Chaotic Dynamics in N-body systems

Boekholt, T.C.N.

2

A Precise N-body Code: Brutus

Based on: *On the Reliability of N-body Simulations* by T. C. N. Boekholt and S. F. Portegies Zwart in Computational Astrophysics and Cosmology, Volume 2, article id. #2, 21 pp. (2015), Ch.1-3

The general consensus in the N-body community is that statistical results of an ensemble of collisional N-body simulations are accurate, even though individual simulations are not. A way to test this hypothesis is to make a direct comparison of an ensemble of solutions obtained by conventional methods with an ensemble of true solutions. In order to make this possible, we wrote an N-body code called Brutus, that uses arbitrary-precision arithmetic. In combination with the Bulirsch–Stoer method, Brutus is able to obtain converged solutions, which are true up to a specified number of digits.

In this chapter we present the structure of Brutus and illustrate the method of convergence by applying it to several small-N systems. For the first time, we can exactly determine how accurate our N-body results are, and obtain true solutions to general N-body configurations. Finally, we construct a model for the scaling of Brutus with N, for converged solutions to reach core collapse. We conclude that it scales roughly exponentially, which is effectively caused by the exponential divergence between neighbouring solutions.

2.1 INTRODUCTION

Analytical solutions to the N-body problem are known for $N = 2$, which are the familiar conic sections. Also, for several systems possessing symmetries, analytical solutions have been found, for example the equilateral triangle (Lagrange, 1772). For a more general initial configuration, solutions have to be obtained by means of numerical integration. Given an initial N-body realisation, one can calculate all mutual forces and subsequently the net acceleration of each particle.

Different integration methods exist which take the accelerations, and update the positions and velocities to a time $t + \Delta t$, with Δt the time-step size. This process is repeated until the end time is reached.

? recognised that obtaining the solution to an N-body problem by numerical integration is difficult. This is caused by exponential divergence. Consider a certain N-body problem, i.e. N point-particles, each with a given mass, position and velocity. This system evolves with time in a definite and unique way. If one goes back to the initial state and slightly perturbs only one coordinate of a single particle, the perturbed N-body problem will also have a definite and unique but different solution than the original one. When the two solutions are compared as a function of time, it is observed that differences can grow exponentially (**????**). If the initial perturbation is due to a numerical error, the calculated solution will also diverge away from the true solution.

Several authors have estimated the time-scale of this divergence (**??**), and arrived at an e-folding time-scale of the order a dynamical, crossing time. Simulation times of interest are typically much longer than a crossing time and therefore staying close to the true solution is numerically challenging.

If the result of a direct N-body simulation of for example a star cluster, has diverged away from the true solution, the result may well be meaningless (**?**). The general consensus however, is that statistically the results are representative for the true solution to the N-body problem (**???**). The underlying idea is that the statistics of an ensemble of N-body simulations are representative for the true statistics, obtained by an ensemble of true solutions, with the same set of initial conditions. We regard this the hypothesis we want to test.

One way to test this hypothesis is to directly compare statistics obtained by conventional methods, with the statistics obtained from an ensemble of true solutions (see Chapter 4). To obtain true solutions, we wrote an N-body code which can solve the N-body problem to arbitrary precision.

Such a code can be realised if the different sources of error are controlled. The error has contributions from the time discretisation of the integrator and the round-off due to the limited precision of the computer (**?**). Another possible source of error is in the initial conditions, for example the configuration of the solar system is only approximately known (Ito & Tanikawa, 2002). However, if the initial condition is a random realisation of a distribution function, this is less often a problem. Using the Bulirsch–Stoer method (**??**), the discretisation error can be controlled to stay within a specified tolerance. Using arbitrary-

precision arithmetic instead of conventional double-precision or single-precision, the round-off error can be reduced by increasing the number of digits.

We obtain converged solutions to the N-body problem by decreasing the Bulirsch–Stoer tolerance and increasing the number of digits systematically. We define a converged solution in our experiments as a solution for which the first specified number of decimal places of every phase-space coordinate in our final configuration in the N-body experiment becomes independent of the length of the mantissa and the Bulirsch–Stoer tolerance. We explain the method of convergence in Sec. 2.2, we give examples of the procedure in Sec. 2.3 and we measure the scaling of Brutus in Sec. 2.4.

