Monthly Notices
of the
ROYAL ASTRONOMICAL SOCIETY

MNRAS **494,** 2465–2470 (2020) doi:10.1093/mnras/staa713

Newton versus the machine: solving the chaotic three-body problem using deep neural networks

Philip G. Breen,[1,2]★† Christopher N. Foley,[3]† Tjarda Boekholt[4] and Simon Portegies Zwart[5]

[1]*School of Mathematics and Maxwell Institute for Mathematical Sciences, University of Edinburgh, Kings Buildings, Edinburgh EH9 3JZ, UK*
[2]*Roar AI, 3rd Floor, 116 Dundas Street, Edinburgh EH3 5DQ, UK*
[3]*School of Clinical Medicine, University of Cambridge, Cambridge, CB2 0SP, UK*
[4]*CIDMA, Departamento de Física, Universidade de Aveiro, Campus de Santiago, P-3810-193 Aveiro, Portugal*
[5]*Leiden Observatory, Leiden University, PO Box 9513, NL-2300 RA Leiden, the Netherlands*

Accepted 2020 March 1. Received 2020 February 28; in original form 2019 October 16

ABSTRACT

Since its formulation by Sir Isaac Newton, the problem of solving the equations of motion for three bodies under their own gravitational force has remained practically unsolved. Currently, the solution for a given initialization can only be found by performing laborious iterative calculations that have unpredictable and potentially infinite computational cost, due to the system's chaotic nature. We show that an ensemble of converged solutions for the planar chaotic three-body problem obtained using an arbitrarily precise numerical integrator can be used to train a deep artificial neural network (ANN) that, over a bounded time interval, provides accurate solutions at a fixed computational cost and up to 100 million times faster than the numerical integrator. In addition, we demonstrate the importance of training an ANN using converged solutions from an arbitrary precise integrator, relative to solutions computed by a conventional fixed precision integrator, which can introduce errors in the training data, due to numerical round-off and time discretization, that are learned by the ANN. Our results provide evidence that, for computationally challenging regions of phase space, a trained ANN can replace existing numerical solvers, enabling fast and scalable simulations of many-body systems to shed light on outstanding phenomena such as the formation of black hole binary systems or the origin of the core collapse in dense star clusters.

Key words: methods: numerical – methods: statistical.

1 INTRODUCTION

Newton's equations of motion describe the evolution of many bodies in space under the influence of their own gravitational force (Newton 1687). The equations have a central role in many classical problems in physics. For example, the equations explain the dynamical evolution of globular star clusters and galactic nuclei, which are thought to be the production sites of tight black hole binaries that ultimately merge to produce gravitational waves (Portegies Zwart & McMillan 2000). The fate of these systems depends crucially on the three-body interactions between black hole binaries and single black holes (e.g. see Breen & Heggie 2013a, b; Samsing & D'Orazio 2018), often referred to as close encounters. These events typically occur over a fixed time interval and, owing to the tight interactions between the three nearby bodies, the background influence of the other bodies can be ignored; i.e. the trajectories of three bodies can be generally computed in isolation (Portegies Zwart & McMillan 2018). The focus of this study is therefore the timely computation of accurate solutions to the three-body problem.

Despite its age and interest from numerous distinguished scientists (de Lagrange 1772; Heggie 1975; Hut & Bahcall 1983; Montgomery 1998; Boekholt & Portegies Zwart 2015; Stone & Leigh 2019), the problem of solving the equations of motion for three bodies remains impenetrable due to the system's chaotic nature (Valtonen et al. 2016), which typically renders the identification of solutions feasible only through laborious numerical integration. Analytic solutions exist for several special cases (de Lagrange 1772) and a solution to the problem for all time has been proposed (Valtonen et al. 2016), but this is based on an infinite series expansion and has limited use in practice. The computation of a numerical solution, however, can require holding an exponentially growing number of decimal places in memory and using a time-step that approaches zero (Boekholt, Portegies Zwart & Valtonen 2019). Integrators that do not allow for this often fail spectacularly, meaning that a single numerical solution is unreliable whereas the average of an ensemble of numerical solutions appears valid in

★ E-mail: philipbreen@gmail.com (PGB); cnf25@cam.ac.uk (CNF)
† These authors contributed equally.

a statistical sense, a concept referred to as nagh Hoch (Portegies Zwart & Boekholt 2018). To overcome these issues, the Brutus integrator was developed (Boekholt & Portegies Zwart 2015), allowing for close-to-zero time-steps and arbitrary precision. Brutus, which is part of the Astrophysical Multipurpose Framework (Portegies Zwart & McMillan 2018), is capable of computing converged solutions to any gravitational N-body problem; however, the process is laborious and can be extremely prohibitive in terms of computer time. In general, there does not exist a theoretical framework capable of determining a priori the precision required to deduce that a numerical solution has converged for an arbitrary initialization (Stone & Leigh 2019). This makes the expense of acquiring a converged solution through brute-force integration unpredictable and regularly impractical.

