Einstein Toolkit + FLRWSolver tutorial

GR simulations in cosmology workshop, September 7 & 8 2020: Zoom

Hayley Macpherson
hayleyjmacpherson (at) gmail (dot) com

In this tutorial we will run through the steps of running a simulation using the Einstein Toolkit (ET) with cosmological initial conditions provided by `FLRWSolver`.

We assume you have successfully downloaded and compiled the ET according to this tutorial.

The version of FLRWSolver you have can set up three kinds of initial conditions:

1. FLRW spacetime (i.e., with no perturbations, pretty boring)
2. FLRW + a single-mode, linear perturbation (with controls for the amplitude, wavelength, direction, and phase offset, slightly less boring)
3. FLRW + Gaussian random, linear perturbations following a given power spectrum (Much less boring)

Here we will be focusing on the third kind of initial conditions, which are similar to those used in Macpherson, Price, & Lasky (2019, arXiv:1807.01711).

The purpose of this tutorial is to help you to perform interesting, realistic cosmological simulations and hopefully help us out in the hunt for important GR effects!
If you do end up using this code structure for interesting science (or non-interesting science), please remember to cite at least this paper. Happy simulating!

The ET executable that you made when compiling (`Cactus/exe/cactus_sim`) is run with a parameter file (extension .par) as an argument. This tells the ET which thorns to use, and sets the required options for each of those thorns. So, we're just telling the code what kind of simulation we want to run.

There are several sample parameter files included in the `FLRWSolver` repository, located in the `par/` directory. Each of these represent one of the spacetimes in the list above.

1. **Choose your parameters:**

   Open the file `par/FLRW_powerspectrum.par` in your favourite editor.

   The parameter `ActiveThorns` tells the ET which thorns to, well, activate during the simulation. The syntax for setting the parameters is:

   ```
   ThornName::parameter   = "value"
   ```

   Parameters can be real, integers, booleans, or characters. If you want to know what a specific parameter means (or what values it can take), you can find it in the relevant thorns `param.ccl` file.

You shouldn't need to change much here, the only thing you **definitely** need to change is the path to the text file containing the power spectrum. This is given by:

```
FLRWSolver::FLRW_powerspectrum_file =
"/some/path/to/file.txt"
```

Which will currently be set to *the path to this file on my laptop*, which is not much use for anyone else. There are two power spectrum files in
`FLRWSolver/powerspectra/`
For this tutorial, use `FLRW_matterpower_z1100.dat` (i.e., at the CMB), to make sure the perturbations are linear (there is also a z=200 file there, for fun, but be careful if using this).
Change the parameter to the path to this file on your own computer.

2. **Use Simfactory to start your simulation:**

Navigate back to your `Cactus/` directory. We will be using Simfactory again to set the simulation running.
Run this command to start your simulation:

```
./simfactory/bin/sim create-run sim_name --parfile
/path/to/FLRW_powerspectrum.par --cores=1 --walltime=0:05:00
```

Where `sim_name` is (unsurprisingly) the name of the simulation, call it whatever you like. Replace `/path/to/FLRW_powerspectrum.par` with the actual path to the parameter file on your machine. The parameter file was set up to only run 10 timesteps, so 5 minutes is plenty (it shouldn't take much more than 1 minute).
(Note: I had to specify `--num-threads` and `--ppn=used` as well to avoid a *warning*, but everything worked fine without these.)

This will create a directory called `sim_name/` in your `$HOME/simulations/` directory (if this doesn't exist, it will create one, I think?). In this directory is a huge amount of information about the simulation you've just run - take a look around and check it out! The simulation data itself (i.e. ascii and HDF5 output) is located in:
`sim_name/output-0000/FLRW_powerspectrum/`

3. **Visualise your output:**
Navigate to the directory containing your simulation data.
(If you want to use your own way to visualise HDF5, go ahead now!)

If you want to use `splash` (and have already downloaded + compiled it), read on:

ET 3D output in HDF5 format includes one file *per variable*, which includes every timestep of output. This format is not ideal to read into `splash`, so we need to split these HDF5 files so that we have *one file per timestep*, which includes every variable we chose to output. I have written a Python script to do this which is included in your `flrwsolver/tools/` directory:

```
python /path/to/split_HDF5_per_iteration3.py
```

There is another with a ‘2’ on the end instead of ‘3’, this is just which Python version it works with. From this you should get 6 files in total (the parameter file only specified 3D output every 2 iterations), with the general naming convention:
`{parfilename}_it{iteration_number}.hdf5`
For us, these are `FLRW_powerspectrum_it000000.hdf5`, etc.
These files are compatible with `splash`, so you can run:

```
csplash-hdf5 FLRW_powerspectrum_it000000.hdf5
```

And follow the prompts to visualise your universe! (note you can use `splash` to visualise the regular ascii files by typing, e.g., `splash rho.average.asc`, etc.)
Feel free to play around and adjust the limits, kernel, etc., as you please. If you can’t be bothered doing this, I have included some files that will adjust these for you in the `flrwsolver/tools/` directory. Copy these to your simulation directory and restart `splash`:

```
cp /path/to/flrwsolver/tools/splash.* .
csplash-hdf5 FLRW_powerspectrum_it000000.hdf5
```

You should now be able to clearly see your power spectrum of perturbations. If you can’t, ask me on Slack.

4. **Load the data into Python for more visualisation and maybe analysis:**
   In this tutorial we won’t be doing any actual analysis (sad face), but I have included a Python notebook to give you an example of how to read in your simulation data to be able to play around with it (but also check out some of the other ways to analyse and/or visualise ET data here).

   There is a Python notebook in the `flrwsolver/tools/` directory:

   ```
   jupyter notebook /path/to/flrwsolver/tools/Plot_ET_data.ipynb
   ```

   Just run through that, and let me know on Slack if you have any questions!

That’s it for the tutorial - but below is some extra stuff for fun

---

5. **Advertisement:** `mescaline`
   Over the last 5 years or so, I have been developing a tool called `mescaline` (extracting interesting things from Cactus) that I started developing with Daniel Price during my PhD.
   This is a code that reads in the HDF5 data in the “split” format you now have, and calculates a bunch of interesting things for inhomogeneous cosmology. This currently includes the Ricci curvature (4D and 3D), expansion scalar, shear, vorticity, acceleration, backreaction, and cosmological parameters. It also includes raytracing, and other interesting stuff.
   Stay tuned - this will be available for public use at some point!

6. **Optional further steps:**

   a. **Feel free to use the other parameter files** to run a super-interesting FLRW spacetime with or without a sine-wave perturbation!

   b. **Use different power spectra from CAMB (or elsewhere)**
   Be careful about which scales you sample here, since CAMB outputs P(k) in the synchronous gauge, which is not the same as the gauge specified in the parameter files supplied. You will need to make sure these coincide.

   c. **Use a restricted number of modes**
   If you want to study your perturbations in a more controlled setting, you can limit the number of modes while still sampling from the power spectrum.
   Say you want to study perturbations that are > 10*dx (where dx is your grid spacing), then you need to find the corresponding wavenumber, k, to which this scale corresponds. Then, just copy your power spectrum file and set the power to zero at all of the k > k_cut that you choose.

   d. **Play around with different box sizes**
   This is controlled using `FLRWSolver::FLRW_boxlength` (physical size of box in Mpc/h), but this is only used when generating a power spectrum of perturbations