2.2 METHODS

2.2.1 The Benchmark Integrator

The gravitational N-body problem aims to solve Newton's equations of motion under gravity for N stars (Newton, 1687). A popular integrator to perform this task is the fourth-order Hermite predictor-corrector scheme (**?**), using double-precision arithmetic. The experiments we discuss in Sec. 4.1 will use this integrator as a benchmark test. We adopt a shared, adaptive time-stepping scheme with the following criterion:

$$\Delta t = \eta \min \sqrt{\Delta r_{ij} / \Delta a_{ij}}. \tag{2.1}$$

Here η is the time-step parameter and Δr_{ij} and Δa_{ij} are the relative distance and acceleration for the pair of particles i and j. We implement no further constraints on the time-step size.

To test how inaccurate we are allowed to integrate while still obtaining accurate statistics (**??**) we vary the time-step parameter η, to obtain statistics from conventional simulations with different precision.

2.2.2 The Brutus N-body Code

The results obtained with the benchmark integrator will be compared to those obtained with Brutus (see Chapter 4), which uses an arbitrary-precision library [1]. With this library we can specify the number of bits, L_w, used to store the mantissa, while the exponent has a fixed word-length. The length of the mantissa can be specified and increased, with the aim of controlling the round-off error.

[1] We use the open-source library GMP: http://gmplib.org/

The integration of the equations of motion is realised using the Verlet-Leapfrog scheme (**?**). The time-step is shared among all particles, but varies for every step according to Eq. 2.1.

To control the discretisation error, we implemented the Bulirsch–Stoer (BS) method, which uses iterative integration and polynomial extrapolation to infinitesimal time-step size (**??**). An integration step is accepted, when two subsequent BS iterations have converged to below the BS tolerance level, ϵ.

The time-step parameter η and the BS tolerance ϵ, both influence the performance. If η is too big, convergence may not be achieved for any tolerance. If η is too small, the many integration steps will render the integration too expensive. There is an optimal value for η as a function of ϵ. We measured this relation empirically, which results in:

$$\log_{10} \eta = A \log_{10} \epsilon + B. \tag{2.2}$$

For $\epsilon < 10^{-50}$ the power law converges to $A = 0.029$ and $B = 0.45$. Extrapolating this relation to $\epsilon > 10^{-50}$ will cause the time-step size to become larger than the time scale for the closest encounter in the system. Therefore this relation saturates to a flatter power law for $\epsilon > 10^{-50}$ with $A = 0.012$ and $B = -0.40$. Compared to a fixed value for η, this relation speeds up the iterative procedure by about a factor three or more. The code is implemented as a community code in the AMUSE framework (**?**) under the name Brutus.

2.2.3 Method of Convergence

For every simulation we have to define the BS tolerance parameter ϵ and the word-length L_w. In an iterative procedure we vary both parameters systematically, each time carrying out a simulation until $t = t_{\rm end}$. We subsequently calculate the phase space distance, $\delta^2_{A,B}$, between two solutions A and B:

$$\delta^2_{A,B} = \frac{1}{6N} \sum_{i=1}^{N} \sum_{j=1}^{6} \left(q_{A,i,j} - q_{B,i,j}\right)^2 . \tag{2.3}$$

The first summation is over all particles and the second summation is over the six phase-space coordinates denoted by q (**?**). We normalise by $6N$, so that δ represents the average difference per phase-space coordinate between two solutions A and B. In our experiments we adopt Hénon units [2] (**??**), in which the typical values for the distance

[2]Formerly known as N-body units. Introduced by D. Heggie at MODEST14.

and velocity are of the same order. We will also use the distance in just position or just velocity space as they might behave differently.

We consider the solutions A and B to be converged when $\delta_{A,B} < 10^{-p}$ at all times during the simulation. Note that converged in this case means convergence of the total solution, contrary to convergence per integration step as in the previous section. This criterion for convergence is roughly equivalent to comparing the first p decimal places of the positions and velocities for all N stars, in two subsequent calculations A, B. In most of our experiments we adopt $p = 3$, i.e. all coordinates have to converge to about three decimal places or more. We perform a subset of simulations with $p = 15$ to investigate the effect of small errors (see Sec. 4.2.4).