Here we demonstrate that, over a fixed time interval, the planar three-body problem can be solved by means of a multilayered deep artificial neural network (ANN; e.g. see LeCun, Bengio & Hinto 2015). These networks are designed for high-quality pattern recognition by mirroring the function of our brains (McCulloch & Pitts 1943; Rosenblatt 1985) and have been successfully applied to a wide variety of pattern recognition problems in science and industry, even mastering the game of Go (Silver et al. 2016). The abundance of real-world applications of ANNs is largely a consequence of two properties: (i) an ANN is capable of closely approximating any continuous function that describes the relationship between an outcome and a set of covariates, known as the universal approximation theorem (Cybenko 1989; Hornik 1991); and (ii) once trained, an ANN has a predictable and a fixed computational burden. Together, these properties lead to the result that an ANN can be trained to provide accurate and practical solutions to Newton's laws of motion, resulting in major improvements in computational economy (Lee, Sode-Yome & Park 1991) relative to modern technologies. Our proof-of-principle method shows that an ANN can accurately match the results of converged solutions found using the arbitrary precision numerical integrator that, for computationally challenging scenarios, e.g. during multiple close encounters, can offer numerical solutions at a fraction of the time cost and CO_2 expense. We demonstrate the importance of training an ANN on converged solutions. This enables the trained ANN to accurately predict particle locations even when a conventional 'double-precision' numerical integrator fails dramatically. By training an ANN that can accurately compute particle trajectories during close encounters, our work extends previous work training neural networks on an n-body-type problem (e.g. Quito, Monterola & Saloma 2001; Battaglia et al. 2016). Our findings also add to the growing body of literature that supports machine learning technologies being developed to enrich the assessment of chaotic systems (Pathak et al. 2018; Stinis 2019) and providing alternative approaches to classical numerical solvers more broadly (Hennig, Osborne & Girolami 2015).

2 METHOD

Every ANN requires a learning phase, where parameters in an adaptive model are tuned using a training data set, which renders prediction accuracy sensitive to whether the training set is representative of the types of patterns that are likely present in future data applications. Training an ANN on a chaotic problem therefore requires an ensemble of solutions across a variety of initializations. The only way to acquire such a training set is by numerically integrating the equations of motion for a large and diverse range of realizations until a converged solution is acquired, which we do

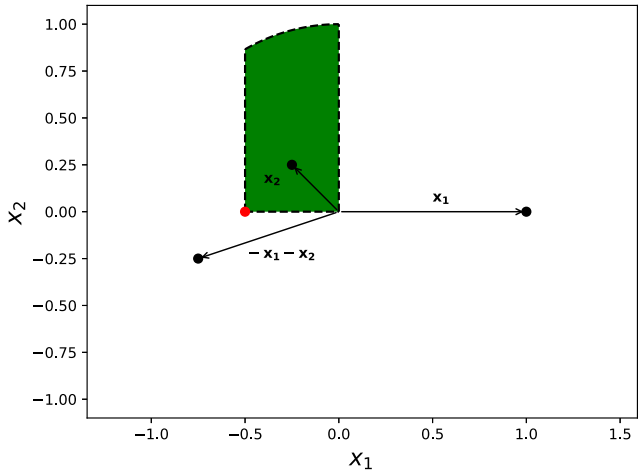

Figure 1. Visualization of the initial particle locations. The origin is taken as the barycentre and the unit of length is chosen to be the distance to the most distant particle x_1, which also orientates the x-axis. The closest particle to the barycentre, labelled x_2, is chosen to orientate the positive y-axis and can be located anywhere in the green region. Once specified, the location of the remaining particle x_3 is deduced by symmetry. There is a singular point at $(-0.5, 0)$, red point, where the positions of x_2 and x_3 are identical. Numerical schemes can fail near the point as the particles are on near-collision orbits, i.e. passing arbitrarily close to one another.

using Brutus (an arbitrary precise N-body numerical integrator). We give an example in Appendix C that demonstrates the importance of training the ANN using solutions derived from an arbitrary precise integrator.

