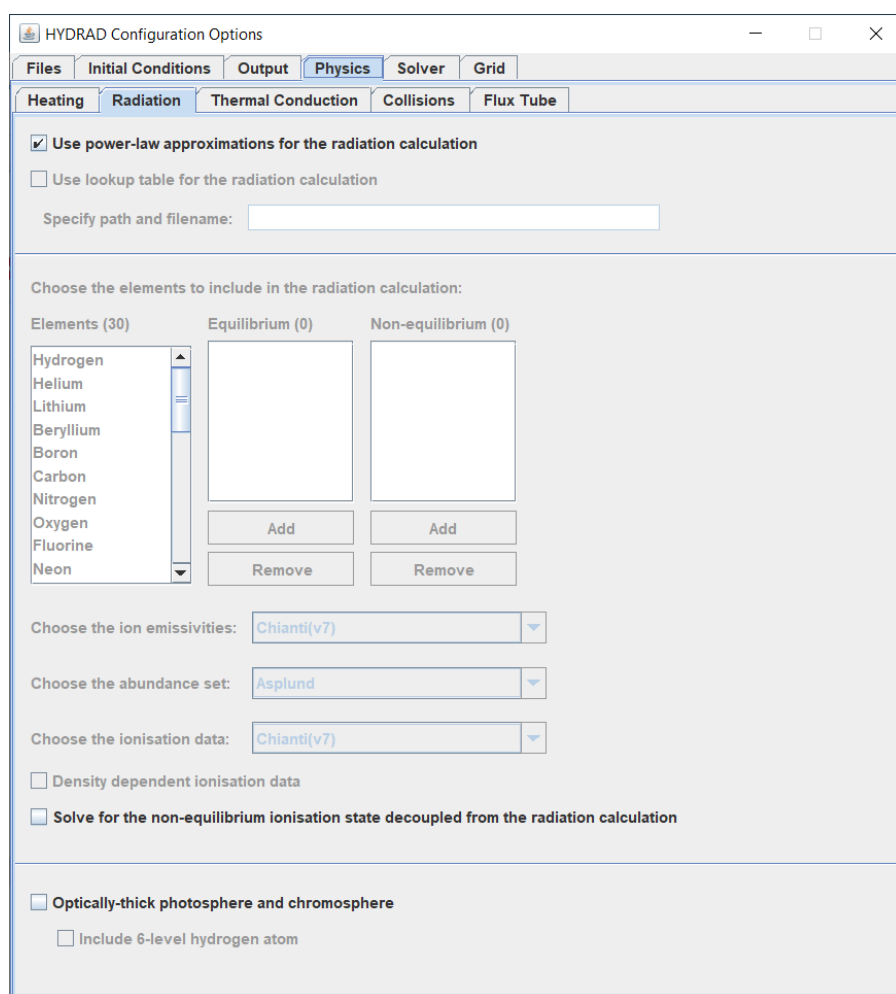


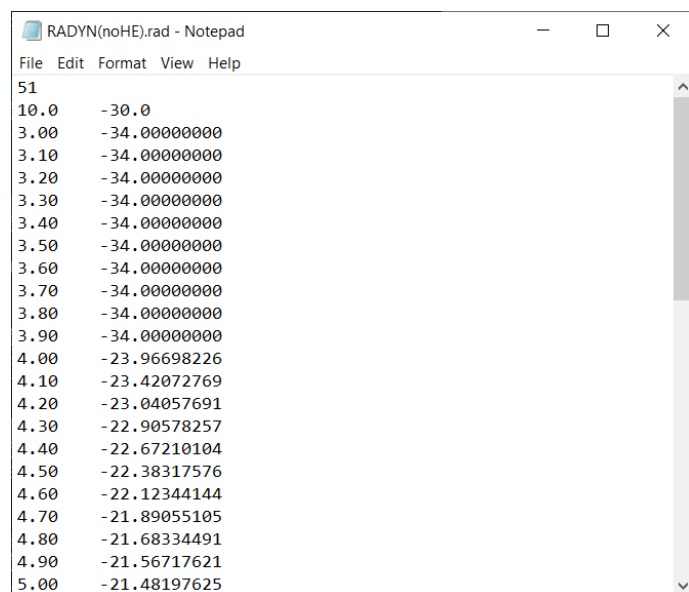
Figure 12: The options for configuring energy loss by radiation in HYDRAD.

