Elias Rubin

Received _____; accepted _____

## 1. Scientific Motivation

Massive stars are believed to form the seed masses for intermediate-mass black holes (IMBHs) and supermassive black holes (SMBHs) (Bromm & Loeb 2003; Hosokawa et al. 2013). Observations of high redshift ($z \geq 6$) quasars place a constraint on the time allowed for IMBHs and SMBHs to form. To explain SMBHs with masses over $10^9$ $M_\odot$ forming under 1 Gyr, we require massive stars with masses around $10^4 - 10^5$ $M_\odot$ to form within a few Myr. These massive stars would then collapse due to general relativistic instability (Chandrasekhar 1964) into black holes which continue to accrete mass from their surroundings at the Eddington rate, as well as by continued mergers (Katz et al. 2015; Latif & Ferrara 2016). These IMBHs and SMBHs could result in powering many of the bright quasars we observe at high redshift.

In this work we examine the possibility of massive star formation by repeated collision in globular clusters and ultracompact dwarf galaxies. Sufficiently dense clusters can be highly collisional, and provide for the possibility of runaway growth. As a single star continues to increase in mass and stellar radius, it increases the likelihood of further collisions, and provides an avenue for runaway star growth (Katz et al. 2015). This mechanism for black hole formation is one of three that are commonly considered, the other two being core collapse black holes which typically are around 10 $M_\odot$, and the direct collapse of gaseous clouds, as described in Bromm & Loeb (2003) and Latif & Ferrara (2016). We follow the work of Portegies Zwart et al. (2004) and others by performing direct N-Body simulations of dense star clusters.

Core collapse black holes are believed to be too small to have much infuence on quasar abundance, as the seed mass is insufficient to form a $10^9$ $M_\odot$ black hole at redshifts above $z = 7$. Although direct collapse black holes have been investigated as a promising path to SMBHs formation, Latif & Ferrara (2016) and others raise questions about the viability of

that pathway because of significant radiation feedback effects which can reduce the rate of mass accretion below the 0.1 $M_\odot$/yr required to develop sufficiently large seed mass black holes.

A time constraint on the formation of our seed mass is the main-sequence lifetime of the most massive stars in the cluster. For a typical initial mass function with a maximum mass of $\sim 100\ M_\odot$, this time is about 3 Myr, afterwards stellar evolution effects such as supernovae and stellar winds can be expected to prevent runaway (Portegies Zwart & McMillan 2002; Latif & Ferrara 2016). We also expect that the largest stars initially dominate the set of stars that do ultimately form runaways, again based on their larger mass and radius increasing the likelihood of collision.

Portegies Zwart & McMillan (2002) suggests runaway collisions as a pathway to IMBH formation, with runaway masses at approximately 0.1% of the total cluster mass in the clusters simulated. Portegies Zwart et al. (2004)

**Should observational constraints or other literature be discussed in this section? What about more detail about alternative SMBH pathways like in Latif?**

**Cite for relationship between quasar luminosity and black hole mass?**

## 2. Model

**Questions to answer: What are the observed bodies that influence model? May need some sources for this. Put in a table/grid of all parameters surveyed.** In their investigation of collision clusters MGG-9 and MGG-11, Portegies Zwart et al. (2004) identify two factors about collisional clusters that can lead to supermassive star formation: concentration parameter $w$ and dynamical time $t_{\rm df}$, defined as follows:

$$w = \log \frac{R_t}{R_c}$$
$$t_{\rm df} = \frac{\langle m \rangle}{100\ M_\odot} \frac{0.138N}{\ln 0.11M/100\ M_\odot} \left(\frac{R^3}{GM}\right)^{1/2}$$

**Should put definitions of each parameter?**

## 3. Simulation Methods

**Does anything else belong in this section? A discussion of units, important cluster properties, more detail on numerical methods?** We are interested in following the evolution of collisional clusters and exploring the range of parameters in which they can lead to SMBH formation. Finding a solution entails solving the equation of motion for each particle specified by the force $\mathbf{F}_i$ and its time derivative $\mathbf{F}_i^{(1)}$ as follows:

$$\mathbf{F}_i = -m_i \sum_{i \neq j} G m_j \frac{\mathbf{R}}{R^3}; \tag{3.1a}$$

$$\mathbf{F}_i^{(1)} = -m_i \sum_{i \neq j} G m_j \left[\frac{\mathbf{V}}{R^3} + \frac{3\mathbf{R}(\mathbf{V}\cdot\mathbf{R})}{R^5}\right], \tag{3.1b}$$