Each simulation starts by specifying the initial positions and velocities of N stars in double-precision (see Sec. 4.1). The simulation is carried out with the parameter set (ϵ, L_w). We start each simulation with $\epsilon = 10^{-6}$ and $L_w = 56$ bits. This corresponds to a level of accuracy similar to what we reach with the conventional Hermite integrator. After this simulation, we increase L_w, for example to 72 bits (~ 22 decimal places), redo the simulation and calculate δ^2. We repeat this procedure until $\delta < 10^{-p}$. When this is achieved, we have obtained a solution in which the round-off error is below a specified number of decimal places for this particular value of ϵ.

We now reduce the tolerance parameter ϵ, for example by a factor of 100, and repeat the procedure of increasing L_w. This series will again lead to a converged solution, but this time it is obtained using a smaller ϵ, and is likely to be different than the previous converged solution. We continue decreasing the value of ϵ by factors of 100 and repeat the procedure, until two subsequent iterations in ϵ lead to a converged solution with a value of $\delta < 10^{-p}$. By this time we are assured of having a solution to the gravitational N-body problem, that is accurate up to at least p decimal places.

In practice, we speed up the procedure by writing the word-length as a function of BS tolerance. Consider for example a BS tolerance of 10^{-20}. To reach convergence up to this level, we need at least 20 decimal places. Adding an extra buffer of 10 digits gives a total of 30 digits, or equivalently a word-length of about 112 bits. For this example, 112 bits turns out to be a good minimum word-length. For a first estimate of the word-length, we use:

$$L_w = 4\,|\log_{10}\epsilon| + 32 \text{ bits}. \tag{2.4}$$

With this relation, we will only have to specify a single parameter ϵ, which directly controls the discretisation error and indirectly controls

the round-off error. For most of the systems in our experiment the discretisation error turns out to be the dominant source of error and as a consequence ϵ has to decrease quite drastically. When ϵ decreases, L_w increases, even up to the point that there are many more digits available than really needed to control the round-off error. In the case when the discretisation error dominates, the above defined minimum word-length for a given BS tolerance will result in the converged solution. When the round-off error dominates the word-length should be varied independently.

2.3 VALIDATION AND PERFORMANCE

2.3.1 The Pythagorean Problem

To show that our method works, we adopt the Pythagorean 3-body system (**?**). Previous numerical studies have shown that this system dissolves into a binary and an escaper (**??**). After many complex, close encounters the dissolution happens at about $t = 60$ time units (**?**), or about 16 crossing times.

We adopt the initial conditions for the Pythagorean problem and integrate up to $t = 100$. To illustrate how the method works we start with a high tolerance and short word-length, ($\epsilon = 10^{-2}$, $L_w = 40$ bits), which is less precise than double-precision. In Fig. 2.1, this calculation is compared to a simulation with ($\epsilon = 10^{-4}$, $L_w = 48$ bits), through the yellow (upper) curves in the first three panels. After the first BS integration step, δ obtains a value of the order of the BS tolerance, and continues to increase due to exponential divergence, to eventually exceed $\delta \sim 10^{-1}$, after which the errors become on the order of the typical distance and speed in the system.

In the following step, we repeat the calculation with a precision of ($\epsilon = 10^{-6}$, $L_w = 56$ bits), and compare the result with the calculation using ($\epsilon = 10^{-4}$, $L_w = 48$ bits). The comparison is represented by the orange curves (second from above) in Fig. 2.1. The overall behaviour of δ is similar, but the system diverges at a later time due to a higher initial precision.

