Python Open source Waveform Extractor (POWER): an open source, Python package to monitor and post-process numerical relativity simulations

Daniel Johnson\textsuperscript{1,2,3,4,5}, E A Huerta\textsuperscript{1} and Roland Haas\textsuperscript{1}

\textsuperscript{1} NCSA, University of Illinois at Urbana-Champaign, Urbana, Illinois, 61801, United States of America
\textsuperscript{2} Department of Physics, University of Illinois at Urbana-Champaign, Urbana, Illinois, 61801, United States of America
\textsuperscript{3} Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, 61801, United States of America
\textsuperscript{4} Students Pushing Innovation (SPIN) undergraduate intern at NCSA, University of Illinois at Urbana-Champaign, Urbana, IL 61801, United States of America

E-mail: dsjohns2@illinois.edu

Received 9 August 2017, revised 20 November 2017
Accepted for publication 22 November 2017
Published 19 December 2017

Abstract

Numerical simulations of Einstein’s field equations provide unique insights into the physics of compact objects moving at relativistic speeds, and which are driven by strong gravitational interactions. Numerical relativity has played a key role to firmly establish gravitational wave astrophysics as a new field of research, and it is now paving the way to establish whether gravitational wave radiation emitted from compact binary mergers is accompanied by electromagnetic and astroparticle counterparts. As numerical relativity continues to blend in with routine gravitational wave data analyses to validate the discovery of gravitational wave events, it is essential to develop open source tools to streamline these studies. Motivated by our own experience as users and developers of the open source, community software, the Einstein Toolkit, we present an open source, Python package that is ideally suited to monitor and post-process the data products of numerical relativity simulations, and compute the gravitational wave strain at future null infinity in high performance environments. We showcase the application of this new package to post-process a large numerical relativity catalog and extract higher-order waveform modes from numerical relativity simulations of eccentric binary black hole mergers and neutron star mergers. This new software fills a critical void in the arsenal of tools provided by the Einstein Toolkit consortium to the numerical relativity community.

\textsuperscript{5} Author to whom any correspondence should be addressed.
Keywords: gravitational waves, numerical relativity, computational astrophysics, open source software

(Some figures may appear in colour only in the online journal)

1. Introduction

Einstein’s theory of general relativity is one of the greatest accomplishments in theoretical physics [1]. The study of this theory has led to the creation of several research programs in mathematics, experimental and theoretical physics, astronomy, and computer science. Analytical studies of this theory by mathematicians and theoretical astrophysicists over the last century have shed light into the physics of new astrophysical objects, and processes that are dominated by strong gravitational interactions. The advent of fully numerical solutions of Einstein’s field equations imparted a major boost to the study of gravitational wave (GW) sources [2–7], i.e. mergers of compact objects, such as binary black holes (BBHs), binary neutron stars (BNSs), supernovae explosions, etc [8].

Numerical relativity (NR) paved the way for the development of tools that were critical for the detection of GWs with the twin advanced laser interferometer gravitational wave observatory (aLIGO) detectors [9–12]. In addition to providing NR waveforms for the calibration of waveform models that were used extensively to detect and characterize GW transients in aLIGO data [13, 14], NR simulations are also directly used in aLIGO data analyses to validate the astrophysical origin of GW transients [15, 16].

NR will continue to shed light on the physics of compact binary mergers that are putative sources for the generation of the most energetic explosions in the Universe [17, 18]. To accomplish these major scientific milestones, numerical relativists are developing software to simulate mergers of NSs and NSBH systems in numerical settings that reproduce, with ever increasing realism, actual astrophysical scenarios [19–23]. While some of this code remains closed source, there have been community efforts to develop, maintain, and exploit open source NR software, such as the Wolfram Language based package SimulationTools [24]. In this article, we further the development of open source software for the study of GW sources, which will be included in upcoming releases of the open source, community software, the Einstein Toolkit [23, 25].

The software we introduce in this article addresses a critical void in the arsenal of open source tools to monitor and post-process the data products of large scale NR campaigns in high performance computing (HPC) environments. In our experience running and post-processing NR simulations, we have found two paradigms: (i) members of NR groups that develop closed source NR software have a comprehensive toolkit for end-to-end analyses, from the generation of initial data to the extraction of the waveform strain at future null infinity. These tools are not available to the NR community at large; (ii) users of open source software, such as the Einstein Toolkit, develop in-house software for initial data generation, and post-processing of NR simulations. These software packages have not been integrated in code releases of the Einstein Toolkit. One of the existing tools available to post-process NR simulations is SimulationTools [24], an open source package written in Wolfram Language. We have explored the use of this package and have found that

• Purchasing Mathematica licenses can create a barrier to entry users
• Even though HPC providers allow the use of commercial software, e.g. Wolfram Language, as long as users bring their own licenses, using SimulationTools demands a major overhaul of existing libraries installed cluster wide on HPC facilities
such as XSEDE and the Blue Waters supercomputer. This is a rather unfeasible prospect for a narrow application of code based on commercial software.