where $G$ is the gravitational constant, $m_j$ is the mass of particle $j$, and $\mathbf{R}$ and $\mathbf{V}$ are the vector distance and velocity respectively between any two particles $i$ and $j$. Because it is not possible to analytically solve the dynamical equations for every body in a cluster, systems must be evolved numerically. Developing numerical methods for N-Body simulations is a long tradition, but the two most popular methods are tree-based simulations and direct simulations. The tradeoff between the two is one of computational cost versus accuracy. Notwithstanding implementation details, direct simulations compute pairwise force terms for every pair of bodies at each discrete simulation timestep, which comes

with a computational cost per timestep of $O(N^2)$ in the number of particles. Tree-based simulations partition the simulation space and compute pairwise terms for close neighbors of a given particle, but aggregate forces from groups outside of a close radius. Thus, the cost per timestep goes as $O(N\log N)$.

The metric generally used for simulation error is energy conservation across the system. Practicioners focus their efforts on increasing simulation speed while keeping increases to errors within acceptable bounds. The acceptable level of error depends on the problem domain and desired fidelity of the result. For our cluster simulations we have many bodies in close proximity and care about the accurate treatment of close encounters, binaries, multiple systems, and collisions, so we use `NBody6++GPU` , a direct code. An alternative we considered is the tree-based code `Starlab`, but our initial testing and recommendations from practicioners suggested that `NBody6++GPU` would be better for our use case.

We perform direct N-body simulations over a range of cluster parameters varying in total mass, half-mass radius, and concentration. We use the `McLuster` software of Küpper et al. (2011) to generate initial conditions and the `NBody6++GPU` software of Wang et al. (2015) to evolve the clusters. We also describe some modifications to the `NBody6++GPU` software made in attempts to accelerate simulations.

**What is the integrator scheme; how does it work? What are the numerical methods used?** `NBody6++GPU` is the latest version in a family of direct N-body simulators beginning with `NBody1` of Aarseth (1963). Aarseth (1999) describes in detail the evolution of the `NBody` family of programs and numerical methods therein. The code is written primarily in Fortran but has modules written in CUDA for GPU extensions and C++ for access to AVX/SSE instructions. It is parallelized with OpenMP.

In order to solve equations 3.1, the `NBody6++GPU` integrator uses a direct fourth-order Hermite integration method, as well as a hierarchical block step (Wang et al. 2015, and refs

within). This is a kind of adaptive timestep system. Instead of evaluating all particles at every timestep, particles are binned into quantized groups, and only those particles at an integer multiple of their timestep are integrated. This speeds up the integration of slow moving particles, while still allowing for accuracy for more extreme particles. The code uses the neighbor scheme of Ahmad & Cohen (1973) to speed up the integration further by separating into the more economical regular force (forces on a body generated by bodies outside a neighbor radius) and irregular force (forces generated by bodies within a neighbor radius). In `NBody6++GPU` , the regular force computations are done on the GPU, and irregular force computations are done on the CPU with the AVX/SSE library of Nitadori & Aarseth (2012). The next part of this section explains each of these in more detail.

The Hermite integration method is used to solve equations 3.1. This is a predictor-corrector method that uses a third-order Taylor expansion. The expression for the force is described in Aarseth (1999) and is as follows:

$$\mathbf{F}_{t+1} = \mathbf{F}_t + \mathbf{F}_t^1 \Delta t + \frac{1}{2}\mathbf{F}_t^{(2)} \Delta t^2 + \frac{1}{6}\mathbf{F}_t^{(3)} \Delta t^3, \tag{3.2a}$$

$$\mathbf{F}_{t+1}^{(1)} = \mathbf{F}_t^{(1)} + \mathbf{F}_t^{(2)} \Delta t + \frac{1}{2}\mathbf{F}_t^{(3)} \Delta t^2. \tag{3.2b}$$

$\mathbf{F}$ and $\mathbf{F}^{(1)}$ can be computed at the beginning and end of a timestep by using equations 3.1. They are used to form the higher order terms

$$\mathbf{F}_t^{(2)} = \frac{2}{\Delta t^2}\left[-3(\mathbf{F}_t - \mathbf{F}_{t+1}) - 2(\mathbf{F}_t^{(1)} + \mathbf{F}_{t+1}^{(1)})\Delta t\right], \tag{3.3a}$$

$$\mathbf{F}_t^{(3)} = \frac{6}{\Delta t^3}\left[2(\mathbf{F}_t - \mathbf{F}_{t+1}) + (\mathbf{F}_t^{(1)} + \mathbf{F}_{t+1}^{(1)})\Delta t\right]. \tag{3.3b}$$

