4 Project Description

4.1 Software Development Plan

The Einstein Toolkit is designed to simulate compact binaries as sources for gravitational waves, i.e., to evolve a system of coupled, nonlinear partial differential equations of elliptic and hyperbolic character on three-dimensional spatial grids in time. It uses the Cactus computational framework to connect different modules and to achieve a clean separation between science and infrastructure components. This enables domain experts in astrophysics and computer engineering to focus their efforts on the components they are most comfortable dealing with. All components within the Einstein Toolkit are distributed using free and open source licenses enabling users to mix and match modules, adapt modules to their own needs and share these modules freely with collaborators.

The Einstein Toolkit can be compiled on most modern POSIX systems, ranging from desktop systems running Linux or MacOS to leadership class supercomputer systems including the new Frontera system at TACC,[195] using a variety of compilers such as the GNU, clang, Intel, PGI, Cray and IBM compilers, on x86, POWER and ARM architectures.

Within the Einstein Toolkit the "driver" is a distinguished module responsible for handling parallelization, memory allocation and input-output operations and thus is of critical importance for the overall performance and scalability of the code. Due to the Einstein Toolkit's modularity this driver can be replaced without affecting the science modules.

The first major focus of the work will be to replace Carpet, the current Adaptive Mesh Refinement (AMR) driver, with CarpetX, a new driver based on the AMReX mesh refinement library.[196] AMReX is developed as part of the DOE's Exascale Computing Project (ECP). This will let the Einstein Toolkit tap into the scalability and functionality improvements of AMReX. Thus, the Einstein Toolkit infrastructure will, effectively, gain the advantage of having full-time development team, currently numbering 10,[197] working on the core of its AMR infrastructure. This will free up Einstein Toolkit developers to work on other issues.

The Einstein Toolkit has a long history[121,198] of using automated code generation to create the kernels of computational relativity codes. In this proposal we will develop NRPy+ as the primary tool for code generation in the Einstein Toolkit. NRPy+[170,199,200] exists as a free and open source piece of software based in Python and its powerful open-source computer algebra system SymPy. While this open software environment makes NRPy+ more easily extensible to target new architectures and numerical methods, its pedagogic, Jupyter-notebook-based documentation make it more easily accessible to new users. Finally, in close collaboration with the group of Hari Sundar, we will target NRPy+-based code generation for GPUs to better integrate the Einstein Toolkit with today's increasingly heterogeneous supercomputing systems.

4.1.1 CarpetX

State-of-the-art The Einstein Toolkit has been very successful with 98 high-quality publications resulting directly from simulations performed using the toolkit and its components during the prior grant's lifetime. Critical to the success of the toolkit is the underlying AMR driver. The Carpet driver, designed for the Cactus infrastructure,[111,113] has been the workhorse for many numerical relativity groups for well over a decade. A survey of the structured AMR frameworks currently in use finds that the majority are facing scalability challenges due to changing performance balances from the time of their original inception.[201] This concerns, in particular, the need to effectively use many core CPUs, GPUs and mechanisms to hide latency in or eliminate data movement. While solutions to these challenges are being explored, doing so in a production code, with limited resources to develop a new code from scratch is challenging for science communities.

With AMReX, the Einstein Toolkit community can leverage the support of the AMReX community to ensure that the underlying AMR driver will scale to ever larger systems. AMReX already implements algorithms to enable efficient use of current CPUs, as well as asynchronous iterators overlapping computation with communication, which will benefit all codes in the Einstein Toolkit. The continued support for AMReX through the DOE's ECP[202] co-design approach,[197] where it is currently used by 7 applications funded by ECP, will provide a stable basis for the Einstein Toolkit past the end of this proposal.

Several attempts, such as the Chemora project,[198] to incorporate GPUs into the Cactus framework have been made over the past years, and they have only met with partial success. AMReX sees active development to improve its GPU support and we will collaborate with the AMReX developers to support this new feature in the Einstein Toolkit. An issue which any GPU framework must address is data movement. AMR requires a number of special operations: i.e. prolongation, by which data is interpolated from coarser regions of the grid to the boundaries of the finer grid; and restriction, by which data is sent from the finer grids to the coarser ones. Optimizing this data movement is a deep problem, probably the dominant problem for any serious AMR code. AMReX will provide this functionality to the Einstein Toolkit as part of its GPU support.