We restricted the training set to the gravitational problem of three equal-mass particles with zero initial velocity, located in a plane. The three particles, with Cartesian coordinates x_1, x_2, x_3, are initially positioned at $x_1 \equiv (1, 0)$ with (x_2, x_3) randomly situated somewhere in the unit semicircle in the negative x-axis, i.e. $x \leq 0$. The reference frame is taken as the centre of mass and, without loss of generality, we orientate the positive y-axis using the particle closest to the barycentre (Fig. 1). In this system, the initial location of only one of (x_2, x_3) needs be specified, as the location of the remaining particle is deduced by symmetry. In addition, we adopt dimensionless units in which $G = 1$ (Heggie & Mathieu 1986). The physical set-up allows the initial conditions to be described by two parameters and the evolution of the system by three parameters (representing the coordinates of x_1 and x_2 at a given time). The general solution is found by mapping the three-dimensional phase space (time t and initial coordinate of x_2) to the positions of particles x_1 and x_2; the position of particle x_3 follows from symmetry.

The training and validation data sets are composed of 9900 and 100 simulations, respectively. In each simulation, we randomly generated initial locations for the particles and computed trajectories, typically for up to 10 time-units (of roughly a dynamical crossing time each), by integrating the equations of motion using Brutus. Each trajectory comprises a data set of some 2561 discrete time points (labels); hence, the validation data set contained over 10^5 time points. A converged solution was acquired by iteratively reducing two parameters during integration: (i) the tolerance parameter (ϵ), controlling accuracy, that accepts convergence of the Bulirsch–Stoer multistep integration scheme (Bulirsch & Stoer 1964) and (ii) the word length (L_w) measured in bits, which controls numerical precision (Boekholt & Portegies Zwart 2015). Our ensemble of initial realizations all converged for values of $\epsilon = 10^{-11}$ and $L_w =$

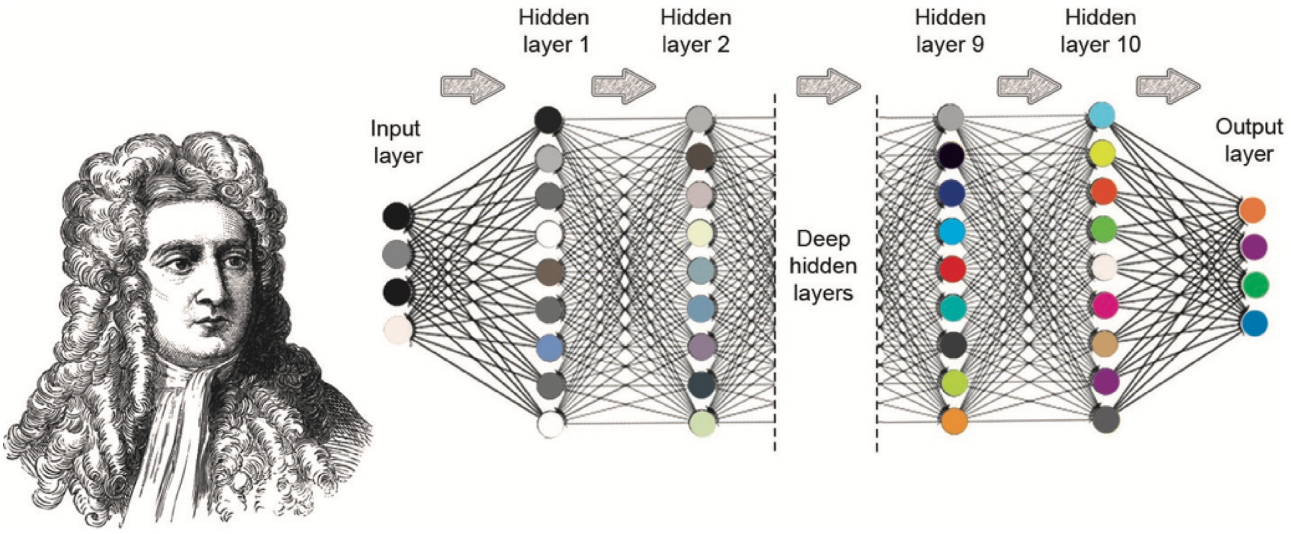

Figure 2. Newton and the machine. Image of sir Isaac Newton alongside a schematic of a 10-layer deep neural network. In each layer (apart from the input layer), a node takes the weighted input from the previous layer's nodes (plus a bias) and then applies an activation function before passing data to the next node. The weights (and bias) are free parameters that are updated during training.

128 (see Appendix A). Generating these data required over 10 d of computer time. Some initializations gave rise to very close encounters between the particles, e.g. mirroring a close encounter, and the computation of converged solutions in these situations is costly[1] (Boekholt et al. 2019).

We used a feed-forward ANN consisting of 10 hidden layers of 128 interconnected nodes (Fig. 2 and Appendix B). Training was performed using the adaptive moment estimation optimization algorithm ADAM (Kingma & Ba 2015) with 10 000 passes over the data, in which each epoch was separated into batches of 5000, and setting the rectified linear unit (ReLU) activation function to max (0, x) (Glorot, Bordes & Bengio 2011). By entering a time t and the initial location of particle x_2 into the input layer, the ANN returns the locations of the particles x_1 and x_2 at time t, thereby approximating the latent analytical solution to the general three-body problem.

