Accelerating NBODY6 with a GPU-Enabled Particle-Particle Particle-Tree Scheme

Anthony D. Arnold,1,* Holger Baumgardt1 and Long Wang2,3
1School of Mathematics and Physics, The University of Queensland, St Lucia QLD 4072, Australia
2Department of Astronomy, School of Science, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan
3RIKEN Center for Computational Science, Minatojima-minami-mach, Chuo-ku, Kobe, Hyogo 650-0047, Japan

Received XXX; in original form ZZZ

Abstract

We describe a modified version of the nbody6 code for simulating star clusters which greatly improves computational efficiency while sacrificing little in the way of accuracy. The distant force calculator is replaced by a GPU-enabled Barnes-Hut code, and integration is done with a standard leapfrog scheme. Short-range forces continue to use the CPU-based fourth-order Hermite predictor-corrector scheme of nbody6. Our code outperforms nbody6 for systems with more than $3 \times 10^3$ particles and runs more than a factor 2 faster for systems of $10^5$ particles with similar energy conservation. Our code should be useful for simulating realistic dense stellar clusters, such as globular clusters or galactic nuclei.

Key words: methods: numerical – software: simulations – gravitation – globular clusters: general

1 INTRODUCTION

The nbody6 code (Aarseth 1999) is the state of the art code for performing direct N-body simulations to study the evolution of dense stellar systems. It is the yardstick against which other N-body codes are measured in terms of computational efficiency and accuracy (Iwasawa et al. 2015; Rodriguez et al. 2016). Moreover, nbody6 remains flexible in its support for arbitrary initial conditions compared to Monte-Carlo methods (Giersz et al. 2013). The code splits the gravitational force acting on a particle into near and distant forces, allowing the fast-changing near forces to be treated separately from the slow-changing distant ones (Ahmad & Cohen 1973; Makino & Aarseth 1992). Recent improvements to nbody6 have included routines for performing distant force calculations on general-purpose graphics processing units (GPUs) (Nitadori & Aarseth 2012), which greatly improves the overall run time. Since the force calculation algorithm used in nbody6 has a calculation cost of $O(N^2)$, simulations of large clusters, exceeding $10^6$ stars, are still very expensive (Iwasawa et al. 2015; Bédorf et al. 2012; Aarseth 2003a; Wang et al. 2015).

An early attempt at an $O(N \log N)$ integration scheme for N-body simulations using a tree code was described by McMillan & Aarseth (1993). More recently, Oshino et al. (2011) described the Particle-Particle Particle-Tree (P3T) scheme which uses a tree code for distant force calculations and a standard leapfrog scheme for integrating these distant forces, while retaining the Hermite scheme for neighbour forces. A working implementation of this scheme with a GPU-enabled tree code was shown in Iwasawa et al. (2015). Over short simulation times, the P3T scheme performs faster than nbody6, however as the system evolves past core collapse, the P3T scheme becomes slow due to the short time steps required by the Hermite integrator. nbody6 is able to cope with these short time steps due to the KS (Kustaanheimo et al. 1965) and chain regularisation code for close encounters (Mikkola & Aarseth 1989; Aarseth 2003b). Additionally, nbody6 includes routines for modelling stellar evolution and post-Newtonian force corrections, which allow for more realistic simulations.

In this paper, we present a version of nbody6, called nbody6+p3t, which incorporates the P3T scheme while retaining nbody6’s routines for KS and chain regularisation. Section 2 describes the construction of the improved nbody6 code. Section 3 presents the results of different runs compared to the original nbody6. Finally in section 4 we present conclusions and discuss future applications of the new code.

2 IMPLEMENTATION

2.1 Formulation

In this section we describe the modifications made to nbody6 in order to incorporate the P3T scheme. We adopt the definition of P3T from Iwasawa et al. (2015) with the following deviations. The Plummer softening is omitted from the scheme as the singularity due to the $1/r$ potential is handled by special treatment of close encounters through KS regularisation (Mikkola & Aarseth 1998).

