Monthly Notices
of the
ROYAL ASTRONOMICAL SOCIETY

MNRAS **492,** 4131–4148 (2020)
Advance Access publication 2020 January 15

doi:10.1093/mnras/staa084

MSTAR – a fast parallelized algorithmically regularized integrator with minimum spanning tree coordinates

Antti Rantala [1,2]★ Pauli Pihajoki [2] Matias Mannerkoski [2] Peter H. Johansson [2] and Thorsten Naab [1]

[1] *Max-Planck-Institut für Astrophysik, Karl-Schwarzschild-Str 1, D-85748 Garching, Germany*
[2] *Department of Physics, University of Helsinki, Gustaf Hällströmin katu 2, FI-00560 Helsinki, Finland*

Accepted 2020 January 9. Received 2020 January 8; in original form 2019 December 19

ABSTRACT

We present the novel algorithmically regularized integration method MSTAR for high-accuracy ($|\Delta E/E| \gtrsim 10^{-14}$) integrations of N-body systems using minimum spanning tree coordinates. The twofold parallelization of the $\mathcal{O}(N_{\rm part}^2)$ force loops and the substep divisions of the extrapolation method allow for a parallel scaling up to $N_{\rm CPU} = 0.2 \times N_{\rm part}$. The efficient parallel scaling of MSTAR makes the accurate integration of much larger particle numbers possible compared to the traditional algorithmic regularization chain (AR-CHAIN) methods, e.g. $N_{\rm part} = 5000$ particles on 400 CPUs for 1 Gyr in a few weeks of wall-clock time. We present applications of MSTAR on few particle systems, studying the Kozai mechanism and N-body systems like star clusters with up to $N_{\rm part} = 10^4$ particles. Combined with a tree or fast multipole-based integrator, the high performance of MSTAR removes a major computational bottleneck in simulations with regularized subsystems. It will enable the next-generation galactic-scale simulations with up to 10^9 stellar particles (e.g. $m_\star = 100\,{\rm M}_\odot$ for an $M_\star = 10^{11}\,{\rm M}_\odot$ galaxy), including accurate collisional dynamics in the vicinity of nuclear supermassive black holes.

Key words: gravitation – methods: numerical – quasars: supermassive black holes – galaxies: star clusters: general.

1 INTRODUCTION

Galactic nuclei and their nuclear stellar clusters are among the densest stellar systems in the entire Universe (Misgeld & Hilker 2011). The nuclei of massive galaxies also host supermassive black holes (SMBHs) with typical masses in the range of $M = 10^6$–$10^{10}\,{\rm M}_\odot$ (Kormendy & Richstone 1995; Ferrarese & Ford 2005; Kormendy & Ho 2013), forming a complex, collisionally evolving stellar-dynamical environment (e.g. Merritt 2013; Alexander 2017). In the ΛCDM hierarchical picture of structure formation, galaxies grow through mergers and gas accretion, resulting in situations where the collisional evolution of a galactic nucleus is intermittently interrupted and transformed by a merger (e.g. White & Rees 1978; Begelman, Blandford & Rees 1980). For gas-poor mergers, the more concentrated nucleus with a steeper stellar cusp will determine the structure of the remnant nucleus immediately after the merger (Holley-Bockelmann & Richstone 1999; Boylan-Kolchin & Ma 2004).

If both of the merging galaxies host central SMBHs, the SMBHs will merge through a three-stage process (Begelman et al. 1980). First, on larger scales, the SMBHs are brought together through dynamical friction from stars and gas until a hard binary is formed with a semimajor axis of $a \sim 10$ pc. In the second phase, the hard binary will interact with the stars in the centre of the merger remnant (Begelman et al. 1980; Milosavljević & Merritt 2001, 2003; Khan, Just & Merritt 2011), scouring a low-density stellar core in the process (e.g. Merritt 2006; Lauer et al. 2007; Rantala et al. 2018, 2019). The largest uncertainty in this process is the rate at which the 'loss cone' is depleted, but there is an emerging consensus that the binary will avoid the so-called final-parsec problem and eventually merge into a single SMBH, even in the collisionless limit (e.g. Berczik et al. 2006; Vasiliev, Antonini & Merritt 2015; Gualandris et al. 2017; Ryu et al. 2018; Mannerkoski et al. 2019), with the final coalescence driven by the emission of gravitational waves (Peters & Mathews 1963).

If the galaxy merger is gas rich, the evolution of the nucleus of the merger remnant proceeds very differently. During galaxy mergers, the resulting asymmetric potential effectively funnels gas inwards into the central regions, causing a central starburst that rapidly increases the nuclear stellar density by several orders of magnitude (e.g. Sanders & Mirabel 1996). In addition, the gas also plays an important role by causing additional drag on the SMBHs (Beckmann, Slyz & Devriendt 2018), as well as by forming circumbinary discs that can have a significant and complicated effect on the angular momentum evolution of the binary (Tang, MacFadyen & Haiman 2017; Duffell et al. 2019; Moody, Shi &

★ E-mail: anttiran@mpa-garching.mpg.de

Stone 2019). In general, SMBH mergers are thought to occur very rapidly in dense gas-rich environments (e.g. Khan et al. 2016) mainly due to the short duration of the stellar interaction phase of the binary evolution (Quinlan 1996).

The growth of SMBHs and the formation of binaries in galaxy mergers have been extensively studied in recent decades. A typical numerical approach is to use grid codes (Kim et al. 2011; Dubois et al. 2013; Hayward et al. 2014), smoothed particle hydrodynamics codes with tree gravity (Springel, Di Matteo & Hernquist 2005; Mayer et al. 2007; Johansson, Naab & Burkert 2009), or direct summation codes (Berczik et al. 2006; Khan et al. 2011). The drawback of grid codes and softened tree codes is that they cannot properly treat collisional systems as the grid cell size or the employed gravitational softening length places a strict spatial resolution limit on the simulation.

Direct summation codes, on the other hand, are very well suited for studying collisional stellar systems with $N_{\rm part} \lesssim 10^6$ particles, such as globular clusters. However, the steep $\mathcal{O}(N_{\rm part}^2)$ scaling of the direct summation algorithm limits the applicability of this method to systems with much higher particle numbers. In addition, the most widely used direct summation codes are rarely coupled with a hydrodynamic solver. One possibility is to use on-the-fly code switching (Milosavljević & Merritt 2001; Khan et al. 2016), but this type of procedure is typically cumbersome and may introduce spurious numerical effects into the simulation. We thus argue that a self-consistent numerical framework for simulating SMBH dynamics in a realistic collisional environment with a high stellar density and including also a gas component still remains to be developed. One of the major obstacles for developing such a code has been the lack of available fast accurate small-scale collisional post-Newtonian integrators, which are also straightforward to couple to both large-scale gravity solvers and hydrodynamical methods.

The most widely used method to treat binaries and close encounters of simulation particles in the collisional regions of simulations is the technique of regularization. The key idea of regularization is to transform the equations of motion of a dynamical system into a form without coordinate singularities, which makes solving the dynamics significantly easier using standard numerical integration techniques (Aarseth 2003). The first such method with practical numerical applications was the two-body Kustaanheimo–Stiefel (KS) regularization method (Kustaanheimo & Stiefel 1965), which transformed both the time and spatial coordinates of the system. A major step forward for regularized dynamics was the introduction of the chain concept. By arranging the simulation particles into a chain of inter-particle vectors, the KS-CHAIN of Mikkola & Aarseth (1993) reduced the number of required KS transformations from $N_{\rm part}(N_{\rm part}-1)/2$ to $N_{\rm part}-1$, yielding a much more efficient regularization method. In the *N*-body codes of Aarseth (1999), the KS-CHAIN treats the mutual encounters of not more than six particles simultaneously.

A new regularization method, which does not require a coordinate transformation but only a time transformation, was discovered by Mikkola & Tanikawa (1999a) and Preto & Tremaine (1999). This algorithmic regularization (AR) method is faster than the previous regularization methods and more accurate, especially in the case of large mass ratios between particles in a *N*-body system. Many current direct summation codes (Harfst et al. 2008; Aarseth 2012) and regularized tree codes (Rantala et al. 2017) use an implementation of the AR-CHAIN integrator (Mikkola & Merritt 2006, 2008) to resolve the small-scale dynamics around SMBHs.

Despite the many successes of regularization methods, their original design as few-body codes still limits their applicability to systems with a very large number of particles, which is crucial for performing galactic-scale simulations at high accuracy. The regularized tree code KETJU (Rantala et al. 2017) is currently limited to a particle resolution of $N_{\rm part} \lesssim 10^7$ stellar particles per galaxy, although the underlying tree code GADGET-3 (Springel et al. 2005) could easily run simulations with ∼10–100 times more stellar particles, up to $N_{\rm part} \sim 10^9$ collisionless particles in a single galaxy. This is because the included AR-CHAIN integrator becomes impractically inefficient with more than ∼300–500 particles in the regularized chain region. We note that similar performance issues with the KS-CHAIN algorithm have already been reported in the literature; see e.g. Milosavljević & Merritt (2001).

In this paper, we present a new algorithmically regularized (AR) integrator MSTAR developed and implemented with the aim of addressing some of the shortcomings of the previous algorithms. Our new code contains two main improvements compared to existing AR integrators. First, we use a minimum spanning tree (MST) coordinate system instead of the chain structure, motivating the name of the code. We note that regularized integration algorithms using particular tree structures have been implemented before: Jernigan & Porter (1989) developed a KS-regularized binary tree code while Mikkola & Aarseth (1989) proposed a 'branching' KS-CHAIN structure for few-body regularization. However, neither of these methods are widely used today. Secondly, a major speed-up compared to the previous regularization methods is achieved by applying a twofold parallelization strategy to the extrapolation method, which is used in regularized integrators to ensure a high numerical accuracy (Mikkola & Tanikawa 1999b).

The remainder of this paper is organized as follows: In Section 2, we review our implementation of AR-CHAIN, as the new code MSTAR was developed based on our earlier experience with the AR-CHAIN integrator. The numerical procedures of the MSTAR integrator and the code implementation are discussed in Section 3. We describe and test the parallel extrapolation method in Section 4. In Section 5, we perform a number of few-body code tests to validate the accuracy of the MSTAR code, whereas in Section 6 we perform a number of scaling and timing tests. Finally, we summarize our results and present our conclusions in Section 7.

2 AR-CHAIN

2.1 Time transformation of equations of motion

The algorithmic regularization chain (AR-CHAIN) integrator is designed to perform extremely accurate orbital integrations of gravitational few-body systems (Mikkola & Merritt 2006, 2008). The equations of motion of the system are time-transformed by extending the phase space to include the original time parameter as a coordinate together with the corresponding conjugate momentum, equal to the binding energy of the system. A new independent variable is then introduced through a Poincaré time transformation. With a specific choice of the time transformation function (Mikkola & Tanikawa 1999a; Preto & Tremaine 1999), the new Hamiltonian and the equations of motion are separable so that the system can be integrated using a leapfrog method. This surprisingly yields an exact orbit for the Keplerian two-body problem even for collision orbits. The only error is in the time coordinate, or the phase of the Keplerian binary. However, this error can be removed by a further modification of the Hamiltonian (Mikkola, Palmer & Hashida 2002), yielding an exact solver, up to machine precision.

We start with the standard N-body Hamiltonian H, defined as

$$H = T - U = \sum_{\rm i} \frac{1}{2} m_{\rm i} \|\boldsymbol{v}_{\rm i}\|^2 - \sum_{\rm i} \sum_{\rm j>i} \frac{G m_{\rm i} m_{\rm j}}{\|\boldsymbol{r}_{\rm j} - \boldsymbol{r}_{\rm i}\|} \tag{1}$$

in which T is the kinetic energy and U is the force function, equal to negative of the potential energy. This Hamiltonian yields the familiar Newtonian equations of motion for the particle positions $\boldsymbol{r}_{\rm i}$ and velocities $\boldsymbol{v}_{\rm i}$:

$$\begin{aligned} \frac{{\rm d}\boldsymbol{r}_{\rm i}}{{\rm d}t} &= \boldsymbol{v}_{\rm i} \\ \frac{{\rm d}\boldsymbol{v}_{\rm i}}{{\rm d}t} &= \boldsymbol{a}_{\rm i} = G \sum_{\rm i \neq j} m_{\rm j} \frac{\boldsymbol{r}_{\rm j} - \boldsymbol{r}_{\rm i}}{\|\boldsymbol{r}_{\rm j} - \boldsymbol{r}_{\rm i}\|^3} \end{aligned} \tag{2}$$

in which we have introduced the Newtonian accelerations $\boldsymbol{a}_{\rm i}$. Possible additional acceleration terms, such as external perturbations $\boldsymbol{f}_{\rm i}$ depending only on particle positions, or velocity-dependent perturbations $\boldsymbol{g}_{\rm i}(\boldsymbol{v})$, such as post-Newtonian corrections, can be directly added to the Newtonian accelerations, yielding $\boldsymbol{a}_{\rm i} \rightarrow \boldsymbol{a}_{\rm i} + \boldsymbol{f}_{\rm i} + \boldsymbol{g}_{\rm i}(\boldsymbol{v})$.

Next, we perform the time transformation (Mikkola & Tanikawa 1999a; Preto & Tremaine 1999). A fictitious time s is introduced as a new independent variable. The original independent variable, physical time t, is promoted to a coordinate of the phase space of the system, while the binding energy of the system $B = -H$ becomes the corresponding conjugate momentum. The old and new time variables are related by the infinitesimal time transformation

$$\frac{{\rm d}t}{{\rm d}s} = \frac{1}{\alpha U + \beta \Omega + \gamma} \tag{3}$$

in which the parameter triplet (α, β, γ) determines the type of regularization. Ω is an arbitrary real-valued function of coordinates, such as the force function for the least massive particles in the system (Mikkola & Aarseth 2002). With the triplet (1, 0, 0), the method becomes the logarithmic Hamiltonian (LogH) method of Mikkola & Tanikawa (1999a) and Preto & Tremaine (1999), while (0, 1, 0) corresponds to the time-transformed leapfrog introduced in Mikkola & Aarseth (2002). The ordinary non-regularized leapfrog is obtained by choosing the triplet (0, 0, 1). Of all the possible choices, Mikkola & Merritt (2006) recommend using the LogH option (1, 0, 0) for its superior numerical performance.

Using the LogH time transformation, the equations of motion for the system become

$$\begin{aligned} \frac{{\rm d}t}{{\rm d}s} &= \frac{1}{T + B} \\ \frac{{\rm d}\boldsymbol{r}_{\rm i}}{{\rm d}s} &= \frac{1}{T + B} \boldsymbol{v}_{\rm i} \end{aligned} \tag{4}$$

for the coordinates and

$$\begin{aligned} \frac{{\rm d}\boldsymbol{v}_{\rm i}}{{\rm d}s} &= \frac{1}{U} (\boldsymbol{a}_{\rm i} + \boldsymbol{f}_{\rm i} + \boldsymbol{g}_{\rm i}(\boldsymbol{v})) \\ \frac{{\rm d}B}{{\rm d}s} &= -\frac{1}{U} \sum_{\rm i} m_{\rm i} \boldsymbol{v}_{\rm i} \cdot (\boldsymbol{f}_{\rm i} + \boldsymbol{g}_{\rm i}(\boldsymbol{v})) \end{aligned} \tag{5}$$

for the velocities. In this discussion, we omit the Ω function and its velocity conjugate. For an unperturbed Newtonian system, i.e. $\boldsymbol{f}_{\rm i} = \boldsymbol{g}_{\rm i}(\boldsymbol{v}) = 0$, the derivatives of the coordinates depend only on the velocities and vice versa; thus, a leapfrog algorithm can be constructed in a straightforward manner. With non-zero external tidal perturbations $\boldsymbol{f}_{\rm i}$, the derivative of the binding energy B depends on the particle velocities, but the dependence is only linear and can thus be analytically integrated over the time-step (see e.g. the appendices A and B of Rantala et al. 2017).

