
On the effects of star formation on the evolution of galaxies in clusters: Development of subgrid models*

RAMESH BALAKRISHNAN
*National Center for Supercomputing Applications,
605 West Springfield Avenue, Champaign, IL 61820*
e-mail: bramesh@ncsa.uiuc.edu

ZARIJA LUKIĆ AND PAUL RICKER
*Department of Astronomy,
University of Illinois at Urbana-Champaign,
1002 West Green Street, Urbana, IL 61801*
e-mail: zlukic@astro.uiuc.edu, pmricker@uiuc.edu

Abstract

Cosmological simulations including the effects of star formation are a grand challenge problem, involving a range of length scales that makes them extremely expensive with respect to computational time and memory requirements. The range of length scales required for direct simulations of galaxy clusters within a cosmological context is $> 10^4$. If star formation within the member galaxies of a cluster is included, the spatial dynamic range needed for direct simulation jumps to $> 10^{11}$. The range of mass scales involved is even greater. One is, therefore, forced to model stellar evolution processes using particle Monte Carlo techniques, in which each computational particle represents a large number of actual stars and couples to the large-scale diffuse gasdynamics (which is simulated directly). Thus this problem bears some similarity to computational fluid dynamics studies of turbulent flows, in which the large eddy simulation (LES) technique is widely used.

The UIUC Cosmological Simulation Group, headed by Prof. Paul Ricker, and the first author are engaged in a collaborative project under the NCSA Strategic Applications Program (SAP) to develop and distribute a star formation module for the publicly available astrophysical simulation code FLASH. Test simulations using more than 10 million star and dark matter particles have been carried out on the NCSA distributed computing platform, Tungsten, and the Teragrid cluster, Mercury. The aim of these simulations was to study the effects of spatial and mass resolution. Our simulations, so far, have yielded results that encourage us to extend this line of work to include the aging and destruction of stars in order to draw more definite conclusions about the effects of star formation on the evolution of galaxies.

This document presents our accomplishments in the first phase of this project.

1 Introduction

Stars form from the diffuse gas, known as the interstellar medium (ISM), which pervades the disk of our Galaxy. The gas in the ISM exists in several different phases characterized by densities and temperatures that vary by several orders of magnitude. Among these phases, the extremely cold ($\sim 10 - 100$ K) and dense ($\sim 1 - 100$ particles cm^{-3}) giant molecular clouds (GMCs), consisting primarily of molecular hydrogen and dust, are the sites of most present-day star formation. The influence of gravity amplifies density perturbations in GMCs, leading to the formation of high-density regions known as cores. The deep interiors of these cores are shielded by dust from ionizing radiation produced by nearby stars, allowing them to reach very high densities by radiating away thermal energy and contracting under the influence of gravity. These contracting objects are known as protostars. The temperature and density of a protostar slowly rise until conditions are right for hydrogen fusion to begin within it, at which point it becomes a star. Fusion energy heats the interior of a star, balancing the effect of gravity with gas and radiation pressure and blowing away the outer parts of the molecular cloud core. This

*This paper is a record of the milestones reached during the first phase of the SAP project that began in October 2004. Owing to the NCSA re-organization, as a result of which the first author's appointment was split between PECCM and Consulting, most of the accomplishments during the first phase occurred after this SAP project was jump-started in the second week of March 2005.

balancing act between pressure and gravity continues throughout the life of the star until all of the star’s central nuclear fuel is spent. Depending on the initial mass of the star, the final outcome may be a white dwarf, neutron star, or black hole. However, a common feature of each end scenario is that the star gives back some of its mass and energy to the ISM in the form of hot gas, enriched by heavy elements and other byproducts of fusion. In the case of massive stars, where $M_\star \gtrsim 5 M_\odot$ ($1 M_\odot = \text{mass of the Sun} \approx 2 \times 10^{33} \text{ g}$), the star ends in a cataclysmic explosion which releases a significant amount of energy to the ISM.

Much remains to be learned regarding the process of star formation; it is one of the most active areas in modern astronomical research. However, the broad outline sketched above has proven remarkably successful in explaining many of the observed properties of our Galaxy and others. As a result simulations aiming to study star formation or its effects on galactic structure have focused on the evolution of GMCs. In such studies the ISM is modeled as a fluid. Stars are treated as non-colliding Lagrangian particles or as a collisionless fluid modeled by the Vlasov-Boltzmann equation. The dimensions of the computational domain in such simulations are approximately 1 kpc on each side. However, protostellar cores develop on much smaller scales, $\sim 10^{-5}$ pc, so their detailed evolution must be treated using subgrid models. Conditions for star formation are set by identifying regions of the computational domain where the flow converges, where the local cooling time is smaller than the local free fall time, or where the local gas density exceeds the threshold for gravitational instability (the so-called Jeans criterion). Star particles are created in the models wherever one or more of these criteria (depending on the model) are satisfied. Each star particle may represent a collection of stars. The effects of star formation on the gasdynamics of the ISM are modeled as sources and sinks of mass and energy in a manner analogous to the large eddy simulations (LES) of fluid turbulence.

While stars ($\sim 0.1 - 100 M_\odot$) are among the smallest objects, clusters of galaxies ($\sim 10^{13} - 10^{15} M_\odot$) are the most massive. Clusters consist primarily of three components: dark matter, which interacts gravitationally with itself and ordinary matter and makes up 80-85% of the total mass; the diffuse, hot ($> 10^7$ K) gas of the intracluster medium (ICM), which contributes 15-20% of the total mass; and galaxies, which contribute only about 1% to the total. Even though stars and gas do not contribute the majority of the mass in clusters, they do produce the radiation that allows us to observe clusters, so it is important to consider the processes governing their evolution when studying clusters via simulation. However, clusters span length scales ~ 1 Mpc, some 11 orders of magnitude larger than the length scales characteristic of star formation.

Subgrid modeling helps us to cope with the enormous dynamic range separating cluster length scales from stellar evolution scales. However, clusters form by gravitationally accumulating matter from regions several tens of Mpc on a side, so a gasdynamics simulation with a spatial resolution of 10 kpc already requires a spatial dynamic range > 1000 just to handle a single cluster properly. If ensembles of clusters are to be simulated — as we must do if we are to make accurate statistical comparisons with cluster survey data — we need resolution equivalent to a uniform mesh with 10^4 or 10^5 zones on a side. Fortunately, this level of resolution is required only in a small fraction of the total volume, so techniques such as adaptive mesh refinement (AMR) can be employed profitably. Even with AMR, however, ensemble simulations of galaxy cluster evolution are necessarily large. Given the processor and memory capacity of modern computers, parallel computers with hundreds to thousands of processors are essential for these simulations.

This report details Phase 1 of an NCSA Strategic Applications Program (SAP) project to develop subgrid models for star formation within a cosmological AMR code, FLASH. In § 2 we describe the basic equations solved by FLASH; this provides the context within which the subgrid model operates. § 3 surveys several phenomenological methods for treating star formation in simulations and describes the method we have chosen for Phase 1. In § 4 we present the code developed to implement the subgrid model together with results of test calculations and scaling studies on NCSA platforms. § 5 summarizes our results.

