ComBinE - Manual

Binary population synthesis code

Matthias U. Kruckow

December 21, 2018

Contents

1 Introduction **1**

2 Run the code **1**

2.1 Set parameters 1

2.1.1 Parameter-file: `data.in` 1

2.1.2 Command line arguments 3

2.1.3 Change the parameters in the code 3

2.2 Input parameter description 4

2.2.1 Numerics 4

2.2.2 Output 5

2.2.3 Physics 5

2.3 Single run 9

2.4 Population synthesis 9

2.5 Output files 10

2.5.1 `data.out` 10

2.5.2 `hist` files 11

2.5.3 `dist` files 11

2.6 Stellar tables 12

3 Code structure **13**

3.1 Main structure 13

3.2 Constants 13

3.3 Structure definitions 14

3.3.1 `t_system` 15

3.3.2 `t_star` 15

3.3.3 `t_SN` 16

3.3.4 `t_HRD` 17

3.3.5 `t_hist` 17

3.4 Read stellar tables 18

3.5 Initial conditions 18

3.6 Evolutionary phases 18

3.6.1 Wind mass loss 19

3.6.2 Roche-lobe overflow 19

3.6.3 Common envelope 20

3.6.4 Supernova/planetary nebula 20

3.7 Analysing the binary 21

3.7.1 Integration tables 21

3.7.2 Galactic motion 21

3.8 Histograms 21

3.9 User interface 22

4 Warnings and errors **25**
4.1 Warning messages 27
4.2 Error messages 30

References **37**

List of Figures **39**

List of Tables **39**

1 Introduction

This code, ComBinE, is based on Starburst developed by Tauris & Voss in 2001, see Voss and Tauris (2003). The new version has the main goal to produce reliable event rates of mergers of stellar mass compact objects for gravitational wave detectors like LIGO. This new code is developed by Matthias Kruckow as part of a DFG-PhD project (Grant: TA 964/1-1) under supervision of Thomas Tauris.

2 Run the code

If the code is not compiled jet, you should use the makefile with the command `make ComBinE` in the `ComBinE` directory.

The easiest way to run the program is by simply typing `./ComBinE` in the main directory of ComBinE[1]. This will use some standard parameters for the run.

To run the code in a different directory you have to place a file called `tablelocation.txt` in the directory where you run the code. In this file the place of the tables[2] should be specified, the default place is `./tables/`. If you use a common directory to store the tables you should make sure that you have write access to this directory. If some integration tables, see section 3.7.1, are missing or not up to date they will be created by the code.

Note that the code will write some files in the current directory. They will be named always in the same way. So you should not run more than one simulation with ComBinE in one directory. Otherwise the output files will be overwritten, see section 2.5.

2.1 Set parameters

To change the parameters there are three ways possible. They will be described in the following sections.

2.1.1 Parameter-file: `data.in`

You can use a parameter-file called `data.in`. This has to be in the directory were you run the code. To read it in just run the command `./ComBinE -2`. An example `data.in` file looks like:

```
NUMBER OF PROCESSORS TO USE (<=0 AUTOMATIC): 1
SEED: 9080706050403020 10
NUMBER: 1000000
MAXIMUM PRIMARY MASS (MSUN): 100.0
MINIMUM PRIMARY MASS (MSUN): 4.0
MAXIMUM SECONDARY MASS (MSUN): 100.0
MINIMUM SECONDARY MASS (MSUN): 1.0
```

[1]All following commands and file references will assume the current directory to be the main directory of ComBinE.

[2]Including the stellar and integration tables, see sections 2.6 and 3.7.1.

```
MAXIMUM SEMI–MAJOR AXIS (RSUN): 10000.0
MINIMUM SEMI–MAJOR AXIS (RSUN): 2.0
WIND MASS LOSS DURING RLO (%): 20.0
RLO MASS EJECTION PARAMETER (%): 75.0
RLO RADIUS PARAMETER OF CIRCUMBINARY TORUS: 2.0
RLO MASS–FRACTION TO CIRCUMBINARY TORUS (%): 0.0
CE EFFICIENCY PARAMETER (%): 50.0
LAMBDA: 1.0
QLIMIT: 2.5
MHECE: −3.3
KICK VELOCITY (KM/S; <0 TAKE RANDOM VALUE): −1.0
LAMBDA INTERPOLATION FACTOR (0.0−1.0 INTERPOLATE; <0.0 OLD
    TABLES; >1.0 CONSTANT FAKTOR OF LAMBDA_G[=2.0 VIRIAL
    EQULIBRIUM]): 0.5
MOTION INTEGRATOR (=0 NO GALACTIC MOTION; =1 RUNGEKUTTA4): 1
GALACTIC POTENTIAL (=0 NO GALACTIC MOTION; =1 MW–POTENTIAL BY
    ALLEN&SANTILLAN 1991): 0
OUTPUT FORMAT (=M MASSIVE OUTPUT; =T TINY OUTPUT): M
INITIAL MASS FUNCTION (=1 SALPETER 1955; =2 KROUPA 2008): 1
INITIAL MASS–RATIO DISTRIBUTION (=1 KUIPER 1935; =2 FLAT; =3
    SANA+ 2012): 1
INITIAL SEMI–MAJOR AXIS DISTRIBUTION (=1 ABT 1983; =2 PERIOD–
    DISTRIBUTION KROUPA 2008; =3 SANA+ 2012): 1
INITIAL ECCENTRICITY DISTRIBUTION (=0 CIRCULAR; =1 THERMAL(
    HEGGIE 1975); =2 FLAT; =3 FLAT IN ANGULAR MOMENTUM): 0
INITIAL AGE DISTRIBUTION (=0 ALL AT ZAMS): 0
INITIAL METALLICITY DISTRIBUTION (=0 MW–METALLICITY; =–1 LMC–
    METALLICITY; =–2 SMC–METALLICITY; =–3 IZw18–METALLICITY): 0
INITIAL ROTATION DISTRIBUTION (=−1 SYNCHRONISED; =0 NON–
    ROTATING): 0
INITIAL POSITION IN THE GALAXY (=−1 SUN; =0 AT GALACTIC CENTER)
    : 0
INITIAL VELOCITY IN THE GALAXY (=−1 SUN; =0 AT REST): 0
INITIAL STELLAR DENSITY (=0 FIELD STAR): 0
```

For more information about the individual parameters see section 2.2. You should make sure, that your `data.in` file only uses ":" right before the value to read in and the order of the parameters stays the same. You can check that all parameters are read in correctly in the `data.out` file, see section 2.5.1.

2.1.2 Command line arguments

You can run the program with command line arguments. Every command line argument should be separated with a space. The first Argument specifies what kind of run is done.

-1 Run a single model with default parameters, where
a second argument sets the primary mass ($M_\odot$),
a third argument sets the secondary mass ($M_\odot$),
a fourth argument sets the semi-major axis ($R_\odot$)
and a fifth argument sets the random seed.
Example: `./ComBinE -1 50.0 40.0 300.0 -21` this calculates one system with a primary mass of 50 $M_\odot$, a secondary mass of 40 $M_\odot$ and a semi-major axis of 300 $R_\odot$, which corresponds to an orbital period of $\approx$ 63.6 d as initial values. If you specify less than five arguments the not specified ones are taken from the default values or specified in the user interface.

-2 Run with the parameters from `data.in` file, see section 2.1.1.

-3 Run a grid between minimum and maximum values of the primary mass, secondary mass and semi-major axis in log-scale. Those values can be changed in the user interface.

-4 Run a single model like in the old code StarBurst (Voss and Tauris, 2003), you can specify parameters like for the option `-1`.

-11 Run a single model in debug mode, you can specify parameters like for the option `-1`.

> 0 Run with default values, see section 2.2, and this and only first argument as random seed.

When ever no random seed is specified it is generated from the current time.

2.1.3 Change the parameters in the code

The last and least way is to change the default parameters in the code, recompile it and run it. This you should only do if you want to change the default values permanently. To do it a bit more suitable, you can modify the values in the desired functions within the user interface, see section 3.9.

2.2 Input parameter description

In this section all important parameters of the program will be described. The parameters are categorised if they belong to numerics, output specifications or to the physics. Within each section the parameters are in the same order as defined in the code. The bullets are the names of the variables in the code.

2.2.1 Numerics

`parallel` The number of parallel processes which will be used by the code. Its default value is 1. It can be specified in the `data.in` file in line 1. You should make sure that the value is smaller or equal to the number of available processors.

To create one process for each processor use a value smaller or equal to 0 and then the program will determine the number of processors. The speed gain with multiple processors is not linear. To create grids of input parameters you should use one processor per ComBinE run and start several threads of ComBinE in different directories yourself.

`seed` The random seed. If it is set to its default value 0 a random seed is generated from the system time. You can also specify the seed in line 2 in the `data.in` file. In this way you can rerun a simulation with the same order of random numbers which might be useful to recreate a single model out of a population synthesis run.

`number` The number of systems which will be calculated. It can be specified in the `data.in` file in line 3 and should be larger than 0 otherwise the code does nothing. Its maximum value is 9 223 372 036 854 775 807 which is the upper boundary of the `long` integer type. The time the program runs scales for large `number` values linearly where 1 000 000 000 corresponds to about 1 d of run time.

`accuracy` The relative accuracy to which the code will check the calculations to be unaffected by numerical uncertainties. Its default value is 10^{-10}. Be aware that lowering this values to close to the precision of `double` type may cause some numerical instabilities.

