Tuesday July 7 - Methodology

Discussion Leads: Bruno Giacomazzo & Federico Cipoletta

Brainstorming Q&A:

1. **Software Infrastructure:**

    a. **What are the alternatives to Lorene (e.g., Kadath)? What about initial data for NS-BH systems?**
    It has been mentioned that there is no fundamental reason why the Lorene software should not be able to produce ID for BH-NS binary. The problem with Lorene seems to be only the need for implementation for a Puncture condition for BH because it seems that it only supports for BH excision up to now. Moreover, it was reported that the TwoPuncture thorn within the EinsteinToolkit (ETK from now on) seems to be broken and the community has lost trace of what and in which way it has been used for. Finally, the audience stressed on particular interest for BH-NS ID produced with Lorene, thus anyone who has news or ongoing work in this direction has been invited to share.

    b. **SphericalNR by Vassilios Mewes**
    A brief presentation of what has been achieved with the SphericalNR framework has been reported: BC (in view of MPI) in vacuum (1st paper). CCZ4 formalism+GRMHD (2nd paper). Developing a filter in order to solve CFL limitations (dt goes as CFL*dr*dtheta*dphi – filtering make it roughly up to 100 times faster - ongoing 3rd paper).TESTS: (T1) Rotating NS with tilted magnetic field wrt to rotation; (T2) Magnetic bar mode instability; (T3) heaviest and closest ID for BNS on Lorene (without symmetries). With Spherical coordinates you need to over-resolve in angular resolution, in order to deal with non-symmetric scenarios. The audience showed interest in the “filtering technique” which consists of Fast Fourier transforming the signals in order to damp down CFL unstable modes and which has been shown to be one essential tool against CFL limitations. The gaining in the dt factor is in some way problem dependent and one should always take care on how the grid and dt are chosen.. It has also been stressed the fact that the SphericalNR framework has been thought in order to render the ETK coordinate agnostic, meaning that x, y and z (ETK coordinates) are used for spherical coordinates, while variable defined within the framework (Cart_x, Cart_y, Cart_z) are used to store cartesian coordinates and used for explicit transformations. The audience seems also to be interested in

comparing of the filtering method with respect to what has been done in other codes (e.g. static mesh refinement or coarsening methods) but every method presents its own pro vs cons (e.g. the refluxing need in the case of SMR), therefore there is space for further investigations.

i. **What was NRPy+ used for in SphericalNR (optimization or other stuff)?**
Zach reported that the reason why NRPy was used is because it adopts reference metric spacetime formulation (several coordinates are available -- spherical-like, cylindrical-like, Cartesian-like, and prolate-spheroidal-like). It contains a covariant BSSN formulation implemented in the reference metric formulation is agnostic to coordinates, and Vassili extended the equations to CCZ4

c. **What is the status of the GR version of Athena++?**
Bruno and the audience in general reported interest in Athena++. This is a publicly available code that was without GR up to now. Bernuzzi and Radice are working in this direction and they should be in one advanced status. Unfortunately, the code developers were not present at the Breakout session, so the audience will for sure try to reach them out in order to ask for updates.

d. **Are there any tools available to port codes from Fortran to C/C++? How do GRMHD codes perform? What was the main factor that contributed to a factor 2 speedup in IGM compared to the original version of the code?**
Zach says: f2c - https://en.wikipedia.org/wiki/F2c. Also see https://software.intel.com/en-us/forums/intel-fortran-compiler/topic/623861 for complications that might arise.
   i. Zach says: The speedup was better use of processor cache & OpenMP; the former by breaking up various pieces of the underlying algorithm and being careful about reading data only once as needed. I optimized the pieces of the IGM algorithm separately to determine what approach worked fastest.
   ii. The opinions of the audience seem to be heterogeneous: some enlight the fact that Fortran presents many advantages when dealing with HPC (every cluster has its own f compiler; the language seems to be highly optimizable; it is easily interfaceable with Python; its wrappers are self-contained; performance of the code mainly depends on the algorithm, so programming skills are the main point in order to obtain one efficient code); other think that C/C++ language is the modern and future way of programming (students are more interested in these languages, which are the ones often required by private companies; it allows for TASK PARALLELIZATION, which is the actual way chosen by the ETK

community, more than GPUs programming). Moreover, taking the programming to a higher level (e.g. via Python which presents several ways of porting to Fortran or C) has been reported as one additional solution because nowadays it seems to be easier to find a good Python programmer rather than a C or Fortran one.