2 Equations and methods used in FLASH

To study galaxy cluster evolution, we use a simulation framework called FLASH that was originally developed by the ASCI Center for Astrophysical Thermonuclear Flashes at the University of Chicago. FLASH (Fryxell *et al.*, 2000) was designed to study X-ray bursts, novae, and Type Ia supernovae using AMR on large parallel computers. FLASH has been used to perform some of the largest AMR calculations ever attempted, earning its authors the 2000 Gordon Bell Prize (Calder *et al.*, 2000). It is nearly unique among astrophysical codes in having been extensively validated against laboratory experiments (Calder *et al.*, 2002). Since its beginning, FLASH has evolved into a general-purpose astrophysical simulation tool, including modules for gravity, magnetohydrodynamics, and particles.

For cosmological simulations with FLASH, calculations are assumed to take place in comoving coordinates $\mathbf{x} = \mathbf{r}/a$, where \mathbf{r} is a proper position vector and $a(t)$ is the time-dependent cosmological scale factor. The present epoch is defined to correspond to $a = 1$; in the following discussion we use $t = t_0$ to refer to the age of the Universe today. For simulations of isolated, collapsed objects such as individual galaxies, we set $a \equiv 1$ and $\dot{a} \equiv 0$.

The evolution of gas (baryons) is described by the Euler equations of hydrodynamics.* A special coordinate transformation

*In most astrophysical contexts diffusive effects are unimportant on the timescales of interest.

casts these equations into a relatively simple form when cosmological expansion is included. The gas velocity \mathbf{v} is taken to be the comoving peculiar velocity $\dot{\mathbf{x}}$. The comoving gas density, pressure, temperature, and internal energy density are defined to be

$$\begin{aligned}\rho &\equiv a^3 \tilde{\rho} \\ p &\equiv a \tilde{p} \\ T &\equiv \frac{\tilde{T}}{a^2} \\ \rho \epsilon &\equiv a \tilde{\rho} \tilde{\epsilon} .\end{aligned}\tag{1}$$

The quantities marked with a tilde ($\tilde{\rho}$ etc.) are the corresponding ‘‘proper’’ or physical quantities. Note that, in terms of comoving quantities, the equation of state has the same form as for the proper quantities in noncomoving coordinates. For example, the equation of state for an ideal gas with adiabatic index γ is

$$\rho \epsilon = \frac{p}{\gamma - 1} = \frac{\rho k T}{(\gamma - 1) \mu} ,\tag{2}$$

where μ is the average mass per particle and k is Boltzmann’s constant. With these definitions, the Euler equations for a perfect gas can be written in the form

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = \Gamma_M\tag{3}$$

$$\frac{\partial \rho \mathbf{v}}{\partial t} + \nabla \cdot (\rho \mathbf{v} \mathbf{v}) + \nabla p + 2 \frac{\dot{a}}{a} \rho \mathbf{v} + \rho \nabla \phi = \mathbf{\Gamma}_P\tag{4}$$

$$\frac{\partial \rho E}{\partial t} + \nabla \cdot [(\rho E + p) \mathbf{v}] + \frac{\dot{a}}{a} [(3\gamma - 1) \rho \epsilon + 2 \rho v^2] + \rho \mathbf{v} \cdot \nabla \phi = \Gamma_E .\tag{5}$$

Here E is the specific total energy, $\epsilon + \frac{1}{2} v^2$. Γ_M , $\mathbf{\Gamma}_P$, and Γ_E are local net mass, momentum, and energy source terms, respectively. The appropriate specification of these source terms is half of our subgrid modeling effort.

Since astrophysical flows are usually highly compressible, FLASH uses the piecewise-parabolic method (PPM) to solve the Euler equations (Colella & Woodward, 1984). PPM is optimized for flows that include sharp flow discontinuities such as shocks, resolving them with 1-2 mesh zones and avoiding the spurious oscillations and excessive dissipation that plague less sophisticated methods.

For collisionless species, including dark matter and stars, we use an N -body or particle representation. Each computational particle corresponds to a large number of physical particles. The i th computational particle is characterized by a comoving position \mathbf{x}_i , a comoving velocity \mathbf{v}_i , a mass m_i , and perhaps additional quantities such as a unique identifier (tag). The particles move in the same potential as the gas, obeying the equations

$$\begin{aligned}\frac{d\mathbf{x}_i}{dt} &= \mathbf{v}_i \\ \frac{d\mathbf{v}_i}{dt} &= -\nabla \phi - 2 \frac{\dot{a}}{a} \mathbf{v}_i\end{aligned}\tag{6}$$

The first term on the right-hand side of the second equation represents the gravitational force on the particles, while the second term represents the redshift effect due to cosmic expansion. Because star formation converts gas into collisionless stars, the second half of our subgrid modeling effort involves the specification of rules for creating and destroying particles intended to follow stellar populations.

The most expensive part of a collisionless particle solver is generally the computation of the force on the particles. Since gravity is a long-range force, a naive calculation of the force on N particles requires of order N^2 operations. For this reason several methods to speed up the force calculation have been developed. FLASH uses an N -body solver based on the particle-mesh method (Hockney & Eastwood, 1988). This method speeds force calculations by converting particle positions to densities on a grid, solving the Poisson equation on the grid, and finally interpolating the potential from the grid to obtain forces at particle locations. Since we already need to solve the Poisson equation on a grid for our hydrodynamical code, the particle-mesh method is ideally suited for integration with it. While the FLASH framework enables other methods such as tree-based methods to be implemented straightforwardly, we have chosen the particle-mesh method because of its simplicity and because the AMR capabilities of the code can be used to yield the correct smoothing length in low- and high-density regions without further changes (Knebe *et al.*, 2001).

The comoving potential ϕ in the above equations is the solution to the Poisson equation in the form

$$\nabla^2 \phi = \frac{4\pi G}{a^3} (\rho_{\text{tot}} - \bar{\rho}) ,\tag{7}$$

where ρ_{tot} is the comoving matter density (gas plus dark matter), $\bar{\rho}$ is the comoving mean matter density, and G is Newton's gravitational constant. The comoving mean matter density is defined in terms of the present-day critical density ρ_{crit} by

$$\bar{\rho} \equiv \Omega_{\text{m}} \rho_{\text{crit}} , \quad (8)$$

where Ω_{m} is the matter density parameter (whose value is specified as part of the cosmological model), and

$$\rho_{\text{crit}} \equiv \frac{3H_0^2}{8\pi G} . \quad (9)$$

Here H_0 is the Hubble constant, often written as

$$H_0 = 100h \text{ km s}^{-1} \text{ Mpc}^{-1} , \quad (10)$$

where h is measured to be about 0.7 (Madore & Freedman, 1998) The Hubble parameter $H(t)$ ($H_0 \equiv H(t_0)$) characterizes the rate of expansion of the Universe and is given by the Friedmann equation:

$$H^2(t) \equiv \left(\frac{\dot{a}}{a}\right)^2 = H_0^2 \left(\frac{\Omega_{\text{m}}}{a^3} + \frac{\Omega_{\text{r}}}{a^4} + \Omega_{\text{de}}(a) - \frac{\Omega_{\text{c}}}{a^2} \right) . \quad (11)$$

Here Ω_{r} is the present-day density parameter of radiation, and $\Omega_{\text{de}}(a)$ is the model-dependent density parameter of dark energy. Ω_{de} is thought to be roughly constant at the present day, and it is this lack of dilution as the Universe expands that is driving the expansion to accelerate. The present-day contribution of the overall spatial curvature of the Universe (thought to be zero) is given by

$$\Omega_{\text{c}} \equiv \Omega_{\text{m}} + \Omega_{\text{r}} + \Omega_{\text{de}} - 1 . \quad (12)$$

The Friedmann equation is solved using a standard ordinary differential equation solver.

