Dissertation

submitted to the

Combined Faculties for the Natural Sciences and for Mathematics

of the Ruperto-Carola-University of Heidelberg, Germany

for the degree of

Doctor of Natural Sciences

Put forward by

Lei Liu

born in: Kaifeng, China

Oral examination: 4th December, 2013

A Multi-Phase Chemodynamic Galaxy Formation Model

Referees: Prof. Dr. R. Spurzem
Prof. Dr. C. Dullemond

Abstract

In this thesis, I present my PhD work: a multi-phase chemodynamic galaxy formation and evolution model. The model is aimed at treating the dynamics of stars, molecular (cold) clouds, and hot/warm diffuse gas individually and allowing for mass, momentum, and energy exchange between them in a self-consistent way, so as to overcome the difficulties of a single-phase description. I introduce the detailed implementation of physical processes in the model including gravity, gas dynamics, heat conduction, cooling, star formation and stellar feedback. A dwarf galaxy model is evolved for 1 Gyr. The corresponding star formation rate decreases from 1 $M_\odot$/Year to 0.1 $M_\odot$/Year. The cloud mass distribution follows a power law with a slope of -2.3. The discrepancies of chemical abundance between hot/warm and cold phase are reproduced. As an extension to the classical multi-phase model, I introduce a transition process such that hot/warm gas can collapse to cold clouds, which solves the problem of cold clouds' initial mass fraction and distribution in the multi-phase simulation. This process is proven to be more suitable for low mass systems. Also I implement an individual star formation model, in which individual stars are created analytically inside a molecular cloud with a stellar mass distribution given by a specific initial mass function (IMF). This model reproduces the life cycle of interstellar medium in a galactic scale simulation and realizes the process of star cluster formation inside one molecular cloud. The multi-phase code is parallelized with MPI and shows good scaling relations. GPUs are used to accelerate the most time consuming parts (gravity, SPH and neighbour search), which results in a speedup of one oder of magnitude for the whole program.

Zusammenfassung

In meiner Dissertation präsentiere ich ein Chemodynamisches Galaxienformationsmodell das mehrere Gasphasen berücksichtigt. Das Simulation beinhaltet die Bewegung von Sternen, Molekularen Wolken und den Energieaustausch zwischen beiden auf selbstkonsistente Weise. Ich zeige detailliert die Implementierung der einzelnen physikalischen Prozessen wie Gravitationskraft, Gasdynamik, Wärmetransport, Abkühlung, Sternentstehung und deren Rückkopplung. Als Anwendungsbeispiel simuliere ich die Entwicklung einer Zwerggalaxie über 10^9 Jahre. Die Sternentstehungsrate fällt von 1 $M_\odot$/Jahr auf $0,1$ $M_\odot$/Jahr. Die Wolkenmassenverteilung folgt einer Potenzfunktion mit einer Steigung von -2,3. Der Unterschied der Elementhäufigkeiten zwischen warmer und kalter Phase wird reproduziert. Als Erweiterung zum klassischen Zweiphasenmodell habe ich einen Übergangsprozess, der den Kollaps von der warmen in die kalte Phase abbildet, eingeführt. Diese Herangehensweise ist gut geeignet für Galaxiensysteme mit geringer Masse. Auch wurde die Sternentstehung überarbeitet, sodass nun einzelne Sterne in den Molekularwolken gemäß einer beliebigen IMF gebildet werden. Das Modell reproduziert den Lebenszyklus des interstellaren Mediums in einer Simulation von Galaxiengröße und schafft es gleichzeitig die Sternenhaufenformation in Molekularen Wolken abzubilden. Der Multiphasencode ist mit MPI parallelisiert und zeigt gute Skalierungseigenschaften. Parallelgeschaltete Grafikprozessoren werden benutzt um die rechenintensivsten Schritte zu beschleunigen, wodurch die Berechnung der gesamten Simulation um eine Größenordnung schneller wird.

献给我的妻子张叔华女士：

在远方默默支持着我，给我安慰，鼓励以及继续前行的勇气。

Contents

List of Figures

List of Tables

"Perceiving the ultimate truth in the morning, one could be satisfied with death in the evening."

Analects, Confucius

1 Introduction

Galaxies are the building blocks of our universe. The commonly accepted theory of galaxy formation suggests that galaxies form inside the dark matter halos. The general picture is: the major matter content in the universe is cold dark matter (CDM); pre-inflation quantum fluctuation provides the seeds for the inhomogeneous distribution of cold dark matter; due to the gravitational instability, cold dark matter evolves to dark matter halos; gas falls into the dark matter halos, and is shock heated to the host halos' virial temperature; galaxies form as gas condenses in the halo center and grow by receiving the cooled gas flow (Rees & Ostriker 1977; White & Rees 1978). Therefore the distribution of the galaxies traces the distribution of the underlying dark matter. In the small scale, gas-dynamical processes play more important roles for the interpretation of galaxies' observational properties. E.g., the formation of a disk galaxy: dark matter mainly stays in the outer part of the galaxy since it only takes part in gravitational interaction; gas contracts as a result of dissipation, thus rotates fast for the conservation of angular momentum and finally forms a thin disk. Other examples include galaxy mergers to form elliptical galaxies, accompanied with a star burst process; supernovae feedback drives gas away from the galaxy center, thus gives a possible explanation to the cuspy halo problem (de Blok 2010).

1.1 Dwarf galaxies

Based on different color and morphology, galaxies can be divided into two classes: spirals (S) and ellipticals (E). Spiral galaxies are featured for their flattened disks and spiral arms, and are referred as late-type galaxies. They are blue and star forming. Around half of them have a bar-shaped structure (Eskridge & Frogel 1999), which is classified as barred spiral galaxies (SB). Elliptical galaxies have ellipsoid shape, and are red with smooth light distribution (Paudel 2011). They are referred as early-type galaxies, inside which the star formation activities are low and stars are much older than that in late-type galaxies. Besides these two types, there is another type called lenticular galaxies (S0), which are red and smooth with disk-like structures.

For galaxies in different mass range, low mass dwarf galaxies are overlooked for quite a long time due to their low surface brightness (Paudel 2011). It was until 1938 that Shapley first discovered two dwarf companions to the Milky Way: Fornax and Sculptor (Shapley 1938; Ferguson & Binggeli 1994). In the recent decades, dwarf galaxies have received more and more attention due to their important roles for the understanding of galaxy formation and evolution

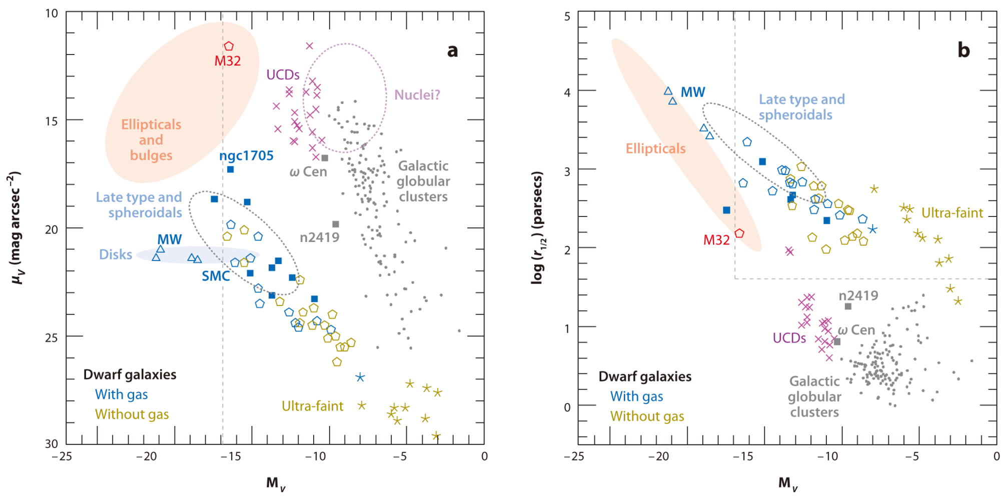


Figure 1.1: Relationships between structural properties for different types of galaxies. Left: absolute magnitude M_V vs. central surface brightness μ_V. Right: M_V vs. half light radius $r_{1/2}$. Local Group dwarf galaxies are plotted as open pentagons, blue and yellow correspond to systems with and without gas. Blue compact dwarfs are marked as blue solid squares. M31, the Milky Way (MW), M33 and LMC are marked as blue open triangles. Reproduced from figure 1 of Tolstoy et al. (2009).

in the cosmological background: in the hierarchical galaxy formation model, galaxies grow mass by merging of small galaxies. Tammann (1994) defined all galaxies that are fainter than $M_V \geq -17$ and more spatially extended than globular clusters as dwarf galaxies. Based on their morphologies, dwarf galaxies can be further divided into different classes: dwarf elliptical (dE), dwarf spheroidal (dSph) [1], dwarf irregulars (dIs), dwarf spiral (dSs), etc. See Grebel (2001) for a comprehensive introduction to classification of dwarf galaxy. For the various types of dwarf galaxies, early-type dwarf galaxies (dEs) are the dominant galaxy population by number in massive galaxy clusters (Jerjen & Tammann 1997; Lisker et al. 2013). Dwarf spirals are quite rare, which were first discovered by Schombert two decades ago (Schombert et al. 1995).

Figure 1.1 demonstrates the position of dwarf galaxies in the M_V - μ_V, M_V - $r_{1/2}$ diagrams. From both diagrams a clear distinction between dwarfs and globular clusters is observed: compared with dwarf galaxies, globular clusters have higher surface brightness and lower absolute magnitude, and are much more compact. Also they follow totally different magnitude - surface brightness relation. On the other side, the properties of dwarfs galaxies are quite different from ellipticals: in the M_V - μ_V diagram, the surface brightness of dwarf galaxies are much lower;

[1]In Grebel (2001), dSphs are characterized as $M_V \gtrsim -14$ mag, $\mu_V \gtrsim 22$ mag arcsec^{-2}, $M_{\rm HI} \lesssim 10^5$ M$_\odot$ and $M_{\rm tot} \sim 10^7$ M$_\odot$. In the study of dwarf galaxies in Virgo cluster, Lieder et al. (2012) demonstrates that the scaling relations (luminosity-size-surface brightness) show no difference between dEs and and dSphs, which suggests these two types have the same origin.

in the M_V - $r_{1/2}$ diagram, they follow different relations, and the scatter of dwarf galaxies are larger. One interesting phenomenon is late-type systems and dwarfs follow the same relation on both diagrams, and their distributions actually overlap with each other. This suggests that the numerous dEs are transformed from late-types systems by environmental processes (Kormendy 1985; Kormendy et al. 2009), which is referred as the morphology transformation mechanism.

Morphology transformation assumes that late-type spirals and irregulars transform to the early-type ellipticals when they fall into the clusters. The higher the density, the more efficiently the transformation is. This assumption is suggested by the famous morphology-density relation in the clustered environments: the dense core regions are richer in early-type galaxies, especially almost all clusters' Bright Central Galaxies (BCG) are early-types. Late-type galaxies mainly distribute in the outer region of clusters (Paudel 2011). Various mechanisms are proposed to explain the transformation process. Gunn & Gott (1972) first propose "ram pressure" stripping, which removes gas efficiently from the infalling galaxies. Using high precision numerical simulation, Moore et al. (1996) demonstrate that high-speed encounters between galaxies can transform small disk galaxies into dwarf ellipticals. They refer this process as "galaxy harassment", so as to distinguish it from the totally different galaxy mergers. Mayer et al. (2001) shows that strong tidal field of Milky Way yields severe mass loss in the dwarf's host halo and disks, and finally transform dwarf disk galaxies into dEs in less than 10 Gyr. They name this process as "tidal stirring".

A large diversity of star formation rate (SFR), morphologies and other aspects still exists among numerous dEs, which suggests that the origin of dEs cannot be simply explained by the mechanism of morphology transformation. E.g., Lisker et al. carry out a systematic study of dEs in Sloan Digital Sky Survey (SDSS) Data Release 4 (DR4; Adelman-McCarthy et al. 2006). By analyzing the imaging data of 413 Virgo cluster dEs, they found that dE class can be divided into multiple subpopulations with different morphology and clustering properties (Lisker et al. 2007). Especially, they have identified a class of dEs with disk features, which is named as "dEdis" (Lisker et al. 2006). The existence of disk featured dEs suggests that at least two different mechanisms are required for the formation scenarios of dEs with and without disk features.

As demonstrated above, numerical modeling is crucial in understanding the formation mechanism of various types of dwarf galaxies. As a first step, in this thesis I focus on the modeling of isolated dwarf galaxy using multi-phase method. The formation and evolution of galaxies are one of the most fundamental questions in modern astronomy. The coupling of complex physics makes it almost impossible to describe the whole scenario analytically. In the next few sections, I will give an introduction to the numerical models and the related physical processes involved in the numerical modeling of galaxies.

1.2 Semi-analytical models

One way to model the baryonic part of galaxy is by using the popular semi-analytical models (SAM). SAM studies galaxy formation in the cosmological framework. The basic idea is simple: baryons and dark matter are only coupled via gravity, therefore more computational resources are preferred to be assigned to pure N-body simulation for a high resolution dark matter distribution. The small scale physical processes which are not fully resolved by numerical simulation, such as star formation and stellar feedback, can be implemented to the model based on the halo merger tree with a set of ordinal differential equations (White & Frenk 1991; Kauffmann et al. 1993, 1999; Cole et al. 1994, 2000; Somerville & Primack 1999; Springel et al. 2001a; Kang et al. 2005). In the model, baryons are divided into several phases: hot gas, cold gas, stars, and an optional ejected gas component. Hot gas cools to cold gas. Cold gas transits to stars as star formation process. The supernova feedback reheats cold gas to the hot. Some of the reheated gas is ejected from the galaxy and will come back to the hot gas phase as the re-incorporation process. Besides these basic processes, galaxy merger to trigger star burst, ram-pressure stripping of gas in satellite galaxies, tidal disruption of satellites, bulge formation, black hole growth and AGN feedback are also implemented in the up to date models (Croton et al. 2006; De Lucia & Blaizot 2007; Fu et al. 2013). Currently the semi-analytical models can be divided into two branches: one branch is the "L-Galaxy" model developed by Munich group (Kauffmann et al. 1999; Springel et al. 2001a; Croton et al. 2006; De Lucia & Blaizot 2007; Fu et al. 2010; Guo et al. 2011); another branch is the "GALFORM" model developed in Durham University (Cole et al. 2000; Benson et al. 2003; Bower et al. 2006). Both of the two branches are based on the halo merger trees extracted from Millennium and Millennium II simulations (Springel et al. 2005b; Boylan-Kolchin et al. 2009), and implement above mentioned physical processes in slightly different ways. Since these small scale gas processes are not fully understood, their study involves the investigation of the large parameter space. The SAM method is very suitable for this work due to its faster speed compared with real numerical simulation. SAMs are able to produce some results that can be compared with observation directly, such as the Tully-Fisher relation, the luminosity function and so on. We can further constrain the parameters of SAM with these observations. In my Master's thesis work, I tried to construct my own model, and wrote a paper about the comparison between three semi-analytical models (Kang et al. 2005; Bower et al. 2006; De Lucia & Blaizot 2007) with the galaxy group catalogue (Yang et al. 2008, 2009a,b) constructed with the Sloan Digital Sky Survey (SDSS), including the conditional stellar mass function, the stellar mass function and the stellar mass - halo mass relation (Liu et al. 2010).

1.3 Numerical simulations

It is quite clear that SAM study are not based on first principles: galaxies are treated as black boxes; the results are only statistically meaningful; observations are required as input parameters to constrain the model. What astronomers want is to build the whole picture of galaxy formation from scratch based on some fundamental physical processes: gravity, hydrodynamics, radiative transfer, etc. Therefore numerical simulations are more favored in this sense. Scientists produce various numerical methods to model these physical processes, and combine them together according to current understanding of the universe, hoping that some phenomenons in galaxy formation can be reproduced. This is a tough task, but we believe that more and more

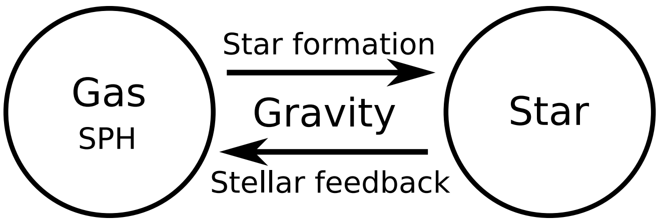


Figure 1.2: Demonstration for a single-phase model.

details of galaxy formation will be unveiled with the development of computational methods and rapid growth of computing power. My work is about numerical simulation, more specifically, the multi-phase chemodynamical galaxy formation model. In this section, I will give a brief introduction to the classification of numerical galaxy formation models.

1.3.1 The single-phase model

In the single-phase model, smoothed particle hydrodynamics (SPH), together with N-body method, imply a Lagrangian description of the galaxies' multiple components including stars, interstellar medium (ISM), and dark matter (Lucy 1977; Monaghan 1992; Katz 1992; Navarro & White 1993; Springel et al. 2001b; Springel 2005; Stinson et al. 2006, 2009; Saitoh et al. 2008; Hopkins et al. 2011). The main features of these models are:

- The ISM is described as a single gas phase and is modeled with SPH method.
- Gas cools to below 10^4 K and collapses gravitationally to massive clumps which can be regarded as giant molecular clouds (GMCs).
- Stars form inside these cold dense regions if the particular star-formation (SF) criteria are satisfied.

Figure 1.2 gives a demonstration for the single-phase model, gas and stars are linked together via star formation and stellar feedback, and are coupled dynamically via gravity. In these single gas phase models, the "Katz" star formation criterion (Katz 1992) is commonly used as the standard star formation recipe. The features can be summarized as follows (Saitoh et al. 2008):

- The physical density is greater than $n_{\mathrm{H}} = 0.1\mathrm{cm}^{-3}$.
- The temperature is lower than $\simeq 10^4$K.
- The velocity field is converging.

Furthermore, some single-phase models requires the local star forming region to be Jeans unstable (Stinson et al. 2006). From the GMCs (low temperature and high density particles in the model), a new star particle is born with a mass usually taken as a fraction of the mass of gas particle, such that multiple generations of stars are formed. By various types of stellar feedback, metals and energy (Springel & Hernquist 2003; Stinson et al. 2006; Valcke et al. 2008; Schroyen et al. 2011; Rahimi et al. 2011) are ejected from stars to the surrounding ISM. GMCs are heated and disrupted and thus regulate the SF process. With arbitrarily adjusted SF recipes and stellar feedback schemes, single-phase models can successfully reproduce some observational features, such as the Schmidt-Kennicutt relation (Schaye & Dalla Vecchia 2008; Hopkins et al. 2012).

Nevertheless, single-phase descriptions cannot allow for the complex multi-phase structure of ISM and its various gas physical and dynamical behaviors. The SPH method itself cannot correctly model the interactions between the phases. Moreover, as pointed out by many authors (Thacker et al. 2000; Stinson et al. 2006; Scannapieco et al. 2006; Booth et al. 2007; Stinson et al. 2012), energy feedback of supernovae is usually not sufficiently efficient in such kind of systems. Since the gas density in the star forming region is high, feedback energy will be easily compensated by radiative cooling. As a consequence, gas cools too rapidly, which yields a star-formation rate (SFR) that is too high. Two methods namely thermal and kinetic, are proposed to improve the energy feedback of supernovae with SPH codes (Pelupessy et al. 2004). In the thermal method, the feedback region is assumed to be adiabatic, or in other words, the cooling process is shut off (Gerritsen & Icke 1997; Thacker & Couchman 2000) for a short time. Usually 30 Myr is taken as the cooling shut off time, which is comparable with the the lifetime of 8 $M_{\odot}$ stars. By doing this, the star forming region becomes hotter than the maximum allowed temperature to form stars, thus regulate the star formation process. What's more, the feedback energy injected to the feedback region drives the gas expanding, an outflow is produced in this way naturally. In the kinetic method, a fraction of supernova energy is transferred to the gas particles in the feedback region as kinetic energy directly, while the rest part is still injected to the feedback region as thermal energy. The fraction is a free parameter in the model (Navarro & White 1993; Springel & Hernquist 2003). In the recent single-phase models, the thermal and kinetic methods are usually used together to give a better feedback treatment. However, the over cooling problem has not been fully solved (Stinson et al. 2006; Pelupessy et al. 2006).

1.3.2 The multi-phase model

To overcome these difficulties of a single-phase description, several multi-phase treatments have been developed (Semelin & Combes 2002). One way is to describe the gas particle as a multi-phase or hybrid system (Springel & Hernquist 2003; Murante et al. 2010), which consists of cold clouds in pressure equilibrium with hot gas, with an optional stellar component. Inside one gas particle, when identified as thermally unstable, hot gas will transit to cold clouds at a rate controlled by the radiative cooling rate. The SF process can be modeled as mass transition from cold clouds to stellar component with SFR given by a specific SF recipe, or creating collisionless star particles from the reservoir of cold clouds. Scannapieco et al. (2006) divide the gas particles into hot and cold components according to their temperature and density, and give them different treatment for stellar feedback. Another way is to treat cold clouds as a new phase according to the "sticky" particle method by Theis & Hensler (1993), while the hot gas is still modeled with SPH method (Semelin & Combes 2002; Harfst et al.

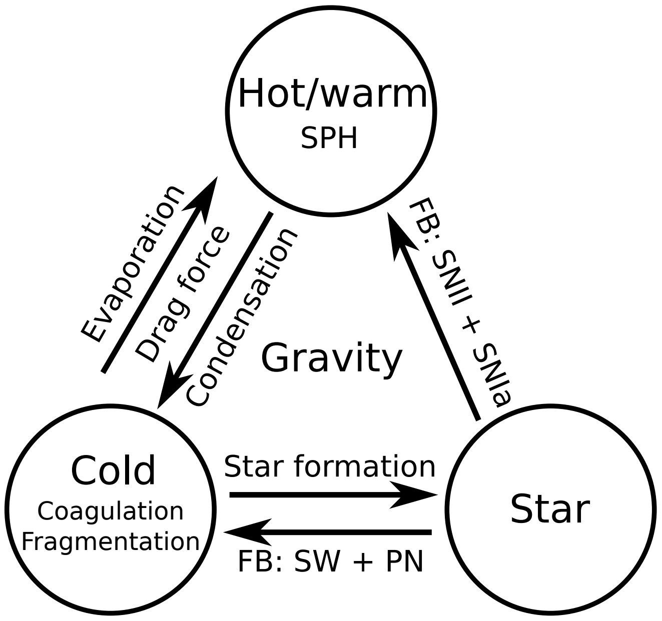


Figure 1.3: Demonstration for a multi-phase model.

2006). By doing this, different phases (hot gas, cold clouds and star particles) are modeled independently, which comes closer to reality because getting rid of any inherent, but artificial single-component mixture of properties and average conditions. Another big advantage of such kind of models is, different components (hot gas, cold clouds and star particles) are treated dynamically separately, but are coupled by means of drag and thermal conduction. Booth et al. (2007) present a multi-phase model, which is the combination of these two ways: When the fraction of molecular clouds in a gas particle reaches the resolution limit of the simulation, a separate sticky particle, representing many sub-resolution molecular clouds, is created, which totally decouples the molecular clouds from the surrounding ambient phase. Grid-based chemodynamical models also exist, such as Theis et al. (1992); Samland et al. (1997); Samland & Gerhard (2003), and a new development based on FLASH: cdFLASH (Mitchell et al. 2012). Figure 1.3 demonstrates the interactions between the phases in our model. Since the multi-phase model is the main focus of this thesis, I will give an elaborate introduction to the implementation of the related physical processes in the next few chapters.

1.4 Physical processes

Numerical programs are designed to run on different architectures (shared memory and distributed memory) and are written in different languages (C, C++, Fortran77, Fortran95, depends on the history of the code), at first glance they have no common features. However, several physical processes must be implemented if the programs are dedicated to the modeling of galaxies. In this section I will give a brief introduction to these processes.

- **Gravity**

Gravity is the dominant force in almost all the astronomical scales. In the non-relativistic case it is simply described by Newton's law of gravitation. However, since it cannot be shielded as the electromagnetic force, gravity interaction must be calculated between every two particles in the system, which yields a computation time of $O(N^2)$ for all N particles. Compared with other local processes, gravity calculation is more time consuming when the particle number is large. Nowadays it has become a special topic in the study of numerical methods: the N-body problem. A set of approximation methods are developed to accelerate its calculation, such as the tree method (Barnes & Hut 1986) that reduces the computation time to $O(N \log N)$ with an adjustable loss of accuracy; the fast multiple method (FMM) that reduces the computation time further to $O(N)$ by calculating the multiple expansion of Helmholtz equation to certain order (Greengard & Rokhlin 1987). I will give an elaborate introduction to gravity calculation together with its GPU acceleration in section 6.1.1.

- **Gas dynamics**

Smoothed particle hydrodynamics (SPH) method is often used for the description of gas in a particle based galaxy formation model. It is first proposed by Lucy (1977); Gingold & Monaghan (1977). Since then this method is widely used in the astrophysical study of almost all the scales: from the large scale structure formation to the small scale planet formation. Also it becomes popular for the modeling of fluid in many other fields of science and engineering. The basic idea of SPH is all physical properties of the fluid field are sampled at the positions of the SPH particles and are distributed with a kernel function $W(r; h)$. Here r is the distance to the particle position; h is the smoothing length, which is often defined in such a way that the kernel function drops to zero at $2h$. To correctly model the shock process during which entropy increases, artificial viscosity is introduced to convert kinetic energy to thermal energy as a creation of entropy. Springel (2010) gives an excellent review for the application of SPH method in astrophysics. The discrete implementation of SPH routines in our model can be found in section 6.1.2.

From another side, it is difficult for SPH method to model gas in such a large density and temperature contrast. In the star forming regions where the density is high and temperature is low, the corresponding Jeans mass is small, which will not be resolved if the resolution of the simulation is not high enough. In this case the system suffers numerical Jeans fragmentation, which renders the modeling of these cold dense regions unreliable. Some methods are proposed to reduce this artificial effect. Robertson & Kravtsov (2008) introduce a density dependent pressure floor into their SPH scheme. The basic idea is, when the corresponding Jeans mass is low, increasing the internal energy of gas particles for SPH calculation. However, this treatment itself is quite artificial, also it is quite difficult to distinguish artificial fragmentation and physical Jeans fragmentation. This method is far from perfect. Actually the behavior of low temperature and high density molecular clouds are more similar with the interaction-less N-body particles instead of the hot diffuse gas as modeled by SPH, if they cannot be resolved properly. In this sense it is more suitable to model the molecular cloud with the "sticky particle" scheme (Theis & Hensler 1993), as we have used in our multi-phase model.

- **Gas cooling**

Gas cooling plays an important role in the process of star formation. In the standard galaxy formation scheme, gas is shock heated to the virial temperature of the host halo. Then gas in the halo center first cools down, since the cooling rate is proportional to the square of gas density. Due to Jeans instability, these low temperature gas fragments and collapses to molecular clouds. Stars form inside these clouds. In section 1.4.1, I will give a detailed review of the modern star formation theory. Since gas transits from fully ionized state to neutral state around 10^4 K, the corresponding cooling rates changes for several orders around this temperature. The dynamic properties of gas are also quite different. Actually this is the main reason that motivates us to divide gas into two phases according to temperature. In the single-phase models, gas can only cool down until 10^4 K. The reason is gas with a temperature below 10^4 K is assumed to be cold gas, whose structure cannot be resolved due to the limited resolution of single-phase model [2]. The gas dynamical time scale is often compared with the cooling time scale. Only those particles that cool fast enough are assumed to collapse and form stars. In the multi-phase models, cold dense gas is modeled as an independent phase, whose structure is resolved analytically. The temperature information is required such that the temperature dependent star formation recipes can be implemented. Therefore we need a cooling curve which covers the range from 10^2 K to 10^7 K.

The cooling rate is a function of temperature and metallicity which consists of the contribution from many elements and various mechanisms at different temperatures. Since it does not take a simple analytical form, usually it is prepared as a two dimensional table for different metallicities and temperatures. The cooling rates above and below 10^4 K are usually calculated separately. Above 10^4 K, the cooling is mainly contributed by continuum emission (free-free, two-photon and recombination radiation) and line emission of specific elements. In our model we use the cooling rate calculated by Böhringer & Hensler (1989). The cooling rate table generated by Sutherland & Dopita (1993) is also often used in many models. Below 10^4 K, cooling is provided by collisions of electrons and atoms together with contribution from H_2 and HD (Dalgarno & McCray 1972; Hollenbach & McKee 1979; Lipovka et al. 2005). The cooling curve used in our model will be introduced in detail in section 2.4.

- **Star formation**

The "star formation" here refers to how star particles are created from the gas particles (single-phase model) or from the molecular clouds (multi-phase model). In single-phase models, the "Katz" recipe is usually taken as the standard star formation recipe (Katz 1992). The common features are summarized in section 1.3.1. In the multi-phase model, various recipes are proposed, some of them are summarized in section 2.5.2. Due to the limited resolution, star particles are not real individual stars, but single stellar populations (SSPs). The stellar mass inside the SSP is distributed according to a given initial mass function (IMF), e.g. the Kroupa

[2]In the galaxy scale simulation, the maximum density of single-phase model is around $1/\mathrm{cm}^3$, see Stinson et al. (2006) as an example. However, the density of molecular clouds is well above $100/\mathrm{cm}^3$. To resolve the structure, we need a particle number of at least 2 orders higher than the current value, which is not realistic at present.

IMF (Kroupa et al. 1993) used in our model. Actually as the programs are parallelized and GPU accelerated, it is possible to resolve individual stars if the resolution is high enough. We will discuss the possibility of creating individual stars in chapter 5. The related theory of star formation and the origin of IMF is discussed in section 1.4.1 and 1.4.2 of this chapter.

- **Stellar feedback and chemical enrichment**

Stellar feedback is the process that eject mass and energy from stars to the surround interstellar medium, which is important in regulating the star formation process and determining the galaxies' chemical evolution. Several kinds of stellar feedback are implemented in galaxy formation models. Massive stars (> 8 M$_\odot$) are featured for their strong stellar wind and UV radiation during their short lifetime (typically less than 100 Myr). They will end their life as type II supernovae or black holes. Intermediate mass stars (0.8 M$_\odot$ to 8 M$_\odot$) strip their outer layers during the AGB (asymptotic giant branch) phase and end as planetary nebulae (PN). The binary systems in this mass range are the progenitors of type Ia supernovae.

The stellar wind and UV radiation of massive stars together with two types of supernovae deposit energy to the interstellar medium of feedback regions. In the single-phase models, the deposited energy will quickly radiate away instead of converting to the ISM's kinetic energy due to the high density of gas in these regions. This is the commonly known as the "over-cooling" problem among single-phase models (Stinson et al. 2006). The rise of this problem is due to the limited resolution to resolve the feedback region. The main idea to solve it is to convert part of the feedback energy to kinetic energy of gas directly (Hopkins et al. 2011). However, such kind of treatment is quite artificial, and introduces extra parameters.

Various feedbacks have different chemical features. Type II supernova produces a large amount of α-elements, iron, and r-process elements. Type Ia is the major source of iron as well as r-process elements. Mass ejecta in the AGN phase and stellar winds mainly contribute to chemical enrichment of carbon, nitrogen and oxygen. In the single phase models, all the chemical elements by different feedback sources are deposited to the only gas phase. The unique chemical features of different gas phases are lost due to the mixture of gas. In the multi-phase model, the hot/warm and cold phases receive the chemical elements from different types of feedbacks. The discrepancies of chemical evolution between the gas phases are clearly shown. This is one of our motivation to develop the multi-phase models.

1.4.1 Star formation in clusters

Theories of star formation and observations indicate that stars frequently form in the clustered environments. The study of Adams & Myers (2001) suggests that the most clustered star formation occurs in clusters with between 10 and 100 stars. Stellar clusters are born embedded in the molecular clouds, their formation and early evolution are only visible at infrared wavelengths, since they are heavily obscured by dust (Lada & Lada 2003). However, the rapid development of observation instruments in the recent decades has made it possible to investigate the inner regions of molecular clouds where star formation occurs. Observations of the Orion B cloud suggest that 96% of infrared resources belong to the cluster. While a more common fraction of stars located in clusters is 50-70%. What's more, a strong association is found between the location of clusters and of dense, massive molecular cloud cores. All of these suggest that the formation of star cluster plays a fundamental role in the study of star formation (Clarke et al. 2000).

Star formation is traditionally divided into two regimes according to the Kelvin-Helmholz time, which is the time scale to radiate away a significant fraction of the thermal energy of a contracting star:

$$t_{\rm KH} = Gm^2/RL, \tag{1.1}$$

where m, R and L are the mass, radius and luminosity of the star, respectively. Low mass stars form in a time scale shorter compared to $t_{\rm KH}$, while high mass stars form in a time longer than $t_{\rm KH}$. The 8 $M_\odot$ is adopted as the mass criterion to distinguish the low and high mass stars, as their formation process are different, and will end their life in different ways, also. Low mass stars form from gravitationally bound cores with masses in the order of the Jeans mass. They undergo extensive pre-main-sequence evolution in the Hertzsprung-Russell diagram, before they cease accreting and move to the main sequence (McKee & Ostriker 2007). The extreme of low mass stars are brown dwarfs, which fail to ignite hydrogen burning in the core since their mass is less than the H-burning limit of ~ 0.08 $M_\odot$. They are interesting objects for the study of the earliest stage of star formation. However, they are excluded from the IMF adopted in this thesis since we only consider the main sequence stars with a mass higher than 0.08 $M_\odot$.

High mass protostars are characterized by Kelvin-Helmholtz times that are shorter than the accretion time, so that they undergo nuclear burning while still accreting. This leads to two physical processes which do not apply to low mass stars: radiation pressure and photoionization. The radiation pressure strongly affects the dynamics of massive star formation. The infall of gas will be stopped if the outward radiation pressure balances gravity, which prevents the formation of massive stars. However, this can be overcome by the combined action of disk formation, protostellar outflows, and radiation hydrodynamic instabilities in the accreting gas. Massive stars ionize their surroundings into HII regions, which corresponds to the transition from cold phase to the hot/warm phase in the multi-phase models. Molecular clouds might be totally ionized by photoionization during the formation of massive stars. Stars are formed inside the molecular clouds with an efficiency of $\sim 5\%$. The rest mass of the molecular clouds returns to the diffuse ISM and starts star formation of the next circle (McKee & Ostriker 2007).

1.4.2 The origin of initial mass function

The initial mass function (IMF) describes the distribution of stellar mass in a newly formed stellar populations. Salpeter (1955) first describes the IMF in a power law form: $\mathrm{d}N = \xi(m)\mathrm{d}m = km^{-\alpha}$, in which $\mathrm{d}N$ is the number of stars in the mass range m, $m + \mathrm{d}m$, k is a normalization factor. He obtains a power law index $\alpha = 2.35$ for $0.4\ M_\odot \lesssim m \lesssim 10\ M_\odot$, which is known as the "Salpeter IMF" (Kroupa et al. 2011). Current observations suggest that the IMF is quite similar in many different locations of the Milky Way (McKee & Ostriker 2007). In the standard IMF of Kroupa (2001), there is evidence for a change in power law index at only two masses: near 0.5 $M_\odot$ and near 0.08 $M_\odot$. The multi-part power law IMF is described as:

$$\xi(m) \propto m^{-\alpha_i}, \tag{1.2}$$

with

$$\begin{aligned}
\alpha_0 &= 0.3 \pm 0.7, \quad 0.01\ \mathrm{M}_\odot \leq m < 0.08\ \mathrm{M}_\odot, \\
\alpha_1 &= 1.3 \pm 0.5, \quad 0.08\ \mathrm{M}_\odot \leq m < 0.50\ \mathrm{M}_\odot, \\
\alpha_2 &= 2.3 \pm 0.3, \quad 0.50\ \mathrm{M}_\odot \leq m < 1.00\ \mathrm{M}_\odot, \\
\alpha_3 &= 2.3 \pm 0.7, \quad 1.00\ \mathrm{M}_\odot \leq m.
\end{aligned} \tag{1.3}$$

Note that in the above form although the power law index α is kept unchanged above 0.5 $\mathrm{M}_\odot$, its uncertainty becomes larger above 1.0 $\mathrm{M}_\odot$. In the simulations presented in this thesis, we use a relatively old form of Kroupa et al. (1993), in which α_3 takes the value 2.7, with the lower and upper mass limits set to 0.08 $\mathrm{M}_\odot$ and 100 $\mathrm{M}_\odot$. The rise of slope at high mass end will not affect our results too much, but will reduce the number of type II supernova events contributed by massive stars.

The key question for the theoretical study of star formation is to understand the origin of IMF and explain its main properties (McKee & Ostriker 2007), such as the Salpeter power law slope at high mass, the break and turnover below $\sim$ 1 $\mathrm{M}_\odot$, the upper mass limit on stellar mass $\sim$ 150 $\mathrm{M}_\odot$, the strong correlation between the mass of the most massive star and the total mass of a star cluster (Kroupa et al. 2011; Peters et al. 2010), and the universality of these features over a wide range of star-forming environments, independent of the mean density, turbulence level, magnetic field strength (Kroupa et al. (2011) has demonstrated that the IMF depends on the star formation rate density and on the metallicity of stars).

The core mass functions (CMFs) of cold clump cores derived from many independent studies via different methods are in good agreement with each other (Johnstone et al. 2006; Lada et al. 2008), and are remarkably similar with the stellar IMF (McKee & Ostriker 2007), which suggests that the cores we detect may be the direct progenitors of individual stars (Motte et al. 1998). Therefore the monolithic collapse models are proposed to explain the origin of IMF. It assumes a one-to-one relation between the two distributions. The inefficiency of single-star formation explains the shifts from CMF to IMF (Peters et al. 2010). Matzner & McKee (2000) predicted that the efficiency of a single star-formation event in an individual core is $\epsilon_{\rm core} \approx 0.25 - 0.7$, and is not sensitive to the parameters involved. With this assumption, the explanation of IMF shifts to the explanation of CMF. Various numerical methods have been used to study the mass distribution of molecular core, such as the SPH and grid based codes including magnetic fields, with an isothermal equation of state (Klessen & Burkert 2001; Padoan et al. 2007), or simulations with non-isothermal equations of state (Li et al. 2003). By analyzing the simulation results with clump-finding algorithms, the derived CMFs are generally dominated by a low-mass end and show high-mass tails which are consistent with the slope of of Salpeter IMF. However, when come to details, the mass distribution depends on the adopted clump-finding algorithms, and on the specific parameters of the simulations (McKee & Ostriker 2007).

The monolithic collapse model links the CMF and IMF in a simple and natural way. However, some caveats still exist (Peters et al. 2010). Many of the pre-stellar cores found in observations appear to be stable and are unlikely to be the star formation site. What's more, some hydrodynamic simulations indicate that massive cores should fragment into many stars rather than collapsing monolithically (Dobbs et al. 2005; Clark & Bonnell 2006). By keeping this in mind, another kind of model, named as competitive accretion, provides a different picture for the explanation to the origin of IMF, especially for its high-mass end (Bate & Bonnell 2005; Bonnell & Bate 2006). In this model, the origin of the peak in IMF is similar with the monolithic collapse model, which is determined by the Jeans mass of pre-stellar clouds. However, all the gas will collapse and form protostars with roughly the same Jeans mass, instead of fragmenting by following the core mass spectrum, with each core collapses to a single star as proposed by monolithic model. Then these protostars accrete and compete for gas in the star forming regions. Stars in the center of the cluster will grow larger, since the potential well is deeper, more gas is hosted there. Those ejected from the cluster center remain low mass. In this sense, the similarity between CMF and IMF is just an illusion, since they have totally different origin. The competitive accretion model can reproduce the correlation between the mass of the most massive star and the total mass of the cluster: $M_{\rm max} \propto M_{\rm stars}^{2/3}$ (Bonnell et al. 2004; Peters et al. 2010), which is suggested by observation.

Although the origin of IMF has not been fully understood, our current knowledge from observations and numerical simulations already allows us to build a general picture. We know that a number of physical processes play important roles in the process of star formation and affect the stellar mass distribution, such as gravity, gas accretion, turbulence, magnetic fields, feedback from young stars and other semi-random processes such as dynamical ejections (Bonnell et al. 2007). The peak in the IMF is due to the gravitational fragmentation and are set by the Jeans mass for fragmentation. The broadened peak can be explained as the dispersion in gas densities and temperature when fragmentation occurs. Turbulence is a necessary condition in the molecular clouds which facilitates the fragmentation. Lower-mass stars most likely form through the gravitational fragmentation of a collapsing region. They are ejected from the cluster center via gravity interactions and are not fueled by the accretion flow of gas, thus remain low mass. Massive stars resides in the cluster center, accumulate mass via competitive accretion. This produces the observed mass segregation in the young stellar cluster, and also the relation between the mass of the most massive star and the total mass of star cluster.

1.5 Thesis outline

The main part of this thesis investigates the modeling of galaxies using multi-phase method, and the related numerical issues. In chapter 2, the physical processes implemented in the multi-phase model are introduced. In chapter 3, the evolution of a dwarf galaxy is presented using the multi-phase model described in previous chapter. In chapter 4, I introduce the transition process, by which the hot/warm gas collapses to cold clouds, as an modification to the traditional multi-phase model. In chapter 5, I introduce the individual star formation model, which mimics the individual star formation process according to IMF. It reproduces the life cycle of interstellar medium and the process of star cluster formation in turbulent velocity field. In chapter 6, all the numerical techniques involved in the previous models are discussed, including the parallelization of the program and the GPU acceleration of the most time consuming parts. A summary of the thesis and an outlook are presented in the last chapter.

"Tao that can be described is not universal and eternal Tao. Name that can be named is not universal and eternal Name."

Tao Te Ching, Lao Tse

2 Introduction to multi-phase model

In this chapter I present our newly developed multi-phase chemodynamic code for the simulation of galaxy evolution. A basic introduction to the multi-phase model has been give in section 1.3.2. We here focus on the introduction to the detailed implementation of the physical processes. We use a two components description of the ISM. The basic idea is to add a cold cloud component to the hot/warm gas represented by SPH. Cold clouds are modeled using the sticky particle scheme. The hot/warm and cold components can exchange mass, energy, and momentum via condensation, evaporation and drag force. The cold clouds can coagulate when they collide, and fragment when they are disrupted by stellar feedback. Mass and energy are ejected to the surrounding hot/warm and cold components via different types of stellar feedback. In this chapter, I will introduce the physical processes involved in the model in detail. The simulation of a dwarf galaxy based on current model will be presented in the next chapter.

2.1 Hot/warm gas

In our model, the hot/warm component refers to the neutral and ionized gas with temperature above 10^3 K. In some multi-phase models, such as Harfst et al. (2006), warm/hot gas is further divided into warm (from a few 10^3 K up to 10^4 K) and hot ($> 10^4$ K) components. Since both hot and warm gas behave dynamically similar as diffuse ISM and are described using the SPH method, they are treated uniformly as hot/warm component in our current model. Note that they are still different since the cooling rate of the warm component depends on the ionization rate fraction, and is several orders of magnitude lower compared with the fully ionized hot component. The hot/warm gas is well described by the SPH method because of their low density and relatively strong self interaction. Section 6.1.2 give a detailed introduction to SPH method and its acceleration with GPU.

In the multi-phase description, hot/warm particles can exchange mass with cold clouds via condensation and evaporation. In our dwarf galaxy model presented in the next chapter, hot/warm particles accumulate mass efficiently via evaporation. For some of the particles their mass can increase up to 10^6 $M_\odot$, which is three orders of magnitude higher than the original mass. The increase of particle mass, however, leads to a lower SPH resolution. To keep the original resolution, we introduce an upper mass limit and a particle splitting mechanism for hot/warm gas. A hot/warm particle will be split if its mass reaches the upper mass limit. This split number is arbitrary and set to 4 in the present model. New particles are placed randomly around 1/4 of old particle's smoothing length, with the same velocity and the temperature of the old one. Our treatment of the SPH particle splitting is simple, a more detailed analysis can be found in Kitsionas & Whitworth (2002).

2.2 Cold clouds

We follow the two-component gas description for the ISM by Theis et al. (1992) and Samland et al. (1997). The basic idea of a multi-phase treatment is to add a cold cloud component to the hot/warm gas which is described by SPH. Because of the cold clouds' relatively high density ($> 10^2$ cm^{-3}) and low temperature, their mutual collisional interactions are weak and can be treated as N-body particles by a kind of "viscosity" (Theis & Hensler 1993).

For the description of cold clouds we use the *mass-radius* relation (Rivolo & Solomon 1988)

$$h_{\rm cl} = 50\sqrt{\frac{m_{\rm cl}}{10^6\ {\rm M}_\odot}}({\rm pc}). \tag{2.1}$$

which is mainly based on observations and some theoretical work (Larson 1981). Several authors (Theis & Hensler 1993; Harfst et al. 2006) have already used this relation to model cold clouds successfully. According to equation 2.1, a typical cloud with the mass of 10^5 M$_\odot$ corresponds to a size of 10 pc. Although the size is small, cold clouds still have the chance to collide with each other, and by this coagulate to the larger cloud. When the size of the cloud is too large, the outward pressure will not be strong enough to balance the internal self gravity. Clouds will collapse due to Jeans instability and form stars in the dense region. Finally the whole object fragments into small pieces because of stellar feedback. In the next few sections, we will describe these processes in detail.

2.2.1 Fragmentation

In our model, the fragmentation of clouds is triggered by stellar feedback. When a star particle forms via Jeans instability or other mechanism depending on the specific SF recipes, its energy release will disrupt the cloud into smaller fragments (Harfst et al. 2006). The time for disruption is determined by the energy release rate: the energy release by means of stellar winds (SW) and supernovae type II (SNeII) drives an expanding shell. The cloud will be disrupted when the shell reaches the edge of cloud. The radius and expansion velocity of the shell can be modeled as a function of time and energy release rate (Brown et al. 1995; Theis et al. 1998) as:

$$r_{\rm sh} = 0.961 \cdot \left(\frac{\dot{E}}{\rho_1}\right)^{0.25} \cdot t^{\,0.75} \tag{2.2}$$

$$v_{\rm sh} = 0.736 \cdot \left(\frac{\dot{E}}{\rho_1}\right)^{0.25} \cdot t^{\,-0.25} \tag{2.3}$$

Here ρ_1 as the environmental density is given by $\rho_1 = m_{\rm cl}/h^2_{\rm cl}$. In the code we fragment the cloud into four pieces symmetrically and give them the velocity of the expanding shell at that time. In this process part of the feedback energy, which is $\lesssim 5\%$ and depends on the mass of cold cloud and the mass of embedded star cluster, transforms to the kinetic energy of the newly formed clouds. The rest will be returned to the surrounding cold and hot media (SW energy to cold phase, SN energy to hot phase) as thermal energy. The mass ejected by SNeII will be added to the surrounding hot medium.

