Black Hole Simulations with CUDA

Frank Herrmann, John Silberholz, Manuel Tiglio
Center for Scientific Computation and Mathematical Modeling,
Department of Physics, Center for Fundamental Physics, Joint Space Institute.
University of Maryland, College Park, MD 20742, USA.

In 1915 Albert Einstein derived the fundamental laws of General Relativity, which predict the existence of gravitational radiation. While indirect evidence is by now well established and led in 1993 to the Nobel Prize in Physics being awarded to Hulse and Taylor, no direct detection has occurred to date. This is because gravitational waves reaching Earth are extremely weak. Recently, researchers have built gravitational wave detectors achieving the required sensitivities to make direct detection possible. A number of laser interferometer experiments are devoted to this task. The currently most sensitive one (LIGO [1]) consists of two crossed laser beams each of which is 4 km long. Over this distance a typical gravitational wave would change each arm's length by $\sim 10^{-18}$ m — a fraction of the size of an atom. Because the signal to be extracted is very small compared to the background noise, an accurate a priori knowledge of inspiral signals that can be searched for in the data is required.

In this chapter we discuss our CUDA and GPU implementation and results from an approximation to Einstein's equations, the *post-Newtonian* one, which allows us to thoroughly study the inspiral phase-space dynamics of the 7-dimensional parameter space of binary black holes in initial quasi-circular orbit.

1 Introduction

With a number of gravitational wave detectors such as LIGO [1], Virgo [2], and GEO600 [15] now measuring at design sensitivity, the prospect of direct detection of gravitational radiation is becoming increasingly real. Due to the

low signal-to-noise ratio encountered at these detectors, researchers need to post-process the data, ideally searching for known signals that would indicate a source of gravitational radiation. One of the most likely sources of detection are stellar-mass binary black hole (BBH) systems. There is a definite need to understand as much as possible about these configurations.

Over recent years numerical solutions of General Relativity (GR) have reached the point were accurate and reliable calculations of the interactions of two black holes (BHs) over many orbits can be produced by a number of different groups (see [12, 4, 3] and references therein). These calculations require enormous computational effort, though, and the generation of a large-scale bank of numerical templates that could be used by gravitational wave detectors is currently intractable. Numerical solutions to Einstein's equations require solving a coupled set of non-linear elliptic-hyperbolic partial differential equations in three spatial dimensions. A single right-hand-side call to advance in time just one point on the three-dimensional spatial grid requires around $13,000$ multiplications, $5,400$ additions, $3,400$ subtractions and 69 divisions, excluding derviative calculations. A single simulation can easily take more than $100,000$ CPU hours. These simulations are typically running on more than 1024 cores simultaneously so they can complete in a reasonable amount of time. Here we report on an approximation to GR, the so-called post-Newtonian expansion (PN), which — as explained in the next section — dramatically reduces the number of computational operations required. Furthermore, recent comparisons between gravitational waveforms obtained from full numerical relativity (that is, numerical solutions of the full Einstein equations) and PN approximations show good agreement until just a few orbits before the two black holes merge [5, 13, 8].

Because of these reasons, performing a large number of PN simulations of BBH systems with different initial configurations is of considerable interest, and the focus of our work described in this chapter.

2 The Post-Newtonian Approximation

The PN approximation leads to a system of coupled ordinary differential equations (ODEs) in time for the orbital frequency ω, the unit orbital angular momentum vector $\hat{\mathbf{L}}$ and the individual spin vectors $\mathbf{S}_i$ for the two BHs. This expansion is valid as long as the two black holes are far separated and slowly moving along a quasi-circular inspiral trajectory. In total this is a

7-dimensional problem as there are three degrees of freedom in each of the spin vectors and one degree of freedom in the mass ratio.

