MANYBODY:

an accompanying software guide to

Advanced School and Workshop on Computational Gravitational Dynamics

Leiden, May 3-13, 2010

and previously

Escuela Internacional de Simulaciones Galacticas Y Cosmológicas de N–cuerpos

International School on Galactic and Cosmological N–body Simulations

INAOE, Tonantzintla,

Puebla, Mexico, 23 July - 4 August 2006

Zomer school over direkte N–deeltjes simulaties

Summer school on direct N–body simulations

Astronomical Institute *'Anton Pannekoek'*,

Amsterdam, Netherlands, 24-30 July 2005

École de Dynamique Numerique Ncorps

School on Numerical N–body Dynamics

Observatoire astronomique de Strasbourg, France,

Strasbourg, 19-22 March 2004

Version 2.0

Spring 2010

Last document revised: May 3, 2010
URL: http://www.astro.umd.edu/nemo/manybody/

Contents

List of Tables

List of Figures

Preface

What lies in front of you is an accompanying guide to some of the software you might find useful this school. It is still very much work in progress, with many sparse and unfinished sections.

A large number of programs are within the NEMO, Starlab and now AMUSE packages, though a number of useful programs and smaller packages that are available elsewhere have been assembled here and made available via the *manybody* environment. See also Appendix B for a more detailed description of this *manybody* environment.

One word of caution though: this Guide is **not** a coherent story that you should necessarily read front to back, it is more encyclopedic and will hopefully give you an impression of the capabilities of this software in case you may need it during this week or in your subsequent research. On most topics in this guide you can find much more in-depth information in other manuals and of course the source code (needless to say, we would like to advocate some type of open source policy for our codes).

Unix Shell

The *manybody* guide assumes some knowledge of Unix. This means Linux, Solaris and MacOSX should be fine, but although cygwin on MS-Windows gets fairly close, your editor couldn't stomach some of its limitations[1]. Most examples show simple snippets of C-shell command lines; their prompt (e.g. `2%`) contains a counter which is normally started at 1 for each section for easy reference. There are only a few minor places where these `csh` and `sh` examples would differ. We just remind you of the most important ones:

1. Redirect and merge stderr and stdout into one file (see also NEMO's `redir` program):

```
 sh:  p1 > log 2>&1
csh:  p1 >& log
```

2. Redirect stdout and stderr into separate files:

```
 sh:  p1 > out 2>err
csh:  (p1 > out) >& err
```

3. Pipe both stdout and stderr into the next program:

[1] YMMV, as they say in this business

```
 sh:  p1 2>&1 | p2
csh:  p1 |& p2
```

Also noteworthy is that in most places where a random seed is used (e.g. the NEMO `seed=0` keyword would take a new seed based on the current date and time) a fixed seed is taken (usually `seed=123`) so the example can actually be exactly reproduced sans compiler and optimization differences. In "real life" you may not always want to do this!

Python shell

Examples

You will find shell script examples in the manual, at the top of the example you should find a line like

```
# File: examples/ex1
```

which means the example can be found in the directory

```
$NEMO/usr/manybody/examples/ex1
```

Getting More Help

After your account has been setup to use the *manybody* environment (see also Appendix A), there are several ways to get more help besides reading this document:

- For most programs the `-h` and/or the `--help` command line argument will give a short reminder of the options and their defaults. This is true for NEMO as well as Starlab programs. For NEMO programs the option `help=h` will also give much more extensive help on the keywords.
- The Unix `man` and `gman` commands can be used to view manual pages for all NEMO programs. Notably the *index(1)* and *programs(8)* manual pages given an overview of most commands.
- For those packages that have html formatted information, they are linked in `manybody/index.html`, whereever this may reside on your system.
- Buy your friendly teachers a Coffee or Beer

Acknowledgements

I would like to thank my co-authors in spirit, Douglas Heggie, Piet Hut, and Junichiro Makino for their contributions, Simon Portegies Zwart and Christian Boily for the MODEST schools in Amsterdam and

Strasbourg respectively, where much of this material was first written down. In the true spirit of open source, numerous authors have contributed their software, which you can find on the DVD. Without their generous contributions this guide would not be half the size and a quarter of the contents. Hopefully we will be able to instill this spirit on the reader too.

Chapter 1

Introduction

Although we will primarely cover direct N-body integrators in this school, we should recall several `types` are commonly in use in astrophysics:

1. Self-consistent direct N-body integrations[1]

$$\ddot{\vec{\mathbf{r}}} = -G \sum \frac{m}{\Delta r^3} \Delta \vec{\mathbf{r}} \tag{1.1}$$

2. Orbit integrator in a general Gravitational Potential[2]

$$\ddot{\vec{\mathbf{r}}} = -\nabla \Phi(\vec{\mathbf{r}}) \tag{1.2}$$

3. Orbit integrator in a Flow Potential[3]

$$\dot{\vec{\mathbf{r}}} = \vec{\mathbf{v}}(\vec{\mathbf{r}}) \tag{1.3}$$

Another way to divide up the field is by methodologies. There are arguably three ways to solve the N-body problem: direct Particle-Particle (PP), Particle-Mesh (PM) and Smooth Field Particle (SFP), also sometimes referred to as Self Consistent Field (SCF) method. Various hybrid schemes also exist, e.g. P^3M.

[1] See Table 2.1
[2] `potcode` in NEMO
[3] `flowcode` in NEMO

Chapter 2

Integrators

We have made a few N-body integrators available for this school, most are summarized in Table 2.1. A brief description of all of them will follow, and for some a more detailed example will follow how they can be used in the *manybody* environment.

Table 2.1: A few good N-body codes

code	*method*	*author(s)/version*	*data format*
`firstn`	PP	von Hoerner (1960; via Aarseth)	
`nbody0`	PP	Aarseth (Binney & Tremaine)	s
`nbody{1,2,4}`	PP	Aarseth (w/ NEMO interface)	b/s
`nbody{6}`	PP	Aarseth	b
`fewbody`	PP	Fregeau	d
`hnbody`	PP	Rauch & Hamilton 2004	
`kira`	PP	Starlab team	d
`treecode`	PP	Barnes & Hut 1986, Hernquist 1989	s
`gyrfalcON`	PP	Dehnen 2000	s
`gadget`	PP	Springel 2001, 2005	g
`scfm`	SFP	Hernquist 1992	205
`quadcode`	SFP	Barnes & White 1986	s
`CGS`	SFP	Trenti et al. 2005	
`galaxy`	PM	Sellwood	b
`superbox`	PM	Fellhauer et al. 2000	
`TPM`	PM	Bode	
`partree`	PP	Dubinski	

2.1 firstn

The first N-body code (skipping the pioneering analog experiments by Holmberg (1941)[1] was arguably written by von Hoerner (1960) and has recently been resurrected by Aarseth and is also available as `firstn` in NEMO. Here is an example running the code for two crossing times on a self-generated 16-body Plummer sphere configuration

```
   # File: examples/ex1
   # sample input parameter file
1% cat $NEMO/usr/aarseth/firstn/input
16 12.0 0.0 1.0 2.0
1.0 1.0 1.0 0
0.5 0

   # integrate
2% firstn < $NEMO/usr/aarseth/firstn/input
     N =   16  ETA =  12.0  EPS =  0.000

 T =   0.0  Q = 0.38  STEPS =      0  DE =  0.000000  E = -0.357554  TC =   0.0
 ERRORS    RCM VCM DE/E DZ     0.00E+00  0.00E+00  0.00E+00  0.00E+00

 T =   2.3  Q = 0.55  STEPS =   1536  DE =  0.000000  E = -0.357554  TC =   1.4
 ERRORS    RCM VCM DE/E DZ     3.17E-16  7.10E-16  2.38E-07  2.97E-08

 T =   4.5  Q = 0.53  STEPS =   3624  DE =  0.000001  E = -0.357554  TC =   2.7
 ERRORS    RCM VCM DE/E DZ     2.26E-16  8.86E-16  7.73E-07  2.99E-08

   # get some online help on this program
3% man firstn
```

It simply prints out the time T, the virial ratio Q (Q=T/W, thus 0.5 means virial equilibrium), number of STEPS taken, energy conservation and total energy, and another line on the center of mass motion. No actual snapshots are stored in this simulation, although the code layout is very similar to that of `nbody0`, and could be adapted quite easily (see below)

You can find the source code in `$NEMO/usr/aarseth/firstn/`. For an explanation of the input parameter file, see the Unix manual page for this program.

2.2 nbody0

A basic implementation of the variable timestep integrator (an essential ingredient to collisional stellar dynamics) was published in Appendix 4B in Binney & Tremaine (1987). You can find this implementation in Fortran (`nbody0`) as well as an identical C (`nbody00`) in NEMO, with both a simple ASCII dataformat, as well as the NEMO *snapshot* format. Aarseth likes to refer to this version as the "Mickey Mouse" version, and really prefers you to use the full version, `nbody1`, or better yet, `nbody4` or `nbody6` (see below).

[1] see also `mkh41`

```
   # File: examples/ex2
   # reminder to the valid command line parameters and defaults
1% mkplummer help=
mkplummer out=??? nbody=??? mfrac=0.999 rfrac=22.8042468 seed=0 time=0.0 zerocm=t
  scale=-1 quiet=0 massname= massexpr=pow(m,p) masspars=p,0.0 massrange=1,1 headline= VERSION=2.6a

   # generate a Plummer sphere with 256 bodies, but with reproducable data
2% mkplummer p256.in 256 seed=123

   # what does nbody00 do?
3% nbody00 help=
nbody00 in=??? out=??? eta=0.02 deltat=0.25 tcrit=2 eps=0.05 reset=t f3dot=f options= VERSION=2.0a

   # some more extensive inline help on the keywords
4% nbody00 help=h
in               : Input (snapshot) file [???]
out              : Output (snapshot) file [???]
eta              : Accuracy parameter - determines timesteps [0.02]
deltat           : When to dump major output [0.25]
tcrit            : When to stop integrating [2]
eps              : Softening length [0.05]
reset            : Reset timestep after datadump (debug) (t|f) [t]
f3dot            : Use more advanced timestep determination criterion? [f]
options          : Optional output of 'step' into AUX []
VERSION          : 13-mar-04 PJT [2.0a]

   # finally, integrate this for about 2 crossing times
5% nbody00 p256.in p256.out
time = 0   steps = 0   energy = -0.257297 cpu =    0.00117 min
time = 0.25   steps = 2519   energy = -0.257303 cpu =      0.006 min
time = 0.5   steps = 5428   energy = -0.257306 cpu =     0.0118 min
time = 0.75   steps = 8207   energy = -0.2573 cpu =     0.0173 min
time = 1   steps = 10914   energy = -0.257304 cpu =     0.0228 min
time = 1.25   steps = 13543   energy = -0.25731 cpu =     0.0282 min
time = 1.5   steps = 16461   energy = -0.257311 cpu =      0.034 min
time = 1.75   steps = 19345   energy = -0.257315 cpu =     0.0398 min
time = 2   steps = 22265   energy = -0.257313 cpu =     0.0462 min
Time spent in searching for next advancement: 0.1
Energy conservation: -1.59973e-05 / -0.257297 = 6.21707e-05
Time resets needed 0 times / 9 dumps
```

Notice that when running `mkplummer` the first two parameters (`in=` and `nbody=`) were not explicitly named, because they were given in the order known to the program. This is a feature of the NEMO command line interface.

2.3 nbody1, nbody2

These two original Aarseth codes are both available with a NEMO wrapper program (`runbody1` and `runbody2`). It will run the compiled Aarseth fortran code (`nbody1` and `nbody2`) in its own run directory and optionally produce *snapshot* files for further analysis. `nbody2` is much like `nbody1`, except it treats forces with the

Ahmad-Cohen neighbor scheme (see Ahmed Cohen 1973).

```
  # File: examples/ex3
  1% mkplummer p100 100 seed=123

  2% runbody1 in=p100 outdir=run1 nbody=100

       N  NRAND    ETA   DELTAT   TCRIT   QE        EPS

     100      0   0.020     0.2     2.0   2.0E-05   5.0E-02


       OPTIONS      1   2   3   4   5   6   7   8   9  10  11  12  13  14  15

                    1   2   0   2   0   1   0   0   0   0   1   0   0   0   0

       SCALING:   SX  =  0.53  E = -4.69E-01  M(1) = 1.00E-02  M(N) = 1.00E-02  <M> = 1.00E-02


       SCALING PARAMETERS:   R* = 1.00E+00  M* = 1.00E+02  V* = 6.56E-01  T* = 1.49E+00


 T =   0.0  Q = 0.00  STEPS =      0  DE =  0.000000  E = -0.251037  TC =   0.0

 <R> =  1.99  RCM =  0.0000  VCM =  0.0000  AZ =   0.00000  T6 =   0  NRUN =  1

   BINARY   71  92  0.010  0.010   0.0  0.2751    1.0  0.5503  1.10  1.000    0
   BINARY   87  92  0.010  0.010  -0.1  0.0756    6.8  0.1512  1.32  1.000    0
...

 T =   5.7  Q = 0.57  STEPS =  37920  DE = -0.000003  E = -0.251050  TC =   2.0

 <R> =  0.85  RCM =  0.0000  VCM =  0.0000  AZ =  -0.00001  T6 =   8  NRUN =  1


         END RUN   TIME =    5.66  CPUTOT =  0.01  ERRTOT = -0.00002
```

Here you are also witnessing a current quirk of this wrapper: Although the number of bodies is known from the input file, it has to be re-specified.

Notice that at T=0 two binaries were already found, in fact star 92 was involved in both binaries.

2.4 nbody4

The Aarseth `nbody4` code has been recently made available with a GRAPE interface (a.k.a. Nbody4). A standalone version (a.k.a. Brut4) is available as `nbody4` in NEMO. A NEMO frontend, much like the previous ones, is available as `runbody4`. One novel addition to this version of `nbody4` is the capability of running one of several stellar evolution codes (e.g. Eggleton, Tout & Hurley, Chernoff–Weinberg). An extensive manual has recently been made available on how to run it (Aarseth, 2006). Currently the only other code capable

of doing combined dynamical and stellar evolution is the `kira` code in Starlab. A very newe project, `muse`[2], aims to modularize this approach. This is now superseded by the AMUSE project[3],

2.5 nbody6

This code is available as is, and with minor modifications could be made to run via clones of `runbody4`. Fairly extensive documentation is available on how to modify and run the code.

`nbody6` is available in `$NEMO/usr/aarseth/nbody6`, There is also a recent manual (see Aarseth 2004). Running `nbody6` is very similar to `nbody4`

2.6 nbody6++

Spurzem (see Khalisi & Spurzem 2003) published an MPI enabled version of NBODY6, dubbed `nbody6++`[4]. The input parameter list (including the well known `KZ()` and `BZ()` arrays, now counts 83 (`nbody1` counted 35 parameters, `nbody2` has 42) !

2.7 fewbody

Fregeau et al. (2004) released *fewbody*, a software package for doing small-N scattering experiments. It handles an arbitrary number of stars, and understands arbitrarily large hierarchies. For example, if the outcome of a scattering experiment between two triples is a stable hierarchical triple of binaries, *fewbody* is smart enough to understand that the system is in that configuration, and terminate the integration once it is considered stable. *fewbody* uses full pairwise K-S regularization and an 8th order Runge-Kutta Prince-Dormand integrator to advance the particles' positions, binary trees as its internal data structures, both to construct the hierarchies, and also to isolate unperturbed hierarchies from the integrator, and the Mardling stability criterion to assess the stability of hierarchies at each level.

fewbody also comes with an OpenGL vizualizer called `GLStarView`, which can visualize a simulation as it is being computed by directly piping the data. You can find fewbody version 0.21 in `$NEMO/usr/fregeau`, as wel as glstarview. Within NEMO the `glnemo2` visualization tool has similar capabilities.

