Michela Mapelli

Structure, compilation and running of STARLAB

Michela Mapelli

Structure, compilation and running of STARLAB

MOST IMPORTANT INGREDIENT: stay calm, breath, don't panic, don't kill yourself, don't kill your office mates

OUTLINE:

1* software for collisional/collisionless dynamics

2* definition and structure of starlab
http://www.sns.ias.edu/~starlab/overview/
http://www.sns.ias.edu/~starlab/structure/

3* the outputs
http://www.sns.ias.edu/~starlab/internals/

4* the dynamics: KIRA
http://www.sns.ias.edu/~starlab/kira/

5* the stellar evolution: SEBA

6* compilation and installation with/without GPU

7* writing initial conditions
http://www.sns.ias.edu/~starlab/examples/

8* running with PBS

1) Simulating collisional systems

You must resolve SINGLE STARS (softening based codes cannot be used)
Solving (i) equations of motion and
possibly ***(ii) stellar and binary evolution***

(i) EQUATIONS of MOTION:

$$\ddot{\vec{r}}_i = -G \sum_{j \neq i} m_j \frac{\vec{r}_i - \vec{r}_j}{|\vec{r}_i - \vec{r}_j|^3}$$

Or

$$\begin{cases} \dot{\vec{r}}_i & = \vec{v}_i \\ \dot{\vec{v}}_i & = -G \sum_{j \neq i} m_j \frac{\vec{r}_i - \vec{r}_j}{|\vec{r}_i - \vec{r}_j|^3} \end{cases}$$

1) Simulating collisional systems

You must resolve SINGLE STARS (softening based codes cannot be used)
Solving (i) equations of motion

→ **DIRECT N-BODY CODES**

1* Forces on binaries are stronger and change more frequently
→binaries need to be updated more frequently than single stars
→ we need a criterion for different timesteps
Timesteps for BINARIES and THREE-BODY ENCOUNTERS
<< timesteps for other bodies!

2* Solve Newton's equations for EACH star directly → scale as N^2
+ relaxation time scales as N
→time complexity $t_{CPU} \propto N^3$

(cfr with tree codes and Monte Carlo $\propto N \ln N$)

1) Simulating collisional systems

INTEGRATION SCHEME:

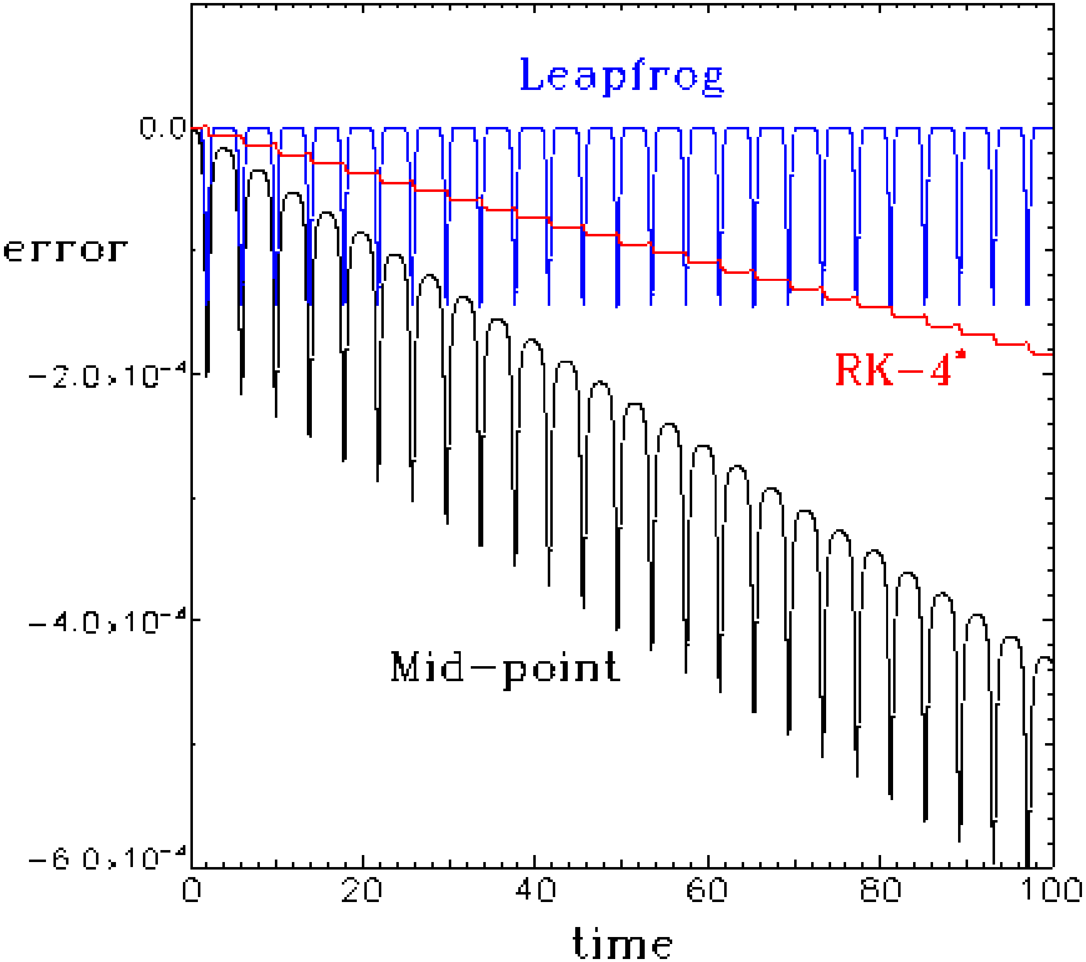
Usage of a high-order integrator (>~4) often with a predictor-corrector scheme, because needs HIGH PRECISION on a SMALL TIME DURATION

EXAMPLEs: better Hermite than Leapfrog

Leapfrog is SYMPLECTIC: solves correctly a Hamiltonian (even if an APPROXIMATED Hamiltonian)

i.e. is TIME REVERSIBLE → good solution on a long time-scale (for conservation of angular momentum, energy, etc), but with large errors on the single timesteps

Hermite is NOT symplectic, but is 4th order on each timestep: much more accurate on single timesteps



1) Simulating collisional systems

2nd order **LEAPFROG** kick-drift-kick (KDK)
or drift-kick-drift (DKD).

Drift=operation that changes only position
Kick= operation that changes only velocity

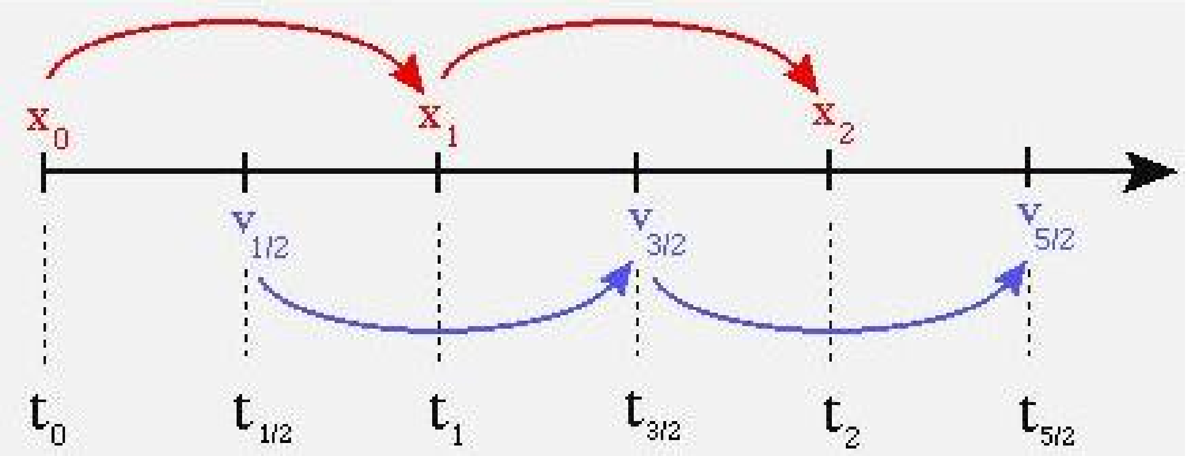
NEEDS an intermediate (auxiliary) quantity (velocity in the KDK, position in the DKD) that will be corrected with a second operation:

1) Simulating collisional systems

D $$x_{n+\frac{1}{2}} = x_n + v_n \frac{\Delta t}{2}$$

K $$v_{n+1} = v_n + a(x_{n+\frac{1}{2}})\,\Delta t$$

D $$x_{n+1} = x_{n+\frac{1}{2}} + v_{n+1} \frac{\Delta t}{2}$$



K $$v_{n+\frac{1}{2}} = v_n + a(x_n) \frac{\Delta t}{2}$$

D $$x_{n+1} = x_n + v_{n+\frac{1}{2}}\,\Delta t$$

K $$v_{n+1} = v_{n+\frac{1}{2}} + a(x_{n+1}) \frac{\Delta t}{2}$$

Hermite 4th order equations (with prediction-correction): Based on **JERK** (time derivative of acceleration)

$$\vec{a}_i = G \sum_{j \neq i} \frac{M_j}{r_{ji}^3} \vec{r_{ij}} \qquad \frac{d\vec{a}_i}{dt} = \vec{j}_i = G \sum_{j \neq i} M_j \left[\frac{\vec{v}_{ji}}{r_{ji}^3} - 3 \frac{(\vec{r}_{ji} \cdot \vec{v}_{ji})\ \vec{r}_{ji}}{r_{ji}^5} \right]$$