At each time step we solve the Poisson equation using a multigrid elliptic solver (Brandt, 1977). Multigrid algorithms solve elliptic equations by accelerating the convergence of relaxation methods. Relaxation methods are easy to implement but converge very slowly. They accomplish the global coupling implied by an elliptic equation using a series of iterations that communicate information across the grid one zone at a time. Hence their convergence rate (fractional reduction in error per iteration) decreases with increasing grid size. The longest-wavelength components of the error require the most iterations to decrease to a given level. By iterating on a sequence of coarser grids, multigrid algorithms bring all wavelengths into convergence at the same rate. This works because long wavelengths on a fine mesh appear to be short wavelengths on a coarse mesh.

Our multigrid solver is based on a multilevel adaptive refinement scheme described by Martin & Cartwright (1996) and Martin (1998). Adaptive mesh refinement provides many benefits in conjunction with a multigrid solver. Where errors are unlikely to have short-wavelength components it makes sense to avoid using fine grids, thus reducing storage requirements and the cost of relaxations on fine levels. The AMR package manages the multilevel mesh data structures and can handle all parallel communication, allowing the multigrid solver to be implemented independently of such details. The AMR package supplies many of the basic functions required by multigrid algorithms in addition to the mesh data structure, including prolongation, restriction, and boundary condition updates. Therefore we use a mesh hierarchy defined by the AMR package.

To implement adaptive mesh refinement, FLASH uses a customized version of the block-structured AMR package PARAMESH (MacNeice *et al.*, 2000). Berger and co-workers (Berger & Olinger, 1984; Berger & Colella, 1989) pioneered block-structured algorithms, which use a hierarchy of logically Cartesian grids to cover the computational domain. This approach is flexible and memory-efficient, but the resulting code is complex and can be difficult to parallelize efficiently. Quirk (1991) and de Zeeuw & Powell (1993) implemented simplified versions which develop the hierarchy of nested grids by bisecting blocks in each coordinate direction and placing each block at a node of a tree. Blocks are distributed among processors using a Morton space-filling curve (Warren & Salmon, 1993). This is the approach on which PARAMESH is based (Figure 1). PARAMESH differs from other AMR libraries in that it was designed with much less abstraction and a greater emphasis on achieving parallel performance. Abstraction and control are implemented at compilation time within the FLASH framework, enabling these performance advantages to be preserved within a flexible application environment.

PARAMESH uses two main data structures: one to hold the solution and another to store the tree information describing the mesh hierarchy. FLASH makes these structures available to the rest of the application through a variety of accessor methods, enabling most physics modules to be coded without explicit reference to AMR. Each block in the mesh hierarchy contains the same number of interior and guard cells. Blocks on different levels are nested. Between any two adjacent levels the spatial resolution changes by a factor of two, and any block's immediate neighbors can be at most one level finer or coarser than it. In exchange for some loss of flexibility, this method gives very good performance, because it gives modern compilers the best opportunity to efficiently manage cache use.

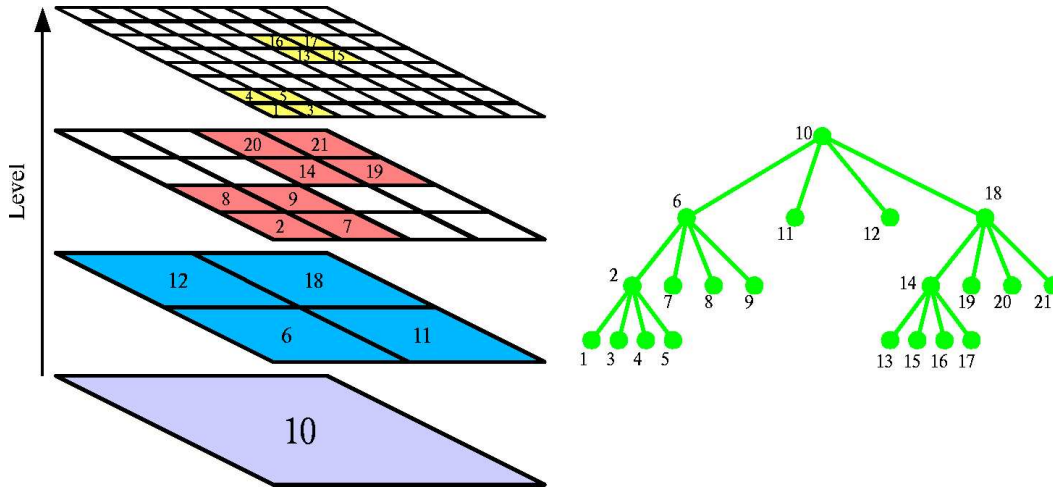


Figure 1: Example of a 2D PARAMESH grid and the corresponding tree structure. Numbers next to each node are assigned using the blocks' Morton ordering.

The criteria for refining blocks are user-defined. For hydrodynamical simulations, we use a normalized second-derivative criterion (Löhner, 1987) to capture shocks. It is important to ensure that flow discontinuities are detected by a block before they move into the interior cells of that block. We therefore test for refinement every four timesteps. Refinement of cosmological simulations must also respond to the need for high resolution in collapsed regions such as clusters without wasting resources in voids. For further discussion of this point see Lukić *et al.* (2005, in preparation).

3 Models of star formation

Simulations of star formation are complicated by the fact that gravity organizes matter on a very wide range of length scales. This calls for a large dynamic range, both in mass and length, and thus incurs the attendant problem of small timesteps. For instance, even if one were simulating star formation in molecular cloud cores of size $\approx 0.1 - 1.0$ pc, the stars that form have a size of a solar radius ($\approx 10^{11}$ cm). This, in itself represents a dynamic range of $\approx 10^7$ with respect to spatial resolution. While such spatial resolutions can be addressed on shared and distributed parallel computing platforms, in conjunction with AMR, one almost inevitably hits a limit while considering extensions of the same methodology to star formation in the interstellar and intracluster media, where the size of the computational domain ranges from ~ 1 kpc to ~ 1 Mpc respectively. One may allude, somewhat crudely, to turbulence simulations and refer to the simulation of star formation in molecular cloud cores as the astrophysical *equivalent* of direct numerical simulation (DNS), simulation of SF in the ISM as the astrophysical *equivalent* of large eddy simulation (LES) and that of SF in the ICM as the astrophysical *equivalent* of very large eddy simulation (VLES).