There are 4 programs that come with *fewbody*: `binbin`, `binsingle`, `triplebin` and `cluster`, each of which understand the `-h` flag if you get lost. You should however read at least Fregeau et al. (2004) before using the code.

```
   # smash a binary and single star into each other
1% binsingle -D 1 > try1
```

[2] `http://www.science.uva.nl/sites/modesta/wiki/index.php/Main_Page`
[3] `http://www.amusecode.org`
[4] and no, it is not rewritten in C++

```
PARAMETERS:
  ks=0  seed=0
  m0=1 MSUN  r0=1 RSUN
  a1=10 AU  e1=0  m10=1 MSUN  m11=1 MSUN  r10=1 RSUN  r11=1 RSUN
  vinf=0.2  b=3.1  tstop=1e+06  tcpustop=3600
  tidaltol=1e-05  abs_acc=1e-09  rel_acc=1e-09  ncount=500  fexp=3

UNITS:
  v=v_crit=11.5357 km/s  l=10 AU  t=t_dyn=4.10953 yr
  M=1.5 M_sun  E=3.97022e+45 erg

OUTCOME:
  encounter complete:  t=439.506 (1806.16 yr)  nstar=2 nobj=1:  [0 1:2]  (binary)

FINAL:
  t_final=439.506 (1806.16 yr)  t_cpu=0.26 s
  L0=0.660373  DeltaL/L0=1.24727e-08  DeltaL=8.2366e-09
  E0=-0.213334  DeltaE/E0=1.5916e-06  DeltaE=-3.39541e-07
  Rmin=0.00092399 (1.98608 RSUN)  Rmin_i=1  Rmin_j=2



   # view the simulation using Fregeau's program
2% glstarview < try1

   # and since it uses Starlab 'dyn' format, xstarplot can be used as well
3% xstarplot < try1
```

2.8 hnbody

HNBody (Hierarchical N-body, see Rauch & Hamilton 2004) is optimized for the motion of particles in self-gravitating systems where the total mass is dominated by a single object; it is based on *symplectic integration* techniques in which two-body Keplerian motion is integrated exactly. For comparison, Bulirsch-Stoer and Runge-Kutta integrators are also available. Particles are divided into three basic groups: HPWs (Heavy Weight Particles), LWPs (Light Weight Particles), and ZWPs (Zero Weight Particles).

You can find the code [5] and support files in `$NEMO/usr/hnbody`. The following example illustrates how to run a simulation with the Sun and the Jovian planets:

```
1% hnbody -h
HNBody version 1.0.3 (linux-x86), released 2004/03/12.
See http://janus.astro.umd.edu/HNBody/ for current information.
Relay questions and bug reports to Kevin Rauch <rauch@astro.umd.edu>.

Usage:  hnbody [options] [file1.hnb ...]
Options:

  -b        Benchmark machine's HNBody performance and exit.
```

[5] HNBody is under very active development, and source code will be available for download later this year. The current version, 1.0.3, is only available in binary form

```
  -h        Display this help message and exit.
  -l LFILE  Log diagnostic output to LFILE instead of standard output.
  -q        Quiet mode--do not produce diagnostic output.
  -r RFILE  Set kill recovery file name to RFILE (default: recover.dat).
              (Specify /dev/null to disable signal handling.)
  -s SFILE  Restart the integration using SaveFile SFILE.
              (The original input files must also be given.)
  -t TCPU   Estimate required CPU time; use TCPU seconds of effort and exit.
              (No output files will be created or modified.)
  -v        Display driver/HNBody version information and exit.

2% hnbody $NEMO/usr/hnbody/input/jovian.hnb
#
# HNBody version 1.0.3 (linux-x86), released 2004/03/12.
# See http://janus.astro.umd.edu/HNBody/ for current information.
# Relay questions and bug reports to Kevin Rauch <rauch@astro.umd.edu>.
#
Integrator = Symplectic
IntegCoord = Jacobi Order2 Drift-Kick
Corrector = True
AngleUnit = rad
LengthUnit = AU
MassUnit = Msun
TimeUnit = d
StepSize = 365.25
Tinitial = 0
M = 1.00000597682
N = 6
NLWPs = 1
InputOrder = Mass x1 x2 x3 v1 v2 v3
HWP = 0.000954786104043042 3.409530427945 3.635870038323 0.03424028779975 -0.005604670182013 0.005524493219597 -2.663981907247e-
HWP = 0.00028558373315056 6.612079829705 6.386934883415 -0.1361443021015 -0.004180230691977 0.004003576742277 1.672376407859e-05
HWP = 4.37273164545892e-05 11.16769742623 16.04343604329 0.3617849409933 -0.003265538550417 0.002070723353855 -2.176677219661e-0
HWP = 5.17759138448794e-05 -30.17366005485 1.917641287545 -0.1538859339981 -0.0002241622739519 -0.003107271885463 3.583760257057
LWP = 1e-30 -21.381833467487 32.077998611553 2.4924585571843 -0.0017760562614312 -0.0020608701590214 0.00065809506351528
Tfinal = 365250000
OutputInterval = 36525000
OutputFiles = plan%d.dat
OutputOrder = Time SemiMajorAxis Eccentricity Inclination LongAscendNode ArgPeriapse MeanAnomaly
OutputCoord = Bodycentric
OutputDigits = 16
SaveInterval = 36525000
SaveFiles = save.dat
EnergyInterval = 36525000
EnergyFile = energy.dat
#
Starting integration (Tinitial = 0).
There are 6 particles in the system (5 HWPs, 1 LWP, 0 ZWPs).

SaveFile save.dat written (Time = 0, Steps = 0).
Energy   errors:  0.0000e+00 ave, 0.0000e+00 rms, 0.0000e+00 max
Momentum errors:  0.0000e+00 ave, 0.0000e+00 rms, 0.0000e+00 max
Estimated CPU time required: 10.2 s.

SaveFile save.dat written (Time = 36525000, Steps = 100000).
Energy   errors:  6.4810e-08 ave, 9.1656e-08 rms, 1.2962e-07 max
Momentum errors:  1.3391e-15 ave, 1.8938e-15 rms, 2.6783e-15 max
```

```
CPU     time since start:    1.020 s (1.0200e-05 s per step)
Elapsed time since start:    1.098 s (92.9% CPU ave; 92.9% local)
Approx. time   remaining:    9.883 s (9.883 s local)

...

Integration complete (Tfinal = 365250000, Steps = 1000000).
Energy   errors:  6.0274e-08 ave, 7.5161e-08 rms, 1.2962e-07 max
Momentum errors:  5.3075e-15 ave, 6.8479e-15 rms, 1.4510e-14 max
CPU     time since start:   10.210 s (1.0210e-05 s per step)
Elapsed time since start:   10.686 s (95.5% CPU ave; 0.0% local)
```

You will find 6 files, `plan0.dat .. plan5.dat` representing the sun, and planets Jupiter through Pluto. HNbody comes with a set of supermongo[6] macros especially designed for the ASCII tables that are produced.

2.9 kira

`kira` is the flagship integrator of the *starlab* package. It is also one of the few codes that has stellar evolution integrated with stellar dynamics. `kira` can optionally be compiled with the GRAPE libraries, to run on the GRAPE hardware, which of course speeds up the gravity computations substantially.

```
   examples/ex4
   # generate a Plummer sphere with 256 bodies, again with reproducable seed
 1% makeplummer -n 100 -s 123 > k100.in
  rscale = 0.979095
  com_pos = 0  0  0

   # integrate to T=100, output steps D=10
   # notice the csh method of splitting stdout and stderr
 3% (kira -t 100 -D 0.2 < k100.in > k100.out) >& k100.log

   # convert to NEMO snapshots
 4% dtos k100.dat < k100.out

   # look at the lagrangian radii
 5% snapmradii k100.dat log=t | tabplot - 1 2:10 0 100 -1 1 line=1,1

   # integrate with nbody0, need to convert the data first
 6% dtos k100.indat < k100.in
 7% nbody00 k100.indat k100.outdat eta=0.01 deltat=0.2 tcrit=100
 8% snapmradii k100.outdat log=t | tabplot - 1 2:10 0 100 -1 1 line=1,1

 9% hackforce k100.dat - |\
      snapcenter - - "-phi*phi*phi" |\
      snapmradii - log=t |\
      tabplot - 1 2:10 0 100 -1 1 line=1,1 xlab=Time ylab="log(M(r))"
10% hackforce k100.outdat - |\
      snapcenter - - "-phi*phi*phi" |\
      snapmradii - log=t |\
```

[6] the `sm` program itself is however not publicly available

```
tabplot - 1 2:10 0 100 -1 1 line=1,1 xlab=Time ylab="log(M(r))"
```

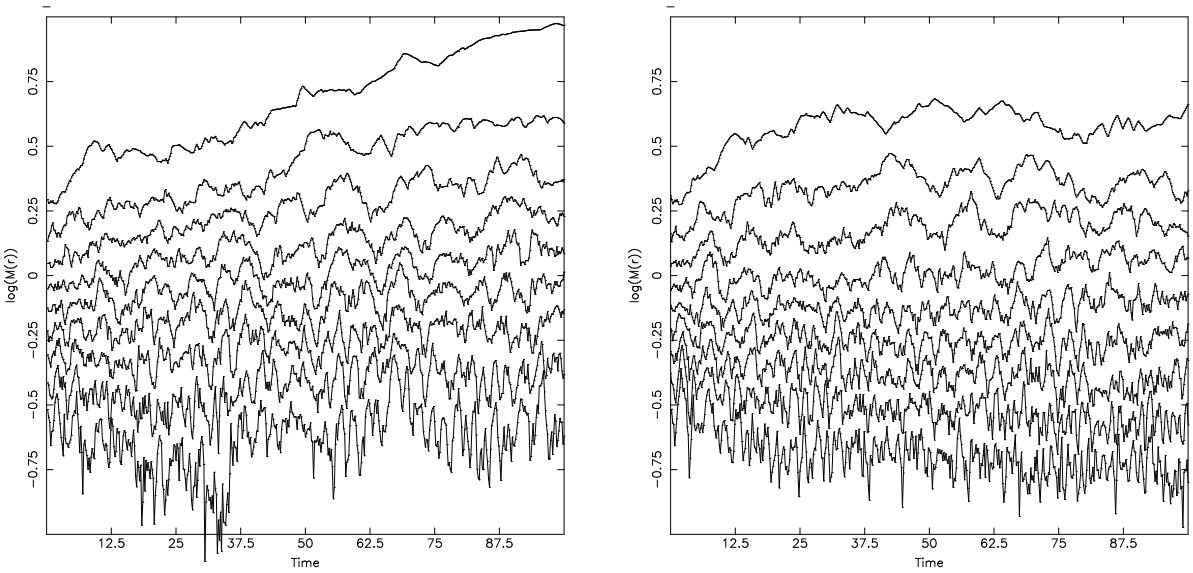

Figure 2.1: Lagrangian radii for `kira` (left, 10%) and `nbody0` (right, 11%). Both have been recentered using the potential $-\Phi^3$.

2.10 treecode

We keep several of the "original" Barnes & Hut (1986) treecodes in NEMO. The code is basically $\mathcal{O}(N \log N)$, which makes it very fast for large N compared to a direct full N-body code. The earliest version also includes the curious self-gravity bug which can occur for large opening angles under peculiar conditions (see Salmon & Warren 1994). The original version around which NEMO was originally developed is still available as `hackcode1`. In this example we'll set up a head-on collision between two Plummer spheres:

```
   examples/ex5
   # create our usual reproducable plummer sphere
1% mkplummer p1 512 seed=123
2% mkplummer p2 512 seed=456

   # stack them , but offset them in phase space on a collision course
3% snapstack p1 p2 p3 deltar=5,0,0 deltav=-1.5,0,0

   # integrate it for a little while
4% hackcode1 p3 p3.out freqout=5 tstop=10

init_xrandom: seed used 456
```

```
       nbody        freq         eps         tol
        1024       32.00      0.0500      1.0000

        options: mass,phase

        tnow       T+U       T/U     nttot     nbavg     ncavg   cputime
       0.000   -0.1236   -0.8953    138503        60        75      0.00

                cm pos   -0.0000    0.0000   -0.0000
                cm vel    0.0000    0.0000   -0.0000
...
        tnow       T+U       T/U     nttot     nbavg     ncavg   cputime
      10.000   -0.1184   -0.8230    111709        42        66      0.37

                cm pos    0.0036    0.0134   -0.0027
                cm vel    0.0007    0.0015   -0.0011

        particle data written

5% snapplot p3.out nxy=3,3 xrange=-5:5 yrange=-5:5 times=0,1,2,3,4,5,6,7,8 nxticks=3 nyticks=3

   # plot up a diagnostics diagram showing center of mass and energy conservation
6% snapdiagplot p3.out
321 diagnostic frames read
Worst fractional energy loss dE/E = (E_t-E_0)/E_0 = 0.0602005 at T = 3.03125

   # show where the center of mass is, notice the out=. (equivalent to /dev/null)
7% snapcenter p3.out  . report=t
-0.000000 0.000000 -0.000000 0.000000 0.000000 -0.000000
-0.000013 0.000002 -0.000012 -0.000073 0.000032 -0.000070
-0.000026 -0.000001 -0.000039 -0.000027 -0.000050 -0.000224
-0.000022 -0.000004 -0.000098 -0.000009 -0.000036 -0.000287
..
0.003339 0.012733 -0.002340 0.000667 0.001810 -0.000922
0.003468 0.013060 -0.002516 0.000708 0.001670 -0.000952
0.003603 0.013355 -0.002703 0.000738 0.001500 -0.001060
```

Note that this code does not conserve linear momentum, since the forces are not symmetric and thus do not exactly cancel. Therefore you will see a drift in the center of mass (CM), as numerically shown by the output of `snapcenter`. For a spherical system this should be a random walk.

In addition to `hackcode1`, there is also `hackcode1_qp`, which is slightly more expensive (but more accurate) and adds quadrupole moment corrections to the force calculations.

2.11 gyrfalcON

Dehnen (2000) introduced a treecode with multipole expansions, which makes the code perform $\mathcal{O}(N)$, and competitive to the Greengard & Rochlin FMM type codes. The code is written in C++ and fully integrated in NEMO, i.e. it reads and writes *snapshot* files and uses the *getparam* user interface. In the next example we'll take the same input condition as before, the colliding Plummer spheres:

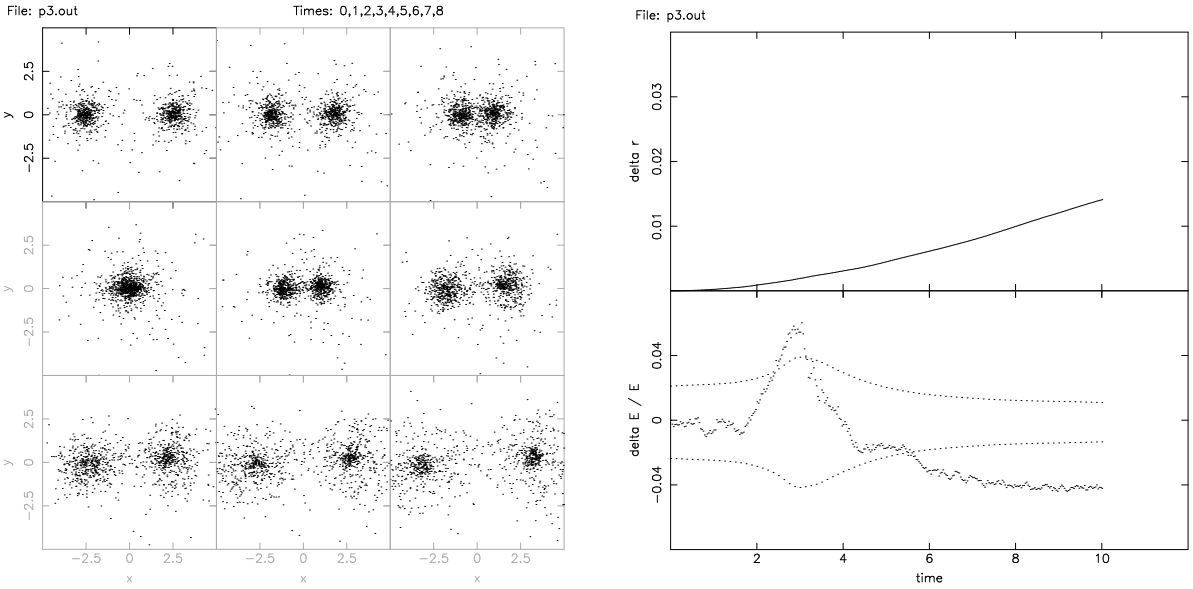

Figure 2.2: Collision between two Plummer spheres (`hackcode1`) and the diagnostics overview (`snapdiagplot`)

```
# ------------------------------------------------------------------------------------------------------------------------
# "gyrfalcON p3 p3.outg tstop=10 step=0.2 hmin=5 VERSION=3.0.5I"
#
# run at  Thu Jul 13 00:17:57
#     by  "teuben"
#     on  "nemo"
#     pid  5815
#
#    time        E=T+V        T          V_in        W         -2T/W     |L|       |v_cm|  tree  grav  step  accumulated
# ------------------------------------------------------------------------------------------------------------------------
 0.0000     -0.1317041    1.0572     -1.1889     -1.1855     1.7835  0.031519   4.7e-10  0.00  0.01  0.02   0:00:00.02
 0.031250   -0.1316010    1.0586     -1.1902     -1.1867     1.7840  0.031522   1.2e-09  0.00  0.01  0.01   0:00:00.04
 0.062500   -0.1315449    1.0600     -1.1916     -1.1883     1.7841  0.031525   9.2e-10  0.00  0.00  0.00   0:00:00.05
 0.093750   -0.1314929    1.0618     -1.1933     -1.1893     1.7857  0.031526   7.6e-10  0.00  0.01  0.01   0:00:00.07
. . .
 9.8750     -0.1310892    0.52518    -0.65627    -0.65459    1.6046  0.031257   1.9e-08  0.00  0.00  0.00   0:00:04.78
 9.9062     -0.1310836    0.52443    -0.65551    -0.65404    1.6037  0.031250   1.9e-08  0.00  0.01  0.01   0:00:04.80
 9.9375     -0.1310790    0.52393    -0.65501    -0.65334    1.6039  0.031241   1.9e-08  0.00  0.00  0.00   0:00:04.81
 9.9688     -0.1310589    0.52360    -0.65466    -0.65291    1.6039  0.031234   1.9e-08  0.00  0.01  0.01   0:00:04.83
 10.000     -0.1311007    0.52330    -0.65440    -0.65275    1.6034  0.031228   1.9e-08  0.00  0.00  0.00   0:00:04.84
```

As you can see, Dehnen's code is quite a lot faster. As for accuracy though, we leave this as an exercise for the interested reader. Notice this code balances all forces and thus conserves total linear momentum unlike the BH86 treecode.