By default, a simple treatment of the radiative losses using piecewise power-law fits to an emissivity curve is selected (Figure 12). This is the fastest way to calculate radiation and suffices for calculations where more precise details of the radiative losses are not required. The set of fits used by HYDRAD are those also used by the EBTEL¹⁶ code. When the “Use power-law approximations for the radiation calculation” box is

¹⁶ Cargill, P. J., Bradshaw, S. J., & Klimchuk, J. A. “Enthalpy-based Thermal Evolution of Loops. II. Improvements to the Model”, *Astrophysical Journal*, 752, 161

unchecked, under the “Physics/Radiation” GUI tab, then other options to calculate the radiative losses become available.

5.2.2 Lookup Tables for Radiation



Temperature (K)	Emissivity (erg cm³ s⁻¹)
10.0	-30.0
3.00	-34.00000000
3.10	-34.00000000
3.20	-34.00000000
3.30	-34.00000000
3.40	-34.00000000
3.50	-34.00000000
3.60	-34.00000000
3.70	-34.00000000
3.80	-34.00000000
3.90	-34.00000000
4.00	-23.96698226
4.10	-23.42072769
4.20	-23.04057691
4.30	-22.90578257
4.40	-22.67210104
4.50	-22.38317576
4.60	-22.12344144
4.70	-21.89055105
4.80	-21.68334491
4.90	-21.56717621
5.00	-21.48197625

Figure 13: The format for radiation lookup files used by HYDRAD.

Checking the “Use lookup table for the radiation calculation” allows the user to specify the path and filename of a table of emissivities given as a function of temperature.

The example in Figure 13 shows that there are 51 rows of temperature (K) and emissivity (erg cm³ s⁻¹) pairs in the table (both given in log₁₀ form). The lower limit of the temperature range is 10³ K. The second row of the table “10.0 -30.0”

gives the parameter values for a linear equation (of the form $y = mx + c$) which maps the (log₁₀) temperature to the array index of the stored quantity, acting as a hashing function for fast, non-sequential access to the table. In this example: x is the temperature; $m = 10.0$; and $c = -30.0$. Thus, $(10.0 \times 3.0) + (-30.0) = 0$ (the first temperature is mapped to the first array index). A linear interpolation is performed between adjacent quantities in the table to find the emissivities for intermediate temperatures.

5.2.3 User-defined Radiation

Where the full power of HYDRAD’s radiation calculation becomes evident is allowing the user to build their own radiative loss function. The user can select from a full list of elements from hydrogen (Z=1) to zinc (Z=30) to include, and they can also choose between sets of element abundances and tabulated ion emissivities, and ionization / recombination rate data. Finally, the user can choose which elements to assume are in equilibrium and which to treat in a time-dependent manner by solving for their non-equilibrium ionization states and associated radiative losses. Choosing which elements radiate in equilibrium and which require a full, non-equilibrium treatment can

substantially shorten computation times relative to assuming all of the elements emit out of equilibrium (particularly when some elements contribute significantly more than others).

