Manual for using the CO^{5}BOLD-code for numerical wave propagation experiments.

Christian Nutto[*]
Kiepenheuer-Institut für Sonnenphysik
Freiburg, Germany

01.04.2010

1 Introduction

This manual will give a brief introduction about how the CO^{5}BOLD-code can be used for numerical wave propagation experiments. The package will not provide the CO^{5}BOLD-code itself as well as CO^{5}BOLD-routines that can handle its specific data output format. These have to be downloaded from the CO^{5}BOLD project website[1]. The changes to the codes for the excitation of artificial waves in the simulation are given in a separate file and need to be copied into the CO^{5}BOLD-modules. The implementation of these code snippets is given in Sect. 2.
Furthermore, IDL-routines[2] are provided that convert the simulation data output into different data formats. The CO^{5}BOLD-code uses a specific binary data format called "uio". In order to read out this "uio"-output, special IDL-routines are supplied by the CO^{5}BOLD software package. The IDL-procedures in this package make use of these routines.
In the VAPOR-directory routines are provided to demonstrate how the CO^{5}BOLD output from 3D simulations can be transformed into the VAPOR data format. In the directory helioseismology, there are IDL-routines that read in the simulation output for further manipulation in order to calculate helioseismic quantities. These derived quantities are stored in a data format that can be read by Gnuplot for plotting. Therefore, the directory "gnuplot" contains example files that can be read by gnuplot and demonstrates how the different quantities can be plotted.

[*]nutto@kis.uni-freiburg.de
[1]see www.astro.uu.se/~bf/co5bold_main.html
[2]The IDL-routines are written for IDL 7.0 and IDL 7.1

The user of the package should feel free to use it to its extent and modify the code if applicable. It should be mentioned that it was specifically created to satisfy the needs of the author. The code was simplified to allow for the use of others, but one still might have to modify the code for own purposes.

2 Using the CO^{5}BOLD-code for wave excitation experiments

In the directory "CO^{5}BOLD_waves" is a txt-file with the fortran code that needs to be implemented into the code in order to use it for 2D/3D wave propagation experiments. For the excitation of the waves we follow the way by Parchevsky & Kosovichev (2009), where the waves are excited by source terms in the momentum equation. The CO^{5}BOLD-code treats these source terms of the momentum equation in a specific subroutine of the MHD-routine **rhd_mhd_module.f90** which can be found in the directory: "for/hd/mhd/" of the CO^{5}BOLD-package. Thus, the code snippets in **wave_excitation_CO^{5}BOLD.txt** have to be placed at the position in the code where this subroutines determines the source terms. The routine in which this occurs is the subroutine *rhd_mhd_Roe1Dstep*.
<u>Block 1</u> in **wave_excitation_CO^{5}BOLD.txt** has to copied to the position where the variables are defined in *rhd_mhd_Roe1Dstep*. The next step is to simple replace the code where the original values are stored in local copies with <u>Block 2</u>. Depending on the version (for24.02.09 in this case) <u>Block 3</u> has to be placed at Line 7217. If a different version is used, than the code has to be placed between the position where the source function *source_rhov1* is calculated and where the output values are updated accordingly to the calculated source function:

```
⋮
source_rhov1=rho(i1,i2,i3) * (phib_grav(i1+1,i2,i3)-phib_grav(i1,i2,i3))
                             /(xb1(i1+1)-xb1(i1))
```

wave excitation code

```
! — Update —
rhoout=rho(i1,i2,i3) - dtimeoverdx(i1) * (flux_rho(i1+1,i2,i3)-flux_rho(i1,i2,i3))
⋮
```

2.1 Different kind of artificial waves

Here the different kind of waves that can be excited are described. What kind of waves will be excited is determined by the control parameter.