To assess the performance of the trained ANN across a range of time intervals, we partitioned the training and validation data sets into three segments: $t \lesssim 3.9$, 7.8, and 10 (which includes all data). For each scenario, we assessed the loss function [taken as the mean absolute error (MAE)] against epoch. Examples are given in Fig. 3. In all scenarios, the loss in the validation set closely follows the loss in the training set. We also assessed sensitivity to the choice of activation function; however, no appreciable improvement was obtained when using either the exponential rectified (Clevert, Unterthiner & Hochreiter 2011) or leaky rectified (Maas, Hannun & Ng 2013) linear unit functions. In addition, we assessed the performance of other optimization schemes for training the ANN, namely an adaptive gradient algorithm (Duchi, Hazan & Singer 2011) and a stochastic gradient descent method using Nesterov momentum, but these regularly failed to match the performance of the ADAM optimizer.

[1]We note that identifying converged solutions for initial conditions near the singular point (0.5, 0) proved challenging. They result in very close encounters between two particles, which could not be resolved within the predetermined precision. Brutus could have resolved these trajectories with higher precision; however, this could result in even more lengthy computation time.

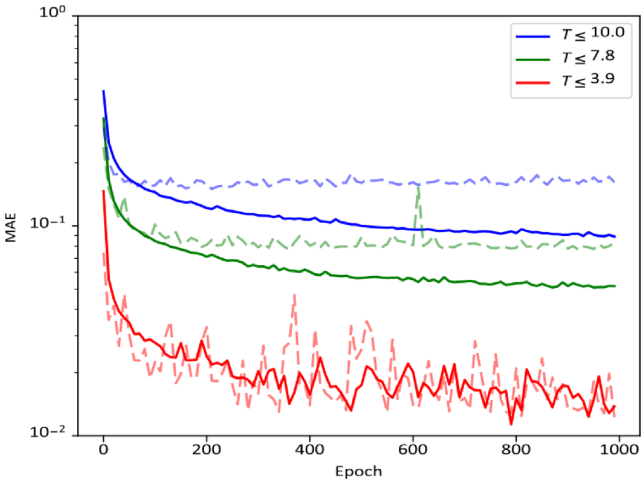

Figure 3. Mean absolute error (MAE) versus epoch. The ANN has the same training structure in each time interval. The solids lines are the loss on the training set and dashed are the loss on the validation set. $T \leq 3.9$ corresponds to 1000 labels per simulation, similarly $T \leq 7.8$ to 2000 labels, and $T \leq 10.0$ to 2561 labels/time points (the entire data set). The results illustrate a typical occurrence in ANN training; there is an initial phase of rapid learning, e.g. about 100 epochs, followed by a stage of much slower learning in which relative prediction gains are smaller with each epoch.

The best-performing ANN was trained with data from $t \lesssim 3.9$ (Fig. 3). We give examples of predictions made from this ANN against converged solutions within the training set (Fig. 4, left) or the validation set (Fig. 4, right). In each scenario, the particle trajectories reflect a series of complex interactions and the trained ANN reproduced these satisfactorily (MAE $\leq$0.1). The ANN also closely matched the complicated behaviour of the converged solutions in all the scenarios that were not included in its training. Moreover, the ANN did this in fixed computational time ($t \sim 10^{-3}$ s), which is on average about 10^5 (and sometimes even 10^8) times faster than Brutus.

We consider the ability of the ANN to emulate a key characteristic of the chaotic three-body system: a sensitive dependence to initial

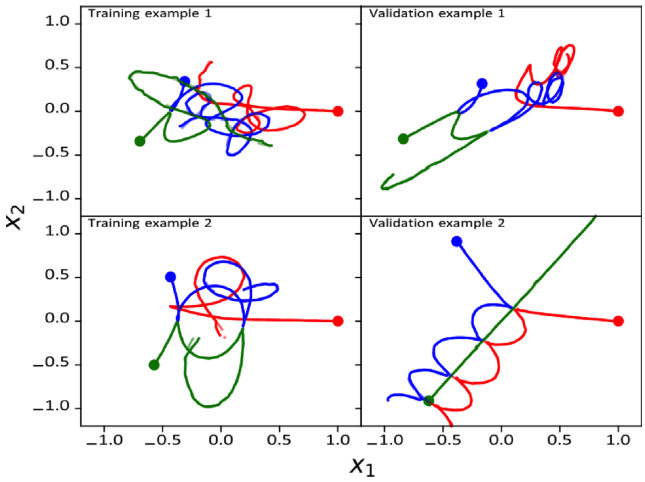

Figure 4. Validation of the trained ANN. Presented are two examples from the training set (left) and two from the validation set (right) from the ANN trained on data $T \leq 3.9$. All examples were randomly chosen from their data sets. The bullets indicate the initial conditions. The curves represent the orbits of the three bodies (red, blue, and green, the latter obtained from symmetry). The solution from the trained network (solid curves) is hardly distinguishable from the converged solutions [dashes, acquired using Brutus (Boekholt & Portegies Zwart 2015)]. The two scenarios presented to the right were not included in the training data set.

