Cauchy-Characteristic Extraction using the Einstein Toolkit

M. C. Babiuc, B. Szilagyi, J. Winicour, Y. Zlochower

Aug, 2011 EinsteinToolkit Telecon

License

This code is available under the GPL General Public License, version 2, or later (see GPLv2 file included).
The code in this arrangement is a results of many years of work by many different authors. Major contributors to this code include (in alphabetical order)

- M.C. Babiuc
- N. T. Bishop
- R. Gomez
- R. A. Isaacson
- L. Lehner
- P. Papadopoulos
- C. Reisswig
- B. Szilagyi
- J. Welling
- J. Winicour
- Y. Zlochower

If you use this code, then we request that you cite the following papers:

- Title: Characteristic Extraction Tool for Gravitational Waveforms Authors: M.C. Babiuc, B. Szilagyi, J. Winicour, Y. Zlochower arXiv:1011.4223 submitted to Physical Review D
- Title: Cauchy-characteristic matching Authors: N. Bishop, R. Isaacson, R. Gomez, L. Lehner, B. Szilagyi, J. Winicour arXiv: gr-qc/9801070 In "Black Holes, Gravitational Radiation and the Universe" eds. B. Iyer and B. Bhawal (Kluwer, Dordrecht, Boston,1999)

This code is maintained by
M. C.Babiuc <babiuc@marshall.edu> B. Szilagyi <bela@caltech.edu> C. Reisswig <reisswig@tapir.caltech.edu> Y. Zlochower <yosef@astro.rit.edu>
Please contact the maintainers of the code if you require write access.

How to checkout the code

1. Download GetComponents.

```
wget --no-check-certificate \
  https://github.com/gridaphobe/CRL/raw/ET_2011_05/GetComponents
chmod 755 GetComponents
```

2. Download the development version of the EinsteinToolkit thornlist

```
wget http://svn.einsteintoolkit.org/manifest/trunk/einsteintoolkit.th
```

3. Edit the thornlist and add the following immediately after the `EinsteinUtils` thorns.

```
PITTNullCode/NullConstr
PITTNullCode/NullDecomp
PITTNullCode/NullEvolve
PITTNullCode/NullExact
PITTNullCode/NullGrid
PITTNullCode/NullInterp
PITTNullCode/NullNews
PITTNullCode/NullPsiInt
PITTNullCode/NullSHRExtract
PITTNullCode/NullVars
PITTNullCode/SphericalHarmonicDecomp
PITTNullCode/SphericalHarmonicRecon
```

4. Run GetComponents on the modified thornlist.

```
./GetComponents -a einsteintoolkit.th
```

Where to find documentation

- ThornGuide for PITTNullCode/SphericalHarmonicDecomp
- ThornGuide for PITTNullCode/SphericalHarmonicRecon
- PITTNullCode/SphericalHarmonicDecomp/par/qc0-mclachlan-CCE_Cauchy.par
- SphericalHarmonicRecon/par/qc0-mclachlan-CCE_Null.par

Overview

The current CCE implementation uses a Spherical harmonic decomposition of the metric components in a 3+1 ADM decomposition (the spatial metric γ_{ij}, the shift β^i, and the lapse α) and their radial derivatives on spheres of fixed coordinate radius R. A CCE waveform extraction would have the following steps:

1. During the main Cauchy evolution the metric is interpolated onto spherical shells and decomposed using spherical harmonics and Chebyshev polynomials. The data is saved for later use.
2. The data may be modified, for example, to filter high frequencies.
3. The data is read into a Characteristic evolution, produces a waveform on $\mathscr{I}^+$. Multiple evolution may be needed for a Richardson extrapolation if very small error tolerances are required.

Overview of Using CCE with the Einstein Toolkit

1. Compile your Cactus/Carpet configuration as usual, but add the thorn `SphericalHarmonicDecomp`.
2. Evolve BBH configuration using your standard BBH parameter files, but add parameters for `SphericalHarmonicDecomp` that will output the Cauchy metric for later post-processing by the `Pitt Null code`.
3. Build a separate configuration with the `Pitt Null code`. The `Pitt Null code` requires the `PUGH` driver and is not compatible with `Carpet`.
4. Run a `Pitt Null code` configuration that will use the saved Cauchy metric data to produce waveforms at $\mathscr{I}^+$.

Outputting the Cauchy metric data

There are at least two thorns that provide this functionality: WorldTube and SphericalHarmonicDecomp (discussed here).