Control parameters:

control=0 no wave will be excited

control=1 excitation of a vertical plane-parallel wave at a specific height in the atmosphere. At which height this vertical plane parallel wave is excited is determined by the index *z0_i*. The excitation takes place at every cell where *i1==z0_i*. The strength of the wave is determined by *coeff_1*, as will be for all other cases.

control=2 excitation of a vertical plane-parallel wave with a Gaussian shaped excitation. At which height this vertical plane parallel wave is excited is determined by the index *z0_i*. However, this time cells above and below *z0_i* are effected depending on the *fwhm*-parameter.

control=3 same as in the case control=2, just scale the amplitude with ρ_0 to account for local density disparity in the excitation zone.

control=4 this excites a tilted plane-parallel wave. The central position of the excitation is given by the indices *x0, z0*. The excitation is again Gaussian in both directions where the width of the excitation is given by *fwhm_x, fwhm_z*. Care has to be taken, that the excitation is tilted by an *angle* and that the coordinates have to be transformed to a new grid x' and z' rotated by a specific degree *angle*.

control = 5 here, we follow the wave excitation described by Parchevsky & Kosovichev (2009) applying it to vertical velocity perturbations. The location of the excitation is given by *x_ex, z_ex*. The radius of the wave source is *r_src*.

control = 6 same as for control=5, however for perturbations of the horizontal and vertical velocities.

control = 7 excitation of a wave in 3D-simulations. The mechanisms are the same as for control=6.

Using these wave excitation cases, an artificial wave is launched that can be used for numerical wave propagation experiments with the CO^{5}BOLD-code.

3 IDL-routines for doing local helioseismology of the atmosphere

This section will describe the different IDL-routines of the package that can be used to do local helioseismology of the wave propagation experiments. The routines are written in a way to serve the needs of the author. The user of these routines can and should modify them for his/her own purpose. All the routines need paths to the necessary data, that is provided by the CO^{5}BOLD-code, e.g. equation-of-state of opacity tables. These paths have to be specifically set by the user according to the file system that is used.

3.1 rhd_create_miscB.pro

This routine lets the user create its own atmospheric setup that can then be used for numerical wave propagation experiments. The example file, **example_atmosphere.sta**, consists of a plane-parallel stratified atmosphere using Model S by Christensen-Dalsgaard et al. (1996). The routine then lets you superimpose a homogenous magnetic field. Including a small perturbation in e.g. velocity, then the model can be evolved in time. Because of the small perturbation, convection should set in after a short while of the simulation run and granules will start to form.
Or, one already uses a snapshot of a simulated hydrodynamic atmosphere and then superimposes a magnetic field. Again, after a short while of the magneto-hydrodynamic simulation a magnetically structured atmosphere will be evolved.
For input, paths and filenames of different data sets described in the routine have to be specified. Especially the input file that should be modified has to be declared. Furthermore the field strength B of the magnetic field in Gauss and its orientation *deg* in degree need to be given.
The routine then writes a new output file in the CO^{5}BOLD data format that can then be used as a start file for wave propagation experiments.

3.2 average_sub.pro

This is the basic routine to calculate wave propagation quantities. Its basic purpose is to calculate the difference of the two simulation runs where in one of them is a artificial wave excited and in the other one there is no specific wave excitation taking place. The residual then shows the actual wave propagating through the magnetically structured atmosphere. The reason for this subtraction method is explained in Steiner et al. (2007).
As input, the routines basically needs the paths of different data already used for

the simulation run (e.g. equation-of-state, opacity, parameter file) as well as the paths and lists of filennames of the simulation runs with and without the wave. The rest of the required paths deals with the output of the data, e.g. the output directory and filenames. The data is stored in both, IDL-save files and in gnuplot data format. All derived quantities are stored in files using the gnuplot data format. Only specific quantities are stored in the IDL-save files. The user itself should decide which of the quantities should be stored in the IDL-save file and has to modify the routine accordingly. There are two IDL-save files at the moment that store different data: one stores the values without the wave (*out_orgdata*) and the other file stores the quantities related to the wave propagation (*out_wavedata*). Furthermore, an output file in the CO^{5}BOLD data format is written, containing the subtracted values of the two simulation runs that can then be viewed by the CO^{5}BOLD analyzing tool (CAT).