- If one wanted to use SimulationTools to post-process the data products of a NR campaign run using Blue Waters, XSEDE or any other supercomputing facility that is not compatible with Wolfram Language, one is required to transfer simulation data to an environment compatible with Wolfram Language, which then leads to unnecessary duplication of large datasets.

In this note, we address these issues by introducing a new open source Python package that facilitates and streamlines the monitoring and post-processing of large scale NR campaigns on any platform, using available software modules on HPC environments.

This paper is organized as follows: in section 2, we introduce POWER (Python Open Source Waveform Extractor), describe how to obtain this package, and provide a brief description of its usage and features. In section 3, we use POWER and SimulationTools to compute the GW strain at future null infinity of a number of Einstein Toolkit NR simulations, that we generated using the blue waters supercomputer. Based on these results, we show that POWER provides a ready-to-use solution to monitor NR campaigns and post-process their data products on HPC environments. In section 4 we showcase the application of POWER for three timely research topics in the context of vacuum and matter NR simulations. We describe future directions of work in section 5.

2. Post-processing of numerical relativity simulations in high performance environments

Among the data products of a typical NR simulation, we obtain a set of HDF5 files that contain numerical solutions for the Weyl scalar $\Psi_4(t)$, which is related to the GW strain $h(t)$ as follows

$$h = h_+ - i h_\times = \int_{-\infty}^{t} \, dt' \int_{-\infty}^{t'} \, dt' \, \Psi_4(t').$$

In the transverse traceless gauge, $(h_+, h_\times)$ are the two polarizations of the strain, and represent the observables that can be measured with a GW detector. In order to compute $h(t)$ at null infinity—where GWs are formally defined—it is necessary to convert $\Psi_4(t)$ into $h(t)$, and then extrapolate the strain to infinite radius. This subtle and delicate numerical procedure is done as follows [27].

We convert the $\Psi_4$ simulation data to strain $h(t)$ at each extraction radii using fixed frequency integration (FFI). The cutoff frequency for FFI is obtained from initial parameters by computing the post-Newtonian orbital frequency for the given separation and angular momentum, and then setting this as the cutoff frequency.

To extrapolate the waveform strain to infinite radius, we convert $\Psi_4$ to strain at each extraction radii; interpolate the phase, $\phi$, and amplitude, $A$, of the strain at each extraction radii to a
fixed retarded time $t_{\text{ret}} = t - r$. Finally, we fit an ansatz $f(r) = f_{\text{inf}} + A/r + B/r^2$ to the phase, and $r \times A$ for the amplitude, respectively. For the phase we fix $B = 0$, while for the amplitude ($f_{\text{inf}}, A, B$) are fitted for. POWER will fail with an error message in the case where the $\Psi_4$ amplitude goes to zero.

Having described the standard method for waveform extraction, we now describe how to obtain and use POWER. Our software is available at NCSA's public gitlab repository, and can be cloned as follows

REPO=https://git.ncsa.illinois.edu/elihu/Gravitational_Waveform_Extractor.git
$ git clone REPO.

The command line arguments required to compute the waveform strain $h(t)$ at future null infinity are:

$ cd Waveform_Extractor/POWER
dollar; python power.py n_radii sim_1 sim_2...

where sim_1 and sim_2 represent the NR simulations from which $h(t)$ is extracted using multiple CPU cores if available, and n_radii specifies how many of the innermost extraction radii are used for the extrapolation. If n_radii is not provided, then all extraction radii are used.
used. The extraction radii are given in the parameter file used to run NR simulations with the Einstein Toolkit. Upon executing the above script, the directory, 'Extrapolated_Strain', is automatically created and the files containing $h(t)$ computed at future null infinity, between POWER and SimulationTools using interpolation order 9.

Figure 2. Top left: as figure 1, but now showing interpolation order 9. Top right: with interpolation order 9, we show the difference in phase and amplitude of the waveform strain extracted at future null infinity between POWER and SimulationTools. The bottom panel shows the real part of the waveform strain, $\Re[h(t)]$, computed at future null infinity, between POWER and SimulationTools using interpolation order 9.
derivatives. We also enable the user to compare these quantities to ADM mass and angular momentum versus time inside a given gravitational waveform extraction radius. Having access to this information while a NR campaign is conducted is critical to determine whether the parameter space and resolutions chosen to study an astrophysical problem are sufficient and adequate, respectively. We have found that extracting this information from NR campaigns in an interactive manner saves times and maximizes the use of computational resources. In the following section, we present a direct comparison between POWER and SimulationTools for a variety of NR simulations that are part of the NCSA numerical relativity catalog of eccentric compact binary coalescence, which is presented in an accompanying publication [28].

