Working Group: Initial Data
Chair: Prof. Pablo Laguna
Participants:

Summary:

Priority Merkers

- * - Less important
- ** - More important
- *** - Most Important

Type of Initial Data:

- Binary Black Holes -
  - High Spin (*) - Conformal Kerr background. Not on the top priority (since we still cannot measure the spins)
  - Extreme mass ratio (*)
  - Eccetric binaries (**)
  - Setting ID equivalent between different NR groups (***)
  - Eccentricity Reduction (***)

- Double Neutron Star:
  - Require better user interface to generate ID using LORENE
  - LORENE - Only support helical symmetry
  - LORENE - Can handle multiple equations of state, but how to tackle with ET (communicate the ID of LORENE for other equations of state with ET)
  - Add Support for spinning binary in LORENE
- Discussion about DNS Initial Data -
  - Definitely requires complete documentation
  - Code is well written and efficient but difficult to use.
  - Generating Initial data is possible but extremely painful process.
  - Adding new physics is almost impossible (unable to map directly the equations listed in paper to the ones in code)
  - Possible people to get help - Phillip, Chris, Koji Uryu, Keisuke Taniguchi
  - Should contact them if they are willing to help with this.

- Action Items :
  - Seek help for LORENE from - Phillip, Chris, Koji Uryu, Keisuke Taniguchi - Rewrite the documentation from scratch (from choosing the BNS configuration to generating the initial data - Josh Faber, Miguel Gracia , Shawn Rosofsky.

  - Add new physics to LORENE (new equation of state, support for spinning systems)
  - Require PDE solver to solve elliptic equations -

- Mixed Binary:
    - Source Code available with LORENE - very few know how to use this - Need documentation.
    - SXS does what exactly LORENE does but their code is strongly coupled with their evolution codes.

  Action Items - Hopeless, don't bother!

  Next Meeting -

    - Extend the meeting in such a way for people to work on this presentation

Instructions on using Lorene to generate Binary initial data.

-1: Read this paper: https://arxiv.org/abs/gr-qc/0007028 (Gourgoulhon, Grandclement, Taniguchi, Marck, Bonazzola)

0. Install Lorene. They have instructions, which often work pretty well. Let's define "HL" to be the directory you end up naming HOME_LORENE for brevity.

Step 1: You need to generate the executables. For Binary Neutron Star data, these reside in HL/Codes/Binary_star/. *There are similar versions in Bin_star; at some point we need to work out the real differences if any.* Make the following

- init_bin
- coal
- lit_bin

Step 1: Lorene can set up individual stars in isolation extremely quickly, since it is just a 1-d problem in spherical coords. You will need the following files, all placed in the same directory, of which all of the "*.d" files will be found in HL/Codes/Binary_star/Parameters:

- init_bin
- par_init.d
    - The coordinate distance is not absolutely critical, since it gets thrown away when we relax a binary configuration.
    - The central enthalpy seems to live on forever, so choose this value wisely! You will get a description of the stellar parameters in physical units, make sure these are what you want.
- par_grid1.d; par_grid2.d
    - Set nz to a handful. nzet=1 is typical.
    - The typical choices for np, nt, and nr respectively are
        - np= A "nice" fft number -- some multiple of powers of 2 and 3 is ideal
        - nt = np+1

        - nr = The next "nice fft number", plus 1
        - e.g,, 12 x 13 x 17; 16 x 17 x 25; 24 x 25 x 33
    - The last "nz" lines of the file contain the value of nr [keep these equal!] and then the inner radii of the given domain in NS radii. The first of these is zero, the second 1, and after that choose a radial ratio between 1.5 and 2 for ideal results.
- par_eos1.d; par_eos2.d
    - This is the EOS choice. Typically you would keep the mean particle mass at unity, but change the pressure coefficient "kappa" to yield the EOS model you want in physical terms -- LORENE does not really use dimensionless units.

The command should just be "./init_bin", possibly ported into a logfile.

The output is a binary file "ini.bin" [EJW: should this be “ini.d”?] containing the unrelaxed binary configuration with given stellar models and specified binary separation.

Step 2: You now have an unrelaxed binary, which is basically just two spherical stars that have been lightly superposed. To generate a relaxed binary configuration, we need the "coal" executable you generated earlier. Place it in a directory with the file "ini.d" you just generated in step 1, along with "parcoal.d" which can be found with the other "*.d" parameter files. Here are the key values to consider changing:

- fact_separ: The first thing the routine does is change the binary separation. Plan accordingly.

Finding a way to get this routine to converge nicely is probably the biggest barrier to use of LORENE for generic data. As measured by "diff_ent", the change in the central enthalpy on a step-by-step basis, one typically sees multistep cycles as it iteratively seeks convergence, and occasionally diverges instead.

Step 3: Take the file "resu.d" and the executable "lit_bin", and try "./lit_bin resu.d". You should get many, many plots showing you various functions for the binary.

Step 4. Next, let's generate a sequence. The routine coal_seq handles this task. Note that the current behavior is hardwired on lines 218 and 219. It begins at 65km separation, and decreases in steps of 5km down to separations of 30km -- somewhere along the line, a crah is likely, since it would take an extremely stiff EOS to allow for any kind of stability at those separations. Files are produced with the separation in the name.

Note that the routine currently picks a distance, computes a relaxed configuration, and once it does, jumps the separation downward by 5km and repeats the process. If you would like something more gradual, it would have to be built into the routine.

OPEN QUESTIONS FOR THE MOMENT:

1. There are BNS initial data routines in two different classes: Bin_star and Binary_star. It is worth checking out:

- Is there any difference between the two, really?
- If so, which is better for modern-day data?

2. There coal_seq routine seems as if it should gradually walk the binary stars together during a relaxation, rather than just doing it in one go. Is this what it actually does?
3. How does one damp the entire relaxation (more)? Try setting all five of the relaxation parameters in parcoal.d to zero ar something of order epsilon, and certain quantities will still evolve significantly step-by-step...

COMPILER FLAGS FOR RECENT MACS w/HOMEBREW -- Modify local_settings_darwin as follows. These worked for both Josh and Maria.

```
F77      = gfortran -ffixed-line-length-132
LIB_CXX = -L/usr/lib   -lstdc++ -lm -L/usr/local/Cellar/gcc/8.1.0/lib/gcc/8/ -lgfortran
##assuming gcc v. 8.1.0
LIB_PGPLOT = -L/usr/local/lib -lcpgplot -lpgplot -lpng -L/opt/X11/lib/ -lX11  #homebrew pgplot, system x11?
LIB_LAPACK = -L/usr/lib -llapack -lblas  #homebrew lapack (not atlas?)
```

Longer term items to better understand Lorene.

1. The original papers describe the iteration loops, but it would be helpful if we could come up with a user's guide to where each of the equations that make up a single step are solved. In some cases, various equations are iterated over within a single sub-step, for instance.

2. The routine that solves for Omega is a particular sticking point, worthy of another look to see if it is as robust as it could be. This has occasionally been a problem in the past.

3. There are five different relaxation parameters in parcoal.d. Setting them all nearly equal to zero does NOT actually freeze the solution in place, it just freezes parts. In order to achieve a better, critically damped iteration loop, we will need to work out guidelines on how to set the initial relaxation parameters, and whether they should evolve themselves as we approach an equilibrium solution.

4. The current scheme to generate a sequence is to pick a radius, relax there, and then change the separation by a finite amount and repeat the process. Since it takes a substantial number of iteration loops each time to attain convergence, more gentle "walking" schemes might be worth considering. We will have to deal with the potential issue of organizing output files, etc., but it seems like a solvable problem.