We continue the iterative procedure until a converged solution has been obtained. In the first three panels of Fig. 2.1, it can be seen that subsequent simulations with higher precision shift the curve to lower values of δ. Superposed on the steady growth of the error are sharp spikes, where the error grows by several orders of magnitude, after which the error restores again (**?**). These spikes are dominated by errors in the velocity, as can be deduced by comparing the magnitude

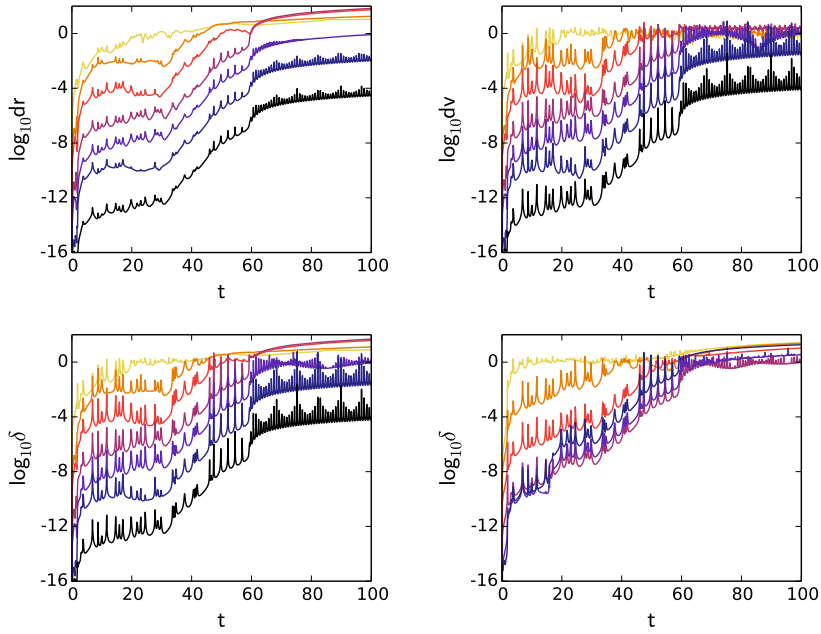

Figure 2.1: Exponential divergence in the Pythagorean problem. In the top two panels and the lower left panel, Brutus is compared with Brutus with increasing precision. The curves at the top of each panel compare a tolerance of 10^{-2} with 10^{-4}, the curve below compares 10^{-4} with 10^{-6} and so on. The word-length is a function of the tolerance as in Eq. 2.4. In the top left panel we show the distance in position-space, in the top right panel in velocity-space and in the bottom left panel in the full phase-space (all normalized by the number of stars and coordinates). The lower right panel compares the converged solution (the lowest curve in the other plots), with Hermite solutions with time-step parameters $\eta = 2^{-3}, 2^{-5}, 2^{-7}$ up to 2^{-13}.

of the spikes in position and velocity-space. Eccentric binaries which are out of phase when comparing two solutions cause large, periodic errors in the velocity. We finish the procedure when a solution is obtained for which the criterion for convergence is fulfilled, considering the magnitude of the error between the sharp spikes (bottom, black curves).

In the bottom right panel of Fig. 2.1, we compare solutions obtained by the Hermite integrator to the converged solution. The different curves belong to different time-step parameters; $\eta = 2^{-3}$, 2^{-5}, 2^{-7} up to 2^{-13}. Note that for a time-step parameter $\eta < 2^{-9}$, the curve is not shifted to lower values of δ, but even increases again. At this point round-off error becomes important, making the solution less accurate. The final close encounter in the Pythagorean problem occurs around 60 time units, after which a permanent binary and an escaper are formed. The Hermite integrator is able to accurately reproduce the evolution up to this point, but not subsequently, because δ has increased to values of order unity or higher. This can be explained by a small error in the final close encounter between all three stars, such that the direction of the escaper is slightly different.

To obtain the converged solution up to the first three decimal places, a tolerance of 10^{-14} and a word-length of 88 bits were needed. The simulation was about twice as slow compared to the Hermite simulation with $\eta = 2^{-9}$. The Hermite simulation, however, had a slightly different solution and a final, relative energy conservation of 10^{-8}, Decreasing the value of η will improve the level of energy conservation, but due to round-off error δ will not decrease.