The specific PN expansion and equations that we use are those of Ref. [6] (with Erratum [7]); namely,

$$\begin{aligned}
\frac{d\omega}{dt} &= \omega^2 \frac{96}{5}\eta (M\omega)^{5/3}\Bigg\{1 - \frac{743+924\eta}{336}(M\omega)^{2/3} \\
&- \left(\frac{1}{12}\sum_{i=1,2}\left(\chi_i \hat{\mathbf{L}}\cdot\hat{\mathbf{S}}_i(\frac{113 m_i^2}{M^2}+75\eta)\right) - 4\pi\right) M\omega \\
&+ \left(\frac{34103}{18144} + \frac{13661}{2016}\eta + \frac{59}{18}\eta^2\right)(M\omega)^{4/3} \\
&- \frac{1}{48}\eta\chi_1\chi_2\left(247(\hat{\mathbf{S}}_1\cdot\hat{\mathbf{S}}_2) - 721(\hat{\mathbf{L}}\cdot\hat{\mathbf{S}}_1)(\hat{\mathbf{L}}\cdot\hat{\mathbf{S}}_2)\right)(M\omega)^{4/3} \\
&- \frac{1}{672}(4159+15876\eta)\pi(M\omega)^{5/3} + \Bigg(\left(\frac{16447322263}{139708800} - \frac{1712}{105}\gamma_E + \frac{16}{3}\pi^2\right) \\
&+ (-\frac{273811877}{1088640} + \frac{451}{48}\pi^2 - \frac{88}{3}\hat{\theta}\eta)\eta \\
&+ \frac{541}{896}\eta^2 - \frac{5605}{2592}\eta^3 - \frac{856}{105}log(16(M\omega)^{2/3})\Bigg)(M\omega)^2 \\
&+ (-\frac{4415}{4032} + \frac{358675}{6048}\eta + \frac{91495}{1512}\eta^2)\pi(M\omega)^{7/3}\Bigg\} \qquad (1)
\end{aligned}$$

$$\frac{d\mathbf{S}_i}{dt} = \mathbf{\Omega}_i \times \mathbf{S}_i \qquad (2)$$

$$\frac{d\hat{\mathbf{L}}}{dt} = -\frac{(M\omega)^{1/3}}{\eta M^2}\frac{d\mathbf{S}}{dt} \qquad (3)$$

where $\mathbf{S} = \mathbf{S}_1 + \mathbf{S}_2$, $\gamma_E = 0.577\ldots$ is Euler's constant, $\hat{\theta} = 1039/4620$, $M = m_1 + m_2$ is the total mass, $\eta = m_1 m_2/M^2$ the symmetric mass ratio, and the magnitude of the orbital angular momentum is $|\mathbf{L}| = \eta M^{5/3}\omega^{-1/3}$.

The evolution of the individual spin vectors $\mathbf{S}_i$ for the two BHs is described by a precession around $\mathbf{\Omega}_i$, Eq. (2), with

$$\mathbf{\Omega}_1 = \frac{(M\omega)^2}{2M}\left(\eta(M\omega)^{-1/3}(4+3\frac{m_2}{m_1})\hat{\mathbf{L}} + 1/M^2(\mathbf{S}_2 - 3(\mathbf{S}_2\cdot\hat{\mathbf{L}})\,\hat{\mathbf{L}})\right), \qquad (4)$$

and $\mathbf{\Omega}_2$ obtained by the exchange $1 \leftrightarrow 2$. The spin vectors $\mathbf{S}_i$ are related to the unit ones $\hat{\mathbf{S}}_i$ via $\mathbf{S}_i = \chi_i m_i^2 \hat{\mathbf{S}}_i$, with $\chi_i \in [0, 1]$ the dimensionless spin parameter of each black hole.

Given mass and spin parameters $(m_1, m_2, \chi_1, \chi_2)$, the above system of coupled ODEs is evolved from an initial frequency ω_0 to a final one ω_f. We typically choose ω_0 corresponding to an initial separation of $r \approx 40M$ and $\omega_f = 0.05$, which is a conservative estimate of where the PN equations still hold [5, 13].

3 Numerical Algorithm

The ODEs are integrated in time using Dormand-Prince's method [10]. This ODE integrator uses an adaptive time step h to keep the solution error below a given threshold. Starting from time t, the time integrator updates the state to $t + h$. It computes an approximation to the solution at six intermediate times between t and $t+h$ (referred to as `k1` to `k6` below) and then computes 4^{th}-order and 5^{th}-order accurate solutions at $t + h$ (here denoted `y4` and `y5`). This requires six right-hand-side calls, i.e. six evaluations of the right-hand-sides of Eqs. (1,2,3). The difference between these solutions is used to estimate the error of `y4` and adapt h if the error is below some specified tolerance. In our simulations we typically set the tolerance to be 5×10^{-7} (i.e. roughly at the level of single-precision accuracy) and start with an adaptive time step of size $h = 10M$. The time step does not change during most of the inspiral. This is crucial for our GPU implementation, because — as discussed later in the chapter — it minimizes thread divergences when multiple inspirals are performed in parallel.

