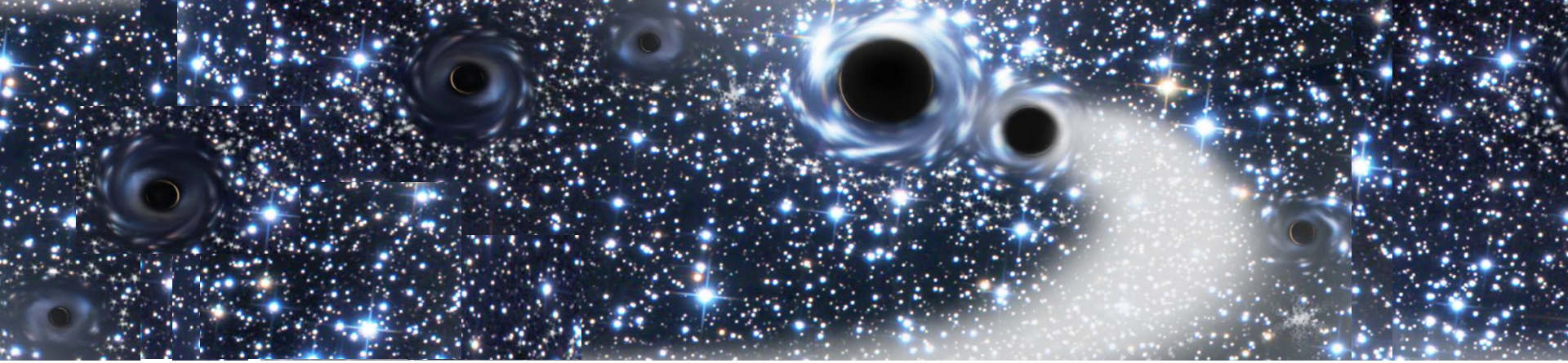


Using computers as telescopes: NBODY6++GPU hands-on

February 19th, 2026

Manuel Arca Sedda

ACME Astrophysics Centre for Multimessenger studies in Europe

Funded by the European Union

MARIE CURIE ACTIONS

MERAC

GRAN SASSO SCIENCE INSTITUTE
SCHOOL OF ADVANCED STUDIES
Scuola Universitaria Superiore

Who am I?

Manuel Arca Sedda (GSSI)
Associate Professor
email: manuel.arcasedda@gssi.it

GRAN SASSO SCIENCE INSTITUTE
SCHOOL OF ADVANCED STUDIES
Scuola Universitaria Superiore

I study the formation and evolution of gravitational wave sources in star clusters and galactic nuclei

What will we learn today?

- Install and run the *N*-body code NBODY6++GPU
- Analyse and interpret simulation outputs

What is the *N*-body problem?

Find solutions to the equations of motion of N bodies for a time t (astrophysics, fluido-dynamics, molecular dynamics …)

$$\ddot{\vec{r}}(t) = \vec{a}(\vec{r}, \vec{v}, t)$$

What is the *N*-body problem?

In Astrophysics (the Gravitational N-body problem) $\ddot{\vec{r}}_i(t) = -\sum_{j \neq i} G m_j \frac{\vec{r}_i - \vec{r}_j}{|\vec{r}_i - \vec{r}_j|^3}$

i.e. a system of 2 x Ndim x N 1st order differential equations

$$\dot{v}_i = -\sum_j \frac{G m_j}{r_{ij}^3} x_{ij}$$

$$\dot{x}_i = v_i$$

What is the *N*-body problem?

In Astrophysics (the Gravitational N-body problem) $\ddot{\vec{r}}_i(t) = -\sum_{j \neq i} Gm_j \frac{\vec{r}_i - \vec{r}_j}{|\vec{r}_i - \vec{r}_j|^3}$

i.e. a system of 2 x Ndim x N 1st order differential equations

$$\dot{v}_i = -\sum_j \frac{Gm_j}{r_{ij}^3} x_{ij}$$

$$\dot{x}_i = v_i$$

*An N-body integrator is a **brute force calculator**, it just integrates the equations of motions of all bodies.*

*The number of operations that must be performed at each time-step is **proportional to N^2**!*

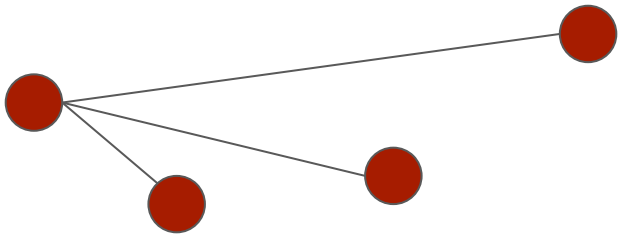
Funded by the European Union

Solving the Gravitational *N*-body problem: divergences

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

Infrared divergence

The gravitational acceleration vanishes at infinity
All particles matter

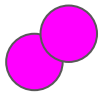


All interactions must be taken into account
- ➢ Calculations scales as ~ NxN
- ➢ Long range effects not negligible

Brute solution: infinite computing power

UV divergence

The gravitational acceleration diverges
as two bodies approach each other



During close approaches acceleration steeply increase
- ➢ Integration errors require small integration steps
- ➢ Spurious ejection of stars

Brute solution: infinite integration time

Solving the Gravitational *N*-body problem: divergences

What do we need?

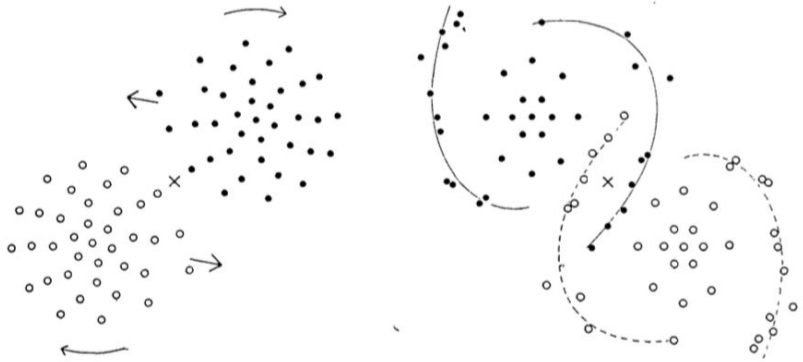
1. high accuracy (close encounters)
2. adjustable time-steps
3. optimisation of computational load

Funded by the European Union

Solving the Gravitational *N*-body problem. A bit of history

1941: **Erik Holmberg** (Lund University) performed the first gravitational N-body simulation using light bulbs:

- N = 74
- each light bulb is a star
- light received can be interpreted as net force
- solve equation of motion based on received light
- move the bulb



1960-1963: **Sebastian von Horner** (University of Heidelberg) and **Sverre Aarseth** (IoA Cambridge University) publish the first direct *N*-body simulations with N = 25-100 members.

Astronomisches Rechen-Institut in Heidelberg
Mitteilungen Serie A Nr. 14

Die numerische Integration des *n*-Körper-Problemes für Sternhaufen I

Von

SEBASTIAN VON HOERNER

Mit 3 Textabbildungen

(Eingegangen am 10. Mai 1960)

Astronomisches Rechen-Institut in Heidelberg
Mitteilungen Serie A Nr. 19

Die numerische Integration des *n*-Körper-Problems für Sternhaufen, II.

Von

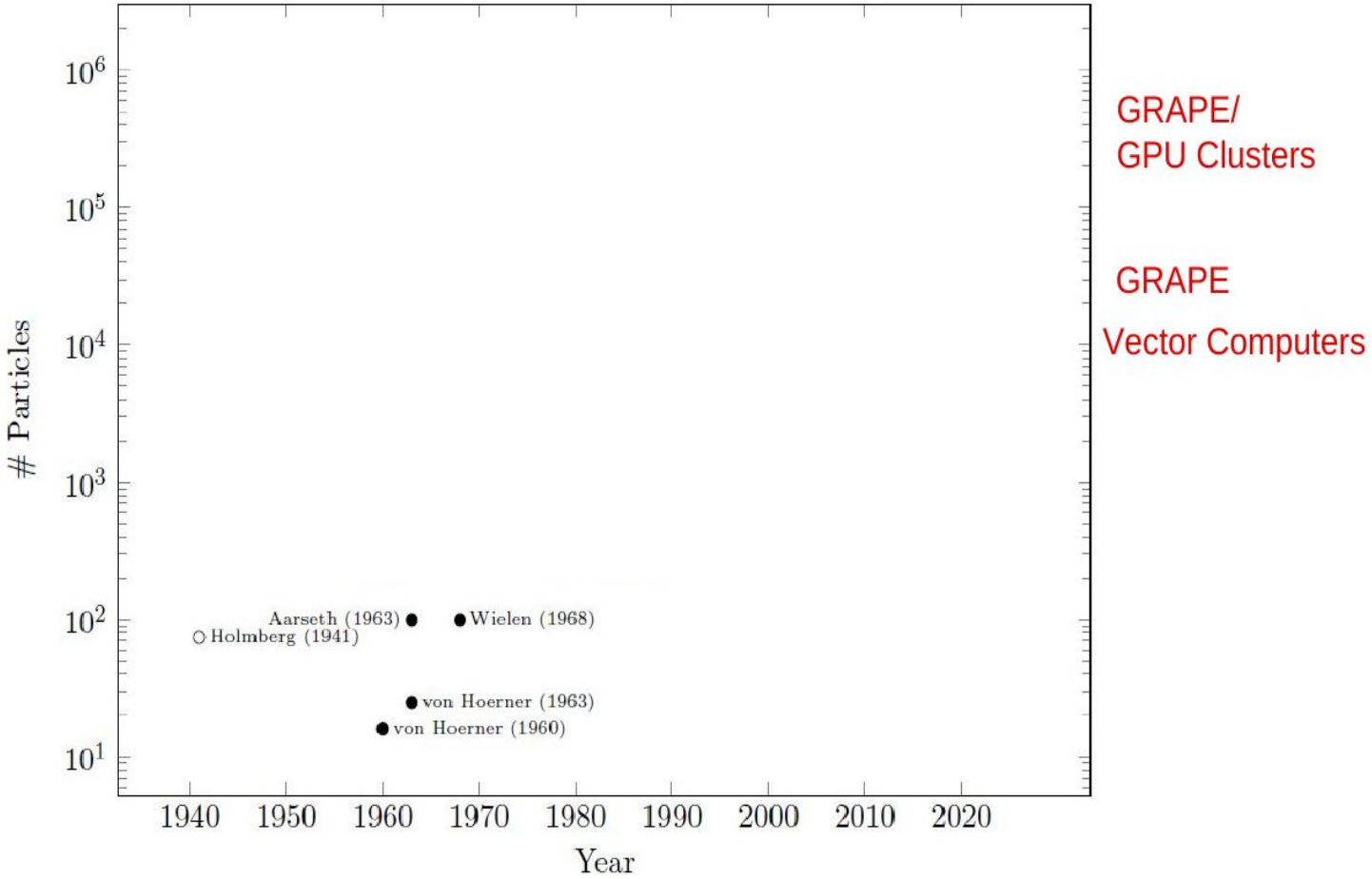
SEBASTIAN VON HOERNER

Mit 10 Textabbildungen

(Eingegangen am 19. November 1962)

Funded by the European Union

Solving the Gravitational *N*-body problem.



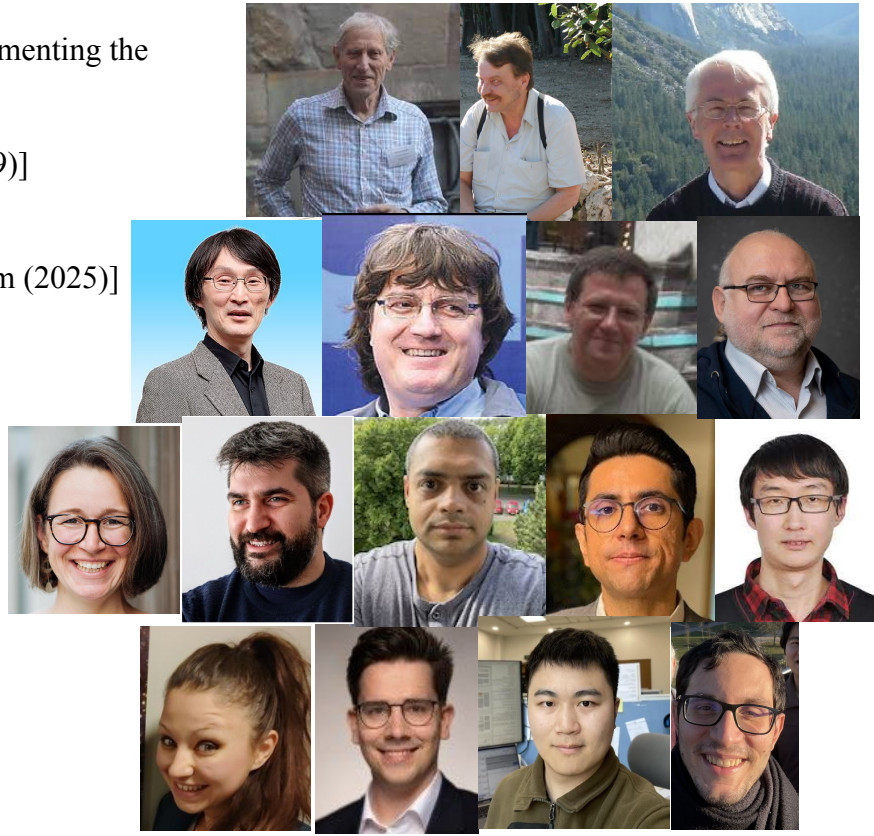
Funded by the European Union

Solving the Gravitational *N*-body problem.

Sverre Aarseth initiated a 60 yr long tradition in N-body simulations by implementing the

NBODY code family series (NBODY1, 2, 3, 4, 6, 6++, 6++GPU, 7..)
[see “From NBODY1 to NBODY6: The Growth of an Industry”, Aarseth (1999)]

Rainer Spurzem and collaborators continue the tradition
[see “From NBODY1 to NBODY7: The Growth of Sverre’s Industry”, Spurzem (2025)]



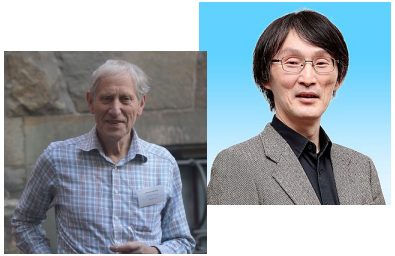
… and many many more

Funded by the European Union

Dynamics of realistic N-body simulations: brute force N-body algorithms

4th order predictor-corrector Hermite scheme (Makino 1991, Makino and Aarseth 1992)

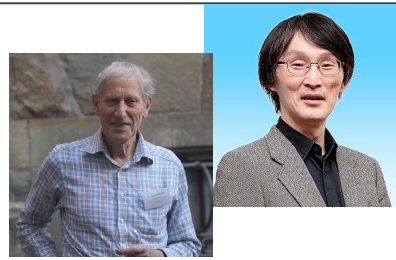
based on the evaluation of the derivative of acceleration (jerk)



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

$$\frac{d\vec{a}_i}{dt} = \vec{j}_i = G \sum_{j \neq i} M_j \left[\frac{\vec{v}_{ij}}{\left(r_{ji}^2 + \epsilon^2\right)^{3/2}} + \frac{3(\vec{v}_{ij} \cdot \vec{r}_{ij})\, \vec{r}_{ij}}{\left(r_{ji}^2 + \epsilon^2\right)^{5/2}} \right]$$

Funded by the European Union



Dynamics of realistic N-body simulations: brute force N-body algorithms

4th order predictor-corrector Hermite scheme (Makino 1991, Makino and Aarseth 1992)

start from 4th order Taylor expansion of pos, vel, acc, jerk:

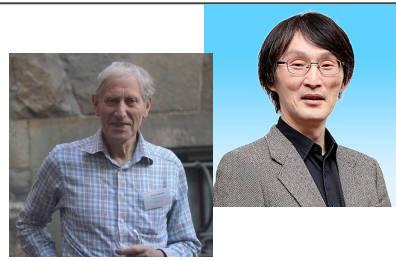
$$\begin{cases} x_1 = x_0 + v_0\,\Delta t + \frac{1}{2}\,a_0\,\Delta t^2 + \frac{1}{6}\dot{j}_0\,\Delta t^3 + \frac{1}{24}\ddot{j}_0\,\Delta t^4 & (1) \\ v_1 = v_0 + a_0\,\Delta t + \frac{1}{2}\,j_0\,\Delta t^2 + \frac{1}{6}\ddot{j}_0\,\Delta t^3 + \frac{1}{24}\dddot{j}_0\,\Delta t^4 & (2) \\ a_1 = a_0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

use (3) and (4) to eliminate the 1st (snap) and 2nd (crackle) derivatives of jerk in (1) and (2) such to obtain a 4th order scheme with 2nd order only terms:

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

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

the method is **implicit:** it depends on a1, j1, v1, which are unknown. We need a way to predict them!