A radius known as the neighbour sphere around each particle separates the force acting on the particle into irregular and regular forces. The irregular forces are those fast-changing forces from nearby particles inside the neighbour sphere. The regular forces are from the distant particles outside of the neighbour sphere, which change more slowly. In nbody6+p3t, the irregular and regular accelerations of particle $i$, $F_{I,i}$ and $F_{R,i}$ respectively, due to particle $j$ are calculated...
as follows:

\[ F_{i,j} = \sum_{j \neq i} m_j \frac{r_{ij}}{|r_{ij}|^3} K_{ij}, \]  
\[ F_{i,j} = \sum_{j \neq i} m_j \frac{\dot{r}_{ij}}{|r_{ij}|^3} K_{ij} - 3 \frac{(r_{ij} \cdot \dot{r}_{ij})r_{ij}}{|r_{ij}|^5} + m_j \frac{r_{ij}}{|r_{ij}|^3} K_{ij}', \]  
\[ F_{R,i} = \sum_{j \neq i} m_j \frac{\dot{r}_{ij}}{|r_{ij}|^3} r_{ij} - F_{i,j}, \]

where \( m_i, x_i, \dot{x}_i \) are the mass, position, and velocity of particle \( i \), respectively and \( r_{ij} \) is the Euclidean distance between particles \( i \) and \( j \). Here, \( K \) and \( K' \) are cutoff functions to ensure a smooth transition of forces between \( F_j \) and \( F_R \) as particles move in and out of neighbour spheres. We adopt the formulae from Iwasawa et al. (2015); Duncan et al. (1998), we have:

\[ K_{ij} = 1 - \begin{cases} 
0 & \text{if } x < 0, \\
-20x^7 + 70x^6 - 84x^5 + 35x^4 & \text{if } 0 \leq x < 1, \\
1 & \text{if } 1 \leq x, 
\end{cases} \]
\[ K_{ij}' = \begin{cases} 
(-140x^6 + 420x^5 - 420x^4 + 140x^3) \frac{(r_{ij} \cdot \dot{r}_{ij})}{|r_{ij}|(|r_{cut} - r_{in})} & \text{if } 0 \leq x < 1, \\
0 & \text{otherwise}, 
\end{cases} \]

where

\[ x = \frac{y - y'}{1 - y'}, \]
\[ y = \frac{|r_{ij}|}{r_{cut}}, \]
\[ y' = \frac{r_{in}}{r_{cut}}, \]

and \( r_{cut} \) and \( r_{in} \) denote the outer and inner cutoff radii for neighbour forces, respectively. If \( |r_{ij}| < r_{cut} \) then the force from particle \( j \) contributes entirely to \( F_{i,j} \), and likewise if \( |r_{ij}| > r_{cut} \) the force counts wholly as part of \( F_{j,i} \). As in Iwasawa et al. (2015), we use \( y = 0.1 \) for all calculations.

In \texttt{nbody6+mr3}, we used the standard criterion from Makino & Aarseth (1992), which determines the individual time step \( \Delta t_i \) for each particle in the Hermite scheme.

\[ \Delta t_i = \min \left( \frac{a_1^{(0)} + a_2^{(1)} + a_3^{(2)}}{\eta a_1^{(1)} + a_2^{(2)} + a_3^{(3)}}, \Delta t_{\text{max}} \right). \]

Here, \( a_1^{(n)} \) is the \( n \)th time derivative of the acceleration of particle \( i \) due to \( F_{j,i} \), with the exception of \( a_1^{(0)} \) which is due to \( F_{i,j} + F_{R,i} \) (Makino & Aarseth 1992). We used the suggestion from Aarseth (1985) and set the accuracy parameter \( \eta = 0.02 \) for all calculations.

The standard P3T implementation includes an initial time step criterion to account for the missing higher order acceleration derivatives. \texttt{nbody6} instead has routines to calculate these derivatives and as such the fourth order standard criterion is used to calculate the initial time step in \texttt{nbody6+mr3}.

\begin{figure}
\centering
\includegraphics[width=0.7\textwidth]{figure1.pdf}
\caption{The integration algorithm used by \texttt{nbody6+mr3}.}
\end{figure}

\subsection{2.2 Modified Integration}

In \texttt{nbody6+mr3} we replaced the regular force Hermite scheme from \texttt{nbody6} with a standard leapfrog integrator. The velocity kick from the regular force is given by

\[ \ddot{x}_i = \dot{x}_i + \frac{\Delta t_{\text{reg}}}{2} F_{R,i}. \]  

The regular time step \( \Delta t_{\text{reg}} \) is a constant shared by all particles and is defined as a multiple of the maximum irregular Hermite step \( \Delta t_{\text{max}} \). In all of our simulations we use \( \Delta t_{\text{reg}} = 4\Delta t_{\text{max}} \). The integration algorithm used in \texttt{nbody6+mr3}, illustrated in Fig. 1, is as follows:

(i) Calculate the regular acceleration \( F_R \) of every particle.
(ii) Apply the velocity kick from equation 12 to each particle.
(iii) Integrate \( F_I \) using the Hermite integrator to time \( t = t + \Delta t_{\text{reg}} \).
(iv) Calculate \( F_R \) again.
(v) Apply another velocity kick.
(vi) Go to step (ii).