An explicit leapfrog algorithm cannot be constructed if the post-Newtonian accelerations $\boldsymbol{g}_{\rm i}(\boldsymbol{v})$ are non-zero. One can, in practice, approach the problem by iterating the implicit equations of motion, but this is very inefficient. An efficient post-Newtonian leapfrog algorithm with velocity-dependent accelerations can be implemented by extending the phase space of the system with auxiliary velocities $\boldsymbol{w}_{\rm i}$. A single leapfrog velocity update (kick) is replaced by an alternating combination of auxiliary and physical kicks, performed in a standard leapfrog manner. For additional details of the auxiliary velocity procedure, see Hellström & Mikkola (2010) and its generalization by Pihajoki (2015).

Nevertheless, the LogH integrator on its own is not accurate enough for high-precision solutions of general N-body systems, even though the systems are regularized against collision singularities. The LogH leapfrog must be supplemented with additional numerical techniques such as chained coordinates and extrapolation techniques. These two methods are introduced in Sections 2.2 and 2.3, respectively.

2.2 Chained coordinate system

In AR-CHAIN, the chained inter-particle coordinate system does not play a role in the regularization procedure itself, unlike in the earlier KS-CHAIN regularization. However, numerical experiments (Mikkola & Tanikawa 1999a,b) have shown that the chained coordinate system is very useful in increasing the numerical accuracy of the method by significantly reducing the numerical floating-point error.

When constructing the chained coordinates, one first finds the shortest inter-particle coordinate vector of the N-body system. These two particles become the initial tail and head of the chain. Next, the particle closest to either the tail or the head of the chain is found among the non-chained particles. This particle is added as the new tail or head of the chain, depending on which end of the chain is closer. The process is repeated until all particles are in the chain.

Labelling the particles starting from the tail of the chain, the inter-particle position, velocity, and various acceleration vectors become

$$\begin{aligned} \boldsymbol{X}_{\rm k} &= \boldsymbol{r}_{\rm j_k} - \boldsymbol{r}_{\rm i_k} \equiv \boldsymbol{r}_{\rm k+1} - \boldsymbol{r}_{\rm k} \\ \boldsymbol{V}_{\rm k} &= \boldsymbol{v}_{\rm j_k} - \boldsymbol{v}_{\rm i_k} \equiv \boldsymbol{v}_{\rm k+1} - \boldsymbol{v}_{\rm k} \\ \boldsymbol{A}_{\rm k} &= \boldsymbol{a}_{\rm j_k} - \boldsymbol{a}_{\rm i_k} \equiv \boldsymbol{a}_{\rm k+1} - \boldsymbol{a}_{\rm k} \\ \boldsymbol{F}_{\rm k} &= \boldsymbol{f}_{\rm j_k} - \boldsymbol{f}_{\rm i_k} \equiv \boldsymbol{f}_{\rm k+1} - \boldsymbol{f}_{\rm k} \\ \boldsymbol{G}_{\rm k} &= \boldsymbol{g}_{\rm j_k} - \boldsymbol{g}_{\rm i_k} \equiv \boldsymbol{g}_{\rm k+1} - \boldsymbol{g}_{\rm k} \end{aligned} \tag{6}$$

in which the last expression on the right-hand side describes the relabelling of the particle indexes along the chain. Note that there are $N_{\rm part} - 1$ inter-particle vectors for a system of $N_{\rm part}$ bodies. The equations of motion for the chained coordinates then become

$$\begin{aligned} \frac{{\rm d}t}{{\rm d}s} &= \frac{1}{T + B} \\ \frac{{\rm d}\boldsymbol{X}_{\rm i}}{{\rm d}s} &= \frac{1}{T + B} \boldsymbol{V}_{\rm i} \end{aligned} \tag{7}$$

while the velocity equations can be expressed as

$$\begin{aligned} \frac{{\rm d}\boldsymbol{V}_{\rm i}}{{\rm d}s} &= \frac{1}{U} (\boldsymbol{A}_{\rm i} + \boldsymbol{F}_{\rm i} + \boldsymbol{G}_{\rm i}) \\ \frac{{\rm d}B}{{\rm d}s} &= -\frac{1}{U} \sum_{\rm i} m_{\rm i} \boldsymbol{v}_{\rm i} \cdot (\boldsymbol{f}_{\rm i} + \boldsymbol{g}_{\rm i}). \end{aligned} \tag{8}$$

It is worthwhile to note that the derivative of the binding energy B is in fact easier to evaluate by using the original coordinate system

than the chained one. For this, the chained velocities need to be transformed back into the original coordinate system during the integration.

Finally, the chained coordinate vectors are needed to evaluate the accelerations $\boldsymbol{a}_\mathrm{i}$ and $\boldsymbol{g}_\mathrm{i}(\boldsymbol{v})$. We use the chained coordinates for computing the separation vectors for N_d closest particles in the chain structure while the original vectors are used for more distant particles, i.e.

$$\boldsymbol{r}_\mathrm{j} - \boldsymbol{r}_\mathrm{i} = \begin{cases} \boldsymbol{r}_j - \boldsymbol{r}_i & \text{if } |i-j| > N_\mathrm{d} \\ \sum\limits_{k=\min\{i,j\}}^{\max\{i,j\}-1} \mathrm{sign}(i-j)\boldsymbol{X}_k & \text{if } |i-j| \leq N_\mathrm{d}. \end{cases} \tag{9}$$

Typically, $N_\mathrm{d} = 2$ in the literature (Mikkola & Merritt 2008). In general, selecting values $N_\mathrm{d} > 2$ for the separation parameter undermines the usefulness of the chain construct as the floating-point error begins to accumulate when summing many inter-particle vectors in equation (9).

2.3 GBS extrapolation method

Even though the chained LogH leapfrog with the time-transformed equations of motion yields regular particle orbits for all non-pathological initial conditions, the numerical integration accuracy is usually too low for high-precision applications. Thus, the chained leapfrog integrator must be accompanied by an extrapolation method to reach a high numerical accuracy (Mikkola & Tanikawa 1999b). A widely used method is the Gragg–Bulirsch–Stoer (Gragg 1965; Bulirsch & Stoer 1966) or GBS extrapolation algorithm. The GBS extrapolation method can only be used with integrators that have an error scaling containing only even powers of the time-step, but fortunately many simple low-order integrators such as the mid-point method and the chained leapfrog fulfil this requirement. In this study, we use only leapfrog-type integrators with the GBS algorithm.

In general, when numerically solving initial value problems for differential equations, the numerical solution will converge towards the exact solution when the step size h of the numerical method is decreased. The numerical accuracy of integrators with an extrapolation method is based on this fact. The key idea of the GBS extrapolation is to successively integrate the differential equation over an interval H using an increasing number of substeps n. The integrations are carried out in small steps of length $h = H/n$ using a suitable numerical method, and the results are then extrapolated to $h \to 0$. Different substep division sequences n_k have been studied in the literature to achieve converged extrapolation results with a minimum computational cost (Press et al. 2007). Popular options include the original GBS sequence (Bulirsch & Stoer 1966), defined as

$$\{n_\mathrm{k}\} = \{2, 4, 6, 8, 12, 16, 24, 32, 48, 64, 96 \dots\}, \tag{10}$$

i.e. $n_\mathrm{k} = 2n_{\mathrm{k}-2}$, $k > 2$ and the so-called Deuflhard sequence (Deuflhard 1983) of even numbers

$$\{n_\mathrm{k}\} = \{2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, \dots\} \tag{11}$$

with $n_\mathrm{k} = 2k$. In our MSTAR code and throughout this paper, we use the Deuflhard sequence.

The chained leapfrog sequence with $n \geq 1$ substeps can be written as

$$\boldsymbol{D}\left(\frac{h}{2n}\right)\left[\boldsymbol{K}\left(\frac{h}{n}\right)\boldsymbol{D}\left(\frac{h}{n}\right)\right]^{n-1}\boldsymbol{K}\left(\frac{h}{n}\right)\boldsymbol{D}\left(\frac{h}{2n}\right), \tag{12}$$

where the drift $\boldsymbol{D}(h)$ operators advance the coordinates and the kick operators $\boldsymbol{K}(h)$ the velocity-like variables by the time-step h. The GBS algorithm starts by computing the first few substep divisions n_k with the leapfrog, after which a rational function or a polynomial extrapolation of the integration results to $h \to 0$ is attempted. The error control is enabled with the convergence criterion

$$\frac{\|\Delta S_k\|}{\|S(s) + \frac{h}{n}S'(s)\|} \leq \eta_\mathrm{GBS}, \tag{13}$$

where S is any dynamical variable of the system, ΔS_k is the extrapolation error estimate after the k-th substep sequence, and $S(s)$ and $S'(s)$ are the value of the dynamical variable and its time derivative obtained after the last complete time-step H, respectively. The GBS tolerance η_GBS is a user-given free input parameter. The extrapolation can typically be carried out successfully even if η_GBS is set near the double-precision floating-point precision $\eta_\mathrm{GBS} \sim 10^{-16}$. In the literature, the most typical values of the accuracy parameter are set in the range of $10^{-12} \leq \eta_\mathrm{GBS} \leq 10^{-6}$.

If convergence is not reached after the first few leapfrog step divisions, the GBS algorithm proceeds to the next substep division and tries the extrapolation again until convergence, or until the maximum number of divisions $n_\mathrm{max} = n_{k_\mathrm{max}}$, is reached. In the case of no convergence after the k_max-th substep sequence, the original step H is halved and the process is started again from the beginning. After convergence, the criterion for the next time-step H_{i+1} after convergence is (Press et al. 2007)

$$H_{\mathrm{i}+1} = a_\mathrm{GBS}\left(\frac{\eta_\mathrm{GBS}}{\epsilon_\mathrm{k}}\right)^{1/(2k-1)} H_\mathrm{i}, \tag{14}$$

where ϵ_i is the maximum error in the dependent variables from the previous step, $a_\mathrm{GBS} \in (0, 1]$ is a safety factor (Hairer, Nørsett & Wanner 2008), and k is the substep sequence at which convergence was achieved. The GBS algorithm also monitors whether trying convergence at different k, or equivalently, changing the order $2k - 1$ of the method, would lead to convergence with a smaller workload. The time-step H is then adjusted accordingly, to bring the k where convergence is expected to the optimal value (Press et al. 2007).

Finally, it should be noted that the extrapolation algorithm used in AR-CHAIN described above is serial in nature, even though the particle accelerations in the individual leapfrog sequences can be computed in parallel as in Rantala et al. (2017). Thus, computing a large number of subsequent subdivision counts or reaching the maximum subdivision count k_max without convergence and restarting with a smaller time-step $H/2$ can be very time consuming.

2.4 Iteration to exact physical time

We next describe an improvement of the time iteration procedure in the new MSTAR integrator over our older AR-CHAIN version. Consider the integration of the equations of motion of a N-body system over a time interval Δt. With the standard leapfrog, there is no problem in arriving at the correct end time, but with the time-transformed leapfrog one has to be more careful (e.g. Mikkola 1997).

Integrating the time transformation of equation (3) over a time interval $H = \Delta t$ with the parameter triplet (1, 0, 0) yields

$$\Delta s = \int_0^{\Delta t} U \mathrm{d}t = G\sum_\mathrm{i}\sum_{\mathrm{j}>\mathrm{i}} m_\mathrm{i} m_\mathrm{j} \int_0^{\Delta t} \frac{\mathrm{d}t}{\|\boldsymbol{r}_\mathrm{j} - \boldsymbol{r}_\mathrm{i}\|}. \tag{15}$$

One can, in principle, approach these $N_\mathrm{part}(N_\mathrm{part} - 1)/2$ integrals in the formula by using the Stumpff–Weiss method, which assumes

that all the particle motions during the time interval are Keplerian (Stumpff & Weiss 1968). However, a simple approximation

$$\Delta s = U \Delta t \tag{16}$$

typically provides sufficiently accurate results, especially when the time interval Δt is short.

In our AR-CHAIN implementation (Rantala et al. 2017), we begin the regularized integration by estimating the amount of fictitious time Δs based on the smallest Keplerian time-scale $P_{\rm Kepler}$ of the system. Using equation (16), we have $\Delta s = qUP_{\rm Kepler}$, in which q is a safety factor $0 < q \leq 1$. After the first step, we let the GBS algorithm decide the step size until we exceed the output time Δt. Then, we take additional GBS iteration steps towards the correct exact time until the relative difference between the integrated time and Δt is small enough, typically $\sim 10^{-4}$–10^{-6}. This requires 2–5 iteration steps in most cases.

We improve the time iteration procedure for our new integrator as the GBS steps are expensive and one should try to converge to the correct physical time with as few iterations as possible. For the first step, we already use equation (16) multiplied by a safety factor instead of the Keplerian criterion. Next, we assume that after a GBS step has been taken we have arrived at the physical time $0 < \tau < \Delta t$. As the GBS routine suggests a new time-step Δs, we check that the estimated time coordinate after the next step $\tau + \Delta s/U$ does not exceed Δt. If it does, we set the next fictitious time-step to $\Delta s = U(\Delta t - \tau)$. If we still end up to time $\tau > \Delta t$ after integration, we proceed as in the old method. This procedure typically requires a few expensive iteration steps less than the approach we used in our previous AR-CHAIN implementation. The speed-up gained by the updated iteration procedure depends on Δt and the N-body system, and the maximum expected speed-up occurs in the integration of very short time intervals.

3 MST COORDINATES

3.1 Shortest Hamiltonian paths and MSTs

The properties of chained inter-particle coordinate systems can be conveniently expressed by using the language of graph theory (e.g. Harary 1969). A graph $G = (V, E)$ is an ordered pair of vertices V and edges E. An edge is determined by the vertices it connects, i.e. $E_{\rm ij} = (V_{\rm i}, V_{\rm j})$. For our purposes, the N-body particles are the vertices of the graph while the inter-particle vectors correspond to the edges of the graph. A graph with N vertices and all possible $N(N-1)/2$ edges is called complete. Complete graphs are also necessarily connected as any vertex can be reached from any other vertex of the graph. Each edge E is weighted with a non-negative real number w. We set the weights of the edges by calculating the Euclidean norm, i.e. the length of the inter-particle vector corresponding to each edge.

In graph theory, a path is a sequence of distinct vertices. A Hamiltonian path is a path that visits each vertex of the graph exactly once. In a complete graph, Hamiltonian paths are guaranteed to exist. We note here that the chain structures of AR-CHAIN are in fact Hamiltonian paths. The problem of finding whether a Hamiltonian path exists in a given graph is NP-complete, i.e. in the practical point of view meaning no solution in polynomial time $\mathcal{O}(N^k)$ with $k > 0$ exists. Furthermore, it can be shown that there is no polynomial time algorithm to find the shortest Hamiltonian path of a complete graph (i.e. shortest chain) either. The computational upper limit of a brute force approach to constructing the shortest chain scales as $\mathcal{O}(N!)$. Thus, strictly speaking, our procedure in finding the chain of inter-particle vectors in Section 2.2 corresponds to finding only the approximately shortest Hamiltonian path of the system. Consequently, there are usually more optimal chained coordinate systems than the one we find with our chain construction algorithm, but going through all the possible chain configurations is not feasible.