3. Comparison to SimulationTools

In this section we compute $h(t)$ at future null infinity using POWER, and directly compare it to SimulationTools. Table 1 lists the simulations taken from our NCSA Catalog of NR simulations [28] for this analysis.

In figures 1–3, we present a direct comparison of the gravitational wave strain computed at future null infinity by POWER and SimulationTools, $\Delta R \equiv R \left(h^{\text{POWER}} - h^{\text{SimTools}}\right)$, for a set of moderately eccentric BBH systems. We also present the difference in amplitude and phase for the waveform strain extracted both from POWER and SimulationTools.

**Figure 3.** Using interpolation order 9, we show the difference in phase and amplitude of the waveform strain extracted at future null infinity between POWER and SimulationTools. See table 1 for details about the mass-ratio and eccentricities of these simulations.
SimulationTools uses Hermite interpolation to produce $h(t)$, whereas POWER uses NumPy, which relies on spline interpolation. We have found that both methods agree, i.e. that $\Delta \Re [h(t)] \leq 10^{-7}$, when we use interpolation order five or above. This result is robust for our catalog of more than 70 Einstein Toolkit BBH NR simulations [28]. The user can modify this interpolation order to meet specific needs.

It is worth emphasizing that in POWER the waveform phase and amplitude are interpolated to compute phase and amplitude at fixed retarded time $t_{\text{ret}} = t - r$ for each extraction radii when radially extrapolating the strain. The interpolation is performed after the strain at the individual detectors is calculated, but before the radially extrapolated strain is computed. The interpolation order refers to spline interpolation order for POWER and Hermite interpolation order for SimulationTools. The examples shown only use the $(\ell, m) = (2, 2)$ mode, but we show in figure 5 that POWER processes all of the modes present in the input files.

The results shown in figures 1 and 3 are those of the highest resolution numerical simulation with the errors quoted being the difference between this value and the Richardson extrapolated value. These results are not Richardson extrapolated to infinite resolution directly though the numbers provided (highest level value plus error estimate) allow for a simple computation of the Richardson extrapolated value.

For the two NR simulations shown in figure 3, we set the interpolation order to 9, and find that $\Delta \Re [h(t)] \leq 10^{-7}$, and that discrepancies in phase and amplitude are always $\leq (10^{-3}, 10^{-7})$, respectively, which is and less than the numerical error of the NR simulations. It is reassuring that the differences $(\Delta \Re [h(t)], \Delta \phi, \Delta \text{Amplitude})$ for different mass-ratios and eccentricities are of similar magnitude for all waveforms in our catalog.
4. Science cases and extension

To showcase the versatility and applicability of POWER both for vacuum and matter NR simulations, in this section we discuss a variety of projects that can make use of our new software package.

We have recently used POWER to post-process our first catalog of eccentric NR simulations that span a parameter space that describes BBH mergers with mass-ratios $q \leq 6$, and eccentricities $e \leq 0.21$ ten orbits before merger. The first batch of waveforms with mass-ratios $q \leq 4$ in shown in figure 4. A detailed analysis of these NR simulations is presented in an accompanying paper [28].

Given that POWER is capable of extracting all of the higher-order waveform multipoles present in the data products of NR simulations, we have exploited this feature to start extracting the higher-order modes of our eccentric BBH merger catalog. These results are shown in figure 5. These results can shed light into the importance of these higher-order contributions for the detectability of eccentric compact binary coalescence, and issue that has not been addressed in the literature.

Finally, even though POWER was developed to post-process vacuum NR simulations, we have now made it flexible enough to post-process the data products of matter simulations. In the bottom right panel of figure 5, we present the waveform strain computed at future null infinity from the merger of two equal mass neutron stars, using the MS1b equation of state.

*Figure 5.* Top panels, and bottom left panel: extraction of higher-order waveform multipoles from the binary black hole NR simulation J0068. Bottom right panel: extraction of the waveform strain of a binary neutron star merger simulated with the GRHydro code, and using the MS1b equation of state.
This simulation was generated and post-processed using the open source GRHydro code [22], and POWER, respectively, in the Blue Waters supercomputer.