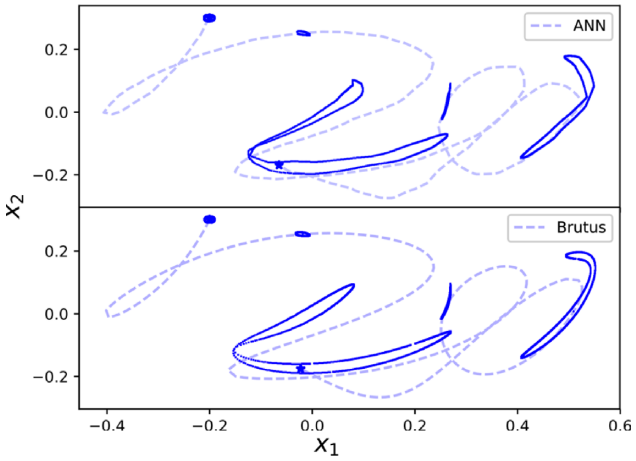

Figure 5. Visualization of the sensitive dependence on initial position. Presented are trajectories from 1000 random initializations in which particle x_2 is initially situated on the circumference of a ring of radius 0.01 centred at $(-0.2, 0.3)$. For clarity, these locations were benchmarked against the trajectory of x_2 initially located at the centre of the ring (hatched line); the star denotes the end of this trajectory after 3.8 time-units. None of these trajectories were part of the training or validation data sets. The locations of the particles at each of the five time points, $t \in \{0.0, 0.95, 1.9, 2.95, 3.8\}$, are computed using either the trained ANN (top) or Brutus (bottom) and these solutions are denoted by the bold line. The results from both methods closely match one another and illustrate a complex temporal relationship that underpins the growth in deviations between particle trajectories, owing to a change in the initial position of x_2 on the ring.

conditions. We illustrate this in two ways and in each case, we generate new scenarios that are not included in either the training or validation data sets. First, we estimated the Lyapunov exponent across 4000 pairs of randomly generated realizations using the simulation framework described previously. The realizations within each pair differed due to a small random change (of $\delta = 10^{-6}$ in both coordinate axes[2]) in the initial location of particle x_2. The trajectories were computed across two time-units and the first, fifth (median), and ninth deciles of the estimated Lyapunov exponent were (0.72, 1.30, and 2.26), indicating some divergence between pairs of initializations. Secondly, we generated 1000 realizations in which particle x_2 was randomly situated somewhere on the circumference of a ring of radius 0.01 centred at $(-0.2,0.3)$ and computed the location of the particle for up to 3.8 time-units, using both the trained ANN and Brutus (Fig. 5). Our findings highlight the ability of the ANN to accurately emulate the divergence between nearby trajectories, and closely match the simulations obtained using Brutus (notably outside of the training scenarios).

We propose that for computationally challenging areas of phase space, our results support replacing classic few-body numerical integration schemes with deep ANNs. To strengthen this claim further, we assessed the ANN's ability to preserve a conserved quantity, taken as the initial energy in the system, as this is an important measure of the performance of a numerical integration scheme. To do this, we required the velocities of the particles. Our ANN was trained to recover the positions of the particles for a given time, the results from which can be used to estimate the velocities by differentiating the network. Instead, we trained a second ANN to produce the velocity information. A typical example of the relative energy error is shown in Fig. 6. In general, the errors are of the order of 10^{-2}; however, these can spike to 10^{1} during close encounters between the bodies in which case energy is highly sensitive to position. Improvements are achieved by adding a projection layer to the ANN (see Appendix D), which projects the phase-space coordinates on to the correct energy surface, thereby reducing errors down to around 10^{-5}. The approach is similar to solving an optimization problem that aims to find a change in coordinates, which reduces the energy error while also remaining close to the coordinates predicted by the ANN.