2.2.2 Coagulation of clouds

Our implementation of coagulation follows the description by Theis & Hensler (1993). Its basic idea is to assign a collisional cross section to each cloud particle and to check whether those of two particles overlap when they approach each other. In general, this check is performed in two steps: At first, all particles are found within a maximum radius r_s around the target particle p. This radius must be larger than the typical distance traveled within the next time step by a particle with a mean particle velocity:

$$r_{s,p} = 2\Delta t\,\sqrt{2}v_{\mathrm{vir},p} = \Delta t\,\sqrt{8/3\phi_p}, \tag{2.4}$$

where Δt is the next time step and ϕ_p the gravitational potential at the position of particle p. In a second step, particles found in the first step are tested by linear extrapolation of their orbits, if their separation from target particle will become less than the sum of the cross-section radii during next time step Δt.

The collisional cross section is then expressed as:

$$A_{\mathrm{cr}} = \eta_{\mathrm{ov}}^2 \cdot \pi h_{\mathrm{cl}}^2 \cdot \left(1 + \frac{2G(m_1 + m_2)}{\eta_{\mathrm{ov}} h_{\mathrm{cl}} v_{1,2}^2}\right), \tag{2.5}$$

where m_1 and m_2 are the masses of two possibly colliding clouds, $v_{1,2}$ is their relative velocity so that the factor in brackets extends the geometrical cross-section by gravitational focusing. Scalo & Pumphrey (1982) pointed out that not each collision evolves fully inelastically due to the incomplete overlap of the clouds. Here we use the factor $\eta_{\mathrm{ov}} = 0.2$ to reduce the geometrical cross section. Moreover, we further test if the orbital angular momentum of the two colliding clouds exceeds the maximum possible spin of the compound object: $m_1 m_2/(m_1 + m_2) \cdot b\ v_{1,2} \le c_{\mathrm{ang}} L_{\mathrm{max}}$, in which b is the impact parameter of the collision and $c_{\mathrm{ang}} = 1$, the maximum angular momentum is written as:

$$L_{\mathrm{max}} = \int \rho(\mathbf{r}) v_{\mathrm{circ}}\, r \sin\theta\, \mathrm{d}\mathbf{r} = \frac{8}{21} \cdot \sqrt{G m_{\mathrm{cl}}^3 h_{\mathrm{cl}}}. \tag{2.6}$$

When all the above criteria are fulfilled, coagulation will occur. The new cloud will be positioned at their center of mass and be given the center's velocity as its new one. Due to momentum conservation the excessive kinetic energy will be converted to thermal energy. By introducing fragmentation and coagulation, the cloud system evolves in a self-regulated way, which helps avoiding any dependence on initial mass configuration.

2.3 Interaction between cold and hot/warm phases

In our multi-phase description, the typical density, temperature and velocity of the hot/warm phase are 0.1cm^{-3}, 10^5 K and 100 km/s, respectively, while for cold clouds these values are 100cm^{-3}, 10^3 K and a few km/s. When the two phases contact and interact with each other, they tend to smear out these physical discontinuities and to exchange momentum by various physical processes. In our model, these are dynamical drag and thermal conduction (condensation and evaporation).

2.3.1 Thermal conduction (condensation and evaporation)

We describe the process of thermal conduction by implementing the analytical formula by Cowie et al. (1981) which result in condensation or evaporation. The basic parameter controlling these process is σ_0, which represents the ratio between the electron mean free path λ_k and the cloud size $h_{\rm cl}$:

$$\sigma_0 = \left(\frac{T_{\rm hot}(\rm K)}{1.54 \times 10^7}\right)^2 \frac{1}{\Phi n_{\rm hot}(\rm cm^{-3}) h_{\rm cl}(\rm pc)} \tag{2.7}$$

Here $T_{\rm hot}$ and $n_{\rm hot}$ are temperature and number density of the hot/warm gas that surrounds the target cold cloud. If $h_{\rm cl}$ is smaller than λ_k ($\sigma_0 > 1$), evaporation occurs. Vice versa, if the temperature of surrounding hot gas is low or the cloud size is large, the cooling length scale becomes shorter than the cloud size and the conditions switch to condensation. We here choose $\sigma_0 = 0.03$ as transition value from evaporation to condensation. The condensation (positive sign)/evaporation rate is given by:

$$\frac{{\rm d}m_{\rm cl}}{{\rm d}t}({\rm kg/s}) = \begin{cases} 0.825 \cdot T_{\rm hot}^{5/2} h_{\rm cl} \sigma_0^{-1} & \sigma_0 < 0.03 \\ -27.5 \cdot T_{\rm hot}^{5/2} h_{\rm cl} \Phi & 0.03 \le \sigma_0 \le 1 \\ -27.5 \cdot T_{\rm hot}^{5/2} h_{\rm cl} \Phi \sigma_0^{-5/8} & \sigma_0 > 1 \end{cases} \tag{2.8}$$

Φ represents the effect a magnetic field has on reducing the mean free path of charged particles and is set to 1, since we neglect it here. In addition to mass exchange between two gas phases, we also include the momentum transfer by the condensed/evaporated matter. Evaporation of cool clouds within hot gas flows lead to the well-known mass load, that hampers e.g. galactic outflows. Furthermore, the added cold material increases the density, reduces the specific thermal energy, and, by this, enhances the cooling of hot gas. Both diminish the pressure which drives an outflow.

2.3.2 Cloud dragging

The different dynamics of the two gas phases leads to a drag force acting on clouds in a homogeneous surrounding hot medium with their relative velocities $\mathbf{v}_{\rm cl} - \mathbf{v}_{\rm hot}$ according to

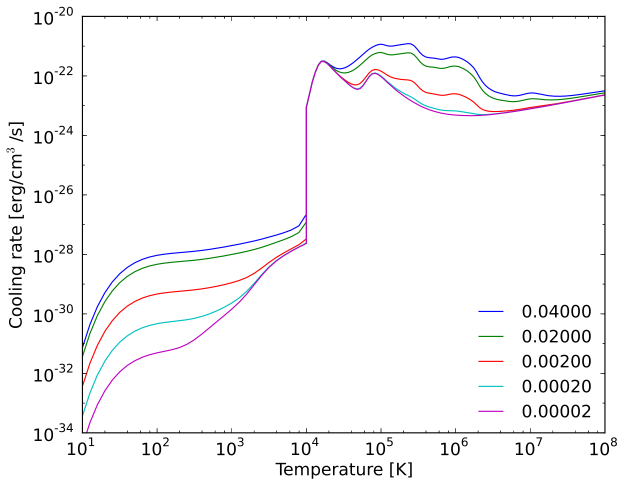


Figure 2.1: Cooling curves as a function of temperature and metallicity.

Shu et al. (1972) of

$$\mathbf{F}_\mathrm{D} = -C_\mathrm{D} \cdot \pi h_\mathrm{cl}^2 \rho_\mathrm{hot} \cdot |\mathbf{v}_\mathrm{cl} - \mathbf{v}_\mathrm{hot}| \cdot (\mathbf{v}_\mathrm{cl} - \mathbf{v}_\mathrm{hot}). \tag{2.9}$$

Here the drag coefficient C_D is the ratio between the effective cross section of a cloud and its geometrical one πh_cl^2. We set it to 0.1, which is consistent with previous works (Harfst et al. 2006; Theis et al. 1992; Samland et al. 1997). Some of our simulations suggest that the coefficient can be different from 0.1, see section 3.3.4 for a discussion.

2.4 Gas cooling

The cooling in the code depends on the metallicity, density and ionization fraction of the gas. The elements which we take into account for calculating the cooling rate are H, He, C, N, O, Ne, Mg, Si, S, Fe, since they are the most abundant elements released during the stellar evolution and are considered in the chemical evolution of galaxies. Above $\sim 10^4$ K, the radiative cooling function is calculated by the metal and temperature dependent prescription of Böhringer & Hensler (1989) of the hot plasma in thermal and ionization equilibrium, which consists of two parts: the continuum emission and the line emission processes. The continuum emission includes the contribution from free-free, recombination and two-photon radiation.

At temperatures below 10^4 K, the cooling is provided by collisions of thermal electrons and hydrogen atoms with singly ionized or neutral metals plus contribution from collisionally excited hydrogen atoms (Dalgarno & McCray 1972). Without detailed element abundance and ionization studies the ionization fraction ($X_\mathrm{e} = n_\mathrm{e}/n_\mathrm{H}$) at low temperatures is a free parameter in the code. We set the X_e to 10^{-4} which is suitable for low metallicity and primordial gas at low temperatures and early epoch of DGs formation which we consider in our reference model. A contribution of H_2 and HD was added to the cooling function based on the theoretical calculation of Hollenbach & McKee (1979) and Lipovka et al. (2005). Their number fractions with respect to hydrogen are 10^{-5} and 10^{-8}, respectively. To calculate the metallicity dependence of the cooling rates we use the abundance distribution in molecular clouds at solar metallicity as determined from observations (Williams et al. 1998) and scale them to the total metallicity of the gas. The total cooling rate is calculated as

$$\Lambda(T, Z) = \Lambda_{\mathrm{Electron}}(T, Z)X_{\mathrm{e}} + \Lambda_{\mathrm{Atom}}(T, Z) + \Lambda_{\mathrm{Molecular}}(T) \tag{2.10}$$

Figure 2.1 demonstrate the cooling curves used in the model as a function of temperature and metallicity. One may notice the huge jump of cooling rate at 10^4K, around which gas transits from fully ionized plasma state to the neutral state. The actual transition happens in range of 5000 K to 10^4 K instead of as steep as in the figure. In our model the cooling curves below and above 10^4 K are calculated separately, and the two parts are not connected together using some smoothed curves. This will not affect our results significantly, since the range of transition temperature is so narrow. The jump also indicates the necessity of a multi-phase description: the physical properties of ISM below and above 10^4 K are so different, which cannot be described precisely using SPH method uniformly. For the hot part ($> 10^4$ K), cooling is dominated by free-free radiation of H and He nuclei. When the metallicity is reduced to less than 1% of the solar value, continuum emission dominates the cooling process. The cooling curves do not change significantly when the metallicity is further reduced. The two peaks at $\sim 1.5 \times 10^4$ K and $\sim 9 \times 10^4$ K correspond to H and He linepeak, which are therefore not affected by the variation of metallicity. To simplify calculations we prepare tables of cooling rates and interpolate for any metallicity in the range of 10^{-5} to 2 times the solar abundance, over the large temperature range ($10 \leq T \leq 10^8$).

2.5 Stars

2.5.1 Single stellar population (SSP)

Star particles in our model are actually star clusters formed at the same time and considered in the code as single stellar populations (SSP) instead of single stars as usually assumed. Stars in each SSP share the same metallicity and age. Their mass are distributed by the Kroupa initial mass function (IMF) (Kroupa et al. 1993):

$$\Phi(m) = \begin{cases} 0.035m^{-1.3} & \text{if } 0.08 \leq m < 0.5, \\ 0.019m^{-2.2} & \text{if } 0.5 \leq m < 1.0, \\ 0.019m^{-2.7} & \text{if } 1.0 \leq m < \infty. \end{cases} \tag{2.11}$$

Here m is in unit of solar mass, the lower and upper mass limits are fixed as 0.08 $\mathrm{M}_{\odot}$ and 100 $\mathrm{M}_{\odot}$, respectively. Stars return mass and energy to the surrounding cold and hot/warm media by means of four types of feedback (SNeII, SNeIa, SWs, and planetary nebulae (PNe)). For the calculation of the stellar lifetime we use the metal-dependent approximation based on the theoretical stellar tracks of the Padova group (Raiteri et al. 1996):

$$\log t_{\star} = a_0(Z) + a_1(Z) \log m + a_2(Z)(\log m)^2, \tag{2.12}$$

where $t_\star$ is in unit of year and m in unit of solar mass. The three coefficients are give by:

$$\begin{aligned}
a_0(Z) &= 10.13 + 0.07547 \log Z - 0.008084(\log Z)^2, \\
a_1(Z) &= -4.424 - 0.7939 \log Z - 0.1187(\log Z)^2, \\
a_0(Z) &= -1.262 + 0.3385 \log Z + 0.05417(\log Z)^2.
\end{aligned}$$

The above formula gives a good approximation of stellar lifetime for stars with mass m between 0.6 $M_\odot$ and 120 $M_\odot$ and metallicity Z between 7×10^{-5} and 3×10^{-2}.

2.5.2 Star formation

Stars are assumed to be usually formed from the collapsing and fragmenting cold clouds. In our model, we use the standard Jeans instability criterion to trigger SF. Meanwhile, a pressure dependent criterion (Köppen et al. 1995; Elmegreen & Efremov 1997; Harfst et al. 2006), controlled by the local ISM properties (mass of the molecular cloud, pressure of ambient diffuse gas) and a self-regulated temperature-dependent SF criterion Theis et al. (1992) are also implemented for comparison. The basic idea is that stars are born inside cold clouds as described in section 2.2.1 of this paper. The star will reside in the cloud before it is destroyed by stellar feedback.

- **Jeans criterion**

For the standard Jeans criterion, at first, we have to compare the Jeans length and the size of the cold cloud $h_{\rm cl}$ (Eq. 2.1) for each cloud particle:

$$\lambda_{\rm J} \equiv c_{\rm cl} \sqrt{\frac{\pi}{G\rho_{\rm cl}}}, \tag{2.13}$$

only those with $\lambda_{\rm J} < h_{\rm cl}$ will collapse due to Jeans instability and thus form stars. The derivation of the maximum SFR uses the picture that all the mass $M_{\rm J}$ inside the Jeans volume will convert to stars during a free-fall timescale $\tau_{\rm cl}^{\rm ff}$:

$$\frac{{\rm d}\rho_*^{\max}}{{\rm d}t} = \frac{M_{\rm J}}{\tau_{\rm cl}^{\rm ff} V_{\rm J}} = \frac{4}{3}\sqrt{\frac{6G}{\pi}}\,\rho_{\rm cl}^{3/2} \tag{2.14}$$

with

$$\tau_{\rm cl}^{\rm ff} \equiv \sqrt{\frac{3\pi}{32G\rho_{\rm cl}}}. \tag{2.15}$$

The actual SFR is set by the maximum one, multiplied by a coefficient ϵ expressing a probability:

$$\frac{{\rm d}\rho_*}{{\rm d}t} = \epsilon \cdot \frac{{\rm d}\rho_*^{\max}}{{\rm d}t} \tag{2.16}$$

ϵ is randomized between 0.01 to 1. For each cold cloud we check if the next SF occurs at least $\tau_{\rm cl}^{\rm ff}$ after the last one. Moreover, a probability is set to further control SFR: for the dwarf galaxy model presented in the next chapter, within each Myr, 0.5% of clouds are allowed to form stars, which corresponds to an inactive timescale of 200 Myr.

- **Pressure dependent criterion**

Köppen et al. (1995) demonstrate a self-regulated star formation model. Harfst et al. (2006) implement the star formation process with a temporally and spatially variable star formation efficiency (SFE) proposed by Elmegreen & Efremov (1997). In their scheme, the SFE of every molecular cloud is determined by its mass and pressure of the surrounding diffuse gas. Usually massive clouds with high pressure of ambient gas yields a high SFE. The specific SFE is obtained by numerically solving the non-linear equation that couples the cloud mass, pressure and SFE. The discrete form of the equation can be found in Appendix A of Harfst et al. (2006).

- **Temperature dependent criterion**

The temperature dependent SF criterion is proposed by (Theis et al. 1992) and adopted in their modeling of an isolated elliptical galaxy. In their recipe the star formation density is expressed as:

$$\rho_{\rm SF} = \eta_{\rm SF}\, \rho_{\rm cl}, \tag{2.17}$$

$\rho_{\rm cl}$ is the density of cold cloud, $\eta_{\rm SF}$ is a temperature dependent coefficient:

$$\eta_{\rm SF} = \begin{cases} 0, & T_{\rm cl} \geq 10^4\ \rm K, \\ \frac{10^4\ {\rm K} - T_{\rm cl}}{99 T_{\rm cl}}, & 10^2\ {\rm K} \leq T_{\rm cl} < 10^4\ \rm K, \\ 1, & T_{\rm cl} < 10^2\ \rm K. \end{cases} \tag{2.18}$$

Apparently, high temperature yields a low SF efficiency. No star is allowed to form when the temperature is above 10^4 K The star formation rate is expressed as:

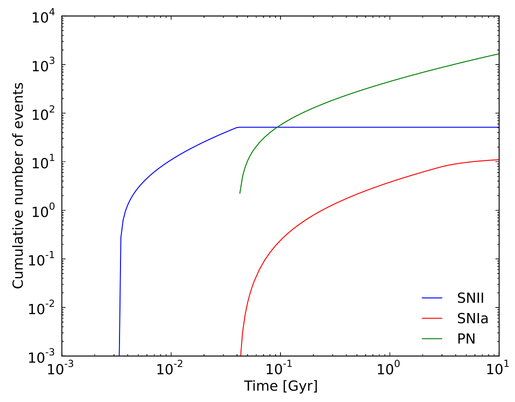
$$\frac{d\rho_*}{dt} = \alpha_{\rm SF} \cdot \rho_{\rm SF}^2, \tag{2.19}$$

with $\alpha_{\rm SF} = 0.55\ {\rm pc}^3/{\rm M}_\odot/{\rm Myr}$.

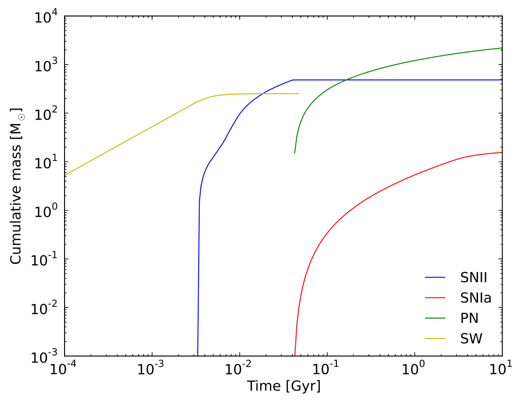
2.5.3 Stellar feedback

As a chemical and energetic stellar feedback we consider the delayed release of individual elements and energy by both, massive stars through SNeII and SWs and intermediate mass stars (SNIa and PNe), respectively. In our chemical enrichment scheme we track eleven elements individually, H, He, C, N, O, Ne, Mg, Si, S, Ca, Fe which also contribute significantly to the cooling rates. Stars release these elements back to the surrounding ISM and subsequent generations of stars are born with different chemical compositions.

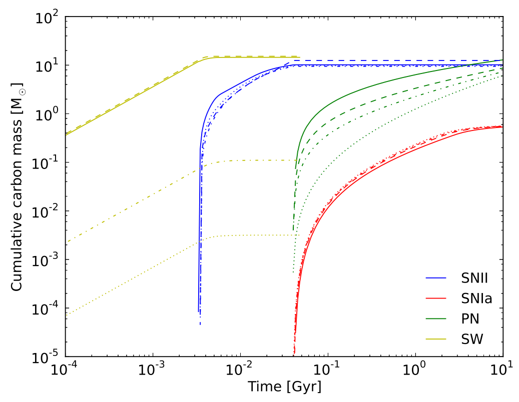
We suggest that all stars are formed in stellar populations and their mass are distributed by the IMF within the mass range of 0.08 $\rm M_\odot$ to 100 $\rm M_\odot$. Massive stars with masses higher than 8 $\rm M_\odot$ release energy in the form of SW until they end their life as SNeII or Black Holes.



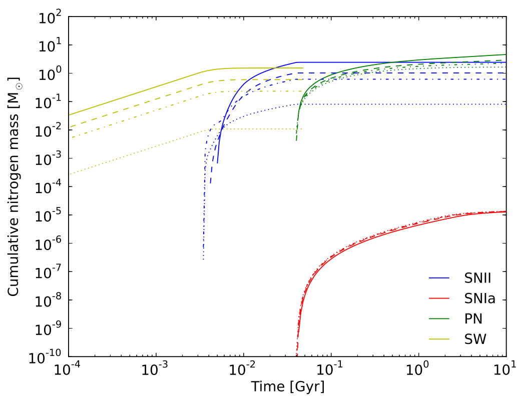
(a) Number of events



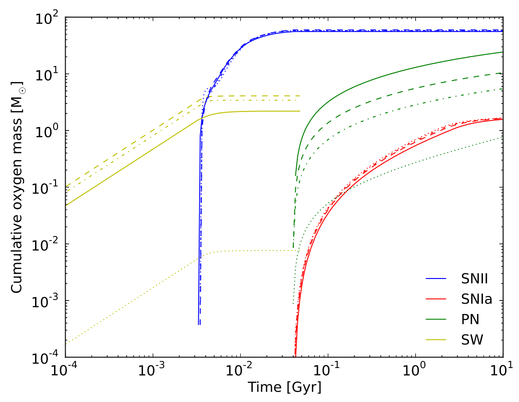
(b) Total mass.



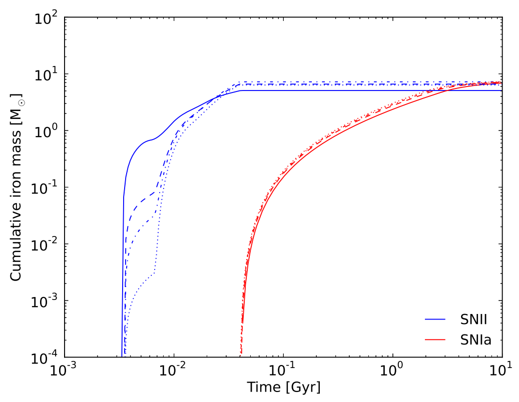
(c) Mass of carbon.



(d) Mass of nitrogen.



(e) Mass of oxygen.



(f) Mass of iron.

Figure 2.2: Cumulative number of events and cumulative ejected mass of chemical elements (C, N, O, Fe) for four different feedback processes (SNII, SNIa, PN, SW) as a function of time. The solid, dashed, dashdot and dotted curves correspond to the metallicity of $Z = 0.02$ (solar value), 0.008, 0.004, 0.0004, respectively. The calculation is based on a single stellar population (SSP) of 10^4 $M_\odot$ in which the stellar mass is distributed by following the Kroupa IMF (Kroupa et al. 1993). The cumulative number of events and cumulative total ejected mass (panel a and b) only exhibit for the solar metallicity, since the deviations of different metallicities are quite small.

Intermediate-mass stars in mass range 0.8 $M_\odot$ – 8 $M_\odot$ which have long main-sequence lifetimes (10^8 yr $\lesssim t \lesssim 10^{10}$ yr) undergo PNe or SNeIa. Low-mass stars with masses less than 0.8 $M_\odot$ do not evolve significantly during a Hubble time and their influence is limited contributing to gravity only and not to the chemical or energetic evolution.

The above stellar feedback processes differ in a number of ways, such as their chemical yields, their energy output, and the timescale on which they act. Because of the short timescale of the feedback processes except SW and ionizing radiation from massive stars, as an approximation we consider the release of the whole ejected mass and energy of a fixed initial stellar mass in a single time step at the end of their lifetimes. Winds and radiation are distributed over multiple time steps according to their lifetimes. Both types of SNe initially deposit their energy release and newly synthesized elements in the hot/warm intercloud medium while SWs and PNe deposit their ejecta into the cold cloud medium. See 1.2 for a demonstration. Processes like condensation and evaporation can provide a chemical and energetic mixing of these two media. All ejected mass and energy are instantaneously and homogeneously distributed in the ISM particles, weighted by the mass of the gas (in case of SNe feedback) or cloud (in case of SWs and PNe) particle that receives them, using the SPH smoothing kernel. Therefore, the feedback is centered on the current position of the star particle and the nearest surrounding gas or cloud particles (N_B) and can be affected until a maximum range of 1 kpc.

Both types of SNe, SWs and PNe have different chemical signatures. SWs and PNe contribute to the galactic chemical evolution mainly by the mass return enriched with C and N and also O in case of SWs and s-process elements in case of PNe. While SNeII release also copious amounts of O, α elements (Ne, Mg, Si, etc.), Fe, and r-process elements on short timescales of massive stars, SNeIa are the major source of iron in the universe as well of r-processes elements but stem from the longer-living binary evolution with White Dwarfs primaries. Because of the time difference between the release of α, r-/s-processes elements and Fe by SNeII, SNeIa, PNe the α/Fe or other element ratios can provide information about the time since the last starburst and evolutionary processes in galaxies.

- **Mass ejecta**

The mass ejected in each element must be obtained from stellar evolution and (explosive) nucleosynthesis models. To calculate the mass feedback we follow Berczik & Petrov (2003) where the authors describe in detail the calculation of the chemical output from SSP. Using the updated software version of this work and nucleosynthetic data for stellar yields we precalculate the tables of metal and time-dependent stellar mass return and mass-weighted by the IMF which we in order to speed up calculations. These feedback tables contain eleven elements returned by one SSP for every stellar evolutionary events (SW, SNII, SNIa, and PN). For the SW and SNeII ejecta we use nucleosynthesis yields from Portinari et al. (1998). For the PNe yields we take data provided by van den Hoek & Groenewegen (1997). The yields for SNeIa are taken from the updated W7 model of the paper by Iwamoto et al. (1999). The mass of each chemical element i ejected by a stellar particle (SSP) with metallicity Z in the time interval $[t, t + \Delta t]$ is:

$$\begin{aligned}\Delta m_i(t, Z) = {} & \Delta m_{i,\mathrm{SNII}}(t, Z) + \Delta m_{i,\mathrm{SNIa}}(t, Z) \\ & + \Delta m_{i,\mathrm{PN}}(t, Z) + \Delta m_{i,\mathrm{SW}}(t, Z),\end{aligned} \tag{2.20}$$

where $\Delta m_{i,\mathrm{SNII}}(t, Z)$, $\Delta m_{i,\mathrm{SNIa}}(t, Z)$, $\Delta m_{i,\mathrm{PN}}(t, Z)$ and $\Delta m_{i,\mathrm{SW}}(t, Z)$ are masses of chemical element weighted by IMF for SNII, SNIa, PN and SW events respectively and taken from feedback tables.

- **Gas heating processes.**

Massive stars heat the ISM by radiation and by SW before they explode as SNII at the end of their lifetime. The energy input by means of Lyman continuum radiation is approximated as a stellar mass-dependent luminosity:

$$L_{Ly} \approx 10^{40}\ (\frac{m_{\mathrm{star}}}{\mathrm{M}_\odot})^6 \text{ photons s}^{-1}\text{ star}^{-1} \tag{2.21}$$

according to Hensler & Burkert (1990) and Theis et al. (1992) with a heating efficiency of $\eta_{Ly} = 10^{-3}$. Energy feedback from massive stars by SWs depends on the metal-dependent mass-loss rate and is approximated from model calculations of Kudritzki et al. (1987) by Hensler (1987) and are described in Theis et al. (1992). The sum of these two stellar energy feedback processes from SSP heat the surrounding cold clouds.

Massive stars in a SSP end after nuclear burning as SNeII. The corresponding energy transfered to the surrounding hot/warm medium as thermal energy in the time interval $[t, t+\Delta t]$ is:

$$\Delta E_{\mathrm{SNII}} = m_{\mathrm{SSP}}\ \Delta N_{\mathrm{SNII}}(t)\ E_{\mathrm{SNII}}, \tag{2.22}$$

where m_{SSP} is the initial mass of the stellar particle and $\Delta N_{\mathrm{SNII}}(t)$ is number of massive stars per unit stellar mass that explode as SNeII within Δt. The energy released by one SNeII is set to 10^{51} erg. Since we create SSP initially embedded into the cold cloud, small part of the SNeII energy, which depends on the cloud mass and the star formation efficiency and is less than 5% maximum, is used to fragment clouds and ends up as the kinetic energy of the newly formed cold fragments as mentioned in Section 2.2.1.

Intermediate-mass stars give rise to PNe that heat the surrounding cold clouds and end as WDs. Those in binary systems undergo SNIa explosions with a large energy release to the hot/warm medium. Energy contribution of SNIa and PN to the ISM is calculated as thermal energy in the time interval $[t, t + \Delta t]$:

$$\Delta E_{\mathrm{SNIa}} + \Delta E_{\mathrm{PN}} = m_{\mathrm{SSP}}\ [\Delta N_{\mathrm{SNIa}}(t)\ E_{\mathrm{SNIa}} + \Delta N_{\mathrm{PN}}(t)\ E_{\mathrm{PN}}], \tag{2.23}$$

where $\Delta N_{\mathrm{SNIa}}(t)$ and $\Delta N_{\mathrm{PN}}(t)$ depend on the IMF-corresponding number of supernovae SNeIa and PNe per unit mass during Δt. The energy released by SNeIa (E_{SNIa})and PNe (E_{PN}) are set to 10^{51} erg and 10^{47} erg respectively. To calculate $\Delta N_{\mathrm{SNIa}}(t)$ we adopt the model proposed by Greggio & Renzini (1983); Matteucci & Greggio (1986).

Figure 2.2 demonstrate the cumulative number of events and ejected mass of chemical elements (C, N, O, Fe) for four different stellar feedback processes (SNII, SNIa, PN and SW) as a function of time in one SSP with a mass of 10^4 M$_\odot$. Stars with mass higher than 8 M$_\odot$ release energy and mass via stellar wind feedback until they end their life as type II supernovae or black holes. The SNII curves start at 3 Myr, which correspond to the lifetime of massive stars with a mass of 40 M$_\odot$. Stars with mass above 40 M$_\odot$ have an even shorter lifetime. They eject energy and mass to the ISM violently via stellar wind feedback, and end their life as black holes. The PN and SNIa curves start at 30 Myr, which corresponds to the life time of stars with a mass of 8 M$_\odot$. Metallicity influences the number of events and total eject mass only a little bit, but has a strong effect on the yields of specific elements.

2.6 Gravity and dark matter potential

In the current multi-phase model, gravity calculation is divided into two parts and is carried out separately. One part is the gravity interactions between particles (self-gravity), which uses the Barnes-Hut tree method (Barnes & Hut 1986) to speedup calculation, as will be explained in section 6.1.1. Another part is the contribution from the dark matter halo which hosts the simulated galaxy. When simulating an isolated galaxy and focusing on the small scale gas processes such as star formation, stellar feedback, etc., the main role of dark matter halo is to provide a density background which contributes to extra gravity that keeps particles bounded in the system. In this sense the gravity contribution of dark matter halo can be modeled in an analytical form instead of with dark matter particles. The advantage of an analytical dark matter potential is, the computing speed is much faster. The spared computing resources can be used for an accurate modeling of gas processes. I have to point out that the particle based description of dark matter is still necessary in some cases, such as in a high resolution simulation, the interaction between baryons and dark matter must be taken into account, since the angular momentum transfer from galaxy to the dark matter halo is very important in keeping the stability of the galaxy disk.

Below I will introduce two types of dark matter density profiles, which are often used in galaxy scale simulations.

- **Burkert profile**

The dark matter density profiles in the inner region of the DM halo is a puzzle which still has not been solved, since the central part is usually dominated by baryons. However, recent observations show that some dwarf spiral systems are completely dark matter dominated, which are the ideal objects to study the inner structure of dark matter halo. Burkert (1995) investigate the dark matter distribution derived from the observed neutral hydrogen rotation curves of four well studied dwarf spiral galaxies. They show that the observed universal mass profile can be fitted nicely with the following form:

$$\rho_{\mathrm{DM}}(r) = \frac{\rho_0 r_0^3}{(r + r_0)(r^2 + r_0^2)}, \tag{2.24}$$

ρ_0 and r_0 are two free parameters. This density distribution yields an isothermal profile in the inner region of the dark matter halo, and has a finite central density ρ_0. In the large radii, the total mass diverges logarithmically with increasing r. Furthermore, they investigate the relation between ρ_0 and r_0 by fitting the theoretical rotation curve to the observed dark matter rotation curves which derives from the accurate neutral hydrogen rotation curves of low-mass disk galaxies. They find a strong, linear correlation between r_0 and $v_0^{1.5}$, and derive the scaling relation for v_0, M_0, ρ_0 and r_0:

$$\begin{aligned} v_0 &= 17.7\ (r_0\ \mathrm{kpc}^{-1})^{2/3}\ \mathrm{km\ s}^{-1}, \\ M_0 &= 7.2\ \times\ 10^7\ (r_0\ \mathrm{kpc}^{-1})^{7/3}\ \mathrm{M}_\odot, \\ \rho_0 &= 4.5\ \times\ 10^{-2}\ (r_0\ \mathrm{kpc}^{-1})^{-2/3}\ \mathrm{M}_\odot\ \mathrm{pc}^{-3}. \end{aligned} \tag{2.25}$$

Here M_0 is the mass within r_0. The Burkert profile gives a good approximation of dark matter halos for dwarf galaxies, and is used in our simulation of an isolated dwarf galaxy which will be introduced in the next chapter.

- **NFW profile**

Navarro et al. (1996) use N-body simulations to investigate the structure of dark matter halos in the standard cold dark matter cosmology. They show that the spherically averaged density profiles of all halos they have studied can be well fitted with a simple "universal" profile:

$$\rho(r) = \frac{\rho_0}{\frac{r}{r_s}\left(1 + \frac{r}{r_s}\right)^2}. \tag{2.26}$$

Here $r_s = r_{200}/c$ is a characteristic radius. r_{200} is the radius within which the mean density of the dark matter halo is 200 times the critical density of the universe. The concentration factor $c = 10-17$ is typical for the halos of Milky Way sized galaxies (Klypin et al. 2002). The above equation is the so called Navarrow-Frenk-White (NFW) profile, which is frequently used in the study of galaxy formation and evolution.

The Burkert profile indicates the almost constant dark matter density in the inner parts of the galaxies, which is derived by fitting to the accurately measured rotation curves of dwarf spiral galaxies. While the NFW profile indicates a steep power-law-like profile, which is suggested by cosmological N-body simulations. This discrepancy between observation and numerical simulation is the famous "core/cusp" problem, which has not been solved until recently. For a detailed introduction to this problem, see de Blok (2010).

"In the northern ocean there is a fish, called the Kun, I do not know how many thousand li in size. This Kun changes into a bird, called the Peng. Its back is I do not know how many thousand li in breadth. When it is moved, it flies, its wings obscuring the sky like clouds."

A Happy Excursion, Chuang Tse

3 Simulation of a dwarf galaxy

In this chapter I present the simulation of a dwarf galaxy using the multi-phase model described in the previous chapter. In the first section the specially prepared equilibrium initial condition is introduced. In the second section the evolution of the dwarf galaxy is exhibited in many aspects. Finally there will be some discussion about the unsolved problems in current model.

3.1 Initial condition

We hope that the simulation starts from an equilibrium state, therefore a special initial condition (IC) is required. The basic idea is, the system will reach the equilibrium or at least a kind of quasi-equilibrium after some relaxation time. To achieve this, the IC is created in two steps. First, both the hot/warm gas and cold clouds are distributed with a Plummer-Kuzmin profile (Miyamoto & Nagai 1975) using the parameters $a = 0.2$ kpc and $b = 0.75$ kpc inside the radius of 20 kpc. The initial properties of these two components are listed in Table 3.1.

The two components rotate around the z-axis in centrifugal equilibrium with an additional 10 km/s velocity dispersion. We run a pure N-body/SPH simulation started with this distribution without any other physical process. In the beginning, the hot/warm gas expands due to the pressure gradient, the cold clouds collapse vertically. After 200 Myr of evolution, the system reaches a dynamic equilibrium state: the outward pressure of the hot/warm gas is balanced by the inward gravity. Because of the expansion of the hot/warm gas, the cold clouds are enclosed by the hot/warm spheres. In a second step, for the initial start all particles outside of 20 kpc are removed. After the removal, 12819 hot/warm particles and 9998 cold particles remain, with a total baryonic mass of 1.9852×10^9 $M_\odot$.

Initially, both hot/warm and cold component are assigned a metallicity of zero. According to our test, the results do not differ too much from runs starting with a low initial metallicity, such as $5 \times 10^{-3} Z_\odot$ because the gas is so quickly metal enriched when stellar feedback starts. Nonetheless, the cooling function allows for individual element abundances deviating from solar abundance ratios but depends mainly on hydrogen and helium at very low metallicity. Another reason to choose zero initial metallicity is, we do not need to consider the initial metal abundances which are not well determined by observation.

As mentioned in previous chapter, an analytical density profile (Burkert 1995) is used for the modeling of the dark matter halo. The parameters are $r_0 = 3$ kpc and $\rho_0 = 0.02$ $M_\odot/\mathrm{pc}^3$

Table 3.1: Initial conditions of two gas components. We first let the system relax as a pure N-body/SPH run. After 200 Myr of evolution, the system reaches a dynamic equilibrium state. All particles outside a radius of 20 kpc are removed.

Component	Mass	Temperature	Particle Number	Particle Number
	(in $M_\odot$)	(in K)	(Initial)	(After removal)
hot/warm	4×10^7	10^6	2×10^4	12819
cold	1.96×10^9	10^3	1×10^4	9998

which yield a dark matter mass of around 10^{10} $M_\odot$ within 20 kpc. Throughout the simulation we use an external pressure of 1 K/cm^3 (around 1.38×10^{-17} pa) in order to avoid that hot/warm gas expands into the vacuum. We fix the integration time step Δt to 0.1 Myr, this value is appropriate for most of the time. But in some extreme cases such as when a cold cloud is heated above 10^4 K, the cooling time becomes very unreasonably short (overcooling problem), we have to give it some special treatment: when the gas temperature (for both components) drops from above 10^4 K to below with the huge cooling rate, we first stop it at a value slightly below 10^4 K, say, 9000 K, such that a low and reasonable cooling rate will be calculated.

In total, we carry out simulations for six dwarf galaxy models with slightly different parameters for comparison and analysis, as listed below:

- Reference model: with all the parameters described above. Our analysis in the next section are mainly based on this model.
- Reduced SN energy model: reduce the energy release of one supernova event to 10% of that in the reference model, which now takes the value 10^{50} erg instead of 10^{51} erg.
- High metallicity value: start with an initial metallicity of 5×10^{-2} $Z_\odot$ instead of zero as in the reference model.
- High resolution model: rise the initial cold particle number from 10^4 to 10^5 and keep the total cold cloud mass unchanged as in the reference model.
- Pressure SF recipe model: the SF efficiency depends on the pressure of the surrounding diffuse gas.
- Temperature SF recipe model: the SF efficiency depends on cold cloud's temperature. High temperature yields a low efficiency.

3.2 Evolution of the model

We start the simulation with the setup of initial conditions. According to the physical processes described above, cold clouds collide and coagulate; stars form inside these clouds, shock waves driven by feedback disrupt clouds into smaller ones; stellar feedback ejects metals to

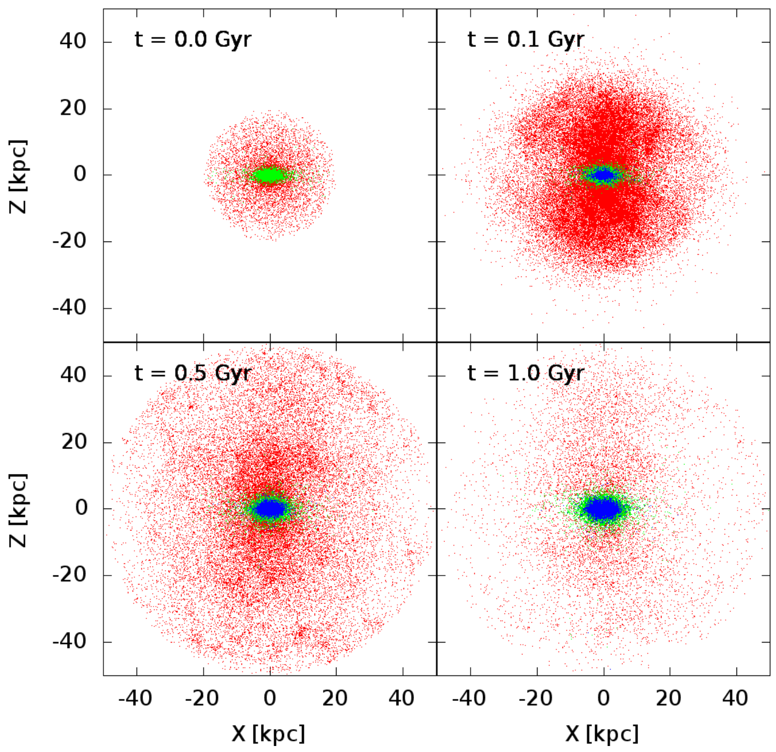


Figure 3.1: Edge on view of the system at four different time. Red, green and blue dots corresponds to hot/warm gas, cold clouds and star particles, respectively.

the ISM, hot/warm gas is heated by supernova feedback. In the center of the galaxy, the temperature can be as high as 10^6 K. Such a high temperature and relatively low gas density (0.1 cm^{-3} at most) leads to very strong evaporation of cold clouds transferring their mass to the surrounding hot/warm gas. When a hot/warm SPH particle reaches an upper mass limit, set to 8000 $M_\odot$ in the current models, it splits into four small particles equally. High temperature corresponds to high pressure, which drives the hot/warm gas expanding due to the pressure gradient. Supernova energy is transfered to kinetic energy of the hot/warm gas in this way. Because dwarf galaxies are of low mass ($\sim 10^9$ $M_\odot$), the potential well is not deep enough to keep the high speed hot/warm gas in the center. Therefore, most of the gas will escape from the galaxy. Below we will analyze these processes in detail.

3.2.1 Snapshots

Figure 3.1 gives a first impression of the evolution of the model. At the beginning of the simulation, cold clouds are embedded in the hot/warm gas. After the system evolves for 0.1 Gyr, there are already a lot of star particles created. Due to evaporation and particle splitting, the number of the hot/warm particles rises 7 times. Although our simulation starts from the quasi equilibrium, SF breaks this state, especially for the hot/warm gas which is very sensitive to

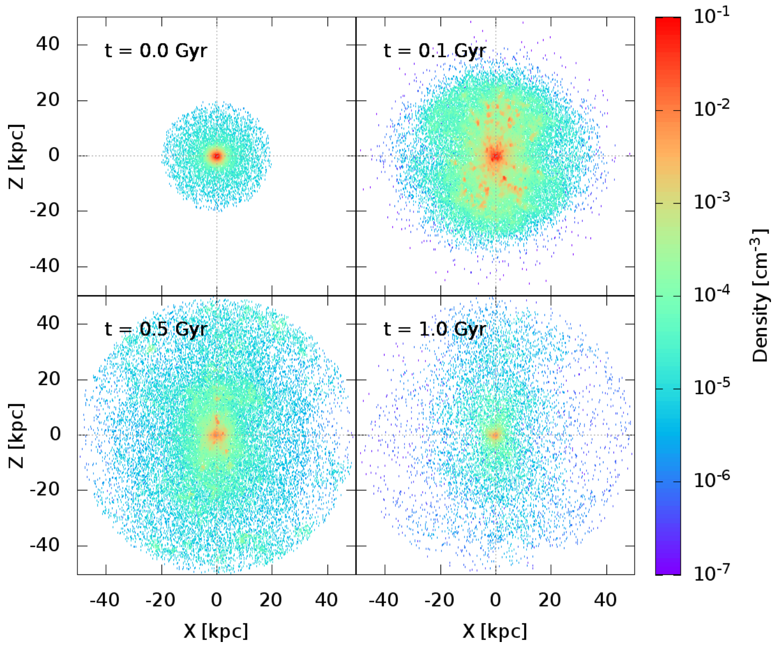


Figure 3.2: Edge on view of the hot/warm gas density at four different time.

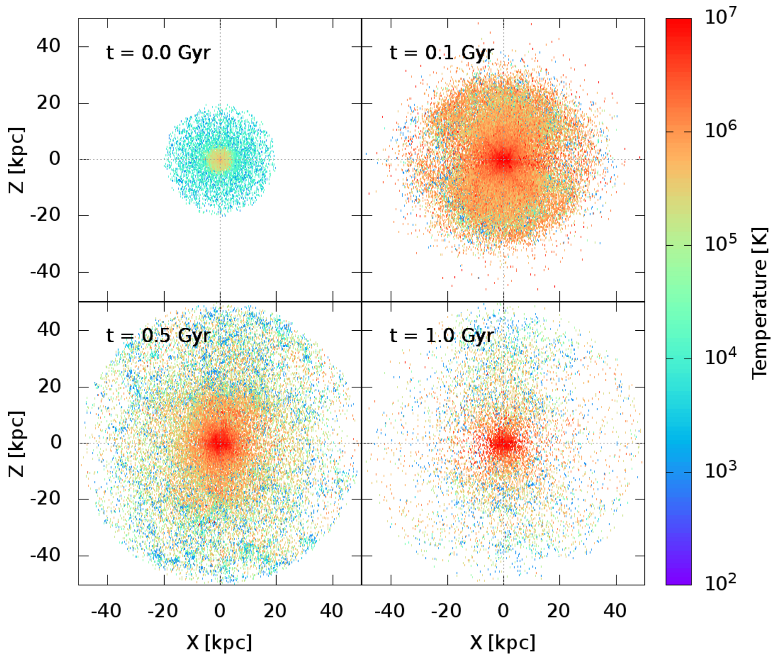


Figure 3.3: Edge on view of the hot/warm gas temperature at four different time.

temperature variation. After the first 100 Myrs evolution, the system reaches another dynamical equilibrium. According to figure 3.4 and 3.5 of energy and mass evolution, there are no big changes for these quantities over the rest of the time. At half Gyr, some of the hot/warm particles already expand outside of 50 kpc and are removed from the system. After 1 Gyr evolution, most of the hot/warm gas has escaped from the galaxy.

Figure 3.2 and 3.3 demonstrate the density and the temperature distribution of the hot/warm gas respectively. The gas density in the center evolves from 0.1 cm^{-3} in the beginning to 0.001 cm^{-3} at 1 Gyr. In the beginning, the gas temperature drops from 8×10^5 K in the center to 10^5 K at a radius of 5 kpc. When the simulation starts, SNII feedback injects a large amount of energy into the hot/warm phase, which leads to a temperature as high as 10^7 K in the center. This high temperature yields a strong evaporation rate, a large amount of cold cloud mass transits to the hot/warm phase.

3.2.2 Energy and mass evolution

Thermal and kinetic energy evolution of the hot/warm gas and the cold clouds, respectively, are shown in figure 3.4. For the first 50 Myr, thermal energy of the hot/warm gas decreases as a result of radiative cooling. Then it rises very quickly when stellar feedback starts. After a delay of several million years, thermal energy transits to kinetic energy and the hot/warm particles are accelerated and escape from the system. By observing the top panel, the kinetic energy of the hot/warm particles correlates well with supernova feedback. Figure 3.5 gives the mass evolution of the three components. Initially, the hot/warm component takes only 1.3% of the total mass, the rest is in the form of cold clouds. After 1 Gyr almost half of the cloud mass has been transferred to the hot/warm component through evaporation and escapes, 15% of the mass transits to stars by SF. Before stellar feedback starts, the hot/warm gas is in a state of low temperature and high density, its mass decreases as a result of condensation. We can still observe this in the very beginning.