- SphericalHarmonicDecomp interpolates the metric onto multiple spherical shells and attempts to fits these data to Chebyshev polynomial / spherical harmonic series. The number of data points is generally larger than the number of functions in the series. This was a design choice to allow for least-squares fitting to the best fit parameters (to help smooth the function).
- In this example we will extract the metric on 3 different shells. The idea here is to make the shell small enough that we can accurately calculate the radial derivatives of the metric function, while also large enough that we can smooth out the grid noise.

parfile section

```
### Your usual BBH parfile parameters  go here. The add ...
ActiveThorns="SphericalHarmonicDecomp"
SphericalHarmonicDecomp::extract_spacetime_metric_every=32
SphericalHarmonicDecomp::num_radii=3
SphericalHarmonicDecomp::EM_Rin[0]=18
SphericalHarmonicDecomp::EM_Rout[0]=22
SphericalHarmonicDecomp::EM_Rin[1]=47
SphericalHarmonicDecomp::EM_Rout[1]=53
SphericalHarmonicDecomp::EM_Rin[2]=94
SphericalHarmonicDecomp::EM_Rout[2]=106
SphericalHarmonicDecomp::num_l_modes=7
SphericalHarmonicDecomp::num_n_modes=7
SphericalHarmonicDecomp::num_mu_points=41
SphericalHarmonicDecomp::num_phi_points=82
SphericalHarmonicDecomp::num_x_points=28
```

Notes about good values for parameters

- extract_spacetime_metric_every: Large enough to resolve the wavelength of the highest mode.
- EM_Rin[i] and EM_Rout[i]: Radii of extraction shell (multiple shells can be specified). Set EM_Rout[i] and EM_Rin[i] far enough apart so that high frequencies can be smoothed. Close enough that radial derivative is accurate.
- num_n_modes: Chebyshev polynomial used to obtain derivative. This depends on EM_Rin[i] and EM_Rout[i].
- num_l_modes: How many spherical harmonic ℓ modes? Note num_l_modes = l_max -1.
- Least-Square Fitting (grid) parameters...
  - num_mu_points: The number of points in the θ direction. Set to multiple of num_l_modes.
  - num_phi_points: set to twice num_mu_points.
  - num_x_points: How many points in radial direction. Set to a multiple of num_n_modes.

Pitt Null Code Cactus Config

Use the following ThornList as a starting point. Note that the `Pitt Null code` requires the `PUGH` driver and is not compatible with `Carpet`.

```
AEIThorns/AEILocalInterp            # AEILocalInterp ( ) [ ] { }
CactusBase/Boundary                 # boundary ( ) [ ] { }
CactusBase/CartGrid3D               # grid (coordbase) [ ] {driver}
CactusBase/CoordBase                # CoordBase ( ) [ ] { }
CactusBase/Fortran                  # Fortran ( ) [ ] { }
CactusBase/IOASCII                  # IOASCII ( ) [ ] {IO}
CactusBase/IOBasic                  # IOBasic (IO) [ ] {IO}
CactusBase/IOUtil                   # IO ( ) [ ] { }
CactusBase/InitBase                 # InitBase ( ) [ ] { }
CactusBase/LocalInterp              # LocalInterp ( ) [ ] { }
CactusBase/LocalReduce              # LocalReduce ( ) [ ] { }
CactusBase/SymBase                  # SymBase ( ) [ ] { }
CactusBase/Time                     # time ( ) [ ] {cactus}
CactusNumerical/MoL                 # MethodOfLines ( ) [ ] { }
CactusNumerical/SpaceMask           # SpaceMask (grid) [ ] { }
CactusPUGH/PUGH                     # Driver ( ) [ ] {cactus}
CactusPUGH/PUGHInterp               # Interp ( ) [ ] { }
CactusPUGH/PUGHReduce               # Reduce ( ) [ ] { }
CactusPUGH/PUGHSlab                 # Hyperslab ( ) [ ] { }
CactusPUGHIO/IOHDF5                 # IOHDF5 ( ) [ ] {IO}
CactusPUGHIO/IOHDF5Util             # IOHDF5Util (IO) [ ] {IO}
CactusUtils/Formaline               # Formaline ( ) [ ] {IO}
CactusUtils/NaNChecker              # NaNChecker (Reduce) [ ] { }
CactusUtils/TimerReport             # timerreport ( ) [ ] {IO}
ExternalLibraries/BLAS              # BLAS ( ) [ ] { }
ExternalLibraries/GSL               # GSL ( ) [ ] { }
ExternalLibraries/HDF5              # HDF5 ( ) [ ] { }
ExternalLibraries/LAPACK            # LAPACK ( ) [ ] { }
ExternalLibraries/libjpeg           # libjpeg ( ) [ ] { }
ExternalLibraries/zlib              # zlib ( ) [ ] { }
ExternalLibraries/FFTW3              # zlib ( ) [ ] { }
```

