# SpECTRE CCE tutorial – ICERM, September 2020

Jordan Moxon, on behalf of the SpECTRE team and SXS
*moxon@black-holes.org*

## I. INTRODUCTION

This is a quick description on how to get up and running with the stand-alone version of the SpECTRE CCE module for working with worldtube data produced by SpEC or other Cauchy evolution system. This is a development code base, and we are always implementing new features or improving the reliability of the system. Because SpECTRE is an open-source code base and the wave extraction system has been sufficiently well verified to trust for SXS projects, we felt it is time to start helping others use our new wave extraction method. As you find errors, bugs, missing documentation, or any other problems or questions, we encourage you to file issues for suggested improvements, or contact the SpECTRE development team.

### A. Known issues

We've done our best to make the CCE system as precise and robust as possible given its current development status; that said, we don't want any of the big problems we already know about to be a surprise for new users. Issues will also be documented on the SpECTRE github issue tracker: https://github.com/sxs-collaboration/spectre/issues.

**The initial data problem:** This is probably the most dire known issue for the waveform data quality produced by CCE. The problem arises because the metric information on the initial-data hypersurface is difficult to prepare in close approximation to the state produced by a true inspiral. This problem ensures that almost all extractions suffer from initial-data transients during the first couple of hundred $M$ in time (for an extraction radius $R = 100M$). Further, the initial data transient also tends to create an offset in the output strain data. For instance, the $(l, m) = (2, 2)$ mode often oscillates about a nonzero value – to the best of our knowledge, after the initial-data transient, the strain is not physically incorrect, but represents the strain in an unusual BMS gauge.

If strain data that oscillates about zero is important for your application, the easiest solution is to either integrate the News (likely backward from the final state), or manually subtract post-merger residual strain value. For more details on the initial-data transient problem and some workarounds, the recent paper led by Keefe Mitman describes some useful methods [1] that were valuable in the investigation of gravitational-wave memory effects.

**Threadsafe HDF5:** We currently require SpECTRE CCE to be built with threadsafe HDF5, as it internally permits the simultaneous read from the input file and write to the output file, and the HDF5 library does not guarantee such simultaneous operations, even when the operations occur for separate files. Without threadsafe HDF5, the CCE evolution will likely eventually segfault when the file operations become simultaneous, possibly some time into the evolution.

**Misordering of timeseries output:** SpECTRE CCE (like most of SpECTRE) is based on task-based parallelism, so the order in which data is written is not guaranteed. This ends up meaning that times in the output file can become transposed, and we have not yet implemented the post-processing routines to put them back in order. See the below Section V for a suggested python snippet to include in your data processing routines to avoid errors when processing possibly misordered times.

## II. PREPARING WORLDTUBE DATA

**Note:** This section represents the current state of compatible input with SpECTRE CCE, but we are willing to work with interested parties to generalize or relax requirements to make the code more amenable to a wider variety of use-cases.

The primary input worldtube format for the standalone mode of SpECTRE CCE is metric data specified in an HDF5 file. The SpECTRE CCE system requires the spatial metric $g_{ij}$, the lapse $\alpha$, and the shift $\beta^i$, as well as their radial and time derivatives. The components of the shift vector and spatial metric must be specified in a set of Kerr-Schild Cartesian-like coordinates, and the extraction sphere on which the metric data is specified must be a sphere of constant radius in that coordinate system.

The worldtube HDF5 file must have the following data ('.dat') entries:

| | | | | | |
|---|---|---|---|---|---|
| /gxx.dat | /Drgxx.dat | /Dtgxx.dat | /Shiftx.dat | /DrShiftx.dat | /DtShiftx.dat |
| /gxy.dat | /Drgxy.dat | /Dtgxy.dat | /Shifty.dat | /DrShifty.dat | /DtShifty.dat |
| /gxz.dat | /Drgxz.dat | /Dtgxz.dat | /Shiftz.dat | /DrShiftz.dat | /DtShiftz.dat |
| /gyy.dat | /Drgyy.dat | /Dtgyy.dat | /Lapse.dat | /DrLapse.dat | /DtLapse.dat |
| /gyz.dat | /Drgyz.dat | /Dtgyz.dat | | | |
| /gzz.dat | /Drgzz.dat | /Dtgzz.dat | | | |

