Code `OllinAxis-BiB` User's manual

Miguel Alcubierre
Instituto de Ciencias Nucleares, UNAM
malcubi@nucleares.unam.mx

June, 2022

Contents

1 Introduction

This program solves the Einstein evolution equations in axial symmetry using a curvilinear version of the BSSN formulation for the 3+1 evolution equations, with different types of matter and different gauge conditions.
The main difference of this version of the code with respect to previous ones is the fact that it uses box-in-box mesh refinement (hence the BiB part of the name), and is parallelized with MPI. Many parts of this code are based o a previous version written by myself and José Manuel Torres. An even older version was written by Milton Ruiz.
Note: This manual is TERRIBLY INCOMPLETE! I haven't has the time to do it. I will be adding a little bit more every now and again.

2 Downloading the code

If you are reading this it means you probably already downloaded the code. But if for some reason you need to download it here is how.

The easiest way to obtain the code is to download it from the CVS server of the Ollin group (dulcinea.nucleares.unam.mx). In order to obtain a username and password for this sever you need to contact Miguel Alcubierre (malcubi@nucleares.unam.mx). Once you have a username and password you first need to log into the repository by typing at the terminal:

```
cvs -d :pserver:username@dulcinea.nucleares.unam.mx:/usr/local/ollincvs login
```

where you should change the word "username" for your personal username! You then need to type your password. After this you must download the code by typing:

```
cvs -d :pserver:username@dulcinea.nucleares.unam.mx:/usr/local/ollincvs co Codes/OllinAxis-BiB
```

This will create a directory `Codes` and inside it directory `OllinAxis-BiB`. This is the main directory for the code.

3 Directory structure

The main directory for the code is `OllinAxis-BiB`. There are several sub-directories inside this main directory:

`CVS`	Contains information about the CVS root and server (see Sec. 13).
`doc`	Contains the tex and pdf files for this user's manual.
`exe`	This directory is created at compile time and contains the executable file. It also contains a copy of the parameter files found in directory `par` (see below).
`fakempi`	Contains fake MPI routines so that the compiler won't complain if MPI is not installed.
`gnuplot`	Contains a couple of simple gnuplot macros for visualization.
`objs`	This directory is created at compile time and contains all the object and module files.
`ollingraph`	Contains the visualization package "ollingraph" for convenient "quick and dirty" visualization (see Section 14 below).
`par`	Contains examples of parameter files (see Section 4 below).
`prl`	Contains perl scripts used at compile time to create the subroutines that manage parameters and arrays.
`src`	Contains the source files for all the code routines.

The directory `src` is itself divided into a series of sub-directories in order to better classify the different routines. These sub-directories are:

`CVS`	Contains information about the CVS root and server.
`auto`	Contains `FORTRAN` files that are automatically generated at compile time by the perl scripts. These files should not be edited manually!
`base`	Contains the routines that control the basic execution of the code, including the parameter and array declarations, the parameter parser, the output routines, the main evolution controllers, and generic routines for calculating derivatives, dissipation, etc. The code in fact starts execution at the routine `main.f90` contained in this directory.

`elliptic`	Contains routines for solving elliptic equations for initial data and/or maximal slicing for example.
`geometry`	Contains routines related to initial data, evolution and analysis of the spacetime geometric variables, including sources, gauge conditions, constraints, horizon finders, etc.
`matter`	Contains routines related to the initial data, evolution and analysis of the different matter models, including a generic routine for calculating the basic matter variables, and routines for evolving scalar fields, electric fields, fluids, etc.

The code is written in `FORTRAN 90` and is parallelized with `MPI` (Message Passing Interface). All subroutines are in separate files inside the directory `src` and its sub-directories.

3.1 Compiling

To compile just move inside the `OllinSphere-BiB` directory and type:

```
make
```

This will first run some perl scripts that create a series of automatically generated `FORTRAN` files that will be placed inside the directory `src/auto`. It will then compile all the `FORTRAN` routines that it can find inside any of the sub-directories of `src` (it will attempt to compile any file with the extension `.f90`).

The resulting object files and `FORTRAN` module files will be placed inside the sub-directory `objs`. The Makefile will then create a directory `exe` and will place in it the final executable file called `ollinaxis`. It will also copy to this directory all the sample parameter files inside the sub-directory `par`, and the visualization package `ollingraph`.

Notice that at this time the Makefile can use the compilers `g95`, `gfortran`, or the Intel compilers `ifc` and `ifort`, and it will automatically check if they are installed. If you have a different compiler then the Makefile will have to be modified (hopefully it won't be very difficult). The code will also attempt to find an `MPI` installation (it looks for the command `mpif90`), and if it does not find it it will use the fake routines inside the directory `fakempi`.

The Makefile has several other useful targets that can be listed by typing: `make help`.

3.2 Running

To run the code move into the directory `exe` and type:

```
ollinsphere name.par
```

Where `name.par` is the name of your parameter file (more on parameter files below). The code will then read data from the parameter file silently and hopefully start producing some output to the screen. The code will also create an output directory and will write the data files to that directory.

For parallel runs using `MPI` one must use instead the command:

```
mpirun -np N ollinsphere name.par
```

where `N` should be an integer number that specifies the number of processors to be used.

4 Parameter files

At run time the code reads the parameter values from a parameter file (parfile), with a name of the form `name.par`, that must be specified in the command line after the executable:

```
ollinaxis name.par
```

The data in this parameter file can be given in any order, using the format:

```
parameter = value
```

Comments (anything after a `#`) and blank lines are ignored. Only one parameter is allowed per line, and only one value is allowed per parameter, with the exception of the parameters `outvars0D` and `outvars1D` that control which arrays get output and take lists of arrays as values, for example:

There is in fact one other parameter that can also take multiple values as input, it is the parameter `mattertype` that can accept several types of matter at once (see Section 8 below).

