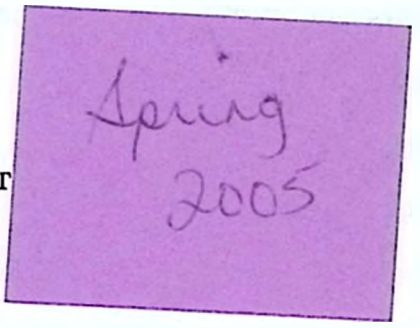
The Effect of Pre-Main Sequence Evolution on Star Cluster Dynamics

The Effect of Pre-Main Sequence Evolution on Star Cluster Dynamics

Robert



A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Master of Science

McMaster University

MASTER OF SCIENCE (2004) McMaster University

(Department of Physics and Astronomy) Hamilton, Ontario

TITLE: The Effect of Pre-Main Sequence Evolution on Star Cluster Dynamics

AUTHOR: Robert Wiersma, B.Sc.

SUPERVISOR: A. Sills

NUMBER OF PAGES: xii, 95

Abstract

The effects of adding pre-main sequence stellar evolution to a stellar dynamics program is investigated. Based on available stellar evolution tracks, pre-main sequence evolution from birth to the zero age main sequence was implemented into the popular dynamics code Starlab. Medium-sized star clusters were modeled under different circumstances, paying special attention to the differences in stellar population. In all, 3 sets of simulations were used. The first was a control set with all stars starting at the main sequence. The second used similar parameters as the first, but with stars beginning their evolution at the pre-main sequence. Because pre-main sequence stars have such large radii, a large number of the binary stars were in contact. For the third set, the binary parameters were adjusted to ensure that all of the binary stars were detached.

The second set of simulations produces a luminosity profile that is dominated by high magnitude stars in the early years of the clusters. It also experiences a large number of mergers, which affect a number of dynamical properties of the models. The mergers lower the binary function of the clusters, which slightly affect the behaviour of its core. More intermediate mass stars abound in the clusters, which leads to higher mass loss through stellar evolution and more high velocity escaping star systems. Fewer blue stragglers are observed since many of the close binaries merge very early on in their existence.

The third set of simulations yields similar results, but mostly for different reasons. There are very few mergers in this implementation, but since there are few hard binaries and more soft binaries many of the multiple systems break up, yielding a similay binary fraction to that of the second set of simulations. Very few of the binaries in these models circularize in stark comparison to the first two sets of models,

which experience circularization in a fraction of its binaries. These models also end up having a slightly higher concentration at the end of the simulation, with a core density of roughly 3 times that of the other sets after 1.5 Gyr.

In general, adding pre-main sequence evolution to star cluster simulations decreases the binary fraction and the number of hard binaries in the cluster. Thus pre-main sequence evolution should be computed for high detail simulations.

Acknowledgements

Thank you to Dr. Alison Sills and the rest of my committee for invaluable input and guidance. I would also like to thank the good people at RHPCS who put up with the temperament of all the software and hardware required to run the simulations. Finally, this work would not be possible without the support and advice of all of the other Starlab/GRAPE users around the globe, most notably, Simon Portegies Zwart.

dedicated to God for strength, and to my wife and son for brightening the worst of days

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview of Star Clusters

Star clusters have been an accessible environment in which to study the properties of stars for the past century. Although some clusters (such as the Pleiades) are visible to the naked eye, telescopes were able to reveal the more precise nature of them. As astronomers such as Messier (1784) began to catalogue objects in the sky (which were recognised as star clusters soon after), their importance became apparent. Herschel (1893) built on Messier's work to dramatically expand the number of recorded clusters. Better telescopes and the ability to record observations as photographs eventually enabled observers to perform star counts of clusters. Bailey (1893) made the first of these counts on ω Centauri and 47 Tucanae which laid down qualitative descriptions of these clusters. The first model was a fit of a function to the stellar distribution to these observations by Pickering (1897).

The emergence of robust theoretical distribution models for star clusters followed. The first serious static models were formulated by Plummer (1905) and Plummer (1911), using various power law forms, and were followed by gradual improvements in the formulations until the 1930s (Meylan and Heggie 1997). After a short period of neglect, star cluster theory was revived, culminating in the model proposed by King (1966) which is still used today. From this point, theoretical research in this area

moved towards dynamical models, which moved the calculations from the pencil and paper and eventually required vast computational resources.

Star clusters represent a halfway point in terms of the scale in which objects in the universe are studied. Any system larger than a star cluster makes the effect of individual evolution and encounters that occur amongst the composite objects negligible and any object that is smaller than a star cluster results in a lot of statistical noise, making the overall dynamics of the system not representative of those sorts of systems. In general, star clusters provide astronomers with laboratories for three major fields: dynamical, binary, and stellar evolution.

Dynamical evolution is found in an ideal form in star clusters. The number of objects is manageable, so the objects of the system can often be tracked individually. It is especially important to have a manageable number of objects when simulating a system where collisions are permitted, as computational requirements can be prohibitive. To study a system of more than 100,000 objects where encounters occur can take months, possibly more depending on the accuracy of gravitational calculations required. On the other hand, star clusters contain enough objects that they are quite protected from the dangers of small number statistics. For systems of less than 500 objects, it is uncertain whether the evolution of a single system can yield general properties of such systems.

Stars often occur in multiple systems with rather tight orbits. Binary systems are the most prominent, and over time can exhibit fascinating phenomena such as mass transfer and encounters with other systems. While these systems are important in themselves, they serve as an interesting interface between the dynamical evolution of the cluster and the stellar evolution of the individual stars. Binary star systems act as energy sinks and can drastically effect the overall properties of the cluster. Before globular clusters were thought to have primordial binaries, dynamically produced

binaries were recognised as a key population for halting core collapse (Elson et al. 1987). If stars in a binary system undergo mass transfer, individual stellar evolution can be affected in a noticeable way as well.

As far as single star evolution is concerned, star clusters are a perfect environment in which to study it. Typically, stars found in clusters have formed roughly at the same time, and with the same material, but have a wide range in masses. This allows astronomers to observe stars that are undergoing many stages of stellar evolution, which is vital for verifying theoretical models. In addition, exotic objects such as so-called "blue stragglers" (which can often only be identified as such in the context of a cluster) manifest themselves most frequently in star clusters.

Two flavours of star clusters exist in the our galaxy: globular clusters and open clusters. Globular clusters in our galaxy represent the old order of stellar associations. Globulars are large clusters with stars numbering in the hundred thousands. As these clusters have ages on order of 10 Gyr, we find that star formation in them is completely finished and high mass stars are absent from their make-up. Found mostly clustered around the galactic centre, old, metal-poor (population II) stars represent standard globular constitution.

Open clusters on the other hand, are young clusters that reside in the galactic disk. These clusters are often less than 1 Gyr in age, and have anywhere from hundreds to tens of thousands of stars. Unlike globulars, a number of open clusters reside within a reasonable range of the solar neighbourhood, so they are often ideal for study. Open clusters contain young, metal-rich (population I) stars which make them attractive for modeling since so much is known about these stars from observations and modelling of stars in the solar neighbourhood. Because open clusters reside in the galactic disk and have a lower total mass, they are more susceptible to tidal effects from the galaxy

itself. This shortens the lifetime of these clusters and masks some of their dynamical properties.

In addition to the above properties, open clusters exhibit many more effects of importance to astronomers. In general, they provide an excellent venue to examine the result of evolution of an N-body system. The processes of mass segregation, core collapse, and evaporation take place on a time scale short enough that they can be noticed when comparing clusters. Their observed high binary fraction (50% (Portegies Zwart et al. 2001)) allows for direct observation of its impact on the dynamical properties of the cluster. Open clusters give an accessible laboratory for various astronomical phenomena.

1.2 Overview of Pre-Main Sequence Evolution

Observable stars exist at various stages of evolution with some on the main sequence, some on the giant branch, and some as remnants. Even the ancients recognised that some stars are of different colour, although it is doubtful that they attributed those colours to an evolutionary sequence. It wasn't until the 20th century that a solid description of how stars evolve formed. The pioneering work of Salpeter (1955) outlined the expected fraction of white dwarf to main sequence stars and speculated that white dwarfs are a later evolutionary stage of main sequence stars. Hayashi (1961) described what a fully convective star would look like, and mapped it to giant stars as well as protostars that are collapsing into main sequence stars.

Sometimes observed stars with the prototype T Tauri, pre-main sequence stars represent the stage of stellar evolution beginning where core deuterium burns and ending where core hydrogen burns. Any object earlier than this phase can be referred to as a protostar and any object that is finished going through this stage should be a

main sequence star. In the protostellar stage, gravitational collapse is the dominant process, but radiation pressure from nuclear burning eventually plays a role throughout the pre-main sequence phase. These stars spend more of this phase of evolution contracting, but have be shown to have subtle nuances in the way their luminosity and temperature evolve (D'Antona and Mazzitelli 1997).

So-called "T Tauri stars" have been known to exist for quite some time, but up until the last half the past century, they were known for their variability. Joy (1945) initially recognised the importance of these objects, probing deep into dense molecular clouds. He assembled the first collection of T Tauri stars and characterized their spectra. Through the work of Herbig (1957) and Walker (1956), it became clear by star cluster aging and H-R diagram placement that these stars represent a very early stage of stellar evolution. Cohen and Kuhi (1979) produced the most complete optical survey of T Tauri stars at that time, and described 4 regions of recent star formation in their work.

Theoretical models of pre-main sequence stars seemed to initially be little more than a distraction to stellar astrophysicists. Modeling efforts in this area began with Henyey et al. (1955) who computed a series of evolutionary tracks based mainly on gravitational contraction. Shortly thereafter, these became known as 'pre-main sequence' stars when Huang (1961) compared the tracks to known early-type stars in NGC 2264 on an H-R diagram. More detailed numerical work followed with Hayashi (1961) and Larson (1969), who made great efforts to make their models self-consistent by adding convection and other physical nuances. Simulations were performed sporadically, with a lot of effort focused on the mechanism for protostellar accretion. This lead to the identification of the mass-radius relation for protostellar cores (Stahler et al. 1980a; Stahler et al. 1980b) and the pre-stellar 'birthline' (Stahler 1983; Cohen and Kuhi 1979). This birthline reflects where deuterium begins to burn and where

stars first appear on the H-R diagram. Work on the pre-main sequence birthline continued into the 1990s and led to improvements in evolutionary models. Recently, models of pre-main sequence evolution have experienced a flurry of interest (Siess et al. 2000; D'Antona and Mazzitelli 1997; Palla and Stahler 1999) as numerous groups are working on producing better stellar evolution tracks, using better understood physics based on more complete observations.

Because the radius of given pre-main sequence star is so much larger than its main sequence counterpart, it is possible that star-star interactions could play an important role in the history of a star cluster. Figure 1.1 demonstrates how the evolution of a 0.8 $M_{\odot}$ star can result in rather drastic changes in radius (Siess et al. 2000). Indeed, the radius of this star remains at a value more than six times the main sequence radius for a for 0.8 $M_{\odot}$ star for a significant period of time and could affect the way in which it reacts to encounters with other stars.

1.3 Overview of Recent Computational Efforts

High-performance computers became the principal method of evolving theoretical models of star clusters somewhere around the middle of the twentieth century. Three major perspectives, gas, Fokker-Planck, and N-body models, dominated these efforts, each with its own benefits and drawbacks. They can often be combined with each other when convenient (Elson et al. 1987). A brief outline of each follows.

As the first inroads into theoretical star cluster dynamics were made, the similarities between a star cluster and a cloud of gas became apparent. Both objects are made up of thousands of components which possess similar apparent randomness in their motions. The main problem with this comparison is that the mean free path of a molecule in a gas cloud is much shorter compared to the size of the cloud than

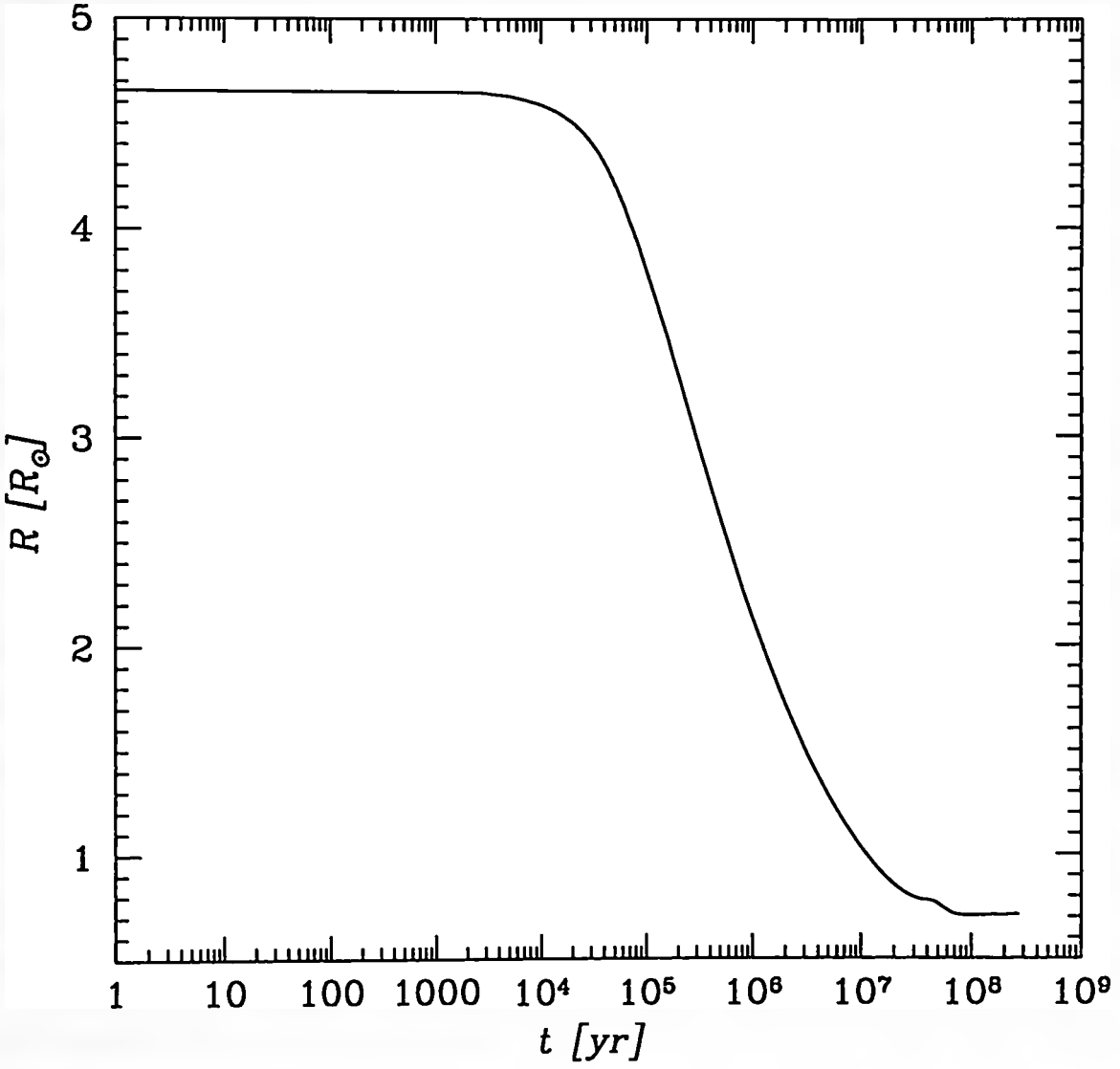


Figure 1.1: Pre-main sequence evolution of the radius of a 0.8 $M_{\odot}$ star using the tracks of Siess et al. (2000). The age of the star is adjusted with the birthline from Palla and Stahler (1999).

a star in a cluster (Meylan and Heggie 1997). Although gas models are rather simplistic, they are relatively easy to implement computationally. These type of models date back to Jeans (1913), but the first serious attempt at gas model simulations was performed by Larson (1970) and work continued along these lines through the 1980s. Interest in this technique has subsequently fallen off.

Towards the middle of this past century, it became apparent that statistical approaches were necessary in order to adequately model large collections of stars. Hénon (1961, 1965) first proposed applying Fokker-Planck equations to star clusters, which is essentially a weak scattering approximation. Great leaps were achieved using this method and some of the first attempts at evolving a cluster through time were made. Hénon (1961) began his models with an infinite central density with an energy flux from a central singularity and later (1965) showed that a star cluster will evolve to this point anyway; a phenomenon now known as "core collapse". Eventually, Hénon (1975) exposed a life beyond this event when he demonstrated that core collapse in a star cluster is better likened to a bounce, as the post core collapse phases of evolution exhibit gravitational oscillations. As the demand for more and more sophistication in simulations increases, the use of Fokker-Planck models has decreased.

Superficially, the most logical way to model star clusters is to keep track of each individual star and calculate the forces between them one by one. This 'N-body' approach was initiated by von Hoerner (1960), who used as many as 16 particles, but serious attempts at N-body modeling didn't occur until the mid-1980s, specifically with Aarseth (1985). Unfortunately, N-body star cluster simulations are rather troublesome. While cosmological simulations of this sort have become quite advanced, the assumptions and resolution used for them produce gross inaccuracies for star clusters (Meylan and Heggie 1997). Because of the scale, cosmological simulations use particles to represent a some mass (dark matter, gas clouds, stars, etc.) of matter

in the universe, rather than an individual object. Star cluster simulations produce a distribution of a small number of objects, while cosmological simulations produce a density distribution using a large number of objects.

The timesteps required to follow a N-body star cluster simulation are typically much smaller than those for cosmological simulations. In fact, if second order differential equations are used, by the time a given interaction has been calculated and carried out, many of the stars have already changed direction completely. This problem is usually solved but checking for energy conservation after each step, but a more proactive approach to ensuring accuracy is by way of a Hermite scheme, see Makino and Aarseth (1992), which uses acceleration and jerk to predict the next location of the particles. Such schemes are quite fast, thus providing computational time that can be spent on other cluster properties.

Over the past decade, dynamical simulations of star clusters have become much more realistic. This realism takes the form of an increasingly complicated treatment of individual stars in the cluster. For years, dynamical models only considered stars as single, equal-mass, non-evolving points. The introduction of a mass function into dynamical models quickly necessitated some treatment of stellar evolution, since high mass stars have much shorter lifetimes than low mass stars. Mass loss from high mass stars can also have a substantial impact on the dynamical evolution of the cluster (Elson et al. 1987). Similarly, the addition of binary evolution to a simulation can significantly impact its outcome.

Stellar dynamicists realized that the point mass approximation for stars was neglecting a number of dynamically significant processes in clusters. Allowing stars to have radii that change as they evolve was an important next addition to stellar dynamics simulations (Kroupa 1995; Portegies Zwart et al. 2001; Hurley et al. 2001). Finite stellar radii are most important for two aspects of these simulations. First,

binary stars can undergo mass transfer as one member of the system fills its Roche lobe, either through evolution of the star or dynamical modification of the orbital parameters of the system. Changing binary systems will change how the binaries affect the evolution of the cluster. In the extreme case, the two components of the binary system can merge. Secondly, stars with finite radii can collide with other stars. The low velocity stellar collisions that occur in star clusters produce blue stragglers (Sills et al. 1997) and could produce other non-standard stellar populations. These populations in turn can change the local dynamical evolution in the cluster.

A number of authors have endeavoured to push the limits of realism in dynamical simulations. Hurley et al. (2001) started their simulations with 10000 - 15000 stars distributed via Plummer and King models with varying fractions of binaries. They assigned masses to these using stars using (Kroupa et al.) (1991, 1993) initial mass functions (IMFs), and evolved them from the main sequence using recipes detailed in Hurley et al. (2000). Their models considered the effects of galactic tidal fields and strip off escaping stars. Because their study focused on blue stragglers, they included collisions and mergers as well as close encounters. Portegies Zwart et al. (2001) started with a somewhat similar model, but with 3096 stars all distributed using a King model, and using a Scalo (1986) mass function. Their treatment of binary and stellar evolution was somewhat different, although they too began their stars on the main sequence. Both these papers were among the first to include a consistent treatment of stellar evolution, a very significant achievement because it is effectively impossible without a high degree of detail. Adding a mass spectrum to an N-body code requires stellar evolution to moderate mass loss, which requires a detailed treatment of binary evolution and accretion since our 'bodies' have radii. Fortunately, this has resulted in data that is looking more and more similar to that taken from observations. Colour-magnitude diagrams and density profiles of these clusters can at

times match up quite nicely with their observed counterparts (Portegies Zwart et al. 2001). In spite of this progress, there is room for improvement in matching observed binarity and luminosity distribution.

Young open clusters are of particular interest when simulating star clusters. Because they tend to have fewer stars, open clusters provide observations that do not require prohibitive computational expense to reproduce. Some open clusters are young enough to have pre-main sequence stars that are observable; indeed numerous authors have reported pre-main sequence stars in the Pleiades (Garcia Lopez et al. 1994; Stauffer et al. 2003). As stated previously, open clusters have both small-scale and large-scale properties that make them an excellent place to start when integrating stellar evolution with dynamics.

Previous work assumed that all stars began their lives on the main sequence (or in some cases, a more evolved state). However, low mass stars make up the bulk of stars by number in a cluster for any reasonable initial mass function. Low mass stars also have a significant pre-main sequence lifetime, significantly influencing a cluster population. These young stars have radii which are upwards of 10 times larger than their main sequence radii. Therefore, some binaries could have undergone an episode of mass transfer that is not taken into account. Also, larger stars are more likely to have experienced a collision; those collision products would have been missed in previous simulations. Including pre-main sequence evolution is regarded as a potentially important step in increasing the realism of stellar dynamics simulations (Sills et al. 2003).

1.4 Thesis Objective

The hypothesis that the addition of pre-main sequence evolution will noticeably affect the overall appearance of the cluster shall be tested. Does this enhancement increase a simulation's ability to mimic observations? Other than apparent age, what properties of the cluster will differ from a simulation begun at the main sequence? Do future star cluster simulations require the inclusion of pre-main sequence stellar evolution or can it be neglected? Simulations begun at the pre-main sequence will be compared to those begun at the main sequence, and the implications for possible observational similarities will be discussed.

Chapter 2

Simulation Environment

2.1 Starlab

The simulations were run using the Starlab environment, version 4.1.1 (Hut 2001). This package features a dynamical integrator which considers stellar encounters by interacting with a binary/stellar evolution module. This way, the entire cluster may be evolved consistently. The two main components of this code are named `Kira` (which controls the N-body integration and the higher order evolution of the system) and `SeBa` (which controls the stellar and binary evolution). The `Kira` component evolves the system dynamically and then sends and retrieves the individual evolution information from `SeBa`.

2.1.1 `Kira`

`Kira` was the main module from which the simulations were performed. From this basic integrator, all of the other required details of the simulation are called. It evolves the system using an adaptive timestep and goes through the following steps each time:

- Flag stars which need to be updated (according to the timestep specified upon the previous iteration). Timesteps are chosen as powers of 2, so there are often a number of stars in this list.

- If necessary, perform various 'housekeeping' operations. These include: system reinitialization, output logging, escaper removal, run termination, and/or snapshot output. Output logging refers to recording certain system parameters whereas snapshot output dumps the position and state of all stars in the system.

- Predict location and velocity of all stars (using the GRAPE (see section 2.2)).

- As per the Hermite scheme (see below), compute acceleration and jerk and correct positions and velocities of flagged stars (using the GRAPE).

- Flag multiple star systems as perturbed or unperturbed by other stars.

- Flag apparent collisions and mergers.

- Reorganise tree, if necessary. That is, if stars are interacting with a multiple system, add them to that node.

- Evolve single and multiple stars with SeBa.

The Hermite predictor-corrector scheme (Makino and Aarseth 1992) is a method of N-body integrating which attempts to reclaim some of the accuracy lost due to the use of discrete timesteps. It uses acceleration and its time derivative (jerk) to compute predicted positions and velocities. New acceleration and jerk values are computed based on the predicted positions and velocities, which are in turn used to correct the motion of the stars. If we let $\mathbf{x}_{proj,i}$, $\mathbf{v}_{proj,i}$, $\mathbf{a}_i$, $\mathbf{v}_i$, and $\mathbf{x}_i$ denote the time-projected position, time-projected velocity, current acceleration, current velocity, and current position respectively for a given timestep dt, then we have

$$\mathbf{x}_{proj,i} = \frac{dt^3}{6}\dot{\mathbf{a}}_i + \frac{dt^2}{2}\mathbf{a}_i + dt\mathbf{v}_i + \mathbf{x}_i \tag{2.1}$$

and

$$\mathbf{v}_{proj,i} = \frac{dt^2}{2}\dot{\mathbf{a}}_i + dt\mathbf{a}_i + \mathbf{v}_i. \tag{2.2}$$

Of course, the acceleration and jerk values are calculated by integration:

$$\mathbf{a}_i = \sum_j Gm_j \frac{\mathbf{r}_{ij}}{r_{ij}^3} \tag{2.3}$$

and

$$\dot{\mathbf{a}}_i = \sum_j Gm_j \left[\frac{\mathbf{v}_{ij}}{r_{ij}} + \frac{3(\mathbf{v}_{ij}\cdot\mathbf{r}_{ij})\mathbf{r}_{ij}}{r_{ij}^5}\right], \tag{2.4}$$

where $\mathbf{r}_{ij}$ and $\mathbf{v}_{ij}$ represent the differences between the *projected* and the current positions and velocities respectively. If $\mathbf{a}_i$ is expanded using Hermite polynomial interpolation, then the second and third order time derivatives calculated at time t, $\ddot{\mathbf{a}}_{0,i}$ and $\mathbf{a}_{0,i}^{(3)}$ are given by

$$\ddot{\mathbf{a}}_{0,i} = \frac{-6(\mathbf{a}_{0,i} - \mathbf{a}_{1,i}) - dt(4\dot{\mathbf{a}}_{0,i} + 2\dot{\mathbf{a}}_{1,i})}{dt^2}, \tag{2.5}$$

and

$$\mathbf{a}_{0,i}^{(3)} = \frac{12(\mathbf{a}_{0,i} - \mathbf{a}_{1,i}) + 6dt(\dot{\mathbf{a}}_{0,i} + \dot{\mathbf{a}}_{1,i})}{dt^2}, \tag{2.6}$$

where $\mathbf{a}_{1,i}$ and $\dot{\mathbf{a}}_{1,i}$ are calculated at time $t + dt$. Finally, the corrected position and velocity will be

$$\mathbf{x}_i(t + dt) = \mathbf{x}_{proj,i} + \frac{dt^4}{24}\ddot{\mathbf{a}}_{0,i} + \frac{dt^5}{120}\mathbf{a}_{0,i}^{(3)}, \tag{2.7}$$

and

$$\mathbf{v}_i(t + dt) = \mathbf{v}_{proj,i} + \frac{dt^3}{6}\ddot{\mathbf{a}}_{0,i} + \frac{dt^4}{24}\mathbf{a}_{0,i}^{(3)}. \tag{2.8}$$

To deal with binary and higher order systems, **Kira** employs a linked-list to compose a tree structure, where single stars are counted as leaves. Figure 2.1 shows how binaries make up branches so that when the integration is performed, these systems make up nodes, and count as one object for gravity to interact with. This simplifies the situation drastically since the orbital positions in binary stars are left to SeBa to calculate. When encounters with other stars or star systems occur, these branches are easily modified to reflect the new systems. For instance, if a binary system with stars A and B encounter a single star C, the tree can go from having A and B being on one node and C being on its own node, to A, B, and C sharing a node as a triple system, and then finally, A and C sharing one node as a binary and B occupying one as a single star.

