総研大集中講義 N 体計算実習資料

牧野淳一郎

September 7, 2009

Abstract

この文書は 2009/9/7 の総研大講義「シミュレーション天文学」に附属する実習のための資料です。

Contents

1 作成状況

1.1 2009/8/18

ソフトウェアのインストールと動作チェックを兼ねたいくつかのコマンドの紹介まで書いた。

1.2 2009/8/20

glnemo2 に関する記述を Troubleshooting に移動。

1.3 2009/8/24

MacOS X で動作確認。2 章の構成変更。

2 準備

この章では、実習当日までにしておくべき準備について述べます。。これができていないと当日の実習は無意味になるので、必ずやっておいて下さい。

この実習では、基本的に受講生が自分で持参した計算機の上で作業します。このため、使用するソフトウェアをあらかじめインストールする必要があります。以下、 OS が Windows (Vista では問題があるかもしれません)、Linux、Mac OS X のそれぞれについて、インストールするべきソフトウェアとその作業手順を述べます。

一応、CfCA の計算機上でも必要なソフトウェアが実行可能 (になる予定) ですが、ネットワークごしになるので可視化ツール等はあまりスムースに動かないです。

インストールするべきソフトウェアは、「nemo」です。 Nemo はメリーランド大学の Peter Teuben が 20 年以上にわたって開発・メンテナンスをしている、主に無衝突 N 体系用 (ですが、衝突系にも使えるし SPH 等へのサポートもある) のシミュレーションをサポートする様々なツールの複合体です。沢山のプログラムが共通のユーザーインターフェースやファイル形式をもつことで、色々便利なことができるようになっています。

nemo のインストールには、グラフィックパッケージ pgplot のインストールが必要ですが、これは nemo のインストールスクリプトの中で自動的に行われるはずなので、上手くいっていればこれは気にする必要はありません。また、今回の実習では

- glnemo
- GalacTICS
- WIP

といったツールを使いますが、これらも以下の手順に従えば自動的にインストールされるはずです。

glnemo はマルセイユ天文台の Jean-Charles Lambert が開発している、nemo のファイル用に作られた 3 次元可視化プログラムです。Linux(Qt が動く UNIX 全般) と Windows の両方で動きますが、まだ色々開発中という感じもあります。が、スナップショットファイルの 3 次元表示や簡単なアニメーション作成には十分に使え、ズーム、回転等を非常に高速に行うことができます。

GalacTICS はバルジ・ディスク・ハローからなる銀河モデルの作成パッケージです。WIP は PGPLOT をインタラクティブに利用するためのプログラムです。

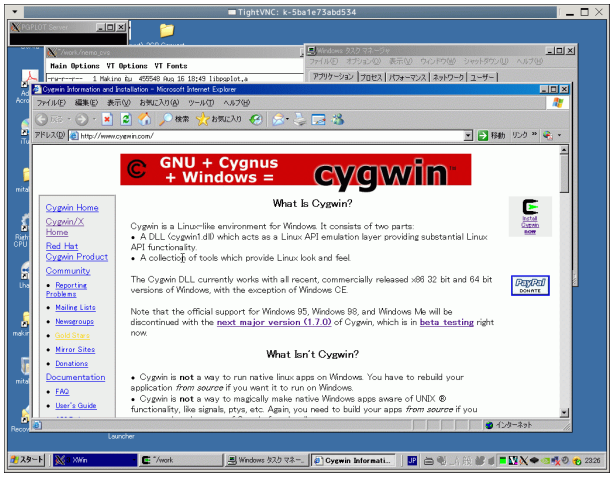

Figure 1: Cygwin のトップページ

2.1 nemo インストールの前にすること

nemo インストールのためにには、OS によって、いくつか別の用意が必要になります。ここではまずそれらについて述べます。

- 2.1.1 Windows の場合
- 2.1.2 Linux の場合
- 2.1.3 Mac OS X の場合

2.1.1 Windows の場合

Windows の場合、Cygwin の利用が前提になるので、Cygwin のインストールについて述べます。なお、X11 およびコンパイラ等がはいっていることが必要なので、Cygwin を既につかっている、という人の場合でも、setup.exe を実行してフルインストールの状態にして下さい。

既に Cygwin を使っていて、X Windows もコンパイラも入っている、という人は 2.2 節に進んで下さい。そうでない人は以下の手順で Cygwin をインストールして下さい。既に Cygwin の基本部分がはいっていて追加インストールする場合でも、以下の手順で問題ありません。

Windows 上の適当なブラウザで以下を実行します。まず、図 1 にある *Cygwin* のトップページ[1]から、

setup.exe [2]をダウンロード (これは実行して OK) します。不明な発行者がなんたらとかでますがかまわずに「実行する」を選択します。

Choose installation type ではデフォルトになっているインターネットからを選択し、

Choose Installation Directory では、デフォルトの C:\cygwin で問題なければそのまま、例えば C ドライブが一杯で別のところにおきたい、という場合にはそのディレクトリ名をいれます。例では D:\cygwin にしています。Install For とか Default Text File Type はデフォルトのままにしておきましょう。

次にでる Select Local Package Directory でも、デフォルトはデスクトップですが、C ドライブに空きがないとかいった事情があれば別のところを指定しましょう。例では変更しています。

[1] `http://www.cygwin.com`
[2] `http://www.cygwin.com/setup.exe`

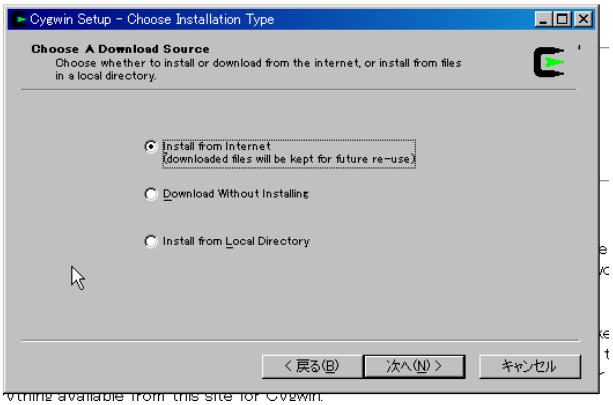

Figure 2: Choose Installation Type のウインドウ

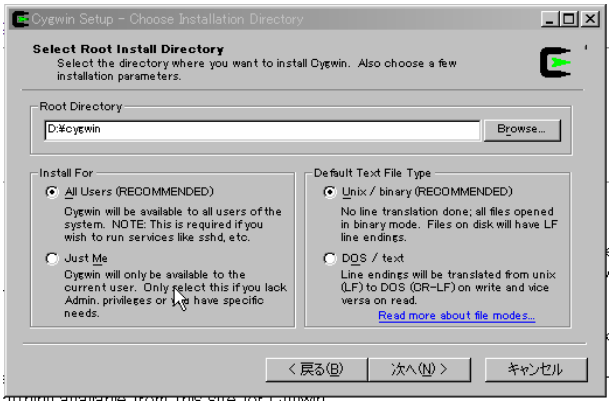

Figure 3: Choose Installation Directory のウインドウ

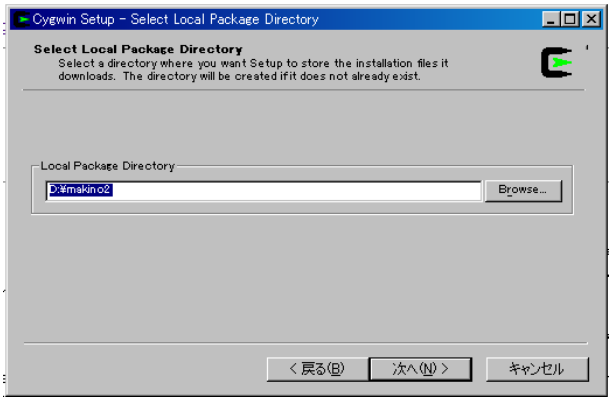

Figure 4: Select Local Package Directory のウインドウ

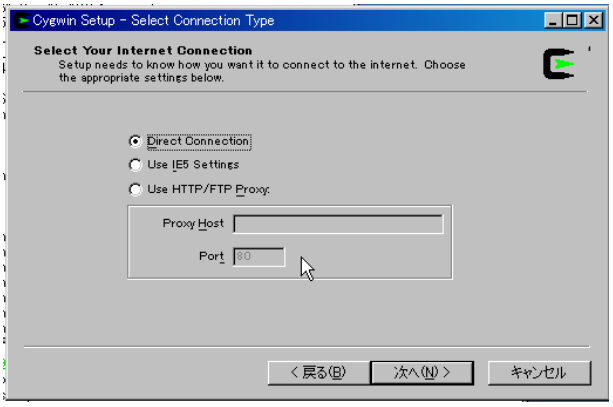

Figure 5: Select Connection Type のウインドウ

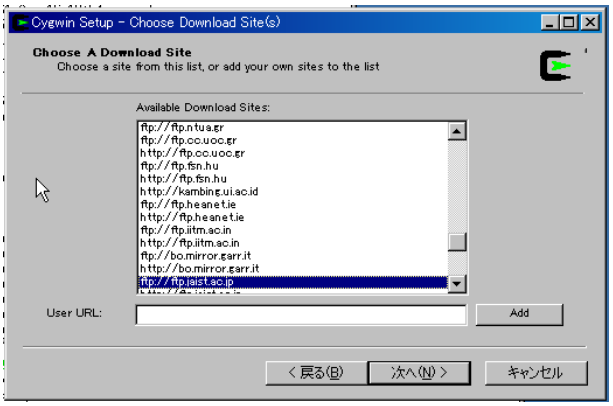

Figure 6: Choose Dounload Site(s) のウインドウ

その次の Select Connection Type は、特殊なネットワーク接続になっていなければデフォルトの Direct Connection でよいはずです。

しばらくたつと、 Choose Dounload Site(s) になります。ここでは、国内のどこかを選びましょう。

このあとまたしばらく待つことになると思いますが、順調にいっていれば Select Packages の画面になります。

ここで、「All」の横にある丸に矢印をクリックして、「Default」になっているのを「Install」に変更します。

これで、無事にインストールが終わると、Installation Status and Create Icons の画面がでます。これは起動方法なので自分の趣味に合わせて下さい。とりあえず、両方チェックしておくのが簡単です。

これで Cygwin のインストールは終了ですが、次に、ちゃんとはいっているかどうかの確認が必要です。スタートアップメニューにいれるオプションを選んでいたなら、「XWin Server」がメニューにあるのでそれをクリックして下さい。上手くはいっていれば、ターミナルウインドウがでるはずです。

