SPIN 2018 Exercise

Prerequisites

You will need a number of packages installed in order to download the Einstein Toolkit components. On a Debian, Ubuntu, Linux-Mint, Fedora or Mac OSX-based system, install them as follows:

```
# Debian (stretch, buster)
su -c 'apt-get install build-essential pkg-config libopenmpi-dev openmpi-bin gfortran git subversion curl
# Ubuntu (16.04.2, 17.04, 17.10)
sudo apt-get install build-essential pkg-config mpich2? python libmpich2?-dev gfortran git subversion curl
# Fedora (FC 25, 27)
su -c ' yum -y install mpich2 python pkg-config mpich2-devel hdf5 hdf5-mpich-devel gcc gcc-c++ gcc-gfortran
patch numactl-devel numactl hwloc subversion git'
# Mac OS (High Sierra), MacPorts
sudo port -N install pkgconfig gcc7 openmpi zlib hdf5 +fortran +gfortran svn
# Mac OS (High Sierra), Homebrew
brew install gnuplot pkg-config gcc hdf5 --with-fortran hwloc jpeg szip open-mpi
# Windows 10
# please install the Ubuntu Linux subsystem (bash) from
# https://msdn.microsoft.com/en-us/commandline/wsl/install_guide
# then follow along using the Ubuntu instructions
```

Please make sure all required packages are correctly installed before proceeding with the next step.

Downloading

A script called GetComponents is used to fetch the components of the Einstein Toolkit. GetComponents serves as convenient wrapper around lower level tools like git and svn to download the codes that make up the Einstein toolkit from their individual repositories. You may download and make it executable it as follows:

```
curl -O -L https://raw.githubusercontent.com/gridaphobe/CRL/ET_2018_02/GetComponents
chmod a+x GetComponents
```

GetComponents accepts a thorn list as an argument. To check out the needed components:

```
./GetComponents https://wiki.ncsa.illinois.edu/download/attachments/58199679/spin2018.th
```

which downloads the master thorn list from the Einstein toolkit server and proceeds to download all thorns. This thornlist checks out Cactus, the Einstein Toolkit, and Simulation Factory.

Configuring

You may proceed to configure Simfactory which requires some changes for some OS.

```
cd SPIN2018

# for Debian
./simfactory/bin/sim setup-silent --optionlist=debian.cfg --runscript debian.sh
# for Ubuntu, Mint
./simfactory/bin/sim setup-silent --optionlist=ubuntu.cfg --runscript debian.sh
# for Fedora (you may have to log out and back in if you have just intalled mpich to make the module command
work)
module load mpi
./simfactory/bin/sim setup-silent --optionlist=fedora.cfg --runscript debian.sh
# OSX+MacPorts
./simfactory/bin/sim setup-silent --optionlist=osx-macports.cfg --runscript osx-macports.run
# OSX+Homebrew
export CPATH=/usr/local/include LIBRARY_PATH=/usr/local/lib
./simfactory/bin/sim setup-silent --optionlist=osx-homebrew.cfg --runscript generic-mpi.run
```

After this step is complete you will find your machine's default setup under ./repos/simfactory2/mdb/machines/<hostname >.ini

You can edit some of these settings freely, such as "description", "basedir" etc. Some entry edits could result in simulation start-up warnings and/or errors such as "ppn" (processor-per-node meaning number of cores per cpu), "num-threads" (number of threads per MPI rank) so such edits must be done with some care.

Building

Now that you have configured Simfactory, you may build:

```
./simfactory/bin/sim build --mdbkey make 'make -j2' --thornlist=thornlists/spin2018.th
```

Adjust -j2 to match the number of cores your machine possesses if you want to use more or less than 2 parallel build processes. This may take a while, as it compiles all the thorns specified in manifest/einsteintoolkit.th.