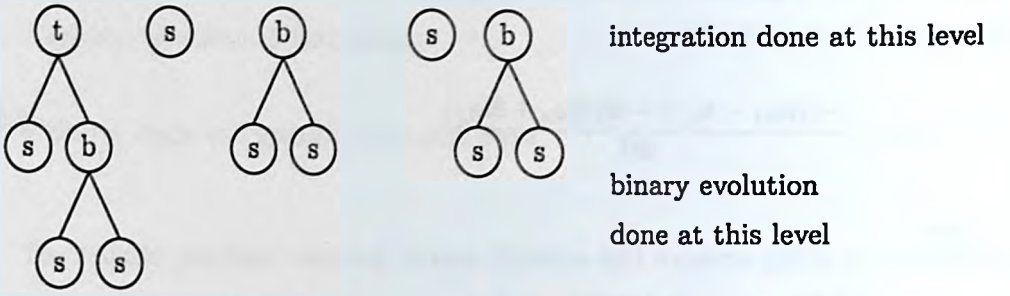


Figure 2.1: The tree structure employed in Starlab.
Circles labeled s, b, and t indicate single, binary, and triple star systems respectively. On the left is a system undergoing a three body exchange (a binary has encountered a single star and is 'trading partners'). The binary system on the right is being perturbed by the nearby single star system.

In addition to the tree structure, Starlab tracks binary systems to detect if other stars are in the neighbourhood to perturb the orbit. If there are not perturbing stars, then the binary orbit is solved analytically by SeBa, but if there are stars nearby, their effect acts to perturb the binary system (even though these other stars are still

on their own node). Perturbed and unperturbed binaries are schematically shown in figure 2.1. Kira attaches a list of nearby perturbers to each binary and updates this list every center of mass step.

Kira gives the option of executing this whole process in the presence of a tidal field. This field will exert a potential upon the cluster and affect the motion of the individual stars. There are a few different tidal field options, but the simulations presented here impose a galactic disk tidal field similar to that of Heggie and Ramamani (1995). The general form for the potential of a tidal field is

$$\phi_{ext} = \frac{1}{2}(\alpha_1 x^2 + \alpha_3 z^2), \tag{2.9}$$

where the line from Galactic center to the cluster center forms the x axis and the z axis represents the axis of rotation for the Galaxy. For a disk field, the equations for the coefficients are given via the Oort constants A and B, with a local Galactic density ρ_g

$$\alpha_1 = -4A(A - B), \tag{2.10}$$

and

$$\alpha_3 = 4\pi G\rho_g + 2(A^2 - B^2). \tag{2.11}$$

Heggie and Ramamani (1995) adopt the values $A = 0.0144$ km s^{-1}, $B = -0.012$ km s^{-1}, and $\rho_g = 0.11$ M$_\odot$ pc^{-3}, and the Jacobi radius (used for stripping escaping stars) takes the form

$$r_j = \left(\frac{-GM_{tot}}{\alpha_1}\right)^{-1/3}. \tag{2.12}$$

The distribution of stars can be given by a number of models. One of the most common models is the King (1966) model. The standard King model can be expressed as:

$$\rho = Ke^{W} \int_0^W e^{-\eta} \eta^{\frac{3}{2}} d\eta, \tag{2.13}$$

where K is some constant, W is a free parameter specified by a central value of the dimensionless depth W_0, and η is a function describing the angular momentum at a certain point which is dependant on the position vector $\vec{r}$. When a tidal field being used, Starlab sets the model up accordingly. The formulation is identical, except that η takes into account an external potential.

The stars on the outer edge of the system may be "stripped" when they get too far away. The stripping radius may be set to any value, but is usually a multiple of the Jacobi radius. Stars will drift this far away from the cluster for three reasons. The first is if there is a tidal field that is imposed on the cluster (usually to represent the effect of the centre of a galaxy), then this may pull outlying stars away from the cluster. Another reason is if the star is has a binary companion that goes supernova, it acquire a velocity kick as a result. Finally, occasionally random motion in the cluster will cause evaporation.

2.1.2 SeBa

SeBa is the collective name for the module that handles stellar and binary evolution. Each star in the cluster is governed by luminosity and temperature relations that depend on the star's mass and age relative to the zero age main sequence. Established formulae can take a star from the main sequence, through the giant branch, and to a compact object (Eggleton et al. 1989). In addition to these two properties, radius, bolometric correction, radius of gyration, mass loss due to stellar wind, and

adiabatic and thermal mass transfer relations are determined for each star. These parameters are used to determine what the star should look like, but also how it behaves in various scenarios. For instance, the adiabatic mass transfer relation will determine how a given star will accrete onto a binary companion, given the opportunity. Stellar properties which are significant dynamically (radius and mass) are updated for kira every 1/64 of a crossing time if they have changed by more than 1%.

Individual stellar evolution is given mainly by the prescriptions found in Eggleton et al. (1989). These recipes assume solar metallicity, and are designed for population I stars. They provide luminosity and temperature for stars in main sequence and post-main sequence stages of evolution. As a given star evolves off the main sequence, it can go through a number of phases, depending mostly on its mass. Prior to this work, SeBa allowed for the following stellar types (and evolved through them more or less as shown here):

proto star. Typically unused, this mostly empty type describes a gravitationally contracting object on the Hayashi track.

planet. Unused type for low mass objects. Initial conditions are usually such that the simulation only describes stellar objects.

brown dwarf. A star with mass below the hydrogen burning limit. Initial conditions normally specify a range of stars that preclude brown dwarfs.

main sequence. A star with a hydrogen burning core.

hypergiant. A post-main sequence star with a mass greater than 25 $M_{\odot}$. A drastic mass loss stage that procedes a Wolf-Rayet star.

Hertzsprung gap. A post-main sequence star that is evolving toward the giant branch.

sub giant. A star with a hydrogen burning shell.

horizontal branch. A star with a helium burning core.

super giant. A star with a two burning shells.

helium star. The helium core of a giant. as a result of stripping of mass by a binary companion.

white dwarf. A low mass stellar remnant.

Thorne-Żytkow. A neutron star with a hydrogen burning shell surrounding it. This can only happen as a result of a merger or collision.

neutron star. An intermediate mass stellar remnant.

black hole. A high mass stellar remnant.

disintegrated. Can be produced via a Type Ia supernova detonation of a white dwarf.

Each of these types have slightly different behaviour and evolves with its own prescription. A star will typically evolve through a number of types until it becomes some sort of compact object (white dwarf, neutron star, or black hole). The limits that determine a star's fate depend on the helium core mass at transition time. For core masses in between 2.2 $M_\odot$ and 5 $M_\odot$ a neutron star is formed, if the core mass is less than 2.2 $M_\odot$, a white dwarf is formed. A black hole is formed for core masses above 5 $M_\odot$. If a star becomes a neutron star, it will receive a velocity kick in a random direction with a magnitude drawn from the distribution of

$$P(u)du = \frac{4}{\pi}\frac{du}{(1+u^2)^2}, \tag{2.14}$$

where $u = v/\sigma$ and $\sigma = 600$ km s^{-1}(Hartman 1997).

Since the main sequence phase can occupy the majority of a star's lifetime, the proper implementation of these parameters is essential to an accurate simulation. Functions expressing luminosity and temperature as a function of mass and time are readily available from Eggleton et al. (1989) (and radius can be obtained from these two). The evaluation of the stellar wind is performed as a simplified version of Schaerer et al. (1996), and is given by

$$\dot{M} = C_w \left[\left(\frac{t_i}{t} \right)^{6.8} - \left(\frac{t_i - dt}{t} \right)^{6.8} \right] M_\odot Myr^{-1}, \tag{2.15}$$

where C_w represents the wind constant, t_i represents the age of this particular evolutionary phase, t represents the age of the cluster, and dt represents the time step over which the wind is calculated. C_w can take on various values depending on the evolutionary state and the mass. For instance, on the main sequence, C_w is given the value of 42 for stars of mass over 85 $M_\odot$, 0 for stars less than 8 $M_\odot$, and 1% of the mass squared for all other masses. On the other hand, when a star is on the asymptotic giant branch, C_w is 80% of the initial envelope mass (Nelemans et al. 2000).

Binary evolution is solved in a few different ways. Unperturbed binaries are evolved analytically as described by Portegies Zwart and Verbunt (1996). This is relatively easy since two body motion is generally quite predictable. This type of motion requires attention if the stars are close enough to produce tidal circularization. The criterion for this is if the stellar radius for either star is greater than 20% of the distance between the star's centers at periastron. The eccentricity, e, of the system will be reduced, but the semi-major axis, a, will be increased to maintain conservation of angular momentum:

$$(1 - e^2{}_1)a_1 = (1 - e^2{}_0)a_0. \tag{2.16}$$

This process will proceed until the orbit is circular or the periastron distance exceeds the stellar radius by a factor of 5.

Perturbed binaries are further divided into strongly and lightly perturbed systems but still experience the same effects of mass loss as their unperturbed counterparts. Strongly perturbed binaries have an external perturbation factor that is greater than ten, and must be resolved into components and solved via integration over each member. Lightly perturbed binaries are slightly more complicated, but faster computationally. Their solution is based on Mikkola and Aarseth (1998), with slight modifications. Mikkola and Aarseth determined that perturbed binaries can be solved in a way similar to perturbed simple harmonic oscillators. This treatment leads to a solution with recursive Taylor series. The method is different because Mikkola and Aarseth employed a regularization scheme to the binary equations of motion, whereas Starlab applies the perturbations directly to the binaries. Small corrections are also applied to the orbits to ensure consistency and low computational costs.

In a two body system, orbital decay can result from a number of processes. Aside from actual contact of the stars, SeBa accounts for two modes of orbital decay. First, angular momentum can be lost via magnetic braking through stellar wind. This loss is applied only when a binary has is circularized or very close to circularization ($e <= 0.001$). Rappaport et al. (1983) give a more thorough treatment of this phenomenon in their population synthesis (that is, binary simulations with no cluster dynamics), and the implementation followed here is a simplified version. The magnetic braking torque is first determined for each star, and then this is translated into angular momentum loss for the binary system. This torque, which applies to main sequence, sub giant, and Hertzpurg gap stars, is given by:

$$\tau_{mb} = -3.8 \times 10^{-30} M R^4_{\odot} \left(\frac{R}{R_{\odot}}\right)^{\gamma} \omega^3 \text{dyn cm}, \tag{2.17}$$

where ω is the angular frequency of the star, and γ is the magnetic braking exponent, set to 2.5 in Starlab. In addition to this, angular momentum can be lost from the system through radiation of gravity waves. The prescription followed is given by Peters (1964), and describes the average decay of eccentricity by:

$$\frac{de}{dt} = -\frac{304}{15} e \frac{G^3 m_p m_s (m_p + m_s)}{c^5 a^4 (1 - e^2)^{5/2}} \left(1 + \frac{121}{304} e^2\right), \tag{2.18}$$

and the semi-major axis a can be determined from this by:

$$a(e) = \frac{c_0 e^{12/19}}{1 - e^2} \left[1 + \frac{121}{304} e^2\right]^{870/2299}. \tag{2.19}$$

c_0 is determined from the values of the previous timestep (or the initial conditions). For large separations this treatment isn't valid, so the binary is first circularized ($e = 0$), and the semi-major axis a is adjusted by:

$$a(t) = (a^4{}_0 - 4\beta t)^{1/4}, \tag{2.20}$$

where

$$\beta = \frac{64}{5} \frac{G^3 m_p m_s (m_p + m_s)}{c^5}. \tag{2.21}$$

Often, the members of a binary system will undergo mass transfer from one star to the other. This will occur usually because of the radius evolution of the individual stars, but dynamic effects can also cause binaries to behave this way. The criterion for this is given by a star's ability to fill its Roche lobe which is given by

$$r_{RL} = \frac{0.49 q^{2/3}}{0.6 q^{2/3} + \ln(1 + q^{1/3})}, \tag{2.22}$$

where q is the ratio of the star's mass to its companion's mass (Eggleton 1983). If a binary star is in a mass transfer phase, the evolution is slowed to ensure that $< 1\%$

of the donor star's mass is lost per timestep. In general, a typical step that a binary would undergo in **SeBa** would involve the following iteration (Portegies Zwart et al. 2001):

1. Reduce angular momentum as a result of magnetic stellar wind.
2. Reduce angular momentum as a result of gravitational wave radiation.
3. Merge the stars if they satisfy the merger criterion.
4. Evolve primary star and correct orbit parameters in light of stellar wind mass loss.
5. Ensure that the binary system is intact. A supernova from one of its stars or a collision would destroy the system's binarity.
6. Evolve secondary star and correct orbit parameters in light of stellar wind mass loss.
7. Ensure that the binary system is intact. A supernova from one of its stars or a collision would destroy the system's binarity.
8. Apply tidal circularization and synchronization as necessary.
9. Determine if either star fills its Roche lobe. If not, return to `Kira`. Otherwise, identify donor and accretor and take subsequent action:
   (a) Determine the time at which mass transfer begins.
   (b) Ensure binary stability. If unstable, apply unstable mass transfer prescription (if this results in a merger), and return to `Kira`.
   (c) Calculate ζ_{ad}, $\zeta_{\rm RL}$, and ζ_{th}.
   (d) Calculate the amount of mass loss from the donor.

(e) Calculate the amount of mass gained to accretor.

(f) Remove the mass for the donor.

(g) Add mass to the accretor, evolve and rejunvenate accretor.

(h) Correct binary orbit parameters.

The instability criterion for unstable mass transfer is simple: if the donor's angular momentum is greater than 1/3 of the binary system's angular momentum, the mass transfer is considered unstable. What happens next is dependent on the stars in the system. If the donor is a main sequence star, the two stars merge because the core/envelope structure is not as defined. If the donor is giant, then any giant in the system ejects its envelope at the expense of the orbital energy. A merger occurs if the donor still fills its Roche lobe after envelope ejection.

For stable mass transfer, it is important to determine which energy transfer mechanism will be dominant and therefore which timescale is significant. The key element to this formulation is the following relation:

$$\zeta_i = \left(\frac{d\ln r}{d\ln m}\right)_i. \tag{2.23}$$

ζ_{th} (for changes in thermal equilibrium) varies between 0 and 0.9 for various masses of main sequence stars, -2 and 15 for Hertzprung gap and horizontal branch stars, and 0 for all other types. ζ_{ad} (for changes due to an adiabatic effect) is set to 2 for main sequence stars with masses greater than 0.7 $M_\odot$, and 4 for all other masses. For other star types ζ_{ad} is assigned to a value based on a polynomial. The other pertinent variable is $\zeta_{\rm RI}$, which describes how the donor's Roche lobe changes. This is calculated by examining how the binary parameters are affected by an infinitesimal mass particle (Portegies Zwart et al. 2001).

The mechanism employed is determined as follows:

$\zeta_{ad} < \zeta_{Rl}$	dynamic mass transfer	timescale: τ_{dyn}
$\zeta_{th} < \zeta_{Rl} < \zeta_{ad}$	thermal transfer	timescale: τ_{th}
$\zeta_{Rl} < \zeta_{ad}, \zeta_{Rl} \leq \zeta_{th}$	nuclear mass transfer	timescale: minimum of τ_{nuc} and τ_J.

The timescales that are given can be expressed:

$$\tau_{dyn} \simeq \sqrt{r^3/m} \cdot 5.1 \times 10^{-11} [Myr], \tag{2.24}$$

$$\tau_{th} \simeq 32m^2/(rL)[Myr], \tag{2.25}$$

$$\tau_{nuc} \simeq 0.1 t_{ms}, \tag{2.26}$$

and

$$\tau_{dyn} \simeq J_{bin}/(\dot{J}_{gr} + \dot{J}_{mb})[Myr], \tag{2.27}$$

where r, m, L, and t_{ms} are the star's radius, mass, luminosity, and main sequence lifetime respectively. $\dot{J}_{gr}$ and $\dot{J}_{mb}$ define angular momentum loss due to gravitational radiation and magnetic braking. In general, it is found that main sequence donors will not employ dynamic mass transfer, while supergiant donors will employ exclusively thermal and dynamic mass transfer.

When a star undergoes accretion from an outside source, the evolutionary state is altered. On one hand, a more massive star will evolve more quickly, but on the other hand, the accretion will add new material for nuclear burning which will make the star seem younger. The treatment of rejuvenation is somewhat simplistic in Starlab, but it is sufficient to produce a fair number of blue stragglers. The star's properties are determined by their relative age (that is, its apparent age independent of when

the cluster was born), t_{rel}. For a star undergoing an accretion of δm, the new relative age can be expressed as:

$$t_{rel}(m+\delta m) = \mathrm{t}_{0..i}(m+\delta m) + \left(\frac{t_{rel} - \mathrm{t}_{0..i}(m)}{\mathrm{t}_j(m)}\right) \mathrm{t}_j(m+\delta m)\mathcal{R}, \tag{2.28}$$

where $\mathrm{t}_{0..i}(m)$ represents the expected time required to evolve through all previous evolutionary states for a star of mass m and $\mathrm{t}_j(m)$ represents the expected time required to evolve through the current evolutionary state for a star of mass m. $\mathcal{R}$ represents the rejuvenation factor. If this is set to unity (which it is if the donated material is not hydrogen), then the accreted star is not rejuvenated beyond changing the relative age to reflect the new, more massive star. Normally, $\mathcal{R}$ is set to a value greater than one:

$$\mathcal{R} = \frac{m+\delta m}{m}. \tag{2.29}$$

2.1.2.1 Collisions/Mergers

One of the major features of using the Starlab environment is that there are prescriptions for mergers or collisions of varying types of stars. Collisions and mergers can be distinguished in that a merger is a result of unperturbed binary evolution, and is preceded by a period of mass transfer. A collision therefore, is an encounter between two stars that are typically not part of a binary system. Currently, the code does not follow collisions that don't result in a merged object, so at the moment when the two stars are replaced by a single star the result will depend only on the types of stars and their mass.

Although SeBa reflects a somewhat simplified view of stellar collisions and mergers, often mergers that occur in the simulations produce blue stragglers. Prescriptions for encounters initially consider which star is the more massive of the two and then follow

a set of rules depending on the types of stars in the encounter. If the secondary (less massive) star is a main sequence star, the rule is that the merger is treated as conservative mass accretion from the less massive secondary to the more massive primary. This results in a rejuvenation of the star which partially 'resets' its age. If this is done with two main sequence stars that are close to the turn off mass, this may be observed as a blue straggler.

In the case of evolved secondary stars, the prescription is slightly different. When the primary is a main sequence star, the product will be a Hertzprung gap star if the secondary is a giant because this will accelerate the star's evolution past the main sequence. If a more massive main sequence star meets a white dwarf, a giant is formed because there will already be helium core. Finally, a Thorne-Żytkow (Thorne and Żytkow 1977) object will be produced in the case of a secondary black hole or neutron star meeting a primary main sequence. Note that these scenarios are most likely to occur if the primary star is a product of a previous merger/rejuvenation and is therefore a blue straggler.

On the other hand, if the primary is a giant star, the product will be giant star as well (with a subsequently larger core mass) for giant and white dwarf secondaries. A neutron star or a black hole secondary that merges with a primary giant star will produce a Thorne-Żytkow object. If the primary is a compact object, most types of stars will form an accretion disc surrounding the primary star. Given enough mass, a white dwarf or neutron star primary could collapse into a neutron star/black hole. In general, accretion onto a neutron star will result in a millisecond pulsar.

2.2 GRAPE-6

Simulations were run employing the tremendous computational power of the GRAPE-6 (Makino et al. 2003). The GRAPE-6 (GRAvity PipelinE) is the 6th generation of special purpose computers designed to perform gravity calculations on a list of objects. It is necessary since the accuracy required for these calculations result in rather short timesteps and high computational demand. While cosmological N-body simulations can be well adapted to parallel computer systems, the nature of a typical star cluster makes this troublesome. The important dynamics of star clusters can be found in a central core that is comparatively smaller than the system. In this core collisions and close interactions become very important. The bottleneck of the simulation is communication latency (Makino et al. 2003). Consequently, the most favourable approach is a dedicated hardware system.

The GRAPE acts as a fully contained 'black box' in that `Kira` sends it a list of particles with their masses and pertinent dynamical information and the GRAPE then outputs the position and velocity assigned to each object. After the initialization of the GRAPE in which all particles with the important data is sent, a single integration would proceed through the following steps (Makino et al. 2003):

1. `Kira` sends a list of particles to be updated.

2. Iterate steps 3-6 for each particle.

3. `Kira` predicts the position and velocity of the particle. This is received by the GRAPE (which calculates the jerk as well), and stored in the force calculation pipeline. The time in the predictor pipeline is reset to the current time.

4. The GRAPE integrates the forces from all other particles. The new position and velocity is calculated in the predictor pipeline.

5. The GRAPE transmits the position and velocity to `Kira`.

6. `Kira` predicts a new timestep.

7. The present system time is updated.

The GRAPE-6 has been able to achieve speeds as high as a few Tflops (Makino et al. 2003), and increases performance greatly for star cluster simulations.

Chapter 3

Pre-Main Sequence Implementation

To implement a pre-main sequence type into an already well-integrated code such as Starlab requires a great deal of care. Instead of revamping the 'Proto Star' type to make it more reflect reality, it was decided that a 'Pre-Main Sequence' type should be inserted in between the 'Proto Star' and 'Main Sequence' types. This ensures the ability to use the 'Proto Star' type to reflect pre-proto stellar cores and other early type objects. The entire pre-main sequence implementation is available as C++ code in Appendix B.

When considering pre-main sequence stars and their properties, there are two main aspects to note. The first is that a pre-main sequence star is almost completely convective. Since there is hardly any energy source other than gravitational contraction, they typically find themselves with very little if any radiative zone. The second is that stars on the pre-main sequence will still possess a composition similar to that of a zero-age main sequence star. That is, there will have been no nuclear reactions (with the exception of a small deuterium burning core) to affect its metallicity.

The most important thing to know about a star at a given time for dynamical simulations is its radius. The radius determines how a star will react when encountering other stars and when to initiate mass transfer in binary systems. The next most important aspect is either the star's temperature or luminosity, as one can be determined from the other if radius is known. Since this data is usually presented in

published stellar evolution tracks, it was necessary to examine contemporary publications of pre-main sequence tracks and find the most suitable.

After a brief search, the evolutionary tracks of Siess et al. (2000) were selected. They give pre-main sequence tracks that begin with a fully convective protostar, and end just after hydrogen burning begins. Their tracks have resolution in mass of 0.1 $M_\odot$ from 0.1 to 2 $M_\odot$ and then transition to a resolution of 1 $M_\odot$ between 4 and 7 $M_\odot$. We used the tracks for a metallicity of $Z = 0.02$ with no convective overshooting. We decided against using the tracks that had convective overshooting for a number of reasons. The first is that the amount of convective overshooting that actually occurs in pre-main sequence stars is uncertain and probably insignificant. The second is that it would add another complication to the system without much gain in realism. The third is that the other stages of evolution already neglect overshoot.

Because the tracks were raw and didn't begin at any particular set time, they were modified to have consistent start and end times. One possible starting point is the deuterium burning limit, which is sometimes referred to as the "pre-main sequence birthline". Deuterium burning typically occurs around a central temperature of 10^6 K, and forms a birthline that travels across the H-R diagram just above the zero-age main sequence. To calculate the birthline, Palla and Stahler (1999) calculate the radius of an object after some mass M accretes on it, and then find the point on its Hayashi track where the stellar radius equals $R_O(M)$. The tracks of Siess et al. (2000) were adjusted to begin at the pre-main sequence birthline and end at the zero-age main sequence (where the main sequence type would take over). Figure 3.1 shows the modified tracks.

These tracks were chosen from a variety of others (D'Antona and Mazzitelli 1997; Palla and Stahler 1999) for a number of reasons. The first is that they have sufficient resolution and span the entire range of masses to which pre-main sequence evolution

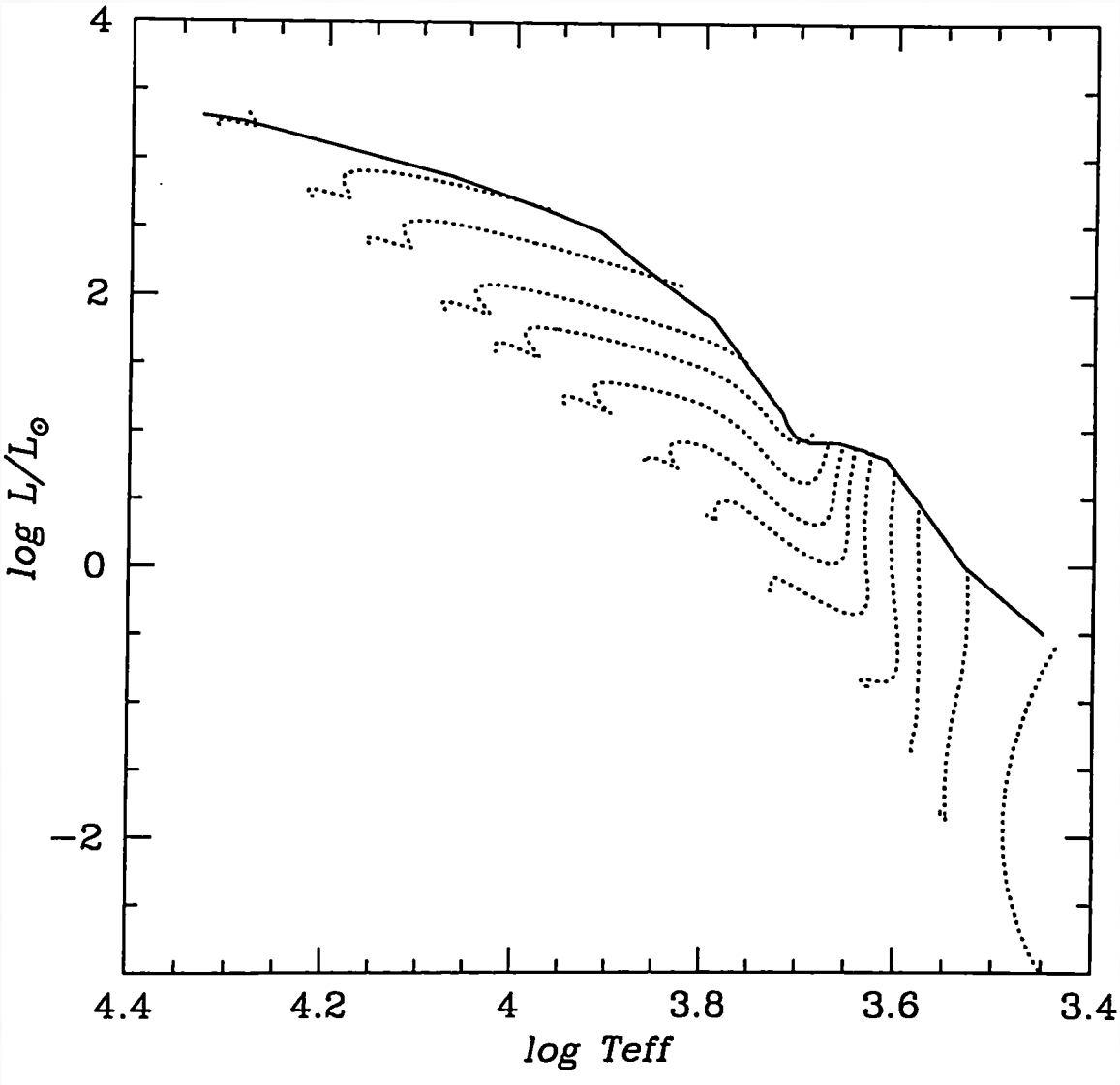


Figure 3.1: Pre-main sequence tracks of Siess et al. (2000), adjusted with the birthline from Palla and Stahler (1999). The masses shown are 0.1, 0.3, 0.5, 0.7, 1.0, 1.3, 1.6, 2.0, 2.5, 3.0, 4.0, 5.0, and 7.0 $M_{\odot}$

applies. Since theoretical tracks from different authors differ amongst themselves it was important to be able to have enough data to infer the properties of the star from a given mass and age. The range of masses is quite fortunate since our initial mass function has a lower limit of 0.1 $M_\odot$, and pre-main sequence evolution isn't very significant at masses much higher than 7 $M_\odot$. Another reason these tracks were chosen is that they had data for stellar radius and temperature that was easily manipulated. Siess et al. (2000) compare their tracks with others and find good correspondence with most other tracks, although there is still some debate concerning the form of very low ($M \leq 0.4 M_\odot$) mass pre-main sequence evolution.