`galintegrator` The integrator routine to use for the motion in the galactic potential. It can be set in line 20 in the `data.in` file. The possible values are:

- `0` no motion in the galaxy
- `1` standard Runge-Kutta 4 integrator — default value

2.2.2 Output

`screen` It enables and disables the output of the evolution information of every calculated system. As default it is `true` for the first ten calculated systems which holds for single and population synthesis runs. If it is true at the end of the calculation of a system its evolution information is printed to the screen, but the intermediate outputs only start when it is `true`.

`output` It specifies the amount of the screen-output. The possible values are `M` for massive output and `T` for tiny output. It can be set in the `data.in` file in line 22. The massive output will give you all information about the evolution of the binary system at any phase, including variable names and units. The tiny output contains only the most important values in a compressed format.

`debug` This is only for debugging and will run the program in debug mode. There it will give a lot of information during the calculation and the full tracks of the stars used for the calculation at the end. The default value is `false`.

2.2.3 Physics

Initial Parameters

Mp, Mp_max, Mp_min, IMF The primary star is the initially more massive star. Its mass in $M_\odot$ will be taken from the given range between `Mp_max` and `Mp_min`. Their default values of 100 and 4 can be changed in lines 4 and 5 of the `data.in` file, respectively. `Mp` is the initial mass for a single system. Its default in a single run is `Mp_max` if not specified otherwise. In a population synthesis run the initial primary mass is taken randomly from a distribution function. This function is controlled by the variable `IMF`.
Possible `IMF`-values are (line 23 of the `data.in` file):

≤`-10`	user defined, see section 3.9	
`1`	Salpeter IMF (Salpeter, 1955; Scalo, 1986)	default value
`2`	canonical IMF (Kroupa, 2008)	

Ms, Ms_max, Ms_min, q, qdist The mass of the secondary star is determined by the primary mass and the mass ratio q. Nevertheless, in the `data.in` file you specify a range for the secondary mass in lines 6 and 7, respectively. Otherwise the default range is between 1 and Mp_max. This will be converted into a range for the mass ratio depending on the primary mass, such that $q \leq 1$ holds. `qdist` tells the program from which distribution the mass ratio should be taken.
Possible `qdist`-values are (line 24 of the `data.in` file):

≤`-10`	user defined, see section 3.9	
`1`	$f(q) = \frac{2}{(1+q)^2}$, see equation (4) in Kuiper (1935)	default value
`2`	flat: $f(q) = 1$	
`3`	$f(q) \propto q^{-0.1}$ (Sana et al., 2012)	

a, a_max, a_min, adist The semi-major axis range in $R_\odot$ is between 2 and 10000 as default or specified in lines 8 and 9 of the `data.in` file. The initial separation has a lower limit depending on the masses of the stars to create no contact systems initially.
Possible `adist`-values to determine the underlying initial orbital period distribution are (line 25 of the `data.in` file):

≤`-10`	user defined, see section 3.9	
`1`	flat in $\log(P)$, where P is the orbital period (Abt, 1983)	default value
`2`	period distribution from Kroupa (2008)	
`3`	$f(P) \propto P^{-0.55}$ (Sana et al., 2012)	

e, edist The eccentricity `e` is determined by `edist` with possible values (line 26 of the `data.in` file):

≤`-10`	user defined, see section 3.9

- `0` $e = 0$, all initially circularised — default value
- `1` $f(e) = 2 \cdot e$, thermal (Heggie, 1975)
- `2` flat: $f(e) = 1$
- `3` flat in orbital angular momentum L and constant orbital binding energy, so $e \propto \sqrt{1 - L^2}$

`t,` `tdist` The initial age is at ZAMS for all stars as default.
The possible distribution values are (line 27 of the `data.in` file):

- $\leq$`-10` user defined, see section 3.9
- `0` start at ZAMS — default value

`metall,` `Zdist` The metallicity is given by `Zdist` which depend on the available stellar tables (section 2.6) and can be (line 28 of the `data.in` file):

- $\leq$`-10` user defined, see section 3.9
- `0` Milky Way metallicity ($Z = 0.0088$) — default value
- `-1` LMC metallicity ($Z = 0.0047$)
- `-2` SMC metallicity ($Z = 0.0021$)
- `-3` IZw18 metallicity ($Z = 0.0002$)

`omegap,omegas,` `rotdist` The spin of the stars ω is determined by the value of `rotdist` and depend on the available stellar tables (section 2.6).
Its possible values are (line 29 of the `data.in` file):

- $\leq$`-10` user defined, see section 3.9
- `-1` synchronised (no effect as long as there are no rotating stellar tables provided)
- `0` non-rotating — default value

`x,y,z,` `Rdist` The galactic position in Cartesian coordinates depends on `Rdist`.
Possible `Rdist`-values are (line 30 of the `data.in` file):

- $\leq$`-10` user defined, see section 3.9
- `-1` solar position in the Milky Way $\vec{R}_{\text{sun}}^{\text{T}} = (-8.5, 0.0, 0.0)$ kpc
- `0` at galactic center — default value
- `1` random in the disk potential of Allen and Santillan (1991)

`vx,vy,vz,` `Vdist` The three dimensional velocity of the binary system in the galaxy is determined by `Vdist` as (line 31 of the `data.in` file):

- $\leq$`-10` user defined, see section 3.9
- `-1` solar velocity in the Milky Way $\vec{V}_{\text{sun}}^{\text{T}} = (10.0, 235.0, 7.0)$ km/s
- `0` at rest — default value

1 perpendicular to the acceleration in the galactic potential to be balanced by the centrifugal force

rhostar, rhodist To estimate the encounter rate with other stars the number density of stars around a system is used. Its unit is pc^{-3}. Stellar encounters are not yet implemented.
The implemented distributions are (line 32 of the `data.in` file):

≤-10 user defined, see section 3.9

0 field star, no encounters — default value

Evolution Parameters

alphaRLO The fraction[3] of material during Roche-lobe overflow which is directly lost from the system as an isotropic wind from the donor star. Its default value is 20% and it can be specified in line 10 of the `data.in` file.

beta, beta_const The fraction[3] of material which cannot be accreted by the companion during Roche-lobe overflow. This material will be isotropically re-emitted from the accretor with its specific orbital momentum. You can specify a minimum constant fraction in `beta_const` in line 11 of the `data.in` file. Its default is 75%. The final value of `beta` is at least `beta_const` but will increase if the Eddington accretion limit is reached.

Gamma The square root of the ratio of the radius of a circumbinary torus and the semi-major axis of the binary, see Soberman et al. (1997). The default value is 2.0 and it can be changed in line 12 of the `data.in` file.

delta This parameter contains the fraction[3] of the material which goes into a circumbinary torus. You can change the default value of 0% in line 13 of the `data.in` file.

qlimit When a normal star is filling its Roche lobe a limit in the mass ratio determines whether the outcome is a stable Roche-lobe overflow or a common envelope. The default of this limit is 2.5 and it can be specified otherwise in line 16 of the `data.in` file.

Mhece For a naked helium star orbiting a non-degenerate star (e.g. a main sequence star) the transition between Roche-lobe overflow and common envelope is given by the mass of the naked helium star. In line 17 in the `data.in` file you can change the default value of $-3.3\ M_\odot$. If you put a negative mass limit there it is ignored and the `qlimit` is applied for mass transfer from a naked helium star onto a non-degenerate star instead.

[3] The sum of `alphaRLO`, `beta_const` and `delta` should be between 0 and 1.

`alphaCE` The efficiency of converting the orbital energy into kinetic energy of the common envelope, to unbind it, is as default 50%. In line 14 of the `data.in` file the value can be changed.

`lambda`, `lambda_const` The λ-value represents the structure of the stellar envelope. It gives information on how easily the common envelope can be ejected. `lambda_const=1.0` is a default if no other values are available and it can be set in line 15 of the `data.in` file. The λ values are normally taken from calculated tables.

`alphaTH` This is used to interpolate between a pure gravitational λ_G and λ_B taking pressure and radiation into account, too. A value like the default of 0.5, which is between 0.0 and 1.0, interpolates the values in the stellar tables, see section 2.6. A value smaller than 0.0 uses the old tables from Dewi and Tauris (2000). A value larger than 1.0 multiplies the value with the pure gravitational λ_G so that a value of 2.0 corresponds to virial equilibrium.
The value in line 19 of the `data.in` file can be changed to:

- < 0 use the old λ tables from Dewi and Tauris (2000)
- $\in [0, 1]$ interpolate between λ_G and λ_B where the value gives the fraction of the pressure and radiation taken into account 0.5 is the default value
- > 1 use as a constant factor of λ_G

`kickv` This parameter specifies the kick velocity in km/s which a new-formed neutron star or black hole gets. A value of 0.0 corresponds to no kick and a negative value like the default -1.0 uses different distribution functions, see section 3.6.4. You can change this value at line 18 in the `data.in` file to:

- ≥ 0 fixed value in km/s for all kicks
- < 0 get kick from distributions, see section 3.6.4 default value is -1.0

`galpotential` This flag indicates which galactic potential should be used for the motion in the galaxy. Possible values are (line 21 of the `data.in` file):

- `0` no galactic potential default value
- `1` Milky Way potential by Allen and Santillan (1991)

2.3 Single run

To evolve a single system, I recommend to use the command line arguments, see section 2.1.2. With them you can specify the primary and secondary mass, the semi-major axis and a random seed for the kick generator. You can although use a `data.in` file and specify the number of systems to 1.