Let's start from 4th order derivative of Taylor expansion:

$$\begin{cases} x_1 = x_0 + v_0\,\Delta t + \frac{1}{2}\,a_0\,\Delta t^2 + \frac{1}{6} j_0\,\Delta t^3 + \frac{1}{24} \dot{j}_0\,\Delta t^4 & (1) \\ v_1 = v_0 + a_0\,\Delta t + \frac{1}{2} j_0\,\Delta t^2 + \frac{1}{6} \dot{j}_0\,\Delta t^3 + \frac{1}{24} \ddot{j}_0\,\Delta t^4 & (2) \\ a_1 = a_0 + j_0\,\Delta t + \frac{1}{2} \dot{j}_0\,\Delta t^2 + \frac{1}{6} \ddot{j}_0\,\Delta t^3 & (3) \\ j_1 = j_0 + \dot{j}_0\,\Delta t + \frac{1}{2} \ddot{j}_0\,\Delta t^2 & (4) \end{cases}$$

We use equations (3) and (4) to eliminate the 1st and 2nd derivative of jerk in equations (1) and (2). We obtain

$$x_1 = x_0 + \frac{1}{2}(v_0 + v_1)\,\Delta t + \frac{1}{12}(a_0 - a_1)\,\Delta t^2 + O(\Delta t^5) \qquad (5)$$

$$v_1 = v_0 + \frac{1}{2}(a_0 + a_1)\,\Delta t + \frac{1}{12}(j_0 - j_1)\,\Delta t^2 + O(\Delta t^5) \qquad (6)$$

WHICH ARE 4th order equations, **IMPLICIT** for a_1, v_1 and j_1

DOUBLE TRICK:

1) we use the 3rd order Taylor expansion to PREDICT x_1 and v_1

$$x_{p,1} = x_0 + v_0\,\Delta t + \frac{1}{2}\,a_0\,\Delta t^2 + \frac{1}{6}\,j_0\,\Delta t^3 \qquad v_{p,1} = v_0 + a_0\,\Delta t + \frac{1}{2}\,j_0\,\Delta t^2$$

and we use these PREDICTIONS to evaluate PREDICTED acceleration and jerk ($a_{p,1}$ and $j_{p,1}$), from Newton's formula.

We then substitute $a_{p,1}$ and $j_{p,1}$ into equations (5) and (6):

$$x_1 = x_0 + \frac{1}{2}\,(v_0 + v_{p,1})\,\Delta t + \frac{1}{12}\,(a_0 - a_{p,1})\,\Delta t^2$$

$$v_1 = v_0 + \frac{1}{2}\,(a_0 + a_{p,1})\,\Delta t + \frac{1}{12}\,(j_0 - j_{p,1})\,\Delta t^2$$

2) It can be shown that this result is only 3rd order, but there is a dirty trick to make it 4th order: we calculate v_1 first and then use the result into x_1

$$v_1 = v_0 + \frac{1}{2}\,(a_0 + a_{p,1})\,\Delta t + \frac{1}{12}\,(j_0 - j_{p,1})\,\Delta t^2$$

$$x_1 = x_0 + \frac{1}{2}\,(v_0 + v_1)\,\Delta t + \frac{1}{12}\,(a_0 - a_{p,1})\,\Delta t^2$$

Further integration trick: **REGULARISATION** by Sverre Aarseth

Definition: *mathematical trick to remove the singularity in the Newtonian law of gravitation for two particles which approach each other arbitrarily close.*

Is the same as softening????

NO, it is a **CHANGE OF VARIABLES**, that removes singularity without affecting the physics

1. Regularisation for binaries and 3-body encounters:

Kustaanheimo-Stiefel (KS) regularisation, from Levi-Civita (1956) regularisation (see Funato et al. 1996, astro-ph/9604025)

*Change from coordinates to offset coordinates (Aarseth): CM and relative particle

* IDEA: a Kepler orbit is transformed into a **harmonic oscillator** and the number of steps needed for the integration of an orbit is reduced significantly

3. Regularisation for multi-body systems:

CHAIN regularisation by Aarseth

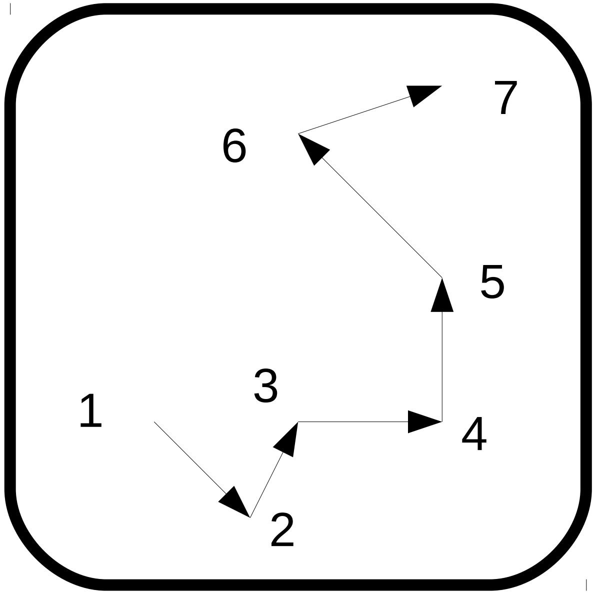
(e.g. Mikkola & Aarseth 1993, Celestial Mechanics and Dynamical Astronomy, 57, 439)

-calculate distances between an active object (e.g. binary) and the closest neighbours

-find vectors that minimize the distances

-use the resulting polygon to change coordinates

-calculate forces with new coordinates



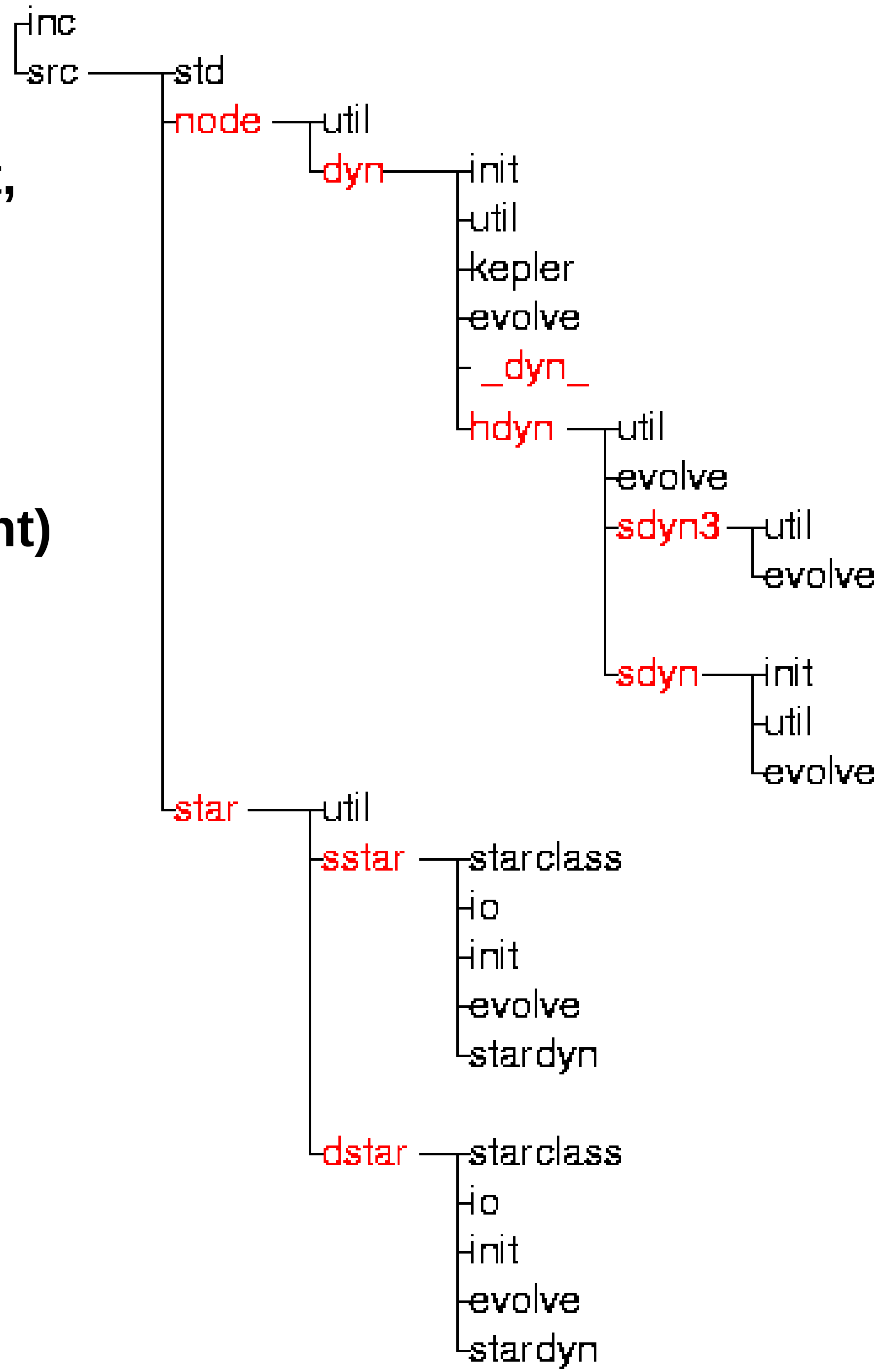
2) definition and structure of starlab

http://www.sns.ias.edu/~starlab/overview/
http://www.sns.ias.edu/~starlab/structure/

*** not a code but a software environment,**
a collection of modular software tools:
generate ICs (plummer, king),
dynamics, stellar evolution,
binary evolution,
plot tools (better not use),
analysis tools (statistics..some important)