2.12 superbox

Fellhauer et al. (2000) wrote a particle-mesh code with high resolution sub-grids and an NGP (nearest grid point) force-calculation scheme based on the second derivatives of the potential. It is also an example of a code where initial conditions benefit from knowing the details of the potential calculation routines. In particular, if the initial conditions are known to be representing a stable configuration (e.g. a Plummer sphere). See the program `Setup/plummer.f` in the `Superbox` code distribution.

2.13 gadget

Springel (2000) made `gadget` publicly available. It is primarily a cosmology code, and based on the treecode, but also incoorporates SPH particles and, for cosmological simulations, the Ewald sum technique to create periodic boundary conditions. The code was originally written in Fortran, but in 2005 re-released and completely rewritten in C. It runs in a parallelized mode under MPI, even if you have a single processor, you will need the infrastructure to use MPI (e.g. mpich2 and openmpi) Also, `gadget2` will typicall be compiled for a specific physical situation. In the standard distribution you will find 4 different ones: *cluster*, *galaxy*, *gassphere* and *lcmd_gas*.

```
1%  mkdir galaxy

2%  ln -s $MANYBODY/gadget/Gadget2/Gadget2/parameterfiles

3%  ln -s $MANYBODY/gadget/Gadget2/ICs

4%  lamboot

5%  mpirun -np 2 Gadget2.galaxy parameterfiles/galaxy.param

6%  lamwipe
```

WARNING: this benchmark example (60,000 particles in a galaxy-galaxy collision) from the Gadget2 source code takes a little over 2000 steps, and can easily take an hour on a single 3 GHz Pentium-4 class CPU.

To use a dataset from NEMO, the data will have to be transformed using the `snapgadget` and `gadgetsnap` pair of programs:

```
mkdir galaxy
ln -s $MANYBODY/gadget/Gadget2/Gadget2/parameterfiles
ln -s $MANYBODY/gadget/Gadget2/ICs

mkplummer p10k 10000
snapgadget p10k galaxy.dat 10000,0,0,0,0
snap

lamboot
Gadget2.galaxy parameterfiles/galaxy.param
mpirun -np 2 Gadget2.galaxy parameterfiles/galaxy.param
```

```
lamwipe

gadgetsnap ...
```

Note that in most circumstances you will want to check the units of mass, length and velocity with those listed in the param file, as Gadget expects this.

The versions of Gadget that are pre-compiled are:

- Gadget2.cluster
- Gadget2.galaxy
- Gadget2.gassphere
- Gadget2.lcdm_gas

2.14 quadcode

An example of an $\mathcal{O}(N)$ code where the potential and forces are derived from a potential expansion in spherical harmonics (White 1983, who dubbed it multipole expansion). See also Hernquist & Barnes, (1990). Fully integrated in NEMO, and very fast. Excellent to study the dynamics of an isolated galaxy, stability, check orbit theory etc.

```
# File: examples/ex7
1% mkplummer p1024.in 1024 seed=1024

2% quadcode p1024.in p1024.out4 tstop=10

       nbody        freq       eps_r       eps_t        mode       tstop
        1024     64.0000      0.0500      0.0700           3       10.00


         tnow         T+U         T/U     cputime
        0.000   -0.247060     -0.4955        0.00

               cm pos    0.0000   -0.0000    0.0000
               cm vel    0.0000   -0.0000   -0.0000

...

         tnow         T+U         T/U     cputime
       10.000   -0.247041     -0.4872        0.04

               cm pos    0.0208   -0.0033   -0.0009
               cm vel    0.0067    0.0007   -0.0004

       particle data written
```

```
#  takes about 2.9" on my laptop

3% snapdiagplot p1024.out4
321 diagnostic frames read
Worst fractional energy loss dE/E = (E_t-E_0)/E_0 = -0.000674741 at T = 6.03125

4% snapmradii p1024.out4 | tabplot - 1 2:10 line=1,1
```

2.15 scfm

Hernquist & Ostriker (1992) deviced a code termed the *self-consistent field (SCF) method*, implemented as `scfm` by Hernquist, with a NEMO wrapper program called `runscfm`.

```
 # File: examples/ex8
 2% runscfm `pwd`/p1024.in p1024.out2 dtime=1.0/64.0 nsteps=640 noutbod=16
### Warning [runscfm]: Resolving partially matched keyword dt= into dtime=
snapprint /tmp/p1024.in m,x,y,z,vx,vy,vz header=t > SCFBI
m x y z vx vy vz
[reading 1024 bodies at time 0.000000]
[reading 1024 bodies at time 0.250000]
..
[reading 1024 bodies at time 9.750000]
[reading 1024 bodies at time 10.000000]

    # a quick overview of the evolution
 3% snapplot p1024.out2/scfm.dat nxy=4,4

    # lagrangian radii, check how much the cluster remains in shape
 4% snapmradii p1024.out2/scfm.dat log=t | tabplot - 1 2:10 line=1,1
```

This will have created a series of `SNAPnnn` files in the `p1024.out2` directory, that have been converted to a file `scfm.dat` in the run directory.

Note this program does not use dynamic memory, so the program needs to be recompiled for more than the default number of particles (10,000 in the current code, see `src/nbody/evolve/scfm/scf.h`

2.16 CGS

The *Collisionless Galactic Simulator* (CGS) N-body code is a spherical harmonics Particle-Mesh code. The current version is a largely expanded and improved version from van Albada by Trenti (Trenti 2005). A convenient NEMO wrapper, called `runCGS` is available:

```
 # File: examples/ex9 :
 1% runCGS in=p1024.in out=run2
>>> snapprint p1024.in x,y,z    > run2/initPOS.dat
x y z
>>> snapprint p1024.in vx,vy,vz > run2/initVEL.dat
vx vy vz
line=1024
Using snapshot in=p1024.in with nbody=1024
>>> CGS.exe >& CGS.log
>>> tabtos fort.90 snap.out nbody,time skip,pos,vel,acc,phi; rm fort.90
[reading 1024 bodies at time 0.000000]
[reading 1024 bodies at time 0.100000]
...
[reading 1024 bodies at time 3.900000]
[reading 1024 bodies at time 4.000000]

    # a quick overview of the evolution
 2% snapplot run2/snap.out nxy=4,4

    # lagrangian radii, but note masses were not present and had to be added
 4% snapmass run2/snap.out - mass=1 | snapmradii - log=t | tabplot - 1 2:10 line=1,1
```

2.17 galaxy

Sellwood (1997) contributed this 3-dimensional cartesian code to NEMO. Is uses the Fourier Analysis/Cyclic Redundancy (FACR) method, pioneered by Hockney (1965, 1970) and James (1977) for galactic dynamics. Running the program itself has some of the usual fortran restrictions, and a NEMO frontend `rungalaxy` is available to simplify running this code:

```
1% mkplummer p1024.in 1024
2% rungalaxy p1024.in run1
 galaxy V1.3
 Maximum number of particles (mbuff):  100000
 Maximum gridsize:  33 33 33
 Greens function creation complete
 1024 8 8192 80000
 305 particles outside the grid at the start
 Run (re)started at time  0.
   and will stop after the step at time   1.05000007
 Time-centering velocities
 Starting step 0
 Starting step 1
...

 Starting step 20
 Un-centering velocities
Time=0
Time=0.5
Time=1
3% snapcenter run1/galaxy.snap . report=t
-0.000846 0.005885 -0.023592 0.003350 -0.004093 0.003552
```

```
-0.006440 -0.002968 -0.026719 0.003096 -0.005531 0.000260
0.009361 -0.004989 -0.012464 0.013971 -0.005660 0.003792
```

2.17.1 AMUSE

Integrators in AMUSE are to be described here.

Table 2.2: Common keywords to N-body integrators differ

code	*time-step*	*output-time-step*	*diag-output*	*stop-time*	*start-time*
`firstn`					
`nbody0`					
`nbodyX` (SJA)	[eta]	deltat	-	tcrit	
`fewbody`					
`hnbody`					
`kira`					
`treecode` v2.2 LH	dtime	nout	-	nsteps	
`treecode1` v1.4 JEB	dtime	dtout	-	tstop	
`gyrfalcON`	[hmin]	step	logstep	tstop	-
`gadget`					
`scfm`	dtime	noutbod	noutlog	nsteps	-
`hackcode1`	freq	freqout	minor_freqout	tstop	
`quadcode`	freq	freqout	minor_freqout	tstop	
`CGS`	dt,dtmin,dtmax	freqout	freqdiag	tstop	
`galaxy`	dy	dtout	dtlog	tstop	
`superbox`					
`partree`					

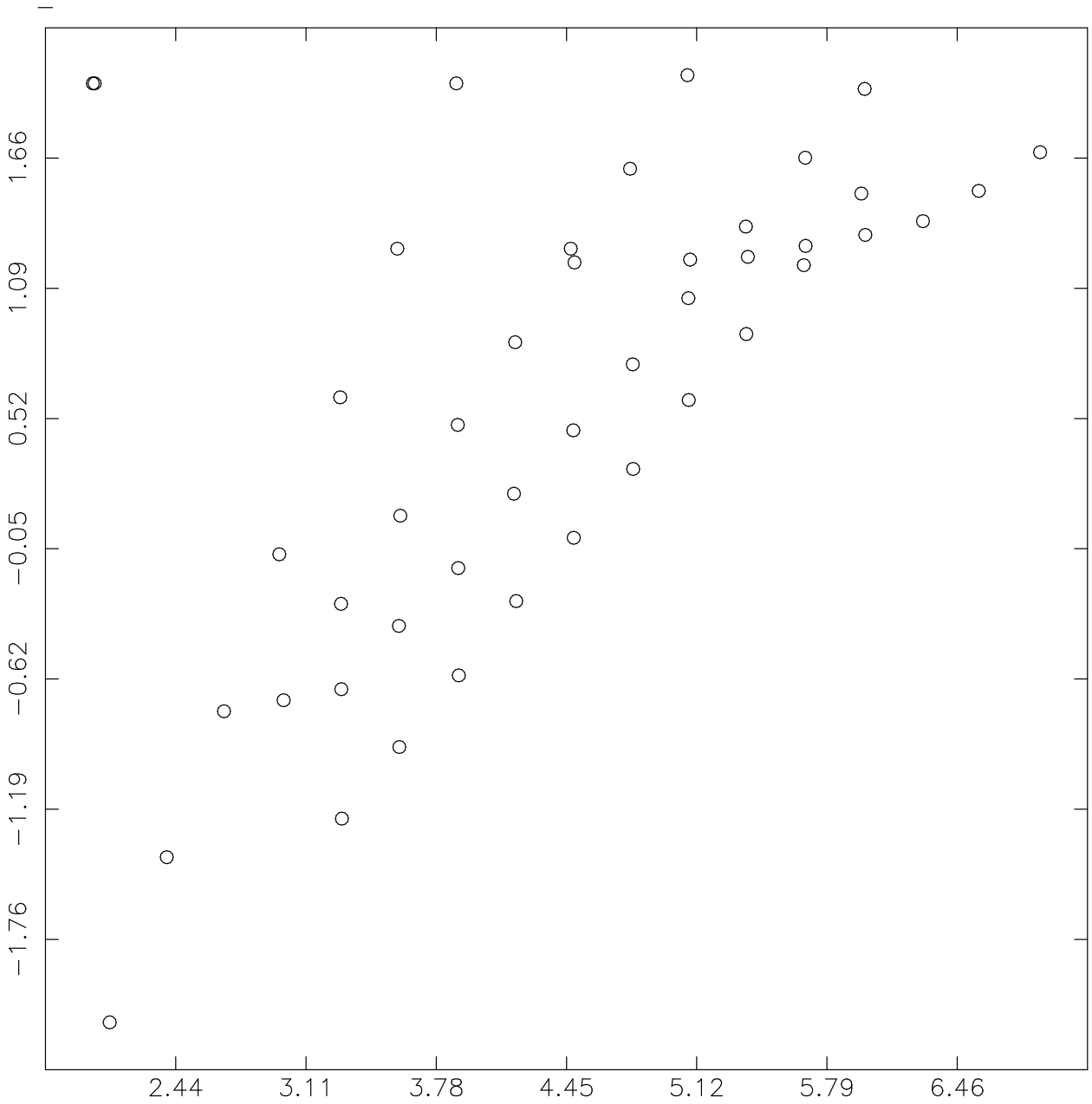

Figure 2.3: Code performances - On the horizontal axis log(N), vertically log(CPU-sec/particle/step). These are the values for a P4/1.6GHz laptop. Seen in this plot are nbody0, galaxy, gyrfalcON, hackcode1 and xxx

Chapter 3

Code Testing

In the field of hydrodynamics rigorous testing using standard test cases is quite commonplace. However, in the field of stellar dynamics this is not the case, largely due to the chaotic behavior of the N-body problem (though most papers that introduce a new N-body code of course discuss their accuracy and performance). Nonetheless, a number of efforts have been undertaken to compare codes, performance and accuracy. We mention a few :

3.1 Lecar: 25-body problem - 1968

This is one of the oldest benchmark in N-body dynamics and also a very difficult one. 11 numerical integrations of a particular 25-body problem were assembled and presented by Myron Lecar (1968). This was essentially a cold collapse model, with all equal-mass particles initially at rest, and in fact the positions of the 25th body was left as a (necessary) exercise for the reader! Data are available as `$NEMODAT/iau25.dat`, though we cheated and added the 25th (as the last) body!

3.2 Pythagoran Problem

A really fun setup which has been used in a number of studies (Burrau in 1913 was the first to report on this, but unlike the real outcome conjectured it would be a periodic solution of the 3-body problem) is 3 particles on a Pythagoran Triangle with sides 3-4-5 and relative masses 3:4:5 (Szebehely & Peters, 1967). See Section G.1 for more details and the coverpage for a sample integration.

3.3 Aarseth, Henon & Wielen - 1974

Aarseth, Henon & Wielen (1974) compared the dynamical evolution of a Plummer sphere as integrated with an N-body integrator, the Monte Carlo method and the fluid dynamical approach. This paper is also well known for its recipe how to create an N-body realization of a Plummer sphere. You can find examples of this in quite a few places, e.g.
`$NEMO/src/nbod4/init/mkplummer.c`
`$STARLAB/src/node/dyn/init/makeplummer.C`
`$NEMO/src/nbody/evolve/aarseth/nbody1/source/data.f`
`$NEMO/usr/aarseth/firstn/data.f` `$NEMO/usr/Superbox/Setup/plummer.f`

cf. long term comparisons Makino et al. did on gravothermal oscillations with Grape

3.4 Kang et al. - Cosmological Hydrodynamics - 1994

Kang *et al.* compared the simulation results of five of cosmological hydrodynamic codes from $z = 20$ to $z = 0$. 5 codes were used: three Eulerian mesh, and two variants of the Lagrangian SPH method. Codes: TVD (Ryue et al 1993), PPM (Bryan et al 1993), COJ (Cen 1992), TSPH (Hernquist & Katz, 1989), PSPH (Evrard 1988).