SeBa uses functions to dictate luminosity and temperature for a star with a given age and mass. Initially, functions were fit to the pre-main sequence tracks through age and mass in order to match the method used for other phases of evolution. Unfortunately, pre-main sequence stars can evolve in a complicated manner and unphysical deviations from the tracks were encountered when trying to fit functions to them. These complications arose for many possible function combinations (fourth order polynomial, logarithmic, and exponential). The use of formulae to dictate the crucial stellar quantities was abandoned. Instead, all the tracks were combined to create a lookup table which was interpolated linearly through mass and age to find radius and temperature. Both linear and logarithmic interpolation were analysed, but it was found that logarithmic interpolation was an improvement over linear interpolation in 119 out of 294 test cases. See appendix A for the analysis details.

Unfortunately, beyond temperature, radius, and luminosity not much is known about pre-main sequence stars. For the other quantities we had to look to stars with similar composition (main sequence) or similar structure (giant) to give us these other characteristics. For instance, the way that we chose to calculate the mass transfer and age rejuvenation from accreting binary star systems in Starlab is similar to the

main sequence recipes, but the mass transfer relations (ζ_{th} and ζ_{ad}) are based on the values for a supergiant star. That is, ζ_{th} is zero (disabled), and ζ_{ad} is governed by the following expression:

$$\zeta_{ad} = 57.8109 {f_m}^4 - 75.6863 {f_m}^3 + 32.0344 {f_m}^2 - 2.84699 f_m - 0.220823, \quad (3.1)$$

where $f_m = M_{core}/M_{tot}$, the core mass divided by the total mass (Hjellming and Webbink 1987). The core mass is set to 0.0001 for all pre-main sequence stars. This yields a number very close to zero, which is consistent since pre-main sequence stars should be very diffuse when undergoing binary accretion. It also means that in almost all instances that pre-main sequence stars are undergoing mass transfer, it is happening via dynamical mass transfer.

Similarly, we chose the value for the radius of gyration to be that of a supergiant star, which is approximately 0.44. It was decided that since pre-main sequence stars do not exhibit mass loss (especially since pre-main sequence recipes were only being used for stars with masses less than 7 $M_{\odot}$), the stellar wind would be disabled for this part of the evolution. The rejuvenation (via accretion from a binary companion) formulae are slightly different that those for main sequence stars. The rejuvenation factor is set to one, which means that the age of the accreted star is adjusted to reflect a younger, more massive star. The accreted star will be proportionally as much along the pre-main sequence as the star was before the accretion began. Normally, rejuvenation makes the star young slightly younger than this, but there is no hydrogen burning occurring in a pre-main sequence star, so it doesn't experience the effects of 're-fueling' due to mass transfer the same way a main sequence star does.

An important aspect of the implementation was how to handle encounters involving pre-main sequence stars. Collisions and mergers play a significant role in affecting the appearance of the cluster. For pre-main sequence/pre-main sequence encounters,

the merged object ends up at the pre-main sequence birthline as its evolutionary state is completely disrupted, and its apparent age is set to zero. The treatment of encounters of pre-main sequence stars with more evolved species was similar to the treatment for main sequence stars. That is, if the pre-main sequence star is the secondary, it behaves as if it instantaneously accreted all of its mass onto the primary. If the secondary is a main sequence star, the pre-main sequence star will rejuvenate it. If the secondary is a giant, it will revert to a Hertzprung gap star; if the secondary is a white dwarf, the product will be a giant star, and finally, a Thorne-Żytkow object will form for a neutron star or black hole primary.

A robust implementation of pre-main sequence evolution is critical for the integrity of the simulation. Consistency with current theory should be achieved mainly through the lookup table. While stars can take upwards of 300 Myr to evolve off the pre-main sequence, a large fraction of that time is spent very close to the main sequence. It is expected that the effects will be noticed early in the simulation.

Chapter 4

Simulation Parameters

A total of 9 simulations were evolved for approximately 1.5 Gyr in order to span the range of well-studied open clusters. The nature of each run is outlined in Table 4.1. The runs prefixed by 'pz' indicate that the snapshots used to begin the runs were the the same snapshots used by Portegies Zwart et al. (2001). For the 'pz-pms' runs, the type of the initial stars (with masses of 7 $M_{\odot}$ and less) was simply changed to pre-main sequence stars, thus keeping the initial distribution of the stars and their binary orbits. Since the binary orbits have a minimum separation of Roche lobe contact based on *main sequence* radii, quite a few of the initial binary systems were in contact. In order to consider more realistic parameters, 3 runs (rw-pmsn - 'rw' denoting the author) were begun with identical parameters, but a minimum orbital separation of pre-main sequence Roche lobe contact. For these runs, the binary orbits had to be completely reset. For statistical purposes, most of the results are averaged over a number of runs for each parameter space.

4.1 Initial Conditions

In the interests of simulating observable results, we choose parameters corresponding to population I, young clusters. These clusters typically have approximately 1000 stars, and are no older than a couple of billion years.

The initial cluster is built from the 'ground up' by adding incrementally various levels of sophistication. It begins by laying out 2048 nodes set up with a King (1966)

model with a dimensionless depth (W_0) of 6. This value seems to give reasonable approximations to some observed clusters (Portegies Zwart et al. 2001). The initial distribution takes into account the velocity anisotropy and non-spherical shape that cluster would experience in the Galactic disk tidal field similar to that of Heggie and Ramamani (1995).

The next step is to add a binary companion to every second node (the nodes are numbered randomly), for a total of 3072 stars. This is comparable to a number of clusters including the Hyades, which have binary fractions close to 50% (Perryman et al. 1998).

Masses are applied to the nodes (but only primary stars) using a initial mass function prescribed by Scalo (1986) with masses varying from 0.1 $M_\odot$ to 100 $M_\odot$, and a mean mass of $\langle m \rangle \simeq 0.6 M_\odot$. This initial mass function ξ takes the form:

$$\xi(\log m) = \begin{cases} \phi_{\rm ms}(\log m) T_0 / \int_{T_0-\tau(m)}^{T_0} b(t)dt & \tau(m) < T_0 \\ \phi_{\rm ms}(\log m) & \tau(m) \geq T_0 \end{cases}, \tag{4.1}$$

where $\phi_{\rm ms}$ is the present day mass function of main sequence stars, T_0 is the age of the Galaxy, $\tau(m)$ is the main sequence lifetime of stars with mass m, and $b(t)$ is the relative birthrate defined as the ratio of the present stellar birthrate to the past

Table 4.1: Parameters investigated

Run Name	Starting Point	Number of Stars	Total Initial Mass $M_\odot$
pz-ms1	Main Sequence	3072	1482.88
pz-ms2	Main Sequence	3072	1594.84
pz-ms3	Main Sequence	3072	1606.40
pz-pms1	Pre-Main Sequence	3072	1482.88
pz-pms2	Pre-Main Sequence	3072	1594.84
pz-pms3	Pre-Main Sequence	3072	1606.40
rw-pms1	Pre-Main Sequence	3072	1482.88
rw-pms2	Pre-Main Sequence	3072	1594.84
rw-pms3	Pre-Main Sequence	3072	1606.40

average birthrate. Scalo (1986) gives numerical tables of these functions based on observational data.

This yields an initial total mass of $M_0 \sim 1600 M_\odot$, which is similar to estimates for the Hyades of Weidemann (1993). For the secondary stars, the masses are selected randomly between 0.1 $M_\odot$ and the mass of their primary (Duquennoy and Mayor 1991).

Binary stars require binary orbits. The eccentricities of the orbits are selected from a thermal distribution between 0 and 1 and orbital separation a is selected with a uniform probability in log a. A minimum separation of Roche lobe contact or 1 $R_\odot$ (whichever is larger) and a maximum separation of 10^6 AU is ensured in this process, with the provision discussed earlier in this chapter. These parameter distributions are similar to those observed in the survey performed by Duquennoy and Mayor (1991).

Attributes are then added to the stars based on their mass. Depending on the type of run, radius, temperature, and luminosity were applied for stars starting at the zero age main sequence, or the pre-main sequence birthline.

The simulations were run for about 1000 dynamical time steps with stellar and binary evolution turned 'on'. Escaping stars were stripped away when they reached 2 Jacobi radii from the cluster's centre of mass. Stars were considered merged when they made contact, where the collision/merger recipes would take over. On average, the simulations took about 2 weeks for each run.

Chapter 5

Results

The properties contained within a star cluster involve effects on a number of scales, so it is beneficial to consider both the dynamical (global) properties as well as the stellar and binary (local) properties. The first attempts at adding stellar evolution to dynamical simulations resulted in dramatic effects on both levels. Adding another phase of stellar evolution should have an effect, albeit more subtle. Although most of the results are averaged over 3 runs, it is still important to ensure that any nuances in the data are due to real, physical processes, and not an artifact from statistical or computational weaknesses.

5.1 Global Dynamical Properties

The most obvious two functions that represent the time evolution of star cluster are its total mass and total number of particles versus time. The two processes through which the total number of cluster members can decrease are through the a merger between two stars and through the escape of a star from the system. Stars may escape as a result of a mixture of influences: dynamical encounter with a binary, supernova kick, or capture by the galactic tidal field. Figure 5.1 shows the total number of particles versus time for each set of runs (averaged). That the number of particles in the pz-pms runs have a significant initial drop is notable. This has mainly to do with early mergers that occur within these clusters. In these runs, all of the binaries that had a small orbital separation in the main sequence configuration are

now in contact. As a result, a large number of binaries merge in the first 1.5 Myr of the simulation. Eventually, the evidence of this event is erased, and the number of remaining cluster members approaches that of the other series. The rw-pms runs experience a decreased number of mergers throughout the life of the cluster since the radii of the stars become smaller throughout the pre-main sequence lifetime, and may only reach a point of Roche lobe overflow when one of the stars becomes a giant. As a result, the number of stars for the rw-pms clusters decreases at a lower rate than the pz-pms runs, but identical to the pz-pms runs (without the initial drop).

The time evolution of mass proceeds similarly, except that initial mergers have no impact on total mass since there is no mass loss in this process. There are two processes through which the total mass of the system is decreased. One is through mass loss of individual stars via stellar winds and the other is mass loss through the escape of star systems. Figure 5.2 shows the total mass of the system versus time, for run 2 of each series. The profiles look identical, and the evolution of the total mass is indifferent to the starting point of the stars. One might expect the pz-ms runs to undergo mass loss from the giant branch sooner, but it appears that those effects aren't very significant.

Although it is insightful to look a the cluster as a whole, it is sometimes only possible to see the very core of the cluster when observing it in practice. Figure 5.3 shows how density in the core as well a larger portion of the cluster changes through time. The overall density of the clusters seem to stay the same, but at around 1.2 Myr, the cores of the pz-ms and pz-pms clusters continue to face a decrease in density, but the rw-pms runs stay steady. This may indicate that the core of these clusters can still absorb energy as massive stars fall toward the centre and low mass stars are ejected, and is related to the number and hardness of binaries in the core.

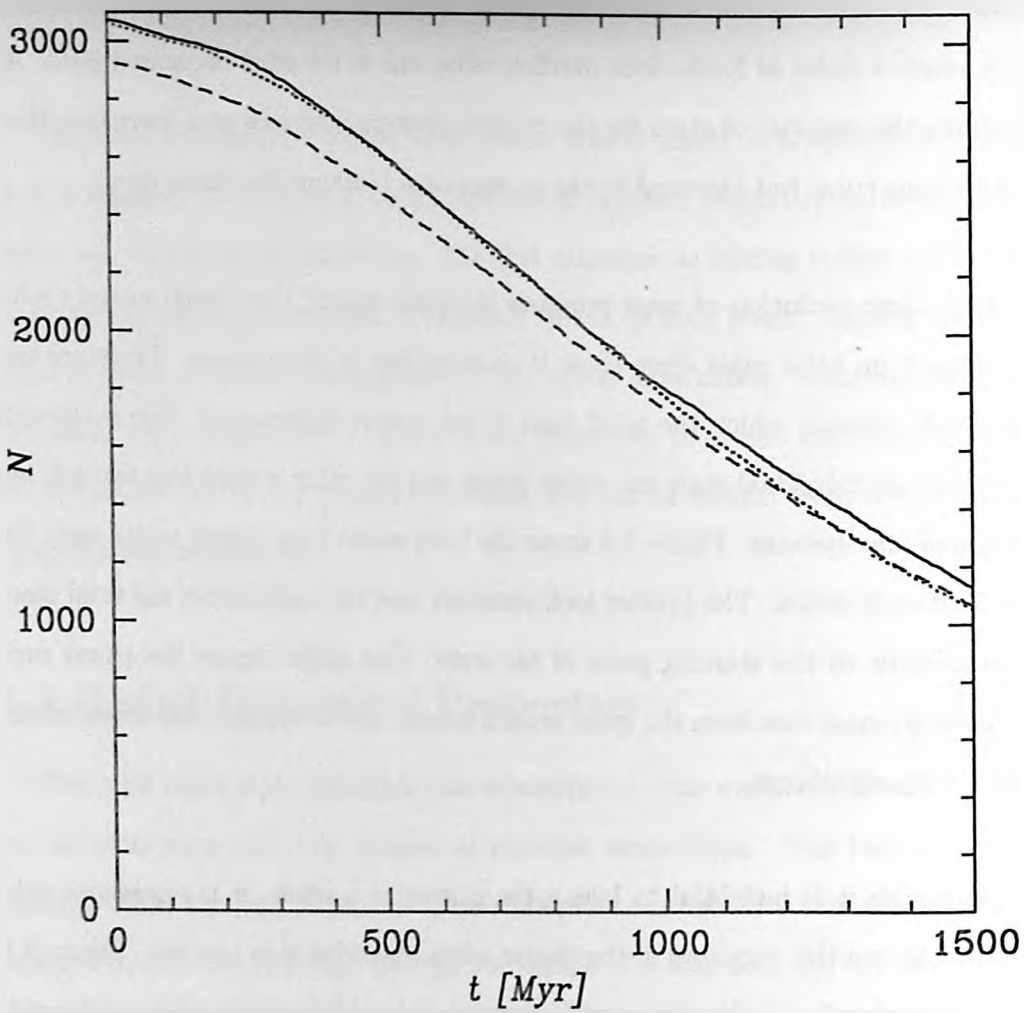


Figure 5.1: Total number of stars versus time averaged over models pz-ms (dotted), pz-pms (dashed), and rw-pms (solid).

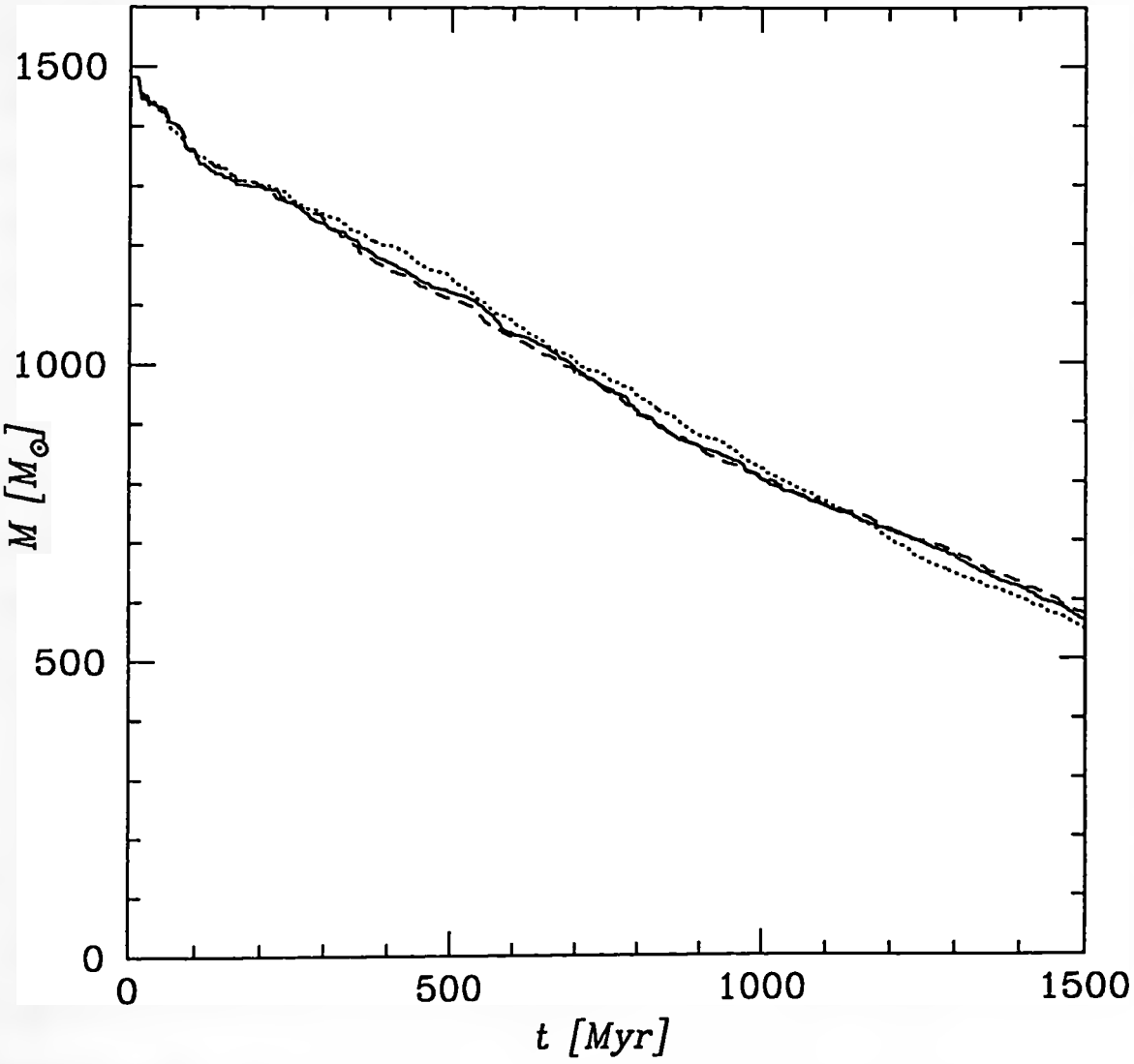


Figure 5.2: Total mass versus time for models pz-ms1 (dotted), pz-pms1 (dashed), and rw-pms1 (solid).

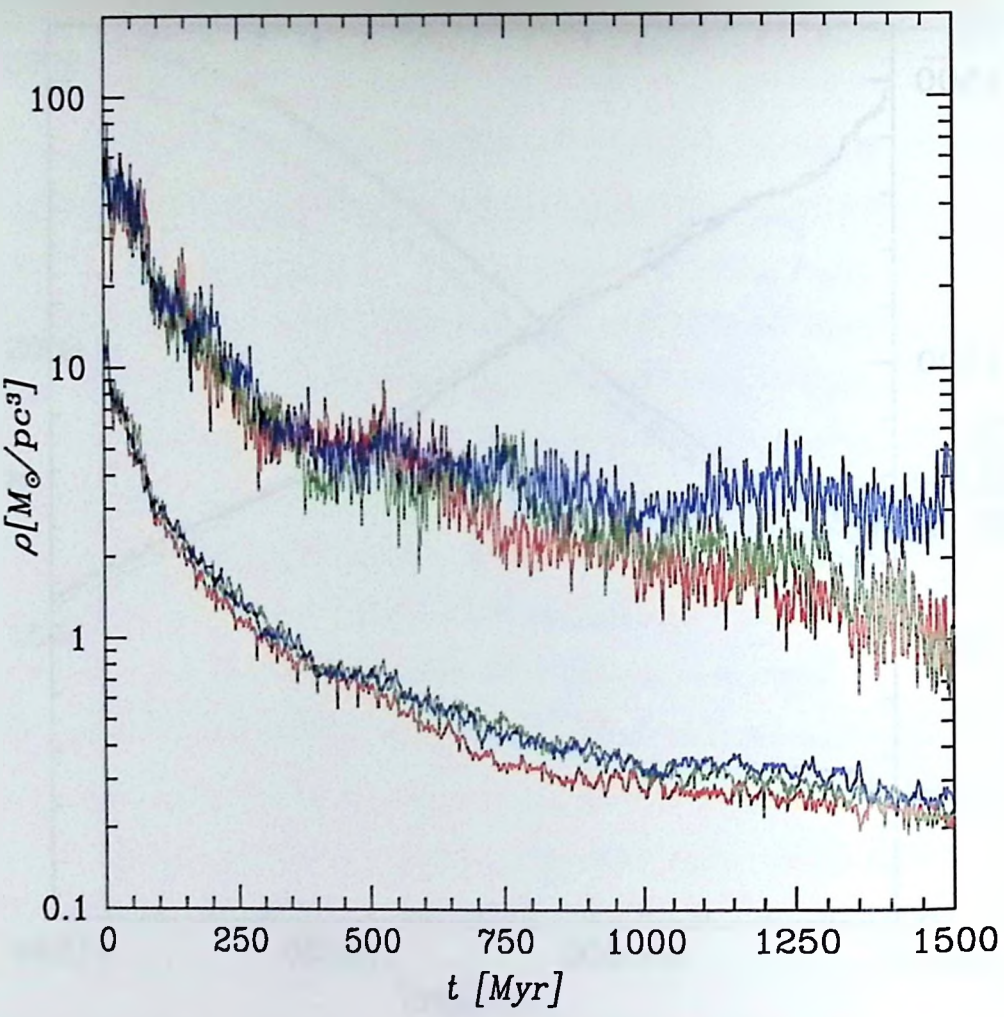


Figure 5.3: Density versus time averaged over models pz-ms (red), pz-pms (green), and rw-pms (blue). The top and bottom lines represent the density within the 10% and 50% Lagrangian radii respectively.

5.1.1 Mass Segregation

While mass segregation occurs in all runs, it is not clear if the starting point of the stellar evolution has much of an impact. The expected cycle that a given cluster will experience is that which the most massive stars fall to the center of the cluster, but as soon as they lose their envelope and become a white dwarf, they end up drifting towards the outside while stars that haven't yet become white dwarfs fall to the center. In a simulation where the stars begin their evolution at the pre-main sequence, this cycle should be offset slightly in time. When evolution of the average mass of the stars contained within various Lagrangian radii is plotted (figure 5.4), the only noticeable difference is in the first 100 Myr. The average mass in the pz-pms and rw-pms runs increase dramatically at first, and then level off to a value slightly above the pz-ms runs. Again, the initial mergers that the pre-main sequence stars experience result in masses higher than normal, yielding a higher average mass.

The value of selected Lagrangian radii would also give an indication of how the mass is distributed through the cluster. Figure 5.5 gives the time evolution of various Lagrangian radii for the average of each type of run. In these figures, the 5% Lagrangian radius tells the story of the core of the cluster. While these clusters are too small to experience core collapse, some interesting fluctuations can be noted. In the first 500 Myr, the core radius of the runs beginning at the pre-main sequence faces a steady increase, somewhat greater than the increase observed in the pz-ms runs. The next period is one of fluctuations for the pz-pms and rw-pms runs, but the pz-ms runs seem rather calm. The last 250 Myr has the core radius of the all the runs increasing.

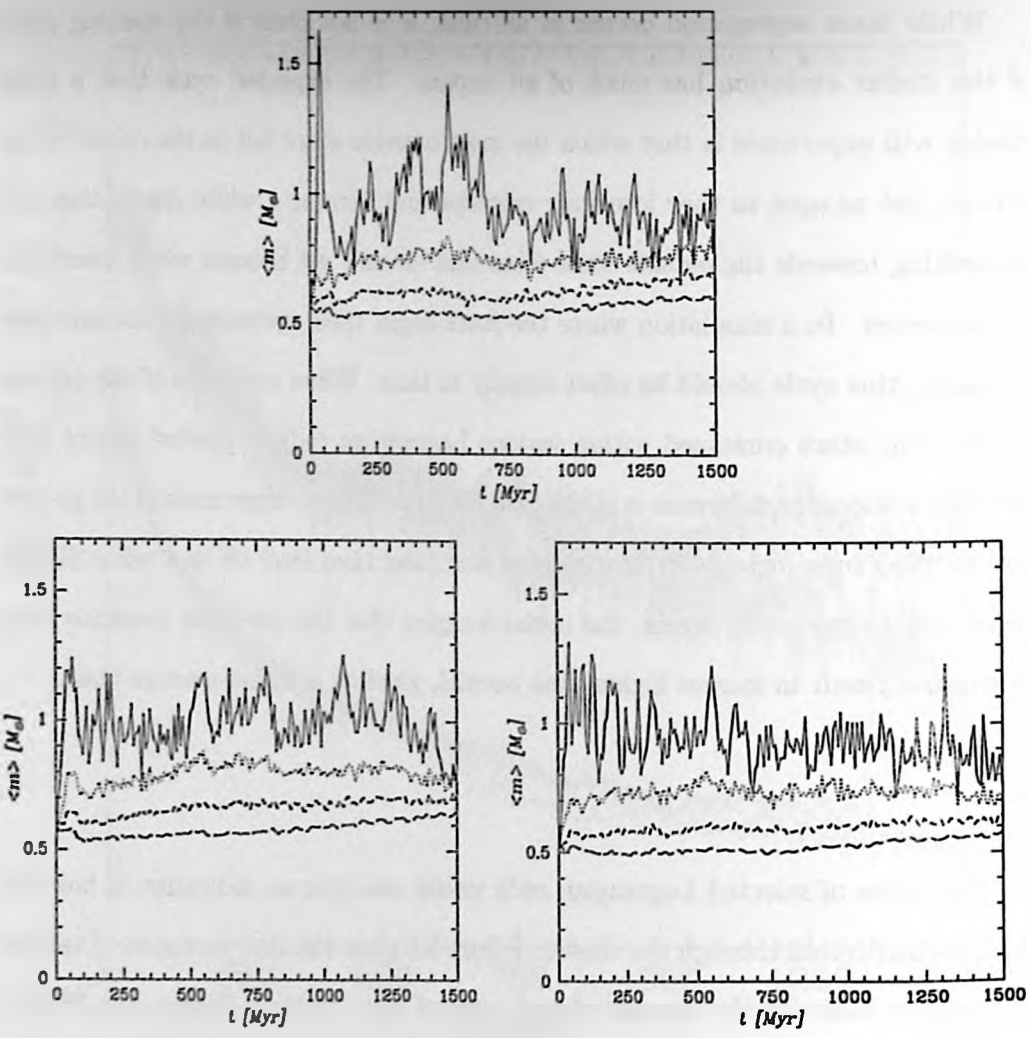


Figure 5.4: Average mass versus time for runs pz-ms1 (top), pz-pms1 (left), and rw-pms1 (right). Shown are the average mass calculated inside the 5% (top line), 25%, 50%, and 75% (bottom line) Lagrangian radii. The values were smoothed over 7.5 Myr intervals.