2.4 Population synthesis

In general, the code is designed to do population synthesis of a large number of binary systems. To check if every thing is running the first 10 calculated systems are displayed like in the single run. With the `output` option `M`, a speed estimate is written to `stderr`. This output is written every $\frac{\texttt{number}}{100}$ systems or every 10^6 systems.

A lot of information about the run can be written to output files which will be described in the section 2.5.

2.5 Output files

ComBinE will create some files with data in the current directory. If such files are already existing they will be overwritten. Therefore you should not run ComBinE in the same directory two times if you do not want to delete the results of the first run.

The following subsections will give you an overview of which files are created and which data will be stored in them.

2.5.1 `data.out`

This file contains the main output of a population synthesis run. The input parameters are written at the top, followed by multiple counters with their descriptions and values. Here is an example file:

```
#parallel    seed    number    (Mp_max,Mp_min)  (Ms_max,Ms_min)  (a_max,a_min)    alphaRLO    beta_const    Gamma    delta    alphaCE  lambda_const    qlimit   Mhece    kickv    alphaTH  galintegrator    galpotential    output   IMF    qdist    adist    edist    tdist    Zdist    rotdist  Rdist    Vdist    rhodist
1    908070605040302010    1000000000    (100,4)  (100,1)  (10000,2)    0.2    0.75    2    0    0.5    1    2.5    -3.3    -1    0.5    1    0    M    1    1    1    0    0    0    0    0    0    0
#counts RLO:  total    case A   case B/C    case BB
360440141    62140512    133943972    164355657
#counts CE:  total    case A   case B/C    case BB
731225685    336322797    386068608    8834280
#counts merger:  total    in RLO   in CE    in SN
567278533    50847    566493576    734110
#counts SN/PN:  SN    NS    BH    PN    WD
121905253    107597339    14307914    754149927    754149927
#number of systems    survive(SS)    destroyed(DS)
1000000000    330457988    669542012
#number of SS    WD-WD    WD-NS    WD-BH    NS-WD    NS-NS    NS-BH    BH-WD    BH-NS    BH-BH    unknown
330457988    328024869    195818    0    1777434  105428    85    87830    231338   35186    0
```

```
15  #SS t_gw<1e+10yr          WD-WD    WD-NS    WD-BH    NS-WD    NS-NS
        NS-BH    BH-WD    BH-NS    BH-BH    unknown
16  20589707         20400867         39953    0         34020    46198
        0          1827     62751    4091     0
17  #number of DS    merged sys.      disrupted sys.   unknown
18  669542012        567278533        102263479        0
```

In lines 1 and 2, you find the header and the values of the input parameters, see sections 2.1.1 and 2.2. The next four pairs of lines, 3-4, 5-6, 7-8 and 9-10, contain the header and the values for the counts of events, like Roche-lobe overflow(RLO), common envelope(CE), merger and white dwarf (WD), neutron star (NS) or black hole (BH) formation, respectively.

Then follows the number of systems which survive as a binary of two compact objects or are destroyed, in lines 11 and 12. The details of survived systems are in the lines 13 to 16. The header in line 13 shows that the survived systems are grouped depending on which compact objects they host. The order of the compact objects is their formation order. The values are in line 14. The last column called `unknown` should be always 0 otherwise there are systems formed in an unconsidered way. The lines 15 and 16 represent a sub-sample of the previous two lines with the additional condition that the time to merge by gravitational wave radiation $t_{\mathrm{gw}} \leq 10^{10}$ yr. The line 18 gives more details how the systems are destroyed, while line 17 contains its header.

This output can be switched off by setting `data` to `false`.

2.5.2 `hist` files

The code provides some routines to create histograms, see section 3.8. These histograms can be written to files and will get the extension `.hist`. The first line contains some dimensional data about the histogram. The second line is the header of the following data: The first data-column contains the binned value. The second one is filled either by the histogram counts or by the normalised counts to the bin width, see section 3.8. All other columns contain the data of the sub-samples if specified.

2.5.3 `dist` files

The code can also print out tables with various specific data, e.g. all values of semi-major axis at the end of the evolution of a system. To enable writing the `.dist` files you have to set `dist=true` in the code. But be aware that this kind of output will store a lot of data when you run a large number of systems. All files with the ending `.dist` contain a header line which tells you about the content in the different columns. Each row will represent one calculated system.

As default an additional table called `distribution.csv` is created. It contains information about a specific kind of systems which appear during a population synthesis run. The systems written their as default are all surviving binaries containing only neutron stars or black holes.

2.6 Stellar tables

The code interpolates tables of stellar evolution. These tables have to be provided for ComBinE from the user. As default the tables are from Brott et al. (2011); Szécsi et al. (2015) and an extension of this grid, calculated by Matthias Kruckow (Kruckow et al., 2018). They should contain the following columns:

1. total mass m in $\mathrm{M}_{\odot}$, where the first row is taken as initial mass
2. age t in yr, where 0.0 indicates the ZAMS
3. radius r in $\mathrm{R}_{\odot}$
4. core mass cm in $\mathrm{M}_{\odot}$
5. luminosity L as $\log_{10}\left(\frac{L}{\mathrm{L}_{\odot}}\right)$
6. effective temperature T_{eff} as $\log_{10}\left(\frac{T_{\mathrm{eff}}}{\mathrm{K}}\right)$
7. $\lambda = \frac{G \cdot m \cdot (m - cm)}{r \cdot E_{\mathrm{bind}}}$ using only the gravitational binding energy
8. λ using the gravitational binding and the thermodynamic energy
9. carbon core mass cc in $\mathrm{M}_{\odot}$

If there are columns missing at the end the code assumed some values, e.g. $cc << 1\ \mathrm{M}_{\odot}$. The code normally differentiates between main sequence and post-main sequence stars. Therefore the last line of the evolutionary tables contains the negative index of the TAMS.

The code uses tables for normal stars and helium stars. These tables have to be placed in the directories `./tables/*stars/` and `./tables/*hestars/` respectively. Here `*` indicated the grid being `MW`, `LMC`, `SMC` or `IZw18`. If you want to place the directory `tables` with its subdirectories elsewhere than in the current directory, have a look at the beginning of section 2 how to provide the location to the code.

3 Code structure

The code is split into several sub-files. The sub-files are in the directory `ComBinElib`. There is a header for all the files in this directory called `ComBinElib.h`. The main directory contains a make-file to compile the code. To make changes, or getting some other output, you can use the user interface, see section 3.9. In the following sections it will be described how the code works.

3.1 Main structure

The following flowchart in Figure 1 shows what the code is doing.

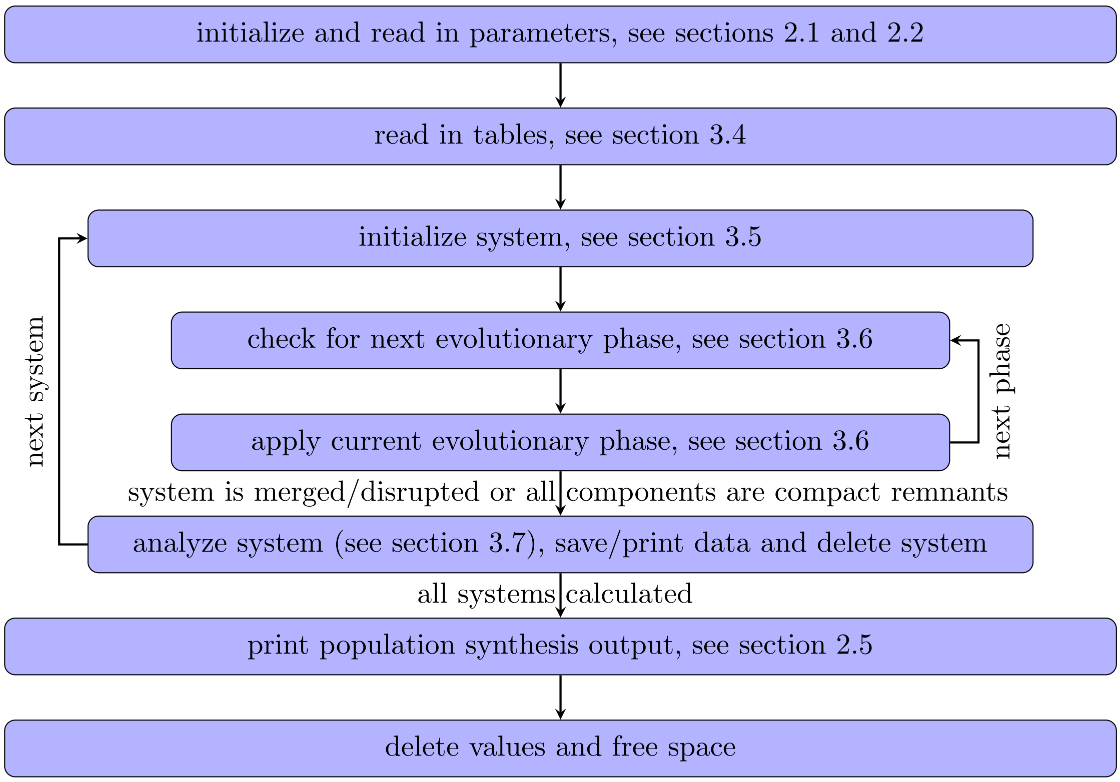

Figure 1: Flowchart of the main routine.