これが出れば、 X を含めて正しくはいっているはずです。

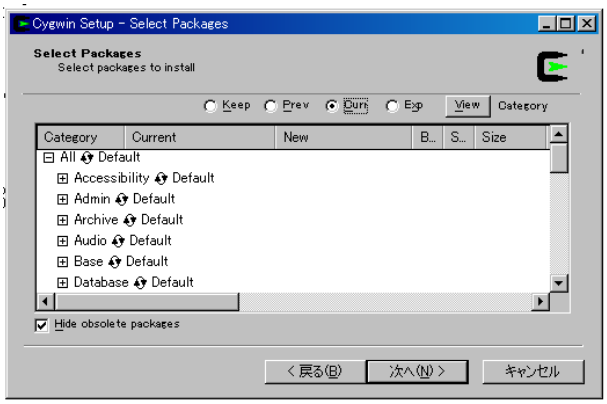

Figure 7: Select Packages の初期ウインドウ

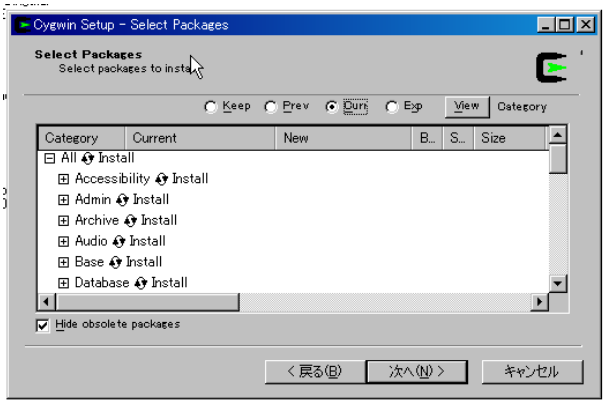

Figure 8: Select Packages の変更後ウインドウ

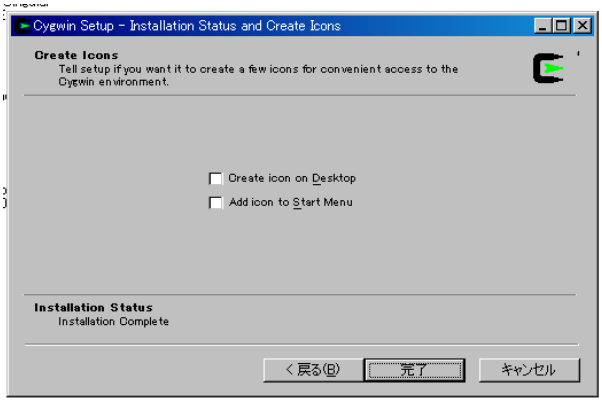

Figure 9: Installation Status and Create Icons ウインドウ

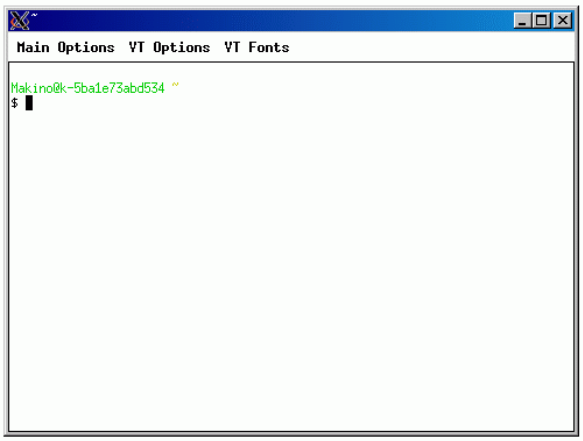

Figure 10: Cygwin の terminal ウインドウ

2.1.1.1 X が動かない時

Cygwin 自体は正しく入っても、 X が動かない時があります。この時には、Cygwin の X サーバーではないものをいれるとそっちは動くこともあります。*Xming* のサイト[3] から、*Xming-fonts*[4] をまずインストールして、それから *Xming*[5] をインストールしましょう。これらはでる画面に従っていけばなんとかなると思います。もしも、上の、 Xming, Xming-fonts へのリンクが壊れていたら、*Xming* のサイト[6] にあるリンクを辿って下さい。

この時は、まず Xming Xserver を起動し、それから Cygwin bash shell を起動して、その中から Xterm 等を起動する必要があります。環境変数 DISPLAY に 127.0.0.1:0.0 を設定しないと上手く動かないかもしれないので注意して下さい。

2.1.2 Linux の場合

Linux の場合も、コンパイラ、 X11 の開発環境といった、デフォルトでははいっていないかもしれないものをインストールする必要があるかもしれません。必要なパッケージ等はディストリビューションによるので、ここで詳細は書きません。追加インストールで関係ありそうなものをいれるとか、 apt-get や yum で入れるとかして下さい。例えば、CentOS の場合

```
yum install gcc compat-gcc-34-g77
yum install qt qt-devel
```

といった辺りだと思います。

2.1.3 Mac OS X の場合

Mac OS X の場合、以下のものをいれる必要があります。

- Xcode
- g77

[3] http://www.straightrunning.com/XmingNotes/
[4] http://sourceforge.net/project/downloading.php?group_id=156984&filename=Xming-fonts-7-4-0-3-setup.exe
[5] http://sourceforge.net/project/downloading.php?group_id=156984&filename=Xming-6-9-0-31-setup.exe
[6] http://www.straightrunning.com/XmingNotes/

- qt3
- fink (qt3 インストールのために必要)

多くの方はすでにこれらのものをインストールしているかもしれません。

以下、手順を簡単に説明します。

まず、Apple の開発環境 Xcode をいれます (いれていない人の場合)。これは、OS の DVD にはいっているので、OS の DVD をいれて XCode とかそういう文字列のほうで何かします。OS の DVD が見つからなくても、*Apple* のサイト[7] でユーザ登録すれば XCodeTools なんとかというのをダウンロード、インストールすることができるはずです。これをいれると gcc 4.0 が入るはずです。 g77 ははいりません。

g77 が入っていない人の場合、 Intel Mac であれば、*HPC Mac OS X*[8]のページから、ここに従って[9] 入れるのが簡単そうです。*g77-intel-bin.tar-gz (Intel cpu* の場合*)*[10]をダウンロードして、それがあるディレクトリで

```
sudo tar -xvf g77-bin.tar -C /
```

を実行するだけです。今回の関係は g77 でしかテストしていませんので、gfortran 等ではコンパイルできません。g77 をいれて下さい。なお、この手順でいれた g77 が動作するためには、 gcc が Xcode 由来のもの (fink 等でいれたものではないもの) である必要があるようです。 fink でいれた gcc をお使いの方は、なんとかして fink で g77 をいれる (すみません、牧野にはやり方が今のところわかっていません) か、ソースから make して下さい。

PowerMac の場合、 fink で g77 が入ります。

qt3 も、普通ははいっていないかもしれません。 fink をいれていれば、これは

```
fink install qt3
```

で入るはずです。 qt3 は可視化ツール glnemo のコンパイルに必要になります。これがコンパイルできるためには、 環境変数 QTDIR に fink で色々なものをいれるパスである /sw を設定しておく必要があります。また、 fink は /sw の下に bin ディレクトリを作るので、環境変数 PATH に /sw/bin を追加して下さい。

csh の場合:

```
setenv QTDIR /sw
setenv PATH ${PATH}:/sw/bin
```

bash 等の場合

```
QTDIR=/sw ; export QTDIR
PATH=${PATH}:/sw/bin ; export PATH
```

まだ fink をいれていなければ *Fink* のダウンロード[11]ページの記述に従ってまず fink をインストールします。この場合、 .profile に勝って fink installer がなんか書くので、それをやっていれば PATH の設定は必要ないはずです。

[7] http://connect.apple.com
[8] http://hpc.sourceforge.net/
[9] http://www006.upp.so-net.ne.jp/mnak/computer/macosx.html
[10] http://prdownloads.sourceforge.net/hpc/g77-intel-bin.tar.gz?download
[11] http://www.finkproject.org/download/index.php?phpLang=ja

2.2 nemo のインストール

2009/8/16 現在の状況では、 *nemo*[12] はソースプログラムの数箇所に変更を加えないと Cygwin にインストールできません。変更したファイル等のコピーから、インストールまでをまとめたスクリプトを牧野が作ったので、これを使ってみることにします。

まず、nemo をインストールするディレクトリを作ります。これは/home/yourid/work (yourid は Cygwin での自分の user id) とか、適当なものを作って下さい。ターミナルウインドウで、そのディレクトリに移動し、wget 等を使って *http://grape.mtk.nao.ac.jp/pub/people/makino/tmp/nemo_cygwin_install.csh*[13] をコピーします。

```
%cd
%mkdir work
%cd work
%wget http://grape.mtk.nao.ac.jp/pub/people/makino/tmp/nemo_cygwin_install.csh
```

MacOS X の場合、 wget がはいっていないようなので、代わりに curl を使います。

```
%cd
%mkdir work
%cd work
%curl -O  http://grape.mtk.nao.ac.jp/pub/people/makino/tmp/nemo_cygwin_install.csh
```

このファイルの中身はこんな感じのものです

```
#!/bin/csh -f
#
# nemo_cygwin_install.csh
#
# J. Makino
# 2009/8/16 created
# 2009/8/18 added hack for qmake on Cygwin
# 2009/8/20 Major rewrite: Try not to download/compile if files are already
#           there
# 2009/8/24 Try curl if wget failed
# 2009/8/25 Hacks to work on OSX 10.4 and 10.5
#
set hostsys = `uname`

# First get the source from Peter's standard location
if ( ! ( -e nemo_cvs  ) ) then
    wget  ftp://ftp.astro.umd.edu/pub/nemo/nemo_cvs.tar.gz
    if ( $status == 1) then
      curl -O     ftp://ftp.astro.umd.edu/pub/nemo/nemo_cvs.tar.gz
    endif
    tar zxf nemo_cvs.tar.gz
    cd nemo_cvs
    cvs login ; cvs update -d          # just to be sure to be up to date
else
    cd nemo_cvs
```

[12] http://bima.astro.umd.edu/nemo/

[13] http://grape.mtk.nao.ac.jp/pub/people/makino/tmp/nemo_cygwin_install.csh

```
endif

# download wip as well
if ( ! ( -e wip) ) then
  wget ftp://ftp.astro.umd.edu/progs/morgan/wip2p3.tar.gz
  if ( $status == 1) then
     curl -O  ftp://ftp.astro.umd.edu/progs/morgan/wip2p3.tar.gz
  endif
  tar xzf wip2p3.tar.gz
endif
# Then get the patched files from Makino's location
wget http://grape.mtk.nao.ac.jp/pub/people/makino/tmp/nemo+pgplot.patch.tgz
if ( $status == 1) then
    curl -O  http://grape.mtk.nao.ac.jp/pub/people/makino/tmp/nemo+pgplot.patch.tgz
endif
tar xvzf nemo+pgplot.patch.tgz
if ( ( -e bin/snapplot ) ||  ( -e bin/snapplot.exe ) ) then
   echo You seem to have nemo already
   echo so skip the installation.
   echo If you want to re-install nemo,
   echo either remove or rename nemo_cvs and try to run this
   echo script again.
   source nemo_start
else


   switch ($hostsys)
# Cygwin does not have /usr/bin/qmake, even after qt3-devel is installed.
# So make a soft link
      case CYGWIN*
         echo This system is Cygwin. prepare /usr/bin/qmake
         if  ( ! ( -e  /usr/bin/qmake)  ) then
            if ( -e  /usr/lib/qt3/bin/qmake )  then
               ln -s /usr/lib/qt3/bin/qmake /usr/bin/qmake
            else
               echo cannot find qmake. Please try some other fix.
            endif
         else
            echo qmake is already in /usr/bin
         endif
      breaksw
      case Darwin*
         echo This system is Mac OS X. Additional setup for QT
         setenv  PATH ${PATH}:/sw/bin
         setenv QTDIR /sw
      breaksw
   default:
      echo none approptiate
   endsw


#and then run the standard script
```

```
   source AAA_SOURCE_ME
#and try to fix the problem with powerpc macs (or OS X 10.4)
# not sure if this problem is due to PPC implementation or OS X version.
   set nemohost = $NEMOHOST
   switch ($nemohost)
      case powerpc-apple*
         echo This system is PowerMac. Things have probably badly failed.
         echo Now try to edit lib/makedef and make again
         if ( ! ( -e lib/makedefs.org ) ) then
             cp -p lib/makedefs   lib/makedefs.org
         endif
         sed -e 's/-lcrt2.o/ /g' <  lib/makedefs.org >   lib/makedefs
         make libs
      breaksw
   default:
      echo Nothing to do here
   endsw
endif

# and then make usual binaries
make bins
# make sure that GalacTICS and glnemo are there
mknemo mkkd95
mknemo glnemo


if ( ( -e bin/wip ) ||  ( -e bin/wip.exe ) ) then
   echo WIP is already installed
else
   echo now compile wip
   cd wip
   switch ($hostsys)
      case Darwin*
        ./makewip -host darwin -pgplot $NEMOLIB -xlib /usr/X11R6/lib
      breaksw
   default:
     ./makewip -host linux -pgplot $NEMOLIB -xlib /usr/X11R6/lib
   endsw
   switch ($hostsys)
      case CYGWIN*
        mv wip.exe $NEMOBIN
      breaksw
   default:
      mv wip $NEMOBIN
   endsw
endif
```

最初の、 cvs update -d までは nemo のサイトに書いてある通りです。
実行には、ターミナルウインドウで、

```
 $ csh -f   nemo_cygwin_install.csh
```