Dynamics of realistic N-body simulations: brute force N-body algorithms

4th order predictor-corrector Hermite scheme (Makino 1991, Makino and Aarseth 1992)

1. **Predictor** use 3rd order Taylor expansion to make the prediction

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

2. **Force evaluation**: we use 1. to calculate $a_{p,1}$ and $j_{p,1}$ from Newton's formula and its derivative

3. **Corrector:** substitute predicted acceleration and jerk into (5) and (6). Beware: the order here is important

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

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

we used $v1$ to updated $x1$, without this trick we should have used the predicted velocity, but this would imply that the method is 3rd order in positions.

Dynamics of realistic N-body simulations: brute force N-body algorithms

Timesteps

In a simulation we can have interactions with typical timescales ~ days - Myr: same time-step for everyone is inefficient

Individual time-step: longer for “unperturbed” particles, shorter for “active” particles

Solution: BLOCK TIME STEP (Aarseth 1985)

1. initial time-step calculated for i-th particle $\Delta t_i = \eta \frac{a_i}{\dot{j}_i}$, with η = 0.01-0.02 typically
2. time $t := t_i + \min(\Delta t_i)$. Particles with time-step min(Δt_i) are called “active”
3. At time t:
    a. Positions and velocities are **predicted for all particles**
    b. Accelerations and jerks are calculated **only for active particles**
    c. Positions and velocities are corrected **only for active particles**

Of course, a single time-step for every particle is unrealistic: too computationally expensive

Dynamics of realistic N-body simulations: brute force N-body algorithms

Timesteps

Solution: BLOCK TIME STEP (Aarseth 1985)

It consists in grouping particles by replacing the individual time-step with time blocks equal to negative powers of 2 $\Delta t_{i,b} = (1/2)^n$

where *n* is chosen such that

$$\left(\frac{1}{2}\right)^n \leq \Delta t_i < \left(\frac{1}{2}\right)^{n-1}$$

this implies that the time is an integer multiple of the timestep, a feature that simplifies the need for synchronising particles at some time.
The individual time-step (which is needed to find the block to which the particle belong) can be calculated as

$$\Delta t_i = \sqrt{\eta \frac{|a_{i,1}|\,|a_{i,1}^{(2)}| + |j_{i,1}|^2}{|j_{i,1}|\,|a_{i,1}^{(3)}| + |a_{i,1}^{(2)}|^2}}$$

Some codes use also higher-order schemes.

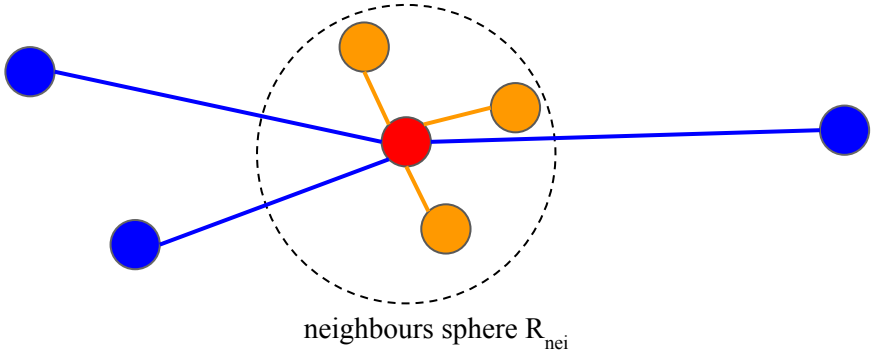
Typical codes are PhiGRAPE, Starlab, PhiGPU, HiGPUs …

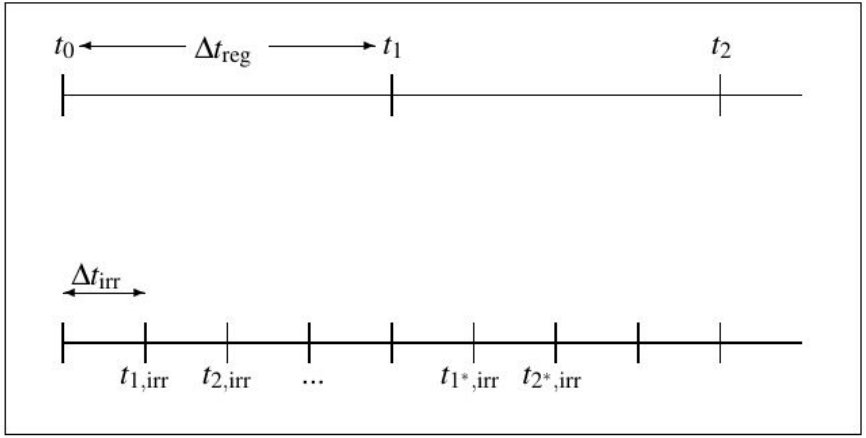
Funded by the European Union

Dynamics of realistic N-body simulations: brute force N-body algorithms

Distance/closeby force calculation: Ahmad Cohen scheme (Ahmad and Cohen 1975)

The force acting on a particle consists of two parts:
one slowly varying due to “distant” stars (**regular force**), and one due to neighbour stars (**irregular force**)





Funded by the European Union

Dynamics of realistic N-body simulations: brute force N-body algorithms

Regularisation techniques for binaries and close encounters

Kustaanheimo-Stiefel (KS) regularisation

Levi-Civita (1956): regularize Kepler orbit of a binary in 2 dimensions

KS (1965): extension to 3 dimensions of Levi-Civita regularization
see Funato et al. (1996, astro-ph/9604025) for improvement
see Waldvogel lecture at Scottish University Summer School in Physics (2007)

www.sam.math.ethz.ch/~joergw/Papers/scotpaper.pdf

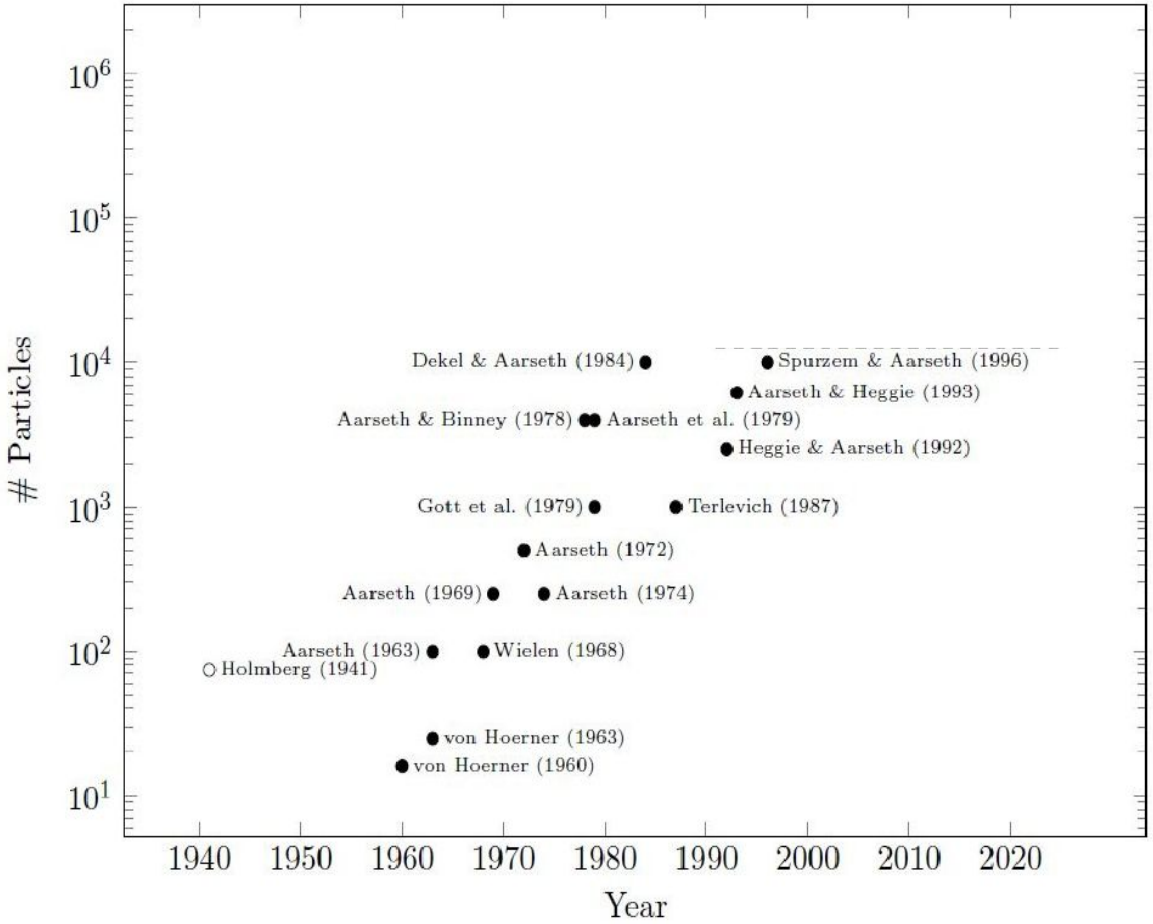
BASIC IDEAS:

*Change from coordinates to offset coordinates: CM and relative particle

$$x_{CM} = \frac{m_1 x_1 + m_2 x_2}{m_1 + m_2} \qquad x_{rel} = x_1 - x_2$$

* a Kepler orbit is transformed into a **harmonic oscillator** and the number of steps needed for the integration of an orbit is reduced significantly & round-off errors reduce too

Solving the Gravitational *N*-body problem.



GRAPE/
GPU Clusters

GRAPE

Vector Computers

ACME
Funded by the European Union

Dynamics of realistic N-body simulations: hardware/software development

An N-body integrator is a brute force calculator, it just integrates the equations of motions of all bodies.

The number of operations that must be performed at each time-step is $O(N^2)$

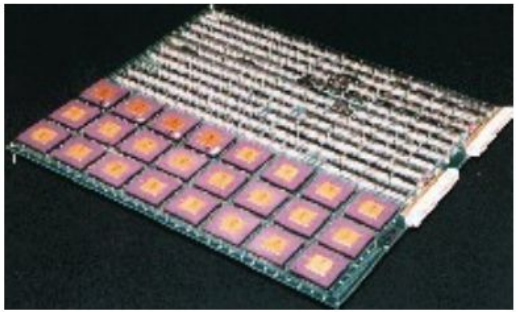
Solution: parallelise the operations

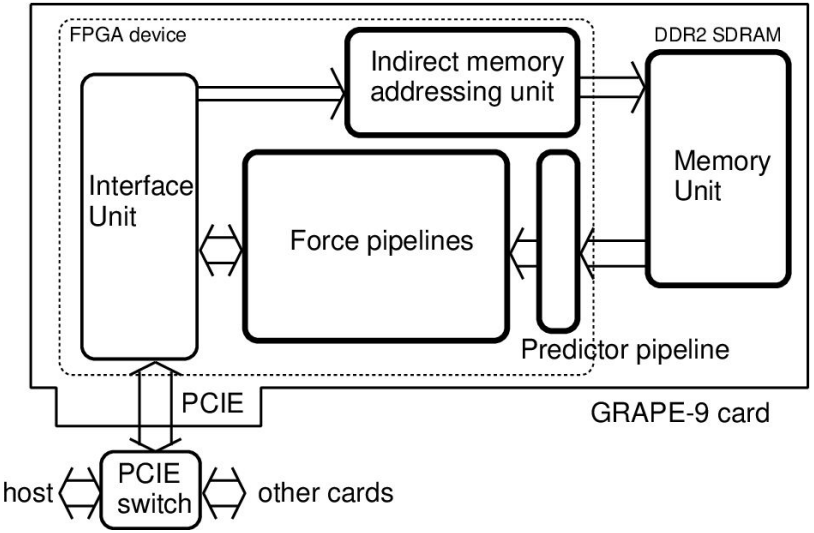
First attempt: GRAPE cards

1989: GRAPE project starts in Tokyo led by Daichiiro Sugimoto and Junichiro Makino

1995: GRAPE-4 becomes the first computer to reach 1 Tflop (# FP operations per second)

Continued almost up to GRAPE-8, but …





Funded by the European Union

Dynamics of realistic N-body simulations: hardware/software development

… but these sober objects came into play,

Graphic Processing Units (GPUs)

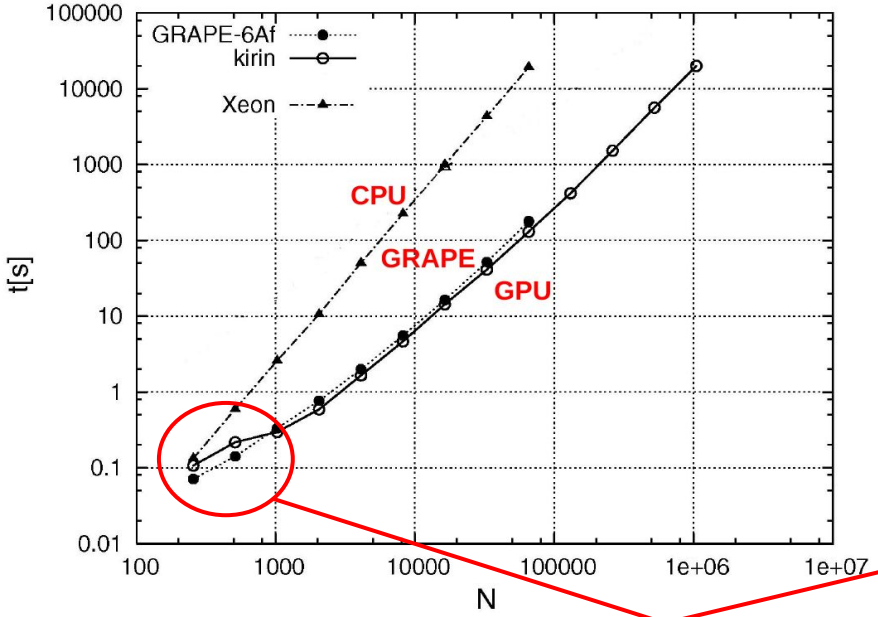
born for videogames market,

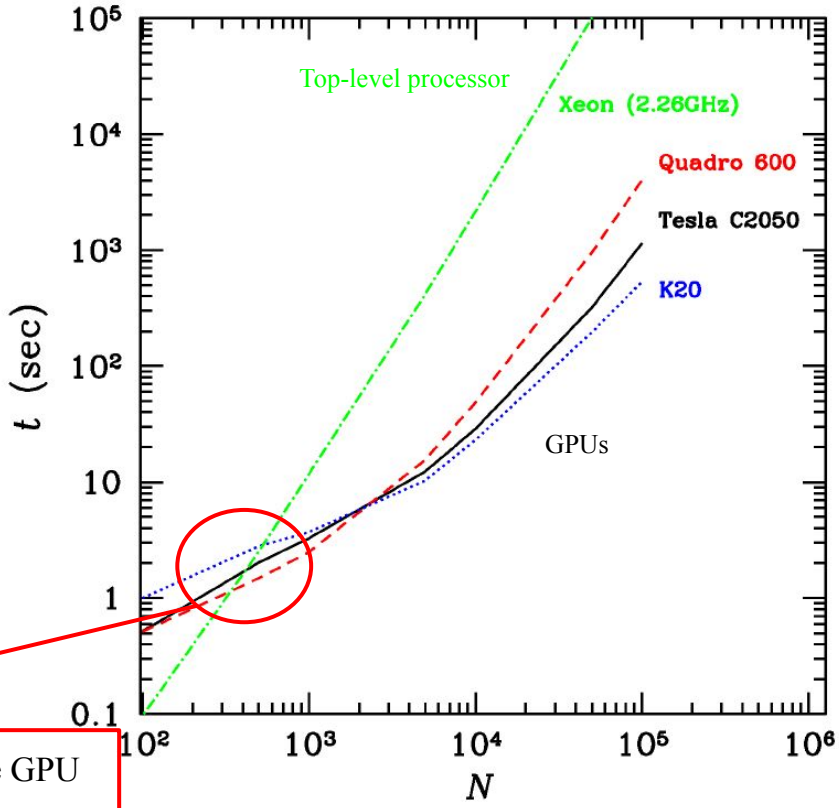
scientist soon realised they're perfect number crunchers!