3.2 Constants

In the code are several constants defined which are either physical constants or conversion factors between different units, see table 1. All constants are from type `double`. The standard units in the code are $\mathrm{M}_\odot$, $\mathrm{R}_\odot$ and yr.

Table 1: Defined constants in the code.

name	description
`Msun`	the mass of the sun: $1\ \mathrm{M}_\odot = 1.9885 \cdot 10^{30}$ kg (Olive and Particle Data Group, 2014)
`Rsun`	the radius of the sun: $1\ \mathrm{R}_\odot = 6.9551 \cdot 10^{8}$ m (Olive and Particle Data Group, 2014)
`yr`	sidereal year: $1\ \mathrm{yr} = 31558149.8$ s (Olive and Particle Data Group, 2014)
`Lsun`	the luminosity of the sun: $1\ \mathrm{L}_\odot = 3.828 \cdot 10^{26}\ \mathrm{kg\,m^2\,s^{-3}}$ (Olive and Particle Data Group, 2014), in the code this value is converted to $\approx 12.5076\ \mathrm{M}_\odot\,\mathrm{R}_\odot^2\,\mathrm{yr}^{-3}$
`G`	the gravitational constant: $G_N = 6.67384 \cdot 10^{-11}\ \mathrm{m^3\,kg^{-1}\,s^{-2}}$ (Olive and Particle Data Group, 2014), in the code it is $\approx 3.92839 \cdot 10^{8}\ \mathrm{R}_\odot^3\,\mathrm{M}_\odot^{-1}\,\mathrm{yr}^{-2}$

Table 1 continued.

name	description
`c`	the speed of light: $c = 299792458\,\mathrm{m\,s^{-1}}$ (Olive and Particle Data Group, 2014), which is converted to $\approx 1.36028 \cdot 10^{7}\ \mathrm{R_\odot\,yr^{-1}}$ within the code
`MH`	the proton mass/mass of ionized hydrogen: $m_p = 1.672621777 \cdot 10^{-27}$ kg (Olive and Particle Data Group, 2014), the code uses the value of $\approx 8.41147 \cdot 10^{-58}\ \mathrm{M_\odot}$
`sigmaT`	the Thomson cross section of eletrons: $\sigma_T = 0.6652458734 \cdot 10^{-28}\ \mathrm{m^2}$ (Olive and Particle Data Group, 2014), while the code uses $\approx 1.37523 \cdot 10^{-46}\ \mathrm{R_\odot^2}$
`cgsEnergy`	energy conversion factor to cgs units 1 erg $\approx 1.03536 \cdot 10^{-40}\ \mathrm{M_\odot\,R_\odot^2\,yr^{-2}}$
`kms`	velocity conversion factor 1 $\mathrm{km\,s^{-1}} \approx 45.3741\ \mathrm{R_\odot\,yr^{-1}}$
`day`	sidereal day: $23^{\mathrm{h}}56^{\mathrm{m}}4.09053^{\mathrm{s}} \approx 2.73033 \cdot 10^{-3}$ yr (Olive and Particle Data Group, 2014)
`au`	astronomical unit: $1.495978707 \cdot 10^{11}$ m $\approx 215.091\ \mathrm{R_\odot}$ (Olive and Particle Data Group, 2014)
`pc`	parsec: $3.08567758149 \cdot 10^{16}$ m $\approx 4.43657 \cdot 10^{7}\ \mathrm{R_\odot}$ (Olive and Particle Data Group, 2014)

3.3 Structure definitions

In the code new structure types are defined for the physical objects. Those are `t_system` and `t_star` and will be described in the sections 3.3.1 and 3.3.2, respectively. The structure `t_SN` contains more data about the SN phase, see section 3.3.3. Additionally there is a structure `t_HRD`, see section 3.3.4, which contains all the phase space information about a star. It is also used to store the stellar gird data, see section 3.4.

3.3.1 `t_system`

This structure represents the total system. Its components are:

Table 2: Components of `t_system`.

type	name	description
`int`	`n`	number of entries in the arrays of `t_system` and its components of type `t_star`
`t_star`	`prim`	primary star, see section 3.3.2
`t_star`	`sec`	secondary star, see section 3.3.2
`double*`	`M`	array of total system mass in $\mathrm{M_\odot}$
`double*`	`qp`	array of mass ratio = mass of primary/mass of secondary
`double*`	`qs`	array of mass ratio = mass of secondary/mass of primary
`double*`	`a`	array of semi-major axis in $\mathrm{R_\odot}$
`double*`	`P`	array of orbital period in yr
`double*`	`rp`	array of Roche-lobe radius of primary in $\mathrm{R_\odot}$

Table 2 continued.

type	name	description
double*	rs	array of Roche-lobe radius of secondary in $R_\odot$
double*	e	array of eccentricity
double*	peri	array of peri-astron separation in $R_\odot$
double*	x	array of galactic x-position in pc
double*	y	array of galactic y-position in pc
double*	z	array of galactic z-position in pc
double*	vx	array of galactic x-velocity in km/s
double*	vy	array of galactic y-velocity in km/s
double*	vz	array of galactic z-velocity in km/s
double*	rhostar	array of density of surrounding stars in pc^{-3}
int*	phase	array of phases of the system, see table 8
int	stagechange	indicates if the stage of one component has changed
int	formation	indicates which formation channel is used
double	tgw	time to merge by gravitational wave emission

3.3.2 t_star

This structure is a single star. Its components are:

Table 3: Components of t_star.

type	name	description
double*	m	array of mass in $M_\odot$
double*	t	array of time in yr
double*	r	array of radius in $R_\odot$
double*	cm	array of core mass in $M_\odot$
double*	llum	array of luminosity in $\log_{10}\left(\frac{L}{L_\odot}\right)$
double*	lteff	array of effective temperature in $\log_{10}\left(\frac{T_{\rm eff}}{K}\right)$
double*	lambda	array of λ
double*	omega	array of angular velocity
int*	stage	array of evolutionary stage, see table 4
t_HRD	track	track of this star, see section 3.3.4
int	rmax	track index where the radius is maximal
int	last	track index where the last time in track is
double	inimass	initial mass of the track
double	metal	metallicity
t_SN	SN	supernova data, see section 3.3.3

Stages

The possible stages are:

Table 4: Evolutionary stages of stars.

value	description
0	hydrogen burning
1	helium burning
2	naked helium burning
3	white dwarf (WD)
4	neutron star (NS)
5	black hole (BH)
-2	star losing its envelope and become 2
-3	star forming a PN or explode in a SN and become 3, 4 or 5, respectively

3.3.3 `t_SN`

This structure stores more data about the supernova (SN) phase. Its components are:

Table 5: Components of `t_SN`.

type	name	description
`double`	`inimass`	initial mass of the star/he-star
`double`	`Hecoremass`	pre-SN He-core mass
`double`	`COcoremass`	pre-SN CO-core mass
`double`	`remnantmass`	post-SN remnant mass
`double`	`w`	kick velocity in km/s
`double`	`theta`	kick angle θ in degree
`double`	`phi`	kick angle ϕ in degree
`double`	`vsys`	systemic velocity change by the kick in km/s
`int`	`startype`	stellar stage prior to the SN, see table 4
`int`	`type`	type of supernova: 0 no supernova; 1 planetary nebula instead of SN; 2 electron capture supernova; 3 iron core collapse supernova; 4 collapse to blackhole

3.3.4 `t_HRD`

This structure is a track of a star. Its components are:

Table 6: Components of `t_HRD`.

type	name	description
`int`	`n`	number of entries in the arrays
`int`	`TAMS`	array index of the TAMS
`double*`	`m`	array of mass in $M_\odot$
`double*`	`t`	array of time in yr

Table 6 continued.

type	name	description
double*	r	array of radius in $\mathrm{R}_\odot$
double*	cm	array of core mass in $\mathrm{M}_\odot$
double*	llum	array of luminosity in $\log_{10}\left(\frac{L}{\mathrm{L}_\odot}\right)$
double*	lteff	array of effective temperature in $\log_{10}\left(\frac{T_\mathrm{eff}}{\mathrm{K}}\right)$
double*	lambda	array of λ
double	inimass	initial mass of the track

3.3.5 t_hist

This structure represents a histogram. It contains:

Table 7: Components of t_hist.

type	name	description
int	n	number of bins in the arrays
int	subs	number of subdivisions in the arrays
long	ctot	total number of counts
long*	c	array of counts
long**	subc	two dimensional array of sub counts
double*	x	array of x values of bin borders
double	min	minimum value
double	max	maximum value
bool	logscale	linear or logscale

3.4 Read stellar tables

As already described in section 2.6, the code interpolates tables for the stellar evolution of each star. To load all available tables, the code searches for them and creates Find files in the directory where the code runs. These files are named Find*.* where the type of searched tables or their location is specified.

When the tables are read in, the λ parameter is only stored as one variable depending on alphaTH, see section 2.2.3. Additionally the core masses of convective cores are adapted. Then the data is stored in a structure of type t_HRD, see section 3.3.4. Furthermore a smoothing is applied to the tables which replaces totally flat parts by very weekly changing parts, depending on the behaviour of the neighbouring data points to avoid numerical instabilities.