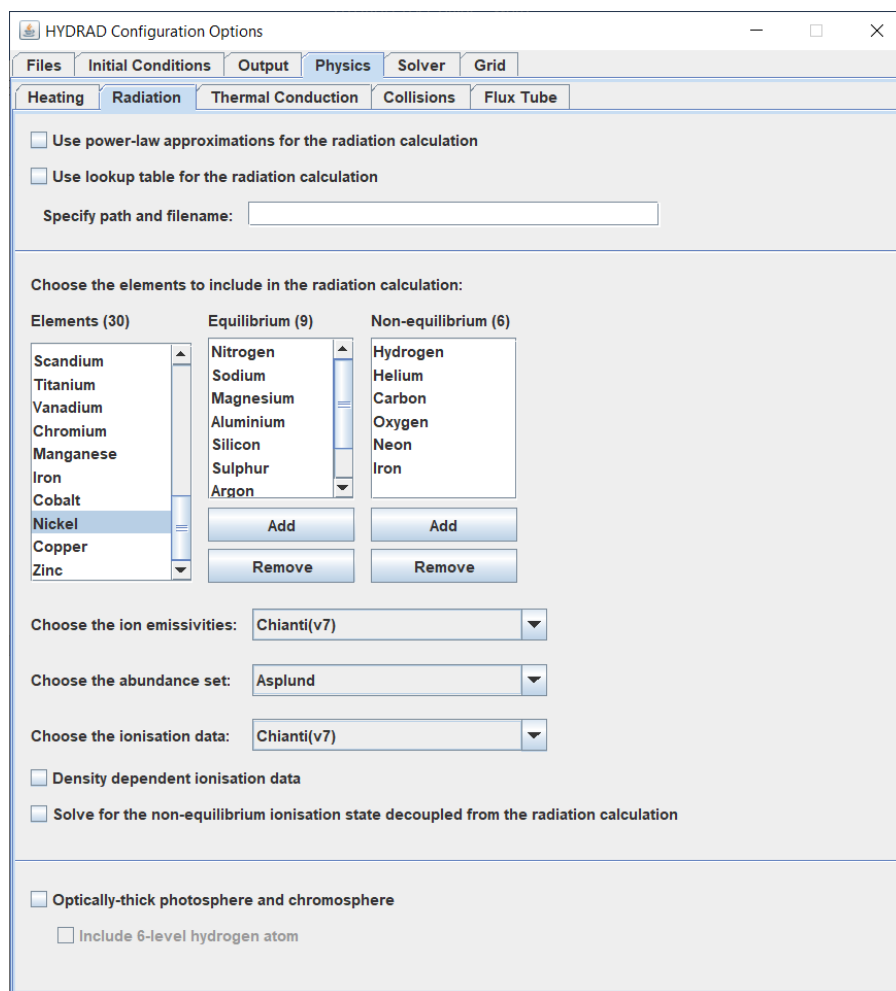


Figure 14: Element selection for a bespoke radiation calculation.

Figure 14 shows a selection of the 15 most important elements with respect to optically-thin radiation from the solar atmosphere. 9 elements will be treated in equilibrium and 6 elements out of equilibrium. The “Equilibrium” and “Non-equilibrium” panels are populated by first selecting the required element in the left-hand panel and then clicking “Add” under the relevant panel. Elements can also be removed from

each panel. It is not necessary to remove an element from one panel before adding it to the other; it will automatically be removed.

Particular sets of ion emissivities, element abundances, and ionization / recombination rate data can be chosen from the drop-down menus. The majority of data is drawn from the well-known Chianti¹⁷ atomic database.

If the chosen set of ionization / recombination rate data is density-dependent in addition to temperature-dependent (denoted by a DD appended to the

¹⁷ <https://www.chiantidatabase.org/>

description in the drop-down menu) then the “Density dependent ionization data” box **MUST** be checked.

When solving for non-equilibrium ionization, the option to write the ion populations at each output interval becomes available under the “Output” GUI tab. Selecting the “Use OpenMP” checkbox under the “Files” GUI tab, and experimenting with the “Number of threads” and the “Number of grid cells / thread”, should decrease the runtime when several elements, and / or high Z elements, are chosen.

5.2.4 Decoupling Non-equilibrium Ionization from the Radiation Calculation

In the event that only non-equilibrium ion populations are needed and a detailed radiation calculation is unnecessary (e.g. when synthesizing / forward-modeling emission lines) then a set of elements to be solved for non-equilibrium ionization can be chosen, just as described, and then the box “Solve for the non-equilibrium ionization state decoupled from the radiation calculation” can be checked. The user can then choose between power-law radiative losses or a lookup table. The ion populations will still be calculated for each selected element, but will not be used to calculate the radiative losses. This decoupling can lead to a substantial decrease in the calculation time. Again, “Use OpenMP” may help to decrease the runtime, though the computational demands are not as heavy as when the ionization state is coupled to the radiation calculation.

5.2.5 Optically-thick Photosphere and Chromosphere

While HYDRAD was originally intended for application to the upper solar atmosphere (transition region and corona), and therefore utilized an idealized lower atmosphere comprising an isothermal chromosphere, it is often desirable (necessary, even) to have a more sophisticated treatment of the lower solar atmosphere which follows its structure and radiative properties with greater fidelity. This is particularly true, for example, in the case of solar flare modeling, whereby most of the energy from electron beam heating is deposited directly into the dense chromosphere.

Full radiative transfer calculations for optically-thick layers are very computationally expensive and can increase computation times to the point of intractability, even in low-dimensional models. HYDRAD’s method

represents a compromise with approximate treatments of optically-thick emission from a VAL model C atmosphere and of radiative transport in the solution of the excitation state of a 6-level hydrogen atom to calculate the electron, neutral hydrogen, and proton number densities. This implementation also includes collisions and the contribution from non-thermal electrons when beam heating is active. The ionization state of any other selected elements, whether in / out of equilibrium and coupled to / decoupled from the radiation calculation, is also used to add their contributions to the total electron density. A full description is given by Reep et al. (2019)¹⁸.