***c++, something in fortran (DON'T USE)**
→ CLASSES!!!

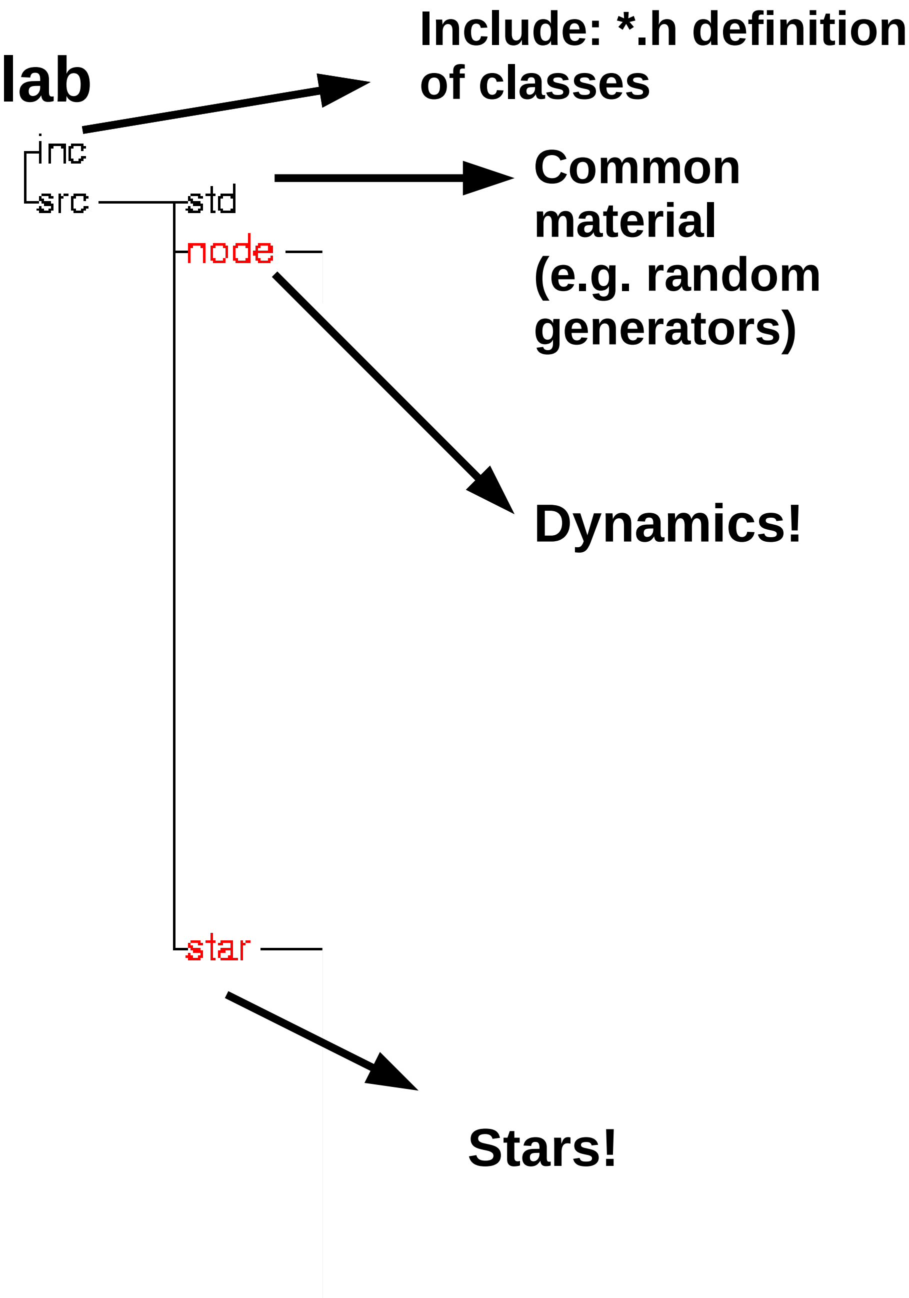
***complex, directory structure:**



2) definition and structure of starlab

http://www.sns.ias.edu/~starlab/overview/
http://www.sns.ias.edu/~starlab/structure/

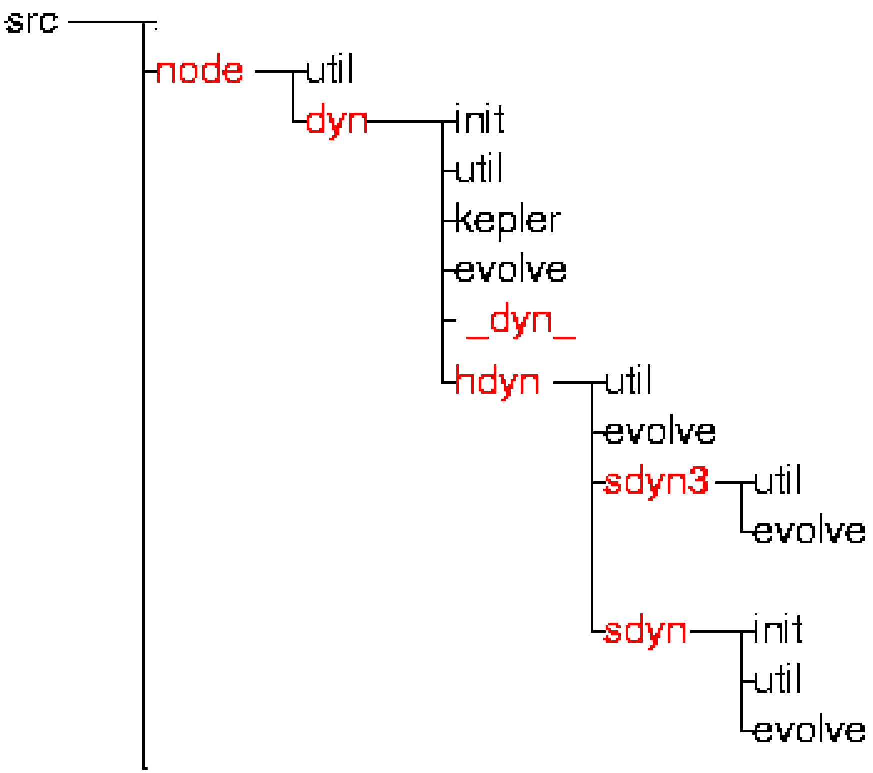
***complex, directory structure:**



2) definition and structure of starlab

http://www.sns.ias.edu/~starlab/overview/

http://www.sns.ias.edu/~starlab/structure/



*** dynamics:**

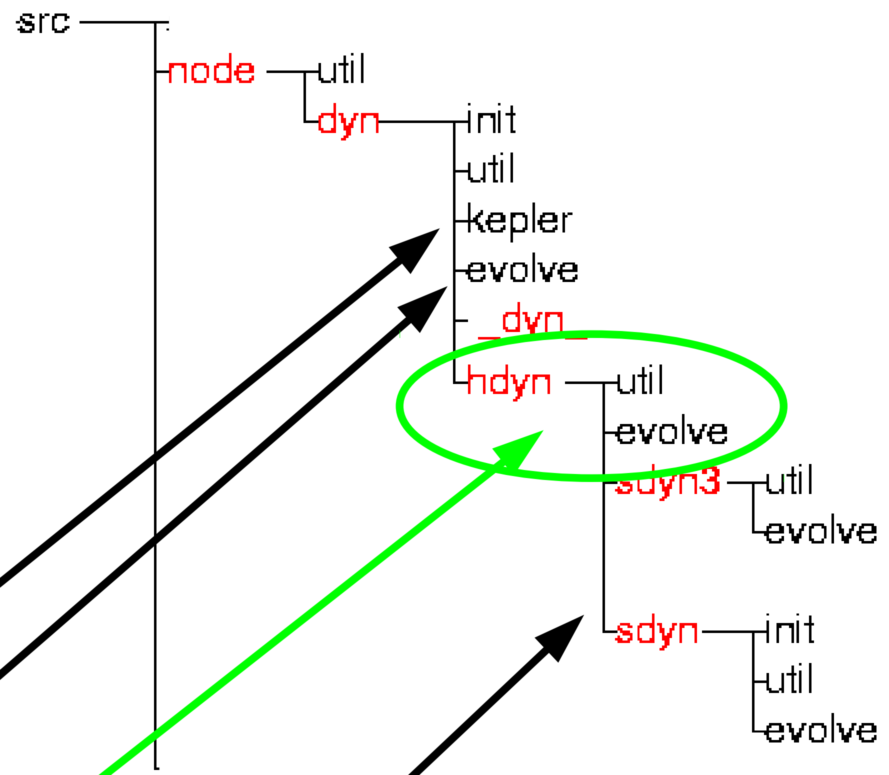
init: contain tool for initialization

util: data analysis or plot

evolve: evolve dynamics in time

2) definition and structure of starlab

http://www.sns.ias.edu/~starlab/overview/
http://www.sns.ias.edu/~starlab/structure/



