The Open Journal of Astrophysics
Preprint typeset using LaTeX style openjournal v. 09/06/15

A NOVEL ALGORITHM FOR GPU-ACCELERATED PARTICLE-MESH INTERACTIONS IMPLEMENTED IN THE QUOKKA CODE

Chong-Chong He[1], Benjamin D. Wibking[2], Aditi Vijayan[1], Mark R. Krumholz[1], Pak Shing Li[3]
[1]Research School of Astronomy and Astrophysics, Australian National University, 233 Mount Stromlo Road, Stromlo, ACT 2612, Australia
[2]Department of Physics and Astronomy, Michigan State University, East Lansing, MI 48824, USA
[3]Shanghai Astronomical Observatory, Chinese Academy of Sciences, 80 Nandan Road, Shanghai 200030, PR China
Version March 19, 2026

Abstract

We present a novel, GPU-optimized algorithm for particle-mesh interactions in grid-based hydrodynamics simulations, designed for massively parallel architectures. This approach overcomes the inefficiency of particle neighbour searches or sorts across multiple GPU nodes by using a new "particle-mesh-particle" interaction scheme, which extends the particle-mesh method for self-gravity. The algorithm proceeds in two main stages: first, quantities exchanged between particles and the mesh – such as mass, energy, and momentum added by stellar feedback or removed by accretion onto a sink – are deposited into a buffer mesh equipped with ghost zones, where multiple contributions per cell are accumulated using atomic additions and then communicated across distributed memory ranks. In the second stage, the buffer states are applied to real mesh states, incorporating cell-wise limiters to enforce physical constraints such as positive density. We implement this scheme in the GPU-native radiation-magnetohydrodynamics code Quokka and validate it through a comprehensive suite of tests, including Bondi and Bondi-Hoyle accretion, and single and multiple supernova remnant evolution at varying spatial resolutions. We show that the algorithm achieves $\approx 50\%$ weak-scaling efficiency running on up to 8192 GPUs on the Frontier supercomputer. This scheme enables efficient, scalable particle-mesh coupling for GPU-optimized simulations of star formation and feedback in galaxies.

Subject headings: accretion, accretion discs — hydrodynamics — methods: numerical — supernovae: general

1. INTRODUCTION

Gravitational collapse is ubiquitous in gaseous astrophysical systems, but following collapsing regions in hydrodynamic simulations demands ever-increasing spatial and temporal resolution. It is therefore inevitable in such calculations that one must define a maximum resolution beyond which one does not follow further collapse. One standard approach to this problem is to replace unresolved high-density peaks with point masses that approximate the behaviour and evolution of unresolved small-scale structures, which may represent individual (proto-)stars, stellar populations, black holes, or similar compact objects. Subsequent to the creation of these particles, gas and particles are integrated simultaneously, including their mutual gravitational forces and possibly also interaction via other mechanisms such as radiation.

The integration of particles into hydrodynamic simulations has emerged as a vital approach in understanding complex astrophysical processes, from star formation and stellar feedback to black hole accretion. Bate et al. (1995) first introduced the technique of sink particles – particles that can accrete gas in addition to interacting with it gravitationally – in smoothed particle hydrodynamics (SPH) codes, while Krumholz et al. (2004) pioneered their use in grid-based codes. Over the past two decades, similar implementations have appeared in various hydrodynamics codes, including gadget (Jappsen et al. 2005), flash (Federrath et al. 2010), enzo (Wang et al. 2010), Ramses (Dubois et al. 2010; Bleuler & Teyssier 2014), and Athena (Gong & Ostriker 2013). Most of these authors primarily focus on star formation simulations, but some also employ sink particles to model supermassive black holes in cosmological contexts.

In addition to accretion, particles are also frequently used as sources of feedback in hydrodynamic simulations. The oldest example of this is supernova feedback in simulations of galaxy formation, wherein gas above some density threshold is converted to collisionless particles that subsequently add energy back to the hydrodynamic particles or grid; the literature contains numerous recipes for treating this process (e.g., Katz 1992; Oppenheimer & Davé 2006; Dalla Vecchia & Schaye 2012; Kimm & Cen 2014; Kim & Ostriker 2017; Hopkins et al. 2018; Hirashima et al. 2025). Other examples include particles representing stars that produce ionizing radiation (e.g., Dale et al. 2005, 2007), thermal radiation (e.g., Krumholz et al. 2007, 2009; Offner et al. 2009), protostellar jets (e.g., Cunningham et al. 2011), and massive stellar winds (e.g., Dale et al. 2014; Gatto et al. 2017), as well as particles representing black holes that similarly exert energetic and thermal feedback on the gas around

chongchong.he@anu.edu.au

them (e.g., Springel et al. 2005; Di Matteo et al. 2005).

All of the algorithms above have been implemented in codes that run on CPU-based architectures, but these are now giving way to more powerful GPU-based systems that present particular programming challenges. Even on CPUs, neighbour searches often represent a major contributor to the computational cost in particle-based hydrodynamics methods, and this problem is worse on GPUs, where efficiency is limited by the data dependencies and irregular memory access patterns inherent to such methods. When running in parallel, the neighbour list generation step requires exchanging particles across all nodes, which becomes inefficient on multiple GPU nodes due to complex synchronization and communication needs. For instance, in GPU-accelerated SPH codes like that of Crespo et al. (2011), the calculation of interactions between a target particle and its neighbours represents the most computationally expensive step.

One might think that particle-mesh methods, in which one follows the hydrodynamics on a mesh while allowing particles to interact with that mesh, would avoid these challenges and allow efficient computation on GPUs. This is true to some extent. Meshes have the advantage of eliminating the neighbour search, since it is trivial to determine the computational cell in which a particular particle is located, and this in turn allows efficient deposition of particle information onto a grid, and conversely interpolation of mesh data back to particles; GPU-accelerated codes such as Nyx (Almgren et al. 2013) and WarpX (Fedeli et al. 2022) exploit this capability to achieve high performance for particle-mesh (PM) gravity and particle-in-cell (PIC) plasma simulations, respectively. However, in these applications particle-mesh interactions are far simpler than is the case for either sink accretion or stellar feedback; particle deposition to the mesh in both PM and PIC simulations is simply additive, but this is *not* the case for either accretion or feedback, which usually involve complex particle-mesh interaction rules that are problematic to implement on GPU.

As one example of this difficulty, most recent implementations of supernova feedback (Kim & Ostriker 2017; Hopkins et al. 2018; Hirashima et al. 2025) involve modifying the properties of cells or particles within some distance of the explosion site with recipes such that the results when multiple supernovae occur are strongly dependent on the order in which the particles contributing supernovae are processed; that is, the results of adding supernova feedback from particle A and then B are not the same as the result of doing so from B and then A. On a CPU one can resolve this issue simply by enforcing an order in which the particles are processed, but ensuring serially-ordered processing of particles that are potentially distributed across multiple nodes in memory is either impossible or prohibitively expensive on GPUs. To make matters worse, if multiple GPUs need to process the supernova feedback recipe (for example because the supernovae are occurring at the edge of a domain decomposition), there is no easy way to guarantee that they will do so in the same order, and thus a naive implementation of CPU-based supernova feedback recipes might yield inconsistent results from one GPU to another. This is just one example of the numerous rules for particle merging, sorting, or prioritization invoked by the particle recipes listed above that do not translate trivially to GPU architectures.

To overcome these problems and allow efficient calculation of more complex particle-mesh interactions on modern GPU hardware, in this paper we introduce a novel "particle-mesh-particle" algorithm. This method specifically tackles the challenges of particle-particle and particle-mesh interactions in a GPU-friendly framework. The major advantages of this algorithm include avoiding the need for hard-to-accelerate particle neighbour searches across distributed GPU memory, producing communication and computation in a predictable pattern with no indirect memory referencing to enable efficient GPU acceleration, and requiring only a single communication step per update cycle. We implement this method in the Quokka GPU-accelerated radiation-hydrodynamics code (Wibking & Krumholz 2022; He et al. 2024a,b).

The remainder of this paper is organized as follows. In Sec. 2, we describe the overall algorithm and communication protocols for particle-mesh and particle-particle interactions. In Sec. 3, we detail our implementations of sink particle formation and accretion (appropriate for simulations of individual star formation) and stellar population formation and supernova feedback (appropriate for galaxy-scale simulations) within this framework, and in Sec. 4 we present a series of tests that demonstrate the accuracy and efficiency of this approach. Finally, Sec. 5 summarizes our findings.

2. THE "PARTICLE-MESH-PARTICLE" ALGORITHM

2.1. *Formulation of the problem*

In this section, we outline the fundamental algorithm that we use for particle-mesh interactions. We consider a mesh consisting of quadrilateral cells with indices ijk, each of which stores a vector of conserved quantities $\mathbf{U}_{ijk}$. In a minimal hydrodynamics application $\mathbf{U}_{ijk}$ would contain the mass density, momentum density, and total energy in each cell, but this can be straightforwardly generalized to more complex methods where $\mathbf{U}_{ijk}$ might include, for example, mass densities of particular chemical species or radiation energy densities and fluxes; we require only that the quantities stored in $\mathbf{U}_{ijk}$ be conserved volumetric quantities. Similarly, the mesh might be uniformly-spaced, or it could be an adaptive mesh in which the cell volume V_{ijk} varies from cell to cell; this too will not matter for our purposes. Finally, the mesh may be domain-decomposed in memory, so that any given GPU only has access to some fraction of the cells making up the computational domain.

Within this mesh we also have a series of particles, indexed by s; each particle is characterised by a position $\mathbf{x}_s$, which lies within some host cell $\{ijk\}_s$. Particles follow the same domain decomposition as the mesh, i.e., if cell $\{ijk\}_s$ lies within the domain of a given GPU, then particle s will also be stored on that GPU. Particles may also carry conserved quantities $\mathbf{u}_s$ that correspond to the quantities stored in the state variable and obey conservation laws; for example, sink particles typically carry a total mass and momentum, and the accretion interaction between particles and gas must conserve total

mass and momentum, so

$$\sum_{ijk} \mathbf{U}_{ijk} V_{ijk} + \sum_s \mathbf{u}_s = \text{constant}, \tag{1}$$

where $\mathbf{u} = (m, m\mathbf{v})$ are the mass and momentum of a particle with velocity $\mathbf{v}$, and $\mathbf{U} = (\rho, \mathbf{p})$ are the cell mass and momentum density.

Now consider an interaction whereby particle s should alter the conserved quantities in cell ijk by an amount $\Delta\mathbf{U}^s_{ijk}$; we refer to $\Delta\mathbf{U}^s_{ijk}$ as the deposition function, which specifies the amount of each conserved quantity that a particle deposits in each cell. Deposition can take the form of a direct and permanent alteration to the conserved quantities themselves (for example accretion of mass by a sink particle), or to a temporary deposition for the purposes of a subsequent calculation (for example depositing particle mass into the gas density field for the purposes of computing the gravitational potential as in a PM gravity method), and can be both positive (for example addition of energy by SN feedback) or negative (for example reduction in density due to sink accretion). Particles can affect cells other than their host cell, but we require that $\Delta\mathbf{U}^s_{ijk}$ be non-zero only over a kernel of radius r_K centred on the particle position, so that $\Delta\mathbf{U}^s_{ijk}$ is zero for any cell whose central position $\mathbf{x}_{ijk}$ satisfies $|\mathbf{x}_{ijk} - \mathbf{x}_s| > r_K$.

Crucially, we allow multiple particles to impose alterations on a single cell, i.e., we can have $\Delta\mathbf{U}^s_{ijk} \neq 0$ and $\Delta\mathbf{U}^{s'}_{ijk} \neq 0$ for two distinct particles s and s', and we allow limiting, so that if $\sum_s \mathbf{U}^s_{ijk}$ is too large – for example if the collective effect of multiple sink particles accreting from a given cell would be to make its density negative, or the collective effect of momentum addition from multiple supernovae would lead to a violation of the total supernova energy budget – we can modify the sum before altering the conserved quantities. These two features are important for GPU-based implementations, because without them we would be forced to implement an algorithm to find neighbouring particles whose deposition zones overlap and implement logic for handling that case (e.g., by merging particles), and it is precisely such an operation that we wish to avoid because it is inefficient on GPU.

2.2. *Algorithm steps*

With this general framework in place, we now describe our algorithm, emphasizing the synchronization and communication protocols critical for GPU optimization. We defer specific details of how we implement different particle types to Sec. 3, and we provide additional technical details of our method to achieve bitwise-reproducible results in Appendix A. The algorithm comprises five steps, which we summarize in Fig. 1:

Step 1: Particle to buffer mesh. First, we allocate a buffer of conserved quantities $\Delta\mathbf{U}^{\rm buf}_{ijk}$ with the same dimensions as the hydrodynamic mesh, with all cells initialized to zero. In a distributed-memory parallel calculation this buffer is domain-decomposed exactly as is the hydrodynamic mesh, but it includes enough ghost zones (also known as guard cells) to fully contain the interaction kernel radius r_K, so if the host cell $\{ijk\}_s$ of particle s is owned by a given GPU, then all the cells ijk for which $\Delta\mathbf{U}^s_{ijk}$ is non-zero are included in the buffer and its ghost zones. We then iterate over all particles and evaluate the deposition function $\Delta\mathbf{U}^s_{ijk}$ for all cells and all particles in local memory, storing the result to the buffer, i.e., we carry out the operation

$$\Delta\mathbf{U}^{\rm buf}_{ijk} = \sum_s \Delta\mathbf{U}^s_{ijk}, \tag{2}$$

where the sum runs over all particles s stored on a given GPU. This deposition must be carried out with some care to avoid race conditions on GPU, as we discuss further in Appendix A.