Selecting the “Optically-thick photosphere and chromosphere” checkbox switches on the approximate treatment of optically-thick radiation from a VAL model C atmosphere, but assumes the electron density is always at least 1.000144 times the hydrogen density (neutral plus protons) to ensure that a minimum number of electrons are present. Some “Geometrical properties” and “Boundary conditions” entries are also altered under the “Initial Conditions” GUI tab, which necessitates recalculating the initial conditions before running HYDRAD.

Selecting the “Include 6-level hydrogen atom” checkbox, in addition, deploys the more sophisticated electron density calculation described above.

When solving the 6-level hydrogen atom, the option to write its state (the level populations) at each output interval becomes available under the “Output” GUI tab. Selecting the “Use OpenMP” check box under the “Files” GUI tab, and experimenting with the “Number of threads” and the “Number of grid cells / thread”, will decrease the runtime.

¹⁸ Reep, J. W., Bradshaw, S. J., Crump, N. A., & Warren, H. P. “Efficient Calculation of Non-Local Thermodynamic Equilibrium Effects in Multithreaded Hydrodynamic Simulations of Solar Flares”, 2019, *Astrophysical Journal*, 871, 18

5.3 Thermal Conduction

5.3.1 Classical (Spitzer Härm) Thermal Conduction

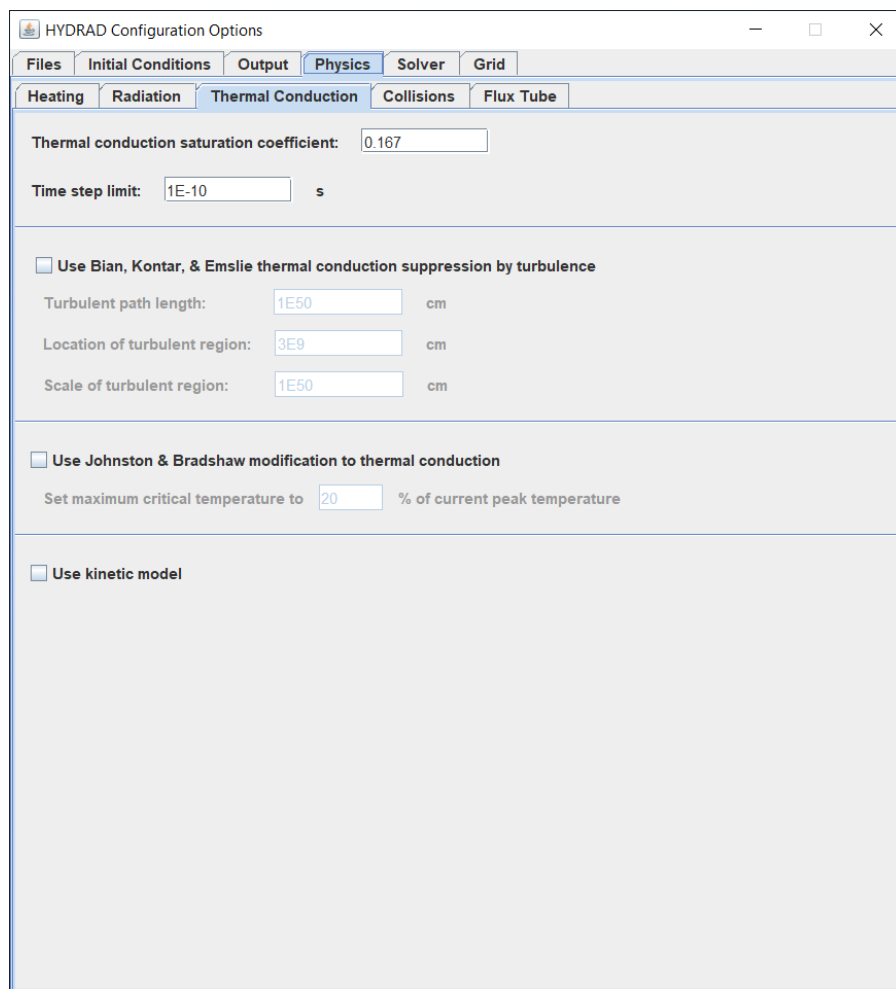


Figure 15: The options for configuring thermal conduction in HYDRAD.