Derived quantities:

dP ratio of pressure perturbation to local unperturbed pressure, $\delta p/p_0$

pressure local unperturbed pressure, p_0

dvpar velocity perturbation parallel to the magnetic field, $\delta|v_{||}|$

dvper velocity perturbation perpendicular to the magnetic field, $\delta|v_{\perp}|$

dv absolute velocity perturbation, $\delta|v|$

dvrho12 absolute velocity perturbation scaled with the square root of the density, $\rho^{1/2}\delta|v|$

dBpar magnetic field perturbation parallel to the magnetic field, $\delta|B_{||}|$

dBper magnetic field perturbation perpendicular to the magnetic field, $\delta|B_{\perp}|$

dB absolute magnetic field perturbation, $\delta|B|$

B unperturbed magnetic field

beta plasma-β

csca ratio of the sound speed c_s to the Alfvén speed c_a, c_s/c_a

F_ac acoustic wave flux (Vigeesh et al. 2009; Bogdan et al. 2003)

F_mag poynting wave flux (Vigeesh et al. 2009; Bogdan et al. 2003)

F_tot absolute total wave flux

These are the quantities that are calculated by this subroutine and saved in gnuplot data format. For plotting these quantities see Sect. 5. They can also be saved in an IDL-save file (*out_wavedata*) if desired, but then the routine has to be modified properly.

3.3 get_power.pro

So far, this routine determines the temporal power spectra of the numerical wave propagation experiments for different horizontal cuts through the atmosphere.
As input, the routine reads the IDL-save file (*out_wavedata*) from **average_sub.pro** containing the components of the perturbation of the velocity and magnetic field stored in the IDL-structure *wave_data*. Further necessary parameters are:

Input parameters:

n_heights number of heights in the atmosphere for which the power spectra should be determined

step_size the distance between two subsequent heights

avg_cut number of time steps that should be neglected at the end of the time series because of the averaging technique in **average_sub.pro** (optional). Can be set to 0 if no averaging is used in **average_sub.pro**.

n_zeros number of zeros that can be added in front of short time series in order to enhance the frequency resolution artificially.

t_sampling temporal resolution of time series

The derived quantities are the following:

Derived quantities:

pow_dvz_zi power spectra of δv_z at each vertical cut and horizontal position

pow_dvx_zi power spectra of δv_x at each vertical cut and horizontal position

pow_dvabs power spectra of $\delta |v|$ at each vertical cut and horizontal position

pow_dbz_zi power spectra of δB_z at each vertical cut and horizontal position

pow_dbx_zi power spectra of δB_x at each vertical cut and horizontal position

pow_dbabs power spectra of $\delta |B|$ at each vertical cut and horizontal position

Again the results are stored in gnuplot data format for plotting and in an IDL-save format.

3.4 wavetime.pro

This routine calculates the vertical wave travel time between two heights in the solar model atmosphere. This can be done for the velocity perturbations and magnetic perturbations. Here, we follow the method used by Finsterle et al. (2004). It is assumed that the simulation box is placed at the center of the solar disc and thus only vertical velocity perturbations δv_z are considered. This time no output is written to a file, since the fitting routines are very sensitive to the guessed initial values for the fitting parameters. Thus, the fit needs to be checked before it can be written to a output file which has then to be done manually.

Again, the necessary data is read from the IDL-save file (*out_wavedata*) from **average_sub.pro** containing the necessary data. The input parameters are the following whereas some of them are basically the same as in **get_power.pro**:

Input parameters:

n_heights number of heights in the atmosphere between which the wave travel time should be calculated

step_size the distance between two subsequent heights

avg_cut number of time steps that should be neglected at the end of the time series because of the averaging technique in **average_sub.pro** (optional). Can be set to 0 if no averaging is used in **average_sub.pro**.

n_zeros number of zeros that can be added in front of short time series in order to enhance the frequency resolution artificially.

t_sampling temporal resolution of time series

f_0 central frequency for the gaussian frequency filter (Finsterle et al. 2004)

width width of the gaussian frequency filter

index_z1 index of the horizontal cut for the lower height between which the wave travel time should be calculated

index_z2 index for the upper height for the wave travel time measurement

The fit to the cross-correlation function is done using the IDL-procedure **curvefit** and the Gabor-wavelet (Finsterle et al. 2004) given in **gfunct5.pro**. The curvefit using the Gabor-wavelet needs the following fitting parameters as an initial guess.