4 GPU implementation

Each ODE integration described above cannot be easily parallelized, because the solution at each time step (even for the intermediate `k1` to `k6` values) depends on the previous one. However, if one wants to study an ensemble of BBH inspiral dynamics (for example in a phase-space study) then a trivial parallelization is available by just performing many inspirals simultaneously.

Fig. 1 shows a CUDA pseudo-code description of our implementation.

```
// compute right hand side (rhs) of ODE
__global__ void ode_rhs(state)

// integrate the ODEs
void integrate(initial_state_and_params) {
  allocate_gpu_storage(...);
  while (all_omega>all_omega_final) {
    ode_rhs<<<nBlocks_rhs,nThreads>>>(...);
    checkCUDAError("first ode_rhs call");
    interm_1<<<nBlocks_interm,nThreads>>>(...);
     ... // more intermediate values and rhs calls
    adjust_timesteps(...);
  }
  transfer_from_gpu_to_host(...);
}
```

Figure 1: Pseudocode describing the parallel ODE integration. The time integration is performed on the CPU with kernels performing the inspirals called on the GPU as described in the text.

The right-hand side (RHS) expressions of Eqs. (1), (2) and (3) are implemented in the `ode_rhs` function on the GPU. The ODE integration is performed on the CPU. First, gpu storage is allocated in `allocate_gpu_storage`. Then the initial state data (i.e. ω, $\mathbf{S}_i$ for the two BHs, and the unit orbital angular momentum vector $\hat{\mathbf{L}}$) is set. In the evolution loop the RHS is called and the different intermediate values `k1` to `k6` are computed in `interm_1`, etc. Next, for each inspiral the timestep h is adjusted separately, i.e. h is a vector with entries for each parallel inspiral. Finally the loop is terminated once all ω have reached the final value ω_f to a given precision. At the end the data is copied back from the GPU to the CPU. During each timestep a memory copy from the GPU to the CPU is needed to check if all simulations are done. In the profiling Section 5 below we will see that no significant amount of time is spent in this copying.

Unfortunately, BBH systems interact differently based on their initial state, which describes their relative masses and the orientations of their spins. In particular this will lead to thread-divergence of the different inspirals. This could potentially be a very serious problem as time-steps are adjusted in different manners and the load/store patterns of the different threads

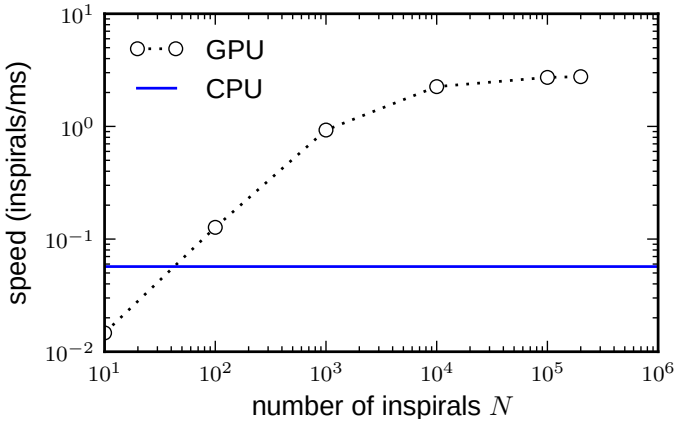
running in parallel changes. In our simulations we have found, however, that the divergence in the threads only appears towards the very end of the inspirals as only then are the time-steps adjusted in a dynamic manner, typically resulting in a runtime difference of at most 10% from just running many copies of the same inspiral. This simplifies the problem significantly, as we do not have to handle thread divergence.

GPUs today provide the most impressive speed-ups over CPUs for single-precision computations. By comparing single-precision and double-precision CPU results on a number of inspiral configurations we have verified single-precision is sufficient for our problem, yielding a maximum error of just over 5% for any final parameter after the simulation, sufficient for our studies.