Funded by the European Union

Dynamics of realistic N-body simulations: software

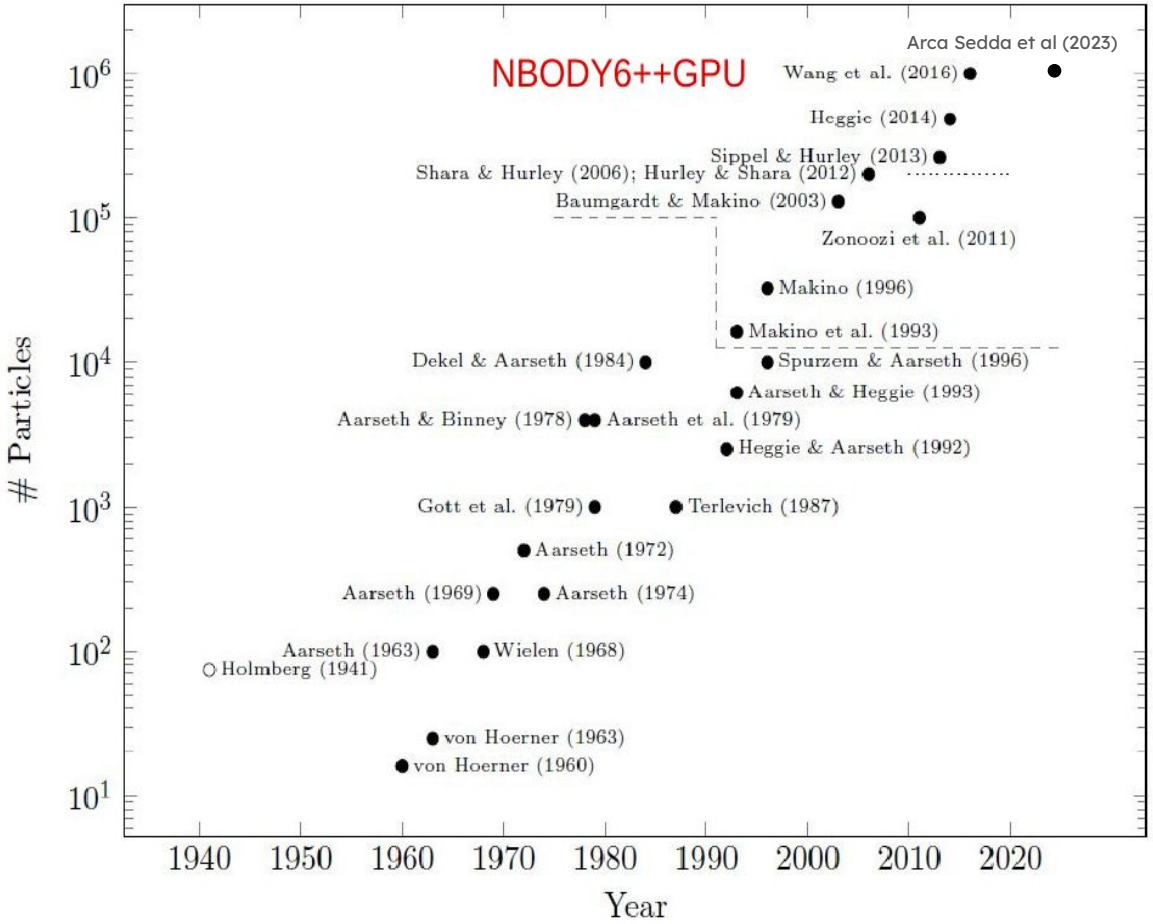
Scalability of the code or efficiency





Communications kill performances at too low N: the GPU needs to be efficiently loaded

Solving the Gravitational *N*-body problem.



GRAPE/
GPU Clusters

GRAPE

Vector Computers

Funded by the European Union

Solving the Gravitational *N*-body problem. The NBODY6++GPU code

The N-body challenge: *when will we see the first star-by-star N-body model of a globular cluster?*

Douglas Heggie

- Honest N-body sim
- Reasonable mass at 12 Gyr (~50,000 Msun)
- Reasonable tide (circular galactic orbit)
- Reasonable IMF (e.g. Kroupa)
- Reasonable binary fraction (~ %)
- Initial model (e.g. Plummer)
- A submitted paper (astro-ph is enough)

Prize: a bottle of single malt Scotch whisky worth €50

Winner (2016): Long Wang - Dragon project (also IAU PhD prize)

Solving the Gravitational *N*-body problem. The NBODY6++GPU code

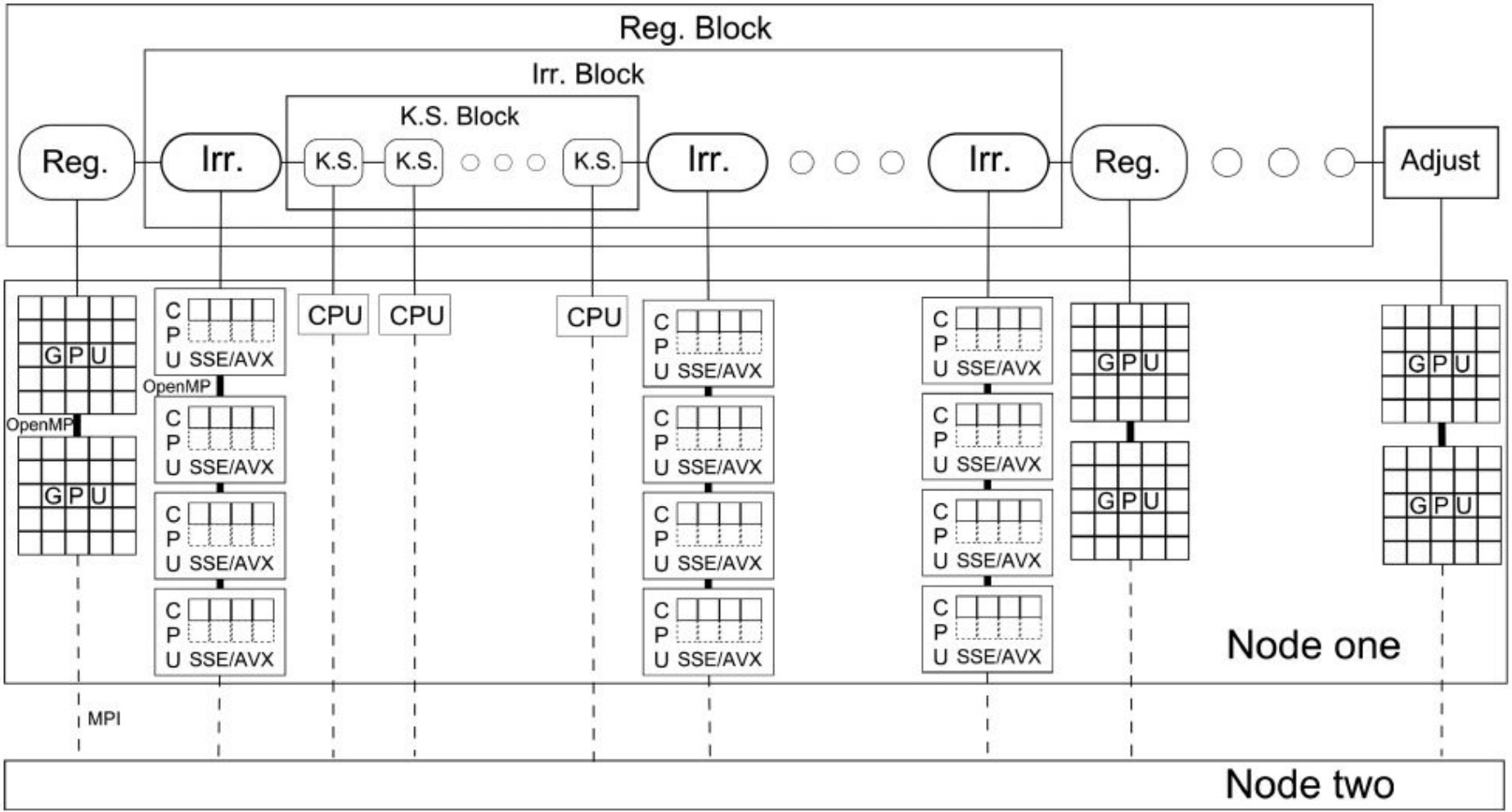
Nbody6++GPU:

Nbody scheme (from NBODY6, developed by S. Aarseth)

MPI implementation (from NBODY6++, developed by R. Spurzem)

GPU implementation (developed by L. Wang)

Written in Fortran90 + CUDA


Reg. Block
Irr. Block
K.S. Block
Reg.
Irr.
K.S.
K.S.
K.S.
Irr.
Irr.
Reg.
Adjust
GPU
CPU
CPU
CPU
C P U SSE/AVX
OpenMP
OpenMP
MPI
Node one
Node two

The NBODY6++GPU
Basic code structure

Basic Code Structure

Input	Read input parameters
Initial conditions	Generate m, $\mathbf{r}$, $\dot{\mathbf{r}}$
Initialization	$\mathbf{F}$, $\dot{\mathbf{F}}$ & Δt
Scheduling	Block-step distribution
Prediction	All N particles
Force calculation	Forces and derivatives
Particle integration	Sequential $\mathbf{F}$&$\dot{\mathbf{F}}$
Corrector	Fourth order
New time-steps	Relative criterion
New block-steps	Determine next group
Results	Cluster parameters

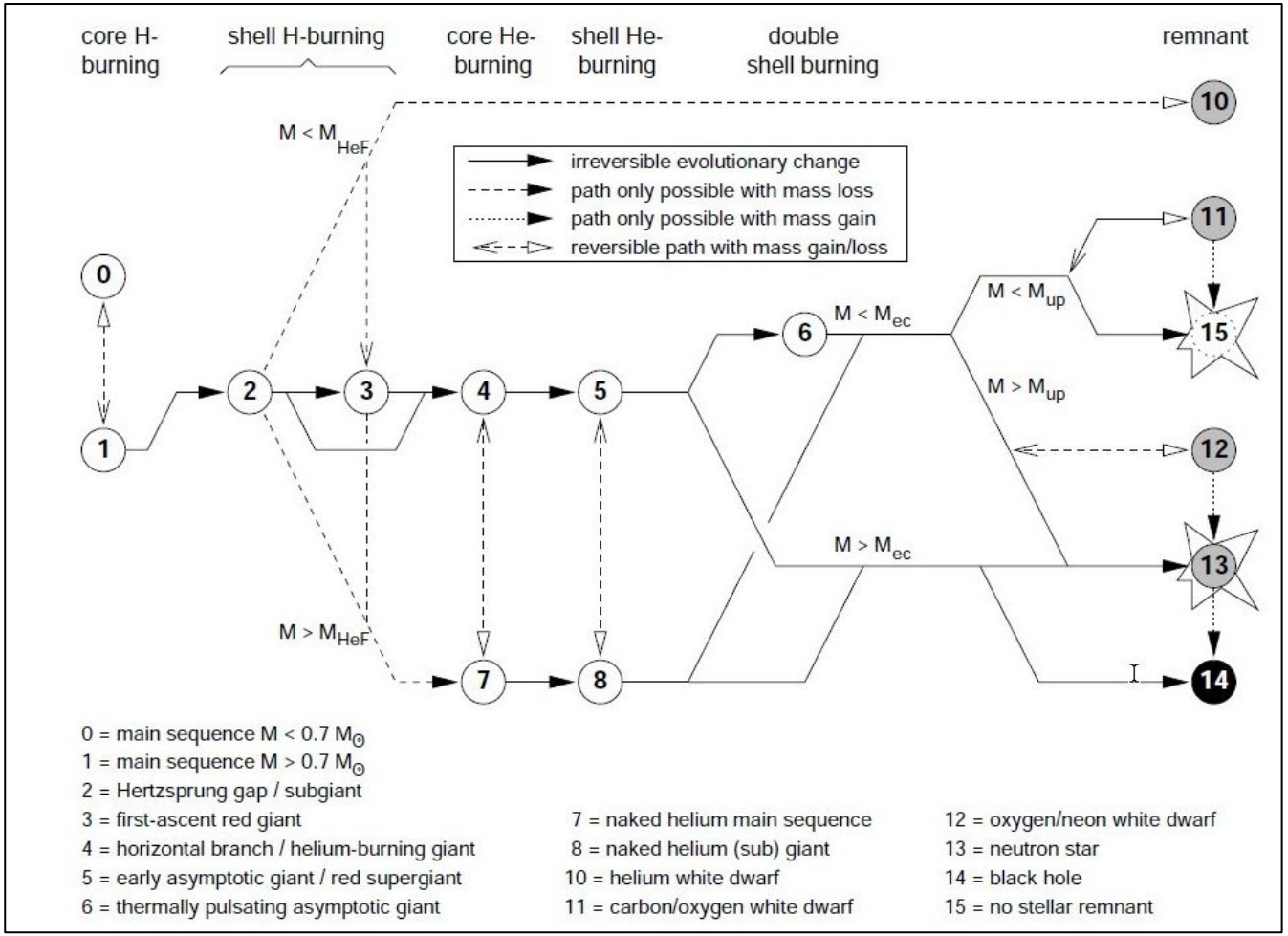
Funded by the European Union

The NBODY6++GPU Stellar Evolution: Single stars

Stellar evolution
is performed via the
SSE/BSE tools
(Hurley et al 2000, 2002)

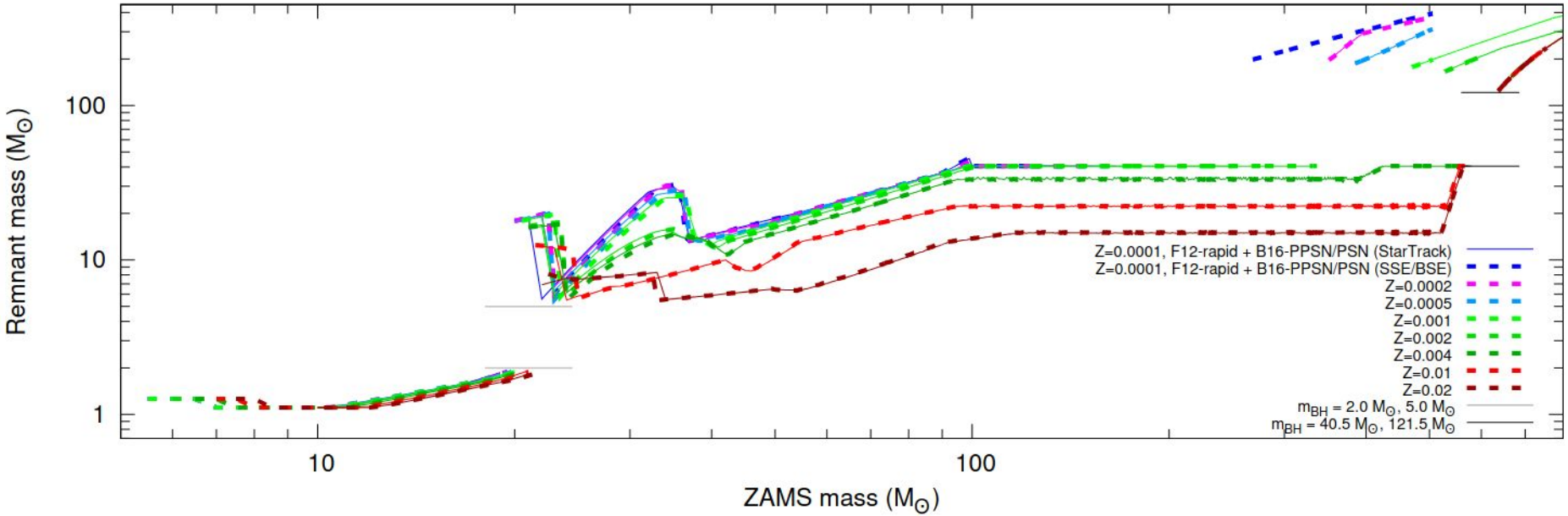
Exploits fitting formulae
to describe stellar evolution

Several upgrades over time:
- Di Carlo et al 2020
- Banerjee et al 2021
- Rizzuto et al 2022
- Kamlah et al 2022
- Arca Sedda et al 2023



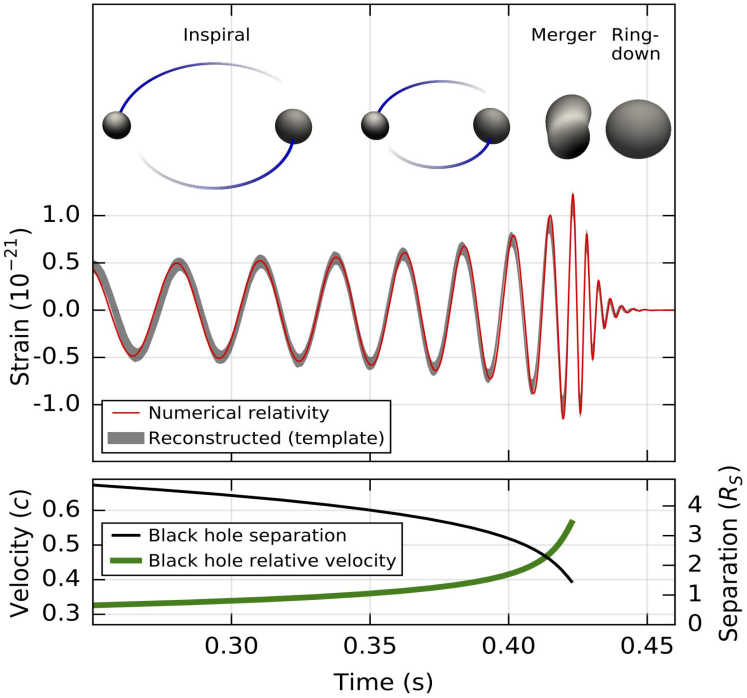
Funded by the European Union

The NBODY6++GPU
Stellar Evolution of massive stars (see Banerjee et al 2020, Kamlah et al 2022)