*** dynamics:**

init: contain tool for initialization
(src/node/dyn/init/makeking.C)

util: data analysis or plot

evolve: evolve dynamics in time

Kepler: only 2-body Keplerian

Only leapfrog

HDYN: high-res dynamics
KIRA INTEGRATOR
./src/node/dyn/hdyn/evolve/kira.C

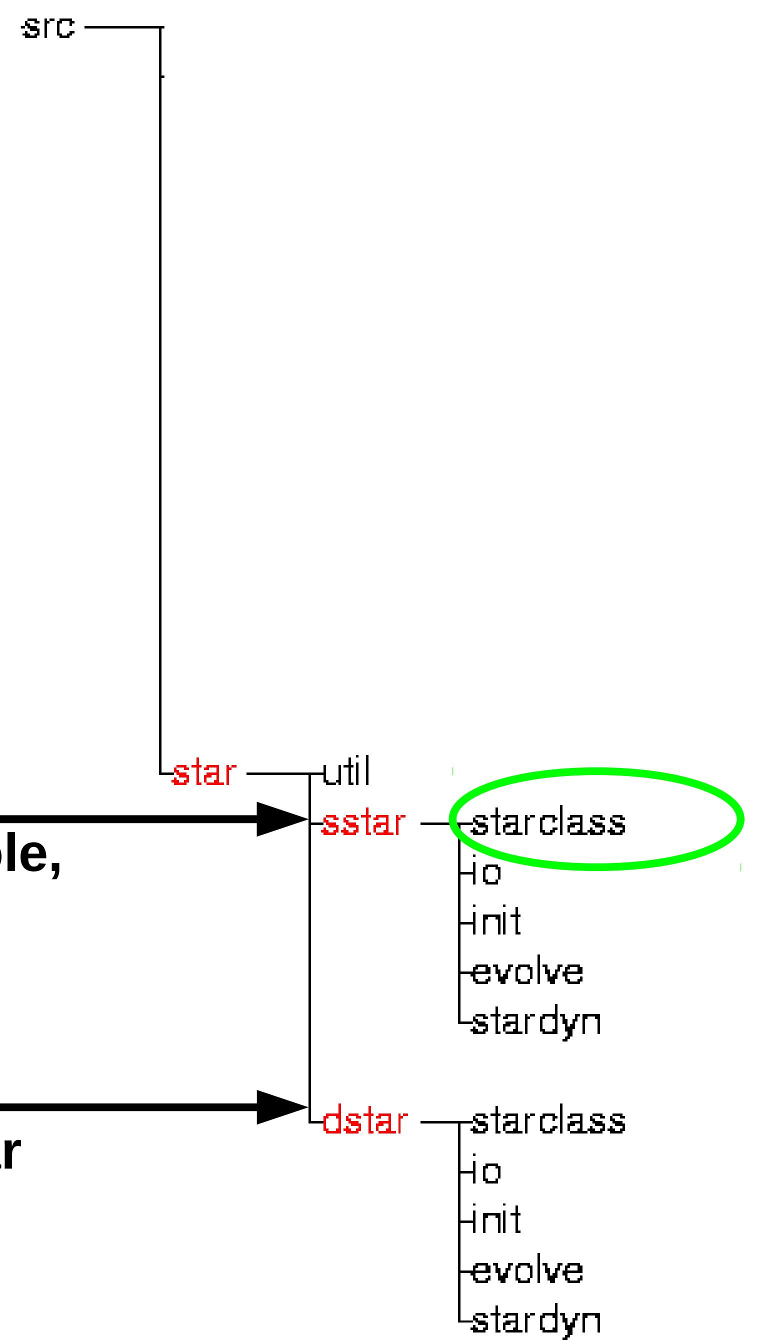
only 3-body scattering

2) definition and structure of starlab

http://www.sns.ias.edu/~starlab/overview/
http://www.sns.ias.edu/~starlab/structure/

*** stars:**

- **init: contain tool for initialization**
- **util: data analysis or plot**
- **evolve: evolve in time star or binary**
- **io: input output of star data**
- **sstar: single stars**
  - **Class: single star,**
  - **derived class: MS star, black hole, hyper-giant, etcetc**
  - **In starclass/**
- **dstar: double star**
  - **starclass: only class double star**



2) definition and structure of starlab

http://www.sns.ias.edu/~starlab/structure/

- CLASS: description of structure+ its functions
- an OBJECT belongs to a class if it is DEFINED as member of the class → star a;

EACH PARTICLE + root belongs to the node class (include/node.h)

```
class node {
        static node* root;  // Global address of the root node.
        long int node_flag; // Indicator of valid node (for internal
                            // bookkeeping purposes only)
        int index;          // Nodes can be numbered,
        char * name;        // or they can receive individual names.
        real mass;
        node * oldest_daughter;    // Define the node's place in
        node * elder_sister;       // the tree.
        node * younger_sister;
        story * log_story;  // Log story is a generalized scratchpad.
        story * dyn_story;  // The dyn story is a placeholder for
                            // dynamical information not recognized by
                            // a program -- this allows the information
                            // to be preserved and passed down a pipe.
        hydrobase * hbase;  // hydrobase is the class underlying all
                            // classes that handle hydrodynamics.
        starbase * sbase;   // starbase is the class underlying all
                            // classes that handle stellar evolution.
    }
```

2) definition and structure of starlab

http://www.sns.ias.edu/~starlab/structure/

EACH PARTICLE + root belongs to the node class

If dynamics is defined, the dyn class is derived from node (include/dyn.h) HEREDITARIETY

```
class dyn : public node {
        static real system_time;
        static bool use_sstar;  // Single star evolution if true.
        vector pos;             // Position (3-D Cartesian vector).
        vector vel;             // Velocity: (d/dt) pos.
        vector acc;             // Acceleration: (d/dt) vel.
        kepler * kep;           // Pointer to a kepler orbit object.
    }
```

NB: mass belongs to node, pos, vel, acc only to dyn

2) definition and structure of starlab

http://www.sns.ias.edu/~starlab/structure/

EACH PARTICLE + root belongs to the node class

If dynamics is defined, the dyn class is derived from node

If high-res hdyn class is derived from _dyn_ which is derived from dyn (include/hdyn.h, include/_dyn_.h)

```
class _dyn_ : public dyn {
        real time;                  // Individual particle time
        real timestep;              // and time step.
        real pot;                   // Potential.
        vector jerk;                // (d/dt) acc
        vector pred_pos;            // Predicted variables for use in the
        vector pred_vel;            // standard predictor-corrector scheme.
        real t_pred;                // Time of prediction.
        real radius;                // Effective (or actual) radius.
    }
```

2) hdyn

Tidal field

Binary evolution

Time dynamical Integration (e.g. softening)

Removal of escapers

Infos on perturbers (see kira)

```
class hdyn : public _dyn_ {
        //-----------------------------------------------------------------
        // Global variables:
        // Tidal field:
        static int  tidal_type; // none, point-mass, halo, disk
        static real alpha1;     // tidal field is conventionally taken
        static real alpha3;     // to be (-alpha1*x, 0, -alpha3*z)
        static real omega;      // system angular speed
        // Binary evolution:
        static bool use_dstar;  // binary evolution if true
        // Stellar encounters and mergers:
        static real stellar_encounter_criterion_sq;
        static real stellar_merger_criterion_sq;
        static real stellar_capture_criterion_sq;
        // Run-time integration parameters:
        static real eta;        // time step parameter
        static real eps;        // softening length
        static real d_min_sq;   // scale term governing tree adjustment
        static real lag_factor; // squared hysteresis factor
        static real mbar;       // mass scale
        static real gamma2;     // squared threshhold for unperturbed motoin
        static real gamma23;    // gamma^{-2/3}
        static real initial_step_limit; // limit on first time step
        static real step_limit;         // limit on all time steps
        // Escaper removal:
        static real scaled_stripping_radius; // stripping radius for unit mass
        //-----------------------------------------------------------------
        // Variables for unperturbed motion:
        real perturbation_squared;   // Relative perturbation squared.
        real unperturbed_timestep;   // Time step for unpert. motion.
        bool fully_unperturbed;      // True if orbit is fully
                                     // unperturbed.
        // Perturber information:
        int n_perturbers;            // Number of perturbers.
        hdyn** perturber_list;       // Pointer to perturber array.
        bool valid_perturbers;       // True if any particle is
                                     // within the perturbation
                                     // radius and the perturber
                                     // list has not overflowed.
        // Other neighbor information:
        hdyn* nn;                    // Pointer to nearest neighbor.
        real d_nn_sq;                // Distance squared to nn.
        hdyn* coll;                  // Pointer to neighbor whose
                                     // surface is closest to this node.
        real d_coll_sq;              // Distance squared to coll.
        // HARP-3 variables:
        int harp3_index;             // HARP-3 address of this particle.
        real harp3_rnb_sq;           // HARP-3 neighbor sphere radius.
    }
```

2) definition and structure of starlab

http://www.sns.ias.edu/~starlab/structure/

Basic class for stars is starbase (include/starbase.h, for root):

```
class starbase {
        node  * the_node;                       // pointer to associated node
        story * star_story;                     // pointer to star story

        static real m_conv_star_to_dyn;    // mass conversion factor
        static real r_conv_star_to_dyn;    // length conversion factor
        static real t_conv_star_to_dyn;    // time conversion factor
        static bool use_hdyn;                   // true iff binary evolution
                                                // is enabled
    /*mmapelli add on December 30 2012*/
        static real starmetal; /* default is solar metallicity*/
    /*mmapelli add on December 30 2012*/
    }
```

2) definition and structure of starlab

http://www.sns.ias.edu/~starlab/structure/

Basic class for stars is starbase (for root)

For each particle, star class is derived from starbase (include/star/star.h)

```
class star : public starbase {
        // No private or
protected data...
        public:
        .
        .
        .
    }
```

2) definition and structure of starlab

http://www.sns.ias.edu/~starlab/structure/

Basic class for stars is starbase (for root)

For each particle, star class is derived from starbase

If star evolution, single_star class is derived from star (include/star/single_star.h)

```
class single_star : public star {
        int identity;
        stellar_type star_type;
 // main sequence,

 // red giant, etc.
        star_type_spec spec_type[no_of_spec_type];
 // spectral type
        real current_time;
        real relative_age;
        real last_update_age;
        real next_update_age;
        real relative_mass;
        real envelope_mass;
        real core_mass;
        real radius;
        real core_radius;
        real effective_radius;
        real luminosity;
    }
```

2) definition and structure of starlab

http://www.sns.ias.edu/~starlab/structure/

Basic class for stars is starbase (for root)

For each particle, star class is derived from starbase

If star evolution, single_star class is derived from star

Each stellar type derives from single_star
E.g. main_sequence (in include/star/main_sequence.h)

```
class main_sequence : public
single_star {

        real main_sequence_core_mass();
        real
main_sequence_core_radius();
        void adjust_donor_age(const
real mdot);
    }
```

2) definition and structure of starlab

http://www.sns.ias.edu/~starlab/structure/

Basic class for stars is starbase (for root)

For each particle, star class is derived from starbase

If star evolution, single_star class is derived from star

If binary evolution, double_star class is derived from star (include/star/double_star.h)

```
class double_star : public star {
        real semi;
        real eccentricity;
        binary_type bin_type;
        int identity;
        real binary_age;
        real minimal_timestep;
        int donor_identity;
        stellar_type donor_type;
        real donor_timescale;
        mass_transfer_type
current_mass_transfer_type;
    }
```

NB: single_star is associated with leaves, double_star with parent (kira!)