An important aspect of SF simulations is the criteria one must use in order to identify regions in the computational domain that favor star formation. In the case of SF in molecular cloud cores (e.g., Krumholz *et al.*, 2004), SF is modeled by the creation of sink particles in regions of the flow where the gas will continue to collapse below the scales that are resolved by the simulation. A condition that a computational cell must meet in order to create sink particles is that it must satisfy the Jeans criterion (Truelove *et al.*, 1997, 1998)

$$\rho > \rho_J = J^2 \frac{\pi c_s^2}{G \Delta x^2}, \quad (13)$$

where Δx is the size of a computational cell, J is a constant that is $\mathcal{O}(1)$, and c_s and ρ are the speed of sound and density in the cell. If the density of a cell exceeds ρ_J , one creates a sink particle with mass

$$m_{sink} = (\rho(\mathbf{x}, t) - \rho_J) \Delta x^3. \quad (14)$$

Another condition that must be satisfied by the cell is that the flow path lines are converging – a condition that is met when

$$\nabla \cdot \mathbf{v}(\mathbf{x}, t) < 0, \quad (15)$$

where $\mathbf{v}(\mathbf{x}, t)$ is the velocity field of the gas. Once the particle is formed, its Lagrangian evolution is coupled with the gas-phase hydrodynamics via gravity. The sink particle accretes mass depending on a model accretion rate and gradually

depletes gas from the region surrounding the particle. With sufficient spatial and mass resolution, one may model all stages of the stellar evolution process.

It is important to note, however, that a direct extension of these criteria, while relevant to the DNS-like simulation of SF in molecular clouds, is not quite accurate when one attempts LES-like and VLES-like simulations of SF in the ISM and ICM respectively. For instance, Cen & Ostriker (1992) model SF on length scales of the order of $80h^{-1}$ Mpc by supplementing the Jeans mass and velocity convergence criteria with the radiative cooling criterion, which requires that the radiative cooling timescale t_{cool} be less than the dynamical timescale t_{dyn} for stars to form:

$$t_{cool} < t_{dyn} = \sqrt{\frac{3\pi}{32G\rho_{tot}}}, \quad (16)$$

where ρ_{tot} is the total density of the dark matter and gas. This criterion is reasonable for molecular cloud core scales, on which the removal of thermal energy by radiation allows collapse to stellar densities to occur. However, in large-scale applications in which the resolution elements are kpc or Mpc across, it is not reasonable to assume that the average properties of the gas on such scales must satisfy the Jeans, velocity, or cooling criteria before stars can form.[†] In simulations of SF in the ISM and ICM, there is, clearly, a need for subgrid modeling that will account for the effects of energy and momentum transfer from the filtered-out (small) scales onto the represented (large) scales of motion.

In a recent paper, Slyz *et al.* (2005) present LES-like simulations of SF in the ISM. The computational volume they employ in their simulations is 1.2 kpc across. Like Cen & Ostriker (1992) they use the velocity convergence and cooling criteria, where

$$t_{cool} = \frac{kT\mu}{\rho_{gas}\Lambda(T)}. \quad (17)$$

Here μ is, as before, the average mass per particle, and $\Lambda(T)$ is the radiative cooling function. Since the behavior of $\Lambda(T)$ strongly depends on the temperature, it is often fit by a piecewise power law which covers the temperature range of interest; for example, Slyz *et al.* (2005) use

$$\Lambda(T) = \begin{cases} 0 & \text{if } T < 310K, \\ (2.2380 \times 10^{-32})T^2 & \text{if } 310K \leq T < 2000K \\ (1.0012 \times 10^{-30})T^{1.5} & \text{if } 2000K \leq T < 8000K \\ (4.6240 \times 10^{-36})T^{2.867} & \text{if } 8000K \leq T < 39811K \\ (3.1620 \times 10^{-30})T^{1.6} & \text{if } 39811K \leq T < 10^5K \\ (3.1620 \times 10^{-21})T^{-0.2} & \text{if } 10^5K \leq T < 2.884 \times 10^5K \\ (6.3100 \times 10^{-6})T^{-3} & \text{if } 2.884 \times 10^5K \leq T < 4.732 \times 10^5K \\ (1.047 \times 10^{-21})T^{-0.22} & \text{if } 4.732 \times 10^5K \leq T < 2.113 \times 10^6K \\ (3.981 \times 10^{-4})T^{-3} & \text{if } 2.113 \times 10^6K \leq T < 3.981 \times 10^6K \\ (4.169 \times 10^{-26})T^{0.33} & \text{if } 3.981 \times 10^6K \leq T < 1.995 \times 10^7K \\ (2.399 \times 10^{-27})T^{0.5} & \text{if } T \geq 1.995 \times 10^7K \end{cases}$$

(Note that the use of a cooling function assumes collisional ionization equilibrium, which is not strictly valid at temperatures below $\sim 10^5$ K. In principle one must track the abundances of several molecular and ionic species and model the photoionizing radiation field in order to obtain the correct radiative cooling rate.) However, rather than use the Jeans criterion, they assume that for a region to favor star formation, the gas density should be greater than a threshold density corresponding to $1 - 10$ atoms cm^{-3} . Further, they assume that the mass of the gas converted into stars is given by

$$M_* = \epsilon \frac{\Delta t}{t_{dyn}} \rho_{gas} \Delta V \quad (18)$$

where Δt is the simulation time step, ΔV is the zone volume, and $\epsilon < 1$ is the star formation efficiency. Slyz *et al.* (2005) experimented with different values for ϵ in the range 0.01 – 0.1, and mention that their results did not change significantly.

Kravtsov (2003) uses an approach similar to that of Slyz *et al.* (2005), but without using Jeans-mass or velocity convergence criteria, to reproduce the statistical Schmidt law of star formation, in which the average star formation rate per unit volume is proportional to a power of the gas density (Schmidt, 1959; Kennicutt, 1989) in the interstellar media of simulated galaxies. Although his simulations took place in cosmological volumes $6h^{-1}$ Mpc across, using AMR he was able to achieve spatial resolutions of a few tens of pc, comparable to those of Slyz *et al.* (2005). As he notes, it is important to use different representations of star formation depending on whether or not star-forming regions are resolved. This insight leads naturally to our division of modeling regimes into DNS, LES, and VLES categories.

[†]One may consider an extreme case where the representation of the physics is so coarse-grained that the resulting velocity field is *almost* uniform! Would it then be correct to say that stars do not form in these regions because the velocity field does not satisfy the convergence criterion?

In order to simulate star formation in poorly resolved interstellar media, analogous to the VLES regime, Springel (1999) uses the Schmidt law to develop a subgrid model for star formation. This approach has been implemented in the smooth particle hydrodynamics (SPH) code GADGET. He assumes the star formation rate per unit volume to be proportional to the gas density and inversely proportional to the dynamical time

$$\frac{d\rho_\star}{dt} = \alpha \frac{\rho_{gas}}{t_{dyn}} \quad (19)$$

where ρ_\star denotes the density of stars. The above model equation leads to a Schmidt-type law of star formation, $\dot{\rho}_\star \propto \rho^n$ with $n = 1.5$. The formation of stars depletes the gas density via

$$\frac{d\rho_{gas}}{dt} = -\alpha \frac{\rho_{gas}}{t_{dyn}} = -\alpha G^{0.5} \rho_{gas}^{1.5} \quad (20)$$

In these equations, α is a free parameter of the model which is calibrated using the observed Schmidt law.

