Mem. S.A.It. Vol. 75, 688
© SAIt 2004

Memorie della

Accelerating stellar evolution

R. P. Church and C. A. Tout

University of Cambridge Institute of Astronomy, The Observatories, Madingley Road, Cambridge. CB3 0HA. England. e-mail: `rpc25@ast.cam.ac.uk`

Abstract. In order to improve the stellar physics in N-body models of clusters a rapid stellar evolution code is being developed from the Eggleton STARS code. We investigate the effect of timestep control and meshpoint number on code runtime and discuss treatments for some difficult phases of stellar evolution. We then briefly describe work in progress on eccentric white dwarf - neutron star binaries with the rapid code.

1. Introduction

Stellar clusters are important systems for testing predictions of stellar evolution theory. The high densities found in the centres of clusters and the presence of a primordial binary population leads to an enhanced probability of close stellar encounters and formation of interacting systems, which provide a significant challenge to stellar evolution codes. Because clusters are thought to form from a single cloud of gas their stars are all approximately the same age and have the same initial composition. This allows us to attempt to synthesise the population of stars in the cluster and thereby measure its age and model its population of unusual stellar objects.

There are two principal approaches to the production of models of globular clusters. A synthetic stellar evolution code allows us to evolve sufficient stars to produce a simulated colour-magnitude diagram for the cluster and also to model chemical enrichment through nucleosynthesis in the stellar population (e.g. Fenner et al. 2004). This approach, however, does not take into account dynamical interactions between stars. In contrast N-body simulations allow us to follow the stars as particles including ejection from the cluster and the formation and breakup of bound systems but not their evolution. The problem with these approaches is that the two are interlinked – mass-loss and changes in the radius of stars affects cluster dynamics whilst encounters and collisions change the stellar evolution. Thus to fully model a globular cluster it is necessary to use a coupled code that models both stellar evolution and dynamics.

Previous attempts to couple N-body and stellar evolution codes have used synthetic stellar evolution (Portegies Zwart et al. 1999; Hurley et al. 2001) to model stars sufficiently quickly to match the N-body computations. However modern computers are sufficiently fast that it is becoming feasible to use a full stellar evolution code in such a simulation. Such a code is superior to a synthetic code (which is based on fits to the output of full code) principally in its treatment of processes that take place on a thermal timescale

Send offprint requests to: R. Church
Correspondence to: Institute of Astronomy, University of Cambridge, Madingley Rd, Cambridge CB3 0HA UK

such as mass transfer from subgiants in the Hertzsprung gap. The rest of this paper outlines some work carried out to modify the Eggleton STARS code (Eggleton 1971; Pols et al. 1995) for such a purpose.

2. Accelerating Stellar Evolution Codes

Using dedicated hardware such as the GRAPE-6 system (Makino et al. 2003) and efficient numerical techniques for dealing with binary and multiple systems and other close encounters (Aarseth 2003) to compute calculations for N-body simulations it is now possible to evolve a cluster of 10^5 stars through the Hubble time in three months. This means that for a stellar evolution code to run in parallel with the N-body code it needs to be able to evolve a star through the Hubble time in approximately one minute on average. However, speed is not the only issue; the code needs to be entirely reliable. It must be able to evolve *any* single or binary stellar system without breaking down. In practice this is a much greater challenge than the speed-up, which can be accomplished mostly through utilisation of sufficiently fast hardware.

2.1. Timestep Control

One of the challenges for stellar evolution codes is the extremely large range of timescales on which a star evolves. Shell flashes on the surface of a forming white dwarf happen in a few years whilst the cooling and crystallisation of the same white dwarf occurs on a timescale of around 10^7 years. It is necessary for a code to choose a useful timestep at each iteration of the model as no single timestep is suitable for all situations. If the timestep that we attempt to use is too long it may not prove possible to converge a physical model for the star or a model may converge but fail to adequately resolve some aspect of the physics. If the timestep is too short then many timesteps are required and the code runs slowly.

The STARS code implements timestep control in an ad hoc manner. The current and next timesteps, $\mathrm{d}t_j$ and $\mathrm{d}t_{j+1}$ are related by

$$\mathrm{d}t_{j+1} = \mathrm{d}t_j \times \frac{\Delta}{\sum_i |\delta X_i|} \qquad (1)$$