Step 2: Inter-GPU summation of the buffer mesh. We next sum the buffer mesh across distributed memory ranks, i.e., we add the contents of the ghost zones on each rank to the corresponding real zones on other ranks, so that at the end of this procedure $\Delta\mathbf{U}^{\rm buf}_{ijk}$ contains the contributions from all particles s that deposit to a given cell, even if those particles are hosted on a different rank. We denote these parallel-summed quantities

$$\Delta\mathbf{U}^{\rm sum}_{ijk} = \sum_s \Delta\mathbf{U}^s_{ijk}, \tag{3}$$

where this expression differs from Eq. 2 in that this summation is over all particles s anywhere in the computational domain, not just those that reside on the same GPU as the cell. Again, some care is required in carrying out this step, as described in Appendix A.

Step 3: Apply limiter. As noted above, to avoid unphysical states when multiple particles deposit to the same hydrodynamic cell, it may be necessary to apply limiters – for example limiting the amount of mass removed from a given cell by sink accretion to avoid making the density negative. We defer discussion of the specific forms of limiting we implement for different particle-mesh interaction types to Sec. 3, and here simply abstract the limiting process as a cell-wise operation of the form

$$\Delta\mathbf{U}^{\rm lim} = f_{\rm lim}(\mathbf{U}, \Delta\mathbf{U}^{\rm sum}), \tag{4}$$

where $f_{\rm lim}$ is some limiter. We store the ratio of the limited and unlimited quantities $\boldsymbol{\eta}_{ijk} = \Delta\mathbf{U}^{\rm lim}_{ijk}/\Delta\mathbf{U}^{\rm sum}_{ijk}$ for use in the next steps. Note that $\boldsymbol{\eta}_{ijk}$ is a vector that can be different for each conserved quantity, though in practice it is often the same for all quantities.

Step 4: Apply limited changes to particles. The fourth step is to update particle properties to account for particle-mesh interaction; note that for particles we carry out this update here, rather than as part of step 1, because limiting may affect the update to particle properties – again returning to the example of sink particles, if we have limited the change in density $\Delta\rho_{ijk}$ in some cell ijk, then the change in mass of the sink particles that accrete from this cell must be similarly affected so as to ensure conservation of total mass. Thus in this step we repeat the calculation of $\Delta\mathbf{U}^s_{ijk}$ for each particle from step 1, summing over all the cells ijk, and we then update the particle properties based on the limited changes

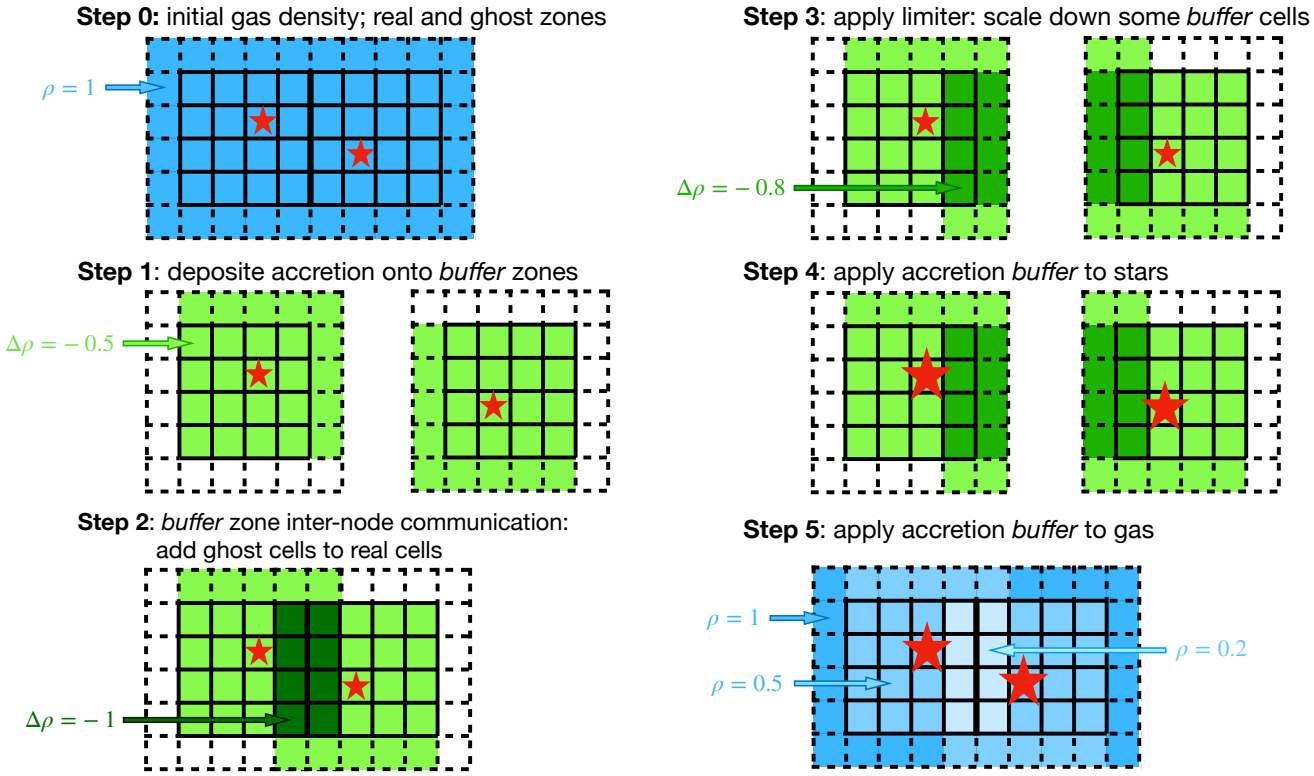

FIG. 1.—: Summary of the particle-mesh-particle algorithm for particle-gas interactions on GPUs, illustrated for the case of sink accretion. In the example, two sink particles (red stars) lie near the boundaries between two grid blocks that reside on different GPUs; the thick black line indicates the block boundary. Cells with dashed edges are ghost zones used for communication. For clarity we show only one layer of ghost zones and two-cell deposition kernel in this schematic, but our production runs employ four or more ghost zones and three-cell deposition kernel. Blue shading indicates gas density, while green shows the buffer quantity (here, the accretion rate). The annotated numbers show example values at various steps: the initial density $\rho = 1$ at Step 0, the change in density $\Delta\rho = -0.5$ computed independently over the accretion kernel for the two particle on each rank at Step 1, the inter-rank buffer sum that increases the mass change to $\Delta\rho = -1$ in the overlapping cells at Step 2, the limiter that scales down this change to $\Delta\rho = -0.8$ at Step 3, the increase in sink mass at Step 4, and the decrease in gas density to new, smaller values in Step 5.

in each cell:

$$\mathbf{u}_s := \mathbf{u}_s - \sum_{ijk} \boldsymbol{\eta}_{ijk} \Delta\mathbf{U}^s_{ijk} V_{ijk}. \quad (5)$$

Step 5: Apply limited changes to the mesh. The final step is to deposit the limited quantities back onto the mesh. Mathematically, this step amounts to carrying out

$$\mathbf{U}_{ijk} := \mathbf{U}_{ijk} + \Delta\mathbf{U}^{\rm lim}_{ijk} = \mathbf{U}_{ijk} + \boldsymbol{\eta}_{ijk} \sum_s \Delta\mathbf{U}^s_{ijk} \quad (6)$$

over all cells. It is manifestly clear that this process obeys the conservation constraint Eq. 1.

Before moving on to the details of how we implement this approach for both sink particle accretion and supernova feedback in Sec. 3, we pause to make some observations. First, not all steps of this algorithm are required for all types of particles. Simple particle-mesh interactions of the type used by PM or PIC simulations only require steps 1, 2, and 5, because there is no limiting and because interactions do not modify particle properties, only mesh properties. Similarly, some of the particle algorithms we describe in Sec. 3 will use only some of the steps.

Second, the only step in the entire algorithm that is not local to a single GPU is step 2, and the communication pattern associated with this step is *exactly* the same as that used for ghost zone filling during the hydrodynamic update. Thus as long as the interaction kernel radius r_K is small enough that interaction buffers do not require more ghost zones than hydrodynamics, the computational cost of this step is guaranteed to be no larger than the computational cost of hydrodynamic ghost cell filling. This cost is minimized by the usual approach of using a space-filling curve to carry out domain decomposition such that physically-adjacent parts of the domain are whenever possible hosted on the same computational node, and this strategy is sufficiently efficient that GPU-optimized hydrodynamics codes such as QUOKKA and ATHENAK (Stone et al. 2024) show near-linear weak scaling even when running on tens of thousands of GPUs.

3. NUMERICAL IMPLEMENTATIONS: SINK AND SUPERNOVA FEEDBACK PARTICLES

Having outlined our general particle-mesh-particle interaction framework, we now provide detailed example implementations for two important use-cases: sink particles for star formation simulations (Sec. 3.1) and stellar population particles with supernova feedback for galaxy formation simulations (Sec. 3.2).

3.1. *Sink particles*

As described in Sec. 1, sink particles are commonly used to represent accreting protostars in simulations of star formation, and thus the key interaction between such particles and the hydrodynamic mesh that we seek to capture is accretion. Our implementation of sink particles in Quokka carries out the following steps during every time step on the finest adaptive mesh refinement level:

1. Integrate the motion of existing sink particles. We carry out this step using the leapfrog method to integrate the particle position and velocities in time, including the effects of the gravitational interactions between sink particles and gas; we compute the gravitational accelerations using a standard PM gravity method, which we implement using steps 1, 2, and 5 of the algorithm described in Sec. 2. We follow a "kick-drift-kick" strategy, which is shown to work better than "drift-kick-drift" approach for variable time step (Springel 2005).

2. Compute accretion onto existing sink particles using all steps of the algorithm in Sec. 2; we discuss the implementation details in Sec. 3.1.1. We retain the buffer mesh containing the cell-wise accretion rates at the end of this step, for use in the subsequent step.

3. Create sink particles following the algorithm outlined in Sec. 3.1.2; this takes the buffer mesh constructed during the accretion stage as an auxiliary input.

The details of the accretion and particle creation steps are as follows.

3.1.1. *Sink particle accretion*

Accretion from sink particles uses all steps of the algorithm outlined in Sec. 2; to fully specify our implementation, we must therefore define our kernel radius r_K and provide the functions $\Delta\mathbf{U}^s_{ijk}$, which describes the change in conserved quantities in each cell due to accretion onto each sink, and $f_{\rm lim}(\mathbf{U}, \Delta\mathbf{U})$, the limiter function.

Our default choice is $r_K = 3\Delta x$, where Δx is the cell spacing, but this is adjustable at compile time, and our tests below show similar results for $r_K = 2\Delta x$ or $4\Delta x$ (see Sec. 4.2). We take our model for the accretion function $\Delta\mathbf{U}^s_{ijk}$ from Krumholz et al. (2004), who use the Bondi-Hoyle accretion formula to approximate the accretion rate. In this approach, we first compute the Bondi-Hoyle radius for each particle

$$r_{\rm BH} = \frac{Gm_s}{v_\infty^2 + c_\infty^2}, \tag{7}$$

where m_s is the sink mass, and v_∞ and c_∞ are the mass-weighted mean relative velocity and mass-weighted mean sound speed over cells within the accretion kernel, respectively. Next, we compute the Bondi-Hoyle-Lyttleton accretion rate (Hoyle & Lyttleton 1939), using the approximation provided by Ruffert (1994) to interpolate between the limits of $v_\infty \ll c_\infty$ and $v_\infty \gg c_\infty$:

$$\begin{aligned}\dot{m}_{\rm BH} &= 4\pi\rho_\infty G^2 m_s^2 \left[\frac{\lambda^2 c_\infty^2 + v_\infty^2}{(c_\infty^2 + v_\infty^2)^4}\right]^{1/2} \\ &= 4\pi\rho_\infty r_{\rm BH}^2 \left(\lambda^2 c_\infty^2 + v_\infty^2\right)^{1/2}\end{aligned} \tag{8}$$

where λ is a constant of order unity that depends on the gas equation of state; we adopt the value for isothermal gas, $\lambda = e^{3/2}/4 \approx 1.120$, throughout this work. Third, we apply an accretion kernel to distribute the accretion rate across the accretion zone, weighting each cell by

$$w = \exp\left(-r^2/r_{\rm acc}^2\right), \tag{9}$$

where $r = |\mathbf{x}_s - \mathbf{x}_{ijk}|$ is the distance from the particle to the cell centre, and

$$r_{\rm acc} = \begin{cases} \Delta x/4, & r_{\rm BH} < \Delta x/4 \\ r_{\rm BH}, & \Delta x/4 \le r_{\rm BH} \le r_K/2 \\ r_K/2, & r_{\rm BH} > r_K/2 \end{cases}. \tag{10}$$

Thus the normalised weight for each cell inside the accretion kernel is

$$\phi_{ijk} = \frac{w_{ijk}}{\sum w_{ijk}}, \tag{11}$$

where the sum runs over all cells inside the accretion kernel. Fourth, we increase the accretion rate in any cell where the density is high enough to violate the Truelove et al. (1997) condition for stability against artificial fragmentation, which requires that the density not exceed

$$\rho_{\rm Tr} = J^2 \frac{\pi c_s^2}{G\Delta x^2}. \tag{12}$$

Here $J = 0.25$ is the Jeans number required for stability, c_s is the sound speed in the cell, and Δx is the cell size. In cells for which $\rho_{ijk} > \rho_{\rm Tr}$, we increase the accretion rate enough to ensure that the density falls to $\rho_{\rm Tr}$, thereby ensuring that the calculation does not undergo artificial fragmentation. Putting these conditions together, our final expression for the accretion function for minimal hydrodynamics (for which the conserved quantities are density, momentum, and total energy) is

$$\Delta\mathbf{U}^s_{ijk} = \max\left(\dot{m}_{\rm BH}, \dot{m}_{\rm Tr}\right) \frac{\phi_{ijk}\Delta t}{V_{ijk}} \begin{pmatrix} 1 \\ \mathbf{v} \\ e_{\rm sp} + v^2/2 \end{pmatrix}_{ijk}, \tag{13}$$

where ϕ_{ijk} is the kernel weight (Eq. 11) and

$$\dot{m}_{\rm Tr} = \frac{(\rho_{ijk} - \rho_{\rm Tr})V_{ijk}}{\phi_{ijk}\Delta t}, \tag{14}$$

is the accretion rate required to enforce the Truelove et al. condition, Δt is the time step, V_{ijk} is the cell volume, and $\mathbf{v}$ and $e_{\rm sp}$ are the cell velocity and specific internal energy. Extensions beyond minimal hydrodynamics are straightforward. Quokka includes a dual-energy formalism to follow high-Mach number flows, and so we also follow the internal energy as an additional, auxiliary conserved

quantity; the accretion formula for this quantity is the same as for total energy, but with $e_{\rm sp} + v^2/2$ replaced by $e_{\rm int,sp}$, where $e_{\rm int,sp}$ is the internal energy per unit mass. In calculations including chemistry or passive scalars, we set the changes in their densities to keep their concentrations invariant, i.e., we choose $\Delta\rho_{\rm sp} = (\rho_{\rm sp}/\rho)\Delta\rho$, where $\rho_{\rm sp}$ is the density of a given species. Finally, in magnetohydrodynamic calculations, we follow Myers et al. (2013) by not altering magnetic fields or magnetic energy densities to avoid artificially creating magnetic divergence.