2.3.2 The Equilateral Triangle

As a second test case, we adopt the 3-body equilateral triangle as an initial condition (Lagrange, 1772). In the exact solution this configuration remains intact, but small perturbations, such as produced by numerical errors, quickly cause the triangle to fall apart. For this problem, we also have a source of error in the initial conditions. Whereas the Pythagorean problem can be set up using integers, the initial condition for the equilateral triangle contains irrational numbers. To control the error in the initial condition, we calculate the initial coordinates with the same word-length as used for the simulation.

In the left panel of Fig. 2.2, a similar diagram is shown as for the Pythagorean problem in the lower left panel of Fig. 2.1. The starting precision is $\epsilon = 10^{-10}$ and the word-length is a function of ϵ as in Eq. 2.4. Subsequent simulations are performed with a 10 orders of

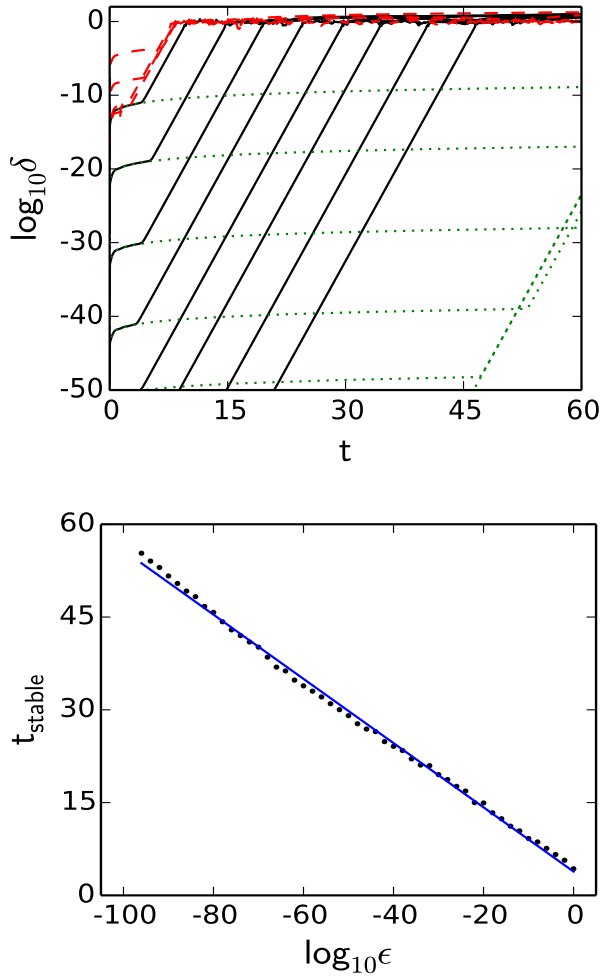

Figure 2.2: Divergence in the equilateral triangle configuration. In the top panel we show the divergence as a function of time. The solid curves compare Brutus solutions with increasing precision, where subsequent precisions are increased by 10 orders of magnitude and where the word-length is a function of tolerance as in Eq. 2.4. The dotted curves show results for similar simulations, but with a much longer, fixed word-length of 512 bits. The initial power law phase of divergence lasts longer in this case. The exponential divergence becomes dominant when the round-off error has had time to accumulate to become of the order the discretisation error. The dashed curves compare the highest precision Brutus solution with Hermite solutions with time-step parameters 10^{-1}, 10^{-2}, 10^{-3} and 10^{-4}. In the bottom panel we show for Brutus, the duration for which the triangular configuration remains intact as a function of Bulirsch–Stoer tolerance ϵ. Note that the time is in units of the period of one complete rotation of the system. The small scatter in the data is due to the discrete times at which we check the triangular configuration.

magnitude higher precision. For a short initial phase of 5 time units, the rate of divergence follows a power law. At later time, the solutions start to diverge exponentially with a characteristic rate independent of the tolerance and word-length. To investigate this transition, we redo the simulations with a large, fixed word-length of 512 bits (green dotted curves). This way, we reduce the amount of round-off error. As a consequence the rate of divergence is first dominated by the accumulation of discretisation errors and this phase lasts for a longer time, until the transition in the behaviour of the divergence, is reached, but now at ~ 45 time units. The time of the transition depends on word-length. Why the exponential divergence starts once the round-off error has kicked in, is a question that is still under investigation.