3.5 Frenk et al. - 1999

Frenk *et al.* took 12 cosmological codes and compared the formation of a rich cluster of galaxies. Again using a number of SPH and Eulerian (various styles) grid simulations were used. These simulations became known as the ‘‘`Santa Barbara Cluster`’’ benchmark.

3.6 Heitmann et al. - 2005

Comparing cosmological N-body simulations. Codes: MC2 (Habib et al 2004), FLASH (Fryxell et al 2000), HOT (Warren & Salmon 1993) GADGET (Springel et al 2001), HYDRA (Couchman et al 1995), TPM (Xu 1995, Bode et al 2000).

Apart from the Frenk et al. *Santa Barbara cluster* comparison, they also compared two other simulations:

The initial conditions, as well as some results at z=0, are available on their website [1].

Power et al - 2003 - on errors.

[1] `http://t8web.lanl.gov/people/heitmann/arxiv/`

3.7 Hozumi & Hernquist - 1995

Testing SCF method on either homogenous spheres or Plummer models, with given initial virial ratio. This also includes the cold collapse. In particular, they studied the resulting shapes.

3.8 Sellwood: Galaxy Dynamics - 1997

Sellwood (1997), in *Galaxy Dynamics by N-body Simulation*, compared 5 different codes and argued that certain codes... Comparing direct N-body, tree and grid codes. Plummer sphere. *do what? stability?*

3.9 Heggie: Kyoto I - 1997

This starcluster benchmark was organized by Heggie, and presented at the IAU GA in Kyoto in August 1997. 10 results have been compiled.

3.10 Heggie: Kyoto II - 2001

This followup starcluster benchmark was presented at IAU Symposium 208 (Tokyo) on 11 July, 2001. 13 results have been compiled to this moment.

3.11 Notes

```
Also:

 - optimal smoothing papers
      Merritt
      Sellwood
      Dehnen
      Zhan      astro-ph/0507237

      Athanassoula?

 - error stuff

      Power at al 2003

 - relaxation:
```

```
      Hernquist & Barnes (1990) :
      Are some N-body algorithms intrinsically less collisional than others?

 - forwards and backwards integration.
      van Albada  & van Gorkom - (1977)

 - various collisional codes compared: MC, nbody6, kira, grid codes etc.
   under various circumstances:   See some SPZ papers by:

   Takahashi et al. (2002)
   Spinnato et al. (2003)
   Joshi et al. (2000)
   Portegies Zwart et al. (2004)
   Gualandris et al. (2004)

- NAM = Numerical Action Method. We'll have code from Shaya. Find solution
  with NAM and integrate it back using e.g. Aarseth and see where they
  come out.
  e.g.  http://arxiv.org/abs/astro-ph/0512405
```

Chapter 4

Data Conversion

With this large number of integrators it is unavoidable that dataformats will be different here. We list a few related to the beforementioned integrations. Note that if your format is not listed here, one of these is usually close enough to be modified to work quickly.

Table 4.1: N-body data interchange programs in NEMO

format	*origin*	*to-NEMO*	*from-NEMO*	*comments*
`snapshot`	NEMO	-	-	
`zeno`	ZENO	zenosnap	-	csf may also work
`dyn`	starlab	dtos	stod	
`tdyn`	starlab	-	-	use
``205''	treecode	atos,atos_sph	stoa, stoa_sph	Hernquist
`tipsy`	tipsy	tipsysnap	snaptipsy	be aware of ascii/binary
`gadget`	gadget	gadgetsnap	snapgadget	
`u3`	nbody1,2	u3tos	-	be aware of endianism
`martel`		martelsnap	-	Martel
`heller`		hellersnap	-	Heller/Shlosman
`rv`		rvsnap	snaprv	Carlberg
`rvc`		rvcsnap		Couchman
`xvp`		xvpsnap		Quinn/Balcells
ascii-tables	-	tabtos	snapprint	versatile converter
idl	-	-	snapidl	binary IDL format
3dv	-	-	snapidl	for a variety of 3D viewers
xyz	NEMO	-	snapxyz	for xyzview
specks	partiview	-	snapspecks	for partiview
...	partree	-	-	-

4.1 Examples

Some examples will follow here... See also the next chapter for some examples.

Chapter 5

Projects

5.1 Units

We need a short session on units. See also NEMO's *units(5NEMO)* manual page. Not just the Heggie & Matthieu (1986) virial units, but also how to scale them to real objects, like clusters and galaxies. Discuss stellar evolution in kira and how this has introduced a hybrid simulation where the scale cannot be changed!

Aarseth uses *N-body Units*, i.e.

`AMUSE` attaches units to numbers, e.g. `m = 100.0 | units.MSun`, `r = 1 | units.parsec`.

5.2 Initial Conditions

Although some initial conditions can be generated from basic principles, others need the integrator for force calculations to place the initial conditions in a specific state consistent with the chosen integrator that uses that force. The position of the 25th body in the IAU 25-body problem is a simple example of where an on-the-fly calculation is needed to guarentee the center of mass is numerically at (0,0,0).

Also important is the issue of quiet starts, as extensively discussed by Sellwood (1987, 1997). Suppression of shot noise. See also the new wavelet based code by Romeo et al. 2003, where the equivalent of a factor of 100 speedup is claimed.

White (1996) suggested a novel way to initialize cosmological simulations, referred to as "glass" initial conditions. They can be obtained when a Poisson sample in an expanding periodic box is evolved with the sign of gravity reversed until residual forces have dropped to a negligible value. Gadget-2 supports this as an option.

5.3 Relaxation

Can we do some interesting experiments in 2D where $\tau_{relax} \approx \tau_{crossing}$?

Also, for a Plummer sphere: show that 2T/W != 1, and that Clausius must be used to study the equilibrium of the system. How much does it 'relax' for given softening? Does it depend on N ?

5.4 Plummer Sphere

Explain sampling a DF, use rejection technique, and using the code show how to make a Plummer sphere. See Aarseth, Henon & Wielen (1974).

5.5 Galaxy Collisions

Where is the dark matter in the S+S remnants sometimes called Ellipticals? (cf. Baes' papers about dark matter in E's). How does the shape of tidal tails depend on the extent of the dark matter.

Also, this is an ideal scenario to study systematic effects of errors in the treecode before and after the overlap. Check conservation of energy and properties of the resulting object(s).

5.6 Galactic Discs

```
- Galactic Disks:
  - mkexpdisk/mkexphot      -
           old NEMO program, nice as toy, but no halo or bulge
           can try and slowly/adiabatically add one. show how bad it is

  - galmake: Hernquist 1993      - has some pitfalls. slow. bad for large N
  - mkkd95: Kuijken & Dubinski
  - MaGalie (Boily et al 2001)
  - Dehnen's ``Galpot''


  Simulations:
     - bar formation - properties, dependance on ICs, random, Q4
     - heating of the disk
     - interactions:
       - M31/MW
       - M51
       - Antenna

  Subsampling: how to generate good initial conditions?
```

5.7 Cold Collapse

Cold Collapse (cf. v Albada 1981, McGlynn?, but also Lecar's 25-body problem). Discussed in many papers, e.g. Hozumi & Hernquist , Aarseth, Popaloizou, Lin; Bertin & Stiavelli; Boily; Theis. Effects of softening on shape.

Cold collapse calculations can be done from any spherical particle distribution by setting the velocities to 0. `snapscale vscale=0.0`, or `snapvirial` can be used to scale to a preferred ratio of $|2T/W|$. Programs like `mkhomsph` have a direct parameter that controls the initial virial ratio, so no external scaling is needed.

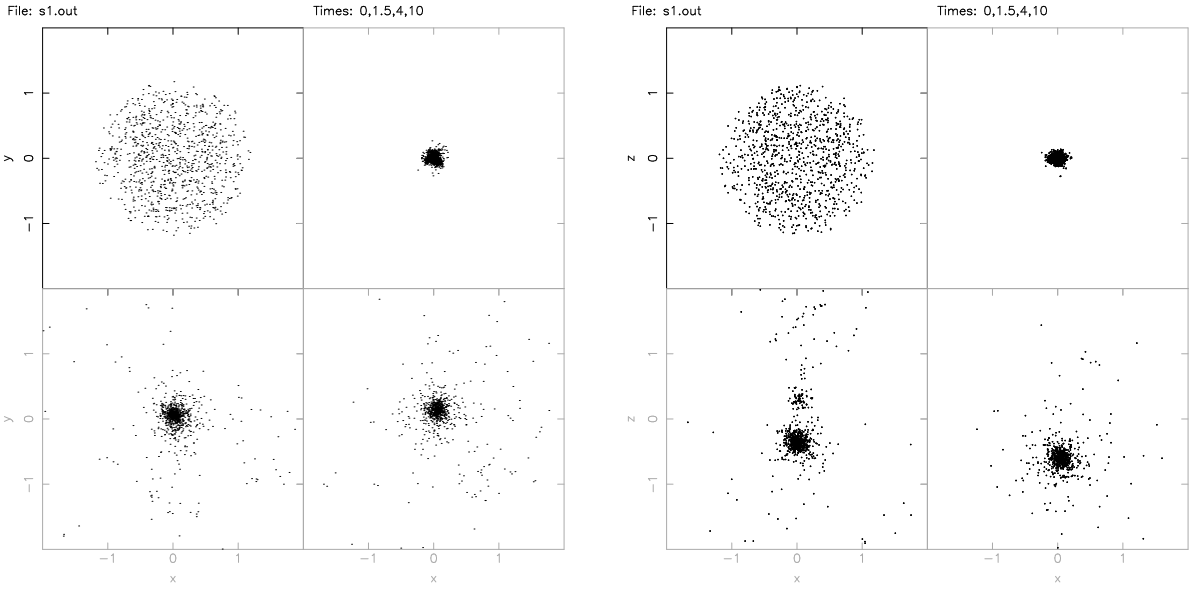

Figure 5.1: Positions of an N=1000 cold collapse at T=0, 1.5 (roughly the first collapse), 4.0 and 10.0 in an XY projection (left) and an XZ projection (right). - try also snapplot3

```
1% mkhomsph s1 1000 rmin=0 rmax=1.2 2t/w=0
2% gyrfalcON s1 s1.out tstop=10 step=0.05 give=m,x,v,p

3% snapplot s1.out times=0,1.5,4,10 nxy=2,2 nxticks=3 nyticks=3 yvar=y yapp=s1xy.ps/vps
4% snapplot s1.out times=0,1.5,4,10 nxy=2,2 nxticks=3 nyticks=3 yvar=z yapp=s1xz.ps/vps
```

As can be seen in the two projections in Figure 5.1, the density center of the collapsed object does not remain at the origin, despite that a perfect momentum-conserving integrator has been used. Why is that?

To study the structure of the resulting object, we will translate the center of mass to the density center. Since we have instructed the integrator to also store the potential (already available to the code during the

integration) in the output data stream, which can now be used to weigh the particles in `snapcenter` with an appropriate factor (why Φ^3 and not Φ^2 or Φ?). See also Cruz et al. (2002).

```
snapmradii s1.out 0.01,0.1:0.9:0.1,0.99 > s1a.mtab
snapcenter s1.out - "-phi*phi*phi" | snapmradii - 0.01,0.1:0.9:0.1,0.99 > s1b.mtab

snapmradii s1.out 0.01,0.1:0.9:0.1,0.99 log=t > s1a.mtab
snapcenter s1.out - "-phi*phi*phi" | snapmradii - 0.01,0.1:0.9:0.1,0.99 log=t > s1b.mtab


tabplot s1a.mtab 1 2:12 0 10 -2 2 nxticks=9 line=1,1 color=2,3,3,3,3,2,3,3,3,3,2
tabplot s1b.mtab 1 2:12 0 10 -2 2 nxticks=9 line=1,1 color=2,3,3,3,3,2,3,3,3,3,2
        xlab=Time "ylab=log(M(r))"

1000: 10?
4000: 37"
16000: 163"

    an interesting observation of this kind of cold collapse is that
    theoretically (N -> \infty) the density would rise arbitrarely high
    at the collapse (the ``big crunch''), thus at some point the
    assumption of a collisionless simulation are violated.
    Not to mention that softening is then not treated correctly,
    since in general Nbody simulations with softening do not take
    the overlap potential into account.

    test this by using a known 2body problem;
    check with the work of Aladin in the 70s or 80s ?
```

5.8 Models for a galactic disk

Kuijken & Dubinksi (1995) came up with a novel way to make reasonably self-consistent disk-bulge-halo models. You can find their code in `$NEMO/usr/kuijken/GalactICS-exp`, the binaries have been placed in `$NEMOBIN`, as well as a wrapper program `mkkd95` is available to simplify creating such galaxies in the NEMO style. In addition, `mkkd95` is optimized to create multiple random realizations.

```
GalactICS/Milky_Way/A>

make galaxy                                     # mergerv disk bulge halo > galaxy
             this will take a while to compute

    D     B    H
A  8000  4000  6000     13sec
B  1000  1000  1000     48sec
C  4000  2000 10000     48sec
D  1000  1000  1000     98sec
```