とします。すると

```
$ csh -f   nemo_cygwin_install.csh
--2009-08-17 01:08:42--  ftp://ftp.astro.umd.edu/pub/nemo/nemo_cvs.tar.gz
           => ‘nemo_cvs.tar.gz’
Resolving ftp.astro.umd.edu... 129.2.14.3
Connecting to ftp.astro.umd.edu|129.2.14.3|:21... connected.
Logging in as anonymous ... Logged in!
==> SYST ... done.    ==> PWD ... done.
==> TYPE I ... done.  ==> CWD /pub/nemo ... done.
==> SIZE nemo_cvs.tar.gz ... 10949723
==> PASV ... done.    ==> RETR nemo_cvs.tar.gz ... done.
Length: 10949723 (10M)

100%[==============================================>] 10,949,723   250K/s   in 48s

2009-08-17 01:09:33 (222 KB/s) - ‘nemo_cvs.tar.gz’ saved [10949723]

Logging in to :pserver:anonymous@cvs.astro.umd.edu:2401/home/cvsroot
CVS password:
```

と、パスワードをいれて、といってくるので、単にリターンをいれます。すると、

```
? .cvspass
? VERSION_cvs
? local
cvs update: Updating .
cvs update: Updating bugs
cvs update: Updating csh
cvs update: Updating data
cvs update: Updating data/GalPot
cvs update: Updating data/filter

(途中省略)

cvs update: Updating usr/vogl/src/msfort
cvs update: Updating usr/vogl/src/sunfort
```

といった感じに延々ファイルの更新をした後に

```
--2009-08-17 01:15:27--  http://grape.mtk.nao.ac.jp/pub/people/makino/tmp/nemo+pgplot.patch.tgz
Resolving grape.mtk.nao.ac.jp... 133.40.7.128
Connecting to grape.mtk.nao.ac.jp|133.40.7.128|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 20361 (20K) [application/x-tar]
Saving to: ‘nemo+pgplot.patch.tgz’

100%[==============================================>] 20,361      --.-K/s   in 0.05s

2009-08-17 01:15:27 (441 KB/s) - ‘nemo+pgplot.patch.tgz’ saved [20361/20361]
```

```
inc/nr.h
inc/mathfns.h
src/kernel/loadobj/loadobjCYGWIN.c
src/kernel/loadobj/loadobj.c
src/nbody/cores/bodytrans.c
src/scripts/pgplot.install
```

という感じのメッセージがでます。ここで、牧野が用意したファイルでいくつかのファイルを置き換えています。その後に、 nemo のコンパイルが始まります。

```
(Sending output to install.log)
Found a local pgplot, Assuming we're using it
mkdir: cannot create directory '/tmp/NEMO_WAS_HERE': File exists
Your filesystem does not support multi-case unique filenames (like A and a)
A real Unix depends on it...
You probably have a Mac, or use Windows, so, good luck to you
rmdir: failed to remove '/tmp/NEMO_WAS_HERE': Not a directory
./configure --with-yapp=pgplot --with-pgplot-prefix=/home/Makino/work2/nemo_cvs/lib
```

これはとても長い時間がかかります。コンパイルの作業状況はこのディレクトリの下にできた nemo_cvs というディレクトリの install.log というファイルに書かれていくので、端末画面をもうひとつ出すとかして、

```
tail -f ~/work/nemo_cvs/install.log
```

で作業状況を確認しましょう。元々のスクリプトを実行した画面では

```
Initializing NEMO environment
Postconfig
Building NEMO library
Building html files for all the manual pages
Building some sample executables by running the testsuite with -b:
   You can later build all executables with "cd $NEMO;make bins"
   or using "mknemo <program-name>" on a per-case basis
    Since you also seem to have a CVS structure (very good!)
    you can also update code directly via CVS, using the -u flag to mknemo:
                mknemo -u <programname>
or
                mknemo -u -l <programname>
if the library also needs to be rebuild.
Compiling falcON and its tools.
Note: Although your current shell now has the NEMO environment loaded,
      new shells will not have NEMO pre-loaded. You would need to add
      the following command/alias to your .cshrc (or equivalent) file:
                  source /home/Makino/work2/nemo_cvs/nemo_start
Found 130 executables (132 at last count)
Found 12/30 problems with the TESTSUITE
TESTSUITE: /home/Makino/work2/nemo_cvs/src/nbody/cores Problems
TESTSUITE: /home/Makino/work2/nemo_cvs/src/nbody/evolve/aarseth/nbody0 Problems
TESTSUITE: /home/Makino/work2/nemo_cvs/src/nbody/evolve/aarseth/tools Problems
TESTSUITE: /home/Makino/work2/nemo_cvs/src/nbody/evolve/dehnen Problems
TESTSUITE: /home/Makino/work2/nemo_cvs/src/nbody/evolve/flowcode Problems
```

```
TESTSUITE: /home/Makino/work2/nemo_cvs/src/nbody/evolve/scfm Problems
TESTSUITE: /home/Makino/work2/nemo_cvs/src/nbody/init Problems
TESTSUITE: /home/Makino/work2/nemo_cvs/src/nbody/io Problems
TESTSUITE: /home/Makino/work2/nemo_cvs/src/nbody/io_nemo Problems
TESTSUITE: /home/Makino/work2/nemo_cvs/src/nbody/reduc Problems
TESTSUITE: /home/Makino/work2/nemo_cvs/src/orbit/misc Problems
TESTSUITE: /home/Makino/work2/nemo_cvs/src/orbit/potential Problems
All done.
Starting at Mon Aug 17 01:46:12 2009, tail -f /home/Makino/work2/nemo_cvs/install.log
cd src;make -i bins >> ../install.log 2>&1
cd usr;make -i bins >> ../install.log 2>&1
Done at Mon Aug 17 01:52:32 2009
```

というような感じで時々出力がでます。

2.3 簡単な動作確認

ここでは、nemo の基本的な概念の学習を兼ねて、簡単な動作確認をします。
そのために、

- mkplummer コマンドで銀河モデルを作り
- tsf および snappprint コマンドでファイルの中身 (数字) をみて
- snapplot コマンドでグラフィック表示し
- glnemo/glnemo2 コマンドで 3 次元グラフィック表示もやってみる

ということをします。

2.3.1 nemo の基本機能の確認

まず、 nemo のコマンドが使えるような環境設定をする必要があります。これには、色々な環境変数やパスを設定する必要がありますが。 csh 系のシェルに対しては nemo_start というスクリプトが用意されているので、

```
% source ~/work/nemo_cvs/nemo_start
```

で設定ができます。 bash 等 sh 系の場合にはまともなスクリプトが用意されてないような気がするので、

```
% csh
% source ~/work/nemo_cvs/nemo_start
% bash
```

(bash の場合)
としてみるのが無難です。
このあと、適当な作業ディレクトリに移動します。例えば ~/work/sandbox を使うなら、

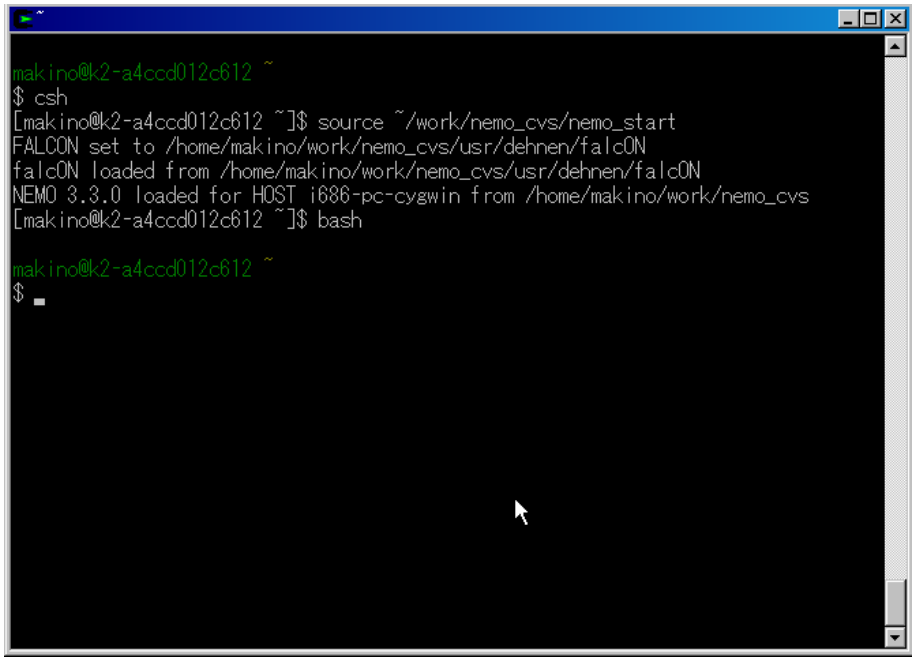

Figure 11: bash の場合の nemo の環境設定

```
% cd ~work
% mkdir sandbox
% cd sandbox
```

です。次に、以下のコマンドを実行してみます。

```
% mkplummer
```

以下のような出力がでるはずです。

```
Insufficient parameters, try 'help=', 'help=?' or 'help=h',
Usage: mkplummer out=??? nbody=??? ...
construct a Plummer model
```

mkplummer に限らず、 nemo のコマンドは基本的にコマンドラインオプションで色々なパラメータを与えます。その中には必須なものがあり、それらが与えられていない、というのが出力の最初の 2 語 "Insufficient parameters," でいいたかったことと思われます。次に try 'help=', 'help=?' or 'help=h' とあるので次にはそれをやりますが、

```
Usage: mkplummer out=??? nbody=??? ...
```

の行を見ると、 out=???, nbody=??? といったパラメータを指定しないといけなさそうだとわかります。

```
% mkplummer help=
mkplummer out=??? nbody=??? mlow=0 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= nmodel=1 VERSION=2.8b
```

mkplummer "help=?" や mkplummer help=h を実行しても、今一つ暗号めいたメッセージしかでない、思ったことでしょう。しょうがないので、

```
% man mkplummer
```

を実行してみます。途中のほうに、

```
      out=snapfile            Output data is  written  into  snapfile,  in
                              standard snapshot format.

      nbody=integer           Number  of  particles nobj in Nbody snapshot
                              realization of the Plummer model.
```

と書いてあるので、 nbody で指定した粒子数のプラマーモデルを、 out で指定したファイルに書いてくれそうです。そこで

```
% mkplummer test3 3
```

と実行してみます。なにもメッセージはでませんが、

```
% ls -l
合計 4
-rw-r--r-- 1 makino makino 427  8月 18 日 18:45 test3
```

といった感じでファイルができているはずです。このファイルの中身はバイナリなので、 cat 等で見ることはできませんが、

```
% tsf test3
char Headline[35] "init_xrandom: seed used 1250588738"
char History[31] "mkplummer test3 3 VERSION=2.8b"
set SnapShot
  set Parameters
    int Nobj 3
    double Time 0.00000
  tes
  set Particles
    int CoordSystem 66306
    double Mass[3] 0.333333 0.333333 0.333333
    double PhaseSpace[3][2][3] 0.789543 -0.771532 -0.263389 -0.220791
      -0.0611593 0.391509 -0.840700 -0.120641 0.277537 0.416235
      -0.0746009 0.219864 0.0511569 0.892173 -0.0141488 -0.195444
      0.135760 -0.611373
  tes
tes
```

で見ることができます。nemo のファイルは単なるバイナリの数字の羅列ではなく、それを作るのに実行したコマンドの履歴等のヘッダ情報と、はいっているデータの実体からなり、実体は、さらに型+名前+データ、という規則的な構造をもっていること、また、「型」に set、つまり「集合」という型を準備することで、構造型のようなものも表現可能にしていることがわかります。

また、質量 (Mass)、位置、速度 (PhaseSpace) については、それぞれを配列にしているらしい、ということもわかるかと思います。

tsf の出力はそういうわけでなかなか高級なものですが、例えば gnuplot とか mongo とか、様々なプロットパッケージで処理するにはあまり向いていません。そういう出力のためにには snapprint を使います

```
% snapprint test3
### nemo Debug Info: x y z vx vy vz
0.789543 -0.771532 -0.263389 -0.220791 -0.0611593 0.391509
-0.8407 -0.120641 0.277537 0.416235 -0.0746009 0.219864
0.0511569 0.892173 -0.0141488 -0.195444 0.13576 -0.611373

% snapprint test3 options=m,x,y,z,vx,vy,vz
### nemo Debug Info: m x y z vx vy vz
0.333333 0.789543 -0.771532 -0.263389 -0.220791 -0.0611593 0.391509
0.333333 -0.8407 -0.120641 0.277537 0.416235 -0.0746009 0.219864
0.333333 0.0511569 0.892173 -0.0141488 -0.195444 0.13576 -0.611373
```