3.5 Initial conditions

The initial conditions of a system are either fixed values or randomly created from generating functions depending on the input parameters, see section 2.2.3. The code can set a fixed value, choose a random value or create a grid of stars, see section 2.1.2.

The generation order of the parameters is the same as in sections 2.2.3 and 3.9. The first determined value is the primary mass, the mass of the initially more massive star. The next one is the mass ratio and accordingly the secondary mass. Then follows the semi-major axis or the period of the system. The last of the basic parameters to be determined is the eccentricity. If specified, a number of additional parameters are chosen: the initial age, the metallicity, the stellar spins, the position and velocity in a galaxy and the stellar density around the system.

3.6 Evolutionary phases

In the code all binary events are related to a phase. These considered evolutionary phases are:

Table 8: Phases for the systems.

value	description
0	wind mass loss for both stars according to the stellar evolution
12	RLO from primary to secondary
21	RLO from secondary to primary
13	CE where the primary fills its Roche-lobe
23	CE where the secondary fills its Roche-lobe
4	early merge
15	SN of primary
25	SN of secondary
16	PN of primary
26	PN of secondary
-1	destroyed, because of a disruptive kick during a SN or after an early merger
94	gravitational merger within 10^{10} yr/both stars are compact remnants
99	end stage/both stars are compact remnants

When the initial conditions, see section 3.5, are fixed the system starts always with a phase of wind mass loss for both stars.

3.6.1 Wind mass loss

During this phase both stars follow their tracks from the evolutionary tables, see section 2.6. The separation of the system changes according to Soberman et al. (1997).

This phase normally ends when either one star is filling its Roche lobe, see section 3.6.2 or the end of the stellar track is reached, see section 3.6.4. It will be followed by another wind mass loss phase when one of the stars finishes its main sequence.

3.6.2 Roche-lobe overflow

Before one star fills its Roche lobe, its is assumed that the tides in such a close orbit will circularise the system before a stable mass transfer starts. When a star is filling its Roche lobe it is checked whether the mass transfer is expected to be a stable Roche-lobe overflow (RLO) or if it will be unstable and lead to a common envelope (CE), see section 3.6.3. Depending on the evolutionary stage of the donor star, a limit on mass ratio or donor mass is used to differentiate between the two cases of mass transfer. For a normal star the two cases are distinguished by the mass-ratio limit `qlimit`, see section 2.2.3. Furthermore the fraction to maximal extend is checked to consider that convective envelopes lead to unstable mass transfer. If the star is a helium star transferring mass onto a non-degenerate star, its mass is compared to `Mhece`, see section 2.2.3. If `Mhece` is negative the `qlimit` is used for the helium stars as well. The mass transfer from a helium star to a compact object is treated as always stable if the orbital period is larger than 0.07 d (Tauris et al., 2015).

As a typical timescale of the RLO, the thermal timescale of the donor is used and is enlarged by a factor of three for a normal star. It is assumed that the donor loses its whole envelope in a RLO. The three parameters `alphaRLO`, `beta` and `delta` specify which part of the lost material leaves the system: directly from the donor, is re-emitted from the accretor, and is transferred to a circumbinary torus, respectively. While `alphaRLO` and `delta` are fixed parameters, which are given by the user, `beta` is determined dynamically. Its lowest value is `beta_const` but it could increase if the companion is not able to accrete all the material. The accretion rate is limited by the Eddington accretion rate.

The orbital parameters change according to Soberman et al. (1997). If the donor was a normal star it becomes a naked helium star after the RLO with a total mass equal to its previous core mass. If it was already a helium star it will go off in a SN or from a WD depending on its mass, see section 3.6.4. If the star does not detach after the mass transfer, the two stars will merge. The accretor will get a new stellar track according to its total and core mass after the mass transfer.

3.6.3 Common envelope

If a mass transfer is unstable the donor will expand rapidly and create a common envelope. During the CE the material of the donors envelope is ejected from the system on a typical timescale of $< 1000\,\mathrm{yr}$. The λ formalism is applied and the parameters `alphaCE` and `lambda_const` are fixed by the user. Depending on `alphaTH` the final λ representing the stellar structure is determined, see section 2.2.3. If the companion is a compact object, some of the envelope material will be accreted onto it which releases some energy helping to unbind the rest of the envelope. For the donor the resulting stages of the CE are the same as the ones of a stable RLO, see section 3.6.2.

3.6.4 Supernova/planetary nebula

Depending on the mass of a star it will end its life as a WD, NS or BH. A WD may form a planetary nebula (PN) around it, while a NS or a BH will be the compact remnant of a supernova. The formation of a BH may produce a dark SN which mainly consists of neutrinos. While a system with a PN will remain bound, a SN may disrupt the binary.

Kicks

For newborn NSs and BHs kicks are expected. The code distinguishes between different kinds of SNe if `kickv` < 0. Associated with them are different kick distributions. For an electron capture SN, the kick is taken from a flat distribution up to 50 km/s, while an iron core-collapse SN uses a Maxwell-Boltzmann distribution. Depending on the mass loss during the evolution of the star the root-mean-square velocity is adjusted (Kruckow et al., 2018). All BHs get kicks from a flat distribution up to 200 km/s.

Shell Impact

A shell impact on the companion of a SN is applied following Tauris and Takens (1998). The code uses a generalisation of the formalism where the circularised semi-major axis is replaced by the separation at the moment of the explosion. After the impact the companion gets a new stellar track depending on its total and core mass. At the end it is checked whether the system remains bound or if it is disrupted or if the orbit gets so small that the system will merge.

3.7 Analysing the binary

When the system reaches its final phase the system is analysed to extract the data of interest. If the `output` option is set to the massive output, the evolution of the system is printed to the screen after the analysis.

3.7.1 Integration tables

To speed up the code, some integrations like the orbital shrinking by gravitational waves is tabulated. These tables will be created when the program runs the first time, or whenever those are missing, which requires write access to the directory with the tables, see subsection 2.

3.7.2 Galactic motion

If it is of interest, the code will follow the motion of a system in a galaxy. Therefore the flags `galintegrator` and `galpotential` have to be set, see section 2.2.

The analysed data should then be printed to the screen, written to a file or stored in a histogram, see section 3.8. The data of a system will be deleted before the next one is calculated.

3.8 Histograms

In the statistic part of the code are some general functions to create histograms during the run time. For the histograms a special type is defined. It is called `t_hist`, see section 3.3.5.

The histogram functions are:

`void inithist(t_hist& hist, int nbin, double min, double max, bool logscale, int subs)`
This function initializes a new histogram. It will have `nbin` bins which cover a total range between `min` and `max`. If you set `logscale=true` then the bins will be of equal size on a logarithmic scale, otherwise on a linear scale. If you want to divide the counts of a bin into subbins use `subs>1`. You should use this function only in the function `PreMainLoop`, see section 3.9.

`void addtohist(t_hist& hist, double value, int sub)`
This function you should use to add data to the histogram. `value` is the new data. If you specify `sub` $\in$ [`0`, `subs-1`] the data will likewise add to the sub histogram with the index `sub`. Adding data to the histogram should be done in the function `AfterEvolution`, see section 3.9.

`void histout(t_hist& hist, bool perbinsize, char* name)`
This function writes the histogram data to a file. With the flag `perbinsize` you can specify whether you want only the counts printed or if you want to get the counts normalized to its bin size. The `name` is the name of the created file in the current directory. It gets an automatic extension `.hist`. For the structure of the `hist` files have a look in section 2.5.2. You should use this function only in the function `AfterMainLoop`, see section 3.9.

`void freehist(t_hist& hist)`
This routine deletes the histogram data. All data in the histogram will no longer be available and the memory of the arrays will be freed. You should use this function only when you do not need the data in the histogram any longer. The desired place of use is in the function `AfterMainLoop`, see section 3.9.

3.9 User interface

The file `ComBinElib/user.cpp` is a user interface to the code. There are several functions provided. They are called by the code and allow the user to make changes during the runtime without changing the main routines of the code. The following list describes the provided functions ordered by the usual call time.

Table 9: User functions.

function(parameter)	description
`void First()`	This function is called directly after the variable declaration in the main function.