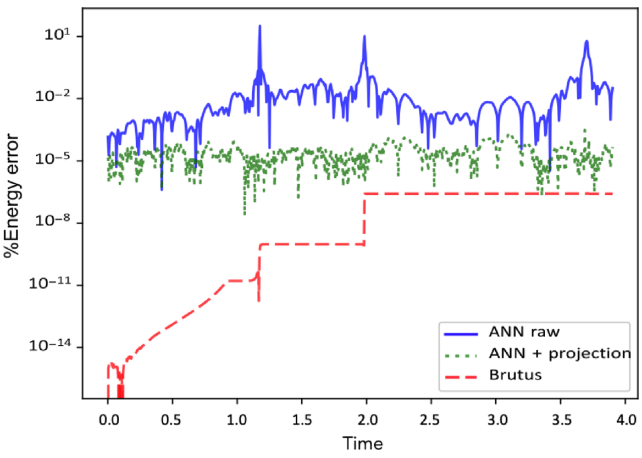

Figure 6. Relative energy error. An example of the relative energy error for a typical simulation. The raw output of the two ANN's typically have errors of around 10^{-2}; after projecting on to a nearby energy surface, the error reduces down to order 10^{-5}.

[2] $\delta = 10^{-6}$ was identified as the minimum distance between a pair of initializations that allowed for the estimation of the Lyapunov exponent and avoided falling below the minimum resolution required to distinguish between a pair of trajectories (owing to the implicit error in the ANN).

3 DISCUSSION

We have shown that deep ANNs produce fast and accurate solutions to the computationally challenging three-body problem over a fixed time interval. Despite the simplifications in our initial set-up, the particle trajectories regularly undergo a series of complex interactions and the ANN captures this behaviour, matching predictions from arbitrarily accurate and precise numerical integrations at a fraction of the computational time cost. For regions of phase space in which a traditional integrator fails to compute a numerical solution within a pre-specified time tolerance, it is possible that the trained ANN can be used to provide accurate predictions of the particle's locations away from the present computationally challenging region. These predictions can then be used as input variables to restart the traditional integrator at a future time point. This idea would see a hybrid numerical integrator developed that combines the traditional integrator with the trained ANN – to calculate particle trajectories local to regions in which the traditional integrator is computationally cumbersome – so that accurate and timely solutions to the three-body problem are obtained over a wider variety of scenarios than what is currently achievable.

Three-body interactions, e.g. between a black hole binary and a single black hole, can form the main computational bottleneck in simulating the evolution of globular star clusters and galactic nuclei. As these events occur over a fixed time length, during which the three closely interacting bodies can be integrated independently of the other bodies comprising the cluster or nuclei, we have demonstrated, within the parameter space considered, that a trained ANN can be used to rapidly resolve these three-body interactions and therefore help towards more tractable and scalable assessments of large systems.

With our success in accurately reproducing the results of a chaotic system, we are encouraged that other problems of similar complexity can be addressed effectively by replacing classical differential solvers with machine learning algorithms trained on the underlying physical processes. Our next steps are to expand the dynamic range and relax some of the assumptions adopted, regarding symmetry and mass equality, in order to construct a network that can solve the general three-body problem. From there, we intend to replace the expensive three-body solvers in star cluster simulations with the network and study the effect and performance of such a replacement. A hybrid numerical integrator, which would employ the ANN iteratively, might be capable of efficiently computing the evolution of the three-body system over an arbitrary time period. To go some way towards validating this, it would be interesting to explore (i) the importance of conserved quantities that are not enforced when training the ANN, e.g. how well does the ANN conserve angular momentum?, and (ii) whether a hybrid scheme conforms to nagh Hoch (Portegies Zwart & Boekholt 2018). Eventually, we envision that network may be trained on richer chaotic problems, such as the four- and five-body problems, reducing the computational burden even more.

ACKNOWLEDGEMENTS

It is a pleasure to thank Mahala Le May for the illustration of Newton and the machine in Fig. 2 and Maxwell Cai for discussions. The calculations were performed using the LGM-II (NWO grant # 621.016.701), the Leiden supercomputer ALICE, and the Edinburgh Compute and Data Facility cluster Eddie (http://www.ecdf.ed.ac.uk/). In this work, we use the MATPLOTLIB (Hunter 2007), NUMPY (Oliphant 2006), AMUSE (Portegies Zwart et al. 2013; Portegies Zwart & McMillan 2018), Tensorflow (Abadi et al. 2016), and Brutus (Boekholt & Portegies Zwart 2015) packages. PGB acknowledges support from the Leverhulme Trust (Research Project Grant RPG-2015-408). TB acknowledges support from Fundação para a Ciência e a Tecnologia (FCT), within project UID/MAT/04106/2019 (CIDMA) and SFRH/BPD/122325/2016.