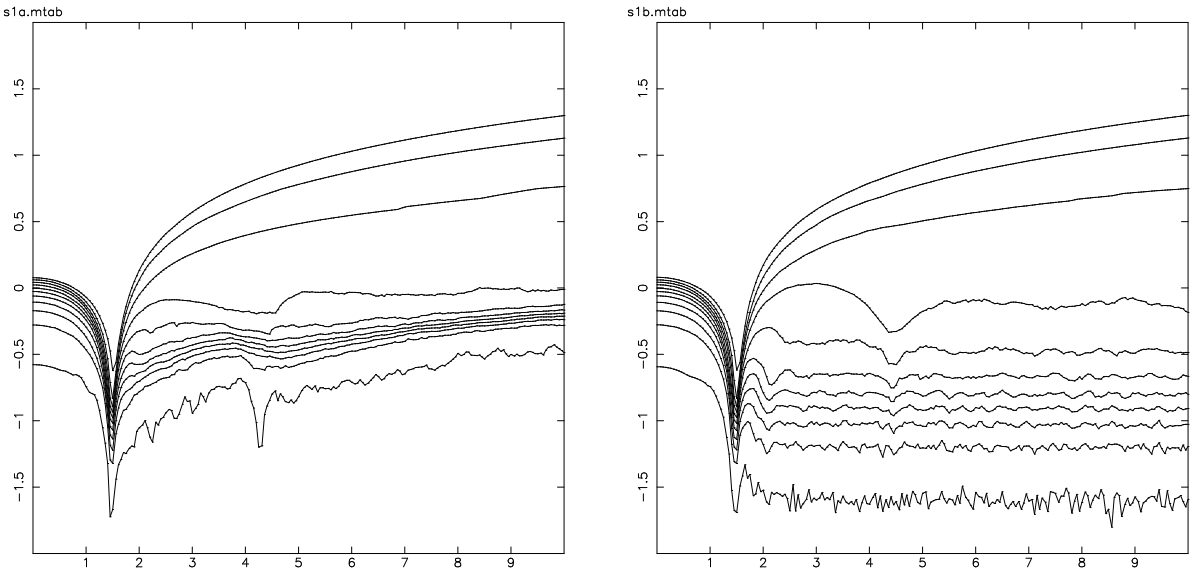

Figure 5.2: Two Lagrangian Radii calculations: one without centering the snapshot (left), and one with (right). Notice that the vertical axis is logarithmic. The mass-radii are plotted at 1,10,20,....90,99% of the mass. Note that between 20% and 30% of the mass is lost in this simulation.

```
tabtos galaxy A0.snap nbody,time m,pos,vel

These are DBH (Disk-Bulge-Halo, in that order in a snapshot) models

snapmstat A0.dat sort=f
0 0:7999  = 8000 Mass= 0.000108822 TotMas= 0.87058 CumMas= 0.87058
1 8000:11999  = 4000 Mass= 0.00010631 TotMas= 0.425242 CumMas= 1.29582
2 12000:17999 = 6000 Mass= 0.000819365 TotMas= 4.91619 CumMas= 6.21201


0=disk   8000
1=bulge  4000
2=halo   6000


snapxyz A0.dat - | xyzview - nfast=18000 maxpoint=18000 scale=16

snapplot A0.dat color='i<8000?2.0/16.0:(i<12000?3.0/16.0:4.0/16.0)'
    colors in yapp are 0..1 and since we use PGPLOT, we only  have
    16 colors.   0/16=b  1/16=white 2=red 3=green 4=blue
e.g. to see the colors

snapplot A0.dat color=x

snapxyz A0.dat -

snapxyz A0.dat - color='i<8000?1:(i<12000?2:4)' | xyzview - maxpoint=18000 nfast=18000 scale=8 fullscreen=t
```

```
snapprint A0.dat  x | tabhist -
minmax about 5
snapprint A0.dat  vx | tabhist -
minmax about 1

thus T=2.pi.R/v = 6*5/1=30 !!!

thus freqout=1/30 for an orbit
thus freq=1/1500

gyrfalcon:   2^hmin = 1500   ::   hmin=10..11  !!


gyrfalcON A0.dat A1.dat tstop=1/30 step=1/300 hmin=10


 0          -2.47565      0.49177 1.2903     7.2e-09  0.02   0.25       0.27      0.27
 0.00097656 -2.475653     0.49177 1.2903     7.4e-09  0.01   0.28       0.29      0.57
 0.0019531  -2.475656     0.49177 1.2903     7.4e-09  0.03   0.25       0.28      0.85
....

 0.033203   -2.475671     0.4919  1.2903     5.1e-09  0.02   0.26       0.28      9.93
 0.03418    -2.475674     0.4919  1.2903     4.9e-09  0.02   0.26       0.28     10.21

snapdiagplot A1.dat
11 diagnostic frames read
### Warning [snapdiagplot]: Autoscaling time. MinMax=-0.00170898 0.0358887
Worst fractional energy loss dE/E = (E_t-E_0)/E_0 = 9.63054e-06 at T = 0.0341797

time gyrfalcON A0.dat . tstop=1 step=1/30 hmin=10
 0          -2.47565      0.49177 1.2903     7.2e-09  0.02   0.26       0.28      0.28
 0.00097656 -2.475653     0.49177 1.2903     7.4e-09  0.03   0.25       0.28      0.58
...
 0.99902    -2.475605     0.49946 1.2903     1.5e-08  0.02   0.27       0.29   298.529
 1          -2.475602     0.49945 1.2903     1.5e-08  0.02   0.25       0.28   298.809


290.800u 8.070s 6:53.62 72.2%   0+0k 0+0io 376pf+0w

snapdiagplot A1.dat
31 diagnostic frames read
### Warning [snapdiagplot]: Autoscaling time. MinMax=-0.05 1.05
Worst fractional energy loss dE/E = (E_t-E_0)/E_0 = 7.54072e-05 at T = 0.5



---

gyrfalcON A0.dat A2.dat tstop=1 step=1/100 hmin=9

 0          -2.47565      0.49177 1.2903     7.2e-09  0.02   0.25       0.27      0.27
 0.0019531  -2.475656     0.49177 1.2903     7.1e-09  0.02   0.21       0.23      0.51
```

```
snapdiagplot A2.dat
101 diagnostic frames read
### Warning [snapdiagplot]: Autoscaling time. MinMax=-0.05 1.05
Worst fractional energy loss dE/E = (E_t-E_0)/E_0 = 8.70601e-05 at T = 0.521484

gyrfalcON A0.dat A3.dat tstop=10 step=1/10 hmin=8
 0          -2.47565      0.49177 1.2903     7.2e-09  0.02   0.26       0.28      0.28
 0.0039062  -2.475652     0.49177 1.2903     7.4e-09  0.02   0.25       0.28      0.56
...
  9.9961     -2.475596     0.49744 1.2903     1.9e-08  0.02   0.26       0.28   744.695
 10         -2.475598     0.49744 1.2903     1.8e-08  0.02   0.26       0.28   744.975
725.720u 19.340s 17:38.84 70.3% 0+0k 0+0io 376pf+0w


snapdiagplot A3.dat
101 diagnostic frames read
Worst fractional energy loss dE/E = (E_t-E_0)/E_0 = 8.21485e-05 at T = 0.5
```

5.8.1 Potential

```
time mkkd95 junk1 40000 80000 60000                          125\"
gyrfalcON junk1 junk1.pot give=m,x,v,p tstop=0                 3.5\"


foreach n (32 64 128 256 512 1024)
  snapgrid junk1.pot junk1-$n.ccd xrange=-10:10 yrange=-10:10 yvar=z zvar=t zrange=-0.1:0.1
        mean=t evar=phi nx=$n ny=$n
end
```

5.9 Detonating Galaxies: testing the treecode

```
mkplummer pp.in 15000
mkplummer ps.in 4000
snapscale ps.in ps1.in mscale=0.2 rscale=0.2
hackforce ps1.in ps2.in
snapvirial ps2.in ps3.in rscale=f virial=1
gyrfalcON ps3.in ps3.out tstop=10 step=0.2
snapmradii ps3.out | tabplot - 1 2:10

snapshift pp.in pp1.in 0.3,0.3,0.3
snapshift ps3.in ps4.in 7,7,7 -0.25,-0.25,-0.25

snapadd pp1.in,ps4.in  detonate.in
```

5.10 Accuracy of a code

Use the van Albada and van Gorkum technique to integrate a code forwards, storing many snapshots. For each snapshot you then integrate backwards to t=0 (some code allow $dt < 0$, others needs to be rescaled v = -v) and comparing the two snapshots with snapcmp. Plot the results as function of T. What kind of bahavior do we see? Liapunov? Which codes are "good", which are bad, and can we understand this?

5.11 Collisionless?

Athanassoula - papers on importance of resonances, live bars. What is a good N to do galactic dynamics. Martin Weinberg now claims $10^7 - 10^8$. Also check the approach Romeo et al. (2003) took using wavelets.

5.12 Comparing

Take two Nbody codes and compare their potential. For example, use Sellwood's *galaxy* PM/FFT code and Dehnen's `gyrfalcON` TreeCode and Trenti's (GCS) SFP expansion code, and compare the potential for a homogenous sphere of different degrees of flattening. Another interesting parametrization is that of two spheres at different separations (collisions of galaxies). See also Figure 2.2.

Chapter 6

Visualization

Some brief but wise words about packages we all have some experience with. Obviously its a big ugly world out there, and there is a lot more under the sun than this.

6.1 NEMO

6.1.1 snap3dv

The `snap3dv` program in NEMO converts data to a variety of popular 3D viewers, but in addition special converters are available for `partiview` (`snapspecks`) and `xyzview` (`snapxyz`). Even simple programs like `snapprint` can work very effectively.

6.1.2 xyzview

Comes with NEMO, needs the VOGL (Very Ordinary GL) library, which emulates GL on classic X with no acceleration. Can only plot colored dots, no finite size points. Poor for presentations. (manual) animations, can also show the orbit for one selected star from the simulation.

```
   # standard model-A from Kuijken&Dubinski with 8000,4000,6000 in disk-bulge-halo
1% mkkd95 a0.dat

   # display them in Red/Green/Blue using snapplot
2% snapplot a0.dat color='i<8000?2.0/16.0:(i<12000?3.0/16.0:4.0/16.0)' yvar=z xrange=-8:8 yrange=-8:8

   # display them in Red/Green/Blue using xyzview
3% snapxyz a0.dat - color='i<8000?1:(i<12000?2:4)' | xyzview - maxpoint=18000 nfast=18000 scale=8
```

need an orbit example here as well

6.1.3 glnemo2

A recent addition from the Marseille group, `glnemo` is a versatile interactive 3D viewer that can also display SPH/gas particles. Examples dataset are in `$NEMO/src/nbody/glnemo/snapshot`. Recent updates to `glnemo` included making animations. A completely rewritten version, `glnemo2` is now the default version, and the previous version is deprecated.

6.2 Starlab

6.2.1 xstarplot

Starlab's visualization program is called `xstarplot`. More to say on this.

```
   # create a plummer
1% makeplummer ...

   # integrate it
2% kira ...

   # display it
3% xstarplot ....
```

6.3 partiview

An NCSA product. Excellent interactive capabilities. Uses GL. A special format, dubbed `tdyn`, can be produced by `kira` in order for `partiview` to seamlessly zoom in space **and** time. NEMO has a conversion program `snapspecks`.

6.4 tipsy

This is a neat X based application. Might have problems with non-8 bit display.[1]

```
1% startx -- :1 -depth 8

2% cd $MANYBODY/tipsy/
3% tipsy

> openascii run99.ascii
> readascii run99.bin
  read time 14.970800
```

[1] See comments in `$NEMO/src/nbody/evolve/aarseth/triple/README.pjt`

```
> loadb 14
  used time 14.970800, hope you don't mind master
> xall
> quit
```

6.5 Tioga

Written in ruby, in active development. Good quality PDF output, animations are also possible.

6.6 glstarview

Comes with Fregeau's code. Data from the integration can be directly piped (or tee'd) into this program for visualization. Notice that `xstarplot` in starlab can also do this.

```
1% binbin -D 1 | glstarview
```

6.7 starsplatter

StarSplatter is a tool for creating images and animations from astrophysical particle simulation data. It treats each particle as a Gaussian "blob", with an exponential fall-off similar to the SPH smoothing length or gravitational softening length common in astrophysics simulations. It also properly anti-aliases points, so even if the particles are very small the results may look better than one would get by simply binning particle counts to produce an intensity image.

See `http://www.psc.edu/Packages/StarSplatter_Home/`

You can also find tools there to help you set up a scene, choose a pan and zoom around.

6.8 gravit

This program has recently been developed, and has various neat visualizaton options and animates simulations. It also displays the oct-tree of the underlying Barnes & Hut treecode. There are plans to integrate this with some of the existing codes.

`http://gravit.slowchop.com/`

6.9 povray and blender

Great to create photo-realistic images. Slightly steep learning curve, but excellents books available. And `blender`. For an example, see `http://www.bottlenose.demon.co.uk/galactic/index.htm`.

`http://www.povray.org/` `http://www.blender.org/`

6.10 spiegel

As part of the gravitySimulator project at RIT, visualization software using Java and Java3D is being developed.

`http://www.cs.rit.edu/~grapecluster/`

6.11 visit

`http://www.llnl.gov/visit/`

6.12 ifrit

`http://home.fnal.gov/ gnedin/IFRIT/` uses VTK and QT and is written in C++. It is a powerful tool that can be used to visualize 3-d data sets.

6.13 xnbody

an online visualization tool for nbody6++, written by Sonja Habbinga for her diploma thesis at the Research Centre Juelich in Juelich, Germany. `http://www.fz-juelich.de/jsc/JSCPeople/habbinga`

6.14 OpenDX

Open source visualization package, written by IBM. See `http://www.opendx.org/`. Can be quite memory intensive, but has excellent tools for particle as well as grid data.

6.15 AstroMD / VisiVO

cosmolab distributes an open-source vizualization package called `AstroMD`. Uses the Visualization Tool Kit "vtk" (see `http://public.kitware.com/VTK/`).

6.16 xgobi, ggobi

Great for multi-variate analysis after you have accumulated hundreds or thousands of simulations. There is also a limited version of this kind of dynamic query technique available in NEMO's program `tabzoom`, which uses pgplot for visualization and a simple command line interface for interactions.

6.17 python

Excellent language, now very popular. Graphics interface `matplotlib`, or alternatively `qwt`.

6.18 R

Although R is a statistics package, it has a very nice integrated way for visualization, and allows extending the package by using your own compiled code. It could potentially be useful for our type of analysis. URL: `http://cran.us.r-project.org/`

6.19 IDL

Although this program is not freely available, many people have written display and analysis routines for IDL that are freely available. Gadget comes with some routines. See also NEMO's inefficient `snapidl` routine to export data for IDL. The Berkeley group (Marc Davis) has some routines on their website and some of your instructors admitted having used it (but they never inhaled).

A public version, GDL, is available which is largely IDL V6 compatible, and uses `gnuplot` are the graphics engine.

6.20 ImageMagic

You always need image conversion programs. `ImageMagic` is one of the better packages for this. Much like NEMO and Starlab, ImageMagic is a collection of image manipulation programs (e.g. convert, montage, mogrify). Reads and writes just about any image format, including some flavors of FITS. Most linux distributions make this package directly available.

6.21 Movies

Animations are still the most popular way to convey your results in a presentation. Good tools exist, but the intrusion of closed sourced formats (or CODECs within a particular format) has scattered the field.

mpeg, animated gif, avi, mpg2avi ....

`http://bmrc.berkeley.edu/frame/research/mpeg/mpeg2faq.html`

`http://www.bergen.org/AAST/ComputerAnimation/Help_FAQs.html`

`http://the-labs.com/GIFMerge/`

`$NEMO/csh/mkmpeg_movie : some example scripts to make movies`

Chapter 7

Exercises

1. Find out what time step criterion the first N-body code (von Hoerner) was using. How do `nbody0` and this program (`firstn`) scale in their performance if you want to reach the same accuracy?

2. The output of the example `firstn` output listed earlier (examples/ex1), can depend on the compiler, and its flags. For example the Q=0.55 with 3624 steps can also become Q=0.65 with 3569 steps! Can you understand this. Is this worrysome?

3. What are these peculiar "`time resets`" quoted in one of the `nbody0` examples.

4. Why was the location of the 25th particle not given in the IAU 25-body problem (Lecar 1968)?

5. Take `hackcode1`, There are two ways to get more accurate results: add a quadrupole term (as implemented in `hackcode1_qp`) or decrease the critical opening angle (`tol=`). Device an experiment to show which of the two is the less compute intensive. Does it depend on the number of particles or the configuration?

6. Back to kindergarden: connect the dots in Figure 2.3 and label which codes they are.

7. Take an original integrator, one without a NEMO interface, and write a user friendly shell script (sh, csh, python, perl, ...) that can take a NEMO snapshot input, and produces a NEMO snapshot output file, for given timestep dt, output timetep dt_out and a final integration time of tstop.

8. What is better if you want to find the center of a snapshot. Using small N and doing a lot of (K) experiments, or

9. For a given configuration (for example a Plummer sphere), define a statistical measure for the accurary of the treecode as a function of the critical opening angle, and compare the classical BH treecode O(NlogN) with and without quadrupole corrections to that of the O(N) Dehnen treecode.

10. Follow up on the previous exercise and add Gadget2 in the comparison.

11. Use CVS (see Appendix C) and study the performance of gyrfalcON on the following dates: now, 1-jul-2006, ..

12. Try out the `galaxy` screensaver in Linux, study their code and generate initial conditions for NEMO integrators. How realistic are they? Compare Toomre & Toomre's (1972) classic work.

13. Study the Holmberg (1941) galactic disk, in particular the stability properties.

Chapter 8

References

Aarseth, S.J. - *Gravitational N-body Simulations : Tools and Algorithms* (Cambridge Univ. Pr., 2003),

Aarseth, S.J. - 2004.. *NBODY6 Users Manual* : **man6.ps**

Aarseth, S.J. - 2006.. *Introduction to Running Simulations with NBODY4* : **nbody4_intro_0602.pdf**

Aarseth, S. J.; Henon, M.; Wielen, R. - 1974 *A comparison of numerical methods for the study of star cluster dynamics* : **1974A+A....37..183A.pdf** [1]

van Albada, T. S.; van Gorkom, J. H. - 1977 - *Experimental Stellar Dynamics for Systems with Axial Symmetry*: **1977A+A....54..121V**

Barnes, J. & Hut, P. - 1986 - Nature - 324, 446.

Barnes & Hut - 1989 *Error analysis of a tree code* - **1989ApJS...70..389B.pdf**.

Binney, J. & Merrifield, M. *Galactic Astronomy*, Princeton University Press. 1998.

Binney, J. & Tremaine, S. - *Galactic Dynamics*, Princeton University Press. 1987.

Cruz, F.; Aguilar, L. A.; Carpintero, D. D *A New Method to Find the Potential Center of N-body Systems* - 2002RMxAA..38..225C **RMxAA..38-2_cruz.pdf**

Bode, Paul, Ostriker, Jeremiah P., Xu, Guohong *The Tree Particle-Mesh N-Body Gravity Solver* - **2000ApJS..128..561B.pdf**

Dehnen, W. - *A Very Fast and Momentum-Conserving Tree Code* - 2000 astro-ph/0003209 : **0003209.ps**, **2002JCP...179...27D.pdf**

Fellhauer et al - 2000 *SUPERBOX - An efficient code for collisionless galactic dynamics* NewA, 5, 305. **2000NewA....5..305F.pdf**

Fregeau, J.M., Cheung, P., Portegies Zwart, S.F., and Rasio, F.A. (2004) *Stellar Collisions During Binary-Binary and Binary-Single Star Interactions* astro-ph/0401004 : **0401004.ps**

[1] papers marked in **boldface** are available electronically on the CD in the `papers/` directory

Frenk, C. et al - 1999 *The Santa Barbara Cluster Comparison Project: A Comparison of Cosmological Hydrodynamics Solutions* **1999ApJ...525..554F**

Goodman, Jeremy; Heggie, Douglas C.; Hut, Piet - 1993 - *On the Exponential Instability of N-body Systems* : **1993ApJ...415..715G.pdf**

Gualandris, A., Portegies Zwart, S., and Tirado-Ramos, A. - 2004 *Performance analysis of direct N-body algortithms on highly distributed systems*: IEEE - **gualandris2004.pdf**

Heggie, D. & Hut, P. 2003, *The Gravitational Million-Body Problem*, Cambridge University Press.