Given that the appropriate star formation model depends on the the maximum resolution possible, we have created a framework for star formation in FLASH that can potentially use any one of the above approaches. In the star formation module the checks to identify SF regions are performed on all zones (cells) of all leaf-node blocks, covering the whole computational domain. When the star formation criteria are met, we create star particles, giving them random positions within the zone in which they form and giving them the same velocity as that of the gas in the zone. No modifications are made to the pressure of the remaining gas, so the gas temperature rises as a result of star formation.

To avoid well-known numerical issues arising when particles have significantly different masses (e.g., dynamical friction), especially in this first phase while we are implementing a new physics module in FLASH, we insisted that all particles have the same mass, m_p . Thus, m_p is in effect our stellar mass resolution. Since, in general, M_\star will not be an integer multiple of m_p , we needed to define a way to treat the remainder properly in order to avoid constant underproduction of stars. Therefore, we implemented a stochastic approach: in each zone, we create

$$n_\star \equiv \left\lfloor \frac{M_\star}{m_p} \right\rfloor \quad (21)$$

stars, then assume the probability for creating one more to be

$$p_\star \equiv \frac{M_\star}{m_p} - \left\lfloor \frac{M_\star}{m_p} \right\rfloor. \quad (22)$$

A uniform deviate in the interval [0,1] is chosen, and if it is less than p_\star , we create $n_\star + 1$ stars rather than n_\star . Since the number of zones is, in general, very large, we expect that the average star formation rate over the entire computational volume will be close to that expected from the definition of M_\star given above.

Our Phase 1 simulations use the method of Slyz *et al.* (2005) together with this partially stochastic prescription. We identify regions favoring SF as those where the density of the gas exceeds a certain preset threshold value $\rho_{threshold}$. Since observations indicate that stars form inside galaxies only, and not everywhere in the ICM, we set $\rho_{threshold} = 0.1 m_H \text{ cm}^{-3}$, which is ~ 10 times smaller than the average gas density in spiral galaxies and a few orders of magnitude higher than the average density of the ICM. In our simulations we set the star formation efficiency $\epsilon = 0.1$.

4 Accomplishments in Phase 1 of the project

4.1 Code Development

FLASH has a modular structure, similar to an object-oriented class hierarchy, in which the largest functional units (*modules*) handle different types of physics or code services and subunits (*submodules*) implement alternative choices for solvers, physics details, or library interfaces, overriding methods defined at higher levels. The source code for these modules and submodules is organized into a directory structure that mirrors the module relationships. The subdirectory for each submodule also contains a `Makefile` fragment and a plain-text configuration file that describes module interrelationships and data structures required by the submodule. As most of the code is written in Fortran 90, this is not a true class hierarchy; however, a Python-based configuration script is used to combine a set of user-chosen submodules into a build directory where a particular instantiation of the code is then created using `gmake`. Thus many of the benefits of a true class hierarchy are retained, with the exception of runtime polymorphism. More details concerning the structure of FLASH can be found in the FLASH User's Guide[‡].

[‡]Available for download from <http://flash.uchicago.edu>.

The `particles` module handles the data structures and algorithms needed to follow the behavior of particles in FLASH. It interacts with the `driver` module, which drives the time evolution and calls all of the solver methods; the `database` module, which provides access to solution variable, mesh, and control parameter data; the `mesh` module, which manages the adaptive mesh; and the `io` module, which handles checkpointing. Particles are defined as objects having a mass, position, and velocity, as well as other necessary attributes. They can represent a variety of physical entities, but at a minimum methods need to be supplied to move them around and to exchange them between processors as needed. Particles are stored together with the leaf-node mesh block in which they are located, allowing particle-mesh transfer operators to be implemented with high parallel efficiency.

The structure of the particle module is depicted in Figure 2. Currently FLASH supports two types of particles: active particles, which contribute to and are affected by force fields such as gravity, and passive particles, which are used to trace gasdynamical flows and obtain their velocities by interpolation from the gas velocity field. The `particles` module includes submodules to handle each of these types of particles. The principal method implemented by these submodules, `AdvanceParticles()`, updates the particle positions and velocities. Two other submodules, `mapping` and `communication`, handle interpolation between particle locations and mesh zones and interprocessor communication, respectively. Our Phase 1

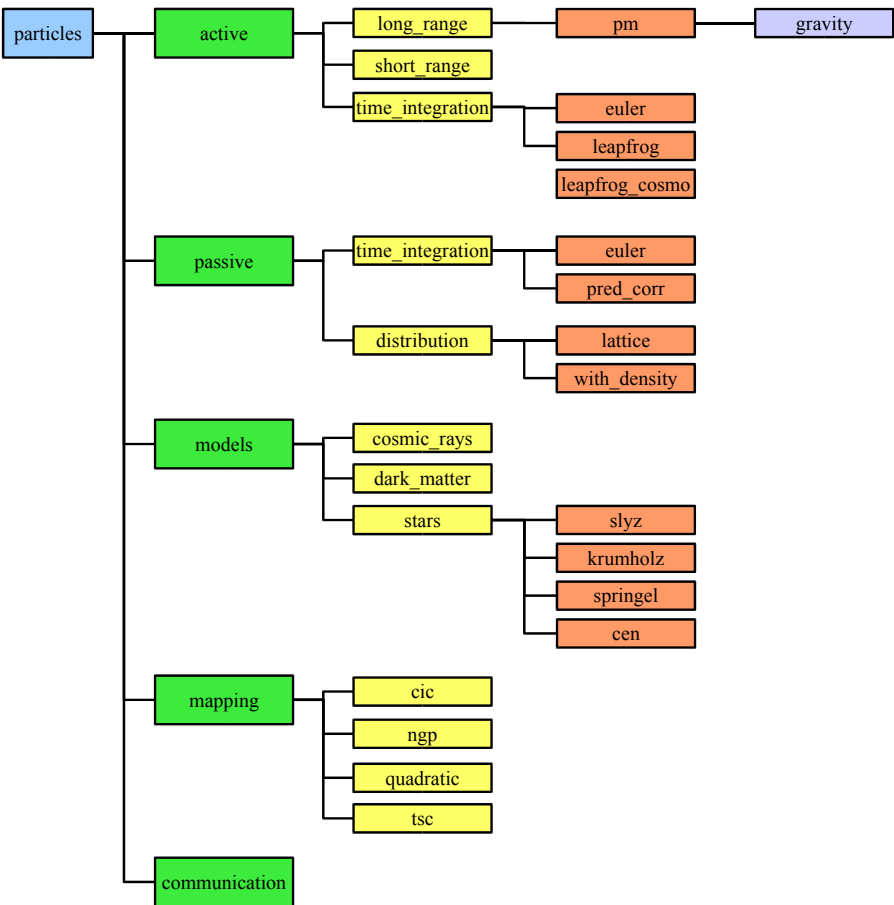


Figure 2: Structure of the particle module in FLASH. Submodules with hashed lines are planned for the next phase of the project.