Input parameters for the fit of the cross-correlation function:

A amplitude of the Gabor-wavelet

om0 central frequency of the gaussian frequency filter

tph phase travel time of the wave

sigma basically 1./(width of the frequency filter)

tgroup group travel time of the wave package

These parameters are then put in the input array *aest0* for the curvefit routine, while the fitting parameters after the fit are then given by *aest*. This fit of the vertical wave travel time is done for each horizontal point of the simulation domain and the fitting parameters are stored in arrays (*w*,*tph*,*dw*,*tgr*) for plotting or further data manipulation.

3.5 get_cf_tseries

This routine calculates the acoustic cut-off frequency for a time series. As input data, the routines needs an IDL-save file that was created by the routine **cobold2idl_tseries.pro**. Further input parameters are:

Input parameters:

order order of the difference scheme that is applied, which uses an evenly spaced grid

z_low lowest cell of the new evenly spaced grid

dz vertical resolution of the new grid

nr_cells number of cells in the vertical direction of the new grid

t_bin number of time bins that should be applied

The output consists of the following quantities

Derived quantities:

cf_t_xz the cut-off frequency for each cell and time step of the time series

cf_t horizontally averaged cut-off frequency for each time step

cf_t_cad horizontally averaged cut-off frequency for binned time steps depending on the cadence of the observations to which it should be compared

cf_tavg horizontally and time averaged cut-off frequency for the complete time series

z_prime2 new vertical grid because of the difference scheme vertical grid points are lost

3.6 cobold2idl_tseries.pro

This routine reads in the CO^{5}BOLD output of a time series which can then be used in IDL-routines. The output is stored in an IDL-save file that then can be used by other IDL-routines. The routine basically just needs the path to the directory of the CO^{5}BOLD output and a list of the filenames if the time series consists of more than one file. The output is saved in an IDL-save file containing the same data as present in the CO^{5}BOLD files.

3.7 idl2gnu

This is just a really short routine to write 2D quantities into the gnuplot data format for 2D data sets.

Input quantities:

data the array containing the 2D-data

x the grid along the first dimension

y the grid along the second dimension

filename output filename

4 Converting CO^{5}BOLD output into VAPOR Data Format

These routines concern 3D simulations carried out with the CO^{5}BOLD-code and can be found in the directory "vapor" provided by the software package. In order to visualize the simulations we use the VAPOR Visualization and Analysis Platform[3]. Therefore, the output by CO^{5}BOLD has to be transformed into the VAPOR data format (vdf). Writing the data into the vdf-format is done by IDL-routines, that are provided by the VAPOR package. Make sure, that IDL knows the paths to these routines.
The following sections will explain, how the CO^{5}BOLD data is converted into VAPOR data.

4.1 cobold2vapor_single.pro

This routine converts a single snapshot of a CO^{5}BOLD simulation into the VAPOR data format. The routine needs the paths and the filenames of the data of the equation of state, opacity, the parameter file, and of course the snapshot that is needed to be converted. These filenames are passed on to the routine **read_cobold.pro**. Furthermore, the output directory and filename have to be given. The following parameters need to be defined.

Input parameters:

dim_x number of cells along the x-direction

dim_y number of cells along the y-direction

stepsize_z VAPOR only accepts regular spaced grids. Since we use an adaptive grid in the vertical direction, a new grid with an evenly spaced distance given by this parameter is defined

nr_data_sets the number of data sets that will be converted into the VAPOR data format

varnames names of the data sets as they will appear in VAPOR. The number of strings has to correspond to the number of data sets that are supposed to be written.

[3]http://www.vapor.ucar.edu/

4.2 cobold2vapor_tseries.pro

This routine converts a time series of a CO^{5}BOLD simulation into the VAPOR data format. It basically works the same as in **cobold2vapor_single.pro**. However, the time step management is more complicated. Again, the paths and filenames to the relevant data has to be specified.

Input parameters:

nr_files number of files the time series consists of

timesteps array that stores the number of time steps that are present in each input file

sum_timesteps the sum of the time steps in all files

cont_step array containing the values of the continuing time step for each file

dim_x number of cells along the x-direction

dim_y number of cells along the y-direction

stepsize_z VAPOR only accepts regular spaced grids. Since we use an adaptive grid in the vertical direction, a new grid with an evenly spaced distance given by this parameter is defined

nr_data_sets the number of data sets that will be converted into the VAPOR data format

varnames names of the data sets as they will appear in VAPOR. The number of strings has to correspond to the number of data sets that are supposed to be written.