3.2.3 Energy and mass transition rates

Figure 3.6 shows the system's energy transition rates between the system components as a function of time. The "PdV" term represents the adiabatic expansion work of the hot/warm gas and correlates well with SNII and SNIa feedback, which suggests that most of the supernova energy is converted to the kinetic energy of the hot/warm gas, this has been shown in the energy diagram (figure 3.4). "UV" and "MECH" together comprise stellar wind feedback, which dominates the heating of cold clouds. Note that the cooling rate shown here excludes the contribution from cold clouds with temperature above 10^4 K, since the cooling rate would be high for cold clouds with such a high density. We will discuss the cooling problem in the discussion session. Figure 3.7 is the system's mass transition rates between phases as a function of time. Clearly, evaporation transfers a large amount of cold cloud mass to hot/warm component. As hot/warm gas becomes less dense and more energetic during the evolution, the condensation rate decreases and almost stops after 800 Myr.

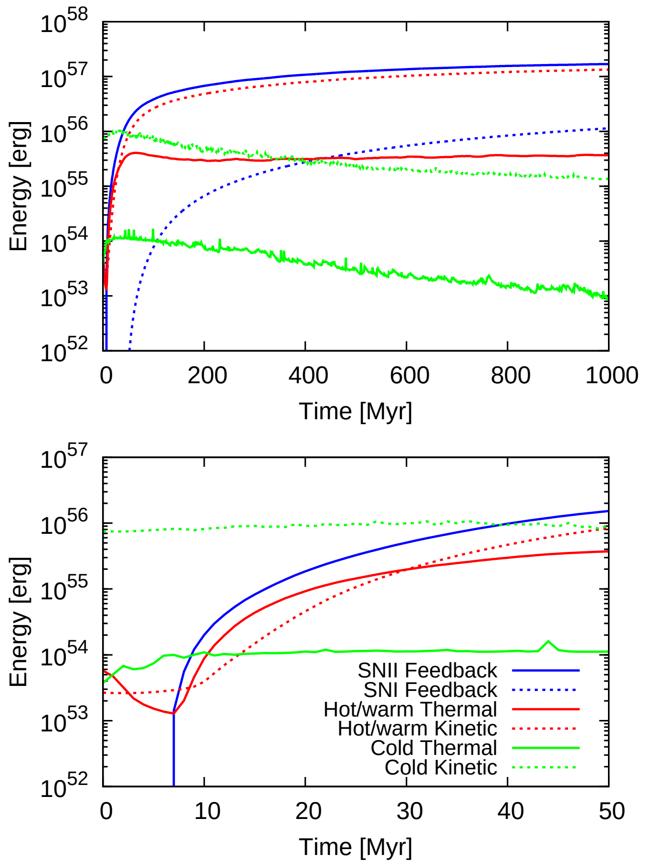


Figure 3.4: Top: energy evolution within 1 Gyr, bottom: energy evolution for the first 50 Myr. Red and green lines correspond to hot/warm gas and cold clouds respectively. The accumulated supernova energies are plotted in the figure as blue lines.

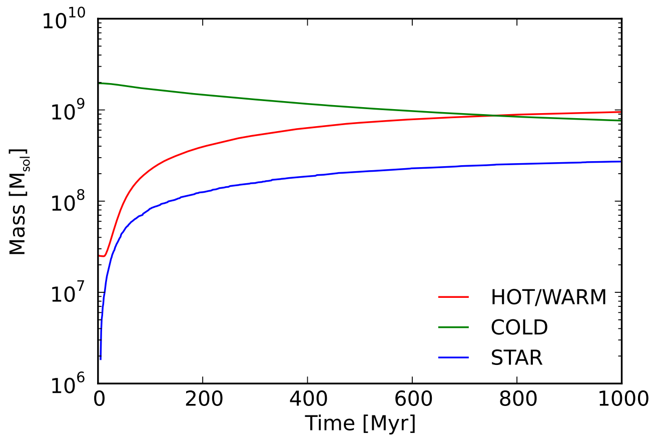


Figure 3.5: Mass evolution of the system. Red, green and blue lines correspond to hot/warm gas, cold clouds and stellar components respectively.

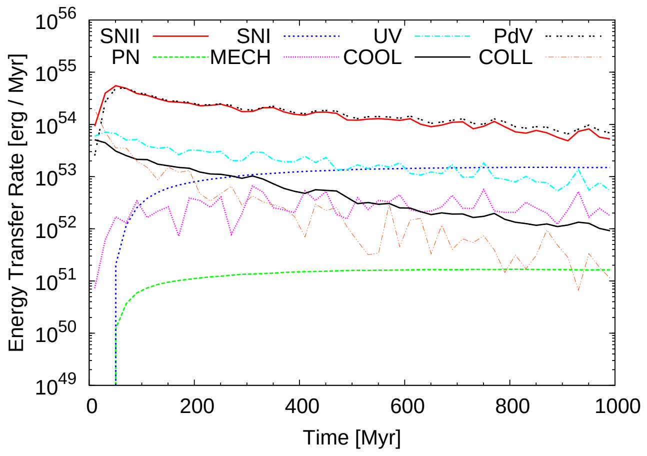


Figure 3.6: Energy transition rates between phases for different physical processes. Here “PdV” refers to the hot/warm particles’ energy transition from thermal to kinetic in the SPH treatment and “COLL” measures the energy transition from kinetic to thermal when two cold clouds coagulate (see section 2.2.2 for a detailed description). Stars’ UV radiation together with mechanical energy release consist of the stellar wind feedback.

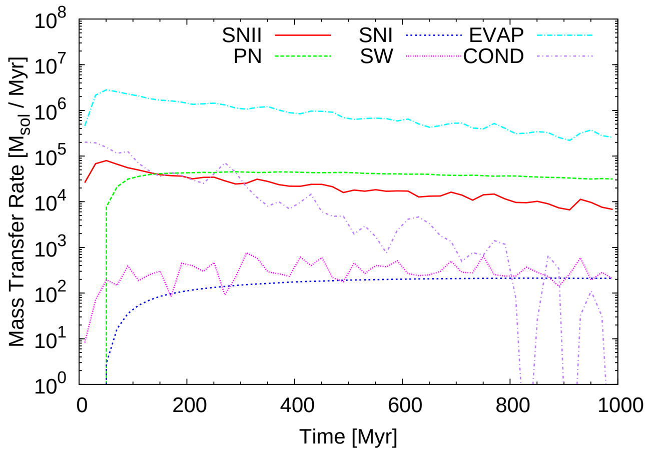


Figure 3.7: Mass transition rates between phases.

3.2.4 Chemical evolution

Figure 3.8 exhibits the evolution of the oxygen abundance within different radii. For the cold clouds and the star particles, the radius of 1 kpc and 5 kpc correspond to the innermost part and the main body of these two components. In the first few million years, the oxygen abundance of the hot/warm gas rises quickly as massive stars produce the oxygen and SNeII inject it to the hot/warm phase. Later on, cloudy gas of low oxygen abundance mixes via evaporation with the SNII gas, the oxygen abundance for this component varies around an almost constant value of 8.4 as for both inner and outer parts of the galaxy. Bigger deviation from average value of O abundance in the inner part is via direct connection of SF and the ejected O mass from Type II SNe: the bigger the SF is, the more O is transferred to the hot/warm gas. The oxygen abundance for cold clouds is 1.5 dex lower. Since the condensation rate is low in our model, especially in the outer part of the galaxy, most of the O is carried away by the outflow of hot/warm gas instead of being transfered to the cold phase. As for Milky Way sized galaxies the gravitational potential well is deeper, more hot/warm gas is retained inside the galaxy and because of its higher density cooling acts more efficiently and mass exchange by condensation between phases is much more effective. We can expect that the oxygen abundance of the cold clouds in larger systems is much higher compared with dwarf galaxies. For the cold clouds and the stellar components, O is always more abundant in the center of the galaxy due to more efficient condensation.

The O and N abundances at 1 Gyrs as a function of radius are shown in figure 3.9. While O in the hot/warm phase stabilizes at a value of 8.4 as a result of the balance between evaporation and SNII feedback, the cold phase forms an O gradient due to condensation which transfers O from the hot/warm phase to the cold clouds. N is mainly produced by PNe and the most massive stars in pre-SN stage, and is ejected to the surrounding cold clouds. One may notice that the nitrogen abundance is declines towards the center. This is due to the massive cold clouds with masses larger than 10^5 $M_{\odot}$. They are metal poor and tend to stay in the galaxy

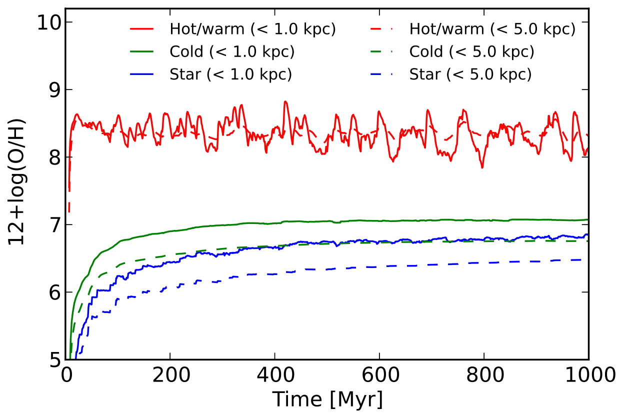


Figure 3.8: The evolution of the oxygen abundance within 1 kpc (solid line) and 5 kpc (dashed line).

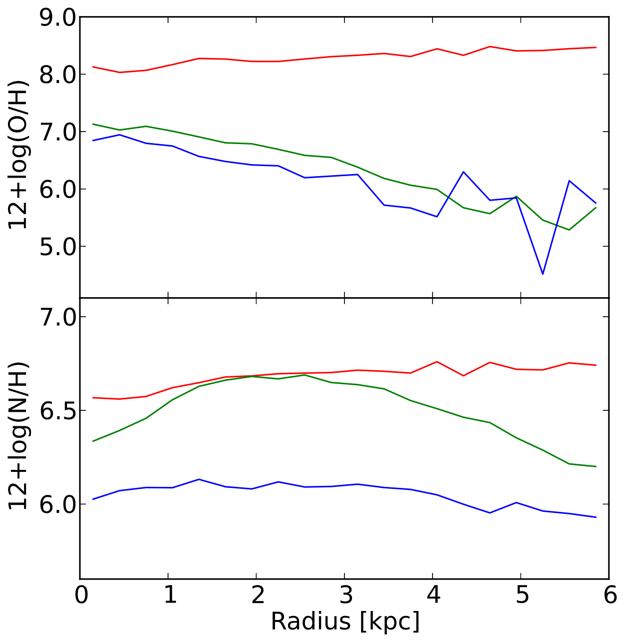


Figure 3.9: Oxygen (top) and nitrogen (bottom) abundance as a function of radius at 1 Gyr. Red, green and blue lines correspond to the hot/warm gas, cold clouds, and stars, respectively.

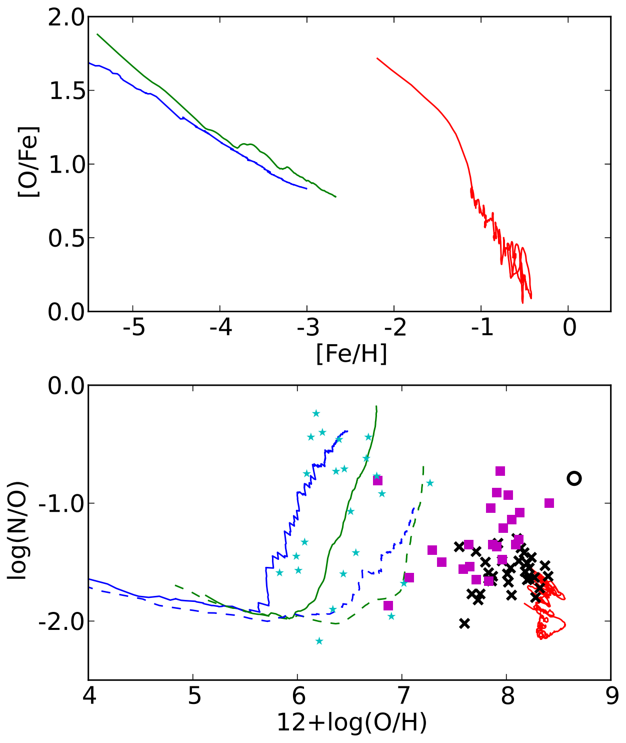


Figure 3.10: Top: the evolution of oxygen to iron ratio as a function of iron abundance at the corresponding time. The ratio and iron abundance are expressed as relative value to solar abundance. Bottom: the evolution of nitrogen to oxygen ratio as a function of oxygen abundance at the corresponding time. Both ratios are calculated during the evolution of the first 1 Gyr within a radius of 5 kpc. Red, green and blue solid lines correspond to the hot/warm gas, cold clouds and stars components of our reference model. Green and blue dashed lines correspond to the cold clouds and stars components of a reduced SN energy model with the energy release of supernova (SNeII and SNeIa) set to 10^{50} erg. Note that the points of the first 10 Myr are excluded from the figures, since feedbacks just start at that time and phases are not mixed enough. As a comparison, observations for the Galactic metal-poor stars from Spitel et al. (2005) and Israelian et al. (2004) are shown as cyan pentagons and magenta squares. Observations for HII regions of dwarf galaxies (van Zee et al. 1997) are shown as black crosses. Solar abundances are shown as a black circle (Asplund 2005; Caffau et al. 2009).

center. Hot/warm gas obtains nitrogen from low-metallicity massive stars and mostly via evaporation, which yields a similar nitrogen abundance distribution as the cold clouds in the center.

The oxygen-to-iron ratio as a function of iron abundance is shown in the top panel of figure 3.10. In the current model, iron is mainly produced by Type Ia SNe and is returned to the hot/warm phase. It cannot be transfered to the cold phase efficiently since the condensation rate is low. This leads to the two orders of magnitude difference of the iron abundance between the hot/warm phase and the cold phase. According to the stellar evolution model, the progenitors of SNeIa live longer than those of SNeII. The latter one produces a large amount of O (Tinsley 1979; McWilliam 1997). This leads to a higher O/Fe ratio at the beginning of the feedback, triggered by short-living massive stars. Very early (in the current model it is around 40 Myr), SNeIa start producing Fe, O/Fe ratio decreases with time. The N/O versus O/H plot is another powerful diagnostic diagram. The evolution of the N/O ratio can be explained in a similar way as for Oxygen. O is first produced by SNeII, and is transited to the cold phase via condensation at the early epoch. The most massive zero-metallicity stars also produce large amounts of N which also condenses into cold gas. As a consequence, a plateau in N/O lies in range of -1.7 to -2.0, is found for $12 + \log(O/H) < 6$ for all components (the plateau for hot gas component is not shown because evolutionary tracks of averaged in inner 5 kpc abundances for all phases in figure 3.10 start since 10 Myr of initial evolution and early evolution of hot/warm gas disappears but indirectly represented in cold gas and star phases). Later PNe as from pristine and next generation stars together with SWs of the very massive non zero-metallicity stars eject a large amount of nitrogen to the surrounding cold clouds. As mentioned earlier, after half Gyr the condensation rate decreases for two orders and even almost vanishes at 1 Gyr (see. figure 3.7), the copious amounts of oxygen produced by SNeII cannot be transferred to the cold phase efficiently, which leads to the low oxygen abundance of cold clouds and stars and the rise of their N/O ratios. As a comparison, the relatively higher condensation rate in the reduced SN energy model leads to a more efficient oxygen transition from the hot/warm phase to the cold phase, which yields a higher oxygen abundance of the cold clouds and stars components. In contrast with cold and stellar phases, in the hot/warm phase N transferred from cold phase via almost constant evaporation reaches a balance with the oxygen ejected by supernovae, the N/O ratio just slightly increases. In addition, the strong galactic outflow due to SNeII feedback from massive stars at later time effectively removes hot/warm phase from galaxy and leaves its abundances not changed. Models assuming strong outflows could delay the ISM enrichment from hot phase so that AGB stars would have time to contribute to what is shown for the evolutionary track of cold phase. Gap between cold and stellar phases is due to the nature of SF: cold clouds have to cool enough and not to undergo the coagulation what would delay the SF and give time to pollute cold phase. Compared with observation, the N/O ratios of cold clouds and stars lie in the same range of observation data of the very metal-poor Galactic halo stars (Spitel et al. 2005; Israelian et al. 2004), the N/O ratio of hot/warm phase is comparable with the observations of van Zee et al. (1997) for HII regions of dwarf galaxies (observation data shown on the figure 3.10 lower panel can represent as the pristine as the already mixed abundances from the ISM, from which the stars formed. Literature data for stars also need corrections owing to non-LTE and 3D effects (Asplund 2005) what would lead

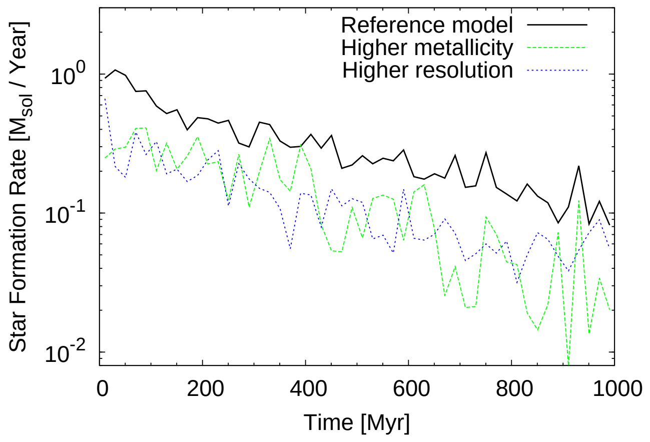


Figure 3.11: Star formation rate as a function of time for different parameters. The black line is the reference model in this paper. The green line is the high metallicity model with an initial metallicity of 5×10^{-2} Z$_\odot$. The blue line is the high resolution result by increasing the initial cold particle number from 10^4 to 10^5 and keep the total cold cloud mass unchanged.

to a slowly increasing trend of N/O with O/H even at the lowest metallicities).

3.2.5 Star formation rate

Figure 3.11 exhibits the SFR as a function of time. In general, it decreases from 1 M$_\odot$/Year to 0.1 M$_\odot$/Year at 1 Gyr. Such kind of decrease cannot be simply interpreted as a result of the decrease of the cold cloud mass. As is described in section 2.5.2, we adopt the Jeans instability criterion as our standard SF recipe in this paper. The Jeans mass which determines the mass of a star particle depends on cloud mass (via the density) and temperature. A low cloud temperature yields a low Jeans mass and thus low star particle mass. Moreover, the derived theoretical mass of a star particle should not be too low in our scheme, because the mass of a single stellar population must fulfill the conditions to cover a full IMF so that we limit the lowest star particle mass to be 10^4 M$_\odot$, which prevents clouds from forming stars if the stellar mass is too small. As the system evolves, the decrease of cloud temperature, together with the consumption of cloud mass leads to the decrease of SFR. We have to admit that the star formation process is highly uncertain, any change of these physical/non-physical quantities such as metallicity and mass resolution will yield totally different result. We will discuss this in the discussion section.

3.2.6 Cloud mass function

For the cloud mass function shown in figure 3.12, initially all clouds have the same mass: 1.96×10^5 M$_\odot$. After the evolution of 100 Myr, a spectrum already forms. There are two peaks in the spectrum: one is the original peak, another appears at 5×10^4 M$_\odot$ which corresponds to

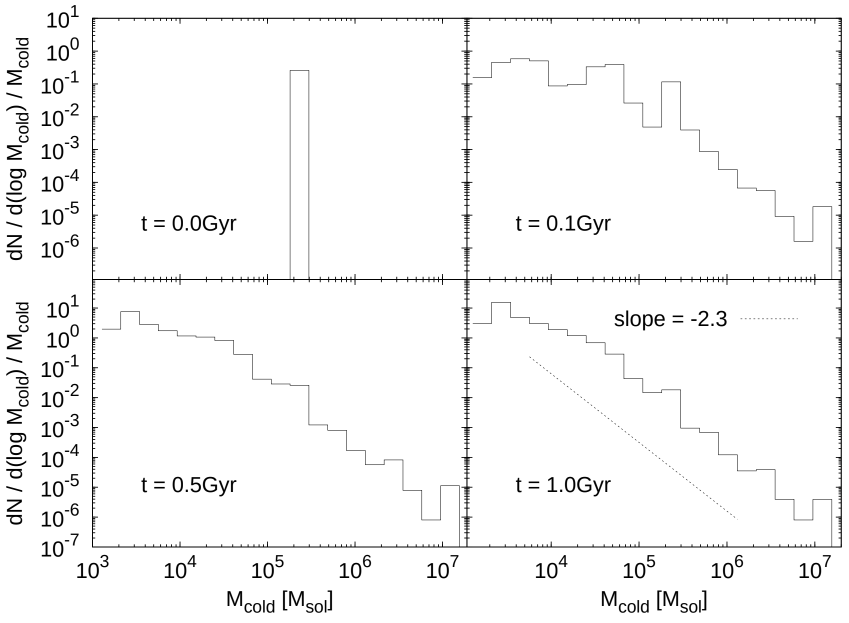


Figure 3.12: Cloud mass distribution (function) for four different times (0.0 Gyr, 0.1 Gyr, 0.5 Gyr, 1.0 Gyr). Dashed line in right bottom panel corresponds to the power law mass function with a slope of -2.3.

clouds after fragmentation. At the high-mass end, some massive clouds form via coagulation. In general, massive clouds have larger cross sections, so that they are more likely to coagulate with others. Also, they are more likely to collapse due to Jeans instability. In the end, most of the clouds populate the low-mass end. Note that the low and high-mass limit of cold clouds are fixed as 10^3 $M_\odot$ and 2×10^7 $M_\odot$, respectively. The lower mass limit is a definition by us to allow a time delay of the clouds to reach Jeans mass but to coagulate before. After the evolution of 1 Gyr, the spectrum becomes smooth, the distribution roughly follows a power law with a slope of -2.3, which is consistent with the simulation of the MW-like galaxy (Harfst et al. 2006) and the observation of molecular clouds in nearby galaxies (Fukui & Kawamura 2010).

Hopkins (2012) attempts to predict the structure of GMC using an excursion-set model. By making some fundamental assumptions for GMC and using the "extended Press-Schechter" formalism which achieves great success for the modeling of the dark matter halo mass function, they get a cloud mass function with a slope of around -2. For the cloud mass function at 1 Gyr, there is still the original peak at about 2×10^5 $M_\odot$. According to our analysis it comprises cloud particles, which are far away from the center (> 8 kpc) so that they can neither coagulate because of the low density in the outer region nor fragment since they cannot receive feedback effectively and thus are too cold to form stars. Also the evaporation is not very effective. They can only keep their initial mass and rotate slowly.

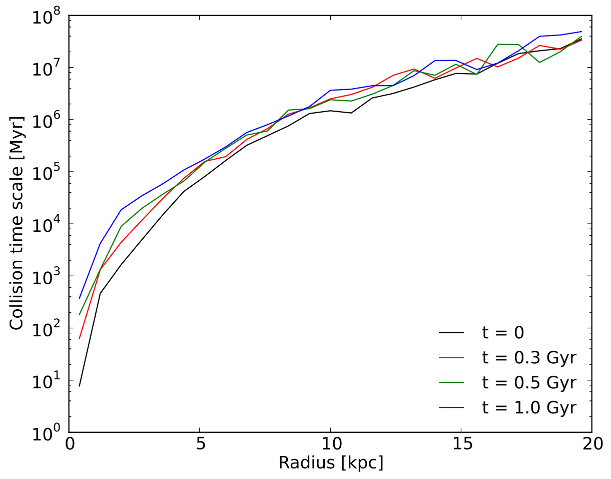


Figure 3.13: Collision time scale of cold clouds at four different times (0, 0.3 Gyr, 0.5 Gyr and 1.0 Gyr).

3.2.7 Collisional time scale and velocity dispersion of cold clouds

To study the coagulation behavior of the cold clouds, we calculate the collisional time scale, which is expressed as:

$$\tau = \frac{1}{n\sigma A_{\rm cr}}. \tag{3.1}$$

In the above equation, n is the number density of cold clouds, σ is the velocity dispersion, $A_{\rm cr}$ is the cloud's cross section expressed by equation 2.5. All of these three quantities are calculated for every cloud particle by averaging their N_B neighbours. Figure 3.13 demonstrate the collisional time scale binned for cloud cloud particles at different radii at 0, 0.3 Gyr, 0.5 Gyr and 1.0 Gyr. It is not difficult to understand that the time scale is shorter in the inner region, since higher particle number density will yield a shorter time scale. For the time evolution, in the outer part, the time scale does not evolve too much. However, it rises for almost two orders in the galaxy center during the 1 Gyr evolution, which means cold cloud particles are more likely to collide with each other and thus coagulate in the beginning of the simulation. Figure 3.14 demonstrate the evolution of velocity dispersion of cold clouds. After 1 Gyr evolution, it decreases 10 km/s in the galaxy center. However, this small changes cannot fully account for the two orders increase of the collisional timescale. According to our analysis, this increase is mainly due to increased number of low clouds. They are less massive and the corresponding cross sections are also small. Collisions are less likely to happen, which naturally leads to the longer collisional time scale. As the system evolves, more low mass clouds are generated as a result of fragmentation, the collisional time scale rises gradually.

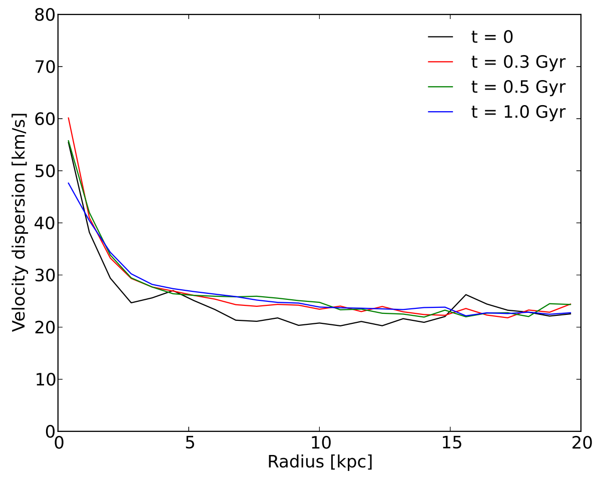


Figure 3.14: Velocity dispersion of cold clouds at four different times (0, 0.3 Gyr, 0.5 Gyr and 1.0 Gyr).

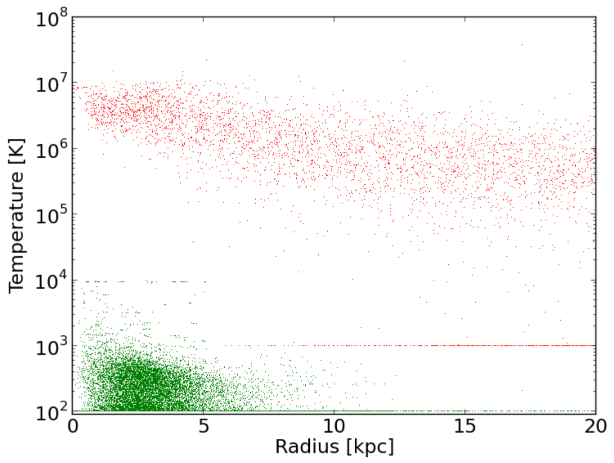


Figure 3.15: Temperature distribution of hot/warm gas and cold clouds as a function of radius at 1 Gyr. Red and green dots correspond to hot/warm and cold components respectively.

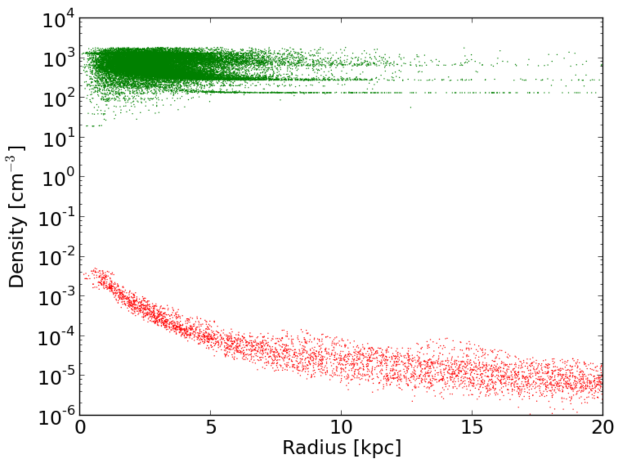


Figure 3.16: Density distribution of hot/warm gas and cold clouds as a function of radius at 1 Gyr. Red and green dots correspond to hot/warm and cold components respectively.

3.2.8 Temperature distribution

The multi-phase nature of ISM is clearly demonstrated in figure 3.15. For the hot/warm component, the temperature of particles in the galaxy center can be as high as 10^7 K due to supernovae feedback. This is also shown in figure 3.3 as the snapshot of temperature for hot/warm component. Note that there is a red line at 10^3 K, which is due to the hot/warm particles at the low temperature limit of this component. They are not allowed to further cool when their temperatures reach this limit. For the cold component, most of cold clouds have a temperature of below 10^3 K. However, some of them stay at 10^4 K. This is due to our special treatment of cooling, as we will discuss in detail in the discussion section. When two cloud particles coagulate, their extra kinetic energy transits to the internal energy of the newly formed clouds, which might lead to a temperature higher than 10^4 K. When it cools with a huge cooling rate, we first let it stop at a temperature slightly below 10^4 K, such that a reasonable low cooling rate can be calculated.

3.2.9 Density distribution

The density distributions of the two gas phases are demonstrated in figure 3.16. In the multiphase model, the hot/warm phase and cold phase are modeled in totally different way. The large density discrepancy between the two phases is reproduced correctly. For the hot/warm phase, gas density decreases from $3 \times 10^{-3}/\mathrm{cm}^3$ in the center to $10^{-5}/\mathrm{cm}^3$ in the outer part, which is also shown in figure 3.3. For the cold phase, since we use the *mass-radius* relation to determine the size of cold cloud, cloud density correlates directly with cloud mass: massive clouds corresponds to a low density. From the figure, most of cloud particles distribute inside a radius of 5 kpc. However there are still three tails in the outer part with similar densities, which means these particles have similar masses. We can find some hints from figure 3.3 of

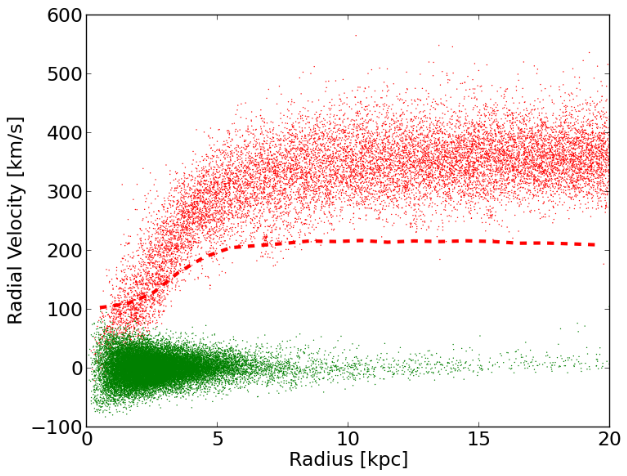


Figure 3.17: Radial velocity distribution for two gas components at 0.5 Gyr. Red and green dots correspond to hot/warm gas and cold clouds of the reference model respectively. The average radial velocity as a function of radius for the reduced SN energy model (the energy release of supernova is set to 10^{50} erg) is presented as red dashed line.

cloud mass distribution: from high to low density, these three tails correspond to three clouds mass: 3×10^3 $M_\odot$, 2×10^5 $M_\odot$ and 3×10^6 $M_\odot$. In particular, the middle one consists of particles with the initial cloud mass. Those particles reside in the outer part of the galaxy, where the collision rates are quite low.

3.2.10 Radial velocity of hot/warm gas

In figure 3.17 we exhibit the radial velocity as a function of distance from the center of the galaxy for both the hot/warm gas and the cold clouds of the reference model. For the cold clouds the radial velocity is low and symmetrically distributed around $v = 0$ within 5 kpc so that they are bounded to the system and will not escape. In contrast, the hot/warm particles are accelerated within the inner region of 5 kpc through supernova feedback and can escape from the galaxy with radial velocities between 300 km/s and 400 km/s, while the virial velocity at $r = 20$ kpc is 50 km/s, which is one order lower than the radial velocity. As a comparison, less energy is transited to the hot/warm phase in the the reduced SN energy model, the corresponding radial velocity is much lower. Actually, such strength of high-velocity winds has only exceptionally been observed but with lower velocity (see e.g. M82), so that our result must be regarded as a numerical artifact of our isolated DG model.

3.3 Discussions

In this section, we discuss the still existing weaknesses of the multi-phase model and study the influences of parameterizations.

3.3.1 Initial metallicity

We set the initial metallicity to zero which yields a similar result as low initial metallicity of 5×10^{-3} $Z_{\odot}$. As a toy model for dwarf galaxy, there is no preferred initial value for such kind of system. We made a test run with a higher initial metallicity of 5×10^{-2} $Z_{\odot}$, the SFR evolution is shown as green line in figure 3.11. As we can see, the SFR decreases immediately by 0.5 dex. This can be explained as a decrease of cold cloud temperature: in the low temperature region ($< 10^4$ K), the cooling rate depends on the metallicity; higher metallicity yields a more efficient cooling and therefore lower temperature. According to section 2.5.2, the corresponding Jeans mass decreases, which yields a lower stellar mass and thus the decrease of SFR. Some multi-phase model (Harfst et al. 2006) does not consider the metallicity evolution of the system by just assuming a constant value such as solar metallicity, and still get reasonable result, since their star formation recipes do not depend on the temperature and metallicity of cloud particles so much.

3.3.2 Cold-phase description

In the current work, we use the *mass-radius* relation (Eq. 2.1) derived by observation to determine the size of the cold clouds and have obtained reasonable results. However, since the structure of the cold cloud is rather complex, and the densities can vary by several orders of magnitude (Klessen 2011), the *mass-radius* relation is not always applicable. Alternatively, one can derive the radius of the cold clouds with other assumptions, such as pressure equilibrium between the cold phase and the hot/warm phase. We have to keep in mind that very likely this will lead to an unreasonably large cloud radius and low density, especially when the hot gas pressure is low in DGs. Another disadvantage of the current relation is the behavior of the cold clouds, such that the SF is affected by the mass resolution since the cloud size and, by this, the density are only determined by the cloud mass. In a proper and reliable model, the resolution effect should be as small as possible. Therefore, we implement coagulation and fragmentation for the cold clouds such that the cloud mass distribution can evolve in a self-regulated way and thus does not depend on the initial mass configuration. We compare the SFR evolution for a higher resolution (10^5 cold particles initially, other parameters including the total cold cloud mass are kept unchanged) as blue line in figure 3.11. The corresponding SFR is again lower, since cold clouds can only form stars after their mass becomes large enough via coagulation. What's more, the change of resolution leads to the change of cloud density, which yields different physical conditions, and affects star formation rate. The SF discrepancy also suggests that current description of cold clouds is not good enough and must be continuously improved.

3.3.3 Star formation recipes

Throughout this paper we use the Jeans instability criterion as the reference SF recipe. Besides this, two other recipes are tested for comparison. One is proposed by Elmegreen & Efremov (1997) and implemented in Harfst et al. (2006), in which the SF efficiency is controlled by the mass of cold clouds and the pressure of the ambient diffuse hot/warm gas (pressure recipe).

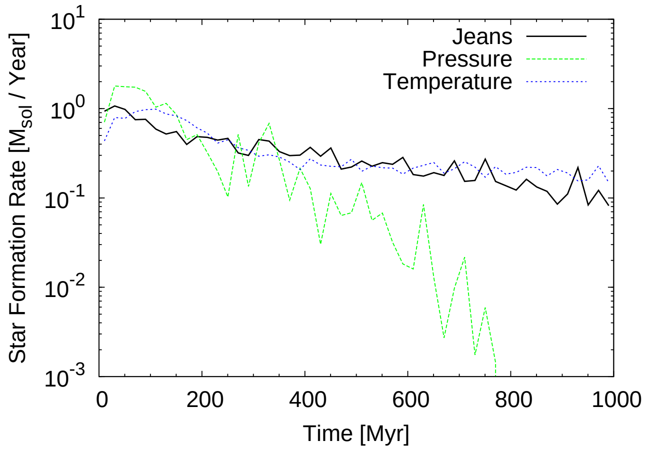
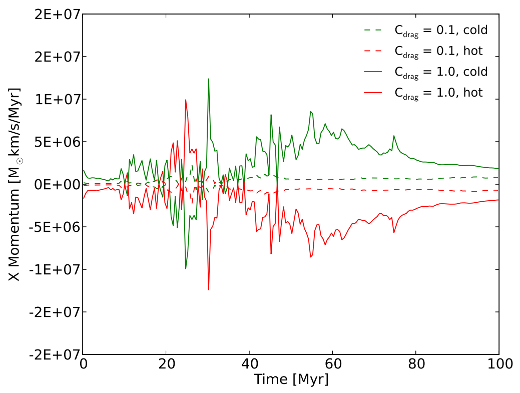


Figure 3.18: Three different kind of star formation recipes.

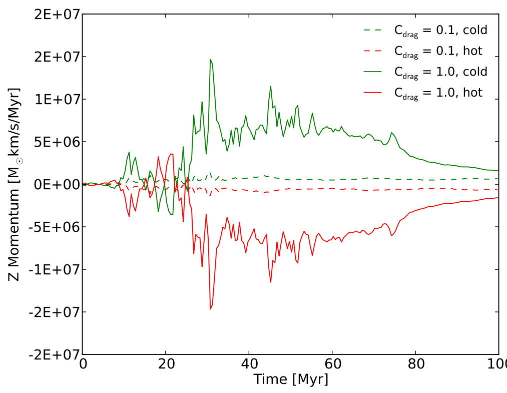
Another one is adopted in Theis et al. (1992), which uses a self-regulated temperature dependent SF criterion (temperature recipe). figure 3.18 presents the SFR evolution for these three different recipes. Note that in the pressure recipe SFR strongly depends on the surrounding hot/warm gas pressure. At 800 Myr almost all the hot/warm gas has escaped from the center of the galaxy, which leads to the quenching of SF. Obviously, for our treatment and the DG objects the Elmegreen-Efremov SF efficiency does not work out properly. We have to admit that although the “Katz” scheme (Katz 1992; Stinson et al. 2006; Saitoh et al. 2008) has been commonly accepted as the standard SF recipe in the single-phase models, there is still no preferred SF recipe for the multi-phase models where cold clouds are modeled as an independent phase.

3.3.4 Drag force coefficient

When we first apply the multi-phase model to the simulation of a dwarf galaxy, it seems that hot/warm gas escapes from the galaxy faster than we have expected (now we know this is the natural result for the current multi-phase model in which the supernova energy transits to the kinetic energy of supernovae efficiently since cooling rate is low due to the low gas density). We consider a larger drag coefficient will enhance the drag force between the cold and hot/warm phases, thus reduce the speed of hot/warm gas. Therefore an experiment is carried out to investigate the effect of different drag coefficient. The result is shown in figure 3.19 for $C_{\mathrm{D}} = 0.1$ and 1.0 respectively. Initially the drag force is small since the system is in an equilibrium state. Then this equilibrium is break down at 10 Myr as supernovae inject the feedback energy to the hot/warm gas and the hot/warm particles start escaping from the system. The larger value of C_{D} yields a more efficient momentum transfer for the hot/warm particles which prevents them leaving the galaxy. However as we can see from the figure that after some fluctuation a new equilibrium forms, the drag force becomes small. According to our observation the drag force cannot prevent the hot/warm particles from escaping the galaxy.



(a) x direction.



(b) z direction

Figure 3.19: Momentum transfer via drag force between the hot/warm and cold phases. Particles in the region $x > 0$, $y > 0$ and $z > 0$ are taken into account. The drag coefficient C_{D} is set to 0.1 and 1.0.

3.3.5 Galactic winds

As is shown in figure 3.17, in our multi-phase model the hot/warm gas escapes from the center of the galaxy with a radial velocity one order higher than the galaxy's virial velocity which seems unreasonable at first glance. While in a single-phase model, the velocity of gas is low, as supernova feedback energy cannot be transformed to kinetic energy of hot/warm phase efficiently. As pointed out by Booth et al. (2007), in the single-phase model, gas in the surrounding of SF sites is at high density and thus cooling is very efficient, leading to the well-known overcooling problem. The feedback energy is just radiated away instead of being transformed into kinetic energy of the gas. In contrast, in the multi-phase model because of the low mixing of hot gas to the cooler ones hot/warm gas does not radiatively cool so quickly. Supernova feedback energy can be kept in the hot/warm phase and drives gas expansion efficiently. Increasing the density of the hot/warm gas can yield a more efficient cooling, therefore decreases the velocity of the high speed outflow. But a more general solution that combines the advantages of both single and multi phase models is by introducing phase transition in a reasonable way: hot/warm gas can cool to cold cloud temperature and forms cold dense clumps. Then these clumps collapse due to Jeans instability and form real cold clouds. By doing this hot/warm gas can transit to cold clouds in a natural way, and keeps a reasonable density that yields an appropriate cooling rate.

Part of this chapter was submitted to Astronomy & Astrophysics (Lei Liu, Mykola Petrov, Peter Berczik, Rainer Spurzem, and Gerhard Hensler, "A Multi-Phase Chemodynamical *N*-Body/SPH code of Galaxy Evolution").

"An artisan hoping to do fine work must first sharpen his tools."

Analects, Confucius

4 Transition of hot/warm gas

In the previous chapter, I present our simulation of a dwarf galaxy (mainly refers to its low mass) using the multi-phase model described in chapter 2. In this chapter, I will discuss the drawbacks of that model, and introduce our implementation of hot/warm gas transition process by which hot/warm gas collapses to cold clouds. This chapter is organized as follows: first, some multi-phase models mentioned in the first chapter are rediscussed. Then I introduce the detailed implementation of transition process. Finally I present the parameter study of six galaxies by taking this new process into account.

4.1 Re-discussion of the multi-phase model

In the multi-phase dwarf galaxy simulation presented in previous chapter, we try to reproduce some observational features, such as the gas outflow, the cloud mass distribution, and the evolution of chemical abundances. It can explain observation to some extent, however, difficulties still exist in some aspects of the multi-phase model.

The first problem is the unknown initial cloud mass function. According to the description in section 3.2.6, the distribution of cloud mass follows a power law with a slope of ~ -2 confirmed by both observations and theories (Fukui & Kawamura 2010; Dobbs & Pringle 2013). As we can see in figure 3.12, a power law distribution already forms after the system evolves for 1 Gyr . However, the mass of 1.98×10^5 $M_\odot$ for all cloud particles initially, is too arbitrary and artificial to be taken in the initial condition. As demonstrated in figure 3.11, different initial values of cold cloud particle yield quite different star formation history, which means our model strongly depends on the initial configuration of cloud mass. A good model should avoid such kind of dependence. One method is to assign the mass of cloud particles according to the already known cloud mass distribution. However in this way the power law distribution becomes input instead of output. We want to find a reasonable method which still guarantees the power law distribution as output, and avoids the problem of initial cloud mass assignment meanwhile.

Another problem is the unknown initial cloud mass fraction. In the model of previous chapter, we fix the fraction of cloud mass to be 98% of the total mass in the initial condition. As the evolution of the mass shown in figure 3.5, part of the cloud mass transits to the hot/warm phase via evaporation, another part transits to stars via star formation. The star formation rate is roughly proportional to the amount of cold gas, which means the star formation process is

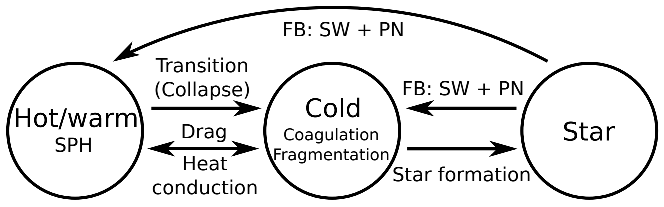


Figure 4.1: Demonstration for the hot/warm transition model

actually depends on the choice of the initial fraction of cloud mass. In one extreme case, no star will form if the initial fraction of the cloud mass is zero. Moreover, if there is no cold cloud particle, heat conduction (condensation and evaporation) will not happen. In a good model, the star formation rate together with other physical processes should be self-regulated, which does not depends on the initial configuration of the model.

Compared with single-phase model, the multi-phase model involves more physical processes between the phases, and therefore introduces the problems described above. We have to admit that the multi-phase description of interstellar medium is more reasonable compared with single-phase, however its complex physical processes are more difficult to manipulate to give reasonable results. The two problems mentioned above can be solved if the hot/warm gas forms dense clumps by following a mass spectrum and transits to the cold phase in an appropriate rate. We therefore implement the transition from the hot/warm phase to the cold phase. The basic idea is, for a clump of hot/warm gas, if its density is high enough, and temperature is low enough due to radiative cooling, we assume it will collapse to a cold cloud.

The transition of hot/warm gas can be regarded as a variation of the popular single-phase model, and combines the advantage of the multi-phase description, such as the modeling of cold clouds, the star formation inside the cold cloud, etc. As described in section 1.3.1, the "Katz" criterion (Katz 1992) is commonly used as the star formation recipe in the single-phase model. However, the density criterion of $0.1/\text{cm}^3$ to form stars is even lower than the typical density ($10/\text{cm}^3$) of giant molecular clouds (Gillmon & Shull 2006). Therefore, compared with the single-phase model in which stars are created from the dense and cold regions of SPH gas directly, it is more reasonable for these regions first collapse to molecular clouds. Then the molecular clouds further collapse and form stars inside. Figure 4.1 demonstrate the model by taking the hot/warm gas transition process into account. All the other processes are kept unchanged as the classical multi-phase model demonstrated in figure 1.3. Also it is somewhat similar with the single-phase model, by introducing a new cold phase between the SPH gas phase and stars. Heat conduction (condensation and evaporation) still exists in the model, but can only be regarded as a second order effect for mass exchange between phases, since its rate is usually lower compared with the hot/warm gas transition.

Actually the transition of hot/warm gas is already not a new idea for various types of multi-phase models. Some authors attempt to model the multi-phase structure of ISM in the single-phase model by modifying the SPH algorithm, since the classical SPH algorithm cannot deal with the large density gradient between the cold and hot phases. Pearce et al. (1999, 2001) prevent the hot gas with temperature above 10^5 K from interacting with with cold gas with temperature below 12000 K but not vice-versa. All other SPH forces remain unchanged. This implementation effectively prevents the over cooling of hot gas induced by the presence of nearby cold gas. Hot gas is decoupled from the cold gas, a two phase distribution is formed according to the density and temperature of SPH particles. Marri & White (2003); Scannapieco et al. (2006) further develop this idea. In their implementations, besides temperature (represented as internal energy and entropy of the two models), density of gas and velocity divergence are also taken into account for the decoupling of the two phases. Moreover, the feedback scheme is modified accordingly. A fraction ϵ_c of supernova energy is injected to the cold phase and mainly radiates away, and another fraction ϵ_h is distributed in the surrounding hot phase. The low density of hot phase leads to the low cooling rate, supernova energy therefore drives the hot gas expanding efficiently. Particles transit between the two phases freely, if their density and temperature fall into the corresponding range of the phase.

