Downloading & Compiling the Einstein Toolkit + FLRWSolver

(this tutorial was written for the GR Simulations in Cosmology workshop "in" London in September, 2020)

Hayley Macpherson
hayleyjmacpherson (at) gmail (dot) com

In this tutorial we will work through the basics of downloading and compiling the Einstein Toolkit (ET), specifically for use with FLRWSolver.
The ET is a collection of codes built for numerical relativity, based on the Cactus infrastructure.

FLRWSolver is an initial condition module initially developed by Paul Lasky at Monash University. It allows us to simulate cosmological spacetimes using the usual evolution thorns in the ET.
This thorn has been extended over the years to initialise an FLRW spacetime with or without different kinds of small perturbations.

---

0. **Dependencies:**
   The ET relies on several external libraries. What you will download does include a copy of some of these libraries, but compilation will be much faster if you can instead link to your own local versions of these libraries instead. See point 6 for how to do this. The main libraries that will be useful to have are: HDF5, FFTW3, GSL, LAPACK, MPI, OPENSSL, ZLIB. Some of these may be included on your machine by default (I think), but if you have issues with any of these while compiling, try installing them using Homebrew (i.e., `brew install` on Mac, or similar for whatever machine you're using).

   The (new and exciting) version of FLRWSolver we are using in this workshop has Python dependencies. These are `cffi, numpy, and scipy`. Make sure you have all of these plus an up-to-date Python version (I tested on 3.8). For the tutorial, we will also use a `jupyter` notebook, so make sure this is installed too.

1. **Download ET:**
   → Head to einsteintoolkit.org
   → Navigate to "Download" page
   → Follow the instructions to download the current release
   → (Mac users: see the end of this document if you get an issue running ./GetComponents … when using Xcode \gtrsim 11.1)

2. **Download FLRWSolver:**
   → Go into your new `Cactus/` directory
   → Navigate to `arrangements/EinsteinInitialData`
   → "*git clone*" or direct download the FLRWSolver repository into this directory
   → Re-name the cloned directory (`mv flrwsolver_public/ flrwsolver/`)
   → Follow the instructions in `README.Python` to set up for Python linking
   → Optional*:* open the following file in your favourite editor:

```
arrangements/EinsteinEvolve/GRHydro/src/GRHydro_UpdateMask.F90
```

and comment out line 76, e.g.:

```
! atmosphere_mask_real(i,j,k) = 1
```

If you're curious why, read Section 5 of the FLRWSolver documentation (in `doc/`).
This is only necessary if you want to do long evolutions (and use this setup for very serious and interesting cosmology), and the tutorial should work fine without doing this.

3. **Add FLRWSolver to the list of thorns to be compiled ("thornlist"):**
   → Open `Cactus/thornlists/einsteintoolkit.th` in your favourite editor
   → Find the list of `EinsteinInitialData` thorns
   → Add in the line: `EinsteinInitialData/flrwsolver` beneath the others *(make sure the naming is consistent with respect to the actual name of the thorn, e.g. capital letters)*

4. **Compile:**
   To compile we use Simfactory which helps us set up and handle simulations with the ET.
   → Set up simfactory using this command:

   ```
   ./simfactory/bin/sim setup
   ```

   from your `Cactus` directory (the answers you give here don't really matter).

   → Choose your config file from the directory `simfactory/mdb/optionlists` (I use a Mac with Homebrew so `osx-homebrew.cfg` works for me, with some minor modifications. For Ubuntu the relevant config file is `generic.cfg`)
   → To compile, run this command:

   ```
   ./simfactory/bin/sim build --thornlist=thornlists/einsteintoolkit.th
   --optionlist=osx-homebrew.cfg
   ```

   With your chosen config file in place of mine.
   This can take a while (about 3 hours for me...), so set it going and do something else.
   (see `FLRWSolver/README.Compilation` *for some common compile-time errors)*

5. **Test your executable:**
   We now should make sure that the executable you created actually works (thanks Francesca & Robyn!). So just run a simple Hello World (from your Cactus/ directory):

   ```
   ./simfactory/bin/sim create-run helloworld --parfile
   arrangements/CactusExamples/HelloWorld/par/HelloWorld.par
   ```

   And hopefully this runs (and greets you) without errors! If not, please ask us in Slack.

