GremlinEq:
Solving the Teukolsky equation with a point-particle source
Version specialized to circular and equatorial orbits

Scott A. Hughes*
Department of Physics
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

February 2, 2019

Abstract

Gremlin[1] is a toolkit for studying solutions to the Teukolsky equation using a point-particle source. GremlinEq is a version of Gremlin that specializes the source to circular and equatorial orbits of Kerr black holes. The full Gremlin package will be released publicly via the Black Hole Perturbation Toolkit; certain proprietary libraries that were used in its development must be cleaned up and replaced with Open Source resources. GremlinEq has been so cleaned, and is hereby provided as an initial release. This document will evolve into a full manual. Presently, it provides directions for installing the package, a synopsis of its contents, and a guide to its use.

1 Installing

Since you are reading this document, I presume you have successfully downloaded it and installed the source files on your computer. Installation should be straightforward:

1. Install libraries. GremlinEq uses the following packages: The GNU Scientific Library (GSL) [1]; the GNU Multiple Precision Library (GMP) [2]; the Fastest Fourier Transform in the West (FFTW) [3]; and the Hierarchical Data Format, version 5 (HDF5) [4]. All of these packages can be easily installed on Apple systems using Homebrew [5]; I would appreciate feedback from users about installation on other unix-type systems. (I offer no advice regarding installation on Windows-based systems, for which I have no relevant expertise at all.)

   The code is known to compile with the following versions of these libraries: GSL version 2.5; GMP version 6.1.2; FFTW version 3.3.8; and HDF5 version 1.10.4. As I type this sentence, these are the most up-to-date version provided by Homebrew.

*sahughes@mit.edu

[1]This package has had multiple names over the years. When the code was initially being put together (c. 1998), it was called "Pebble," to evoke a small body orbiting a black hole. When I began sharing version with people, I started calling it "Gremlin," which was intended to more-or-less stand for Gravitational Radiation in the Extreme Mass-ratio LIMit (and then magically changing that "M" to an "N"). When I changed the core integrator to one based on the Mano-Suzuki-Takasugi (MST) method, I renamed it "Spectre," as the MST method is essentially a spectral method for solving Teukolsky. In the meantime, the numerical relativity community has taken ownership of code with names that sound like "spec," so it was decided to go back to the name Gremlin.

2. Edit the Makefile. Depending on the version of "make" on your system, the variable CURDIR may not be defined. If that is the case, change the variable TOP to point to whichever directory on your system holds GremlinEq. You may also need to teach the Makefile about the location of the different libraries and include files associated with those libraries. Homebrew tends to put relevant files into /usr/local/include and /usr/local/lib; if that is the case for you, then the present variables should be sufficient. Otherwise, you may need define additional variables defining the location of header files and libraries, and modify the linking steps in the Makefile appropriately.

3. "make". If all goes well, the output of this should be a set of libraries in the "lib" directory, and a set of executables in "bin". If anything fails, you may need to discuss with Hughes (at least until there has been sufficient testing to document a set of failure modes).

2 GremlinEq's contents

Gremlin and GremlinEq are at heart a set of libraries which define C++ classes that can be used to study the Teukolsky equation's solutions. The executables are essentially wrapper codes for these libraries. In this section, I very briefly describe the key classes. *This is quite minimal at present.* For now, I presume that the user is sufficiently expert that they can read the code and deduce how to use the different classes once their purpose is defined.

2.1 Class SWSH

SWSH defines methods for computing spin-weighted spheroidal harmonics. The class prototype is in SWSH.h (all header files are in the "include" directory), and the code lives in src/swsh. SWSH uses class Clebsch, which defines methods for computing Clebsch-Gordan coefficients; Clebsch is also prototyped in SWSH.h, and has code in src/swsh.

2.2 Class FT

FT defines methods for computing solutions to the homogeneous Teukolsky equation as laid out in papers by Ryuichi Fujita and Hideyushi Tagoshi. See FT.h and src/fujtag.

2.3 Class CEKG

CEKG defines methods for computing various quantities related to circular, equatorial Kerr geodesics. See CEKG.h, and code in src/circeq.

2.4 Class CEKR

CEKR describes methods for computing various quantities to radiation from circular, equatorial Kerr geodesics. It relies on input from instances of the classes SWSH, FT, and CEKG. See CEKR.h and code in src/circeq.

2.5 Class CETD

CETD stands for circular, equatorial Teukolsky driver, and is used to "drive" studies that look at a many circular equatorial orbits. It is essentially a holder for methods that solve CEKR repeatedly, as well as to . See CETD.h and code in src/circeq.