Another approach to describe the multi-phase structure is the so called hybrid model (Springel & Hernquist 2003; Scannapieco et al. 2006; Booth et al. 2007; Murante et al. 2010), in which every particle is assumed to be composed of several components (hot gas, cold gas and stars). Various physical processes, such as gas cooling, star formation and stellar feedback, are modeled inside the particle as a set of ordinal differential equations. In this way the transition between hot and cold phases are modeled under the regulation of heating and cooling processes. By adjusting the coefficients carefully, the hybrid model can reproduce the Schmidt–Kennicutt relation, the star formation rate, the galactic outflows, etc. Some phase transition models involve complex calculation of physical processes, such as the implementation of HI to H_2 transition in Pelupessy et al. (2006). However, the sub-grid physics which determine heat conduction, cooling, star formation, stellar feedback, etc., are still not clear, which means incorporating all of these equations to numerical models can only lead to larger deviations. In another words, the more we did, the more mistake we make. What we can do is to construct a phenomenological only model, and keep the picture simple. This is the principle for our implementation of hot/warm phase transition, as I will introduce in the next section.

In the following part of this chapter, I will first give a introduction to the transition process from hot/warm phase to the cold phase. Then I present our work of parameter study on six galaxies by taking the phase transition into account.

4.2 Introduction to the transition process

Our implementation of transition from the hot/warm phase to the cold phase is different from the two main approaches described in previous section, but still can be regarded as a combination of them. The basic idea is to first distinguish particles in the cold clumps based on their density and temperature. Then these clumps are identified with a similar method of identifying the sub-structures in the dark matter halos which is quite common in the cosmological simulation. Finally we assume these clumps will collapse to the cold clouds if they fulfill some criteria. In general, this process can be summarized in the following steps:

(a) Identify candidate particles: a hot/warm particle is regarded as a candidate particle

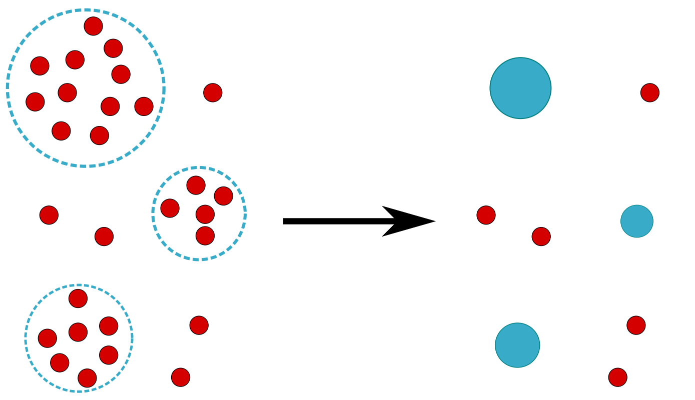


Figure 4.2: Demonstration for the hot/warm transition process

which might form cold clumps if it fulfills the following criteria:

$$T < T_{\rm crit},$$

$$\rho > \rho_{\rm crit}.$$

These two critical values can be borrowed from the star formation recipe of single-phase model, since we assume the diffuse gas will first collapse to cold clouds and then form stars, which corresponds to the star forming process in the single-phase model. For galaxy evolution, $T_{\rm crit}$ and $\rho_{\rm crit}$ take values in the order of 10^4 K and $0.1/{\rm cm}^3$.

(b) Create neighbour list: find the nearest N_B neighbours for all of these candidate particles. N_B is the neighbour number for SPH calculation which is set to 50.

(c) Create groups: link particles using a Friend of Friend (FoF) method with a linking length b: if the distance of two particles is shorter than b, they are linked together and are labeled as belonging to the same group. This is done in the code by checking the N_B neighbours for every candidate particle. In this way the candidate particles are assigned to a set of groups.

(d) Create peaks: for every group created in step (c), sort particles inside the group according to their density. Start with the highest density, for every particle i in the group, check its nearest $N_{\rm NGB}$ neighbours. If there is a neighbour particle j in the same group and belongs to a peak, assign particle i also to this peak. If there is no neighbour with higher density in the same group, create a new peak based on particle i for this group. This algorithm guarantees that neighbour particles in the same group with higher density already belong to some peaks, otherwise there must be something wrong. $N_{\rm NGB}$ is a free parameter, which controls the size of peaks. Larger $N_{\rm NGB}$ means more massive peaks. Usually it is set to 20.

(e) Transit to cold cloud: for every peak in one group, check if its mass is higher than the minimum mass allowed for transition. If yes, further check if its cooling time scale $t_{\rm cool}$

is shorter than its dynamic time scale $t_{\rm dyn} \sim 1/\sqrt{G\rho}$. We assume that the cold clump (peak) should experience sufficient cooling when collapse to cold clouds. Finally we fix the transition probability as

$$p_{\rm transit} = \Delta t_{\rm transit}/t_{\rm dyn}.$$

Here $\Delta t_{\rm transit}$ is the time interval to check transition.

(f) Create cold cloud: when the peak fulfills all the above criteria and passes the probability check, it transits to a cold cloud particle. All the candidate particles in the peak are removed from the hot/warm particle list. The newly formed cold cloud takes the center of mass as its position, the mass weighted average velocity as its velocity. The radius are calculated according to the *mass-radius* relation.

Figure 4.2 demonstrate the transition from hot/warm gas to cold clouds. The most technical part is to identify these cold and dense regions encircled by the blue dashed line. One may notice that step (c) and (d) are somewhat similar with the method of identifying the substructures of dark matter halos in the cosmological simulation. More specifically, our treatment is a simplified version of the SUBFIND algorithm (Springel et al. 2001a). In the SUBFIND algorithm, the saddle points which connect two subhalo candidates are more likely to be assigned to the smaller subhalos, since the purpose the algorithm is to identify the substructures. In our treatment, the candidate particle is always assigned to its nearest high density peaks, which simplifies the treatment of saddles, since we do not care about the substructures of cold clumps.

Figure 4.3 give an example of the transition process using the method described above. Initially, 5×10^5 SPH particles with a total mass of 5×10^7 M$_\odot$ are randomly distributed in a cubic box with a box size of 100 pc. The simulation is setup for the study of supernovae effect on a turbulent field of pc scale. The reason of not using a galaxy scale simulation is, it is much clearer to demonstrate this process in a background of uniform density distribution, which is quite common in the study of turbulent field in the giant molecular clouds. We choose a linking length b of 0.2 pc, a minimum transition density of $10^4/\rm cm^3$ and a maximum transition temperature of 1000 K for the transition process. As a test of the transition process only, other processes, such as coagulation between clouds, star formation, stellar feedback, etc., are switched off (cooling is still included due to its influence on gas temperature). According to the snapshots, the cold cloud distribution correlates well with the gas distribution. Massive clouds are always created in the higher gas density region. Figure 4.4 demonstrate the distribution of cloud mass at 0.7 and 1.3 Myr. Compared with the distribution at 0.7 Myr, there are more massive clouds at 1.3 Myr, since the gas becomes more clustered as the system evolves. Moreover, the cloud mass distribution follows a power law with a slope of -2 at 1.3 Myr, which is consistent with previous study (Harfst et al. 2006; Fukui & Kawamura 2010; Hopkins 2012). This fulfills our requirement that the mass of newly created clouds follows a reasonable distribution. Because of the hierarchical structure of molecular clouds, this transition process can be used in different scales.

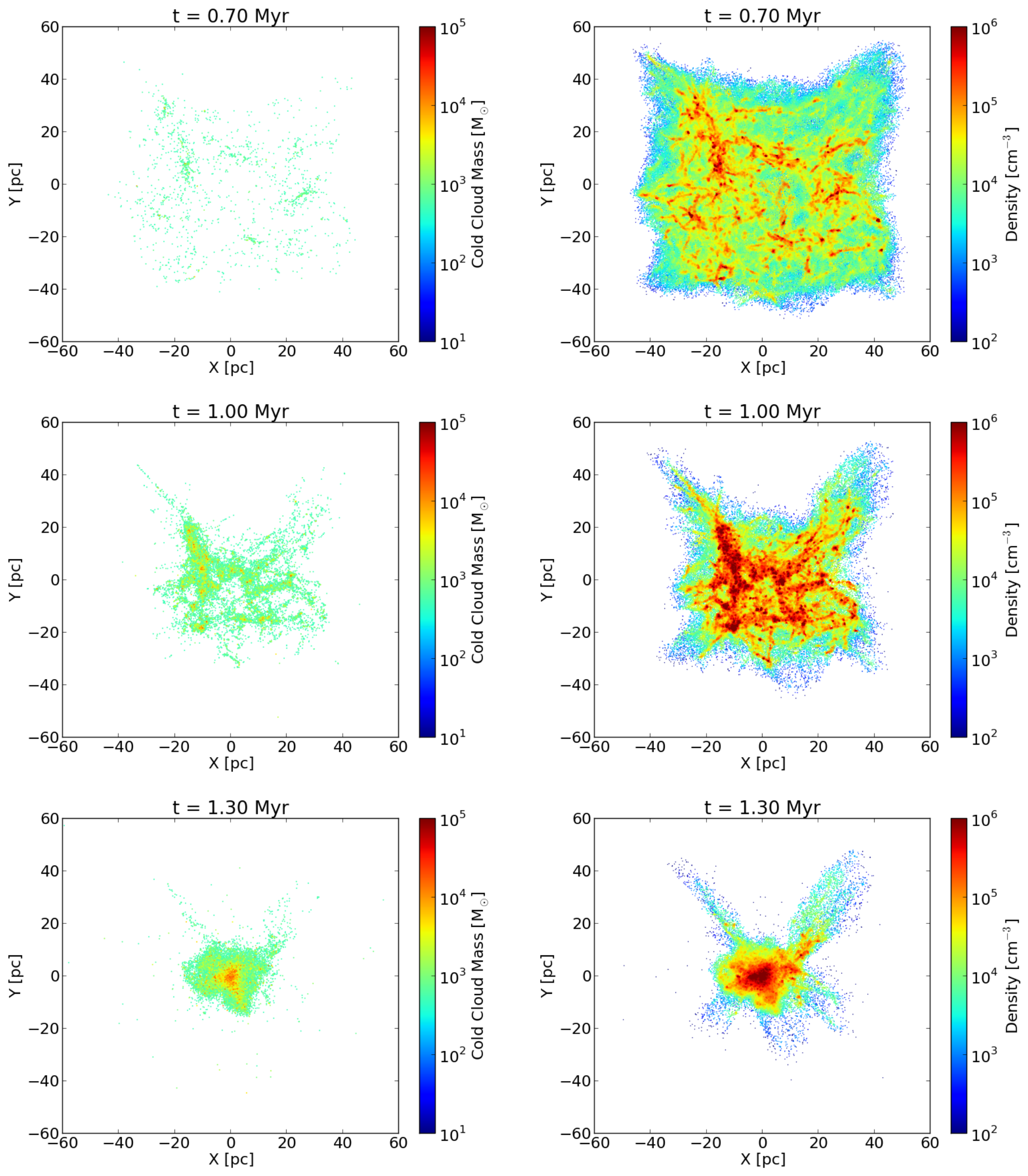
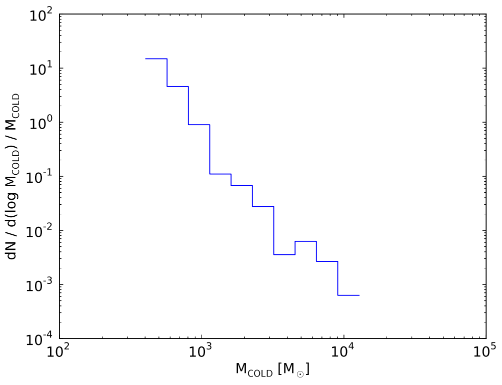
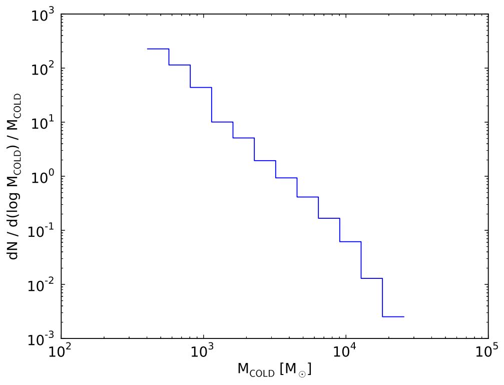


Figure 4.3: Simulation of the transition process in a cubic box. Left column: the distribution of molecular clouds generated by the hot/warm transition process. The cloud mass spectrum is mapped to different colors, as specified by the color bar. Right column: the density distribution of hot/warm gas at the corresponding time. Upper, middle and bottom panels represent three different times: 0.7 Myr, 1.0 Myr and 1.3 Myr.



(a) t = 0.7 Myr.



(b) t = 1.3 Myr.

Figure 4.4: Cloud mass distribution at 0.7 and 1.3 Myr.

4.3 Parameter study of galaxies

The simulation in previous chapter is carried out using the serial version of our multi-phase code. In the past two years, I parallelized the code such that the code can run on the modern computer clusters of distributed memory. Now the code supports a much larger particle number (5 million particles) and runs in a relatively faster speed after the SPH and neighbour search parts are also accelerated using GPU [1]. With this powerful code, on one hand, we can do some simulation with higher resolution, such as the individual star formation model in the next chapter. On the other hand, the parameter study can be carried, since it is always time consuming especially when the parameter space are large. Our GPU accelerated parallel code is proved to be extremely suitable for such kind of work, as I will introduce below.

4.3.1 Setup of initial conditions

In this section, I present our work on the parameter study of galaxies with different mass and rotation. The phase transition described in the previous section is included to generate the cold phase. The basic idea is to compare the properties of galaxies with different mass and rotation. As the first step, we prepare the initial conditions for galaxies with initial mass of 2×10^{10} M$_\odot$, 2×10^{9} M$_\odot$ and 2×10^{8} M$_\odot$. For the two galaxies with the same mass, we consider the case of no rotation and 35% of the circular velocity. In total, six galaxies are setup for comparison.

We use the Plummer-Kuzmin density profile (Miyamoto & Nagai 1975) for the initial distribution of gas, and the Burkert profile (Burkert 1995) for the modeling of an analytical dark matter potential. The 91 and 90 models are assumed to have similar density distribution and dark matter halo as the dwarf galaxy model in the previous chapter. For the rest four models, their sizes are adjusted according to their mass, as listed in table 4.1. The central density ρ_0 of Burkert profile is calculated according to the scaling relation between r_0 and ρ_0.

[1]A detailed introduction to these numerical issues is available in chapter 6.

Table 4.1: Parameters of the six models. a and b are for the Plummer-Kuzmin profile of initial gas distribution. ρ_0 and r_0 are for the Burkert profile of dark matter distribution.

Model	Total mass	Particle	Rotation	b_{FoF}	a	b	ρ_0	r_0
name	($\mathrm{M}_\odot$)	number		(kpc)	(kpc)	(kpc)	($\mathrm{M}_\odot/\mathrm{pc}^3$)	(kpc)
101	2×10^{10}	2×10^6	Yes	0.02	0.43	1.62	0.013	6.463
100	2×10^{10}	2×10^6	No	0.02	0.43	1.62	0.013	6.463
91	2×10^9	2×10^6	Yes	0.01	0.20	0.75	0.022	3.000
90	2×10^9	2×10^6	No	0.01	0.20	0.75	0.022	3.000
81	2×10^8	2×10^5	Yes	0.01	0.093	0.35	0.036	1.392
81	2×10^8	2×10^5	No	0.01	0.093	0.35	0.036	1.392

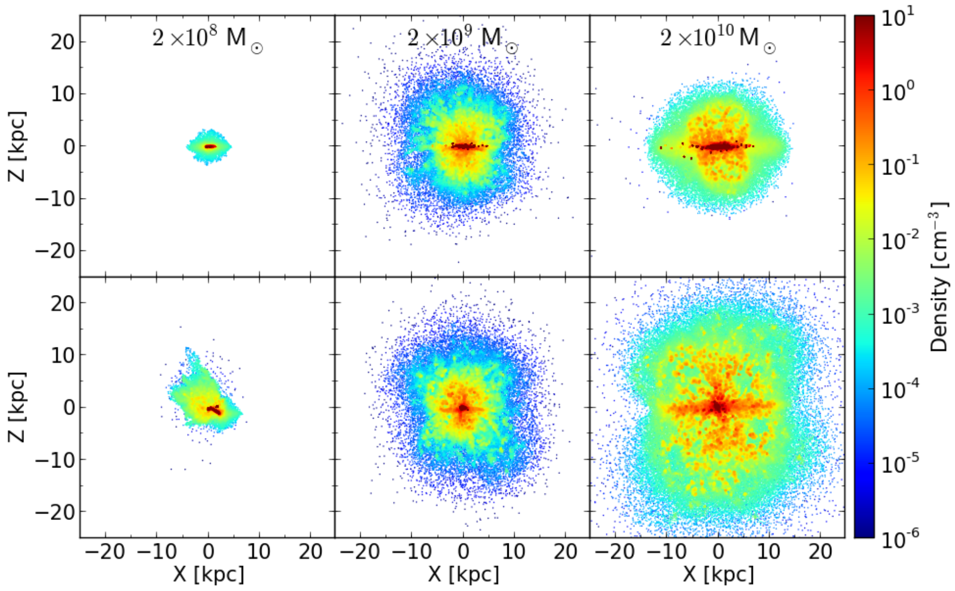
According to the relation, smaller size corresponds to a core with higher density, as low mass galaxies are dark matter dominated. As listed in the table, model 81, 80 and 91, 90 have the same mass resolution. In model 101 and 100, the mass resolution is one order lower. The main reason is the "laohu" cluster cannot support support a run with 20 million particles for quite a long time (to do that we have to use at least 16 nodes and keep the cluster busy for one week or even longer due to the load imbalance).

All the models start with a setup of pure hot/warm particles. The cold clouds are generated via the transition process. This help us avoid the problem of initial cloud mass fraction and distribution, especially when we have to deal with different parameters of mass and rotation. The criteria for transition are $\rho_{\mathrm{crit}} = 0.2/\mathrm{cm}^3$, $T_{\mathrm{crit}} = 10^4$ K. The linking lengths b_{FoF} for different models have been listed in the table. In model 101 and 100 this value is two times larger than it is in other four models, since the corresponding mass resolution is one order lower. We have to point out that the transition process is highly experimental, there is still no commonly accepted recipe to determine the linking length. The six models are kept running until 2 Gyr. We choose a timestep of $\mathrm{d}t = 0.01$ Myr. The parallel architecture together with the GPU acceleration make such kind of computing intensive task possible: the 2 million particle run utilizes 8 GPUs and keeps the four nodes (8 threads) of the "laohu" cluster busy for more than two days.

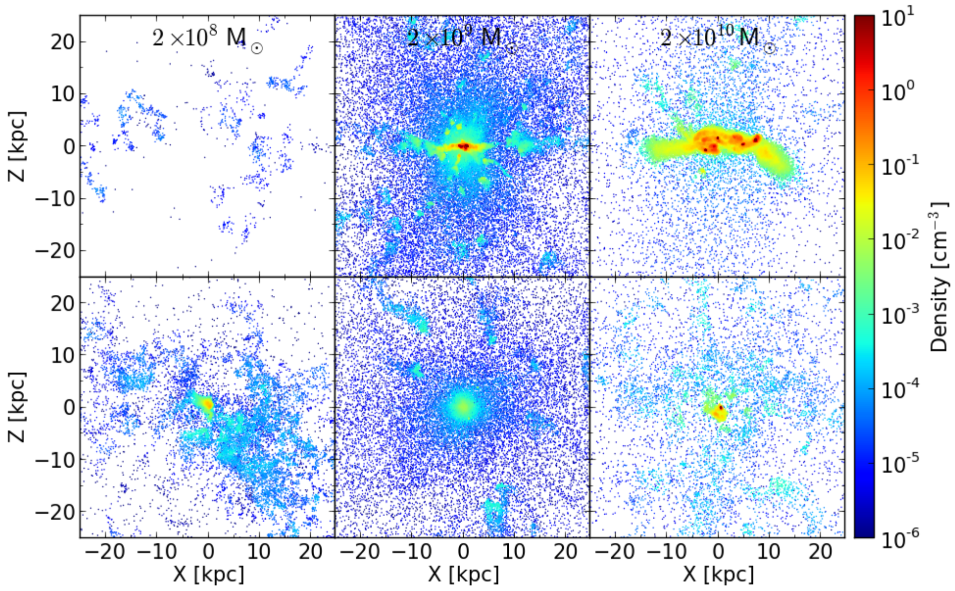
4.3.2 Evolution of the models

As for the dwarf galaxy model in chapter 3, we present the edge on snapshots, evolution of mass and mass transition rate, star formation rate, cloud mass function and velocity distribution of the models.

Figure 4.5 and 4.6 present the edge on view snapshots of hot/warm gas and cold clouds at 0.1 and 1.0 Gyr. When the simulations start, most of hot/warm gas transits to cold clouds in the first 100 Myr. Stellar particles form inside the molecular clouds. Their feedbacks disrupt the clouds and drive the hot/warm gas expanding. In the rotation case, gas collapses mainly along the z-axis, and forms a disk in the xy-plane. Cold clouds follow the distribution of

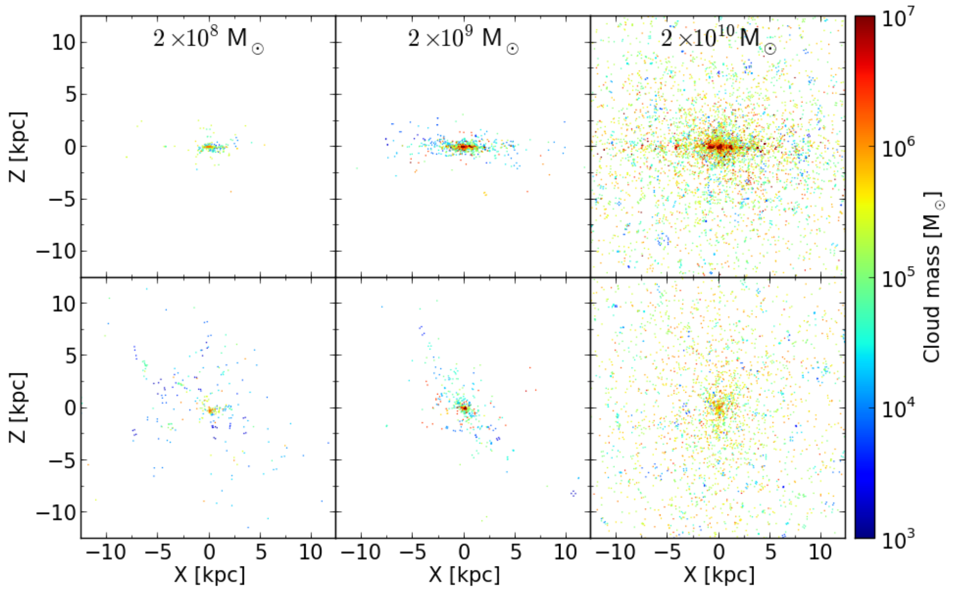


(a) t = 0.1 Gyr.

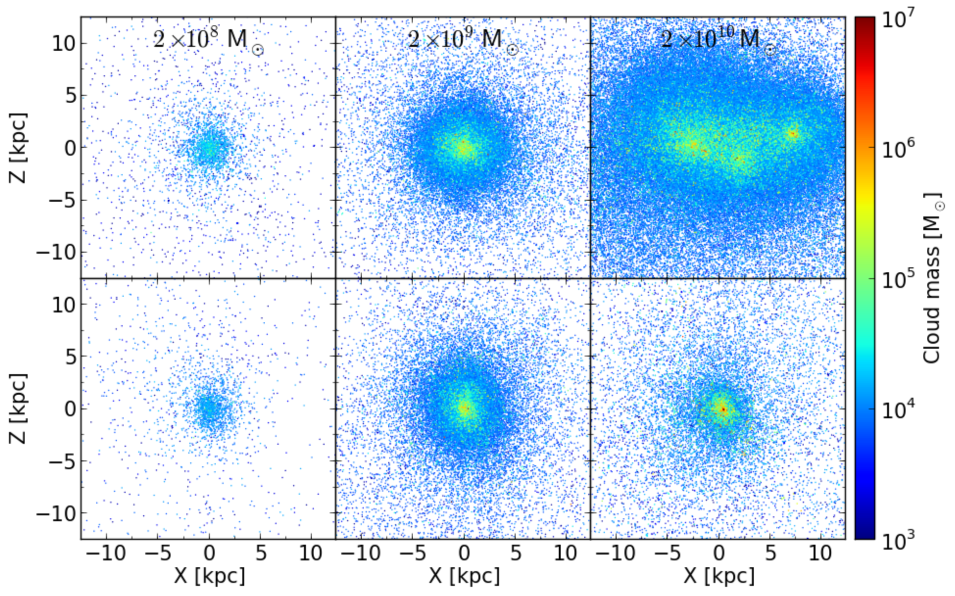


(b) t = 1.0 Gyr

Figure 4.5: Edge on view snapshots of hot/warm gas density at 0.1 and 1.0 Gyr.



(a) t = 0.1 Gyr.



(b) t = 1.0 Gyr

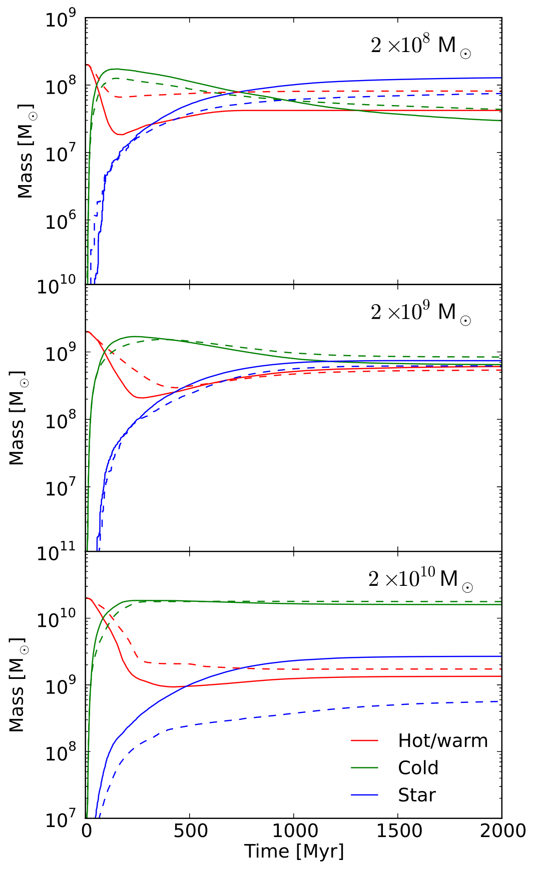
Figure 4.6: Edge on view snapshots of cold clouds at 0.1 and 1.0 Gyr. Color bar represents the corresponding cloud mass.

hot/warm gas, forming disk with similar size. The main difference between the rotation and non-rotation models is, in the non-rotation model, gas collapses not only along the z- axis, but also radially to the center. Therefore a small core forms instead of a disk. One may note that in the outer regions, the low mass particles with deep blue color are not observed in the 101 and 100 models. This is simply a matter of mass resolution: in 101 and 100 models the initial mass of hot/warm particle is one order higher than that in the low mass models.

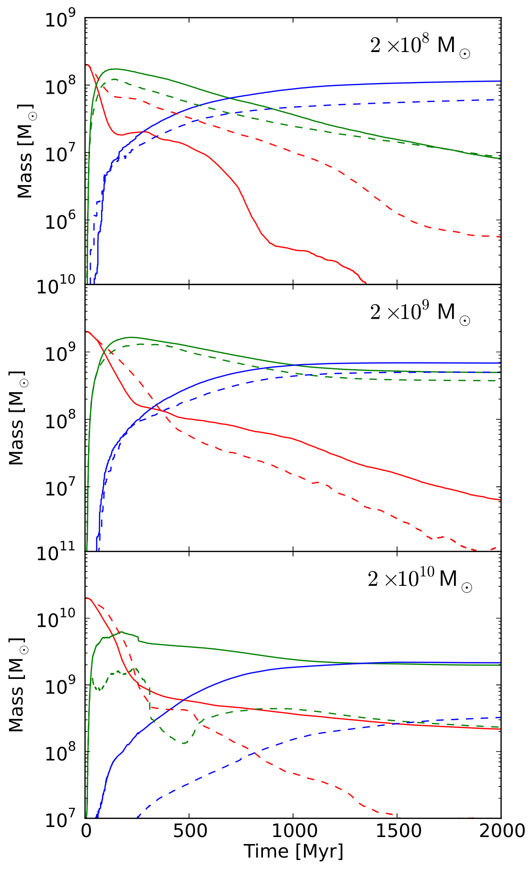
One thing I have to point out is, the density of hot/warm gas is very high ($> 10/\mathrm{cm}^3$) at the beginning of the simulations, since we start from a pure hot/warm gas initial condition. According to equation 2.9, the drag force is large. The motion of cold clouds are strongly coupled with hot/warm phase. When the simulations start, hot/warm gas collapses to the center (or along the z-axis in the rotation case) with high speed. The corresponding cold clouds also obtains this high speed via drag force between the phases. In current implementation, these clouds will collide with other low speed clouds when they leave the galaxy with high speed. A significant amount of mass is taken away in this way. Currently there is no good solution for this problem, if the initial density of hot/warm gas is so high. What we can do is to switch off drag force in the current model. This will reduce the dynamical coupling of the two phases, but the general evolution does not change too much. Even with this modification, the velocity of newly formed clouds are still too high in the massive galaxies (100 and 101). In the non-rotation case (100), most of the cloud particles leave the galaxy with high velocity. In the rotation case (101), although most of the clouds still stay in the galaxy, the disk structure is very unstable due to the high angular momentum. We can see several cold cloud “cores” in the snapshot at 1.0 Gyr.

Figure 4.7 present the mass evolution of three components. For the total mass evolution, at the beginning of simulations, hot/warm gas transits to cold clouds very quickly in the first 100 Myr. Then stars form inside the cold clouds. Their feedbacks heat the surrounding hot/warm medium. Due to the high temperature and low density of hot/warm gas after the initial collapse phase, evaporation is strong. Mass transits back to the hot/warm phase from the cold phase. For the mass within 20.0 kpc, the evolution of cold and stellar components for model 80, 81, 90 and 91 are almost the same as they are in the total case (left panel), which means most of these particles stay inside the galaxy. For the two massive galaxies, both of them lose most of their cloud mass in the initial collapse phase, and even more mass is lost in the non-rotation case. For the evolution of hot/warm gas within 20.0 kpc, the mass always decreases, which means gas is leaving the galaxy, although more mass is transiting from the cold phase via evaporation. By observing the stellar mass evolution, one may find that the stellar mass in the non-rotation models is always lower than that in the rotation models. This is also confirmed by the evolution of SFR in figure 4.8: the SFRs of non-rotation model are always lower than that of rotation models. This can be explained by the lower cloud mass in the non-rotation models, since the SFR is proportional to the cloud mass with current SF recipe. However, this rises another question: why the cloud mass in the non-rotation models are lower than that in the rotation models. This question cannot be fully answered at present. What we know is this must be related with the transition process: from the figure the cloud mass is already lower in the non-rotation case at the beginning of the simulation.

Figure 4.9 demonstrate the mass transition rate as a function of time for model 91. Other



(a) General.



(b) Within a radius of 20 kpc.

Figure 4.7: Mass evolution of the six models. Red, green and blue lines correspond to the hot/warm, cold and star components. Solid and dashed lines correspond to the rotation and non-rotation models, respectively. The left panel represents the total mass evolution, in which all particles are taken into account, including those far away from the galaxy center and are thus removed from the system to save the computation resources. In the right panel, only those inside the radius of 20 kpc are taken into account, which represents the actual mass of the galaxy.

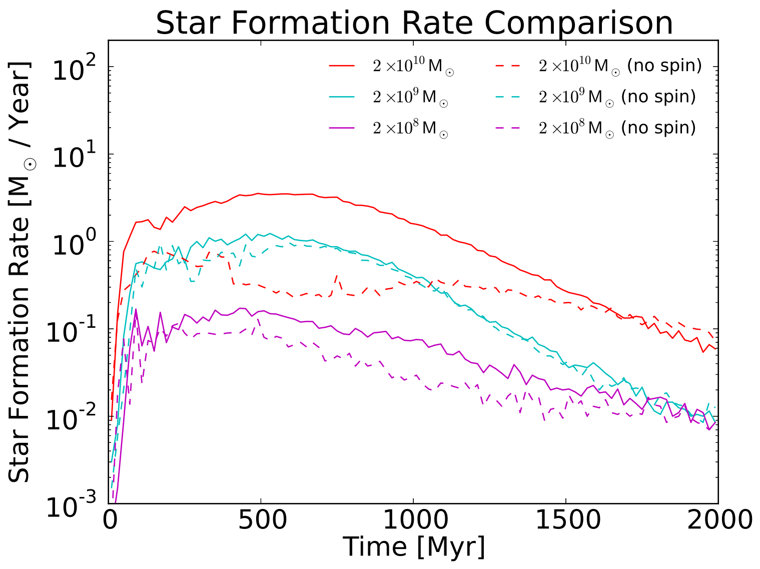


Figure 4.8: Evolution of star formation rates in the six models.

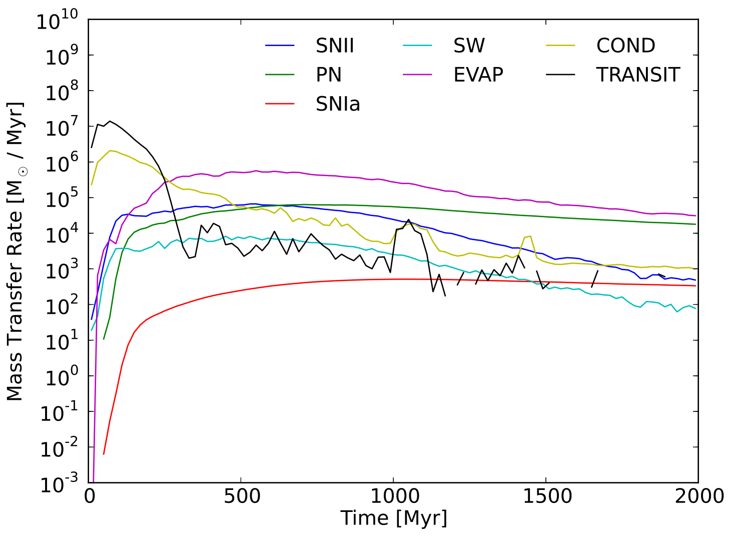
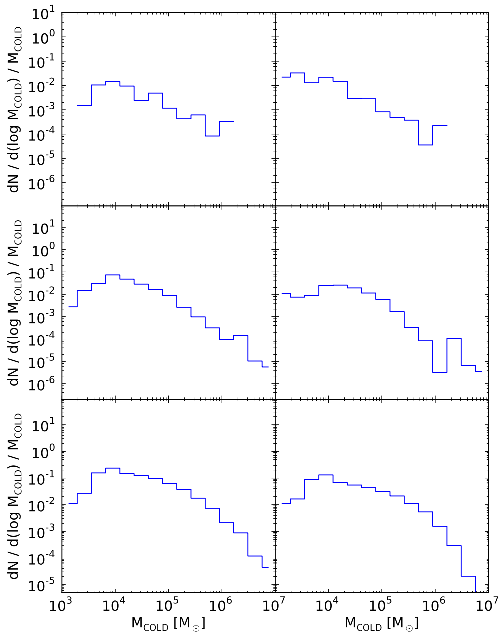
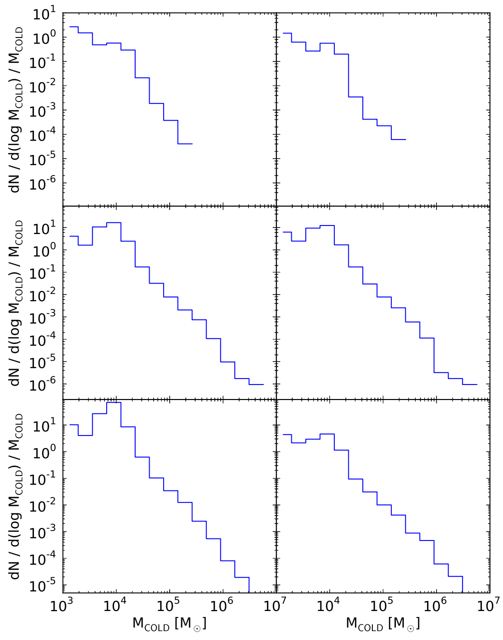


Figure 4.9: Evolution of mass transition rate in model 91. The transition rates of another five models are not shown here since their evolutions are similar.



(a) t = 0.1 Gyr.



(b) t = 1.0 Gyr.

Figure 4.10: Cloud mass distribution at 0.1 and 1.0 Gyr. For each panel, histograms in the left and right columns correspond to the rotation and non- rotation models, respectively. The top, middle and bottom rows correspond to galaxies with a mass of 2×10^8 $M_\odot$, 2×10^9 $M_\odot$ and 2×10^{10} $M_\odot$.

models are not shown since their evolution are similar. At the beginning of the simulation, hot/warm gas collapses and transits to cold clouds, which provides the seeds for the hot/warm gas to condense to. Both transition and condensation rates are high in this stage. After that stellar feedback especially type II supernovae inject a large amount of energy to the hot/warm phase, which drives gas expanding. With the high evaporation rate, mass transits from cold phase back to the hot/warm phases, the total mass of hot/warm gas increases, which is demonstrated in figure 4.7 of mass evolution.

Figure 4.10 demonstrate the cloud mass distribution for six models at 0.1 and 1.0 Gyr. Very clearly, the slope at 0.1 Gyr is much flatter than that at 1.0 Gyr, which means there are more low mass clouds at the late stage of the evolution. This is also shown in the snapshots of cold clouds. In the current treatment, low mass clouds are created when stellar feedback of newly formed stars disrupt the massive clouds. We set the minimum mass of stellar particles to be 10^4 $M_\odot$. Clouds around this mass cannot form stars and thus further fragment. Meanwhile, their coagulation rate is low since their cross section is small. More and more small clouds form a peak around 10^4 $M_\odot$. For all the six models, the distribution follows a power law with a slope of -2, which confirms the simulation results in section 3.2.6 and is consistent observations and theoretical predictions (McKee & Ostriker 2007; Fukui & Kawamura 2010;

Hopkins 2012). This result also suggests that the clouds with current implementation have a strong ability of self-regulation. Its final distribution does not depends on the initial configuration so much. Another interesting thing is, from the figure we cannot differentiate the rotation and the non-rotation models, which means rotation does not affect the cloud mass distribution too much.

4.3.3 Summary

In the previous section, we present the simulation of six galaxies in three different mass ranges with and without rotation. All models start with pure hot/warm gas distributed with Plummer-Kuzmin density profile. Cold component is generated by the hot/warm transition process. Burkert profile is adopted to provide an analytical dark matter potential. The parameters for these models are listed in table 4.1. All models are evolved for 2.0 Gyr, during which the evolution of mass, mass transition rates and SFR, together with the snapshots of hot/warm gas and cold clouds, cloud mass distribution at 0.1 and 1.0 Gyr are recorded. The main results for the investigation of the six models and the experimental hot/warm transition process are listed below:

- In the initial collapse phase, more hot/warm gas transits to the cold phase in the rotation models compared with the non-rotation models. This discrepancy of cold clouds mass further affects the star formation rate.
- Some of the cold clouds escape from the galaxy in high velocity. They coagulate with other clouds when leaving the galaxy. A large amount of cold cloud mass is taken away from the galaxy in this way. We find that this is mainly due to the drag force between the cold and hot/warm phases. In the initial collapse phase, the density of hot/warm phase is high, thus leads to the unphysically large drag force on some cold cloud particles. To prevent the cold clouds from escaping in high velocity, drag force must be switched off. However, some newly formed cold clouds still inherit the high velocity of the collapsing hot/warm gas, which is more likely to happen in massive systems. E.g., the two massive galaxies with mass 2×10^{10} $\mathrm{M}_{\odot}$ are still not stable even though the drag force has been switched off. In this sense the hot/warm transition process is more applicable for smaller object. This can be regarded as the shortcoming of current hot/warm transition process.
- The clouds mass distribution follows a power law with a slope of ~ -2 after 1.0 Gyr, which is consistent with observations and theoretical predictions. The behavior of cold clouds is highly self-regulated, which means the cloud mass distribution does not depend on the initial configuration so much. Hot/warm transition process is only effective at the beginning of the simulation for the creation of cold clouds, but will not affect the evolution of the system so much.

In the next chapter, I will present the individual star formation model, in which hot/warm transition process is still used to generate the cold phase. The shortcoming of hot/warm transition process is, the velocity for some of the newly generated cold cloud particles are too high even if drag force has been switched off. One possible solution is to first carry out a pure N-body/SPH run by taking cooling into account. The extra kinetic energy generated during the initial collapse phase will transit to thermal energy via SPH viscosity, and radiate away by cooling. The real simulation starts after the system relaxes for a sufficient time and reaches equilibrium. In this way the high velocity of cold clouds can be avoided. More test is need to verify this idea.

"High-brow music receives little understanding from people; highly reputed people rarely live up to their reputation."

Hou Han Shu, Fan Ye

5 An individual star formation model

In this chapter, I present our individual star formation model. The basic idea is to create stars in molecular clouds analytically with mass distribution given by a specific IMF. We trace the evolution of individual stars in the system and study their feedbacks to the surrounding interstellar medium. I will first explain our motivations for this study. Then I introduce our implementations of the related physical processes based on the theory of star formation. After that I present two applications of the individual star formation model. One is in galaxy scale, another is in star cluster scale. Finally there will be a summary.

5.1 Motivations

As will be introduced in the next chapter, I parallelize the multi-phase code and speedup the most time consuming parts using GPU. Currently the code is able to run in up to 32 threads and supports a particle number as large as 20 million. With this powerful code and the huge computing resources of GPU clusters, it is possible to carry out several studies. One possibility is parameter study, as we have done in previous chapter on six galaxies with different mass and rotation. Although it is not so time consuming for every individual model, the total time consumption for the investigation of the whole set of parameters is still large. Another possibility is to model some large objects, such as the Milky Way sized disk galaxy. The third possibility is to model some small objects with high resolution, such as the evolution of a low mass dwarf galaxy, or the formation of star cluster inside one molecular cloud.

Several reasons motivate us for this study. The first reason, from engineering side, both our softwares and hardwares now allow us to carry out such a large simulation that utilizes so many computation resources simultaneously and for such a long time. It is a big challenge to coordinate the work load between CPUs and GPUs, and the communication between different threads. Also it is difficult to keep the GPU cluster busy for more than three days and integrating 1.5×10^5 timesteps without making any mistake. What's more, the various types of physical processes and large particle number make this problem even more complex. It is a good opportunity to investigate the potential of the code and the GPU cluster.

The second reason is, we are the first to investigate the idea of individual star formation inside the molecular cloud. It would be very interesting to trace the evolution of every individual star and study their feedback to the surrounding ISM. The study of numerical star formation can be classified into two types according to the scale they investigate. In the small scale of inside one molecular cloud, the formation of individual star is studied by implementing the necessary physical processes, such as gas accretion and radiative heating. These small scale models can explain the mechanism for the formation of massive stars and reproduce some observations, e.g., the observed relation between total stellar mass and the mass of the largest star in the stellar cluster (Peters et al. 2010). In the galaxy scale, star particles in the form of single stellar population with a mass as high as 10^4 $M_\odot$ are created from molecular clouds (multi-phase model) or SPH particles (single-phase model) analytically. Their feedbacks such as supernovae to the surrounding ISM are also modeled with some methods (Harfst et al. 2006; Stinson et al. 2006). We try to combine these two scales, although individual star inside the molecular clouds are still created analytically. The power of modern GPU cluster makes this study possible. Another advantage for this study is, most of the processes we have to consider are already there, such as gravity, gas dynamics, phase transition, cooling, etc. Moreover, the software architecture can be kept unchanged. What we need to do is just rewriting the routines related with star formation, such that individual stars are created according to IMF. What's more, the feedback part needs to be updated. The reuse of existing routines greatly reduces our work load compared with writing a code from scratch.

The third reason is, this model provides the multi-phase based framework for the successive study on the relation between interstellar medium and stars, such as the formation of star clusters in the turbulent velocity field.

5.2 Implementation of physical processes

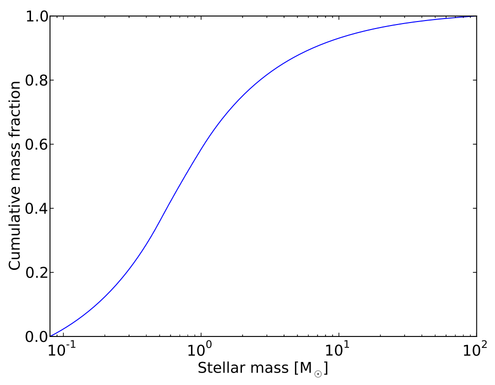
In this section, I introduce the detailed implementation of the model. Since the model is based on previous multi-phase model, most of the physical processes, such as the gravity, gas dynamics (described by SPH), condensation and evaporation, hot/warm transition, are kept the same and have been introduced in chapter 2 and 4. Therefore I will focus on the newly implemented parts, which are the individual star formation inside molecular clouds and the corresponding stellar feedback.

5.2.1 Individual star formation

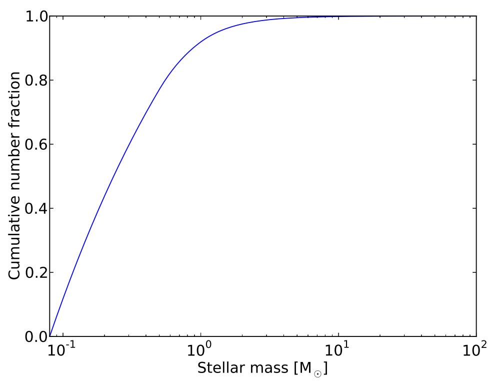
We implement the star formation recipe based on several principles:

- Stars form in the cluster environment inside the molecular clouds with a star formation efficiency of 10 % [1].
- Stars are created inside the clusters with the mass distribution determined by a given IMF.
- The cloud will be destructed by photoionization if massive stars (> 8 $M_\odot$) form inside. Later we will see this destruction process is crucial in the recycle of interstellar medium.

[1]In the Milky Way, the observed value is around 0.05 but varies by more than 2 orders of magnitude from cloud to cloud (Williams & McKee 1997). Therefore the 10% is a reasonable value.



(a) Cumulative mass.



(b) Cumulative number.

Figure 5.1: Cumulative mass and number fraction as a function of time according to IMF (Kroupa et al. 1993). The cumulative mass are integrated from the low mass limit (0.08 M$_\odot$) to the high mass limit (100 M$_\odot$).

We set a maximum allowed temperature for star formation, currently this value is 200 K. Meanwhile, a probability is set to control the star formation rate, which is model dependent. Our star formation recipe is simple, since it does not involve any specific physical process, but just a phenomenological description based on current observation of star forming regions.