Our limiter function is chosen to ensure stability by ensuring that the mass in a given cell is reduced by no more than 25% per time step (Krumholz et al. 2004) unless doing so is necessary to enforce the Truelove et al. condition. If the mass accreted is limited, then the momentum and energy changes are limited by the same factor. Thus our limiting function is

$$f_{\rm lim}(\mathbf{U}, \Delta\mathbf{U}^{\rm sum}) = \left(\frac{\Delta\rho^{\rm lim}}{\Delta\rho}\right)\Delta\mathbf{U}, \tag{15}$$

where

$$\Delta\rho^{\rm lim} = \max\left[\min\left(\Delta\rho, \frac{\rho}{4}\right), \rho - \rho_{\rm Tr}\right]. \tag{16}$$

Equivalently, the ratio of the limited to unlimited changes is $\boldsymbol{\eta}_{ijk} = \Delta\rho^{\rm lim}/\Delta\rho$ for all quantities.

3.1.2. *Sink particle creation*

We also follow a slightly-modified version of the Krumholz et al. (2004) criterion for sink particle creation, which is to insert a sink particle in cells that violate the Truelove et al. (1997) condition for artificial self-gravitating fragmentation. This condition amounts to requiring that the ratio of the cell size to the local Jeans length remain below a threshold $J \approx 0.25$, which in turn yields the density threshold given by Eq. 12. We therefore mark cells within which $\rho_{ijk} > \rho_{\rm Tr}$ as candidates for sink particle creation. However, we only create sink particles in cells that satisfy two additional criteria that were not present in the original Krumholz et al. formulation: we require that a sink particle is only created if: (1) the cell is a local density maximum within a sphere of radius r_K; (2) no other sink particle exists within r_K.[1]

We enforce the latter condition by forbidding sink particle creation in any cell within which $\Delta\mathbf{U}_{ijk}$ is non-zero. We add these two additional criteria due to a subtle change required in the Krumholz et al. approach to make it GPU-friendly. In the original approach, sink particles could be created in multiple adjacent cells, but were then immediately merged, which prevented the creation of large numbers of sink particles in close proximity. We do not use this approach here because selecting particles to merge requires constructing particle trees across distributed GPU memory, an inefficient operation that we wish to avoid. Our approach of only allowing sink creation at local maxima and not within the accretion kernel of another sink particle, but then increasing the accretion rate itself enough to enforce the Truelove et al. condition, achieves the same effect as the original Krumholz et al. prescription without the additional computational overhead of constructing parallel particle trees on GPU.

If a cell does satisfy the sink creation conditions, we set the initial sink mass to $m_s = (\rho - \rho_{\rm Tr})V_{ijk}$, and reduce the cell density to $\rho_{\rm Tr}$. We leave the cell velocity and specific internal and kinetic energies unchanged, and set the initial momentum of the sink particle to $m_s\mathbf{v}_{ijk}$, where $\mathbf{v}_{ijk}$ is the velocity of the cell in which the sink particle is created.

3.2. *Massive stellar particles and supernova feedback*

In simulations of entire galaxies, the resolution is generally too low to capture individual accreting protostars, which begin their accretion upon second collapse after reaching masses of only $\sim 10^{-3}$ $M_\odot$. Instead, the particles used in galaxy-scale simulations are intended to represent stellar populations. Depending on the resolution, such populations can be either integrated, representing a collection of stars large enough to treat supernovae as a continuous wind, or stochastic, representing stellar populations small enough to treat each supernova individually. The implementation for GPU we present here is an example of the latter, and thus is similar in spirit to the stochastic stellar population models implemented by, for example, Gatto et al. (2017), Fujimoto et al. (2018), Armillotta et al. (2019), Applebaum et al. (2020), and Jeffreson et al. (2021) for the FLASH, ENZO, GIZMO, CHANGA, and AREPO codes, respectively.

Our basic steps for supernova feedback particles are much the same as for sink particles. During each time step, we

1. Integrate the motion of existing particles under the influence of gravity using a "kick-drift-kick" integration, with the potential computed using a PM method.
2. Inject supernova feedback from stars at the ends of their lives (Sec. 3.2.1).
3. Create new particles in cells where the density has risen past our ability to resolve (Sec. 3.2.2).

We describe the second and third steps in this algorithm in detail in the next two sections.

3.2.1. *Supernova feedback*

As we discuss further in Sec. 3.2.2, each of our particles represents a single massive star, which has a known formation time and an initial mass assigned at birth. We determine the lifetime of each such star by interpolating on the MIST tracks for rotating, Solar-metallicity stars (Choi et al. 2016), and we determine which stars end their lives in core collapse supernovae from the tabulation of Sukhbold et al. (2016). For each time step we

[1] We do not add the further restrictions on sink particle formation (e.g., requiring $\nabla \cdot \mathbf{v} < 0$) advocated by some authors (e.g., Federrath et al. 2010; Haugbølle et al. 2018). We do not do so because these criteria substantially affect the outcome of fragmentation only when the gas remains isothermal or close to it up to the sink particle creation density; if the effective equation of state stiffens at lower densities, either due to artificial stiffening (e.g., Jappsen et al. 2005; Kratter et al. 2010) or inclusion of physical processes such as radiative transfer (e.g., Krumholz et al. 2007, 2009), fragmentation occurs on resolved scales and the results become relatively insensitive to the details of the sink particle prescription. In the isothermal limit where details of the sink particle creation criteria do matter, it is far from clear that it is possible or meaningful to obtain a converged result for fragmentation (e.g., Guszejnov et al. 2018). For this reason, we do not attempt to implement more involved sink particle creation recipes.

first find all the stars whose lives end during that time step, and then inject supernova back from those stars. Feedback injection follows the general scheme laid out in Sec. 2, with the exception that we skip step 4 because there is no feedback from the computational grid to the properties of the stellar particles. The remaining steps can be fully specified by the choice of accretion kernel radius r_K (as with sink particles, we use $r_K = 3\Delta x$ by default), the deposition function $\Delta\mathbf{U}^s_{ijk}$, and the limiter $f_{\rm lim}(\mathbf{U}, \Delta\mathbf{U})$.

The deposition function.— We calculate the deposition function using a variant of the TIGRESS prescription (Kim & Ostriker 2017) adapted to be GPU-friendly. Following the TIGRESS approach, we first calculate the total gas mass within the kernel radius r_K, which we take to be $M_{\rm snr} = \sum \omega_{ijk}\rho_{ijk}V_{ijk} + M_{\rm ej}$, where the sum runs over all cells within the stencil and ω_{ijk} is the fraction of the grid cell that is covered by the stencil, and $M_{\rm ej}$ is the mass ejected by the supernova itself. In a production simulation we obtain $M_{\rm ej}$ by interpolating on the tabulated values provided by Sukhbold et al. (2016) in the same way we determine which stars end their lives as supernovae, but for the purposes of comparing with the results of Kim & Ostriker we follow them in setting $M_{\rm ej} = 10$ M$_\odot$ for all stars in the tests presented below. We then compute the ratio of this mass to the shell-formation mass, defined as the approximate swept-up mass at the point when the interior of a supernova remnant transitions from adiabatic to radiative and thus forms a dense shell. This ratio is $\mathcal{R}_M \equiv M_{\rm snr}/M_{\rm sf}$, where $M_{\rm sf} = 1679(n_{\rm H}/{\rm cm}^{-3})^{-0.26}$ M$_\odot$ and $n_{\rm H}$ is the ambient number density of H nuclei (Kim & Ostriker 2015). We estimate the latter quantity as $n_{\rm H} = M_{\rm snr}/\mu_{\rm H}V_{\rm snr}$, where $\mu_{\rm H}$ is the mass per H nucleus ($\approx 1.4m_{\rm H}$ for standard He abundances) and $V_{\rm snr} = \sum \Delta V_{ijk}$ is the total volume of the accretion kernel; here the sum again goes over cells satisfying $|\mathbf{x}_{ijk} - \mathbf{x}_s| < r_K$.

We then calculate the deposition function using three different prescriptions depending on the value of $\mathcal{R}_M$; following Kim & Ostriker, we refer to these as the `EJ` (Ejecta), `ST` (Sedov-Taylor), and `MC` (Momentum Conserving) regimes, depending on whether the resolution is sufficient to capture the transition between free expansion of the ejecta and the adiabatic Sedov-Taylor phase (the `EJ` case), the transition between the Sedov-Taylor phase and the momentum-conserving snowplow phase (the `ST` case), or neither of these transitions (the `MC` case). We describe our deposition function with the state array (with mass, momentum, and total energy as conserved quantities)

$$\Delta\mathbf{U}^s_{ijk} = \begin{pmatrix} \omega_{ijk}M_{\rm ej}/V_{\rm snr} \\ \rho_{\rm new}\vec{v}_{\rm COM} - \rho_{ijk}\vec{v}_{ijk} + \vec{p}_{\rm radial} \\ \omega_{ijk}\left(E_{\rm sn} + E_{\rm kin}\right)/V_{\rm snr} + \vec{v}_{\rm COM}\cdot\vec{p}_{\rm radial} \end{pmatrix}, \tag{17}$$

where $\rho_{\rm new} = \rho_{ijk} + \omega_{ijk}M_{\rm ej}/V_{\rm snr}$ is the updated cell density, $E_{\rm sn} = 10^{51}$ erg is the supernova energy, $E_{\rm kin} = \frac{1}{2}m_{\rm ej}v_s^2$ is the kinetic energy of the supernova ejecta, $\vec{p}_{\rm radial} = |\vec{p}_{\rm radial}|\hat{r}$ is the supernova momentum injection in the radial direction in a coordinate system centred on the exploding particle, and $\vec{v}_{\rm COM}$ is the centre-of-mass velocity of the supernova remnant,

$$\vec{v}_{\rm COM} = \frac{\sum \omega_{ijk}\rho_{ijk}V_{ijk}\vec{v}_{ijk} + m_{\rm ej}\vec{v}_{\rm ej}}{M_{\rm snr}}. \tag{18}$$

The cross term $\vec{v}_{\rm COM}\cdot\vec{p}_{\rm radial}$ accounts for the work done by the radial expansion against the bulk motion of the remnant. This formulation effectively smooths the velocity field (but not the density) before deposition. Because $\vec{p}_{\rm radial}$ is isotropic, the cross term cancels when summing over all cells in the stencil, and thus the work done on all cells by the radial expansion sums to zero, ensuring global energy conservation. This also guarantees the supernova deposition is Galilean invariant, i.e., the outcome of a SN event is independent of the velocity of the frame in which it is computed.

To motivate this procedure, it is helpful to evaluate what occurs if we do not smooth the background velocity and only add the radial component, i.e. $\Delta p = \vec{p}_{\rm radial}$. In this case, the cross term in the energy update becomes $\vec{v}_{ijk}\cdot\vec{p}_{\rm radial}$, which, when summed over the stencil, becomes $\sum \vec{v}_{ijk}\cdot\vec{p}_{\rm radial} = \sum(\vec{v}_{\rm COM} + \delta\vec{v}_{ijk})\cdot\vec{p}_{\rm radial} = \sum \delta\vec{v}_{ijk}\cdot\vec{p}_{\rm radial}$, where $\delta\vec{v}_{ijk}$ is the cell velocity relative to the centre of mass. This introduces spurious energy into the system, which can be significant when a supernova event occurs shortly after another at the same location, such that $\delta\vec{v}_{ijk}$ is parallel to $\hat{r}$ and thus $\vec{p}_{\rm radial}$ in all cells. We note that, by smoothing the background velocity, the kinetic energy in the centre-of-mass frame before deposition is lost. However, because we update the total energy and the internal energy is derived from the other conserved quantities, this lost energy effectively becomes internal energy, which is subsequently handled by radiative cooling.