Running

```
exe/cactus_sim arrangements/CactusExamples/WaveMoL/par/gaussian.par
```

Which will run a short test simulation and should produce lots of output like this

```
       10
  1   0101       ************************
  01  1010 10      The Cactus Code V4.2.3
 1010 1101 011      www.cactuscode.org
  1001 100101    ************************
    00010101
     100011     (c) Copyright The Authors
      0100      GNU Licensed. No Warranty
      0101
------------------------------------------

Cactus version:    4.2.3
Compile date:      Feb 22 2018 (13:53:09)
Run date:          Feb 22 2018 (13:53:54-0600)
Run host:          ekohaes8.ncsa.illinois.edu (pid=63822)
Working directory: /data/rhaas/postdoc/gr/cactus/SPIN2018
Executable:        exe/cactus_sim
Parameter file:    arrangements/CactusExamples/WaveMoL/par/gaussian.par
...
INFO (IOBasic): Periodic scalar output requested for 'WAVEMOL::phi'
INFO (IOBasic): Periodic info output requested for 'WAVEMOL::phi'

-------------------------------------------------
   it  |          | WAVEMOL::phi                |
       |    t     | minimum      | maximum      |
-------------------------------------------------
     0 |    0.000 |-3.332752e-11 |   1.00000000 |
     1 |    0.010 |-2.449807e-21 |   0.97117632 |
     2 |    0.020 |-2.626456e-11 |   0.88848624 |
     3 |    0.030 |-1.124029e-10 |   0.75905317 |
     4 |    0.040 |-1.943181e-10 |   0.59370187 |
     5 |    0.050 |-1.136465e-19 |   0.40576706 |
...
    98 |    0.980 |  -0.02801168 |   0.00291381 |
    99 |    0.990 |  -0.02545322 |   0.00287969 |
   100 |    1.000 |  -0.02429672 |   0.00289761 |
Done.
```

It will also create a directory `gaussian/` with more output files (plain text files). Open the file `gaussian/phi_maximum.xg` which contains the maximum value of the variable phi over time and note how it changes over time (the first column is time, the second one is phi).

Exercise

Open the file `arrangements/CactusExamples/WaveMoL/src/WaveMoL.c` which contains the C code used to compute phi and a quantity called energy. Find the function `WaveMoL_Energy` which is located at the end of the file. The function is called once per timestep and computes a quantity `energ` which is always positive (or zero). The loops

```
for (k=0; k<cctk_lsh[2]; k++)
{
  for (j=0; j<cctk_lsh[1]; j++)
  {
    for (i=0; i<cctk_lsh[0]; i++)
    {
      index = CCTK_GFINDEX3D(cctkGH,i,j,k);
      energy[index] = phit[index]*phit[index] +
        phix[index]*phix[index] +
        phiy[index]*phiy[index] +
        phiz[index]*phiz[index];
    }
  }
}
```

compute the array of energies and store the results in `energy[index]`. Add code to `WaveMoL_Energy` that

1. computes the maximum value of `energy[index]`
2. writes this value as a function of time to a file `energy_maximum.xg` using the same format as `phi_maximum.xg` . Time is accessible as the variable `cctk_time` .

Please compile the code once more, run it and send me the modified source file as well as `energy_maximum.xg` to me before the interview.

```
./simfactory/bin/sim build --mdbkey make 'make -j2'
exe/cactus_sim arrangements/CactusExamples/WaveMoL/par/gaussian.par
```

Hint

You can access the value of `phi` as `phi[index]` and check that your code produces the same results as found in `phi_maximum.xg`.

If you prefer to use C++ instead of plain C you can rename the file `WaveMoL.c` to `WaveMoL.cc`, update the file `arrangements/CactusExamples/WaveMoL/src/make.code.defn` to list `WaveMoL.cc` and recompile.

This fragment outputs time:

```
printf("Current time: %g\n", cctk_time);
```

Solution

If you get stumped with the exercise, here's a worked out solution. Obviously you cannot send me this one before the interview 🙂.