Heitmann, Katrin; Ricker, Paul M.; Warren, Michael S.; Habib, Salman - 2005 *Robustness of Cosmological Simulations. I. Large-Scale Structure* **2005ApJS..160...28H.pdf** - or **0411795.pdf** from astro/ph

Hernquist, L. 1989. *Performance characteristics of tree codes* Ap J Supp. 64, 715. **1987ApJS...64..715H.pdf**.

Hernquist, L. & Barnes, J. 1990 - *Are some N-body algorithms intrinsically less collisional than others?* Ap.J. 349, 562. **1990ApJ...349..562H.pdf**

Hernquist & Ostriker - 1992 - *A self-consistent field method for galactic dynamics* **1992ApJ...386..375H.pdf**.

Hernquist - 1993 - ApJS - how to create initial conditions (we have the code, but doesn't seem to work well)

Hockney & Eastwood - 1970? *Simulations using Particles* - there is now a 2nd edition.

Hozumi & Hernquist - 1995 - ....

Holmberg, E. - 1941 - Ap.J. 94, 385. **1941ApJ....94..385H.pdf**.

Hurley et al.- astro-ph/0507239

Hut, P. & Makino, J., 2003 *Moving Stars Around* : **acs_v1_web.pdf**

Hut, P. & McMillan, S.L, 1987 *The Use of Supercomputers in Stellar Dynamics* :

Joshi, Kriten J.; Rasio, Frederic A.; Portegies Zwart, Simon - 2000 - *Monte Carlo Simulations of Globular Cluster Evolution. I. Method and Test Calculations*, ApJ 540, 969: **2000ApJ...540..969J.pdf**

Kang, H. et al - 1994 *A comparison of cosmological hydrodynamic codes* **1994ApJ...430...83K**

Khalisi, E., & Spurzem, R., 2003 - *NBODY6 : Features of the computer code.* (last update: 2003, May 02)

Kuijken & Dubinski - 1995 *Nearly self-consistent disc-bulge-halo models for galaxies.* MNRAS, 277, 1341 : **1995MNRAS_277_1341K.pdf**.

James, R. - 1977 - J.Comp. Phys. 25, 71.

Lecar, M. - *The Standard IAU 25-body problem* - 1968, Bull.Astr. 3, 91.

Steven Phelps, Vincent Desjacques, Adi Nusser, Edward J. Shaya - 2005 *Numerical action reconstruction of the dynamical history of dark matter haloes in N-body simulations* - astro-ph/0512405: **0512405.pdf**

Portegies Zwart, Simon F.; Baumgardt, Holger; Hut, Piet; Makino, Junichiro; McMillan, Stephen L. W. - 2004 - *Formation of massive black holes through runaway collisions in dense young star clusters* Nature 428,

724 (astro-ph/0402622): **0402622.pdf**

Rauch, K. & Hamilton, D. - 2005 - in preparation. See also `http://janus.astro.umd.edu/HNBody`

Romeo, Alessandro B.; Horellou, Cathy; Bergh, Jran - 2003 - *N-body simulations with two-orders-of-magnitude higher performance using wavelets* MNRAS, 342, 337 : **2003MNRAS.342..337R.pdf**

Salmon, J.K. & Warren, M.S. - 1994 - *Skeletons from the Treecode Closet* J. of Comp. Phys, **111**, 136: **skeletons.ps**

Sellwood, J.A. - *The art of N-body building* - 1987, Ann.Rev A&A, 25, 151 : **1987ARA+A..25..151S.pdf**

Sellwood (1997, *Computational Astrophysics* : in *Galaxy Dynamics by N-body Simulations*, ed Clarke & West, ASP Conf series v123, p215) - astro-ph/9612087 **9612087.ps**

Spinnato, Piero F.; Fellhauer, Michael; Portegies Zwart, Simon F. - 2003 - *The efficiency of the spiral-in of a black hole to the Galactic Centre* : MNRAS 244, 22S.: **2003MNRAS.344...22S.pdf**.

Springel V. - 2005 - *The cosmological simulation code GADGET-2* **2005MNRAS.364.1105S.pdf gadget2-paper.pdf**

Springel V., Yoshida N., White S. D. M., 2001, New Astronomy, 6, 51. *GADGET: A code for collisionless and gasdynamical cosmological simulations*: **gadget-code-paper.pdf**.

Szebehely, Victor, Peters, C. Frederick - 1967 - *Complete solution of a general problem of three bodies*: **1967AJ.....72..876S.pdf**

Takahashi, Koji; Portegies Zwart, Simon F. - 2002 - *The Evolution of Globular Clusters in the Galaxy* - ApJ 535, 759: **2000ApJ...535..759T.pdf**

Theis, C. it Two-body relaxation in softened potentials - 1998, Astron.Astroph. 330, 1180: **theis98.pdf**.

Trenti 2005.

von Hoerner, S. - 1960 - Z.f.Astrophys. 50, 184. **1960ZA.....50..184V.pdf**

Weinberg, M. - 2001a - *Noise-driven evolution in stellar systems - I. Theory*: **2001MNRAS.328..311W**

Weinberg, M. - 2001b - *Noise-driven evolution in stellar systems - II. A universal halo profile*: **2001MNRAS.328..321W**

White,S.D.M. - 1983 - Ap.J. 274, 53 : **1983ApJ...274...53W.pdf**

White,S.D.M. - 1996 *...DaTitle....*, in: *Les Houches Lecture Notes* **94100043.ps**

White & Barnes - 1984 - **1984MNRAS.211..753B.pdf**

Zhan - astro-ph/0507237 **Zhan-0507237.pdf**

URLs

http://modesta.science.uva.nl/modest/modest5c/index.html	this N-body school (2005)
http://astro.u-strasbg.fr/scyon/School/	the previous N-body School (2004)
http://www.artcompsci.org/	The Art of Computational Science
http://www.manybody.org/	Manybody portal (including MODEST)
http://www.astro.umd.edu/nemo	NEMO website
http://www.ids.ias.edu/~starlab/	Starlab website
http://muse.li/	MUSE
http://www.amusecode.org/	AMUSE

Appendix A

Setting Up Your Account

A.1 Leiden 2010 setup

We will be using Linux workstations, running Suse 10.1 (x86-64) with dual 64bit Intel 3 GHz CPU processors (totaling 150 GHz!). Each workstation comes with 1 GB memory. Bla bla.

GH06: A common data directory `/soft/gh06` is available on all 25 linux lab workstations. These machines are named `c01tgh`, `c02tgh` ... `c24tgh` and the server is called `ctgh06`. Each student has been assigned an account `stud01`, `stud02`, ... `stud50`, the initial passwords are written on the white-board, but please change them. Also note that depending on your `studXX` account, you will need to be on one of the specific `cYYtgh` machines. There is a simple algorithm between `XX` and `YY`. See also the whiteboard.

Printing should be done via the HP LaserJet 4050 in the lab. For those connecting their laptops, use the "Networked JerDirect" queue type, with printer at 192.168.100.176 on port number 9100[1]

Here it is assumed that you are using the C-shell (`csh` or `tcsh`) or Bourne Shell (`bash`). Easiest is to add the following command to your `.cshrc` file for csh

```
source /soft/gh06/manybody/manybody_start.csh
```

or `.bashrc` for bash users

```
source /soft/gh06/manybody/manybody_start.sh
```

This will add the NEMO and STARLAB packages to your environment, as well as a number of other tools. It will also give you access to a modest amount of source code that has not been compiled, and a number of papers drawn from ADS and astro-ph and placed in `$MANYBODY/papers`. If you want to install/copy this onto your laptop or home machine, please see Appendix B.

[1] in CUPS this shows up as `DeviceURI socket://192.168.100.176:9100`

A.2 A quick test

To verify the installation was done correct, here are some quick one liners, without much of an explanation. Some of them will bring up a picture. Most commands, except where noted, take a few seconds. If you prefer, you can also execute the `quick_test` script in the `$MANYBODY` root directory.

1. `firefox $MANYBODY/index.html` : local HTML
2. `mkplummer p1 1000` : NEMO, create a Plummer sphere
3. `snapplot p1 color=x` : NEMO, check if pgplot works. check color bands
4. `gyrfalcON p1 . tstop=0.1` : NEMO, check if an integrator works
5. `mkkd95 a0.dat`: NEMO: create composite galaxy (takes 45 secs on a P1.6GHz)
6. `snapplot a0.dat color='i<8000?2.0/16.0:(i<12000?3.0/16.0:4.0/16.0)' yvar=z xrange=-8:8 yrange=-8:8` : NEMO: parse bodytrans functions?
7. `snapxyz a0.dat - color='i<8000?1:(i<12000?2:4)' | xyzview - maxpoint=18000 nfast=18000 scale=8 fullscreen=t` : NEMO 3D viewer? Hold down key '1', '2' or '2' and move the mouse. Hit ESC to quit.
8. `glnemo a0.dat select=0:7999,8000:11999,12000:17999` : NEMO 3D viewer. Press left mouse to view the object from different angles, use the scroll mouse (if you have one) to zoom in and out.
9. `snapgrid a0.dat - xrange=-8:8 yrange=-8:8 nx=128 ny=128 yvar=z | ccdsmooth - - 0.1 | ccdfits - a0.fits` : NEMO make a CCD type observation of dark matter

   `ds9 a0.fits` : vizualize it. right mouse down changes contrast. Try zooming in and changing color map to SLS.
10. `nemobench gadget` : a small Gadget2 benchmark (TBA)
11. `makeplummer -n 32 > p32.dyn` : Starlab : create a Plummer sphere
12. `makeplummer -n 10 | dtos - | snapprint -` : a Starlab-NEMO pipe and converting data
13. `makeplummer -n 32 | kira -D 0.1 -t 2 | xstarplot` : Starlab w/ graphics. 'q' to quit.
14. `make -f $MANYBODY/partiview/data/primbin16.mk` : Starlab: 16 body integrator. Takes about 15 secs.
15. `partiview $MANYBODY/partiview/data/primbin16.cf` : Partiview: view animation. Resize window. Left mouse down rotates. Right mouse down zooms. >> button starts movie.
16. `binsingle -D 1 | glstarview` : fewbody + visualizer
17. `g++ -o forward_euler1 $MANYBODY/acs1/chap3/forward_euler1.C; forward_euler1` (ACS 1)
18. `...ruby...` (ACS 2)

A.3 Available packages and commands

```
NEMO            package with 200+ tools
Starlab         package with 200+ tools

partiview       4D (space+time) data viewer
ds9             a FITS image display program
fewbody         programs that come with Fregeau's ``fewbody''
nbody6          Aarseth's nbody6 integrator
gadget2         SPH-treecode for cosmology
```

Appendix B

manybody: N-body Compute Toolbox

B.1 Linux Cluster

The directory `/soft/gh06` on the 25 Linux Lab machines contains all the *manybody* software (and then some) we are discussing at this school. We have also made a DVD available, from which you should be able to reproduce this toolbox on your own laptop or workstation at home. Some details on the installation of this are described in the next sections.

First an overview of the *manybody* hierarchy under `/soft/gh06`:

```
manybody/                   root directory
    opt/                    contains a {bin,lib,include,...} tree for a private --prefix
    acs/                    kali code
    nemo_cvs/               NEMO package
    starlab_cvs/            Starlab package
    partiview_cvs/          partiview visualization
    starcluster_cvs/        starlab interface to web based programs
    fftw-2.1.5/             fftw library
    papers/                 ADS and astro-ph papers (PDF and PS mostly)

manybody_pkg/               (mostly) tar balls of codes
```

B.2 The DVD

In the root directory of your DVD you will find the following files and directories:

```
    README              a small intro
    install_fc5         shell script that should install binary "manybody"
    index.html          top level index to all HTML in this tree
```

```
quick_test            shell script to test MANYBODY (see Appendix A.2)

manybody_fc5.tar.gz   FC5 binaries of manybody, without the papers
papers.tar.gz         astro-ph and ADS paper for the manybody material

manybody_pkg/         directory with all sources in case of re-install
```

B.3 Installation

If your linux system is not Fedora Core 5 or compatible, recompilation may be needed, since its shared library requests may be conflicting with the ones your system has. The script `install_manybody` should guide you through this. You can find a copy of this in either the `manybody` directory, or its original CVS location, `$NEMO/usr/manybody`.

```
1% tar zxf /mnt/cdrom/manybody_fc5.tar.gz
2% source manybody/manybody_start.csh
3% tar zxf -C manybody /mnt/cdrom/papers.tar.gz
```

if you need to do a source install, the following should be your starting point[1]

```
11% /mnt/cdrom/install_manybody pkg=/mnt/cdrom/manybody_pkg
12% source manybody/manybody_start.csh
```

but more than likely some tweaking is needed if you have missing or incompatible libraries or tools.

```
4% setenv CC  gcc-32
5% setenv CXX g++-32
6% setenv F77 g77
7% /mnt/cdrom/install_manybody pkg=/mnt/cdrom/manybody_pkg
```

A typical install takes about an hour.

B.4 Importing new Packages into manybody

Much of the available open source software we use, use *autotools* to `configure`, `make` and `install` their software on the users system. Where exactly this software is going to be located, is free to the user to decide. On most Unix-based system the `/usr`, `/usr/local`, `/opt/local` e.g. DarwinTools on MacOSX or `/sw` (e.g. Fink on MacOSX) are used for this. As long as the user then puts the respective `''bin''` directory in their search path, and perhaps the respective `''lib''` directory in their shared library search path, all is well.

[1] the root name `/mnt/cdrom` may very well be different on your machine

The drawback of this approach is that the user needs have administrative privilages (username `root` in Unix parlor). For **manybody** we cannot always assume that, so we choose a similar approach whereby libraries, programs and their ancillary material are placed in a directory under user control.

So, for almost all such packages we can use these kinds of commands:

```
1% tar zxf /tmp/package-X.Y.Z.tar.gz
2% cd package-X.Y.Z
3% ./configure --prefix=$MANYBODY/opt
4% make
5% make install
```

and for packages that depend on other packages, we would use something like

```
3% ./configure --prefix=$MANYBODY/opt --with-fltk=$MANYBODY/opt
```

B.5 Example install session

```
% install_manybody
install_manybody : version ...
Wed Jul 5 14:56:00 EDT 2006 Using pkg=/scratch11/teuben/manybody_pkg
Wed Jul 5 14:56:01 EDT 2006 Installing cvsutils
Wed Jul 5 14:56:02 EDT 2006 Installing ruby
Wed Jul 5 14:57:18 EDT 2006 Installing acs
Wed Jul 5 14:57:22 EDT 2006 Installing starlab
Starlab version 4.4.2 loaded with
        STARLAB_PATH         = /scratch11/teuben/manybody/starlab_cvs
        STARLAB_INSTALL_PATH = /scratch11/teuben/manybody/starlab_cvs/usr
STARLAB-COUNT: 176
Wed Jul 5 15:05:09 EDT 2006 Installing starcluster
ERRORS compiling starcluster, check /scratch11/teuben/manybody/tmp/starcluster.log
ERRORS installing starcluster, check /scratch11/teuben/manybody/tmp/starcluster.log
STARLAB+CLUSTER-COUNT: 176
Wed Jul 5 15:05:09 EDT 2006 Installing fltk
Wed Jul 5 15:06:03 EDT 2006 Installing partiview
ERRORS compiling partiview, check /scratch11/teuben/manybody/tmp/partiview.log
cp: cannot stat ‘partiview’: No such file or directory
Wed Jul 5 15:06:25 EDT 2006 Installing NEMO
NEMO-COUNT: 209
Wed Jul 5 15:10:51 EDT 2006 Installing gadget
Wed Jul 5 15:29:07 EDT 2006 Installing firstn
Wed Jul 5 15:29:08 EDT 2006 Installing nbody6
Wed Jul 5 15:29:29 EDT 2006 Installing galactICS
Wed Jul 5 15:29:34 EDT 2006 Installing gsl
Wed Jul 5 15:33:44 EDT 2006 Installing fewbody,glstarview
ERRORS compiling glstarview, check /scratch11/teuben/manybody/tmp/fewbody.log
cp: cannot stat ‘glstarview’: No such file or directory
Wed Jul 5 15:33:47 EDT 2006 Installing hnbody- for linux only
```

```
Wed Jul 5 15:33:47 EDT 2006 Installing ds9 - for linux only
Wed Jul 5 15:33:47 EDT 2006 Installing xyz from NEMO
Wed Jul 5 15:33:55 EDT 2006 Installing EZ
ERROR: no known fortran compiler for EZ available (ifort, f95 g95)
Wed Jul 5 15:33:56 EDT 2006 Installing Tioga
Wed Jul 5 15:34:04 EDT 2006 Installing MMAS
Wed Jul 5 15:34:04 EDT 2006 Installing FFTW
Wed Jul 5 15:34:48 EDT 2006 Instaling StarCrash
Wed Jul 5 15:34:53 EDT 2006 Done.
```

In this particular example you see some errors occurred, `glstarview` and `partiview` did not properly build on this machine (a missing library), and there was no fortran-95 compiler available for EZ to be compiled. Even though NEMO returned with 209 binaries, this is not a full success, where 222 have been seen.