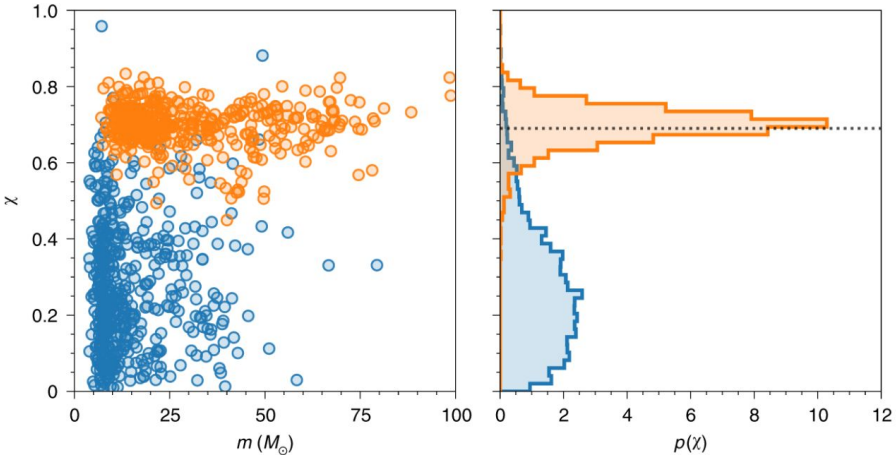
Funded by the European Union

The NBODY6++GPU
Relativistic effects (see Arca Sedda et al 2023)



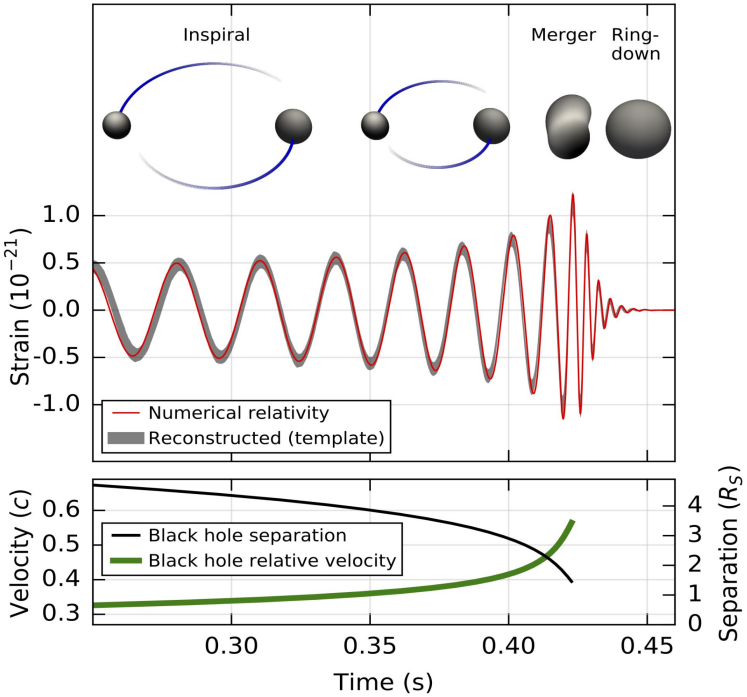
- **Mass-loss:** m1 + m2 != (m1+m2). Around 5% of mass is converted into energy

- **Spins:** what are natal BH spins? We don't actually now, but in general the merger product has a spin of ~ 0.6

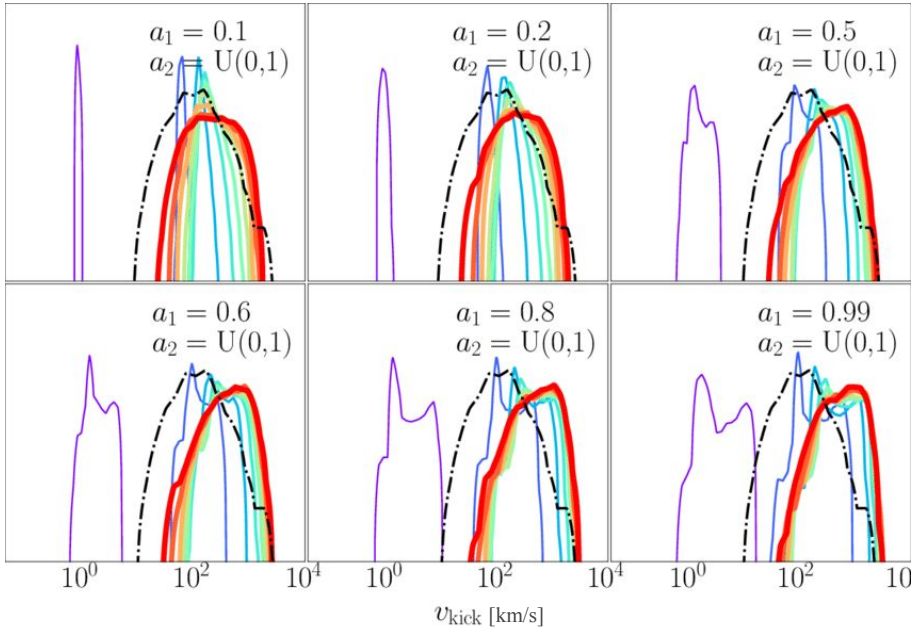


Funded by the European Union

The NBODY6++GPU
Relativistic effects (see Arca Sedda et al 2023)



- **Kicks:** asymmetries in the GW emission impart on the merger product linear momentum, a GW kick



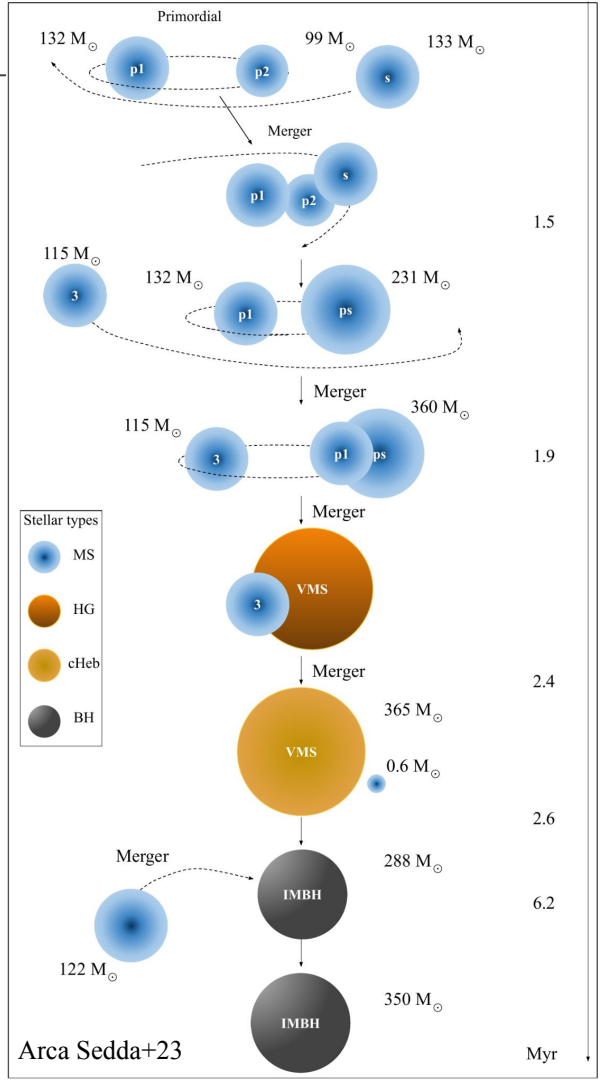
Funded by
the European Union

The NBODY6++GPU
Stellar mergers

Star-BH mergers: BH accretes a fraction f_c = [0-1] of the star mass

In NBODY6++GPU default: f_c = 0.5 **(routine input.F)**

Star-Star merger: depends on stellar types



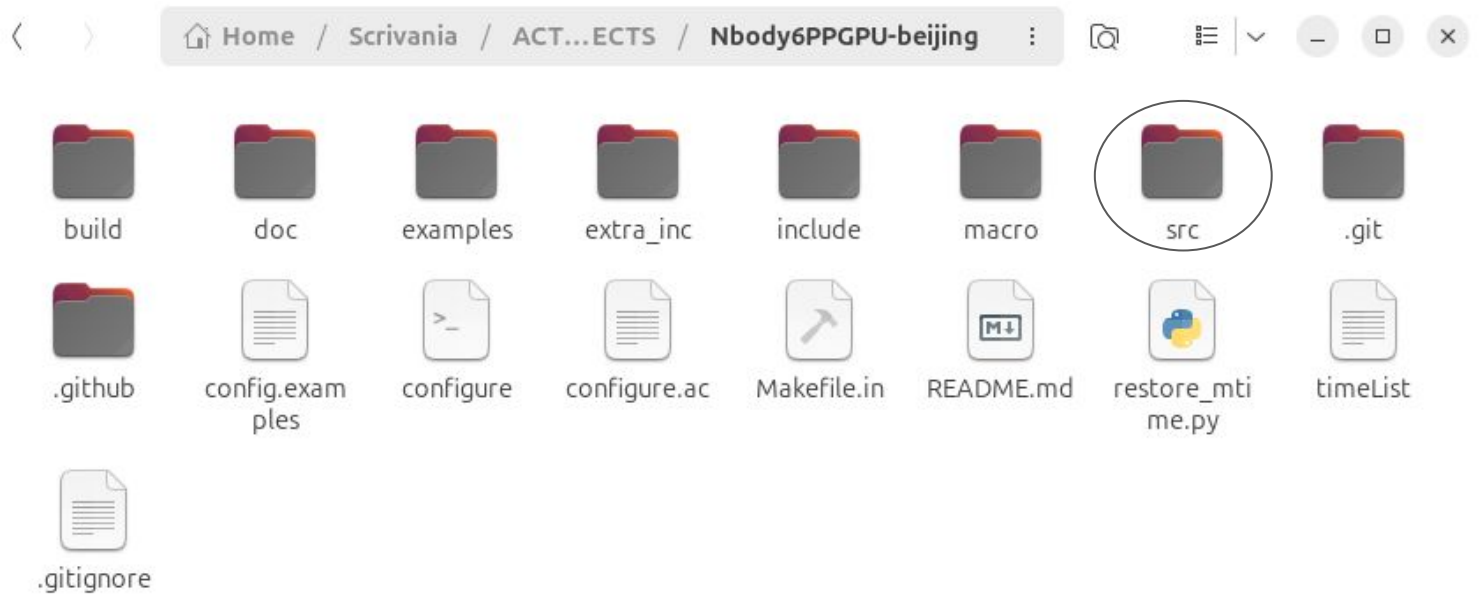
Funded by the European Union

The NBODY6++GPU: let's get started

```
git clone -b dev https://github.com/nbody6ppgpu/Nbody6PPGPU-beijing
```

cd Nbody6PPGPU-beijing

ls



Funded by the European Union

The NBODY6++GPU: let's get started

cd src ; ls ; cd Main ; ls → 371 routines!!!!

```
manuel@AstroBook:~/Scrivania/ACTIVE_PROJECTS/Nbody6PPGPU-beijing/src/Main$ ls
absorb.f          cmfirr_cor.f            eigenvalue.f      fpoly1_mpi.f            hut2.f              kspert.f          nbody6.F          reg.sse.cpp       stablc.f          ttcal.f
add_tlist.f       cmfirr.f                ellan.f           fpoly2.f                hut.f               ksphys.f          nbpot.f           reinit.f          stablz.f          ttforce.f
adjust.F          cmfirr_ucor.f           encorr_mdot.F     fpoly2_ks.f             iblock.f            kspinit.f         nbrem.f           remove.f          star.f            ttgalaxy.f
asa147.f          cmfreg.f                endreg.f          fpoly2_mpi.f            ichain.f            kspoly.f          nbrest.f          remove_ks.F       start3.f          ttinit.f
assess.f          coal.f                  energy.F          fpowercut.f             imbhinit.F          kspred.f          newreg.f          remove_ks.F.diag  start4.f          units.f
bhplot.f          comenv.f                energy_mpi.F      gcinit.f                imf2.f              kspreg.F          newsys.f          remove_ks.F.old   start.F           unpert.f
bindat.f          const.f                 erel3.f           gcint.F                 imfbd.f             ksrect.f          newtev.f          remove_tlist.f    status.f          update.F
binev.f           core.f                  erel4.f           ghost.f                 imf.f               ksrect.f.diag     next_tlist.f      rename_ks.F       stepk.f           util_gpu.F
binout.f          corerd.f                erel.f            giant2.f                impact.f            ksrect.f.old      nstab.f           renew.f           steps2.f          vector.f
binpop.F          counter_reset.f         escape.F          giant3.f                inclin.f            ksreg.F           ntint.f           repair_tlist.f    steps.f           verify.f
block.f           cputim.F                events.f          giant.f                 indexx.f            ksreg.F.diag      offset.f          replace_tlist.f   string_left.f     xbpredall.F
bodies.f          cstab2.f                exchange_tlist.f  global_output.F         induce.f            ksreg.F.old       orbit.f           reset2.f          stumpf.f          xbpredall.F.new
brake2.f          cstab3.f                expand.f          global_params_gether.f  input.F             ksres2.f          output.F          reset.f           subint.F          xbpredall.f.old
brake3.f          cstab4.f                expel2.f          gntage.f                insert.F            ksres.f           peri.f            resolv.f          subsys.f          xcpred.f
brake.f           cstab5.f                expel.f           gpunb.gpu.cu            instar.F            ksres_op.f        permit.f          rkint.f           swcond.f          xtf.f
bsetid.f          custom_output.F         extend.f          gpunb.velocity.cu       intgrt.F            k_step.f          pfac.f            rl.f              switch.f          xtpert.f
cfuncs.f          custom_output_facility.F  fbulge.f        gpunb.velocity.cu.intel  invert.f            ksterm.F          phicor.f          rmesc_tlist.f     synch.f           xtrnl0.F
chain.f           data.F                  fchain.f          gpunb.velocity.cu.multi  irr.avx.cpp         kstide.f          physks.f          roche.f           tail0.f           xtrnld.f
chaos0.f          decide.f                fclose.f          gpunb.velocity.cu.openmp  irr.sse.cpp         kstran.f          pmwpot.f          rpmax2.f          tchain.f          xtrnlf.F
chaos2.f          deform.f                fcloud.f          gpunb.velocity.cu.single  isnan.f             lagr.f            pot.avx.cpp       rpmax.f           tcirc.f           xtrnlp.f
chaos.f           degen.f                 fcorr.F           gpupot.gpu.cu           jacobi.F            levels.f          poti.f            rpmin.f           tides2.f          xtrnlt.F
chdata.f          delay_add_tlist.f       fdisk.f           grrad.f                 jacobi_transform.f  magbrk.f          pot.sse.cpp       rsort.f           tides3.f          xtrnlv.F
check.f           delay.f                 fhalo.f           hcorr.f                 jpred.f             matrix.f          proto_star.f      scale.F           tides.f           xvpred.f
checkl.F          delay_remove_tlist.f    ficorr.F          hiarch.f                jpred_int.f         mdot.F            qforce.f          search.f          touch.f           ycopy.f
chfind.f          delay_store_tlist.f     file_init.F       hicirc.f                kcpert_cor.f        mdot.F.diag       qpmod3.f          select.f          tperi.f           ysave.f
chfirr.f          derqp3.f                findj.f           hidat.f                 kcpert.f            mdot.F.standard   qpmod4.f          setsys.f          tpert.f           zare.f
chinit.f          derqp4.f                findm.f           higrow.f                kick2.f             merge2.f          qpmod.f           setup.F           trans3.f          zcnsts.f
chlist.f          derqp.f                 flush.f           himax.f                 kick.F              merge.f           qtides.f          shift_tlist.f     trans4.f          zero.f
chmod_chain.f     dgcore.f                flyby.f           himod.f                 kickGW.f            mix.f             quad.f            short.f           transk.f          zfuncs.f
chpot.f           difsy1.f                fmiyamoto.f       hipop.F                 ksapo.f             mloss.f           r2sort.f          shrink.f          transq.f
chrect.f          difsy3.f                fmwpot.f          hirect.f                kscorr.f            mlwind.f          ran2.f            shrink_tlist.f    transx.f
chstab.f          difsy4.f                fnfw.f            histab.f                ksin2.f             modify.F          rchain.f          slow.f            trdot2.f
chterm.f          dtchck.f                fnuc.f            hivel.f                 ksinit.F            mrenv.f           recoil.f          sort1.f           trdot.f
clint.f           eccmod.f                fpcorr.F          hmdot2.f                ksint.f             mwpotinit.F       redraw.f          sort_tlist.f      trflow.f
cloud0.f          ecirc.f                 fpert.f           hmdot.f                 kslist.f            mydump.F          reduce.f          spiral.f          triple.f
cloud.f           edot.f                  fpoly0.F          hpsort.f                ksmod.f             nbint_cor.f       reg.avx.cpp       stability.f       tstab.f
cmbody.f          efac2.f                 fpoly1.f          hrdiag.f                ksparmpi.F          nbint.F           regcor_gpu.F      stabl3.f          tstep2.f
cmcorr.f          efac3.f                 fpoly1_ks.f       hrplot.F                ksperi.f            nblist.f          regint.f          stabl4.f          tstep.f
```