これは、粒子一つを 1 行にまとめています。 nemo の特徴として、粒子データから計算したものを出力にできる、ということがあります。

```
% snapprint test3 options="vx*vx+vy+vy+vz*vz"
### nemo Debug Info: [bodytrans_new: invoking cc +saving .o]
### nemo Debug Info: vx*vx+vy+vy+vz*vz
0.079709
0.0723902
0.683496
```

この場合、粒子データに対して ”vx*vx+vy+vy+vz*vz” という式を評価する関数を作り、コンパイルして、自分自身にリンクした上で評価して出力する、という、なかなか気が効いたことをしています。

粒子 3 個ではつまらないので、増やしてみます。

```
% mkplummer test8k 8192
```

tsf や snapprint で見るにはデータが膨大過ぎるので、グラフィック表示してみます。

```
% snapplot test8k
```

図 12 が出れば大成功です。折角なので、少し違うことをしてみます。

なお、 Cygwin の場合 (他の OS でもあるかもしれません)、

```
%PGPLOT, PGSCI: no graphics device has been selected
```

とかいったメッセージが大量にでて、グラフィック表示はされない、ということがあるかもしれません。この時の対応は 3.2.1 節をみてみて下さい。

```
% snapplot test8k xvar=r yvar=v xrange=0:5 yrange=0:2
```

この場合、x 軸に $r = \sqrt{x^2 + y^2 + z^2}$, y 軸に v をとり、またそれぞれの軸の範囲も指定しています。半径の関数としての速度分布がなんとなくわかるプロットになっています。

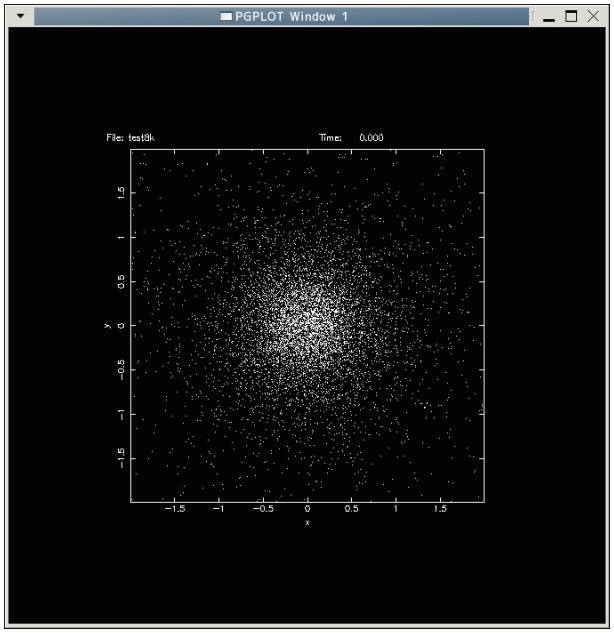

Figure 12: snapplot test8k の出力結果

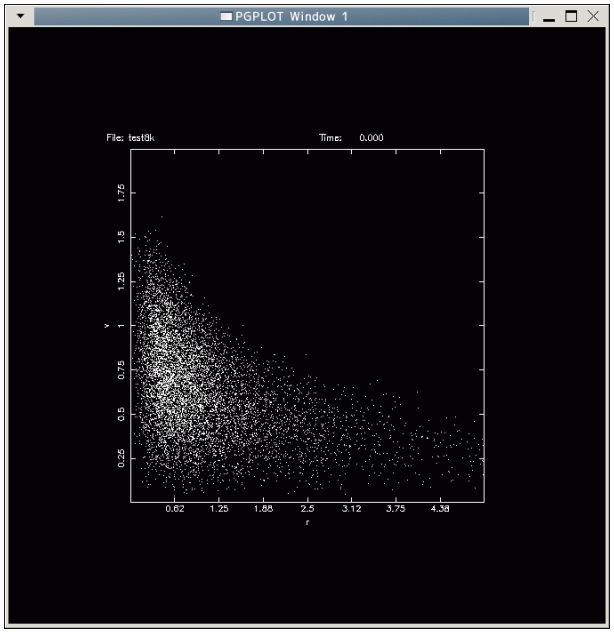

Figure 13: snapplot test8k xvar= ... の出力結果

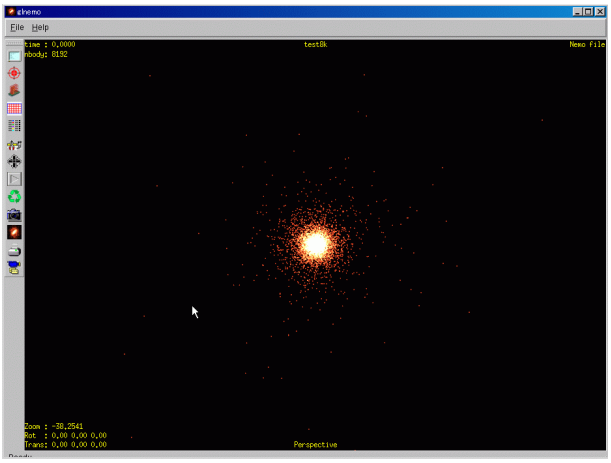

Figure 14: glnemo での test8k の表示

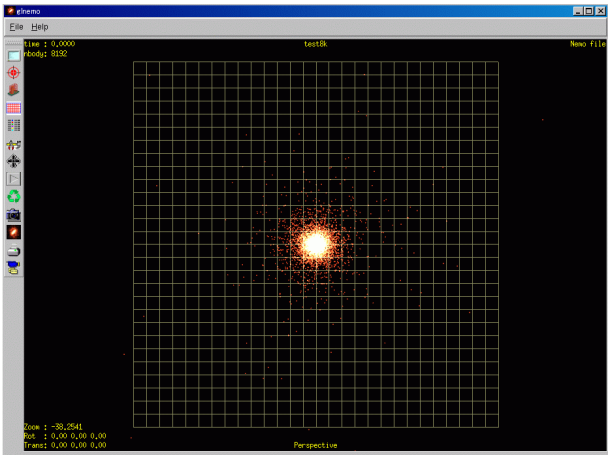

Figure 15: 格子付き表示

2.3.2 glnemo の利用

というわけで、こちらでは glnemo を使ってみます。

```
% glnemo test8k
```

画面の左の赤い格子をクリックすると格子がでます。
あとは表示画面にマウスカーソルをいれて

- 左ドラッグで回転
- マウスホイールでズーム

ができるはずです。色々やってみて下さい。
これくらいが全部動くなら、基本的な機能は大丈夫なはずです。

Figure 16: glnemo2 の画面。ファイルをオープンする途中。

2.3.3 glnemo2 の利用

ここは、Windows で glnemo2 が動いた人向けです。

さらに、 glnemo2 を使ってみます。

```
../glnemo2-win32/glnemo2
```

で、図 16 のウインドウがでます。

図のように左上の「File」メニューをクリックすると、Windows の普通のファイル選択のダイアログウインドウがでますから、 test8k を選択します。そうすると、

図 17 のウインドウがでます。ここではとりあえず OK をクリックします。

図 18 のように粒子データが表示されたら、表示ウインドウにマウスカーソルをいれると、

- 左ドラッグで回転
- マウスホイールでズーム

ができるはずです。色々やってみて下さい。

これくらいが全部動くなら、基本的な機能は大丈夫なはずです。

2.3.4 mkkd95 のテスト

nemo には Kuijken と Dubinski の作った、 GalactICS という銀河モデル作成パッケージがはいっています。

```
%mkkd95 testkd
```

を実行してみて下さい。 testkd というファイルができていれば OK です。駄目な時には、、、色々調べてみて下さい。

2009/9/3 20 時前後以前に Cygwin に nemo をインストールした場合には、mkkd95.c にバグがあってファイルができません。

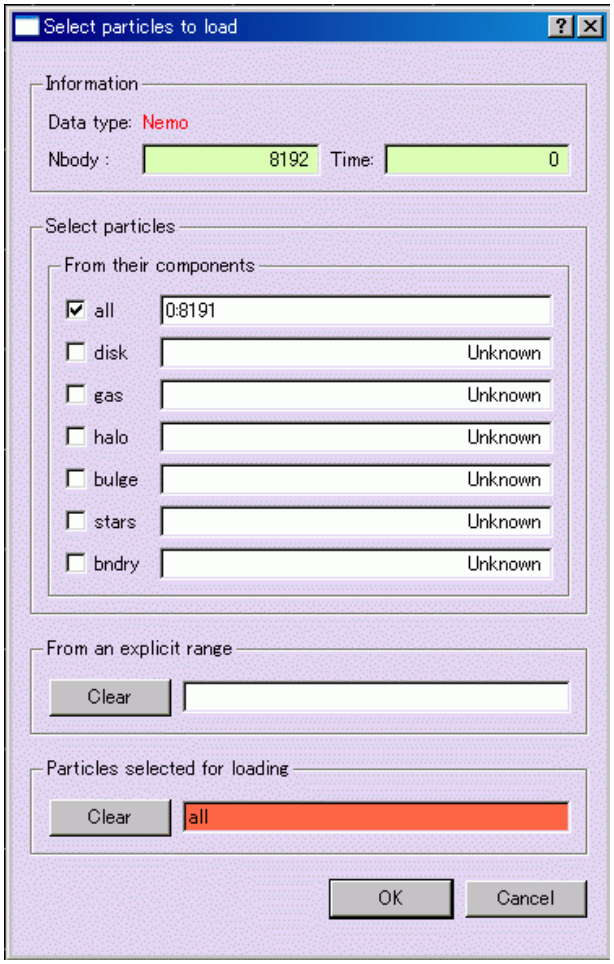

Figure 17: glnemo2 の画面。ファイル読み込みの詳細指定

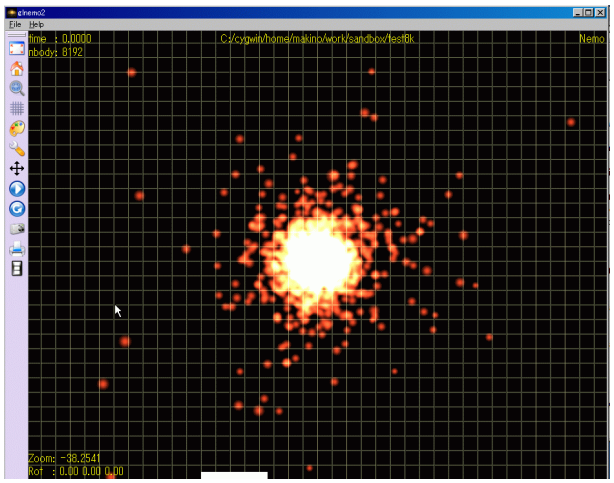

Figure 18: glnemo2 での test8k の表示

http://grape.mtk.nao.ac.jp/pub/people/makino/tmp/mkkd95.c で ∼/work/nemo_cvs/src/nbody/init/mkkd95.c を置き換えてから

```
% mknemo mkkd95
```

を実行して mkkd95 を作り直して下さい。

なお、すみません、 an03 では mkkd95 が動きません。

3 Troubleshooting

色々、トラブルが起こって上手く動かない時にどうするか、というのを、実例があったものをまとめてみます。

3.1 空きディスクが不足で Cygwin 等がインストールできない

開発環境まで含めた Cygwin は数 GB の空きディスク領域を要求するので、既にハードディスクが色々なもので一杯で、インストールできない、という人がいることと思います。

もちろん、色々努力して空きを作ればよいのですが、どうしても無理な時には、外付の USB ハードディスク等を使うのが 1 つの方法です。Choose Installation Directory、Select Local Package Directory の両方のウインドウで、そちら、例えば、ドライブ名が F: になっていれば

```
F:\CYGWIN
F:\CYGWINPACKAGES
```