The magnitude of the momentum deposition $|\vec{p}_{\rm radial}|$ changes between regimes as

$$|\vec{p}_{\rm radial}| = \begin{cases} P_{\rm snr}, & \mathcal{R}_M > 1\,(\texttt{MC}) \\ \sqrt{2M_{\rm ej}\epsilon_K E_{\rm sn}}, & 0.027 < \mathcal{R}_M \le 1\,(\texttt{ST}) \\ 0, & \mathcal{R}_M \le 0.027\,(\texttt{EJ}) \end{cases} \tag{19}$$

Here $\epsilon_K = 0.28$ is the fraction of energy in kinetic form during the Sedov-Taylor phase and $P_{\rm snr}$ is the terminal momentum we expect at the end of the Sedov-Taylor phase. These expressions amount to assuming that in the `MC` case the supernova adds only kinetic energy and momentum and so the internal energy per unit mass stays constant, while in the `ST` regime the changes in kinetic and internal energy match those expected for the Sedov-Taylor solution, and in the `EJ` case the supernova adds only thermal energy. Note that, because we are always depositing an energy of $E_{\rm sn}$ into the gas total energy, in the `MC` case the supernova adds a small amount of thermal energy to the gas because the total kinetic energy deposited is $P_{\rm snr}^2/2M_{\rm ej} = 4.6\times10^{50}$ erg $= 0.46E_{\rm sn}$. The remaining 54 per cent of the energy is deposited as thermal energy. This assignment of thermal and kinetic energy is handled with the limiter step discussed below.

We adopt a numerical value

$$P_{\rm snr} = 2.8\times10^5\,{\rm M}_\odot\,{\rm km\,s^{-1}}\left(\frac{n_{\rm H}}{\rm cm^{-3}}\right)^{-0.17} \tag{20}$$

for the terminal momentum directly from the Kim & Ostriker prescription. This is a fit to results from the simulations of Kim & Ostriker (2015); however, this numerical

value is roughly consistent with those produced by other studies of the remnants of single, isolated supernovae in both one dimension (Blondin et al. 1998; Thornton et al. 1998) and three dimensions (Iffrig & Hennebelle 2015; Martizzi et al. 2015).[2]

The limiter.— Our remaining task is to specify the limiter $f_{\rm lim}(\mathbf{U}, \Delta\mathbf{U})$. Limiting is required because Eq. 17 does not guarantee that the state produced after deposition obeys the constraint that $e - |\mathbf{p}|^2/2\rho \geq e_{\rm int,0}$, where e and $\boldsymbol{p}$ are the total energy and momentum per unit volume, and $e_{\rm int,0}$ is the internal energy in the initial state. Indeed, in rare cases the prescriptions above can cause the kinetic energy to exceed the total energy. For example, consider a region where the mass inside the deposition zone is large enough that we are in the MC regime, but where there is a particular cell with very low density, such that the change in its kinetic energy density, $\Delta e_K = (p+\Delta p)^2/[2(\rho+\Delta\rho)] - p^2/2\rho$, is larger than the change in its total energy density. To ensure that the states we produce after deposition are valid, we apply a limiter to rescale the momentum update vector to enforce consistency. The state is consistent if

$$C_V(\rho+\Delta\rho)T + \frac{|\boldsymbol{p}+\lambda\Delta\boldsymbol{p}|^2}{2(\rho+\Delta\rho)} \leq e + \Delta e, \qquad (21)$$

where C_V is the gas specific heat capacity, and we determine the value of the limiter λ by solving the resulting equation for λ and choosing the largest root in the range $[0, 1]$. The internal energy is then obtained by subtracting the kinetic energy from the total energy. In cases where the maximum solution for λ equals 1, this procedure effectively heats up the gas. This can be necessary, for example, in regions between two SN explosions, where the momentum vectors from two depositions cancel each other and thus $|\Delta\boldsymbol{p}|$ is close to zero; in this case our prescription correctly deposits the SN energy as heat.

Comparison to the original TIGRESS model.— Compared to the standard TIGRESS prescription outlined in Kim & Ostriker (2017), the primary changes to our approach are that we do not homogenise conditions inside the supernova deposition kernel, and we apply limiting only to the collective action of all supernovae rather than one supernova at a time. These changes lead to an algorithm that does not depend on the order in which supernova injection is processed, and that can cope with multiple supernovae depositing material in a single cell even if those particles reside on different parts of a decomposed domain – both crucial features for the algorithm to operate smoothly on GPU. An added benefit of our approach even on CPU is that the TIGRESS prescription, which involves homogenising the material inside the supernova deposition zone prior to adding energy or momentum, produces a result that depends on the order in which particles are processed. This means either that one must adopt an arbitrary rule for which particle is processed first, or that the results are non-repeatable because the particles are processed in whatever order they happen to be stored in memory, which may vary depending on details such as the number of MPI ranks used in the simulation. Our prescription does not depend on arbitrary ordering choices, avoiding these undesirable features.

Another difference from the TIGRESS prescription is that we employ a fixed kernel size for all prescriptions, whereas TIGRESS varies the kernel size from $R_{\rm snr,min} = 3\Delta x$ to a model-dependent $R_{\rm snr,max}$ (typically 128 pc). TIGRESS adopts this variable approach to mitigate overcooling issues when the Sedov-Taylor phase is unresolved. In contrast, we observe no such overcooling issue in our code, even with a fixed $R_{\rm snr}$ in the unresolved regime (see the test in Sec. 3.2). We attribute this to our use of a more accurate solver for temperature evolution: unlike TIGRESS, which relies on a single backward Euler step, we implement an ODE solver with adaptive time stepping to compute precise temperature updates. This enhancement avoids overcooling without requiring more than three ghost cells, thereby eliminating the need for additional communication beyond standard ghost-cell exchanges – a critical advantage for efficient GPU implementation.

3.2.2. *Formation of supernova feedback particles*

Our method of depositing supernovae on the grid is independent of our choice for how to generate the particles that eventually produce those supernovae. However, for completeness here we also describe our default particle creation prescription in Quokka. Since our target resolution regime is one where individual massive stars can be resolved but the process of fragmentation that determines the initial mass function (IMF) cannot (typical cell mass at maximum resolution $\sim 1 - 10^2$ $M_\odot$), we implement a stochastic stellar population model to emulate the formation and evolution of individual massive stars. In this model, the star formation rate is controlled through two parameters: efficiency per free-fall time, $\epsilon_{\rm ff}$, and the fraction of gas cell mass that is converted into stars when star formation does occur, ϵ_*. The first of these is a physical parameter describing an underlying theory for star formation on unresolved scales, which we set to $\epsilon_{\rm ff} = 0.01$ based on high-resolution observations of individual star-forming clouds (e.g., Krumholz et al. 2019; Pokhrel et al. 2021; Hu et al. 2022), while the latter is a numerical parameter that we set to $\epsilon_* = 0.5$, a choice that balances efficiency (favouring larger ϵ_*, so that we avoid having to spawn stars repeatedly from the same cell) against stability (favouring smaller ϵ_*, so that we perturb the hydrodynamic state minimally).

Given these parameters, our procedure is as follows: first, during each time step of size Δt, we identify cells on the finest adaptive mesh level where the density is above the Truelove et al. (1997) threshold given by Eq. 12. For this purpose we take $J = 0.5$ to be conservative; this leaves us vulnerable to artificial fragmentation, but since in this regime we are prescribing the mass distribution of stars rather than attempting to calculate it directly, this is not a major concern. However, we do not place a star particle in every cell above the threshold density, since doing so would be computationally expensive. Instead, following the common approach in cosmological galaxy formation simulations, we treat star formation stochas-

[2] The terminal momentum of the remnants produced by multiple supernova clustered closely enough in time that several explosions contribute to the expansion remains significantly uncertain, and may well depend on the degree of inhomogeneity or other properties of the background into which those remnants expand; see Gentry et al. (2017, 2019, 2020), Kim et al. (2017), and El-Badry et al. (2019) for further discussion.

tically by assigning a probability P of forming stars in any eligible cell. To determine P, note that the expected stellar mass formed from a cell of gas mass $M_{\rm cell}$

$$\langle M_* \rangle = P \epsilon_* M_{\rm cell} \, . \tag{22}$$

We then demand that $\langle M_* \rangle$ match the value predicted by the star formation efficiency per free-fall time $\epsilon_{\rm ff}$, which requires

$$\langle M_* \rangle = \epsilon_{\rm ff} M_{\rm cell} \frac{\Delta t}{t_{\rm ff}} \, , \tag{23}$$

where $t_{\rm ff} = \sqrt{3\pi/32G\rho}$. Inverting the above two equations, the probability that a star that is eligible to do so forms a star during a time step is

$$P = \min \left[\left(\frac{\epsilon_{\rm ff}}{\epsilon_*} \right) \left(\frac{\Delta t}{t_{\rm ff}} \right), 1 \right] . \tag{24}$$

In each time step we therefore randomly assign each cell that is eligible to form star to be star-forming or non-star-forming with probability P.[3]

After identifying a cell for star formation, we spawn a stellar population drawn from a Chabrier (2005) IMF, following the "Poisson" sampling strategy implemented by the SLUG stochastic stellar populations code (Krumholz et al. 2015). In our implementation, every stellar population comprises one particle representing the low-mass end of the IMF and a random number of individual high-mass stars; we draw the boundary between "low-mass" and "high-mass" at 9 $M_\odot$ because this is the minimum initial stellar mass (at Solar metallicity) predicted to produce a SNe in the Sukhbold et al. (2016) tables on which we rely. We draw the number of high-mass stars, $N_{\rm high}$, from a Poisson distribution with an expectation value $f_{*,{\rm high}} \epsilon_* M_{\rm cell} / \langle m_{*,{\rm high}} \rangle$, where $f_{*,{\rm high}} = 0.20055$ is the fraction of stellar mass contained in stars with masses > 9 $M_\odot$, and $\langle m_* \rangle = 19.39$ $M_\odot$ is the mean mass per star for stars with masses > 9 $M_\odot$; the numerical values given here are for our chosen IMF, but can be adjusted for different IMFs as desired. We then draw the masses of individual high-mass star particles from the high-mass end of the IMF, and set the mass of the low-mass star particle[4] to $\epsilon_* M_{\rm cell} (1 - f_{*,{\rm high}})$.

[3] We note here that Eq. 23 implicitly assumes that $\epsilon_{\rm ff}$ describes the star formation rate averaged over Δt. In principle one could also calculate the mass of stars formed by treating $\epsilon_{\rm ff}$ as describing an instantaneous star formation rate, and integrating the ordinary differential equation describing the gas mass, $dM_{\rm cell}/dt = -\epsilon_{\rm ff} M_{\rm cell}/t_{\rm ff}$, for a time Δt; one can then set $M_* = M_{\rm cell}(0) - M_{\rm cell}(\Delta t)$. For fixed $t_{\rm ff}$, this would yield $M_* = M_{\rm cell}(1 - e^{-\epsilon_{\rm ff} \Delta t / t_{\rm ff}})$, corresponding to $P = 1 - e^{-\epsilon_{\rm ff} \Delta t / t_{\rm ff}}$, the expression often adopted in cosmological star formation prescriptions. This choice and Eq. 23 obviously yield very similar results if $\Delta t \ll t_{\rm ff}/\epsilon_{\rm ff}$, but the difference is non-negligible for larger time steps. That said, there is no good reason to assume that $t_{\rm ff}$ remains constant over such larger time steps, so there is no reason to believe that the exponential form is more accurate. In practice, however, the difference between the exponential form of $\langle M_* \rangle$ and Eq. 23 is unimportant, both because the CFL condition almost always guarantees that $\Delta t \lesssim t_{\rm ff} \ll t_{\rm ff}/\epsilon_{\rm ff}$, and because if $\Delta t \gg t_{\rm ff}$ then we should not expect to obtain precisely the correct star formation rate from *any* prescription based on an operator-split treatment of star formation and hydrodynamics.

[4] We reiterate that this "low-mass" particle represents a collection of stars inhabiting the low-mass end of the IMF, unlike the "high-mass" particle(s) which represent a single high-mass (> 9 $M_\odot$) star.

It is important to note that in this implementation the mass of stars we produce will *not* exactly equal the gas mass converted to stars on a cell-by-cell basis; instead the gas mass converted to stars is equal to the *expectation value* of the stellar mass produced. Thus our star formation prescription guarantees mass conservation only in a statistical sense, not to machine precision. While this behaviour might seem undesirable, the alternative is worse: as discussed in Krumholz et al. (2015, see also Haas & Anders 2010), there is no way to sample the distribution of stellar masses produced in events of finite total mass that simultaneously respects exact mass conservation within each event (i.e., that guarantees that the mass of stars drawn is exactly equal to the mass assigned to that event) and that also produces an IMF integrated over all star formation events that is independent of the mass assigned to each event. Nor is this a small effect: if the typical mass of a star-forming event is 100 $M_\odot$, then over many such events a prescription that requires the masses of stars formed in each event total exactly 100 $M_\odot$ could easily produce fewer than half as many supernovae as one would have obtained if the event size were 1000 or 10^4 $M_\odot$. Thus if we were to enforce exact mass conservation, the result would be that simulations carried out at different resolutions (which determines the characteristic mass of cells that can form stars) would effectively have very different IMFs. Given a choice between this and reducing mass conservation from exact to statistical, we choose the latter as the lesser evil.

Once we have drawn high-mass star particles, we follow Andersson et al. (2021) and Steinwandel et al. (2023) by giving them initial velocities equal to the velocity of the cell which spawned them plus an additional component whose direction is random and whose magnitude is drawn from a power-law distribution $\propto v^{-1.8}$ with $v \in [3, 385]$ km s^{-1}. This distribution of velocities is motivated by simulations of dynamical ejections from young clusters (Oh & Kroupa 2016), and ensures creation of runaway stars with an incidence rate of $\sim 14\%$, matching empirical estimates – see Andersson et al. (2021) and Steinwandel et al. (2023) for full details. To ensure no net momentum is added just by star particle creation, we set the velocity of the low-mass star particle such that, in the frame comoving with the cell that spawns the particles, its momentum is equal and opposite to the total high-mass momentum.

4. TESTS

We now present a series of tests of our implementations of sink and supernova feedback particles. While the isothermal sphere test (Sec. 4.1), the Bondi/Bondi-Hoyle accretion tests (Sec. 4.2, Sec. 4.3), and single SN feedback test (Sec. 4.4) are not new, they demonstrate the correctness of multi-rank communication, as in all the tests we use multiple MPI ranks and place the particle at block boundaries.