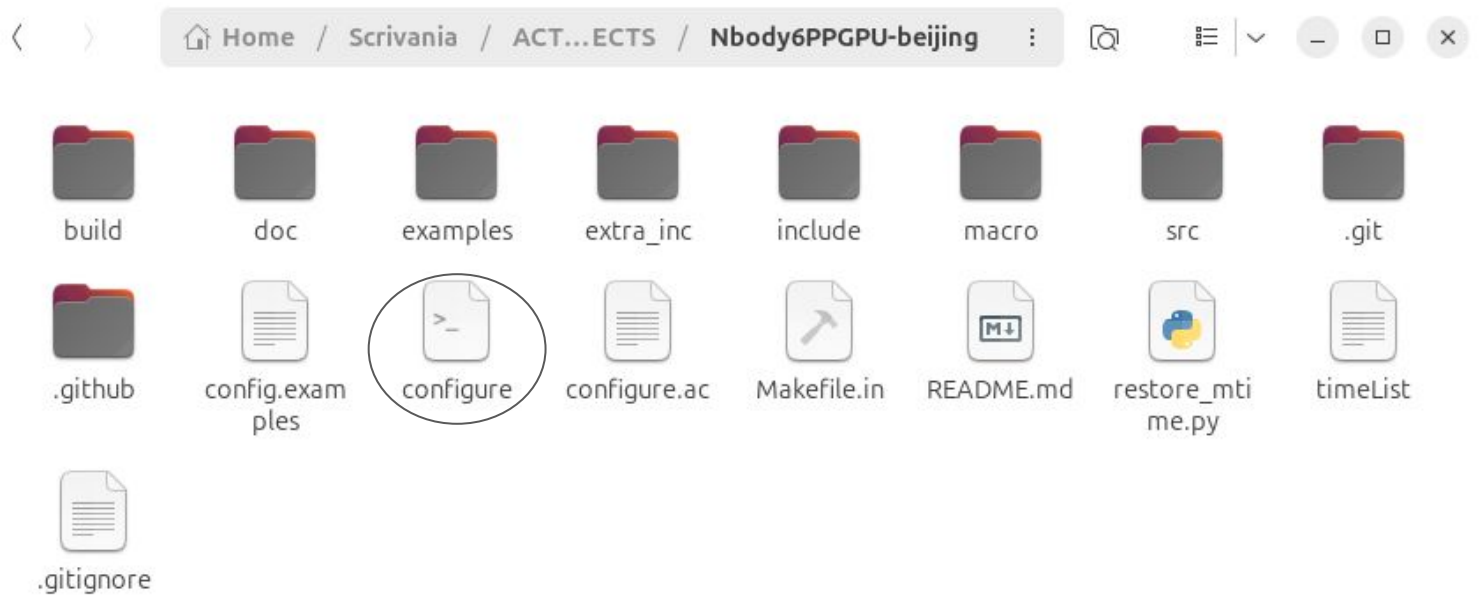
Funded by the European Union

The NBODY6++GPU: let's get started

```
git clone -b dev https://github.com/nbody6ppgpu/Nbody6PPGPU-beijing
```

cd Nbody6PPGPU-beijing

ls



Funded by the European Union

The NBODY6++GPU: let's get started

head -20 config.log

$ ./configure --disable-mpi --enable-simd=sse --with-par=32k --enable-openmp --disable-gpu

(keep finger crossed #1 …)

$ make

(keep finger crossed #2 …)

an executable will be saved in the **build/nbody6++.sse** directory, let's check

the manual of the code can be reached here https://www.overleaf.com/read/hcmxcyffjkzq#89d2bb

The NBODY6++GPU: let's get started: the infamous file for initial conditions (now based on namelists)

The NBODY6+ namelists)

```
&INNBODY6
KSTART=2,TCOMP=1.0E8,TCRTP0=10,isernb=40,iserreg=40,iserks=0 /

&ININPUT
N=6000,NFIX=1,NCRIT=10,NRAND=4332,NNBOPT=80,NRUN=1,NCOMM=10,
ETAI=0.02,ETAR=0.02,RS0=0.15,DTADJ=1.0,DELTAT=1.0,TCRIT=200.0,QE=1.0,RBAR=0.5,ZMBAR=42.,
KZ(1:10)= 1 1 1 0 1 0 3 1 3 2
KZ(11:20)=0 1 0 2 2 0 1 0 3 6
KZ(21:30)=1 1 2 0 2 2 3 2 0 2
KZ(31:40)=1 0 2 2 1 0 1 1 2 1
KZ(41:50)=0 0 0 0 0 4 0 0 3 0 ,
DTMIN=2.E-6,RMIN=5.E-5,ETAU=0.1,ECLOSE=1.0,GMIN=1.0E-06,GMAX=0.01,SMAX=1.0,
Level='C' /

&INSSE /

&INBSE /

&INCOLL /

&INDATA
ALPHAS=1.0,BODY1=150.0,BODYN=20.0,NBIN0=0,NHI0=0,ZMET=0.0001,EPOCH0=0,DTPLOT=1.0 /

&INSETUP SEMI=,ECC=,APO=,N2=,SCALE=,ZM1=,ZM2,ZMH,RCUT= /

&INSCALE
Q=0.5,VXROT=0.0,VZROT=0.0,RTIDE=0.0 /

&INXTRNL0
GMG=1.78E11,RG0=13.3,DISK=,A=,B=,VCIRC=,RCIRC=,GMB=,AR=,GAM=,RG=,,,VG=,,,MP=,AP2=,MPDOT=,TDELAY= /

&INBINPOP
SEMI0=0.0005,ECC0=-1.0,RATIO=1.0,RANGE=5.0,NSKIP=5,IDORM=0 /

&INHIPOP
SEMI0=,ECC0=,RATIO=,RANGE= /
```

Funded by the European Union

The NBODY6++GPU: let's get started: the infamous file for initial conditions #1

```
1 1.2E+7 1.2E+10 40 40 0
KSTART, TCOMP, TCRITp, isernb,iserreg                    (nbody6.F)
```

`nbody6.F`:

KSTART	Run control index =1: new run (construct new model or read from `dat.10`) =2: restart/continuation of a run, needs `fort.1` =3: restart + changes of DTADJ, DELTAT, TADJ, TNEXT, TCRIT, QE, J, KZ(J) =4: restart + changes of ETAI, ETAR, ETAU, DTMIN, RMIN, NCRIT, NNBOPT, SMAX =5: restart containing the combination of the control index 3 and 4
TCOMP	Maximum wall-clock time in seconds (parallel runs: wall clock)
TCRITP	Termination time in Myr
isernb	For MPI parallel runs: only irregular block sizes larger than this value are executed in parallel mode (dummy variable for single CPU)
iserreg	For MPI parallel runs: only regular block sizes larger than this value are executed in parallel mode (dummy variable for single CPU)
iserks	For MPI parallel runs: only ks block sizes larger than this value are executed in parallel mode (dummy variable for single CPU)

The NBODY6++GPU: let's get started: the infamous file for initial conditions #1

```
1000 1 10 100 80 1 10
N, NFIX, NCRIT, NRAND, NNBOPT, NRUN                    (input.F)
```

N	Total number of particles (single + c.m.s. of binaries; singles + 3×c.m.s. of binaries < NMAX−2)
NFIX	Multiplicator for output interval of data on `conf.3` and of data for binary stars (output each DELTAT×NFIX time steps; compare KZ(3) and KZ(6))
NCRIT	Minimum particle number (alternative termination criterion)
NRAND	Random number seed; any positive integer
NNBOPT	Desired optimal neighbour number (< LMAX−5)
NRUN	Run identification index

The NBODY6++GPU: let's get started: the infamous file for initial conditions #1

```
0.02 0.02 0.08 1.0 1.0 10000.0 1.0E-01 0.6 0.591865
ETAI, ETAR, RS0, DTADJ, DELTAT, TCRIT, QE, RBAR, ZMBAR        (input.F)
```

ETAI	Time–step factor for irregular force polynomial
ETAR	Time–step factor for regular force polynomial
RS0	Initial guess for all radii of neighbour spheres (*N*–body units)
DTADJ	Time interval for parameter adjustment and energy check (*N*–body units)
DELTAT	Time interval for writing output data and diagnostics, multiplied by NFIX (*N*–body units)
TCRIT	Termination time (*N*–body units)
QE	Energy tolerance: – immediate termination if DE/E > 5*QE & KZ(2) $\leq$ 1; – restart if DE/E > 5*QE & KZ(2) > 1 and termination after second restart attempt.
RBAR	Scaling unit in pc for distance (*N*–body units)
ZMBAR	Scaling unit for average particle mass in solar masses (in scale-free simulations RBAR and ZMBAR can be set to zero; depends on KZ(20))

The NBODY6++GPU: let's get started: the infamous file for initial conditions #1

```
0.02 0.02 0.08 1.0 1.0 10000.0 1.0E-01 0.6 0.591865
ETAI, ETAR, RS0, DTADJ, DELTAT, TCRIT, QE, RBAR, ZMBAR        (input.F)
```

ETAI	Time–step factor for irregular force polynomial
ETAR	Time–step factor for regular force polynomial
RS0	Initial guess for all radii of neighbour spheres (*N*–body units)
DTADJ	Time interval for parameter adjustment and energy check (*N*–body units)
DELTAT	Time interval for writing output data and diagnostics, multiplied by NFIX (*N*–body units)
TCRIT	Termination time (*N*–body units)
QE	Energy tolerance: – immediate termination if DE/E > 5*QE & KZ(2) $\leq$ 1; – restart if DE/E > 5*QE & KZ(2) > 1 and termination after second restart attempt.
RBAR	Scaling unit in pc for distance (*N*–body units)
ZMBAR	Scaling unit for average particle mass in solar masses (in scale-free simulations RBAR and ZMBAR can be set to zero; depends on KZ(20))

These come from McLuster, and represents the scales for masses and lengths, thus it is crucial double-check them before starting the run

KZ(1) Save COMMON to file `comm.1`
= 1: at end of run or when dummy file STOP is created
= 2: every 100*NMAX steps

KZ(2) Save COMMON to file `comm.2` and `comm.2_[TIME[NB]]`
= 1: save at output time and every time interval of NCOMM (NB time unit). Notice after restart, the adjust/output time (TADJ) should be integer times of NCOMM, otherwise the COMMON data would not be dumped.
= 2: save at output time and every time interval of NCOMM and restart simulation if energy error DE/E > 5*QE

KZ(3) Save basic data to file `conf.3` at output time (unformatted)

KZ(4) (Suppressed) Binary diagnostics on `bdat.4` (# = threshold levels <10)

KZ(5) Initial conditions of the particle distribution, needs KZ(22)=0
= 0: uniform & isotropic sphere
= 1: Plummer random generation
= 2: two Plummer models in orbit (extra input)
= 3: massive perturber and planetesimal disk (each pariticle has circular orbit, constant separation along radial direction between each neighbor and random phase) (extra input)
= 4: massive initial binary (extra input)
= 5: Jaffe model (extra input)
≥ 6: Zhao BH cusp model (extra input if KZ(24)<0)

KZ(6) Output of significant and regularized binaries at main output (`bodies.f`)
= 1: output regularized and significant binaries (|E|>0.1 ECLOSE)
= 2: output regularized binaries only
= 3: output significant binaries at output time and regularized binaries with time interval DELTAT
= 4: output of regularized binaries only at output time

Funded by the European Union

KZ(7) Determine Lagrangian radii and average mass, particle counters, average velocity, velocity dispersion, rotational velocity within Lagrangian radii (`lagr.f`)
= 1: Get actual value of half mass radius RSCALE by using current total mass
≥ 2: Output data at main output and `lagr.7`
≥ 6: Output Lagrangian radii for two mass groups at `lagr.31` and `lagr.32` (`lagr2.f`; based on KZ(5)=1,2; cost is O(N^2))
—- methods:
= 2, 4: Lagrangian radii calculated by initial total mass
= 3, ≥ 5: Lagrangian radii calculated by current total mass (The single/K.S-binary Lagrangian radii are still calculated by initial single/binary total mass)
= 2, 3: All parameters are averaged within the shell between two Lagrangian radii neighbors
≥ 4: All parameters are averaged from center to each Lagrangian radius

KZ(8) Primordial binaries initialization and output (`binpop.f`)
—- Initialization:
= 0: No primordial binaries
= 1, ≥ 3: generate primordial binaries based on KZ(41) and KZ(42) (`binpop.F`)
= 2: Input primordial binaries from first 2×NBIN0 lines of `dat.10`
—- Output:
> 0: Save information of primordial binary that change member in `pbin.18`; binary diagnostics at main output (`binout.f`)
≥ 2: Output KS binary in `bdat.9`, soft binary in `bwdat.19` at output time

KZ(9) Binary diagnostics
= 1, 3: Output diagnostics for the hardest binary below ECLOSE in `hbin.39` (`adjust.f`)
≥ 2: Output binary evolution stages in `binev.17` (`binev.f`)
≥ 3: Output binary with degenerate stars in `degen.4` (`degen.f`)

KZ(10) K.S. regularization diagnostics at main output
> 0: Output new K.S. information
> 1: Output end K.S. information
≥ 3: Output each integrating step information

Funded by the European Union

KZ(11) (Suppressed)

KZ(12) > 0: HR diagnostics of evolving stars with output time interval DTPLOT in `sse.83` (single star) and `bse.82` (K.S. binary)
$= -1$: used if KZ(19)$= 0$ (see details in KZ(19) description)

KZ(13) Interstellar clouds
$= 1$: constant velocity for new cloud
> 2: Gaussian velocity for new cloud

KZ(14) External tidal force
$= 1$: standard solar neighbor tidal field
$= 2$: point-mass galaxy with circular orbit (extra input)
$= 3$: point-mass + disk + halo + Plummer (extra input)
$= 4$: Plummer model (extra input)
$= 5$: Milky-Way potential (extra input) based on Python Galpy package (Bovy, 2015)

KZ(15) Triple, quad, chain and merger search
≥ 1: Switch on triple, quad, chain (KZ(30)>0) and merger search (`impact.f`)
≥ 2: Diagnostics at main output at begin and end of triple, quad
≥ 3: Save first five outer orbits every half period of wide quadruple before merger and stable quadruples accepted for merger in `quastab.89`

KZ(16) Auto-adjustment of regularization parameters
≥ 1: Adjust RMIN, DTMIN & ECLOSE every DTADJ time
≥ 3: modify RMIN for GPERT > 0.05 or < 0.002 in chain; output diagnostics at `kscrit.77`

KZ(17) Auto-adjustment of ETAI, ETAR and ETAU by tolerance QE every DTADJ time (`check.f`)
≥ 1: Adjust ETAI, ETAR
≥ 2: Adjust ETAU

Funded by the European Union

KZ(18) Hierarchical systems
= 1, 3: diagnostics (`hiarch.f`)
≥ 2: Initialize primordial stable triples, number is NHI0 (`hipop.F`)
≥ 4: Data bank of stable triple, quad in `hidat.87` (`hidat.f`)

KZ(19) Stellar evolution mass loss
= 0: if KZ(12)= −1, the output data will keep the input data unit if KZ(22)= 2 − 4 or *N*-body units if KZ(22)= 6 − 10
= 1, 2: supernova scheme
≥ 3: Eggleton, Tout & Hurley
≥ 5: extra diagnostics (mdot.F)
= 2, 4: Input stellar parameters from `fort.21` (`instar.f`)
N lines of (MI, KW, M0, EPOCH1, OSPIN)
MI: Current mass
KW: Kstar type
M0: Initial mass
EPOCH1: evolved age of star (Age = TIME[Myr] − EPOCH1)
OSPIN: angular velocity of star

KZ(20) Initial mass functions, need KZ(22)=0 or 9:
= 0: self-defined power-law mass function using ALPHAS (`data.F`)
= 1: Miller-Scalo-(1979) IMF (`imf.f`)
= 2, 4: KTG (1993) IMF (`imf2.f`)
= 3, 5: Eggleton-IMF (`imf2.f`)
= 6, 7: Kroupa(2001) (`imf2.f`), extended to Brown Dwarf regime (`imfbd.f`)
—- Primordial binary mass
= 2, 6: random pairing (`imf2.f`)
= 3, 4, 5, 7: binary mass ratio corrected by $(m_1/m_2)\prime = (m_1/m_2)^{0.4}$ + constant (Eggleton, `imf2.f`)
= 8: binary mass ratio $q = m_1/m_2$ $(m_2 \leq m_1)$ use distribtution $0.6q^{-0.4}$ (Kouwenhoven)

Funded by the European Union

KZ(21) Extra diagnostics information at main output every DELTAT interval (`output.F`)
≥ 1: output NRUN, MODEL, TCOMP, TRC, DMIN, AMIN, RMAX, RSMIN, NEFF
≥ 2: Number of escapers NESC at main output will be counted by Jacobi escape criterion (cost is $O(N^2)$, `jacobi.f`)

KZ(22) Initialization of basic particle data mass, position and velocity (`data.F`)
—- Initialization with internal method
$= 0, 1$: Initial position, velocity based on KZ(5), initial mass based on KZ(20)
$= 1$: write initial conditions in `dat.10` (`scale.F`)
—- Initialization by reading data from `dat.10`
$= 2$: input through NBODY-format (7 parameters each line: mass, position(1:3), velocity(1:3))
$= 3$: input through Tree-format (`data.F`)
$= 4$: input through Starlab-format
$= 6$: input through NBODY-format and do scaling
$= 7$: input through Tree-Format and do scaling
$= 8$: input through Starlab-format and do Scaling
$= 9$: input through NBODY-format but ignore mass (first column) and use IMF based on KZ(20), then do scaling
$= 10$: input through NBODY-format and all units are astrophysical units (mass: $M_\odot$; position: pc; velocity: km/s)