Then the predictor and corrector $\mathbf{r}_{p,t+1}, \mathbf{v}_{p,t+1}, \Delta\mathbf{r}$, and $\Delta\mathbf{v}$ are given thusly

$$\mathbf{r}_{p,t+1} = \mathbf{r}_t + \mathbf{v}_t\Delta t + \frac{1}{2}\mathbf{F}_t\Delta t^2 + \frac{1}{6}\mathbf{F}_t^{(1)}\Delta t^3, \tag{3.4a}$$

$$\mathbf{v}_{p,t+1} = \mathbf{v}_t + \mathbf{F}\Delta t + \frac{1}{2}\mathbf{F}_t^{(1)}\Delta t^2, \tag{3.4b}$$

$$\Delta\mathbf{r} = \frac{1}{24}\mathbf{F}_t^{(2)}\Delta t^4 + \frac{1}{120}\mathbf{F}_t^{(3)}\Delta t^5, \tag{3.4c}$$

$$\Delta\mathbf{v} = \frac{1}{6}\mathbf{F}_t^{(2)}\Delta t^3 + \frac{1}{24}\mathbf{F}_t^{(3)}\Delta t^4, \tag{3.4d}$$

with the final state at $t+1$ given by $\mathbf{r}_{t+1} = \mathbf{r}_{p,t+1} + \Delta\mathbf{r}$ and $\mathbf{v}_{t+1} = \mathbf{v}_{p,t+1} + \Delta\mathbf{v}$.

The Hermite predictor is used for each time step in the hierarchical block step scheme. Block steps are the way the `NBODY` family of codes addresses the large dynamic range in the velocities and proximities of bodies in clusters. This is necessary for performance because otherwise the time step for the entire system would be determined by the separation of the closest bodies. Instead each particle is given its own timestep using the following formula of Khalisi, E. and Wang, L. and Spurzem, R. (2017)

$$\Delta t_i = \sqrt{\eta \frac{|\mathbf{a}_{1,i}||\mathbf{a}_{1,i}^{(2)}| + |\mathbf{a}_{1,i}^{(1)}|^2}{|\mathbf{a}_{1,i}^{(1)}||\mathbf{a}_{i,1}^{(3)}||\mathbf{a}_{1,i}^{(2)}|^2}}. \tag{3.5}$$

We use an initial value of 0.01 for $\eta$ which we arrived at as it generally allowed our simulations to progress further before energy error becomes too great which forces a restart. This value may be further adjusted if the relative energy error $dE$ (defined as the difference in total simulation energy between checkpoint times) becomes close (within a factor of 5) to the maximum tolerance $Q$. The correction factor is given by $\sqrt{dE/Q}$. Our simulations use a checkpoint timestep of 0.01 $t_{\rm nb}$, where $t_{\rm nb}$ is the simulation time unit and is equal to $\frac{1}{2\sqrt{2}}t_{\rm df}$ (Khalisi, E. and Wang, L. and Spurzem, R. 2017). We set $Q = 0.05$, which allows for a maximum energy error of 20%, and causes a readjustment of $\eta$ when the energy error reaches 4%.

Figure 1 shows an example of how the particles are advanced.

The `NBody6++GPU` employs the neighbor scheme of Ahmad & Cohen (1973) in conjunction with the Hermite integrator and block timestep. This is a further optimization to speed up the calculations, this time by identify particles that are spatially close to each other. Thus, a given particle will actually have two timesteps, a regular timestep and (smaller) irregular timestep. Particles have a neighbor list of size at most $N_{\rm nb} \ll N_{\rm tot}$. These are advanced using a similar predictor-corrector as the Hermite scheme above, computing the full force for neighbor particles and the last regular values for non-neighbors. Membership in the neighbor list is determined by the size of the neighbor sphere and the length of the neighbor list $N_{\rm nb}$. The first $N_{\rm nb}$ particles within a sphere of radius $r_{\rm search}$ are included. We found that too small values for $N_{\rm nb}$ would lead to code crashes and settled on a value of 1024. We use a search sphere of radius $1 \times 10^{-4}$ pc.

The distinguishing feature of the `NBODY` family is their use of Kustaanheimo & Stiefel (1965) (KS) regularization to generate solutions for binaries, triples, and multiple systems with high levels of accuracy. This treatment for close encounters adds a good deal of complexity to the implementation, but the scheme essentially works by searching for bodies closer to each other than $r_{\rm min}$, replacing them in the next integration step by their center of mass, and integrating them separately at a smaller timescale and in a different reference frame. This is useful to avoid numerical truncation errors caused by bodies being too close in the originally reference frame. The KS regularization is extended to multiple systems and adds perturbing bodies using a variant of the Ahmad & Cohen (1973) neighbor scheme.