といったものを指定します。必要な容量は 8GB 程度なので USB メモリでもよさそうなものですが、安物の USB メモリで実験したところでは 5 時間程度たっても Cygwin のインストールが終わりませんでした。ただ、これは、USB メモリのプロパティをなんかすることで高速化できるようです。

3.2 X ウインドウ関係

glnemo で

```
glnemo: cannot connect to X server
```

というメッセージがでたり、

snapplot で

```
%PGPLOT, PGSCI: no graphics device has been selected
```

というメッセージがでた場合、色々な可能性があります。

3.2.1 X サーバーが正常に起動しているかどうか

まず、 X server が起動しているかどうか確認して下さい。 Linux や Mac の場合には起動していないということはないと思いますが、 Cygwin の場合にはデスクトップアイコンにできた「Cygwin」やスタートアップメニューから Cygwin を起動して、シェルウインドウがでても X は起動していません。

スタートアップメニューの「すべてのプログラム」から Cygwin-X, XWin Server の順番に選択して、 X server を起動して下さい。Xming のほうをインストールした場合には、 XLaunch か Xming のアイコンかスタートアップメニューから起動して下さい。これはどちらでもよいですが、 XLaunch では細かい設定ができます。基本的にはデフォルトでよいはずなので、 Xming を起動で大丈夫なはずです。

X サーバが起動したら、画面の右下隅に X のアイコンがでるはずです。

この状態で、シェルから

```
% xterm &
```

と入力してみます。

```
xterm Xt error: Can't open display:
xterm: DISPLAY is not set
```

とでるはずです。もっとも、なぜかちゃんと xterm の画面がでるかもしれません。この場合には以下は不要です。csh 系なら

```
% setenv DISPLAY 127.0.0.1:0.0
```

bash 等なら

```
% DISPLAY=127.0.0.1:0.0; export DISPLAY
```

と入力します。そうすると、X ウインドウを使うプログラムがどこに表示するかが指定されるので、今度は

```
% xterm &
```

を実行すればウインドウがでるはずです。但し、Cygwin だとウインドウがでるのですが他のウインドウに隠れているかもしれないので、タスクバーとかで捜して下さい。

xterm が起動できたウインドウからなら、snapplot や glnemo も動くはずです。なお、ここで起動した xterm のほうに移動してしまうと、最初のほうの

```
% source ~/work/nemo_cvs/nemo_start
```

をもう一度実行する必要があります。

Unix にある程度なれている人なら、 ~/.cshrc や ~/.bash_login で上の設定や、 DISPLAY の設定もしておくとあとが楽です。

3.2.2 xterm は動くが snapplot が動かない

xterm は起動でき、 glnemo も起動できるけれど snapplot が動かなくて、相変わらず

```
%PGPLOT, PGSCI: no graphics device has been selected
```

というメッセージが大量にでる、ということがひょっとしたらあるかもしれません。これは、 PGPLOT が使っている Xwindow 用の表示プログラム pgxwin_server の起動に何故か失敗した、という場合がありえます。この時には

```
% ~/work/nemo_cvs/lib/pgxwin_server &
```

で、 pgxwin_server を起動しておくと、幸せになれるかもしれません。

また、既に pgplot がインストール済みであった場合には、そっちを使っているかもしれないので、 nemo が使うことを想定している、独自にいれた pgplot とは違うデバイスを使うようになっているかもしれません。この時には

```
% unsetenv YAPP (csh)
% unset YAPP (bash 等)
```

してから snapplot を実行すると、

```
%snapplot test8k
Graphics device/type (? to see list, default /NULL):
```

というふうに何を使うか聞いてきます。ここで、 ? をいれてリターンすると、例えば以下のようなリストがでるはずです。

```
PGPLOT v5.2.2 Copyright 1997 California Institute of Technology
Interactive devices:
   /XWINDOW   (X window window@node:display.screen/xw)
   /XSERVE    (A /XWINDOW window that persists for re-use)
Non-interactive file formats:
   /CGM       (CGM file, indexed colour selection mode)
   /CGMD      (CGM file, direct colour selection mode)
   /GIF       (Graphics Interchange Format file, landscape orientation)
   /VGIF      (Graphics Interchange Format file, portrait orientation)
   /NULL      (Null device, no output)
   /PGMF      (PGPLOT metafile)
   /PPM       (Portable Pixel Map file, landscape orientation)
   /VPPM      (Portable Pixel Map file, portrait orientation)
   /PS        (PostScript file, landscape orientation)
   /VPS       (PostScript file, portrait orientation)
   /CPS       (Colour PostScript file, landscape orientation)
   /VCPS      (Colour PostScript file, portrait orientation)
   /WD        (X Window Dump file, landscape orientation)
   /VWD       (X Window Dump file, portrait orientation)
Graphics device/type (? to see list, default /NULL):
```

でたデバイスの中で、画面にでそうなものを選んでみて下さい。

3.2.3 snapplot で、 yvar=v としたらそんなのは知らないと言われた。

nemo が、 v とかはあらかじめ知っているはずなのですが、上手くいかないケースがあります。この時は、 yvar=v の代わりに、

```
yvar="sqrt(vx*vx+vy*vy+vz*vz)"
```

としてみて下さい。このようなエラーがでる場合でも、 x, y, z, vx, vy, vz, m といった基本的な変数の組合せであれば上手くいくはずです。

3.2.4 glnemo が動かない

Linix, Cygwin, MacOS のどれでも、運がよければ glnemo はコンパイルされていますが、

```
% glnemo
```

と実行したら「そんなコマンドは知らない」的なメッセージがでるかもしれません。コマンド自体がない、つまり make に失敗している場合、色々な理由がありえるので一般的な対応は困難です。

一つのありえる理由は、 glnemo は Qt Version 3.x を想定していることです。ディストリビューションのほうがもしも Qt4 以降をデフォルトにする設定になっていたら、コンパイルできないことはありえます。

なお、 glnemo2 は 4.3 以降を想定しています。ですので、ディストリビューションが新しすぎて glnemo がコンパイルできない、といったことがもしもあれば、glnemo2 ならコンパイルできるかもしれません。但し、 glnemo2 は gcc もバージョン 4.1 以降であることを要求するようです。

また、ディストリビューションによっては、 Qt の make コマンドである qmake が、通常の実行パスのあるところのどこにもない、というようなこともあります。その時には、適切な qmake に例えば /usr/local/bin からシンボリックリンクをはってから、の、nemo が使える環境で

```
% mknemo glnemo
```

でコンパイルできるかもしれません。

牧野の手元の CentOS 5.2 のシステムでは、システムについてくる gcc 4.1.2 と Qt 3.3.6 (それぞれ、 gcc -v と qmake -v でバージョンをチェックできます) で glnemo は問題なくコンパイルされました。また、 Qt 4.5.2 をいれたら glnemo2 も問題なくコンパイルできました。

MacOS X の場合、環境変数 QTDIR が上手く設定されない等の理由で、glnemo のコンパイルに失敗することがあります。この時には

```
(csh の場合)
%setenv QTDIR /sw
%mknemo glnemo

(bash 等の場合)
%QTDIR=/sw; export QTDIR
%mknemo glnemo
```

で上手くいく可能性があります。もしも /sw/bin/qmake がないなら、 fink で qt3 をインストールして下さい:

```
fink install qt3
```

Cygwin でも、何故か失敗していることがあります。

/usr/lib/qt3/bin/qmake というファイルあることを確認して、あれば、

```
ln -s /usr/lib/qt3/bin/qmake /usr/bin/qmake
```

を実行してから

```
mknemo glnemo
```

を実行して下さい。

ここは、少し大変かもしれませんが皆様色々試してみて下さい。

3.2.5 Ubuntu で glnemo が動かない。

すみません、動かないようです。 snapplot を使って下さい。

メニューの、システム -¿ 外観の設定というので、「視覚効果」というものをオフにすると表示できるそうです。

3.2.6 glnemo2 を使う

Cygwin 環境で glnemo がどうしても動かない時には、 glnemo2 のほうは動くかもしれないのでこっちをインストールしてみましょう。

Glnemo のダウンロードページ[14]から、*Microsoft Win 32 binary*[15]をダウンロードし、展開します。これは、ブラウザのオプションで「展開する」みたいなのがあると思いますので、それを選択します。で、できたファイル glnemo2.exe をダブルクリックして実行します。そうすると、ちゃんと展開したファイルをセーブするかどうか、みたいなことを聞いてきたら、そうするようにします。そうすると、 glnemo2-win32 というディレクトリを新しく作って、その下にファイルを展開するはずです。これをもう一度ダブルクリックして実行してみて下さい。起動すればとりあえず OK です。

3.3 nemo の色々なコマンドが全然できていない

PowerMac G4 + MacOS X 10.4 だと、そもそも configure + make の基本的なやり方で失敗することがわかっています。

nemo のインストールの項で csh -f nemo...csh を実行したあとで、

nemo_cvs/lib/makedefs を以下のように編集して下さい

```
*** makedefs    Tue Aug 25 14:50:41 2009
--- makedefs.original   Tue Aug 25 14:30:17 2009
***************
*** 92 ****
! YAPPLIB = $(NEMOLIB)/yapp_pgplot.o -L/Users/makino/work/nemo_cvs/lib -lcpgplot -lpgplot   -L/usr/
--- 92 ----
! YAPPLIB = $(NEMOLIB)/yapp_pgplot.o -L/Users/makino/work/nemo_cvs/lib -lcpgplot -lpgplot   -L/usr/
```

ちょっとなんだかわからないですが、

```
  -lcrt2.o
  -L/sw/lib/gcc/powerpc-apple-darwin8.4.0/3.4.3/../../..（こっちはこのままでもいいかも）
```

を消して下さい。で、 csh で

```
cd ~/work/nemo_cvs
source nemo_start
make libs
make bins
```

のあと、もう一度

```
cd ~/work
csh -f nemo*.csh
```

[14] http://www.oamp.fr/lam/equipes/dynamique/jcl/glnemo/download.html
[15] http://www.oamp.fr/lam/equipes/dynamique/jcl/glnemo/download/glnemo2-win32.zip

を実行してみて下さい。

また、 OSX 10.5 の場合でも、同じような問題が発生することがあります。

nemo_cvs/lib/makedefs の中に -lf2c という文字列があったら、それを-lg2c に変更してから、上と同様に

```
cd ~/work/nemo_cvs
source nemo_start
make libs
make bins
```

を実行してみて下さい。

3.4 なんだか全然わからなくなった時

連絡用アドレス simulation2009 - at - grape.mtk.nao.ac.jp に質問してみて下さい。時間がある限り対応します。

4 N 体シミュレーション実習

ここからが実習本番です。準備は大変だったかもしれませんが、実際に研究に使われているパッケージですので、役に立つこともあると思います。

この実習では、

- いくつかの簡単に作成できる系の進化をシミュレーションしてみる。
- その時に、グローバルな物理量、例えばビリアル比等の進化を調べる。
- 計算精度と、その結果への影響について調べる。

といったことができるようになる、ということを目標にします。

4.1 Cold Collapse

4.1.1 前置き

Cold Collapse とは、「冷たい」、つまり、粒子の速度分散等が系のサイズを支えるだけの大きさがない粒子分布を作って、そこから時間発展をさせるとどんなことが起こるか、という問題です。もちろん、宇宙でそのままのことが起こるわけではないのですが、 現在のビッグバンモデルでは、例えば銀河といった現在ある構造は、一様な宇宙膨張から重力ゆらぎが成長していって、膨張から外れて収縮に移ってできるわけです。大雑把な理解としては、膨張から収縮に移る、ということはどこかで最大膨張になったところがあり、そこではその中の粒子は全部止まっているわけですから、 Cold Collapse というのはそういう状態のモデルであると考えることができます。

もっとも、現在の CDM 宇宙論では、小さいスケールの構造が先にできるので、銀河系のような大きなサイズのものが一気に Cold Collapse でできるわけではありませんが、それでも Cold Collapse で何が起きるか、ということを理解しておくことは宇宙における構造形成を理解する基本といえます。