4.3 read_cobold.pro

This procedure reads out the CO^{5}BOLD output of a snapshot passed on by **cobold2vapor_single** or of a time series passed on by **cobold2vapor_tseries**. The data is then transformed onto a regular vertical grid by **transform_cobold-data.pro**.

Input parameters:

modelfile consisting of the filename (without extension) of the CO^{5}BOLD data

mextens extension of the filename (.full, .end, .sta)

nrtimestep the time step number that needs to be read out

4.4 transform_cobolddata.pro

This routine interpolates the CO^{5}BOLD output data, which is usually on a vertical adaptive grid, onto a vertical regular spaced grid. This routine is called by **read_cobold.pro**.

Input parameters:

data the data that needs to be interpolated onto the regular grid

n3_new new number of grid cells in the vertical direction

quantity specifying what kind of quantity is interpolated (hydrodynamic quantity - cell centered values; magnetic quantity - cell boundary values)

b_comp keyword specifying which magnetic field component is transformed; if no magnetic component is processed leave the string empty.

4.5 get_thermoquantities.pro

Using this routine, one is able to derive thermodynamic quantities, e.g. pressure or temperature, from the CO^{5}BOLD output data. However, the routine and routines within **cat_eosbox.pro** that derive these quantities are provided by the CO^{5}BOLD-code and not by this software package. The procedure returns an IDL-structure containing all the quantities that are derived. If other thermodynamic quantities than the one already determined by this procedure are necessary, then the user should feel free to modify the script for own purposes.

Input parameters:

parfile name of the parameter file, including path

opafile name of the opacity table, including path

eosfile name of the equation-of-state table, including path

modelfile name of the CO^{5}BOLD output file, including path, excluding extension

mextens extension of the CO^{5}BOLD output file that is used (.full, .end, .sta)

nrtimestep time step of the model file that should be processed

As mentioned, the output is returned to **cobold2vapor_single.pro** or **cobold2vapor_tseries.pro** via the *cobold_data.common* block for further manipulation or conversion into the VAPOR data format.

5 Using Gnuplot: Quick example

This section describes two quick examples, how gnuplot can be used to plot the data that is written in Gnuplot data format by several routines.

5.1 plot_dv.gnu

This script is a short example of how the output by **average_sub.pro** can be plotted. This script plots the velocity perturbation scaled with the density, $\rho^{1/2}\delta|v|$, but can be modified easily to for all other output quantities.
The script plots a map of the 2D-simulation for the specified quantity. Overplotted, using multiplot, is the equipartition layer where the sound speed equals the Alfvén speed, $c_s = c_a$. The script uses a loop to plot the different time steps defined by the variable a. Thus before calling the script this variable needs to be defined in Gnuplot.
The script is pretty much self explanatory and can easily be modified.

5.2 plot_pow_dB.gnu

This script can be used to plot the output from **get_power.pro**. At the moment the script plots the power as seen in magnetic field perturbations $\delta|B|$ but can again easily be modified to plot all other variables from **get_power.pro**. The height index is given by a and has to be set before the script is called. The loop runs over all heights starting from the height with the index a.

References

Bogdan, T. J., Carlsson, M., Hansteen, V. H., et al. 2003, Astrophysical Journal, 599, 626

Christensen-Dalsgaard, J., Dappen, W., Ajukov, S. V., et al. 1996, Science, 272, 1286

Finsterle, W., Jefferies, S. M., Cacciani, A., Rapex, P., & McIntosh, S. W. 2004, Astrophysical Journal, Letters, 613, L185

Parchevsky, K. V. & Kosovichev, A. G. 2009, Astrophysical Journal, 694, 573

Steiner, O., Vigeesh, G., Krieger, L., et al. 2007, Astronomische Nachrichten, 328, 323

Vigeesh, G., Hasan, S. S., & Steiner, O. 2009, Astronomy and Astrophysics, 508, 951