e. **The GRMHD software infrastructure community is concentrated in writing several codes to simulate BH/NS and BNS e.g. IGM/NRPy+, Spritz, ETK/GRHydro, BAM, Athena++, Harm3D, SphericalNR, Spectre, MHDuet, etc. Can we start sharing best code practices, tools and enhance collaborative environments in a more systematic way so that progress can be accelerated? How to do this? Suggestions?** Mark A: What about starting an equivalent to "stackoverflow" for GRMHD in the community? Centered around a place for students/postdocs/people change or expanding codes? The advantage of having such a tool is that it can cut time to learn systematic infrastructure points.

The audience seems to homogeneously accept this as a good suggestion. More is to be done in this direction. How to do that?

Some reported that the documentation is another way of going (NRPy, Doxygen were explicitly mentioned there). Performing extensive testing is really important even for documentation (eg. in the case of NRPy), in order to be sure of not breaking anything with new commits.
Bruno expressed the necessity of following software engineering best-practises in order to avoid "reinventing the wheel" every time.

i. **Efficiency: AMRex/Carpetx, Simflowny, Athena++, Spectre, Patchwork, LAMA, etc**
Carlos Palenzuela: Simflowny is software to generate code; there are publicly available codes for infrastructure or AMR (AMRex); you only need to put eqs in infrastructure and this is the same with Simflowny; you can measure speed and scalability in a very realistic way adopting such tools. SUGGESTIONS: 1) infrastructure – AMRex; 2) design some tests for scalability. Top500 list of GRMHD code - what was done in the "Apples to Apples" project. The **IDEA** that arises is to pursue one common "ranking project", listing and assigning rates to GRMHD codes, focusing on explicitly measuring the ACCURACY, STABILITY and EFFICIENCY of each code, keeping attention to what the code is meant for (e.g. some codes may be best performing in presence or absence of particular symmetries) - a sort of "Apples2Apples" round 2.

2. **Is anyone working on GPU codes for GRMHD? E.g., Tchekhovskoy reported very good results for his accretion code (https://arxiv.org/abs/1912.10192). (IH:https://github.com/AlexJamesWright/METHOD is SR only, but written with GR in mind)**
   Manuela: he was calculating jets – any Microphysics? HARM is pretty efficient. What is the reason for going for GPUs?
   Mark: they wanted very high resolution/grid cell counts for some of their physics problems. They also have AMR, which allows them to achieve bigger timesteps.

   a. **Manuela: there are some hardware limitations with GPU systems as they are not cost/effective from the hardware point of view e.g. some large supercomputers in the US didn't invest in a large GPU based system. This is a factor to consider when one writes complex codes plies the human cost involved.**
      The audience seems to agree on the fact that the GPU's cost (as for every product in the world) is given by the relation between the material production and the human cost involved. The main concept that comes out is that the Numerical Relativity community is still not ready to directly go for GPUs. Some people are already involved in this direction but not all want to invest their own human time in order to port the codes, if there is no means of using them. GPUs are not for general-usage but rather seem to require ad-hoc coding. What physical scenario may gain the best from GPUs? ETK seems to be moving in allowing for that (in future). There is probably room for collaboration.

   b. **Bruno: Italy just invested in a GPU cluster (Marconi 100). This is our main publicly available cluster in Italy and it is part of the EU PRACE network. It is the 9th fastest supercomputer in the world (Top500 list).**
      The EU PRACE network accepts proposals from investigators from every country. These are very competitive grants. For sure, the part of the community using the ETK framework is in some way dependent on the decisions of this. The actual trend of the ETK is to prefer Task Parallelization wrt to GPUs porting, but the discussion topic may be brought to the ETK North American Workshop (https://www.cct.lsu.edu/Einsteintoolkitworkshop for infos).

   c. **Ari: I also know that Frontier will be an amazing supercomputer with GPU power. https://www.olcf.ornl.gov/frontier/**