Although `NBody6++GPU` uses MPI parallelization to scale across multiple compute nodes, there are some sequential bottlenecks including KS regularization. In our experiments, we experienced no gain from using multiple nodes, and it's possible that the KS regularization method was responsible. We touch on this further in section **??**.

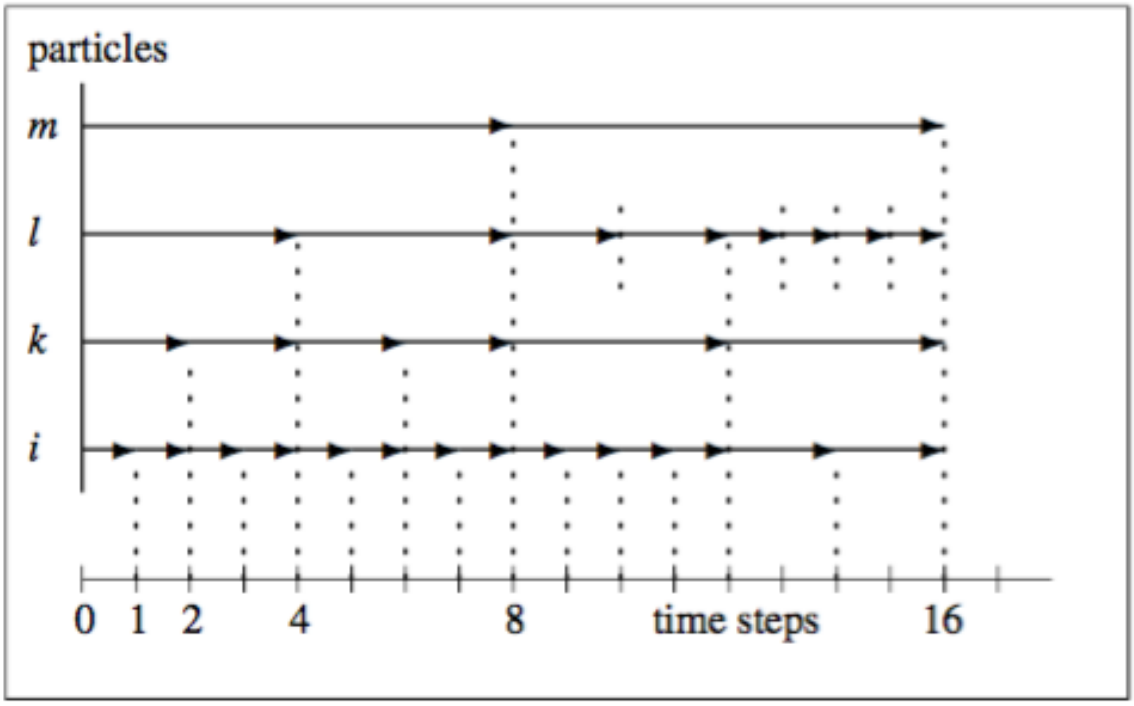



Fig. 1.— Particles $i$, $k$, $l$, and $m$ have separate time steps. A full force computation is done at each arrow, otherwise, the position and velocity are extrapolated from the values already computed. These time steps are reevaluated by equation 3.5 after a full integration cycle. Subsystems which are evaluated independently are replaced by their center of mass in the block scheme. Figure reproduced from Khalisi, E. and Wang, L. and Spurzem, R. (2017).

We performed our simulations on the `Tiger2` GPU cluster at Princeton University. Most experiments were performed on one compute node which each use four NVIDIA Tesla P100 GPUs and 2.4 GHz Broadwell processors parallelized with OpenMP.

**What are the initial conditions and how are they generated?** We generate our initial conditions using the `McLuster` software of Küpper et al. (2011). For the initial distribution of positions and velocities, we use a King (1966) profile, varying half-mass radius and concentration. We use the initial mass function of Kroupa (2001) which degenerates to a Saltpeter IMF above 0.5 $M_\odot$. For our simulations with cluster mass $2 \times 10^5$ $M_\odot$, we use a mass range of 1 to 100 $M_\odot$. For cluster mass $6 \times 10^5$ $M_\odot$, we increase the lower bound to 3.3 $M_\odot$. For cluster mass $2 \times 10^6$ $M_\odot$, we use a lower bound of 13 $M_\odot$ and an upper bound of 120 $M_\odot$. We do not apply an external tidal field, initialize primordial binaries, or apply primordial mass segregation.