Extras:

6. **Choosing config options:**
   The config file specifies things like which compilers to use (e.g. variables F90,CPP,CC), which flags to send to those compilers (e.g. F90FLAGS, CPPFLAGS, etc.), some other flags to do with optimisation, debugging, etc. It also specifies the path to some libraries necessary to run the ET. These are found at the bottom of the file, examples include FFTW3, BLAS, GSL, HDF5, etc.

   → *What do you need to change?*
   You may be able to just run with the config file as it is, but this would be very lucky. Chances are your compilers and libraries are **not** in the same location as specified in the config file. You may need to locate some of the compiler binaries, and external libraries, on your machine.
   We will use HDF5 as an example. The config file defines:

   ```
   HDF5_DIR
   HDF5_INC_DIRS
   HDF5_LIB_DIRS
   ```

   These specify the path to where the various files needed to use the HDF5 library are located. Don't worry too much about the details unless you're interested.
   It's quite likely your HDF5 will be located in `/usr/local`, in which case you would specify

   ```
   HDF5_DIR = /usr/local
   HDF5_INC_DIRS = /usr/local/include
   HDF5_LIB_DIRS = /usr/local/lib
   ```

   From which you hopefully get the idea. A good way to check where your local installation of HDF5 is is to use the `which` command on one of the HDF5 binaries, for example typing

   ```
   which h5ls
   ```

   Will return the path to where the binary `h5ls` (which is used to list the contents of a HDF5 file) is located, and your HDF5 will be installed in a directory or two above this.

7. **Things to remember:**

   → Cactus is notoriously annoying to compile, so take some deep breaths and stay calm

   → Stack Overflow is your best friend

   → Please ask us on Slack if you get stuck, because anyone who has compiled the ET as many times as me (and some others here) are *very likely* to have seen your exact error before.

Visualisation

When we actually run a simulation during the tutorial, we will want to look at our beautiful universes somehow! The ET outputs HDF5 data for 3-dimensional output (and regular ascii for scalar output), so if you already have your own way to visualise HDF5 that you're used to (e.g. VisIt, Python), then you can skip this and just use that if you like. Another way (specific to ET output) is to use splash (written by Daniel Price).

If you choose to use splash, download and compile it according to the instructions on the website (here).

Potential issues when downloading / compiling:

Many of the issues you will run into when downloading and compiling will be able to be answered by one of us on Slack. Also check the end of `FLRWSolver/README.Compilation` for common issues when compiling / linking with Python.

Known issues when downloading ET:

Downloading the components of the ET, i.e. running (as per the download page)

```
./GetComponents --parallel

https://bitbucket.org/einsteintoolkit/manifest/raw/ET_2020_05/einsteintoolkit.th
```

Using Linux: you might see an error like this (on Linux Fedora):

```
Can't call method "enqueue" on an undefined value at ./GetComponents line 1032.
```

Try removing the `--parallel` flag and re-running. If this doesn't work, let us know.

Using MacOS: some of the components are downloading OK, but you see a warning for some of them (for me this was several of the `ExternalLibraries` thorns), e.g. (on MacOS Catalina with Xcode 11.1):

```
Warning: Could not checkout module ExternalLibraries/libjpeg
svn: error: The subversion command line tools are no longer provided by Xcode.
```

Hopefully this error is quite self-explanatory. New versions of Xcode no longer include support for subversion (svn). If you use Homebrew, run this:

```
brew install svn
```

Which fixed the problem for me. If it doesn't fix it for you (or you don't use Homebrew), there are other suggestions here:

https://stackoverflow.com/questions/60869347/mac-command-line-tools-11-4-no-longer-has-svn

If none of those work, let us know.