Table 9 continued.

function(parameter)	description
`void AfterReadParameters(double& Mp_max, double& Mp_min, double& Ms_max, double& Ms_min, double& a_max, double& a_min, double& Mp, double& Ms, double& a, double& e, double& metall, double& vp, double& vs, double& x, double& y, double& z, double& vx, double& vy, double& vz, double& rhostar, int& parallel, int& galintegrator, int& galpotential)`	This function is called after the parameters are read in from the command line or `data.in` , see sections 2.1.2 and 2.1.1 respectively. All parameters are described in section 2.2. You can use this function to change the variables during the run time to rescale units if you prefer different input units.
`void PreReadTables()`	This function is called directly before the tables are read in.
`void AfterReadTables()`	This function is called after the tables are read in. You can use this function to check the tables.
`void SetSeed(long& seed, long* discard, int n)`	This function is called directly before the random seeds are set. `n` is the number of entries in `discard`. You can manipulate `seed` and `discard` to reproduce a specific system which occurred in a population synthesis run.
`void PreMainLoop()`	This function is called directly before the main loop of calculating all the systems. You can use it to set up new histograms, see section 3.8.
`double UserInitial_Mp(double Mp_max, double Mp_min)`	In this function you can define your own initial primary mass function. The return value is the primary mass in $M_\odot$.
`double UserInitial_q(double Ms_max, double Ms_min, double Mp)`	In this function you can define your own initial mass ratio function. The return value is the secondary mass in units of the primary mass or simply the mass ratio.
`double UserInitial_a(double a_max, double a_min, double Mp, double Ms)`	In this function you can define your own initial semi-major axis function. The return value is the semi-major axis in $R_\odot$.
`double UserInitial_e(double e_max, double Mp, double Ms, double a)`	In this function you can define your own initial eccentricity. The return value is the initial eccentricity the system should have.
`double UserInitial_t(double Mp, double Ms, double a, double e)`	In this function you can define your own initial age. The return value is this age in yr. (currently no effect)

Table 9 continued.

function(parameter)	description
`double UserInitial_Z(double Mp, double Ms, double a, double e, double t)`	In this function you can define your own initial metallicity. The return value is the metallicity of the system. (only metallicities corresponding to the stellar tables are considered)
`void UserInitial_omega(double& omegap, double& omegas, double Mp, double Ms, double a, double e, double t, double metall)`	In this function you can define your own initial angular spin. The values of the spin has to be stored in `omegap` and `omegas` for the primary and the secondary, respectively. Their unit is yr^{-1}. (currently no effect)
`void UserInitial_R(double& x, double& y, double& z, double Mp, double Ms, double a, double e, double t, double metall, double omegap, double omegas)`	In this function you can define your own initial position in a galactic potential. The position vector components are `x`, `y` and `z` as Cartesian coordinates of the galactic rest frame in pc.
`void UserInitial_V(double& vx, double& vy, double& vz, double Mp, double Ms, double a, double e, double t, double metall, double omegap, double omegas, double x, double y, double z)`	In this function you can define your own initial velocity in a galactic potential. The Cartesian velocity components of the galactic rest frame in km/s should be stored in `vx`, `vy` and `vz`.
`double UserInitial_rho(double Mp, double Ms, double a, double e, double t, double metall, double omegap, double omegas, double x, double y, double z, double vx, double vy, double vz)`	In this function you can define your own initial stellar density. The value of the stellar density is to be returned in pc^{-3}. (currently no effect)
`void AfterEvolution(t_system& system)`	This function is called after the system is evolved. `system` contains all the evolution information of the system, see section 3.3.1 for the structure information of the type `t_system`. Here the data should be written out or stored in the histograms.
`long ToDebug(long i)`	This function defines which system is printed with debug output, `i` is the number of the current system. Any value ≤ 0 will give non debug output.
`void AfterMainLoop()`	This function is called directly after the main loop of calculating all the systems. You can use it to write out and delete your own defined histograms, see section 3.8.

4 Warnings and errors

All warning or error messages created by the code start with `#Warning` and `#Error` respectively, see sections 4.1 and 4.2. They are written to `stderr`. Additionally there are some outputs starting with `#` to `stderr` which will be printed if some values are non-physical. All important values are checked if they are sensible in a physical context. If one of these checks fails, you will get one of the outputs summarised in table 10.

Table 10: Description of check of physical parameters.

output (* denotes some value)	description
`#prim.m=*Msun`	the primary's mass has a non-physical value
`#prim.t=*yr`	the primary's age has a non-physical value
`#prim.r=*Rsun`	the primary's radius has a non-physical value
`#prim.cm=*Msun`	the primary's core mass has a non-physical value
`#prim.llum=*`	the primary's logarithmic luminosity has a non-physical value
`#prim.lteff=*`	the primary's logarithmic effective temperature has a non-physical value
`#prim.lambda=*`	the primary's λ has a non-physical value
`#prim.omega=*/yr`	the primary's spin has a non-physical value
`#prim.stage=*`	the primary's stage has no value
`#sec.m=*Msun`	the secondary's mass has a non-physical value
`#sec.t=*yr`	the secondary's age has a non-physical value
`#sec.r=*Rsun`	the secondary's radius has a non-physical value
`#sec.cm=*Msun`	the secondary's core mass has a non-physical value
`#sec.llum=*`	the secondary's logarithmic luminosity has a non-physical value
`#sec.lteff=*`	the secondary's logarithmic effective temperature has a non-physical value
`#sec.lambda=*`	the secondary's λ has a non-physical value
`#sec.omega=*/yr`	the secondary's spin has a non-physical value
`#sec.stage=*`	the secondary's stage has no value
`#sys.M=*Msun`	the total mass of the system has a non-physical value
`#sys.qp=*`	the primary mass in units of the secondary mass has a non-physical value
`#sys.qs=*`	the secondary mass in units of the primary mass has a non-physical value
`#sys.a=*Rsun`	the semi-major axis has a non-physical value
`#sys.P=*yr`	the period has a non-physical value
`#sys.rp=*Rsun`	the Roche-lobe radius of the primary has a non-physical value
`#sys.rs=*Rsun`	the Roche-lobe radius of the secondary has a non-physical value

Table 10 continued.

output (* denotes some value)	description
`#sys.e=*`	the eccentricity has a non-physical value
`#sys.x=*pc`	the x coordinate of the system has a non-physical value
`#sys.y=*pc`	the y coordinate of the system has a non-physical value
`#sys.z=*pc`	the z coordinate of the system has a non-physical value
`#sys.vx=*km/s`	the velocity in x direction of the system has a non-physical value
`#sys.vy=*km/s`	the velocity in y direction of the system has a non-physical value
`#sys.vz=*km/s`	the velocity in z direction of the system has a non-physical value
`#sys.rhostar=*/pc^3`	the stellar density around the system has a non-physical value
`#sys.phase=*`	the phase of the system has no value
`#system.prim.lambda[n]=* ...`	the λ of the primary has a non-physical value
`#system.sec.lambda[n]=* ...`	the λ of the secondary has a non-physical value

4.1 Warning messages

Warnings should awake the curiosity of the user. Some of them only show up if the screen output is enabled, see section 2.2.2. The following list gives you an alphabetically ordered overview of possible warning messages.

Table 11: Warning messages.

warning message (* denotes some value)	description
`#Warning: accuracy not reached in RLFT2: rs=*Rsun roche=*Rsun diff=*Rsun 1.0-roche/rs=*=*`	the solution when a star fills its Roche lobe has not the required accuracy, see section 2.2.1, only displayed if `screen=true`
`#Warning: companion[0].t[*]=*yr set to *yr`	the accretor of the CE would exceed its lifetime during the CE, therefore the time of the CE phase is reduced
`#Warning: end track time newtrack.t[0]=*yr=*yr reached: newtrack.t[*]=*yr newtrack.t[0]-newtrack.t[*]=*yr Warning: newtrack.t[*]=*yr=*`	the ages in the new track exceed the determined maximum, the new track is truncated

Table 11 continued.

warning message (* denotes some value)	description
`#Warning: generate new seed(i):*`	the given random seed curses a problem during initialising of the random number generator, so a new seed is generated
`#Warning: Invalid integrator spezified: use RungeKutta 4 integrator`	the flag of the galactic integrator has an undefined value and is set to the default value, see section 2.2.1
`#Warning: Invalid potential spezified: use no potential`	the flag of the galactic potential has an undefined value and is set to the default value, see section 2.2.3
`#Warning: No initial age distribution spezified: use tdist=0`	the value of the `tdist` flag is invalid, see section 2.2.3
`#Warning: No initial eccentricity distribution spezified: use edist=0`	the value of the `edist` flag is invalid, see section 2.2.3
`#Warning: No initial mass function spezified: use IMF=1`	the value of the `IMF` flag is invalid, see section 2.2.3
`#Warning: No initial mass ratio distribution spezified: use qdist=1`	the value of the `qdist` flag is invalid, see section 2.2.3
`#Warning: No initial metallicity distribution spezified: use Zdist=0`	the value of the `Zdist` flag is invalid, see section 2.2.3
`#Warning: No initial semi-major axis distribution spezified: use adist=1`	the value of the `adist` flag is invalid, see section 2.2.3
`#Warning: No initial space distribution in a galaxy spezified: use Rdist=0`	the value of the `Rdist` flag is invalid, see section 2.2.3
`#Warning: No initial stellar density distribution in a galaxy spezified: use rhodist=0`	the value of the `rhodist` flag is invalid, see section 2.2.3
`#Warning: No initial stellar spin distribution spezified: use rotdist=0`	the value of the `rotdist` flag is invalid, see section 2.2.3
`#Warning: No initial velocity distribution in a galaxy spezified: use Vdist=0`	the value of the `Vdist` flag is invalid, see section 2.2.3
`#Warning: no solution in RLFT2`	there is no solution found when a star fills its Roche lobe, while it was expected, only displayed if `screen=true`
`#Warning: no track update, because star out of grid: star.m[*]=*Msun and star.cm[*]=*Msun`	the conditions of the star are outside of the stellar gird, therefore the mass change remains unconsidered, only displayed if `screen=true`
`#Warning: prim.t[*]=*yr prim.track.t[*]=*yr sec.t[*]=*yr sec.track.t[*]=*yr`	one of the stars got older than its age defined by its stellar evolution track

Table 11 continued.