The red dashed curves in the same diagram in Fig. 2.2 give the results of the fourth-order Hermite, which are compared with the most precise Brutus simulation (with $\epsilon = 10^{-80}$, $L_w = 352$ bits). The time-step parameter $\eta = 10^{-1}$, 10^{-2}, 10^{-3} and 10^{-4} for subsequent curves. The Hermite integrations show a similar behaviour as the Brutus results, which could imply that the rate of divergence is a physical property of the configuration, rather than a property of the integrator.

In the right panel of Fig. 2.2 we show the duration for which the triangular configuration remains intact as a function of BS tolerance. For this experiment we halt the simulation when the distance between any two particles has increased or decreased by 10%, after which the triangle falls apart quickly. This diagram also illustrates the linear relation between accuracy and time in this system, which is caused by the constant number of digits being lost during every unit of time. The small scatter is due to the discrete times at which we check the triangular configuration. The solid, blue line is a fit to the data and its slope is $-0.52(3)$, which is equivalent to a loss of 1.9(1) digits per cycle.

2.3.3 A Plummer Distribution with N=16

As a third test we simulate the dynamical formation of the first hard binary in a small star cluster. We select a moderate number of sixteen equal mass stars and draw them randomly from a Plummer distribution (**?**). We integrate this system for about ten crossing times and apply the method of convergence. In Fig. 2.3 we present how two solutions with the same initial conditions, but different precisions, diverge as a function of time. The rate of exponential divergence, on average, starts rather constant, with a loss of $\sim 2/3$ digits per time unit. This is equivalent to an e-folding time of $t_e = 0.65$, which is consistent with

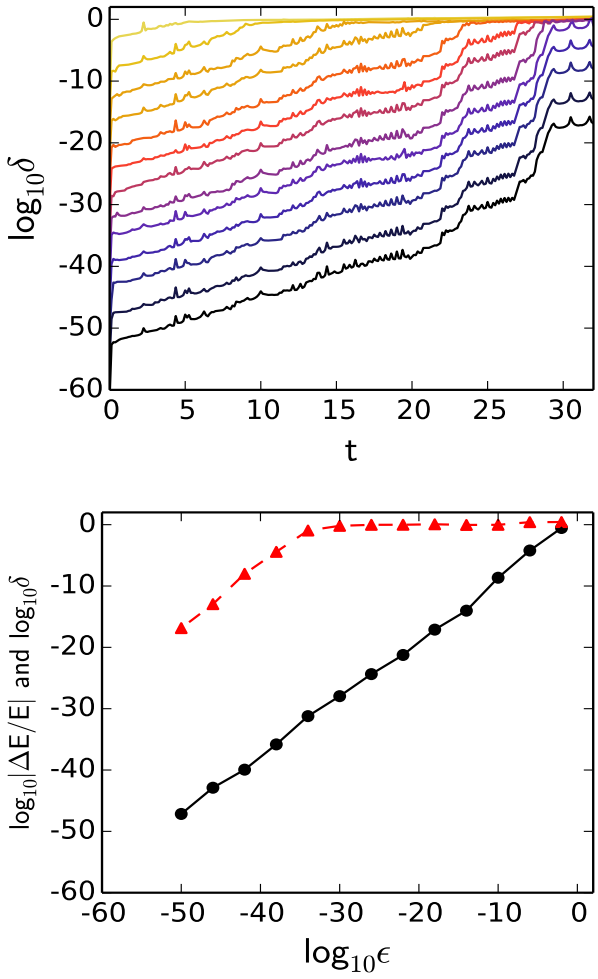

Figure 2.3: Exponential divergence in a 16-body cluster. In the top panel we illustrate the exponential divergence between Brutus simulations with increasing precision. In the bottom panel we show the final relative energy conservation (bullets, solid line) and the final normalized phase space distance between two subsequent simulations (triangles, dashed line) versus the Bulirsch–Stoer tolerance parameter ϵ. The solution starts to converge at a level of final relative energy conservation of $\sim 10^{-34}$.

the results of Goodman, Heggie and Hut (1993) (see their Fig. 8). From $t = 20$ onwards, the rate of divergence experiences systematic changes, in particular a steep rise of the error of about 10 orders of magnitude between $t = 26$ and $t = 29$. Such rises are a signature for the presence of a hard binary interacting with surrounding stars.