A spanning tree $T = (V, E_{\rm T})$ of the graph G is a subgraph of G connecting all the vertices of the original graph with a minimum possible number of its edges. A spanning tree connecting N vertices has $N - 1$ edges, the same as the number of inter-particle vectors in the chain structure of AR-CHAIN. In addition, T is an MST of G if the sum of the edge weights w in T is the smallest among all the possible spanning trees of G. It turns out that if the graph G has unique edge weights w there is a unique MST T. This is usually the case in practical applications of MSTs, such as our N-body systems. Unlike for the shortest Hamiltonian path problem, there are algorithms that find the MST for a given complete graph in a polynomial time.

There are two important graph properties that aid in finding the MST or in filtering out edges that are certainly not in the MST of a graph. The first is the cycle property. A cycle C of graph G is a path that forms a loop. The cycle property states that the edge $E_{\rm i}$ with the highest weight in any cycle C in G cannot be in the MST of G. The second property is the cut property. We divide the vertices $V_{\rm i}$ of G arbitrarily into two distinct groups. The essence of cut property is that the edge $E_{\rm ij}$ with the minimum weight connecting the two vertex groups is necessarily in the MST of G.

3.2 Finding the MST: Prim's algorithm

The problem of efficiently finding the MST of a given graph has been extensively studied in the literature, the classic solutions to the problem being found by Borůvka (1926), Kruskal (1956), and Prim (1957). We select the classic Prim's algorithm due to its relative simplicity and the fact that the algorithm somewhat resembles the chain construction in our AR-CHAIN implementation with the difference that the chain is now allowed to branch. In addition, Prim's algorithm makes labelling the edge vectors easier for our purposes than the other two classic algorithms.

Prim's algorithm proceeds as follows. First, one selects a single vertex $V_{\rm i}$ of the graph G. For N-body systems, we suggest starting from the particle spatially closest to the centre of mass of the system. Next, the edge $E_{\rm ij}$ with a minimum weight w connected to the first vertex is found and added as the first edge to the MST. Then the algorithm proceeds by finding the successive minimum weight edges among the edges in G connected to the MST and adds them into the MST until all vertices are connected with the MST. Our parallel implementation uses sorted adjacency lists on different tasks to effectively find the consecutive edges to add to the MST. For complete graphs, even the most sophisticated MST-finding algorithms typically scale as $\mathcal{O}(N^2)$ as the Prim's algorithm does.

The crucial difference between the chain construction and Prim's algorithm manifests itself here. In the chain construction, it is allowed to add new inter-particle vectors only to the tail and the head of the chain while in Prim's algorithm it is allowed to add new edges to any location in the MST. This ensures that spatially close N-body particles are always located near each other in the MST data structure, which is necessarily not the case in the chain. The differences between a chain and an MST built on a same group of particles are illustrated in Fig. 1.

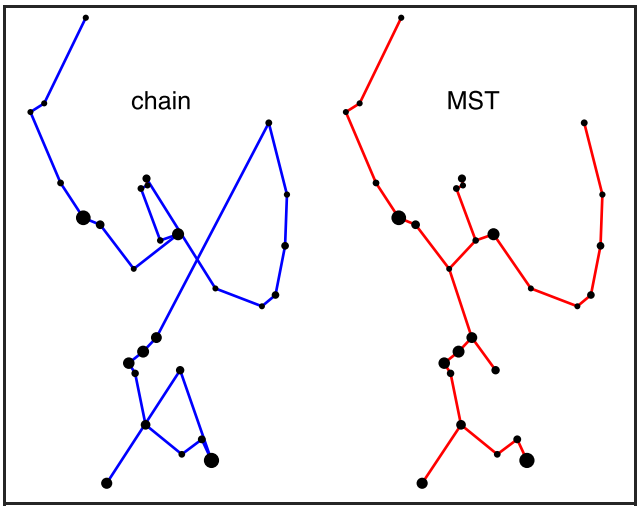

Figure 1. A two-dimensional illustration highlighting the difference between a chained coordinate system and the MST coordinates. Both the chain and the MST are constructed on a collection of 28 points corresponding to the locations of the brightest stars in the familiar constellation of Orion. The total length of the MST edges is smaller than the length of the chain. The chain also occasionally suffers from the fact that spatially close particles might be distant in the chain, which the MST avoids by branching its tree structure.

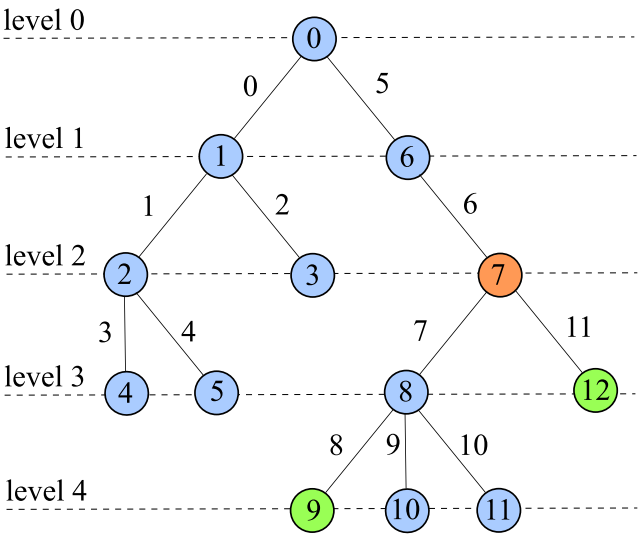

Figure 2. A tree structure with 12 vertices, 11 edges, and 5 levels. The parent vertex of vertices V_2 and V_3 is the vertex V_1. In addition, the vertex V_1 is the 2nd ancestor of vertex V_4. The lowest common ancestor of vertices V_9 and V_{12} (both in green) is the vertex V_7 (in orange).

3.3 Finding the MST: divide-and-conquer method

It is possible to quickly find spanning trees T with a total weight very close to the total weight of the MST of the graph (e.g. Wang, Wang & Mitchell Wilkes 2009; Zhong et al. 2013). For simplicity, we also refer to these approximate minimum spanning trees as MSTs throughout this study.

Our preferred method is the divide-and-conquer approach to the Prim's algorithm. First one divides the original graph G into $\sim\sqrt{N}$ subgraphs $G^{'}$. We use a simple octree-based spatial partition to find these subgraphs. A so-called meta-graph $G^{''}$ is formed by contracting the subgraphs $G^{'}$ into vertices of $G^{''}$ and including all possible edges between the vertices. The edge weights of the meta-graph are the minimum edge weights between the subgraphs by the cut property. Next, we use Prim's algorithm to construct the MSTs of each $G^{'}$ and the meta-graph $G^{''}$. To speed up the local MST construction, we eliminate edges that cannot be in the final MST using the cycle property before applying Prim's algorithm. Now we have all the edges of the total MST, which are then ordered and labelled by performing a standard tree walk.

We find that the spanning trees T found using the approximate algorithm are typically 5 per cent longer than the true MST of the graph. However, this difference is not a serious problem as the spanning trees are locally true MSTs of the subgraphs. Furthermore, our motivation to use MSTs is to minimize numerical error originating mostly from computation operations involving spatially close particles by choosing a clever local coordinate system. In addition, the divide-and-conquer approach is faster than the original Prim's algorithm. Concerning the speed of the algorithms, we find that in our code it is always profitable to use the divide-and-conquer method when the particle number is $N \gtrsim 10^3$. With smaller particle numbers, both approaches yield very similar results as the wall-clock time elapsed in the MST construction is negligible.

3.4 MST as a coordinate system

As both the chain and the MST consist of $N_{\rm part} - 1$ inter-particle vectors, the MST coordinate system can be constructed with a recipe similar to the chained coordinates in equation (6) with relatively small modifications to the procedure.

First, we need to index the MST edge vectors $E_{\rm ij}$ as in equation (6). In the chain construct, setting the chained indices is straightforward as one simply starts the indexing from the tail of the chain and proceeds towards the head. In the MST, indexing is more complicated because of the branching of the MST structure. However, we can take full advantage of the fact that the MST is a tree structure. The first chosen vertex V_0 is the root vertex of the MST and gets the index and level zero in the MST, i.e. $L(V_0) = 0$. We index the subsequent vertices and edge vectors in the order they were added to the MST. The vertex levels and the parents of the vertices are assigned simultaneously as well. A simplified illustration describing the indexing of our MST structure is presented in Fig. 2. After the indexing, the inter-particle vectors corresponding to the MST edges can be computed just as in equation (6) with the additional rule that all the inter-particle vectors point away from the root vertex, just as the chained vectors are oriented away from the tail towards the head of the chain.

Next, we generalize equation (9) to determine the rules when two vertices are close to each other in the MST, i.e. within $N_{\rm d}$ edges of each other, just as in the chain construct. For the force calculation of the nearby vertices, the MST edge vectors are used while the original coordinate system is used for the rest of the vertex pairs. The criterion of two vertices $V_{\rm i}$ and $V_{\rm j}$ being within $N_{\rm d}$ MST edges of each other can be conveniently expressed by using the lowest common ancestor (LCA) of the two vertices. The parent vertex is the 1st ancestor of the vertex, the 2nd ancestor is the parent of the parent vertex, and so on. The LCA is the vertex among the common ancestors of both $V_{\rm i}$ and $V_{\rm j}$ that has the highest level in the MST. We label this vertex $V_{\rm LCA}$. Note that $V_{\rm i}$ or $V_{\rm j}$ itself may be the $V_{\rm LCA}$ of the vertex pair. Now we can state that if

$$|L(V_{\rm i}) - L(V_{\rm LCA})| + |L(V_{\rm j}) - L(V_{\rm LCA})| \leq N_{\rm d} \qquad (17)$$

the two vertices $V_{\rm i}$ and $V_{\rm j}$ are close to each other in the MST. Here, $L(V_{\rm i})$ again signifies the level of the vertex $V_{\rm i}$.

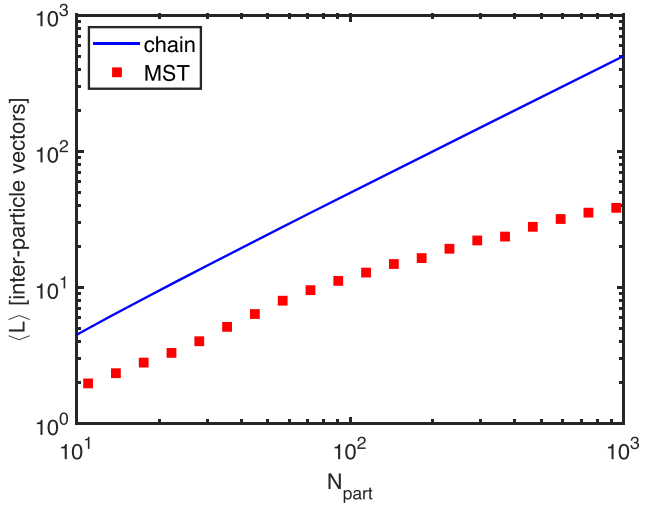

Figure 3. Comparing the mean length $\langle L_{\rm chain}\rangle$ of the chain structure (solid blue line) and the mean level $\langle L_{\rm MST}\rangle$ of particles in an MST (red symbols). The particle distribution on which the coordinate systems are build follows a $\rho(r) \propto r^{-1}$ density profile. We see that the mean level of particles in the inter-particle coordinate structure is always lower in the MST by a factor of ∼2–10. Changing the underlying density profile has a very small effect on the result. The small mean particle level corresponds to a smaller level of floating-point error when constructing and deconstructing the coordinate systems as discussed in the text.

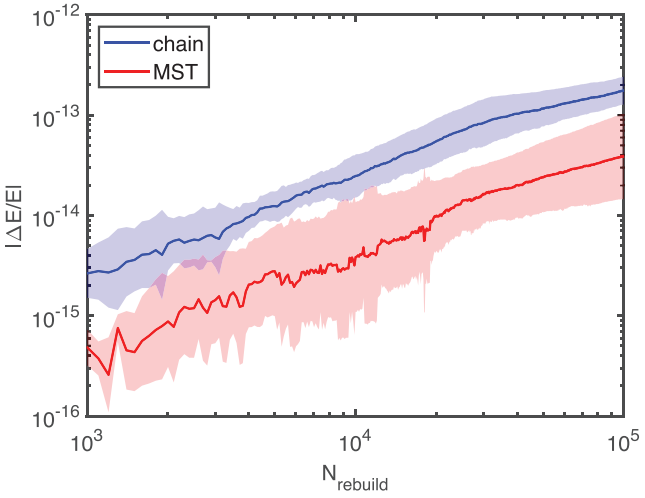

Figure 4. The build, deconstruct, and rebuild test of the two coordinate systems on a simple N-body system with $N_{\rm part} = 379$ particles. With the chained coordinates (blue line), the energy error is approximately an order of magnitude larger than with the MST coordinates (red line). The results are averages over 10 N-body systems with differing random seeds. The coloured regions represent the scatter of one standard deviation.

Finally, we write down the recipe for selecting which vectors to use in the force calculation. If $V_{\rm j}$ and $V_{\rm i}$ are within $N_{\rm d}$ edges of each other, we simply walk the MST from $V_{\rm i}$ to $V_{\rm j}$ via $V_{\rm LCA}$ and sum the traversed edge vectors to obtain the MST separation vector $\boldsymbol{X}_{\rm k}$, just as in equation (9) with the chain. If the condition of equation (17) does not hold, we use the original coordinate vectors to compute the separations $\boldsymbol{r}_{\rm j} - \boldsymbol{r}_{\rm i}$.

3.5 Numerical error from consecutive coordinate transformations

During integration, the MST (or chain) coordinate structure is frequently built, deconstructed, and rebuilt to keep the coordinate structure up to date as the simulation particles move. For N-body systems with a large number of particles, these coordinate transformations require a large number of summation operations (e.g. Mikkola & Aarseth 1993), which introduces a new source of numerical error in the integration. This fact has received little attention in the literature thus far, most probably due to the fact that for few-body systems ($N_{\rm part} \lesssim 10$) the accumulated summation errors always remain small. The chain construction and deconstruction both require summing on average $\langle L_{\rm chain}\rangle = (N_{\rm part} - 1)/2$ coordinate vectors. For MST, the corresponding number is the mean level of the vertices in the MST, i.e. $\langle L_{\rm MST}\rangle = \langle L(V_{\rm i})\rangle$. Now the choice of the root vertex in Section 3.1 becomes important. If the root vertex is spatially close to the centre of mass of the system, $\langle L_{\rm MST}\rangle$ should always be smaller than $\langle L_{\rm chain}\rangle$. We demonstrate this fact in Fig. 3. Our results show that $\langle L_{\rm chain}\rangle/\langle L_{\rm MST}\rangle \sim 2$–10 with particle numbers of $N_{\rm part} = 10$–1000.

This difference somewhat affects the numerical performance of the two algorithms. We perform an additional numerical test in which we build, deconstruct, and rebuild the two coordinate systems consecutively $N_{\rm rebuild} = 10^5$ times while monitoring the accumulating numerical error. The N-body system on which the coordinate systems are built contains $N_{\rm part} = 379$ particles with its full details presented in Section 6.1. The results of the rebuild test are presented in Fig. 4. The results indeed show that the MST coordinate system accumulates less numerical error than the chained coordinates, the difference being approximately an order of magnitude in energy error. Apart from the difference of an order of magnitude, the cumulative error behaves very similarly with the two coordinate systems. As the energy error in the chain coordinate test reaches $|\Delta E/E| \sim 10^{-13}$ after $N_{\rm rebuild} = 10^5$ rebuilds, we conclude that the MST coordinate system is recommended for regularized N-body simulations requiring extremely high numerical accuracy.