warning message (* denotes some value)	description
`#Warning: reduce core mass from star.cm[*]=*Msun to *Msun`	the star got a too large core mass and is placed at the end of the stellar grid
`#Warning: reduce mass from star.m[*]=*Msun to *Msun`	the star got a too large mass and is placed at the end of the stellar grid
`#Warning: replace negative lambda with 1.0e+10: *`	a negative λ would mean that the envelope is unbound, instead it is replace by a very loosely bound value
`#Warning: Roche-lobe-overflow setup: prim.r[0]=*Rsun rp[0]=*Rsun sec.r[0]=*Rsun rs[0]=*Rsun`	one of the stars fills its Roche lobe form the initial conditions
`#Warning: star[0].t[*]=*yr set to *yr`	the donor of the CE would exceed its lifetime during the CE, therefore the time of the CE phase is reduced
`#Warning: The user defined eccentricity is out of range: e=* e_min=0.0 e_max=*`	The eccentricity returned by `UserInitial_e` is smaller than `e_min` or larger than `e_max`
`#Warning: The user defined semi-major axis is out of range: a=*Rsun a_min=*Rsun a_max=*Rsun`	The semi-major axis returned by `UserInitial_a` is smaller than `a_min` or larger than `a_max`
`#Warning: t=*yr star.t[*]=*yr star.track.t[j]=*yr star.track.t[j-1]=*yr j=* last=* star.rmax=* star.track.n-1=* r_ratio=* roche=*Rsun star.track.r[j]=*Rsun star.track.r[j-1]=*Rsun cloop=* cjump=*`	the time when a star filled its Roche lobe is in the past, only displayed if `screen=true`
`#Warning: wrong formation value`	the system has an unconsidered formation channel, this warning enables the screen output for this system
`#Warning: *track(s) not updated, because star out of grid`	tells you that the stellar grid is too small, for some stars the mass change is not fully considered, they are placed at the end of the stellar grid

4.2 Error messages

If an error occurs in a run you should not trust the output of the simulation. As a normal user you should never see one of the error messages. The following list gives you an alphabetically ordered overview of implemented error messages.

Table 12: Error messages.

error message (* denotes some value)	description
`#Error: can't find *-tables in *, code: *`	some tables cannot be found in the given subdirectory of `tables`, see section 2.6
`#Error: can't open data.in`	you specified with `-2` to read the `data.in` file, but there is no such file in the current directory or you have no read access to it
`#Error: can't write to text, code: *`	creating the filename for the integration table fails
`#Error: cm=*Msun t_ratio=* j=* star.track.cm[j]=*Msun star.track.cm[j-1]=*Msun`	the core mass of the star would be negative
`#Error: common envelope: system.rp[*]=*Rsun system.prim.r[*]=*Rsun system.rs[*]=*Rsun system.sec.r[*]=*Rsun`	no star is selected as donor for the common envelope
`#Error: companion[0].r[*]=*Rsun companion[0].track.r[*]=*Rsun companion[0].track.r[*]=*Rsun t_ratio=*`	the accretor gets a negative radius after common envelope
`#Error: end track time: jlow=* nlow=* jup=* nup=*`	the determined end time of the track is reached, normally it should only happen for `jlow=nlow` and `jup=nup`
`#Error: e0=* eccarray.ecc[*]=* eccarray.ecc[*]=* eratio=*`	the eccentricity cannot be interpolated from the integration table for merger time due to gravitational wave radiation, see section 3.7.1
`#Error: He-star mass of *Msun not in table`	the given initial mass of the naked He-star is outside the He-star grid
`#Error: integer overflow n=*`	the galactic integration needs more steps than `INT_MAX`
`#Error: i=*>star.rmax=*`	no zero before maximum radius found, previous checks have failed
`#Error: jlow=*<>jup=* nlow=* nup=*`	the reduction of a track by one data point failed
`#Error: j=*>jmax=*`	no zero before the companion finishes its life time found, previous checks have failed
`#Error: mass increase of the system: dm=*Msun`	the system gains mass in a non-physical way

Table 12 continued.

error message (* denotes some value)	description
`#Error: memory allocation failed: *`	the memory allocation for a variable is not possible, please check if there is enough memory available
`#Error: memory reallocation failed: *`	the memory for a variable cannot be increased, please check if there is enough memory available
`#Error: Mp/Msun=* not in [*,*]`	the generation of a random primary mass from the IMF failed
`#Error: Mp=*Msun m1=*Msun value=* int1IMF=*`	the generation of a random primary mass from the canonical IMF failed
`#Error: mratio out of range: mratio=* star.m[*]=*Msun stararray0[*].m[*]=*Msun stararray0[*].m[*]=*Msun`	the mass of the star which will get a new track cannot be found in the prior determined mass range in the stellar grid
`#Error: mratio=* not in [*,*] star.cm[*]=*Msun star.m[*]=*Msun newtrack.m[*]=*Msun newtrack.m[*]=*Msun`	the mass is outside the considered accuracy region
`#Error: mratio-cmratio=*>* mratio=*=*/*=(*-*)/(*-*) cmratio=*=*/*=(*-*)/(*-*)`	the position defined by the mass and the core mass should be the same unless the numerical errors are too large
`#Error: m=*Msun t_ratio=* j=* star.track.m[j]=*Msun star.track.m[j-1]=*Msun`	the mass of the star would be negative
`#Error: negative core mass: inimass=*Msun cm[*]=*Msun jump=*`	during the correction for convective cores a core mass gets negative, see section 3.4
`#Error: negative psi=*`	the ψ for the kick/shell impact is negative, see section 3.6.4
`#Error: newtrack of m_ini=*Msun t_max=*yr: j=* jlow=* jup=* m=*Msun t=*yr r=*Rsun core mass(cm)=*Msun`	the mass or the age of a new track gets negative
`#Error: new companion mass=*Msun companion.m[*]=*Msun companion.track.m[j]=*Msun companion.track.m[j-1]=*Msun j=* last=* companion.track.n-1=* t_ratio=* t=*yr companion.track.t[j]=*yr companion.track.t[j-1]=*yr`	the companion mass would be too small for a star or negative

Table 12 continued.

error message (* denotes some value)	description
`#Error: new star mass=*Msun star.m[*]=*Msun star.track.m[j]=*Msun star.track.m[j-1]=*Msun j=* last=* star.rmax=* star.track.n-1=* r_ratio=* roche=*Rsun star.track.r[j]=*Rsun star.track.r[j-1]=*Rsun`	the star mass would be too small for a star or negative
`#Error: not all new track points copied: j=* newtrack.n=*`	the new track consists of less track points than previously determined
`#Error: no corresponding trk1 found`	if the tables are split into `trk1` and `trk2` files the code tries to match which belong together and the matching failed
`#Error: No initial age distribution spezified! t set to 0.0`	the `tdist` flag is invalid and was not automatically changed, see section 4.1
`#Error: No initial eccentricity distribution spezified! e set to 0.0`	the `edist` flag is invalid and was not automatically changed, see section 4.1
`#Error: No initial mass function spezified! Mp set to Mp_max=*`	the `IMF` flag is invalid and was not automatically changed, see section 4.1
`#Error: No initial mass ratio distribution spezified! q set to q_max=*`	the `qdist` flag is invalid and was not automatically changed, see section 4.1
`#Error: No initial metallicity distribution spezified! metallicity set to 0.02`	the `Zdist` flag is invalid and was not automatically changed, see section 4.1
`#Error: No initial semi-major axis distribution spezified! a set to a_max=*`	the `adist` flag is invalid and was not automatically changed, see section 4.1
`#Error: No initial space distribution in a galaxy spezified! position set to center`	the `Rdist` flag is invalid and was not automatically changed, see section 4.1
`#Error: No initial stellar density distribution in a galaxy spezified! star is set in the field`	the `rhodist` flag is invalid and was not automatically changed, see section 4.1
`#Error: No initial stellar spin distribution spezified! omegap and omegas set to 0.0`	the `rotdist` flag is invalid and was not automatically changed, see section 4.1
`#Error: No initial velocity distribution in a galaxy spezified! v set to 0.0`	the `Vdist` flag is invalid and was not automatically changed, see section 4.1
`#Error: no mergertime.int`	the integration table for the merger time cannot be read

Table 12 continued.

error message (* denotes some value)	description
`#Error: no random number generator selected to get random value`	the index of the used random number generator is invalid while trying to get a random value
`#Error: no random number generator selected(i=*) to set new seed`	the index of the used random number generator is invalid while setting a new seed
`#Error: no star for supernova/planetary nebula selected: system.prim.stage[*]=* system.sec.stage[*]=*`	no star is selected to explode in a supernova or to form a planetary nebula around it
`#Error: no TAMS expected, negative mass? value=*`	a negative mass is detected in a stellar track where no TAMS value is expected, see section 2.6
`#Error: no track update: star.stage[n]=*`	the star has a stage where no track update routine is defined
`#Error: overflow: system.phase[*]=*`	no star is selected as donor for the Roche-lobe overflow
`#Error: radius out of track: r=*Rsun system.prim.track.r[*]=*Rsun system.prim.track.r[*]=*Rsun diff=*Rsun system.prim.track.n-1=* ratio_p=* system.prim.last=*`	the determined radius when the primary fills its Roche lobe cannot be found in the evolutionary track of the primary
`#Error: radius out of track: r=*Rsun system.sec.track.r[*]=*Rsun system.sec.track.r[*]=*Rsun diff=*Rsun system.sec.track.n-1=* ratio_s=* system.sec.last=*`	the determined radius when the secondary fills its Roche lobe cannot be found in the evolutionary track of the secondary
`#Error: red=*`	undefined value of `red`, possible values are only 1 and −1
`#Error: RL-radii=(*, *)Rsun stellar radii=(*, *)Rsun ratiop=* ratios=* system.prim.track.t[*]=*yr t=*yr system.prim.track.t[*]=*yr system.sec.track.t[*]=*yr t=*yr system.sec.track.t[*]=*yr system.prim.m[*]=*Msun system.sec.m[*]=*Msun`	the star which is expected to fill its Roche lobe is not doing so
`#Error: Roche lobe=*Rsun na=*Rsun new star mass=*Msun new companion mass=*Msun nq=* r_ratio=* t_ratio=*`	the Roche lobe has no value
`#Error: roche=*Rsun star.track.r[*]=*Rsun star.track.r[*]=*Rsun`	the star cannot fill its Roche-lobe, previous checks have failed