According to the different types of feedback, stars are divided into four types:

- Type II SN stars. Stars with mass higher than 8 M$_\odot$. They are massive stars which will end their life as type II supernovae or black holes. Before that they release energy via stellar wind feedback (Portinari et al. 1998).
- Type Ia SN stars. They are binary systems which will explode as type Ia supernovae. The lifetime of the system is determined by the lifetime of the secondary star (Greggio & Renzini 1983). We assume that the number of type Ia SN events in one single stellar population is 20 % of type II SN events (Portinari et al. 1998).
- PN stars. Intermediate mass stars with mass between 0.8 M$_\odot$ and 8 M$_\odot$. They loss most of their envelope mass in the asymptotic giant (AGB) phase and form planetary nebulae at the end of their life (van den Hoek & Groenewegen 1997).
- Low mass stars (particles). Stars with mass between 0.08 M$_\odot$ and 0.8 M$_\odot$ are represented by one particle as a single stellar population in this mass range. The single particle is assigned a mass of 1.6 M$_\odot$. An extensive explanation for this treatment is given below.

As mentioned in previous section, brown dwarfs with mass less than 0.08 M$_\odot$ have been excluded from the IMF. Stars with mass between 0.08 and 0.8 M$_\odot$ are classified as red dwarfs stars. They evolves slowly and have a lifetime of more than 10 Gyr. Their luminosity is low and have little effect to the surrounding ISM. Therefore they can be regarded as N-body particles in our model over the typical simulation time (2 Gyr). According to figure 5.1, they take up 50% of the total mass in a single stellar population. However, their number fraction is higher than 80%. This means the stellar component will mainly consist of such kind of N-body like particles, if we sample the IMF until the low-mass end, which will consume a significant amount computation resources. To avoid this problem, this part of mass is represented by particles with the type of low mass stars, which can be regarded as a single stellar population. Each particle is assigned a mass of 1.6 $M_\odot$, which is the average mass of stars between 0.8 $M_\odot$ and 100 $M_\odot$.

For type II SN and type PN stars, the determination of mass is quite straightforward, since their mass distribution follows IMF directly. The idea is to first randomly generate a mass m in the corresponding mass range. Then check its probability to be kept according to its position in the IMF $\xi(m)$. High mass stars are more likely to be rejected since their probabilities are small. For the generation of type II stars, the algorithm can be described as follows:

(a) Calculate the maximum IMF value $\xi_0 = \xi(m_0)$ as the normalization factor. Here $m_0 =$ 8 $M_\odot$, which is the minimum mass for type II stars.

(b) Randomly generate a mass m in the mass range of type II SN stars: 8 $M_\odot$ – 100 $M_\odot$. Its probability to be kept is $p = \xi(m)/\xi_0$.

(c) Randomly generate a number r between 0 and 1.

(d) Compare r with p. If $r < p$, the probability criterion is fulfilled: m is adopted by the newly created star. Otherwise m is rejected: repeat (b) - (d), until $r < p$.

For type PN stars, the algorithm is the same, except the mass range is set to 0.8 $M_\odot$ – 8 $M_\odot$.

For type Ia SN stars, the conditions are more complex, since we have to determine the mass of both primary and secondary stars. We set the type Ia SN rates to be 20 % of type II SN rates in current model. Once a type Ia SN star is doomed to form, we follow the modeling of binary system in Greggio & Renzini (1983) and Matteucci & Greggio (1986), and assume the carbon-oxygen (C/O) white dwarfs are the most common type of type Ia SN progenitors. The upper boundary for the mass of primary star m_1 is set by the condition that the primary develops a C/O core before filling its Roche lobe (Greggio & Renzini 1983), which means the primary mass $m_1 \leq 8$ $M_\odot$ and the mass of the secondary star $m_2 < m_1$. This implies the maximum mass for the total mass of the system $m_B = m_1 + m_2 < 16$ $M_\odot$. Moreover, binaries with mass higher than $m_{B,\min}$ can produce type Ia events. Following Matteucci & Greggio (1986), we set $m_{B,\min} = 3$ $M_\odot$. The distribution function $f(\mu)$ for the mass fraction of the secondary star $\mu = m_2/m_B$ is assumed to take the form (Greggio & Renzini 1983):

$$f(\mu) = 2^{1+\gamma}\,(1+\gamma)\,\mu^{\gamma} \qquad (0 < \mu \leq \frac{1}{2}). \tag{5.1}$$

As adopted by Matteucci & Greggio (1986), $\gamma = 2$. Obviously $f(\mu)$ takes the maximum value when $\mu = \frac{1}{2}$.

Now we have all the necessary physical conditions to determine the mass of primary and secondary stars in the binary system. The algorithm is described as follows:

(a) Determine the mass of the secondary star m_2 in the mass range 0.8 $M_\odot$ - 8 $M_\odot$ with the previous algorithm. Its mass distribution is assumed to follow the IMF.

(b) The lower mass limit of the system $m_{B,\mathrm{inf}}$ is set to 2 m_2, since $m_1 > m_2$. If $m_{B,\mathrm{inf}} < m_{B,\mathrm{min}}$, set $m_{B,\mathrm{inf}} = m_{B,\mathrm{min}}$.

(c) The upper mass limit of the system $m_{B,\mathrm{sup}}$ is set to m_2 + 8 $M_\odot$, since the maximum mass for primary mass is 8 $M_\odot$.

(d) Determine m_B in the mass range $m_{B,\mathrm{inf}}$ - $m_{B,\mathrm{sup}}$ with the previous algorithm. Its mass distribution is assumed to follow the IMF.

(e) Calculate the mass fraction of the secondary: $\mu = m_2/m_B$. The probability for this μ is $p = f(\mu)/f(\frac{1}{2})$.

(f) Randomly generate a number r between 0 and 1.

(g) Compare r with p, if $r < p$, the probability criteria is fulfilled: m_2 and m_B are adopted by the newly created star, with the lifetime determined by the lifetime of the secondary star (Greggio & Renzini 1983). Otherwise m_2 and m_B are rejected: repeat (a) - (g), until $r < p$.

For each type of stars, we calculate their numbers in one star cluster according IMF, then create stars with the above algorithms. After the type II SN, type Ia SN, type PN stars are created, the rest part of the stellar mass is assigned to the low mass star particles. As a simple treatment, we randomly place the newly created stars inside the radius of the molecular cloud with the same velocity of the molecular cloud. When a massive star (type II SN star) is created in the cluster, we assume the host molecular cloud is destructed by photoionization. The corresponding mass transits to the hot/warm phase, with a fixed velocity dispersion of 5 km/s.

5.2.2 Stellar feedback

The stellar feedback processes have already been introduced in section 2.5.3, in which we focus on the time evolution of stellar feedback in one single stellar population. In the individual star formation model, we trace the evolution and feedback of every individual star. Figure 5.2 demonstrate the stellar lifetime as a function of stellar mass and metallicity. Obviously massive stars have a lifetime much shorter than low mass stars. Type II SN stars with a stellar mass higher than 8 $M_\odot$ have a lifetime of less than hundred Myr. Low mass stars with mass lower than 0.8 $M_\odot$ stay in the main sequence for more than 10 Gyr with low luminosity, which only take part in gravity interaction in the system. In the following part of this section, I introduce the feedbacks by different type of stars separately.

5.2.2.1 Stellar wind and type II SN stars

Massive stars (> 8 $M_\odot$) release energy and mass in the form of stellar wind during their whole evolution, and end their life as type II or type Ib, c supernovae (Portinari et al. 1998) [2]. They

[2] We only consider type II supernovae in current model

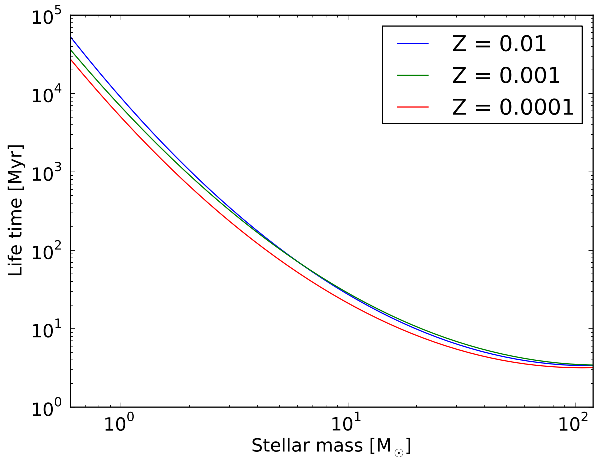
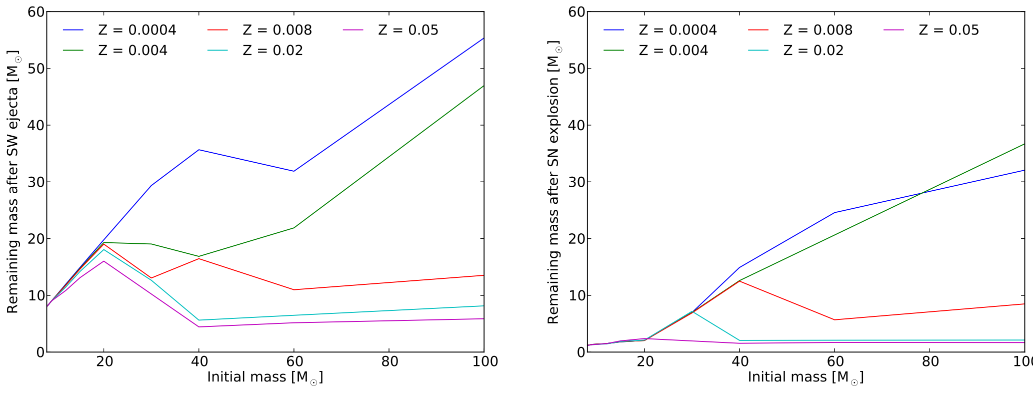


Figure 5.2: Stellar life time as a function of stellar mass and metallicity, based on the fit to the theoretical stellar tracks of Padova group (Raiteri et al. 1996).



(a) Remaining mass after ejecta of stellar wind. (b) Remaining mass after supernova explosion.

Figure 5.3: Remaining mass of massive stars after ejecta of stellar wind and after supernova explosion. Colors corresponds to different metallicity. Data is taken from Portinari et al. (1998).

Table 5.1: Nucleosynthesis products of type Ia supernova explosion. Yields are taken from W7 model of Iwamoto et al. (1999).

Species	^{12}C	^{14}N	^{16}O	^{20}Ne	^{24}Mg	^{28}Si	^{32}S	^{40}Ca	^{56}Fe
Yields (in $M_\odot$)	0.0483	1.16E-6	0.143	2.02E-3	8.5E-3	0.154	0.0846	0.0119	0.626

heat the surrounding molecular clouds by means of radiation and stellar wind. The Lyman continuum radiation is approximated as a function of stellar mass described in equation 2.21, with a heating efficiency of $\eta_{Ly} = 10^{-3}$ (Theis et al. 1992). The ejected mass from stellar wind as a function of stellar mass are taken from Portinari et al. (1998), in which the metallicity dependent mass loss rate is modeled as $\dot{M} \propto Z^{0.5}$ (Kudritzki et al. 1989). The remaining mass after ejecta of stellar wind as a function of initial stellar mass for different metallicities are plotted in the left panel of figure 5.3. According to the figure, high metallicity corresponds to a low remaining mass, since mass loss rate is proportional to the square root of metallicity. Especially, massive stars (> 40 $M_\odot$) with high metallicities are referred to as Wolf-Rayet stars. Their mass loss rates strongly depend on stellar mass ($\dot{M} \propto M^{2.5}$). They loss mass rapidly with strong stellar wind and end up with very low remaining mass (Portinari et al. 1998). The final wind velocity of as a function of stellar mass and metallicity is approximated as (Theis et al. 1992):

$$v_\infty = 3 \times 10^3 \left(\frac{m}{M_\odot}\right)^{0.15} Z^{\,0.08} \text{ km/s}. \tag{5.2}$$

Therefore the total heating from massive star is expressed as:

$$\dot{E}_{\rm SW} = \frac{1}{2}\dot{m}v_\infty^2 + \eta_{Ly}L_{Ly}. \tag{5.3}$$

The remaining mass after supernova explosion is given in the right panel of figure 5.3. A detailed explanation can be found in Portinari et al. (1998). Type II supernovae are the major source of oxygen, α-elements as well as iron. For each supernova event, the energy and chemical elements are distributed to all hot/warm particles within a radius of 0.1 kpc of the explosion site in the galactic scale model.

5.2.2.2 Type Ia SN stars

In a binary system which will end its life as type Ia supernova explosion, the primary star evolves to white dwarf first. It accretes mass from the secondary star which becomes a red giant at the late stage of its evolution. When the mass of the white dwarf reaches the Chandrasekhar limit (~ 1.41 $M_\odot$), supernova explosion is triggered. The nucleosynthesis products in this process is released to the surrounding hot/warm gas. For the yields of type Ia SN event, we adopt the W7 model of Iwamoto et al. (1999), as listed in table 5.1. For each type Ia SN event, we assume the white dwarf as the progenitor is destroyed. The corresponding mass comparable to the Chandrasekhar limit is ejected to the surrounding hot/warm medium. From the table, type Ia SN is the major source of iron.

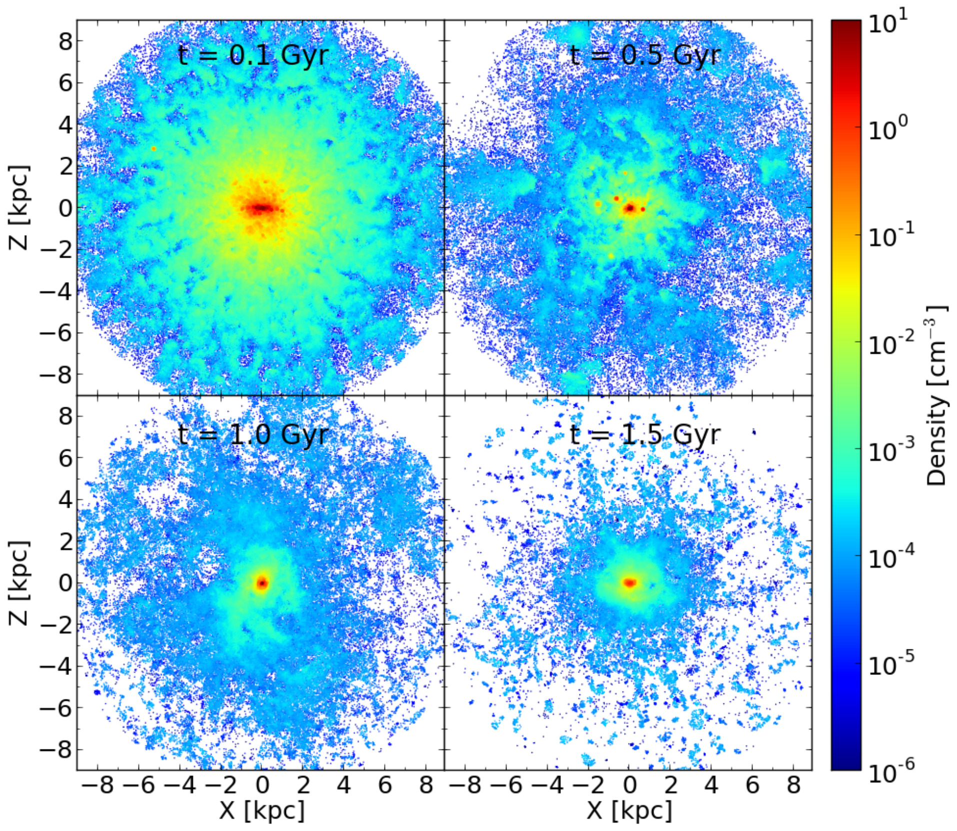


Figure 5.4: Edge on view of hot/warm gas density distribution at 0.1, 0.5, 1.0 and 1.5 Gyr.

5.2.2.3 PN stars

Intermediate mass in the mass range 0.8 $M_{\odot}$ to 8 $M_{\odot}$ have a lifetime of 10^{8} to 10^{10} years. They go through the asymptotic giant branch (AGB) phase at the end of their life. In this process they lose most of their envelope mass and contribute substantially to the interstellar abundances of He, C, N, and s-process elements. The envelopes finally evolve to planetary nebulae. We adopt the yields of intermediate mass stars by van den Hoek & Groenewegen (1997). The ejected mass is distributed in the surrounding molecular clouds of the PN star.

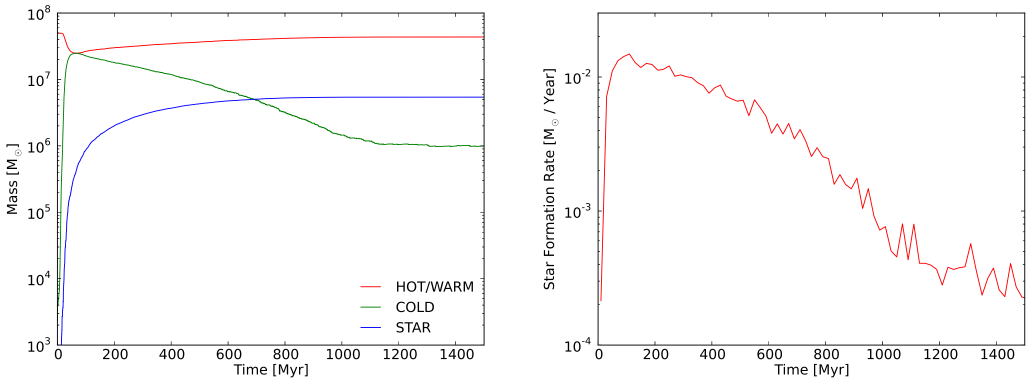
5.3 A galaxy scale model

In this section, I present the simulation results of a galaxy scale individual star formation model. As mentioned earlier, most of the physical processes are kept the same as multi-phase models in the previous chapters. The individual star creation and stellar feedback parts are newly implemented processes and have been introduced in previous sections. The size of molecular cloud is still determined by the *mass-radius* relation. According to observation, this relation is still valid for a cloud mass as low as 100 $M_\odot$ (Lombardi et al. 2010).

For the setup of initial condition, 5 million SPH particles with a total mass of 5×10^7 $M_\odot$ are distributed in a Plummer-Kuzmin sphere (Miyamoto & Nagai 1975), with parameter a=0.08 kpc and b=0.4 kpc. Burkert profile (Burkert 1995) is used to model the analytical dark matter potential, with parameter $\rho_0 = 0.045$ $M_\odot/\mathrm{pc}^3$ and $r_0 = 1.0$ kpc. The initial temperature of hot/warm gas is set to 10^3 K, which is the low limit for this component. The initial metallicity is set to 1% of the solar metallicity ($\sim 2\times10^{-4}$). Element (e.g. nitrogen and iron) abundances are scaled proportional to the solar value. Hot/warm transition described in previous chapter is adopted here to create the cold phase, with the linking length b set to 0.01 kpc. Maximum temperature and minimum density for transition are set to 10^4 K and $0.1/\mathrm{cm}^3$, which are comparable with the criterion of star formation in the single phase models. The model is evolved for 1.5 Gyr, with the individual star formation and feedback process described in the previous section. For each type II and type Ia SN event, we inject an energy of 2×10^{50} erg into the surrounding hot/warm phase as thermal energy, which is the standard SN energy with an efficiency of 20%. This efficiency is a free parameter in our model, we will play with it together with other parameters in the future work. The simulation is carried out on the "laohu" cluster in NAOC, CAS with 16 threads and lasts four days.

Figure 5.4 demonstrate the edge on view of hot/warm gas density distribution at four different snapshots. At 0.1 Gyr, most of the gas still stays in the center, where gas density is high. SN feedback already starts, which ejects a large amount of chemical elements and drives gas expanding. In the previous models, the SN feedback is distributed within the nearest 50 hot/warm particles with a maximum radius of 1.0 kpc. However, this is not appropriate for the individual star formation model, in which mass resolution is much higher, 50 nearest neighbour particles correspond to a radius much smaller than 1.0 kpc. To guarantee the feedbacks are distributed to more particles in a larger region, we fix the feedback radius to 0.1 kpc, which means all hot/warm particles within a radius of 0.1 kpc of feedback site will have the chance to receive the feedback. For the implementation, in each feedback timestep, type Ia SN and type II SN particles which are about to explode are broadcasted to all threads. Then a fixed radius neighbour search is carried out in each node, to find out how many hot/warm particles reside in the feedback region. Usually this number spans from 10^3 to 10^5, depends on the local density of feedback site. Actually the feedback scheme still has great chance to be improved, such as by adopting a time dependent radius of supernova bubble (Dobbs et al. 2011). At 0.5 Gyr, the gas becomes less dense. We can see some dense clumps of gas are leaving the center. They come from the photoionized molecular clouds when massive stars form. After 1.0 Gyr, the system does not evolve too much. The dense core becomes smaller. Later we will see the destruction of cold clouds is balanced by the hot/warm gas condensation and transition. The hot/warm gas core is sustained in this way.

Figure 5.5 and 5.6 demonstrate the evolution of mass, star formation rate and mass transition rate of the system. The transition from hot/warm gas to cold clouds is mainly in the first 100 Myr, which is demonstrated as a peak in the mass transition figure. Cloud component quickly accumulates mass in this way, and reaches the maximum at 50 Myr. After that cold clouds start to be photoionized by massive stars. Their mass returns to the hot component.



(a) Mass evolution. (b) Star formation rate.

Figure 5.5: Mass evolution and star formation rate as a function of time.

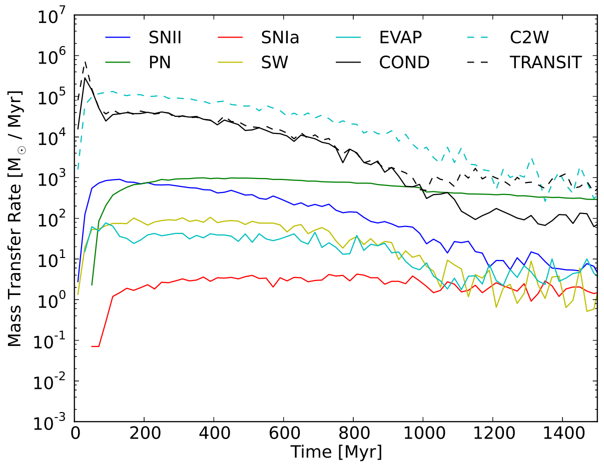
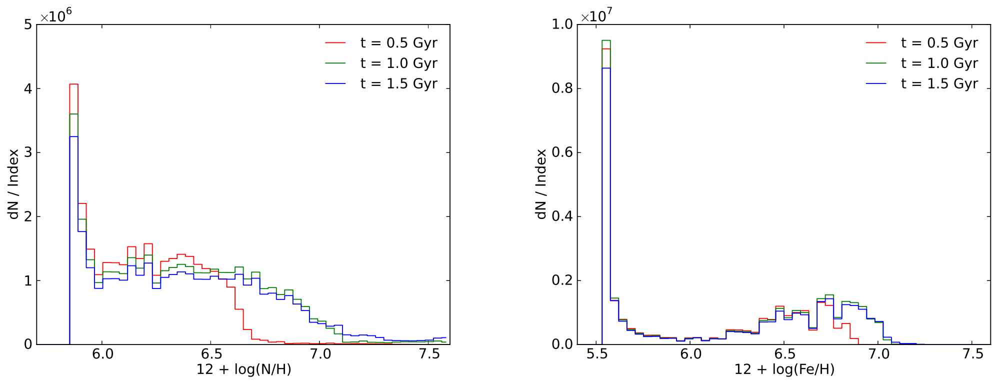


Figure 5.6: Evolution of mass transition rate. “TRANSIT” means the transition process from hot/warm to cold phase. “C2W” means the destruction of cold clouds to hot/warm gas by the photoionization of massive stars (> 8 M$_\odot$).



(a) Nitrogen.

(b) Iron.

Figure 5.7: The stellar nitrogen and iron abundance distribution at 0.5, 1.0 and 1.5 Gyr. Low mass type particles are excluded from the distribution.

An interesting phenomenon is that the condensation rate correlates with the transition rate in the first 1.0 Gyr. This can be explained as the transition process provides the seeds of cold clouds, such that hot/warm gas can condense to. Meanwhile the gas density is high and the temperature is low ($\sim$ 1000 K), which leads to an efficient condensation from hot/warm to cold phase. After 100 Myr, the system reaches a dynamic equilibrium. Evaporation of cold clouds ("C2W" in figure 5.6) are balanced by transition and condensation in the opposite direction. However, the destruction rate of cold clouds is still higher than the sum of transition and condensation rates. The final result is more hot/warm gas is produced. This is shown as the gradually rise of hot/warm mass in mass evolution panel. Note that the newly generated hot/warm particles do not stay in the system. They escape after receiving enough energy from supernovae feedback. The star formation rate correlates well with the cold cloud mass. This is not surprising, since we assume a constant star formation efficiency (10%) when creating star cluster inside a molecular cloud. Therefore SFR reaches its maximum at 50 Myr. Later we will see that a significant fraction of stars with initial metallicity is created around this time. After 1.0 Gyr, transition rate does not decrease any more as the first 1.0 Gyr, while the condensation rate continues decrease as the hot/warm gas becomes less dense. This time the sum of the condensation rate and the transition rate becomes equal to the destruction rate of cold clouds. The mass does not evolve too much for the three components. Meanwhile the system continues to produce stars, although the corresponding star formation rate is very low.

Figure 5.7 present the nitrogen and iron abundance distribution of stars at three different times. As mentioned before, for both elements, we assume an initial abundance of 1% solar value. The two distributions share one common feature: a peak at the position of the initial abundance. This peak corresponds to the stars form at the beginning of the simulation, when the cold component has not been metal enriched by stellar feedback and has not received metal from the hot/warm phase via transition and condensation processes. Meanwhile, star formation rate is of the highest at the beginning of the simulation, a large amount of stars with

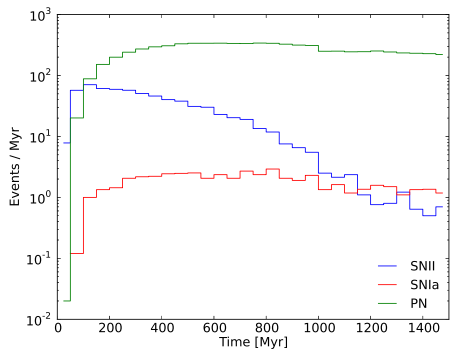


Figure 5.8: Feedback events as a function of time.

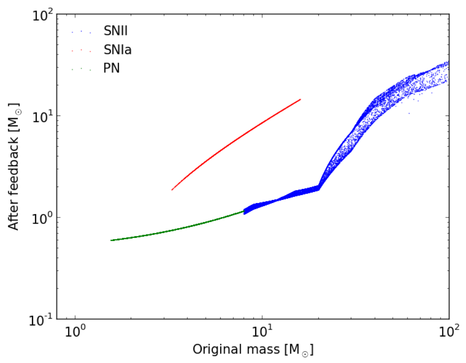


Figure 5.9: Comparison of stellar mass before and after feedback.

initial metallicity form. Besides that, the two distributions show strong evolutions between 0.5 and 1.0 Gyr, since more high metallicity stars form during that time. However the distributions do not evolve too much between 1.0 and 1.5 Gyr, since star formation rate is relatively low after 1.0 Gyr. The corresponding stellar feedback is also reduced. Besides these similarities, the two distributions show some difference. Nitrogen is mainly produced by the intermediate mass stars (type PN) during the AGB phase at the end of their life, and is also contributed a little bit by the massive stars via stellar wind feedback before they explode as type II supernovae. In the model we assume these two types of low energy feedback processes (PN and SW) mainly contributes to the cold phase. Since most of the intermediate mass stars have a lifetime of longer than 100 Myr, the enrichment of nitrogen is relatively slow, compared with that of iron. Therefore the nitrogen distribution moves to the high abundance end slowly and continuously. Iron is mainly produced by two types of supernova feedback, and is returned to the hot/warm phase. It is transferred to the cold phase via mass exchange processes such as phase transition and condensation. Since massive stars as the progenitor of type II SN have a life time much shorter than intermediate stars, the iron enrichment in the hot/warm phase is very fast. Due to the high transition and condensation rate at the beginning of the simulation, iron is transferred to the cold phase efficiently. The newly formed stars inside the iron enriched clouds have an abundance much higher than those formed initially. The iron abundance demonstrates a double peaks distribution.

Figure 5.8 demonstrate the frequency of feedback events as a function of time for three types of feedback. Due to the short lifetime of massive stars, the frequency of type II SN responses to the star formation rate with just a short delay ($<$ 100 Myr). The frequencies of type Ia SN and PN correlate well with each other, since their lifetimes follow the same distribution (the lifetime of type Ia star is determined by the secondary component, whose mass follows the same distribution as the type PN stars). Also their delay to star formation rate is much longer. The remnant mass as a function of initial mass for stars is shown in figure 5.9. For type Ia stars, as an approximation, we assume the primary star as the progenitor of supernova explosion is totally disrupted. The corresponding mass of Chandrasekhar limit is

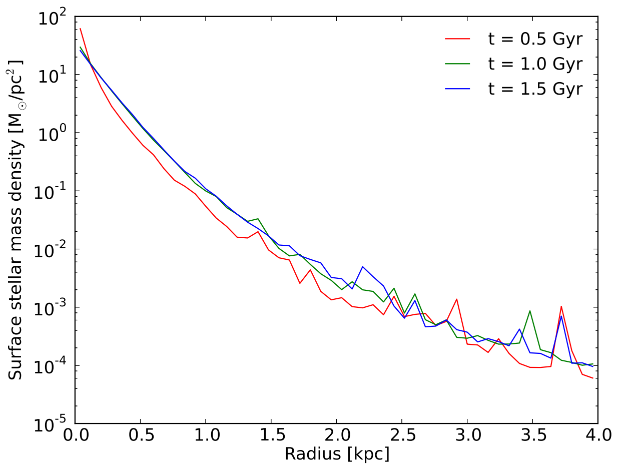


Figure 5.10: Distribution of surface stellar mass density at 0.5, 1.0 and 1.5 Gyr.

ejected to the surrounding hot/warm medium. The remnant mass does not decrease so much. For Type II SN stars, the remnant mass is metal dependent. As shown in figure 5.7, most of stars in the current model are in low metallicity ($< 0.1\ Z_\odot$), which do not go through the Wolf-Rayet phase. Therefore, a significant amount of mass is still kept in the remnant. What's more, due to the strongly metal dependent evolution of massive stars, the remnant masses of type II SN stars demonstrate a double peaks distribution, which corresponds to the double peaks of metal abundance distribution in figure 5.7. For the type PN and type Ia SN stars, their remnant mass are assumed to be independent of metallicity, but only as a function of initial mass.

Figure 5.10 demonstrate the evolution of stellar surface mass density. All the stars, including the remnants after stellar feedback and low mass type particles, are taken into account. According to the mass and SFR evolution in figure 5.5, more than half of the stellar mass is accumulated before 0.5 Gyr, when SFR is high. At 1.0 Gyr, the distribution becomes less steeper. The density in the center decreases a little bit. Stars shifts to the outer region, and rise the density there. From 1.0 to 1.5 Gyr, the distribution shows almost no evolution, which means the system reaches a dynamical equilibrium. The star formation activity is in a very low level.

5.4 A star cluster scale model

In this section I present the experimental work of modeling the individual star formation process in the turbulent velocity field. The basic idea is first setup a molecular cloud with SPH particles for one specific density distribution, then assign velocity to these particles according to a turbulent velocity field. When the simulation starts, gas collapses to the cloud center and transits to even smaller cold clumps. After a significant amount of gas transits to stars, a fraction of stars should form a bounded system, which can be regarded as a young star cluster

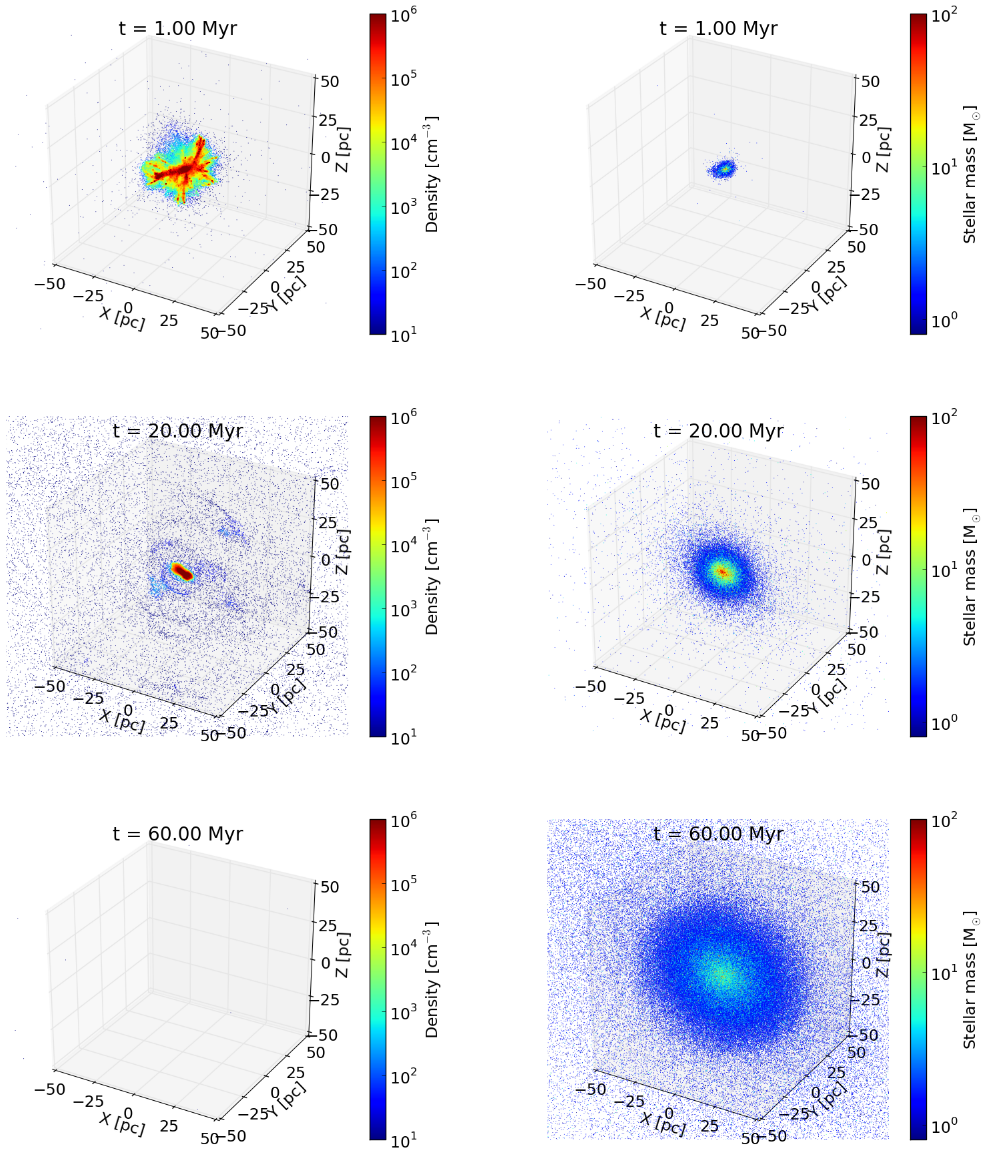


Figure 5.11: Simulation of star cluster formation. Left column: snapshots of gas component. Colors corresponds to different density, as specified by the color bar. Right column: snapshots of individual stars. Colors corresponds to different stellar mass. Upper, middle and bottom panels represent three different times: 1 Myr, 20 Myr and 60 Myr.

at its initial stage. The bounded and star-only system can be used as the initial condition for some high precision N-body integration program, such as “nbody6” (Aarseth 2003). Note that here the “cold clumps” are already different from the cold clouds whose sizes are determined by *mass-radius* relation. Since we are now focusing on the evolution of one single molecular cloud which is equivalent to one cold cloud in the galaxy scale model, the *mass-radius* relation is not valid any more: if we still use this relation to calculate the size of cold clumps, the densities of the SPH particles in the collapsing regions are already higher than that of cold clumps which they will collapse to. A more reasonable viewpoint is, these cold clumps are regarded as the core star forming regions where gas transits to stars. Therefore in this star cluster scale simulation, I have switched off the interactions (drag force, condensation and evaporation) between the cold and hot/warm phases, since they are already not applicable in this scale.

The above star cluster formation process is implemented with small modification to the galaxy scale model in the previous section. First, when stars are created inside the cold clumps analytically, the velocity dispersion is determined by the Larson’s relation (Larson 1981): $\sigma(\mathrm{km\ s^{-1}}) = 1.10\ L\ (\mathrm{pc})^{0.38}$, where L is the size of the cold clumps. As I have pointed out previously, the *mass-radius* relation does not hold in such a high density environment. However, due to some historical reasons, this relation is still used to estimate velocity dispersion in our model. The good thing is, the velocity dispersion only weakly depends on the mass of cold clumps (see Larson 1981, equation 2). A cold clump with a mass around 10^3 M$_\odot$ corresponds to a velocity dispersion of $\sim$ 1 km/s, which is adopted in the program. Second, to emphasize the effect of supernova feedback to the interstellar medium, I have modified the feedback part. Now the supernova energy is distributed in the nearest 256 gas particles as kinetic energy only, which drives the gas in the feedback region expanding. For the setup of initial condition, we place 5×10^5 gas (SPH) particles with a total mass of 5×10^6 M$_\odot$ in a gas sphere within a radius of 50 pc. The particles are distributed by following a density profile of $\rho \propto (r/r_0)^{-2}$. Here r_0 is a characteristic scale, which is set to 10 pc in the current model. The initial turbulent velocity field is created by imposing a Gaussian perturbation on a random velocity field in Fourier space, and transforming it back into real space. The power spectrum of the modes is described by a power-law function in wavenumber space with $E_k \propto k^{-2}$, which corresponds to Burgers turbulence (Girichidis et al. 2011). The amplitude of the velocity field is normalized according to the ratio of gas particles’ total kinetic energy to potential energy, which is specified in the models. Initially, all gas particles are assigned with a temperature of 1000 K and a metallicity of 0.1 Z$_\odot$. For the transition from gas to cold clumps, I use a linking length of 0.1 pc, a minimum density of $10^3/\mathrm{cm}^3$ and a maximum temperature of 1000 K. The probability for transition is set to 0.5% per 0.01 Myr. When the cold clump mass reaches 1000 M$_\odot$ by transition or by coagulation with other cold clumps, individual stars are created immediately without referring to another probability. The stellar mass is sampled according to Kroupa IMF (Kroupa et al. 1993). In total, four models with different parameters are set up, which are summarized below:

- Reference model: the initial kinetic energy to potential energy ratio ($E_\mathrm{K}/|E_\mathrm{P}|$) is set to 0.04, which corresponds to a velocity dispersion of 8.3 km/s. The supernova energy is set to a standard value of 10^{51} erg.

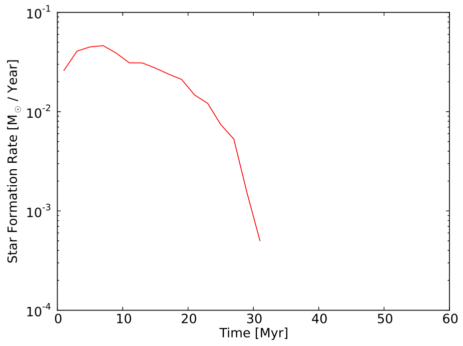


Figure 5.12: Star formation rate of the reference model.

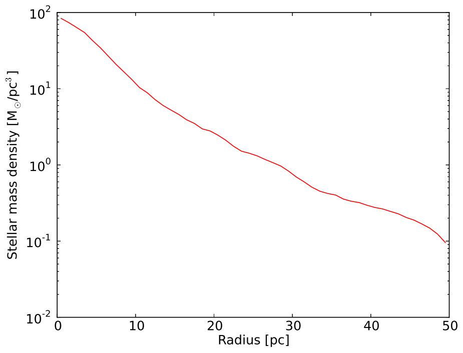
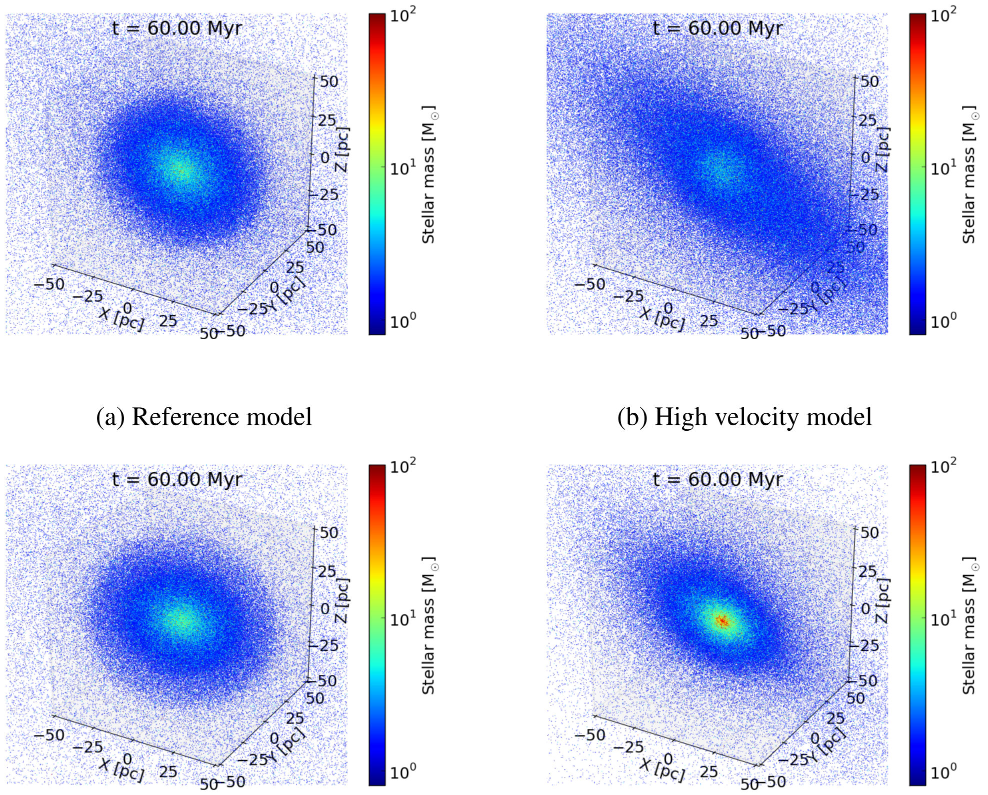


Figure 5.13: Stellar density profile of the reference model.

- High velocity model: the initial $E_K/|E_P|$ is set to 0.3, which corresponds to a velocity dispersion of 22.7 km/s. Other parameters are kept the same as the reference model.
- Zero velocity model: the initial velocity of particles are set to zero. Other parameters are kept the same as the reference model.
- Reduced supernova efficiency model: the initial $E_K/|E_P|$ is set to 0.3, which is the same as the high velocity model. The supernova feedback efficiency is reduced to 10% of the reference model (10^{50} erg). Other parameters are kept the same as the reference model.

The simulations are carried out on the "Milky Way" cluster of Jülich Supercomputing Center. Since the particle number is relatively small, for each run, four threads are used, with each thread utilizes one Tesla M2070 graphics card. All runs adopt a fixed timestep of $\mathrm{d}t = 0.001$ Myr and last for 60 Myr, which take about one day.

Figure 5.11 demonstrates the snapshots of gas and stellar components at three different times. At 1 Myr, two collapsing modes are observed. The dominant mode is gas collapsing to the center of mass, which forms an even denser central region. Another mode is collapsing to the filaments, which is triggered by the turbulent velocity field. Due to the hierarchical structure of molecular cloud, these two collapsing modes have been observed in the simulation of the core star forming regions in a much smaller scale (Girichidis et al. 2011). At 20 Myr, most of gas particles have been expelled from the cluster center by supernova feedback. The star cluster grows larger. At 60 Myr, all gas has left the system. The cluster becomes a bounded system as we have expected. The border of the cluster is quite clear. By calculating the star cluster's kinetic energy to potential energy ratio, we know that the system has reached virial equilibrium. For the stellar mas distribution shown in the figure, at first glance massive stars mainly reside in the center of the cluster, which demonstrates a clear feature of mass segregation. However, I have to point out that this is not the case. It just depends on the way we generate snapshots of the stellar system: star particles are sorted according to their mass; massive stars are always plot at last, thus overlap the less massive ones and seem more concentrate in the center. From the physical side, the primordial mass segregation can be



(a) Reference model

(b) High velocity model

(c) Zero velocity model

(d) Reduced SN efficiency model

Figure 5.14: Stellar component of four models at 60 Myr.

excluded: star particles are created by sampling the IMF; the initial fraction of massive stars is always the same in every place. Nor can it be dynamical mass segregation: according to our calculation, the system's relaxation time scale is as long as 6 Gyr; the star cluster is far from being relaxed when the simulation stops at 60 Myr. The mass segregation can only be observed by evolving current system for several Gyr with the high precision gravity integrator "nbody6", which is what we plan to do as the next step work.

Figure 5.12 is the star formation rate of the reference model, which shows that the star formation rate almost stops at 30 Myr. Actually this is the time when most of the gas has left the system. Also the stellar component becomes bounded around this time. Figure 5.13 is the density profile of stars in the reference model at 60 Myr. The stellar density in the center is as high as 100 $M_\odot/pc^3$, which is consistent with the cluster formation model of Nakasato et al. (2000). The corresponding velocity dispersion of the cluster at this time is around 7 km/s.

Figure 5.14 demonstrates the snapshots of the stellar component for four models with different parameters at 60 Myr. For the reference model and the zero velocity model, the bounded

Table 5.2: Properties of four models after 60 Myr evolution. The parameters are calculated based on the stellar particles within 50 pc of the cluster center. Those massive stars which have exploded as type II supernovae are excluded from statistics.

Model name	Total mass ($M_\odot$)	Particle number	Half mass radius (pc)	Velocity dispersion (km/s)	Final $E_K/\|E_P\|$	Relaxation time[1] (Gyr)	Bounded
Reference	4.58×10^5	285402	19.4	6.98	0.552	6.36	Yes
High velocity	2.07×10^5	129552	25.4	4.15	0.549	6.90	Yes
Zero velocity	4.74×10^5	294917	16.5	7.54	0.545	5.05	Yes
Reduced feedback	1.22×10^6	749398	5.1	37.05	1.716	-	No

[1] The relaxation time is calculated as the *medium relaxation time*, which uses the mean density inside the system's half mass radius. See page 514 of Binney & Tremaine (1987) for a detailed explanation. The relaxation time of the reduced feedback model is not calculated, since there is still gas resides in the center and the stellar system itself is not bounded.

clusters are quite prominent. The cluster in the high velocity model are more scattered. However, it is also bounded according to our calculation. The cluster in the reduced SN feedback efficiency model seems very compact. However, this is due to a significant amount of gas still resides in the system. The stellar component itself are not bounded. Table 5.2 presents the basic parameters of the four clusters after 60 Myr evolution. The three bounded clusters have a kinetic energy to potential energy ratio of around 0.55, and a relaxation time of around 6 Gyr, which is consistent with the observations for galactic globular clusters (Webbink 1985; Binney & Tremaine 1987)[3]. What's more, it demonstrates a clear trend that the lower initial velocity yields a more compact cluster, and more mass is bounded, the corresponding relaxation time is shorter.