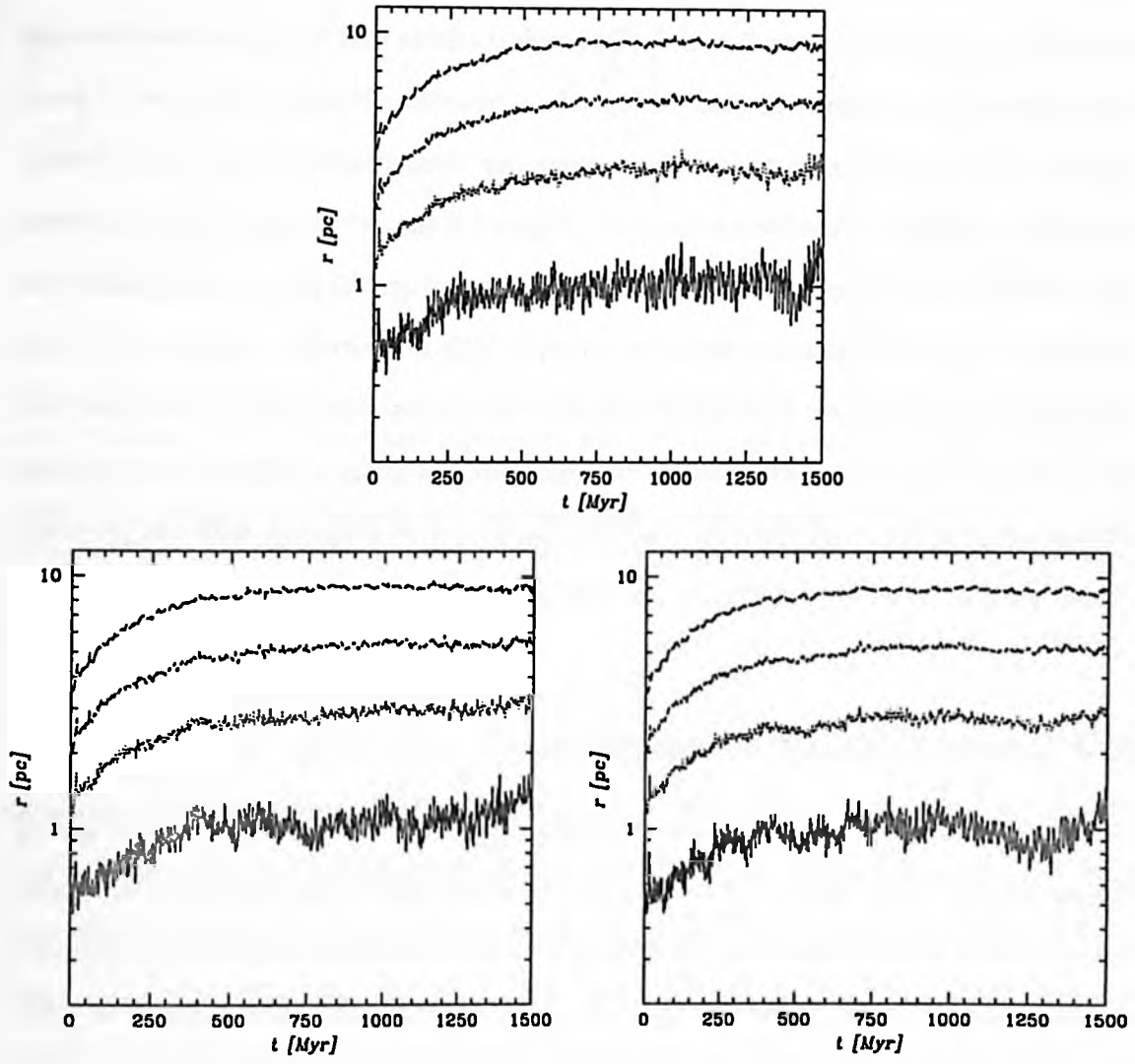


Figure 5.5: Lagrangian radii versus time. Each figure shows the 5%, 25%, 50%, and 75% Lagrangian radii (starting at the bottom and moving up). Shown are those averaged over the pz-ms runs (top), the pz-pms runs (left), and the rw-pms runs (right).

5.1.2 Luminosity/Mass Functions

Mass and luminosity functions aid in determining the initial conditions of star formation. The present-day mass and luminosity functions play a crucial role in determining the initial mass function. Dynamical effects and binary interactions are rarely taken into account in these calculations however. If there are a lot of early mergers in a cluster for instance, the present day mass function may be artificially weighted towards intermediate mass stars. Figure 5.6 shows the evolution of the mass and luminosity functions over time. The run beginning on the pre-main sequence has an initial luminosity function weighted towards high magnitude, and after 600 Myr still does not fill the low luminosity bins like the pz-ms1 run does because the low mass stars still have not completely evolved onto the main sequence. In the mass function diagram, the pz-ms1 and rw-pms1 models evolve similarly, with the pz-pms1 model having fewer low mass stars (mainly due to a high number of early mergers).

5.2 Local Stellar Properties

In addition to the impact that pre-main sequence evolution has on the global properties of a star cluster, it is desirable to know what form various phenomena take on under these conditions. For instance, does the addition of pre-main sequence evolution increase the number of blue stragglers in a cluster? The number of collisions? These questions will gauge the relevance of the difference between the three implementations.

5.2.1 Population Evolution

The stellar populations of the different runs are quite different. Pre-main sequence stars are quite numerous throughout the lifetime of all the clusters starting at the

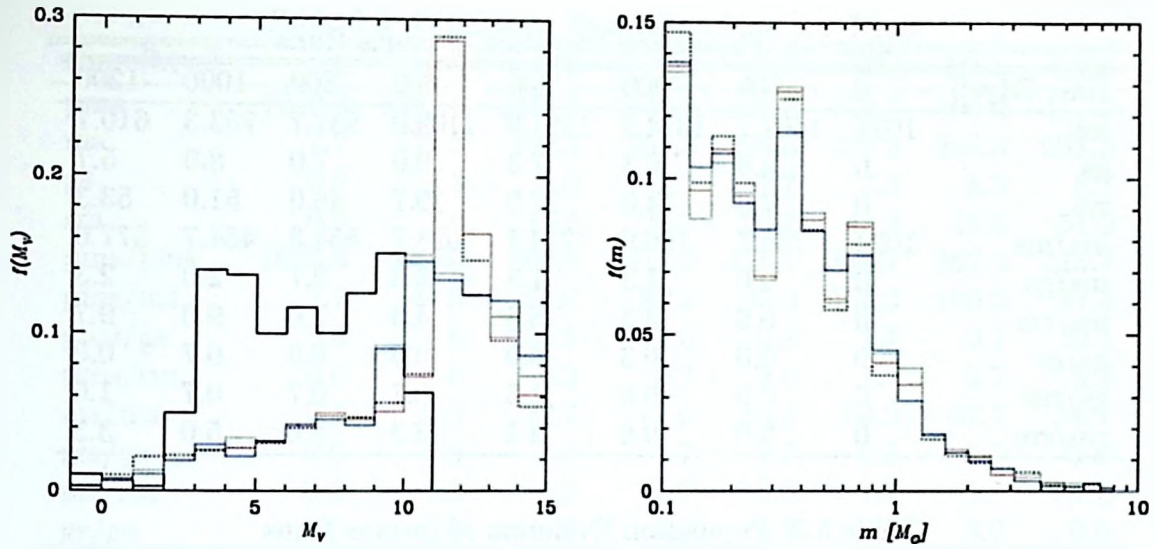


Figure 5.6: Luminosity (left) and mass (right) functions for models pz-ms1, pz-pms1, and rw-pms1. The dotted line represents the initial configuration for the pz-ms run (which is identical to the models starting on the pre-main sequence for the mass function) and the solid black line represents the initial configuration for the pz-pms run and the rw-pms run. Also shown is the data at 600 Myr for pz-ms (red), pz-pms (green), and rw-pms (blue).

pre-main sequence. Aside from that, there is a slight decrease in the number of stellar remnant and giant stars. Tables 5.1, 5.2 and 5.3 show the time evolution of the various populations (single stars and binaries) in the cluster (here and throughout, pms represents pre-main sequence stars, ms represents main sequence, gs represents giant star, and rm represents stellar remnant). Almost all of the features are attributable to either the difference in the number of mergers or the shift in age that the entire population experiences due to starting on the pre-main sequence. The fact that the lower mass stars start their main sequence evolution later results in a non-uniform shift in the population. It is interesting to note that in a lot of instances, the pz-ms runs sit in between the ps-pms and rw-pms runs. For instance, the number of white dwarfs in the pz-ms runs stays steadily between the rw-pms and the ps-pms runs. The reason that there are so many white dwarfs in the pz-pms runs with respect to

Table 5.1: Population Evolution of pz-ms Runs

time [Myr]:	0	100	200	400	600	800	1000	1200
ms	1024	1407.7	1374.3	1251.0	1109.0	931.7	763.3	610.7
gs	0	4.0	6.3	7.3	9.0	7.0	8.0	5.7
rm	0	7.0	14.0	27.0	39.7	48.0	51.0	53.3
ms/ms	1024	788.3	766.0	714.7	628.7	554.3	464.7	377.0
ms/gs	0	0.6	1.3	1.3	3.3	2.7	2.0	2.3
ms/rm	0	0.6	3.3	5.0	4.3	7.0	9.0	9.7
gs/gs	0	0.0	0.3	0.0	0.0	0.0	0.7	0.0
gs/rm	0	0.0	0.6	1.3	0.7	0.7	0.7	1.0
rm/rm	0	0.0	0.6	3.3	3.3	4.0	5.0	3.3

Table 5.2: Population Evolution of pz-pms Runs

time [Myr]:	0	100	200	400	600	800	1000	1200
pms	1020.3	1395.3	1274.7	1071.0	868.7	684.7	524.3	379.7
ms	3.7	144.7	219.0	288.3	320.0	333.3	320.3	302.3
gs	0.0	3.7	5.7	6.3	7.7	8.3	8.7	7.0
rm	0.0	7.0	15.3	31.3	43.3	53.0	58.3	60.7
pms/pms	1020.6	600.7	553.3	473.3	392.7	322.3	258.3	200.3
pms/ms	2.7	34.7	56.0	74.3	86.0	85.3	85.0	75.7
pms/gs	0.0	0.3	0.0	0.3	0.0	1.0	1.0	0.3
pms/rm	0.0	0.0	0.3	0.3	0.3	0.3	1.0	1.7
ms/ms	0.7	23.7	28.7	45.0	55.0	59.3	61.7	62.0
ms/gs	0.0	0.3	1.7	0.7	2.7	1.0	0.7	1.3
ms/rm	0.0	0.0	1.7	3.0	4.0	5.3	5.7	5.7
gs/gs	0.0	0.3	0.0	0.0	0.3	0.0	0.0	0.0
gs/rm	0.0	0.0	0.7	0.7	0.3	0.7	0.3	0.3
rm/rm	0.0	0.0	1.7	2.0	1.7	2.0	1.7	1.3

the rw-pms runs is that the merger products from early in a ps-pms cluster's lifetime will have a high mass, and thus tend to move quickly to the remnant stage.

In general, these populations end up being similarly distributed throughout the cluster. Figure 5.7 shows the distribution of various classes of stars at 600 Myr the first run of each of the series. At this point in time, the cluster has relaxed quite a bit and is cycling its core population by having the high mass giant stars fall into the centre, lose their mass as they become white dwarfs, and then drift out to the outer parts of the cluster. The low mass stars (main sequence in pz-ms, and pre-

Table 5.3: Population Evolution of rw-pms Runs

time [Myr]:	0	100	200	400	600	800	1000	1200
pms	1020.3	1450.0	1346.7	1148.3	936.3	735.7	566.0	471.7
ms	3.7	127.3	192.3	255.3	296.3	301.3	284.3	291.3
gs	0.0	4.3	5.0	3.7	7.7	7.3	8.3	8.7
rm	0.0	7.0	15.3	28.7	35.7	43.3	48.3	51.0
pms/pms	1020.6	643.0	600.0	512.3	418.7	334.7	262.3	226.7
pms/ms	2.7	44.3	61.0	87.3	95.0	104.3	100.3	97.3
pms/gs	0.0	0.0	0.3	1.0	0.3	1.7	0.7	0.3
pms/rm	0.0	0.0	0.3	0.7	2.0	1.7	2.7	3.3
ms/ms	0.7	22.3	32.7	47.0	56.3	64.3	67.7	66.7
ms/gs	0.0	0.0	2.3	1.7	2.3	0.3	1.0	1.0
ms/rm	0.0	1.3	1.3	4.7	6.0	7.7	6.7	6.3
gs/gs	0.0	0.0	0.0	0.0	0.3	0.0	0.0	0.0
gs/rm	0.0	0.0	0.0	0.3	1.3	0.3	0.3	1.0
rm/rm	0.0	0.3	1.0	2.3	3.0	4.7	5.0	4.0

main sequence in pz-pms and rw-pms) tend to have the flattest distribution, while the giant stars are almost always concentrated in the core. The binary stars sit with the intermediate mass stars, as would be expected since their total masses are close to that range.

The time evolution of the binary orbit parameters is as expected. Figures 5.8 and 5.9 show the binary orbits of all stars at the start of the simulation and at 600 Myr for runs using the original binary orbits and using the more realistic binary orbits respectively. For the pz-pms model it is evident that almost all of the binaries with a low orbital period have merged or circularized by 600 Myr. For the rw-pms model almost no circularization occurs. This is again attributable to the fact that the stars are contracting and are therefore rarely in a period of mass transfer. This is encouraging since a quick note of observational data (Duquennoy and Mayor 1991) indicates that fully circularized binaries are not very common. It is also noticeable that all of the softer binaries (those with orbital periods larger than 10^4 yr) have been broken up in all models by 600 Myr.

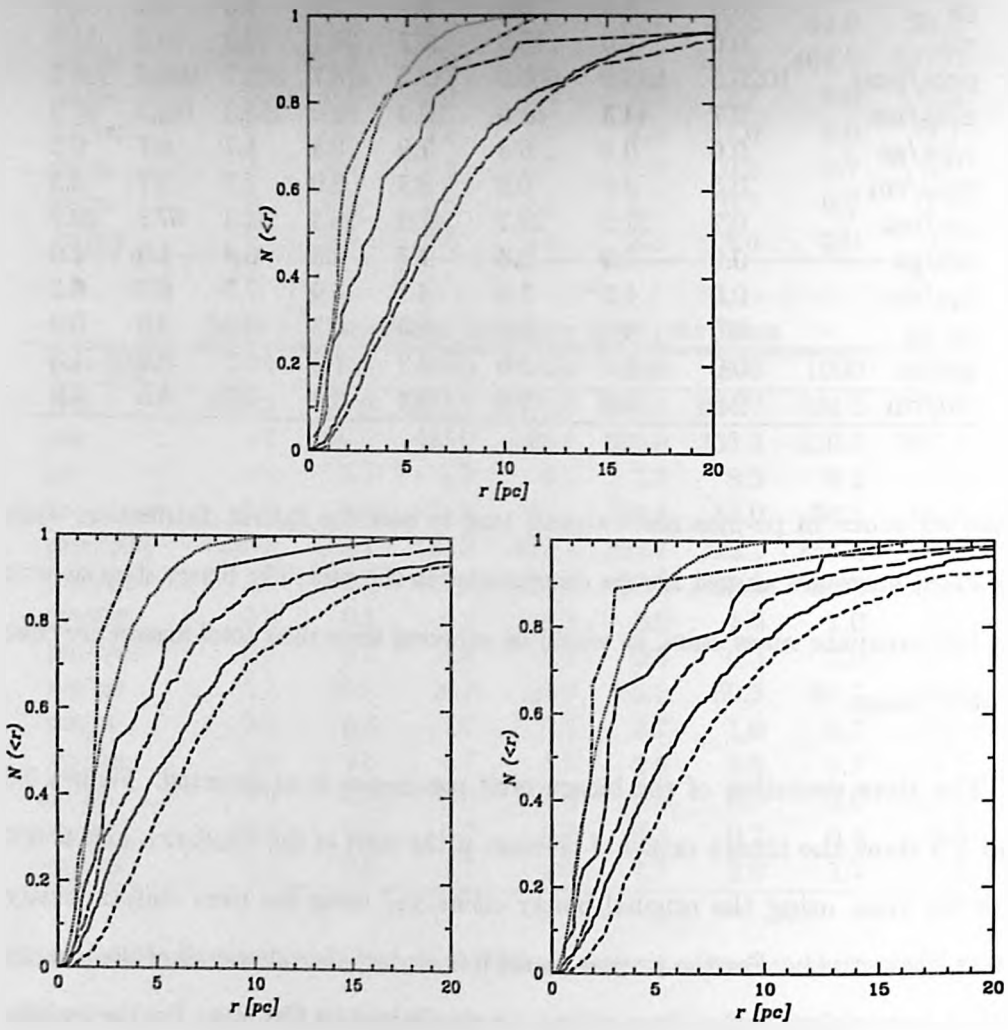


Figure 5.7: Population distribution for runs pz-ms1 (top), pz-pms1 (left), and rw-pms1 (right) at 600 Myr. The dotted line represents the initial distribution of all objects (a King model). Also shown are the binary stars (solid), pre-main sequence stars (short dashed), main sequence stars (long dashed), giant stars (dotted-short dashed), and remnants (dotted-long dashed).

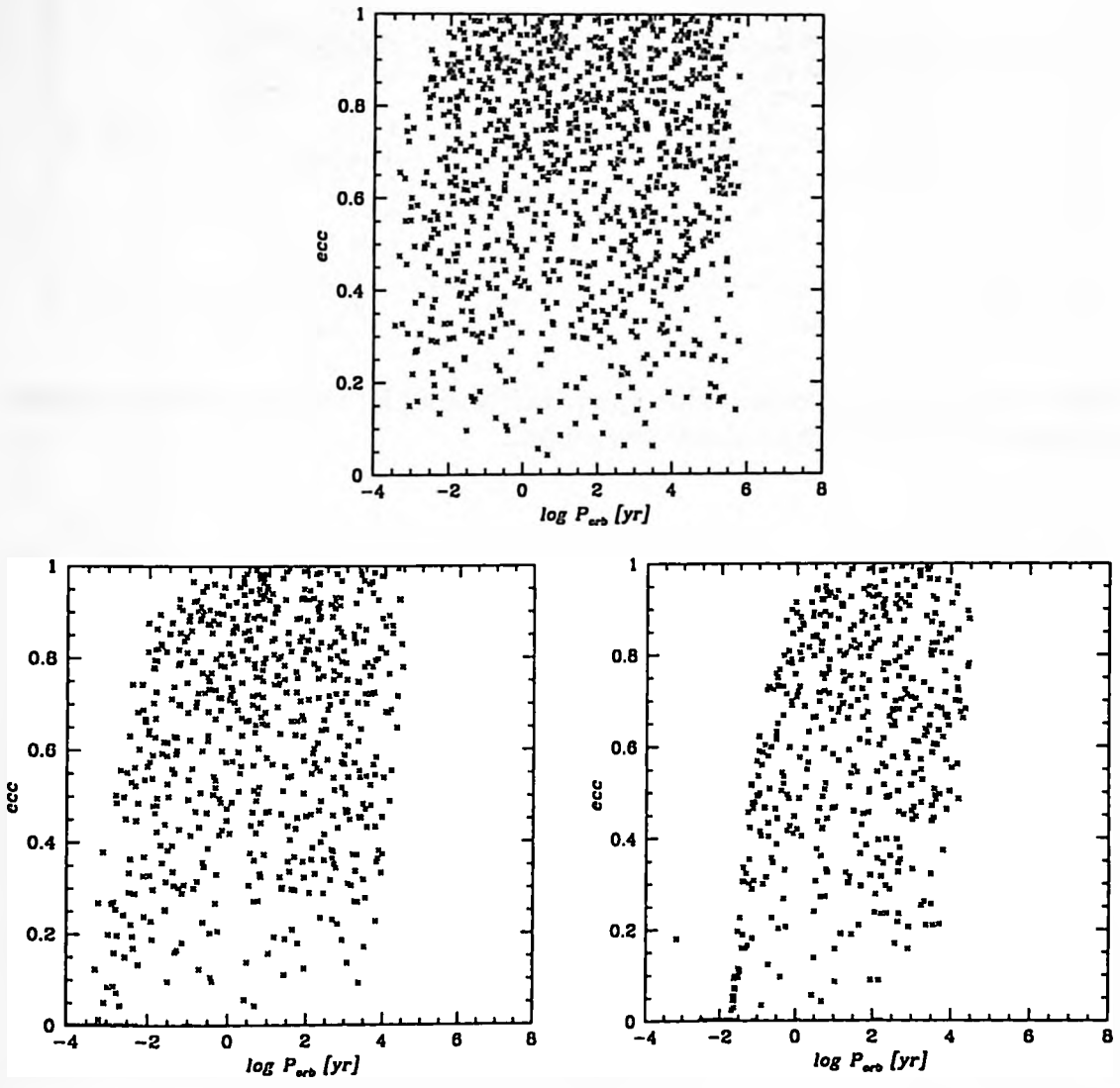


Figure 5.8: Eccentricity versus orbital period of binaries for pz-ms1 and pz-pms1. Shown are the initial parameters shared by both runs (top), and the parameters at 600 Myr for pz-ms1 (left) and pz-pms1 (right).

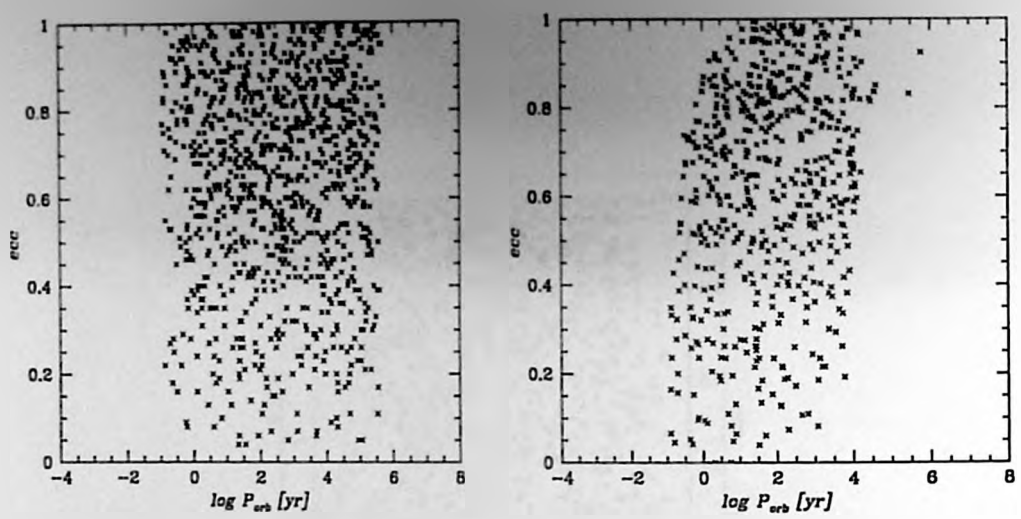


Figure 5.9: Eccentricity versus orbital period of binaries for rw-pms1. Shown are the parameters at 0 Myr (left) and 600 Myr (right).

Throughout the pz-pms and rw-pms runs, the binary fraction stays dramatically below the pz-ms runs, as seen in figure 5.10. The initial dip for the pz-pms runs is a result of the large number of mergers that occur immediately after the simulation begins. For the rw-pms runs, the binary orbits were selected such that there were a higher concentration of soft binaries. This lead to a lot of multiple systems breaking up into single stars. The binary fraction stays at a low level for the runs that begin at the pre-main sequence until about 800 Myr when it begins to creep up again. This is around the time at which a new crop of massive stars fall into the center of the cluster, and their concentration leads to the formation of binary star systems. In the case of the pz-pms runs, the binary fraction continues to increase through time and might possibly meet the pz-ms value at some point. It is possible that there is an equilibrium binarity of this configuration. If so, it seems that at a binary fraction of approximately 0.37, the number of binaries that are dynamically created is equal to the number of binary stars that are destroyed through mergers and dynamics.

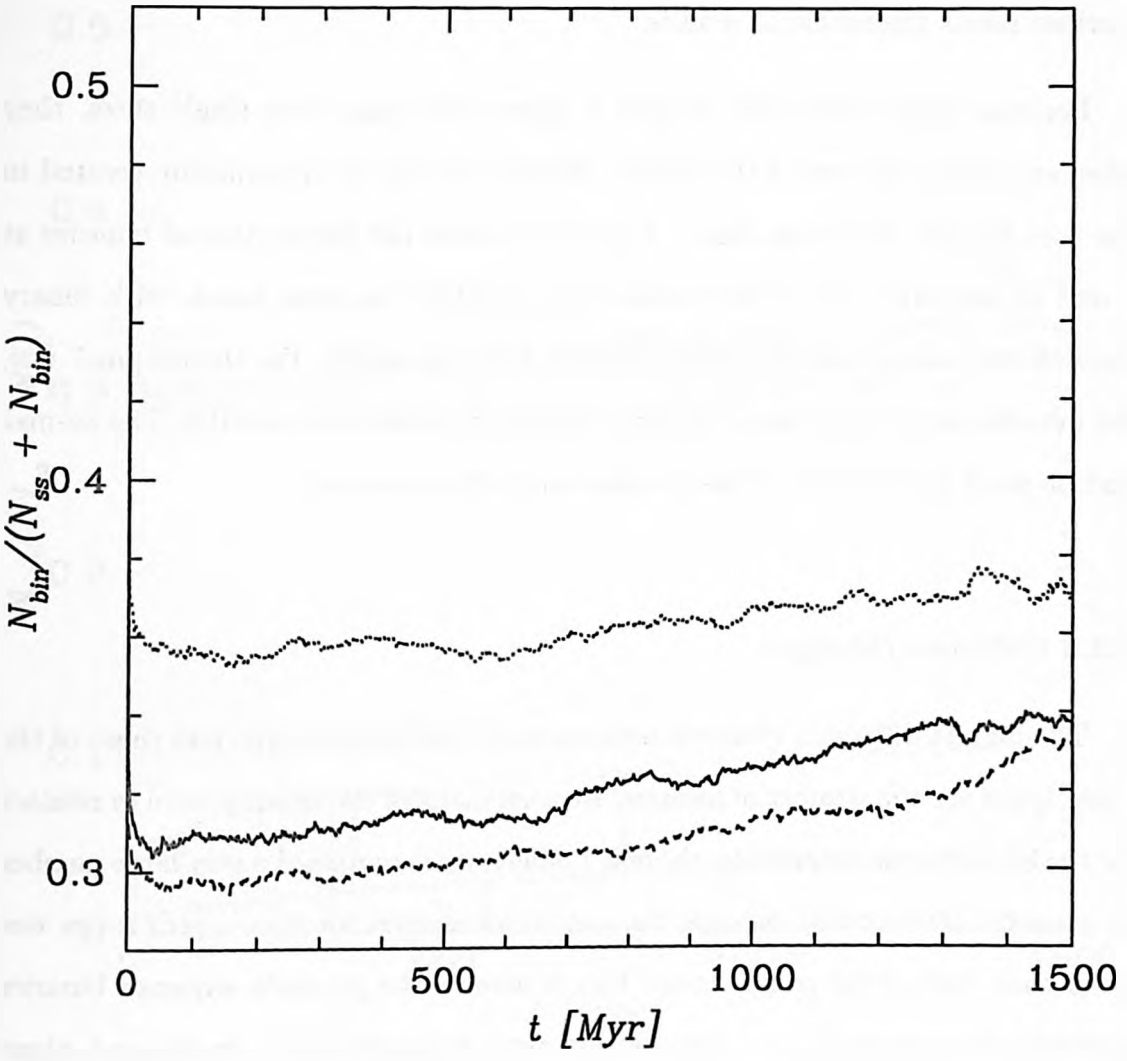


Figure 5.10: Binary fraction versus time for all binaries. Shown are pz-ms (dotted), pz-pms (dashed), and rw-pms (solid) averaged over all runs.