where X_i are the model variables and Δ is a constant for a given run (usually between 0.5 and 5). As X_i are logarithmic ($\log r$, $\log m$, etc.; the luminosity is excluded from the sum) or mass fractions the changes δX_i are all of order unity, and so summing them is less arbitrary than it might at first appear. This formula works reasonably well; we find that $\Delta = 1$ is sufficient for almost all evolution, including the main sequence, red-giant branch (RGB), core helium burning, asymptotic giant branch (AGB) and white dwarf phases. However this value may not always be optimally fast. For a rapid code it is desirable for us to choose a timestep that is as close to optimal as possible to avoid unnecessarily slow progress in phases where the model changes quite rapidly (e.g. the AGB) and to avoid instabilities in some phases of evolution (e.g. the pre-main sequence). Phases of evolution where there is mass transfer also require smaller timesteps and it is desirable to reduce the timestep just prior to the onset of mass transfer to avoid instabilities. These issues all point towards the desirability of us developing an improved method of timestep control. Possibilities include retention of the present approach but varying Δ with the evolutionary phase of the system, use of a different metric to calculate the next timestep or implicit calculation of the timestep, although the latter has proved difficult to implement successfully in the past (Eggleton, private communication).

2.2. Number of Meshpoints

The STARS code allows us to vary the number of meshpoints used between runs, although this is typically kept constant throughout a single run. With fewer meshpoints the code performs fewer calculations at each timestep and thus, in general, runs faster. Models with fewer meshpoints also use less memory and disc space – the disc space required to store a large number of models can be significant even though the

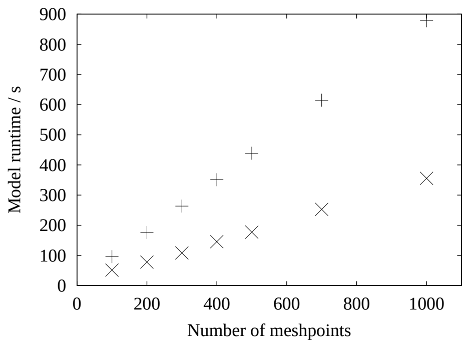

Fig. 1. The effect of varying the number of meshpoints on the STARS code runtime. A $1M_\odot$ star was evolved from the zero age main sequence to the end of the AGB without mass-loss. For each model Δ (see section 2.1) was scaled with the number of meshpoints so that the timestep on each iteration is the same. The two sets of points show different sets of timesteps.

memory required to store a single model generally is not.

Figure 1 shows how the number of meshpoints affects code runtime. In general we find that the runtime varies linearly with the number of meshpoints. Simple profile analysis suggests that this is because the code spends the majority of its time evaluating the equation of state, which is computed once per meshpoint. We could therefore accelerate the code by interpolation of the equation of state within precompiled tables instead of the current explicit evaluation, although this would lead to some loss of accuracy and thermodynamic consistency.

A reduction in the number of meshpoints usually increases execution speed but there are a few caveats. The solutions produced are less detailed and may not have sufficient resolution to model some aspect of the star's physics (e.g. thermal pulses on the AGB). This can reduce code stability and force it to take shorter timesteps. It may be possible to ameliorate some of these problems by adjustments to the mesh spacing function to distribute the meshpoints more usefully throughout the model.

3. Difficult Phases of Evolution

Difficult phases of evolution are those where the model changes very rapidly with time. The Henyey method (Henyey et al. 1964) expresses the equations of stellar evolution in matrix form and solves them by a relaxation method. This assumes that a stellar model can be obtained by making small changes to the model at the previous timestep. For steady evolution this is a good assumption. However when the star is evolving very rapidly it requires the code to take small timesteps in order to converge models to the required degree of accuracy causing numerical instabilities usually associated with the finite precision of numerical derivatives. Such codes also rely on the assumption that the star is in hydrostatic equilibrium. This is usually a good assumption because the dynamical timescale for a star is smaller than either the thermal or nuclear timescales by several orders of magnitude. However, during very rapid phases of evolution, the star may change on a timescale of the order of the dynamical timescale, at which point this assumption becomes invalid. Three examples of difficult phases of evolution and how we treat them in the rapid code are given below.

3.1. The Helium Flash

The helium flash is the ignition of helium under degenerate conditions in the cores of low-mass stars($M \lesssim 2.3M_\odot$) . Helium burning proceeds via the triple-alpha reaction:

$$^4\mathrm{He} + ^4\mathrm{He} \rightleftharpoons ^8\mathrm{Be}^*$$

$$^8\mathrm{Be}^* + ^4\mathrm{He} \rightleftharpoons ^{12}\mathrm{C}^*$$

$$^{12}\mathrm{C}^* \rightarrow ^{12}\mathrm{C} + \gamma$$

Being a three-body reaction – three ^{4}He nuclei have to come together within the lifetime of the unstable ^{8}Be* particle – its rate is extremely temperature-sensitive. Usually the increased energy generation caused by an increased burning rate leads to expansion, which then reduces the density and temperature of the burning material and thus the reaction rate, providing thermostatic control. However in degenerate material the equation of state is such that the pressure depends only on the density, so the material does not expand with increasing temperature. As the rate of the triple-alpha reaction is strongly temperature dependent this leads to a runaway reaction, with the energy generation rate changing on a dynamical timescale. This makes helium flash models very difficult to evolve.