B.6 ACS

Say something about the Hut & Makino *ACS* series. ACS1 (C++) vs. ACS2 (ruby).

Appendix C

Using CVS

If you have never heard of CVS it is worth reading this appendix and considering to use it (a few packages in manybody are CVS enabled, encouraging you to use this timesaving mode). CVS is one of the more popular source code control systems, which simplifies keeping your source code up to date with a master version. It can be useful for collaborators to work on a project (source code, a paper), but also for a single developer testing out code on various machines.

C.1 Anonymous CVS

Most likely you will first start by using the so-called "anonymous CVS" method, which works much like anonymous-ftp. It is however important to enable your CVS account first, using the `cvs login` command[1]:

```
1% cvs -d :pserver:anonymous@cvs.astro.umd.edu:/home/cvsroot login
Logging in to :pserver:anonymous@cvs.astro.umd.edu:2401/home/cvsroot
CVS password:
```

simply hit return here, since there is no password. You only need to do this once per CVS account, as the account information is added to a file `.cvspass` in your home directory.

C.2 Starting from scratch

1. The environment variable CVSROOT, or the -d flag to the cvs command, is needed to get accesss to a repository. Use the one listed in the previous section after the `-d` flag.
2. You then need to checkout a new sandbox that mirrors a repository module (the `-Q` flag makes it much less verbose:

[1] In this and age of security, there is a chance this command will hang and not continue. Port 2401 may be blocked by a router near your connection

```
   # checkout nemo, assuming CVSROOT has been set
%1 cvs -Q co nemo

   # checkout starlab, notice the somewhat odd looking module name under manybody
%2 cvs -Q co -d starlab manybody/starlab
```

C.3 Starting with an existing package that is CVS enabled

If you already have a directory tree that is "CVS enabled" (each directory will have a `CVS` subdirectory in which administrative details of that directory are stored), life is even easier.

1. By default the contents of the `CVS/Root` file(s) will be used to get access to the repository. Otherwise, as before, the CVSROOT environment variable, or the -d command option flag, can be used. Basically you don't need to worry about setting CVSROOT or using the -d flag here.
2. To check your source code for any needed updates: `cvs -n -q update`. You might get a response like:

```
? src/kernel/io/try.c            <-- file not under CVS control
U man/man1/mkplummer.1           <-- newer file on the server
M man/man5/data.5                <-- you have a modified file
C src/nbody/init/mkplummer.c     <-- a conflict!!! both server and you have modified
```

   The first column designates the status of the file. We will treat the listed 4 cases seperately (there are a few more, but not so common):

`?` These files can be safely ignored. They happened to be in the directory as a side-effect of installation or some other tinkering you did. They could be added to CVS using the `cvs add` command.

`U` This means the file is new(er) at the server, and your CVS sandbox needs to be updated. A simple command: `cvs update`.

`P` You might see this when you `cvs update` a file, instead of Updating the file, it is patched, taking much less bandwith.

`M` This means your version of the file is newer than on the server, and as long as you have write permission on the server, it can be returned with a simple command: `cvs commit`.

`C` This is the more complicated case of a conflict. Both your local version, as well as the server version, have been modified independently. Most of the time CVS is actually able to make a version that merges both modifications, though the developer should now check if the resulting code is correct. The correct CVS order to fix this is a three step process: the code is updated, then edited and checked for correctness, and finally committed back to the repository:

```
1% cvs update mkplummer.c
2% make/edit/check/debug mkplummer.c
3% cvs commit mkplummer.c
```

C.4 cvsutils

In your *manybody* environment you will find a few perl scripts (dubbed `cvsutils`)[2] that help with some tedious CVS interactions, notably some tools when you are not online. Another common enough operation is to change the `CVS/Root` (where your local `$CVSROOT` lives) from anonymous to somebody with write permission, viz.

```
1% cd $NEMO
2% cat CVS/Root
:pserver:anonymous@cvs.astro.umd.edu:/home/cvsroot
3% cvschroot :pserver:pteuben@cvs.astro.umd.edu:/home/cvsroot
```

C.5 svn

The future of CVS is unclear. A lot of open source packages are switching to SubVersion (command: `svn`). In practice the `cvs` and `svn` commands are nearly interchangeable.

C.6 git

A more recent innovation is the use of distributed CMS, such as `git`. This has the advantage of maintaining/creating your own repository and only later merge them. The linux kernel is currently maintained using *git*.

[2]See `http://www.red-bean.com/cvsutils/`

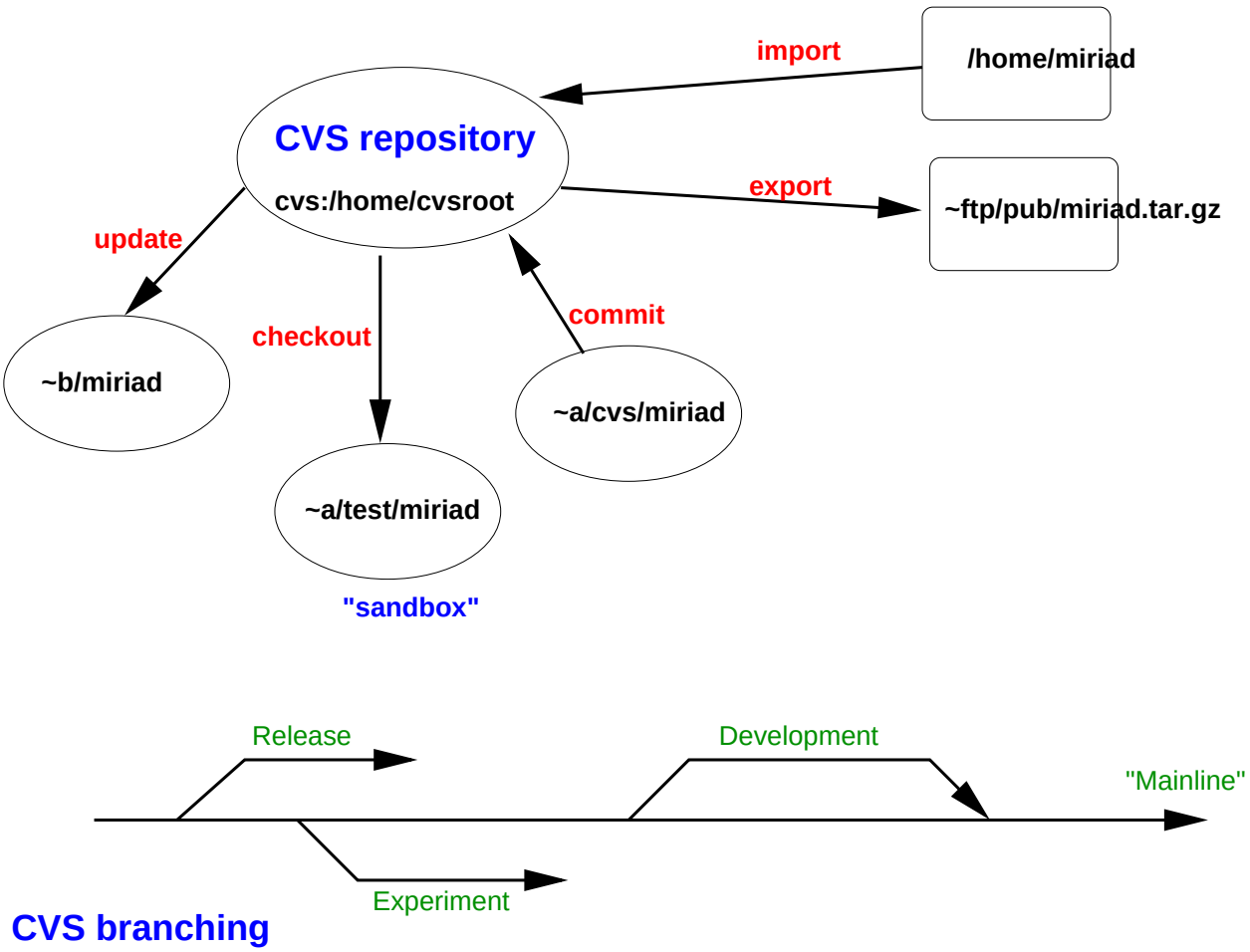

Figure C.1: CVS flow diagram

Appendix D

Using NEMO

A summary of useful things to know about NEMO:

1. NEMO is a large (250+) collection of programs, each performing a small task controlled by a set of parameters.
2. the command line user interface (CLI) is a set of *program keywords* (unique to each program) and *system keywords* (available to each program). Most often used system keywords are `yapp=` (plotting device), `debug=` (increase debug level) and `error=` (pass over fatal errors at your own risk).
3. each program displays internal help using the option `--help`, `help=` or `help=h`. Use `help=\?` to see all the kinds of help the CLI can give. Use the `man` or `gman` command to view or browse manual pages.
4. command line parameters do not need to have their keyword name given if the parameters are given in the right order. E.g. `mkplummer p10 10` is the same as `mkplummer out=p10 nbody=10`. Use `help=` to see that order.
5. most programs have a manual page. Use the unix `man` command, or GUI tools like `gman` [under redhat this is hidden in the yelp package]
6. data is mostly stored in binary files, for speed, accuracy and portability. Programs like `tsf` display the contents of these files.
7. in Unix tradition data is often piped from one program to the next, the `in=/out=` filename needs to be designated with a dash (e.g. `in=-, out=-`. Most programs have in/out as their 1st and 2nd parameter.
8. a large number of programs produce ASCII tables. Programs like `tabplot` and `tabhist` are convenient endpoints to quickly present results graphically.

One or two overview plots on the packages and file formats in NEMO will now follow.

Appendix E

Using Starlab

A summary of useful things to know about Starlab:

1. Starlab is a large (250+) collection of programs, each performing a small task controlled by a set of parameters.
2. the command line user interface (CLI) is a set of flags
3. each program displays internal help using the option `--help`. It actually does this by tracking down the source code and displaying the relevant help section of the source code.
4. data is mostly stored in special ascii files, with a hierarchical structure
5. in Unix tradition data is piped from one program to the next, and unlike in NEMO, unknown data is generally passed onwards unmodified.

Appendix F

Using AMUSE

Appendix G

Cover Art

The figures on the cover page were made in the following way, printed out and cut and pasted the classical way.

G.1 Collisional Orbit

Although the *Figure8* orbit is perhaps very artistic, it is not very representative for orbits in collisional systems, in fact, almost more appropriate for collisionless systems! The collisional orbit choosen was drawn from the Pythagoran problem: 3 stars with masses 3, 4 and 5, on a pythagoran triangle where the long sides opposite their mass have a length proportional to their mass. Initially all three stars are at rest. A file `$NEMODAT/pyth.dat` contains the initial conditions:

```
   # print out the initial conditions
1% snapprint $NEMODAT/pyth.dat m,x,y,z,vx,vy,vz
m  x  y  z  vx vy vz
3  1  3  0  0  0  0
4 -2 -1  0  0  0  0
5  1 -1  0  0  0  0

   # integrate for a while
2% nbody00 $NEMODAT/pyth.dat pyth.out eta=0.01 deltat=0.01 tcrit=100 eps=0
time = 0   steps = 0   energy = -12.8167 cpu =          0 min
time = 0.01   steps = 1   energy = -12.8167 cpu =          0 min
...
time = 99.98   steps = 33940   energy = -12.7912 cpu =     0.0115 min
time = 99.99   steps = 33945   energy = -12.7912 cpu =     0.0115 min
time = 100   steps = 33949   energy = -12.7912 cpu =     0.0115 min
Time spent in searching for next advancement: 0.1
Energy conservation: 0.0254786 / -12.8167 = -0.00199189
Time resets needed 5977 times / 10001 dumps

   # view Red=0 Green=1 Blue=2
```

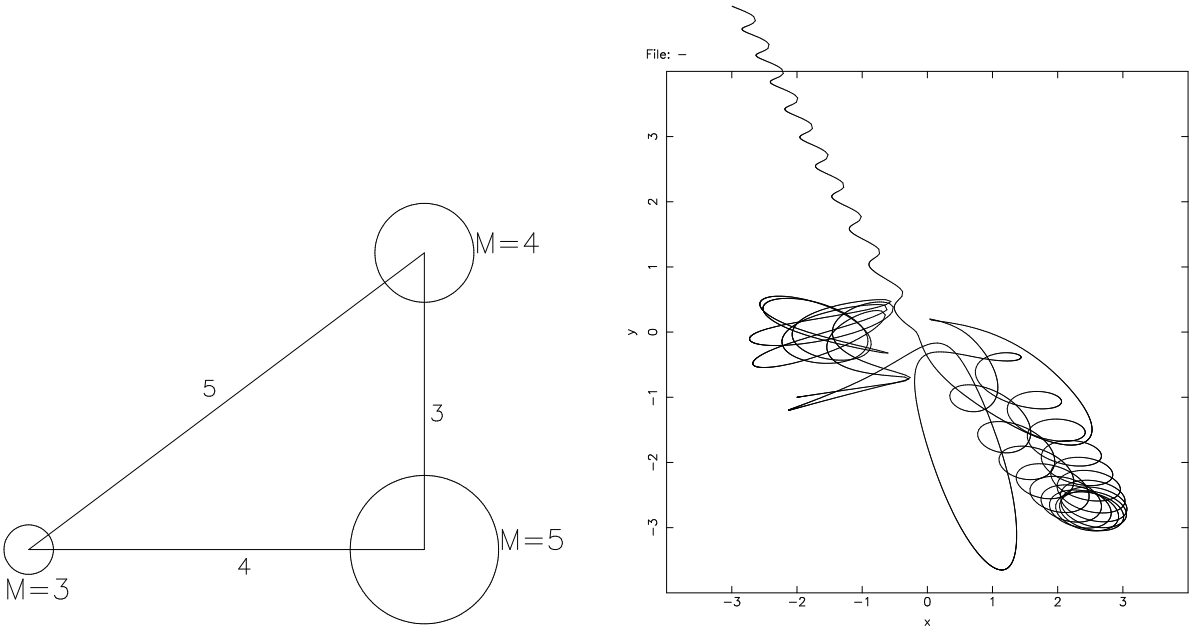

Figure G.1: Initial conditions for the Pythagorean problem (left, but notice **???**but they are** these are not the official Szebeheley & Peters configuration we use in the example) and the orbit of the intermediate mass (M=4) particle starting at (x,y) = (-2,-1).

```
   # notice that although initially blue/green pair up, red intervenes
   # and takes off with green heading up, leaving blue to wander south...
3% snapplot pyth.out trak=t xrange=-4:4 yrange=-4:4 color='i==0?2.0/16.0:i==1?3.0/16.0:4.0/16.0'

   # notice the final breakup occurs around time=82.4
4% snapprint pyth.out t,r | tabplot -
5% snapprint pyth.out t,r | tabplot - xmin=80 xmax=84

   # need to track down some unexpected dependancies of the results on compiler version
   # and options. this is of course somewhat disturbing.
   # all the trouble starts at that very close encounter around T=82

6% stoo pyth.out - 1 100000 | orbplot - xrange=-4:4 yrange=-4:4
```

The 3-body problem is a chaotic problem. By perturbing the initial conditions even a little bit, you can get drastically different results. This was shown in another paper by Shebehely & Peters (1967AJ.....72.1187S), where the positions of the 3 bodies were all perturbed by of order 0.04, and this created a periodic solution.