For each of these datasets, the time-series data is represented as one row per time value, with the corresponding function values on that row represented by spherical harmonic coefficients. The data should be stored in double-precision values (floats will lose important precision), and must take the order of the time value, followed by the real and imaginary modes in $m$-varies-fastest order; $l$ ascending and $m$ descending. Explicitly, the row legend is:

```
time, Re(0,0), Im(0,0), Re(1,1), Im(1,1), Re(1,0), Im(1,0), Re(1,-1), Im(1,-1),
 Re(2,2), Im(2,2), Re(2,1), Im(2,1), Re(2,0), Im(2,0), Re(2,-1), Im(2,-1), Re(2,-2), Im(2,-2) ...
```

The worldtube data must be constructed as spheres of constant coordinate radius, and (for the time being) written to a filename of the format `...CceRXXXX.h5`, where the `XXXX` is to be replaced by the integer for which the extraction radius is equal to XXXXM. For instance, a 100M extraction should have filename `...CceR0100.h5`. The `...` indicates that the filename the extraction radius may be arbitrarily prefixed. This scheme of labeling files with the extraction radius is constructed for compatibility with SpEC worldtube data. We'll work to relax this constraint in the future, as variables specified by filename is not a desirable design choice long-term.

For performance, we also recommend that the HDF5 chunking for the input file be specified such that only a comparatively small ($< 256$) number of rows occupy the same chunk. If the file is chunked such that e.g. the entire time-series for each mode (column) shares a chunk, the CCE I/O performance will suffer for long runs.

## III. COMPILING SPECTRE CCE

For full documentation of SpECTRE dependencies and how to establish a native build environment, see the SpECTRE documentation https://spectre-code.org/installation.html.

For this guide, I will focus on the workflow in which a SpECTRE environment is obtained via the docker container. If you're building on an HPC system and it has a singularity module, we encourage using that with the SpECTRE build container (See https://sylabs.io/docs/). Note that the standard SpECTRE build container will not be suitable for running on multiple nodes in an HPC environment, as it will not know anything about the interconnects between nodes – a custom container for each supercomputing system will be necessary for multi-node runs under our current setup. However, CCE runs comfortably on a single node (it only parallelizes to 3 cores, but runs very quickly).

Please follow the instructions at https://spectre-code.org/installation.html to obtain the SpECTRE build container and start an instance, using the docker `-v` argument to mount a portion of your local file system where you want to work on SpECTRE into the container (see the docker documentation on the argument https://docs.docker.com/engine/reference/commandline/run/). You should clone the SpECTRE repository (sxs-collaboration/SpECTRE) into that mounted location so that it can be built inside the container.

Unfortunately, for now SpECTRE CCE requires a thread-safe version of the HDF5 library, and the HDF5 library supplied by the Ubuntu distribution on which the container is constructed is not built with the thread-safe options. So, we need to go through some small amount of work to prepare a thread-safe HDF5 version. Here, I'll step through a method of building the thread-safe version of HDF5 in a directory mounted in the docker container for use in the SpECTRE build. Once it's built in the mounted directory, it will similarly be available on later instances of the docker container provided the directory mounting remains the same.

In the appropriate mounted directory where you'd like to place the HDF5 installation, perform the following steps (replacing `[/path/to/mounted/hdf5/destination]` with the appropriate directory associated with the mount argument you provided to the docker instance):

```
wget https://support.hdfgroup.org/ftp/HDF5/current/src/hdf5-1.10.5.tar.gz
tar -xvzf hdf5-1.10.5.tar.gz
cd hdf5-1.10.5
./configure --enable-threadsafe --disable-hl --prefix=[/path/to/mounted/hdf5/destination]
make -j4
make install
```

At this point, you should now have a threadsafe HDF5 version built in the container path `/path/to/mounted/hdf5/destination`.

Now, navigate to the working directory in which you want to build the SpECTRE CCE executable. Once there, you can use the cmake command (again replacing the paths in `[]` with the appropriate paths for your mounted filesystem)

```
cmake -D CHARM_ROOT=/work/charm_6_10_2/multicore-linux-x86_64-clang/ \
 -D CMAKE_CXX_COMPILER=/usr/bin/clang++-10 \
 -D CMAKE_Fortran_COMPILER=/usr/bin/gfortran-7 \
 -D HDF5_ROOT=[/path/to/mounted/hdf5/destination] \
 -D CMAKE_BUILD_TYPE=Release \
 [/path/to/spectre/root]
make -j4 CharacteristicExtract
```

Provided the build succeeds, you should find yourself with the `CharacteristicExtract` executable in the `bin` subdirectory.