**What are the boundary and escape conditions?** We use an open boundary condition so particles that do escape do not return to the simulation. Our escape condition is a distance from center of greater than twice the tidal radius. Both of these conditions remain uniform over all of our simulations.

**What is the merger scheme?** The default merger criteria in `NBody6++GPU` follows that of Kochanek (1992). For a two-body system with pericenter $R_p$, the bodies are merged if

$$\frac{R_p}{R_1} \leq 1.3 \times (\frac{M_T}{2M_1})^{1/3}$$

.

This merger scheme is conservative. In the case of a central mass with $m = 200$ $M_\odot$ (a typical case after a couple of mergers) and an additional mass with $m = 1$ $M_\odot$, and using

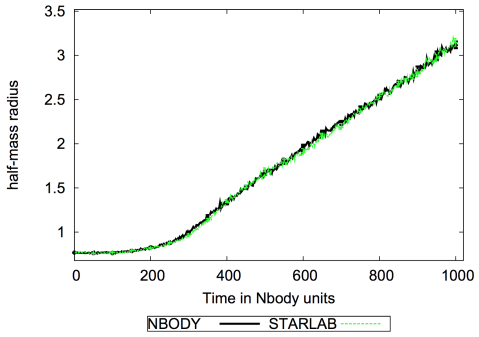
$r_{\rm central} = (\frac{m}{M_\odot})^{0.55}$, we would not merge the two stars unless they had a pericenter distance smaller than 20 stellar radii, or a deviation of only about 3% from the radius of the central mass.

In an effort to increase the speed of our simulations by spending less time in KS regularization for insignificant bodies, we experimented with short-circuiting this merger scheme for pairs meeting certain conditions. We experimented with merging binaries where one member was 5 or 10 times larger than the other, provided the pericenter distance was smaller than $10r_{\rm large}$ (in the more conservative trial) and $100r_{\rm large}$ in the more liberal one. We avoided merging with the central mass, but did want to reduce the number of KS pairs for peripheral bodies. We found that the overall runtime was not affected significantly by this change to the merger scheme. Simulation results appeared robust to the factor 10 difference, but were affected by the factor 100 difference. We tested changes to our merger scheme at approximately the transition between collisional and non-collisional clusters: a mass of $2 \times 10^5$ $M_\odot$, a concentration of 2, and a half-mass radius of 2.9 pc.

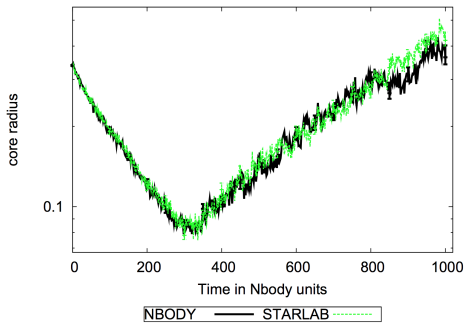
**How was verification performed?** Anders et al. (2009, 2012) perform comparisons between `NBody4` of Aarseth (1999) and `Starlab` environment. In order to verify our installation of `NBody6++GPU` we repeated the tests from Anders et al. (2009). We used `McLuster` to generate similar initial conditions to the ones used in their test. Our verification clusters consist of 1024 1 $M_\odot$ stars, distributed in a Plummer profile with initial half-mass radius 0.5 pc. We performed test runs with and without stellar evolution mass loss, as `NBody6++GPU` requires stellar evolution mass loss to perform mergers.

The trends of our verification runs generally agreed with those in Anders et al. (2009). The authors left some uncertainty about the exact parameters used for their runs, and they also sampled many runs to obtain an average. Figures 2 and 3 compare the two.

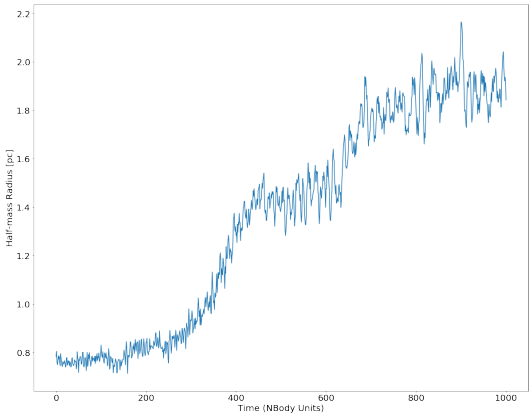
**What other issues with original code? How were they resolved?** We had



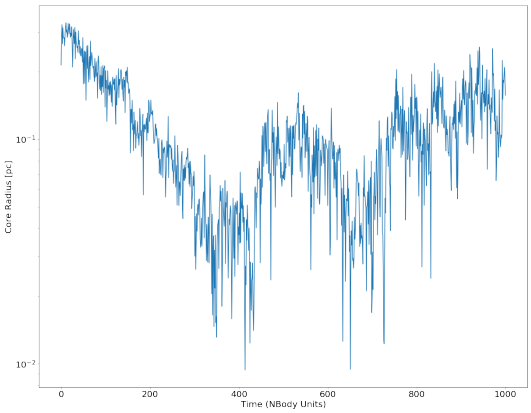

(a) Half-mass radius vs. time reproduced from Anders et al. (2009).