SAP project has focused on the development of a new submodule, `models`, which allows particle properties other than position and velocity to be manipulated using prespecified models. The primary method implemented by the `models` submodule is called `AdvanceParticleModel()`. Submodules of `models` implement models for star formation, cosmic ray propagation,

and any other situations in which the creation, destruction, or modification of particles needs to be handled specially. Each submodule provides an update method which is called by `AdvanceParticleModel()` for those particles whose types are appropriate. To allow for this possibility, we added a new integer-valued particle attribute, `particle_type`, whose value is tested against constants defined in a new Fortran 90 module called `ParticleModelData`.

The `stars` submodule implements methods needed to create, destroy, and age star particles, as well as to implement star-related feedback effects on the gas. Each star particle represents a large number of stars, each of which is assumed to have the same mass. (An alternative representation, in which each star particle represents a distribution of masses, is also allowed.) The attribute `particle_age` is defined in addition to the standard particle attributes; it provides a clock which is set to zero when the particle is created. Each clock is advanced as the code steps through time. Together with the masses of the stars associated with the particle, the particle’s age determines when various feedback effects such as stellar winds and supernovae are introduced by the model.

Four methods are provided by the `stars` submodule. These are `AdvanceStarParticleModel()`, `CreateStars()`, `AgeStars()`, and `DestroyStars()`. The first of these invokes the other three and is only provided as an interface to the method `AdvanceParticleModel()`. Currently most of the work is done by `CreateStars()`; this method implements our star formation model and is embarrassingly parallel because particles are stored together with the mesh zones containing the gas from which they form. (Note, however, that star formation models which rely on $\nabla \cdot \mathbf{v}$ must perform a finite-difference operation and thus first perform a guardcell transfer.) VLES models also use `CreateStars()` to implement the effects of Type II supernovae, which arise from massive stars; because the main-sequence lifetimes of these stars are shorter than the typical timesteps in VLES simulations, they are born, live, and die entirely within one step and therefore do not need to be tracked.

The simplest form of `AgeStars()` simply increments the ages of stars that have already been created. It is also possible to implement stellar winds and other pre-death stellar feedback within this method.

Finally, `DestroyStars()` handles the final steps in stellar evolution for those stars that travel some distance before dying. This includes Type Ia supernovae, which arise in star systems of relatively low mass, as well as Type II supernovae in DNS and LES simulations. Despite the name of this method, stars are only destroyed in Type Ia supernovae; other end states always include a compact stellar remnant such as a white dwarf, neutron star, or black hole, whereas Type Ia supernovae result in the disruption of a white dwarf. Thus in most cases we do not actually remove the star particle from the simulation, but rather return a significant fraction of its mass and energy to the diffuse gas.

4.2 Results of the Phase 1 simulations

In order to reduce the computational cost of the first phase of this project, we created a model of a galaxy in which we can examine all star formation issues. An isolated galaxy is 10 – 100 times smaller in size than a cluster of galaxies, and also $10^3 - 10^5$ times less massive, allowing us to perform all code accuracy checks without running costly high-resolution simulations.

We created a model of an elliptical galaxy consisting of all three important components – dark matter (DM), stars, and interstellar gas. The extended DM halo is simulated using particles distributed according to the NFW density profile (Navarro *et al.*, 1996, 1997). Star particles present in the initial conditions are distributed according to the Hernquist profile (Hernquist, 1990), known to be an excellent description of the light profile in elliptical galaxies. Finally, the gas is set up using the β -model (Cavaliere & Fusco-Femiano, 1976), and its density is set to an unrealistically high value; we put 20% of all baryonic matter into gas. While this high value is reasonable for spiral galaxies, ellipticals are known to be poor in gas, implying a very low present SF rate. The rationale behind our choice is that our goal here was not to make an elliptical model as accurately as possible, but rather to create a testbed for star formation models. Also, we made this model in such a way as to allow us to easily simulate collisions of two galaxies, an important test for the star formation module. We set the luminous mass of the galaxy to $10^{10} M_{\odot}$, while the total mass was about $10^{11} M_{\odot}$.

After setting these initial conditions, we allowed the galaxy to evolve. Since there is no interaction with any other object, the only things we expect to happen are some relaxation (because all three distributions mentioned above end with a sharp cutoff for numerical reasons), and, of course – star formation.

To check the validity of our results we took recent observational data regarding the supernova (SN) rate per unit mass (Mannucci *et al.*, 2005). The core-collapse SN rate varies between 0.1 and 2 SN per century per $10^{10} M_{\odot}$ of luminous matter, depending on galaxy morphology. The lowest value, 0.1, is measured for S0a/b type galaxies; Sbc/d have ~ 0.7 ; and the highest is for irregular galaxies, ~ 2 . We have to emphasize here that we are implementing an algorithm to be used in large-scale simulations where the maximum achievable resolution is \sim kpc, so we are not taking morphology into account at this point. Assuming a Salpeter initial mass function (Salpeter, 1955),

$$N(M) dM = N_0 M^{-2.35} dM , \tag{23}$$

where $N(M) dM$ denotes the probability for a star to have a mass M in the range M to $M + dM$, we can find the fraction

of stars that will end as supernovae (stars with mass $> 5 M_{\odot}$):

$$f_{sn} = \frac{\int_5^{20} N(M)dM}{\int_{0.1}^{20} N(M)dM} = 0.43\% . \tag{24}$$

Here we take $20 M_{\odot}$ as an upper limit, because the Salpeter function has been confirmed up to that mass; but even theoretical integrations to infinity will lead to almost the same number. (The lower threshold used to normalize the fraction is based on the minimum mass needed for a star to exist.) This means we can expect ~ 1 SN per $250 M_{\odot}$ formed per year, implying a SF rate in our simulation (luminous mass $10^{10} M_{\odot}$) of

$$0.25 - 5 M_{\odot} \text{ yr}^{-1} . \tag{25}$$

Figure 3 shows this interval (red dashed lines) together with the cumulative SF rate we obtained for an isolated galaxy on a 64^3 grid using a mass resolution of $1.5 \times 10^4 M_{\odot}$. Although we have not explicitly calibrated the star formation model against observations, the computed stellar mass lies within the observational bounds.

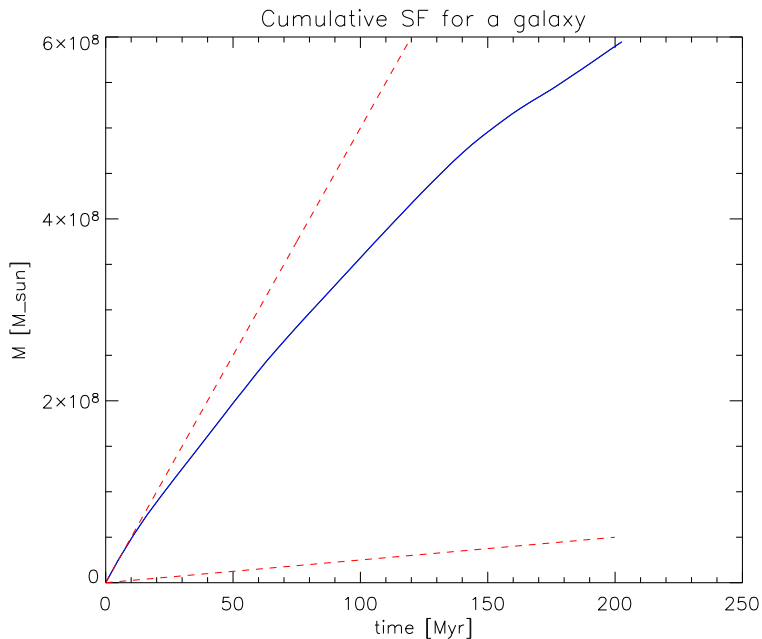


Figure 3: Cumulative star formation rate (blue) obtained on a 64^3 grid using a mass resolution of $1.5 \times 10^4 M_{\odot}$. Red dashed lines indicate the observational bounds given by equation (25).