4.1. *Self-similar collapse of an isothermal sphere*

Our first test of sink accretion is the classic self-similar collapse of a singular isotehrmal sphere, which represents the inside-out gravitational collapse of a cold, nearly isothermal molecular cloud core forming a central protostar and infalling envelope (Shu 1977). In this problem, the initial state is a spherically symmetric, isothermal

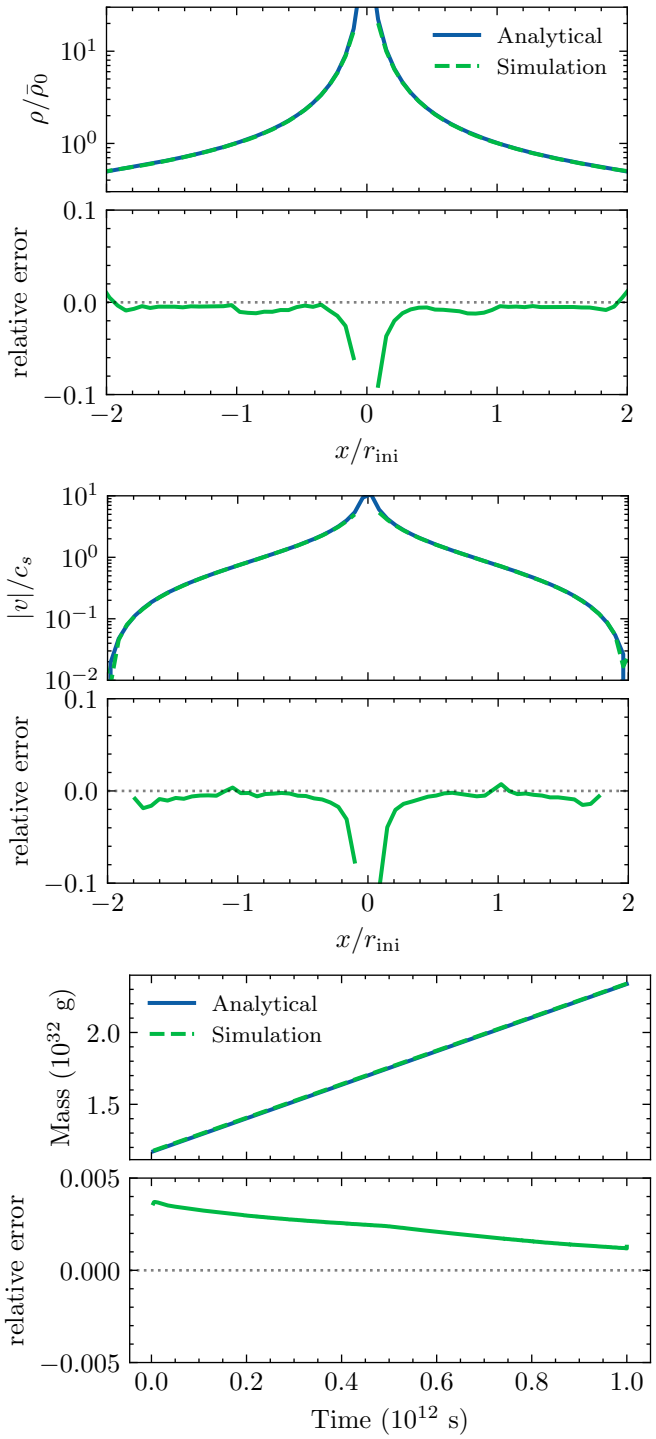

FIG. 2.—: Isothermal sphere test. The top two panels show the mean density and radial velocity as a function of radius in the simulation (dashed green line) compared to the analytic solution (blue solid line). The lower panel shows the sink particle mass versus time, again comparing the simulation and exact analytic solutions.

gas cloud in unstable hydrostatic equilibrium, and the subsequent evolution describes Shu's famous "expansion-wave" solution: the outer part retains a static singular isothermal equilibrium profile before the expansion wave arrives, and the inner part is free-falling toward the center, with the entire structure evolving self-similarly.

The density profile for the outer, hydrostatic part of a generalized singular isothermal sphere is

$$\rho(r) = \frac{Ac_s^2}{4\pi G r^2}. \tag{25}$$

where c_s is the isothermal sound speed and A is a dimensionless parameter that measure the degree of overdensity relative to the marginally stable equilibrium. The special case $A = 2$ corresponds to an isothermal sphere that is in unstable hydrostatic equilibrium everywhere outside the expansion wave.

We consider a sphere of isothermal gas with sound speed $c_s = 0.2$ km/s. To avoid the singular initial configuration of the Shu solution, we do not start from the $t = 0$ state with a formal $\rho \propto r^{-2}$ divergence at the origin. Instead, we initialize our density and velocity profile to the analytic solution at a finite time $t_0 = 10^{12}$ s, by which point a central point mass has already formed an the expansion wave has propagated outward; the choice of initial time is essentially arbitrary, since the analytic solution is self-similar – the structure is the same at all times > 0, and the choice of t_0 simply serves to set the initial position of the expansion wave. For $x \in [0, 1]$, where $x = r/c_s t$ is the similarity variable, we use the asymptotic solution of the expansion-wave collapse problem, taking the case $A = 2.0001$ as an approximation to the $A = 2^+$ limit; for $x > 1$, we adopt the isothermal solution $v = 0$, $\alpha = 2/x^2$. We run the simulation in cgs units; to convert the dimensionless variables in the initial condition to dimensional quantities, we use the following definitions of the unit length, mass, velocity, and density:

$$u_l = c_s t_0 = 2 \times 10^{16} \text{ cm} \tag{26}$$

$$u_m = c_s^3 t_0 / G = 1.198627571 \times 10^{32} \text{ g} \tag{27}$$

$$u_v = c_s = 0.2 \text{ km/s} \tag{28}$$

$$u_\rho = \frac{1}{4\pi G t_0^2} = 1.192296893 \times 10^{-18} \text{ g cm}^{-3}. \tag{29}$$

Initially, we place a sink particle with mass $m_0 = 0.975\ u_m = 1.168661882 \times 10^{32}$ g at the domain centre. The computational domain is a box with size $8u_l$, and we use a base grid of 128^3 cells with two refinement levels to achieve a maximum resolution $\Delta x = u_l/64$ within $2u_l$ of the origin.

In the top two panels of Fig. 2, we plot the density and velocity profiles at $t = 10^{12}$ s, enough time for the expansion wave to have expanded in radius by a factor of two; we plot the sink particle mass versus time in the lower panel. The numerical results agree with the analytic solution extremely well: the relative error in both density and velocity remains within 2% across radii inside the expansion wave front, with larger deviations only in the innermost region where spatial resolution is poor. The temporal evolution of the sink particle mass is in excellent agreement with theoretical expectations.

4.2. *Bondi accretion*

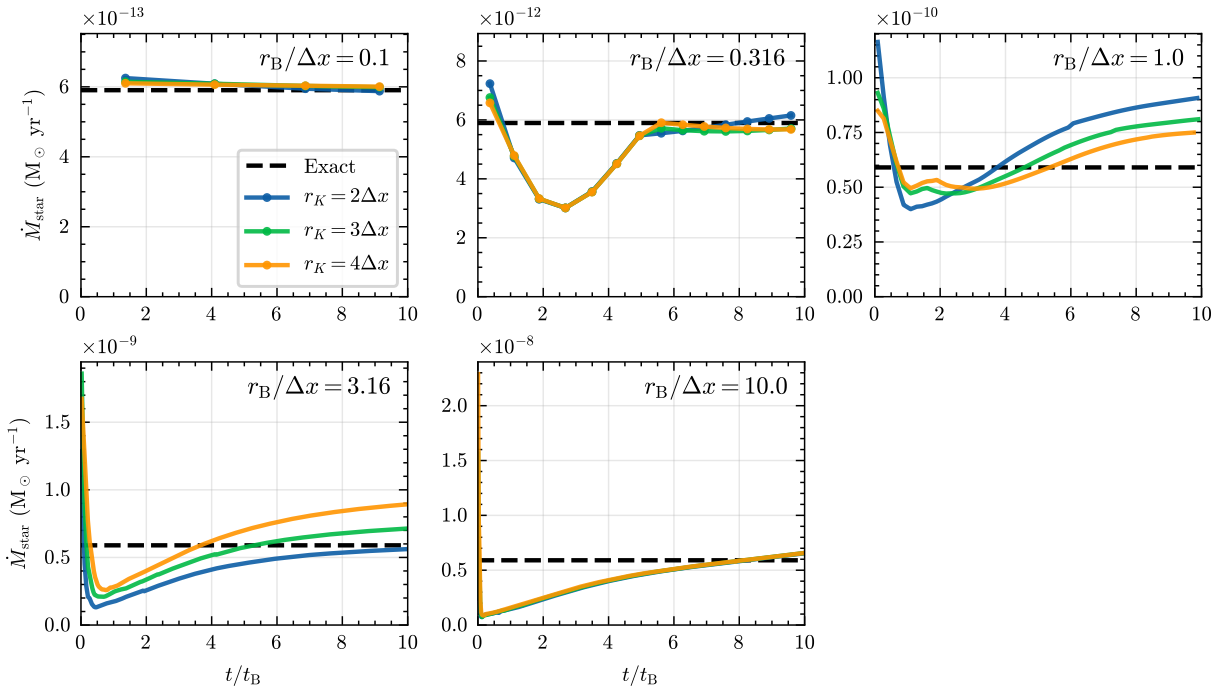

FIG. 3.—: Comparison of simulated accretion rates over time with theoretical values for the Bondi accretion tests presented in Sec. 4.2. The ratios of the Bondi radius to spatial resolution are, from left to right and top to bottom, 0.1, 0.32, 1.0, 3.16, and 10. The dashed horizontal line denotes the theoretical accretion rate, and the solid lines show the accretion rate measured in the simulations. The horizontal axis gives time in units of the Bondi time.

Our second test of the sink particle algorithm is against the Bondi (1952) solution for accretion onto a point mass by an infinite, uniform medium with negligible self-gravity. Since we have built the Bondi accretion formula into our method (Eq. 8), we expect it to reproduce the exact analytic accretion rate in the unresolved regime $r_B \ll \Delta x$, where r_B is the Bondi radius given by Eq. 7 with $v_\infty = 0$. In the opposite regime, $r_B \gg \Delta x$, the accretion rate should depend primarily on the hydrodynamics and gravity solver, so most reasonable schemes for accretion by a point mass should yield an accurate solution, but the marginally resolved case ($r_B \sim \Delta x$) is more challenging. To assess the performance of our scheme, we conduct a suite of simulations with largely identical initial conditions but a range of values for $r_B/\Delta x$, following the setup of Krumholz et al. (2004). We initialize a gaseous sphere of radius $r_{\rm sph} = 1.21 \times 10^{19}$ cm at a constant temperature of 10 K with mean molecular weight $\mu = 2.33\, m_p$ with a sink particle of mass m_s at its centre. The gas sphere is centred in a box of size 16 $r_{\rm sph}$. The mesh comprises a coarse level of 128 cells, with level-1 refinement covering a box of size 8 $r_{\rm sph}$, and level-2 refinement covering 4 $r_{\rm sph}$. With $m_s = 1\ M_\odot$, this configuration yields $\Delta x = r_B$ at the finest level. We then vary m_s so that $r_B/\Delta x$ ranges from 0.1 to 10 in steps of 0.5 dex. In all cases we initialize the density and velocity profiles of the sphere to the analytic Bondi solution, which is given implicitly by the solutions to the system of coupled ordinary differential equations

$$x^2 \alpha v = \lambda \tag{30}$$

$$\frac{v^2}{2} + \ln(\alpha) - \frac{1}{x} = 0, \tag{31}$$

where $x \equiv r/r_B$, $v \equiv |u|/c_\infty$, $\alpha \equiv \rho/\rho_\infty$, and λ is the mass accretion rate in units of the fiducial mass flux $\rho_\infty c_\infty$ through area $4\pi r_B^2$,

$$\lambda \equiv \frac{\dot{M}}{4\pi\rho_\infty (GM)^2/c_\infty^3}. \tag{32}$$

We allow the simulation to evolve until the accretion rate reaches a steady state; the natural unit of time for this system is $t_B = r_B/c_s$, and the time to reach equilibrum is generally $\sim 5\ t_B$.

The results are summarized in Fig. 3. We obtain high accuracy (error $< 2\%$) in the steady-state accretion rate when the Bondi radius r_B is 10 times smaller than the cell spacing Δx. When r_B and Δx are comparable, the error increases to $\sim 20\%$. As argued by Krumholz et al. (2004), the larger error for $r_B \approx \Delta x$ is expected because the transition from subsonic to supersonic accretion flow is poorly resolved, leading to substantial errors in the density and velocity in cells near the sink particle. For $r_B = 10\Delta x$ the error at $t = 10 t_B$ drops to a few percent.

We also use this test to explore kernel sizes from $r_K = 2\Delta x$ to $4\Delta x$ and find that the choice of r_K makes only minor differences when $r_B/\Delta x \sim 1$; in particular, at $r_B/\Delta x = 1$, larger r_K yields slightly improved accuracy. However, for $r_B/\Delta x \gg 1$ or $r_B/\Delta x \ll 1$, the difference is negligible. This behaviour is consistent with the arguments above regarding the scheme's accuracy in the well-resolved and unresolved limits.