The routine which predicts the position and velocity of a particle at time \( t \) uses the total force in \texttt{nbody6} in the form

\[ x_{\text{pred},i} = x_i + \Delta t_i \dot{x}_i + \frac{\Delta t_i^2 (F_{I,i} + F_{R,i})}{2} + \frac{\Delta t_i^3 (\dot{F}_{I,i} + \dot{F}_{R,i})}{6}, \]
\[ \dot{x}_{\text{pred},i} = \dot{x}_i + \Delta t_i (F_{I,i} + F_{R,i}) + \frac{\Delta t_i^2 (\ddot{F}_{I,i} + \ddot{F}_{R,i})}{2}, \]
\[ \Delta t_i = t - t_i, \]

however due to the velocity kick from the leapfrog routine in \texttt{nbody6+mr3}, the regular force would be counted twice. The new routine eschews the regular force and it’s derivative, becoming

\[ x_{\text{pred},i} = x_i + \Delta t_i \dot{x}_i + \frac{\Delta t_i^2 \dot{F}_{I,i}}{2} + \frac{\Delta t_i^3 \ddot{F}_{I,i}}{6}, \]
\[ \dot{x}_{\text{pred},i} = \dot{x}_i + \Delta t_i \dot{F}_{I,i} + \frac{\Delta t_i^2 \ddot{F}_{I,i}}{2}. \]
2.3 Neighbour Radius

\texttt{nbody6+p3t} uses a constant neighbour radius \( R_{\text{c}} \) for all particles and introduces a new constant parameter, \( R_{\text{buff}} \) such that \( \text{cut} = R_{\text{c}} + R_{\text{buff}} \). The fixed neighbour radius leads to a much smaller average neighbour number, usually \(< 1\), with most particles being isolated. In contrast to \texttt{nbody6} which adjusts size of the neighbour sphere to avoid isolated particles, routines in \texttt{nbody6+p3t} must tolerate empty neighbour lists.

2.4 Force Calculation

Historically, the dominant component with regards to computational time in N-body gravitational simulations has been the calculation of the \( N^2 \) gravitational interactions (Iwasawa et al. 2015; Bédorf et al. 2012; Aarseth 2003a; Mikkola & Aarseth 1998; Barnes & Hut 1986). A Barnes-Hut tree (Barnes & Hut 1986) is a data structure and algorithm for approximating the force acting on each particle in \( O(n \log n) \) time; the increase in efficiency is a trade-off with accuracy. This trade-off is managed through the opening angle parameter \( \theta \). A value of \( \theta = 0 \) will result in no approximations, reducing the tree code algorithm to the equivalent of a direct-N code. For all runs in this paper we choose \( \theta = 0.4 \).

The regular force calculator in \texttt{nbody6+p3t} is a GPU-enabled tree code provided by the \texttt{bonsai} library (Bédorf et al. 2012). Modifications were made to \texttt{bonsai} to populate the neighbour list and return the results in a format compatible with \texttt{nbody6}.

Irregular forces, the gravitational force acting on each particle due to its neighbours, are calculated with a family of routines which were modified in \texttt{nbody6+p3t} to include the smoothing function from (6) and (7) and the alternative prediction formulation from (16) and (17).

2.5 Regularisation Parameters

\texttt{nbody6} defines two values, \( \Delta T_{\text{min}} \) and \( R_{\text{min}} \), which control when two particles are candidates for KS regularisation. These parameters describe the minimum distance and irregular time step required for two particles to form a regularised pair. Put simply, if for any single particle \( i \), \( \Delta t_i < \Delta T_{\text{min}} \) and for its nearest neighbour \( j \), \( |x_i - x_j| < R_{\text{min}} \), then the pair \( i,j \) may be removed from the simulation and replaced with a single centre-of-mass particle to represent the binary. Choosing the wrong values for these parameters leads to large energy errors. The values of \( R_{\text{min}} \) and \( \Delta T_{\text{min}} \) are calculated in the \texttt{adjust} routine and are defined as follows.

\[
R_{\text{min}} = \frac{4r_{\text{p}}}{N^{1/3} \rho_{\text{core}}}, \tag{18}
\]

\[
\Delta T_{\text{min}} = 0.01 \sqrt{\frac{\eta/0.02}{\sqrt{\frac{\Delta t}{\min}}}}, \tag{19}
\]

where \( r_{\text{p}} \) is the half-mass radius and \( \rho_{\text{core}} \) is the central density of the system. In \texttt{nbody6+p3t}, we use the same definitions for \( \Delta T_{\text{min}} \) and \( R_{\text{min}} \) as in \texttt{nbody6}.