3) the outputs

http://www.sns.ias.edu/~starlab/internals/

comes naturally from the class structure

PARTICLE: each single node

LOG: log story of the node

DYN story of the node

Hydro story of the node

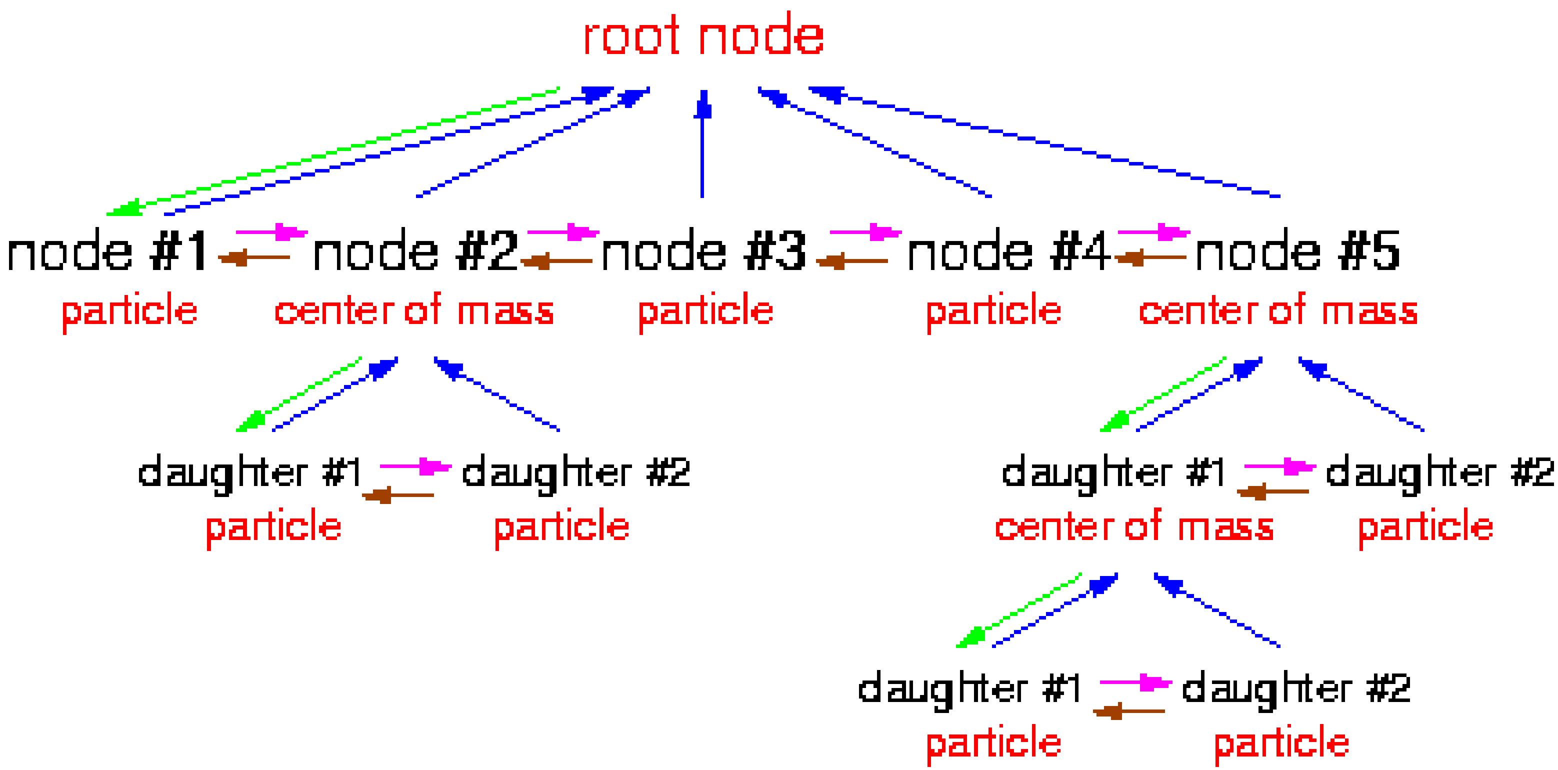
STAR class story

```
(Particle
  i = 4
  N = 1
(Log
Close encounter with black hole #7 at time 10 Myr
)Log
(Dynamics
  m  =  0.5
  r  =  -0.1  0.2   0.5
  v  =  0.3  -0.4  -0.3
)Dynamics
(Hydro
)Hydro
(Star
  Type   =  main_sequence
  T_cur  =  0
  M_rel  =  1
  M_env  =  0.99
  M_core =  0.01
  T_eff  =  6000
  L_eff  =  1
)Star
)Particle
```

3) the outputs

http://www.sns.ias.edu/~starlab/internals/

comes naturally from the class structure



3) the outputs

http://www.sns.ias.edu/~starlab/internals/

comes naturally from the class structure

PARTICLE: can be single (N = 1), or a binary (N=2) with 2 daughter particles, or a root (name=root), or more complicated dependence

```
(Particle        <--------.
  N = 1                   |
(Log                      |
)Log                      |
(Dynamics                 |
  m  =  1                 |
)Dynamics                 |
)Particle        <--------'
(Particle        <--------.
  N = 1                   |
(Log                      |
)Log                      |
(Dynamics                 |
  m  =  1                 |
)Dynamics                 |
)Particle        <--------'
(Particle        <--------.
  N = 1                   |
(Log                      |
)Log                      |
(Dynamics                 |
  m  =  1                 |
)Dynamics                 |
)Particle        <--------'
```

```
(Particle        <----------------.
      N = 2                       |
    (Log                          |
    )Log                          |
    (Dynamics                     |
      m  =  1                     |
    )Dynamics                     |
    (Particle        <--------.   |
      N = 1                   |   |
    (Log                      |   |
    )Log                      |   |
    (Dynamics                 |   |
      m  =  0.5               |   |
    )Dynamics                 |   |
    )Particle        <--------'   |
    (Particle        <--------.   |
      N = 1                   |   |
    (Log                      |   |
    )Log                      |   |
    (Dynamics                 |   |
      m  =  0.5               |   |
    )Dynamics                 |   |
    )Particle        <--------'   |
    )Particle        <----------------'
```

4) kira

http://www.sns.ias.edu/~starlab/kira/
based on 4th order Hermite with corrector/predictor

STEPS:

1. determines which stars need to be updated

2. checks for: reinitialization, log output, escaper removal, termination, snapshot output

3. perform low-order prediction (grape)

4. calculates acceleration/jerk and correct position/velocities (grape)