The hard binaries follow a slightly different trend as shown by figure 5.11. The number of hard binaries increases steadily over the entire lifetime of the cluster. For pz-pms and rw-pms runs, the hard binarity is significantly lower that their pz-ms counterpart due to early mergers and initial conditions respectively. The rate of increase seems similar for all models.

Because binary stars tend to have a higher total mass than single stars, they often end up in the core of the cluster. Binaries can also be dynamically created in the core because it is more dense. Figure 5.12 shows the distribution of binaries at 0 and at 600 Myr. All of the models seem to follow the same trend, with binary fraction decreasing with decreasing distance from the centre. For the rw-pms1 run, the binaries seem to dominate the core, with binary fractions of over 0.8. The pz-ms1 and pz-pms1 models seem to be have similar profiles, however.

5.2.2 Collisions/Mergers

The biggest difference observed between runs of the pz-pms type and those of the other types are the number of mergers. Because many of the binaries were in contact for the initial pz-pms snapshots, the first 1 million years contained a very large number of mergers. Interestingly enough, the number of mergers for the rw-pms series was lower than that of the pz-ms series. This is because the pre-main sequence binaries started with large radii and then contract away from each other. In the end, these binaries require more of a perturbation in order to merge. The differences between the models are showed in figure 5.13.

The most common form of merger in the pz-pms runs is the pre-main sequence/pre-main sequence merger. Since these mergers occurred so soon, their impact on the number of blue stragglers isn't the same as the mergers in the pz-ms runs. The early

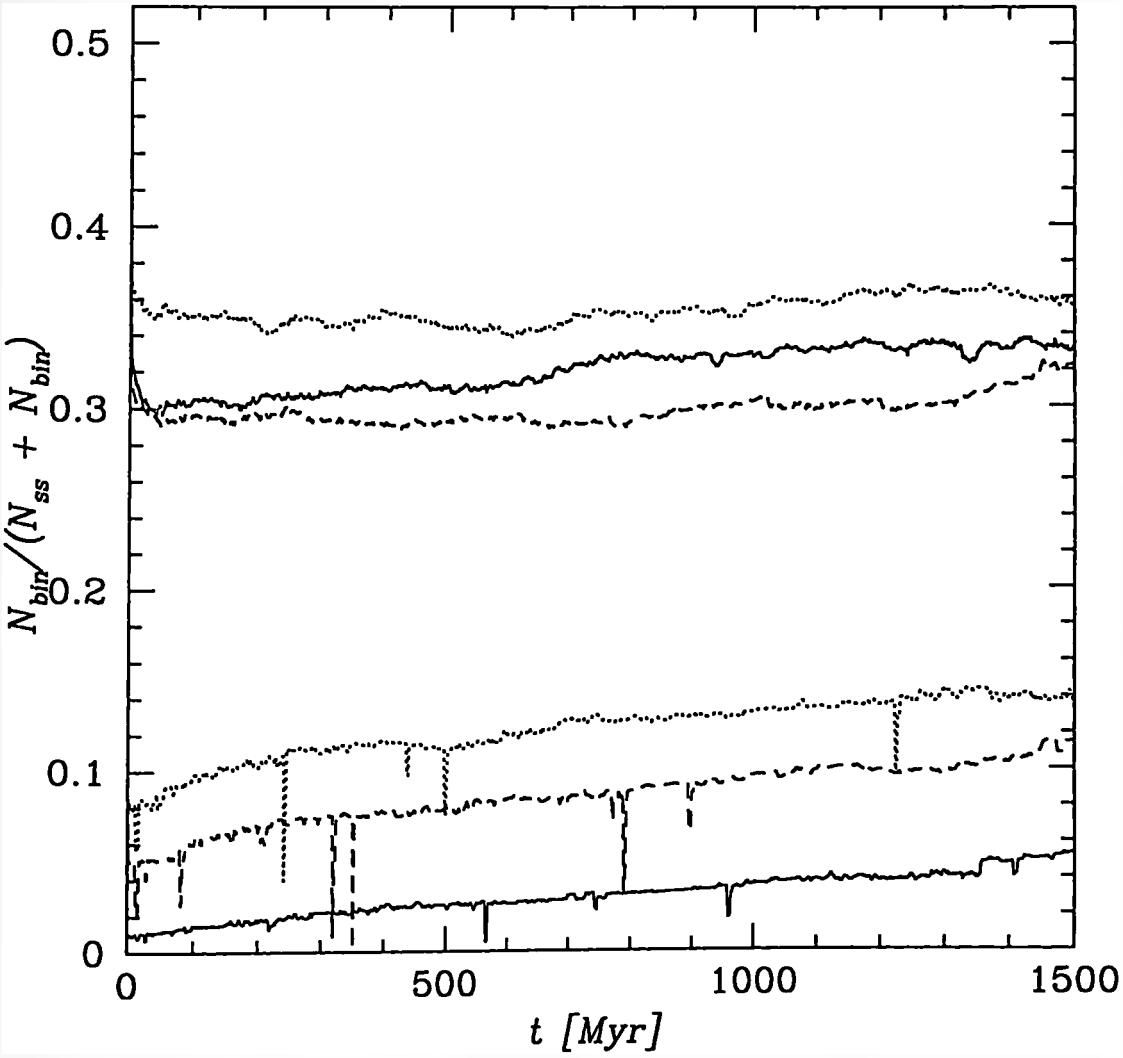


Figure 5.11: Binary fraction versus time for hard ($E < 1000kT$) binaries. Shown are the values for pz-ms1 (dotted), pz-pms1 (dashed), and rw-pms1 (solid). The total binary fractions are given as a reference using the same legend.

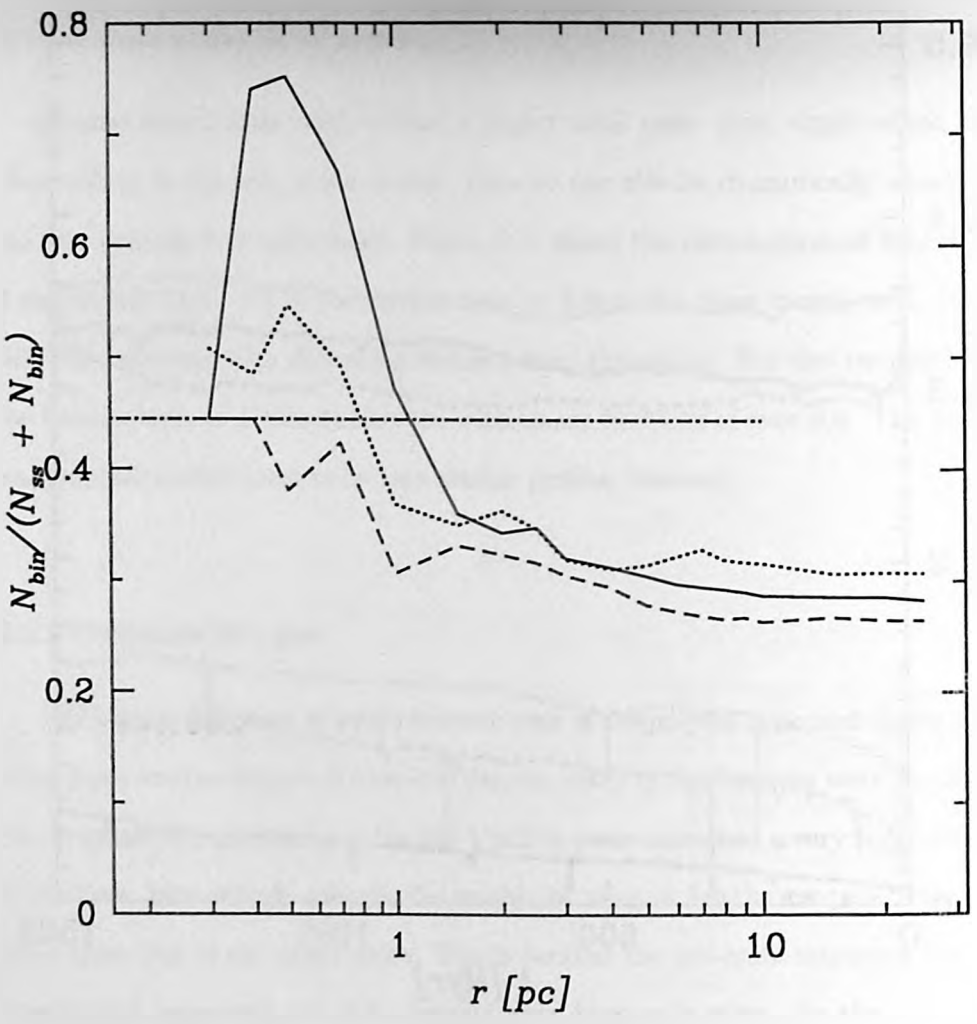


Figure 5.12: Binary fraction versus radius initially (dotted-dashed line), and at 600 Myr. Shown are the values for pz-ms1 (dotted), pz-pms1 (dashed), and rw-pms1 (solid).

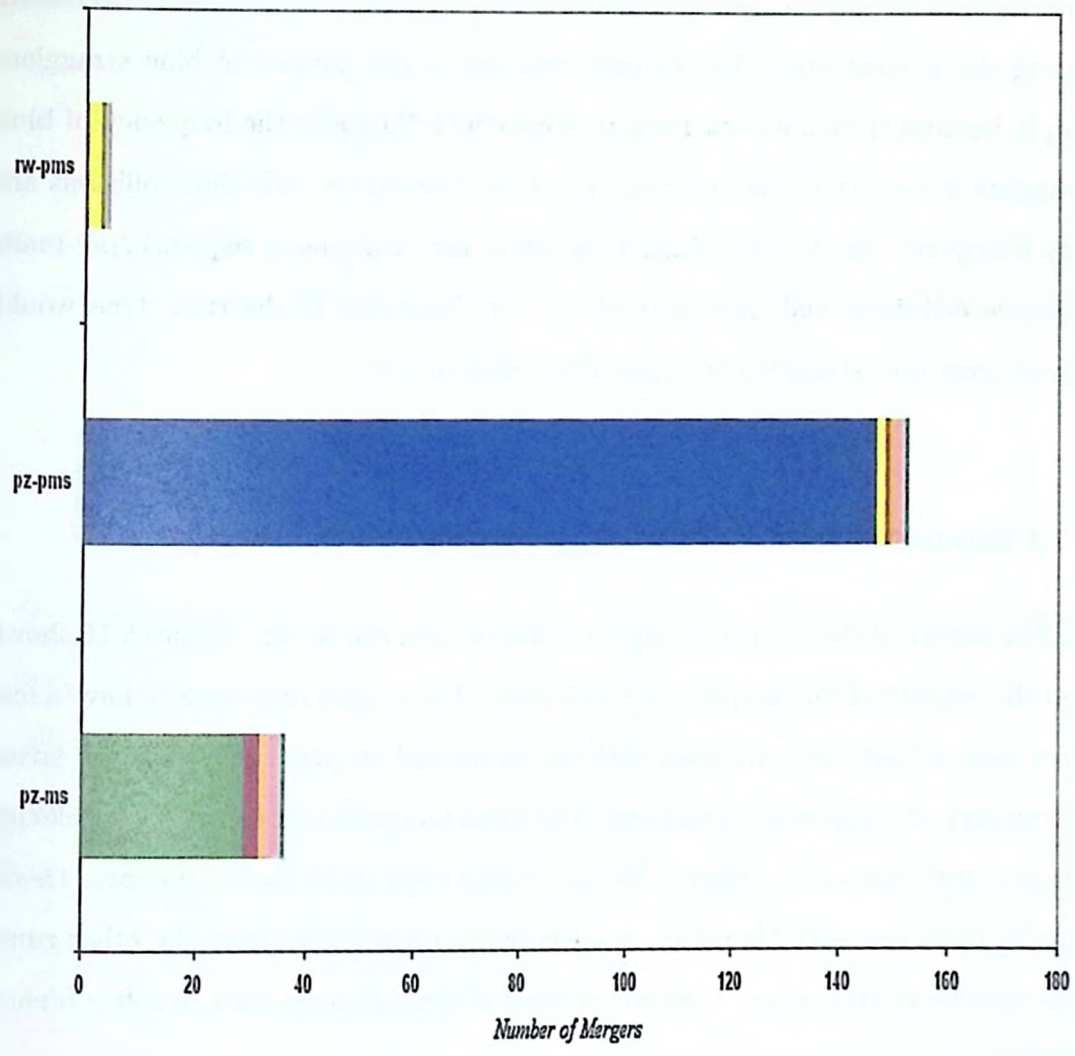


Figure 5.13: Types of mergers in all models. The legend is as follows: blue - pre-main sequence/pre-main sequence mergers, red - pre-main sequence/main sequence mergers, yellow - pre-main sequence/giant mergers, green - main sequence/main sequence mergers, violet - main sequence/giant star mergers, orange - main sequence/remnant mergers, grey - giant star/giant star mergers, pink - giant star/remnant, light blue - remnant/remnant.

merged pre-main sequence binaries do not become blue stragglers because their adjusted age is so close to the real age of the cluster. Also, because so many of the close binaries have already merged, the number of blue stragglers is significantly decreased among the pz-pms runs. The rw-pms runs have a low number of blue stragglers simply because there are fewer mergers. Figure 5.14 illustrates the frequency of blue stragglers in each run. The configuration of these clusters is such that collisions are very infrequent. In all of the simulations, there were 2 pre-main sequence/pre-main sequence collisions, and these were at the very beginning of the run. One would expect more blue stragglers in a more dense environment.

5.2.3 Escapers

The nature of the escapers is slightly different from run to run. Figure 5.15 shows how the velocity of the escapers vary with mass. The pz-pms runs seem to have a few more high velocity escapers than both the pz-ms and rw-pms runs, with the latter having very few high velocity escapers. The trend is opposite for the medium velocity escapers with masses less than 1 $M_{\odot}$, as pz-pms seem to be void of escapers there. Finally, there are more high mass escapers in the rw-pms runs than the other runs. This may be an indication of the fate of some of the high mass stars in softer binary systems.

Mass can leave the system through escapers as well as during stellar evolution. Table 5.4 summarises which modes of mass loss and number loss are experienced in each of the models. There is slightly more mass lost through escapers in the rw-pms runs even though these runs have fewer escapers than the pz-pms runs. This is probably because stars have have a greater likelihood of being in a wide binary orbit (and thus able to disassociate) in the rw-pms runs. A rather surprising result

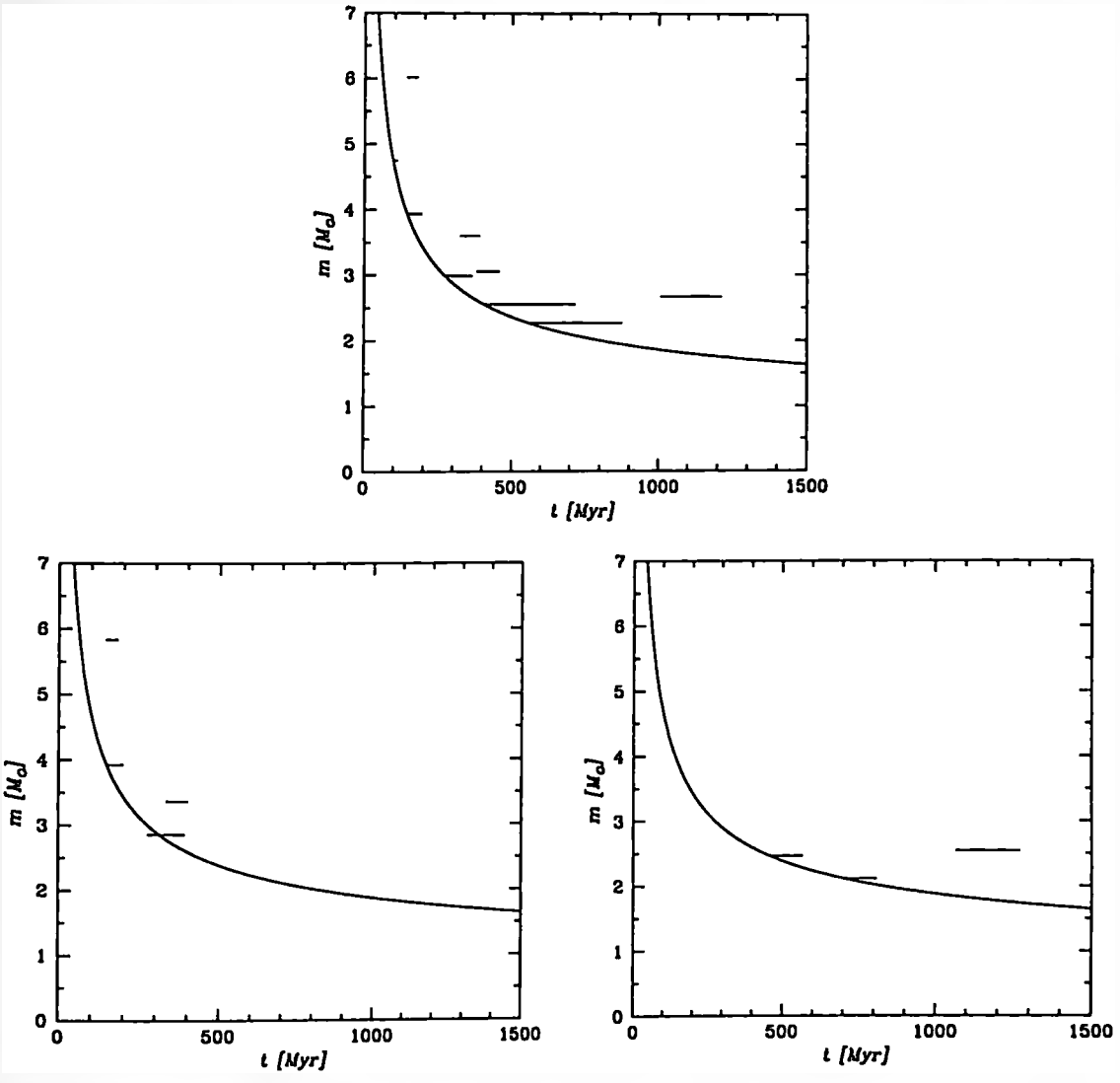


Figure 5.14: Blue stragglers with respect to the turn-off mass. Each horizontal line represents a blue straggler that exists for a given period of time. Shown are those from the pz-ms2 runs (top), the pz-pms2 runs (left), and the rw-pms2 runs (right).

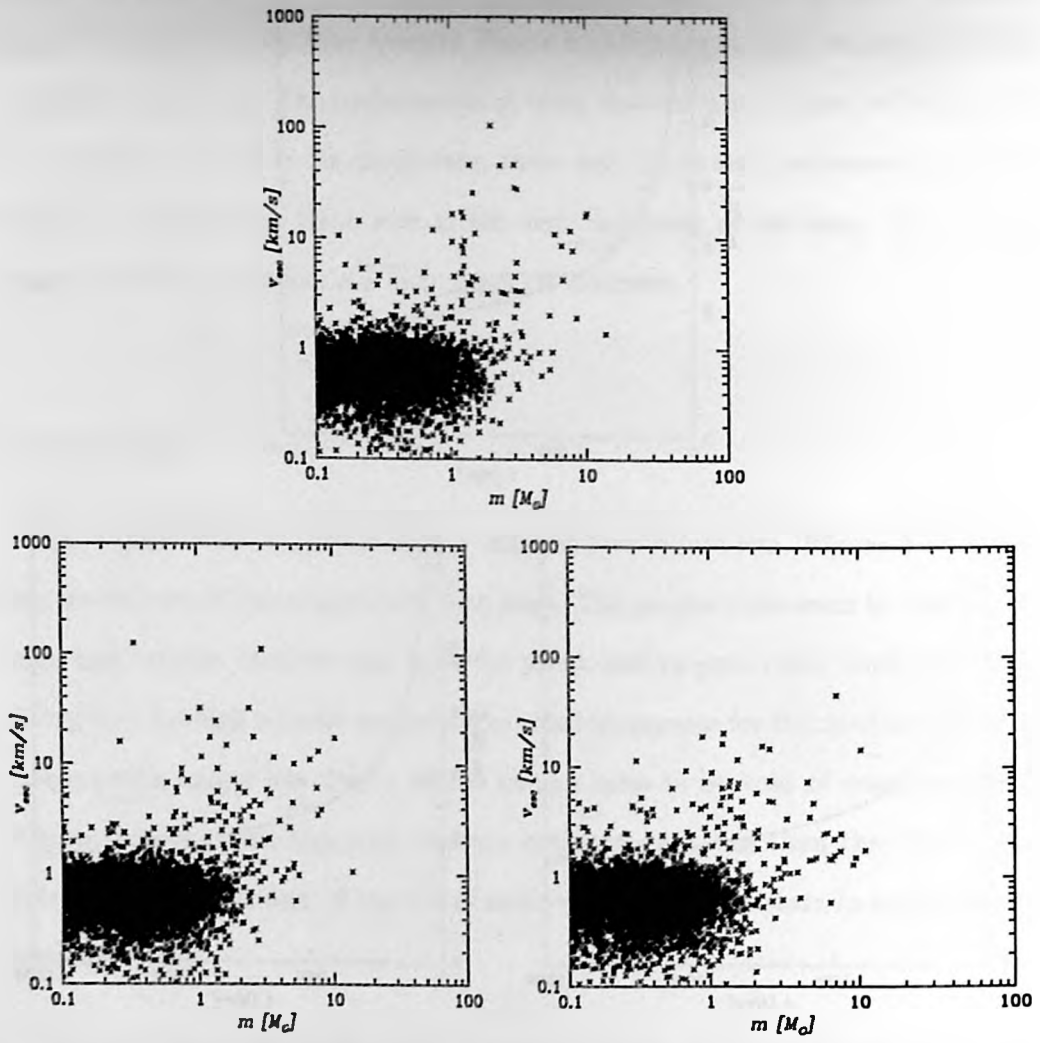


Figure 5.15: Mass versus velocity of all escapers. Shown are those from the pz-ms runs (top), the pz-pms runs (left), and the rw-pms runs (right).

Table 5.4: Number Loss and Mass Loss as of 1.5 Gyr for all runs

	pz-ms	pz-pms	rw-pms
% objects that have escaped	51.1	49.5	50.1
% objects that have merged	3.1	14.2	0.4
% mass lost via escapers	47.5	46.6	47.8
% mass lost via stellar evolution	18.6	19.6	18.9

is that the amount of mass lost through stellar evolution for the runs beginning at the pre-main sequence is higher than the pz-ms runs. One might expect that because the pz-ms stars are started further along than the pz-pms stars, more stars will have a chance to get to the giant branch and blow off their envelopes. For the pz-pms runs, the mergers result in more high mass stars which lose more mass during stellar evolution than their low mass counterparts. On the other hand, the rw-pms stars are rarely accreting so there is less rejuvenation of binary companions. Rejuvenation retards a star's evolution, and thus slows mass loss.

5.3 Simulations of Observations

One significant aspect of computational work is to bridge the gap between theory and observations. A useful way to do this is to try display the data as it would look if an observer was documenting the object.

Colour-magnitude diagrams (CMDs) provide an excellent snapshot of a particular cluster. In figure 5.16 the time evolution of the appearance of the cluster is shown. Features such as the binary main sequence, blue stragglers, and a collection of giants and white dwarfs are clearly visible. The pre-main sequence stars begin above the main sequence and descend down towards the main sequence. In the more evolved colour-magnitude diagrams, a gap is noticeable in the main sequence for the rw-pms and pz-pms at the 300 Myr mark. This gap has been observed in NGC 3603 (Eisenhauer et al. 1998), and although NGC 3603 is less than 5 Myr old, the gap

should continue to move down the zero age main sequence until it becomes too small to notice, especially at the lower magnitudes. The gap itself is a result of non-linearities in the mass-absolute magnitude relation. These non-linearities arise from out of equilibrium CNO burning that is initially out of equilibrium in pre-main sequence stars (Piskunov and Belikov 1996). Another notable feature of the CMDs is that the binary main sequence is so sparsely populated in the pz-pms1 runs. The decreased binary fractions due to mergers is evidenced through this, but the effect is uniform over all colours. This may be actually one of the only distinguishable features of the rw-pms1 CMDs, along with perhaps the population of the white dwarf regime.

Finally, it is desirable to know what the cluster 'looks like' to an observer. That is, what sort of image would show up in a telescope from earth looking at one of the clusters in space. In figures 5.17 and 5.18, the spatial concentration in the x-y plane of the magnitude is shown at 0 Myr, and 600 Myr respectively. In the initial snapshots, the greater luminosity of the pre-main sequence stars is evident. Of course, if this was a real young cluster, a large amount of that luminosity would be obscured by dust. In the more evolved snapshots, the difference is more subtle, but it seems that the rw-pms run has more of its luminosity concentrated in the core.