Our work is somewhat similar with Nakasato et al. (2000) one decade ago. They present the formation of globular cluster in the inner region of the proto-globular clouds (PGCs) with their SPH code for various initial conditions. In their simulation, a gas sphere is setup with the density profile of $\rho \propto r^{-2}$ within a radius of 150 - 300 pc. The total mass is assumed to be 10^6 $M_\odot$ with an initial temperature of 10^4 K. They implement the physical processes required for globular cluster formation in a SPH code, such as radiative cooling, star formation, energy feedback by stellar winds and supernovae, chemical enrichment. The treatment of star formation is based on the classical "Katz" recipe (Katz 1992) which is quite common in single phase models. By investigating various models with different initial radii and metallicities, they conclude that the initial metallicity must be high enough to produce a globular cluster-like system, and a shell-like structure of gas forms in all cases. Although in both of our work, the bounded structures are formed, our models are still different from theirs, as summarized below:

- Our model is multi-phase. Gas first collapses to cold clumps. Afterwards individual

[3]The results are based on data from Webbink (1985), by assuming an average stellar mass $m = 0.7$ $M_\odot$, which is different from $m = 1.6$ $M_\odot$ in our system. However, this will not change our conclusion, since the deviations by adopting these two values are within the same order.

stars are created inside cold clumps. In the “Katz” recipe adopted in Nakasato et al. (2000), stellar particles are created as a fraction of SPH particles’ mass, and are regarded as a single stellar population due to the limited resolution. What’s more, the feedback scheme are quite different. We distribute the supernova energy as kinetic to the surrounding 256 particles. Nakasato et al. (2000) adopt a totally thermal way.

- In Nakasato et al. (2000), a shell-like gaseous structure is formed, which is due to the energy inputs of stars in the cluster center. The bounded cluster consists of stars form before the shell formation, which happens at the beginning of the simulation. In our simulation, the shell like structure does not form. The star cluster becomes bounded after most of gas has left the system. In addition, the time scale of the cluster formation in our model is much longer than theirs (50 Myr vs. 10 Myr).

5.5 Summary

In the above sections, I introduce the implementation of the individual star formation model, and present its application on two objects at different scales.

In the galaxy scale model, I model a low mass dwarf galaxy with 5 million SPH particles, each of them has a mass of 10 $M_\odot$. The cold molecular clouds are created via hot/warm gas transition and accumulate mass via condensation. Individual stars are created inside these clouds with mass distribution given by IMF. We can show that with a star formation efficiency of 10%, three millions star particles form with a total mass of 5×10^6 $M_\odot$ after the system evolves for 1.5 Gyr. The model mimics the individual star formation process according to IMF, and reproduces the life cycle of interstellar medium. More importantly, the individual star formation scheme provides a possible solution for the future galaxy scale simulation when the newly formed star particles are not massive enough to be regarded as a single stellar population when the resolution further increases. Note that our current model does not aim at reproducing observations, which requires a large amount of work to investigate the parameter space.

In the star cluster scale model, I present the process of star cluster formation inside the turbulent velocity field. In this model, the *mass-radius* relation is not applicable to the cold clumps due to the high density. Instead, these cold clumps should be regarded as the core star forming regions where individual stars are created. After all the gas has left the system, the bounded star clusters form and reach virial equilibrium. Our main discovery can be summarized below:

- Our star cluster formation model is consistent with the model of Nakasato et al. (2000) in the aspect of central stellar density and cluster size. However, due to the different implementation of physical processes such as star formation and stellar feedback, the gas behavior is quite different.

- The density profile is crucial in determining the bounding state of the star cluster. Before switching to the current $\rho \propto r^{-2}$ profile, a uniform density distribution has been tested. In that distribution the bounded system cannot form even if the initial velocity is reduced to a very low level.

- The feedback efficiency is important in expelling the gas. In the reduced feedback efficiency model, although the initial velocity is high, the low feedback energy cannot clear the gas inside the system. The star cluster itself is not bounded, which is not what we expect.

- The initial velocity dispersion is important in determining the compactness of the final star cluster. Low initial velocity yields a more compact star cluster. The corresponding stellar velocity dispersion is higher and the star cluster is more massive.

What's more, I have to point out that we still have some work to do before putting the star cluster into "nbody6" for further integration. One problem is binary stars. The accuracy of our current code is not high enough to trace the evolution of binaries. Therefore only single stars are taken into account in the current simulation (type Ia SN is implemented as a fraction of type II SN, but only as single stars with a special type). Of course, this can be solved by assuming a binary fraction and creating binaries when preparing IC for "nbody6". Another problem is the treatment of low mass stars. In the current multi-phase model, stars in the mass range of 0.08 $M_\odot \sim 0.8$ $M_\odot$ are approximated as low mass type particles, each with a mass of 1.6 $M_\odot$. It is still under discussion whether this type of particles should be resolved as individual stars when the star cluster is put into "nbody6". In that case the particle number will be one order higher, which is a big burden for both multi-phase program and the "nbody6".

"The article (algorithm) is originally natural, the wonderful hand gets it accidentally."
Wen Zhang, Lu You

6 Numerical method

When the computer was first invented in the 1940s, no one could predict that it will revolutionize scientific research. In the field of astrophysics, where most of the phenomenons cannot be reproduced in the laboratory, numerical simulation might be the only way to validate theory when observations are far away from reaching the desired object in the universe, temporally or spatially, such as the neutral hydrogen distribution before reionization (Morales & Wyithe 2010), the very inner part of molecular clouds where massive stars are born (Peters et al. 2010). Also, simulation makes it possible to watch the whole process of evolution, instead of merely snapshots by observation, such as the merging process of two galaxies (Springel et al. 2005a), the formation of large scale structure (Springel et al. 2005b). As various numerical methods are invented to utilize the power of the computers (Press et al. 2007) in the recent decades, some of them come from the field of astrophysics and are mainly used in astrophysical simulations, such as the Barnes-Hut (BH) tree method (Barnes & Hut 1986) for the acceleration of approximate gravity calculation, the smoothed particle hydrodynamics (SPH) method (Lucy 1977; Gingold & Monaghan 1977) for a particle based description of the fluid field.

To achieve the desired resolution in a simulation, a huge amount of computational resources are required. Because of the limited memory size and CPU number on one computer, multiple computers are usually connected together by high speed network to consist of a computer cluster. Since memories and CPUs are distributed on different nodes (computers), how to write programs running efficiently on these nodes becomes a special topic in computer science: high performance computing. Some well written parallel codes are publicly available for astrophysical simulation, such as Gadget (Springel 2005), FLASH (Fryxell et al. 2000) and GAMER (Schive et al. 2010). What's more, besides CPU, various hardwares are investigated for the potential of accelerating astrophysical simulations. The famous ones includes the GRAPE series by Jun Markino (University of Tokyo), the FPGA based MPRACE board developed by Guillermo Marcus of Heidelberg University. Due to the highly parallel architecture and high memory bandwidth, nowadays GPU quickly rises and has become the mainstream of hardware accelerator. Its main features will be introduced in section 6.2.1 of this chapter.

As a PhD student whose topic is numerical simulation, my work is somewhat similar to a programmer: writing code and making test. Some "good" simulations are picked out and presented as scientific results. In this chapter, I will introduce the numerical issues involved in the model, including the parallelization scheme and the GPU acceleration of the most time consuming parts in the code (gravity calculation, SPH and neighbour search).

6.1 Numerical treatment

For numerical simulation, speed is what we care most besides the reliability of the physical processes, especially when the size of the simulation is large: the famous “Millennium Simulation” kept the supercomputer of Max-Planck Society’s Supercomputing Center in Garching busy for more than one month, although the program had been specially optimized for this task. The simulations presented in this thesis are mainly carried out on the “laohu” cluster of National Astronomical Observatories of China (NAOC), on which the maximum running time for one job is one week, and up to 64 threads are supported (actually 16 - 32 threads are preferred, the 64 threads queue is usually busy). I have to mention that this is already one of the largest GPU clusters aimed at astrophysical simulations. What’s more, several reasons further limit the maximum durable time for our simulation. First, our multi-phase code is continuously under development, which is not as stable as some well written codes which are able to run for months without stopping. Second, the parameter space for the multi-phase model is large, we have to investigate the combination of several parameters, which is not a trivial work.

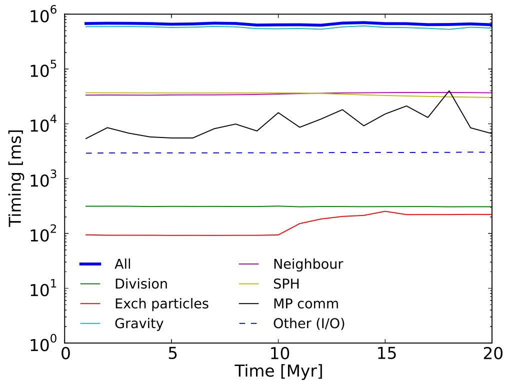
To find out the bottle neck of our program, we make detailed benchmark for the main processes of a typical run, the result is shown in figure 6.1. According to the pie chart, gravity calculation consumes most of the time, followed by SPH and neighbour search parts. As a parallel program which runs on multiple nodes, the parallel part takes only 1.8% of the total time. I have to point out that this part is influenced by many factors: load imbalance, particle distribution, physical processes, and so on. Since the program runs on more than one node, other programs of the same node would affect the communication efficiency, which possibly leads to the peak (black curve) in the left panel at 18 Myr. When the program is accelerated by GPU, especially in the late stage of the simulation, half of the time might be spent on communication due to the synchronization between threads caused by the load imbalance. Obviously, the bottle neck of the program is the gravity part, then SPH and neighbour search (based on current domain decomposition and integration scheme, the load imbalance and communication time cannot be reduced easily). In the next few subsections, I will introduce our numerical treatment and GPU acceleration scheme for these three parts.

6.1.1 Gravity

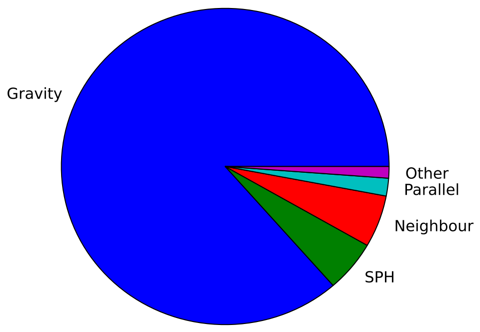
6.1.1.1 Introduction to gravity calculation

When doing astrophysical simulation, gravity is an unavoidable topic. The main reason is it plays the dominant role in almost every scale: from the formation of large scale structure to the evolution of planetary system. The gravity is so strong that in many studies, one can already get reasonable results by just taking it into account. In a particle system which contains N particles, the gravity acceleration and potential for particle i are calculated as:

$$\mathbf{a}_i = -\sum_j \frac{G m_j \mathbf{r}_{ij}}{\left(r_{ij}^2 + \epsilon^2\right)^{3/2}} \tag{6.1}$$



(a) Timing as a function of time.



(b) Fraction of total time for the most time consuming tasks.

Figure 6.1: Benchmark for a typical CPU run of the first 20 Myr: 2 million particles, 8 threads, neighbour lists are updated every 10 timesteps. According to the pie chart, the gravity, SPH, neighbour search and parallel parts take 86.4%, 5.2%, 5.3% and 1.8% of the total time, respectively. Note that the parallel fraction includes time consumptions of domain decomposition and multi-phase related particle exchange.

and

$$\phi_i = -\sum_j \frac{Gm_j}{\left(r_{ij}^2 + \epsilon^2\right)^{1/2}}. \tag{6.2}$$

Here $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$, ϵ is the gravity softening length to prevent the unphysically large acceleration when two particles are too close to each other (Aarseth 1963). Since gravity cannot be shielded like electromagnetic forces in plasmas, the summations is made with all the other particles in the system. This is the so called direct summation method, which is often used for high precision integration, such as the famous “nbody” series (Aarseth 2003) for the modeling of star clusters. However, the computation time of direction summation method is $O(N^2)$. The time consumption will become unbearably large when the particle number increases. At present only up to 10^5 particles can be modeled with this method (Aarseth 1999, 2003; Makino & Hut 1988; Spurzem 1999; Harfst et al. 2007).

In galaxy and cosmological scale, much larger particle number is required such that the cosmological potential and density field can be traced with enough resolution. The accuracy of the gravity is already not the first priority. By keeping this in mind, a set of approximation methods are developed, such as the BH tree method (Barnes & Hut 1986; Springel 2005), in which the simulation boxes are divided into a series of child boxes hierarchically during tree construction. When calculating gravity, particles in one box are treated as one **pseudoparticle** located at the center of mass of that box (Bertschinger 1998), if the size of the box l and the distance to the target particle r satisfies $l/r < \theta$. Here θ is the opening angle used to control the error of force. One can choose $\theta < 1$ to achieve enough accuracy. By setting $\theta = 0$, the tree method degenerates to the direct summation method. By sacrificing some accuracy (remote particles are approximated by a set of pseudoparticles), the BH tree method achieves a computation time of $O(N \log N)$, which is much faster when the particle number is large. In our model, we choose a BH tree method which is specially optimized for the architecture of GPU, which will be described later.

Another kind of method which is widely used for the fast calculation of gravity is the particle-mesh (PM) method, which is widely used in both particle based and grid based numerical codes (Fellhauer 2001; Schive et al. 2010; Federrath et al. 2010). The basic idea is to first derive the gravitational potential field on a set of equally spaced grids of the simulation box. Then gravity force is calculated by interpolating the potential field among the nearby grids. The potential field on the grids can be obtained by solving the Poisson's equation:

$$\nabla^2 \Phi(\mathbf{r}) = 4\pi \, G \, \rho(\mathbf{r}). \tag{6.3}$$

In Fourier space this partial differential equation takes a polynomial form:

$$\hat{\Phi}(\mathbf{k}) = -4\pi \, G \, \frac{\hat{\rho}(\mathbf{k})}{k^2}, \tag{6.4}$$

which can be solved easily. By using the Fast Fourier Transform (FFT) algorithm (Cooley & Turkey 1965; Press et al. 2007), the transform yields a computation time of $N_g \log N_g$ (N_g is the number of grids along one dimension), which makes the fast solving of Poisson equation possible Bertschinger (1998). To compute the density field in Fourier space $\hat{\rho}(\mathbf{k})$, the mass of the particles must be assigned to the nearby grids first. Various assignment methods exist, such as the frequently used nearest grid point (NGP) scheme, the cloud in cell (CIC) scheme and the triangular shaped cloud (TSC) scheme. By introducing the assignment function $W(\mathbf{r})$ to describe the assignment methods, the density field we get on the grid is the actual field convolved with $W(\mathbf{r})$, which inevitably introduces some aliasing errors (Jing 2005). For a more detailed description of these methods, see Hockney & Eastwood (1988). Some implementations of the FFT algorithm are publicly available, such as the FFTW library [1] (Frigo & Johnson 1998). It supports distribute memory FFT by using the MPI interface, which is very suitable for the gravity calculation in parallel codes. One big disadvantage of PM method is, the force resolution is limited by the size of the mesh. The rise of resolution is a memory and CPU expensive operation. To overcome this difficulty, some hybrid methods are developed, by decomposing the gravity force into two parts, the PM method is used only for the long range force, while the direct summation or tree method is used for short range force (interaction with nearby particles). A typical example is Gadget-2, which uses the combination of Tree and PM methods (TreePM) to balance speed and accuracy of gravity calculation (Springel 2005).

From the hardware side, some devices are developed specifically for the speedup of gravity calculation. One successful example is the "GRAPE (GrAvity piPE)" series (Sugimoto et al. 1990; Makino & Taiji 1998), which is an Application Specific Circuit (ASIC) computer engine designed to accelerate the N-body simulations by calculating gravity using multiple pipelines for many particles simultaneously. With GRAPE-DR, this series have reached the

[1]`http://fftw.org/`

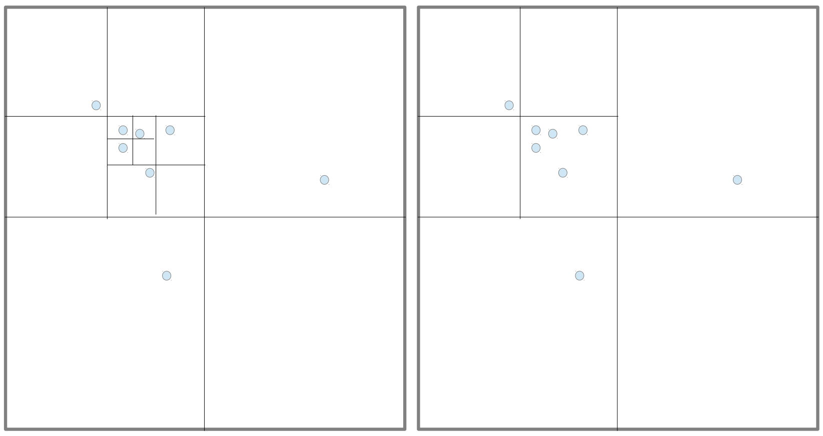


Figure 6.2: Construction of the BH tree. Left: classical tree method, the simulation box is divided recursively until there is at most one particle in each box. Right: modified tree method, box will not be further divided if the number of particles in this box is less than a critical number, in this case it is 5.

peak performance of 85 Tflop/s, and a sustained speed of 22 Tflop/s. It has already achieved a speed comparable with the fastest general-purpose computers at that time, and won several Gordon Bell Prizes for the main designer Junichiro Makino and his collaborators. However, the GRAPE boards can only be used for classical gravity calculation, which strongly limited its use in other fields. Also, due to the relatively small market, it has not been fully commercialized, which leads to the high price of the product. Only a few research institute could afford its cost and deploy in their machines. Meanwhile, other hardwares, such as FPGAs are also explored as a possible hardware accelerator (Spurzem et al. 2009).

6.1.1.2 An optimized tree code for parallel architecture

We choose Fukushige's tree code (Fukushige et al. 2005; Makino et al. 2003) for the serial version, and Hamada's tree code for the parallel version (Hamada & Iitaka 2007) of our model. Both of them use the modified tree method (Barnes 1990), which is first implemented by Makino on a RISC-based host computer and GRAPE-1A with a peak performance of 240 Mflops (Makino 1991). They are just different implementation of Barnes's original idea, with no big difference in the algorithm level. Before introducing the algorithm, two terms must be introduced first: as shown in equation 6.1, the target particle for which the gravity is calculated is called i particle, the particles whose gravity contribution is taken into account for the gravity calculation of i particle are called j particles. In the tree method, j particles are not all necessarily real particles, they might be pseudoparticles as described in previous section.

In general, the parallel tree construction and gravity calculation algorithm can be summa-

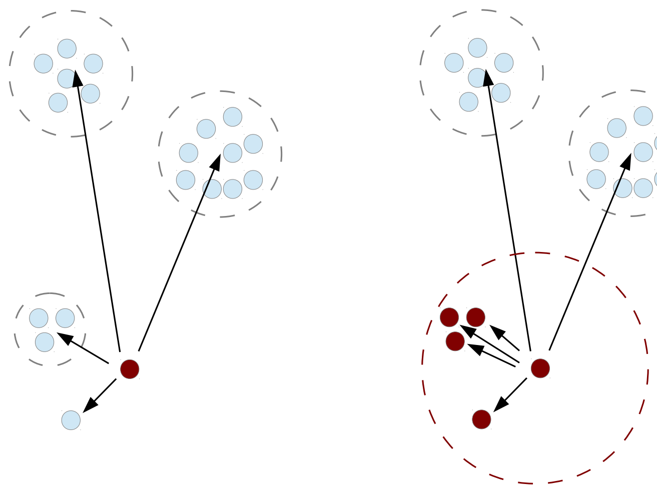


Figure 6.3: Gravity calculation based on the BH tree. Left: classical tree method, the interaction list is only prepared for one target particle (the red one). Right: modified tree method, the interaction list is prepared for a group of particles (particles in the red circle). Courtesy of Peter Berczik.

rized in the following steps:

(a) Construct the tree for particles in local domain by inserting particles into the tree one by one.

(b) Calculate the center of mass for all nodes of the local tree.

(c) Treat the local domain as one node, using the opening criterion to determine if a tree node in other domains should be imported to the local domain. Exchange external nodes between every two domains. The domain decomposition algorithm guarantees that the partition is always in the form of cuboid, which can be treated as one box when determining whether an external nodes should be imported.

(d) If there are external nodes imported, insert these nodes into the the local tree, using the same method in step (a). Update the center of mass for all node based on the updated tree.

(e) Group particles into small groups (i particle list) and prepare the interaction list (j particle list) for this group, then send both lists to device, calculate gravity and retrieve gravity information.

Since the introduction to BH tree method can be found in many references related with gravity calculation, I will only focus on Makino's modification to the classical tree for parallel architecture. The basic idea of this modified tree method is, in step (e), when calculating gravity, first transverse the tree for a group of target particles (*i* particles), then prepare the interaction list (*j* particles) for this group, which will be used by every particles in this group. In traditional tree method, interaction list is prepared for every individual target particle. Figure 6.3 demonstrate this difference: all particles in the red circle share the same interaction list in the modified tree method. What's more, the tree structure itself provides an subdivision of the space, which can be used to group particles. The algorithm to construct groups by using the tree is demonstrated in appendix A.1.2. In the algorithm, `ncrit_for_group` is a parameter which controls how many particles should be grouped together. Usually it is a multiple of the pipeline number, say, 8192, such that all particles in the group will be loaded to the pipelines and processed via several loops. Another difference from the traditional tree method is, when the tree is constructed, the simulation box is subdivided into 8 child boxes recursively until the particle number in leaf box (box with no child) is less than a critical value, say, 16. While in the traditional method, this critical value is 1, which means each leaf box contains at most 1 particle, or empty (see figure 6.2 for a demonstration). The algorithm is shown in appendix A.1.1, in which a parameter `ncrit_for_tree` is used to control the maximum number of particles in one leaf box. The reason for this modification is, the number of particles in one group is quite large, further subdivision of the tree would only increase the time consumption of tree construction without speedup the calculation. This improved BH tree method is extremely suitable for hardwares with multiple pipelines: each pipeline is assigned with one target particle in one group, then the force is calculated simultaneously for all target particles using the same interaction list. The preparation of interaction list is similar with classical method, which will not be explained here.

Based on Makino's original implementation for GRAPE architecture, Hamada & Iitaka (2007) made further optimization, such that the algorithm is suitable for GPU architecture. One difference between GRAPE and GPU is, GRAPE has a broadcasting mechanism, with which the information of *j* particle is broadcasted to all pipelines. In this way the time spent for data moving is hidden, which yields a high efficiency for pipeline calculation. For GPU, which is designed for more complicated operation and data transfer, no broadcasting is available. Instead, it introduces the shared memory: small size but provides fast access to computing unit. The "Chamomile Scheme" is developed to utilize this new feature of GPU. The basic idea is, first each thread (corresponding to the pipeline in GRAPE card, the smallest computing unit) loads one *i* particle. The interaction list (*j* particles) are grouped into J-BAGs with the size of threads and are loaded into shared memories each time for one bag. Then each thread calculates interaction for the corresponding *i* particle with every *j* particle in the shared memory. After all *j* particles are grouped into J-BAGs and are loaded by the threads, gravity calculations for *i* particles are done. The use of shared memory greatly reduces the data transfer time for *j* particles, since the access to shared memory is two orders faster than global memory. Moreover, by packing the sequential data of *j* particles into J-BAGs, the data transfer from global memory to shared memory is "coalesced", which further reduces the data transfer time for *j* particles.

The "Chamomile Scheme" can be summarized in the following steps (assuming there are `NTHREADS` threads in each block. The size of `NTHREADS` is determined such that the data size of

`NTHREADS` j particles (each particle takes four 4 Bytes floating point units, `NTHREADS` particles take $4 \times 4\times$`NTHREADS` Bytes) can be loaded into the shared memory):

(a) The host (CPU) prepares a set of i particle lists and the corresponding j particle lists, and sends them to the computing device (GPU).

(b) Each block is assigned one i particle list and the corresponding j particle list.

(c) Within one block, each thread loads one i particle from the i particle list, in total `NTHREADS` i particles are loaded. Then each thread resets the gravity force registers to zero.

  (d) Each thread loads one j particle from the corresponding j particle list into shared memory, and synchronize inside the block.

  (e) Each thread calculates gravity between i particle and all j particles in shared memory in one loop. The results is accumulated in gravity force register. Synchronize inside the block **again** to guarantee that the shared memory will not be overwritten by another thread which has started its next loop.

  (f) Repeat (d) - (f) until all j particles in j particle list are loaded for calculation.

(g) The gravity calculation for i particle is done. The result is transferred from register to the array of force in global memory.

(h) Repeat (c) - (g) until the gravities of all i particles in i particle list are calculated.

(i) The array of force is sent back to host. Calculations finish.

6.1.2 Smoothed particle hydrodynamics (SPH)

We adopt the standard SPH description for the hot/warm gas in the model, in which gas is modeled by a set of smoothed particles (Lucy 1977; Gingold & Monaghan 1977) with their individual mass m_i and smoothing length h_i. All physical properties of the hydrodynamical field are sampled at the positions of particles and are distributed with a kernel function $W(r; h)$. The basic SPH equations used in the model can be found in many classical publications on this method (Lucy 1977; Monaghan 1992; Navarro & White 1993; Price & Monaghan 2007). In this section we will give a brief introduction to the method and describe the acceleration scheme on GPU.

6.1.2.1 Introduction to SPH method

The density at i particle's position $\mathbf{r}_i$ with respect to all neighbours j is approximated as:

$$\rho_i = \sum_j m_j \cdot W(|\mathbf{r}_i - \mathbf{r}_j|; h_{ij}), \tag{6.5}$$

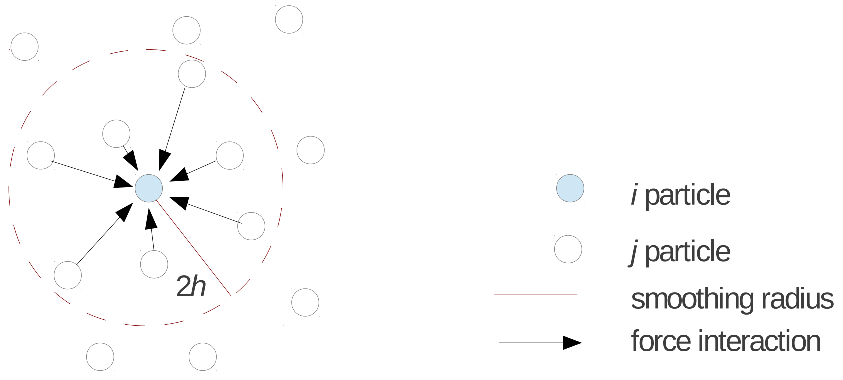


Figure 6.4: Demonstration for SPH smoothing length and force interaction. Courtesy of Guillermo Marcus.

here $h_{ij} = (h_i + h_j)/2$. The sum is made for all the neighbours (j) of particle i. $W(r;h)$ is the SPH kernel, which determines how the mass of the SPH particle is distributed in the nearby region. According to Monaghan & Lattanzio (1985), $W(r;h)$ takes the form:

$$W(r;h) = \frac{1}{\pi h^3}\begin{cases} 1 - \frac{3}{2}\left(\frac{r}{h}\right)^2 + \frac{3}{4}\left(\frac{r}{h}\right)^3, & \text{for } 0 \le \frac{r}{h} \le 1, \\ \frac{1}{4}\left(2 - \frac{r}{h}\right)^3, & \text{for } 1 \le \frac{r}{h} \le 2, \\ 0, & \text{for } \frac{r}{h} > 2. \end{cases} \tag{6.6}$$

The smoothing length h is defined as the **half** distance from the target particle i to the most distant neighbour, as is shown in figure 6.4. In SPH calculation, the neighbour number is either kept constant (our model) or varied in a small range (Gadget), such that the smoothing length can represent the local density of the fluid field. The chosen of the neighbour number is a technical problem: it should be neither too small nor too large. A too small value will amplify the fluctuation of the local density field, which leads to the smoothing length statistically unmeaningful. While a too large value is appropriate for statistics, but will smooth out the variation of the density field in small scale. In our model, the neighbour number is set to 50. The equation of motion is derived from the Euler equation:

$$\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = -\frac{\nabla P}{\rho}, \tag{6.7}$$

and takes the form:

$$\frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = -\sum_j m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2}\right) \nabla_i W(|\mathbf{r}_i - \mathbf{r}_j|, h_{ij}), \tag{6.8}$$

P is the pressure determined by the equation of state. The above form is often used in astrophysical simulations. A detailed derivation and other forms of equation of motion can be found in Dalrymple's lecture [2].

To simulate the dissipate energy within shock wave and to prevent two fluid flows from passing through each other when collide, the artificial viscosity is introduced into the equation of motion, which takes the form (Monaghan 1992; Balsara 1995; Steinmetz 1996; Springel et al. 2001b):

$$\Pi_{ij} = \begin{cases} \frac{-\alpha c_{ij}\mu_{ij}+\beta\mu_{ij}^2}{\rho_{ij}} & \mathbf{v}_{ij}\cdot\mathbf{r}_{ij} < 0, \\ 0 & \text{else.} \end{cases} \tag{6.9}$$

Here

$$\mu_{ij} = \frac{h_{ij}(\mathbf{v}_i - \mathbf{v}_j)\cdot(\mathbf{r}_i - \mathbf{r}_j)}{\mathbf{r}_{ij}^2 + \epsilon h_{ij}^2} f_{ij}. \tag{6.10}$$

$c_{ij} = (c_i + c_j)/2$ is the mean sound speed and $\rho_{ij} = (\rho_i + \rho_j)/2$ a mean density. ϵ is a small factor which prevents the denominator becoming too small and is set to 0.01 here. $f_{ij} = (f_i + f_j)/2$ is the so-called Balsara factor:

$$f_i = \frac{|(\nabla\cdot\mathbf{v})_i|}{|(\nabla\cdot\mathbf{v})_i| + |(\nabla\times\mathbf{v})_i| + \epsilon^2 c_i/h_i}. \tag{6.11}$$

Now the equation of motion 6.8 becomes:

$$\frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = -\sum_j m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} + \Pi_{ij}\right)\nabla_i W_{ij}, \tag{6.12}$$

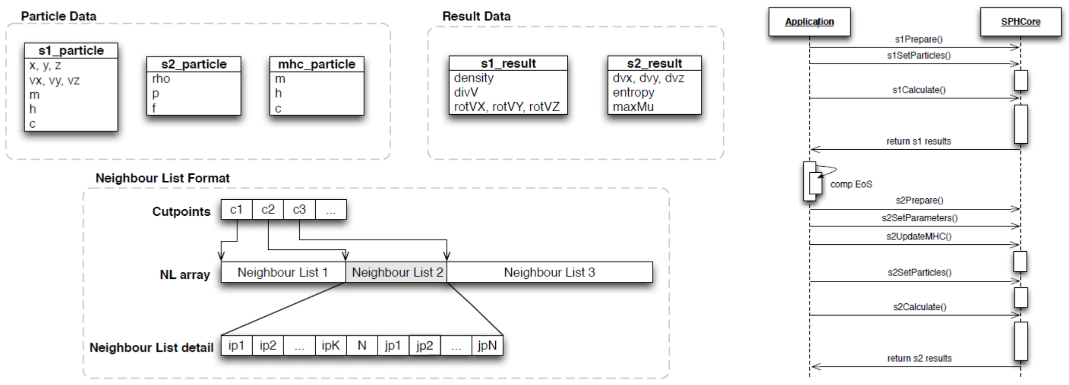
and the corresponding equation of internal energy takes the form:

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = \frac{1}{2}\sum_j m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} + \Pi_{ij}\right)(\mathbf{v}_i - \mathbf{v}_j)\nabla_i W_{ij}. \tag{6.13}$$

6.1.2.2 The SPH method on GPU

The GPU acceleration scheme used in current model is based on Guillermo Marcus's raceSPH library which is designed for FPGA architecture originally. The library is further extended to support other hardware architectures, such as CPU, SSE extension and GPU (using CUDA interface) with the same interface. The reason to develop this library can be somewhat explained by figure 6.1: when every process is running on CPU, definitely the gravity calculation is the most time consuming part: it takes 86.4% of the total time. However, if the gravity is accelerated by GPU or other hardwares and achieves an amazingly two orders of speedup, as we will see in section 6.2.2.1, it is already not the most time consuming part. SPH calculation

[2]`http://www.ce.jhu.edu/dalrymple/classes/785/Interpolation.pdf`



(a) Data structures (b) Data flows

Figure 6.5: Data structures and data flows of raceSPH library. Courtesy of Guillermo Marcus.

together with neighbour search become the new bottle neck. Therefore it is very necessary to accelerate these two parts by using some kind of special hardwares, just as what has been done for gravity calculation. I have to point out that the benchmark result shown here strongly depends on the particle distribution and the configuration of hardwares. But in general gravity calculation is the most time consuming part, then the SPH and neighbour list construction. In this section I will focus on the GPU acceleration for the SPH part, while the neighbour search part will be introduced in the next section. In our current code, we do not use the raceSPH library directly. The main reason is, when moving to a new platform, especially when the GPU architecture is different, we have to recompile the library. However, the compilation is not a easy task due to the dependence of tools and libraries on that platform. The portability of our code will be reduced if we still use the library. What's more, when the SPH equations or algorithms are changed, which is very likely to happen during the development of the model, we have to modify the corresponding part of the code in the library. However, the raceSPH library is designed to support multiple hardwares, which involves various architectures. The modification of the code is not a trivial work, especially when we want to keep consistency among multiple hardware implementations (CPU, SSE, GPU and FPGA). To avoid the above problem, I extract the GPU part only and simplify the architecture, while the interfaces and algorithms are kept identical to the raceSPH library. The final codes are organized into 3 CUDA files and can be easily compiled as part of the multi-phase code. Below I will give a brief introduction to the interfaces and algorithms used in the raceSPH library. A more detailed description can by found in Guillermo Marcus's PhD thesis (Marcus 2011).

For SPH calculation, the basic idea is still assigning the target particles to the multiple pipelines, and calculating the interactions for multiple target particles simultaneously. However, due to the relatively complex physics, the implementations of SPH routines on GPU are more difficult, and we could predict that the efficiency for this part will not be as high as for gravity calculation. Consider the physical quantities involved in SPH calculation, the input parameters include position (x, y, z), velocity (vx, vy, vz), mass (m), smoothing length (h), pressure (P), sound speed (c) and the neighbour list for each target particle. The outputs include density (ρ), rotation, divergence, acceleration ($\mathrm{d}vx, \mathrm{d}vy, \mathrm{d}vz$) and change of internal energy (described as entropy in raceSPH library). The intermediate results (density, divergence and rotation) are calculated according to position, smoothing length and mass, and are needed for the calculation of other outputs.

Figure 6.5 demonstrate the data structures and the corresponding data flows of raceSPH library. The SPH calculations are divided into two steps. In the first step, particle information are grouped into structure `s1_particle` and then sent to the device. After calculation, the results are transferred back to the host via the structure `r1_result`. In the second step, the pressure and Balsara factor are calculated in the host and are grouped into structure `s2_particle` together with density. Actually they can also be calculated by the device, which is not difficult for implementation. However, this will reduce the flexibility of the library, since the pressure and Balsara factor depend on the equation of state, and take different forms according to different SPH implementation. Besides `s1_particle` and `s2_particle` as input, there is an additional data structure `mhc_particle`, which is used to update smoothing length and sound speed, which might change after first step calculation. After the second step calculation, the acceleration and the change of internal energy are transferred back to the host via the structure `s2_result`, the whole calculations finish. The format of neighbour list is shown in the lower panel of figure 6.5a. The list is organized as a 1-D integer array, such that it can be transferred to the device via one translation. In principle, the neighbour list (j particles) can be shared by several particles (i particles). However, usually each neighbour list is prepared only for one particle.

6.1.3 Neighbour search

As shown in the above section, SPH calculation, together with neighbour search become the bottleneck of the program when gravity calculation is accelerated by GPU. The neighbour search is important in our model: when different types of particles exchange mass and energy via various physical processes, e.g. stellar feedback, condensation and evaporation, the first step is to locate their neighbours within a fixed radius or the nearest N_B particles. In this section, I will introduce some commonly used method for neighbour search, especially the k-dimensional tree (k-d tree) method used in our code, and the GPU acceleration for this part.

6.1.3.1 The linklist method

At first glance, the neighbour search is a simple problem. For every target particle i, first calculate its distance to another $N - 1$ particles in the system, then sort the distance, pickup the nearest N_B neighbours or cut at a certain radius. However, this yields a computation time of $O(N^2)$, if we prepare the neighbour lists for all particles in the system. In this sense, the neighbour search is somewhat similar with the direct summation method used for gravity calculation. Actually in the direct summation code (Aarseth 2003), one can obtain the distance information between particles without extra calculation. Neighbour search is not a big problem in such kind of codes. However, when gravity is calculated in an approximate way, e.g. the

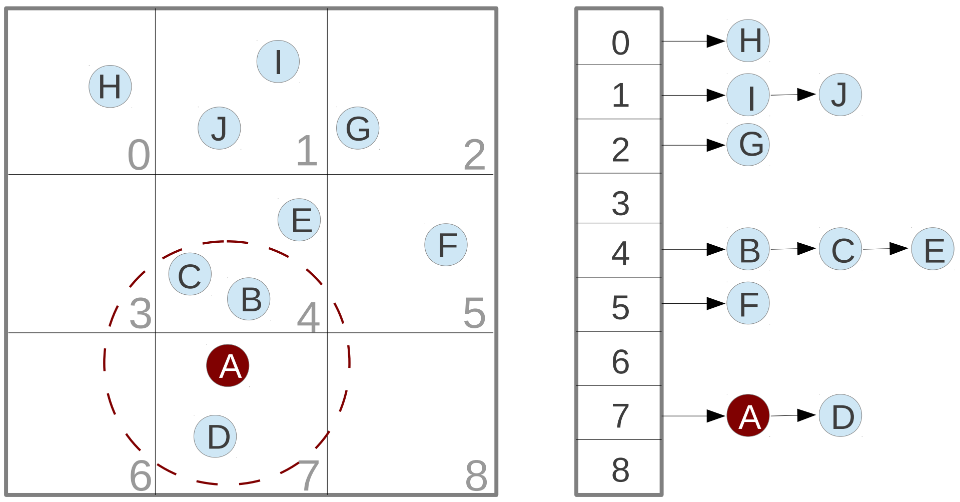


Figure 6.6: Demonstration of the linklist method for neighbour search.

BH tree method, distance information is not available for free, we have to find other faster algorithm.

One popular method is the linklist method. The basic idea is first dividing the space equally into small search cells. Then inserting particles into the cells according to their positions using a linklist (Hockney & Eastwood 1988). The linklist is a kind of data structure which consists of two parts, one part stores the data, another part is a pointer to the next element in the linklist. For neighbour search these two parts correspond to the index of current particle and the index of the next particle. For a target particle, when the searching radius has been fixed, one only needs to open the cell it resides in and the nearby cells which have an overlap with the searching range. The neighbour search process can be summarized in the following steps:

(a) Linklist construction: divide the space into equal sized cells.

(b) For every particle in the system, locate the cell it resides in according to its position and insert it to the cell's linklist. The linklists are constructed in this way.

(c) For every target particle, locate the cells overlapped with its search range.

(d) For every overlapped cell, retrieve all particles in the corresponding linklist.

(e) For every retrieved particle, calculate its distance to the target particle, determine if it falls in the search range.

(f) Repeat (c) - (f) for all target particles.

Figure 6.6 illustrates the corresponding process in 2-D case. First the searching area is equally divided into 9 (3×3) cells. Then particle A - J are inserted into the corresponding linklist. For the target particle A, first locate the search range (encircled by the red dashed line), which is overlapped with cell 3, 4, 5, 6, 7, 8. Then these six cells are opened and the corresponding linklists are traversed to find all particles (D, C, B, E, F) in these cells. These particles' distances to A are calculated. Particle B, C, D are identified as A's neighbour. The linklist method still yields a computation time of $O(N^2)$. However, for one target particle, only the nearby region is needed to be searched, which is much faster compared with searching the whole region. This method is more suitable for a homogeneous particle distribution. Therefore it is widely used in cosmological studies for the calculation of correlation function, in which the number of neighbours within a given spherical shell is counted.

6.1.3.2 The *k*-d tree method

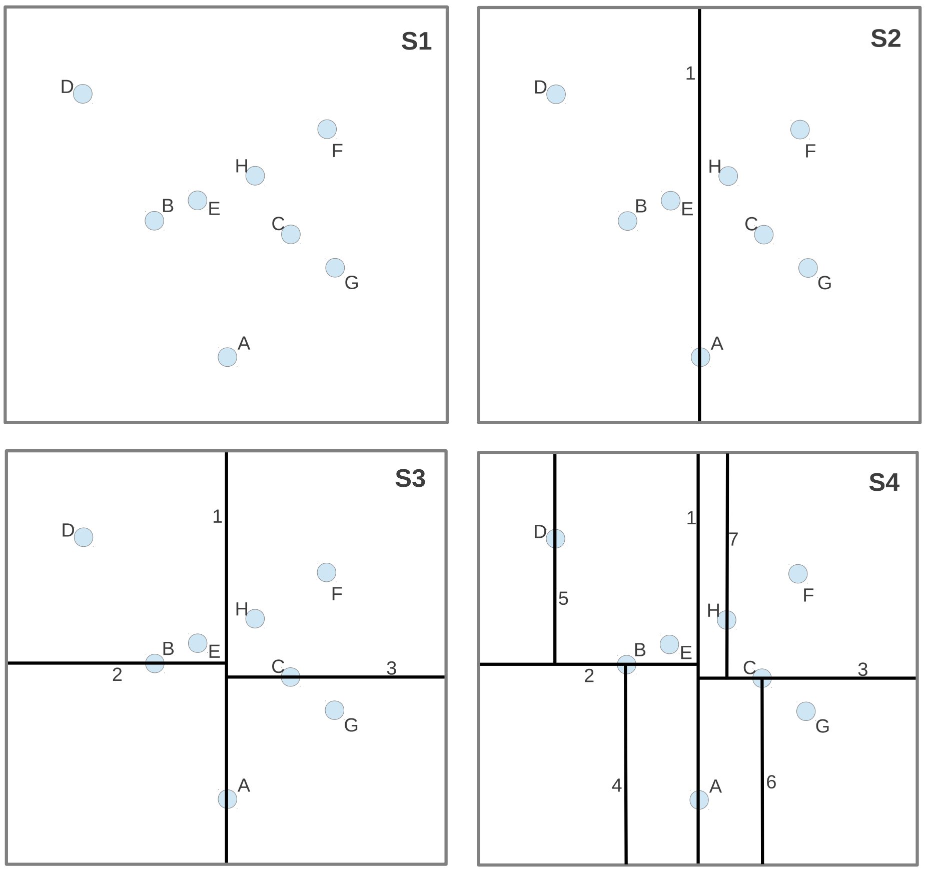
The efficiency of the linklist method decreases sharply for an inhomogeneous particle distribution, such as in the simulation of galaxies: the density of particles decreases exponentially from the center to the edge. In this case, some hierarchical trees are more suitable for the organization of particles, such as the BH tree for gravity calculation. Although most hierarchical trees can be used for neighbour searching, the *k*-d tree (Bentley 1975; Press et al. 2007) is still the most frequently used method for this task. The construction of the *k*-d tree is somewhat similar with that of the BH tree: start with the root node, particles are inserted into the tree one by one, which yields a computation time of $O(N \log N)$. N is the total particle number. The time for finding one point's neighbour is $O(\log N)$, which corresponds to a total computation time of $O(N \log N)$ for all particles. Compared with the direct calculation and the linklist method, the *k*-d tree method is much faster when N is large, which makes it very suitable for our model: large particle number ($> 10^4$), highly inhomogeneous particle distribution. According to Press et al. (2007), the principles of *k*-d tree are summarized as follows:

- Boxes are successively partitioned into two daughter boxes.
- Each partition is along the axis of one coordinate.
- The coordinates are used cyclically in successive partitions.
- In making the partition, the position of the "cut" is chosen to leave equal numbers of points on the two sides (or differing by one in the case of an odd number of points).

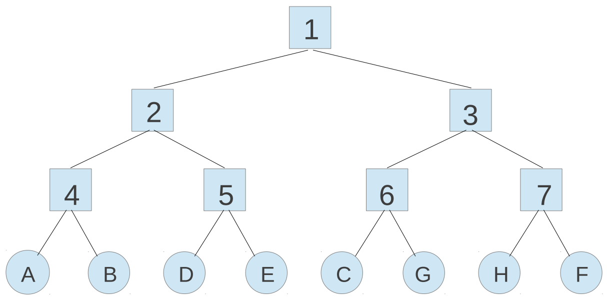
To find out the nearest neighbour of target particle i in the tree, the following two steps are carried out:

(a) Locate the box in which i lies. Find the nearest neighbour of i in that box.

(b) Traverse the tree in a deep first order. When encounter a new box, check whether it might contain a particle closer to i than the previously identified nearest neighbour. A box will be opened only if it is closer to the i particle than the current nearest neighbour.



(a) Construction process of the *k*-d tree.



(b) The corresponding tree structure after construction.

Figure 6.7: Top: demonstration of *k*-d tree construction in 2-D case. S1: all particles reside in the root box before division. S2: sort particles according to the *x*-coordinate. Divide the root box into two daughter boxes with line 1. S3: sort particles on the right and left side according to the *y*-coordinate. Divide each of them into two daughter boxes with line 2 and 3. S4: sort particles according to the *x*-coordinates again. Divide each of them into two daughter boxes with line 4, 5, 6, 7. Bottom: rectangles and circles represent boxes and particles respectively. Box 4, 5, 6, 7 are leaf boxes, which only contain particles and will not be further divided.