REFERENCES

Abadi M. et al., 2016, preprint (arXiv:1603.04467)
Battaglia P. W., Pascanu R., Lai M., Rezende D., Kavukcuoglu K., 2016, Interaction Networks for Learning about Objects, Relations, and Physics, NIPS
Boekholt T., Portegies Zwart S., 2015, Comput. Astrophys. Cosmol., 2, 2
Boekholt T., Portegies Zwart S., Valtonen M., 2019, MNRAS, 493, 3932
Breen P. G., Heggie D. C., 2013a, MNRAS, 432, 2779
Breen P. G., Heggie D. C., 2013b, MNRAS, 436, 584
Bulirsch R., Stoer J., 1964, Numer. Math., 6, 413
Clevert D., Unterthiner T., Hochreiter S., 2015, Conference paper at ICLR 2016, preprint (arXiv:1511.07289)
Cybenko G., 1989, Math. Control Signals Syst., 2, 303
de Lagrange J.-L., 1772, Chapitre II: Essai sur le Probleme des Trois Corps. Œuvres de Lagrange, 6, 229)
Duchi J., Hazan E., Singer Y., 2011, J. Mach. Learn. Res., 12, 2121
Hennig P., Osborne M. A., Girolami M., 2015, Proc. R. Soc. A, 471, 20150142
Glorot X., Bordes A., Bengio Y., 2011, Proc. Fourteenth Int. Conf. Artif. Intell. Stat., AISTATS 2011. Fort Lauderdale, USA, p. 315
Heggie D. C., 1975, MNRAS, 173, 729
Heggie D. C., Mathieu R. D., 1986, in Hut P., McMillan S. L. W., eds, Lecture Notes in Physics, Vol. 267, The Use of Supercomputers in Stellar Dynamics. Springer-Verlag, Berlin, p. 233
Hornik K., 1991, Neural Netw., 4, 251
Hut P., Bahcall J. N., 1983, ApJ, 268, 319
Hunter J. D., 2007, Comput. Sci. Eng., 9, 90
Kingma D. P., Ba J., 2015, 3rd Int. Conf. for Learning Representations, San Diego (CoRR, abs/1412.6980)
LeCun Y., Bengio Y., Hinton G., 2015, Nature, 521, 436
Lee K. Y., Sode-Yome A., Park J. H., 1998, IEEE Trans. Power Syst., 13, 519
Maas A. L., Hannun A. Y., Ng A. Y., 2013, 30th Int. Conf. Mach. Learn. (ICML), Rectifier Nonlinearities Improve Neural Network Acoustic Models, Atlanta, Georgia
McCulloch W., Pitts W., 1943, Bull. Math. Biophys., 7, 115
Miller R. H., 1964, ApJ, 140, 250
Montgomery R., 1998, Nonlinearity, 11, 363
Newton I., 1687, Philosophiae Naturalis Principia Mathematica, Benjamin Motte, London
Oliphant T. E., 2006, A Guide to NumPy, Vol. 1. Trelgol Publishing, USA
Pathak J., Hunt B., Girvan M., Lu Z., Ott E., 2018, Phys. Rev. Lett., 120, 024102
Portegies Zwart S. F., Boekholt T. C., 2018, Commun. Nonlinear Sci. Numer. Simul., 61, 160
Portegies Zwart S. F., McMillan S. L. W., 2000, ApJ, 528, L17
Portegies Zwart S., McMillan S., 2018, Astrophysical Recipes: The Art of AMUSE. AAS IOP Astronomy, IOP Publishing
Portegies Zwart S., McMillan S. L. W., van Elteren E., Pelupessy I., de Vries N., 2013, Comput. Phys. Commun., 183, 456
Quito M., Monterola C., Saloma C., 2001, Phys. Rev. Lett., 86, 4741
Rosenblatt F., 1958, Psychol. Rev., 65, 386
Samsing J., D'Orazio D. J., 2018, MNRAS, 481, 5445
Silver D. et al., 2016, Nature, 529, 484

Stinis P., 2019, Enforcing Constraints for Time Series Prediction in Supervised, Unsupervised and Reinforcement Learning
Stone N. C., Leigh N. W. C., 2019, Nature, 576, 406
Valtonen M. et al., 2016, The Three-body Problem from Pythagoras to Hawking. Springer Int. Publishing, Switzerland

APPENDIX A: TUNING AND ASSESSING BRUTUS PERFORMANCE PARAMETERS

The Brutus integrator is sensitive to the choice of two tuning parameters: (i) the tolerance parameter (ϵ), which accepts convergence of the Bulirsch–Stoer method (Bulirsch & Stoer 1964) and (ii) the word length ($L_{\rm w}$), which controls numerical precision. To account for this, interactions with the same initial condition and two choices for the pair ϵ, $L_{\rm w}$ were performed ($\epsilon = 10^{-11}$, $L_{\rm w} = 128$ and $\epsilon = 10^{-10}$, $L_{\rm w} = 88$). We then calculated the average phase distances $\delta_{a,b}$ between two solutions a and b (Miller 1964), i.e.

$$\delta^2_{a,b} = \frac{1}{12}\sum_{i=1}^{3}\sum_{j=1}^{4}\left(q_{\{a,i,j\}} - q_{\{b,i,j\}}\right)^2, \tag{A1}$$

to assess sensitivity between the choices over the phase-space coordinates $q_{\{\cdot,i,j\}}$, where i denotes a particle and j its position or velocity. Note that δ is equivalent to the mean squared error of the phase-space coordinate. Converged solutions were identified when the average phase distance was <0.1 over the duration of a simulation. In over 90 per cent of simulated scenarios, we identified converged solutions; exceptions were generally owing to particles initialized near the singularity (Fig. 1). We also noted sensitivity to the maximum time-step assumed; however, we found good resolution of the trajectories when returning results every 2^{-8} time units.