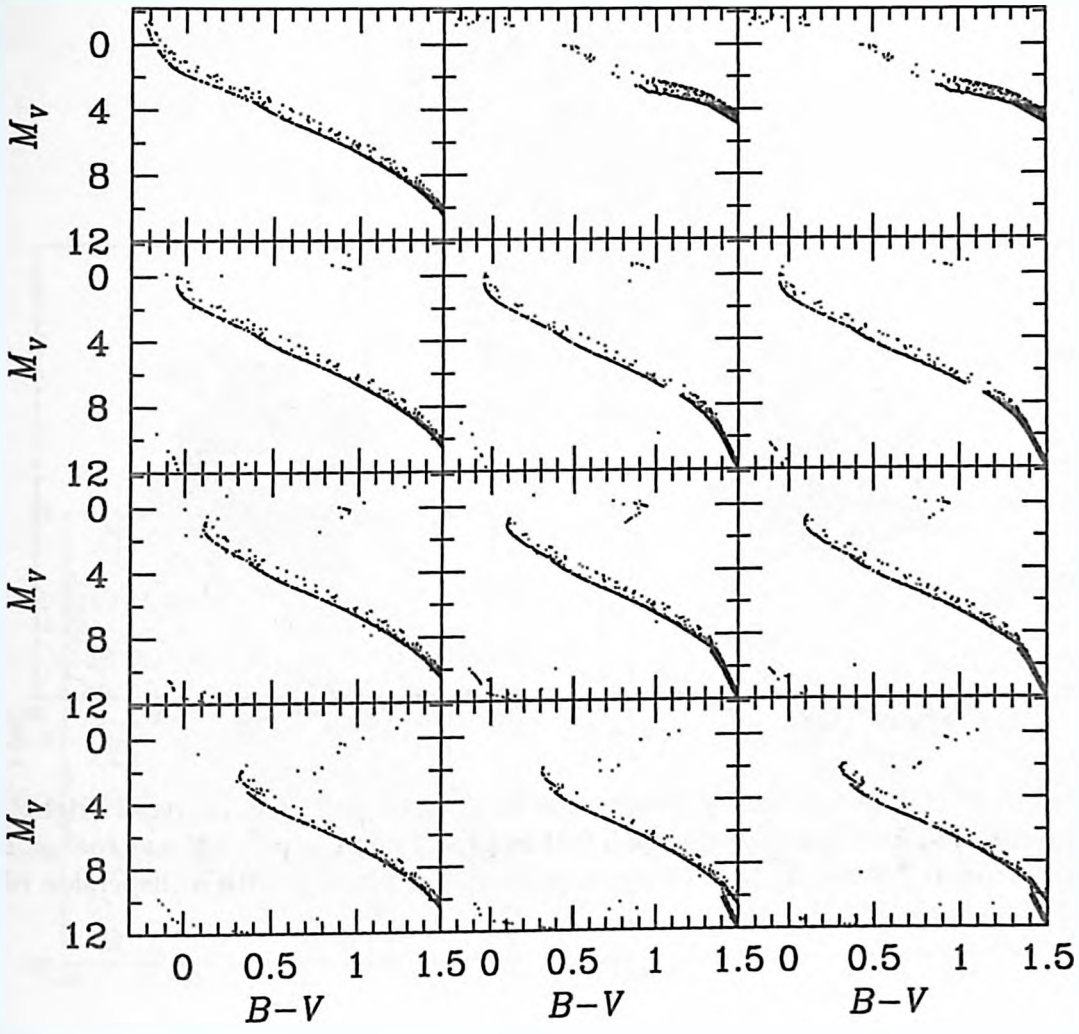


Figure 5.16: Colour-Madnitude Diagrams (CMDs) for models pz-ms1 (left), pz-pms1 (center), and rw-pms1 (right). Descending chronologically, the CMDs shown represent the clusters at approximately 0, 300, 600, and 1200 Myr.

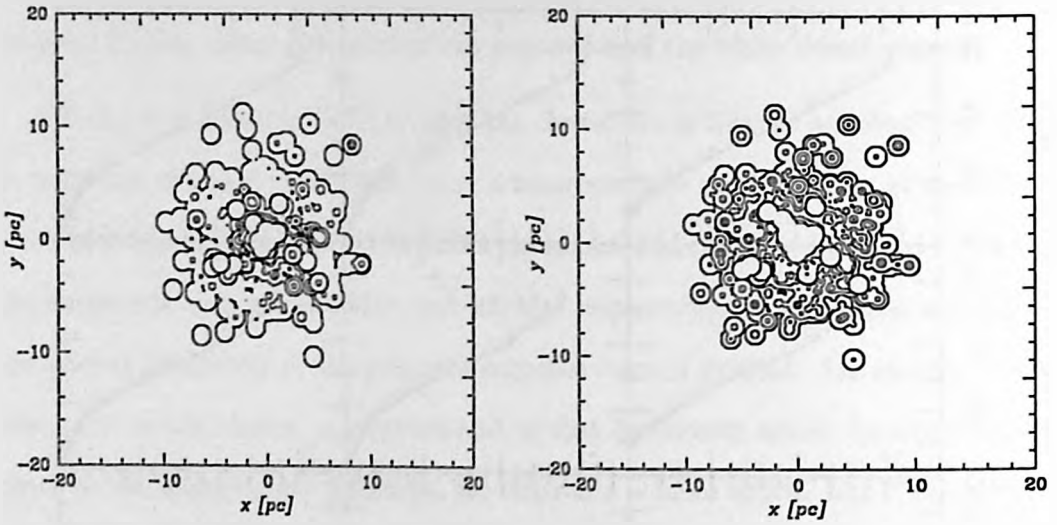


Figure 5.17: Initial Luminosity Distribution for pz-pms1 (left) and rw-pms1 (right). Contours are plotted at -0.86 mag/pc^2, 0.34 mag/pc^2, 1.6 mag/pc^2, 2.8 mag/pc^2, 4.1 mag/pc^2, and 7.8 mag/pc^2. A Gaussian point spread function with a dispersion of 0.35 pc is applied to the stars.

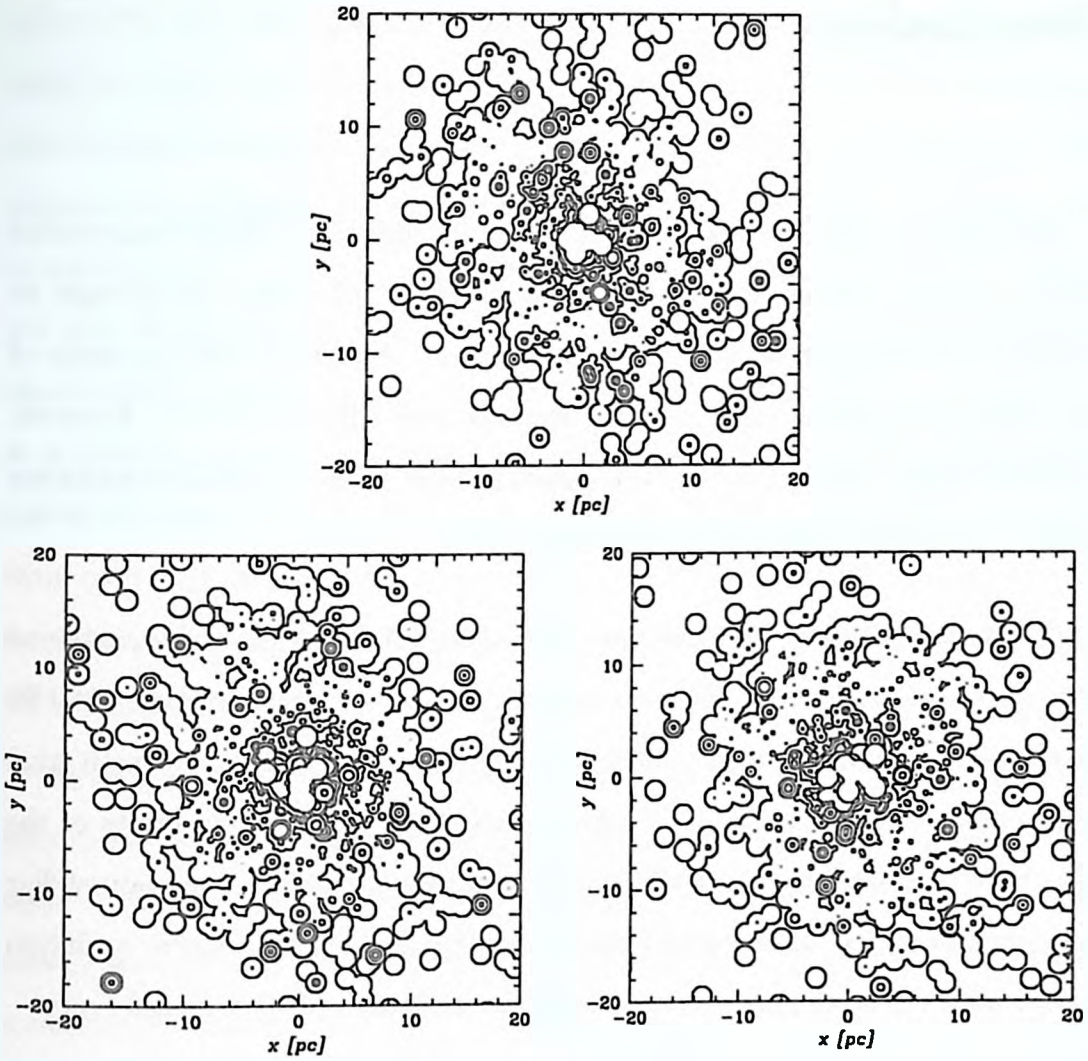


Figure 5.18: Luminosity Distribution for pz-ms1 (top), pz-pms1 (left), and rw-pms1 (right) at 600 Myr. Contours are plotted at -0.86 mag/pc^2, 0.34 mag/pc^2, 1.6 mag/pc^2, 2.8 mag/pc^2, 4.1 mag/pc^2, and 7.8 mag/pc^2. A Gaussian point spread function with a dispersion of 0.35 pc is applied to the stars.

Chapter 6

Conclusions

Pre-main sequence stars abound in the galaxy, yet their role in dynamical systems remains largely ignored. This is in part due to the fact that they are thought to be dynamically insignificant, but also because of dust obscuration and the rarity of very young clusters that makes pre-main sequence stars difficult to study. However, research is being pushed to the limits of detail, and the inclusion of this evolutionary stage in dynamical models is the next logical step.

In this work, 3 series of 3 runs were evolved for 1.5 Gyr. One series contained 3096 main sequence stars in different configurations, with a binary fraction of 50 %. The second series was made up of identical configurations as the first, but with stars beginning at the pre-main sequence. For the third series, the binary orbits of the stars were reassigned such that the stars were initially detached. The main stellar properties for the pre-main sequence stars were taken from available stellar evolution tracks, while the other properties were taken as adaptations of other stellar types.

One of the main problems facing star cluster theorists is that it is difficult to determine whether the results represent a degenerate end of a number of input parameters/physics or a unique solution. It is therefore important to continually compare results with the previous configurations and implementations. This work focused on first changing one parameter - the starting point of stellar evolution - and also compared to changing another parameter - orbital separations of the binary systems.

The intermediate step is crucial in determining which effects are uniquely a result of adding pre-main sequence evolution.

Although the differences between the various runs are small, they are mostly due to the different number of mergers. In the pz-pms runs, a lot of the binaries begin in contact, since the binary orbits are based on *main sequence* radii, and pre-main sequence stars have much larger radii. These early pre-main sequence/pre-main sequence mergers show up as a decrease in the total number of stars, but an increase in average mass. This effect drastically changes the mass function of the cluster, not just initially, but the effect continues to be noticed for quite a large portion of the cluster's lifetime. On the other hand, since the stars in the rw-pms runs start at a greater separation, and since the initial pre-main sequence radius is the largest radius for a given star until it becomes a giant star, there is an absence of mergers in these runs. This increases the binary fraction, and affects the mass function to match something one would expect from a population synthesis (stellar/binary evolution with no dynamics).

In spite of the initial drop of total number of objects in the pz-pms models, the time evolution of the total mass isn't noticeably affected by the starting point of the stars. The total mass evolution of the rw-pms runs behaves similarly. One might expect the pms runs to have a flatter or steeper curve for total mass versus time for a number of reasons, but it seems that these reasons are countered by other effects, or are not very dominant. For instance, there is no mass loss due to stellar wind for pre-main sequence stars, and their pre-main sequence lifetime offsets the evolutionary stage in which the stars would lose a lot of mass. In this way, the total mass of the system should be decreasing at a lower rate.

Similarly, the total number of stars in both pms clusters seem to decrease at a slower rate than their pz-ms counterparts. This could be due to a smaller number of

escapers ejected from the cluster through an encounter. Perhaps the inclusion of pre-main sequence evolution causes the stellar interactions to be less violent. During a binary star/single star exchange, accretion during the stellar encounter may affect the ensuing binary parameters which in turn could affect the resulting ejection velocity. Indeed, the merger of close binaries (as in the pz-pms runs) or the absence of close binaries in the rw-pms runs should affect the cluster in the same way since binaries act as a heat sink for the total energy of the cluster. Since the predominant form of binary that stars in these models will encounter is not very hard, the heat sinks of the cluster can absorb more, thus decreasing the energy available for other purposes.

In the rw-pms models very little circularization is observed. This is because the stars are contracting away from each other, and therefore cannot maintain contact as required for circularization. The pz-pms runs contain the largest number of circularized binaries, while the pz-ms runs contain quite a few as well. As surveys of binary parameters become more complete, it will be possible to determine how common circularization is. Currently it seems that there are not a large number of binary stars with eccentricities less than 0.0001 (Duquennoy and Mayor 1991).

Another item of note is the frequency of the different types of mergers. Almost all of the mergers were due to the normal binary evolution in which a star's radius becomes too big and contacts the other star. The two most frequent mergers in the pms runs are pre-main sequence/pre-main sequence and pre-main sequence/giant star mergers (with the latter being more dominant in the rw-pms case). Detailed work on these types of encounters is still in an embryonic stage (Laycock and Sills, in preparation), but clearly they need to be better understood if they occur this frequently. As it stands, it appears that the result will be a larger pre-main sequence star in the former case, and a larger giant star in the latter case, but this needs confirmation.

The differences between the frequency and attributes of escapers of the models is subtle, but since there are more low mass, high velocity escapers in the rw-pms models, perhaps their numbers are being underestimated in standard main sequence cluster evolution models. This could be confirmed if the velocity distribution of a large number of field stars was available as it is doubtful that the galactic tidal field could affect the velocities of said stars significantly.

There is a significant amount of future work that could be done in this area and could involve simulating other cluster configurations (for instance, with more stars or in different tidal field). Since pre-main sequence stars in very young clusters have been studied (e.g., Eisenhauer et al. (1998)), a focus on the first 100 Myr of a simulation could be insightful. Another avenue of research would be to perform population synthesis in order to better determine which initial distribution of binary parameters will result in the observed distribution of these parameters.

The end result then is that while consideration of pre-main sequence evolution is not important in basic star cluster theory, any detailed treatment of stellar populations within such clusters warrants such care. These stars can affect the dynamics as well as the population evolution of the cluster in very important ways which cannot be overlooked. Research in this area will continue to progress until a completely satisfactory match with observations has occurred.

Bibliography

Aarseth, S. J. 1985, In *IAU Symp. 113: Dynamics of Star Clusters*, pp. 251.

Bailey, S. I. 1893, Astronomy and Astro-Physics, XII, 689.

Cohen, M. & Kuhi, L. V. 1979, ApJS, 41, 743.

D'Antona, F. & Mazzitelli, I. 1997, Memorie della Societa Astronomica Italiana, 68, 807.

Duquennoy, A. & Mayor, M. 1991, August, A&A, 248, 485.

Eggleton, P. P. 1983, ApJ, 268, 368.

Eggleton, P. P., Tout, C. A., & Fitchett, M. J. 1989, ApJ, 347, 998.

Eisenhauer, F., Quirrenbach, A., Zinnecker, H., & Genzel, R. 1998, ApJ, 498, 278.

Elson, R., Hut, P., & Inagaki, S. 1987, ARA&A, 25, 565.

Garcia Lopez, R. J., Rebolo, R., & Martin, E. L. 1994, A&A, 282, 518.

Hartman, J. W. 1997, A&A, 322, 127.

Hayashi, C. 1961, Publications of the Astonomical Society of Japan, 13, 450.

Heggie, D. C. & Ramamani, N. 1995, MNRAS, 272, 317.

Hénon, M. 1961, Annales d'Astrophysique, 24, 369.

Hénon, M. 1965, Annales d'Astrophysique, 28, 61.

Hénon, M. 1975, In *IAU Symp. 69: Dynamics of the Solar Systems*, pp. 133.

Henyey, L. G., Lelevier, R., & Levée, R. D. 1955, PASP, 67, 154.

Herbig, G. H. 1957, In *IAU Symp. 3: Non-stable stars*, pp. 3.

Herschel, W. 1893, ApJ, 419, 479.

Hjellming, M. S. & Webbink, R. F. 1987, ApJ, 318, 794.

Huang, S.-S. 1961, ApJ, 134, 12.

Hurley, J. R., Pols, O. R., & Tout, C. A. 2000, MNRAS, 315, 543.

Hurley, J. R., Tout, C. A., Aarseth, S. J., & Pols, O. R. 2001, MNRAS, 323, 630.

Hut, P. 2001, In *IAU Symp. 208: Astrophysical Supercomputing Using Particles*, pp. 1.

Jeans, J. H. 1913, MNRAS, 74, 109.

Joy, A. H. 1945, ApJ, 102, 168.

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

Kroupa, P. 1995, MNRAS, 277, 1522.

Kroupa, P., Gilmore, G., & Tout, C. A. 1991, MNRAS, 251, 293.

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

Larson, R. B. 1969, MNRAS, 145, 271.

Larson, R. B. 1970, MNRAS, 147, 323.

Makino, J. & Aarseth, S. J. 1992, April, Publications of the Astonomical Society of Japan, 44, 141.

Makino, J., Fukushige, T., Koga, M., & Namura, K. 2003, Publications of the Astonomical Society of Japan, 55, 1163.

Messier, C. 1784, ApJ, 546, 681.

Meylan, G. & Heggie, D. C. 1997, ARA&A, 8, 1.

Mikkola, S. & Aarseth, S. J. 1998, New Astronomy, 3, 309.

Nelemans, G., Verbunt, F., Yungelson, L. R., & Portegies Zwart, S. F. 2000, August, A&A, 360, 1011.

Palla, F. & Stahler, S. W. 1999, ApJ, 525, 772.

Perryman, M. A. C., Brown, A. G. A., Lebreton, Y., Gomez, A., Turon, C., de Strobel, G. C., Mermilliod, J. C., Robichon, N., Kovalevsky, J., & Crifo, F. 1998, A&A, 331, 81.

Peters, P. C. 1964, Physical Review, 136, 1224.

Pickering, E. C. 1897, Annals of Harvard College Observatory, 26, 193.

Piskunov, A. E. & Belikov, A. N. 1996, Astronomy Letters, 22, 466.

Plummer, H. C. 1911, MNRAS, 71, 460.

Plummer, W. E. 1905, MNRAS, 65, 801.

Portegies Zwart, S. F., McMillan, S. L. W., Hut, P., & Makino, J. 2001, MNRAS, 321, 199.

Portegies Zwart, S. F. & Verbunt, F. 1996, A&A, 309, 179.

Rappaport, S., Verbunt, F., & Joss, P. C. 1983, ApJ, 275, 713.

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

Scalo, J. M. 1986, Fundamentals of Cosmic Physics, 11, 1.

Schaerer, D., de Koter, A., Schmutz, W., & Maeder, A. 1996, A&A, 310, 837.

Siess, L., Dufour, E., & Forestini, M. 2000, A&A, 358, 593.

Sills, A., Deiters, S., Eggleton, P., Freitag, M., Giersz, M., Heggie, D., Hurley, J., Hut, P., Ivanova, N., Klessen, R. S., Kroupa, P., Jr., J. C. L., McMillan, S., Zwart, S. P., & Zinnecker, H. 2003, New Astronomy, 8, 605.

Sills, A., Jr., J. C. L., Bailyn, C. D., Demarque, P., Rasio, F. A., & Shapiro, S. L. 1997, ApJ, 487, 290.

Stahler, S. W. 1983, ApJ, 274, 822.

Stahler, S. W., Shu, F. H., & Taam, R. E. 1980a, ApJ, 241, 637.

Stahler, S. W., Shu, F. H., & Taam, R. E. 1980b, ApJ, 242, 226.

Stauffer, J. R., Jones, B. F., Backman, D., Hartmann, L. W., y Navascus, D. B., Pinsonneault, M. H., Terndrup, D. M., & Muench, A. A. 2003, AJ, 126, 833.

Thorne, K. S. & Żytkow, A. N. 1977, ApJ, 212, 832.

von Hoerner, S. 1960, Zeitschrift fur Astrophysics, 50, 184.

Walker, M. F. 1956, ApJS, 2, 365.

Weidemann, V. 1993, A&A, 275, 158.

Appendix A

Interpolation Analysis

Determining which method of interpolation would be appropriate for implementation in the look-up table was of great importance. Although the time steps in the lookup table should provide good resolution for calculating the required parameters, it isn't obvious whether logarithmic or linear interpolation would more accurately reflect the trends in a pre-main sequence evolutionary track.

The linear interpolation is rather straight forward. For a given parameter X, the value for X at some mass m and time t is given by the following set of equations:

$$X(m_j,t) = \frac{t_{m_j,i+1} - t}{t_{m_j,i+1} - t_{m_j,i}} X_{m_j,i} + \frac{t - t_{m_j,i}}{t_{m_j,i+1} - t_{m_j,i}} X_{m_j,i+1}, \tag{7.1}$$

$$X(m_{j+1},t) = \frac{t_{m_{j+1},i+1} - t}{t_{m_{j+1},i+1} - t_{m_{j+1},i}} X_{m_{j+1},i} + \frac{t - t_{m_{j+1},i}}{t_{m_{j+1},i+1} - t_{m_{j+1},i}} X_{m_{j+1},i+1}, \tag{7.2}$$

$$X(m,t) = \frac{m_{j+1} - m}{m_{j+1} - m_j} X(m_j,t) + \frac{m - m_j}{m_{j+1} - m_j} X(m_{j+1},t), \tag{7.3}$$

where subscripts denote values from the lookup table.

Since some of the inflections in the values of the tracks take place over small time scales when compared to the length of the entire track, logarithmic interpolation deserved to be examined. Logarithmic interpolation takes the same form as linear interpolation with the exception that the values in the equations are the logarithm (base 10) of the values in the lookup table. The final value is then raised as an exponent corresponding to the logarithm base.

In order to test which method is better, test points at selected intervals were examined. These test points would be removed, and values would be interpolated in their place. The interpolated values would then be measured against the real values. Admittedly, this method only gives a lower limit on accuracy as one level of resolution is removed, but it should indicate which method is superior.

In total, linear interpolation was more accurate 175 out of 294 times. In most cases, the difference was minimal, but at higher masses the effect of lost resolution was more pronounced. For instance, the largest discrepancy for masses less than 1.5 $M_{\odot}$ was as high as 17.1% for logarithmic interpolation and 15.2% for linear interpolation, but the largest discrepancy all other masses was 45.4% for logarithmic interpolation and 52.9% for linear interpolation. A complete summary of all cases is given by the following tables. In the table the value from the track is given, followed by the value calculated through linear interpolation, followed by a percent difference, with data from the logarithmic interpolation listed along side.

Table 7.1: Interpolation comparison for M = 0.13 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	2.167	2.20	1.71%	2.20	1.51%	2829	2825.34	-0.13%	2824.28	-0.17%
1.00E+05	1.795	1.81	0.96%	1.81	0.62%	2885	2881.01	-0.14%	2880.28	-0.16%
1.00E+06	1.016	1.15	13.38%	1.15	12.96%	3026	2993.76	-1.07%	2993.05	-1.09%
1.00E+07	0.497	0.49	-2.22%	0.48	-2.89%	3146	3140.17	-0.19%	3139.46	-0.21%
1.00E+08	0.210	0.21	-0.01%	0.21	-0.70%	3147	3124.42	-0.72%	3122.53	-0.78%
3.00E+08	0.157	0.16	0.49%	0.16	-0.61%	3090	3048.38	-1.35%	3044.77	-1.46%
6.00E+08	0.148	0.15	-0.35%	0.14	-2.07%	3072	3005.77	-2.16%	3000.20	-2.34%
1.00E+09	0.146	0.13	-12.05%	0.14	-3.35%	3069	2894.16	-5.70%	2978.95	-2.93%

Table 7.2: Interpolation comparison for M = 0.16 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	2.372	2.31	-2.560%	2.31	-2.634%	2898	2914.86	0.58%	2912.86	0.51%
1.00E+05	1.965	1.93	-1.67%	1.93	-1.98%	2945	2955.29	0.35%	2954.15	0.31%
1.00E+06	1.25	1.36	8.77%	1.30	4.30%	3059	3048.04	-0.358%	3047.93	-0.362%
1.00E+07	0.538	0.53	-0.86%	0.53	-1.25%	3208	3207.60	-0.01%	3206.83	-0.04%
1.00E+08	0.223	0.22	0.16%	0.22	-0.54%	3229	3217.14	-0.37%	3215.95	-0.40%
3.00E+08	0.178	0.18	-0.16%	0.18	-1.38%	3193	3173.62	-0.61%	3171.97	-0.66%
6.00E+08	0.175	0.17	-0.68%	0.17	-2.23%	3189	3166.13	-0.72%	3164.33	-0.77%
1.00E+09	0.174	0.17	-1.07%	0.17	-2.55%	3188	3163.71	-0.76%	3161.84	-0.82%

Table 7.3: Interpolation comparison for M = 0.2 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	2.429	2.47	1.84%	2.47	1.74%	3040	3039.33	-0.02%	3035.25	-0.16%
1.00E+05	2.179	2.18	0.05%	2.18	-0.12%	3046	3056.00	0.33%	3052.80	0.22%
1.00E+06	1.623	1.55	-15.20%	1.51	-17.14%	3077	3129.92	1.72%	3128.93	1.69%
1.00E+07	0.611	0.62	0.71%	0.61	0.25%	3283	3279.39	-0.11%	3278.25	-0.14%
1.00E+08	0.26	0.26	-0.80%	0.26	-1.52%	3321	3325.37	0.13%	3323.73	0.08%
3.00E+08	0.211	0.21	-0.79%	0.21	-1.98%	3293	3301.99	0.27%	3299.86	0.21%
6.00E+08	0.207	0.21	-0.63%	0.20	-1.99%	3290	3297.91	0.24%	3295.68	0.17%
1.00E+09	0.207	0.21	-0.91%	0.20	-2.31%	3290	3297.33	0.22%	3295.08	0.15%

Table 7.4: Interpolation comparison for M = 0.25 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	2.601	2.59	-0.62%	2.58	-0.80%	3216	3203.50	-0.39%	3199.32	-0.52%
1.00E+05	2.325	2.40	3.44%	2.39	2.99%	3213	3205.50	-0.23%	3201.52	-0.36%
1.00E+06	1.917	2.00	4.09%	1.99	3.68%	3218	3219.20	0.04%	3216.07	-0.06%
1.00E+07	0.651	0.64	-1.07%	0.64	-1.47%	3383	3380.87	-0.06%	3379.59	-0.10%
1.00E+08	0.294	0.29	0.22%	0.29	-0.41%	3442	3425.71	-0.47%	3424.12	-0.52%
3.00E+08	0.246	0.25	-0.06%	0.24	-1.06%	3435	3408.71	-0.77%	3406.76	-0.82%
6.00E+08	0.244	0.24	-0.41%	0.24	-1.51%	3434	3407.00	-0.79%	3404.99	-0.84%
1.00E+09	0.244	0.24	-0.38%	0.24	-1.49%	3434	3407.00	-0.79%	3404.99	-0.84%