4.3. *Bondi-Hoyle accretion*

We evaluate our sink particle scheme's ability to model accretion onto a moving particle by simulating the classical Bondi-Hoyle accretion problem. For this test, we ini-

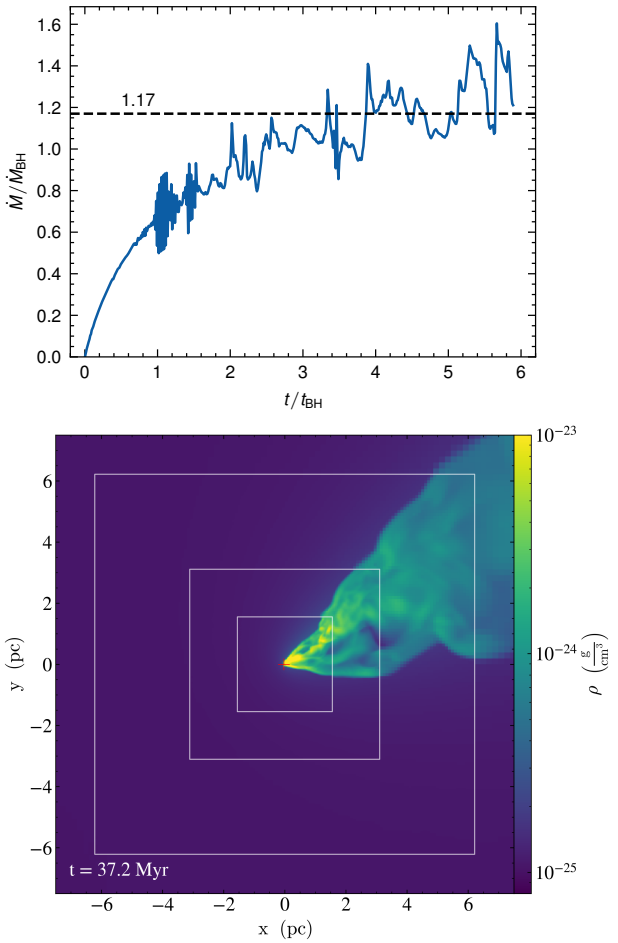

FIG. 4.—: Top: Accretion rate versus time for the Bondi-Hoyle accretion test described in Sec. 4.3. The accretion rate is normalized to the Bondi-Hoyle accretion rate, and the time is normalized to Bondi-Hoyle time. Bottom: Snapshot of the density field in the xy-plane at $t \approx 6t_{\rm BH}$. White squares indicate the borders of the AMR levels.

tialize the system in the regime $r_{\rm BH}/\Delta x \gg 1$. We use the same grid hierarchy as in Sec. 4.2, but reduce the domain size – and thus the grid spacing – by a factor of 5. The gas has a uniform density $\rho = 10^{-25}$ g cm^{-3} and an initial velocity with Mach number $\mathcal{M} = 3$. The velocity lies in the $x-y$ plane, forming a 30° angle with the x-axis. We place a 100 $M_\odot$ sink particle at the domain center. Using Eq. 7, we obtain the Bondi-Hoyle radius $r_{\rm BH} = 0.1\ r_{\rm B}$, which in turn yields $r_{\rm BH}/\Delta x = 100 \times 0.1 \times 5 = 50$ (see calculations in Sec. 4.2).

Fig. 4 shows the accretion rate versus time. It takes several Bondi-Hoyle times, $t_{\rm BH} = r_{\rm BH}/v_\infty$, for the system to reach a quasi-steady value. As discussed by Krumholz et al. (2004), this setup is in a regime where the interpolation formula is not particularly accurate. Ruffert (1996) and Krumholz et al. (2004) performed similar simulations and found a steady-state accretion rate close to $\dot{M} \approx 1.17\dot{M}_{\rm BH}$, where $\dot{M}_{\rm BH}$ is defined in Eq. 8. Our results agree well with these studies: the difference in the steady-state mean accretion rate is smaller than the intrinsic fluctuations. Both works also reported temporal variability in the accretion rate and flow morphology, and suggested Rayleigh-Taylor and Kelvin-Helmholtz instabilities as plausible causes.

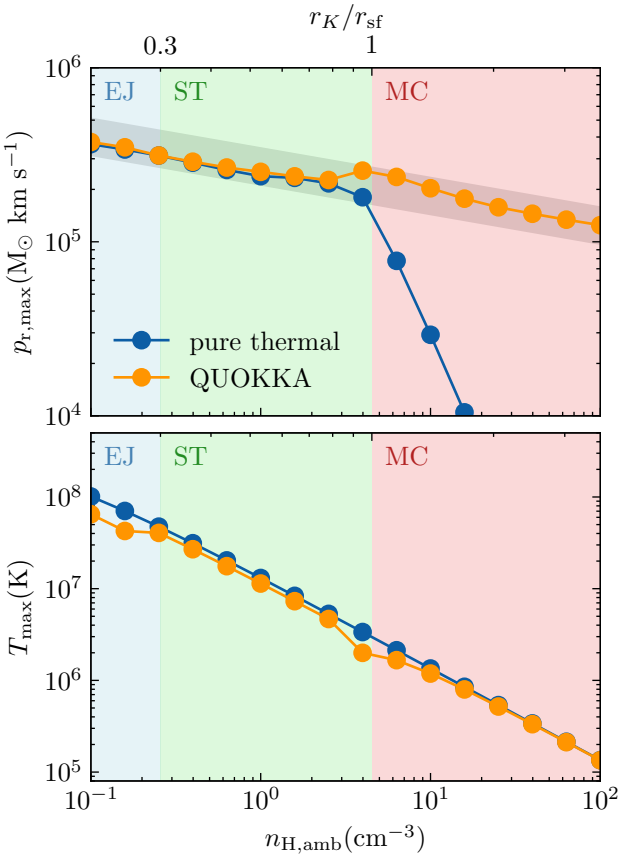

FIG. 5.—: Tests of SN feedback prescriptions using simulations of radiative SNR evolution in a uniform medium with spatial resolution $\Delta x = 4$ pc, both using our fiducial prescription (orange) and a scheme that deposits thermal energy only (blue). The top panel shows the final radial momentum, while the bottom shows the maximum temperature attained during the simulation, both shown as functions of the ambient hydrogen number density. The grey stripe in the upper panel indicates the empirical $p_{\rm r,max} - n_{\rm H,amb}$ relation with 25% tolerance. The upper axis shows the ratio of kernel radius ($= 3\Delta x$) to the shell-formation radius, and the background shading indicates which case of our fiducial SN deposition scheme is used for each simulation given that ratio.

4.4. *Single SN feedback*

We assess the accuracy of our SN feedback implementation by simulating SNR evolution with radiative cooling in a uniform medium at fixed spatial resolution $\Delta x = 4$ pc in a domain of size 256 pc and no adaptivity, following Kim & Ostriker (2017). We vary the ambient density from $n_{\rm amb} = 0.1$ cm^{-3} to 100 cm^{-3}. We initialize each simulation with a SN particle that explodes at $t = 0$, and we evolve the problem to the point where the total radial momentum in the domain, defined as $p_{\rm r} = \int \rho \mathbf{v} \cdot \hat{\mathbf{r}}\, dV$, reaches steady-state. Fig. 5 presents the final radial momentum (top) and the maximum temperature achieved in any cell over the full simulation time (bottom) as a function of $n_{\rm amb}$; the corresponding ratios of $r_K = 3\Delta x$ to shell-formation radius $r_{\rm sf} = 22.6$ pc$(n_{\rm amb}/{\rm cm}^{-3})^{-0.42}$ are indicated on the top axis. Results for our QUOKKA

prescription are shown in orange. For comparison, we also show the result of the "pure thermal" model in blue, which injects thermal energy at all resolutions.

In the EJ regime, where $\mathcal{R}_M \leq 0.027$ (roughly $r_K/r_{\rm sf} \leq 0.3$), our fiducial model applies pure kinetic energy deposition. In this regime the ST stage is fully resolved, and Kim & Ostriker (2015) find that both thermal and kinetic deposition recover the correct final radial momentum. Consistent with this expectation, our fiducial model coincides with the pure-thermal model in the top panel. The maximum temperature reached with kinetic deposition is only slightly lower than with pure-thermal deposition because in kinetic mode the gas is shock-heated shortly after SN ejection.

As the Sedov-Taylor stage becomes marginally resolved, $0.3 < r_K/r_{\rm sf} < 1$, the prescriptions diverge at the level up to a few tens of percent; in the unresolved regime ($r_K/r_{\rm sf} > 1$), the discrepancies are even larger. In these regimes our fiducial model (orange) remains within 25% of the empirical $p_{\rm r,max} - n_{\rm H,amb}$ relation (shaded area), demonstrating the the scheme is robust. By contrast, the pure-thermal model deviates substantially from the empirical relation owing to overcooling, consistent with previous studies (e.g. Kim & Ostriker 2015, 2017). Interestingly, while still incorrect, our pure thermal calculations yield a curve of $p_{\rm r,max}$ versus $n_{\rm H,amb}$ that lies closer to the empirical relation than reported by Kim & Ostriker (2017), which we attribute to improvements in our cooling solver (see Sec. 3.2.1.0).

4.5. *Multiple SN feedback*

One advantage of our algorithm with a limiter is its robust treatment of multiple SNe depositing into a single cell. To test numerical convergence in overlapping SNR regions, we simulate the evolution of two SNe exploding simultaneously at $t = 0$. We consider two ambient densities, 0.1 cm^{-3} and 10 cm^{-3}, corresponding to shell-formation radii of 59 pc and 8.6 pc, respectively. We place the exploding particles symmetrically about the domain centre, separated by $d_{\rm sep} = 30$ pc, which lies between the two values of $r_{\rm sf}$: for the two lower and higher initial densities, $r_{\rm sf}/d_{\rm sep} \approx 2.0$ and 0.3, respectively. Consequently, for the higher ambient density the SNRs should form shells before they begin to interact, while for the lower ambient density interaction will begin during the Sedov-Taylor phase.

We simulate each of our two configurations at spatial resolutions $\Delta x = 2, 4, 8$ and 16 pc in a domain of size 256 pc, again with no adaptivity. Given the particle separation of 30 pc and our fiducial kernel size of $r_K = 3\Delta x$, the deposition kernels of the two particles are well-separated in the higher-resolution runs, but overlap in the lower-resolution runs. Comparing the low- and high-resolution cases therefore allows us to test the performance of our scheme for handling multiple SNe in cases where the deposition regions for those SNe overlap, in both the well-resolved, $r_{\rm sf}/r_K \gg 1$, and poorly-resolved, $r_{\rm sf}/r_K \lesssim 1$, regimes. We summarise the full set of configurations we have tested in Table 1.

Fig. 6 and Fig. 7 illustrate the momentum-density evolution across resolutions and initial densities. Consistent with our expectations, we see shell formation prior to interaction between the two SNRs at higher density and when the resolution is sufficient for the deposition kernels not to overlap, while for lower density the two SNRs merge before forming shells.

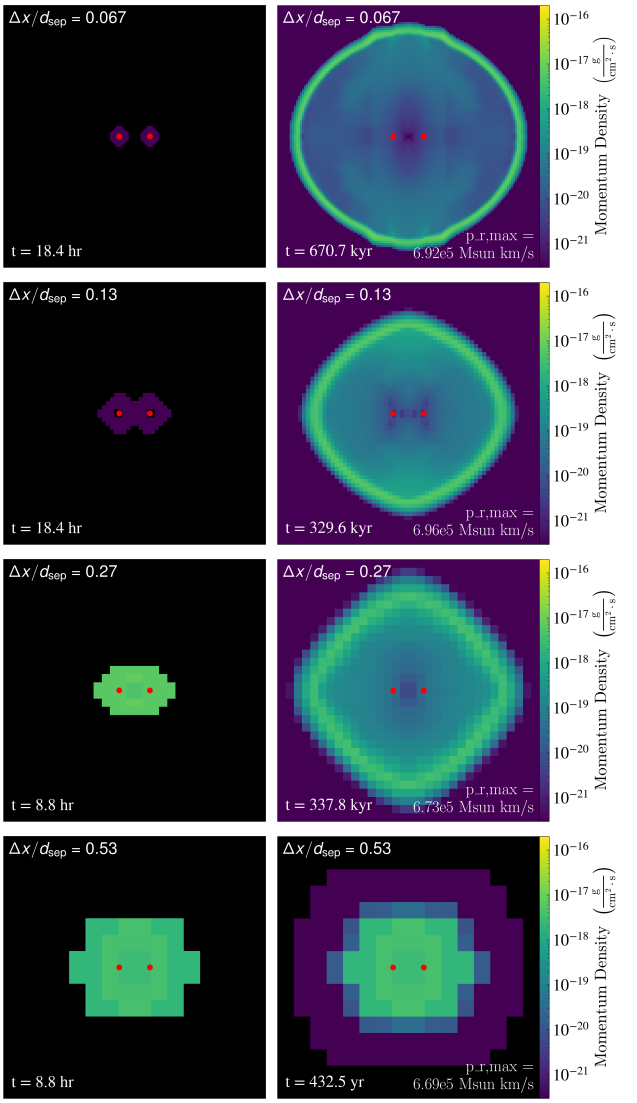

FIG. 6.—: Slices showing radial momentum density in the double-SN test with $r_{\rm sf}/d_{\rm sep} \approx 2.0$ (corresponding to ambient density $n_{\rm H,amb} = 0.1$ cm^{-3}), so shell formation occurs only *after* the two SNRs begin to interact. Red circles mark the locations of the two SNe. The left column shows the configuration after the first time step, immediately after SN deposition, while the right column shows the state at the end of the simulation when the radial momentum has ceased increasing, at the time indicated. Spatial resolutions are 2 pc, 4 pc, 8 pc, and 16 pc, from top to bottom. Our numerical method demonstrates converging results for the terminal radial momentum (see labels at bottom-right corner of the second column) as we vary the spatial resolution.

