Solving the gravitational N-body problem with machine learning

Saz Ulibarrena, V.

ENGLISH SUMMARY

Few problems in the history of science have drawn as much fascination as the motion of celestial bodies in space. What we now refer to as the N-body problem, has been a subject of study since the beginning of history and remains an active area of research nowadays. One of the main approaches to studying this problem is through numerical simulations.

Astronomers have historically made discoveries by looking through the eyepiece of a telescope. Nowadays, however, scientists can take advantage of the known laws of physics to develop computer programs that mimic how the Universe works. These programs, or simulations, can be used together with observations from telescopes as two faces of the same coin, to compare what is happening against what we think is supposed to happen. Just as the strength of astronomical observations relies on the quality of telescopes and their proper use, the effectiveness of simulations hinges on the numerical tools that are used to create them. The strength of computational astrophysics relies upon the quality of the numerical methods employed.

Our contribution to the field of computational astrophysics is the exploration and creation of new methods that optimize simulations of the gravitational N-body problem. Specifically, we take advantage of the recent, paramount popularity of Machine Learning methods to find tools that can suit our problem.

5.7.1 The gravitational *N*-body problem

Celestial bodies exert gravitational forces on one another following Newton's universal law of gravitation. Knowing the current position and velocity of a celestial body, or a group of celestial bodies, one can predict their future states to gain insights into various astrophysical phenomena such as the planetary formation or the long-term evolution of planetary systems. While the motion of two bodies can be solved analytically using Kepler's equations, the problem becomes significantly more complex for three or more bodies. In such case, the equations of motion must be solved with numerical integrators.

Numerical integration involves discretizing the trajectory of a system and solving the equations of motion step by step. The time between those steps is denominated the time-step size,a decision parameter when setting up a simulation.A smaller time step enhances accuracy by approximating the continuous trajectory more closely but increases computational expense. A large time step can be advantageous to speed up the simulations at the expense of a larger numerical error. In this thesis we find solutions to optimize the trade-off between accuracy and efficiency.

In a system of N-bodies interacting only via their gravitational forces, the total energy and angular momentum are conserved. We can therefore use deviations in these quantities

as an indication of the presence of numerical errors. These errors tend to accumulate over time, potentially leading to unphysical, and therefore unreliable, solutions. This issue is particularly pronounced in chaotic systems, where errors grow exponentially. Most instances of the N-body problem are chaotic for $N \geq 3$, which will influence the choices of numerical integrator and data analysis. To address the accuracy requirements of chaotic problems, many integrators have been designed. For example, some implement a variable time-step criterion that dynamically adapts to the needs of the problem (see Chapter 4), while others take advantage problem-specific characteristics (such as Wisdom-Holman integrator in Chapter 2).

5.7.2 Machine Learning in Astronomy

Numerical integrators have been developed for decades and are currently optimized for different problems. However, despite the work put on perfecting these methods, there is a limit to what they can achieve in terms of performance. The rapid advancement of Machine Learning (ML) in recent years, has led to many new Machine Learning (ML) methods being created. Despite the exceptional interest that the field has generated, most of those new methods are usually tested for the solution of simple equations or for overly-simplified test cases. There are many possible applications of ML methods to scientific fields, but identifying the transferability of those methods from simple applications to complex cases such as the ones found in computational astrophysics is a nontrivial challenge.

This thesis focuses on two types of machine learning techniques: physics-aware neural networks and reinforcement learning. The first are a specific case of supervised learning where physical knowledge is included in the neural network. The latter refers to a group of algorithms that learn to perform actions through trial and error using a reward system. Additionally, we identify other potential applications of machine learning techniques for N-body simulations and the challenges associated with their implementation.

Throughout this work, we identify common obstacles in applying machine learning techniques to astrophysics and propose solutions to address them.

5.7.3 In this thesis

In this thesis we explore the applications of machine learning techniques -particularly physics-aware neural networks and reinforcement learning- to numerical simulations of the gravitational N-body problem.