The core quantities, \( R_{\text{core}}, N_{\text{core}}, \) and \( \rho_{\text{core}} \) posed a challenge. The method, due to Casertano & Hut (1985), used to find \( R_{\text{core}} \) and the core density \( \rho_{\text{core}} \) relies on knowing the 6 nearest neighbours to each particle. \texttt{nbody6} uses the neighbour lists of the particles for this purpose and while \texttt{nbody6} ensures that most neighbour lists have sufficient numbers by varying \( r_{\text{cut}} \) for individual particles, \texttt{nbody6+p3t} uses a global fixed \( r_{\text{cut}} \).

The core quantities are used to determine the regularisation parameters, \( R_{\text{min}} \) and \( \Delta T_{\text{min}} \). These parameters describe the minimum distance and irregular time step required for two particles to form a regularised pair. If these parameters are too high, \texttt{nbody6+p3t} chooses to regularise pairs in sub-optimal situations, leading to frequent and unnecessary regularisations. Conversely, when the parameters are too low, particles will get too close to each other before they are regularised, leading to small irregular integration steps. We show a solution to this problem in subsection 3.2.1.

2.6 Other Modifications

Assumptions are present throughout \texttt{nbody6}, particularly with regards to the variability of \( R_{\text{c}} \) and the avoidance of isolated particles. Also prevalent in the code are numerous code loops to perform local Hermite scheme predictions and force calculations, outside of the usual routines, which needed to be modified to follow the formulations from (16) and (17). To explain all of the changes required in \texttt{nbody6+p3t} would be prohibitively verbose. Tables listing the modifications made to \texttt{nbody6} are made available in appendix B.

3 RESULTS

3.1 Accuracy and Performance

We performed a number of test runs in order to compare the accuracy and performance of \texttt{nbody6+p3t} against \texttt{nbody6}. In this section, we describe the configuration of each program and the results of the test runs. For initial conditions, we adopted a Plummer model (Plummer 1911) of equal-mass particles for each run. We used N-Body units (Heggie 1986) and set the total mass \( M = 1 \), the gravitational constant \( G = 1 \), and the total energy \( E = -1/4 \). Softening is not used as KS regularisation will avoid the singularity of the gravitational potential.

All runs were performed on a single compute node, utilising 28 Intel Xeon Gold 6132 CPUs (2 sockets, 14 cores per socket), 32GB of memory, and 1 NVIDIA V100 GPU. For each run, we evolved the system over 10 N-body time units, with the exception of \( N = 2048k \) which was evolved for only 1 N-body time unit to avoid lengthy run times. In our results we show the average wall clock time per N-body time unit.

3.1.1 Parameters

The number of particles, \( N \), in each run ranged from \( N = 32k \) to \( 2048k \) (where \( k = 1024 \)), doubling in size for each successive run. Unless explicitly stated, run time parameters were chosen to follow the standard \texttt{nbody6} scheme. The parameters \( r_{\text{buff}}, r_{\text{cut}}, \Delta t_{\text{reg}} \) were determined according to a modified version of the optimal set of accuracy parameters set out in Iwasawa et al. (2015) and summarised below.

\[
r_{\text{buff}} = 3 \sigma \tau, \tag{20}
\]

\[
r_{\text{cut}} = 4 \tau, \tag{21}
\]

\[
\tau = \frac{1}{128} \left( \frac{N}{2^{14}} \right)^{-1/3}, \tag{22}
\]

\[
\Delta t_{\text{reg}} = \text{lowest power of } \frac{1}{2} \text{ s.t. } \Delta t_{\text{reg}} \leq \tau, \tag{23}
\]

\[
\Delta t_{\text{max}} = \frac{\Delta t_{\text{reg}}}{4}, \tag{24}
\]

\[
\sigma = \frac{1}{\sqrt{2}}. \tag{25}
\]
Here, $\sigma$ is the global three-dimensional velocity dispersion and $\tau$ is the calculated regular time step used to get $r_{\text{buf}}$ and $r_{\text{cut}}$. $\Delta_{\text{max}}$ identifies the maximum irregular time step for an individual particle in $\text{body6}+p3r$; for $\text{body6}$ the standard value of 1/8 was used.

For all runs in this paper, $\sigma$ is considered constant throughout the run. In future runs, requiring longer integration times, a periodic recalculation would be necessary.