5. checks for all unperturbed motion

6. checks for collisions and mergers

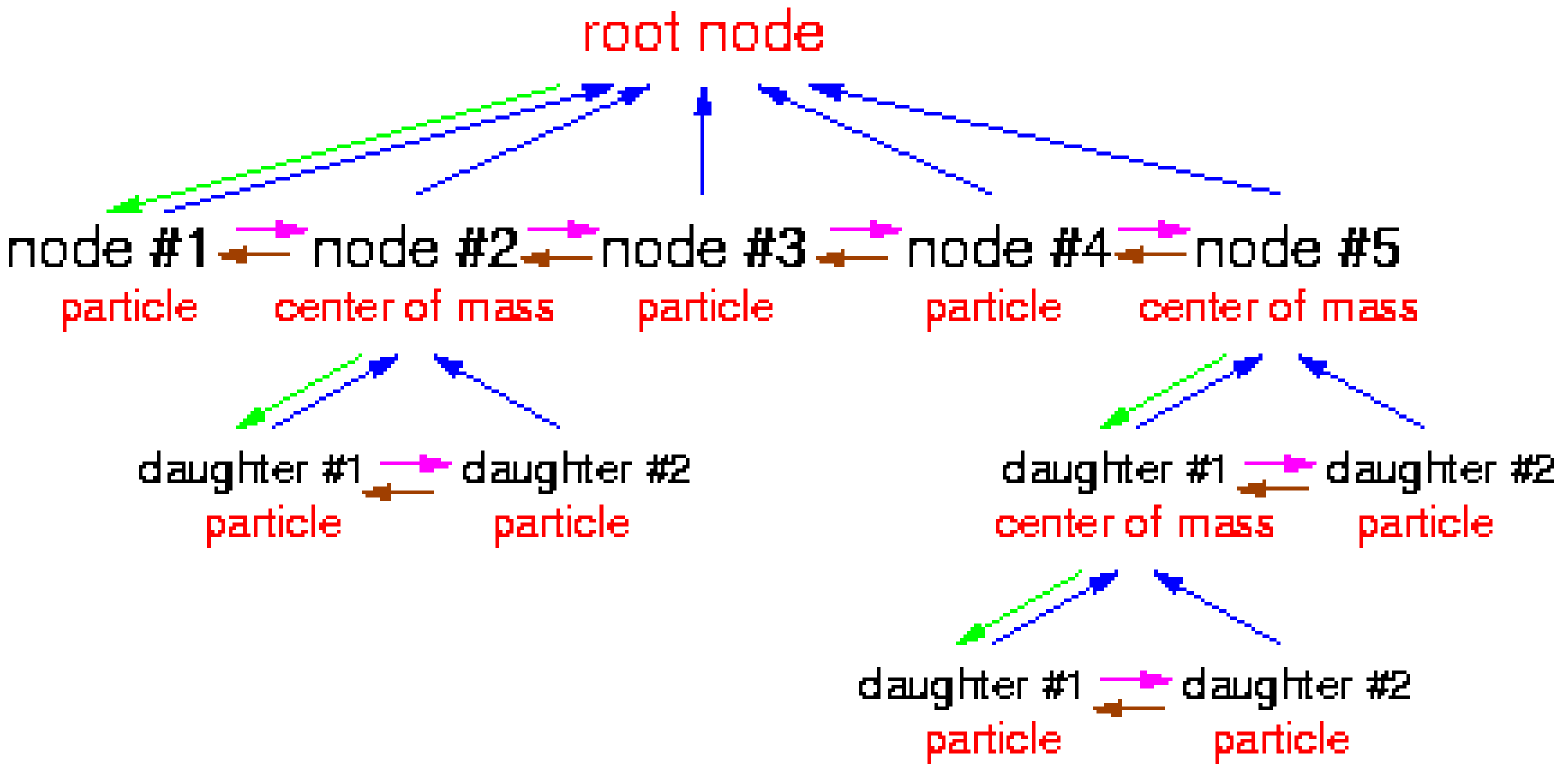
7. checks tree reorganization

8. checks for stellar/binary evolution

4) kira

http://www.sns.ias.edu/~starlab/kira/
based on 4th order Hermite with corrector/predictor

TREE simpler than tree code: leaves are single stars, parents can be binaries or multiples, no more



Forces are computed using direct summation over all other particles in the system; no tree or neighbor-list constructs are used!!!

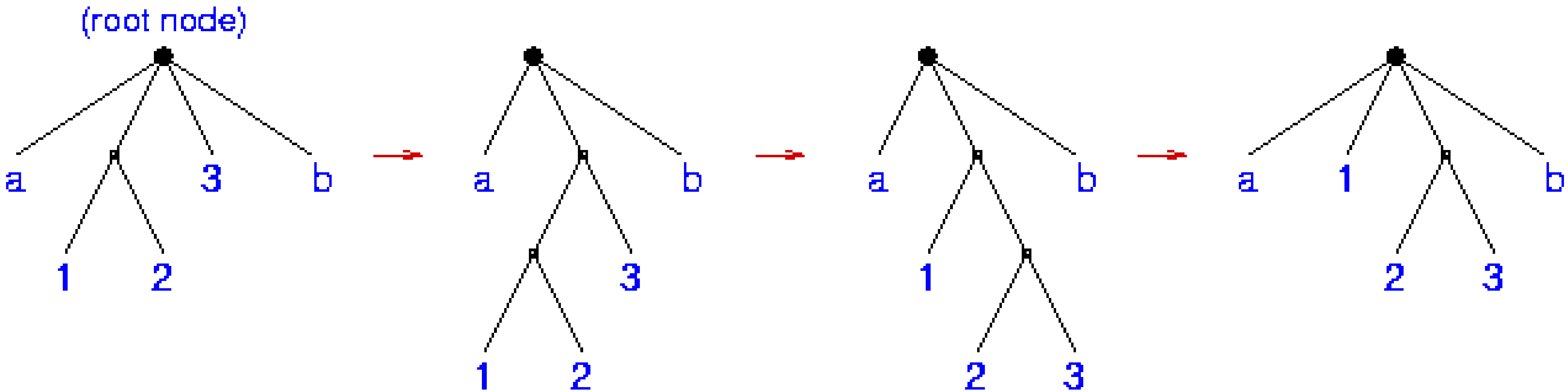
NO O(N logN)

4) kira

http://www.sns.ias.edu/~starlab/kira/
based on 4th order Hermite with corrector/predictor

TREE simpler than tree code: leaves are single stars, parents can be binaries or multiples, no more

Example of a 3-body encounter



PERTURBED binaries (3-body) are splitted into components

UNPERTURBED binaries are evolved ANALYTICALLY

Critical point: how to decide perturber list!!!

4) kira

http://www.sns.ias.edu/~starlab/kira/

based on 4th order Hermite with corrector/predictor

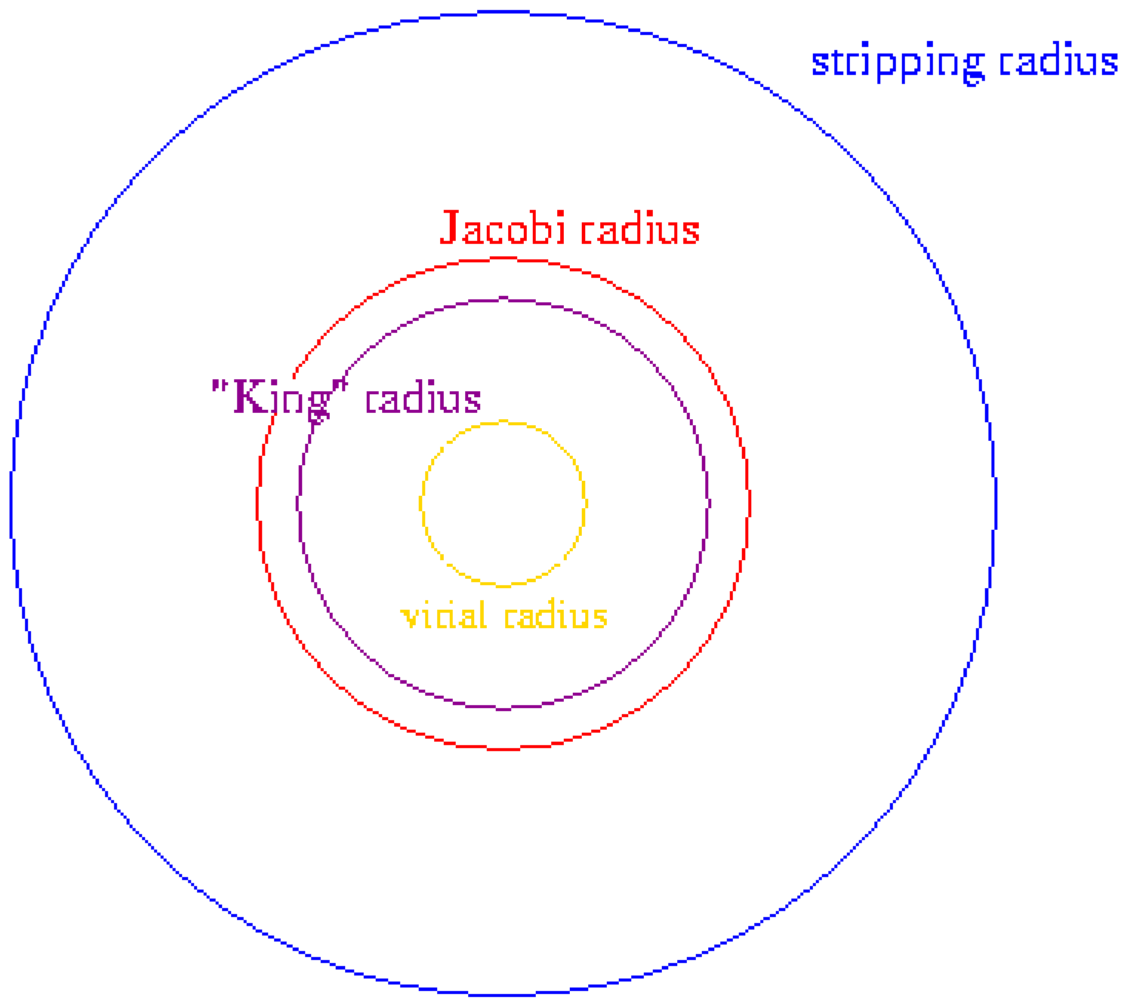
ESCAPER REMOVAL (not in current simulations)

*** Virial radius**

*** King radius:**
cutoff of King model

*** Jacoby radius:**
tidal radius

***stripping radius:**
radius for escaper removal
(eg 2 Jacobi)



4) kira

http://www.sns.ias.edu/~starlab/kira/

RUNNING KIRA

./kira -t 150 -d 1 -D 1 -b 1 -f 0 -n 10 -e 0.000 -B -s 31107

-t number of timesteps

-d log output interval

-D snapshot interval

-b specify frequency of full binary output

-f add analytic formula for internal dynamical friction (should already be accounted for by integrator). 0 means no friction, 1 means friction.
It works only with Plummer & Power-law.
If you use King, you can avoid specifying -f.
WARNING: in old versions of kira -f indicates the minimum energy to form a binary (-f 0.3 means that only binaries with |E|>0.3 kT can form).

-n minimum number of particles (below n terminate simulation, *i.e. if >(N-n) stars escape and are removed, terminate the simulation*)

-e softening

-B with binary evolution (and also star, otherwise -S)

-s random seed (default internal clock)

5) the stellar evolution: SEBA

http://www.sns.ias.edu/~starlab/seba/

Portegies Zwart & Verbunt 1996

proto star (0) Non hydrogen burning stars on the Hyashi track

planet (1) Various types, such as gas giants, etc.; also includes moons.

brown dwarf (2) Star with mass below the hydrogen-burning limit.

main sequence (3) Core hydrogen burning star.