4.1.2 初期条件の作成とその検証

能書きはさておき、まず初期条件を作ってみます。

nemo の中に役に立ちそうなプログラムはあるかをみます。nemo のプログラムでモデルを作るものは大体 mk なんとかという名前なので、ls してみると、、、

```
%ls $NEMOBIN/mk*
/home/makino/work/nemo_cvs/bin/mkbaredisk
/home/makino/work/nemo_cvs/bin/mkconfig
/home/makino/work/nemo_cvs/bin/mkcube
/home/makino/work/nemo_cvs/bin/mkdisk
/home/makino/work/nemo_cvs/bin/mkexpdisk
/home/makino/work/nemo_cvs/bin/mkexphot
/home/makino/work/nemo_cvs/bin/mkflowdisk
/home/makino/work/nemo_cvs/bin/mkhom
/home/makino/work/nemo_cvs/bin/mkhomsph
/home/makino/work/nemo_cvs/bin/mkkd95
/home/makino/work/nemo_cvs/bin/mknemo
/home/makino/work/nemo_cvs/bin/mkommod
/home/makino/work/nemo_cvs/bin/mkop73
/home/makino/work/nemo_cvs/bin/mkorbit
/home/makino/work/nemo_cvs/bin/mkpdoc
/home/makino/work/nemo_cvs/bin/mkplummer
/home/makino/work/nemo_cvs/bin/mkpolytrope
/home/makino/work/nemo_cvs/bin/mksphere
/home/makino/work/nemo_cvs/bin/mkspiral
/home/makino/work/nemo_cvs/bin/mktestdisk
```

やたら沢山あります。ディスクっぽいものが多いですが、ここでは mkhomsph というのを使ってみます。

```
%mkhomsph help=
mkhomsph out=??? nbody=2048 rmin=0 rmax=1.2 2t/w=0.0 vmax=0 power=0 seed=0 zerocm=t headline= VERSI
```

```
%mkhomsph test4k 4096
%glnemo test4k
```

という感じで、どんなものができるかみてましょう。

なんとなくそれらしい丸いものができている、というのはわかると思います。しかし、粒子分布は本当に一様でしょうか？これを確認するためには、中心からの距離によって粒子を並べかえて、累積質量分布を作ってみるのが一つの方法です。もちろん、半径方向の密度プロファイルでもよくて、これは単に累積質量分布を半径で微分したものですが、 N 体で作った粒子分布は離散的なので微分は簡単ではありません。

並べかえには snapsort というプログラムが使えます。どんなオプションがあるかは、ヘルプオプションや man ページで確認して下さい。

```
%snapsort test4k test4k.sort rank=r
```

で、半径方向に並べかえてくれます。次に、「半径方向の累積質量分布」を作って、これを半径をグラフにしてみたい、ということになります。

ここでは、 PGPLOT に簡易なユーザーインターフェースをつけた wip というプログラムでグラフを書き、そのためのデータは snapprint の出力を awk で加工してみます。

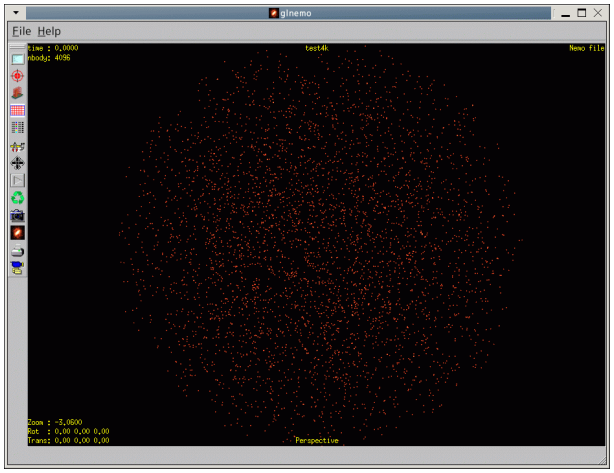

Figure 19: mkhomsph test4k 4096 でできた初期モデルを glnemo で見る。

```
%snapprint test4k.sort options=r,m | awk '{macc += $2; print macc, $0}' > tab.out
```

ruby や perl を使ってもよいですが、こういう使い方なら awk が簡単です。ここでは、ソートした test4k.sort から、粒子の位置の半径と質量が 1 行になった出力を作り、それを awk で処理します。 awk でやっていることは macc という変数に 1 行の 2 つめの数 ($2) を加算し、加算した macc と読み込んだ行全体を出力する (print macc, $0) というものです。 {} でくくった中を 1 行毎に実行する、というのが awk の基本動作で、さらに " でくくっているのはこれ全体をシェルに解釈させないで awk に渡すためです。 tab.out がどんなファイルかみてみましょう。

```
% head tab.out
0.000244141 0.0480947 0.000244141
0.000488282 0.0717042 0.000244141
0.000732423 0.120009 0.000244141
0.000976564 0.137944 0.000244141
0.00122071 0.154036 0.000244141
0.00146485 0.157324 0.000244141
0.00170899 0.163142 0.000244141
0.00195313 0.163527 0.000244141
0.00219727 0.164162 0.000244141
0.00244141 0.164364 0.000244141
```

こんな感じ (2 列目の距離の数字は乱数の初期値によって違うはず) で、最初のほうでは距離の数字が小さく

```
% tail tab.out
0.997804 1.21369 0.000244141
0.998048 1.2137 0.000244141
0.998293 1.21397 0.000244141
0.998537 1.2148 0.000244141
0.998781 1.21576 0.000244141
0.999025 1.21585 0.000244141
0.999269 1.21661 0.000244141
0.999513 1.21795 0.000244141
0.999757 1.21842 0.000244141
1 1.2211 0.000244141
```

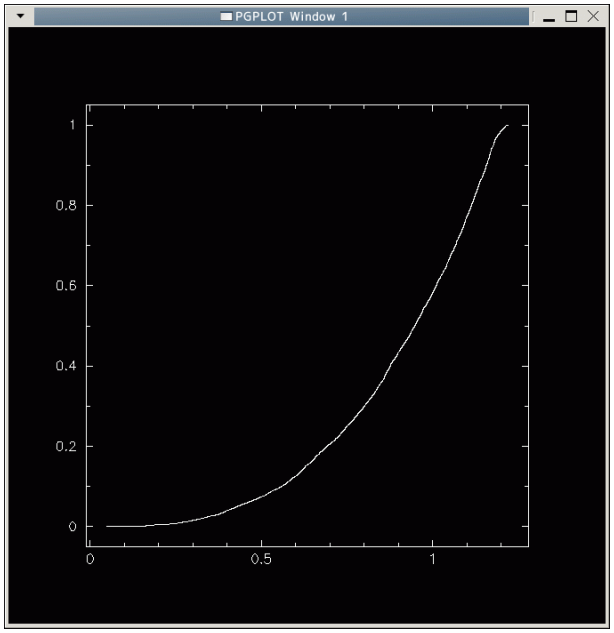

Figure 20: mkhomsph test4k 4096 でできた初期モデルの半径方向質量累積分布

最後のほうでは大きくなっていればもっともらしいです。グラフは

```
%wip
Could not open user's WIP initialization file.
Setting up default device [/XWINDOW]
Welcome to WIP Version: 2.3 22jan98

WIP> device /xs
WIP> data tab.out
WIP> xcol 2
WIP> ycol 1
WIP> limit
WIP> box
WIP> conn
```

こんな感じでコマンドをいれると図 20 のようなグラフがでるはずです。なお、 wip は end で終了します。
これは、誤差の範囲で、 $y = (x/1.2)^3$ と一致しているはずです。折角なので、これも計算してみましょう。

```
%snapprint test4k.sort options=r,m | awk '{macc += $2; rs=$1/1.2; print macc, $0, rs*rs*rs}' > tab.

wip
Could not open user's WIP initialization file.
Setting up default device [/XWINDOW]
Welcome to WIP Version: 2.3 22jan98

WIP> device /xw
WIP> data tab.out
WIP> xcol 2
```

```
WIP> ycol 1
WIP> era
WIP> limit
WIP> box
WIP> conn
WIP> ycol 4
WIP> color 2
WIP> conn
WIP> end
```

$y = (x/1.2)^3$ が 4 列目のデータになっているはずなので、それを ycol 4 で y 軸方向のデータとして読み、 color 2 で赤に変えてプロットしています。グラフは載せませんが、ちゃんとあっていることを各自確認して下さい。

人が作ったプログラムだから、自分が作ったものよりは信用できますが、だからといってチェックしないで使ってはいけません。これは、プログラムが間違っている可能性は (それまでに沢山の人が使っているものでも) 必ずあるし、また、プログラムが正しくても自分の使いかたが間違っていることもある、というかこちらは良くある、からです。

4.1.3 時間積分

さて、時間積分です。今回は、 hackcode1_qp という、 nemo に初めからはいっているプログラムを使ってみることにします。これは、Barnes-Hut ツリーアルゴリズムで有名な Barnes 本人が 1985 年頃に書いたものです。この、nemo というパッケージ自体が、その頃にプリンストン高等研究所にいた Piet Hut, Josh Barnes と、現在も開発、メンテナンスを 続けている Peter Teuben の 3 人が共同で始めたもので、構造化されたバイナリファイルや、引数で数式を与えるとコンパイルして自分自身にリンクする、といった仕掛けの最初の実装は Barnes によるものです。

と、これは余談ですが、 gadget とか、もっと速いコードを使ってもよいですが、今日の実習ではそこまではしないで、ユーザーインターフェースが簡単で安全に使える hackcode1_qp を使うことにします。例によって help をみてみます。

```
%hackcode1_qp help=
hackcode1_qp in= out= restart= continue= save= nbody=128 seed=123 cencon=false freq=32.0 eps=0.05 t
```

色々謎なパラメータがあるので、 man hackcode1_qp 、、、あれ、そんなの知らない、とでますね。 man hackcode1 でマニュアルを見ることができます。 qp がつくと 4 重極モーメントを使って、より正確に力を計算するものになっています。

試しに、入力、出力だけを指定して、あとはデフォルトでやってみましょう。

```
%hackcode1_qp test4k test4k.outsnap
```

4.1.4 エネルギー誤差のチェック

画面に一杯出力がでますが、なんだかわからないかもしれません。最後の数行は

```
      tnow        T+U        T/U      nttot      nbavg      ncavg    cputime
    2.000    -0.4579    -0.7211     796020         39        155       0.13

             cm pos     0.0007    -0.0011    -0.0015
             cm vel     0.0026    -0.0014    -0.0037
```

```
        particle data written
```

という感じのはずです。(細かい数字は違います) tnow は時刻、 T+U は系のトータルエネルギー、 T/U は運動エネルギー T とポテンシャルエネルギー U の比、いわゆるビリアル比です。力学平衡の状態ならこれは-0.5 です。その次の 3 つの数字はツリーコードでの計算量を示していて、1 ステップで粒子への力を計算した回数、1 粒子への、粒子からの力の数の平均、1 粒子への、ツリーノードからの力の数の平均、となっています。その下の cm pos, cm vel は、系全体の重心の位置と速度です。大体 0 ですが少し動いているのがわかります。

ウインドウをスクロールして上のほうに戻ると、最初のほうの出力は

```
 Hack code

        nbody        freq         eps         tol
         4096       32.00      0.0500      1.0000

         options: mass,phase

         tnow       T+U       T/U     nttot     nbavg     ncavg   cputime
        0.000   -0.4983   -0.0000    425175        17        86      0.00

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

         particle data written

         tnow       T+U       T/U     nttot     nbavg     ncavg   cputime
        0.031   -0.4983   -0.0003    424385        17        86      0.01

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

というような感じだと思います。freq はタイムステップの逆数、 eps は重力ソフトニングの値、 tol はツリーコードの opening angle で、小さくすると精度は上がりますが計算量も増えます。ここでエネルギーである T+U の値を比べると、 -0.4983 だったのが -0.4579 になって、大きく変わってしまっています。これでは駄目そうな感じがするので、タイムステップを小さくしてみましょう。

```
 %hackcode1_qp test4k test4k.outsnap freq=128

 Hack code

        nbody        freq         eps         tol
         4096      128.00      0.0500      1.0000

         options: mass,phase
 ### Fatal error [hackcode1_qp]: stropen: file "test4k.outsnap" already exists
```

nemo のプログラムは、出力ファイルがすでにあったらアボートするようになっています。