Specific objectives The CarpetX driver in Cactus acts as a high level interface translating between Cactus and AMReX. Beyond these, a Cactus driver has to provide functionality to compute reductions (integrals, extrema, various norms) over data on the grid, interpolation functionality to evaluate date on the grid at arbitrary coordinates, not necessarily coinciding with grid points, and input-output functionality used to checkpoint a simulation, output 1D, 2D, 3D data and read in initial data.

Haas and Schnetter will take the lead in providing the core, low level functionality using their prior experience as developers of Carpet. For this they will collaborate with the AMReX developers to make best use of the infrastructure and keep abreast of its current developments. They, together with students and postdoctoral researchers, will also implement the high-order transport (prolongation and restriction) operators. These are imperative to evolve Einstein's equations with the toolkit, but are not currently provided by AMReX since their client codes are mostly Newtonian (magneto-)hydrodynamic codes. Brandt, with students at the collaborating institutions, will ensure that CarpetX supports the data dependent scheduling developed for Cactus during the lifetime of the prior awards from the start. The Cactus runtime has access to a rich set of contextual information about data dependency between the functions present in the active Cactus modules. This information together with the support for asynchonous task scheduling currently under development in AMReX, will be employed for overlap computing, e.g. of spacetime and of fluid quantities, and to increase the number of parallel work items available to the function scheduler.

Input-output, checkpointing, reduction and interpolation functionality will be built on top of this framework. Haas, Brandt, and Zlochower will work with students and postoctoral researchers to implement these. While some functionalities such as interpolation can be mapped to existing AMReX functionality, others like reductions over the grid will be implemented from scratch. Supplying this functionality will allow the majority of the existing Einstein Toolkit modules to be used with CarpetX without code changes. Importantly this will allow for early performance evaluation of CarpetX with real-world workloads and identification of potential weak areas.

The initial implementation of CarpetX will use a uniform timestep across the domain, but we plan to add subcycling in time, also called local timestepping later in the project. This choice simplifies the code and simplifies using refluxing methods (to ensure mass conservation at mesh refinement boundaries) and divergence preserving transport operators (to avoid creating magnetic monopoles) already present or in the process of being implemented in AMReX. Schnetter, Haas and the postdoctoral researcher will lead the effort of adding full Berger-Oliger sub-cycling in time, using expertise based on the existing sub-cycling capability in Carpet.

The Einstein Toolkit offers excellent backwards compatibility in the interfaces provided to the science code, a crucial feature for reproducable scientific simulations. However, we will use the transition to CarpetX to re-visit interface decisions that, while beneficial at the time, have proven to a be bottleneck to overall code performance. Modules adopting the new interfaces will improve in performance at the cost of a loss of backwards compatibility.

4.1.2 NRPy+

State-of-the-art Code generation has been an important part of the Einstein Toolkit for a long time. The Einstein equations are sufficiently complex, that experimenting with them in any meaningful way is difficult without a symbolic manipulator of some flavor. Traditionally the Einstein Toolkit relied on Kranc, a Mathematica-based code generator targeting the Cactus framework. Using Kranc, researchers are able to automatically generate vectorized, highly tuned code.

NRPy+[170, 199, 200] on the other hand uses Python and its computer algebra package SymPy[203], lowering the entrance barrier for users by avoiding the use of Kranc's custom, domain specific language while offering greater control through a more hands-on interface. NRPy+ generated code uses SIMD instructions of modern CPUs including fused multiply-add instructions which are inserted during the common-subexpression elimination (CSE) stage. NRPy+ generated code has been shown to exceed Kranc's code performance by 5–20%. Existing modules targeting the Einstein Toolkit that use NRPy+ include a spacetime evolution code, an initial data module for an accretion disk around a black hole, and modules to compute the gravitational waves in a spacetime.

NRPy+ focuses on ease of use for new users, being documented in pedagogical, interactive Jupyter notebooks accessible from https://mybinder.org[204] (via GitHub[200]). Since its release two years ago, NRPy+ has been used by groups at 11 institutions world-wide. Typical use involves modifying one of the fully functional, example NRPy+ standalone (i.e., non-Einstein Toolkit) codes to suit their needs. Thus, as the NRPy+ user community grows, the need for better integration with the Einstein Toolkit becomes essential, so that both communities can grow together and mutually benefit from overlapping expertise.