Because there is no empirical $p_{\rm r,snr} - n_{\rm H,amb}$ relation for the dual-SNR problem, we take the 2 pc-resolution run as the reference solution for each ambient density. We measure the terminal radial momentum in that simulation, and report the fractional deviation from that value in Table 1 as our error estimate. For $n_{\rm H,amb} = 10$ cm^{-3}, the reference solution lies in the regime where the ST phase is

$n_{\rm H,amb}$ [cm^{-3}]	$r_{\rm sf}$ [pc]	Δx [pc]	$\Delta x/d_{\rm sep}$	$2r_K/d_{\rm sep}$	$r_K/r_{\rm sf}$	$p_{\rm r,max}$ [M$_\odot$ km s^{-1}]	ϵ_p [%]
0.1	59	2	0.067	0.40	0.1	6.92×10^5	-
0.1	59	4	0.13	0.80	0.2	6.96×10^5	0.6
0.1	59	8	0.27	1.60	0.4	6.73×10^5	−2.7
0.1	59	16	0.53	3.20	0.8	6.69×10^5	−3.3
10	8.6	2	0.067	0.40	0.7	3.00×10^5	-
10	8.6	4	0.13	0.80	1.4	3.42×10^5	14
10	8.6	8	0.27	1.60	2.8	3.05×10^5	1.7
10	8.6	16	0.53	3.20	5.6	3.59×10^5	20

TABLE 1: Parameters and results of the numerical convergence tests for the evolution of two SNRs separated by $d_{\rm dep} = 30$ pc. Symbols: $n_{\rm H,amb}$ – ambient gas number density; $r_{\rm sf}$ – shell-formation radius, Δx – spatial resolution; $p_{\rm r,max}$ – total terminal radial momentum; ϵ_p – error relative to the value at the best resolution for a given $n_{\rm H,amb}$. In all tests, the SN separation is 30 pc. The strength of SN feedback, characterized by $p_{\rm r,max}$, matches the high-resolution case to within 20 % for all cases and resolutions.

Level	Δx [pc]	$N_{\rm cell}^{1/3}$	$V^{1/3}$ [kpc]
0	1171.9	1024	1200
1	585.9	1024	600
2	293.0	1024	300
3	146.5	1024	150
4	73.2	1024	75
5	36.6	1342	49
6	18.3	1886	35
7	9.2	2954	27
8	4.6	4721	22

TABLE 2: Parameters of the weak-scaling benchmark at the highest node count of 1024. For each AMR level we list the cell size Δx, the cube root of the total cell count $N_{\rm cell}^{1/3}$, and the cube root of the simulated volume $V^{1/3}$. The numbers given are for the highest node count of 1024; for smaller node counts the volume V remains unchanged, but the cell size Δx increases in factor of 2 steps, and the cell and particle counts decrease in factor of 2^3 steps. The total number of particles in the simulation is $1.088 \times 10^9 = 1029^3$, and about half of them are star particles at the finest level.

resolved and the initial injection regions do not overlap. For $n_{\rm H,amb} = 0.1$ cm^{-3}, the shells of the two SNRs overlap, but the reference solution still resolves the ST phase and avoids overlap at the moment of injection. Across all resolutions and both ambient densities, the terminal radial momentum $p_{\rm r,max}$ matches the high-resolution result to within 20%, demonstrating the effectiveness of the limiter.

4.6. *Parallel performance*

We evaluate the parallel performance of the Quokka particle module using weak-scaling benchmarks on the Frontier supercomputer at Oak Ridge National Laboratory. Each node of this system has 8 AMD MI-250X logical GPUs. In a weak-scaling experiment the total problem size is increased in proportion to the number of compute nodes so that the workload per node remains approximately fixed; ideal weak scaling therefore corresponds to constant wall-clock time per step as the node count increases.

We consider an isolated Milky Way analogue initial condition from the AGORA project (Kim et al. 2016), a standardised equilibrium disk galaxy model comprising a gas disk, a stellar disk and bulge, and a dark matter halo with masses $M_{\rm gas} \approx 9 \times 10^9$ M$_\odot$, $M_{*,\rm disk+bulge} \approx 4 \times 10^{10}$ M$_\odot$, and $M_{\rm halo} \approx 10^{12}$ M$_\odot$. The stellar and dark matter components are represented by live collisionless particles that interact gravitationally with the gas and with one another, and are initialised in equilibrium using a numerical solution of the Jeans equations.

This benchmark uses the full physics stack adopted in our production galaxy simulations: piecewise parabolic method hydrodynamics with an RK2 update that is second-order accurate in time and space, multigrid self-gravity, radiative cooling using a GPU Runge-Kutta ODE integrator with the Grackle tabulated cooling and heating rates (Smith et al. 2017) with temperature evolution integrated to a relative tolerance of 10^{-6}, and the star-formation and supernova feedback prescriptions described in Sec. 3. We enable AMR and construct a sequence of problem sizes that preserves the refinement pattern while increasing the total problem size with node count.

For this test all runs use the same nine-level AMR hierarchy (levels 0–8). We generate the 2-, 16-, 256-, and 1024-node cases (corresponding to 8 to 8192 GPUs) by resampling a checkpoint so that the refinement geometry is preserved but the resolution is increased uniformly at all AMR levels. Across this sequence, the linear resolution increases by factors of 2, 4, and 8 relative to the two-node case (and the number of cells correspondingly increases by factors of 2^3, 4^3, and 8^3), yielding finest-level cell sizes of $\Delta x = 36.6$, 18.3, 9.2, and 4.6 pc, respectively. We similarly scale the particle load by increasing the particle number and decreasing the particle masses by factors of 2^3, 4^3, and 8^3 across the three resolution increases. We report counts for the numbers of cells on each AMR level for the 1024-node case in Table 2. The total number of particles in the simulation domain for this node count is 1.088×10^9; about half of these are star particles located in the disk that overlap with cells refined to the maximum level, and the other half are dark matter particles distributed among all levels.

To test the performance of the particle code, we measure the wall-clock time of the particle update per step, which we obtain from an inline profiler in our code that

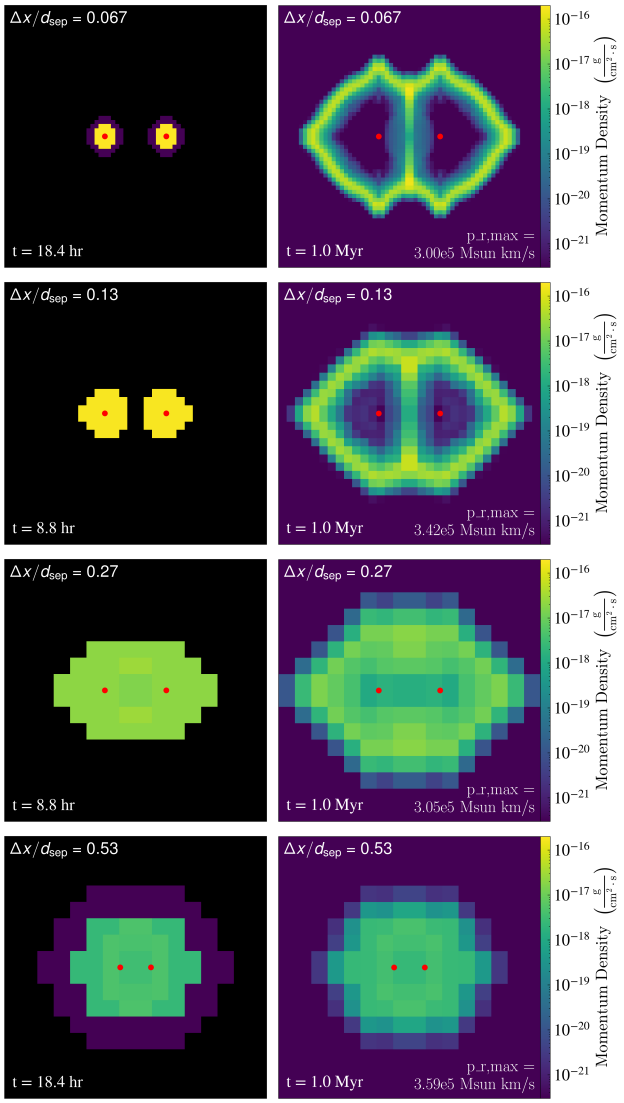

FIG. 7.—: Same as Fig. 6 but for a setup where $R_{\rm shell}/d_{\rm sep} = 0.3$ (corresponding to an ambient density $n_{\rm H,amb} = 10$ cm^{-3}). The full simulation box is 256 pc in length, but for improved visibility the first three rows are cropped to a 128 pc region. In this configuration, shell formation occurs *before* SNRs begin to interact. Our numerical method demonstrates convergent terminal radial momentum across spatial resolutions, with discrepancies within 20%.

isolates the particles routines, by averaging over 100 steps with I/O disabled. The results are shown in Fig. 8; the blue line shows the absolute update time per step, while the orange line shows the ratio of the time spent updating particles to the time spent updating hydrodynamics. Since in this test the numbers of particles and cells per GPU are fixed, perfect weak scaling corresponds to a horizontal line. We measure a weak scaling efficiency of 50% from 2 to 1024 nodes for the particle code, but even at the largest node count the particle update requires less than 10% of the hydrodynamic update time. This good scaling is achieved because the star-formation and feedback modules introduced in this work execute natively on GPUs without GPU–CPU transfers of memory, and

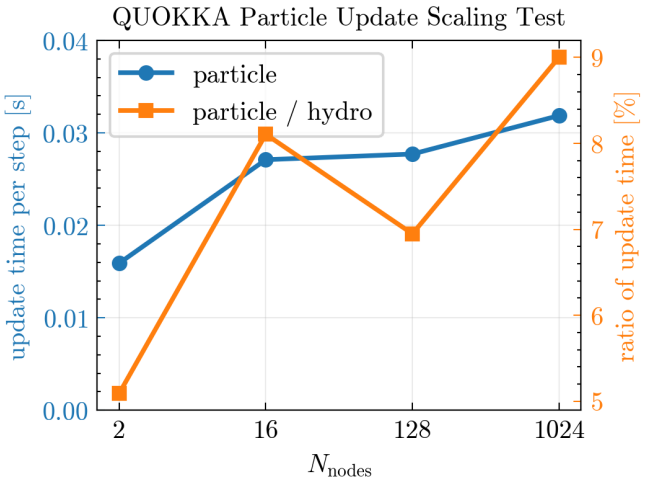

FIG. 8.—: Weak-scaling test of the particle update in QUOKKA for a Milky Way–like disk galaxy simulation. The left y axis shows the wall-clock time for the particle update per coarse step, and the right y axis shows this time expressed as a fraction of the hydrodynamic update time, both as a function of the number of Frontier nodes. We find 50% weak scaling efficiency from 2 to 1024 nodes.

because the particle module avoids all-to-all communication by restricting communication to the same nearest-neighbour ghost-zone exchanges required by the hydrodynamic update.

5. SUMMARY

We present a new algorithm, particle-mesh-particle, to provide a robust and efficient solution to the challenges inherent in particle–mesh and particle-particle interactions on modern GPU architectures. By avoiding expensive neighbour searches through the use of a deposition-buffer mesh with ghost zones and using efficient space-filling communication of these zones, our method achieves both high numerical accuracy and excellent performance. We provide implementations of two types of particle-mesh interaction – sink particles and SN feedback particles – built around this algorithm, and show that these implementations pass multiple tests, including accretion from a singular isothermal sphere, Bondi and Bondi-Hoyle flows, and expansion of SNRs driven by single and multiple SNe at a range of resolutions. To our knowledge, this approach enables, for the first time, simulations of star formation and stellar feedback on massively parallel, GPU-based architectures. The algorithm is also readily extensible to other feedback processes, such as stellar winds and protostellar outflows, and more generally to any linear or nonlinear particle-particle or particle-mesh interactions.

Our particle-mesh-particle strategy represents a paradigm shift by rethinking how particle contributions are aggregated and updated on GPUs. It requires only one extra communication per hydrodynamic step and reuses the communication pattern of the hydrodynamic update, which is already well-optimized for GPUs. We demonstrate the computational efficiency of this approach through weak-scaling benchmarks on the Frontier supercomputer using full-physics galaxy simulations with adaptive mesh refinement, self-gravity, radiative cooling, star formation, and supernova feedback. We find that

the particle module demonstrates ≈ 50% weak scaling from 2 to 1024 nodes (8192 GPUs). This performance demonstrates that our method scales comparably to pure hydrodynamics and can run efficiently on thousands of GPUs.

Despite these promising results, several limitations remain. First, the interaction distance is constrained by the number of hydrodynamic ghost cells. For complex stellar-feedback prescriptions that combine multiple feedback mechanisms (e.g., sink accretion, stellar wind, and outflows), additional ghost cells may be required to maintain accuracy. While increasing the number of ghost cells beyond what is required for the hydrodynamic solver can improve the accuracy of particle feedback, it incurs additional memory and compute costs. Second, although we have minimized inter-GPU communications to a single ghost-zone synchronization per hydrodynamic step, this constitutes an additional synchronization beyond that needed for the hydrodynamic update. This ghost zone update must be applied on levels that host particles and across all domains, regardless of the number of particles, and there is no obvious way to merge this synchronization with that of the hydrodynamic solver.

Future work will focus on extending Quokka's particle module to include additional feedback processes such as stellar winds and protostellar outflows. GPU-optimized methods for particle-particle-particle-mesh N-body calculations will also be explored.

ACKNOWLEDGEMENTS

CCH, AV, and MRK acknowledge support from the Australian Research Council through Laureate Fellowship FL220100020. This research was undertaken with the assistance of resources and services from the National Computational Infrastructure (NCI) and the Pawsey Supercomputing Centre, which are supported by the Australian Government, through award jh2. PSL acknowledges the supports by the NSFC through grant No. 1241101426 and the National Key R&D Program of China (No. 2022YFA1603101). The Quokka code is based on the AMReX module (Zhang et al. 2019).

DATA AVAILABILITY

The Quokka code, including the source code for all tests presented in this paper, is available from https://github.com/quokka-astro/quokka under an open-source license.