Table 7.5: Interpolation comparison for M = 0.3 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	2.741	2.81	2.467%	2.79	1.92%	3367	3344.00	-0.68%	3339.22	-0.83%
1.00E+05	2.619	2.60	-0.80%	2.57	-2.00%	3365	3341.89	-0.69%	3337.03	-0.83%
1.00E+06	2.169	1.92	-11.66%	1.92	-11.684%	3361	3342.33	-0.56%	3337.82	-0.69%
1.00E+07	0.695	0.69	-0.59%	0.69	-0.99%	3475	3469.32	-0.16%	3467.25	-0.22%
1.00E+08	0.318	0.32	-0.76%	0.31	-1.58%	3529	3523.00	-0.17%	3521.21	-0.22%
3.00E+08	0.279	0.28	0.00%	0.28	-1.34%	3524	3514.17	-0.28%	3512.40	-0.33%
6.00E+08	0.279	0.28	-0.01%	0.27	-1.47%	3524	3512.01	-0.34%	3510.30	-0.39%
1.00E+09	0.279	0.28	0.13%	0.28	-1.36%	3524	3512.18	-0.34%	3510.46	-0.38%

Table 7.6: Interpolation comparison for M = 0.4 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	3.223	3.28	1.75%	3.23	0.37%	3600	3566.00	-0.94%	3560.44	-1.10%
1.00E+05	3.179	3.18	0.00%	3.13	-1.61%	3600	3566.70	-0.93%	3560.97	-1.08%
1.00E+06	1.925	2.09	8.54%	2.09	8.42%	3691	3594.69	-0.73%	3558.86	-0.89%
1.00E+07	0.791	0.79	-0.60%	0.78	-1.13%	3638	3617.63	-0.56%	3614.66	-0.64%
1.00E+08	0.374	0.38	0.71%	0.37	-0.62%	3683	3677.49	-0.15%	3674.49	-0.23%
3.00E+08	0.346	0.35	0.68%	0.34	-1.34%	3673	3666.15	-0.19%	3663.39	-0.26%
6.00E+08	0.349	0.35	0.18%	0.34	-1.88%	3668	3665.00	-0.08%	3662.20	-0.16%

Table 7.7: Interpolation comparison for M = 0.5 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	3.818	3.67	-3.76%	3.65	-4.49%	3765	3746.50	-0.49%	3743.63	-0.57%
1.00E+05	3.741	3.53	-5.62%	3.51	-6.10%	3770	3749.40	-0.55%	3746.42	-0.63%
1.00E+06	2.059	2.00	-2.90%	2.00	-2.91%	3770	3746.64	-0.62%	3743.45	-0.70%
1.00E+07	0.843	0.84	-0.51%	0.84	-0.85%	3764	3751.24	-0.34%	3749.62	-0.38%
1.00E+08	0.441	0.45	2.38%	0.45	1.15%	3823	3845.41	0.59%	3841.95	0.50%
3.00E+08	0.418	0.42	1.57%	0.42	-0.18%	3808	3822.77	0.39%	3819.82	0.31%
6.00E+08	0.42	0.43	1.64%	0.42	-0.08%	3806	3818.64	0.33%	3815.68	0.25%

Table 7.8: Interpolation comparison for M = 0.6 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	4.126	4.10	-0.73%	4.09	-0.96%	3893	3877.50	-0.40%	3875.87	-0.44%
1.00E+05	3.874	3.88	0.18%	3.88	0.07%	3899	3885.06	-0.36%	3883.36	-0.40%
1.00E+06	1.991	2.10	5.70%	2.10	5.63%	3901	3893.12	-0.20%	3891.16	-0.25%
1.00E+07	0.918	0.93	0.76%	0.92	0.46%	3862	3858.29	-0.10%	3857.14	-0.13%
1.00E+08	0.512	0.52	1.48%	0.51	-0.01%	4011	4045.86	0.87%	4039.99	0.72%
3.00E+08	0.503	0.51	1.71%	0.50	-0.01%	3973	4023.80	1.28%	4018.01	1.13%
6.00E+08	0.506	0.51	1.56%	0.51	-0.12%	3969	4019.96	1.28%	4014.26	1.14%

Table 7.9: Interpolation comparison for M = 0.7 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	4.374	4.40	0.63%	4.39	0.43%	3990	3980.00	-0.25%	3979.05	-0.27%
1.00E+05	4.083	3.90	-4.45%	3.90	-4.454%	3999	3997.95	-0.03%	3996.71	-0.06%
1.00E+06	2.102	2.04	-2.97%	2.04	-3.09%	4014	4009.53	-0.11%	4008.05	-0.15%
1.00E+07	1.02	1.01	-0.61%	1.01	-0.86%	3954	3954.16	0.00%	3953.12	-0.02%
1.00E+08	0.61	0.61	0.09%	0.60	-1.14%	4268	4323.70	1.31%	4312.31	1.04%
3.00E+08	0.605	0.61	0.46%	0.60	-1.04%	4240	4297.38	1.35%	4285.14	1.06%

Table 7.10: Interpolation comparison for M = 0.8 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	4.601	4.44	-3.58%	4.43	-3.61%	4069	4072.00	0.07%	4071.19	0.05%
1.00E+05	3.831	3.94	2.73%	3.93	2.66%	4100	4089.12	-0.27%	4088.14	-0.29%
1.00E+06	2.142	2.18	1.62%	2.18	1.56%	4119	4109.05	-0.24%	4107.96	-0.27%
1.00E+07	1.034	1.04	0.54%	1.04	0.32%	4047	4053.09	0.15%	4051.82	0.12%
1.00E+08	0.706	0.70	-0.59%	0.70	-1.47%	4637	4637.05	0.00%	4622.06	-0.32%
3.00E+08	0.709	0.70	-1.04%	0.69	-2.02%	4625	4623.84	-0.03%	4608.22	-0.36%

Table 7.11: Interpolation comparison for M = 0.9 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	4.533	4.58	1.12%	4.58	1.119%	4153	4140.24	-0.31%	4139.63	-0.32%
1.00E+05	3.808	3.86	1.24%	3.85	1.23%	4179	4169.62	-0.22%	4169.06	-0.24%
1.00E+06	2.223	2.23	0.12%	2.22	0.02%	4203	4199.11	-0.09%	4198.34	-0.11%
1.00E+07	1.113	1.12	0.32%	1.11	0.11%	4152	4160.24	0.20%	4158.68	0.16%
1.00E+08	0.795	0.80	0.65%	0.79	-0.05%	5010	4996.80	-0.26%	4983.85	-0.52%

Table 7.12: Interpolation comparison for M = 1.0 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	4.614	4.52	-1.93%	4.52	-1.95%	4209	4215.39	0.15%	4214.90	0.14%
1.00E+05	3.902	3.85	-1.38%	3.85	-1.39%	4238	4243.49	0.13%	4243.00	0.12%
1.00E+06	2.33	2.32	-0.39%	2.32	-0.46%	4280	4273.63	-0.15%	4273.06	-0.16%
1.00E+07	1.192	1.20	0.32%	1.19	0.11%	4271	4276.15	0.12%	4274.31	0.08%
1.00E+08	0.895	0.90	0.49%	0.89	-0.18%	5357	5340.92	-0.30%	5330.63	-0.49%

Table 7.13: Interpolation comparison for M = 1.1 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	4.408	4.58	1.86%	4.58	1.84%	4278	4266.50	-0.27%	4266.07	-0.28%
1.00E+05	3.889	3.92	0.89%	3.92	0.88%	4308	4295.94	-0.28%	4295.55	-0.29%
1.00E+06	2.411	2.41	0.13%	2.41	0.06%	4344	4339.97	-0.09%	4339.55	-0.10%
1.00E+07	1.289	1.29	0.32%	1.29	0.12%	4384	4400.51	0.38%	4398.36	0.33%

Table 7.14: Interpolation comparison for M = 1.2 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	4.497	4.49	-0.20%	4.49	-0.2004%	4327	4326.00	-0.02%	4325.73	-0.03%
1.00E+05	4.041	4.04	-0.17%	4.04	-0.19%	4350	4349.42	-0.01%	4349.12	-0.02%
1.00E+06	2.562	2.56	-0.05%	2.56	-0.12%	4401	4398.73	-0.05%	4398.30	-0.06%
1.00E+07	1.339	1.35	0.77%	1.35	0.52%	4594	4621.26	0.59%	4616.54	0.49%

Table 7.15: Interpolation comparison for M = 1.3 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	4.478	4.46	-0.51%	4.45	-0.52%	4374	4377.50	0.08%	4377.21	0.07%
1.00E+05	4.01	3.99	-0.43%	3.99	-0.45%	4403	4399.50	-0.08%	4399.25	-0.09%
1.00E+06	2.557	2.55	-0.10%	2.55	-0.16%	4462	4454.92	-0.16%	4454.58	-0.17%
1.00E+07	1.456	1.47	1.14%	1.47	0.87%	4745	4760.04	0.32%	4755.11	0.21%

Table 7.16: Interpolation comparison for M = 1.4 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	4.413	4.49	1.84%	4.49	1.83%	4428	4418.50	-0.21%	4418.28	-0.22%
1.00E+05	4.102	4.16	1.47%	4.16	1.46%	4442	4438.31	-0.08%	4438.13	-0.09%
1.00E+06	2.642	2.65	0.39%	2.65	0.34%	4510	4507.44	-0.06%	4507.21	-0.0619%
1.00E+07	1.591	1.65	3.75%	1.64	2.94%	5093	5132.65	0.78%	5122.88	0.59%

Table 7.17: Interpolation comparison for M = 1.5 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	4.31	4.28	-5.10%	4.28	-5.15%	4463	4476.00	0.29%	4475.74	0.2855%
1.00E+05	4.198	4.02	-4.34%	4.01	-4.37%	4479	4488.90	0.22%	4488.66	0.2156%
1.00E+06	2.731	2.69	-1.36%	2.69	-1.39%	4553	4556.01	0.07%	4555.78	0.06%
1.00E+07	1.865	1.92	2.79%	1.89	1.31%	5464	5563.34	1.82%	5544.36	1.47%

Table 7.18: Interpolation comparison for M = 1.6 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	4.147	4.37	5.39%	4.37	5.34%	4524	4508.00	-0.35%	4507.78	-0.36%
1.00E+05	3.854	4.02	4.20%	4.01	4.18%	4540	4530.70	-0.20%	4530.46	-0.21%
1.00E+06	2.715	2.75	1.33%	2.75	1.30%	4603	4604.32	0.03%	4604.06	0.02%
1.00E+07	2.243	1.93	-14.16%	1.92	-14.19%	6070	6344.40	4.52%	6286.58	3.57%

Table 7.19: Interpolation comparison for M = 1.7 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	4.231	4.23	-0.13%	4.22	-0.15%	4553	4556.00	0.07%	4555.89	0.06%
1.00E+05	3.936	3.93	-0.11%	3.93	-0.14%	4577	4573.47	-0.08%	4573.35	-0.08%
1.00E+06	2.796	2.80	0.07%	2.80	0.02%	4653	4646.65	-0.14%	4646.45	-0.14%
1.00E+07	1.964	2.00	1.86%	1.99	1.13%	7209	6689.79	-7.20%	6661.79	-7.59%

Table 7.20: Interpolation comparison for M = 1.8 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	4.304	4.15	-3.51%	4.15	-3.63%	4588	4594.50	0.14%	4594.31	0.14%
1.00E+05	4.009	3.89	-2.91%	3.89	-2.92%	4607	4615.84	0.19%	4615.67	0.19%
1.00E+06	2.984	2.95	-1.13%	2.95	-1.15%	4682	4687.86	0.13%	4687.67	0.12%
1.00E+07	1.759	1.85	5.11%	1.85	4.90%	7301	7786.67	6.65%	7765.20	6.36%

Table 7.21: Interpolation comparison for M = 1.9 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	4.076	4.15	1.99%	4.15	1.92%	4636	4635.00	-0.02%	4634.76	-0.03%
1.00E+05	3.933	4.00	1.77%	4.00	1.71%	4644	4644.64	0.01%	4644.43	0.01%
1.00E+06	2.944	2.96	0.68%	2.96	0.66%	4737	4737.33	0.01%	4737.05	0.00%
1.00E+07	1.739	1.70	-2.09%	1.70	-2.15%	8345	8079.45	-3.18%	8041.58	-3.64%

Table 7.22: Interpolation comparison for M = 2.0 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	4.008	4.00	-0.20%	4.00	-0.24%	4682	4679.67	-0.05%	4679.26	-0.06%
1.00E+05	3.869	3.86	-0.16%	3.86	-0.21%	4690	4689.76	-0.01%	4689.36	-0.01%
1.00E+06	3.025	3.03	0.16%	3.03	0.11%	4783	4788.08	0.11%	4787.43	0.09%
1.00E+07	1.646	1.73	4.92%	1.73	4.91627%	8861	8747.40	-1.28%	8728.55	-1.49%

Table 7.23: Interpolation comparison for M = 2.2 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	3.85	4.06	5.45%	4.06	5.44%	4767	4750.00	-0.36%	4749.27	-0.37%
1.00E+05	3.723	3.90	4.87%	3.90	4.83%	4780	4764.28	-0.33%	4763.48	-0.35%
1.00E+06	3.088	3.20	3.61%	3.19	3.20%	4915	4916.51	0.03%	4914.18	-0.02%
1.00E+07	1.699	1.72	0.94%	1.71	0.84%	9571	9529.58	-0.43%	9496.86	-0.77%

Table 7.24: Interpolation comparison for M = 2.5 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	4.138	3.93	-4.94%	3.93	-4.96%	4852	4980.00	2.64%	4976.93	2.57%
1.00E+05	4.025	3.90	-3.13%	3.90	-3.19%	4870	5010.33	2.88%	5006.64	2.81%
1.00E+06	3.56	4.10	15.12%	4.02	12.95%	5050	5410.09	7.13%	5393.10	6.79%

Table 7.25: Interpolation comparison for M = 2.7 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	3.989	4.68	17.26%	4.63	16.13%	5122	5184.40	1.22%	5168.73	0.91%
1.00E+05	3.989	4.68	17.26%	4.63	16.13%	5122	5184.40	1.22%	5168.73	0.91%
1.00E+06	4.796	3.20	-33.20%	3.17	-33.80%	6013	7475.41	24.32%	6948.19	15.55%

Table 7.26: Interpolation comparison for M = 3.0 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	5.487	5.07	-7.54%	4.89	-10.81%	5683	5473.00	-3.70%	5454.73	-4.02%
1.00E+05	5.554	4.93	-11.20%	4.80	-13.61%	5839	5701.31	-2.36%	5660.76	-3.05%
1.00E+06	2.724	3.95	44.94%	3.78	38.63%	11048	8302.01	-24.86%	7807.97	-29.33%

Table 7.27: Interpolation comparison for M = 3.5 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	6.849	6.54	-4.54%	6.45	-5.76%	6105	6225.08	1.97%	6202.72	1.60%
1.00E+05	6.435	6.14	-4.62%	6.11	-5.05%	6629	6806.15	2.67%	6738.53	1.65%
1.00E+06	2.558	2.61	2.16%	2.61	1.85%	12062	12723.97	5.49%	12605.49	4.51%

Table 7.28: Interpolation comparison for M = 4.0 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	7.606	7.13	-6.27%	7.12	-6.40%	6727	7203.39	7.08%	7046.11	4.74%
1.00E+05	6.692	6.02	-10.01%	5.99	-10.42%	7798	8718.67	11.81%	8283.12	6.22%
1.00E+06	2.405	2.59	7.59%	2.59	7.56%	14461	13570.53	-6.16%	13421.06	-7.19%

Table 7.29: Interpolation comparison for M = 5.0 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	7.676	5.72	-25.53%	5.39	-29.73%	9421	11968.31	27.04%	10758.71	14.20%
1.00E+05	5.24	5.04	-3.82%	4.76	-9.17%	12839	12554.24	-2.22%	11615.70	-9.53%
1.00E+06	2.663	2.68	0.59%	2.66	0.04%	16507	16453.64	-0.32%	16331.23	-1.06%

Table 7.30: Interpolation comparison for M = 6.0 $M_\odot$

t [yr]	R [$R_\odot$]	Lin	% diff	Log	% diff	T [K]	Lin	% diff	Log	% diff
1.00E+04	3.921	6.00	52.91%	5.70	45.36%	17306	16577.33	-4.21%	16144.35	-6.71%
1.00E+05	3.385	4.35	28.51%	4.25	25.70%	17291	14212.00	-17.81%	13303.79	-23.06%