KZ(23) Removal of escapers (`escape.F`)
≥ 1: remove escapers and ghost particles generated by two star coalescence (collision)
$= 2, 4$: write escaper diagnostics in `esc 11`
≥ 3: initialization & integration of tidal tail

Funded by the European Union

KZ(24)	Initial conditions for subsystems < 0: ZMH & RCUT (N-body units) Zhao model (Need KZ(5)$\geq$6, `setup.F`) $= 1$: Add one massive black hole (extra input: mass, position, velocity and output frequency), will output black hole data in `mbh.45` and its neighbor data in `mbhnb.46`
KZ(25)	Velocity kicks for white dwarfs (`kick.F`) $= 1$: Type 10 Helium white dwarf & 11 Carbon-Oxygen white dwarf $= 2$: All WDs (type 10, 11 and type 12 Oxygen-Neon white dwarf)
KZ(26)	Slow-down of two-body motion, increase the regularization integration efficiency ≥ 1: Apply to KS binary ≥ 2: Apply to chain $= 3$: Rectify to get better energy conservation
KZ(27)	Two-body tidal circularization (Mardling & Aarseth, 2001; Portegies Zwart et al. 1997) (Please suppress in KS parallel version) $= 1$: sequential $= 2$: chaos $= -1$: Only detect collision and suppress coalescence
KZ(28)	Magnetic braking and gravitational radiation for NS or BH binaries (Need KZ(19)=3) ≥ 1: GR coalescence for NS & BH (`brake.f`, `brake3.f`) ≥ 2: Diagnostics at main output (`brake.f`) $= 3$: Input of ZMH = 1/SQRT(2*N) (Need KZ(5)$\geq$6) (`setup.F`) $= 4$: Set every star as type 13 Neutron star (`instar.f`)
KZ(29)	(Suppressed) Boundary reflection for hot system
KZ(30)	Hierarchical system regularization $= -1$: Use chain only $= 0$: No triple, quad and chain regularization, only merger $= 1$: Use triple, quad and chain (`impact.f`) ≥ 2: Diagnostics at begin/end of chain at main output ≥ 3: Diagnostics at each step of chain at main output

Funded by the European Union

KZ(31)	Centre of mass correction after energy check (`cmcorr.f`)
KZ(32)	Adjustment (increase) of adjust interval DTADJ, output interval DELTAT and energy error criterion QE based on binding eneryg of cluster (`check.f`)
KZ(33)	Block-step statistics at main output (diagnostics) ≥ 1: Output irregular block step; and K.S. binary step if KZ(8)>0 ≥ 2: Output regular block step
KZ(34)	Roche-lobe overflow $= 1$: Roche & Spin synchronisation on binary with circular orbit (`synch.f`) $= 2$: Roche & Tidal synchronisation on binary with circular orbit by BSE method (`bsetid.f`)
KZ(35)	TIME reset to zero every 100 time units, total time is TTOT = TIME + TOFF (`offset.f`)
KZ(36)	(Suppressed) Step reduction for hierarchical systems
KZ(37)	Neighbour list additions (`checkl.F`) ≥ 1: Add high-velocity particles into neighbor list ≥ 2: Add small time step particle (like close encounter particles near neighbor radius) into neighbor list
KZ(38)	Force polynomial corrections during regular block step calculation $= 0$: no corrections
KZ(39)	Neighbor radius adjustment method $= 0$: The system has unique density centre and smooth density profile $= 1, \geq 3$: The system has no unique density centre or smooth density profile skip velocity modification of RS(I) (`regint.f`, `regcor_gpu.f`) do not reduce neighbor radius if particle is outside half mass radius reduce RS(I) by multiply 0.9 instead of estimation of RS(I) based on NNBOPT/NNB when neighbor list overflow happens (`fpoly0.F`, `util_gpu.F`) $= 2, 3$: Consider sqrt(particle mass / average mass) as the factor to determine the particle's neighbor membership. (`fpoly0.F`, `util_gpu.F`)
KZ(40)	$= 0$: For the initialization of particle time steps, use only force and its first derivative to estimate. This is very efficent. > 0: Use Fploy2 (second and third order force derivatives calculation) to estimate the initial time steps. This method provide more accurate time steps and avoid incorrent time steps for some special cases like initially cold systems, but the computing cost is much higher ($O(N^2)$)

KZ(41) proto-star evolution of eccentricity and period for primordial binaries initialization (`proto_star_evol`, `binpop.F`)

KZ(42) Initial binary distribution
= 0: RANGE>0: uniform distribution in log(semi) between SEMI0 and SEMI0/RANGE
RANGE<0: uniform distribution in semi between SEMI0 and -1*RANGE.
= 1: linearly increasing distribution function $f = 0.03438 * logP$
= 2: $f = 3.5logP/[100 + (logP) * *2]$
= 3: $f = 2.3(logP - 1)/[45 + (logP - 1) * *2]$; This is a "3rd" iteration when pre-ms evolution is taken into account with KZ(41)=1
= 4: $f = 2.5(logP - 1)/[45 + (logP - 1) * *2]$; This is a "34th" iteration when pre-ms evolution is taken into account with KZ(41)=1 and RBAR<1.5
= 5: Duquennoy & Mayor 1991, Gaussian distribution with mean $\log P = 4.8$, SDEV in $\log P = 2.3$. Use Num.Recipes routine `gasdev.f` to obtain random deviates given "idum1"

KZ(43) (Unused)

KZ(44) (Unused)

KZ(45) (Unused)

KZ(46) HDF5/BINARY/ANSI format output and global parameter output (main output, see chapter 13 for details)
= 1,3: HDF5(if HDF5 is compiled)/BINARY format
= 2,4: ANSI format
= 1,2: Only output active stars with time interval defined by KZ(47)
= 3,4: Output full particle list with time interval defined by KZ(47)

KZ(47) Frequency for KZ(46) output
Output data with time interval $0.5^{KZ(47)} \times SMAX$

KZ(48) (Unused)

KZ(49) Computation of Moments of Inertia (with Chr. Theis) in `fort.60` (`ellan.f`)

KZ(50) For unperted KS binary. The neighbor list is searched for finding next KS step. It is safer to get correct step but not efficient when unperted binary number is large. To suppress this, set to 1

The NBODY6++GPU: let's get started: the infamous file for initial conditions #1

```
3.0E-05 3.0E-05 0.1 1.0 1.0E-06 0.01 1.0
DTMIN, RMIN, ETAU, ECLOSE, GMIN, GMAX, SMAX                    (input.F)
```

DTMIN	Time–step criterion for regularization search
RMIN	Distance criterion for regularization search
ETAU	Regularized time-step parameter (6.28/ETAU steps/orbit)
ECLOSE	Binding energy per unit mass for hard binary (positive)
GMIN	Relative two-body perturbation for unperturbed motion
GMAX	Secondary termination parameter for soft KS binaries
SMAX	Maximum time-step (factor of 2 commensurate with 1.0)

Funded by the European Union

The NBODY6++GPU: let's get started: the infamous file for initial conditions #1

```
 2.35 150.0 0.08 0 0 0.0002 0 1.0
ALPHA, BODY1, BODYN, NBIN0, ZMET, EPOCH0, DTPLOT              (data.F)
```

ALPHA	Power-law index for initial mass function, routine `data.F`
BODY1	Maximum particle mass before scaling (based on KZ(20); solar mass unit)
BODYN	Minimum particle mass before scaling
NBIN0	Number of primordial binaries (need KZ(8)>0) – by routine `imf2.F` using a binary IMF (KZ(20)≥2) – by routine `binpop.F` splitting single stars (KZ(8)>0) – by reading subsystems from `dat.10` (KZ(22)≥2)
ZMET	Metal abundance (in range 0.03 - 0.0001)
EPOCH0	Evolutionary epoch (in 10^6 yrs)
DTPLOT	Plotting interval for stellar evolution HRDIAG (N-body units; ≥ DELTAT)

The NBODY6++GPU: let's get started: the infamous file for initial conditions #1

```
0.5 0.0 0.0 0.0
Q, VXROT, VZROT, RSPH2                                    (scale.F)
```

Q	Virial ratio (routine `scale.F`; Q=0.5 for equilibrium)
VXROT	XY–velocity scaling factor (> 0 for solid-body rotation)
VZROT	Z–velocity scaling factor (not used if VXROT = 0)
RTIDE	Unscaled tidal radius for KZ(14)=2 and KZ(22)≥2. If not zero, RBAR = RT/RTIDE where RT[pc] is tidal radius calculated from input GMG and RG0

The NBODY6++GPU: let's get started: the infamous file for initial conditions #1

```
1.78E11 13.3
GMG, RG0                                                    (xtrnl0.F) KZ(14)=2
```

GMG	Point-mass galaxy (solar masses, linearized tidel field in circular orbit)
RG0	Central distance (in kpc)

The NBODY6++GPU: first simulation*

We need the executable, an input file, and sometimes an initial condition file

Create a directory to perform your tests

```
mkdir TEST
mkdir TEST/exec
```

Copy the executable

```
cp build/nbody6++.sse TEST/exec
```

Copy the input file stored in the examples directory on git

```
cp $PATH_TO_DOWNLOAD/examples/N10k_noDat10.inp TEST/exec
```

In this first example we let NBODY create the initial conditions (masses, positions, velocities), so we don't need a "dat.10" file.

```
OMP_NUM_THREADS=1 ./nbody6++.sse< N10k_noDat10.inp 1> run_test.out 2> run_test.err &
```

On some machines, it may be needed to set the following

```
ulimit -s unlimited
```

*See also the guide on our git repo: QuickStart.md

Funded by the European Union

```
&INNBODY6
KSTART=1,TCOMP=1.0E8,TCRTP0=2,isernb=40,iserreg=40,iserks=0 /
&ININPUT
N=1000,NFIX=1,NCRIT=10,NRAND=4332,NNBOPT=80,NRUN=1,NCOMM=10,
ETAI=0.01,ETAR=0.01,RS0=0.15,DTADJ=1.0,DELTAT=1.0,TCRIT=1000.0,QE=1.0,RBAR=0.1,ZMBAR=125.,
KZ(1:10)= 1 1 1 0 1 0 3 1 3 2
KZ(11:20)=0 1 0 2 2 0 1 0 3 6
KZ(21:30)=1 1 2 0 2 2 3 2 0 2
KZ(31:40)=1 0 2 2 1 0 1 1 2 1
KZ(41:50)=0 0 0 0 0 4 0 0 3 0 ,
DTMIN=2.5E-6,RMIN=8.E-5,ETAU=0.1,ECLOSE=1.0,GMIN=1.0E-06,GMAX=0.01,SMAX=1.0,
Level='C' /
&INSSE /
&INBSE /
&INCOLL /
&INDATA
ALPHAS=1.0,BODY1=150.0,BODYN=100.0,NBIN0=0,NHI0=0,ZMET=0.0001,EPOCH0=0,DTPLOT=1.0 /
&INSETUP SEMI=,ECC=,APO=,N2=,SCALE=,ZM1=,ZM2,ZMH,RCUT= /
&INSCALE
Q=0.5,VXROT=0.0,VZROT=0.0,RTIDE=0.0 /
&INXTRNL0
GMG=1.78E11,RG0=13.3,DISK=,A=,B=,VCIRC=,RCIRC=,GMB=,AR=,GAM=,RG=,,,VG=,,,MP=,AP2=,MPDOT=,TDELAY= /
&INBINPOP
SEMI0=0.0005,ECC0=-1.0,RATIO=1.0,RANGE=5.0,NSKIP=5,IDORM=0 /
&INHIPOP
SEMI0=,ECC0=,RATIO=,RANGE= /
```

Funded by the European Union

What is printed in the main output?

```
RANK:             0  OpenMP Number of Threads:

               N  NFIX  NCRIT  NRAND  NNBOPT  NRUN NCOMM

            5000     1     10  27943     100     1    10

             ETAI       ETAR       RS0        DTADJ      DELTAT     TCRITP     TCRIT      QE         RBAR        ZMBAR

             2.0E-02    2.0E-02    6.5E-01    1.0E+00    1.0E+00    5.0E+00    1.0E+05    1.0E-02    5.0E-01    3.5E-01


            OPTIONS

           1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37
38 39 40 41 42 43 44 45 46 47 48 49 50

           0  2  1  0  1  1  5  2  3  2  0  1  0  1  2  0  0  0  3  6  1 10  2  0  1  2  2  0  0  1  1  0  3  2  0  0  0
 2  2  0  0  0  0  0  0  2 -3  0  0  0


            DTMIN      RMIN       ETAU       ECLOSE     GMIN       GMAX       SMAX

            1.0E-03    1.0E-02    1.0E-01    1.0E+00    1.0E-06    1.0E-03    1.2E-01
```

Repetition of some input variable

Funded by the European Union

```
          SCALING:   SX =  1.68378D+00  E = -1.48E-01  M(1) = 6.11E-05  M(N) = 2.48E-04  <M> = 2.00E-04  Q =   0.49919

          TIME SCALES:   TRH = 1.1E+02  TCR = 2.8E+00  2<R>/<V> = 2.8E+00


          PHYSICAL SCALING:    R* =   1.683784063659E+00  M* =   1.771647206537E+03  V* =   2.127418169251E+00  T* =
7.739394786414E-01  <M> =   3.543294413074E-01  SU =   7.468182747310E+07
```

Scaling Factor and info about Physical Scaling

```
          SCALING:   SX =  1.68378D+00  E = -1.48E-01  M(1) = 6.11E-05  M(N) = 2.48E-04  <M> = 2.00E-04  Q =   0.49919

          TIME   M/MT:  1.00D-03  3.00D-03  5.00D-03  1.00D-02  3.00D-02  5.00D-02  1.00D-01  2.00D-01  3.00D-01  4.00D-
01  5.00D-01  6.00D-01  7.00D-01  8.00D-01  9.00D-01  9.50D-01  9.90D-01  1.00D+00       <RC
     0.0000D+00 RLAGR:   5.59D-02  8.84D-02  1.10D-01  1.39D-01  2.06D-01  2.32D-01  3.09D-01  4.23D-01  5.37D-01  6.59D-
01  7.40D-01  8.93D-01  1.12D+00  1.51D+00  2.16D+00  3.13D+00  8.06D+00  8.98D+01  3.35D-01
     0.0000D+00 RSLAGR:   5.59D-02  8.84D-02  1.10D-01  1.39D-01  2.06D-01  2.32D-01  3.09D-01  4.23D-01  5.37D-01  6.59D
-01  7.40D-01  8.93D-01  1.12D+00  1.51D+00  2.16D+00  3.13D+00  8.06D+00  8.98D+01
     0.0000D+00 RBLAGR:   0.00D+00  0.00D+00  0.00D+00  0.00D+00  0.00D+00  0.00D+00  0.00D+00  0.00D+00  0.00D+00  0.00D
+00  0.00D+00  0.00D+00  0.00D+00  0.00D+00  0.00D+00  0.00D+00  0.00D+00  0.00D+00
     0.0000D+00 AVMASS:  1.68D-04  1.41D-04  1.49D-04  1.62D-04  1.65D-04  2.28D-04  2.05D-04  2.10D-04  1.94D-04  1.92D-
04  2.09D-04  2.05D-04  2.00D-04  1.98D-04  2.00D-04  2.00D-04  2.00D-04  2.00D-04  1.94D-04
     0.0000D+00 NPARTC:         6        22        34        62       183       250       489       956      1544       20
83      2388      2927      3495      4038      4492      4743      4952      5000       600
```

Time, Lagrangian Radii info etc

Funded by the European Union

```
             ADJUST:  TIME    0.00000D+00  T[Myr]    0.00  Q 0.50  DE   0.000E+00 DELTA   0.000E+00 DETOT
0.000E+00 E  -2.475E-01 E
KIN   2.492E-01 POT   4.992E-01 ETIDE   2.489E-03 ETOT  -2.475E-01 EBIN   0.000E+00 EMERGE   0.000E+00 ESUB
0.000E+00 E
COLL   0.000E+00 EMDOT   0.000E+00 ECDOT   0.000E+00
```

Any ADJUST indicate any Integration Step

```
   RMIN = 1.0E-02  DTMIN = 1.0E-03 RHOM = 6.8E+01 RSCALE = 7.4E-01 RSMIN = 6.5E-01  ECLOSE = 1.00  TC =    0
```

KS integration parameters

```
      END RUN    TIME[Myr] =    0.00  TOFF/TIME/TTOT=      0.00000000      1.00000000      1.00000000  CPUTOT =    0.0  ERRTOT =-9.82615D-08  DETOT =-2.457

     0 INTEGRATION INTERVAL =    1.00    NIRR=          0 NIRRB=          0 NREG=          0 NKS=          0


       PER TIME UNIT: NIRR= 0.00000D+00 NIRRB= 0.00000D+00 NREG= 0.00000D+00 NKS= 0.00000D+00
Total CPU=  0.48609003140882123
```