APPENDIX B: DEEP ANN

Our ANN consisted of 10 densely connected layers with 128 nodes using the $\max(0, x)$ (ReLu) activation function; the final output layer used a linear activation function. We considered a variety of ANN architectures. Starting with a network consisting of 5 hidden layers and 32 nodes, we systematically increased these values until the ANN accurately captured the complex trajectories of the particles, which we identified as a network containing 10 hidden layers with 128 nodes. We further assessed performance using an ANN with transposed convolution layers (sometimes referred to as a de-convolution layer), which allows for parsimonious projections between a medium and high-dimensional parameter space. However, performance, as measured by the MAE, was poorer under these networks.

APPENDIX C: EMULATING AN ARBITRARY PRECISION VERSUS DOUBLE-PRECISION NUMERICAL INTEGRATOR

The converged set of solutions from Brutus required over 10 d of computer time. This was largely a consequence of computationally challenging scenarios in which there were multiple close encounters between the particles. In these situations, the computation of a converged solution typically required more than the conventional $L_{\rm w} = 64$ bits of precision, i.e. 'double precision'. In Fig. C1, we provide an example of this, showing that, after a close encounter between particle x_2 and one of the other particles, a conventional double-precision numerical integrator fails dramatically whereas the ANN – trained on (arbitrary precise) converged solutions – continues to furnish accurate solutions. Note that no part of the (blue) Brutus trajectory appeared in the training data set for the ANN, i.e. the ANN is performing out-of-sample prediction. The result highlights the need to acquire converged solutions using Brutus in order to train an ANN.

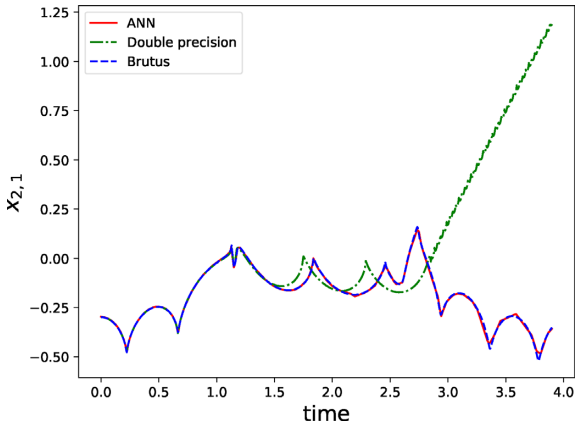

Figure C1. Arbitrary precise trained ANN versus conventional double-precision integrator. Presented are computations of the location of the first dimension, i.e. the horizontal location, of particle x_2 across 4 time units and using three methods: (i) the arbitrary precise Brutus; (ii) a precision restricted Brutus, i.e. using the Brutus integrator with the numerical precision fixed to double precision ($L_{\rm w} = 64$ bits) – referred to as 'double precision' above; and (iii) the ANN, trained using (arbitrary precise) converged solutions. Note the presence of a close interaction between particle x_2 and at least one other particle at $t \approx 1.1$, eventually predicting the particle escaping. However, the ANN – which is performing out-of-sample prediction here – closely matches the converged solution computed using Brutus.

APPENDIX D: PROJECTION LAYER

To better preserve a conserved physical quantity, e.g. energy, during the training of a single ANN, we introduced a projection layer. The projection layer adjusted the coordinates by minimizing the following optimization problem:

$$Er(x, v)^2 + \gamma_1 D_x(x)^2 + \gamma_2 D_v(v)^2, \tag{D1}$$

where $Er(x, v)$ is the energy error, and D_x (D_v) is the distance from the initial position (initial velocity) produced by the ANN. Additionally, γ_1 and γ_2 are constants that penalize deviation from the initial values of x and v, respectively. The optimization problem was solved using the Nelder–Mead method. Instead of this, a training metric, e.g. a fixed multiple of the MAE, can be introduced to bound the error of a prediction. If a single ANN was trained to predict both position and velocity information for each particle, an alternative strategy would be to introduce a penalty term in the cost function during training (similar to a regularization process).

This paper has been typeset from a TEX/LATEX file prepared by the author.