Appendix B

Pre-Main Sequence Additions to STARLAB

The main addition to the STARLAB code was the pre-main sequence class that dictates the evolution of pre-main sequence stars from the deuterium burning birthline until the main sequence. This is done primarily through a lookup table. What follows is the main addition to the computer code.

```
///
// pre_main_sequence.C
//
// derived class of class star.
// class pre_main_sequence describes stellar evolution
// for a pre-main sequence star.
// Class pre_main_sequence will automatically create the following
// base classes: star, single star and starbase.


#include "proto_star.h"
#include "main_sequence.h"
#include "brown_dwarf.h"

real pms_log:parray_data [10000][10000];
real pms_log:parray_masses [10000];
int pms_status=0;
int pms_number_of_masses=0;

// Default (empty) constructor in pre_main_sequence.h

#if 0
void pre_main_sequence::adjust_initial_star() {


  if(relative_age<=0)
    relative_age = max(current_time, 0.0);

  core_mass = pre_main_sequence_core_mass();
  core_radius = pre_main_sequence_core_radius();
  update_wind_constant();

  instantaneous_element();

  update();

}
#endif

pre_main_sequence::pre_main_sequence(proto_star & p) : single_star(p) {

        delete &p;

        last_update_age = 0;
        relative_age = 0;
        relative_mass = envelope_mass + core_mass;
```

```
        envelope_mass = relative_mass - 0.0001;
        core_mass = 0.0001;

        adjust_next_update_age();
        update_wind_constant();

        instantaneous_element();
        update();

        post_constructor();
}

void pre_main_sequence::update() {

  // last update age is not after stellar expansion timescale is set.
// (CN+SPZ May  4 1999) last_update_age now used as time of last type change
//  last_update_age = relative_age;

  detect_spectral_features();

}

// wind_constant is fraction of envelope lost in nuclear lifetime
// of stars. Should be updated after mass accretion
// (SPZ+CN: 1 Oct 1998)
void pre_main_sequence::update_wind_constant() {

  wind_constant = 0;

}

// Adjust radius & luminosity at relative_age
void pre_main_sequence::instantaneous_element() {


  real T_eff;
  pms_status = 2;

  if(relative_mass <=7.0)
  {
    int m_count = 0;
    real m2 = 0.00;
    real m1, R1, R2, T1, T2;
    char buffer[2000];
    char filename[60];


    while (m2 < relative_mass && m_count < pms_number_of_masses) //know the mass range we're dealing with
    {
      m1 = m2;
      m2 = pms_lookuparray_masses[m_count];
      m_count += 1;
    }


    if (m1 == 0.0) // if less than lowest mass in the lookup table
    {
      radius = pms_lookuparray_data[0][1];
      T_eff = pms_lookuparray_data[0][2];
    }
    else if (m_count == pms_number_of_masses)  // if higher than the highest mass in the lookup table
    {
      radius = pms_lookuparray_data[0][(pms_number_of_masses*3)-2];
      T_eff = pms_lookuparray_data[0][(pms_number_of_masses*3)-1];
    }
    else
    {
      R1 = pms_lookuparray_data[0][(m_count*3)-5];
      T1 = pms_lookuparray_data[0][(m_count*3)-4];

      R2 = pms_lookuparray_data[0][(m_count*3)-2];
      T2 = pms_lookuparray_data[0][(m_count*3)-1];

      //1D linear interpolation
      radius = ((m2 - relative_mass)/(m2-m1))*R1 +  ((relative_mass-m1)/(m2-m1))*R2;
      T_eff = ((m2 - relative_mass)/(m2-m1))*T1 +  ((relative_mass-m1)/(m2-m1))*T2;

    }
  }
  else //main_sequence recipe
  {
    real alpha, beta, gamma, delta, kappa, lambda;
    real log_mass = log10(relative_mass);

    alpha = 0.08353 + 0.0565*log_mass;
    beta  = 0.01291 + 0.2226*log_mass;
    gamma = 0.1151 + 0.00267*log_mass;
    delta = pow(relative_mass, 1.25)
      * (0.1148 + 0.8604*relative_mass*relative_mass)
```

```
    / (0.04651 + relative_mass*relative_mass);
  kappa  = 0.092088 + 0.058138*log_mass;
  lambda = -0.05713 + log_mass*(0.3764 -0.1744*log_mass);

  real ff    = relative_age/main_sequence_time();
  luminosity = base_main_sequence_luminosity()
    * pow(10., (ff*(ff*lambda + kappa)));

  radius = delta*pow(10., (ff*(ff*(ff*gamma + beta) + alpha)));
  T_eff = cnsts.parameters(Tsun)*pow(luminosity/(effective_radius*effective_radius), 0.25);

 }

 effective_radius = radius;
 luminosity = pow(effective_radius, 2.)* pow(T_eff/cnsts.parameters(Tsun), 4.);

}

void pre_main_sequence::evolve_element(const real end_time) {


 real dt = end_time - current_time;
 current_time = end_time;
 relative_age += dt;

 if (relative_mass > 7.0)  // We have no data for PMS stars more massive than this
 {                         // just put some numbers in and send it on it's way
   star_transformation_story(Main_Sequence);
   new main_sequence(*this);
   return;
 }

 real relageinyrs = relative_age*1.0E6;
 real T_eff;

 int m_count = 0;
 int t_count = 0;
 real m2 = 0.00;
 real t1ii = 0.00;
 real R1ii = 0.00;
 real T1ii = 0.00;
 real t2ii = 0.00;
 real R2ii = 0.00;
 real T2ii = 0.00;
 real m1, t1i, t2i, R1i, R2i, T1i, T2i, R1, R2, T1, T2;
 char buffer[2000];
 char filename[50];


 while (m2 < relative_mass && m_count < pms_number_of_masses) //know the mass range we're dealing with
 {
   m1 = m2;
   m2 = pms_lookuparray_masses[m_count];
   m_count += 1;
 }


 if (m1 == 0.0) // if less than lowest mass in the lookup table
 {

   while (t1ii < relageinyrs) // get the ages, radii, and Temperature for the first mass
   {
     t1ii = pms_lookuparray_data[t_count][0];
     R1ii = pms_lookuparray_data[t_count][1];

     if (R1ii == 0.0) // if at the end of pre-main sequence, but not changed to MS yet
break;

     t_count += 1;
   }
   if (R1ii == 0.0)
   {
     t1i = pms_lookuparray_data[t_count-2][0];
     R1i = pms_lookuparray_data[t_count-2][1];
     T1i = pms_lookuparray_data[t_count-2][2];
     t1ii = pms_lookuparray_data[t_count-1][0];
     R1ii = pms_lookuparray_data[t_count-1][1];
     T1ii = pms_lookuparray_data[t_count-1][2];
   }
   else
   {
     t1i = pms_lookuparray_data[t_count-2][0];
     R1i = pms_lookuparray_data[t_count-2][1];
     T1i = pms_lookuparray_data[t_count-2][2];
     T1ii = pms_lookuparray_data[t_count-1][2];
   }

   radius = ((t1ii-relageinyrs)/(t1ii-t1i))*R1i+((relageinyrs-t1i)/(t1ii-t1i))*R1ii;
   T_eff = ((t1ii-relageinyrs)/(t1ii-t1i))*T1i+((relageinyrs-t1i)/(t1ii-t1i))*T1ii;
```

```
  }
  else if (m_count == pms_number_of_masses)  // if higher than the highest mass in the lookup table
  {
    while (t1ii < relageinyrs) // get the ages, radii, and Temperature for the first mass
    {
      t1ii = pms_lookuparray_data[t_count][(pms_number_of_masses*3)-3];
      R1ii = pms_lookuparray_data[t_count][(pms_number_of_masses*3)-2];

      if (R1ii == 0.0) // if at the end of pre-main sequence, but not changed to MS yet
break;

      t_count += 1;
    }
    if (R1ii == 0.0)
    {
      t1i = pms_lookuparray_data[t_count-2][(pms_number_of_masses*3)-3];
      R1i = pms_lookuparray_data[t_count-2][(pms_number_of_masses*3)-2];
      T1i = pms_lookuparray_data[t_count-2][(pms_number_of_masses*3)-1];
      t1ii = pms_lookuparray_data[t_count-1][(pms_number_of_masses*3)-3];
      R1ii = pms_lookuparray_data[t_count-1][(pms_number_of_masses*3)-2];
      T1ii = pms_lookuparray_data[t_count-1][(pms_number_of_masses*3)-1];
    }
    else
    {
      t1i = pms_lookuparray_data[t_count-2][(pms_number_of_masses*3)-3];
      R1i = pms_lookuparray_data[t_count-2][(pms_number_of_masses*3)-2];
      T1i = pms_lookuparray_data[t_count-2][(pms_number_of_masses*3)-1];
      T1ii = pms_lookuparray_data[t_count-1][(pms_number_of_masses*3)-1];
    }

    radius = ((t1ii-relageinyrs)/(t1ii-t1i))*R1i+((relageinyrs-t1i)/(t1ii-t1i))*R1ii;
    T_eff = ((t1ii-relageinyrs)/(t1ii-t1i))*T1i+((relageinyrs-t1i)/(t1ii-t1i))*T1ii;
  }
  else
  {
    while (t1ii < relageinyrs) // get the ages, radii, and Temperature for the first mass
    {
      t1ii = pms_lookuparray_data[t_count][(m_count*3)-6];
      R1ii = pms_lookuparray_data[t_count][(m_count*3)-5];

      if (R1ii == 0.0) // if at the end of pre-main sequence, but not changed to MS yet
break;

      t_count += 1;
    }
    if (R1ii == 0.0)
    {
      t1i = pms_lookuparray_data[t_count-2][(m_count*3)-6];
      R1i = pms_lookuparray_data[t_count-2][(m_count*3)-5];
      T1i = pms_lookuparray_data[t_count-2][(m_count*3)-4];
      t1ii = pms_lookuparray_data[t_count-1][(m_count*3)-6];
      R1ii = pms_lookuparray_data[t_count-1][(m_count*3)-5];
      T1ii = pms_lookuparray_data[t_count-1][(m_count*3)-4];
    }
    else
    {
      t1i = pms_lookuparray_data[t_count-2][(m_count*3)-6];
      R1i = pms_lookuparray_data[t_count-2][(m_count*3)-5];
      T1i = pms_lookuparray_data[t_count-2][(m_count*3)-4];
      T1ii = pms_lookuparray_data[t_count-1][(m_count*3)-4];
    }

    t_count =0;

    while (t2ii < relageinyrs) // get the ages, radii, and Temperature for the second mass
    {
      t2ii = pms_lookuparray_data[t_count][(m_count*3)-3];
      R2ii = pms_lookuparray_data[t_count][(m_count*3)-2];

      if (R2ii == 0.0) // if at the end of pre-main sequence, but not changed to MS yet
break;

      t_count += 1;
    }
    if (R2ii == 0.0)
    {
      t2i = pms_lookuparray_data[t_count-2][(m_count*3)-3];
      R2i = pms_lookuparray_data[t_count-2][(m_count*3)-2];
      T2i = pms_lookuparray_data[t_count-2][(m_count*3)-1];
      t2ii = pms_lookuparray_data[t_count-1][(m_count*3)-3];
      R2ii = pms_lookuparray_data[t_count-1][(m_count*3)-2];
      T2ii = pms_lookuparray_data[t_count-1][(m_count*3)-1];
    }
    else
    {
      t2i = pms_lookuparray_data[t_count-2][(m_count*3)-3];
      R2i = pms_lookuparray_data[t_count-2][(m_count*3)-2];
      T2i = pms_lookuparray_data[t_count-2][(m_count*3)-1];
      T2ii = pms_lookuparray_data[t_count-1][(m_count*3)-1];
```

```
    }

    //2D linear interpolation
    R1 = ((t111-relage1myrs)/(t111-t11))*R11+((relage1myrs-t11)/(t111-t11))*R111;
    R2 = ((t211-relage1myrs)/(t211-t21))*R21+((relage1myrs-t21)/(t211-t21))*R211;
    radius = ((m2 - relative_mass)/(m2-m1))*R1 +  ((relative_mass-m1)/(m2-m1))*R2;
    T1 = ((t111-relage1myrs)/(t111-t11))*T11+((relage1myrs-t11)/(t111-t11))*T111;
    T2 = ((t211-relage1myrs)/(t211-t21))*T21+((relage1myrs-t21)/(t211-t21))*T211;
    T_eff = ((m2 - relative_mass)/(m2-m1))*T1 +  ((relative_mass-m1)/(m2-m1))*T2;

  }


  if (relative_age <= next_update_age) {

    effective_radius = Starlab::min(effective_radius, radius);
    luminosity = pow(effective_radius, 2.)* pow(T_eff/cnsts.parameters(Tsun), 4.);

  } else {

    // Pre-main sequence star's age exceeds deuterium core burning
// lifetime.

    if (relative_mass < cnsts.parameters(minimum_main_sequence)) {
      star_transformation_story(Brown_Dwarf);
      new brown_dwarf(*this);
      return;
    } else {
      star_transformation_story(Main_Sequence);
      new main_sequence(*this);
      return;
    }
  }


  update();
  stellar_wind(dt);
}

real pre_main_sequence::bolometric_correction() //use single_star perscription
{
  // temperature() is defined in Kelvin.
  // here we should use old 10^3K implementation
  // (SPZ+GN: 1 Oct 1998)
  real temp_in_kK = 0.001 * temperature();
  real bc;

  if (temp_in_kK < 4.195)
    bc = 2.5*log10((1.724e-7*pow(temp_in_kK,11.) + 1.925e-2)
   / (1. + 1.884e-9*pow(temp_in_kK,14.)));
  else if (temp_in_kK >=  4.195 &&
   temp_in_kK <= 10.89)
    bc = 2.5*log10((7.56e-2*pow(temp_in_kK,1.5))
   / (1. + 6.358e-5*pow(temp_in_kK, 4.5)));
  else
    bc = 2.5*log10((2728/pow(temp_in_kK,3.5) + 1.878e-2*temp_in_kK)
   /(1. + 5.362e-5*pow(temp_in_kK,3.5)));

  return bc;
}


// pre_main_sequence stars do not have a compact core.
// for convenience the core is set to a small value.
real pre_main_sequence::pre_main_sequence_core_mass()
{
    real m_core = 0.0001;
    //m_core = Starlab::max(core_mass, m_core);

    if (m_core > get_total_mass())
      m_core = get_total_mass();

    return m_core;
}

real pre_main_sequence::pre_main_sequence_core_radius()
{
    return Starlab::min(0.0001, radius);
}


// used for RLOF
star* pre_main_sequence::subtract_mass_from_donor(const real dt, real& mdot)
{

  mdot = relative_mass*dt/get_binary()->get_donor_timescale();
```

```
  mdot = mass_ratio_mdot_limit(mdot);


  if (mdot<=envelope_mass)
    envelope_mass -= mdot;
  else if (mdot>envelope_mass)
    envelope_mass = 0;

  return this;
}


star* pre_main_sequence::merge_elements(star* str) {

      star* merged_star = this;

      add_mass_to_core(str->get_core_mass());


      if (str->get_envelope_mass()>0)
         add_mass_to_accretor(str->get_envelope_mass());

      spec_type[Merger]=Merger;

      switch(str->get_element_type()) {
         case Pre_Main_Sequence:
   relative_age = 0; // back to the birthline
   break;
         case Main_Sequence:
   star_transformation_story(Main_Sequence);
   return dynamic_cast(star*, new main_sequence(*this));
 case Hyper_Giant:
         case Hertzsprung_Gap:
         case Sub_Giant:
         case Horizontal_Branch:
         case Super_Giant:
         case Carbon_Star:
         case Helium_Star:
         case Helium_Giant:
         case Carbon_Dwarf:
    case Oxygen_Dwarf:
         case Helium_Dwarf:
   cerr << "Merge PMS+Helium Dwarf"<< endl;
      if (relative_mass <
  cnsts.parameters(massive_star_mass_limit)) {
star_transformation_story(Hertzsprung_Gap);

// (GN+SPZ May  4 1999) should return new
//  merged_star = dynamic_cast(star*,
//  new hertzsprung_gap(*this));
//  dump(cerr, false);

// Chose relative_age to be next update age!
// otherwise sub_giants become unhappy.
cerr << "Merge PMS+wd"<<endl;
PRC(relative_age);PRC(next_update_age);

return dynamic_cast(star*, new hertzsprung_gap(*this));
      }
      else {
star_transformation_story(Hyper_Giant);
//  merged_star = dynamic_cast(star*,
//  new wolf_rayet(*this));
return dynamic_cast(star*, new hyper_giant(*this));
      }
         case Thorn_Zytkow :
 case Xray_Pulsar:
         case Radio_Pulsar:
         case Neutron_Star :
         case Black_Hole   :
              star_transformation_story(Thorn_Zytkow);
      return dynamic_cast(star*, new thorne_zytkow(*this));
         default:    instantaneous_element();
      }

      return merged_star;

}


// Star is rejuvenated by accretion.
void pre_main_sequence::adjust_accretor_age(const real mdot,
const bool rejuvenate=true) {


      real m_rel_new;
      real m_tot_new = get_total_mass() + mdot;
      if (m_tot_new>relative_mass)
         m_rel_new = m_tot_new;
```

```
    else m_rel_new = relative_mass;

    real t_pms_old = pre_main_sequence_time();
    real t_pms_new = pre_main_sequence_time(m_rel_new);

    relative_age *= (t_pms_new/t_pms_old);

    // next_update_age should not be reset here.
    // is done in add_mass_to_accreter, where also relative_mass
    // is updated (SPZ+CM: 1 Oct 1998)
    // next_update_age = t_ms_new;

}


// Low-mass main-sequence donor lifetimes are expanded by
// reducing relative_mass
// (SPZ+CM:25 Sep 1998)
void pre_main_sequence::adjust_donor_age(const real mdot) {

     real m_rel_new = get_relative_mass() - mdot;
     relative_age *= pre_main_sequence_time(m_rel_new)
          / pre_main_sequence_time();

}


// Adiabatic response function for main_sequence star.
// Used for determining mass_transfer_timescale.
// Increasing zeta stabilizes binary.
real pre_main_sequence::zeta_adiabatic() {

  // same as super giant (both convective)

  real x = core_mass/get_total_mass();
  real a = -0.220823;
  real b = -2.84699;
  real c = 32.0344;
  real d = -75.6843;
  real e = 57.8109;

  real z = a + b*x + c*x*x + d*x*x*x + e*x*x*x*x;

  return z;

}

// Thermal response function for pre_main_sequence star.
// Used for determining mass_transfer_timescale.
// (SPZ+CM: 1 Oct 1998)
real pre_main_sequence::zeta_thermal() {

  real z = 0; // from super_giant

  return z;

}

star* pre_main_sequence::reduce_mass(const real mdot) {

     if (envelope_mass<=mdot)
        envelope_mass = 0;
     else
envelope_mass -= mdot;

     return this;
   }

void pre_main_sequence::adjust_next_update_age() {

// (CM+SPZ May  4 1999) last update age is time of previous type change
     last_update_age = 0;
     next_update_age = pre_main_sequence_time(relative_mass);
   }

void pre_main_sequence::detect_spectral_features() {

     single_star::detect_spectral_features();

     if (accreted_mass>=cnsts.parameters(B_emission_star_mass_limit))
spec_type[Emission]=Emission;
     if (get_relative_mass() > turn_off_mass(current_time)
                 * (1+cnsts.parameters(Blue_straggler_mass_limit)))
   spec_type[Blue_Straggler]=Blue_Straggler;
   }

#if 0
real pre_main_sequence::stellar_radius(const real mass, const real age)
{
```

```
  real r_pms;
  real relageinyrs = age*1.0E6;
  real T_eff;

  int m_count = 0;
  int t_count = 0;
  real m2 = 0.00;
  real t1i = 0.00;
  real R1i = 0.00;
  real T1i = 0.00;
  real t2i = 0.00;
  real R2i = 0.00;
  real T2i = 0.00;
  real m1, t1i, t2i, R1i, R2i, T1i, T2i, R1, R2, T1, T2;
  char buffer[2000];
  char filename[50];


  while (m2 < relative_mass && m_count < pms_number_of_masses) //know the mass range we're dealing with
  {
    m1 = m2;
    m2 = pms_lookuparray_masses[m_count];
    m_count += 1;
  }


  if (m1 == 0.0) // if less than lowest mass in the lookup table
  {

    while (t1i < relageinyrs) // get the ages, radii, and Temperature for the first mass
    {
      t1i = pms_lookuparray_data[t_count][0];
      R1i = pms_lookuparray_data[t_count][1];

      if (R1i == 0.0) // if at the end of pre-main sequence, but not changed to MS yet
break;

      t_count += 1;
    }
    if (R1i == 0.0)
    {
      t1i = pms_lookuparray_data[t_count-2][0];
      R1i = pms_lookuparray_data[t_count-2][1];
      t1i = pms_lookuparray_data[t_count-1][0];
      R1i = pms_lookuparray_data[t_count-1][1];
    }
    else
    {
      t1i = pms_lookuparray_data[t_count-2][0];
      R1i = pms_lookuparray_data[t_count-2][1];
    }

    radius = ((t1i-relageinyrs)/(t1i-t1i))*R1i+((relageinyrs-t1i)/(t1i-t1i))*R1i;
//     if (isnan(radius))
//       cerr << "here R1" << R1 << " R2 " << R2<<" radius " << radius<< endl;
  }
  else if (m_count == pms_number_of_masses)  // if higher than the highest mass in the lookup table
  {
    while (t1i < relageinyrs) // get the ages, radii, and Temperature for the first mass
    {
      t1i = pms_lookuparray_data[t_count][(pms_number_of_masses*3)-3];
      R1i = pms_lookuparray_data[t_count][(pms_number_of_masses*3)-2];

      if (R1i == 0.0) // if at the end of pre-main sequence, but not changed to MS yet
break;

      t_count += 1;
    }
    if (R1i == 0.0)
    {
      t1i = pms_lookuparray_data[t_count-2][(pms_number_of_masses*3)-3];
      R1i = pms_lookuparray_data[t_count-2][(pms_number_of_masses*3)-2];
      t1i = pms_lookuparray_data[t_count-1][(pms_number_of_masses*3)-3];
      R1i = pms_lookuparray_data[t_count-1][(pms_number_of_masses*3)-2];
    }
    else
    {
      t1i = pms_lookuparray_data[t_count-2][(pms_number_of_masses*3)-3];
      R1i = pms_lookuparray_data[t_count-2][(pms_number_of_masses*3)-2];
    }

    radius = ((t1i-relageinyrs)/(t1i-t1i))*R1i+((relageinyrs-t1i)/(t1i-t1i))*R1i;
//     if (isnan(radius))
//       cerr << "here R1" << R1 << " R2 " << R2<<" radius " << radius<< endl;
  }
  else
  {
    while (t1i < relageinyrs) // get the ages, radii, and Temperature for the first mass
    {
```

```
      t1ii = pms_lookuparray_data[t_count][(m_count*3)-6];
      R1ii = pms_lookuparray_data[t_count][(m_count*3)-5];

      if (R1ii == 0.0) // if at the end of pre-main sequence, but not changed to MS yet
break;

      t_count += 1;
    }
    if (R1ii == 0.0)
    {
      t1i = pms_lookuparray_data[t_count-2][(m_count*3)-6];
      R1i = pms_lookuparray_data[t_count-2][(m_count*3)-5];
      t1ii = pms_lookuparray_data[t_count-1][(m_count*3)-6];
      R1ii = pms_lookuparray_data[t_count-1][(m_count*3)-5];
    }
    else
    {
      t1i = pms_lookuparray_data[t_count-2][(m_count*3)-6];
      R1i = pms_lookuparray_data[t_count-2][(m_count*3)-5];
    }

    t_count =0;

    while (t2ii < relageinyrs) // get the ages, radii, and Temperature for the second mass
    {
      t2ii = pms_lookuparray_data[t_count][(m_count*3)-3];
      R2ii = pms_lookuparray_data[t_count][(m_count*3)-2];

      if (R2ii == 0.0) // if at the end of pre-main sequence, but not changed to MS yet
break;

      t_count += 1;
    }
    if (R2ii == 0.0)
    {
      t2i = pms_lookuparray_data[t_count-2][(m_count*3)-3];
      R2i = pms_lookuparray_data[t_count-2][(m_count*3)-2];
      t2ii = pms_lookuparray_data[t_count-1][(m_count*3)-3];
      R2ii = pms_lookuparray_data[t_count-1][(m_count*3)-2];
    }
    else
    {
      t2i = pms_lookuparray_data[t_count-2][(m_count*3)-3];
      R2i = pms_lookuparray_data[t_count-2][(m_count*3)-2];
    }

    //2D linear interpolation
    R1 = ((t1ii-relageinyrs)/(t1ii-t1i))*R1i+((relageinyrs-t1i)/(t1ii-t1i))*R1ii;
    R2 = ((t2ii-relageinyrs)/(t2ii-t2i))*R2i+((relageinyrs-t2i)/(t2ii-t2i))*R2ii;
    radius = ((m2 - relative_mass)/(m2-m1))*R1 +  ((relative_mass-m1)/(m2-m1))*R2;

  }
  r_pms = radius;
  return r_pms;
}
#endif

real pre_main_sequence::gyration_radius_sq() {

  return ctsts.parameters(convective_star_gyration_radius_sq);

}
```

A more minor addition to the code was made in the add star module. It was necessary to initialise the lookup table there since it is called at the beginning of the kira command.

```
       //=====================================================//    -\|/-
      //  __  _____        ___                   ___      //      /|\ -
     // /       |      .    |   \  |        .     |   \   //         -\|/-
    //  \__    |     / \   |___/  |       / \    |___/  //           /|\ -
   //      \   |    /___\  |   \  |      /___\   |   \ // -\|/-
  //    ___/   |   /     \ |    \ |____ /     \  |___/ //  /|\ -
 //                                                  //          -\|/-
//==================================================//            /|\ -

//// addstar:  Add physical stellar data to existing node structure.
////
//// Options:     -c    comment to put in the starbase log structure [none]
////              -M    mass scaling (system mass unit, in solar units)
////              -Q    virial ratio (used if time scaling is omitted) [0.5]
////              -R    dynamical size scaling for stars
////                        (system length unit, in parsecs)
////              -s    initial types of stars [main_sequence]
////              -T    dynamical time scaling  (system time unit, in Myr)
////              -t    stellar age (in millions of years) [0]
////
//++ Notes:
//++  Command-line arguments overrule parameters from input snapshot.
//++
//++  If no time unit is explicitly specified it is derived
//++  from the other units.
//++
//++  The mass of the system can be set independently from the total
//++  mass of the input N-body system (e.g. when modeling a large
//++  cluster by a smaller N-body system).
//++
//++  Should run makemass first to establish stellar masses.
//++
//++ Example of usage:
//++  makenode -n 10 | makemass -u 100 -l 10 | add_star -T 1
//++
//++ See also:  mkmass
//++            mknode
//++
//// Version 1.0: Apr 1993   Piet Hut, Steve McMillan, Jun Makino
//// Version 2.0: May 1993   Simon Portegies Zwart
//                           spz@grape.c.u-tokyo.ac.jp

#include "single_star.h"
extern real pms_lookuparray_data [10000][10000];
extern real pms_lookuparray_masses [10000];
extern int pms_number_of_masses;
extern int pms_status;

#ifndef TOOLBOX

//-----------------------------------------------------------------------------
//  addstar  -- for all particles, add a star part using "new star()".
//-----------------------------------------------------------------------------

void  addstar(node * b, real t_current, stellar_type type, bool verbose)
{
  char pms_lut_fname[60];
  char test[10];
  char buffer[2000];
  int t_count = 0;

  if (pms_status == 0)
  {
    ifstream pms_lookuptable;
    strcpy(pms_lut_fname, getenv("STARLAB_PATH") );
    strcat(pms_lut_fname, "/lookuptables/PMSLOOKUP");
    pms_lookuptable.open(pms_lut_fname);
    if (! pms_lookuptable.is_open())
    {
      cerr << "No pre-main sequence lookup table.  Ok if you're just doing ms, but if not,";
      cerr << " it needs to be called $STARLAB_PATH/lookuptables/PMSLOOKUP"<<endl;
    }
    else
    {

      pms_lookuptable.getline(buffer, 2000); // bypass comment line
      do
```

```
      {
pms_lookuptable>>test;
if (strcmp(test, "\n") == 0 || strcmp(test, "t") == 0)
{
  break;
}
else
{
  pms_lookuparray_masses[pms_number_of_masses] = atof(test);
  pms_number_of_masses += 1;
}
      }
      while (strcmp(test,"\n") != 0); //know the mass range we're dealing with
      pms_lookuptable.getline(buffer, 2000); // bypass label line

      while (! pms_lookuptable.eof() )
      {
for (int i = 0; i<pms_number_of_masses; i++)
{
  pms_lookuptable>>pms_lookuparray_data[t_count][i*3];
  pms_lookuptable>>pms_lookuparray_data[t_count][(i*3)+1];
  pms_lookuptable>>pms_lookuparray_data[t_count][(i*3)+2];
}
t_count += 1;
      }
      for (int i = 0; i<pms_number_of_masses; i++)
      {
pms_lookuparray_data[t_count][i*3] = 0.0;
pms_lookuparray_data[t_count][(i*3)+1] = 0.0;
pms_lookuparray_data[t_count][(i*3)+2] = 0.0;
      }

      pms_status = 1;
      pms_lookuptable.close();
    }
  }


  if (b->get_eldest_daughter() != NULL) {

real m_min = VERY_LARGE_NUMBER;
real m_max = -VERY_LARGE_NUMBER;
real m_av = 0;
int  n_star = 0;

for_all_daughters(node, b, bi) {
    addstar(bi, t_current, type, verbose);

    if (verbose) {
real m = bi->get_starbase()
  ->conv_m_dyn_to_star(bi->get_mass());
m_min = Starlab::min(m_min, m);
m_max = Starlab::max(m_max, m);
m_av += m;
n_star++;
    }
}

if (verbose) {
    if (n_star > 0) m_av /= n_star;
}

    } else {


// MAS is the default return value of get_element_type, in
// cases where only a starbase (but no star) exists.

if (b->get_starbase()->get_element_type() != MAS) return;

int id = b->get_index();
real t_cur=0, t_rel=0, m_rel=1, m_ar.=0, m_core=0.01;
real T_eff, L_eff;
real p_rct=0, b_fld=0;
real m_tot;


// Create a (single) star part, using the information obtained from
// the star story read in by get_node(), or created from scratch.

stellar_type local_type = MAS;
starbase * old_starbase = b->get_starbase();
story * s = old_starbase->get_star_story();

real mco_core = 0;
extract_story_chapter(local_type, t_cur, t_rel,
      m_rel, m_ar., m_core, mco_core, T_eff, L_eff,
      p_rct, b_fld, s);
```

```
m_tot = m_env+m_core;
old_starbase->set_star_story(NULL);
delete s;

if (local_type != NAS) { // Is the star properly defined?

    // Use the data extracted from the star story.  Note that the
    // input values of the type and t_rel arguments are IGNORED.


    type = local_type;

    single_star* new_star = new_single_star(type, id, t_cur,
    t_rel,
    m_rel, m_tot, m_core,
    mco_core,
    p_rot, b_fld, b);



} else { // No star story present, or at least no
      // proper definition for a single star.

    // Create a default star of the specified type.

    real t_cur=0, t_rel = 0, m_rel = 1, m_core = 0.01;
    real p_rot=0, b_fld=0;
    real m_tot;
    t_cur = t_current;

    starbase * old_starbase = b->get_starbase();
    id = b->get_index();
    m_rel = m_tot = b->get_starbase()
     ->conv_m_dyn_to_star(b->get_mass());

    stellar_type local_type = type;

    // Treat by dynamics pre-requested black holes
    if(getiq(b->get_log_story(), "black_hole")==1) {
      local_type = Black_Hole;
      m_core = m_tot;
      mco_core = m_tot;
    }
    else if(m_tot<cnsts.parameters(minimum_main_sequence)) {
      local_type = Brown_Dwarf;
      m_core = 0.01*m_tot;
      mco_core = 0;
    }

    single_star* new_star = new_single_star(local_type, id,
    t_cur, t_rel,
    m_rel, m_tot, m_core,
    mco_core,
    p_rot, b_fld, b);

}

// Should not be needed since new_single_star does the job.
// (SPZ:25 Nov 1998)
        // delete old_starbase;
    }
}

#else

main(int argc, char ** argv)
{
    int  c;
    bool  t_flag = FALSE;
    bool  M_flag = FALSE;
    bool  R_flag = FALSE;
    bool  T_flag = FALSE;
    bool  Q_flag = FALSE;
    bool  s_flag = FALSE;
    bool  c_flag = FALSE;
    real  m_tot = -1;
    real  r_vir = -1; // NB code length unit; may or may not
// actually be the virial radius
    real  t_vir = -1;
    real  t_rel = 0;
    real  q_vir = 0.5;
    real  T_start = 0;
    stellar_type type = Main_Sequence;
    char * star_type_string;

    char  *comment;
    extern char *poptarg;
    int  pgetopt(int, char **, char *);
    char * param_string = "M:R:Q:T:t:s:c:";
```

```
    check_help();

    while ((c = pgetopt(argc, argv, param_string)) != -1)
switch(c)
    {
    case 'M': M_flag = true;
      m_tot = atof(poptarg);
      break;
            case 'R': R_flag = TRUE;
                      r_vir = atof(poptarg);
                      break;
    case 'Q': Q_flag = TRUE;
              q_vir = atof(poptarg);
              break;
            case 'T': T_flag = TRUE;
                      t_vir = atof(poptarg);
                      break;
    case 't': T_flag = TRUE;
      t_rel = atof(poptarg);
      break;
            case 's': s_flag = true;
      star_type_string = poptarg;
                      type = extract_stellar_type_string(star_type_string);
    case 'c': c_flag = TRUE;
      comment = poptarg;
      break;
            case '?': params_to_usage(cerr, argv[0], param_string);
          get_help();
          exit(1);
    }

    node *b;
    b = get_node(cin);

    if (c_flag == TRUE)
      b->log_comment(comment);

    b->log_history(argc, argv);

    // See if any scaling parameters are specified in the input snapshot.

    real new_r_vir=-1, new_t_vir=-1, new_m_tot=-1;

    // System mass in solar units:
    real old_m_tot = b->get_starbase()->conv_m_dyn_to_star(1);

#define Rsun_pc 2.255e-8 // R_sun/1 parsec = 6.960e+10/3.086e+18;

    // Code length unit in parsecs:
    real old_r_vir = b->get_starbase()->conv_r_dyn_to_star(1) * Rsun_pc;

    // Code time unit in Myr:
    real old_t_vir = b->get_starbase()->conv_t_dyn_to_star(1);

    // Set new parameters from old ones or command line.

    if (old_m_tot > 0 && !M_flag)
new_m_tot = old_m_tot;
    else if (M_flag)
new_m_tot = m_tot;
    else
err_exit("addstar: No mass scaling available.");

    if (old_r_vir > 0 && !R_flag)
new_r_vir = old_r_vir;
    else
new_r_vir = r_vir;

    if (old_t_vir > 0 && !(T_flag || Q_flag))
new_t_vir = old_t_vir;
    else
new_t_vir = t_vir;

    // Try to derive quantities not yet known.

    bool check_consistent = true;

    if (new_t_vir <= 0) {
if (new_m_tot > 0 && new_r_vir > 0) {

    // If no time unit is explicitly specified, derive it from
    // the other units.

    // Standard (N-body / Heggie & Mathieu) time unit, in Myr:

    new_t_vir = 21.0 * sqrt(q_vir/new_m_tot) * pow(new_r_vir, 1.5);
    check_consistent = false;
}
```

```
else
    err_exit("addstar: Unable to set time scaling");
    }

    if (new_r_vir <= 0) {
if (new_m_tot > 0 && new_t_vir > 0) {

    // If no length unit is explicitly specified, derive it from
    // the other units.

    new_r_vir = pow(new_t_vir / (21.0 * sqrt(q_vir/new_m_tot)), 2.0/3);
    check_consistent = false;
}
else
    err_exit("addstar: Unable to set radius scaling");
    }

    if (check_consistent) {

// Check that the various units are consistent.

real tv = 21.0 * sqrt(q_vir*pow(new_r_vir, 3)/new_m_tot);
if (abs(1-new_t_vir/tv) > 1.e-4) {
    cerr << "Warning: inconsistent units: ";
    PRC(new_t_vir);
    PRL(tv);
}
    }

    // Finally, set the scalings and create star parts.

    if (old_m_tot > 0 && !M_flag) {

// Existing dynamical masses are to be interpreted as
// solar units.  Set scaling accordingly before setting
// up star parts.

b->get_starbase()->set_stellar_evolution_scaling(1,1,1);
addstar(b, T_start, type, true);

// New scaling assumes that total dynamical mass will be
// rescaled to 1.

b->get_starbase()->set_stellar_evolution_scaling(new_m_tot,
 new_r_vir,
 new_t_vir);
    }
    else {

// New scaling assumes that total dynamical mass will be
// rescaled to 1.

b->get_starbase()->set_stellar_evolution_scaling(new_m_tot,
 new_r_vir,
 new_t_vir);
addstar(b, T_start, type, false);
    }


#if 0
    real m_sum = 0;
    for_all_daughters(node, b, bi) {
      m_sum += bi->get_mass();
    }
    b->set_mass(m_sum);

    real old_mtot = b->get_starbase()->conv_m_dyn_to_star(1);
    if(old_mtot!=m_sum) {
      real old_r_vir= b->get_starbase()->conv_r_star_to_dyn(1);
      real old_t_vir= b->get_starbase()->conv_t_star_to_dyn(1);
      b->get_starbase()->set_stellar_evolution_scaling(m_sum,
       old_r_vir,
       old_t_vir);
    }
#endif

    put_node(cout, *b);
    delete b;
}

#endif

/* endof: addstar.c */
```