REFERENCES

Ahrens W., Demmel J., Nguyen H. D., 2020, ACM Trans. Math. Softw., 46
Almgren A. S., Bell J. B., Lijewski M. J., Lukić Z., Van Andel E., 2013, ApJ, 765, 39
Andersson E. P., Renaud F., Agertz O., 2021, MNRAS, 502, L29
Applebaum E., Brooks A. M., Quinn T. R., Christensen C. R., 2020, MNRAS, 492, 8
Armillotta L., Krumholz M. R., Di Teodoro E. M., McClure-Griffiths N. M., 2019, MNRAS, 490, 4401
Bate M. R., Bonnell I. A., Price N. M., 1995, MNRAS, 277, 362
Bleuler A., Teyssier R., 2014, Monthly Notices of the Royal Astronomical Society, 445, 4015
Blondin J. M., Wright E. B., Borkowski K. J., Reynolds S. P., 1998, ApJ, 500, 342
Bondi H., 1952, MNRAS, 112, 195
Chabrier G., 2005, in Corbelli E., Palla F., Zinnecker H., eds, Astrophysics and Space Science Library Vol. 327, The Initial Mass Function 50 Years Later. Springer, Dordrecht, pp 41–+
Choi J., Dotter A., Conroy C., Cantiello M., Paxton B., Johnson B. D., 2016, ApJ, 823, 102
Collange C., Defour D., Graillat S., Iakymchuk R., 2015, Parallel Computing, 49, 83
Crespo A. C., Dominguez J. M., Barreiro A., Gómez-Gesteira M., Rogers B. D., 2011, PLOS ONE, 6, 1
Cunningham A. J., Klein R. I., Krumholz M. R., McKee C. F., 2011, ApJ, 740, 107
Dale J. E., Bonnell I. A., Clarke C. J., Bate M. R., 2005, MNRAS, 358, 291
Dale J. E., Ercolano B., Clarke C. J., 2007, MNRAS, 382, 1759
Dale J. E., Ngoumou J., Ercolano B., Bonnell I. A., 2014, MNRAS, 442, 694
Dalla Vecchia C., Schaye J., 2012, MNRAS, 426, 140
Defour D., Collange C., 2015, in 2015 Federated Conference on Computer Science and Information Systems (FedCSIS). pp 721–728, doi:10.15439/2015F86
Di Matteo T., Springel V., Hernquist L., 2005, Nature, 433, 604
Dubois Y., Devriendt J., Slyz A., Teyssier R., 2010, MNRAS, 409, 985
El-Badry K., Ostriker E. C., Kim C.-G., Quataert E., Weisz D. R., 2019, MNRAS, 490, 1961
Fedeli L., et al., 2022, in SC22: International Conference for High Performance Computing, Networking, Storage and Analysis. pp 1–12, doi:10.1109/SC41404.2022.00008
Federrath C., Banerjee R., Clark P. C., Klessen R. S., 2010, ApJ, 713, 269
Fujimoto Y., Krumholz M. R., Tachibana S., 2018, MNRAS, 480, 4025
Gatto A., et al., 2017, MNRAS, 466, 1903
Gentry E. S., Krumholz M. R., Dekel A., Madau P., 2017, MNRAS, 465, 2471
Gentry E. S., Krumholz M. R., Madau P., Lupi A., 2019, MNRAS, 483, 3647
Gentry E. S., Madau P., Krumholz M. R., 2020, MNRAS, 492, 1243
Gong H., Ostriker E. C., 2013, The Astrophysical Journal Supplement Series, 204, 8
Guszejnov D., Hopkins P. F., Grudić M. Y., Krumholz M. R., Federrath C., 2018, MNRAS, 480, 182
Haas M. R., Anders P., 2010, A&A, 512, A79
Haugbølle T., Padoan P., Nordlund Å., 2018, ApJ, 854, 35
He C.-C., Wibking B. D., Krumholz M. R., 2024a, Monthly Notices of the Royal Astronomical Society, 531, 1228
He C.-C., Wibking B. D., Krumholz M. R., 2024b, Monthly Notices of the Royal Astronomical Society, 535, 3059
Hirashima K., Moriwaki K., Fujii M. S., Hirai Y., Saitoh T. R., Makino J., Steinwandel U. P., Ho S., 2025, ApJ, 987, 86
Hopkins P. F., et al., 2018, MNRAS, 477, 1578
Hoyle F., Lyttleton R. A., 1939, Proceedings of the Cambridge Philosophical Society, 35, 405
Hu Z., Krumholz M. R., Pokhrel R., Gutermuth R. A., 2022, MNRAS, 511, 1431
Iffrig O., Hennebelle P., 2015, A&A, 576, A95
Jappsen A. K., Klessen R. S., Larson R. B., Li Y., Mac Low M. M., 2005, A&A, 435, 611
Jeffreson S. M. R., Krumholz M. R., Fujimoto Y., Armillotta L., Keller B. W., Chevance M., Kruijssen J. M. D., 2021, MNRAS, 505, 3470
Katz N., 1992, ApJ, 391, 502
Kim C.-G., Ostriker E. C., 2015, The Astrophysical Journal, 802, 99
Kim C.-G., Ostriker E. C., 2017, The Astrophysical Journal, 846, 133
Kim J.-h., et al., 2016, The Astrophysical Journal, 833, 202
Kim C.-G., Ostriker E. C., Raileanu R., 2017, ApJ, 834, 25
Kimm T., Cen R., 2014, ApJ, 788, 121
Kratter K. M., Matzner C. D., Krumholz M. R., Klein R. I., 2010, ApJ, 708, 1585
Krumholz M. R., McKee C. F., Klein R. I., 2004, The Astrophysical Journal, 611, 399
Krumholz M. R., Klein R. I., McKee C. F., 2007, ApJ, 656, 959

Krumholz M. R., Klein R. I., McKee C. F., Offner S. S. R., Cunningham A. J., 2009, Science, 323, 754
Krumholz M. R., Fumagalli M., da Silva R. L., Rendahl T., Parra J., 2015, MNRAS, 452, 1447
Krumholz M. R., McKee C. F., Bland -Hawthorn J., 2019, ARA&A, 57, 227
Martizzi D., Faucher-Giguère C.-A., Quataert E., 2015, MNRAS, 450, 504
Myers A. T., McKee C. F., Cunningham A. J., Klein R. I., Krumholz M. R., 2013, ApJ, 766, 97
Offner S. S. R., Klein R. I., McKee C. F., Krumholz M. R., 2009, ApJ, 703, 131
Oh S., Kroupa P., 2016, A&A, 590, A107
Oppenheimer B. D., Davé R., 2006, MNRAS, 373, 1265
Pokhrel R., et al., 2021, ApJ, 912, L19
Ruffert M., 1994, ApJ, 427, 342
Ruffert M., 1996, A&A, 311, 817
Shu F. H., 1977, ApJ, 214, 488
Smith B. D., et al., 2017, MNRAS, 466, 2217
Springel V., 2005, Monthly Notices of the Royal Astronomical Society, 364, 1105
Springel V., Di Matteo T., Hernquist L., 2005, MNRAS, p. 623
Steinwandel U. P., Bryan G. L., Somerville R. S., Hayward C. C., Burkhart B., 2023, MNRAS, 526, 1408
Stone J. M., et al., 2024, arXiv e-prints, p. arXiv:2409.16053
Sukhbold T., Ertl T., Woosley S. E., Brown J. M., Janka H.-T., 2016, ApJ, 821, 38
Thornton K., Gaudlitz M., Janka H.-T., Steinmetz M., 1998, ApJ, 500, 95
Truelove J. K., Klein R. I., McKee C. F., Ii J. H. H., Howell L. H., Greenough J. A., 1997, The Astrophysical Journal, 489, L179
Wang P., Li Z.-Y., Abel T., Nakamura F., 2010, ApJ, 709, 27
Wibking B. D., Krumholz M. R., 2022, Monthly Notices of the Royal Astronomical Society, 512, 1430
Zhang W., et al., 2019, JOSS, 4, 1370

APPENDIX

BITWISE-REPRODUCIBILITY

We define software as bitwise-reproducible if it has the property that running it multiple times using the same inputs (including seeds for any random number generators) on the same hardware (i.e., using the same number and configuration of GPUs) is guaranteed to produce outputs that are bit-by-bit identical. This property is desirable for numerical software because it greatly aids in debugging and correctness testing – for example, if software is reproducible then we can verify that code changes that should not affect the outcomes of particular tests indeed do not affect them by performing bitwise differences between results produced before and after the change. However, bitwise reproducibility is very difficult to guarantee for GPU-based software, and strategies for maintaining it are an active area of research in the computer science community (e.g. Defour & Collange 2015; Ahrens et al. 2020).

Bitwise-reproducibility is challenging on GPU due to the interaction between the unpredictable order of execution of different GPU threads and the non-associativity of floating point arithmetic. The latter property means that, when we sum a list of floating point numbers, the sum is not guaranteed to be bitwise-invariant under a re-ordering of the list, while the former means that if the sum is being computed by multiple GPU threads, or if insertion of numbers into the list is carried out by multiple threads, we cannot guarantee that the list will be summed in the same order every time the program is run. This means that a naive program that runs reproducibility on a single CPU thread will not be reproducible when run on GPU if it involves any summations of lists of floating point numbers or analogous operations. The first two steps of the algorithm described in Sec. 2.2 both involve summations that, if not carried out carefully, would lead to non-reproducibility. We limit this problem as follows.

Particle to buffer mesh

The first step in our algorithm is to sum the contributions $\Delta\mathbf{U}^s_{ijk}$ for all particles s to the buffer mesh, which is exactly the type of list summation operation for which a naive implementation will not be reproducible on GPU. To improve reproducibility, we modify our deposition algorithm as follows:

1. In addition to creating a buffer to hold the amount of conserved quantity we are depositing in each cell $\Delta\mathbf{U}^{\rm buf}_{ijk}$, we also create a second temporary buffer $\mathbf{E}_{ijk}$ covering the same cells, which we will use to accumulate an upper bound on the error due to non-associativity.

2. When we update $\Delta\mathbf{U}^{\rm buf}_{ijk}$ by depositing the contribution from a given particle to it, we also update the value of $\mathbf{E}$ for the corresponding cell by carrying out the operations
$$\Delta\mathbf{U}^{\rm buf}_{ijk} := \Delta\mathbf{U}^{\rm buf}_{ijk} + \Delta\mathbf{U}^{s}_{ijk}$$
$$\mathbf{E}_{ijk} := \mathbf{E}_{ijk} + \varepsilon\left|\Delta\mathbf{U}^{\rm buf}_{ijk}\right|,$$
where ε is the machine precision. The amount we are adding to $\mathbf{E}$ represents an upper limit on the amount by which the sum could be changed by re-ordering the additions.

3. After we have completed summing over all particles s, we compute an upper bound on the relative error in each cell as $\mathbf{r}_{ijk} = \mathbf{E}_{ijk}/|\Delta\mathbf{U}^{\rm buf}_{ijk}|$.

4. We then round $\Delta\mathbf{U}^{\rm buf}_{ijk}$ in every cell by setting the least significant $N_{\rm bit}$ bits of the floating point mantissa to zero, with $N_{\rm bit} = \log_2\mathbf{r}_{ijk} + N_{\rm mantissa} + M$, where $N_{\rm mantissa} = 52$ is the number of bits in the mantissa and the extra $+M$ is a safety margin.

The choice of M requires some thought, and involves a trade-off between reproducibility and accuracy. Naively one might expect that choosing any $M \gtrsim 1$ would be sufficient to ensure reproducibility, since in this case the amount by which the results could differ between successive runs the program as a result of re-ordering of summations is much smaller than the amount by which we are rounding. However, rounding involves mapping all the bit values within a certain interval to a single value, and there is always a chance that, when the code is run twice, the result from the first run will be so close to the boundary between rounding intervals that the small amount by which the results differ in the second run pushes the sum into a different rounding interval. The probability of this happening is roughly the amount by which the results can change between runs divided by the size of the rounding interval, and thus the probability is $p \approx 2^{-M}$. We can therefore choose M large enough that, for a particular application, it is very unlikely that a non-reproducible

result will occur within a specified run duration. The value of M required for this purpose depends on the typical number of cells in which there are contributions from multiple particles and on the number of time steps for which the simulation is expected to run. For example the choice $M = 28$, which for cases where at most a few particles contribute to each cell is roughly equivalent to rounding the results from double precision to single precision, is sufficient to make a non-reproducible result improbable as long as the number of cell-advances during which multiple particles deposit to the same cell is $\lesssim 10^9$. In future work we will also explore the option of using superaccumulators, as suggested by Defour & Collange (2015) and Collange et al. (2015), to guarantee full bitwise-reproducibility for an arbitrary number of steps.

Summation of the buffer mesh

The second location in our algorithm where potential non-reproducibility arises is in the summation of the buffer mesh, where we add the buffer meshes that contain the contributions from all particles that can influence a given cell. As shown in Fig. 1, for most buffer cells there are at most two contributions, and thus there is no problem with associativity, but for cells that lie near the corner (in 2D) or edge (in 3D) of a cubical domain, there may be more than two buffers that contribute to that cell, in which case the potential for non-reproducibility appears.

To handle this problem we flag cells in the overlapping region over which the sum is taking place to which more than two different buffers contribute. We sum the majority cells with only two contributors immediately, and for the remaining cells we accumulate the contributions from all buffers in memory, sorting them based on the (x, y, z) position of the contributing grid, and then sum only once all the contributions have been transferred. Since the ordering of the summation is then uniquely determined and guaranteed to be the same every time the program is run, the result is bitwise-reproducible.

This paper was built using the Open Journal of Astrophysics LaTeX template. The OJA is a journal which provides fast and easy peer review for new papers in the `astro-ph` section of the arXiv, making the reviewing process simpler for authors and referees alike. Learn more at http://astro.theoj.org.