### 3.1.2 Results

When $N \leq 256k$, $\text{body6}+p3r$ performs similarly to $\text{body6}$ in terms of run time. For larger runs, $\text{body6}+p3r$ is measurably faster, completing the run in less than half the time for $N = 1024k$ and more than 5 times faster for $N = 2048k$. Fig. 2 shows the evolution of total execution time as the system size increases for both codes.

We identified four functional components common to both codes: force calculation and integration, each further separated into regular and irregular parts. Irregular and regular force calculation execution time are denoted $T_{F_I}$ and $T_{F_R}$ respectively. Similarly, irregular and regular integration are denoted $T_{I}$ and $T_{R}$ respectively. Here, the irregular force calculation and integration corresponds to the direct-N force calculator and Hermite integrator of both $\text{body6}$ and $\text{body6}+p3r$. Furthermore, the regular force calculation corresponds to the GPU-enabled direct-N force calculator in $\text{body6}$ and to the GPU-enabled tree code in $\text{body6}+p3r$. Finally, the regular integration component corresponds to the Hermite integrator in $\text{body6}$ and to the leap frog integrator in $\text{body6}+p3r$. We measured the execution time of each component to show how the choice of component algorithm affects the total execution time.

Fig. 3 shows $T_{F_I}$ and $T_{F_R}$ for different system sizes. $\text{body6}+p3r$ consistently spends less time calculating either force when $N > 128k$. For smaller $N$ the difference is negligible; only a few seconds per N-body time.

The advantage of the tree code can be seen in Fig. 3. As $N$ increases the execution time clearly increases much slower in $\text{body6}+p3r$ than the direct-N algorithm of $\text{body6}$. At $N = 1024k$, the tree code is faster than the direct-N code by approximately a factor of 20.

In terms of $T_{I}$, $\text{body6}+p3r$ performs comparably with $\text{body6}$ as seen in Fig. 4. Fig. 5 shows that $\text{body6}+p3r$ performs slightly more irregular integration steps than $\text{body6}$, but Fig. 6 shows that particles in $\text{body6}+p3r$ have far fewer neighbours on average which would lead to less time spent calculating irregular forces per particle. Both programs also use the same irregular integration algorithm.

Fig. 4 also shows that $\text{body6}+p3r$ spends more time performing regular integration $T_{R}$ than $\text{body6}$. The regular integration algorithms used by each program are not comparable, making it difficult to diagnose the disparity. The total number of regular steps performed by $\text{body6}+p3r$ is fixed at $N/\Delta_{\text{reg}}$ per unit time. The number of regular steps performed by $\text{body6}$ is dependent upon the Hermite algorithm which drives regular integration. Fig. 5 shows that...
Figure 5. The average number of integration steps per particle per N-body unit time as a function of \(N\). Both regular and irregular integration steps are shown. All runs are the same as those in figure 2.

Figure 6. The average number of neighbors per particle at start time as a function of \(N\). All runs are the same as those in figure 2.

\textsc{nbbody6+3r} performs more regular steps per particle than \textsc{nbbody6}, over four times as many for \(N > 10^6\), which would explain why \textsc{nbbody6+3r} spends more time overall performing regular integration than \textsc{nbbody6}.

The average number of neighbors in \textsc{nbbody6+3r} is much smaller, which is reflected in the irregular force calculation time. However, the total integration time is slower than in \textsc{nbbody6} because the number of irregular and regular steps per particle is higher due to the smaller maximum time step value. In larger system sizes this is not so obvious because the regular force calculation is much more computationally expensive.

Fig. 7 shows the execution time of each component as a proportion of the total execution time for that run. Included in this figure is the time \(T_{\text{List}}\), which measures the time needed for the particle scheduler to determine which particles are due to be integrated at each step. As \(N\) gets large in \textsc{nbbody6}, the regular force calculation, \(T_{F_R}\), becomes the dominant component accounting for nearly all of the computation time.

However, in \textsc{nbbody6+3r}, \(T_{F_R}\) is no longer dominant. In fact, irregular integration \(T_{I}^{\text{int}}\) becomes the dominant component accounting for approximately 50% to 60% of the run time consistently. The total time spent performing both irregular and regular integration is consistently 70% to 80% of the total run time. Moreover, the proportion of time spent performing force calculations, both regular and irregular, decreases significantly as \(N\) gets large; less than 10% altogether at \(N = 2048k\).