Specific objectives NRPy+ was not developed with the Einstein Toolkit in mind and as a result, writing Einstein Toolkit modules requires significant prior expertise. To lower the barrier to entry for new Einstein Toolkit users, and ready the Einstein Toolkit and NRPy+ for next-generation multimessenger simulations, we will improve NRPy+'s integration with the Einstein Toolkit and extend NRPy+'s capabilities to better serve the needs of the astrophysics computational modeling community.

This work will take 4 basic forms: generating complete Einstein Toolkit modules instead of compute kernels only, generating GPU kernel code, expanding the set of curvilinear coordinate systems supported by NRPy+, and developing a toolbox for finite-volume schemes.

With students at WVU and LSU, Etienne and Brandt will lead the effort to extend NRPy+ to output complete ET modules, including dependency tracking in the function scheduler developed during the prior grant. As part of this work, they will work to extend NRPy+ to accept LaTeX-like human-friendly input for tensorial equations using Einstein notation, thereby reducing index errors as one of the most common sources of bugs. Etienne, collaborator Hari Sundar, the postdoc and Brandt will develop code to generate GPU kernels usable with AMReX's GPU offload functionality, including an abstract interface to loops over the computational grid which will be fed back to the CPU code. This high level interface will allow the same NRPy+ source code be used to generate both CPU and GPU kernels optimized for the specific target showcasing the benefit of using code generation. Etienne and student will extend the suite of curvilinear coordinate systems supported by NRPy+ within the toolkit; of interest when modeling systems possessing near symmetries, and an excellent avenue for the smallest-of-scale NR simulations (i.e., NR on laptops & desktops).

Etienne, Haas, Witek and the students and postdoc will develop a toolbox of finite volume methods to quickly explore novel formulations of hydrodynamics and evolution schemes.

4.1.3 Canuda: enabling fundamental physics

State-of-the-art: Numerical relativity has become a mature tool to investigate the nonlinear regime of gravity. In addition to its original playground in astrophysics, it has become ever more important in the context of cosmology, theoretical gravity and high-energy physics. In fact, the last year has witnessed the emergence of the new research direction "numerical relativity beyond general relativity".[205–213] The novel library Canuda,[214] an open-source software that is fully compatible with the Einstein Toolkit and has been developed by members of this team, has played a key role in promoting these exciting new developments. Its current design capabilities include (i) evolution and initial data modules to simulate the interplay between *single* black holes and ultra-light bosonic fields[215,216] that have already been employed by the wider scientific clientele,[217] and (ii) initial data and evolution modules that enabled first proof-of-principle simulations of black hole binaries in quadratic gravity up to first order in an effective-field-theory-like (EFT) treatment.[213,218,219]

Specific objectives: These have only been the first steps and we intent to further develop Canuda's capabilities to enable new simulations of compact binaries in extensions of GR, as well as the production of waveforms to be exploited by the LIGO-Virgo Collaboration, the LISA Consortium and the broader scientific community. In particular, we plan to develop new initial data and evolution modules, together with modules providing observables such as gravitational waves or (apparent) horizons that are collected in specific arrangements. These arrangements will be designed to (i) simulate black-hole *binaries* coupled to ultra-light scalar or vector fields; (ii) simulate black-hole binaries in quadratic gravity to higher order in an EFT expansion such that effects on the gravitational wave signal are included; (iii) enable "parametrized numerical relativity" where modifications of a new theory (expanded in an EFT fashion) are treated as source terms that can be easily coupled to the new software infrastructure. Furthermore, all modules are implemented in a user-friendly, readable fashion and are accompanied by educational examples and documentation.

Witek, Laguna, Shoemaker and Zilhão, together with Postdoc and students, will further develop Canuda. After its completion, Canuda's scientific applications are manifold, and encompass the exploration of compelling dark matter candidates using black holes, predictions of gravitational wave signatures of quantum gravity as well as the investigation of the *nonlinear* stability of black holes in GR that has remained a century-old problem.