Funded by the European Union

The NBODY6++GPU: let's get started: Output files

```
bdat.8      bev.82_7     bwdat.19_6  conf.3_7   merger.84      sev.83_4
bdat.9_0    binary.40_1  bwdat.19_7  dat.10     mix.15         sev.83_5
bdat.9_1    binary.40_2  chstab.81   degen.4    nbflow.41      sev.83_6
bdat.9_2    binary.40_3  cirdiag.71  esc.11     nbody6++.avx   sev.83_7
bdat.9_3    binary.40_4  cirdiag.75  event.35   shrink.14
bdat.9_4    binary.40_5  cirdiag.96  fort.34    out            single.40_0
bdat.9_5    binary.40_6  coll.13     global.30  pbin.18        single.40_1
bdat.9_6    binary.40_7  comm.2      hbin.39    rocdeg.22      single.40_2
bdat.9_7    binev.17     comm.2_0    highv.29   roche.85       single.40_3
bev.82_0    bs.91        conf.3_0    hinc.44    sediag.25      single.40_4
bev.82_1    bwdat.19_0   conf.3_1    hirect.16  sediag.38      single.40_5
bev.82_2    bwdat.19_1   conf.3_2    histab.73  sediag.43      single.40_6
bev.82_3    bwdat.19_2   conf.3_3    input      sev.83_0       single.40_7
bev.82_4    bwdat.19_3   conf.3_4    lagr1.31   sev.83_1       status.36
bev.82_5    bwdat.19_4   conf.3_5    lagr2.32   sev.83_2       symb.20
bev.82_6    bwdat.19_5   conf.3_6    lagr.7     sev.83_3       wdcirc.95
```

The NBODY6++GPU: let's get started: Output files

single.40_X,

NAME, M[M*], X(1:3)[pc], V(1:3)[km/s], POT[NB], RS[R*], L[L*], Teff[K], MCORE[M*], RSCORE[R*], K*

```
## N_SINGLE          247  Time[NB]    0.0000000000000000
          69  0.138743058      0.193287954     -0.171732694      6.00698441E-02 -0.720279574     -4.86821942E-02  0.782
          70  0.162363857      0.186007276     -0.149012357      2.69592479E-02 -0.777357697     -6.53275624E-02  0.781
         121  0.582071304     -1.47736184E-02  0.158564091      0.135727867     0.604919076     -1.06824756     -0.162
         122  0.363979876     -6.89574927E-02  0.164750874      0.152300507     0.537554562     -0.973333299    -8.98
           5  0.349002808      0.274906069      3.04735322E-02 -0.132446244    -0.320778221     -0.678983867    -0.840
           6  0.168906897      0.269460618      5.25103509E-02 -0.112933353    -0.525037766     -0.603348255    -0.952
          31  0.923223257      0.109903283      1.84809079E-03  0.166919202     0.133556470     -1.24844599     -0.982
          32  0.333079964      0.146618441      3.18649132E-03  0.156022549     0.235582769     -1.19925678     -0.727
           3  0.363134623      0.124351598      0.150900245    -0.236131668     0.305598438     -1.66195190     -1.43
           4  0.189846113      0.140859872      0.162547126    -0.234480426     0.613352358     -1.68615890     -1.18
         157  3.84371376      -8.53632912E-02  -8.65354612E-02   7.96079040E-02 -0.942823648     -0.727606595    -0.296
         158  3.44998264       6.48547933E-02  0.107971109     -6.95804320E-03 -7.21068010E-02 -0.363535583     6.69
```

Funded by the European Union

The NBODY6++GPU: let's get started: Output files

binary.40_X,

```
NAME1, NAME2, NAME(ICM), M1[M*], M2[M*], XCM(1:3)[pc],
*          VCM(1:3)[km/s], XREL(1:3)[AU], VREL(1:3)[km/s], POT[NB],
*          semi[AU], ecc, P[days], Gamma,
*          RS1[R*], RS2[R*], L1[L*], L2[L*], Teff1[K], Teff2[K],
*          MCORE1[M*], MCORE2[M*], RSCORE1[R*], RSCORE2[R*],
*          K*1, K*2, K*(ICM)
*
*     (binary parameter: 0 means inner binary, 1 means outer component )
*     (single parameter: 1/2, means inner binary components, 3 means outer component )
*     (merger parameter: 0 inner binary, 1 outer binary)
*     (POT: potential, sum of - G m_j/r_j)
```

```
## N_BINARY          73  Time[NB]   0.0000000000000000
          15          16       446  0.174306944      0.333937973     -0.122557029      -1.14016794E-02   9.93460068E-04
          67          68       447  0.195164412      0.640439093      0.194081068       2.11306568E-02  -0.180667594
         111         112       448  0.197046697      0.226330519      0.220915616       3.47995162E-02  -0.128341794
          71          72       449  0.218287528      0.106719159     -0.103002414      -0.168809712      0.108195014
```

Funded by the European Union

The NBODY6++GPU: let's get started: Output files

merger.40_X,

```
*    merger: NAME1,NAME2,NAME3,NAME(INCM), M1[M*], M2[M*], M3[*],
*          XCM(1:3)[pc], VCM(1:3)[km/s], XREL0(1:3)[AU], VREL0(1:3)[km/s],
*          XREL1(1:3)[AU], VREL1(1:3)[km/s],POT[NB],
*          semi0[AU], ecc0, P0[days], semi1[AU], ecc1, P1[days],
*          RS1[R*], RS2[R*], RS3[R*], L1[L*], L2[L*], L3[L*], Teff1[K], Teff2[K], Teff3[K],
*          MCORE1[M*], MCORE2[M*], MCORE3[M*], RSCORE1[R*], RSCORE2[R*], RSCORE3[R*]
*          K*1, K*2, K*3, K*(INCM),
*
*    (binary parameter: 0 means inner binary, 1 means outer component )
*    (single parameter: 1/2, means inner binary components, 3 means outer component )
*    (merger parameter: 0 inner binary, 1 outer binary)
*    (POT: potential, sum of - G m_j/r_j)
```

```
## N_MERGER          4  Time[NB]   8.0000000000000000
        57         58         34        -57   2.45512795      0.917761147      1.13693309     -0.221200913      4.4
        95         96        186        -95   0.230416968     0.150982484     0.944365740     0.621198297     -0.96
```

The NBODY6++GPU: let's get started: Output files

How to recognise stellar types?

Type	Description
0.	Deep or fully convective low-mass MS star (CS)
1.	Main-sequence star (MS)
2.	Hertzsprung gap star (HG)
3.	First giant branch (RGB)
4.	Core helium Burning
5.	First asymptotic giant branch (early AGB)
6.	Second asymptotic giant branch (late AGB)
7.	Main-sequence naked helium star
8.	Hertzsprung gap naked helium star
9.	Giant branch naked helium star
10.	Helium white dwarf (He white dwarf)
11.	Carbon/oxygen white dwarf (CO white dwarf)
12.	Oxygen/neon white dwarf (ONe white dwarf)
13.	Neutron star
14.	Black hole
15.	Massless supernova/remnant

The NBODY6++GPU: let's get started: Output files

Simplified output

sev.83_#: info about single stars

bev.82_#: info about binary stars

coal.24: info about stellar/compact object coalescence

coll.13: info about stellar/compact object collisions

esc.11: info about escaping objects

global.30: general info about the system and the run

lagr.7: lagrangian radii (284 columns, 18 values of the lagrangian radii)

Funded by the European Union

The NBODY6++GPU: let's get started: Output files

Simplified output

sev.83_#

Header-1	NS, TIME[Myr]
N-Label	TIME[NB], I, NAME, K*, RI[RC], M[M*], log10(L[L*]), log10(RS[R*]), log10(Teff[K])

The NBODY6++GPU: let's get started: Output files

Simplified output

bev.82_#

Header-1	NPAIRS, TIME[Myr]
N-Label	TIME[NB], I1, I2, NAME(I1), NAME(I2), K*(I1), K*(I2), K*(ICM), RI[RC], ECC, log10(P[days]), log10(SEMI[R*]), M(I1)[M*], M(I2)[M*], log10(L(I1)[L*]), Log10(L(I2)[L*]), Log10(RS(I1)[R*]), Log10(RS(I2)[R*]), Log10(Teff(I1)[K]), Log10(Teff(I2)[K])

The NBODY6++GPU: let's get started: Output files

Simplified output

esc.11 | TIME[Myr], M[M*], EESC, VI[km/s], K*, NAME

to obtain info about escaping binaries:

grep 'BINARY ESCAPE' *.out

Funded by the European Union

The NBODY6++GPU: let's get started: Output files

Simplified output

coal.24 (new format)

WHICH1, IQCOLL, TTOT [NB], NAME(I1,I2), K*(I1,I2,I), M(I1)[NB], M(I2)[NB],
R12[NB], ECC, SEMI, EB, DP, P[days],
M1[M*], M2[M*], MTOT[M*], DM[M*],
RAD1[R*], RAD2[R*],
RCOLL[R*], VINF[km/s],
RI[NB], VI[NB]

The NBODY6++GPU: let's get started: Output files

Simplified output

coll.13

RBAR, $\langle M \rangle$[M*], M1[M*], TSCALE, NB0, NZERO
TIME[NB], NAME(I1), NAME(I2), K*(I1), K*(I2), K*(INEW), M(I1)[M*], M(I2)[M*], M(INEW)[M*], DM[M*], RS(I1)[R*], RS(I2)[R*], RI/RC, R12[R*], ECC, P[days]

The NBODY6++GPU: let's get started: Restart?

"If something can go wrong, will go wrong"

Dealing with *N*-body simulations means understanding this sentence very well.

Luckily, when this happens, we can restart our simulations:

- Copy one of the comm.2_# files (possibly the last) in comm.1, which will be used to get the latest info about the simulations to restart
- Change KSTART = 2 in the input file
- Run the code

BUT! Good practice's rule:
In the good old times it could happen that performing the above would overwrite some files (quite randomly). Now everything should be fixed, nonetheless…

Good practice #1: save your files in separate folders, so to make sure that all run's chunks are safely stored.

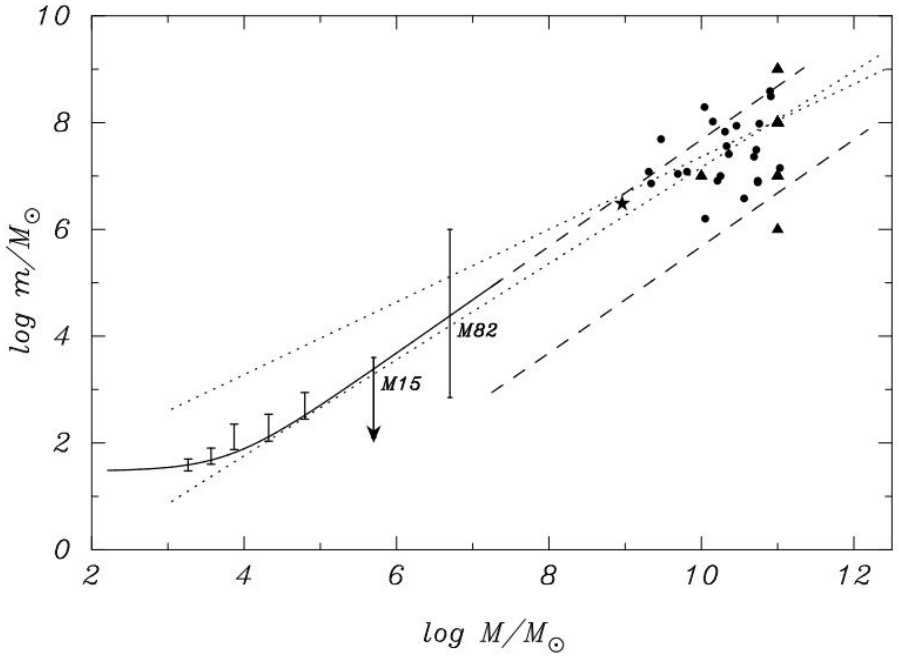
Good practice #2: assign to the output and error files the shell's PID

```
i=$(($$+1000000))
OMP_NUM_THREADS=6 ./nbody6++.sse < N10k_noDat10.inp 1> run_Arca."$i".out 2> run_Arca."$i".err &
```

Funded by the European Union

NBODY6++GPU exercise. The Runaway Growth of Intermediate-Mass Black Holes in Dense Star Clusters

Simon Portegies-Zwart and Steven McMillan, in 2002, developed a theoretical framework, supported by a series of *N*-body simulations, showing that intermediate-mass black holes could form from the collapse of a very massive star built-up via runaway stellar collisions.



Our exercise

N = 6000

Nbin = 0

$m_{*min,max}$ = (20-150) M_{SUN} → m_{aver} ~ 42 M_{SUN}

IMF = Kroupa (2001)

R_{HALF} = 0.5 pc

Z = 0.0001

Stellar Evolution Level = "C"

External Field: point-like

$M_{Galaxy} = 1.78 \times 10^{11} M_{SUN}$

R_{orbit} = 13.3 kpc

NBODY6++GPU exercise. The Runaway Growth of Intermediate-Mass Black Holes in Dense Star Clusters

Results discussed and commented in the notebook

https://github.com/marcasedda/acme_tutorials

Backup Slides

What is the *N*-body problem?

Consider the gravitational potential of the cluster as a smooth function Φ(x), the distribution of energy is described by a

$$f(\vec{r}, \vec{v}, t)\mathrm{d}x^3\,\mathrm{d}v^3\,\mathrm{d}t$$ *distribution function*

i.e. the probability that a star is in the given range of space, velocity, and time.

What is the *N*-body problem?

Consider the gravitational potential of the cluster as a smooth function Φ(x), the distribution of energy is described by a

$$f(\vec{r},\vec{v},t)\mathrm{d}x^3\,\mathrm{d}v^3\,\mathrm{d}t$$ *distribution function*

i.e. the probability that a star is in the given range of space, velocity, and time.

If encounters are negligible the phase-space probability density around a star remains constant, i. e.

$$\frac{\mathrm{d}f}{\mathrm{d}t} = 0$$ collisionless Bolzmann or Vlasov equation

What is the *N*-body problem?

Consider the gravitational potential of the cluster as a smooth function Φ(x), the distribution of energy is described by a

$$f(\vec{r},\vec{v},t)\mathrm{d}x^3\,\mathrm{d}v^3\,\mathrm{d}t \qquad \textit{distribution function}$$

i.e. the probability that a star is in the given range of space, velocity, and time.

If encounters are negligible the phase-space probability density around a star remains constant, i. e.

$$\frac{\mathrm{d}f}{\mathrm{d}t} = 0 \qquad \text{collisionless Bolzmann or Vlasov equation}$$

The motion of stars can be described in terms of a global acceleration determined by the gravitational potential energy, i.e.

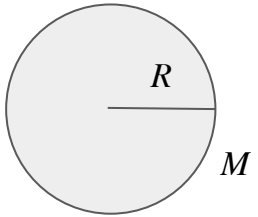
$$\ddot{\vec{r}} = -\nabla\Phi$$

And the numerical problem reduces to solve N independent equations. The computational load would simply by ~ N.