ThornList cont

```
PITTNullCode/NullConstr       # NullConstr (NullGrid,NullVars,NullInterp) [ ] {NullEvolve,Nul
PITTNullCode/NullDecomp       # NullDecomp (NullGrid) [ ] {NullGrid,IO}
PITTNullCode/NullEvolve       # NullEvolve (NullInterp,NullGrid,NullVars,Time) [NullSHRExtrac
PITTNullCode/NullExact        # NullExact (NullVars,NullGrid,NullNews,NullConstr,NullInterp,Nu
PITTNullCode/NullGrid         # NullGrid ( ) [ ] { }
PITTNullCode/NullInterp       # NullInterp (NullGrid) [ ] {NullGrid}
PITTNullCode/NullNews         # NullNews (NullGrid,NullVars,NullInterp) [ ] {NullGrid,cactus}
PITTNullCode/NullPsiInt       # NullPsiInt (NullVars,NullGrid,NullNews,NullInterp,NullEvolve)
PITTNullCode/NullSHRExtract   # NullSHRExtract (NullInterp,NullGrid,NullVars) [] {NullGrid,cac
PITTNullCode/NullVars         # NullVars ( ) [ ] {NullGrid}
PITTNullCode/SphericalHarmonicRecon # SphericalHarmonicRecon ( ) [ ] { }
```

Compiling utilities

The thorn `SphericalHarmonicRecon` provides several utilities that may be necessary to in order to process the data the obtained during the initial Cauchy evolution. Copy the above ThornList to CCEThornList and create a new configuration.

```
[yosef@quasar Cactus]$ gmake cce
[yosef@quasar Cactus]$ cp CCEThornList configs/cce/ThornList
[yosef@quasar Cactus]$ gmake cce
[yosef@quasar Cactus]$ gmake cce-utils
```

The utilities will now be in `Cactus/exe/cce/`.

Modify the Cauchy data

- Merge the hdf5 files (for a given radial shell) from all runs in the simulation into a single hdf5 file using `hdf5_merge`.
- Find the last "iteration" (dump number) using the command `findlast` [`./findlast file.h5`]. Output is iteration number and time.

```
./findlast  metric_obs_0_Decomp.h5
8086: 4.010913e+02
```

- Examine the data. To obtain ascii output use the `ascii_output` utility [`ascii_output nlev n l m comp file.hdf5`]. 'comp' refers to the component in the metric (0=gxx, 1=gxy,..., 6=betax,..., 9=alp). In this example, we output the ($\ell = 4$, $m = 4$) mode (for $n = 1$) of gxx.

```
  ./ascii_output 8086 1 4 4 0 metric_obs_0_Decomp.h5 > gxx.asc
```

- Determine the highest frequency of the actual signal. For equal mass BBHs it's about 0.5 (ω).
- Filter the data (not required) using the `fftwfilter` program [`./fftwfilter nlast omegamax kdamp file.h5`]. Choose omegamax larger than the physical maximum frequency. Check the filtered data to make sure the true signal is not suppressed and no significant Gibbs oscillations are introduced.

```
./fftwfilter 8086 3 1 metric_obs_0_Decomp.h5
  ... there will be a lot of output to the screen
  ... a new data file metric_obs_0_Decomp_ft.h5 will be created
  ... that contains the filtered metric. It will also contain the
  ... time derivative of the metric (obtained using the Fourier transform)
```

- Note omegamax is in units of $1/M$, while kdamp is in units of integer (fftw) frequency. Larger kdamp leads to smoother windowing but can also contaminate the true signal.
- Check the filtered data using `ascii_output`.