Parameters that do not appear in the parfile get the default values given in the file `param.f90`. Examples of parameter files can be found in the subdirectory `par`.

IMPORTANT: Even though `FORTRAN` does not distinguish between upper and lower case in variable names, the names of parameters are handled as strings by the parameter parser, so lower and upper case are in fact different. The name of parameters in the parameter file should then be identical to the way in which they appear in the file `param.f90`.

5 Output files

At run time, the codes creates an output directory whose name should be given in the parameter file. It then produces a series of output files with the data from the run. There are so called `0D` files (with extension `*.tl`), `1D` files (with extensions `*.rl`, `*.zl` and `*.dl`), and `1D` files (with extension `*.2D`).

The `0D` files refer to scalar quantities obtained from the spatial arrays as functions of time. These scalar quantities include the maximum (`max`), the minimum (`min`), and three different norms of the spatial arrays: maximum absolute value (`nm1`), root mean square (`nm2`), and total variation (`var`). The value of different variables at the origin is also output.

The `1D` files contain the complete arrays along the coordinate axis and diagonals at different times, while the `2D` files output the full arrays. Beware: If you do output very often these files can become quite large!

Since we can have several grid refinement levels, and grid boxes, the file names are appended with a number corresponding to the specific box and level (all grid levels have output). For example:

`alphab0l0.rl`:	Level 0 (coarsest grid)
`alphab0l1.rl`:	Level 1

...

All files are written in ASCII, and using a format adapted to XGRAPH (but other graphic packages should be able to read them).

Output is controlled by the following parameters:

`directory`	Name of directory for output.
`Ninfo`	How often do we output information to screen?
`Noutput0D`	How often do we do 0D output?
`Noutput1D`	How often do we do 1D output?
`Noutput2D`	How often do we do 2D output?
`outvars0D`	Arrays that need 0D output (a list separated by commas).
`outvars1D`	Arrays that need 1D output (a list separated by commas).
`outvars2D`	Arrays that need 2D output (a list separated by commas).

6 Numerical grid

7 Spacetime and evolution equations

The code uses cylindrical coordinates (r, z) is space. Notice that what the code calls r is frequently called ρ. The distance to the origin in the code is in fact called $rr := (r^2 + z^2)^{1/2}$.

The spatial metric is written in the following way:

$$\begin{aligned} dl^2 = e^{4\phi(r,z,t)} \big[& A(r,z,t)dr^2 + B(r,z,t)dz^2 + r^2 H(r,z,t)d\varphi^2 \\ & + 2\, r \left(C(r,z,t)drdz + r^2 C_1(r,z,t)drd\varphi + rC_2(r,z,t)dzd\varphi \right) \big] \ , \end{aligned} \tag{7.1}$$

with ϕ the conformal factor.We also define the function $\psi = e^{\phi}$ and use it instead of ϕ in many expressions. For the lapse function we use the array $\alpha(r, z, t)$, while the shift in principle has three components: $\beta^r(r, z, t)$, $\beta^z(r, z, t)$, $\beta^\varphi(r, z, t)$.

Notice that when there is no angular momentum the metric can be simplified since in that case we have $\beta^\varphi = 0$, and $C_1 = C_2 = 0$. The code controls this using the logical parameter `angmom`: if this parameter is true then those arrays are turned on, while if it is false they are off. By default the parameter is false, so that those arrays are turned off.

8 Matter

9 Slicing conditions

10 Shift conditions

11 Initial data

The type of initial data is controlled by the character-type parameter `idata`. If you add a new type of initial data it should be appended to the list of allowed values for this parameter in the file `src/base/param.f90`. You should also add a corresponding call to your initial data routine in the file `src/base/initial.f90`.

12 List of main code parameters

13 Editing the code

14 Ollingraph

The code includes a simple 1D visualization package called "ollingraph". It is written in Python and uses Matplotlib to plot simple line plots and animations. At the moment it should work fine under python 2.7, but is seems to have problems with python 3 (I will try to fix this later). It is supposed to have similar functionality to the old xgraph and ygraph packages, and expects the data files in the same format (see below).

The plots are quite simple, and meant for quick/dirty interactive visualization. These plots are not supposed to be used for figures intended for publication (use gnuplot or something similar for that). To use the package type:

```
ollingraph file1 file2 ...  filen
```

When plotting several data files at once, they are all assumed to be of the same type. One can also add a custom name for the plot using the option –title:

```
ollingraph --title="Plot name" file1 file2 ...  filen
```

If this option is not there the plot is just named after the name of the data file being plotted, assuming there is only one, or just says “Multiple data files” if there is more than one file.

Before using it make sure that `ollingraph` has execution permission:

```
chmod +x ollingraph
```

There is also a package for simple plots of 2D arrays called `ollingraph2D`. It is called in a similar way:

```
ollingraph2D file
```

But in this case the file to be plotted must have extension `.2D`.

The data files are expected to have the following format:

- 0D files (those ending in .tl):

  1. One comment line starting with `#` or `"` with the file name (this line could be missing and it should still work).
  2. A series of lines with the data points, with the x and y values separated by blank spaces.

- 1D evolution-type files (those ending in .rl):

  1. Each time step begins with a comment line starting with `#` or `"` that contains the time in the format:

     ```
     #Time = (real number)
     ```

  2. A series of lines with the data points, with the x and y values separated by blank spaces.
  3. One or more blank lines to separate the next time step.