(b) Core radius vs. time reproduced from Anders et al. (2009).

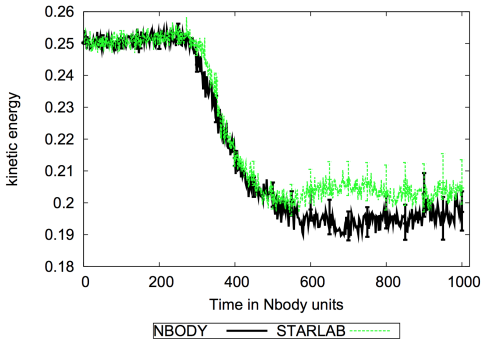



(c) Half-mass radius vs. time from our verification run.

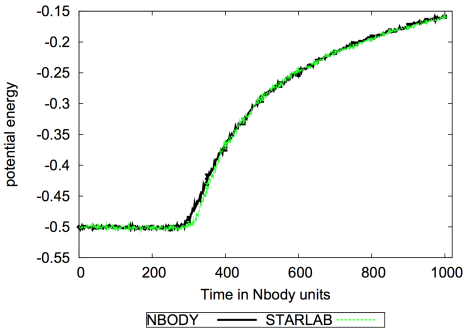



(d) Core radius vs. time from our verification run.

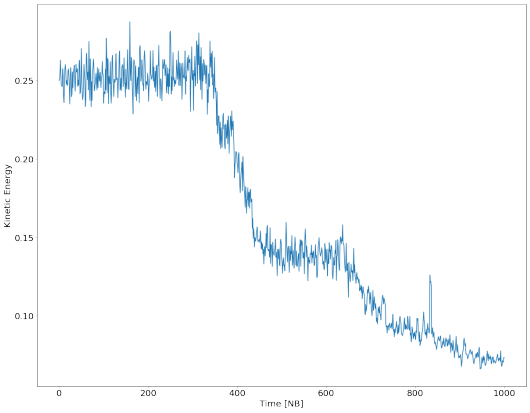
Fig. 2.— Our verification run shows generally good agreement with those of Anders et al. (2009). The half mass radius doesn't start to expand significantly until 200 time steps in both sets, and then follows a general upwards trend. Our core radius shows a similar peak-trough distance of oscillation and a return close to the starting point by the end of the run. Our result appears to oscillate more because we show a single run instead of an entire sample.



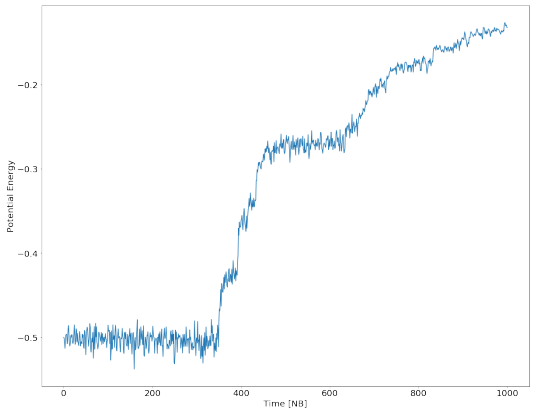

(a) Kinetic energy vs. time reproduced from Anders et al. (2009).




(b) Potential energy vs. time reproduced from Anders et al. (2009).




(c) Kinetic energy vs. time from our verification run.




(d) Potential energy vs. time from our verification run.

Fig. 3.— We see the same general trend with kinetic and potential energy from our verification run, although the kinetic energy from our run seems to end up lower than that of Anders et al. (2009). This is where there is also the greatest divergence between `NBODY4` and `Starlab`, so it's possible that changes in between `NBODY4` and `NBody6++GPU` account for this difference.

significant difficulty running more collisional clusters to completion. We found that clusters with many collisions stretched the limits of default `NBody6++GPU` parameters. Among other issues, we encountered a persistent problem where runs would stop progressing during a KS regularization. We settled on a two-part workaround for this problem: first by reducing the KS step size which seemed to forestall these failures at the cost of some additional compute time and second by simply restarting runs once they stopped progressing.