G.2 Collisionless Orbit

Here we are taking orbits from the logarithmic potential as defined by Binney & Tremaine (1987) in eq. (2.54) and (3.77).

$$\Phi(x,y) = \frac{1}{2}v_0^2 \ln\left(r_c^2 + x^2 + y^2/q^2\right)$$

The orbits in their Figure 3-7, our Figure G.2, are for $q = 0.9, R_c = 0.14, v_0 = 0.07$.

```
   # Create an orbit, much like the one in Figure 3-7 of BT87
1% mkorbit orb1 y=0.1 vx=-1 etot=-0.337 potname=log potpars=0,0.07,0.14,0.9
Using pattern speed = 0
pos: 0.000000 0.100000 0.000000
vel: -1.000000 1.330308 0.000000
etot: -0.337000
lz=0.100000

2% orbint orb1 orb1.out nsteps=10000 dt=0.01 ndiag=100
0.000000 1.384859 -1.721859               -0.337 -0
1.000000 0.074342 -0.411342     -0.3369998357925 -4.87263e-07
...
98.970000 1.434173 -1.771178     -0.3370046579526 1.38218e-05
99.980000 0.016410 -0.353415     -0.3370045122688 1.33895e-05
Energy conservation: 1.33895e-05

3% orbplot orb1.out xrange=-0.8:0.8 yrange=-0.8:0.8
```

G.3 Barred Galaxy

Here we produce an image of a simulated barred galaxy, overlayed with a contour diagram of a smoothed distribution of stars. First a barred galaxy is created by integrating an unstable exponential disk for a few rotation times.

```
   # Create an unstable disk
1% mkexpdisk disk.in 4096

   # integrate for a few rotation times
2% gyrfalcON disk.in disk.out tstop=5 step=1
#    time      energy       -T/U     |L|       |v_cm|  build   force    step     accum
# ------------------------------------------------------------------------------------
 0          -0.5751588    0.492   0.42204    2.2e-10  0.01   0.04       0.05      0.05
 0.015625   -0.575151     0.49177 0.42204    5.8e-10  0      0.05       0.05      0.11
 0.03125    -0.5751344    0.49121 0.42204    7.9e-10  0      0.06       0.06      0.17
...
 4.9531     -0.5766627    0.50137 0.42187    1.3e-08  0.01   0.03       0.04     14.54
```

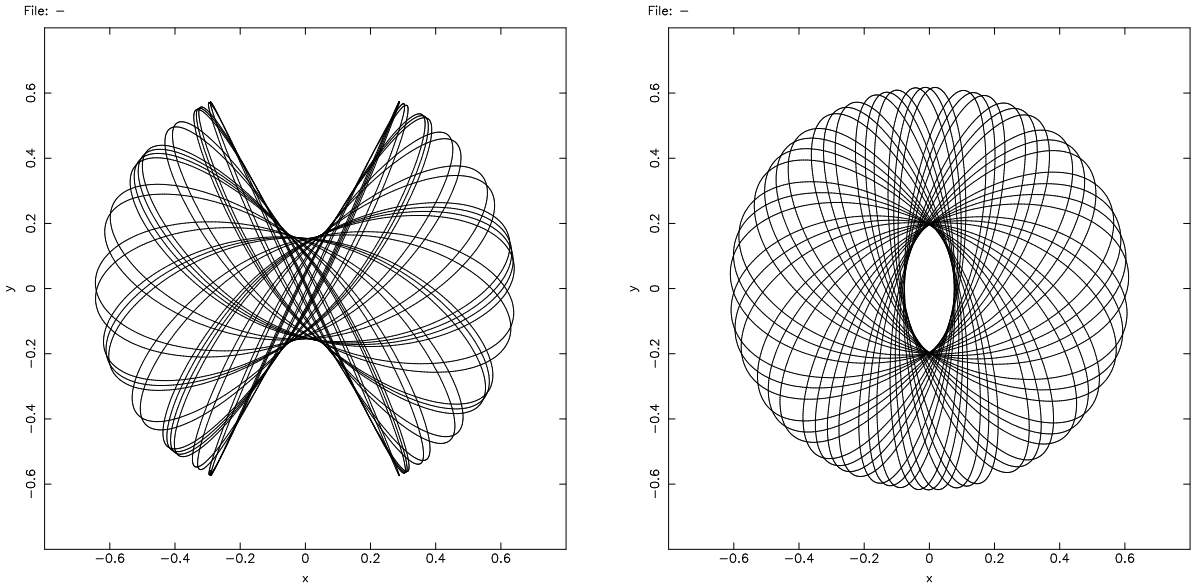

Figure G.2: Orbits in a Logarithmic potential: Initial conditions y=0.1 result in a box orbit (left), y=0.2 in a loop orbit (right). See also Figure 3-7 in BT87

```
 4.9688     -0.5766588    0.50042 0.42186    1.3e-08  0      0.04       0.04     14.59
 4.9844     -0.5766436    0.49937 0.42186    1.4e-08  0.01   0.04       0.05     14.64
 5          -0.5766284    0.49822 0.42186    1.4e-08  0      0.04       0.04     14.68

   # check up on energy conservation of the code
3% snapdiagplot disk.out
6 diagnostic frames read
Worst fractional energy loss dE/E = (E_t-E_0)/E_0 = 0.00255499 at T = 5

   # nice composite diagram showing the initial, bar, and messed up state.
4% snapplot disk.out times=0,1,2,5 nxy=2,2 nxticks=3 nyticks=3

   # now combine a particle plot with a smooth contour diagram
5% snapgrid disk.out ccd1 times=2 nx=256 ny=256
6% ccdsmooth ccd1 ccd2 0.1
7% ccdplot ccd2 0.03,0.1,0.3,1,3,10      yapp=p1.ps/vps
8% snapplot disk.out times=2 psize=0.01  yapp=p2.ps/vps
9% catpgps p1.ps p2.ps > p12.ps

10% snapgrid disk.out - times=2 nx=256 ny=256 zvar=vy mom=0 | ccdsmooth - ccd3.0s 0.1
11% snapgrid disk.out - times=2 nx=256 ny=256 zvar=vy mom=1 | ccdsmooth - ccd3.1s 0.1
12% ccdmath ccd3.1s,ccd3.0s ccd3.vel %1/%2
13% ccdplot ccd3.vel -0.9:0.9:0.3 blankval=0 yapp=p3.ps/vps
14% catpgps p2.ps p3.ps > p23.ps

   # or if gif is used, left as an exercise to the reader

10% ccdplot ccd2 0.03,0.1,0.3,1,3,10          yapp=p1.gif/gif
11% snapplot disk.out times=2 color=2.0/16.0  yapp=p2.gif/gif
```

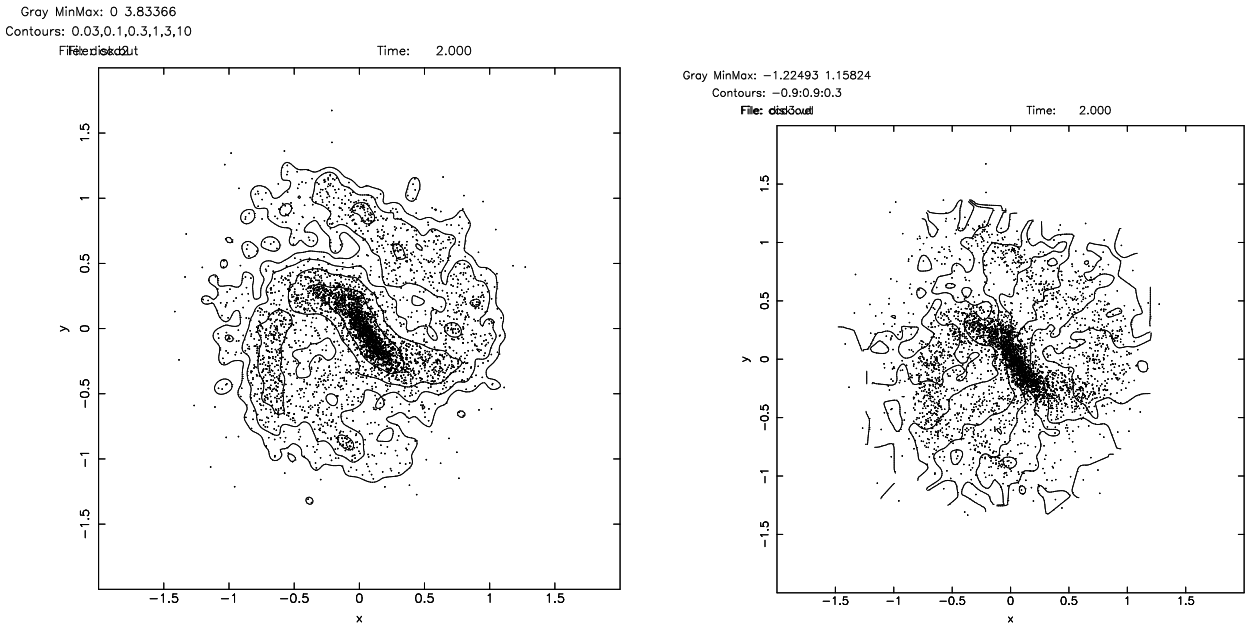

Figure G.3: the making of a Barred Galaxy. On the left the particle distribution is overlayed with a 0.1 beam smeared density distribution. On the right the overlayed contours are that of the Y velocities. Notice the S shaped contours in the bar region, due to the elliptical orbits along the bar.

```
12% See ImageMagick(1)
```

G.4 Disk-Bulge-Halo galaxy

A composite Disk-Bulge-Halo model in near equilibrium can be created using the Kuijken-Dubinski prescription. A NEMO frontend to their tools is available via the program `mkkd95`. By default it will create a disk with 8000 particles, a bulge with 4000 particles and a halo with 6000 particles. `snapplot` can be told to color particles (assuming it has been compiled with a graphics driver that knows color, e.g. pgplot)

```
 1% mkkd95 a0.dat
 2% snapplot a0.dat color='i<8000?2.0/16.0:(i<12000?3.0/16.0:4.0/16.0)' yvar=z xrange=-8:8 yrange=-8:8 yapp=/gif
```

This particular plot actually has a black background, so after replacing black with white (e.g. using Gimp or ImageMagic) and some axis cropping, the cover version is ready.

G.5 Millenium Simulation

Here we had to cheat for moment. This picture was piched from the Millenium simulation, where Volker Springel deserves all the credit! The cover picture is a small section sized 125 Mpc/h in the horizontal direction.

Appendix H

NEMO programming

NEMO is written in vanilla (mostly ANSI) C, with moderate use of macros. Most of this is written up in the *NEMO Users and Programmers Guide*. We'll just highlight a few things you might encounter if you would browse the source code:

1. nemo_main(); stdinc.h;
2. Instead of the laborious work of opening files and error checking

```
FILE *fp = fopen(argv[1],"r");
if (fp==NULL) {
  fprintf(stderr,"File %s cannot be opened\n",argv[1]);
  exit(1);
}
```

   we typically use one-liners

```
stream istr = stropen(getparam("in"),"r");
```

   which cause a fatal error in those common cases of "not found", "no permission" etc.etc. `stropen` also takes care of special files such as tt "-" and ``.''.
3. Instead of different types of `printf()` statements, use

```
error("Bad value of n=%d",n);
warning("Fixing n=%d",n);
dprintf(1,"Iteration n=%d sum=%g",n,sum);
```

   controlled with the `error=` and `debug=` system keywords.
4. Random number are normally initialized in a generic way, as follows:

```
    iseed = set_xrandom(getparam("seed"))

with the following meaning for seed:

    -2   pid
    -1   centi-seconds since boot
    0    seconds since 1970.0
    n>0  your personal seed selection
```

there are controlled with the `seed=` program keyword.

H.1 Creating a new program

There are roughly three ways one can decide on how to add functionality to NEMO (i.e. creating a new program):

1. Add a keyword to an existing program with nearly the same functionality. Be aware that you should then make it backwards compatible, i.e. the default value for the keyword should reflect the way it was working before to prevent surprising the old-timers. Also, don't forget to add this new keyword to the manual page (NEMO/man/man1 typically)
2. Clone an existing program, and give it a nice new name. This means you'll also have to clone the manual page (NEMO/man/man1 typically), and perhaps cross-references (in the SEE ALSO section) of a few other manual pages. In most cases you may have to add an entry to the Makefile, at least in the BINFILES= macro, to ensure it gets installed. Sometimes special rules are needed for binaries. If you want the program to be part of the standard distribution, add an entry to the `Testfile`, so it gets built when NEMO is installed and runs the testsuite. Also test, from `$NEMO`, if `mknemo` does not get confused about naming conventions before the program is committed to CVS.
3. Write a whole new program. You can either use `NEMO/src/scripts/template`, and fill in the blanks, or write it from scratch. See also previous comments on Makefile, Testfile and mknemo.

```
1% $NEMO/src/scripts/template foo a=1 b=2 n=10
2% bake foo
3% foo c=1
### Fatal error [foo]: Parameter "c" unknown
```

Appendix I

Maintenance and Updates

Updates to the code can either be done via releases (tar files or tagged CVS) or in a "continuum" (CVS).

I.1 NEMO (version 3.2.3)

After installation, it can occur that it appears as if you have a missing program in NEMO, despite that a manual page exists.

There are 3 quick and easy ways to install/update a program

1. The program just needs to be compiled, or you modified it,and needs to be placed back in the system: `mknemo program`
2. The program was updated at the master site, so you need to update your CVS repository, and recompile it: `mknemo -u program`
3. A number of library routines were updated, and therefore the program should be recompiled and/or relinked: `mknemo -l -u program`

I.2 STARLAB (version 4.2.2)

be sure to update this: since the last summerschool starlab install was overhauled

Similar to NEMO there is a script `mkstarlab` available which makes it relatively easy to update or recompile the code under most circumstances.

1. Just (re)compile the program: `mkstarlab program`

2. Fetch a new version via CVS and recompile a program: `mkstarlab -u program`
3. Update the code, recompile the library and the recompile the program: `mkstarlab -l -u program`

I.3 Other

...not yet...

Appendix J

Libraries

Appendix K

Troubleshooting

Compiling `manybody` can be a chore, since it contains many packages and modules, each of which can have some dependancies on the system. If you are running linux, one of the most common things not installed is the development libraries for X windows and perhaps a few other libraries. For RH9 this would be `XFree86-devel`, for FC4 `xorg-devel`. Mesa is another often missed package (needed for partiview).

K.1 Known Problems/ToDo's

1. The default shell should be (t)csh, there is some limited (ba)sh support.
2. nbody4 is now available, though may be called Brute4 (Nbody4 needs the GRAPE hardware and Grape libraries)
3. `kira --help` sometimes will complain "source file unavailable"
4. The `gman` command could be be missing on your machine. It's very nice to have though!
5. Q=0 for runbody1 when in= is used... bug in the wrapper interface
6. `dtos` suffers from some quick hacks put in for the AAS NVO demo: Aux/Acceleration/Potential are meaningless.
7. `snapplot` multipanel has odd colors (grey) showing up
8. fix MANPATH for NEMO under linux
9. `tkrun` demos / python?
10. overview diagrams of the hierarchy and layout of NEMO, Starlab, Manybody
11. add Kawaii's code
12. units(1) and units(5) unfinished

13. (i)python, matplotlib and various interfaces for NEMO?
14. Fink/DarwinPorts
15. NAM (Shaya/Peel)
16. Romeo's wavelet code (has some GPL issues)
17. SuperBox
18. starlab has some installation problems some compilers (e.g. on FC5, but mdk10 ok) - related to their internal autoconf version?
19. Greengard & Rochlin's FMM Multipole Moment Code (simple C version via Umiacs)
20. single vs. double precision: gadget can actually be compiled in double precision mode, but the gadgetsnap/snapgadget programs are likely to break. gyrfalcON is mostly compiled in double mode in NEMO, but Walter will most likely prefer float.
21. (gyr)falcON may need to be re-compiled without the `-ffast-math` option (see `OPTFLAGS` in `make/defs`). Otherwise it has the tendency to crash. (e.g. AMD64 with gcc 3.4.5 and intel with gcc 4.1.0)
22. intel compiler can speed up a lot
23. 64 bit issues: one of the current problems with fortran is the switch from g77 to gfortran. The former cannot handle many 64 bit issues, where gfortran is able to do so. gfortran is however more ANSI strict in language features, but does handle (some) fortran-95 . To make things more confuising, your system may also contain `g95`. If you decide to compile NEMO with gfortran, make sure no g77 is used where data is transformed with `unfio`, as the header size is 4 in g77. There are also compiler issues. Sometimes gfortran is faster (magalie improved from 49s to 22s, but mkkd95 ran slower, from 38s to 56s!).

K.2 Unknown Problems

This section merely exists to not confuse Piet, and is otherwise left to your investigation.