(Maxwellian) distribution function to maintain its validity. To avoid straying into areas of inapplicability, the heat flux carried by each species is limited to its saturated value, reached when approximately all of the particles move in the same direction at the thermal speed. This prevents the heat flux from becoming arbitrarily large (e.g. due an increasing temperature gradient) and unphysical when there are too few particles of sufficient energy to support it. An additional multiplying factor (less than 1.0), which is indicated by Fokker-Planck calculations, is often applied to the saturated value. By default, this “Thermal conduction saturation coefficient” (Figure 15) is set to one sixth (0.167).

Thermal conduction in HYDRAD is calculated using the classical (Spitzer Härm) formulation, such that the heat flux is a strongly non-linear function of the local temperature and a function of the local temperature gradient. Underpinning this approximation is the assumption that conducting species are in near-collisional equilibrium, with only small perturbations allowed on the

The time scale for energy diffusion across a grid cell by thermal conduction ($\tau \sim n/T^{5/2}$) is often the shortest of all the active physical processes. If it becomes too short then continuing the calculation may become unfeasible. In this event, the “Time step limit” for thermal conduction can be increased from its default value of 10^{-10} seconds (Figure 15); the coefficient of thermal conduction is calculated such that it cannot yield time steps shorter than the specified limit. While this can render calculations tractable once again, the compromise is that thermal conduction is artificially limited in some regions of the domain.

5.3.2 Suppression of Thermal Conduction by Magnetic Turbulence

Selecting the “Use Bian, Kontar, & Emslie thermal conduction suppression by turbulence” checkbox (Figure 15) enables the method of Bian et al. (2016)¹⁹ for including the effects of particle scattering by turbulence associated with magnetic fluctuations. The user specifies the “Turbulence path length” (shorter path lengths correspond to stronger scattering), and they can choose to localize the turbulence region in the domain (by providing a location and (Gaussian) spatial scale) or apply it to the entire domain via a spatial scale that is much larger than the domain size.

Since turbulence is expected to be associated with the coronal heating mechanism (reconnection and / or waves) then setting the turbulent region to be spatially coincident with impulsive heating can emulate the effects of turbulence in the heating region.

5.3.3 The Johnson & Bradshaw Modification to Thermal Conduction

It is well-known that the demands placed on the transition region when it is required to handle a large incoming heat flux from the corona lead to very steep temperature gradients and correspondingly high spatial resolution (though hot and tenuous coronal regions can be the more limiting factor early on during heating and prior to significant filling by ablation). The resulting time steps can then be very short and yield unacceptable increases in execution time.

¹⁹ Bian, N. H., Kontar, E. P., & Emslie, A. G. “Suppression of Parallel Transport in Turbulent Magnetized Plasmas and Its Impact on the Non-thermal and Thermal Aspects of Solar Flares”, 2016, Astrophysical Journal, 824, 78

Various techniques exist to mitigate these issues including, for example, super-time stepping, but HYDRAD implements the method devised by Johnson & Bradshaw (2019)²⁰ which has been shown to be extraordinarily accurate in recovering the temperature – density relationship in phase space during the full heating / cooling cycle, while offering orders of magnitude decreases in runtime.

The user chooses a critical temperature (Figure 15), which is expressed as a percentage of the peak temperature at the current time step. Below this critical temperature, the divergence of the heat flux, radiation, and heat input are modified to spatially broaden the transition region and significantly reduce its spatial resolution requirements. Full details are given by Johnston et al. (2020)²¹. The inherent trade-off is that predictions of the emission spectrum below the critical temperature may not agree with those produced when this method is not used.

5.3.4 Non-classical Thermal Conduction

HYDRAD offers an experimental option to solve a kinetic equation for the electron distribution function in parallel with the hydrodynamic equations. The heat flux is calculated from the third moment of the distribution function in each cell of the domain and substituted into the electron energy equation. The purpose of this feature is to study the effects of a streaming / collisionless high-energy electron population, particularly with respect to heat flux delocalization. The distribution functions could also be integrated over collision cross-sections to estimate the effects on inter-species collisions and ionization / recombination rates.

The kinetic equation is based on a linear relaxation operator (e.g. a BGK-type equation) for the sake of speed. This yields the high-energy tail part of the distribution function. The low-energy part (up to about 3.4 thermal speeds) is calculated by applying the Spitzer Härm perturbation method to the \sim Maxwellian bulk of the distribution function.