Both \textsc{nbbody6} and \textsc{nbbody6+3r} spend around 10% of the run time in \(T_{\text{List}}\), however this proportion is decreasing in \textsc{nbbody6} and increasing in \textsc{nbbody6+3r}. Both codes use the same algorithm; first the particles with small time steps are copied in to a smaller list \(L_Q\) at regular intervals. Next, the smaller list is scanned at every block step to determine which particles are due at that step. The first part has a run time complexity of \(O(N)\), and the second is \(O(N_Q)\) where \(N_Q\) is the size of \(L_Q\). If \(L_Q\) is repopulated too frequently, or if \(N_Q\) is too often larger than the actual number of particles due to be integrated, then it is possible for \(T_{\text{List}}\) to take longer than \(T_I^{\text{int}}\). The distributions of time steps are different in \textsc{nbbody6} and \textsc{nbbody6+3r}, which leads to different measurements of \(T_{\text{List}}\) in each. In \textsc{nbbody6+3r} it is possible to replace the scheduling technique described above with a priority queue, however it is beyond the scope of this paper.
3.2 Core Collapse

As a proof-of-concept, we performed a long run using \texttt{nbody6+p3r} to
the initial core collapse. The cluster, containing 64\textit{k} particles, was
arranged as a Plummer sphere with the following mass spectrum:

\[
m_i = \begin{cases} 10^{-9} & \text{if } i \equiv 0 \pmod{10}, \\ 10^{-8} & \text{otherwise}. \\ \end{cases}
\]

In this way, approximately half of the system’s mass is contained in
just 10% of the particles. All other parameters were the same as in
section 3.1.

An \texttt{nbody6} run with identical initial conditions was performed
to compare the evolution of system properties. Lagrange radii for
a range of mass proportions, total energy error \(\frac{|E(t) - E_0|}{E_0}\), core
density \(\rho_{\text{core}}\), core radius \(R_{\text{core}}\), core membership \(N_{\text{core}}\), and
the KS regularisation parameters \(R_{\min}\) and \(\Delta T_{\min}\) were all measured
in both runs.

3.2.1 Results

Fig. 8 shows the evolution of the Lagrange radii to the initial
core collapse. Both runs exhibit near-identical overall evolution for
the measured mass proportions. Additionally, Fig. 9 shows the
total energy change as a function of time; the \texttt{nbody6+p3r} run
recorded an error rate of the same order of magnitude as that of
the \texttt{nbody6} run. A linear regression over both series gave a slope of
6.47 \times 10^{-6} \pm 1.27 \times 10^{-7} E/T and 3.06 \times 10^{-6} \pm 3.02 \times 10^{-7} E/T
for the \texttt{nbody6} and \texttt{nbody6+p3r} runs, respectively.

As was mentioned in subsection 2.5, a fixed neighbour radius can be
problematic if too many particles do not have enough neighbours
to contribute to the calculation of the core quantities. Fig. 6 shows that
the average number of neighbours per particle in \texttt{nbody6+p3r} is quite
low. In fact, few particles have the requisite 6 neighbours required to
contribute to the calculations of \(R_{\text{core}}\) and \(N_{\text{core}}\).

A new algorithm was introduced to \texttt{nbody6+p3r} to accurately
approximate \(R_{\text{core}}, N_{\text{core}},\) and \(\rho_{\text{core}}\) values. Instead of using the
neighbour list of each particle to determine its 6 nearest neighbours,
our code takes a random sample of \(N, \text{SV000N}\) particles, then
performs a brute-force search for those particles’ 6 nearest neigh-
bour.

Figures 10, 11, and 12 respectively show the changes in these
values as a function of time using this new algorithm. With the
core quantities correctly determined, then regularisation parameters
are confidently determined. Fig. 13 shows both \(R_{\min}\) and \(\Delta T_{\min}\)
calculations; \texttt{nbody6+p3r} maintains small offsets from the output of
\texttt{nbody6}, but they converge to the same solutions during core collapse
when correct regularisation parameters are critical. Furthermore,
Fig. 14 shows the number of KS pairs formed as a function of time
for each program. \texttt{nbody6+p3r} produces roughly four times as many
pairs, and slightly more again during the later stages of the run; this
may indicate that the new core quantity algorithm requires some
adjustment.

4 CONCLUSIONS

We described a modified \texttt{nbody6} version to incorporate a GPU-
enabled P3T scheme. Using the optimal set of accuracy parameters
outlined in Iwasawa et al. (2015), our code outperformed \texttt{nbody6} for
system sizes \(N \geq 512k\).