The right panel in Fig. 2.3 shows the energy conservation (black bullets, solid line) and the normalized phase space distance (red triangles, dashed line) versus ϵ. Energy conservation is proportional to ϵ, but the solutions only start to converge for $\epsilon < 10^{-34}$. More generally, even if conserved quantities like total energy are conserved to machine-precision or better, it is not guaranteed that the solution itself has converged.

The highest precision Brutus simulation in this example, ($\epsilon = 10^{-50}$, $L_w = 232$ bits), took about a day of wall-clock time, which is about 7000 times slower than a simulation with Hermite using $\eta = 2^{-9}$.

2.4 SCALING OF THE WALL-CLOCK TIME

The use of arbitrary-precision arithmetic dramatically increases the CPU time of N-body simulations. Also the BS method, which performs integration steps iteratively, makes an integration scheme more expensive by at least a factor two or more. To investigate for example how feasible it would be to run a converged N-body simulation for 10^3 stars through core collapse, we perform a scaling test in which we vary the number of particles and the precision, ϵ and L_w.

We randomly select positions and velocities for N equal mass stars from the virialised Plummer distribution (**?**), for $N = 2, 4, 8$, ..., up to 1024. The BS tolerance is fixed at a level of 10^{-6} and the word-length at 64 bits. We integrate the systems for one Hénon time unit and measure the wall-clock time. In the top left panel in Fig. 2.4 we show the wall-clock time as a function of N, which fit the relation $\mathrm{t_{CPU}} \propto N^{2.6}$.

For $N > 32$, it becomes efficient to parallellise the code. Our version implements i-parallellisation (Portegies Zwart et al., 2008) in the calculation of the accelerations. In the top right panel of Fig. 2.4, we plot the speed-up, S, against the number of cores. For $N = 1024$, we obtain a speed up of a factor 30 using 64 cores.

In the lower panels of Fig. 2.4 we present the scaling of the wall-clock time with BS tolerance and word-length. To measure the dependence on BS tolerance, we simulated a 16-body cluster for 1 Hénon time unit. We varied the BS tolerance while keeping the word-length fixed at $L_w = 1024$ bits. The relation obtained converges to $\mathrm{t_{CPU}} \propto \epsilon^{-0.032}$.