We avoid the helium flash by pseudo-evolution - a process whereby stellar models are altered in a manner that would not have happened in normal circumstances (e.g. we can implement only hydrogen burning reactions) but such that they are still valid solutions to the equations of stellar structure. To pseudo-evolve through the helium flash we obtain a model of a higher-mass star that has just ignited helium non-degenerately remove mass from its surface until it is the same mass as the target star (the star that we are taking through the helium flash). Then we burn hydrogen until the core reaches the correct mass. Finally we reset the chemical compositions in the envelope to the values they had in the last target star model.

This process is not physically rigorous: it assumes, for instance, that the compositions of the stellar envelope are not affected by the helium flash. It is plausible that the large energy pulse produced by degenerate ignition could drive convective mixing, although detailed modelling suggests that this is only the case for quite a limited range of circumstances (Brown et al. 2001; Cassisi et al. 2003). However as an approach to the problem of evolution through a difficult and slow process it is useful because it allows us to carry out calculations that would otherwise be impossible.

3.2. The Thermally Pulsing Asymptotic Giant Branch (TP-AGB)

Thermal pulses on the AGB are caused by the Härm-Schwarzschild instability (Schwarzschild & Härm 1965), which occurs when hydrogen-burning and helium-burning shells are separated by a thin layer of material. Thermal pulses and the difficult task they pose for stellar evolution codes are well described elsewhere in this publication (see e.g. Stancliffe et al. in this volume). Given the extreme difficulties surrounding the TP-AGB it is not practical to attempt to evolve through thermal pulses in a rapid code. Thus a sufficiently small number of meshpoints are used with sufficiently long timesteps that thermal pulses are not resolved.

There are three significant inaccuracies associated with not resolving thermal pulses. The process of third dredge-up, where the convective envelope penetrates through the extinguished hydrogen burning shell and dredges up processed material from the intershell region to the core both reduces the rate of core growth and chemically enriches the envelope. Luminosity peaks and opacity enhancements due to thermal pulses and third dredge-ups may contribute to the strong winds, which remove the envelope, truncating the star's life. Whilst the composition changes are not especially significant for dynamical models the changes in lifetime and structure may be. However it will hopefully be possible to emulate these effects via an alternative method, such as using reduced reaction rates (but retaining the normal rate of energy generation to) to model the slow rate of core growth.

3.3. Dynamical Timescale Mass Transfer

In an interacting binary, the accretor[1] cannot gain mass on a timescale much shorter than its thermal timescale, as the accreted mass needs to cool in order to settle on to the stellar surface. This means that if mass is transferred more rapidly than this limiting rate it builds up in a cloud or disc around the accretor. If this cloud then fills the accretor's Roche Lobe we end up with the envelope of the donor forming a diffuse cloud around the two stars, a "common envelope". It is believed that the stars then spiral together, by some frictional process that is poorly understood, and the orbital energy

[1] In this paper I use the following convention for referring to stars in a binary system: The accretor is the star gaining transferred mass, the donor is the star losing it, the primary is the star that had the higher mass *at the beginning of the simulation* and the secondary is the star that had the lower mass.

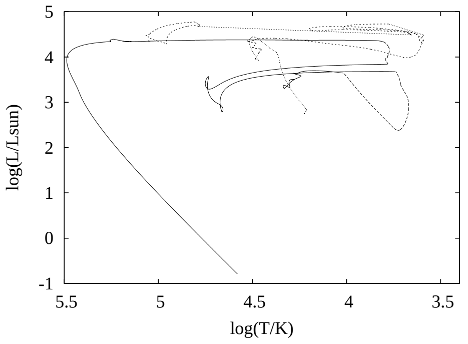

Fig. 2. A typical HR diagram from a WD-NS binary ($M_1 = 7.25\,M_\odot$, $M_2 = 5.0\,M_\odot$, $P_\mathrm{i} = 11.7$ d. The solid and long dashes are the primary during normal evolution and mass transfer respectively. The short dashes and dots are the secondary. The loops in the primary track coming on to and back off steady core helium burning are a general feature, as are the wide loop replacing the RGB and the truncated AGB. Until the common envelope phase, (the long straight dotted line) the secondary behaves much like a single star of the mass it becomes after accretion.

thus released causes the common envelope to be ejected. This results in a planetary nebula surrounding a close binary system or, in the case where the cores of the two stars merge, a rapidly rotating giant.