3. **Suggestions for the Handoff atmosphere problem? (Angle-average the atmosphere. See if order of operations matters. Pseudo-evolve the atmosphere to stability by freezing the interior. Accept that all atmosphere**

algorithms are wrong, and change IGM's to get its values from HARM. Change the interpolation scheme to a full recursive-order-reduction WENO scheme, taking care of the finite volume scale factors.)
Zach: we have some interpolation induced issues in the Handoff

a. **MC: Interesting idea. We should discuss it more.**

b. **Zach says: Yes, these are neat ideas. Modifying IGM would be challenging as it would require an extensive overhaul, with the likely result being a less robust code (IGM must simulate dynamical spacetimes including BHs, and stability inside horizons involves a number of routines that extend to the atmosphere).**

c. **IH: I will not be able to make the discussion, but I would note (1) shocks and vacuum are both much easier to generate at low density, and this depends on the details of the numerics (hence the shock generation issue). (2) the atmosphere is essentially a numerical equilibrium; at the transition to atmosphere I would expect the equivalent of "startup error" in the handoff. (3) treating a radius in the atmosphere as a transition/boundary between the two different schemes and doing an analysis of that BC (like in the Ghost Fluid Method - see http://eprints.soton.ac.uk/375551/, or like MR boundary analysis) may give useful information, particularly on thermodynamic consistency. None of the hacks I suggested above (with the possible exception of angle-averaging to re-init the atmosphere) address thermodynamic consistency.**

The discussion here focused on the atmosphere treatment in the Hand-off project. Porting data from one coordinate framework to another (namely from cartesian to spherical) necessarily injects some numerical noise which may break the code stability. Moreover, the atmosphere treatment is a delicate point in each GRMHD code, because every code has its own prescription or requirements for the atmosphere, while no one is really sure of what is physically the correct treatment for that. Some (if not all) codes should add further prescriptions when treating some problems (e.g. a BNS merger + post merger disk) and this is not new in the community (several papers report different prescriptions). The suggestion of simply averaging the values in the atmosphere may be too crude and one algorithmic average has been suggested by Zach. Moreover, as the cleaning procedure based on pre-hand-off VS post-hand-off data in the disk 1D profile has brought a huge improvement, maybe this is the way to go. It was asked what is done for the velocities in the atmosphere and it seems that they are kept to zero. This choice may present some issues. There was a mention to a possibility of cleaning the lapse function, but then the time was over. More is to come in the next hand-off telecons.

d. Zach says: Here are a couple ideas that might help:
    i. At each point below a certain density threshold implement the following “minesweeper” approach
        1. Look at all the 6 (or 26) nearest neighbors. Find the neighboring point (that also falls below the density threshold) with the minimum value of the pressure and copy those GRMHD (or just pressure?) data over to the current point. This would remove some of the high-pressure noise from the atmosphere. One could in principle apply this algorithm through multiple passes (though in the limit of infinite passes, it’ll replace the entire atmosphere with the lowest values -- not desirable). Also one needs to be careful with the order at which the algorithm is applied… maybe store a copy of the original grid function. I think it’d probably be a good idea to use this approach only with 1st order interpolation data.
    ii. Instead of sharp or threshold transitions, instead modify quantities smoothly, as a function of density using a transition function like erf(f(rho)) or tanh(f(rho)). I fear that current on/off thresholding can cause (or exacerbate) the shocky behavior we see.
    iii. If you’d like to try a prescription that evolves the atmosphere but freezes the disk, you could try simply setting the lapse to zero in the disk, but transitioning to the handoff value in the atmosphere. This will cause outer layers of the disk to evolve very slowly, but the atmosphere to evolve at the normal rate. My worry is that a sharp evolve/no-evolve transition might cause problems.

Key References

- Simflowny is a platform that runs under SAMRAI: https://bitbucket.org/iac3/simflowny/wiki/Home
- https://stackoverflow.com/help/on-topic
- Nuitka, automatic Python -> C conversion: http://nuitka.net/
- tensorflow GPU support is native to Google Collab: https://colab.research.google.com/notebooks/gpu.ipynb, https://towardsdatascience.com/heres-how-to-use-cupy-to-make-numpy-700x-faster-4b920dda1f56
- https://github.com/kokkos/kokkos