```
 %rm test4k.outsnap ; hackcode1_qp test4k test4k.outsnap freq=128

        tnow       T+U       T/U     nttot     nbavg     ncavg   cputime
```

```
2.000   -0.5006   -0.7081    780717        38       151      0.92

          cm pos    0.0012   -0.0003   -0.0008
          cm vel    0.0038    0.0004   -0.0020
```

freq=128 を指定して、タイムステップをデフォルトの 1/4 にしてみた結果です。劇的にエネルギー保存が改善されているのがわかります。最初のでは 10% 程度だったものが、 0.5% 程度になっています。

理論的には、hackcode1 で使っている leap frog 公式では、エネルギーの誤差は時間ステップの 2 乗に比例します。なので、これは大体正しい振る舞いといえます。

4.1.5 アニメーション

結果を snapplot や glnemo で見てみます。

```
%snapplot  test4k.outsnap
%glnemo  test4k.outsnap
```

どちらも、パラパラ漫画ふうですが、出力間隔が大き過ぎて気分がでません。間隔を変えるオプションは freqout なので、これも指定してもう一度実行してみます。

```
%rm test4k.outsnap ; hackcode1_qp test4k test4k.outsnap freq=128 freqout=32
```

今度は snapplot 等でもう少し気分がでると思います。

もうちょっとがんばって、ムービーを作ってみましょう。glnemo で、アニメーションのアイコン (左のメニューの一番下) を押すと、アニメーションのメニューがでます。ここでまず、 Recording/Playing の「O」を押すとアニメーションの実行記録が始まります。そこで、左側の play ボタンを押し、表示が終わる数フレーム前に (ここが極めて重要) Recording/Playing の「X」を押します。

それが済んだら、入力スナップショットを巻き戻し (play の 3 角ボタンの下の巻き戻しっぽいアイコン)、アニメーションのメニューウインドウのほうの play の「¿」ボタンを押して、

- アニメーションが動くこと
- 最後に、Snapshot ... end of snapshot reached! の警告ウインドウがでないこと

を確認して下さい。警告ウインドウがでてしまったら、 Recording からやり直して下さい。

警告ウインドウがでたもので、以下の Rendering を実行すると、何故か X が固まります。大変危険ですのでここは十分に注意して下さい。

上手くいったら、 Rendaring のメニューに移動して、スナップショットを巻き戻した上で、「Start」をクリックします。そうすると、どこかに (デフォルトは /tmp/render/frame.xxxxx.png, xxxxx は連番) フレーム毎の png ファイルができます。これを、例えば animation gif に変換するなら、ファイルができているディレクトリで

```
%convert *.png anim.gif
```

で OK です。gif アニメは、ブラウザで見ることもできますが、何が起こっているか、の物理を直観的に理解するためには、1 コマずつ進めるとか、逆に戻るとかが簡単にできるツールが必須です。結構、なかなか適当なものがありません。 gif アニメの表示には xanim が便利ですが、開発が止まって 10 年ほどたつこともあり Cygwin や Linux の最近のディストリビューションにははいっていません。 Gnome があれば、GImageView はあまり便利ではないですが一応それらしいことができます。Windows でも Explorer で沢山の画像ファイルを見る、というのが簡単な方法のような気がします。

4.1.6 エネルギー誤差の時間変化グラフを作る

さて、さっきは、エネルギーの最終の値をチェックしましたが、これだけでは普通は十分ではありません。最後の値は良くても、途中でデタラメな値になっていることもあるからです。これをチェックするためには、時間変化のグラフを作る必要があります。

hackcode1_qp は、出力のスナップショットファイルの中に、エネルギーや他の色々な情報を埋め込んでいます。これは

```
%tsf  test4k.outsnap
```

で見ることができます。最後のほうの出力が

```
set SnapShot
  set Parameters
    int Nobj 4096
    double Time 2.00000
  tes
  set Particles
    int CoordSystem 66306
    double Mass[4096] 0.000244141 0.000244141 0.000244141 0.000244141
      0.000244141 0.000244141 0.000244141 0.000244141 0.000244141
      0.000244141 0.000244141 0.000244141 0.000244141 0.000244141
      0.000244141 0.000244141 0.000244141 0.000244141 0.000244141
      . . .
    double PhaseSpace[4096][2][3] 0.153245 0.0136813 -0.205303 1.79778
      -0.760285 -0.378773 -0.454063 0.919896 -0.552425 -0.704771
      1.34618 -1.00689 -0.983543 -0.662581 -0.412249 -1.66848 -1.23127
      -0.691078 0.153962 0.0313744 -0.0690967 1.43265 -0.379112
      . . .
  tes
  set Diagnostics
    double Energy[3] -0.500630 1.21438 -1.71501
    double KETensor[3][3] 0.377060 0.0249498 -0.0361606 0.0249498
      0.380638 -0.0253557 -0.0361606 -0.0253557 0.456682
    double PETensor[3][3] -0.413097 0.0255456 0.0248519 0.0249459
      -0.417775 0.0157845 0.0236208 0.0148032 -0.599691
    double AMTensor[3][3] 0.00000 -0.000262578 0.000150161 0.000262578
      0.00000 -0.000102650 -0.000150161 0.000102650 0.00000
    double CMPhaseSpace[2][3] 0.00120219 -0.000280852 -0.000787409
      0.00384288 0.000367961 -0.00195136
    double cputime 0.778333
  tes
tes
```

ですから、

- Time がある行の数字
- Energy がある行の最初の数字

をプロットすればよさそうです。また awk を使うことにすると

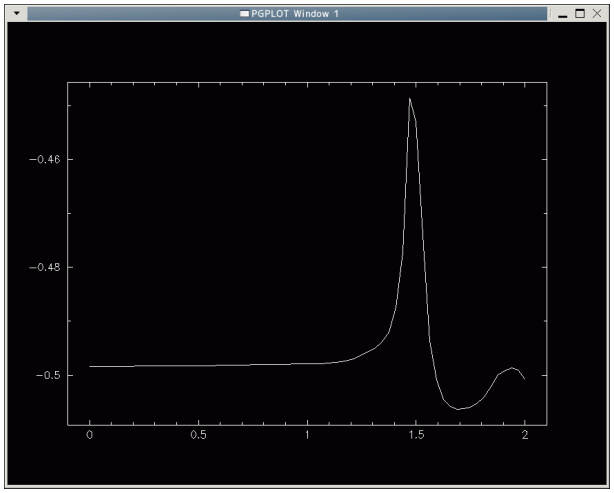

Figure 21: エネルギーの時間変化

```
%tsf test4k.outsnap | awk '/Time/{t=$3}/Energy/{print t, $3}' > tab.out
```

です。 /Time/{t=$3} は、入力行が正規表現 /Time/ と一致したら、括弧内を実行、で、 Time という文字列 w 含む行の 3 個めの文字列 (区切りはスペース) を t という変数に代入、/Energy/{print t, $3} は同様に Energy を含む行にきたら、 t の値と 3 個めの文字列をプリント、となります。

wip でグラフを書いた例が図 21 です。結構駄目ですね。

T=1.5 くらいでずれが非常に大きくなっています。これは、系全体が非常に小さく収縮し、速度も大きくなるためですが、10% はいかにも大きいので、この場合にはタイムステップをもっと小さくしたほうが安心できます。

4.1.7 初期条件を変えてみる

エネルギー誤差が大きいのは、タイムステップが大き過ぎるせいではありますが、初期条件が「一様」という特別なものであるためでもあります。

密度一様な球のコラプスは、密度一様なまま進む、という特徴があります。実際の計算では粒子数が有限なので 1 点にはあつまらないですが、原理的には 1 点に集まってしまうわけで、そうするとどんなにタイムステップを短くしても上手くいきません。そういう、破綻する場合の積分方法の話も講義では少し触れたかもしれませんが、実習ではそこまで高度なことはしません。

逆にいうと、密度が一様でなければ、そこまでひどいことは起きないわけです。密度一様でなくて、半径の-1 乗にしてみます。

```
%mkhomsph test4kb 4096 power=1
```

これも、累積質量分布がちゃんと出来ていることを確認しましょう。

```
% hackcode1_qp test4kb test4kb.outsnap freq=128 freqout=32 tstop=10
```

で少し長い計算をして glnemo でアニメーションを作ったりしてみましょう。

少しありえないほど細長いですが、楕円銀河のように見えないこともないものが出来ているはずです。エネルギーエラーも、ずっと小さくなっていることが確認できます。

エネルギーエラーを計算する awk のスクリプトを少し変えると、ビリアル比 T/U が計算できます。この場合にビリアル比のグラフを出す、というのが最初の課題、ということにします。 Energy の行で、 4 個めが T, 5 個めが U なので、 $3 の代わりに $4/$5 をプリントさせればよいわけです。

さて、タイムステップをどんどん小さくしていくと、エネルギーエラーは小さくなりますが、ある程度以下にはならないことがわかります。それはなぜか、また、さらに小さくするにはどうすればよいか、を調べるのは発展課題とします。

4.1.8 密度分布を書く

前に、初期条件をチェックするには累積質量分布を使いましたが、「どのような銀河ができたのか」といった物理的な性質を見るのには累積分布はあまり便利ではありません。なので、半径方向の空間密度分布を書いてみます。

nemo には radprof というプログラムがあるのですが、動作が気に喰わないというか今一つ使えないので、snapprint の出力から書いてみることにします。前に書いたように、どうしても N 体では粒子数によるゆらぎがあります。ここでは粒子 128 個毎に、その 2 つの粒子にはさまれた球殻の平均密度を、2 つの粒子の半径の平均の半径での密度とします。

式で書いたほうがわかりやすいですね。粒子が半径順に並んでいて、半径が r_i で質量が m_i とすると、粒子 i, j にはさまれた範囲の密度を

$$\begin{aligned} r &= \frac{r_j + r_i}{2} \\ \rho_r &= \frac{\sum_{i<k\leq j} m_k}{\frac{4\pi}{3}(r_j^3 - r_i^3)} \end{aligned} \tag{1}$$

とします。これを計算するプログラムですが、C でも Fortran でも好きな言語でよいですが、以下は awk の例です。

```
{
    m+= $2
    if ( NR % 128 == 0){
r1 = $1
print (r1+r0)/2, m/(r1*r1*r1-r0*r0*r0)*0.2387
m=0
r0=r1
    }
}
```

awk には、

- 未定義の変数は値が 0
- 変数の値は文字列または実数、どっちになるかは適当
- 暗黙に、1 行づつ読んで処理、というループになっている
- 1 行の、複数のフィールドの数字があらかじめ $1, $2, ... という名前で使える

といった特性があるので、割合簡潔に処理を表現できているのがわかります。なお、 awk なんていう大昔の言語は嫌だ、もうちょっと現代的なものがよい、という向きには、例えば Ruby を使うなら

```
r0=m=0
while s=gets()
  a=s.chomp.split
    m+= a[1].to_f
    if ( $. % 128 == 0)
r1 = a[0].to_f
print (r1+r0)/2, " ",m/(r1*r1*r1-r0*r0*r0)*0.2387, "\n"
m=0
r0=r1
    end
end
```

という感じですが、これでは awk に比べて面倒なだけでメリットはありません。どちらの場合でも

```
 %snapprint test4k.sort options=r,m | awk -f density.awk > tab.out
```

とか

```
 %snapprint test4k.sort options=r,m | awk -f density.rb > tab.out
```

とかすれば、それらしいデータがでているはずです。(もちろん、上のコードをそういう名前でファイルにセーブする必要があります) 前と同様にグラフを書いてみましょう。折角 ruby を使うのでもう少し気の効いたことをするなら、プログラムを 2 行ばかり変更して

```
r0=m=0
open("|snapprint #{ARGV[0]} options=r,m"){|io|
  while s= io.gets()
    a=s.chomp.split
    m+= a[1].to_f
    if ( $. % 128 == 0)
      r1 = a[0].to_f
      print (r1+r0)/2, " ",m/(r1*r1*r1-r0*r0*r0)*0.2387, "\n"
      m=0
      r0=r1
    end
  end
}
```

として見る、といったこともできます。この場合は、ruby プログラムの中で snapprint #{ARGV[0]} options=r,m を実行して、その出力を io.gets() で読む、というふうにしています。こうしておくと、