Chapter 2. We study a planetary system with a large number of small bodies and apply neural networks to replace parts of the integration to speed up the simulations. We use Wisdom Holmann integrator which separates the contributions to the acceleration of a body in two terms: the ones by the central body and the perturbing ones by the other bodies in the system. We adapt two types of neural networks to this case: a deep neural network and a Hamiltonian neural network. While the standard neural networks are easy to set up, they do not lead to results that adhere to the laws of physics and do not extrapolate well to unseen data. In contrast, Hamiltonian Neural Networks managed to find solutions that adhere better to the physics and are capable of extrapolating for longer periods of time.

However, there were also major challenges. Hamiltonian neural networks proved to be difficult to train and apply to problems with large differences in scale. With both networks, we find that prediction errors by the neural networks accumulate and grow exponentially, leading to unphysical solutions. To address this problem, we create a hybrid method that evaluates the network prediction at each step. If the prediction does not satisfy a requirement of accuracy, the calculation is repeated using the analytical equations. Thanks to this method, the integrator can make use of neural networks to speed up the simulation while ensuring robustness in its accuracy.

Chapter 3. We compare different implementations of neural networks with structure-preserving architectures and develop a new type of neural network that represents a generalization of many of those types. SRNNs, SympNets, and HénonNets implement hard physical constraints into their network structure. Those networks can be thought of as specific cases of a new type of structure-preserving neural networks: Generalized Hamiltonian Neural Networks (GHNNs). The performance achieved with those neural networks in solving Hamiltonian systems is compared for different experiments. The results of the experiments show that the performance of the networks depends on the experiment. For simple cases such as the single and double pendulum, networks that included a larger number of simple updates dominated over those with fewer updates. In contrast, in the 3-body problem, networks with more complex updates (such as Deep HénonNets) achieved a better performance. In any case, MultiLayer Perceptrons, a simple type of physics-unaware neural network, showed the worst extrapolation capabilities compared to the physics-aware ones. It is concluded that the added symplecticity in the structure-preserving neural networks resulted in better performances, both inside and outside the training data.

Chapter 4. After studying the advantages and disadvantages of replacing parts of the integration with neural networks, we moved our focus to the use of Machine Learning techniques to choose simulation parameters. When setting up a simulation, lack of expertise makes it common to use the default parameters of the integrator. This can lead to inaccurate solutions or computationally expensive simulations. Thus, we explore the idea of using Reinforcement Learning techniques that eliminate the need for expert knowledge by choosing the time-step parameter. We apply this method to the chaotic 3-body problem. In this case, the bodies might move close to each other, and the interactions will heavily determine the outcome of the simulation. In these moments, the integrator should choose smaller time-step sizes to capture those interactions in detail. When the bodies are far from each other, the time-step size should be increased to save computation time. The time-step parameter directly scales the time-step size chosen at each moment. By allowing this parameter to change dynamically to adapt to the needs of the simulation, we obtain an optimal choice that balances accuracy and computational effort at each time step. We explore the extrapolation of this trained algorithm to other integrators and determine that it can be used without retraining for similar algorithms. Additionally, the method setup can be easily extrapolated without major development changes.

Chapter 5. We show the generalization capabilities of the algorithm designed in Chapter 4 by applying a similar algorithm to a more complex astrophysics problem: the evolution of a star cluster in which some stars have planetary systems. In this case, we use different integrators for the star cluster and the planetary system to adapt to the orders of magnitude difference in their scales. Then, those two different parts are linked using

the Bridge method from AMUSE. The interaction time between both integrators is a fixed parameter that has to be chosen manually in advance. We denominate it the bridge time-step size. Instead of choosing a value and keeping it fixed throughout the simulation, we apply our RL algorithm to allow it to change dynamically to find the optimal choice that balances accuracy and computation time. We find that our algorithm outperforms all of the current options and can adapt to different initializations. Knowing that our system is highly chaotic, we want to create a method that is robust against suboptimal choices of the RL algorithm. Therefore, we create a hybrid method, similar to the one in Chapter 2, that evaluates the prediction of the RL method and reduces the time-step size if considered inadequate. We find that this method leads to improvements of up to orders of magnitude in the energy error without a major increase in computational effort.