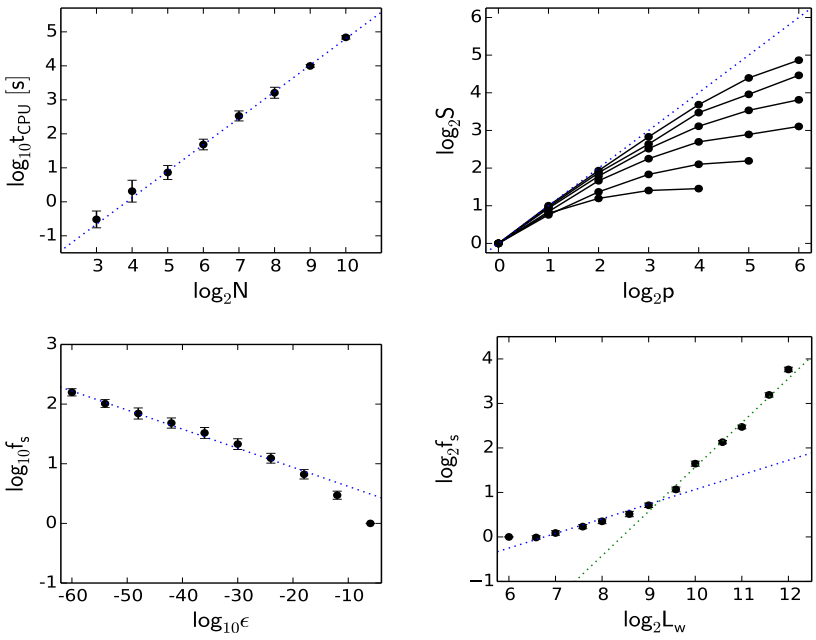

Figure 2.4: In the top left panel we show the scaling of the wall-clock time for Brutus as a function of number of stars N. The dotted curve is a fit to the data given by $t_{CPU} \propto N^{2.6}$. In the top right panel we show the speed-up when the number of cores, p, is increased. The bottom, solid curve represents $N = 32$ and each curve above has an N a factor two higher than the previous curve. The dotted curve represents ideal scaling. In the bottom left panel we plot the slowdown factor as a function of the Bulirsch–Stoer tolerance ϵ, for a fixed word-length of 1024 bits. In the bottom right panel we plot the slowdown factor as a function of word-length L_w, for a fixed tolerance of 10^{-10}. The slowdown of the simulations is mainly caused by the very small Bulirsch-Stoer tolerances required.

A similar experiment was performed to measure the dependence on word-length. This time we fixed the BS tolerance at $\epsilon = 10^{-10}$ and varied the word-length. For $L_w < 1024$, the relation can be estimated as $t_{CPU} \propto L_w^{0.33}$, while for $L_w > 1024$, $t_{CPU} \propto L_w$. This transition depends on the internal workings of the arbitrary-precision library which we will not discuss here.

Using a very long word-length of 4096 bits, i.e. $\sim 10^3$ digits, results in a slowdown of a factor $f_s \sim 16$ compared to 64 bits. But for some simulations a BS tolerance smaller than 10^{-50} can easily be required to reach convergence, and this will result in a slowdown of a factor $f_s > 100$. The very small BS tolerance is often the main cause for the slowdown of the simulations, instead of the increased word-length.

Using the above results, we can construct the following model to estimate the wall-clock time for integrating 1 Hénon time unit with $L_w < 1024$ bits:

$$t_{CPU} = \left(\frac{N}{512}\right)^{2.6} \left(\frac{\epsilon}{10^{-6}}\right)^{-0.032} \left(\frac{L_w}{64}\right)^{0.33} 10^4\,[s]. \qquad (2.5)$$

Integrating $N = 1024$ with standard precision, ($\epsilon = 10^{-6}$, $L_w = 64$ bits), up to core collapse at ~ 300 time units, and taking into account a speed up of a factor 30 due to parallellisation, we estimate a total wall-clock time of a week. Increasing the precision to ($\epsilon = 10^{-20}$, $L_w = 112$ bits), will take about a month. A precision of ($\epsilon = 10^{-50}$, $L_w = 232$ bits) will take roughly a year. To estimate how much precision is needed, we will assume that the rate of exponential divergence before the formation of the first hard binary is approximately constant. In the left panel of Fig. 2.3, the initial slopes correspond to a loss of $\sim 2/3$ digits per time unit. We construct the following approximate model for the initial BS tolerance needed to end up with a converged solution:

$$\log_{10} \epsilon = \log_{10} \delta_{\rm final} - R_{\rm div} t_{\rm cc}. \qquad (2.6)$$

Here ϵ is the BS tolerance parameter, δ_{final} is the final precision of all the coordinates in the system, $R_{\rm div}$ is the approximately constant rate of divergence, e.g. the number of accurate digits lost per unit of time, and $t_{\rm cc}$ is the core collapse time. We set the final precision to 10^{-6}, i.e. convergence to the first 6 decimal places, and we set the core collapse time to ~ 300 as before. If we adopt $R_{\rm div} = 2/3$, we estimate that we need an $\epsilon \sim 10^{-206}$. This would take about 10^5 years to finish. It would be more practical to simulate a 256-body cluster. If we set the

core collapse to 100 time units we estimate $\epsilon \sim 10^{-73}$, which would take about a month on a cluster of 64 Intel(R) Xeon(R) E5530 cores.

For direct N-body codes, the time for integrating up to core collapse usually scales as $\mathcal{O}(N^3)$. Using the analysis above, we estimate that the time for converged core collapse simulations scales approximately exponentially. This is effectively caused by the exponential divergence.