²⁰ Johnston, C. D., & Bradshaw, S. J. “A Fast and Accurate Method to Capture the Solar Corona/Transition Region Enthalpy Exchange”, 2019, *Astrophysical Journal*, 873, L22

²¹ Johnston, C. D., Cargill, P. J., Hood, A. W., De Moortel, I., Bradshaw, S. J., & Vaseekar, A. C. “Modelling the solar transition region using an adaptive conduction method”, 2020, *Astronomy and Astrophysics*, 635, 168

The resulting heat fluxes are reasonable to, roughly, order of magnitude, given their sensitivity to their shape (skew) of the underlying distribution function. Hence this can be a useful qualitative, rather than quantitative, approach. A more sophisticated Fokker-Planck approach, based on a diffusion operator, may be implemented in the future.

When this option is selected a “.dfn” file is automatically written at each output period. This file contains the electron distribution function in each cell of the domain. The format of the “.dfn” file is:

Field-aligned position in first cell (cm)	...	Field-aligned position in last cell (cm)
Lower particle velocity limit (negative) (cm s ⁻¹)	...	Upper particle velocity limit (positive) (cm s ⁻¹)
Value of (log ₁₀) distribution function at lower velocity limit in the first grid cell	...	Value of (log ₁₀) distribution function at upper velocity limit in the first grid cell
Value of (log ₁₀) distribution function at lower velocity limit in the second grid cell	...	Value of (log ₁₀) distribution function at upper velocity limit in the second grid cell
.	.	.

This part of HYDRAD has been parallelized and the execution time can be substantially reduced when this feature is used with the “Use OpenMP” option enabled.

5.4 Inter-species Collisions

HYDRAD is a multi-species code. At present, hydrogen and electrons are treated with separate energy equations, and electrons and hydrogen can have different number densities (particularly in the lower atmosphere when the 6-level hydrogen atom is selected). These species are coupled via collisional terms in their respective energy equations. It is reasonably straightforward to couple energy equations for more species into this formulation.

A key challenge associated with collisionality in a domain with orders-of-magnitude variations in number density is the enormous range of time scales. Inter-species collisions and energy exchange are sufficiently frequent in the dense lower atmosphere to render integrating on the associated time scales unfeasible. Consequently, the user can select a “Min. collisional coupling time scale” in the “Physics/Collisions” GUI tab. This has the effect

of ensuring that species remain collisionally coupled where they should be (in the lower atmosphere), on the larger time scales that are typically of interest, while allowing decoupling in the hotter and less dense regions of the domain (upper atmosphere). The inherent trade-off is that this time scale generally sets the largest time step that HYDRAD can integrate on.

Most numerical models of this type employ a single-fluid approach. To facilitate comparisons between HYDRAD and other models, the option exists under the “Physics/Collisions” tab to force a single-fluid approximation in HYDRAD by coupling the species using an arbitrarily short collisional time scale everywhere (taken to be the minimum collisional coupling time scale already defined).

5.5 Flux Tube Properties

The flux tube properties pertain to the field-aligned gravitational acceleration, which determines the geometry of the flux tube, and the field-aligned variation in the cross-sectional area, determined by the inverse of the magnetic field strength. The paths and names of files containing piecewise polynomial fitting coefficients to spatial profiles of these field-aligned quantities can be specified in the “Physics/Flux Tube” GUI tab.

The piecewise polynomial fits and coefficient files can be obtained from tabulated data (e.g. produced by a magnetic field extrapolation), as described in Section 4.2.1.

6. Configuring and Optimizing the Numerical Solver

HYDRAD Configuration Options

Files Initial Conditions Output Physics **Solver** Grid

CFL condition safety coefficients (≤ 1.0)

Radiation:

Min. temperature of radiation: K

Decrease to zero over temperature interval: K

Max. density of optically-thin plasma: / cm³

Atomic processes:

Ion fraction cut-off:

Major ion control parameter:

Minor ion control parameter:

Thermal conduction:

Transport:

Viscosity:

☐ Include artificial viscosity

Damping time scale: (in multiples of the CFL transport time step)

Min. plasma species temperature: K

Limit time-step increases to % greater than the previous time-step

Figure 16: The options for configuring the numerical solver in HYDRAD.

The system of coupled hydrodynamic and atomic rate equations solved by HYDRAD are comprised of an assortment of hyperbolic, parabolic, stiff, and ordinary terms, with varying degrees of non-linearity (e.g. none to strongly). Solving such a variety of evolution equations presents a substantial numerical challenge and requires robust, but flexible, methods that can be tailored to particular kinds of problem.