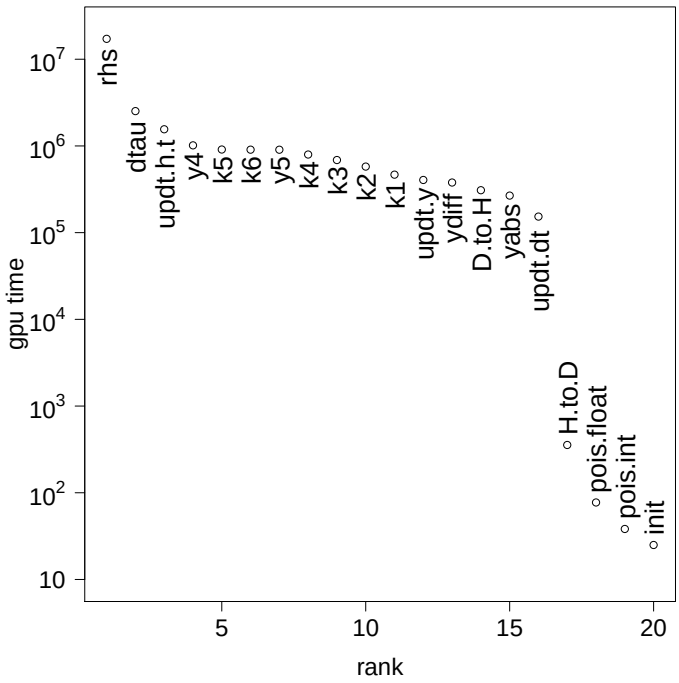
5 Performance Results

We now evaluate the performance advantages GPUs deliver in our application. We executed our code on a single core of a quad-core Intel Xeon E5410 CPU running at 2.33 GHz, which is rated at around 5 GFlops in double-precision. Note that since the problem is embarrassingly parallel the CPU would be able to provide excellent scaling over its 4 cores. We integrated one of our test inspirals 100 times serially and found that we could achieve around 0.057 inspirals per millisecond (ms) on a single core — which we will use for our performance comparison.

For comparison, each of the four GPUs on a high-end unit (the NVIDIA Tesla S1070) is currently rated at 1035 GFlops in single-precision, delivering a theoretical performance advantage of about a *factor 200* over the single-core CPU. We ran our test setup on a single GPU, spawning a large number N of simultaneous inspirals in parallel. Fig. 2(a) shows results for the performance of the GPU card. As we increase the number of simultaneously scheduled inspirals N the speed levels off at about 2.7 inspirals/ms. The GPU has 240 processors that work in parallel, but the runtime CUDA scheduler has to be able to keep these 240 processors working simultaneously by switching out threads that would pause while waiting for a memory access to complete. This is the reason why the performance still rises even after $N = 10,000$, as the runtime has a better chance of squeezing the optimal performance out of the card by interleaving different inspirals without having to wait for costly memory transfers. Fig. 2(b) shows more detailed profiling information. Listed is the time spent on different routines for one of our simulations



(a)



(b)

Figure 2: Performance of the GPU. Fig. (a) shows scaling as we increase the number of inspirals N simultaneously scheduled on the GPU. Fig. (b) shows profiling results — right-hand-side (RHS) computations dominate, as expected.

run with $N = 50,000$; note the log-scale. The right-hand-side (`RHS`) computations dominate, as expected from the arithmetic intensity of Eqs. (1), (2) and (3). The next most expensive routines relate to the time update step. The `y4` and `y5` are the routines performing the estimation updates from the ODE integrator mentioned in Sec. 3, the `k1` to `k6` are the routines for the intermediate computations, while `D.to.H` and `H.to.D` refer to device-to-host and host-to-device `memcopy` operations. Finally, `pois.float`, `pois.int` and `init` are all initialization steps, which are only run once.

We performed this study using block sizes of 256 threads. We found similar behavior with 128-thread blocks. Based on this data, we achieve a *speed-up of about a factor 50*. While this comparison is slightly unfair to the CPU because we only use a single core and double-precision on the CPU, there is no doubt that the performance gains are very significant. Note also that for this test problem we integrated the exact same problem in all cases. This means that the individual threads run in perfect lock-step, the best possible case for the GPU. As mentioned above, we typically see differences of about 10% in the runtime of different inspirals, and this would result in a slight inefficiency on the GPU as some of the threads in a block may finish before others.