For longer runs, we found that the \texttt{nbody6+p3r} code is sufficiently
accurate by comparing the total energy conservation and Lagrange

---

![Figure 8. Lagrange radii at different proportions of the total mass \(M\). The solid and dashed lines represent the \texttt{nbody6} and \texttt{nbody6+p3r} runs, respectively. Both runs used the same initial conditions; \(N = 64k\) particles in a Plummer sphere with a mass spectrum.](image8.png)

![Figure 9. Energy change as a function of time as a proportion of the initial energy. Both runs are the same as those in figure 8.](image9.png)
Accelerating NBODY6 with a GPU-Enabled P3T Scheme

Figure 10. The core radius of the cluster as a function of time. Both runs are the same as those in figure 8.

Figure 12. The core density of the cluster as a function of time. Both runs are the same as those in figure 8.

Our faster code provides the ability to accurately simulate the evolution of dense star clusters, like globular clusters or galactic nuclei, over long time scales (one Hubble time.) A realistic cluster could contain $10^6$ stars, making it advantageous to use our NBODY6+p3t code. Simulations of these systems can be used to study the conditions of black hole and neutron star mergers (Wang et al. 2016).

We found that the calculation of the KS regularisation parameters needed to be modified due to an inaccurate calculation of the core density $\rho_{\text{core}}$, which in turn was caused by the lower average neighbour number due to a fixed $r_{\text{cut}}$ value. We have mitigated the issue with a new core density algorithm, although this algorithm may need further adjustment, depending on the type of system that is being simulated.

Additional future improvements may include scaling the GPU-
enabled force calculator to utilise multiple GPUs, and to scale the computation over many nodes. A method to deal with higher ratios of binaries, like the one in Wang et al. (2020), would also be useful.

ACKNOWLEDGEMENTS

Parts of this work were performed on the OzSTAR national facility at Swinburne University of Technology. The OzSTAR program receives funding in part from the National Collaborative Research Infrastructure Strategy (NCRIS) Astronomy allocation provided by the Australian Government.

L.W. thanks the financial support from JSPS International Research Fellow (School of Science, The University of Tokyo).

DATA AVAILABILITY

The (benchmark) data underlying this article were generated by using the \texttt{Nbody6} and \texttt{Nbody6+P3t} codes at the supercomputer, Getafix, at the University of Queensland. The data underlying this article will be shared on reasonable request to the corresponding author.

Our \texttt{Nbody6+P3t} code is free for use and is available at 
\url{https://github.com/anthony-arnold/nbody6-p3t/tree/v1.0.0}.

REFERENCES

Aarseth S. J., 1985, in Multiple time scales. pp 377–418
Aarseth S., 1999, Celest. Mech. Dyn. Astron., 73
Aarseth S. J., 2003a, Gravitational N-Body Simulations: Tools and Algorithms. Cambridge Monographs on Mathematical Physics, Cambridge University Press, Cambridge, doi:10.1017/CBO9780511535246
Aarseth S. J., 2003b, in Makino J., Hut P., eds, IAU Symposium Vol. 208, Astrophysical Supercomputing Using Particle Simulations. p. 295 (\url{arXiv:astro-ph/0110148})
Ahmad A., Cohen L., 1973, Journal of Computational Physics, 12, 389
Barnes J., Hut P., 1986, Nature, 324, 446
Bédorf J., Gaburov E., Portegies Zwart S., 2012, in Capuzzo-Dolcetta R., Limongi M., Tornambé A., eds, Astronomical Society of the Pacific Conference Series Vol. 453, Advances in Computational Astrophysics: Methods, Tools, and Outcomes. p. 325 (\url{arXiv:1204.2288})
Casertano S., Hut P., 1985, ApJ, 298, 80
Duncan M. J., Levison H. F., Lee M. H., 1998, ApJ, 116, 2067
Giersz M., Heggie D. C., Hurley J. R., Hrypi A., 2013, MNRAS, 431, 2184
Heggie D. C., 1986, Star Cluster Dynamics: Mathematical Models. Springer-Verlag Berlin Heidelberg New York, p. 13, doi:10.1007/BFb0116390
Iwasawa M., Portegies Zwart S., Makino J., 2015, Computational Astrophysics and Cosmology, 2, 6
Kustaanheimo P., Schnizel A., Davenport E., Steifel E., 01 Apr. 1965, Journal für die reine und angewandte Mathematik, 1965, 204
Makino J., Aarseth S. J., 1992, Publications of the Astronomical Society of Japan, 44, 141
McMillan S. L. W., Aarseth S. J., 1993, ApJ, 414, 200
Mikkola S., Aarseth S., 1989, Celestial Mechanics and Dynamical Astronomy, 47, 375
Mikkola S., Aarseth S. J., 1998, New Astronomy, 3, 309
Nitadori K., Aarseth S. J., 2012, MNRAS, 424, 545
Oshino S., Funato Y., Makino J., 2011, Publications of the Astronomical Society of Japan, 63, 881–892
Plummer H. C., 1911, MNRAS, 71, 460
Rodriguez C. L., Morscher M., Wang L., Chatterjee S., Rasio F. A., Spurzem R., 2016, MNRAS, 463, 2109
Wang L., Spurzem R., Aarseth S., Nitadori K., Berczik P., Kouwenhoven M. B. N., Naab T., 2015, MNRAS, 450, 4070
Wang L., et al., 2016, MNRAS, 458, 1450
Wang L., Iwasawa M., Nitadori K., Makino J., 2020, MNRAS, 497, 536–555