In HYDRAD, the numerical methods can be configured with the options available under the “Solver” GUI tab (Figure 16). The default set of values have been chosen to work in the majority of cases, but occasionally some tweaking may be required to optimize accuracy and runtimes.

6.1 Radiation

In the “Radiation” section (Figure 15) the user can choose the additional safety factor (the default value is 0.1) by which to multiply the estimated radiation time scale, for numerical stability (e.g. preventing the electron energy dropping to negative values in regions of very strong radiation). During simulations where very high densities are encountered, leading to large radiative losses, which might be the case in flare models, this safety

factor may need to be further reduced by several orders of magnitude. Unfortunately, it cannot be rigorously determined beforehand because the characteristic speed associated with radiation is the speed of light (photon speed) and thus would be overly restrictive with respect to time-stepping if uniformly applied. The other options available in this section allow the user to choose the minimum temperature at which an optically-thin plasma radiates (given by the temperature of the chromosphere in the isothermal treatment), the temperature interval over which radiation is smoothly decreased to zero (to avoid precipitous decreases and possible instability), and the maximum density of the optically-thin plasma used in the radiative energy loss calculation (n^2), again to limit the losses and avoid non-physical consequences. The maximum density must be larger than the density at the base of the transition region and a reasonable choice would be the density somewhere in the mid-chromosphere.

6.2 Atomic Processes

The parameters associated with “Atomic processes” configure the solver for non-equilibrium ionization. While the rate equations themselves are numerically stiff, the method employed is robust and the safety factor can be left at 1.0. The “Ion fraction cut-off” sets a floor on the minimum allowable ion population fraction, to alleviate the numerical stiffness, and the ion control parameters set the accuracy of minor vs. major ion populations in the solutions. These latter parameters are described by MacNeice et al. (1984)²² and the defaults should suffice for essentially all conceivable situations.

6.3. Thermal Conduction and Bulk Transport

The additional safety factors applied to “Thermal conduction” and “(Bulk) Transport” are set to 1.0 by default. The numerical methods used to evaluate these terms are very robust and their safety factors can typically be left at 1.0.

6.4. Kinematic and Artificial Viscosity

The safety factor associated with “(Kinematic) Viscosity” can also be left at 1.0. HYDRAD also provides an option to “Include artificial viscosity” to help

²² MacNeice, P., McWhirter, R. W. P., Spicer, D. S., & Burgess, A. “A Numerical Model of a Solar Flare Based on Electron Beam Heating of the Chromosphere”, 1984, Solar Physics, 90, 357

reduce / eliminate any small scale wave modes that arise close to the spatial resolution limit. It is not usually needed, but can be used to provide smoother profiles with negligible quantitative changes in the solutions. The “Damping time scale” is set to multiples of the time scale associated with bulk transport; consequently, small scale wave modes are typically damped out over periods coupled to this time scale (e.g. the default is roughly 10 sound wave crossing times in the smallest cell when the bulk flow is much smaller than the sound speed).

6.5 Other Physics and Stability-related Parameters

The “Min. plasma species temperature” is set by the user to prevent unphysically low (and negative) temperatures being encountered when strong cooling processes are operating. Increases in consecutive time steps can be pegged to the previous time step by a user-defined amount to prevent large changes and associated issues (instabilities, etc.). There is no limit on the amount a time step is allowed to decrease between consecutive steps, which is essential to maintain numerical stability.

7. Configuring and Managing the Adaptive Grid

The screenshot shows the 'HYDRAD Configuration Options' dialog box with the 'Grid' tab selected. The 'Uniform grid' checkbox is unchecked. The 'Min. number of grid cells' is set to 60. The 'Max. refinement level' is 10, and the 'Initial refinement level' is also 10. The 'Min. grid cell width' is 97656.25 cm, and the 'Max. grid cell width' is 1.0E8 cm. Under 'Refine on:', 'Density' and 'Electron energy' are checked, while 'Hydrogen energy' is unchecked. 'Refine every' is set to 10 time steps. 'Aim for a difference of >= 10 % and' is selected, and 'aim for a difference of <= 20 % between quantities in adjacent cells' is also present. 'Use linear restriction' is unchecked, and 'Enforce conservation upon restriction' is checked.

Figure 17: The options for configuring the adaptive grid in HYDRAD.