This class is also used to write the HDF5 data files (the "d" stands for data as well as driver). See src/circeq/CETD.cc for detailed discussion. The synopsis is that the HDF5 file contains 2 groups: the

modes (i.e., the indices ℓ and m) and the parameters (orbital radius r, spin parameter a, energy E, axial angular momentum L_z, and axial frequency Ω_ϕ). Into the modes group goes 19 different bits of data:

- The spin-weighted spheroidal harmonic eigenvalue λ
- The real and the imaginary parts of Z^∞ (the amplitude of the "to infinity" Teukolsky amplitude)
- The real and the imaginary parts of $Z^{\rm H}$ (the amplitude of the "down horizon" Teukolsky amplitude)
- The real and the imaginary parts of $R^{\rm in}$ (the ingoing separated radial part of the Teukolsky function, at the orbit)
- The real and the imaginary parts of $\partial_r R^{\rm in}$
- The real and the imaginary parts of $R^{\rm up}$ (the outgoing separated radial part of the Teukolsky function, at the orbit)
- The real and the imaginary parts of $\partial_r R^{\rm up}$
- 4 fluxes carried by the radiation, $\dot{E}^\infty$, $\dot{E}^{\rm H}$, $\dot{L}_z^\infty$, $\dot{L}_z^{\rm H}$
- The rate of change of orbital radius arising from radiation flux to infinity, $\dot{r}^\infty$
- The rate of change of orbital radius arising from the down-horizon radiation flux, $\dot{r}^{\rm H}$.

2.6 Class CEDR

CEDR stands for circular, equatorial data reader. This class collects methods which read the output of a data run (i.e., the HDF5 data produced by the code) and present it in human readable format. See CEDR.h and code in src/circeq.

2.7 Class CEID

CEID stands for circular, equatorial inspiral data. This class uses CEDR, and provides methods for smoothly interpolating through a large set of output from many orbits to study waveforms and inspirals. See CEID.h and code in src/circeq.

2.8 Class Kerr

This is a simple container class for a collection of useful simply static functions that pop up over and over again when studying Kerr black hole orbits. The class and all of its functions are provided in Globals.h.

2.9 Class GKG

This class likewise defines functions that are useful for generic Kerr geodesics. It is not used very much; its content may be subsumed into a different piece of the code in a future release. See GKG.h and code in src/utility.

2.10 Class RRGW

This class defines functions that are useful for computing gravitational waves and radiation reaction. Aspects of this class may fit better in another module, and may be moved elsewhere in a future release. See RRGW.h and code in src/utility.

2.11 Class TidalH

This class defines functions which are useful for analyzing how the on-horizon Teukolsky solution affects the geometry of a black hole. This methods were extensively used in work with former MIT student Stephen O'Sullivan, but may not be of broad interest; as such, they may be removed from general release. See TidalH.h and code in src/utility.

2.12 Class Tensors

This is a container class for memory allocation routines which define multi-index objects with arbitrary index range. They are inspired by Numerical Recipe's such routines [e.g., vector() and matrix()], but have been implemented in a new way and use C++ templates to make arrays of arbitrary type. All code is in Tensors.h.

3 Current executables

As mentioned above, the code's executables are essentially just wrappers for the different class libraries. In this section, I briefly describe the purpose of each method that you will find in the bin directory. Every one of these codes takes arguments; run the code without any arguments in order to get a brief message describing what the arguments are.

Note that several of these executables were written for a particular analysis or paper and may not be of broad interest. This list will be cleaned up (and likely reduced) for general release in the future. The somewhat random nature of what is available often reflects the fact that certain specific bits of data have been needed for projects over the years; many of these executables were written with a very specific purpose in mind and then never used again. For example, the "TidalH" and "Embed" executables were used extensively to produce Refs. [6, 7].

3.1 Circ_Eq

Circ_Eq solves the Teukolsky equation for a single (ℓ, m) mode of a single orbit. Produces an HDF5 datafile containing the data on that solution, and reports some summary information to stdout.

3.2 Circ_Eq_Seq

Solves the Teukolsky equation for all m modes up to some $\ell_{\rm max}$ over a range of radii.

3.3 Circ_Eq_Seq2

Solves the Teukolsky equation over a range of radii for all m modes, including enough ℓ modes to insure that the fractional convergence in flux is within a specified ϵ, but that at least $(\ell_{\rm max})_{\rm min}$ ℓ modes have been included. The data is evenly spaced in $v \equiv (M\Omega_\phi)^{1/3}$, where Ω_ϕ is the axial frequency of circular equatorial Kerr orbits.

3.4 Fluxes

Take data produced by one of the codes above and produce various fluxes (energy, angular momentum, breakdown of Teukolsky amplitudes) from a single orbit.

3.5 GW

Take data produced by one of the codes above and produce the gravitational waveform from a single orbit.

3.6 Circ_Eq_TotFlux_r

Produce summary data for the total flux of energy and angular momentum carried from an orbit, for data evenly spaced in orbital radius. Typically paired with Circ_Eq_Seq.

3.7 Circ_Eq_TotFlux_v

Produce summary data for the total flux of energy and angular momentum carried from an orbit, for data evenly spaced in $v = (M\Omega_\phi)^{1/3}$. Typically paired with Circ_Eq_Seq2.