Without a full hydrodynamic simulation of the system (and perhaps not even with one) it is impossible to model the detailed physics of common envelope evolution. Instead we use the empirical α prescription. The ratio of binding energy of the envelope to orbital energy change is given by

$$\alpha_\mathrm{CE} = \frac{\Delta E_\mathrm{bind}}{\Delta E_\mathrm{orb}} \qquad (2)$$

where the efficiency factor α is thought to lie in the range $0.1 < \alpha < 10$. Various factors affect α. For example radiative losses from the envelope could decrease it below unity whilst extra burning caused by accretion on to compact objects or magnetic reconnection could increase it. For more details see the review by Iben & Livio (1993). The implementation of common envelope evolution in the rapid code reduces the effective mass-loss rate to maintain numerical stability but assumes that no nuclear burning takes place during common envelope evolution as it is expected to happen on a timescale much shorter than the nuclear timescale of the star. Owing to this small timescale it is assumed that no material is accreted and that if both stars overflow their Roche Lobes during the process they merge.

4. An Application - Eccentric Compact Object Binaries

Binaries containing a white dwarf (WD) and a neutron star (NS) are generally found to be circular because although aspherical explosions in supernovae are thought to provide a kick to the nascent NS, tides driven by the newly formed neutron star in its low-mass main-sequence companion circularise the orbit. However two examples of eccentric WD-NS binaries are known, J1141-6545 and B2303+46. It is conjectured that circularisation has been avoided by the white dwarf forming first. Davies et al. (2002) suggested that such systems form in close binaries consisting of a primary with mass $5 < M/M_\odot < 7$ and a lower mass secondary. The primary evolves first and transfers mass via Roche Lobe overflow as it evolves up the RGB and later the AGB. This reduces its mass below that necessary for core-collapse and thus it leaves a white dwarf. The secondary then evolves up its RGB and loses most of its mass in a common envelope but still retains enough to form a neutron star via a core-collapse supernova. The velocity kick provided by the supernova produces a system with the required eccentric orbit. Previous attempts to model such systems have used synthetic formulae for stellar evolution. These have some significant shortcomings. Our calculations of detailed models using the rapid code has found that these synthetic models fail to accurately take into account the effect of mass transfer on the secondary star, which sometimes cannot absorb all the mass being transferred and thus forms a common envelope system prematurely. Some close systems undergo multiple phases of common envelope evolution and lose most of their mass, eventually forming WD-WD binaries and some systems merge. This work is still in progress but the formation pathway looks plausible. Figure 2 shows an HR diagram of one model that produces a WD-NS system.

5. Conclusions

A rapid full stellar evolution code that is sufficiently fast and reliable to integrate with an N-body dynamics code is a useful goal to work towards. Such a code is currently being developed and some details of ways to accelerate the code have been discussed. Automatic pseudo-evolution has been introduced to avoid some difficult phases and the approximations involved justified. Finally some work in progress on WD-NS binaries is outlined.

Acknowledgements. RPC is grateful to PPARC for a grant. CAT is grateful to Churchill College for a fellowship. RPC would like to thank the members of the IoA stellar evolution group, particularly R. Stancliffe and J. Eldridge, for many useful conversations, and all the attendants at the workshop who made helpful suggestions on the work presented here.

References

Aarseth, S. 2003, Gravitational N-Body Simulations: Tools and Algorithms (Cambridge University Press)
Brown, T. M., Sweigart, A. V., Lanz, T., Landsman, W. B., & Hubeny, I. 2001, ApJ, 562, 368
Cassisi, S., Schlattl, H., Salaris, M., & Weiss, A. 2003, ApJ, 582, L43
Davies, M. B., Ritter, H., & King, A. 2002, MNRAS, 335, 369
Eggleton, P. P. 1971, MNRAS, 151, 351
Fenner, Y., Campbell, S., Karakas, A. I., Lattanzio, J. C., & Gibson, B. K. 2004, MNRAS, 280
Henyey, L. G., Forbes, J. E., & Gould, N. L. 1964, ApJ, 139, 306
Hurley, J. R., Tout, C. A., Aarseth, S. J., & Pols, O. R. 2001, MNRAS, 323, 630
Iben, I. J. & Livio, M. 1993, PASP, 105, 1373
Makino, J., Fukushige, T., Koga, M., & Namura, K. 2003, PASJ, 55, 1163
Pols, O. R., Tout, C. A., Eggleton, P. P., & Han, Z. 1995, MNRAS, 274, 964
Portegies Zwart, S. F., Makino, J., McMillan, S. L. W., & Hut, P. 1999, A&A, 348, 117
Schwarzschild, M. & Härm, R. 1965, ApJ, 142, 855