Finally, we note that advanced floating-point summation methods such as the Kahan summation (Kahan 1965) or the Neumaier summation (Neumaier 1974) algorithm might be used to further diminish the numerical error from the coordinate transformation operations. However, the inclusion of such more advanced summation algorithms is left for future work, as our current MSTAR code is numerically accurate enough for all current target applications.

4 PARALLEL EXTRAPOLATION METHOD

4.1 Force loop parallelization

The most straightforward way to begin to parallelize a serial AR integrator is to parallelize the force computation of the code. The MPI parallelization standard is adopted for this study. We use the basic parallelization strategy in which the $\mathcal{O}(N_{\rm part}^2)$ iterations of the force computing loop are divided evenly for $N_{\rm force}$ MPI tasks, speeding up the loop calculation. However, the inter-task communication required to collect the final results after the force computations uses an increasing amount of CPU time when the number of MPI tasks is increased.

In a serial direct summation integrator using a GBS extrapolation method, the total wall-clock time T elapsed for the force calculation

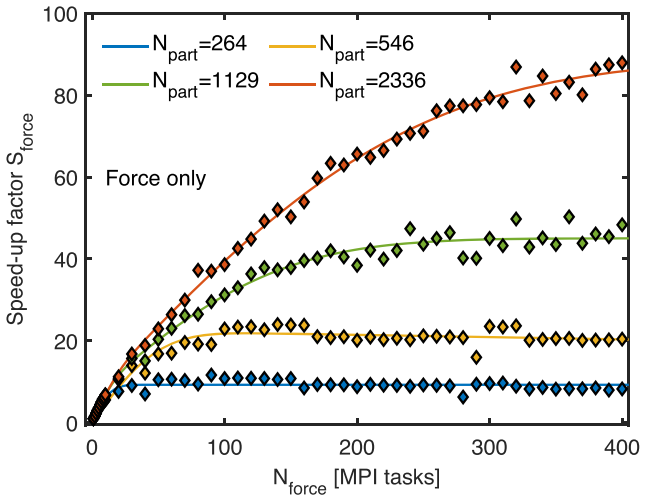

Figure 5. The speed-up factor of the force computation $S_{\rm force}$ as a function of the number of MPI tasks $N_{\rm force}$ for four different N-body particle numbers $N_{\rm part}$. The symbols represent the measured force computation speed-up factors while the solid lines are error function fits (equation 22) to the speed-up data as described in the text. The speed-up factor initially grows linearly when $N_{\rm force} \ll N_{\rm part}$ and saturates to a roughly constant level after $N_{\rm force} \gtrsim 0.1 \times N_{\rm part}$.

during the integration of a single step H can be expressed as

$$T \approx \sum_{k=1}^{k_{\rm max}} n_{\rm k} t N_{\rm part}^2 = t N_{\rm part}^2 \sum_{k=1}^{k_{\rm max}} n_{\rm k}, \tag{18}$$

where $k_{\rm max}$ is the maximum number of GBS substep divisions and t is a proportionality constant. Without the GBS algorithm, the sum expression would not appear in the formula. Thus, the wall-clock time elapsed in the force calculation depends not only on the particle number $N_{\rm part}$ but also on the sequence $n_{\rm k}$ introduced in equations (10) and (11).

We define the speed-up factor of the parallelized force computation as

$$S_{\rm force}(N_{\rm force}) = \frac{T_{\rm serial}}{T_{\rm parallel}(N_{\rm force})} \tag{19}$$

in which $T_{\rm serial}$ and $T_{\rm parallel}$ are the respective wall-clock times elapsed during the force computation. Assuming ideal zero-latency communication between the tasks, the speed-up factor $S_{\rm force}$ of the parallelized force calculation scales initially linearly with the number of MPI tasks, i.e. $S_{\rm force} \propto N_{\rm force}$. The linear scaling behaviour continues until the number of tasks equals the particle number $N_{\rm force} = N_{\rm part}$ at which point one MPI task handles one particle. After this, the speed-up factor $S_{\rm force}$ remains constant. With realistic non-instantaneous inter-task communication, the flat scaling behaviour is reached with $N_{\rm force}$ well below the particle number $N_{\rm part}$ due to the increasing communication costs.

We illustrate the results of a series of force computation scaling tests in Fig. 5. The N-body systems in the test are selected from our sample of initial conditions with logarithmically spaced particle numbers in the range $10^1 \leq N_{\rm part} \leq 10^4$. The results are averages over three different random realizations of the systems. In the scaling test, we use $1 \leq N_{\rm force} \leq 400$ MPI tasks. With a small number of MPI tasks ($N_{\rm force} \lesssim 10$), the speed-up factor increased linearly, as expected. After the linear phase, $S_{\rm force}(N_{\rm force})$ flattens to a constant function at higher $N_{\rm force}$. Eventually, the speed-up factor actually begins to decrease as the communication costs start to dominate over the time elapsed in the force computation. We define the maximum reasonable task number $N_{\rm force}^{\rm max}$ for N-body systems as the task number in which ∼95 per cent of the maximum speed-up has been achieved, i.e. $S_{\rm force}(N_{\rm force}^{\rm max}) = 0.95 \times S_{\rm force}^{\rm max}$. When $N_{\rm force} \gtrsim N_{\rm force}^{\rm max}$, the addition of subsequent MPI tasks has only a negligible effect on the speed-up of the force computation. We find in our scaling experiments that $N_{\rm force}^{\rm max}$ can be approximated with a simple relation

$$N_{\rm force}^{\rm max} \approx q \times N_{\rm part} \tag{20}$$

in which the constant factor is between $0.05 \leq q \leq 0.1$. In addition, the maximum force computation speed-up factor $S_{\rm force}^{\rm max}$ can be approximated with the formula

$$\log_{10}(S_{\rm force}^{\rm max}) \approx a_1 \log_{10}(N_{\rm part}) - a_2, \tag{21}$$

with $a_1 \approx 0.505$ and $a_2 \approx -0.58$. For quantifying the behaviour of the speed-up factor $S_{\rm force}$ in the intermediate range of MPI tasks between the linearly increasing and the constant speed-up factor, a suitable fitting function is required. A suitable choice is the error function erf that has the correct asymptotic behaviour both with small and large values of its argument. We use fitting functions of the form

$$S_{\rm force}(N_{\rm force}) = \sum_{\rm i}^{N_{\rm coeff}} b_{\rm i}\, {\rm erf}\,(c_{\rm i} N_{\rm force}) \tag{22}$$

in which $b_{\rm i}$ and $c_{\rm i}$ are constant coefficients. As expected, $\sum_{\rm i}^{N_{\rm coeff}} b_{\rm i} \approx S_{\rm force}^{\rm max}$ by definition. We find that using $N_{\rm coeff} \approx 1$–2 terms yields good results. The fit coefficients are presented in Table 1 and are later used in Section 4.3 to estimate the optimal division of computational resources when additional layers of parallelization are implemented into the extrapolation method of the integrator.

Table 1. The force speed-up coefficients $b_{\rm i}$ and $c_{\rm i}$ obtained by fitting the data from Fig. 5 using the function equation (22). For the smallest particle number $N_{\rm part} = 264$, fits beyond a single term were not profitable.

$N_{\rm part}$	b_1	c_1	b_2	c_2
264	3.60	5.10	×	×
546	3.24	5.33	3.16	4.97
1129	5.03	9.08	4.02	31.87
2336	6.50	12.35	6.09	79.49

4.2 Substep division parallelization

Solving initial value problems numerically for individual ordinary differential equations was long considered to be an inherently sequential process. However, numerical techniques employing extrapolation methods are an important exception to this rule (e.g. Rauber & Rünger 1997; Korch, Rauber & Scholtes 2011, and references therein). As the N-body problem is an initial value problem for a coupled system of ordinary differential equations, it is possible to introduce another layer of parallelization besides the parallel force loop computation into N-body codes that use extrapolation methods. To our best knowledge, the only work studying orbital dynamics with a code including a parallelized extrapolation method is the study by Ito & Fukushima (1997). Unfortunately, this pioneering study has not received wide attention in the literature.

For the MSTAR code implementation, we use the Neville–Aitken algorithm (e.g. Press et al. 2007) for polynomial extrapolation. Error control is implemented as in equation (13) by studying the

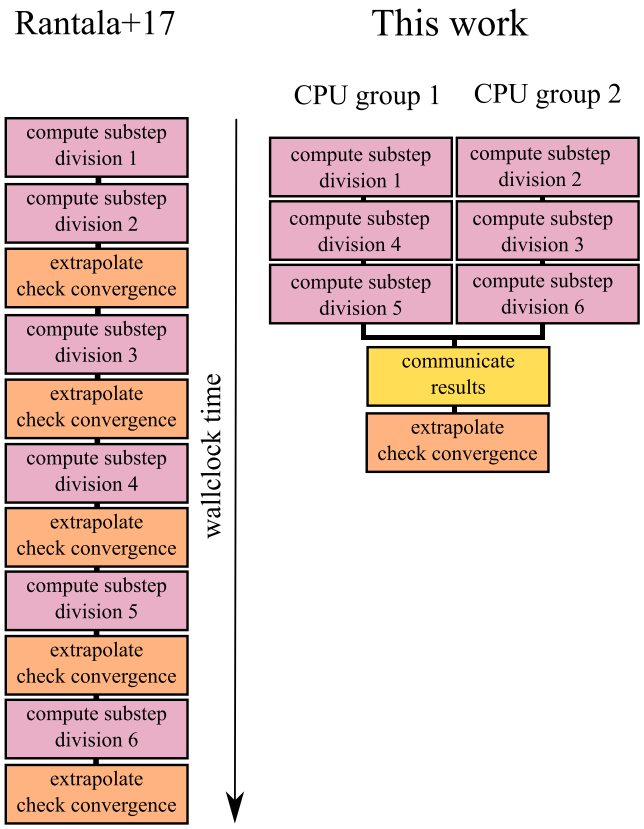

Figure 6. The parallelization strategy of the GBS extrapolation method without substep parallelization as in our previous AR-CHAIN implementation (left) and including it in our new MSTAR code (right). In this example, the extrapolation method converges after six substep divisions. The previous extrapolation method computes the different substep divisions, tries extrapolation, and checks convergence in a sequential manner. The parallelized extrapolation method computes the different substep divisions in parallel using different CPU groups, communicates the results to each node, and then performs the extrapolation procedure.

relative difference of consecutive extrapolation results. We chose polynomials over rational functions for extrapolation as we have observed that using rational functions may occasionally lead to spurious extrapolation results even though the convergence criteria of equation (13) are fulfilled. We do not use the sophisticated recipes intended to optimize the amount of computational work per unit step in serial extrapolation methods, present in some GBS implementations (Press et al. 2007; Hairer et al. 2008).

Implementing another parallelization layer into an AR integrator begins with the observation that the computation of a single substep division in the GBS method is independent of the other required substep divisions. Thus, the different substep divisions can be integrated using different CPU groups in parallel (Rauber & Rünger 1997), after which the results of subdivisions are communicated and the actual extrapolation is performed. The Neville–Aitken polynomial extrapolation for all the dynamical variables can be parallelized as well. In this work, we focus on the case of a single N-body system, but the method described here can be extended to integrate multiple independent N-body systems simultaneously. A simple example with two CPU groups and six substep divisions is illustrated in Fig. 6. We label the number of CPU groups $N_{\rm div}$. As we use a single MPI task per CPU, the total number of CPUs, $N_{\rm CPU}$, the number of CPU groups, $N_{\rm div}$, and the number of CPUs in force computation, $N_{\rm force}$, are connected by the simple relation

$$N_{\rm CPU} = N_{\rm div} \times N_{\rm force}. \tag{23}$$

As stated in Section 2.3, the standard GBS method computes the substep divisions in a sequential manner, calculating subsequent substep divisions until the results converge or the user-given maximum number of substep divisions $k_{\rm max}$ is reached. The parallelization of the substep divisions requires that the number of substep divisions to be computed must be set by user in advance in our implementation. We call this fixed number of substep divisions $k_{\rm fix}$. We note that techniques to use a non-fixed $k_{\rm max}$ exist even with parallelization (Ito & Fukushima 1997) but the simple approach with a fixed number of subdivisions has proven to be sufficient for our purposes. The optimal value for $k_{\rm fix}$ depends on the GBS tolerance parameter $\eta_{\rm GBS}$ and the particular N-body system in question. Thus, numerical experimentation is needed for determining $k_{\rm fix}$. If $k_{\rm fix}$ is set to too low a value, the extrapolation method must shorten the time-step H in order to reach convergence, increasing the running time of the code. On the other hand, if $k_{\rm fix}$ is too high, extra computational work is performed as the extrapolation would have converged with fewer substep divisions. However, this slowdown is somewhat compensated by the fact that longer time-steps H can be taken with a higher $k_{\rm fix}$.

The number of CPU groups $N_{\rm div}$ can have values between $1 \leq N_{\rm div} \leq k_{\rm fix}$, leaving $N_{\rm force} = N_{\rm CPU}/N_{\rm div}$ for the parallelization of the force computation. The individual substep divisions are divided into the $N_{\rm div}$ CPU groups with the following recipe. Initially, each CPU group has the computational load $C_{\rm i} = 0$. Starting from the substep division with the highest number of substeps, i.e. $\max(n_{\rm k}) = 2k_{\rm fix}$, we assign the substep divisions one by one into the CPU group that has the lowest computational load $C_{\rm i}$ at that moment until no divisions remain. If there are several CPU groups with the same computational load, we select the first one, i.e. the CPU group with the lowest CPU group index in the code. Keeping $N_{\rm force}$ fixed, we define the parallel substep division speed-up $S_{\rm div}$ as

$$\begin{aligned} S_{\rm div}(N_{\rm div}) &= \frac{T_{\rm serial}}{T_{\rm parallel}(N_{\rm div})} = \frac{\sum_{k=1}^{k_{\rm fix}} n_k}{\max_i C_i} \\ &= \frac{\sum_{k=1}^{k_{\rm fix}} n_k}{\max_i \left(\left[\sum_j^{N_{\rm i}} n_{k_j} \right]_i \right)}. \end{aligned} \tag{24}$$

If $N_{\rm div} = 1$, there is no speed-up in the force computation and the running time is the same as in equation (18). When $N_{\rm div} > 1$, there is a wall-clock time speed-up as the force computation is divided into multiple CPU groups. With $N_{\rm div} = k_{\rm fix}$, we have $S_{\rm div} = (k_{\rm fix} + 1)/2$ assuming the Deuflhard sequence from equation (11).

Now we can compute the speed-up factor $S_{\rm div}$ once $k_{\rm fix}$ and $N_{\rm div}$ are set. The computed results are presented in Fig. 7. The speed-up factor follows the line $S_{\rm div} = N_{\rm div}$ until the point $N_{\rm div} = k_{\rm fix}/2$ is reached, after which the $S_{\rm div}(N_{\rm div})$ rapidly flattens into the constant value of $S_{\rm div} = (k_{\rm fix} + 1)/2$. Thus, the maximum reasonable number of CPU groups is $N_{\rm div}^{\rm max} = \lceil k_{\rm fix}/2 \rceil$ in which we use the ceiling function $\lceil \cdot \rceil$.