4.3 Code performance on NCSA platforms

We have ported FLASH together with the star formation module to the Tungsten and Mercury Linux clusters at NCSA. Porting to Tungsten using the Intel 8 compiler and Champion MPI revealed several double-precision memory alignment errors that had not caused problems on other machines, but led to difficulties when Fortran 90 derived data types were passed to MPI. The Totalview compiler proved instrumental in tracking down these errors. Another issue which has still not been completely resolved on Tungsten pertains to reading and writing HDF5 checkpoint files using the Lustre parallel file system. I/O performance on Lustre has been extremely erratic. Some runs will read and write checkpoint files at acceptable speeds (typically completing within a few minutes), but identical runs often get stuck reading or writing at speeds that would have them complete only after several hours. Greg Bauer (NCSA) has been able to reproduce these problems and is still investigating their origin. For the meantime we have developed two acceptable workarounds: using the `/nfs/scratch` filesystem and writing to node local disks. While NFS is slower than Lustre’s theoretical peak speed, in practice it has been much more reliable. Using node local disks is very fast but runs the risk of data loss if the queue window closes before the

job script runs to completion (since the script copies data off of the nodes when FLASH finishes, and the queuing system denies user access to the node when the requested wall-clock time is exceeded).

No major issues were identified when porting to Mercury. We are able to compile and run on Mercury using the Intel 8 compiler, and we experience no I/O performance problems. However, we have encountered problems when using the ‘watch’ capability of Totalview – the values given by Totalview differ from those produced by print statements placed in the code by hand.

To study scaling performance on Mercury, we ran the isolated galaxy problem described in § 4.2 with minimum and maximum refinement levels of 3 and 9, respectively (corresponding to a maximum effective resolution of 2048^3). Using a particle mass of $3 \times 10^3 M_\odot$ gave us almost 17 million dark matter particles and 2.7 million star particles to begin with. This run was carried out for twenty timesteps on different numbers of processors. The results are shown in Figure 4.

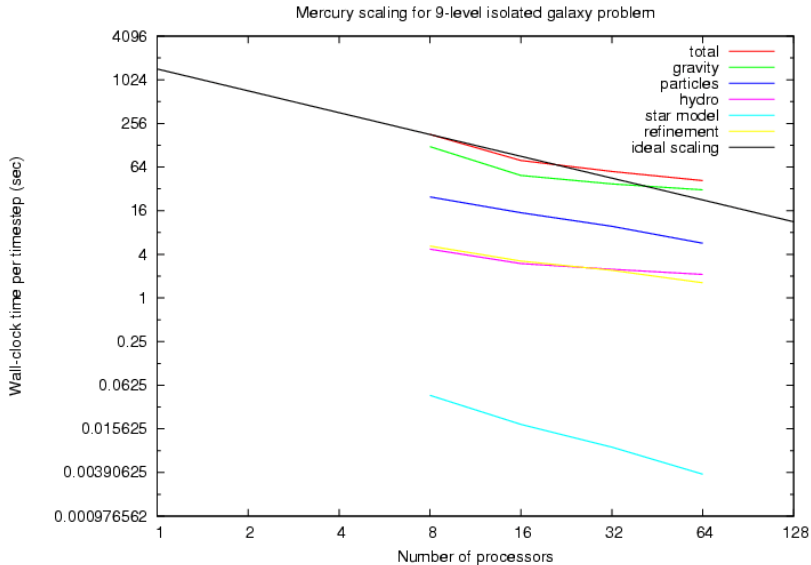


Figure 4: Constant total work scaling on Mercury of FLASH for the isolated galaxy problem using AMR with an effective mesh size of 2048^3 zones and a particle mass of $3 \times 10^3 M_\odot$ (yielding about 20 million particles).

This job runs on fewer processors, but the length of runs on 1-4 processors precluded their inclusion in this report. Scaling from eight to sixteen processors appears to be superlinear but may be due to lack of control over which set of nodes were used. Comparing eight and 64 processors, the parallel efficiency is about 54%. The execution time is dominated by the multigrid Poisson solver, and most of the scaling behavior is dominated by this solver. (In Phase 2 we plan to investigate ways to improve the overall performance of the multigrid solver.) Most of the time associated with particles is devoted to computing forces on them and exchanging them between processors; the star model itself scales perfectly (as expected) and contributes a negligible fraction of the total execution time.

5 Conclusion

During the first phase of this SAP project, we have created a framework to include the effects of star formation and stellar feedback in FLASH. Within this framework we have implemented a simple algorithm to create stars according to a prescription similar to that described by Slyz *et al.* (2005). Using this solver, we have included star formation in the interstellar medium (LES/VLES-like regime) for simulations of an isolated spherical galaxy and mergers between galaxies. The solver has been successful in creating and evolving more than ten million particles with nine levels of adaptive mesh refinement. We have examined the convergence properties of the solver and determined what particle mass is needed to achieve reasonably converged results for meshes of different sizes. We have also shown that the basic solver works with adaptive meshes.

Acceptable scaling performance was demonstrated on Mercury up to 64 processors; the star formation module has a negligible impact on overall performance. We identified the multigrid Poisson solver as the main contributor to deviations from perfect scaling.

6 Acknowledgments

RB would like to thank John Towns, Senior Associate Director of the Persistent Infrastructure Division at NCSA, for giving him the opportunity to collaborate with PR and ZL under the aegis of the Strategic Applications Program (SAP). RB would also like to thank the manager of the Consulting Group, Ms. Susan John, for her help and understanding in granting time to complete the first phase of this project. The authors are immensely grateful to NCSA for allowing them to use the platforms Tungsten and Mercury during this project.

PR and ZL acknowledge support from NCSA and UIUC Department of Astronomy faculty startup funds. ZL acknowledges support from the National Academy of Sciences under a Grant-in-Aid of Research administered by Sigma Xi, The Scientific Research Society.

The authors would like to thank Dr. Alessandro Gardini for sharing with us his thoughts on star formation. The authors would also like to thank Dr. Greg Bauer of the Performance Engineering and Computational Methods (PECM) group at NCSA for help regarding the I/O problem with the Lustre filesystem on Tungsten.