We were unable to get `NBody6++GPU` 's internal restart module to function in our environment, so our restart method required building a tool to parse the checkpoint data and create initial conditions for a fresh simulation. The raw checkpoint data does not perfectly reflect the state of the system because it contains duplicate bodies when binaries, triples, quads, and chains are involved. We implemented an additional heuristic in our restart code to remove spurious binaries and other multiples, leaving only the constituent bodies in the new initial conditions.

`NBody6++GPU` 's default configuration does not always seem to handle collisional clusters well. This led to other failure modes for our simulations which we tried to correct. In dense clusters we found that the neighbor lists would rapidly become full leading to code crashes. Here we tried reducing the size of neighbor search spheres per the suggestion of 201 (????, personal communication), which did not resolve the issue, so we instead doubled the size of the neighbor lists which was enough to alleviate the issue. Another failure mode which we have not yet been able to find a solution to occurs after restarts during GPU force initialization. It appears some initial conditions cause GPU out-of-memory errors which cause `NBody6++GPU` to crash. The resolution we have adopted for that failure mode is to binary search snapshots until finding one that proceeds and begin the run at that point. We then have to invalidate data after the selected snapshot.

## 4. Results

**What clusters had runaway central masses?**

**What fraction of the total cluster mass did the runaway achieve? How long didit take?**

**What was the rate of growth? Did this vary with cluster parameters? What about the shape of the growth curves?**

**Any clusters that were too expensive to simulate or didn't work for some other reason?**

**What other observed phenomena? Did any/all clusters experience mass loss? What happened with core radius? Was core collapse correlated with larger central mass?**

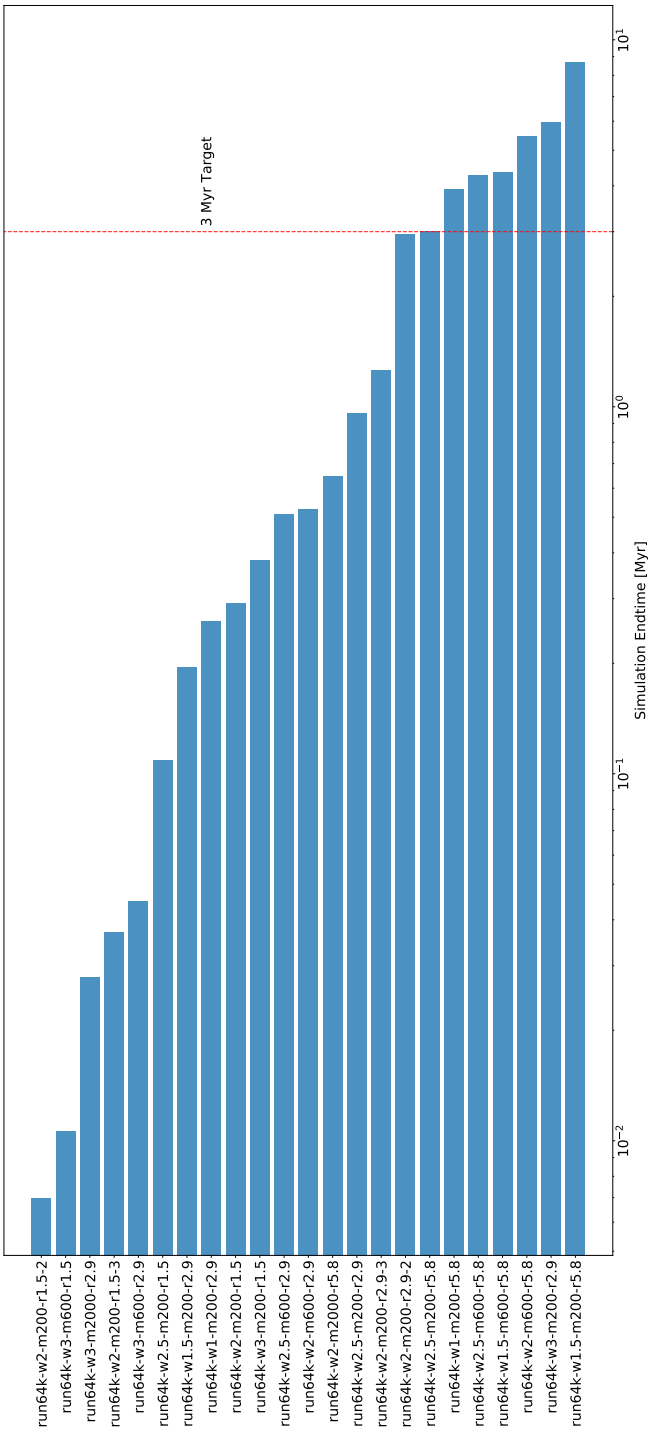



Fig. 4.— Not all jobs ran to completion.