4.3 Speed-up with full parallelization

Now we are ready to combine the force loop and the substep division layers of parallelization. The primary advantage of using two layers of parallelization compared to the simple force loop computation parallelization is that we can efficiently use more MPI tasks to speed up the MSTAR integrator. Without the subdivision parallelization, it

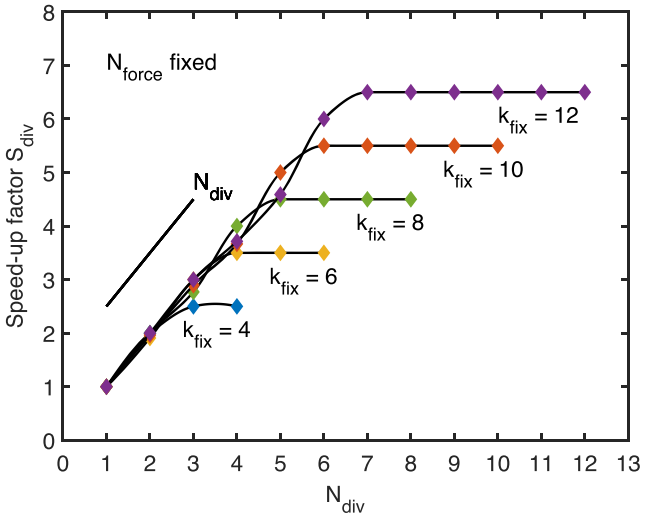

Figure 7. The speed-up factor $S_{\rm div}$ for the parallelized substep divisions (symbols) and their interpolated continuous counterparts (solid lines) as a function of the number of substep divisions $k_{\rm fix}$. The interpolants are only shown to clarify the visualization as there are no groups with non-integer $k_{\rm max}$. We see that $S_{\rm div} = N_{\rm div}$ until $N_{\rm div} = k_{\rm fix}/2$ after which the speed-up factor attains a constant value of $S_{\rm div}^{\rm max} = (k_{\rm fix} + 1)/2$.

is not reasonable to use more than $N_{\rm force} \approx 0.1 \times N_{\rm part}$ MPI tasks as shown in Section 4.1. With the subdivision parallelization included, the maximum reasonable CPU (or task) number becomes $N_{\rm CPU} \approx 0.05 \times k_{\rm fix} N_{\rm part} = 0.4 \times N_{\rm part}$ with the typical value of $k_{\rm fix} = 8$. This is the value of $k_{\rm fix}$ we use in the simulations of this study.

Next, we estimate how the computational resources should be divided to ensure the maximum total speed-up factor $S_{\rm total}$ if the number of CPUs, $N_{\rm CPU}$, and thus MPI tasks are fixed. The values of $N_{\rm part}$ and $k_{\rm fix}$ are assumed to be fixed as well. The optimal division of computational resources corresponds to finding the maximum of the function

$$\begin{aligned} S_{\rm total}(N_{\rm force}, N_{\rm div}) &= S_{\rm force}(N_{\rm force}) \times S_{\rm div}(N_{\rm div}) \\ &= S_{\rm force}(N_{\rm CPU}/N_{\rm div}) \times S_{\rm div}(N_{\rm div}) \end{aligned} \tag{25}$$

with the constraints $N_{\rm force}, N_{\rm div} \in \mathbb{N}$ and $N_{\rm CPU} = N_{\rm force} \times N_{\rm div}$. For arbitrary $N_{\rm CPU}$, there are typically only a few solutions. For the force computation speed-up $S_{\rm force}$, we need to use the approximate methods, i.e. the fitting function equation (22) and its coefficients $b_{\rm i}$ and $c_{\rm i}$ from Table 1. The substep division speed-up factor $S_{\rm div}$ can be exactly estimated by using equation (24).

In Fig. 8, we present the speed-up factor $S_{\rm total}$ for four different particle numbers and four different values for $N_{\rm CPU}$. We set $k_{\rm max} = 8$ for each of the 16 combinations of the particle number and the number of CPUs. For a fixed particle number, the total speed-up factor $S_{\rm total}$ increases until $N_{\rm CPU} \sim 0.4 \times N_{\rm part}$. We find that the maximum of $S_{\rm total}$ is typically located near $\lceil N_{\rm div}/2 \rceil$, which corresponds to finding the optimal $N_{\rm force}$ around $N_{\rm CPU}/(2N_{\rm div})$.

However, we find that the best strategy for finding the optimal pair $(N_{\rm force}, N_{\rm div})$ is to relax the requirement of having a pre-set value for $N_{\rm CPU}$. One computes the values for $S_{\rm total}$ for all the integer pairs $(N_{\rm force}, N_{\rm div})$ satisfying $1 \le N_{\rm force} \le \lceil 0.1 \times N_{\rm part} \rceil$ and $1 \le N_{\rm div} \le \lceil k_{\rm fix}/2 \rceil$. The location of the maximum value of $S_{\rm total}$ determines which values of $N_{\rm force}$ and $N_{\rm div}$, and thus also $N_{\rm CPU}$, should be used. Additional constraints such as the number of CPUs per supercomputer node should also be taken into account, i.e. $N_{\rm CPU}$ should be a multiple of this number.

Finally, we present the results of a strong scaling test of our force calculation algorithms in Fig. 9. In a strong scaling test, the problem size remains fixed while the number of CPUs is increased. We examine both the force loop parallelized version and the code with full parallelization. We see that the force calculation algorithm with full parallelization follows the ideal scaling behaviour to higher CPU numbers than the force loop parallelized version. With the fully parallelized force computation, one can use CPU numbers approximately up to $N_{\rm CPU} \sim 0.5 \times N_{\rm part}$ before the scaling begins to deviate from the ideal case. Including only the force loop parallelization, the scaling behaviour becomes non-ideal with roughly 10 times smaller $N_{\rm CPU}$. We tested the force algorithms up to $N_{\rm CPU} = 400$ and all fully parallelized tests with particle numbers $N_{\rm part} \gtrsim 10^3$ followed ideal scaling. The total speed-up factors $S_{\rm total}$ are consistent with our estimations in this section.

5 CODE ACCURACY: FEW-BODY TESTS

5.1 Eccentric Keplerian binary

Next, we demonstrate the numerical accuracy of our MSTAR integrator by studying the standard Keplerian two-body problem by comparing the results to the analytical solution and to our AR-CHAIN code (Rantala et al. 2017). In the following sections, we also run tests with two additional three-body set-ups.

All the simulation runs of this study are run on the FREYA[1] cluster of the Max Planck Computing and Data Facility (MPCDF). Each computation node of FREYA contains two Intel Xeon Gold 6138 CPUs totalling 40 cores per computation node. However, in the context of this article, we refer to these core units as CPUs.

The Keplerian two-body problem is completely described by its six integrals of motion. As the final integral, the periapsis time can be arbitrarily chosen; we only need five integrals of motion to describe the orbit. The first conserved quantity is the energy E of the two-body system, defined as

$$E = \frac{1}{2}\mu \|\boldsymbol{v}\|^2 - \frac{G\mu M}{\|\boldsymbol{r}\|} \tag{26}$$

in which $M = m_1 + m_2$, $\mu = m_1 m_2/M$ and $\boldsymbol{r}$ and $\boldsymbol{v}$ are the relative position and velocity vectors, respectively. The energy of the two-body system uniquely defines its semimajor axis a as

$$a = -\frac{G\mu M}{2E}. \tag{27}$$

Next, the conserved angular momentum vector $\boldsymbol{L}$ is defined as

$$\boldsymbol{L} = \mu \boldsymbol{r} \times \boldsymbol{v}. \tag{28}$$

Together, E and $\boldsymbol{L}$ determine the orbital eccentricity e of the two-body system as

$$e = \left(1 + \frac{2EL^2}{G\mu^3 M^2}\right)^{1/2}. \tag{29}$$

Finally, we have the constant Laplace–Runge–Lenz vector

$$\boldsymbol{A} = \mu \boldsymbol{v} \times \boldsymbol{L} - GM\hat{\boldsymbol{r}} \tag{30}$$

in which $\hat{r} = \boldsymbol{r}/\|\boldsymbol{r}\|$. The Laplace–Runge–Lenz vector lies in the orbital plane of the two-body system pointing towards the periapsis. As we have now in total seven conserved quantities and only five

[1] www.mpcdf.mpg.de/services/computing/linux/Astrophysics

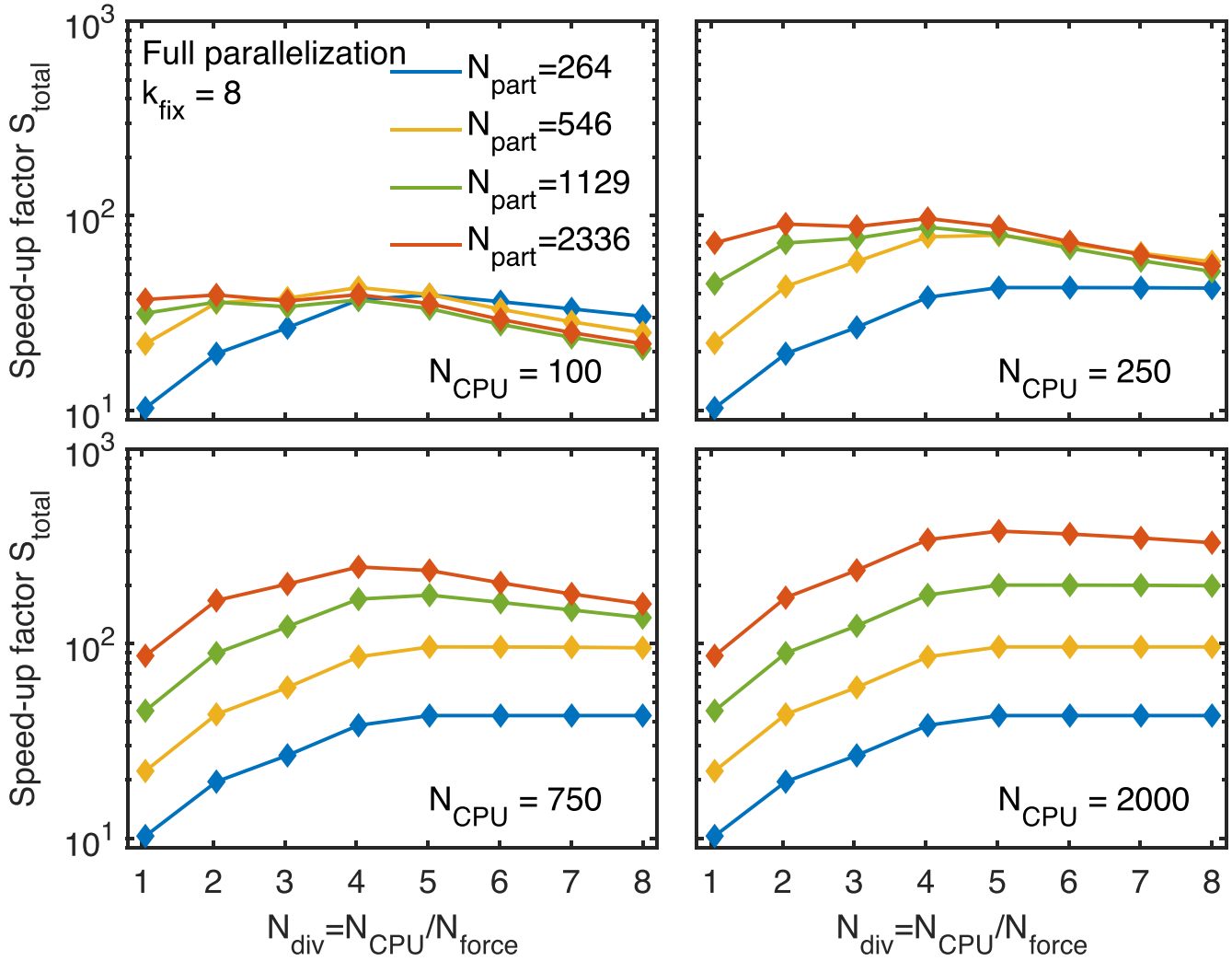

Figure 8. An example of finding the optimal division of computational resources between the force and substep parallelization. The substep part of the speed-up factor can be computed analytically while the force parallelization part is estimated by using simple numerical tests as explained in the text. Starting from the top-left corner, the four panels have increasing CPU numbers of $N_{\rm CPU} = 100$, 250, 750, and 1000. Each panel shows the speed-up factor $S_{\rm total}$ as a function of the number of CPU groups $N_{\rm div}$ for four different particle numbers of $N_{\rm part} = 264$ (blue line), $N_{\rm part} = 546$ (yellow), $N_{\rm part} = 1129$ (green line), and $N_{\rm part} = 2336$ (red line). The maximum speed-up factor $S_{\rm total}^{\rm max}$ is typically found near $N_{\rm div} = k_{\rm fix}/2$. The corresponding number of CPUs for the force computation is obtained by using the relation $N_{\rm force} = N_{\rm CPU}/N_{\rm div}$.

integrals are required, the conserved quantities cannot be independent. The first relation is simply $\boldsymbol{A} \cdot \boldsymbol{L} = 0$ while the non-trivial second relation reads $e = \|\boldsymbol{A}\|/(GM)$, connecting both the energy E and the norm of the angular momentum vector L to the norm of $\boldsymbol{A}$.

It is convenient to study the accuracy of a numerical integrator by observing the behaviour of E, $\boldsymbol{L}$, and $\boldsymbol{A}$ during a simulation run. Symplectic integrators such as our chained leapfrog typically conserve quadratic invariants such as the angular momentum exactly and the energy approximately but with no long-term secular error growth (Hairer, Lubich & Wanner 2006). However, the Laplace–Runge–Lenz vector is a third-order invariant, and its conservation is not guaranteed. Thus, the orbit can precess in its orbital plane (e.g. Springel et al. 2005). This makes the rotation angle of the Laplace–Runge–Lenz vector

$$\theta_{\rm LRL} = \arctan(A_{\rm y}/A_{\rm x}) \tag{31}$$

a very suitable probe for testing the accuracy of an integrator.

We perform a series of two-body simulations both with our MSTAR integrator and our AR-CHAIN implementation. For the tests in this section, serial code implementations are used. We initialize 360 equal-mass SMBH binaries with $M = 2 \times 10^9\,{\rm M}_\odot$, $a = 2$ pc, and $e = 0.9$. We orient the initial binaries in a random orientation in space in order to have a sample of binaries with initially the same integrals of motion but with differing numerical errors during the simulation. We run the binary simulations for $T = 10^4 \times P$ in which P is the Keplerian orbital period of the binary. The GBS tolerance is set to $\eta_{\rm GBS} = 10^{-12}$. We always use $k_{\rm fix} = 8$ substep divisions in the serial GBS procedure.

The results of the binary simulations are presented in Figs 10 and 11. The panels of Fig. 10 illustrate the relative error of the energy and the norm of the angular momentum vector as well as the absolute rotation angle of the Laplace–Runge–Lenz vector. In addition, we show the maximum GBS error $\epsilon_{\rm GBS}$ after convergence in each step. Fig. 11 in turn presents the elapsed wall-clock time, the GBS step fail rate, and the length of the fictitious time-step during the simulations.