## IV. PERFORMING A TEST RUN

We have made available an example worldtube file so that new users can test their build of the SpECTRE characteristic evolution system. This can be downloaded from: https://zenodo.org/record/4033408#.X2KAa5plBhE

In a new directory for the run within the directories mounted in the docker container, copy over the example input file for `CharacteristicExtract`:

```
cp [/path/to/spectre/source]/tests/InputFiles/Cce/CharacteristicExtract.yaml [/path/to/run/directory/]
```

And use your favorite terminal-based text editor to open the yaml input file.

There's a bunch of options in there, but mostly you should only need to modify a handful of the parameters:

1. `BoundaryDataFilename` needs to be set to the filename of the worldtube data from which you want to extract the waveforms.

2. `LMax` should be set to the angular resolution you'd like for the evolution system. A good rule of thumb is that `LMax` of 12 will give a decent waveform and a quick evolution; an `LMax` of 16 will give generally good results that will often reach a precision level beyond the Cauchy simulation from which it is extracting (and so the waveform cannot be further improved). We have generally not seen much further convergence of of numerical residuals beyond `LMax` of 24.
   *NOTE:* be sure to update the `FilterLMax` option to around 2 less than the `LMax` option, or the simulation will just be filtering out all the extra resolution.

3. `EndTime` - this represents the final time you want to evolve to. If this option is omitted, the duration of the extraction is inferred from the worldtube file, and the wave is extracted from the full duration of the input. For most uses, it is best to simply omit the `EndTime` option.

4. `TargetStepSize` sets the step size used in the CCE system. We do not yet have adaptive time-stepping implemented for our evolution systems, so this needs to be manually specified. Typically, if the worldtube is at $200M$ or greater, a step size of `1.0` should work nicely. At around $100M$, the step size should be decreased to around `0.5` as the feature size on the domain will be smaller, and smaller yet for tighter extraction radii.

5. `ScriOutputDensity` sets the number of extra points between timesteps to write to the output waveforms. This is useful if analysis of the waveforms needs to take finite-difference derivatives, but most use-cases can safely set this option to 1 and save on disk space for the output waveforms.

6. `VolumeFileName` - in the SpECTRE systems, the waveform data is considered 'volume' data (as opposed to 'reduction' data), so the final waveforms will be written to an hdf5 file with filename specified by this option.

The options also have some descriptions that can be found at the SpECTRE documentation: https://www.spectre-code.org.

Once the input options are chosen, you can start a SpECTRE run in the container via

```
[/path/to/spectre/build]/bin/CharacteristicExtract +p4 --input-file ./CharacteristicExtract.yaml
```

And this will extract the waveform. For the supplied test run, depending on your system processor, and for `LMax` of 12 you should expect it to complete in around 10 minutes. See Fig. 1 for plots of the $l = 2, 3$ modes from the example run at `LMax` of 12.