In summary, we have developed an open source tool that is ready to use to monitor the progress of NR campaigns, and to compute the GW strain $h(t)$ at future null infinity. These results indicate that POWER is an ideal open source tool to post-process NR simulations on any leadership HPC facility that provides numpy and scipy.

5. Conclusion

We have developed, and made publicly available, a Python package to monitor and post-process NR simulations on HPC environments. This software is available as a stand alone tool, and will be included in upcoming releases of the open source, community software the Einstein Toolkit. This tool will facilitate and streamline the continuous monitoring needed in NR campaigns, and the subsequent post-processing required to compute the GW strain at future null infinity. This work fills in a critical void in the arsenal of tools that is available to numerical relativists interested in assisting the LIGO Scientific Collaboration in the validation of GW transients, or to develop new waveform models from the ground up, among other activities.

Acknowledgments

This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the State of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications (NCSA). The eccentric numerical relativity simulations used in this article were generated with the open source, community software, the Einstein Toolkit on the Blue Waters petascale supercomputer and XSEDE (TG-PHY160053). This work was partially supported by the NSF S12-SSI award OAC-1550514. We acknowledge support from the NCSA and the SPIN (Students Pushing Innovation) Program at NCSA. We thank Ian Hinder and Barry Wardell for the SimulationTools analysis package. Plots were generated with Matplotlib [29]. POWER uses numpy [30] and scipy [31].

ORCID IDs

Daniel Johnson © https://orcid.org/0000-0001-7717-5640

References

[1] Einstein A 1915 Königlich Preussische Akademie der Wissenschaften Zu Berlin 1915 844–7
[2] Pretorius F 2005 Phys. Rev. Lett. 95 121101
[3] Nakamura T, Oohara K and Kojima Y 1987 Prog. Theor. Phys. Suppl. 90 1–218
[4] Shibata M and Nakamura T 1995 Phys. Rev. D 52 5428–44
[5] Baumgarte T W and Shapiro S L 1999 Phys. Rev. D 59 024007
[6] Baker J G, Centrella J, Choi D I, Koppitz M and van Meter J 2006 Phys. Rev. Lett. 96 111102
[7] Campanelli M, Lousto C O, Marronetti P and Zlochower Y 2006 Phys. Rev. Lett. 96 111101
[8] Sathyaprakash B S and Schutz B F 2009 Living Rev. Relativ. 12 2
[9] Abbott B P et al 2016 Phys. Rev. Lett. 116 061102
[10] Abbott B P et al 2016 Phys. Rev. Lett. 116 241103
[11] Abbott B P et al 2017 Phys. Rev. Lett. 118 221101
[12] Abbott B P et al 2017 Phys. Rev. Lett. 119 141101
[13] Taracchini A et al 2014 Phys. Rev. D 89 061502
[14] Khan S, Husa S, Hannam M, Ohme F, Pürrer M, Forteza X J and Bohé A 2016 Phys. Rev. D 93 044007
[15] Abbott B P et al 2017 Class. Quantum Grav. 34 104002
[16] Abbott B P et al 2016 Phys. Rev. D 94 064035
[17] Lehner L and Pretorius F 2014 A&A 52 661–94
[18] Abbott B P et al 2016 Living Rev. Relativ. 19 1
[19] Kidder L E et al 2017 J. Comput. Phys. 335 84–114
[20] Haas R et al 2016 Phys. Rev. D 93 124062
[21] Etienne Z B, Paschalidis V, Haas R, Mösta P and Shapiro S L 2015 Class. Quantum Grav. 32 175009
[22] Mösta P, Mundim B C, Faber J A, Haas R, Noble S C, Bode T, Löffler F, Ott C D, Reisswig C and Schnetter E 2014 Class. Quantum Grav. 31 015005
[23] Löffler F et al 2012 Class. Quantum Grav. 29 115001
[24] Hinder I and Wardell B 2017 Simulation Tools is a free software package for the analysis of numerical simulation data in mathematica http://simulationtools.org/
[25] Einstein Toolkit Consortium 2017 The Einstein Toolkit http://einsteintoolkit.org
[26] The HDF Group 1997–2017 Hierarchical data format, version 5 http://hdfgroup.org/HDF5/
[27] Reisswig C and Pollney D 2011 Class. Quantum Grav. 28 195015
[28] Huerta E A et al 2017
[29] Hunter J D 2007 Comput. Sci. Eng. 9 90–5
[30] van der Walt S, Colbert S C and Varoquaux G 2011 Comput. Sci. Eng. 13 22–30
[31] Jones E et al 2001–2017 SciPy: open source scientific tools for Python [Online; accessed 20 November 2017]