4.2 Community Needs and Alternatives

Before the first release of the Einstein Toolkit in 2010, the only numerical relativity, software infrastructures available were those developed by individual investigators or groups. The availability of those codes was mostly confined within the walls of the collaborations of their developers.[220,221] The exception perhaps is the Cactus infrastructure, the precursor to the Einstein Toolkit. The Einstein Toolkit is the first example of a fully open-source, production code to tackle numerical relativity problems. Inspired by the Einstein Toolkit several groups have since started developing free and open-source astrophysics codes such as GRChombo,[222] Dendro-GR[223] and SpECTRE.[224]

None however has reached the widespread acceptance that the 55 publications acknowledging use of the Einstein Toolkit from groups other than this proposal's demonstrates. Thus it is not an exaggeration that the future research of several groups would be severely compromised without the Einstein Toolkit infrastructure.

In an effort to make functionality present in the Einstein Toolkit available to other codes we will work with external collaborators to developed libET linking the Einstein Toolkit into their codes via a new Einstein Toolkit library interface. This will make the state of the art analysis modules present in the Einstein Toolkit available to these codes.

5.2 Outreach & Education

To infuse interaction with the broader community and the public, the Einstein Toolkit will continue to develop the main website as a portal to a wealth of research and education material. In addition to the code itself, the website will disseminate educational material, including projects using the data for high school and college instructors, a schedule of the workshops and summer schools for both teachers and researchers, as well as the material used in Einstein Toolkit workshops and tutorials. The web-portal will also host a speakers' bureau and a series of online review articles suitable for scientists entering our research field.

The Einstein Toolkit will be used increasingly in the local education of students, e.g. as integral part of the "High-performance computing" course taught at RIT within their new Mathematical Modeling PhD program, and for the training of all PhD students envisioned in this grant.

5.2.1 Tutorial Server

The Einstein Toolkit community has devised an outstanding teaching and tutorial system. To this end, we set up the NDSLabs machine at https://etkhub.ndslabs.org. The system uses CILogon to manage credentials and provide users with access to Jupyter notebooks that demonstrate how to compile the Einstein Toolkit, run it, and develop modules for it.

We will continue the development of this system by capitalizing on the feedback that we receive during in-person training at workshops, and expand it to a larger variety of Cactus features as well as new software modules. This will include (i) step-by-step videos (including screencasts) that will demonstrate how to install, build, run, and extend the toolkit; (ii) curated virtual machines with the toolkit pre-configured on various flavors of Linux; and (iii) step-by-step instructions for configuring and running simulations of BNS or BBH.

To further enhance the efficiency of our training activities we plan to increasingly use the Python-based NRPy+ as user-friendly and open-source interface with Einstein Toolkit. During the course of this grant, NRPy+ will develop the ability to easily generate entire modules, and with each module generated associated pedagogical Jupyter notebooks (exportable as PDFs) will be created and disseminated.

5.2.2 Portal

In the future, we plan to add a portal or "Science Gateway" to our online teaching tools. This will allow students to become familiar with what the Toolkit does and how to use it before understanding how to build and develop it. It will also allow researchers who are not interested in the mechanics of computational science to make use of the toolkit. The use of Science Gateways on XSEDE resources is on the rise, increasing by a factor of five during the years 2011-2017,[229] which indicates this to be an important trend in computational research. To implement this portal, we plan to leverage the Agave framework[230][†]. Agave/Tapis provides a system to manage credentials, providence tracking, share results, and archive them. Its ToGo[231] application provides a basic web-portal application out of the box which may be customized to fit the needs of our particular application. The customization will include graphical tools for visualizing results and the interface to the parameter files.

Provenance for all simulations run through the portal will be tracked in two ways. First, through the Agave/Tapis Frameworks logs, which track all details of the "when," "where," and "who" associated the run, including the user id that submitted the job, the date and time it queued, ran, etc. Cactus's Formaline module will keep a copy of the entire source code and parameter file used to generate the simulation, providing the details of "what" was run. In addition, once NRPy+ is fully developed, it will become possible to easily generate whole modules from mathematical expressions within the portal.

[†]Note that Agave has recently been rebranded as Tapis by TACC.