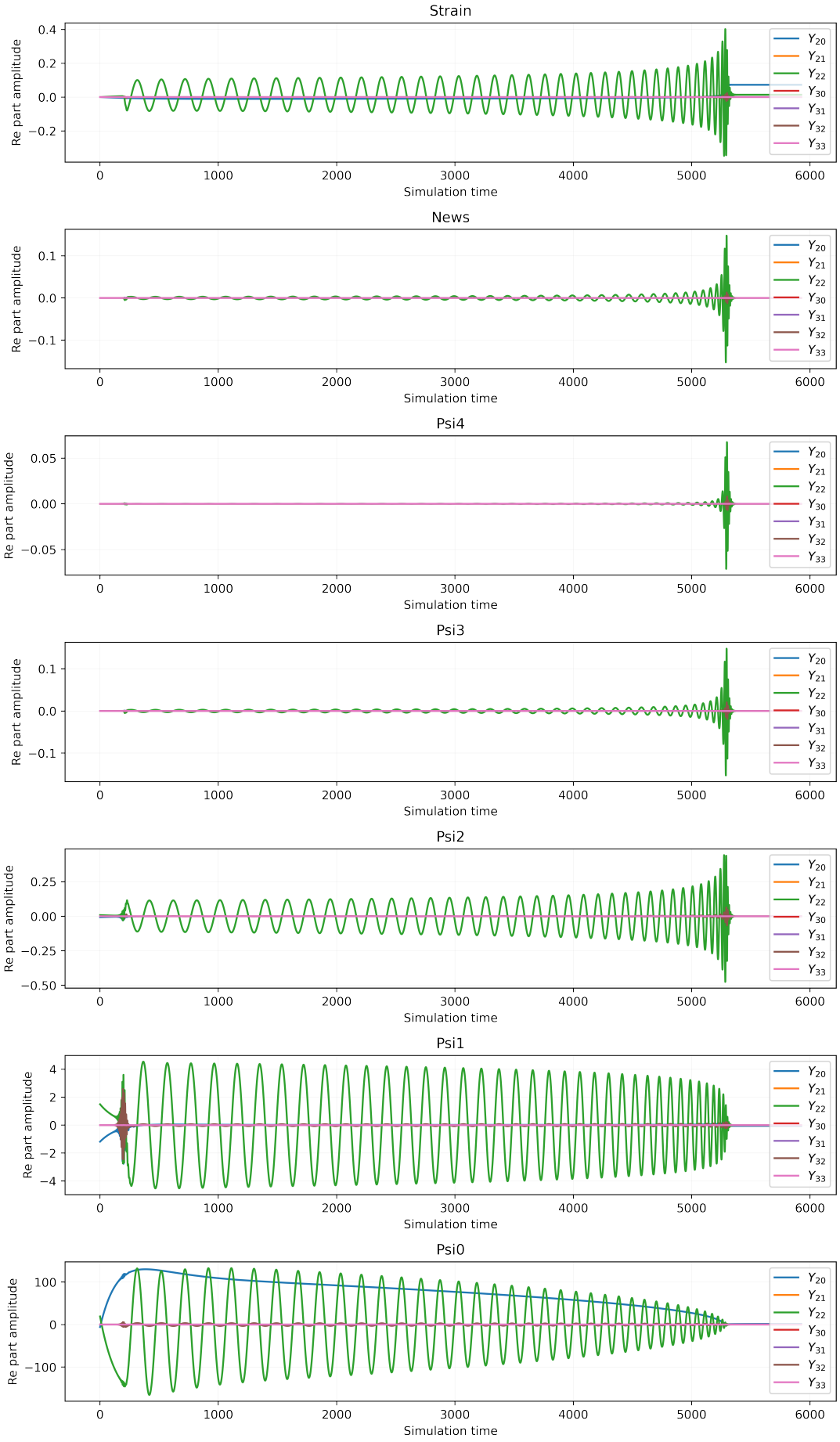


FIG. 1. Plots of each of the waveform outputs from the example worldtube file, at `LMax` =12.

## V. WAVEFORM OUTPUT FORMAT

The HDF5 format for the waveform outputs is very similar to the format used for the waveform inputs, with a couple of notable alterations. The data fields that are written by SpECTRE CCE are:

```
News.dat
Psi0.dat
Psi1.dat
Psi2.dat
Psi3.dat
Psi4.dat
Strain.dat
```

The data is represented as spin-weighted spherical harmonic modes, where the spin-weights for the quantities are: News(-2), Psi0(2), Psi1(1), Psi2(0), Psi3(-1), Psi4(-2), and Strain(-2).

The output is again produced as one time-step per row, with the first entry of each row being the asymptotic inertial time of the waveform, followed by the real and imaginary parts of the spin-weighted spherical harmonic modes in $m$-varies-fastest format (both $l$ and $m$ ascending). To be explicit, the legend for one of these data fields reads:

```
time, Real Y_0,0, Imag Y_0,0, Real Y_1,-1, Imag Y_1,-1, Real Y_1,0, Imag Y_1,0, Real Y_1,1, Imag Y_1,1,
Real Y_2,-2, Imag Y_2,-2, Real Y_2,-1, Imag Y_2,-1, Real Y_2,0, Imag Y_2,0, Real Y_2,1, Imag Y_2,1,
Real Y_2,2, Imag Y_2,2 ...
```

**WARNING:** The SpECTRE CCE output is not (yet) guaranteed to be placed in the output H5 in time-order. This is because of our reliance on loosely ordered task-based parallelism utilities. To avoid errors when analyzing the data arising from surprising time ordering, we suggest that you perform a sort of the time-series rows. This can be done easily, for instance, in python with the following snippet:

```
import h5py as h5
import numpy as np

#... your script setup

with h5.File(spectre_filename, 'r') as in_h5:
    for dset in in_h5:
        data_array = in_h5[dset][()]
        # sort the array according to the time
        data_array = data_array[data_array[:, 0].argsort()]
#... do something with the data from `data_array`
```

## VI. REDUCED CCE WORLDTUBE DATA TYPE AND CONVERSION UTILITY

The metric data format contains a great deal of redundant information and data not needed to perform the CCE evolution, and that inefficiency can be problematic if users want to store the input worldtube data to extract again with CCE in the future (for instance, with different input file settings or as improved initial data routines become available). In SXS we've taken the first step to solving this data efficiency problem by reprocessing the worldtube data to store Bondi quantities, and to omit the redundant negative-$m$ modes for real quantities.