3.8 Circ_Eq_Traj

Take data evenly spaced in radius and use the flux information to compute the adiabatic inspiral trajectory a small body would follow as it spirals into the larger black hole.

3.9 Circ_Eq_Wave

The same as Circ_Eq_Traj, but produce the accompanying gravitational waveform.

3.10 Circ_Eq_Clm

The same as Circ_Eq_Wave, but only output a single amplitude $C_{\ell m}$.

3.11 Circ_Eq_Clm2

The same as Circ_Eq_Clm, but use input data that is evenly spaced in v.

3.12 Circ_Eq_Ymode_v

The same as Circ_Eq_Clm2, but provide output decomposed onto a spin-weighted *spherical* harmonic basis, and provide flux information decomposed into spherical harmonics as well.

3.13 Circ_Eq_Smode_v

The same as Circ_Eq_Ymode_v, but provide decomposes into spin-weighted *spheroidal* harmonics. Overlaps substantially with Circ_Eq_Clm2.

3.14 Circ_Eq_lmode_v

Take a sequence of data, evenly spaced in v, and examine how the energy flux is distributed over different spheroidal harmonic ℓ modes.

3.15 Circ_Eq_mmode_v

Take a sequence of data, evenly spaced in v, and examine how the energy flux is distribute over different m modes.

3.16 TidalHEq

Examines the curvature of an event horizon distorted by an orbiting companion; focuses on the equatorial plane.

3.17 TidalHSurf

Examines the curvature of an event horizon distorted by an orbiting companion; surveys the entire horizon surface.

3.18 EmbedEq

Examines the embedding geometry of an event horizon distorted by an orbiting companion; focuses on the equatorial plane.

3.19 EmbedSurf

Examines the embedding geometry of an event horizon distorted by an orbiting companion; surveys the entire horizon surface.

Note that the executables TidalHEq, TidalHSurf, EmbedEq, and EmbedSurf require GSL version 2.5. These bits were used for one project and may not be of general interest. In the current release, the Makefile is set up such that "make all" will not make these executables; "make horizgeom" will make them.

Social contract

This code uses multiple packages and libraries that use the Free Software Foundation's open source "copyleft" license, the GNU Public License. As such, this code likewise will be released using the GPL. In this first not-quite-public release, I ask that users use the following "social contract" until it has been tested some more and is ready to be inflicted on the world:

1. If your use of this code leads to results that appear in a published article, please reference the code as follows:

   S. A. Hughes, computer code GREMLINEQ, available from the Black Hole Perturbation Toolkit (http://bhptoolkit.org)

2. If you modify or improve the code, please do not branch a new release. Instead, please submit your change, improvement, or bugfix to Hughes. Your change may then be incorporated into the main codebase and included in the main public Toolkit branch. Note that I cannot legally enforce this, hence I consider this to be more of a social contact while we gear up for a true public release.

3. If you find the code useful and like it, and you run into me (Hughes) at a conference, you agree to buy me 1 beer. I like ales, porters, and stouts.

Acknowledgments

Over the 20+ years that this package has been developed, this work has been partially supported by numerous grants from the National Science Foundation and NASA to the California Institute of Technology, the Kavli Institute for Theoretical Physics at the University of California at Santa Barbara, the Massachusetts Institute of Technology, and Cornell University, as well as various fellowships and internal support provided to Hughes and students at MIT. On a later release, I intend to catalog and enumerate all these different sources of support.

The development of this software benefitted greatly from interactions with many different people. Special mention for contributions in the earliest days goes to Daniel Kennefick (a particularly helpful and a valuable source of advice and wisdom in the development of this work, not to mention a source of comparison data that helped validate the core Teukolsky code), Sam Finn (who provided a great deal of advice and validation data), Saul Teukolsky (who originally suggested that the spin-weighted spheroidal harmonics could be spectrally decomposed using the spin-weighted spherical harmonics as basis functions), and Eric Poisson (who supervised my first project in black hole perturbation theory [in 1993!], guided me through the literature on Teukolsky-equation-based radiation reaction, and was wonderfully patient as I pestered him with oodles of naive and [undoubtedly] irritating questions).

Special mention for later work and contributions goes to Steve Drasco (whose work on the generic source composed a major part of his Ph.D. thesis), William Throwe (whose work implementing the MST method as described in papers by Fujita-Tagoshi formed his undergraduate thesis), Niels Warburton, and Maarten van de Meent.

4 REFERENCES

[1] https://www.gnu.org/software/gsl/

[2] https://gmplib.org/

[3] http://www.fftw.org/

[4] https://en.wikipedia.org/wiki/Hierarchical_Data_Format

[5] https://brew.sh/

[6] S. O'Sullivan and S. A. Hughes, Phys. Rev. D **90**, 124039 (2014); arXiv:1407.6983

[7] S. O'Sullivan and S. A. Hughes, Phys. Rev. D **94**, 044057 (2016); arXiv:1505.03809