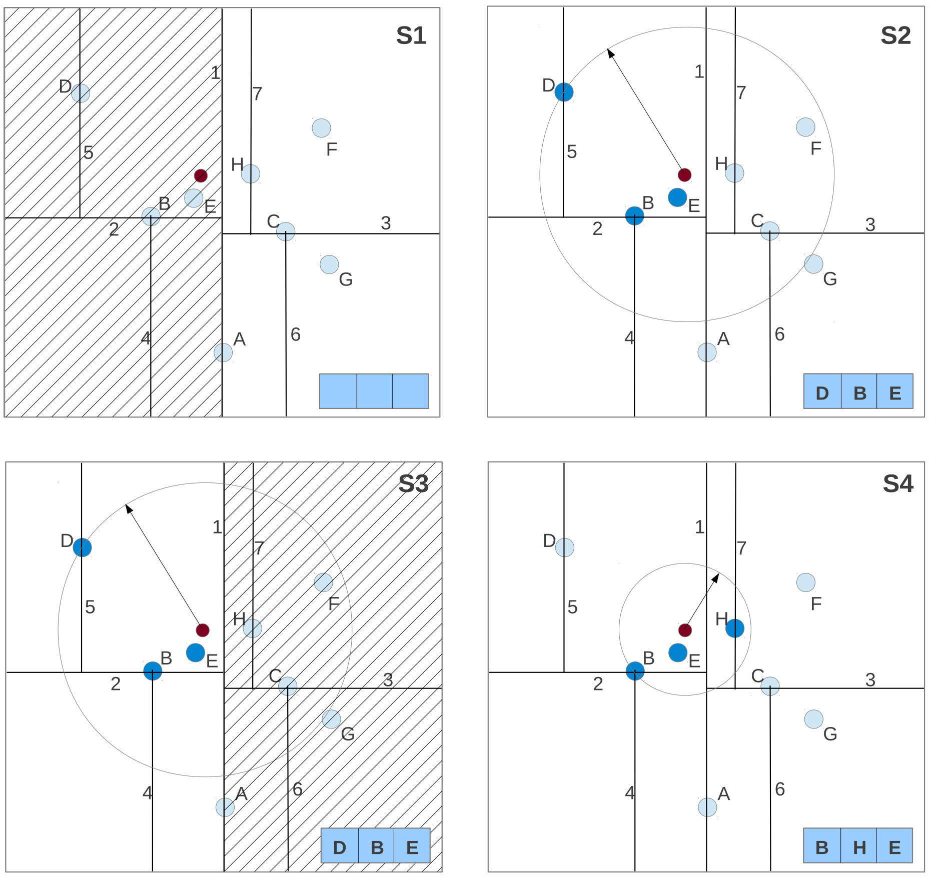


Figure 6.8: Demonstration of neighbour search with k-d tree. The target particle is labeled as red. The nearest 3 neighbours will be located ($N_B = 3$). S1: the target particle resides in box 2 (shadowed box), which contains 4 particles ($> N_B$). S2: find the nearest 3 particles in box 2: D, B, E. Insert them into the nearest neighbour list and record the distance between the target particle and the most distant neighbour (particle D) $d_{\max}$. S3: traverse the tree again. Box 3 might contain particles with distance smaller than $d_{\max}$. Open this box and check its daughter nodes. S4: particle H is closer to target. Particle D is kicked out from the nearest neighbour list. Neighbour search is done.

Note that in our model, what we want to obtain is not just the nearest neighbour, but the nearest N_B neighbours. In this case the above algorithm needs to be modified a little bit. The trick is to maintain a neighbour list with the size of N_B and always record i's distance to the most distant neighbour as $d_{\max}$. When doing neighbour search, in the first step, locate the box that i particles resides in and contains enough particles to fill in the neighbour list. In the second step, traverse the tree, a box will be opened only if its distance to particle i is smaller than $d_{\max}$. In the leaf box (box contains no daughter box), if there is a particle with a distance smaller than $d_{\max}$, insert this particle into the neighbour list and update $d_{\max}$. When the traverse finishes, the nearest N_B neighbours are stored in the neighbour list. A detailed description of k-d tree construction and neighbour search can be found in appendix A.2.1 and A.2.2. Meanwhile, figure 6.7 and 6.8 demonstrate the corresponding process.

For the task of neighbour search, in the serial version of the model that developed earlier, we use the "Approximate Nearest Neighbors" library[3] (Arya et al. 1998) which is based on the k-d tree algorithm and demonstrates a good performance on CPU. In the parallel version, we implement the algorithms described in Press et al. (2007) and make modification to the data structure such that the codes are capable of running on GPU. The idea is to organize the data for tree structures into a set of arrays with simple data types (`int`, `float`, `int4` and texttttfloat4), such that each value (vector) is stored in an independent array and is transferred to GPU separately, which allows the compiler to give a better optimization for the data access in global memory. Actually this method has been proposed by Marcus (2011) for SPH calculation on GPU: he demonstrates that the structure of arrays (SOA, organizing data into independent arrays as described above) yields a computation speed of 10% faster on GPU compared with the array of structures (AOS) which is frequently used for CPU codes. The GPU codes for neighbour search are quite similar with the CPU version, except that the target points are transferred to GPU global memory as an array such that each processing unit on GPU will be assigned one target particle and the search is carried for multiple targets simultaneously. In general, the GPU neighbour search can be summarized in the following steps:

(a) Construct the k-d tree using the algorithm described in appendix A.2.2 on the host by CPU. Transfer all the necessary arrays of tree structures to GPU's global memory.

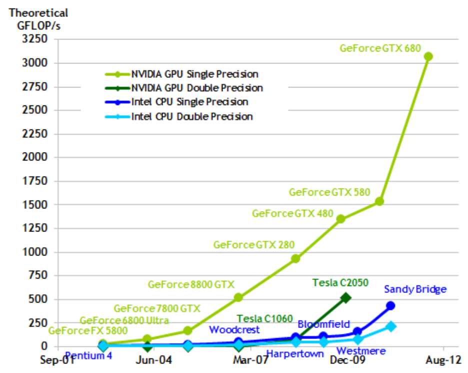
(b) Transfer the target particle array which contains the positions of target particles to GPU. Allocate space on GPU global memory to store the neighbour list. Set the number of neighbours needed to be found.

(c) Assign each target particle to a processing unit of GPU. Do neighbour search for multiple targets simultaneously.

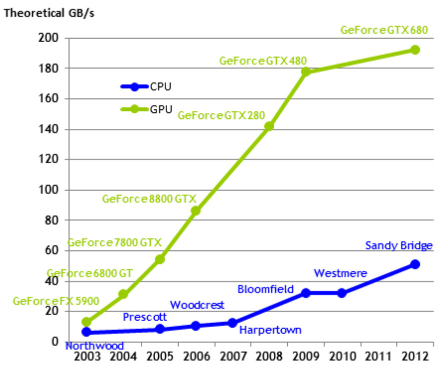
(d) Transfer the neighbour list back to hosts. Delete all space allocated on GPU global memory and host memory. Neighbour search is done.

We can expect that the speedup for neighbour search on GPU will not be as high as that for gravity and SPH calculation, since this process contains too many control flow instructions

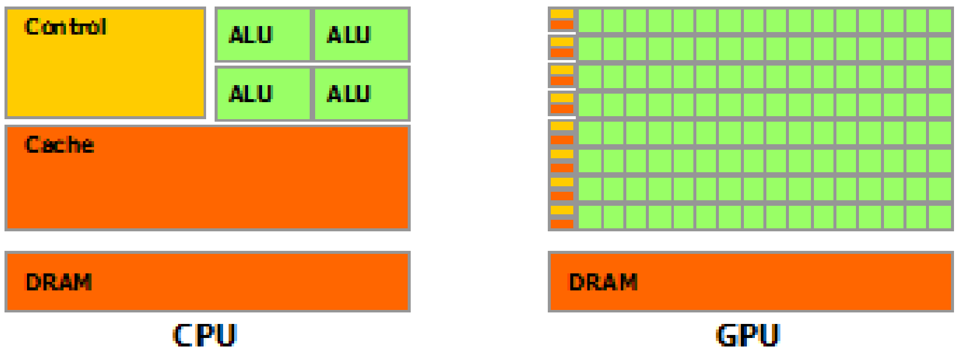
[3]`http://www.cs.umd.edu/ mount/ANN`



(a) Floating point operation per second (FLOPS).



(b) Memory bandwidth.



(c) Architecture.

Figure 6.9: Comparison of performance and architecture between CPU and GPU. Figures are taken from "CUDA C PROGRAMMING GUIDE", October 2012, NVIDIA Corporation.

(`if`, `do`, `for`, `while`), which are actually not suitable for GPUs. The benchmark result for neighbour search part will be presented in the benchmark section.

6.2 GPU acceleration

6.2.1 Introduction to GPGPU

As mentioned earlier in this chapter, many hardwares other than CPU have been investigated for the potential of accelerating numerical calculation. Among them the GPU (Graphics processing unit) quickly emerges and has become the most successful one. Below I summarize the main features of GPU:

- Multi processing units. Different from CPU, a GPU is built on a set of multithreaded *Streaming Processors* (SMs). When the host evokes a GPU kernel, the grids and blocks specified by the program are mapped and assigned to these SMs. The SM is able to execute multiple threads concurrently due to its SIMD[4] nature. For NVIDIA GPU, one SM consists of multiple CUDA cores depends on compute capability (from 8 cores in 1.x to 196 cores in 3.x, but SM always executes in groups of 32 threads called *warps*). Within one decade, the GPU core number has risen from 128 cores in the first generation programmable GPUs (GeForce 8800GTX) to 2496 cores in the latest Tesla K20 of Kepler architecture. Moreover, this fast increase of core number guarantees the corresponding increase of the computing performance, as shown in figure 6.9a.

- High memory bandwidth. To fulfill the requirement of data transfer between multiple CUDA cores and GPU memory, modern GPU cards support a high memory bandwidth, which increases from 86.4 GB/s of GeForce 8800GTX to 250GB/s of the latest Tesla K20.

- More transistors for data processing. As demonstrated in figure 6.9c, for CPU, which is designed for general purpose application, the complex memory cache (L1, L2, L3) takes up a large area of the CPU chip. For GPU, which is originally designed for computing intensive and highly parallel tasks of graphics processing, more transistors are devoted to data process on a GPU chip.

- Low price and fast update. Big companies (NVIDIA, AMD, Intel) provide huge support for the development of GPU card and guarantee the good product quality. Meanwhile the mature technique and large scale production of the card reduce the price, which makes it much more competitive compared with other hardware accelerators.

The idea of GPGPU (General purpose GPU) is to speedup traditional CPU applications with GPU. In the early days, the main method for GPGPU is to cheat GPU by using Cg (C for graphics) or GLSL (OpenGL Shading Language) which are originally designed for programming shaders. The release of CUDA (Compute Unified Device Architecture) together with the first generation of programmable graphics cards (the G80 series) fully extract the power of GPU and make GPGPU popular. Now the GPGPU technique can be found in every field related with high performance computing. Meanwhile, the OpenCL framework maintained by the Khronos Group develops quite fast and has become a possible competitor of CUDA. Compared with CUDA, OpenCL is open standard and has got support from many companies. Also it supports multiple architectures, which means program written in OpenCL programming language can run on GPUs (AMD, Intel) other than NVIDIA, and even on DSPs and FPGAs, if they provide OpenCL support or at least a subset. The current relations for OpenCL and CUDA is very similar with that for OpenGL and Direct3D when Direct3D was first released by Microsoft in 1996. Who will be the winner is still difficult to predict.

[4]SIMD: single instruction, multi data, which describes computing devices with multiple processing units that can execute the same instruction on multiple data points simultaneously. NVIDIA use the term SIMT (Single instruction, multi threads) to distinguish its GPU from traditional SIMD architecture, such as the Cray style vector machine. But in general SIMT can be regarded as a subset of SIMD. For a detailed introduction, see: `http://semipublic.comp-arch.net/wiki/Single_Instruction_Multiple_Threads_(SIMT)`

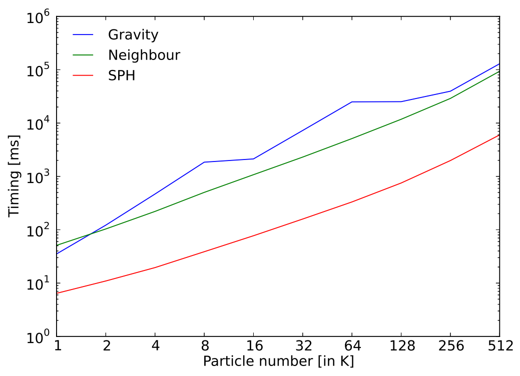
6.2.2 Benchmark result

In this section, I present the benchmark results as a set of performance comparisons between CPU and GPU, for different processes (gravity, SPH and neighbour search) and different particle numbers. In addition, some benchmarks for the raceSPH libraries, ANN libraries and GRAPE card will also be presented. I have to point out that the benchmark results are determined by many ingredients, such as the CPU and GPU occupancy at the testing time, which means other program running on the same node will affect the performance of our test program, by means of shared bandwidth of bus, disk I/O, network traffic and the schedule of operating system, even if they are using different CPU cores and GPU cards. What's more, the results only represent the performance of the codes on specific hardwares (CPU, GPU, etc.) of the test platform. Other platform might yield quite different results.

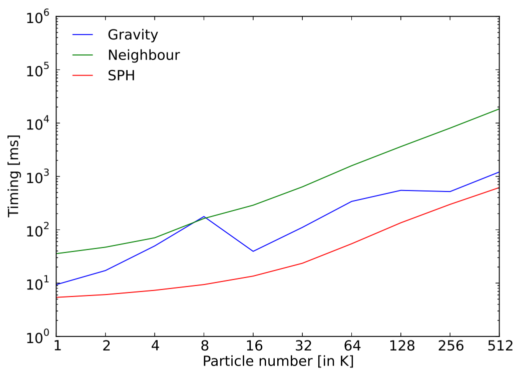
6.2.2.1 Benchmark for gravity, SPH and neighbour search parts at different particles numbers

The benchmark in this section is carried out on "laohu" cluster of National Astronomical Observatories, Chinese Academy of Science (NAOC, CAS). The "laohu" cluster is one of the largest GPU clusters which are specialized on astrophysical numerical simulations and data analysis. It consists of 85 nodes. Each node is equipped with 2 NVIDIA Tesla C1060 GPU cards (240 CUDA cores grouped to 30 SMs on each card with a compute capability of 1.3), two quad-core Intel Xeon E5620 of 2.40GHz and 11GB memory available for applications. The whole cluster was build in 2010. The initial conditions are generated for the same density distribution (Miyamoto & Nagai 1975) with particle numbers from 1 K to 512 K.

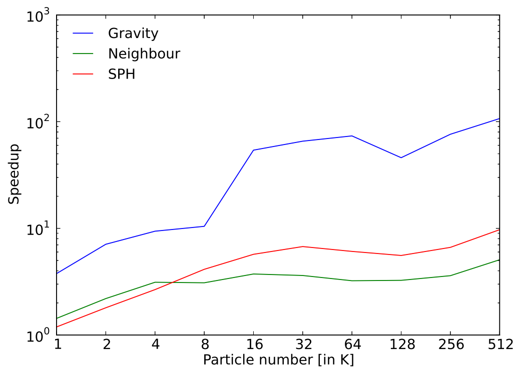
Panel a and b of figure 6.10 demonstrate the benchmark of CPU and GPU for gravity, SPH and neighbour search calculation with particle numbers from 1 K to 512 K (1 K = 1024). Clearly the computation time rises with the increasing particle number. For the gravity calculation on GPU, there is a peak at a particle number of 8 K, which means the computation time is shorter for a larger particle number. Also notice that the corresponding CPU curve becomes flat at the same particle number.This is due to the grouping of *i* particles which is described in appendix A.1.2. In our current program, the parameter `ncrit_for_group` is set to 8192, which means all *i* particles will be grouped into one group when the particle number is 8 K. This group is packed as one block and sent to GPU global memory. However there are only 128 threads in each block in our current setting for the gravity kernel of CUDA, which means the calculations are carried out in 64 loops. For other particles numbers, due to the inhomogeneous particle distribution, the number of particles in each group will not reach 8192, which requires less loops and a shorter calculation time. This peak can be avoided by changing `ncrit_for_group` to a smaller value, but we will not do that since our code is designed for a particle number much larger than that, say, 64 K to 128 K. 8192 is a appropriate value in that case: all GPU cores will be fully occupied. For the SPH and neighbour search calculation, the curve rises more smoothly, since their calculations have nothing to do with groups and loops. Instead, each CUDA core is assigned with one target particle, the calculation time is only proportional to the total particle number. One may notice that the time consumed for neighbour



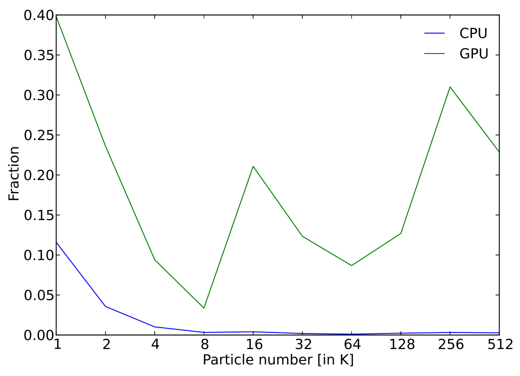
(a) CPU performance.



(b) GPU performance.



(c) Speedup.



(d) Tree construction as a fraction of total time for gravity calculation.

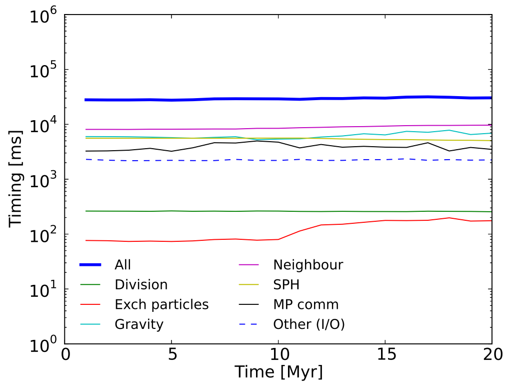
Figure 6.10: Performance comparison between CPU and GPU and the corresponding GPU speedup.

search is already comparable with gravity calculation in the CPU case, but previously we have mentioned that the gravity calculation takes up more than 85% of the total time. The explanation is in our model neighbour search is carried out every 10 timesteps, which means its time consumption can be compared with gravity and SPH calculation only after it is divided by a factor of 10.

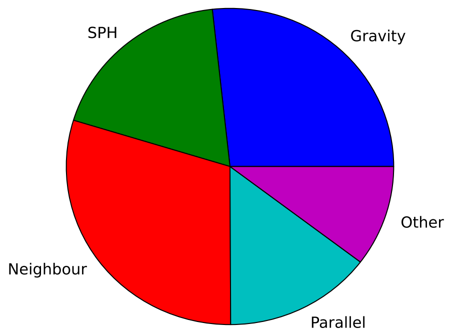
Panel c exhibits the corresponding GPU speedup. One can easily find out that the gravity part achieves an amazing speedup of 100× at 512 K particles, and stabilizes at a value above 50×, which proves that gravity calculation is very suitable for GPU acceleration. The SPH part achieves a maximum speedup of 9.5×, and stabilizes at a value around 5.5× when the particle number is larger than 16 K. Section 6.2.2.4 reports a maximum speedup of 24×, however that is in "titan" cluster whose hardware configuration is totally different from that of "laohu". By analyzing the CUDA kernel, we may find that the too many conditional statements and the high frequency non-coalesced global memory data transfer are the main reason for the low efficiency of the GPU code. The neighbour search part only achieves a maximum speedup of 5× and sustained at 3.5×, considering so many CUDA cores (240) compared with one CPU core, the speedup is quite low. The reason is similar with that for SPH: in neighbour search there are even more conditional statements since the algorithm is much more complex as demonstrated in appendix A.2.2. The algorithms of k-d tree is quite mature which is almost impossible to be improved. To further speedup GPU neighbour search, one can only rely on the development of GPU such that the conditional statement can be handled more efficiently in the future. Another method is by adopting a better GPU cards such as the K20 series, which has more CUDA cores and higher memory bandwidth.

Panel d demonstrates the fraction of tree construction among the total time of gravity calculation as a function of particle number. The tree construction corresponds to a computation time of $O(N \log N)$, whose time is negligible when the particle number is large. For CPU, the time fractions are less than 0.5% when particle numbers are above 8 K. However, for GPU, the fraction fluctuates between 10% and 30%, since the gravity calculation is already accelerated by GPU, and the tree construction is still done by CPU. Very naturally one may think about accelerating this part also by GPU, but unfortunately this is not realistic based on current GPU architecture. In general the tree is constructed by inserting particles into the tree one by one, which is a pure serial process. One way to parallel this process is to insert multiple particle simultaneously, and use atomic lock to guarantee that each time only one tree node can be created or divided, so as to keep the consistency of the tree. However, this will lead to the blockage of memory writing, which yields a tree construction time much longer than the CPU version. Actually there is already efficient way for tree construction with GPU, as I will demonstrate in section 6.2.2.2. The construction of k-d tree on GPU has similar problem, but it is not so important since the speedup of GPU for neighbour search part is low. The fraction of k-d tree construction is still small.

Figure 6.11 demonstrate the benchmark result for a typical run mentioned at the beginning of this chapter. According to the pie chart, gravity part is already not the bottleneck of the program after it is accelerated by GPU for more than 70 times. The corresponding time consumption is already comparable with that of SPH and neighbour search. Neighbour search consumes most of the time, due to its relatively low efficiency on GPU. The fraction of parallel



(a) Timing as a function of time.



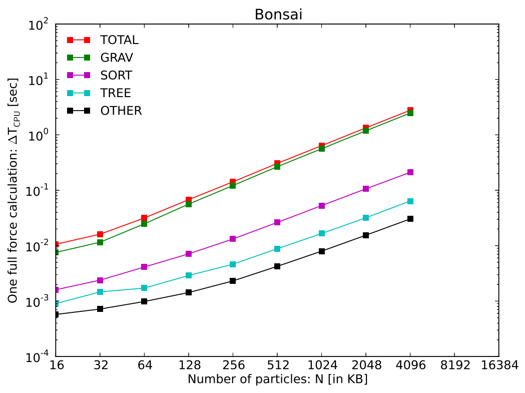
(b) Fraction of total time for the most time consuming tasks.

Figure 6.11: Benchmark of a GPU run for the first 20 Myr: 2 million particles, 8 threads, neighbour lists are updated every 10 timesteps. According to the pie chart, the gravity, SPH, neighbour search and parallel parts take 26.7%, 18.6%, 29.8% and 14.6% of the total time, respectively. The corresponding GPU speedup for gravity, SPH and neighbour search parts are 73.2×, 6.4×, 4.0×. A speedup of 22.6× is achieved for the whole program.

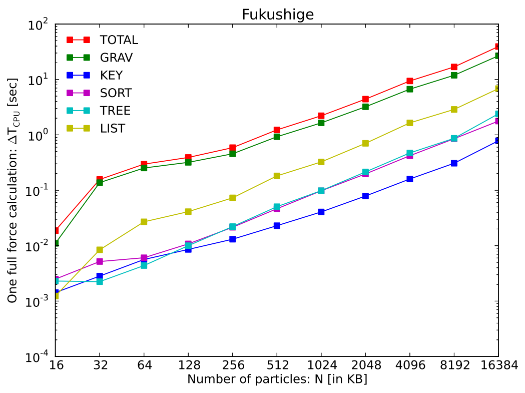
part also rises, which is mainly due to the load imbalance. The program achieves an overall speedup of 22.6×, which means the pure GPU run which would take more than one month can be finished within two days, as our ideal running time on the GPU cluster "laohu".

6.2.2.2 Benchmark for Bonsai and Fukushige GPU tree code

Bédorf et al. (2012) present their GPU Tree-Code "Bonsai", in which all parts of the tree algorithms are executed on GPU. To construct the tree using GPU, they first sort the particles according to a Z-order space filling curve (Morton-order), and then construct the tree based on the reordered particle list. We carry out a set of benchmarks on the Bonsai code and the Fukushige tree code for a Plummer sphere particle distribution with particle number ranges from 16 K to 16384 K (1 K = 1024). The results is shown in figure 6.12. These benchmarks are carried out on the "Hydra" cluster located in ARI, Heidelberg. The "Hydra" is a small GPU cluster which consists of 8 nodes. Each node is equipped with an Intel Core i5 CPU @ 3.30GHz, and one NVIDIA GeForce GTX 570 graphics card with 1.3 GB global memory. Due to the limited size of GPU memory, the benchmarks for Bonsai stops at a particle number 4096 K. From the figure, The Bonsai code is 3 to 4 times faster than the Fukushige code at large particle numbers. This is mainly due to all the algorithms are executed on GPU, which greatly eliminates the data transfer time between CPU and GPU. What's more, with their well written GPU code, the tree construction on GPU (sort and tree build) is 3 times faster than that on CPU. However, executing all the algorithms on GPU makes the hybrid code much



(a) Bonsai.



(b) Fukushige.

Figure 6.12: Timing for Bonsai and Fukushige GPU tree gravity code.

more complex and difficult to modify, which is not suitable for our model that many physical processes need to be implemented. Therefore we still prefer the Fukushige tree code, in which the GPU tree gravity part is packed as several functions with simple interface, and can be easily integrated with other physical processes.

6.2.2.3 Benchmark for Grape card

Figure 6.13 demonstrate the benchmark result for a Grape card. The test was made on the GPU cluster "grape" of Institute for Astronomy, University of Vienna, Austria. Each node of the "grape" cluster is equipped with two quad-core Intel Xeon X5365 of 3.00 GHz and 16 GB memory, one NVIDIA GeForce GTX 580 GPU card (512 CUDA cores grouped to 16 SMs with a compute capability of 2.0) and a Grape6-Blx64 card (4 Grape6 VLSI Chips, 6 pipeline per chip, see Makino et al. (2003) for a detailed introduction). The program uses exactly the same host code for tree construction and the preparation of interaction group list (Fukushige et al. 2005). The switch between different hardwares is done by linking the corresponding libraries. As shown in the figure, GPU achieves a speedup of 200× compared with CPU, while for Grape card the speedup is 60×, which is 3 times lower compared with GPU.

What is shown in figure 6.13 are not merely several curves, but a history of hardware evolution. The research group of Grape cards has been the pioneer of using special hardwares for high performance computing for more than 10 years, during which they have won several Gorden-Bell prizes. The Grape cards achieve somewhat success also on the commercial side. They are adopted by many laboratories specialize on N-body simulations. However, with the release of CUDA and the first generation of programmable graphics cards (the G80 series), GPGPU becomes popular and quickly dominates the field of hardware acceleration. The GPU cards are much faster, and usually come with a much lower price. Take the Grape and GPU card in the figure for example. The Grape6-Blx64 is released in 2003 with a performance of 131.3 GFLOPS and a price of $6000, while the GeForce GTX 580 was released at the end

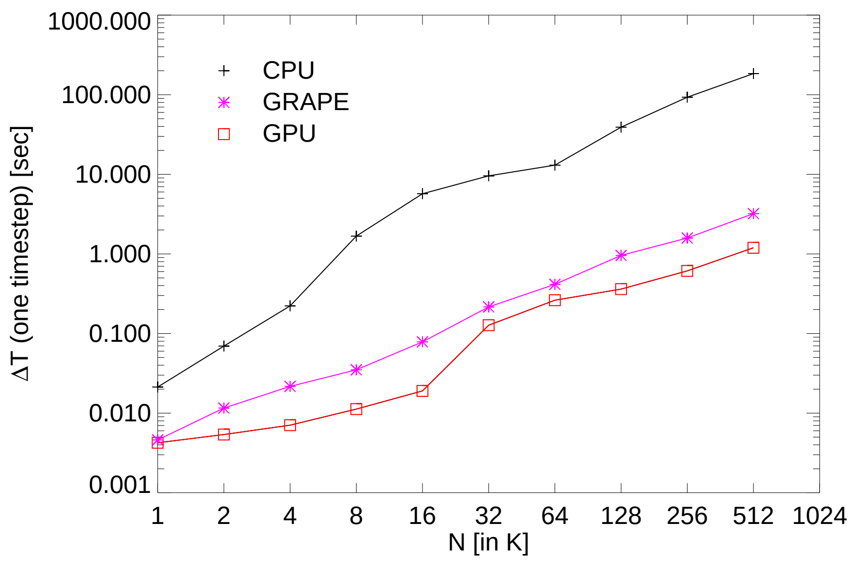


Figure 6.13: Performance comparison between CPU, Grape card and GPU for gravity calculation.

of 2010 with a performance of 1581.1 GFLOPS and a price of $500. Within one decade, the performance is improved for more than 10 times, and the price is reduced for more than 10 times. The hardwares become more general purpose. Softwares become much easier to develop. The development of Grape cards stops upon the emerge of GPGPU. In the last decade, we have witnessed the end of an old age, and the start of a new age. We should show our respect to the precursors of special hardware computing. This field cannot develop so fast without their effort.

6.2.2.4 Benchmark for raceSPH library

Figure 6.14 presents the benchmark result for the raceSPH library. The benchmark is carried out on the GPU cluster "titan" of Astronomisches Rechen-Institut (ARI), which is part of the Center for Astronomy of Heidelberg University (Zentrum für Astronomie der Universität Heidelberg, ZAH). The cluster consists of 32 nodes, each node is quipped with two single core Xeon CPUs (family 14, model 4, 1MB cache, 3.2GHz), one NVIDIA GeForce 8800 GTS 512 graphics card (128 cores, 1.8GHz) and 4GB memory. Nodes are interconnected with 20 Gbit Infiniband links.[5] As introduced in section 6.1.2.2, the raceSPH library developed by Guillermo Marcus provides a uniform interface for SPH calculations. The library is capable of computing with several hardware accelerators, including SSE instructions in the CPU, FPGAs and GPUs. However, the benchmark result for FPGA (MPRACE board) is not available due

[5]Currently the "titan" cluster is updating, the hardwares described above will be abandoned after the update.

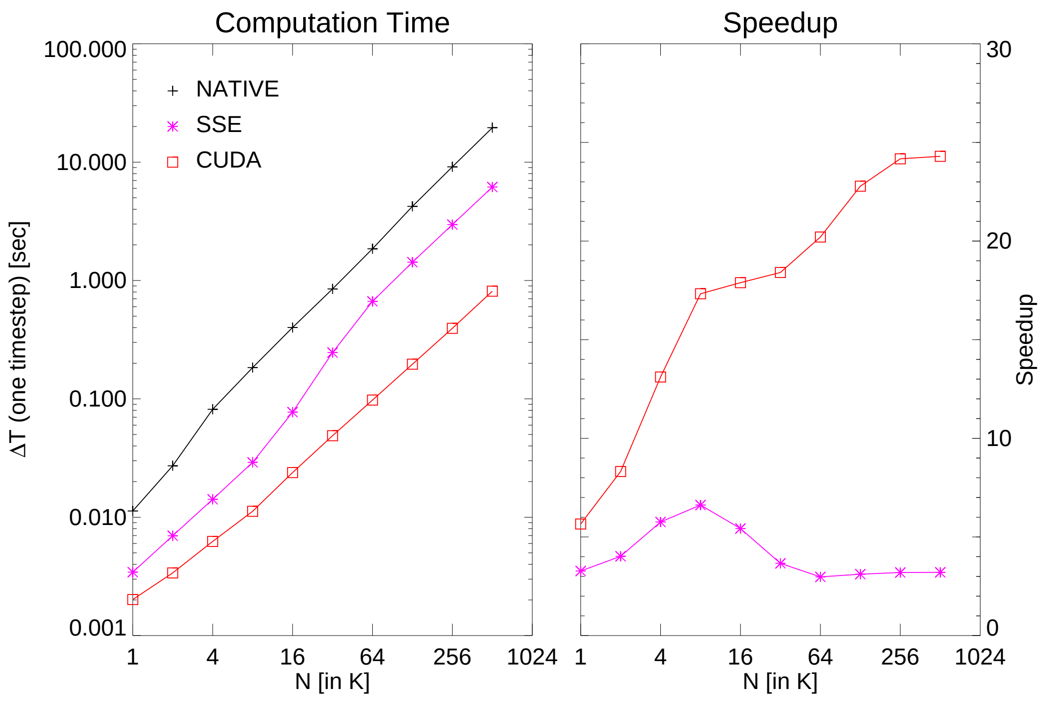
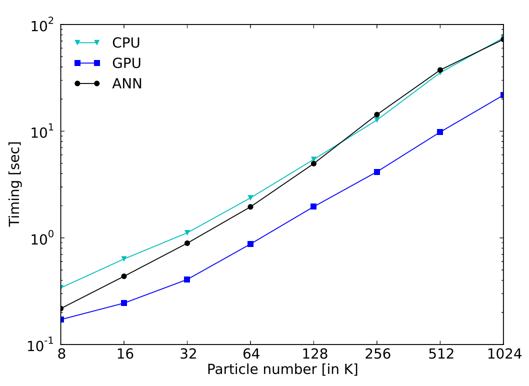
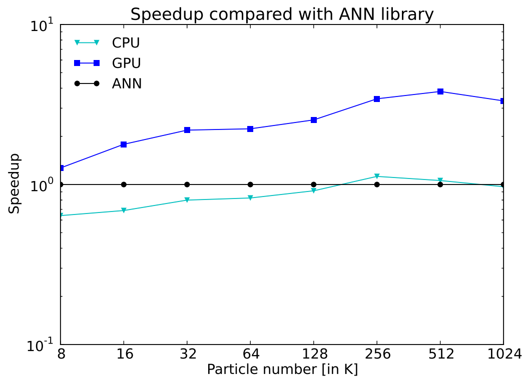


Figure 6.14: Benchmark of raceSPH library with different hardware accelerators.



(a) Time consumption



(b) Speedup.

Figure 6.15: Benchmark of neighbour search as a comparison with ANN library

to the update of the linux kernel when the test was made. According to the figure, GPU is definitely more powerful compared with the use of SSE extensions in CPU. A maximum speedup of 24× is achieved in the GPU run for 512 K particles.

6.2.2.5 Benchmark for neighbour search as a comparison with ANN library

We compare the performance of neighbour search between our implementation based on Press et al. (2007) and the widely used ANN library. The benchmark is carried out on "laohu" cluster for a Plummer sphere particle distribution. According to figure 6.15b, our CPU implementation achieves 80% of ANN library's performance in 64 K particle case, and becomes faster than ANN library when the particle numbers are larger than 128 K. In general, the performance of our implementation is comparable with that of ANN library. This is not surprise, since both libraries uses almost the same algorithm but just different implementation using C++. Moreover, our GPU implementation achieves a maximum speedup of 4× relative to ANN library, which is consistent with the benchmark result in section 6.2.2.1.

6.3 Parallelization

Our model is developed in two steps: first, a serial version was developed which mainly focuses on the implementation of physical processes. In this version only the gravity calculation is accelerated using GPU with Fukushige's tree code. After the model became mature and had already produced some scientific results, we developed the parallel version to support a larger particle number and to utilize the computing resources of modern GPU clusters. In the parallel version particles are divided into several domains according to their position. Different domains exchange particle information via MPI (Message Passing Interface). All of the three time consuming processes (gravity, SPH and neighbour search) are accelerated with GPU. Currently the code has been tested and runs well in 32 threads with up to 20 million particles.

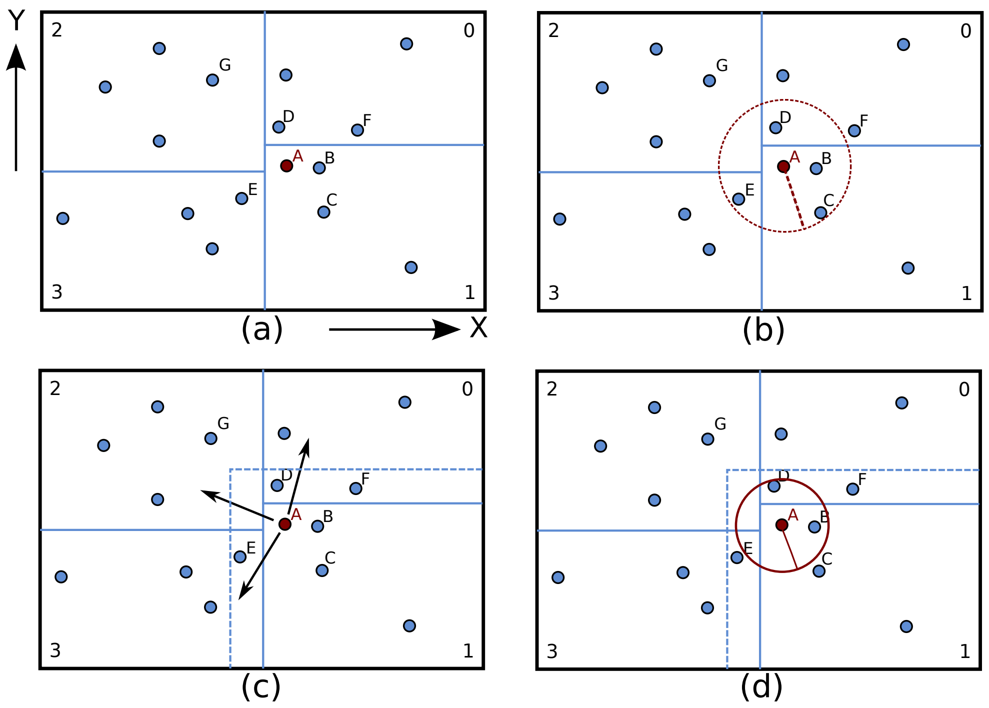


Figure 6.16: Demonstration of domain decomposition and external particle import process.

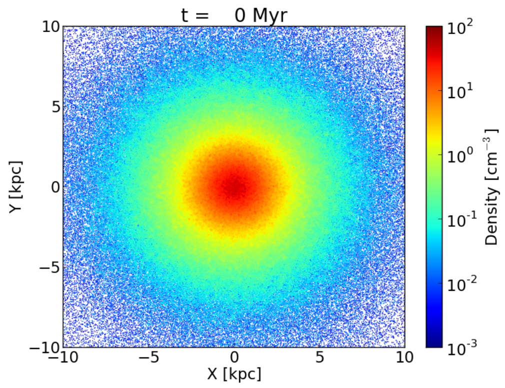
Parallelization is an important part of my work, and takes me a significant amount of time. In this section I explain the parallelization part in detail, including the domain decomposition algorithm, how particles are imported and export between nodes and some scaling tests for different thread numbers.

6.3.1 Domain decomposition and particle exchange

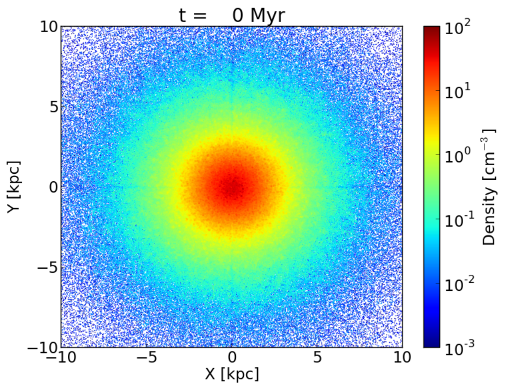
The domain decomposition scheme used in current model is taken from Hamada's GPU tree code. In their scheme the simulation box are subdivided along one coordinate cyclically, such that roughly the same number of particles are assigned on both sides of the dividing plane. This process is somewhat similar with the construction of k-d tree in three dimension. The difference is, in the k-d tree case the simulation boxes are subdivided when one particle is inserted into the tree, the number of nodes is determined by particle number and distribution. In the domain decomposition scheme, the number of nodes has been fixed to the thread number. In general, the domain decomposition process can be summarized in the following steps:

(a) Determine the number of nodes along each dimension such that `nx`×`ny`×`nz`=`nproc`. The total number of threads `nproc` can only take the form of 2^n.

(b) Collect sample particles. It is not necessary to take all particles into account for the division process when the particle number is large. A subset in the order of 100 K is already enough to represent the particle distribution.



(a) With external particles.



(b) Without external particle.

Figure 6.17: Comparison of snapshots with and without including external particles

(c) Sort the x, y, z coordinates in a cyclical way and divide particles into different domains. The minimum and maximum coordinates of domains are determined in this way (domain always takes the form of box, which is defined by its minimum and maximum coordinates).

(d) Determine if a local particle should be exported to other nodes. Prepare the exporting list. Each time one node can only exchange particles with one another node, therefore `nproc` nodes require `nproc-1` times exchange.

(e) Exchange particles between nodes. Domain decomposition is done.

Panel a of figure 6.16 demonstrates the decomposed domain in the two dimension case. In total there are 16 particles in the box. First sort the x coordinate, and divide the box into left and right domains. Then sort the y coordinate in the left and right domains separately, and divide the whole box into four sub domains. The advantage of this domain decomposition scheme is, domains are always regular boxes, which are suitable for gravity calculation when determining whether one tree node (in gravity calculation) or particle (in SPH and other neighbour related processes) should be exported to other domains. Such kind of scheme is suitable for cosmological simulations, in which particles distribute homogeneously, the decomposed domains will have similar size, thus achieve a high efficiency. However, when particles are highly clustered, such as in the simulation of galaxy and star clusters, the size of domains might be very different. Compared with the domains in the edge of the galaxy, domains in the galaxy center will be very small, since the density of particles is high there. This leads to the load imbalance problem, which will reduce the efficiency of gravity calculation. Some advanced domain decomposition methods are adopted to reduce the load imbalance in the parallel code, such as the Peano-Hilbert curve that maps the three dimension space to a one dimension line, which has been used in some astrophysical simulation programs (Springel 2005; Schive et al. 2010). In our current model, we still use the decomposition scheme described above. For it will greatly reduce the complexity when determining whether a particle should be exported to other domains.

As mentioned in section 6.1.1.2, after the local tree is constructed, the external tree nodes are imported and inserted to the local tree. The opening angle determines whether a local node should be exported to other nodes or it needs to be further opened. While in the calculation of other processes, such as SPH, stellar feedback, condensation and evaporation, etc., we have to calculate the interactions between target particle and its neighbours. When the target particle resides on the border of the domain, very likely it has neighbours in other domains. In this case, there are two ways to calculate interactions with these external neighbours. One way is to send the target particle to other domains, and take it back after interactions in that domain are done. The value of interactions are accumulated. This way is suitable for the case of individual time step, in which only a small fraction of particles are updated in every sub step. This treatment is adopted by Gadget, for the calculation of SPH force in other domains. Another way is to import all the possible neighbours in other domains to the local node, which enclose the local node like a ghost layer. Then the interactions are carried out for the target particle exactly the same as for other particles without external neighbours. This treatment is suitable in the shared time step scheme, as we have adopted in our model.

The most difficult problem for particle exchange is to determine if a particle should be exported to other domains or not. Our solution for this problem relies on the following simple and reasonable principle:

- The distribution of particle is coherent, two particles close to each other have similar smoothing length.

This principle can be used to determine whether a particle should be exported to other domains or not: if a particle i's smoothing region (a sphere centered at the particle position with a radius of $2h$, h is smoothing length, defined as the half distance to the most distant neighbour) is overlapped with other domains, possibly there is a particle j resides in this region, in this case they are neighbours of each other, particle i should be exported to j's domain. The algorithm of this particle exchange scheme can be described in the following steps:

(a) Do neighbour search for all target particles in the local region. Find out the nearest N_B neighbours in the local region. Calculate the corresponding smoothing lengths.

(b) If one particle's smoothing area is overlapped with another domain, label this particle. Later it will be exported to that domain. If this particle is overlapped with more than one domain, it will be exported to all these domains.

(c) Prepare the export list for all other domains, exchange particles between domains.

(d) Do neighbour search again, by including particles imported from other domains. Create the new neighbour list. Particle exchange is done.

Panel 6.16b, c, d of figure 6.16 demonstrate this process: panel b: do neighbour search on local nodes, find particle A's most distant neighbour (assuming a neighbour number $N_B = 2$)

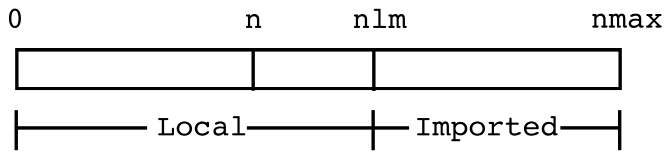


Figure 6.18: Arrangement of particle array.

is particle C. The corresponding smoothing region (encircle by red dashed circle) is overlapped with domain 0, 2, 3. Panel c: particle A is exported to domain 0, 2, 3, while particle E, D, F are exported to domain 1, as their smoothing regions are overlapped with domain 1. They consist of the ghost layer of domain 1, as is marked by the blue dashed lines. panel d: do neighbour search again by including the external particles. This time particle D in domain 0 is included as A's neighbour. Particle C is kicked out from the neighbour list.

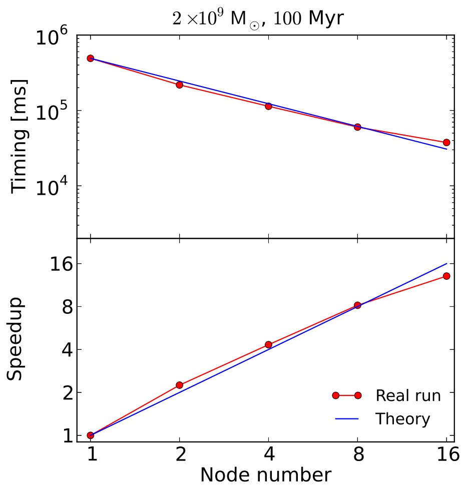
Figure 6.17 present the snapshot of density map for a particle distribution of Plummer-Kuzmin density profile (Miyamoto & Nagai 1975) with and without taking external particles into account. The simulation is carried out in 8 threads. Therefore the simulation box is divided into 8 domains. Due to the symmetry of the particle distribution, the splitting planes are $x = 0, y = 0, z = 0$ respectively. When observe along the z axis, the borders of domains are along the x and y axis. In the right panel, when the density contribution of external particles are not taken into account, the borders of the domains are clearly shown as two lines along the x and y axis, which are due to the lower density of particles on the border compared with inner regions. While in the left panel, when the contribution from external particles is taken into account, the two lines which mark the border of domains disappear. The borders cannot be differentiated. In our model, the total number of particles varies under the regulation of some processes (SPH particle splitting, cold cloud fragmentation and coagulation, star formation, etc.). We have to allocate some redundant space in the particle array such that the newly created particles can be accommodated. Moreover, the external particles should also reside in the same particle array since they are treated the same as local particles in the process of neighbour search, SPH calculation, condensation and evaporation, stellar feedback, and so on. Figure 6.18 demonstrate the arrangement of particle array in the model. Local particles reside in the first `n` slots. From `n` to `nlm-1` (short for `n` local max) are prepared for the newly created local particles. From `nlm` to `nmax-1` are allocated for external particles. Routines in the model differentiate if a particle is external or not simply by its index.

The parallelization of the program still involves how the quantities required in multi-phase calculation transferred back and forth between domains. This will not be explained here since it involves too many details and is of pure technical issues. We leave the demonstration and explanation of this part to appendix A.3.

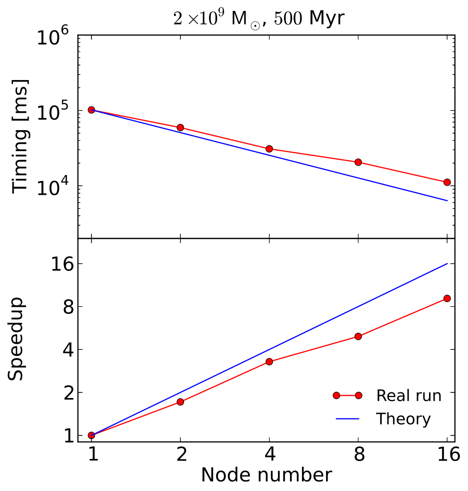
6.3.2 Scaling and consistency test

As a convention for parallel program, in this section we present the scaling and consistency test for parallel version of the multi-phase model.

