Binary_c days, September 8th and 9th, 2015

Hoyle Committee Room, Institute of Astronomy, Madingley Road, Cambridge

Start time: 9.30am for 10am; Coffee 11am; Lunch at Churchill; Tea 3.30pm

Curry night on the 8th

Laptops can be connected to the screen; white boards available; internet via Eduroam

Participants

- Carlo Abate
- Anna Hourihane
- Jarrod Hurley
- Robert Izzard
- Denise Keller (in the cloud)
- Selma de Mink
- Coen Neijssel
- Matt North
- Fabian Schneider
- Christopher Tout
- Manos Zapartas
- Anna Żytkow

Programme

Tuesday 8th

History and reasoning

- Stellar evolution based on Jarrod Hurley's SSE/BSE code
- Full evolution vs "synthetic" evolution
- WTTS2 demo: *binary_c* vs *TWIN* (Eggleton's code)

Science in binary_c

- Single star evolution
  - SSE models
- Binary star evolution
  - BSE = SSE + binary algorithm
  - common envelope (+mass accretion)
  - Wind mass transfer
  - RLOF and mergers
  - Tides

  - Gravitational waves
  - Explosions (SNIa, novae)
- Nucleosynthesis
  - AGB
  - Massive stars
  - Supernovae, novae etc.
  - Binaries: e.g. thermohaline mixing

Recent science updates

- Contributions please! Suggestions:
  - Carlo: wind-RLOF, s-process at low metallicity
  - Selma: massive stellar structure, rotation, rejuvenation
  - Rob: Adaptive RLOF, tidal E_2, mass transfer algorithms
  - Fabian: mergers and blue stragglers
  - Denise: planetary nebulae

Binary_c code structure

- Original fork from *BSE*
  - C rewrite, extensive testing on "random" systems
  - Extended with nucleosynthesis, particularly AGB
- Modularization and directory structure
  - *src/*
    - *hrdiag* : stellar evolution
    - *claet* (=calc_lum_and_evol_time) : stellar timescales and luminosities used in the fitting formulae
    - *RLOF* : Roche-lobe overflow routines
    - *single_star_functions* : routines which involve single-star evolution
    - *binary_star_functions* : ditto for binary-star physics (mass transfer, tides etc.)
    - *nucsyn* : nucleosynthesis (Rob's PhD + follow-up papers)
    - *zfuncs* : fitting functions (based on *BSE*'s *zfuncs.f*)
    - *wind* : stellar wind routines
    - *maths* : mathematics, notably interpolate.c
  - *doc/*
    - Documentation goes here
  - *tbse*
    - Test script for a single system (single/binary star)
- Structures vs small arrays
  - instead of `x[2]`, `y[2]`, `z[2]` ... have `{x,y,z}[2]`
  - All data in `stardata`
  - Stars in `stardata->star[1]` and `stardata->star[2]`

  - copy stars easily and fast (`memcpy`)
  - `NUMBER_OF_STARS` stars: could be extended, some code assumes this is 2
- configure and Makefile
  - `./configure`
  - `make`
  - `make clean; make cleanall`
  - `make libbinary_c.so`
- concept of single, static executable
  - good for use on multiple machines of same (similar?) architecture, e.g. 64-bit Linux clusters
- compile-time flags
  - *binary_parameters.h*
    * default parameters, constants, limits etc.
    * `ADAPTIVE_RLOF`
    * `WRLOF`
    * logging
    * `NUCSYN`
  - *nucsyn_parameters.h*
    * read if `#define NUCSYN`
  - API
    * `#define BINARY_C_API` (in binary_parameters.h)
    * some obscure things are broken with the API (probably affects nobody)
- Shared library
- Optimization (setup for grid runs)
- Recent updates
  - *interpolate.c*
  - API (C and FORTRAN)

Wednesday 9th

Accessing binary_c

- SVN access
- tree model: trunk and branches
- mutual agreement

How to use *binary_c*

- Interactive tutorial
  - *tbse* for a single system
  - *binary_grid* script for many systems (see below)
- There's a comprehensive manual, but it might be a bit out of date

- Mailing lists
  - Announcements
    `https://groups.google.com/forum/?hl=en-GB#!forum/binary_c-nucsyn-announce`
  - Development
    `https://groups.google.com/forum/?hl=en-GB#!forum/binary_c-nucsyn-devel`
  - Should use these more!
- Online interface
  - `http://www.ast.cam.ac.uk/~rgi/cgi-bin/binary4.cgi`

The *binary_grid* software

- Perl (bi-directional pipe) grid
  - binary_c run in "batch mode"
  - Arguments sent to it (stdin) control the system parameters
  - Send a "go" to run the system
  - Output (stdout) is captured until "fin" received
  - Data stored in Perl hashes, native, fast.
  - Data is processed by Perl parsing function, statistics added up, processed, output (text and/or graphs)
  - All ASCII, no binary, hence cross-platform.
  - Threaded queue (Perl's `Thread::Queue`) for multi-CPU, usually memory bandwidth (not CPU) limited.
- Example grid script
  - *distribution_functions*
    - Initial mass(es), q, period/separation, eccentricity etc.
  - *spacing_functions*
    - how many points and where?
- Data extraction
  - On-the-fly vs post-processing
  - Hashes slow? Use MySQL (Fabian!), or CHI memory cache?
- Latest features
  - many speed optimisations
  - compressed buffer streams
  - better error handling
  - better handling of "failed" systems
- Visualisation (*WTTS2*)
  - demo!
  - Download from `http://www.ast.cam.ac.uk/~rgi/window.html`

Future of binary_c and friends

- Science that needs to be implemented
  - New stellar model grids?
  - Nucleosynthesis (new AGB models, extend M,Z range)
  - Rotation and mixing
  - Mass transfer (X-ray binaries, etc.)
  - Novae (efficiency)
  - SNeIa? (helium stars?)
  - Populations (field vs clusters)
  - Input distributions (IMF, etc.)
  - Cluster evolution (impact of dynamics on binaries)
  - More than two stars
  - ?
- Code updates
  - More modularization
  - Better timestepping
  - Set derivatives and apply them (like a real stellar code)
  - More than two stars (triples etc., see Hamers' work)
  - git instead of SVN? or something else?
- Visualisation
  - Update WTTS2 for 2D stars, more outreach opportunities (openGL?)
- Future projects
  - Galactic models / Galactic chemical evolution
    - GCE code using binary_grid already exists
    - needs debugging!
  - Updated nucleosynthesis
    - Carolyn Doherty's new AGB/superAGB grid
    - TZ objects
    - update SN yields
    - SNeIa (triples, wind accretion)
    - Novae (nucleosynthesis and WD accretion efficiency)
  - Critical mass ratios
  - Selection effects
    - Difficult!
    - No systematic way to do this (yet?)
  - Model verification pipeline
    - WTTS2 allows visual comparison
    - Again, no systematic testing procedure (yet?)