The executable for producing the reduced format can be compiled similarly to the standalone `CharacteristicExtract` binary:

```
make -j4 ReduceCceWorldtube
```

The input options can be found via

```
ReduceCceWorldtube --help
```

The `--lmax_factor` is 2 by default, which indicates a higher resolution for the intermediate computation steps than the worldtube file, to defeat aliasing problems. We have found empirically that 2 is sufficient to get to the truncation error of most simulations, and that 3 or 4 is sufficient to reach numerical roundoff (i.e. ensuring the conversion process is truly lossless).

The reduced worldtube input data may be used the the CCE evolution input file identically to the original worldtube file with the exception that you must include an additional boolean flag indicating the format change:

```
#...
Cce:
# ...
  H5IsBondiData: true
# ...
```

The file format is the same as the spin-weighted output files, with the exception that the $-m$ modes for the real spin-0 quantities are omitted, as well as the imaginary part of the $m = 0$ modes.

## VII. REFERENCE MATERIALS FOR SPECTRE AND CCE

There are a number of papers on the formal development of the mathematics of the CCE system [2, 3], including our recent paper on the formalism refinements that we use specifically for SpECTRE [4]. For more complete technical details about the SpECTRE CCE implementation, see the SpECTRE documentation page https://spectre-code.org/namespaceCce.html#details. There are also several publications regarding the implementations PittNull [5–8] and the SpEC CCE system [9–12]. For information about SpECTRE generally, there are a couple publications describing the computational systems we're developing [13]; and [1] uses SpECTRE CCE for investigations of gravitational-wave memory.


[1] K. Mitman, J. Moxon, M. A. Scheel, S. A. Teukolsky, N. Deppe, L. E. Kidder, and W. Throwe, (2020), arXiv:2007.11562 [gr-qc].
[2] N. T. Bishop, R. Gomez, L. Lehner, and J. Winicour, Phys. Rev. **D54**, 6153 (1996), arXiv:gr-qc/9705033 [gr-qc].
[3] N. T. Bishop, R. Gomez, L. Lehner, M. Maharaj, and J. Winicour, Phys. Rev. **D56**, 6298 (1997), arXiv:gr-qc/9708065 [gr-qc].
[4] J. Moxon, M. A. Scheel, and S. A. Teukolsky, Phys.Rev.D **102**, 044052 (2020), arXiv:2007.01339 [gr-qc].
[5] M. Babiuc, B. Szilagyi, J. Winicour, and Y. Zlochower, Phys.Rev.D **84**, 044057 (2011), arXiv:1011.4223 [gr-qc].
[6] N. T. Bishop, R. Gomez, L. Lehner, B. Szilagyi, J. Winicour, and R. A. Isaacson, in *Black Holes, Gravitational Radiation and the Universe: Essays in Honor of C.V. Vishveshwara*, edited by B. R. Iyer and B. Bhawal (1998) pp. 383–408, arXiv:gr-qc/9801070 [gr-qc].
[7] C. Reisswig, N. Bishop, D. Pollney, and B. Szilagyi, Class.Quant.Grav. **27**, 075014 (2010), arXiv:0912.1285 [gr-qc].
[8] C. Reisswig, N. Bishop, D. Pollney, and B. Szilagyi, Phys.Rev.Lett. **103**, 221101 (2009), arXiv:0907.2637 [gr-qc].
[9] C. J. Handmer and B. Szilagyi, Class. Quant. Grav. **32**, 025008 (2015), arXiv:1406.7029 [gr-qc].
[10] C. J. Handmer, B. Szilágyi, and J. Winicour, Class. Quant. Grav. **32**, 235018 (2015), arXiv:1502.06987 [gr-qc].
[11] C. J. Handmer, B. Szilágyi, and J. Winicour, Class. Quant. Grav. **33**, 225007 (2016), arXiv:1605.04332 [gr-qc].
[12] K. Barkett, J. Moxon, and B. Scheel, Mark A.and Szilágyi, arXiv (2019), arXiv:1910.09677 [gr-qc].
[13] L. E. Kidder *et al.*, J. Comput. Phys. **335**, 84 (2017), arXiv:1609.00098 [astro-ph.HE].