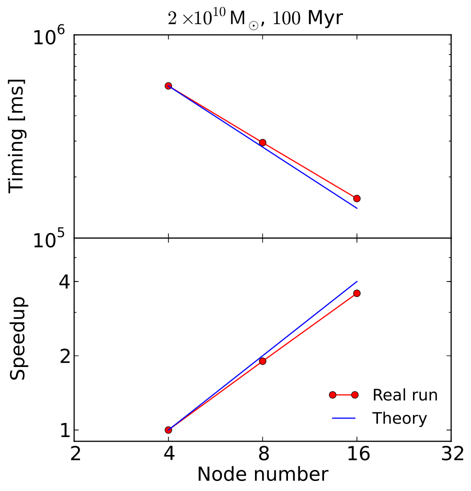
Figure 6.19 demonstrate the results for scaling tests of different particle numbers at differ-



(a) $2{\times}10^5$ SPH particles in IC, time sums from 90 to 98 Myr.



(b) $2{\times}10^5$ SPH particles in IC, time sums from 490 to 498 Myr.



(c) $2{\times}10^6$ SPH particles in IC, time sums from 95 to 99 Myr.

Figure 6.19: Scaling test for different particle number at different time range. Note that the SPH part has not been accelerated by GPU when the tests are carried out.

ent time ranges. The tests are carried out on "titan" cluster of ARI described in section 6.2.2.4. For both 2×10^9 M$_\odot$ (2×10^5 particles) and 2×10^{10} M$_\odot$ (2×10^6 particles) runs, the test starts with IC consists of pure hot/warm (SPH) particles. These hot/warm particles transit to cold clouds. Stars form inside the cold clouds. The 2×10^9 M$_\odot$ runs stop at 500 Myr. 10^5 and $2{\times}10^4$ SPH particles stay in the system at 100 Myr and 500 Myr, respectively. The $2{\times}10^{10}$ M$_\odot$ runs stop at 100 Myr, since such kind of large particle runs are very time consuming on the not so powerful "titan" cluster. 10^6 SPH particles still stay in the system at 100 Myr. In the scaling tests present in this section, SPH calculation is the most time consuming part since it has not been accelerated by GPU. The scalability of the parallel code depends on the fraction of SPH calculation among the total computing time. For both 2×10^9 M$_\odot$ and 2×10^{10} M$_\odot$ runs at 100 Myr (panel a and c), the computation time scales well with node (thread) numbers, since half of SPH particles still stay in the system, SPH calculation takes a significant fraction of total computing time. For 2×10^9 M$_\odot$ runs at 500 Myr, the parallel efficiency at 16 nodes even decreases to 50% of the theoretical value (panel b). This is due to the number of SPH particles is already quite small (10% of initial value), other routines which do not scale well with thread number dominates the computation, such as communications of multi-phase processes. Note that in panel a the speedup is even higher than the theoretical values. This can be explained similarly as the peak in figure 6.10b of gravity calculation: the group of *i* particles leads to too many loops, which finally yields the low efficiency of GPU calculation on one node. On one hand, we can expect that the parallel efficiency will further decrease if the SPH part is also accelerated by GPU, since communication between nodes and load imbalance will consume a significant amount of time. To solve the problem of load imbalance we need to choose the more advanced domain decomposition scheme, which will change the architecture of the code, and cannot be finished in a short time. We leave it to future development since our current program is already eligible to support the expected particle number. On the other hand, one can easily obtain a higher parallel efficiency by running all processes on CPU, as many traditional CPU programs have demonstrated excellent scaling curves. In this sense it is more difficult to achieve a high parallel efficiency for GPU code. This is also the reason I do not present more scaling test.

Figure 6.20 presents the consistency check for star formation rate (SFR) of the $2{\times}10^9$ M$_\odot$ runs. In current model, star formation involves several random numbers, and sensitively depends on the physical quantities (temperature, mass) of cold clouds, which is a highly non-linear process. Any small perturbation will be amplified and finally leads to different results. Such kind of process is demonstrated in the figure, for star formation rate of different thread numbers. The non-linear effect leads to the discrepancies of SFR curves, a maximum difference of 43% is found for the stellar mass at 500 Myr between 2 and 4 nodes. For the mass evolution of hot/warm and cold phase, the differences are within 10%.

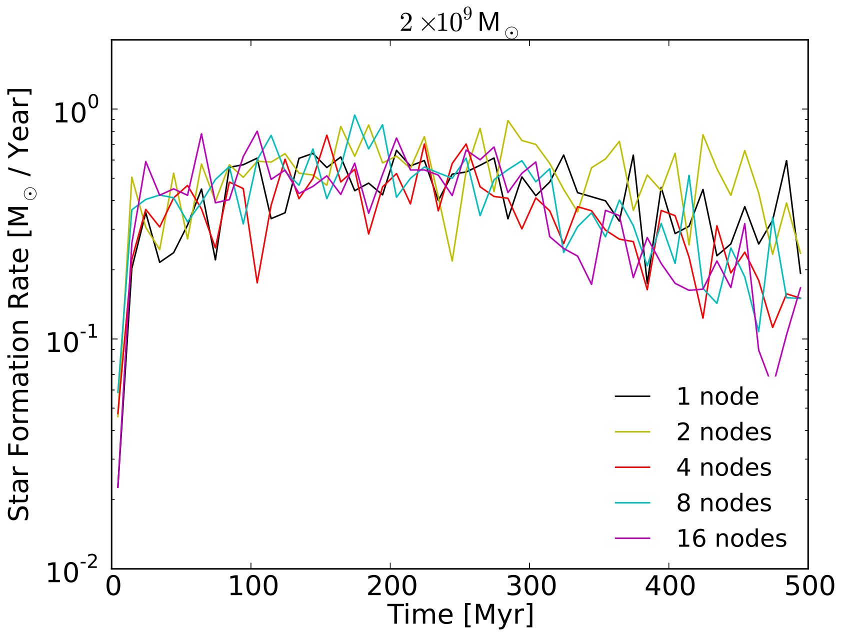


Figure 6.20: Consistency check for star formation rate of the 2×10^9 M$_\odot$ runs

"Pages full of idle word,
Penned with hot and bitter tears;
All men call the author fool,
None his secret message hears."
The Story of the Stone, Cao Xueqin

7 Summary and outlook

7.1 Summary

In this thesis I present our multi-phase model, which gives a better description to the complex structure of ISM compared with the single-phase model. We have formed a realistic cloud mass spectrum from the very generic initial conditions, and reproduced the discrepancies of chemical abundance between different gas phases.

In our model, the ISM is divided into a hot/warm gas component modeled by the SPH method and a cold cloud component described by the sticky particle scheme. The two components can exchange mass and energy via condensation and evaporation and momentum by means of the drag force. Cold clouds can collide and coagulate with each other. Stars are formed inside cold clouds with various star formation recipes. Their feedback can disrupt clouds into small pieces, which is named fragmentation. Four kinds of stellar feedback are implemented in the current model: SNII and SNIa for hot/warm gas, stellar wind and planetary nebulae for cold clouds. All of these processes have different mass and energy ejection rates as a function of time and metallicity.

The evolution of an isolated dwarf galaxy is modeled for 1 Gyr with the multi-phase model described above. The basic picture is that the hot/warm gas accumulates mass from cold clouds via evaporation and expands by converting supernova feedback energy to their kinetic energy. The hot/warm gas particles are accelerated within 5 kpc radius of the galaxy and then escape from the system with a radial velocity between 300 km/s and 400 km/s. The potential well of a dwarf galaxy is not strong enough to hold these high speed hot/warm gas. After 1 Gyr of evolution, 50% of the cold cloud mass transits to hot/warm gas. The mass distribution of cold clouds follows a power law with a slope of -2.3, which is consistent with previous simulation and theoretical prediction. In the end 15% of total baryon mass transits to stars. The SFR decreases from 1 $M_\odot$/yr to 0.1 $M_\odot$/yr at 1 Gyr. High metallicity yields a low SFR, since the corresponding high cooling rate leads to a low temperature and low Jeans mass.

To solve the problem of classical multi-phase model, such as the initial mass fraction and the initial mass distribution of cold clouds, I introduce the transition process such that the hot/warm gas can collapse to cold clouds. According to the parameter studies of galaxies with different mass and rotation, this process is proved to be more suitable for low mass system (see section 4.3.3 for the explanation). Based on this process, I further introduce the process of creating individual stars analytically according to IMF, which aims at resolving individual stars when the newly formed star particles are not massive enough to be regarded as a single stellar population in the high resolution simulations. I apply this process to two simulations of different scales. In the galaxy scale simulation, the life cycle of interstellar medium is reproduced. In the star cluster scale simulation, the star clusters consist of individual stars are created from turbulent velocity field. I find that supernovae feedback is crucial in expelling gas from the star cluster. The initial velocity dispersion is important in determining the compactness of the final star cluster. Lower initial velocity yields a more compact star cluster.

What's more, code development is an important part of my PhD work. I give a detailed introduction to the parallelization of the multi-phase program and the GPU acceleration of the most time consuming parts (gravity, SPH calculation and neighbour search). Our code demonstrates a good scaling relations between performance and node number (figure 6.19), and supports a particle number as large as 2×10^7. With the use of GPU, the program runs one order faster compared with the pure CPU version, which reduces the time consumption of our largest run presented in the thesis from one month to four days.

7.2 Outlook

Several authors have demonstrated that the cold molecular clouds can be studied in a similar way as dark matter. By assuming the Gaussian distribution of the logarithmic density field, the excursion-set model (extended Press-Schechter formalism) is proved to be very successful for the study of cosmology and galaxy evolution. Hopkins (2012) develop it for the study of formation and time evolution of gas structures in the ISM. They use this model to calculate the mass function of the bound gas structures and derive a power-law slope close to -2, which is consistent with observations. What's more, they propose an algorithm based on the excursion-set model for the construction of time-dependent merger/fragmentation trees which can be used to trace the evolution of clouds. The idea is similar with the halo merger trees for the SAM study of galaxy formation. Dobbs & Pringle (2013) study the evolution of molecular clouds in a galactic disc simulation by identifying clouds with a simple clump-finding algorithm. They define the clouds at other times which contains the greatest number of original cloud particles as the *continuation* of the target cloud at specific time, which is very similar with the definition of *progenitor* and *successor* in the construction of halo merger trees. In chapter 4, I demonstrate how cold clumps are identified using an algorithm similar to “SUBFIND”. The latter one are widely used to identify dark matter halos in cosmological simulations (Springel et al. 2001a). As the research focus is shifted from cosmological scale to galaxy scale, more similarities will be found between dark matter particles and molecular clouds, and the research methods will be reused thereby.

In the current single-phase model, star particle is created with the mass taken as a fraction of the hosted SPH particle. It does not correspond to real star, instead, it can be regarded as a single stellar population (SSP), inside which the distribution of stellar mass is assumed to follow IMF. However, there is a lower mass limit ($\sim$ 200 $M_{\odot}$) for such kind of stellar particles. Below this limit, the number of massive stars (> 8 $M_{\odot}$) inside one SSP is less than one. The SSP becomes statistically unmeaningful. As the resolution of simulation increases, this mass limit will be reached inevitably. In that case, one solution is to resolve individual stars (or at least resolve the individual massive stars which are more important for galaxy evolution), which is suggested in chapter 5. Another solution is to link several SPH particles together, and create a massive star particle well above the mass limit. Both of the two solutions involve identifying cold clumps in the dense regions, which corresponds to the transition process discussed in chapter 4.

From the numerical side, as GPU becomes the main stream of hardware accelerator, it is time to reconsider the bottleneck of numerical programs. Traditionally, when the code is running on CPU only, the time consumption of one numerical routine is proportional to the number of floating point operations. Therefore gravity calculation is usually the most time consuming part. However, in the GPU era, the large amount of simple and repeatable floating point operations can be easily mapped to the large number of processing units in GPUs. This is the reason gravity calculation can achieve a speedup of two orders, and has not been the bottleneck any more. However, the SIMD (single instruction, multiple data) architecture of GPU is not suitable for dealing with conditional statements (`if`, `while`, etc.), since each of them induces the divergence of the control flow, which greatly reduces efficiency. One example is neighbour search, whose GPU speedup is much lower than that of gravity. The main reason is its algorithm is much more complex, which leads to a much lower efficiency.

This also gives us some hints on the design of modern numerical program with the presence of hardware accelerator. For the traditional CPU program, many algorithms are invented with the purpose to reduce the total number of floating point operations, which make the software architecture more complex, but save the computing time on CPU. One example is the "nbody" series, in which the gravity force is divided into two parts (Ahmad & Cohen 1973). One part is the regular force, which includes the contribution from distant particles and changes slowly. Another part is the irregular force, which is due to particles in the immediate vicinity of the target particle. The regular force changes much slowly than the regular force, which can be updated in a low frequency. Since the high frequency calculation of irregular force only involves a small number of neighbour particles, force calculation of the whole system is accelerated. However, after the regular force is accelerated using GPU, its time consumption is negligible. The irregular force which is calculated on CPU becomes the bottleneck. Even worse, in the parallel version, all nodes maintain the same particle array which stores all the particle information (position, velocity, etc.). Irregular force calculation requires the high frequency update of this array, which consumes a large amount of communication time. In a modern view, it is already not necessary to distinguish the regular and irregular force, which is designed for the slow CPUs in 1970s. Unifying force calculation will greatly reduce the complexity of architecture, and keep the computation speed comparable with the current version. I have to point out this is just a prediction, the final performance still depends on benchmark results.

Moreover, in the GPU program, the data transfer between CPU and GPU is always time consuming compared with the calculation itself. Therefore it is very necessary to reduce the data transfer frequency between CPU and GPU. One way is to move the main data structure to GPU global memory, and use GPU to finish most of the tasks. Some program has achieved this goal, one example is Bonsai (Bédorf et al. 2012), which implements all the possible parts on GPU, including tree construction, tree walk (gravity calculation), particle reordering, updating of particle information (velocity, position). Because of the data transfer time between CPU and GPU is saved, it is 3× faster than the Fukushige's tree code. However, GPU programming still suffers some problems: the developing and debugging of GPU program are more difficult due to the parallel architecture; the compiler and analyzing tools are still not so mature compared with CPU; what's more, there is even no commonly accepted programming model (CUDA or OpenCL). I believe with the development of GPU technique, these problems will be solved. The use of GPU and other hardware accelerators are still the trend of modern numerical programs.

A Appendix

A.1 Algorithm for BH tree

In this and next section, the algorithms are described in python, which is easier to understand. The actual program of the mutli-phase model is written in C++, which is much more efficient. The purpose of these two sections is to present the algorithms in python instead of providing runnable programs.

A.1.1 BH tree construction

```
# Define a BH tree node:
class Node:
  def __init__(self):
# Total particles in this node (include subnodes):
    self.nparticle = 0
# Subnode list:
    self.child_list = []
# Particle list (useful only if it is a leaf node)
    self.particle_list = []
    self.center = [0.0, 0.0, 0.0]
    self.length = 0.0
# Center of mass when this node is treated as a pseudoparticle:
    self.cm = [0.0, 0.0, 0.0, 0.0]
# Leaf node, defined as no child box
    self.isleaf = True

# Insert a particle to the BH tree:
def insert_particle(ncrit_for_tree, root, particle):
  node = root
  while(True):
    if node.nparticle < ncrit_for_tree:
      node.particle_list.append(particle)
      node.nparticle++
      return
    else:
      if node.isleaf:
        node.particle_list.append(particle)
        node.nparticle++
        allocate_children_and_assign_particles(node)
        return
      else:
# Get the index of the subnode where particle will be inserted
        index = get_index(node.center, particle.pos)
        node.nparticle++
        node = node.child_list[index]
        continue

# Allocate 8 subnodes and insert particles to these nodes:
def allocate_children_and_assign_particles(node):
# Allocate subnodes:
  for i in range(0, 8):
    subnode = Node()
# Set subnode's center according to its index
    set_node_center(i, subnode, node)
    node.child_list.append(sub_node)
# Insert particles of father node:
  for particle in node.particle_list:
    index = get_index(node.center, particle.pos)
    node.child_list[index].particle_list.append(particle)
    node.child_list[index].nparticle++
    del node.particle_list
# Check subnodes such that nparticle <= ncrit_for_tree
  for i in range(0, 8):
    if(node.child_list[i].nparticle > ncrit_for_tree):
      allocate_children_and_assign_particles(node.child_list[i])
  isleaf = False
```

A.1.2 Create group list for *i* particles

Group the *i* particles into several groups. Each group contains a particle number no more than `ncrit_for_group`.

```
# Start with root node and an empty list:
def find_group_list(ncrit_for_group, node = root, group_list = []):
  if node.nparticle > ncrit_for_group:
    for child in node.child_list:
      find_group_list(ncrit_for_group, child, group_list)
  else:
    if node.nparticle > 0:
      group_list.append(node)
```

A.2 Algorithm for *k*-d tree

The algorithm are taken from the book “Numerical Recipe” (Press et al. 2007), which is originally in C++.

A.2.1 *k*-d tree construction

```python
import numpy as np

# Define a k-d tree node:
class KDNode:
  def __init__(self, lo, hi, mom, dau1, dau2, ptlo, pthi):
# Min and max point of a box, in type of point:
    self.lo = lo
    self.hi = hi
# Mother box:
    self.mom = mom
# Two daughter boxes:
    self.dau1 = dau1
    self.dau2 = dau2
# Low and high index of particles in this box:
    self.ptlo = ptlo
    self.pthi = pthi

# Construct a k-d tree:
# pts: particle array
# Particle array size:
npts = np.size(pts)
# Dimension of the tree:
DIM = 3
# The assumed largest number:
BIG = 1.0E20
# Declare a box list:
boxes = []
# Declare two index arrays:
ptindx = np.zeros(npts, dtype = int)
rptindx = np.zeros(npts, dtype = int)
def kdtree_construction(DIM, pts):
  ptss = pts
# Stack for mother box and dimension, in this implementation we use stack
# instead of recursive, although they are equivalent:
  taskmom = np.zeros(50, dtype = int)
  taskdim = np.zeros(50, dtype = int)
  for k in range(0, npts):
    ptindx[k] = k
# Copy point coordinates:
  coords = np.zeros(DIM * npts, dtype = float)
  for j in range(0, DIM):
    kk = 0
    for k in range(0, npts):
      coords[kk + k] = pts[k][j]
    kk += npts
# Initialize the root box:
  lo = [-BIG, -BIG, -BIG]
  hi = [BIG, BIG, BIG]
  boxes.append(KDNode(lo, hi, 0, 0, 0, 0, npts - 1))
  jbox = 0
  taskmom[1] = 0
  taskdim[1] = 0
  nowtask = 1
  while(nowtask > 0):
    tmom = taskmom[nowtask]
    tdim = taskdim[nowtask]
    nowtask -= 1
    ptlo = boxes[tmom].ptlo
    pthi = boxes[tmom].pthi
    hp = ptindx[ptlo:]
    cp = coords[tdim * npts:]
    kk = (np - 1) / 2
    selecti(kk, hp, np, cp)
    hi = boxes[tmom].hi
    lo = boxes[tmom].lo
    hi[tdim] = lo[tdim] = coords[tdim * npts + hp[kk]]
    jbox += 1
    boxes.append(KDNode(boxes[tmom].lo, hi, tmom, 0, 0, ptlo, ptlo + kk))
    jbox += 1
    boxes.append(KDNode(lo, boxes[tmom].hi, tmom, 0, 0, ptlo + kk + 1, pthi))
    boxes[tmom].dau1 = jbox - 1
    boxes[tmom].dau2 = jbox
    if(kk > 1):
      nowtask += 1
      taskmom[nowtask] = jbox - 1
      taskdim[nowtask] = (tdim + 1) % DIM
    if(np - kk > 3):
      nowtask += 1
      taskmom[nowtask] = jbox
      taskdim[nowtask] = (tdim + 1) % DIM
  for j in range(0, npts) rptindx[ptindx[j]] = j
  np.delete(corrds)

# Permutes indx[0..n-1] to make arr[indx[0..k-1]] <= arr[indx[k]] <=
# arr[indx[k+1..n-1]]:
def selecti(k, indx, n, arr): pass
```

A.2.2 Neighbour search with *k*-d tree

```
# Given a point pt, return a list nn of index for the n nearest neighbours:
# nn = np.zeros(n, dtype = int)
def nnearest(pt, nn, n):
  task = np.zeros(50, dtype = int)
  if(n > npts - 1):
    print "too␣many␣neighbours␣requested!";
    return None
  dn = np.zeros(n, dtype = float)
  for i in range(0, n):
    dn[i] = BIG
# Find the smallest box with enough particles to fill in the neighbour list:
  kp = boxes[locate(pt)].mom
  while(boxes[kp].pthi - boxes[kp].ptlo < n):
    kp = boxes[kp].mom
# Save the closest n neighbours:
  for i in range(boxes[kp].ptlo, boxes[kp].pthi + 1):
    d = dist(pts[ptindx[i]], pt)
# The most distant neighbour always locates in index 0:
    if(d < dn[0]):
      dn[0] = d
      nn[0] = ptindx[i]
      if(n > 1):
        sift_down(dn, nn, n)
# Traverse the tree to find closer boxes, using stack:
  task[1] = 0
  ntask = 1
  while(ntask > 0):
    k = task[ntask]
    ntask -= 1
    if(k == kp):
      continue
    if(dist(boxes[k], pt) < dn[0]):
      if(boxes[k].dau1 > 0):
        ntask += 1
        task[ntask] = boxes[k].dau1
        ntask += 1
        task[ntask] = boxes[k].dau2
      else:
        for i in range(boxes[k].ptlo, boxes[k].pthi + 1):
          d = dist(boxes[ptindx[i]], pt)
          if(d < dn[0]):
            dn[0] = d
            nn[0] = ptindx[i]
            if(n > 1):
              sift_down(dn, nn, n)
  return nn

# Given a point pt, return the index of kdtree box it resides in:
def locate(pt):
  nb = 0
  jdim = 0
  while (boxes[nb].dau1 > 0):
    d1 = boxes[nb].dau1
    if (pt[jdim] <= boxes[d1].hi[jdim]):
      nb = d1;
    else:
      nb = boxes[nb].dau2
      jdim += 1
      jdim = jdim % DIM
  return nb

# Calculate the distance between a box and a point:
def dist(box, pt): pass

# Sort array dn, put the largest value to dn[0]. We just need to move
# largest value to index 0, which is an O(n) operation at first glance.
# However, O(log n) can be achieved in a sift down process on the heap,
# as is demonstrated in ‘‘Numerical Recipe’’. We make test to these
# two methods, the discrepancy of speed is quite small.

def sift_down(dn, nn, n): pass
```

A.3 Control flows of the multi-phase program

In this section I describe the control flows of the multi-phase program using pseudo C++ code, together with some comments.

```
// Initialize program: load parameters (unit normalization, cooling table,
// feedback table), read initial condition, distribution particles for the
// first time, activate GPU device, etc.:
mp_initialize();

// Start the main loop. step is the starting step, nstep is the ending step:
for(istep = step + 1; istep <= nstep; ++istep){

  time = istep * dt;
  step = istep;

// Kick (vel, u, mass) to the half step:
  mp_kick();

// Drift (pos) for the full step:
  mp_drift();

// Predict (vel, u, mass) for another half step:
  mp_predict();

// Calculate gravity in a parallel way, with the algorithm described in
// section 6.1.1.2:
  mp_calculate_gravity();

// Import position, velocity, mass, metallicity and type of external
// particles to the local domain:
  mp_exchange_pos_m_type_new();
  mp_exchange_mass_z_new();
  mp_exchange_vel_h_new();

// Density, pressure, Balsara factor (f) calculation:
  mp_calc_den();
  mp_calc_pres();

// Import density, Balsara factor, internal energy (u) and pressure of
// external particles to the local domain:
  mp_exchange_den_f_u_pres_new();

// Calculate SPH interaction, together with the internal energy change due
// to cooling and stellar feedback:
  mp_calc_acc_du();

// Calculate drag force, condensation and evaporation:
  mp_calc_dr_ce();
```

```
// Export internal energy, acceleration, mass, metallicity change of
// external particles due to condensation and evaporation back to their
// original domains:
  mp_exchange_de_ce_back();
  mp_echange_a_dm_back();
  mp_exchange_dm_z_back();

// Kick (vel, u, mass) for another half step:
  mp_kick();

// Coagulation:
  mp_coagulate();

// Some cold clouds coagulate with others, and are set as inactive. This
// change must be exported to other domaines where these clouds serve as
// external particles:
  mp_exchange_is_active();

// Fragmenation:
  mp_fragmentation();

// Recycle cold particles when their masses are small:
  mp_recycle_cold_particle();

// Star formation:
  if(istep % sf_fb_step == 0){

// Choose one SF recipe:
    mp_sf_fragment();
  }

// Domain decomposition and neighbour list construction:
  if(istep % nb_step == 0){

// Split hot particles when they are massive:
    mp_split_hot_particle();

// Recycle hot particles when their masses are small:
    mp_recycle_hot_particle();

// Remove remote particles:
    mp_remove_remote();

// Fill in the empty slots in the particle array with particles at the end
// of the array:
    mp_move_particle();

// Domain decomposition:
    mp_setup_division();

// Hot/warm gas transits to cold clouds (in the code it is referred as
// "hot dump"):
    mp_hot_dump();

// Construct neighbour list after domain decomposition:
    mp_set_neighbours();
  } // domain decomposition and neighbour search

// Stellar feedback:
  if(istep % sf_fb_step == 0 && n_star_tot > 0){

    mp_fb();

// Feedbacks to the external particles are taken back to their own domains:
    mp_exchange_fb_energy_back();
    mp_exchange_fb_dm_back();
    mp_exchange_fb_dm_z_back();
  }

// Control output:
  if(istep % contr_step == 0){

    mp_contr();
    mp_mass();
    mp_trans();
  }

// Snap output:
  if(istep % output_step == 0){

    mp_write();
  }

} // main loop
```

Bibliography

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

Aarseth, S. J. 1999, Celestial Mechanics and Dynamical Astronomy, 73, 127

Aarseth, S. J. 2003, Gravitational N-Body Simulations, Cambridge University Press

Adams, F. C., & Myers, P. C. 2001, ApJ, 553, 744

Adelman-McCarthy, J. K., Agüeros, M. A., Allam, S. S., et al. 2006, ApJS, 162, 38

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

Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R. & Wu, A. 1998, ACM, 45, 891

Asplund, M. 2005, ARA&A, 43, 481

Balsara, D. S. 1995, Journal of Computational Physics, 121, 357

Barnes, J., Hut, P. 1986, Nature, 324, 446

Barnes, J. E. 1990, Journal of Computational Physics, 87, 161

Bate, M. R., & Bonnell, I. A. 2005, MNRAS, 356, 1201

Bédorf, J., Gaburov, E., & Portegies Zwart, S. 2012, Advances in Computational Astrophysics: Methods, Tools, and Outcome, 453, 325

Benson, A. J., Bower, R. G., Frenk, C. S., et al. 2003, ApJ, 599, 38

Bentley, J. L. 1975, Communications of the ACM, 18, 509

Berczik, P. & Petrov, M. 2003, KFNT, 19, No. 1, 36

Bertschinger, E. 1998, ARA&A, 36, 599

Binney, J., & Tremaine, S. 1987, Galactic Dynamics, Princeton University Press

Böhringer H. & Hensler G., 1989, A&A, 215, 147

Bonnell, I. A., & Bate, M. R. 2006, MNRAS, 370, 488

Bonnell, I. A., Larson, R. B., & Zinnecker, H. 2007, Protostars and Planets V, 149

Bonnell, I. A., Vine, S. G., & Bate, M. R. 2004, MNRAS, 349, 735

Booth, C. M., Theuns, T., & Okamoto, T. 2007, MNRAS, 376, 1588

Bower, R. G., Benson, A. J., Malbon, R., et al. 2006, MNRAS, 370, 645

Boylan-Kolchin, M., Springel, V., White, S. D. M., Jenkins, A., & Lemson, G. 2009, MNRAS, 398, 1150

Brown, J. H., Burkert, A., & Truran, J. W. 1995, ApJ, 440, 666

Burkert, A. 1995, ApJ, 447, 25

Caffau, E., Maiorca, E., Bonifacio, P., et al. 2009, A&A, 498, 877

Clarke, C. J., Bonnell, I. A., & Hillenbrand, L. A. 2000, Protostars and Planets IV, 151

Clark, P. C., & Bonnell, I. A. 2006, MNRAS, 368, 1787

Cole, S., Aragon-Salamanca, A., Frenk, C. S., Navarro, J. F., & Zepf, S. E. 1994, MNRAS, 271, 781

Cole, S., Lacey, C. G., Baugh, C. M., & Frenk, C. S. 2000, MNRAS, 319, 168

Cooley, J. W., Tukey, J. W. 1965., Math. Comp., 19, 297

Cowie, L. L., McKee, C. F. & Ostriker, J. P. 1981, ApJ, 247, 908

Croton, D. J., Springel, V., White, S. D. M., et al. 2006, MNRAS, 365, 11

Dalgarno, A. & McCray, R. A. 1972, ARA&A, 10, 37

de Blok, W. J. G. 2010, Advances in Astronomy, 2010,

De Lucia, G., & Blaizot, J. 2007, MNRAS, 375, 2

Dobbs, C. L., Bonnell, I. A., & Clark, P. C. 2005, MNRAS, 360, 2

Dobbs, C. L., Burkert, A., & Pringle, J. E. 2011, MNRAS, 417, 1318

Dobbs, C. L., & Pringle, J. E. 2013, arXiv:1303.4995

Elmegreen, B. G. & Efremov, Y. N. 1997, ApJ, 480, 235

Eskridge, P. B., & Frogel, J. A. 1999, Astrophysics and Space Science, 269, 427

Federrath, C., Banerjee, R., Clark, P. C., & Klessen, R. S. 2010, ApJ, 713, 269

Fellhauer, M. 2001, Dynamics of Star Clusters and the Milky Way, 228, 422

Ferguson, H. C., & Binggeli, B. 1994, The Astro Astrophys Rev, 6, 67

Frigo, M., & Johnson, S. G. 1998, Proc. IEEE Intl. Conf. Acoustics Speech and Signal Processing, 3, 1381

Fryxell, B., Olson, K., Ricker, P., et al. 2000, ApJS, 131, 273

Fu, J., Guo, Q., Kauffmann, G., & Krumholz, M. R. 2010, MNRAS, 409, 515

Fu, J., Kauffmann, G., Huang, M., et al. 2013, arXiv:1303.5586

Fukushige, T., Makino, J., & Kawai, A. 2005, PASJ, 57, 1009

Fukui, Y., & Kawamura, A. 2010, ARA&A, 48, 547

Gerritsen, J. P. E., & Icke, V. 1997, A&A, 325, 972

Gillmon, K., & Shull, J. M. 2006, ApJ, 636, 908

Gingold, R., & Monaghan, J. 1977, MNRAS, 181, 375

Girichidis, P., Federrath, C., Banerjee, R., & Klessen, R. S. 2011, MNRAS, 413, 2741

Grebel, E. K. 2001, Astrophysics and Space Science Supplement, 277, 231

Greengard, L., & Rokhlin, V. 1987, Journal of Computational Physics, 73, 325

Greggio, L., & Renzini, A. 1983, A&A, 118, 217

Gunn, J. E., & Gott, J. R., III 1972, ApJ, 176, 1

Guo, Q., White, S., Boylan-Kolchin, M., et al. 2011, MNRAS, 413, 101

Hamada, T., & Iitaka, T. 2007, arXiv:astro-ph/0703100

Harfst, S., Theis, Ch., & Hensler, G. 2006, A&A, 449, 509

Harfst, S., Gualandris, A., Merritt, D., et al. 2007, New Astronomy, 12, 357

Hensler, G. 1987, Mitt. Astron. Ges., 70, 141

Hensler, G., & Burkert, A. 1990, Ap&SS, 171, 149

Hockney, R. W., & Eastwood, J. W. 1988, Computer Simulation Using Particles, Bristol: Hilger

Hollenbach, D., & McKee, C. F. 1979, ApJS, 41, 555

Hopkins, P. F. 2012, MNRAS, 423, 2016

Hopkins, P. F., Quataert, E., & Murray, N. 2011, MNRAS, 417, 950

Hopkins, P. F., Quataert, E., & Murray, N. 2012, MNRAS, 421, 3488

Israelian, G., Ecuvillon, A., Rebolo, R., et al. 2004, A&A, 421, 649

Iwamoto, K., Brachwitz, F., Nomoto, K., et al. 1999, ApJS, 125, 439

Jerjen, H., & Tammann, G. A. 1997, A&A, 321, 713

Jing, Y. P. 2005, ApJ, 620, 559

Johnstone, D., Matthews, H., & Mitchell, G. F. 2006, ApJ, 639, 259

Kang, X., Jing, Y. P., Mo, H. J., Börner, G. 2005, ApJ, 631, 21

Katz, N. 1992, ApJ, 391, 502

Kauffmann, G., Colberg, J. M., Diaferio, A., & White, S. D. M. 1999, MNRAS, 303, 188

Kauffmann, G., White, S. D. M., & Guiderdoni, B. 1993, MNRAS, 264, 201

Kitsionas, S., & Whitworth, A. P. 2002, MNRAS, 330, 129

Klessen, R. S. 2011, Proceedings of the 2010 Evry Schatzman School in Aussois, arXiv:1109.0467[astro-ph]

Klessen, R. S., & Burkert, A. 2001, ApJ, 549, 386

Klypin, A., Zhao, H., & Somerville, R. S. 2002, ApJ, 573, 597

Köppen, J., Theis, Ch., & Hensler, G. 1995, A&A, 296, 99

Kormendy, J. 1985, ApJ, 295, 73

Kormendy, J., Fisher, D. B., Cornell, M. E., & Bender, R. 2009, ApJS, 182, 216

Kroupa, P. 2001, MNRAS, 322, 231

Kroupa, P., Tout, C. A., & Gilmore, G. 1993, MNRAS, 262, 545

Kroupa, P., Weidner, C., Pflamm-Altenburg, J., et al. 2011, arXiv:1112.3340

Kudritzki, R. P., Pauldrach, A., & Puls, J. 1987, A&A, 173, 293

Kudritzki, R. P., Pauldrach, A., Puls, J., & Abbott, D. C. 1989, A&A, 219, 205

Lada, C. J., & Lada, E. A. 2003, ARA&A, 41, 57

Lada, C. J., Muench, A. A., Rathborne, J., Alves, J. F., & Lombardi, M. 2008, ApJ, 672, 410

Larson, R. B. 1981, MNRAS, 194, 809

Li, Y., Klessen, R. S., & Mac Low, M.-M. 2003, ApJ, 592, 975

Lieder, S., Lisker, T., Hilker, M., Misgeld, I., & Durrell, P. 2012, A&A, 538, A69

Lipovka, A., Núñez-López, R., & Avila-Reese, V. 2005, MNRAS, 361, 850

Lisker, T. 2012, Astronomische Nachrichten, 333, 405

Lisker, T., Grebel, E. K., & Binggeli, B. 2006, AJ, 132, 497

Lisker, T., Grebel, E. K., Binggeli, B., & Glatt, K. 2007, ApJ, 660, 1186

Lisker, T., Weinmann, S. M., Janz, J., & Meyer, H. T. 2013, MNRAS, 432, 1162

Liu, L., Yang, X., Mo, H. J., van den Bosch, F. C., & Springel, V. 2010, ApJ, 712, 734

Lombardi, M., Alves, J., & Lada, C. J. 2010, A&A, 519, L7

Lucy, L. 1977, AJ, 82, 1013

Makino, J., Fukushige, T., Koga, M., & Namura, K. 2003, PASJ, 55, 1163

Makino, J., & Hut, P. 1988, ApJS, 68, 833

Makino, J. 1991, PASJ, 43, 621

Makino, J., & Taiji, M. 1998, Scientific Simulations with Special-Purpose Computers–the GRAPE Systems

Marcus, G. 2011, *Acceleration of Astrophysical Simulations with Special Hardware*, Ph.D. Thesis, Heidelberg University: Germany

Marri, S., & White, S. D. M. 2003, MNRAS, 345, 561

Matteucci, F., & Greggio, L. 1986, A&A, 154, 279

Matzner, C. D., & McKee, C. F. 2000, ApJ, 545, 364

Mayer, L., Governato, F., Colpi, M., et al. 2001, ApJ, 559, 754

McKee, C. F., & Ostriker, E. C. 2007, ARA&A, 45, 565

McWilliam, A. 1997, ARA&A, 35, 503

Mitchell, N., Vorobyov, E., Hensler, G. 2013, MNRAS, 428, 2674

Miyamoto, M., & Nagai, R. 1975, PASJ, 27, 533

Monaghan, J. J. 1992, ARA&A, 30, 543

Monaghan, J. J., & Lattanzio, J. C. 1985, A&A, 149, 135

Moore, B., Katz, N., Lake, G., Dressler, A., & Oemler, A. 1996, Nature, 379, 613

Morales, M. F., & Wyithe, J. S. B. 2010, ARA&A, 48, 127

Motte, F., Andre, P., & Neri, R. 1998, A&A, 336, 150

Murante, G., Monaco, P., Giovalli, M., Borgani, S., & Diaferio, A. 2010, MNRAS, 405, 1491

Nakasato, N., Mori, M., & Nomoto, K. 2000, ApJ, 535, 776

Navarro, J. F., Frenk, C. S., & White, S. D. M. 1996, ApJ, 462, 563

Navarro, J. F., & White, S. D. M. 1993, MNRAS, 265, 271

Padoan, P., Nordlund, Å., Kritsuk, A. G., Norman, M. L., & Li, P. S. 2007, ApJ, 661, 972

Pang, X. 2012, *A comprehensive study of the young star cluster HD97950 in NGC3603*, Ph.D. Thesis, Heidelberg University: Germany

Paudel, S. 2011, *Early-type dwarf galaxies: Insight from stellar population studies*, Ph.D. Thesis, Heidelberg University: Germany

Pearce, F. R., Jenkins, A., Frenk, C. S., et al. 1999, ApJL, 521, L99

Pearce, F. R., Jenkins, A., Frenk, C. S., et al. 2001, MNRAS, 326, 649

Pelupessy, F. I., Papadopoulos, P. P., & van der Werf, P. 2006, ApJ, 645, 1024

Pelupessy, F. I., van der Werf, P. P., & Icke, V. 2004, A&A, 422, 55

Peters, T., Klessen, R. S., Mac Low, M.-M., & Banerjee, R. 2010, ApJ, 725, 134

Portinari, L., Chiosi, C. & Bressan, A., 1998, A&A, 334, 505

Press, W. H., Teukolsky, S. A., Vetterling, W. T., Flannery, B. P. 2007, Numerical Recipes Third Edition, Cambridge University Press

Price, D. J., & Monaghan, J. J. 2007, MNRAS, 374, 1347

Rahimi, A., Kawata, D., Allende Prieto, C., et al. 2011, MNRAS, 415, 1469

Raiteri, C. M., Villata, M., & Navarro, J. F. 1996, A&A, 315, 105

Rees, M. J., & Ostriker, J. P. 1977, MNRAS, 179, 541

Rivolo, A. R., & Solomon, P. M. 1988, Lecture Notes in Physics (Berlin: Springer Verlag), 315, 42

Robertson, B. E., & Kravtsov, A. V. 2008, ApJ, 680, 1083

Saitoh, T. R., Daisaka, H., Kokubo, E., et al. 2008, PASJ, 60, 667

Salpeter, E. E. 1955, ApJ, 121, 161

Samland, M., & Gerhard, O. E. 2003, A&A, 399, 961

Samland, M., Hensler, G., Theis, Ch. 1997, ApJ, 476, 544

Scalo, J. M., & Pumphrey, W. A. 1982, ApJ, 258, 29

Scannapieco, C., Tissera, P. B., White, S. D. M., & Springel, V. 2006, MNRAS, 371, 1125

Schaye, J., & Dalla Vecchia, C. 2008, MNRAS, 383, 1210

Schive, H.-Y., Tsai, Y.-C., & Chiueh, T. 2010, ApJS, 186, 457

Schombert, J. M., Pildis, R. A., Eder, J. A., & Oemler, A., Jr. 1995, AJ, 110, 2067

Schroyen, J., de Rijcke, S., Valcke, S., Cloet-Osselaer, A., & Dejonghe, H. 2011, MNRAS, 416, 601

Semelin, B., & Combes, F. 2002, A&A, 388, 826

Shapley, H. 1938, Harvard College Observatory Bulletin, 908, 1

Shu, F. H., Milione, V., Gebel, W., Yuan, C., Goldsmith, D. W., Roberts, W. W. 1972, ApJ, 173, 557

Somerville, R. S., & Primack, J. R. 1999, MNRAS, 310, 1087

Spite, M., Cayrel, R., Plez, B., et al. 2005, A&A, 430, 655

Springel, V. 2010, ARA&A, 48, 391

Springel, V., White, S. D. M., Tormen, G., & Kauffmann, G. 2001a, MNRAS, 328, 726

Springel, V., Yoshida, N., & White, S. D. M., 2001b, New Astronomy, 6, 79

Springel, V., & Hernquist, L. 2003, MNRAS, 339, 289

Springel, V. 2005, MNRAS, 364, 1105

Springel, V., Di Matteo, T., & Hernquist, L. 2005a, MNRAS, 361, 776

Springel, V., White, S. D. M., Jenkins, A., et al. 2005b, Nature, 435, 629

Stinson, G. S., Brook, C., Prochaska, J. X., et al. 2012, MNRAS, 425, 1270

Stinson, G.S., Dalcanton, J.J., Quinn, T., et al. 2009, MNRAS, 395, 1455

Stinson, G.S., Seth, A., Katz, N., et al. 2006, MNRAS, 373, 1074

Steinmetz, M. 1996, MNRAS, 278, 1005

Spurzem, R. 1999, Journal of Computational and Applied Mathematics, 109, 407

Spurzem, R., Berczik, P., Marcus, G., Kugel, A., Lienhart, G., Berentzen, I., Männer, R., Klessen, R., Banerjee, R. 2009, Computer Science - Research and Development (CSRD), 23, 231

Sugimoto, D., Chikada, Y., Makino, J., et al. 1990, Nature, 345, 33

Sutherland, R. S., & Dopita, M. A. 1993, ApJS, 88, 253

Tammann, G. A. 1994, European Southern Observatory Conference and Workshop Proceedings, 49, 3

Thacker, R. J., & Couchman, H. M. P. 2000, ApJ, 545, 728

Thacker, R. J., Tittley, E. R., Pearce, F. R., Couchman, H. M. P., & Thomas, P. A. 2000, MNRAS, 319, 619

Theis, Ch., Burkert, A., & Hensler, G., 1992, A&A, 158, 17

Theis, Ch., Ehlerová, S., Palouš, J. & Hensler, G. 1998, Lecture Notes in Physics (Berlin: Springer Verlag), 506, 409

Theis, Ch., & Hensler, G. 1993, A&A, 280, 85

Tinsley, B. M. 1979, ApJ, 229, 1046

Tolstoy, E., Hill, V., & Tosi, M. 2009, ARA&A, 47, 371

Valcke, S., de Rijcke, S., & Dejonghe, H. 2008, MNRAS, 389, 1111

van den Hoek, L. B., & Groenewegen, M. A. T. 1997, A&AS, 123, 305

van Zee, L., Haynes, M. P., & Salzer, J. J. 1997, AJ, 114, 2479

Webbink, R. F. 1985, Dynamics of Star Clusters, 113, 541

White, S. D. M., & Frenk, C. S. 1991, ApJ, 379, 52

White, S. D. M., & Rees, M. J. 1978, MNRAS, 183, 341

Williams, J. P., Bergin, E. A., Caselli, P., Myers, P. C., & Plume, R. 1998, ApJ, 503, 689

Williams, J. P., & McKee, C. F. 1997, ApJ, 476, 166

Yang, X., Mo, H. J., van den Bosch, F. C. 2008, ApJ, 676, 248

Yang, X., Mo, H. J., van den Bosch, F. C. 2009, ApJ, 693, 830

Yang, X., Mo, H. J., van den Bosch, F. C. 2009, ApJ, 695, 900

Acknowledgements

Here comes the joyful moment to write the acknowledgements. First I would like to give my gratitude to Prof. Dr. Rainer Spurzem, my supervisor, who encourages me and gives me support during my PhD study. He teaches me how to overcome difficulties of living and working in totally different cultures for the pursuit of dream. I would like to thank Prof. Dr. Gerhard Hensler, who gives me so many directions on my topic and helps me a lot for my visit to Vienna. I would like to thank Mr. Mykola Petrov, I miss the time we are working together at Vienna, Beijing and Heidelberg. I am also thankful to Dr. Peter Berczik, Dr. Guillermo Marcus, Dr. Hsi-Yu Schive (薛熙于), Prof. Dr. Andreas Just, Dr. Thorsten Lisker, Prof. Dr. Ralf Klessen, Dr. Paul Clark, Dr. Christoph Olczak. Our discussions at different stage of my study form the basis of this thesis. I would like to thank Prof. Dr. Cornelis Dullemond for spending time reviewing my thesis and Prof. Dr. Reinhard Männer for attending my PhD exam.

I would like to thank all my colleagues and friends I have met in the last three years. I thank Dr. Jose Fiesta, Dr. Klaus Rieger, Dr. Christoph Olczak, Dr. Andreas Ernst, Dr. Sophia Lianou, Mr. Justus Schneider, Dr. Sanjaya Paudel, Dr. Alexander Hansson, Dr. Svitlana Zhukovska, Mr. Jan Rybizki, Mr. Daniel Bialas, Ms. Julia Weniger, Dr. Shuang Gao (高爽), Dr. Xiaoying Pang (庞晓莹) and Dr. Shuo Li (李硕), who give me so many help for my life and work in Heidelberg, Beijing and Vienna. I thank all the group members in ARI and NAOC, from whom I learn a lot. I thank Frau Mayer, Frau Pisch, Frau Buchhaupt, Ms. Hazel Wei (魏玉芳) and Ms. Jie Yao (姚洁), who help me get settled in ARI and NAOC. I thank Dr. Peter Schwekendiek, Mr. Changhua Li (李长华) for helping me fix problems of my computer and GPU clusters. I thank Mr. Jayanta Dutta, Mr. Jun Zhou (周骏), Mr. Hao Lu (吕昊), Mr. Fei Xing (邢飞), Dr. Zhongmu Li (李忠木), Dr. Ruizhi Yang (杨睿智), Mr. Difeng Guo (郭迪砜), for their friendship during this beautiful time in Heidelberg. I thank my office mates in NAOC: Ms. Dan Wu (吴丹), Dr. Lingzhi Wang (王灵芝), Mr. Shiyan Zhong (钟诗言), for the pleasant atmosphere in the office. I thank Dr. Fang Zhang (张放), for our relaxed and enlightening walks after dinner in Beijing's "awesome" air.

I would like to thank IMPRS, DFG SFB 881 project, Silk Road Project of NAOC, OeAD for supporting my study in Heidelberg, Beijing and Vienna in the last three years. Special thanks are given to IMPRS coordinator Dr. Christian Fendt, for helping me get the chance to study in Heidelberg University.

I would like to express my gratitude to my parents and my wife. This thesis will never come to an end without their support.

Heidelberg
Lei Liu
September 23, 2013