6 GPU Supercomputing Clusters

To further speed our computations, we used the NCSA Lincoln cluster, which is a cluster of 96 NVIDIA Tesla S1070 units connected to 192 8-core Dell PowerEdge 1950 servers (each scalar unit allows access to two GPUs). To perform these computations, we used Message Passing Interface (MPI), designating one node as a lead node that delegates tasks to others and keeps track of the progress of the overall job. We subdivided each large computation into a set of smaller jobs, which were completed by individual GPUs. To limit network communication overhead, we sent only basic information about the jobs to the nodes performing the computation, and they calculated the parameters for each of the runs and then performed the computations. On completion of a job, the resulting datafile (stored in a binary format to conserve space) was stored on tape storage provided by the cluster. Throughout the course of the run, we would copy files from that storage to a local hard drive so we had all the files stored locally by the end of the computation.

Several robustness measures were necessary to successfully complete the

computation and transfer of the resulting data. Occasionally a node would never complete a job, blocking completion of the overall computation. To prevent this, we had the lead node reassign abnormally long-running jobs. Further, in our real-time data transfer system, it was sometimes difficult to determine if a file on the tape storage represented a partially or fully transferred datafile. To ensure we only copied complete datafiles to our local hard drive, we added an indicator file to the tape storage to mark that a file had been fully transferred. Last, sometimes our transfers from the tape storage to our local hard drive — using high performance enabled secure copy (HPN SCP) — only partially completed. We validated the copies by comparing the hash values of the datafiles on the tape storage to those on the local hard drive.

By using a GPU supercomputing cluster, we were able to significantly speed our investigation on the phase space of a binary black hole system.

7 Statistical Results for Black Hole Inspirals

As described in [14], we can gain significant insight into the interactions of BBH systems through a systematic sampling and evolution of the parameter space of initial configurations. In particular, in a recent statistical study [11], through a principal component analysis of our simulation results we found quantities which are nearly conserved, both in a statistical sense with respect to parameter variation and as functions of time. Figure 3 shows the variance σ^2 with respect to changes in the initial black hole spin orientations for one such quantity,

$$\Delta\mathcal{E}_2^{\rm SO} = \hat{V}_2^1\,\Delta\left(\hat{\mathbf{S}}_1\cdot\hat{\mathbf{L}}\right) + \hat{V}_2^2\,\Delta\left(\hat{\mathbf{S}}_2\cdot\hat{\mathbf{L}}\right)\,. \tag{5}$$

In the previous expression the superscript SO stands for *spin-orbit* interactions, Δ denotes the difference between final and initial quantities, and $\hat{V}_2^i$ $(i=1,2)$ are the components of the normalized eigenvector of the covariance matrix associated with $\Delta\left(\hat{\mathbf{S}}_1\cdot\hat{\mathbf{L}}\right)$ and $\Delta\left(\hat{\mathbf{S}}_2\cdot\hat{\mathbf{L}}\right)$. These components, as well as the variance of $\Delta\mathcal{E}_2^{\rm SO}$, depend on the black hole masses m_i and spin magnitudes χ_i $(i=1,2)$. In Figure 3 we show the dependence of the variance on these parameters. The range of $\sigma^2 \sim 2\times10^{-4} - -10^{-9}$ gives an indication of how narrowly peaked the probability distribution for this quantity is or,

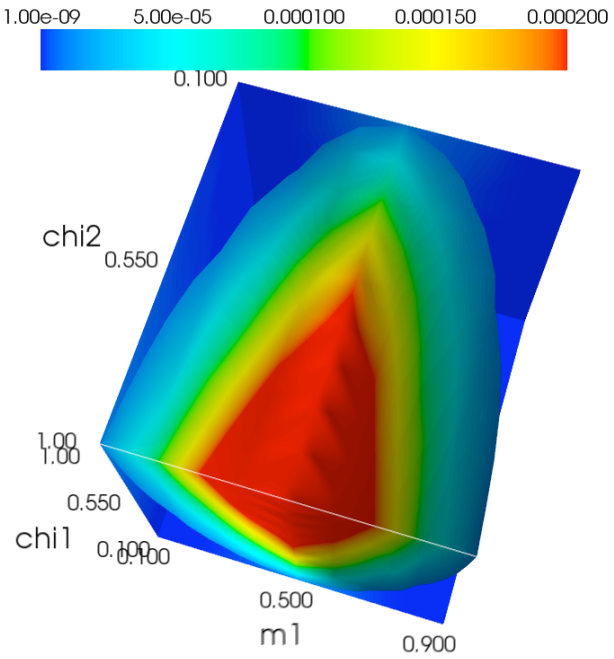


Figure 3: Variance of the principal component in Eq. (5) as a function of the black holes spin magnitudes χ_i $(i = 1, 2)$ and one of the black hole masses, m_1 (the total mass was set to $M = m_1 + m_2 = 1$).

in other words, how well it is conserved with respect to variations in initial spin orientations for different masses and spin magnitudes.