The results of the binary simulations systematically show that the new MSTAR implementation conserves the orbital energy E, angular

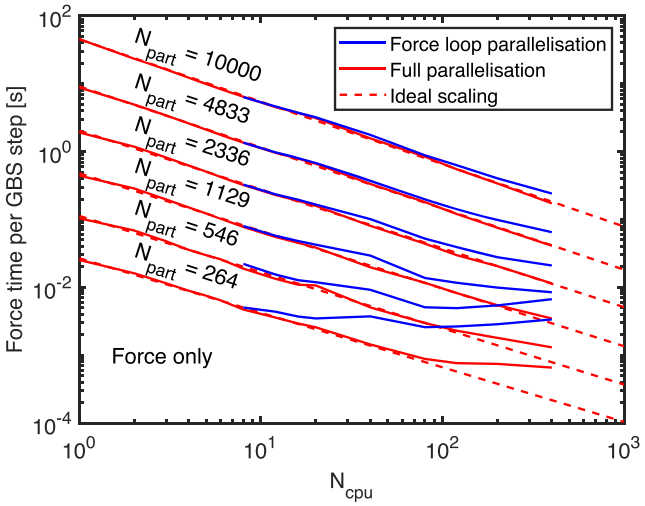

Figure 9. The strong scaling test of the force loop parallelized (solid blue line) and the fully parallelized (solid red line) force computation algorithms. The serial running time per GBS step corresponds to equation (18). The dashed red line shows the ideal scaling behaviour of the codes. The fully parallelized algorithm follows the ideal scaling behaviour up to $N_{\rm CPU} \sim 0.5 \times N_{\rm part}$ while the loop parallelized algorithm begins to deviate from the ideal scaling law already with roughly 10 times smaller CPU numbers.

momentum $\boldsymbol{L}$, and the Laplace–Runge–Lenz vector $\boldsymbol{A}$ ∼1–2 orders of magnitude better than our old AR-CHAIN implementation. In addition, the new code is faster than the old code by a factor of few. The difference in the code speed originates from the fact that the GBS recipe of Hairer et al. (2008) that we are using in our AR-CHAIN implementation optimizes the computational work per unit step and also aggressively attempts longer steps H after convergence. This leads to a large number of failed GBS steps, slowing down the code in the test. However, the implementation is not very transparent and thus it is somewhat difficult to point to exactly where the speed and accuracy differences originate compared to our own implementation of the extrapolation algorithm in MSTAR.

5.2 Pythagorean three-body problem

The Pythagorean three-body problem (Burrau 1913; Szebehely & Peters 1967) is a famous zero angular momentum set-up to test integrators and to study chaotic and orderly motion in a relatively simple gravitating system (Aarseth et al. 1994; Valtonen & Karttunen 2006). Three SMBHs with masses of $M_1 = 3 \times 10^8\,{\rm M}_\odot$, $M_2 = 4 \times 10^8\,{\rm M}_\odot$, and $M_3 = 5 \times 10^8\,{\rm M}_\odot$ are placed at the corners of a right-angled triangle with side lengths of $r_{13} = 30$ pc, $r_{12} = 40$ pc, and $r_{23} = 50$ pc, i.e. in such a manner that the least massive SMBH is opposite to the shortest side and so on. Initially, all velocities are set to zero. Once the simulation run is started, the three bodies experience a series of complicated interactions finally resulting in the ejection of the least massive M_1 body while the remaining two SMBHs form a bound binary recoiling to the opposite direction.

The final outcome of the system can be parametrized by the orbital elements ($a_{2,3}$, $e_{2,3}$) of the formed binary and the escape direction β_1. If the initial triangle set-up is oriented as in fig. 1 of Aarseth et al. (1994), the escape angle becomes

$$\beta_1 = \arctan(y_1/x_1) \tag{32}$$

in which the subscript refers to the least massive SMBH. The system is extremely sensitive to the initial conditions and the numerical accuracy of the used integrator, thus providing an ideal test set-up for demonstrating that our old AR-CHAIN and the new MSTAR implementations yield the same final results.

We perform the integration with the same code parameters as the two-body tests. We show the orbits of the three SMBHs in the Pythagorean three-body problem in Fig. 12 both with the old AR-CHAIN and the new MSTAR integrator. The overall behaviour of the system is as expected from the literature results: the least massive body becomes unbound and the binary of the two remaining bodies recoils in the opposite direction. At this level of scrutiny, there are no noticeable differences between the two integrator implementations.

The escape angle β_1, as well as the orbital elements of the pair of two most massive bodies, is presented in Fig. 13. After a period of complicated gravitational dynamics, the values of β_1, $a_{2,3}$, and $e_{2,3}$ settle to their final values as the motion of the system becomes ordered after the escape of the least massive SMBH. Both the AR-CHAIN and the MSTAR integrator implementations provide the results $\beta_1 \approx 71.4^\circ$, $a_{2,3} \approx 5.5$ pc, and $e_{2,3} \approx 0.99$ with a relative difference of only $\sim 10^{-4}$ in each value. Due to the extreme sensitivity of the Pythagorean three-body problem to numerical errors during integration, we conclude that the two integrators produce the same results within an accuracy sufficient for our purposes. These final results also agree very well with the literature values (Szebehely & Peters 1967; Aarseth et al. 1994).

5.3 Lidov–Kozai oscillations

The Lidov–Kozai mechanism (Kozai 1962; Lidov 1962) is a widely studied dynamical phenomenon present in a family of hierarchical three-body systems. The mechanism has a large number of applications in dynamical astronomy, reaching from dynamics of artificial satellites to systems of SMBHs (Naoz 2016). An inner binary consisting of a primary and a secondary body is perturbed by a distant orbiting third body. The inner binary and the perturber form the outer binary. The time-varying perturbation causes the argument of pericentre of the secondary body to oscillate around a constant value. Consequently, the eccentricity and the inclination of the inner binary with respect to the orbital plane of the outer binary oscillate as well. The time-scale of the oscillations exceeds by far the orbital periods of the inner and outer binaries.

In the limit of the secondary body being a test particle, the quantity

$$l_z = \sqrt{1 - e_2^2}\cos i_2 \tag{33}$$

is conserved. Here, the subscripts of the orbital elements refer to the secondary body with respect to the primary body. The Lidov–Kozai oscillations are present in the three-body system if the inclination i_0 of the secondary exceeds the critical value $i_{\rm crit}$, defined as

$$i_{\rm crit} = \arccos\left(\sqrt{\frac{3}{5}}\right), \tag{34}$$

which is approximately $i_{\rm crit} \approx 39.2^\circ$. The maximum achievable eccentricity $e_{\rm max}$ depends only on the initial inclination i_0 as

$$e_{\rm max} = \sqrt{1 - \frac{5}{3}\cos^2 i_0}. \tag{35}$$

We set up a hierarchical three-body system with masses of $M_1 = M_3 = 10^9\,{\rm M}_\odot$ and $M_2 = 10^3\,{\rm M}_\odot$ using the following orbital parameters. The outer binary is circular ($e_{\rm outer} = 0$) with a semimajor axis of $a_{\rm outer} = 20$ pc. The inner binary is initially almost circular

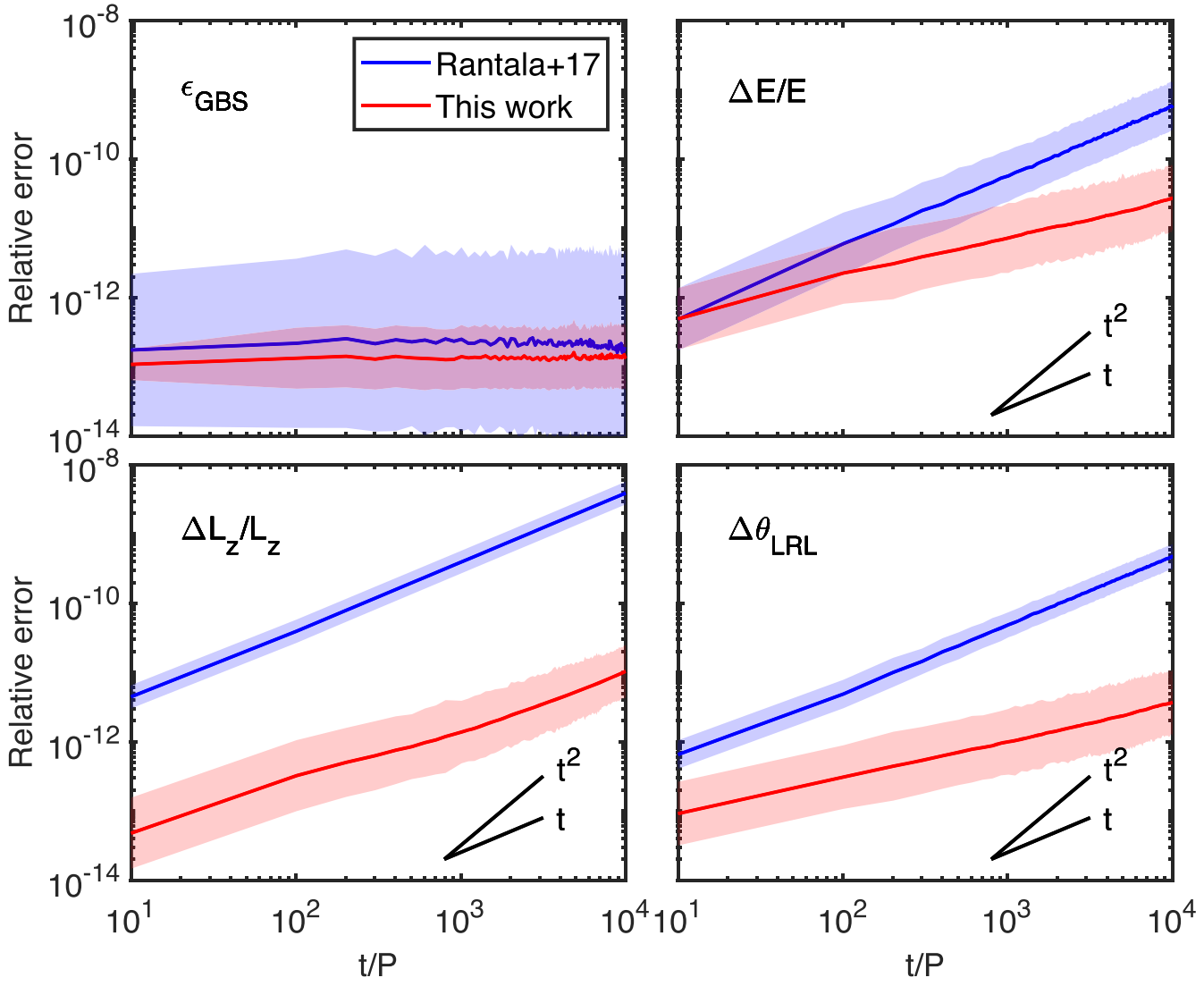

Figure 10. Results of the SMBH binary simulations with our AR-CHAIN implementation (blue) and MSTAR (red). Starting from the top left panel, the four panels show the maximum extrapolation error ϵ_{GBS} after an accepted step, the relative error in energy E and angular momentum L_z and the rotation angle of the Laplace–Runge–Lenz vector θ_{LRL}. The error regions depict a single standard deviation. The lower scatter in the maximum extrapolation error in MSTAR indicates that the code does not exceedingly increase the step size after a successful step, which would lead to divergence and step split during the next step. Our the new code clearly performs better than our AR-CHAIN implementation with the numerical errors being smaller throughout.

($e_{inner} = 10^{-3}$) and has a semimajor axis of $a_{inner} = 2$ pc. The orbital plane of the secondary is initially inclined $i_0 = 80^\circ$ with respect to the orbital plane of the outer binary, exceeding the critical inclination i_{crit} so the system exhibits Lidov–Kozai oscillations. The test-particle approximation predicts the maximum eccentricity of $e_{max} \approx 0.975$ for the system.

We simulate the evolution of the three-body system for 100 Myr using both the AR-CHAIN and MSTAR integrators. The integrator accuracy parameters are identical to the ones in the previous section. The oscillations of eccentricity and inclination of the secondary during the simulation are presented in Fig. 14. The system experiences roughly 10 strong oscillations in 100 Myr, reaching a maximum eccentricity of $e_{max} = 0.983$. The minimum inclination during the oscillations is very close to the critical value of $i_{crit} \approx 39.2^\circ$. The system evolves practically identically when run with the old AR-CHAIN and the new MSTAR integrator. The relative difference of the value of e_{max} with the two integrators is only of the order of 10^{-8}.

6 CODE SCALING AND TIMING: *N*-BODY TESTS

6.1 *N*-body initial conditions

We construct gravitationally bound clusters of equal-mass point particles in order to perform code timing tests. We use 20 different particle numbers, N_{part}, selected logarithmically between $N_{part} = 10^1$ and 10^4 particles with three different random seeds for each run, totalling 60 cluster initial conditions. The particle positions are drawn from the spherically symmetric Hernquist sphere (Hernquist 1990) with a density profile of

$$\rho(r) = \frac{M}{2\pi}\frac{a_H}{r(r+a_H)^3}, \tag{36}$$

where M is the total mass of the system and a_H its scale radius. We set $M = 10^7\,M_\odot$ and a_H in such a manner that the half-mass radius of the system equals $r_{1/2} = 10$ pc. The particle velocities

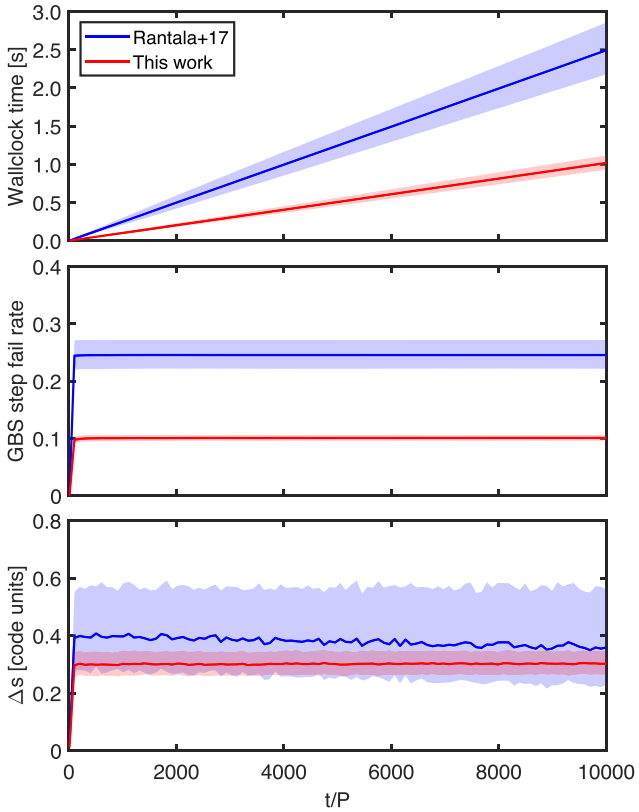

Figure 11. Additional results of the binary simulations with our previous AR-CHAIN implementation (blue) and the new code (red). The top panel presents the wall-clock times spent by the integrators with the new implementation being 2–3 times faster than the old one. The middle and bottom panels show the GBS step fail rate and the GBS step size H in fictitious time Δs. These panels confirm that the new AR-CHAIN implementation is faster in two-body tests due to its factor of ∼2.5 smaller GBS step fail rate.

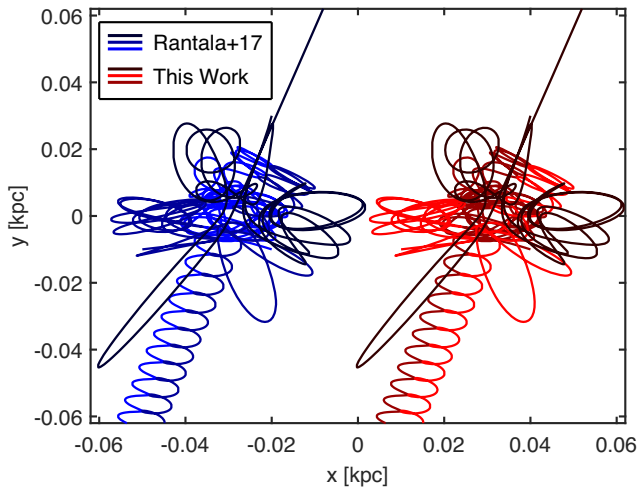

Figure 12. The general overview of the orbits of the three bodies in the Pythagorean three-body problem. Initially, the three bodies are gravitationally bound, but after a series of complicated interactions the least massive body (black line) is ejected while the other two bodies (red and blue) form a bound binary recoiling at the opposite direction. By eye, there are no noticeable differences in the results with the two integrators.