Solving the Gravitational *N*-body problem: long term evolution

- Under the simplest approximation possible, a star crosses its cluster over a “Crossing” Timescale

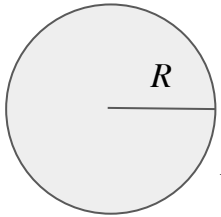
$$T_{\rm crs} = \frac{2\pi R}{v} = 2\pi \left(\frac{R^3}{GM}\right)^{1/2} = 0.3{\rm Myr}\left(\frac{M}{10^5 {\rm M}_\odot}\right)^{-1/2}\left(\frac{R}{1{\rm pc}}\right)^{3/2}$$



Solving the Gravitational *N*-body problem: long term evolution

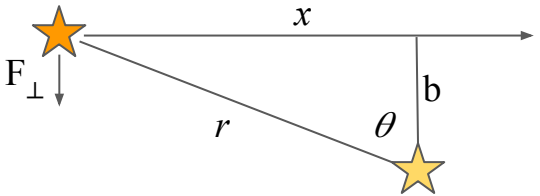
- Under the simplest approximation possible, a star crosses its cluster over a “Crossing” Timescale

$$T_{\rm crs} = \frac{2\pi R}{v} = 2\pi \left(\frac{R^3}{GM}\right)^{1/2} = 0.3{\rm Myr}\left(\frac{M}{10^5 {\rm M}_\odot}\right)^{-1/2}\left(\frac{R}{1{\rm pc}}\right)^{3/2}$$



- A star passes within a distance b from another star, which exerts a force

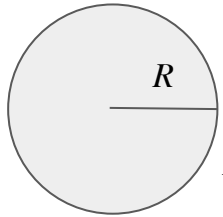
$$F_\perp = Gm^2 / (b^2 + x^2) \cos\theta = Gm^2\, b\, (b^2 + x^2)^{-3/2} = Gm^2/b^2 [1+(vt/b)^2]^{-3/2}$$



Solving the Gravitational *N*-body problem: long term evolution

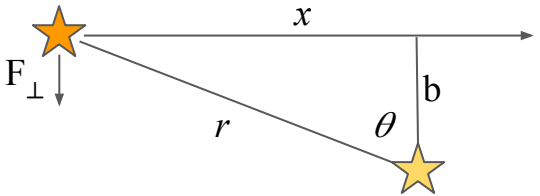
- Under the simplest approximation possible, a star crosses its cluster over a “Crossing” Timescale

$$T_{\rm crs} = \frac{2\pi R}{v} = 2\pi \left(\frac{R^3}{GM}\right)^{1/2} = 0.3{\rm Myr}\left(\frac{M}{10^5 {\rm M}_\odot}\right)^{-1/2}\left(\frac{R}{1{\rm pc}}\right)^{3/2}$$



- A star passes within a distance b from another star, which exerts a force

$$F_\perp = Gm^2 / (b^2 + x^2) \cos\theta = Gm^2\, b\, (b^2 + x^2)^{-3/2} = Gm^2/b^2 [1+(vt/b)^2]^{-3/2}$$



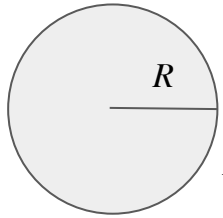
- Repeated interactions slowly changes the star trajectory, until it “loses memory” of its initial conditions

Funded by the European Union

Solving the Gravitational *N*-body problem: long term evolution

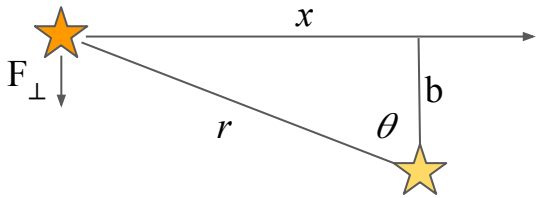
- Under the simplest approximation possible, a star crosses its cluster over a “Crossing” Timescale

$$T_{\rm crs} = \frac{2\pi R}{v} = 2\pi \left(\frac{R^3}{GM}\right)^{1/2} = 0.3{\rm Myr}\left(\frac{M}{10^5 {\rm M}_\odot}\right)^{-1/2}\left(\frac{R}{1{\rm pc}}\right)^{3/2}$$



- A star passes within a distance b from another star, which exerts a force

$$F_\perp = Gm^2 / (b^2 + x^2)\cos\theta = Gm^2\, b\, (b^2 + x^2)^{-3/2} = Gm^2/b^2\,[1+(vt/b)^2]^{-3/2}$$



- Repeated interactions slowly changes the star trajectory, until it “loses memory” of its initial conditions

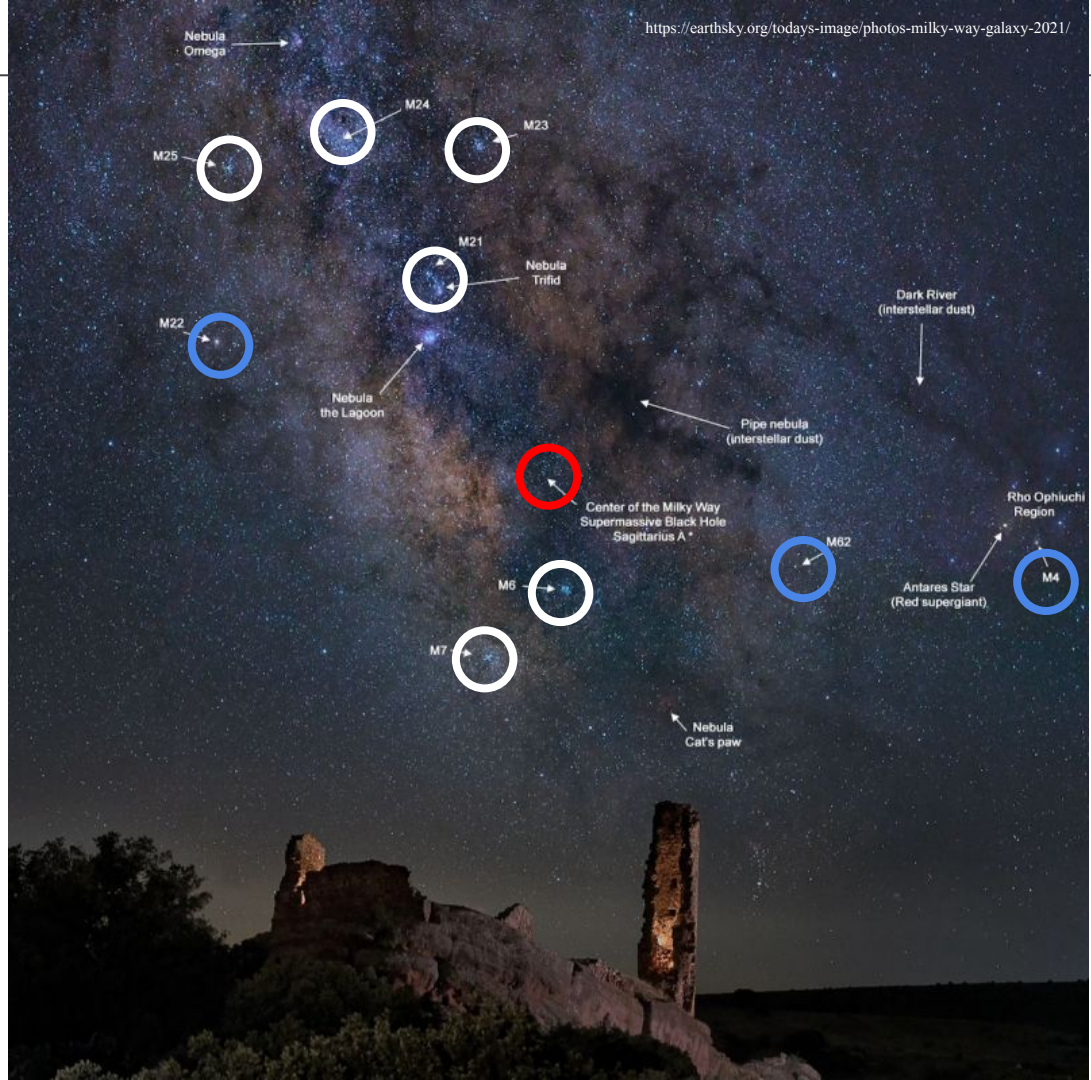
- The process that reshuffles the energy and momentum among stars is called “Relaxation”, operating on a timescale

$$T_{\rm rlx} = 0.34\frac{\sigma^3}{G^2 m \rho \ln\Lambda} = \frac{0.78{\rm Gyr}}{\ln\Lambda}\frac{1\,{\rm M}_\odot}{m_*}\left(\frac{M}{10^5{\rm M}_\odot}\right)^{1/2}\left(\frac{r}{1\,{\rm pc}}\right)^{3/2}$$

Funded by the European Union

Galaxy's zoology - many star cluster flavours:

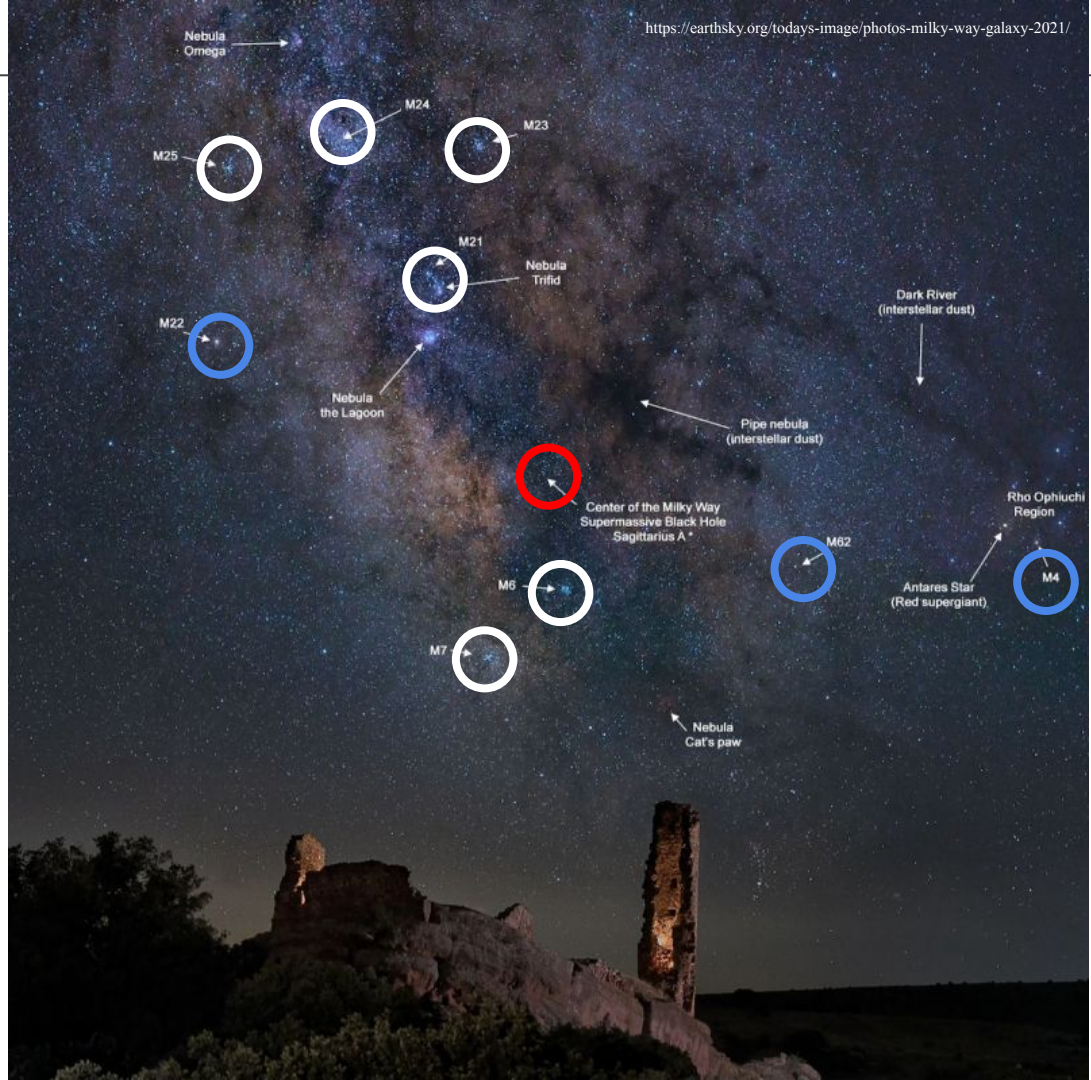
cluster type	M (M_{SUN})	R (pc)	T_{crs} (Myr)	T_{rlx} (Gyr)	T_{age} (Gyr)
open clusters	10^2	2	26	0.03	0.01-1
young clusters	10^4	2	2.6	0.1	0.1-2
globular clusters	10^5	2	0.8	0.2	10-14
nuclear clusters	10^7	2	0.1	1.6	1-10



ACME
Funded by the European Union

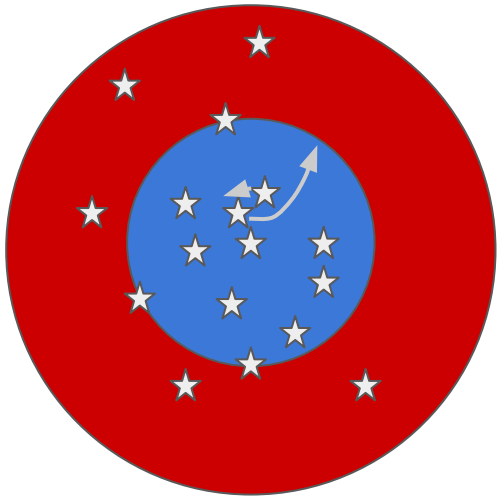
Galaxy's zoology - many star cluster flavours:

cluster type		T_{rlx} (Gyr)	T_{age} (Gyr)
open clusters	**STAR CLUSTERS ARE COLLISIONAL SYSTEMS!**	0.03	0.01-1
young clusters		0.1	0.1-2
globular clusters		0.2	10-14
nuclear clusters		1.6	1-10

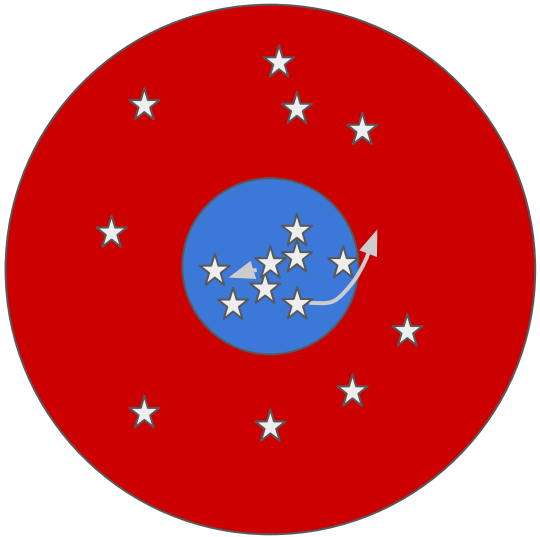


Solving the Gravitational *N*-body problem: relaxation, collapse, and long-term evolution

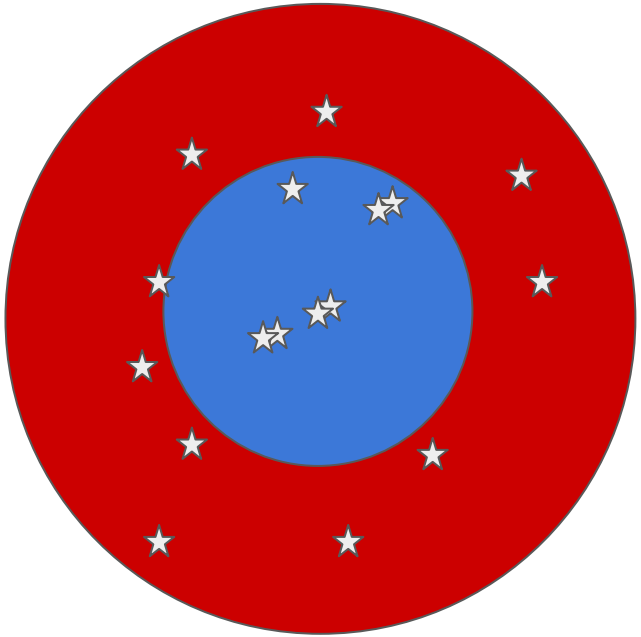
Halo (colder)
Core (hotter)



Interactions starts, the core slowly contracts, while the halo starts expanding



The core reaches maximum contraction (core collapse) and rebound



Both layers keep expanding, interactions become rarer and less energetic

Time

Funded by the European Union

Solving the Gravitational *N*-body problem: relaxation, collapse, and long-term evolution

(file input.F, routine READSE)

- ecflag = 1 Electron-capture SN
- wdflag = 1 Modified Mestel cooling for WDs
- nsflag = 4 Delayed model from Fryer+12 for NSs
- psflag = 1 PPISNe/PISNe from Belczynski+16
- mdflag = 4 Mass-loss without bi-stability jump + metallicity correction from Belczynski+10
- bhflag = 2 Fallback scaled kicks for BHs
- nwind = 0.477 Reimer's mass loss coefficient for giant winds from McDonald & Zijlstra 2015
- bwind = 0.0 Companion-enhanced mass-loss factor
- flbv = 1.5 Coefficient for LBV rate (if mdflag==2)
- kmech = 4 Neutrino-driven natal kick for BHs and NSs (Banerjee+20)
- disp = 265 Dispersion of Maxwellian distribution for kick distribution (all velocities in km/s)
- ecsig = 3 Dispersion of Maxwellian distribution for kick distribution for ECSNe remnants
- wdsig1 = 2 Dispersion of Maxwellian distribution for kick distribution for COWDs
- wdsig2 = 2 Dispersion of Maxwellian distribution for kick distribution for ONeWDs
- wdkmax = 6 Maximum kick for WDs
- vvfac = 0 No kick rescaling
- lambd1 = 0.0 If star is in between HG and GB naked He phases, λ_{CE} is calculated depending on the stellar phase, wheres is λ_{CE}= 0.5 for MS and naked MS stars
- alphac = 1 Factor scaling the amount of orbital energy used for envelope liberation in CEE (α_{CE})
- bhspinfl = 0 BH natal spins (0: no spins, 1: uniform, 2: Gaussian with mean 0.5 and sigma 0.2, 3: Maxwellian with sigma=0.2, 4: Geneva models, 5: MESA models)
- kicktype = -2 Relativistic kicks (-2: no kicks, -1: v=10^4km/s, 0: analytic fit to zero spins GW kicks, 1: full kicks e.g. Arca Sedda+20)
- xbeta = 0.125 Wind velocity factor (lower limit from Hurley et al 2002)
- xxi = 1 Factor to reduce spin angular momentum change owing to wind accretion (Hurley et al 2002)
- epsnov = 0.001 Fraction of material accreted onto a WD ejected in a nova
- eddfac = 100 Eddington limit for accretion on a degenerate object (Hurley et al 2002)
- gamm1 = -1 Lost material carries with it the specific angular momentum of the primary (Hurley et al. 2002)