8 Conclusion

In this chapter, we have shown that CUDA and GPUs are well suited for the exploration of the binary black hole system parameter space in the post-Newtonian approximation, enabling significant insights into the binary black hole inspiral phase space. Looking to the future, GPU computing and CUDA have the potential to make a significant impact in the quest for direct measurement of gravitational radiation.

9 Acknowledgments

This work has been supported by NSF Grant PHY0908457 to the University of Maryland and NVIDIA Corporation through a Professor Partnership award. The simulations were carried out at the Teragrid under allocation TG-PHY090080 with some of them using GPUs on the Lincoln cluster.

References

[1] B Abbott et al. LIGO: The Laser Interferometer Gravitational-Wave Observatory. 2007.

[2] F. Acernese et al. Status of Virgo. *Class. Quant. Grav.*, 25:114045, 2008.

[3] Benjamin Aylott et al. Status of NINJA: the Numerical INJection Analysis project. *Class. Quant. Grav.*, 26:114008, 2009.

[4] Benjamin Aylott et al. Testing gravitational-wave searches with numerical relativity waveforms: Results from the first Numerical INJection Analysis (NINJA) project. *Class. Quant. Grav.*, 26:165008, 2009.

[5] M. Boyle, A. Buonanno, L. E. Kidder, A. H. Mroue, Y. Pan, H. P. Pfeiffer, and M. A. Scheel. High-accuracy numerical simulation of black-hole binaries: Computation of the gravitational-wave energy flux and comparisons with post-Newtonian approximants. *Phys. Rev. D*, 78:104020, 2008.

[6] Alessandra Buonanno, Yan-bei Chen, and Michele Vallisneri. Detecting gravitational waves from precessing binaries of spinning compact objects: Adiabatic limit. *Phys. Rev.*, D67:104025, 2003.

[7] Alessandra Buonanno, Yan-bei Chen, and Michele Vallisneri. Erratum: Detecting gravitational waves from precessing binaries of spinning compact objects: Adiabatic limit. *Phys. Rev.*, D74:029905, 2006.

[8] Manuela Campanelli, Carlos O. Lousto, Hiroyuki Nakano, and Yosef Zlochower. Comparison of Numerical and Post-Newtonian Waveforms for Generic Precessing Black-Hole Binaries. *Phys. Rev.*, D79:084010, 2009.

[9] Shin Kee Chung, Linqing Wen, David Blair, Kipp Cannon, and Amitava Datta. Application of Graphics Processing Units to Search Pipeline for Gravitational Waves from Coalescing Binaries of Compact Objects. *Class. Quant. Grav.*, 27:135009, 2010.

[10] J. R. Dormand and P. J. Prince. A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19 – 26, 1980.

[11] Chad R. Galley, Frank Herrmann, John Silberholz, Manuel Tiglio, and Gustavo Guerberoff. Statistical constraints on binary black hole inspiral dynamics. Preprint arXiv:1005.5560. 2010.

[12] Mark Hannam. Status of black-hole-binary simulations for gravitational-wave detection. *Class. Quant. Grav.*, 26:114001, 2009.

[13] Mark Hannam, Sascha Husa, Bernd Bruegmann, and Achamveedu Gopakumar. Comparison between numerical-relativity and post-Newtonian waveforms from spinning binaries: the orbital hang-up case. *Phys. Rev.*, D78:104007, 2008.

[14] Frank Herrmann, John Silberholz, Matias Bellone, Gustavo Guerberoff, and Manuel Tiglio. Integrating Post-Newtonian Equations on Graphics Processing Units. *Class. Quant. Grav.*, 27:032001, 2010.

[15] Benno Willke. GEO600: Status and plans. *Class. Quant. Grav.*, 24:S389–S397, 2007.

[16] Burkhard Zink. A General Relativistic Evolution Code on CUDA Architectures. Technical report, CCT-TR-2008-1, `http://www.cct.lsu.edu/CCT-TR/CCT-TR-2008-1`.