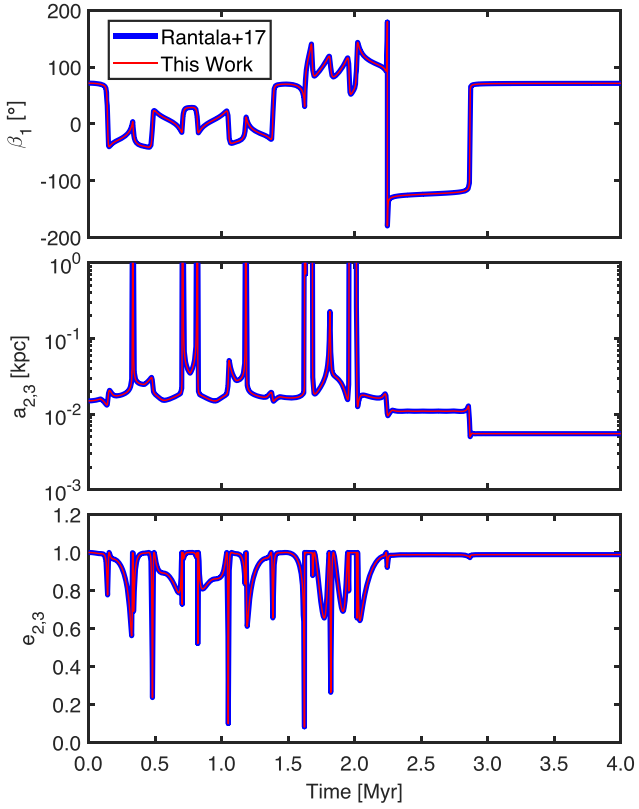

Figure 13. The three variables $\beta_1, a_{2,3}$, and $e_{2,3}$ parametrizing the outcome of the Pythagorean three-body problem as described in the main text. The results of the MSTAR and the AR-CHAIN integrator always agree within a relative factor of 10^{-4}.

are sampled from the Hernquist density–potential pair using the Eddington's formula technique (e.g. Binney & Tremaine 2008).

Even though we do not intentionally include primordial binaries in our cluster construction set-up, binaries may still form when sampling particle positions and velocities. A binary is considered hard if its binding energy exceeds the average kinetic energy of a particle in the cluster, i.e.

$$\frac{G\mu M}{2a} \gtrsim \frac{1}{2} m\sigma^2, \tag{37}$$

where m is the mass of a single particle and σ is the velocity dispersion of the cluster. While our MSTAR integrator can easily compute the dynamics of hard binaries, the possible existence of such binaries is problematic for the N-body timing tests. This is because an integrator using the LogH (or equivalent) time transformation can propagate physical time only for an amount Δt per one step, where Δt is of the order of the orbital period P of the hardest binary in the cluster, defined as

$$P = 2\pi \left(\frac{a^3}{GM}\right)^{1/2} \tag{38}$$

by the Kepler's third law. Consequently, the total running time of the simulation will scale as $\Delta t^{-1} \propto a^{-3/2}$. This is very inconvenient as the clusters with the same $N_{\rm part}$ but a different binary content may have a very large scatter in their simulation times up to several orders of magnitude. Thus, we exclude all clusters that contain even a single hard binary and generate a new cluster until we have 60 clusters in total. For the same reason, we do not include a single heavy point mass (SMBH) at the centre of the cluster as the orbital

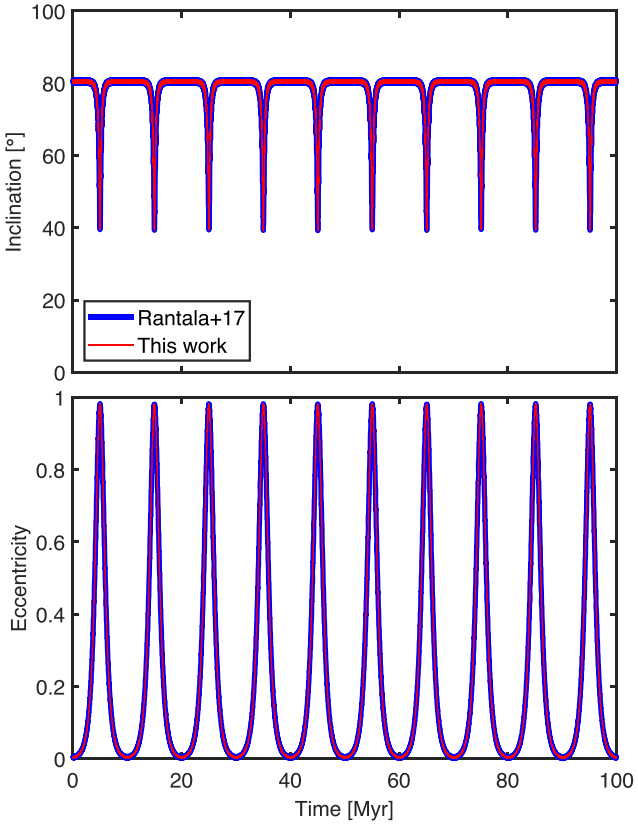

Figure 14. The Lidov–Kozai mechanism test with the AR-CHAIN (blue line) and the MSTAR integrator (red line). The hierarchical three-body system shows strong oscillations both in the inclination (top panel) and the eccentricity (bottom panel) of the secondary body. The results of the two integrators agree very well with each other and analytical estimates as described in the text.

period of the most bound light particle (star) would then determine the running time of the simulation.

6.2 Strong scaling tests

We perform a series of strong scaling tests to study the overall scaling behaviour of our MSTAR integrator. The results of the strong scaling test of the force calculation part of the code were presented in Fig. 9. As before in a strong scaling test, the problem size remains fixed while the number of CPUs is increased. In our six tests, we use six different logarithmically spaced particle numbers with $264 \leq N_{\rm part} \leq 10^4$ as in Section 4.3. The strong scaling tests consist of in total 270 short N-body simulations with initial conditions described in the previous section. In the simulations, each of the point-mass clusters is propagated for $T = 0.1$ Myr, which is close to the crossing times of the point-mass clusters. The GBS tolerance is set to $\eta_{\rm GBS} = 10^{-6}$ in these tests. We test CPU numbers up to $N_{\rm CPU} = 400$. The CPU number $N_{\rm CPU}$ is always divided between the force loop tasks ($N_{\rm force}$) and substep parallelization ($N_{\rm div}$) in a way that minimizes the simulation running time as explained in Section 4.3.

The results of the strong scaling tests are shown in Fig. 15. The maximum speed-up factors with $N_{\rm CPU} = 400$ range from $S_{\rm total} \approx 15$ with $N_{\rm part} = 264$ to $S_{\rm total} \approx 145$ when $N_{\rm part} = 10^4$. At this point, the scaling of the set-up with the lower particle number $N_{\rm part}$ is completely saturated while the set-up with $N_{\rm part} = 10^4$ would still benefit from additional computational resources. However, we do

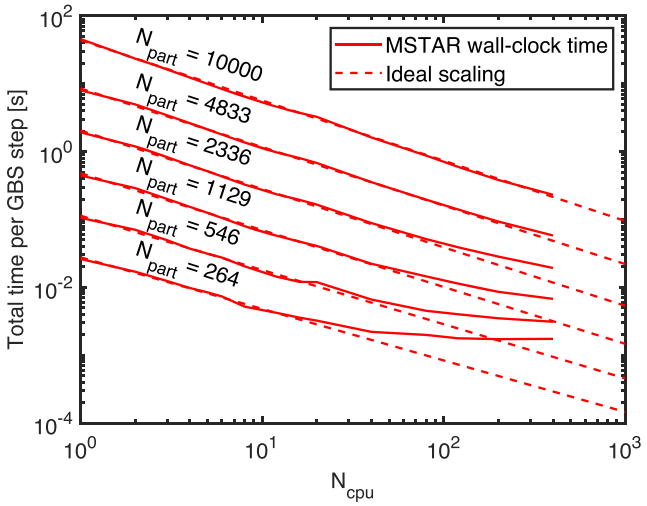

Figure 15. The results of the strong scaling test of our MSTAR integrator (solid red line). The ideal scaling behaviour is indicated by the dashed red line. The scaling behaviour of the integrator begins to deviate from the ideal scaling law around $N_{\rm CPU} \sim 0.1 \times N_{\rm part}$ and it is completely saturated around $N_{\rm CPU} \sim 0.2 \times N_{\rm part}$. The test set-ups with the two highest particle numbers tested retain the ideal scaling behaviour up to the largest CPU number used in these tests, $N_{\rm CPU} = 400$.

not pursue numerical experiments beyond $N_{\rm CPU} = 400$ in this study. We find that the integrator follows ideal scaling behaviour roughly up to the CPU number of $N_{\rm CPU} \sim 0.1 \times N_{\rm part}$ and a flat, saturated scaling occurs with $N_{\rm CPU} \gtrsim 0.2 \times N_{\rm part}$. The scaling of the entire code starts to deviate from the ideal scaling behaviour at a factor of a few smaller $N_{\rm CPU}$ than the scaling of only the force calculation part of the code. We attribute this difference to Amdahl's law (Amdahl 1967), which states that maximum speed-up of a parallelized code depends on the fraction of serial code or code that cannot be efficiently parallelized. The force calculation part of the code can be almost entirely parallelized except for the necessary MPI communication between the CPUs. The entire integrator contains additional parts that cannot be parallelized as efficiently as the force computation. The main functions containing these serial code sections or functions that are difficult to parallelize efficiently are the MST construction functions and the GBS extrapolation procedure.

6.3 Timing tests

We perform another set of N-body simulations to evaluate how much faster the parallelized version of the MSTAR integrator is than our AR-CHAIN implementation in Rantala et al. (2017). In the simulations, each of the 60 point-mass clusters is again propagated for $T = 0.1$ Myr with a GBS tolerance of $\eta_{\rm GBS} = 10^{-6}$. The other parameters remain as in the previous sections.

We test four different integrator configurations. The integrator details are collected in Table 2. Our old AR-CHAIN implementation is used both in a serial mode (R17-S-1) and a parallel mode (R17-P-24) with 24 CPUs as in Rantala et al. (2017). We test the MSTAR integrator in serial and parallel modes as well (R20-S-1 and R20-P-max). In the test set-up R20-P-max, we experimented with CPU numbers within $2 \leq N_{\rm CPU} \leq 400$ and chose the $N_{\rm CPU}$ that gave the smallest running time for each particle number. In general, adding more CPUs speeds up the computation until the scaling stalls around $N_{\rm CPU} \sim 0.2 \times N_{\rm part}$ as already illustrated in Fig. 15. This type of test is not performed with our old AR-CHAIN integrator as the scaling of

Table 2. The integrators and their serial/parallel configurations studied in the running time test. In the set-up R20-P-max, we selected the CPU number within $2 \leq N_{\rm CPU} \leq 400$ for each particle number, which yielded the fastest simulation times.

Label	Integrator	Mode	Resources
R17-S-1	AR-CHAIN	serial	1 CPU
R17-P-24	AR-CHAIN	parallel	24 CPU
R20-S-1	MSTAR	serial	1 CPU
R20-P-max	MSTAR	parallel	2-400 CPU

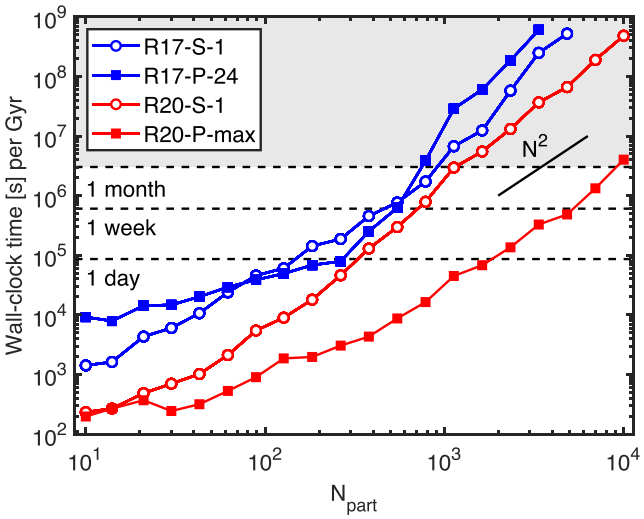

Figure 16. The timing test of our AR-CHAIN (blue) and MSTAR (red) integrators. The parallel runs are indicated with the filled squares while the serial runs are labelled with the open circles. Benchmarks of 1 d, 1 week, and 1 month are also included. The grey shaded area marks the region that we deem too time consuming for practical applications. We note that the new MSTAR serial code is even faster than the old parallel AR-CHAIN integrator and that the new parallel code is extremely fast compared to the other implementations, especially for large particle numbers.

the code becomes poor beyond a few tens of CPUs (Rantala et al. 2017).

The wall-clock times elapsed in the timing tests of the four integrator configurations are presented in Fig. 16 as a function of the particle numbers of the simulated clusters. The results are scaled into the units of wall-clock time in seconds per simulated Gyr. We restrict our tests to simulations that last less than 10^9 s of wall-clock time in the scaled units. Studying the results with the AR-CHAIN integrator first, we see that the parallel implementation is faster than the serial version in the particle number range $50 \lesssim N_{\rm part} \lesssim 700$. Our simulations in previous studies (Rantala et al. 2018, 2019) have used regularized particle numbers in the lower half of this range. With lower particle numbers, the communication time exceeds the speed-up from parallelized force loops. The slowdown of the parallelized old integrator at high particle numbers $N_{\rm part} \gtrsim 700$ is attributed to the fact that the code is only partially parallelized as serial functions still remain in the parallelized version.

Comparing the serial implementations of the MSTAR and the AR-CHAIN integrator (R17-S-1 and R20-S-1), we see that the new integrator is faster by a factor of $\sim$6 when $N_{\rm part} \lesssim 100$. The speed difference of the two serial codes reaches its minimum value of $\sim$2 around $N_{\rm part} = 10^3$. After this, the speed difference of the codes begins to increase again, reaching a value of $\sim$8 at $N_{\rm part} = 5 \times 10^3$. We note that the MSTAR serial implementation R20-S-1 is even faster than the AR-CHAIN parallel version R17-P-24 in every simulation we run for this study. Code run-time performance analysis tools reveal that the cache efficiency of our AR-CHAIN code is poor compared to the new MSTAR code. This fact explains the run-time speed difference of the two serial codes. All the four integrator configurations fairly closely follow the $\mathcal{O}(N_{\rm part}^2)$ scaling, being consistent with the theoretical scaling of the regularization algorithm of $\mathcal{O}(N_{\rm part}^{2.13-2.20})$.

Studying the results of the MSTAR integrator, we can see that the parallel set-up R20-P-max is always faster than the serial set-up R20-S-1, even with small particle numbers. In addition, the test runs with R20-P-max become increasingly faster than the serial set-up towards high particle numbers. Within $10^3 \lesssim N_{\rm part} \lesssim 10^4$, the speed-up factor is $\sim$55–145. Compared to the fastest old AR-CHAIN implementation, the new parallel MSTAR code is faster by a very large factor of $\sim$1100 in this range of particle numbers.

