NZ Summer School on Gravitational Waves 2023

Lab 2

Getting started: To open the Jupyter environment on your browser, go to $130.216.216.114$: 9000, login with yesterday's credentials and open g_wave_setup.ipynb in the Lab2 directory.

Background: Now that we have performed checks of correctness and are confident the implementation is sound, let's have a play!

- **CFL**: The CFL number is very important for stability of numerical evolutions. A common range for numerical relativity is 0.2 to 0.5, and theory says it should be no more than 1.
- **SAT method**: Our system has two PDEs of the form $\partial_t u + c\,\partial_z u = F$, which propagate with a direction determined by the sign of c: If c is positive then u propagates from left to right, and the other way round if c is negative. Equations of this form are called an *advection equations*. We have two advection equations, corresponding to the two waves, with the opposite propagation direction. When the direction of propagation is *into* the computational domain at a boundary, we need to impose boundary conditions there. The SAT method is a way to stably impose boundary conditions when paired with a finite difference operator that has the Summation-By-Parts (SBP) property – the discrete version of integration by parts. It works by adding a term to the evolution equation at said boundary that drives the variable to the desired value. Simply replacing the variables values at a boundary with another is known to be unstable in general.
- **g_wave.py**: This file contains the following methods:
  - kpbump_pol: This returns the wave profile at the boundaries. This and an appropriate number of its derivatives should smoothly deviate from the initial data.
  - left/right: Auxillary functions to grab the boundary conditions.
  - evaluate: Evaluates the evolution equations.
  - initial_data: Sets initial data, in this case, Minkowski space-time.
  - timestep: Returns the timestep given a spatial step size and CFL.
  - constraint_violation: Calculates the constraints.
- **The Kretschmann scalar**: This is the scalar curvature invariant $R_{abcd}R^{abcd}$ that is independent of your choice of frame. This means that it can help decide whether a space-time singularity is physical or not. The code is commented throughout, have a look to see what is going on.

Tasks:

1. Run the code and watch the animation for different CFL values. What happens when the CFL is chosen too large?

2. In the evaluate routine, C_left and C_right are the characteristic speeds of the waves. What would happen if you had made a mistake and these were the wrong sign?

3. In g_wave.py there is a blank method called "my_profile" where a represents the amplitude of the waves and t is time. What happens if you change the wave profiles to be the constant $a/2$? This is the wave profile of a gravitational shock wave. You will have to change the left and right methods to call your new method instead of "kpbump_pol". Do you see the little wiggles in the waves? What are these?

4. Make another wave profile that imposes $a\sin(8t)^7$ for $0 \leq t \leq 2\pi/8$ and zero afterward. Assign this profile to the left and right boundaries as before. Run the simulation using a resolution of 400 and save your data. What happened during the simulation?

5. Open kretschmann.ipynb and run it, making sure the name of your HDF file is correct. This reads in the simulation data and calculates the Kretschmann scalar. Write some code to plot the Kretschmann scalar as a contour or surface, loading in the appropriate matplotlib libraries. Looking at your visualization, what happens to the Kretschmann scalar? What does this say about the future of this space-time?

6. **Challenge question:** Experiment with different boundary conditions. Can you shoot in a train of waves from each boundary? What happens to all these simulations if they're run long enough?

Debrief: There are many different technical parameters that go into numerical relativity simulations, even one as basic as this. A stable simulation requires all of them to interact together well. It is clear that if even one of these parameters, such as the CFL, are not chosen well then the simulation will crash.

Scalar curvature invariants are a useful way of checking whether a singularity is physical or an artifact of the gauge.