Characteristic evolution

- Determine the appropriate world tube radius (avg. of Rin and Rout). Rin and Rout can be determined using the `readmeta` utility.

```
./readmeta metric_obs_0_Decomp.h5
Run Parameters
... nn = 7
... na = 49
... Rin = 1.800000e+01
... Rout = 2.200000e+01
```

- Make sure that all thorns use the correct radius.
  - `SphericalHarmonicRecon::r_extract=20.0`
  - `NullSHRExtract::cr = 20.0`
  - `NullGrid::null_rwt = 20.0`
- Additional parameters:
  - `SphericalHarmonicRecon::time_derivative_in_file = "yes"` # set this only if you filtered the data
  - `SphericalHarmonicRecon::metric_data_filename = "metric_obs_0_Decomp_ft.h5"`

Characteristic ParFile

```
ActiveThorns = "Fortran MoL SymBase CoordBase CartGrid3D Time"
ActiveThorns = "IOASCII IOBasic IOUtil"
ActiveThorns = "Boundary LocalReduce  SpaceMask"
ActiveThorns = "Pugh PughReduce PughSlab PUGHInterp"
ActiveThorns = "AEILocalInterp NullGrid NullInterp NullVars"
ActiveThorns = "NullEvolve NullDecomp NullSHRExtract SphericalHarmonicRecon"
ActiveThorns = "NullNews NullConstr"
ActiveThorns = "LocalInterp  HDF5 IOHDF5 IOHDF5Util"
NullEvolve::boundary_data = "SHRE"
NullEvolve::initial_J_data = "smooth_J"
NullEvolve::first_order_scheme = "yes"

##Make sure the following three are consistent with the actual
## extraction radius (average of R_out and R_in).
SphericalHarmonicRecon::r_extract=20.0
NullSHRExtract::cr = 20.0
NullGrid::null_rwt = 20.0
SphericalHarmonicRecon::time_derivative_in_file = "yes"
# set the filename appropriately
SphericalHarmonicRecon::metric_data_filename = "metric_obs_0_Decomp_ft.h5"
```

Characteristic parfile cont.

```
NullSHRExtract::mass = 1
NullSHRExtract::WT_metric = "Full"
NullSHRExtract::l_max = 6 # should match num_l_modes -1
NullGrid::null_xin = 0.45
NullGrid::N_ang_ghost_pts = 3
NullGrid::N_ang_stencil_size = 3

##The following four are Null Grid Resolution parameters ....
Time::timestep = 0.05 #Set to a multiple of the dump time for reduced errors
NullGrid::N_ang_ev_outside_eq = 20
NullGrid::N_ang_pts_inside_eq = 101
NullGrid::N_radial_pts = 113
NullInterp::interpolation_order = 4
NullInterp::stereo_patch_type = "circle"
NullInterp::deriv_accuracy = 4
```

Characteristic parfile cont.

```
NullEvolve::dissip_J = 0.0001
NullEvolve::dissip_Jx = 0.001
NullEvolve::dissip_mask_type = "zero at rD0, one at rD1"  #0r01r1
NullEvolve::N_dissip_zero_outside_eq = 11
NullEvolve::N_dissip_one_outside_eq = 10
NullDecomp::use_rsYlm = no
NullDecomp::l_max = 4
NullSHRExtract::WT_spherical_harmonics = yes
NullNews::write_spherical_harmonics = yes
NullNews::interp_to_constant_uBondi = yes
NullNews::news_interp_order = 4
NullNews::mask_Psi4 = yes
NullNews::mask_NewsB = yes
```

Obtaining Highly-Accurate Waveforms

- The CCE waveform will be first-order convergent with a small first-order error.
- Optional: Repeat the Characteristic evolution with higher resolutions and Richardson extrapolate to obtain a highly-accurate waveform.
- Instructions for increasing resolution:

Example for a resolution sequence

- Halve `Time::timestep`.
- Double `NullGrid::N_ang_ev_outside_eq`.
- Double `NullGrid::N_ang_pts_inside_eq` and subtract by 1.
- Double `NullGrid::N_radial_pts` and subtract by 1.

```
Time::timestep = 0.125
NullGrid::N_ang_ev_outside_eq = 10
NullGrid::N_ang_pts_inside_eq = 51
NullGrid::N_radial_pts = 61

Time::timestep = 0.0625
NullGrid::N_ang_ev_outside_eq = 20
NullGrid::N_ang_pts_inside_eq = 101
NullGrid::N_radial_pts = 121

Time::timestep = 0.03125
NullGrid::N_ang_ev_outside_eq = 40
NullGrid::N_ang_pts_inside_eq = 201
NullGrid::N_radial_pts = 241
```