One of the most powerful features of HYDRAD, which enables it to perform calculations that are out-of-reach of other numerical codes, is its flexible adaptive grid. The adaptive regridding algorithm in HYDRAD can place and remove spatial resolution, to maximize efficient of the calculation, anywhere in the domain. It is not limited to one or two sub-domains, as is the case for other “adaptive” methods. It is also a

genuinely adaptive scheme, which physically adds and removes grid cells, unlike a stretching algorithm which adds increases resolution at the expense of resolution elsewhere and keeps the total number of grid cells constant. Familiarity with configuring the adaptive grid to suit a particular calculation is key to its successful completion. The user is encouraged to experiment to build an intuition for configuring the adaptive grid and how the code responds to changing the parameter values.

7.1 Basic Grid Properties

Selecting the “Uniform grid” checkbox in the “Grid” GUI tab (Figure 17) switches off all adaptive capability. The uniform size of the grid cells in the domain is then given by the size of the domain divided by the “Min. number

of grid cells” (which simply becomes “Number of grid cells”). For example, a flux tube length of 60 Mm and 60 grid cells yields a uniform cell size of 1 Mm (1000 km). This option is useful for fast exploratory calculations.

The adaptive grid is enabled by default. The user specifies the “Min. number of grid cells”, which yields the “Max. grid cell width”. It is **STRONGLY** recommended to choose this number of grid cells such that the maximum cell width is a whole number that can be represented with low precision (e.g. few decimal places in normalized scientific (mantissa / exponent) form).

The user then chooses the “Max. refinement level” and the “Initial refinement level”. Each time a grid cell is refined (restricted) it is split into two new (daughter) grid cells that are half the size of the original (parent) cell. The maximum refinement level (RL_{\max}) determines the “Min. grid cell width” in the domain by dividing the maximum cell size by $2^{RL_{\max}}$. The initial refinement level specifies the amount of refinement that should be applied to the “Initial_Conditions/profiles/initial.amr” file (Section 4.2.1). The initial conditions don’t typically require the degree of refinement that may be encountered later on during a run, to be properly resolved, and choosing an initial refinement level that is smaller than the maximum allowable produces smaller initial conditions (“initial.amr”) files. This is particularly useful in the case of very large domains (e.g. solar wind models that extend out to many solar radii).

7.2 Adaptive Refinement Parameters

The user can specify the conditions which must be satisfied for restriction (splitting cells) and prolongation (merging cells). By default, HYDRAD restricts / prolongs depending on the cell-to-cell changes in the particle number density and the electron energy, and aims to maintain these changes between 10% and 20% in adjacent cells. Changes less than 10% allow cells to be merged, while changes greater than 20% require cell splitting. Choosing smaller values increases the total number of grid cells (and cell sizes) required to satisfy these constraints, increasing accuracy but extending execution times commensurately.

By default, HYDRAD applies the adaptive regridding algorithm every 10 time steps, since it demands fairly significant computational overhead, although the overall performance gains are well worth this penalty. If the solution changes slowly in time / space, and / or a few time steps of under-resolution

can be tolerated, then this number can be relatively large. 10 is a reasonable number for most use cases, but the user is free to experiment and tailor it to their own preferences for solution accuracy and runtime.

7.3 Additional Restriction Parameters

Using the default third order polynomial interpolation to calculate the conserved quantities in newly refined cells can introduce new maxima and minima, particularly in cases where structures arise in the solution at small spatial scales and vary rapidly. These can grow in amplitude as local wave-like solutions, requiring further refinement and smaller time steps, potentially rendering the calculation intractable. While the interpolation algorithm attempts to avoid this scenario by employing efforts at mitigation, sometimes it is not sufficient. Under such circumstances, the user can choose “Linear restriction” whereby a simple linear interpolation (which cannot introduce new maxima and minima) to calculate the conserved quantities in newly refined cells. This generally resolves the problems associated with non-linear interpolation and allows calculations to proceed.

Since HYDRAD solves the conservation equations, it is important to ensure that mass, momentum, and energy is not lost from the domain when new cells are created upon splitting. To this end, the conserved quantities interpolated into the newly created cells are corrected to guarantee that the sum of the conserved quantities in the new cells equals the amount in the original cell. Occasionally, this condition can become somewhat restrictive and unchecking the “Enforce conservation upon restriction” box can help calculations along.

Introducing linear interpolation and relaxing the conservation constraint is essentially equivalent to introducing a small amount of numerical diffusion, which acts to smooth out potentially problematic parts of the solution and allows calculations to be completed in a reasonable amount of time.

The prolongation operator simply sums conserved quantities in the two cells to be merged and takes the average to populate the merged cell. Thus, interpolation is trivial and conservation is guaranteed.