The adopted GBS accuracy parameter affects the wall-clock time elapsed in the simulations. We perform an additional brief series of timing tests using MSTAR with simulation parameters $N_{\rm part} = 1129$, $N_{\rm CPU} = 200$, and $10^{-12} \leq \eta_{\rm GBS} \leq 10^{-6}$. We find that in our tests the elapsed wall-clock time T scales well as a power law as

$$\frac{T}{T(\eta_{\rm GBS} = 10^{-6})} = \left(\frac{\eta_{\rm GBS}}{10^{-6}}\right)^{-\alpha} \tag{39}$$

in which the power-law index $\alpha \approx 0.05$ when $10^{-10} \lesssim \eta_{\rm GBS} \leq 10^{-6}$ and $\alpha \approx 1$ when $\eta_{\rm GBS} \lesssim 10^{-10}$. Due to the mild scaling of the wall-clock time T as a function of the GBS accuracy parameter $\eta_{\rm GBS}$, we conclude that the results in the section run with $\eta_{\rm GBS} = 10^{-6}$ apply in general for GBS tolerances $\eta_{\rm GBS} \gtrsim 10^{-10}$. However, we stress that the timing results of the codes depend on the N-body particle set-ups used, with more centrally concentrated stellar systems requiring in general more computational time.

Finally, we end this section by discussing how large simulations can be run with our new MSTAR code within a reasonable wall-clock time. We have marked the running times of 1 d, 1 week, and 1 month in Fig. 16. The grey area in the illustration marks the running times beyond 1 month per Gyr, which we consider unacceptably time consuming. The parallel MSTAR code can perform simulations with of the order of 10 times more N-body particles with similar wall-clock times as our old integrator implementations. Simulations with 4000–7000 particles are expected to last a few weeks with the parallel MSTAR code with $N_{\rm CPU} = 400$. Running a simulation with $N_{\rm part} = 10^4$ in a similar wall-clock time would require $N_{\rm CPU} \approx 2000$ cores.

7 CONCLUSIONS

We have developed and implemented the MSTAR integrator, a new fast AR integrator. While the time transformation scheme of the regularized integrator remains the same as in the earlier AR-CHAIN integrator, the coordinate system and the GBS extrapolation method are significantly improved. A brief summary of the main ingredients of our integrator code and a selection of related integrators from the literature is collected in Table 3.

In our new MSTAR implementation, the chained coordinate system of AR-CHAIN is replaced by an MST coordinate system, which can be viewed as a branching chain structure. Due to its ability to branch, the MST avoids the possible pathological chain configurations in which spatially close particles can be found in very different parts of the chain. We find that the numerical error originating from building and deconstructing the coordinate

Table 3. A brief summary of the main properties of the MSTAR integrator presented in this work alongside related integration methods from the literature.

	Ito & Fukushima (1997)	Mikkola & Merritt (2008)	Rantala et al. (2017)	This work
Regularization	×	✓	✓	✓
Extrapolation method	✓	✓	✓	✓
Chained coordinates	×	✓	✓	✓
MST coordinates	×	×	×	✓
Serial code	✓	✓	✓	✓
Parallel force loops	×	×	✓	✓
Parallel GBS subdivisions	✓	×	×	✓

structures is approximately smaller by a factor of ∼10 for the MST compared to the chain. The reason for this is that the MST is a much shallower data structure than the chain as the average number of inter-particle vectors to reach the root vertex of the system is smaller. Thus, we recommend using the MST coordinate system instead of the chained coordinates even though the code implementation becomes somewhat more complicated.

Our MSTAR integrator includes a simplified GBS extrapolation method with two layers of MPI parallelization. First, the force loop computation is parallelized with $N_{\rm force}$ CPUs with one MPI task each. The second layer is included in order to compute the $k_{\rm fix}$ substep divisions in parallel using $N_{\rm div}$ CPU groups. We also provide a recipe for estimating how to divide the computational resources to the different parallelization layers and estimates for the maximum reasonable number of CPUs for parallelization before the code scaling stalls.

We validate the numerical accuracy of our MSTAR integrator in a series of demanding two- and three-body test simulations. The simulation set-ups include an eccentric Keplerian binary, the classic Pythagorean three-body problem, and Lidov–Kozai oscillations in a hierarchical three-body system. Overall, the particle orbits in the test runs are practically identical with both the MSTAR and AR-CHAIN integrators. In fact, MSTAR conserves energy, angular momentum, and the direction of the Laplace–Runge–Lenz vector somewhat better than our previous regularized integrator AR-CHAIN.

We test the speed and scaling behaviour of the MSTAR integrator in a series of N-body stellar cluster simulations with up to $N_{\rm part} = 10^4$ particles. We find that the new integrator is always faster than the AR-CHAIN. With full parallelization, we can efficiently use ∼10 times more CPUs with adequate code scaling behaviour compared to our integrator implementation with only force loop parallelization. The speed-up gained by the new fully parallelized integrator is substantial. The parallel MSTAR code is up to a factor of ∼145 faster than the serial MSTAR code and up to a factor of ∼1100 faster than the AR-CHAIN code when the simulation particle number is in the range $10^3 \lesssim N_{\rm part} \lesssim 10^4$.

The MSTAR integrator will be important when pressing towards the ultimate goal of running collisionless simulations containing regularized regions with collisional stars and SMBHs with up to $N_{\rm part} \sim 5 \times 10^8$–$10^9$ simulation particles in individual galaxies. We estimate that the MSTAR integrator is able to run a Gyr-long simulation with $N_{\rm part} = 10^4$ in approximately two weeks of wall-clock time using $N_{\rm CPU} \approx 2000$ CPUs. In our previous studies with the KETJU code (Rantala et al. 2017), which couples the GADGET-3 tree code to the AR-CHAIN integrator, the total particle number in galaxies was limited to $N_{\rm part} \lesssim 10^7$ particles (Rantala et al. 2018, 2019). This is due to the fact that our AR-CHAIN integrator could efficiently handle only up to 200–300 particles in the regularized regions. Based on these numbers, we estimate that MSTAR can treat ∼50 times more regularized particles than our AR-CHAIN implementation in a reasonable wall-clock time. Thus, galaxy simulations with accurate SMBH dynamics using MSTAR in KETJU instead of AR-CHAIN containing $5 \times 10^8 \lesssim N_{\rm part} \lesssim 10^9$ simulation particles seem achievable in the immediate future. These particle numbers yield stellar mass resolutions down to $m_\star \approx 100\,{\rm M}_\odot$ even for simulations of massive early-type galaxies.

Finally, the improved numerical scaling and performance will be crucial when simulating the dynamics of SMBHs in gas-rich galaxies, which are expected to have steep central stellar concentrations due to elevated levels of star formation in their nuclei. This is, in particular, important as the upcoming LISA gravitational wave observatory will be the most sensitive for SMBHs with masses in the range of $M_{\rm BH} \sim 10^6$–$10^7\,{\rm M}_\odot$ (Amaro-Seoane et al. 2007), which are expected to reside at the centres of gas-rich late-type galaxies.

ACKNOWLEDGEMENTS

We would like to thank Seppo Mikkola, the referee of the paper. The numerical simulations were performed on facilities hosted by the CSC – IT Center for Science, Finland and the MPCDF, Germany. AR, PP, MM, and PHJ acknowledge the support by the European Research Council via ERC Consolidator Grant KETJU (no. 818930). TN acknowledges support from the Deutsche Forschungsgemeinschaft (DFG; German Research Foundation) under Germany's Excellence Strategy – EXC-2094 – 390783311 from the DFG Cluster of Excellence 'ORIGINS'.

REFERENCES

Aarseth S. J., 1999, PASP, 111, 1333
Aarseth S. J., 2003, Gravitational N-Body Simulations. Cambridge Univ. Press, Cambridge
Aarseth S. J., 2012, MNRAS, 422, 841
Aarseth S. J., Anosova J. P., Orlov V. V., Szebehely V. G., 1994, Celest. Mech. Dyn. Astron., 58, 1
Alexander T., 2017, ARA&A, 55, 17
Amaro-Seoane P., Gair J. R., Freitag M., Miller M. C., Mandel I., Cutler C. J., Babak S., 2007, Class. Quantum Gravity, 24, R113
Amdahl G. M., 1967, Proc. April 18-20, 1967, Spring Joint Comput. Conf. AFIPS '67 (Spring). ACM, New York, NY, USA, p. 483
Beckmann R. S., Slyz A., Devriendt J., 2018, MNRAS, 478, 995
Begelman M. C., Blandford R. D., Rees M. J., 1980, Nature, 287, 307
Berczik P., Merritt D., Spurzem R., Bischof H.-P., 2006, ApJ, 642, L21
Binney J., Tremaine S., 2008, Galactic Dynamics, 2nd edn. Princeton Univ. Press, Princeton, NJ
Borůvka O., 1926, Pr. Moravské přírodovědecké společnosti, sv. III, 7, 37

Boylan-Kolchin M., Ma C.-P., 2004, MNRAS, 349, 1117
Bulirsch R., Stoer J., 1966, Numer. Math., 8, 1
Burrau C., 1913, Astron. Nachr., 195, 113
Deuflhard P., 1983, Numer. Math., 41, 399
Dubois Y., Gavazzi R., Peirani S., Silk J., 2013, MNRAS, 433, 3297
Duffell P. C., D'Orazio D., Derdzinski A., Haiman Z., MacFadyen A., Rosen A. L., Zrake J., 2019, preprint (arXiv:1911.05506)
Ferrarese L., Ford H., 2005, Space Sci. Rev., 116, 523
Gragg W. B., 1965, SIAM J. Numer. Anal., 2, 384
Gualandris A., Read J. I., Dehnen W., Bortolas E., 2017, MNRAS, 464, 2301
Hairer E., Lubich C., Wanner G., 2006, Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations, Vol. 31. Springer-Verlag, Berlin
Hairer E., Nørsett S., Wanner G., 2008, Solving Ordinary Differential Equations I: Nonstiff Problems. Springer Series in Computational Mathematics. Springer, Berlin
Harary F., 1969, Graph Theory. Addison-Wesley, Reading
Harfst S., Gualandris A., Merritt D., Mikkola S., 2008, MNRAS, 389, 2
Hayward C. C., Torrey P., Springel V., Hernquist L., Vogelsberger M., 2014, MNRAS, 442, 1992
Hellström C., Mikkola S., 2010, Celest. Mech. Dyn. Astron., 106, 143
Hernquist L., 1990, ApJ, 356, 359
Holley-Bockelmann K., Richstone D., 1999, ApJ, 517, 92
Ito T., Fukushima T., 1997, AJ, 114, 1260
Jernigan J. G., Porter D. H., 1989, ApJS, 71, 871
Johansson P. H., Naab T., Burkert A., 2009, ApJ, 690, 802
Kahan W., 1965, Commun. ACM, 8, 40
Khan F. M., Just A., Merritt D., 2011, ApJ, 732, 89
Khan F. M., Fiacconi D., Mayer L., Berczik P., Just A., 2016, ApJ, 828, 73
Kim J.-h., Wise J. H., Alvarez M. A., Abel T., 2011, ApJ, 738, 54
Korch M., Rauber T., Scholtes C., 2011, Concurrency Comput.: Pract. Exp., 23, 1789
Kormendy J., Ho L. C., 2013, ARA&A, 51, 511
Kormendy J., Richstone D., 1995, ARA&A, 33, 581
Kozai Y., 1962, AJ, 67, 591
Kruskal J. B., 1956, Proc. Am. Math. Soc., 7, 48
Kustaanheimo P., Stiefel E., 1965, J. Reine Angew. Math., 218, 204
Lauer T. R. et al., 2007, ApJ, 664, 226
Lidov M. L., 1962, Planet. Space Sci., 9, 719
Mannerkoski M., Johansson P. H., Pihajoki P., Rantala A., Naab T., 2019, ApJ, 887, 35
Mayer L., Kazantzidis S., Madau P., Colpi M., Quinn T., Wadsley J., 2007, Science, 316, 1874
Merritt D., 2006, ApJ, 648, 976
Merritt D., 2013, Dynamics and Evolution of Galactic Nuclei. Princeton Univ. Press, Princeton, NJ
Mikkola S., 1997, Celest. Mech. Dyn. Astron., 67, 145
Mikkola S., Aarseth S. J., 1989, Celest. Mech. Dyn. Astron., 47, 375
Mikkola S., Aarseth S. J., 1993, Celest. Mech. Dyn. Astron., 57, 439
Mikkola S., Aarseth S., 2002, Celest. Mech. Dyn. Astron., 84, 343
Mikkola S., Merritt D., 2006, MNRAS, 372, 219
Mikkola S., Merritt D., 2008, AJ, 135, 2398
Mikkola S., Tanikawa K., 1999a, Celest. Mech. Dyn. Astron., 74, 287
Mikkola S., Tanikawa K., 1999b, MNRAS, 310, 745
Mikkola S., Palmer P., Hashida Y., 2002, Celest. Mech. Dyn. Astron., 82, 391
Milosavljević M., Merritt D., 2001, ApJ, 563, 34
Milosavljević M., Merritt D., 2003, ApJ, 596, 860
Misgeld I., Hilker M., 2011, MNRAS, 414, 3699
Moody M. S. L., Shi J.-M., Stone J. M., 2019, ApJ, 875, 66
Naoz S., 2016, ARA&A, 54, 441
Neumaier A., 1974, Z. Angew. Math. Mech., 54, 39
Peters P. C., Mathews J., 1963, Phys. Rev., 131, 435
Pihajoki P., 2015, Celest. Mech. Dyn. Astron., 121, 211
Press W., Teukolsky S., Vetterling W., Flannery B., 2007, Numerical Recipes, 3rd edn: The Art of Scientific Computing. Cambridge Univ. Press, Cambridge
Preto M., Tremaine S., 1999, AJ, 118, 2532
Prim R. C., 1957, Bell Syst. Tech. J., 36, 1389
Quinlan G. D., 1996, New Astron., 1, 35
Rantala A., Pihajoki P., Johansson P. H., Naab T., Lahén N., Sawala T., 2017, ApJ, 840, 53
Rantala A., Johansson P. H., Naab T., Thomas J., Frigo M., 2018, ApJ, 864, 113
Rantala A., Johansson P. H., Naab T., Thomas J., Frigo M., 2019, ApJ, 872, L17
Rauber T., Rünger G., 1997, Concurrency, Pract. Exp., 9, 181
Ryu T., Perna R., Haiman Z., Ostriker J. P., Stone N. C., 2018, MNRAS, 473, 3410
Sanders D. B., Mirabel I. F., 1996, ARA&A, 34, 749
Springel V., Di Matteo T., Hernquist L., 2005, MNRAS, 361, 776
Stumpff K., Weiss E. H., 1968, J. Astronaut. Sci., 15, 257
Szebehely V., Peters C. F., 1967, AJ, 72, 876
Tang Y., MacFadyen A., Haiman Z., 2017, MNRAS, 469, 4258
Valtonen M., Karttunen H., 2006, The Three-Body Problem. Cambridge Univ. Press, Cambridge, UK
Vasiliev E., Antonini F., Merritt D., 2015, ApJ, 810, 49
Wang X., Wang X., Mitchell Wilkes D., 2009, IEEE Trans. Knowl. Data Eng., 21, 945
White S. D. M., Rees M. J., 1978, MNRAS, 183, 341
Zhong C., Malinen M., Miao D., Fränti P., 2013, in Wilson R., Hancock E., Bors A., Smith W., eds, Computer Analysis of Images and Patterns. Springer, Berlin, p. 262

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