APPENDIX A: COMPILATION

\texttt{Nbody6} traditionally uses the \texttt{GNU Make} program to build executables. The \texttt{Nbody6+P3t} environment is more complex, with the use of \texttt{bonsai} as a shared library and the need to ensure that compilation and linkage flags and \texttt{CUDA} versions remain consistent. To facilitate consistency, \texttt{Nbody6+P3t} uses the \texttt{CMake} program to build libraries and executables.

APPENDIX B: CODE MODIFICATIONS

Table B1 lists the routines from \texttt{Nbody6} which were updated in \texttt{Nbody6+P3t} along with a short description of what changes were necessary and Table B2 lists the new routines that were written for \texttt{Nbody6+P3t}. Table B3 describes the new common variables introduced to the program.

This paper has been typeset from a \TeX/\LaTeX file prepared by the author.
| Table B1. Modifications made to nbody6 routines for nbody6+p3t. |
|-----------------|---------------------------------------------------------------|
| **ADJUST**      | Avoid modifying the neighbour radius or regular step.       |
| **CORE**        | Determine core quantities using a nearest neighbour search of a sample of random particles. |
| **CMFIRR,**     |                                                               |
| **CMFIR2,**     |                                                               |
| **CPERTI,**     |                                                               |
| **CPERTX,**     |                                                               |
| **FCHAIN,**     |                                                               |
| **FIRR,**       |                                                               |
| **FPERT,**      |                                                               |
| **KCPERT,**     |                                                               |
| **KSPERT,**     |                                                               |
| **KSPOLY**      | Apply the smoothing functions to calculated forces.         |
| **FPOLY0**      | Read additional parameters and set constant neighbour radius. |
| **INTGR**       | Replace the regular Hermite scheme with a leapfrog scheme.   |
| **IPRED,**      |                                                               |
| **IPRED2,**     |                                                               |
| **KSPRED,**     |                                                               |
| **RESOLV,**     |                                                               |
| **XCVPRED,**    | Calculate predictions from equations (13) and (14).           |
| **KSINT**       | Apply smoothing functions. Avoid modifying the neighbour radius or regular step. |
| **KSTERM**      | Use the constant neighbour radius.                            |
| **MYDUMP**      | Read and write new common variables.                          |
| **NBINT,**      |                                                               |
| **NRINTP**      | Handle empty neighbour lists and $F_{R,t} = 0$.               |
| **NBLIST**      | Use the constant neighbour radius. Accept empty neighbour lists. |
| **OFFSET**      | Offset the regular block time variable TBLCKR.                |
| **REGF**        | Use BNSAI to calculate regular forces and determine the neighbour list. |
| **STEPS**       | Use a constant regular time step $\Delta t_{reg}$. Set $\Delta t_i = \Delta t_{max}$ if $F_{R,t} = 0$. |
| **TPERT**       | $\Delta t_i = \Delta t_{max}$ if the neighbour list is empty. |

| Table B2. New routines in nbody6+p3t |
|--------------------------------------|
| **CUTOFF**                          | Calculate $F_{1,t}$ and $\hat{F}_{1,t}$ from equations (1) and (2). |
| **CUTOFF0**                         | Calculate $F_{1,t}$ from equation (1). |
| **GPUMP**                           | Apply external forces from XTRNLF and FCLoud after REGF. |
| **GPUMP2**                          | Apply a velocity kick via RKICK and determine new time steps. |
| **RKICK**                           | Apply a velocity kick from equation (12). |

| Table B3. New common variables in nbody6+p3t |
|---------------------------------------------|
| **RBUFF**                                   | The parameter $r_{nuff}$. |
| **TBLCKR**                                  | Used to track the time of the last regular for calculation. |
| **NRMUL**                                   | A constant factor to determine the regular time step; $\Delta t_{reg} = NRMUL \times \Delta t_{max}$. Defaults to 4. |