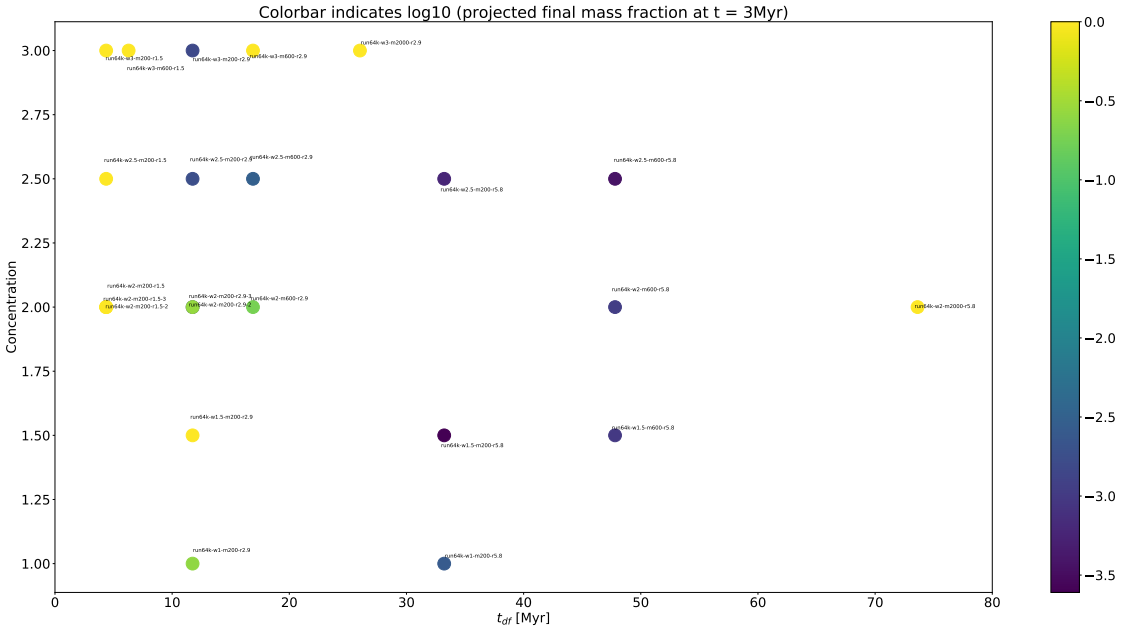



Fig. 5.— Plot of clusters with their parameter $K$

## REFERENCES

????

Aarseth, S. J. 1963, MNRAS, 126, 223

—. 1999, PASP, 111, 1333

Ahmad, A., & Cohen, L. 1973, Journal of Computational Physics, 12, 389

Anders, P., Baumgardt, H., Bissantz, N., & Portegies Zwart, S. 2009, MNRAS, 395, 2304

Anders, P., Baumgardt, H., Gaburov, E., & Portegies Zwart, S. 2012, MNRAS, 421, 3557

Bromm, V., & Loeb, A. 2003, ApJ, 596, 34

Chandrasekhar, S. 1964, ApJ, 140, 417

Hosokawa, T., Yorke, H. W., Inayoshi, K., Omukai, K., & Yoshida, N. 2013, ApJ, 778, 178

Katz, H., Sijacki, D., & Haehnelt, M. G. 2015, MNRAS, 451, 2352

Khalisi, E. and Wang, L. and Spurzem, R. 2017

King, I. R. 1966, AJ, 71, 64

Kochanek, C. S. 1992, ApJ, 385, 604

Kroupa, P. 2001, MNRAS, 322, 231

Küpper, A. H. W., Maschberger, T., Kroupa, P., & Baumgardt, H. 2011, MNRAS, 417, 2300

Kustaanheimo, P., & Stiefel, E. 1965, J. Math. Bd, 218, 27

Latif, M. A., & Ferrara, A. 2016, PASA, 33, e051

Nitadori, K., & Aarseth, S. J. 2012, MNRAS, 424, 545

Portegies Zwart, S. F., Baumgardt, H., Hut, P., Makino, J., & McMillan, S. L. W. 2004, Nature, 428, 724

Portegies Zwart, S. F., & McMillan, S. L. W. 2002, ApJ, 576, 899

Wang, L., Spurzem, R., Aarseth, S., et al. 2015, MNRAS, 450, 4070