Hypergiant (4) Massive (m>25Msun) post main sequence star with enormous mass-loss rate in a stage of evolution prior to becoming a Wolf-Rayet star.

Hertzsprung gap (5) Rapid evolution from the Terminal-age main sequence to the point when the hydrogen-depleted core exceeds the Schonberg-Chandrasekhar limit.

sub giant (6) Hydrogen shell burning star.

horizontal branch (7) Helium core burning star.

supergiant (8) Double shell burning star.

helium star (9-11) Helium core of a stripped giant, the result of mass transfer in a binary. Subdivided into carbon core (9), helium dwarf (10) and helium giant (11).

white dwarf (12-14) Subdivided into carbon dwarf (12) , helium dwarf (13) and oxygen dwarf (13).

Thorne-Zytkow (15) Shell burning hydrogen envelope with neutron star core.

neutron star (16-18) Subdivided into X-ray pulsar (16), radio pulsar (17) and inert neutron (18) star (m<2Msun).

black hole (19) Star with radius smaller than the event horizon. The result of evolution of massive (m>25Msun) star or collapsed neutron star.

disintegrated (20) Result of Carbon detonation to Type Ia supernova.

5) the stellar evolution: SEBA
http://www.sns.ias.edu/~starlab/seba/
Portegies Zwart & Verbunt 1996

WHAT I CHANGED:

- include/starbase.h → add starmetal
- src/star/sstar/starclass/hertzsprung_gap.C → remove spurious wind (8-20Msun)
- star/sstar/starclass/main_sequence.C → Hurley+ 2000 metal dependent radii
  Vink+2001 winds for MS
- star/sstar/starclass/horizontal_branch.C → remove spurious wind (8-20Msun)
- star/sstar/starclass/single_star.C → winds for MS, WR and LBV
- star/sstar/starclass/sub_giant.C → remove spurious wind (8-20Msun)
- star/sstar/starclass/helium_giant.C → remove Disintegrated stars
  change winds to adapt to WR
- star/sstar/starclass/black_hole.C → insert direct collapse (failed supernova)

SEE YOURSELF with `grep -R mmapelli *`

6) compilation & installation

tar xvfz starlabapr19_2013.tgz

cd starlabapr19_2013/

make clean

./configure

make

make install ← **Copy executables on /usr/bin**

6) compilation & installation, ADVANCED

* if you have grape or GPU+CUDA, put **grape.sh** (optimized for grape or GPU) in **local/** → configure will find it and configure starlab for grape or GPU

otherwise configure will optimize for serial CPU

* if you have the file local/grape.sh but you DO NOT WANT TO COMPILE FOR GRAPE or GPU, change name to grape.sh or configure with

./configure --without-grape

* the only important for us is GPU+CUDA → NEEDs:

1. configure for GPU (no grape) → my file
2. NVIDIA GPU
3. CUDA (https://developer.nvidia.com/cuda-downloads)
4. SAPPORO LIBRARY (Gaburov et al. 2009)

   (http://castle.strw.leidenuniv.nl/software/sapporo.html)

* if you have fortran, please make configure not to use it:

./configure --without-f77

6) compilation & installation, ADVANCED

* on PLX @ cineca use setup_starlab_mm2.sh

```
#!/bin/bash
#PBS -N test1
#PBS -A PROJECTNAME
#PBS -q debug
#PBS -l walltime=0:20:00
#PBS -l select=1:ncpus=1:ngpus=2

module load gnu/4.1.2
module load profile/advanced
module load boost/1.41.0--intel--11.1--binary
#module load boost/1.41.0--gnu--4.1.2
module load cuda/4.0

LD_LIBRARY_PATH=/cineca/prod/compilers/cuda/4.0/none/lib64:/cineca/prod/compilers/cuda/4.0/none/lib:/cineca/prod/lib
raries/boost/1.41.0/intel--11.1--binary/lib:/cineca/prod/compilers/intel/11.1/binary/lib/intel64

export LD_LIBRARY_PATH

cd /plx/userexternal/mmapelli/starlabapr19_2013/
make clean
./configure --without-f77
make
make install
```

*** NB to run setup_starlab_mm2.sh you need to be on the computing nodes:**

qsub setup_starlab_mm2.sh

PROJECTNAME found with saldo -b

6) compilation & installation, ADVANCED

* **on PLX @ cineca you can also compile interactively (e.g. if you debug):**

***Submit an interactive job as* qsub -I**

qsub -I walltime=0:10:00 -l select=1:ncpus=1 -q debug -A projectname

and then type in the shell

```
module load gnu/WHICH_VERSION
module load profile/advanced
module load boost/WHICH_VERSION
module load cuda/WHICH_VERSION
cd /plx/userexternal/USER/starlabYOUR_VERSION/
make clean
./configure --without-f77
make
make install
```

7) writing initial conditions

http://www.sns.ias.edu/~starlab/examples/

Use a sh script (easier and keep memory)

```
./makeking -n 5000 -w 5 -i -u \
     | ./makemass -f 8  -l 0.1 -u 150 \
     | ./makesecondary -f 0.1 -q -l 0.1 \
     | ./add_star -R 1 -Z 0.01 \
     | ./scale -R 1 -M 1\
     | ./makebinary -f 2 -o 1 -l 1 -u 107836.09 \
       > cineca110_bin_N5000_frac01_W5_Z001_IC.txt
```

* makeking: generates a king profile with
  - -n number of centres of mass
  - -w dimensionless central potential
  - -i number the particles sequentially
  - -u leave final N-body system unscaled

  src/node/dyn/init/makeking.C

Useful alternative: makeplummer (**src/node/dyn/init/makeplummer.C**)

7) writing initial conditions

http://www.sns.ias.edu/~starlab/examples/

Use a sh script (easier and keep memory)

```
./makeking -n 5000 -w 5 -i -u \
     | ./makemass -f 8  -l 0.1 -u 150 \
     | ./makesecondary -f 0.1 -q -l 0.1 \
     | ./add_star -R 1 -Z 0.01 \
     | ./scale -R 1 -M 1\
     | ./makebinary -f 2 -o 1 -l 1 -u 107836.09 \
       > cineca110_bin_N5000_frac01_W5_Z001_IC.txt
```

* makemass: generates mass of primary & single stars from IMF
  - -f 1-8: kind of IMF (1 Power-law, 2 Miller & Scalo, 3 Scalo, 4 old Kroupa, 5 DeMarchi, 6 old Kroupa+ 1991, 7 two power law, 8 Kroupa 2001)
  - -l minimum star mass (units of Msun)
  - -u maximum star mass (units of Msun)

src/node/util/makemass.C

7) writing initial conditions

http://www.sns.ias.edu/~starlab/examples/

Use a sh script (easier and keep memory)

```
./makeking -n 5000 -w 5 -i -u \
      | ./makemass -f 8  -l 0.1 -u 150 \
      | ./makesecondary -f 0.1 -q -l 0.1 \
      | ./add_star -R 1 -Z 0.01 \
      | ./scale -R 1 -M 1\
      | ./makebinary -f 2 -o 1 -l 1 -u 107836.09 \
        > cineca110_bin_N5000_frac01_W5_Z001_IC.txt
```

* makesecondary: generates mass of secondary from flat distribution
  -f binary fraction
  -q if present, secondary mass ratio is chosen uniformly
    on [lower_limit, upper_limit]
  -l lower limit secondary mass (if -q in fraction of primary mass)
  -u upper limit secondary mass (if -q in fraction of primary mass)
    If not specified =1

src/node/util/makesecondary.C

7) writing initial conditions

http://www.sns.ias.edu/~starlab/examples/

Use a sh script (easier and keep memory)

```
./makeking -n 5000 -w 5 -i -u \
      | ./makemass -f 8  -l 0.1 -u 150 \
      | ./makesecondary -f 0.1 -q -l 0.1 \
      | ./add_star -R 1 -Z 0.01 \
      | ./scale -R 1 -M 1\
      | ./makebinary -f 2 -o 1 -l 1 -u 107836.09 \
        > cineca110_bin_N5000_frac01_W5_Z001_IC.txt
```

* add_star: generates physical properties of stars (radius)
  - -M   mscale - mass scale for stars. If not set uses Mtot → better!
  - -R   lscale - dynamical size scaling (in parsecs)
    Error if you do not put anything. May be virial radius of cluster or other scale. Suggestion: put 1 (1 parsec=44370956 sun radii), otherwise you lose control on units.
  - -Z   star cluster metallicity (in units of solar=0.019)
    added by MMapelli on December 31 2012

src/star/sstar/init/add_star.C

7) writing initial conditions

http://www.sns.ias.edu/~starlab/examples/

Use a sh script (easier and keep memory)

```
./makeking -n 5000 -w 5 -i -u \
      | ./makemass -f 8  -l 0.1 -u 150 \
      | ./makesecondary -f 0.1 -q -l 0.1 \
      | ./add_star -R 1 -Z 0.01 \
      | ./scale -R 1 -M 1\
      | ./makebinary -f 2 -o 1 -l 1 -u 107836.09 \
        > cineca110_bin_N5000_frac01_W5_Z001_IC.txt
```

* add_star: produces in output

```
(Star
  mass_scale     =  0.000299852183843106945
  size_scale     =  2.25500000000000001e-08
  time_scale     =  3.88903717906355428
  metallicity    =  1
)Star
```

1/Mtot in Msun
1/Rsun in pc NB! BUG!!!
1/tscale in Myr
in Zsun

7) writing initial conditions

http://www.sns.ias.edu/~starlab/examples/

Use a sh script (easier and keep memory)

```
./makeking -n 5000 -w 5 -i -u \
     | ./makemass -f 8  -l 0.1 -u 150 \
     | ./makesecondary -f 0.1 -q -l 0.1 \
     | ./add_star -R 1 -Z 0.01 \
     | ./scale -R 1 -M 1\
     | ./makebinary -f 2 -o 1 -l 1 -u 107836.09 \
       > cineca110_bin_N5000_frac01_W5_Z001_IC.txt
```