Table 12 continued.

error message (* denotes some value)	description
`#Error: r=*Rsun t_ratio=* j=* star.track.r[j]=*Rsun star.track.r[j-1]=*Rsun`	the radius of the star would be negative
`#Error: r_ratio=* roche=*Rsun star.track.r[j]=*Rsun star.track.r[j-1]=*Rsun j=*`	r_ratio has no value
`#Error: stararray0[*].TAMS=* >stararray0[*].n=*`	the TAMS is past the stellar evolution track
`#Error: System is not synchronous`	the two stars in a binary system have different ages
`#Error: system.a[*]=*Rsun system.phase[*]=*`	the semi-major axis is negative
`#Error: system.formation=* system.prim.stage[*]=* system.sec.stage[*]=*`	something went wrong when the formation channel is determined: neither the primary nor the secondary corresponds to the last formed remnant
`#Error: system.prim.r[*]=*Rsun`	the primary has a negative radius
`#Error: system.prim.r[*]=*Rsun system.prim.track.r[*]=*Rsun system.prim.track.r[*]=*Rsun ratio_p=*`	the primary got a negative radius when one star fills its Roche-lobe
`#Error: system.sec.r[*]=*Rsun`	the secondary has a negative radius
`#Error: system.sec.r[*]=*Rsun system.sec.track.r[*]=*Rsun system.sec.track.r[*]=*Rsun ratio_s=*`	the secondary got a negative radius when one star fills its Roche-lobe
`#Error: table length: stararray0[*].n=* stararray0[*].n=* Error: j=* jlow=* nlow=* jup=* nup=*`	the positions in the two neighbouring tracks are undefined
`#Error: time out of track: companion[0].t[*]=*yr companion[0].track.t[*]=*yr companion[0].track.t[*]=*yr diff=*yr companion[0].track.n-1=* t_ratio=*`	the age after the CE cannot be found in the stellar track of the accretor
`#Error: time out of track: t=*yr system.prim.track.t[*]=*yr system.prim.track.t[*]=*yr diff=*yr system.prim.track.n-1=* ratio_p=* system.prim.last=*`	the determined age when the secondary fills its Roche-lobe cannot be found in the evolutionary track of the primary
`#Error: time out of track: t=*yr system.sec.track.t[*]=*yr system.sec.track.t[*]=*yr diff=*yr system.sec.track.n-1=* ratio_s=* system.sec.last=*`	the determined age when the primary fills its Roche-lobe cannot be found in the evolutionary track of the secondary
`#Error: too many new track points: i=* star.track.n=*`	the new track consists of more track points then previously determined

Table 12 continued.

error message (* denotes some value)	description
`#Error: too massive neutron star(*):*`	the primary/secondary is a NS with a mass above the NS mass limit
`#Error: too massive white dwarf(*):*`	the primary/secondary is a WD with a mass above the Chandrasekhar limit
`#Error: t=*yr star.track.t[*]=*yr star.track.t[*]=*yr ts=" << ts << "yr r_ratio=*`	the star cannot fill its Roche-lobe, previous checks have failed
`#Error: t_ratio=* t=*yr companion.track.t[j]=*yr companion.track.t[j-1]=*yr j=*`	t_ratio has no value
`#Error: t_roche_p=*yr tp=*yr ts=*yr`	at the time when the primary fills its Roche-lobe one of the stars exceeded its life time
`#Error: t_roche_s=*yr tp=*yr ts=*yr`	at the time when the secondary fills its Roche-lobe one of the stars exceeded its life time
`#Error: unconsidered kind: *`	the kind of specified stellar grid is not implemented
`#Error: unknown calculation of age`	the positions in the two neighbouring tracks are undefined, therefore the age cannot be calculated properly
`#Error: white dwarf mass=*Msun McoreCO=*Msun star[0].cm[n]=*Msun`	the calculated WD mass exceeds the Chandrasekhar limit
`#Error: wind mass increase of primary: dm=*Msun`	the primary gains mass in a non-physical way
`#Error: wind mass increase of secondary: dm=*Msun`	the secondary gains mass in a non-physical way
`#Error: wrong time span: star.track.t[*]=*yr time=*yr star.track.t[*]=*yr t_ratio=*`	the star never reaches its given start age for the calculation
`#Error: wrong track position: star.cm[*]=*Msun star.cm[*]=*Msun star.track.cm[*]=*Msun star.track.cm[*]=*Msun`	the current core mass is compatible with the core mass at the current track position
`#Error: wrong track position: star.m[*]=*Msun star.m[*]=*Msun star.track.m[*]=*Msun star.track.m[*]=*Msun`	the current mass is compatible with the mass at the current track position
`#Error: wrong track position: star.r[*]=*Rsun star.r[*]=*Rsun star.track.r[*]=*Rsun star.track.r[*]=*Rsun`	the current radius is compatible with the radius at the current track position

Table 12 continued.

error message (* denotes some value)	description
`#Error: wrong track position: star.t[*]=*yr star.t[*]=*yr star.track.t[*]=*yr star.track.t[*]=*yr`	the current age is compatible with the age at the current track position
`#Error: zero mass`	one of the star has no mass

References

Abt, H. A. (1983). Normal and abnormal binary frequencies. ARA&A, 21:343–372.

Allen, C. and Santillan, A. (1991). An improved model of the galactic mass distribution for orbit computations. Rev. Mexicana Astron. Astrofis., 22:255–263.

Brott, I., de Mink, S. E., Cantiello, M., Langer, N., de Koter, A., Evans, C. J., Hunter, I., Trundle, C., and Vink, J. S. (2011). Rotating massive main-sequence stars. I. Grids of evolutionary models and isochrones. A&A, 530:A115.

Dewi, J. D. M. and Tauris, T. M. (2000). On the energy equation and efficiency parameter of the common envelope evolution. A&A, 360:1043–1051.

Heggie, D. C. (1975). Binary evolution in stellar dynamics. MNRAS, 173:729–787.

Hobbs, G., Lorimer, D. R., Lyne, A. G., and Kramer, M. (2005). A statistical study of 233 pulsar proper motions. MNRAS, 360:974–992.

Kroupa, P. (2008). Initial Conditions for Star Clusters. In Aarseth, S. J., Tout, C. A., and Mardling, R. A., editors, *The Cambridge N-Body Lectures*, volume 760 of *Lecture Notes in Physics, Berlin Springer Verlag*, page 181.

Kruckow, M. U., Tauris, T. M., Langer, N., Kramer, M., and Izzard, R. G. (2018). Progenitors of gravitational wave mergers: Binary evolution with the stellar grid based code ComBinE. *ArXiv e-prints*.

Kuiper, G. P. (1935). Problems of Double-Star Astronomy. I. PASP, 47:15.

Olive, K. A. and Particle Data Group (2014). Review of Particle Physics. *Chinese Physics C*, 38(9):090001.

Salpeter, E. E. (1955). The Luminosity Function and Stellar Evolution. ApJ, 121:161.

Sana, H., de Mink, S. E., de Koter, A., Langer, N., Evans, C. J., Gieles, M., Gosset, E., Izzard, R. G., Le Bouquin, J.-B., and Schneider, F. R. N. (2012). Binary Interaction Dominates the Evolution of Massive Stars. *Science*, 337:444.

Scalo, J. M. (1986). The stellar initial mass function. Fund. Cosmic Phys., 11:1–278.

Soberman, G. E., Phinney, E. S., and van den Heuvel, E. P. J. (1997). Stability criteria for mass transfer in binary stellar evolution. A&A, 327:620–635.

Szécsi, D., Langer, N., Yoon, S.-C., Sanyal, D., de Mink, S., Evans, C. J., and Dermine, T. (2015). Low-metallicity massive single stars with rotation. Evolutionary models applicable to I Zwicky 18. A&A, 581:A15.

Tauris, T. M., Langer, N., and Podsiadlowski, P. (2015). Ultra-stripped supernovae: progenitors and fate. MNRAS, 451:2123–2144.

Tauris, T. M. and Takens, R. J. (1998). Runaway velocities of stellar components originating from disrupted binaries via asymmetric supernova explosions. A&A, 330:1047–1059.

Voss, R. and Tauris, T. M. (2003). Galactic distribution of merging neutron stars and black holes - prospects for short gamma-ray burst progenitors and LIGO/VIRGO. MNRAS, 342:1169–1184.

List of Figures

1 Flowchart 13

List of Tables

1 Constants 14
2 `t_system` components 15
3 `t_star` components 15
4 Stages 16
5 `t_SN` components 16
6 `t_HRD` components 17
7 `t_hist` components 17
8 Phases 19
9 User functions 23
10 Physical checks 25
11 Warning messages 27
12 Error messages 30