FLASH development is supported by DOE under contract B341495 to the ASCI Flash Center at the University of Chicago and also partially by the University of Illinois and the National Center for Supercomputing Applications, the University of California, and NASA Goddard Space Flight Center. FLASH is freely available at <http://flash.uchicago.edu/>.

References

- BERGER, M. J. & COLELLA, P. 1989 Local Adaptive Mesh Refinement for Shock Hydrodynamics. *Journal of Computational Physics* **82**, 64.
- BERGER, M. J. & OLIGER, J. 1984 Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations. *Journal of Computational Physics* **53**, 484.
- BRANDT, A. 1977 Multi-Level Adaptive Solutions to Boundary Value Problems. *Math. Comp.* **31**, 333.
- CALDER, A. C., CURTIS, B. C., DURSI, L. J., FRYXELL, B., HENRY, G., MACNEICE, P., OLSON, K., RICKER, P., ROSNER, R., TIMMES, F. X., TUFO, H., TRURAN, J. W. & ZINGALE, M. 2000 High-Performance Reactive Fluid Flow Simulations Using Adaptive Mesh Refinement on Thousands of Processors. In *Proceedings of SC2000*.
- CALDER, A. C., FRYXELL, B., PLEWA, T., ROSNER, R., DURSI, L. J., WEIRS, V. G., DUPONT, T., ROBEY, H. F., KANE, J. O., REMINGTON, B. A., DRAKE, R. P., DIMONTE, G., ZINGALE, M., TIMMES, F. X., OLSON, K., RICKER, P., MACNEICE, P. & TUFO, H. M. 2002 On Validating an Astrophysical Simulation Code. *Astrophysical Journal Supplement* **143**, 201–229.
- CAVALIERE, A. & FUSCO-FEMIANO, R. 1976 X-rays from hot plasma in clusters of galaxies. *Astronomy and Astrophysics* **49**, 137–144.
- CEN, R. & OSTRICKER, J. P. 1992 Galaxy formation and physical bias. *Astrophysical Journal* **399**, L113–L116.
- COLELLA, P. & WOODWARD, P. 1984 The Piecewise Parabolic Method (PPM) for Gas-Dynamical Simulations. *Journal of Computational Physics* **54**, 174.
- FRYXELL, B., OLSON, K., RICKER, P., TIMMES, F. X., ZINGALE, M., LAMB, D. Q., MACNEICE, P., ROSNER, R., TRURAN, J. W. & TUFO, H. 2000 FLASH: An Adaptive Mesh Hydrodynamics Code for Modeling Astrophysical Thermonuclear Flashes. *Astrophysical Journal Supplement* **131**, 273–334.
- HERNQUIST, L. 1990 An Analytical Model for Spherical Galaxies and Bulges. *Astrophysical Journal* **356**, 359–364.
- HOCKNEY, R. W. & EASTWOOD, J. W. 1988 *Computer Simulation using Particles*. IOP.
- KENNICUTT, R. C. 1989 The Star Formation Law in Galactic Disks. *Astrophysical Journal* **344**, 685–703.
- KNEBE, A., GREEN, A. & BINNEY, J. 2001 Multi-level adaptive particle mesh (MLAPM): a c code for cosmological simulations. *Monthly Notices of the Royal Astronomical Society* **325**, 845–864.
- KRAVTSOV, A. V. 2003 On the Origin of the Global Schmidt Law of Star Formation. *Astrophysical Journal Letters* **590**, L1–L4.
- KRUMHOLZ, M. R., MCKEE, C. F. & KLEIN, R. I. 2004 Embedding Lagrangian Sink Particles in Eulerian Grids. *Astrophysical Journal* **611**, 399–412.

- LÖHNER, R. 1987 An Adaptive Finite Element Scheme for Transient Problems in CFD. *Comp. Meth. App. Mech. Eng.* **61**, 323.
- MACNEICE, P., OLSON, K. M., MOBARRY, C., DE FAINCHEIN, R. & PACKER, C. 2000 PARAMESH: A Parallel Adaptive Mesh Refinement Community Toolkit. *Comp. Phys. Comm.* **126**, 330.
- MADORE, B. F. & FREEDMAN, W. L. 1998 Calibration of the Extragalactic Distance Scale. In *Stellar astrophysics for the local group: VIII Canary Islands Winter School of Astrophysics*, p. 263.
- MANNUCCI, F., DELLA VALLE, M., PANAGIA, N., CAPPELLARO, E., CRESCI, G., MAIOLINO, R., PETROSIAN, A. & TURATTO, M. 2005 The Supernova Rate per Unit Mass. *Astronomy & Astrophysics* **433**, 807–814.
- MARTIN, D. 1998 An Adaptive Cell-centered Projection Method for the Incompressible Euler Equations. PhD thesis, University of California at Berkeley.
- MARTIN, D. & CARTWRIGHT, K. 1996 Solving Poisson’s Equation using Adaptive Mesh Refinement. *Tech. Rep.* UCB/ERL M96/66. University of California at Berkeley.
- NAVARRO, J. F., FRENK, C. S. & WHITE, S. D. M. 1996 The Structure of Cold Dark Matter Halos. *Astrophysical Journal* **462**, 563.
- NAVARRO, J. F., FRENK, C. S. & WHITE, S. D. M. 1997 A Universal Density Profile from Hierarchical Clustering. *Astrophysical Journal* **490**, 493.
- QUIRK, J. J. 1991 An Adaptive Grid Algorithm for Computational Shock Hydrodynamics. PhD thesis, Cranfield Institute of Technology.
- SALPETER, E. E. 1955 The Luminosity Function and Stellar Evolution. *Astrophysical Journal* **121**, 161.
- SCHMIDT, M. 1959 The Rate of Star Formation. *Astrophysical Journal* **129**, 243.
- SLYZ, A. D., DEVRIENDT, J. E. G., BRYAN, G. & SILK, J. 2005 Towards Simulating Star Formation in the Interstellar Medium. *Monthly Notices of the Royal Astronomical Society* **356**, 737–752.
- SPRINGEL, V. 1999 On the formation and evolution of galaxies. PhD thesis, Ludwig-Maximilians-Universität München.
- TRUELOVE, J. K., KLEIN, R. I., MCKEE, C. F., HOLLIMAN, J. H., HOWELL, L. H. & GREENOUGH, J. A. 1997 The Jeans Condition: A New Constraint on Spatial Resolution in Simulations of Isothermal Self-gravitational Hydrodynamics. *Astrophysical Journal* **489**, L179.
- TRUELOVE, J. K., KLEIN, R. I., MCKEE, C. F., HOLLIMAN, J. H., HOWELL, L. H., GREENOUGH, J. A. & WOODS, D. T. 1998 Self-gravitational Hydrodynamics with Three-dimensional Adaptive Mesh Refinement: Methodology and Applications to Molecular Cloud Collapse and Fragmentation. *Astrophysical Journal* **495**, 821.
- WARREN, M. S. & SALMON, J. K. 1993 A Parallel Hashed Oct-Tree N-Body Algorithm. In *Supercomputing 1993*.
- DE ZEEUW, D. & POWELL, K. G. 1993 An Adaptively Refined Cartesian Mesh Solver for the Euler Equations. *Journal of Computational Physics* **104**, 56.