* kira + add_star: produces in stderr

```
scale factors taken from input snapshot
    [m]: 3335.0 M_sun
    [R]: 1 pc
    [T]: 0.257133 Myr
```

Mscale =Mtot/Msun
lscale in pc
tscale in Myr

7) writing initial conditions

http://www.sns.ias.edu/~starlab/examples/

Use a sh script (easier and keep memory)

```
./makeking -n 5000 -w 5 -i -u \
     | ./makemass -f 8  -l 0.1 -u 150 \
     | ./makesecondary -f 0.1 -q -l 0.1 \
     | ./add_star -R 1 -Z 0.01 \
     | ./scale -R 1 -M 1\
     | ./makebinary -f 2 -o 1 -l 1 -u 107836.09 \
       > cineca110_bin_N5000_frac01_W5_Z001_IC.txt
```

* scale: generates physical scales for final SC

-R   specify virial radius

in parsecs, if add_star -R 1 →

(1) ./add_star -R 1 | ./scale -R 5 means rvir=5 in units of 1 pc → rvir=5 pc!!

in units of add_star, if add_star -R != 1 →

(2) ./add_star -R 5 | ./scale -R 1 means rvir=1 in units of 5 pc → rvir=5 pc!!

Almost equivalent, (1) easier, (2) gives more physical meaning to timescale

-M   specify star cluster mass

in units of Mtot, if add_star has no -M option →

-M 1 means that mass units in the output file are /Mtot

IMPORTANT THAT SCALE BE AFTER ADD_STAR IF STAR EVOL

src/node/dyn/util/scale.C

7) writing initial conditions

http://www.sns.ias.edu/~starlab/examples/

Use a sh script (easier and keep memory)

```
./makeking -n 5000 -w 5 -i -u \
      | ./makemass -f 8  -l 0.1 -u 150 \
      | ./makesecondary -f 0.1 -q -l 0.1 \
      | ./add_star -R 1 -Z 0.01 \
      | ./scale -R 1 -M 1\
      | ./makebinary -f 2 -o 1 -l 1 -u 107836.09 \
        > cineca110_bin_N5000_frac01_W5_Z001_IC.txt
```

* makebinary: generates orbital properties of primordial binaries
  - -f function select option
    - 1: angular momentum per unit reduced mass
      (L^2 = am[1-e^2]), solar units
    - 2: semi-major axis or peri/apo, solar units
    - 3: energy
  - -o specify interpretation of limits - With -f 2 -o 1: semi-major axis,
  - -l lower limit on selected binary parameter (sma in Rsun)
  - -u upper limit on selected binary parameter (sma in Rsun)

src/node/dyn/init/makebinary.C

7) writing initial conditions
http://www.sns.ias.edu/~starlab/examples/

Last note on units

-units of stdoutput:

In (Dynamics ..)Dynamics units scaled to Mscale, lscale, tscale (note *)

In (Star ..)Star units scaled to Msun, Rsun=6.95e10 cm, Myr

-units in stderr: units scaled to Msun, Rsun=6.95e10 cm, Myr

Note * = lscale is that in stderr ([R]: .. pc)

or 2.255e-8/(value in stdout)

where 2.255e-8=Rsun in pc

8) running with PBS

SEE launch_starlab.sh

`#!/bin/bash`	**Shell**
`#PBS -N bigZ1N9`	**Job name**
`#PBS -A IscrC_GClife2`	**Project name**
`#PBS -q longpar`	**Queue type**
`#PBS -l walltime=24:00:00`	**Time**
`#PBS -l select=1:ncpus=1:ngpus=2`	**1 node, 1 cpu, 2 gpu**
`module load gnu`	**Load modules**
`module load profile/advanced`	
`module load boost`	
`module load cuda`	
`LD_LIBRARY_PATH=/cineca/prod/compilers/cuda/4.0/none/lib64:/cineca/prod/compilers/cuda/4.0/none/lib:/cineca/prod/libraries/boost/1.41.0/intel--11.1--binary/lib:/cineca/prod/compilers/intel/11.1/binary/lib/intel64`	
`export LD_LIBRARY_PATH`	**New library path**
`sh /plx/userexternal/mmapelli/Z001/big_Z1_9.sh`	**Runs big_Z1_9.sh**

8) running with PBS

launch_starlab.sh calls big_Z1_9.sh:

```
echo $PWD
echo $LD_LIBRARY_PATH
/plx/userexternal/mmapelli/Z001/kira -t 500 -d 1 -D 1 -b 1 -f 0.3 \
                    -n 10 -e 0.000 -B -s 1361557926 \
      < $CINECA_SCRATCH/Z1bb/ICs/npppp9_645 \
      > $CINECA_SCRATCH/Z1bb/new_cineca9_bin_N50000_frac00_W5_Z1.txt5 \
      2> $CINECA_SCRATCH/Z1bb/ew_cineca9_bin_N50000_frac00_W5_Z1.txt5
```

To submit launch_starlab.sh

qsub launch_starlab.sh

To see if running (R) or queued (Q)

qstat -u username

To delete if wrong

qdel job_id

9) CREDITS for STARLAB:

* Thank the authors in the acknowledgments (Portegies Zwart, McMillan, Makino, Hut,...)
* Cite Portegies Zwart+ 2001MNRAS.321..199
  Portegies Zwart & Verbunt 1996A&A...309..179P
* If use GPU, thank the authors of Sapporo: Gaburov, Harfst, Portegies Zwart
  and cite Gaburov+ 2009NewA...14..630G
* If use my metallicity-dep. Version
  cite Mapelli+ 2013MNRAS.429.2298M

10) Online material:

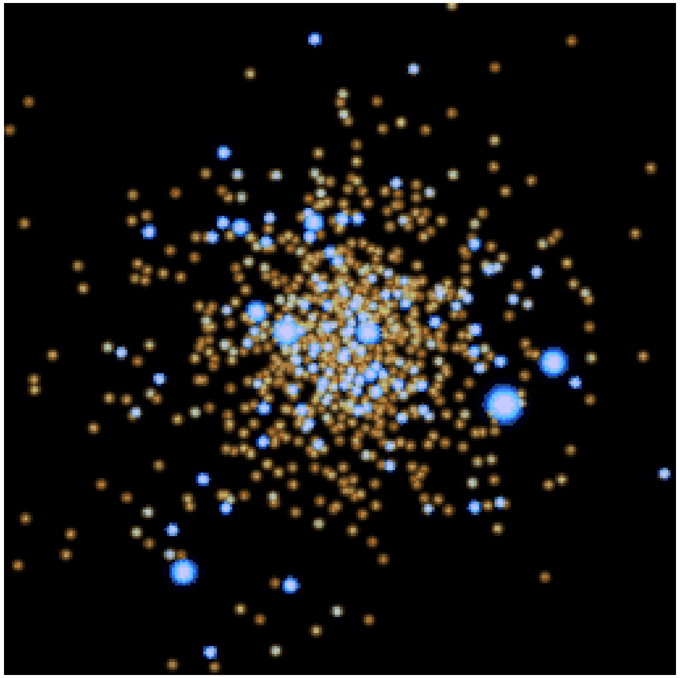
http://www.science.uva.nl/sites/modesta/wiki/index.php/Starlab_tools

and of course

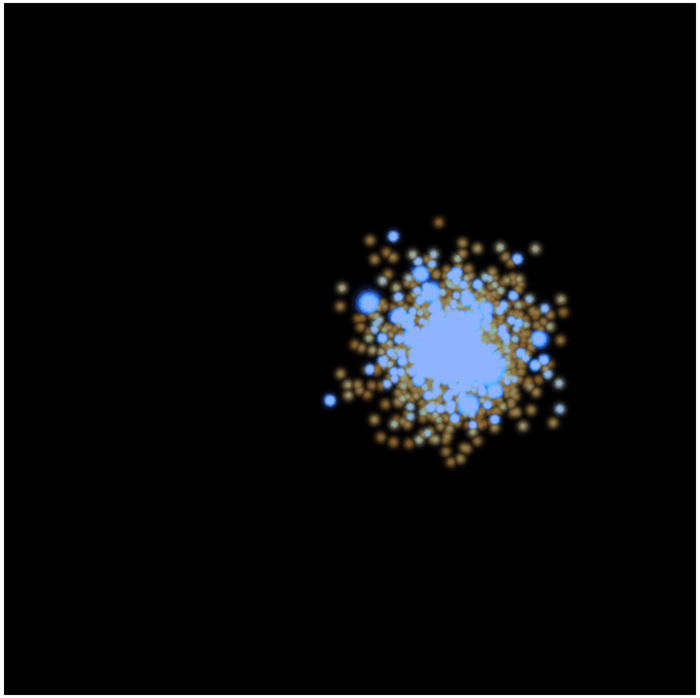
http://www.sns.ias.edu/~starlab/index.html

Download my version and templates

RUN some EXAMPLES on your laptop:



1- isolated star cluster, 2000 stars, NO BINARIES
- **createIC_template.sh: initial conditions**
- **run_template.sh: run with kira**
- **./xstarplot < stdout_N2000_W5_Z01.txt**
  - **on the fly movie**



2- star cluster in Plummer tidal field, 700 stars, no bin.
- **IC_tidal_field_template.sh: initial conditions**
- **run_tidal_field_template.sh: run with kira**
- **./xstarplot -l 20 < stdout_td_N700_W5_Z01.txt**
  - **on the fly movie**