```
 %ruby density2.rb test4k.sort
```

という形で実行できるので、沢山のファイルについて密度分布を書く、といった場合にやりやすいし、後になってこのプログラムの使いかたをすっかり忘れた時にも、 snapprint をどう使ったかは書いてあるので安全です。まあ、ヘルプとか、色々つけておくとあとで助かるのですが、世間で普通に使われている方法は面倒であまり使いやすくないように思います。プリンストン高等研究所の Piet Hut (Barnes-Hut ツリーコードの Hut) と牧野が少し前に「コマンドラインオプションのパーザはどうあるべきか」だけで本一冊分くらいの 原稿[16]を書いたことがあるので、そういうのに興味がある人は読んで見ると得ることがあるかもしれません。

グラフは、この場合には

[16] http://www.artcompsci.org/kali/vol/command_line/title.html

```
%wip
Could not open user's WIP initialization file.
Setting up default device [/XWINDOW]
Welcome to WIP Version: 2.3 22jan98

WIP> device /xw
WIP> data tab.out
WIP> xcol 1
WIP> ycol 2
WIP> limit
WIP> era
WIP> box
WIP> conn
```

という感じで書けると思います。これを、シミュレーションの最後のスナップショットに適用してみましょう。最後のスナップショットに snapsort とかを適用するには、最後のスナップショットを切り出すコマンドを使うのが便利です。

```
%snaptrim test4kb.outsnap - times=10 | snapsort - - rank=r |\
? snapprint - options=r,m | ruby density.rb > tab.out
### nemo Debug Info: r m
### nemo Debug Info: time = 10  npart =   1     ndiag =   1     outputing particles
```

4.1.9 提出課題

1. test4k.outsnap と test4kb.outsnap のそれぞれについて、最終状態での密度分布を計算し、両対数グラフにして提出せよ。
2. test4kb.outsnap のグラフはスナップショットの絵と比べて「もっともらしい」、つまり、構造の特徴を捉えていると考えられるか？もしも捉えていないならそれはなぜで、どう処理を変更すればよいか検討し、「もっともらしい」ものになった結果を提出せよ。
3. test4k からの数値積分について、時間刻みを 2 桁以上の範囲で変化させて、t=0 から t=2 まで積分した時の最大誤差と最終誤差を時間刻みの関数としてプロットしたグラフを作り、提出せよ。

なお、提出用のファイルを作るには、画面のスクリーンショットとかでもいけなくはないですが、例えば TeX の場合 ps ファイルを作るのが便利です。WIP の場合、

```
%wip
Could not open user's WIP initialization file.
Setting up default device [/XWINDOW]
Welcome to WIP Version: 2.3 22jan98

WIP> device xxx.ps/vcps
```

と、device コマンドで出力を ps ファイル (この場合、名前が xxx.ps) にすることができます。この後で

```
WIP> viewport 0.2 0.9 0.3 0.8
```

を実行すると大体正方形のグラフになるはずです。

4.2 円盤銀河

実際の実習ではこの辺で既に力つきるかもしれませんが、折角なのでもう少し気の効いた銀河モデルを作ってみましょう。nemo には Kuijken と Dubinski の作った、 GalactICS という銀河モデル作成パッケージがはいっています。

```
%mkkd95 testkd
```

で何かができます。 glnemo でぐるぐる回してみると、ハロー、ディスク、バルジからできているらしいことがわかります。

4.2.1 単位系の修正

N 体計算をする時の割合大事な話として、どういう単位系で考えるか、ということがあります。例えば、銀河系の計算をするとして、 kpc, Myr, solar mass といった単位系を使っていたのでは、物理的な理解が必ずしも簡単ではありません。

従って、多くの場合に、いわゆる standard unit あるいは N-body unit といわれる単位系を使います。これは、系の質量が 1、重力定数も 1、系の全エネルギーが -1/4 であるような単位系です。

何故 -1/4 で -1 でないのか、というと、この場合ポテンシャルエネルギーが-1/2 であり、全粒子ペアについて $\frac{1}{|\mathbf{r}_i - \mathbf{r}_j|}$ の平均の逆数、つまり粒子間距離の調和平均をとると、それが 1 になっている、ということで -1/4 を選んでいます。 nemo の多くのプログラムはこれに従っているか、大体従っています。例えば mkplummer の場合にはそうなっています。

しかし、GalactICS は作った人にそういう考え方はないので、全く違う単位系 (但し、重力定数は 1) ででてきます。なので、ここではまず、GalactICS の出力を standard unit に変換してみます。

元の系の質量が M_0、ポテンシャルエネルギーが V_0、運動エネルギーが T_0 であったとして、質量を α 倍、位置座標を β 倍、速度を γ 倍すれば、新しい質量とエネルギーは

$$\begin{aligned} M_1 &= \alpha M_0 \\ V_1 &= \alpha^2 \beta^{-1} V_0 \\ T_1 &= \alpha \gamma^2 T_0 \end{aligned} \tag{2}$$

となります。この時に、ビリアル比 T/V を変化させないように速度をスケールするには、

$$\gamma = \sqrt{\alpha/\beta} \tag{3}$$

でないといけないことがわかります。また、 M_1 と V_1 から α と β を求めると

$$\begin{aligned} \alpha &= M_1/M0 \\ \beta &= \alpha^2 V_0/V_1 \end{aligned} \tag{4}$$

となるわけです。というわけで、まずできたモデルの質量とエネルギーを計算してみます。

```
% hackforce_qp testkd - tol=0.5 eps=0.025 |\
? snapprint - options=m,"m*(phi+vx*vx+vy*vy+vz*vz)*0.5" |\
```

```
? awk '{mtot += $1; etot += $2}END{ print mtot, etot}'
### nemo Debug Info: m m*(phi+vx*vx+vy*vy+vz*vz)*0.5
### nemo Debug Info: initial rsize: 4.000000    rmin: -2.000000  -2.000000  -2.000000
### nemo Debug Info:   final rsize: 128.000000    rmin: -38.000000  -54.000000  -86.000000
6.21204 -2.48885
```

ちょっと意味不明ですね。 "—" はパイプで、左のコマンドの出力を右のコマンドの入力にします。 nemo のプログラムは、ファイル名に "-" を指定すると出力なら標準出力、入力なら標準入力になるので、上の例のようにパイプでつなぐことができます。このために、中間ファイルとかを作る必要が (少なくとも見かけ上は、、、) なく、複雑な処理をパイプで実現できます。

この例では、最初の hackforce_qp は、ツリーコードでポテンシャルと加速度を計算して、それをスナップショットに付け加えて出力するものです。次の snapprint では、各粒子の質量と、「運動エネルギーとポテンシャルエネルギーの半分の合計」を書かせています。ポテンシャルエネルギーを半分にするのは、ペアを 2 重に数えてしまっているのを補正するためです。最後の awk では、質量、エネルギーをそれぞれ合計していって、最後に (END {} のところで) 出力しています。

さて、これで質量とエネルギーがわかったので、あとは電卓でスケールを計算して、

```
%snapscale testkd testkd.scaled mscale=0.1609777 \
?   rscale=0.25798248826425 vscale=0.789928431392108
```

とすれば一件落着なのですが、これだと、GalactICS で銀河モデルを作る度に電卓で面倒な計算をして、数字を人間が打ち込む (まあ、 xcalc とか使っていればコピペもできますが) ことになります。式に従って計算する、というのは計算機のほうが人間よりも得意に決まっているので、これも計算機にやらせてみましょう。

```
#!/bin/csh -f
#
# scaletoheggie.csh
#
# scale a snapshot to the standarr (aka Heggie) units
#
# Usage: scaletoheggie.csh snapshot_file
# output file: snapshot_file.scaled
snapscale $1 $1.scaled \
`hackforce_qp $1 - tol=0.5 eps=0.025 |\
  snapprint - options=m,"m*(phi+vx*vx+vy*vy+vz*vz)*0.5" |\
  awk '{mtot += $1; etot += $2}END{a=1/mtot;b=-a*a*etot/0.25; print "ms=" a, "rs=" b, "vs=" sqrt(a/
```

こんな感じのスクリプトを作って、実行すると

```
%csh -f scaletoheggie.csh testkd
### nemo Debug Info: initial rsize: 4.000000    rmin: -2.000000  -2.000000  -2.000000
### nemo Debug Info: m m*(phi+vx*vx+vy*vy+vz*vz)*0.5
### nemo Debug Info:   final rsize: 128.000000    rmin: -38.000000  -54.000000  -86.000000
### Warning [snapscale]: Resolving partially matched keyword ms= into mscale=
### Warning [snapscale]: Resolving partially matched keyword rs= into rscale=
### Warning [snapscale]: Resolving partially matched keyword vs= into vscale=

%tsf testkd.scaled
char Headline[12] "-1250698778"
```

```
char History[80] "snapscale testkd testkd.scaled ms=0.160978 rs=0.25798
  3 vs=0.789928 VERSION=3.2b"
char History[82] "tabtos galaxy ../testkd nbody,time mass,pos,vel headl
  ine=-1250698778 VERSION=1.5a"
set SnapShot
  set Parameters
    int Nobj 18000
    double Time 0.00000
  tes
  set Particles
    int CoordSystem 66306
    double Mass[18000] 1.75180e-05 1.75180e-05 1.75180e-05 1.75180e-05
      1.75180e-05 1.75180e-05 1.75180e-05 1.75180e-05 1.75180e-05
      1.75180e-05 1.75180e-05 1.75180e-05 1.75180e-05 1.75180e-05
      1.75180e-05 1.75180e-05 1.75180e-05 1.75180e-05 1.75180e-05
      . . .
    double PhaseSpace[18000][2][3] -0.0729338 0.257252 0.00723250
      -0.607239 -0.289207 0.212093 -0.261455 -0.103321 0.0670730
      0.159549 -0.373926 0.129851 -0.849309 -0.927010 -0.00670115
      0.560895 -0.433799 0.0387852 0.0114802 0.336401 -0.0339088
      . . .
  tes
tes
```

で、 tsf の出力から、 snapscale に渡っているオプションはそれらしいことがわかります。本当にエネルギー等が正しいかどうかはもう一度全エネルギーと質量を計算して確認しましょう。上のスクリプトでは ‘command‘、つまり、バッククォートで囲ったコマンドは、そのコマンドの実行結果に展開される、という shell の割合便利な機能を使って、awk の出力を snapscale のオプションの格好で出したものを直接 snapscale に渡しています。 ms= とか書くと、nemo のコマンドライン解釈ルーチンが、一致するオプションを見つけてきます。この場合は、 in out mscale rscale vscale の順でオプションを与えているので ms= とかは書かなくてもよいのですが、ここでは書いてみました。

では、適当な初期条件で円盤銀河 2 つをぶつけてみましょう。

```
%snaprotate testkd.scaled testkd.rotated theta=45 spinvector=1,0,0
%snapstack  testkd.rotated testkd.rotated  merger.in deltar=5,0,0\
? deltav=-0.7,0.35,0 zerocm=true
%hackcode1_qp merger.in merger.out freq=128 eps=0.02 tstop=20\
? freqout=16 minor_freqout=16 &
```

これはとても長い時間がかかるので、 時々 glnemo で様子をみながら休憩、というところで、今日の実習はおしまいにします。

```
%glnemo merger.out select=0:7999,8000:17999,18000:25999,26000:35999 blending=t
```

とすると、ディスク、ハローで色を変えることができます。また、オプションメニューからさらに色々変更することもできます。

ここで作った銀河モデルは粒子数 18000 で、円盤銀河の色々な性質を調べるには到底十分とはいえません。従って、実際の研究でシミュレーションをするには、 hackcode1_qp よりももっと速くシミュレーションできる必要があります。

これには、

- プログラムを改良して速くする
- 速い計算機を使う。使えるようにプログラムを改良する
- 速い計算機を作る

といった、様々なアプローチが行われています。今日やった計算は 36000 粒子ですが、現在の宇宙論研究の最先端では 100 億を超える粒子数を使った計算も行われています。

5 CfCA の計算機を使う

ネットワークにつながった状態で、 ssh で X の画面が飛ぶようになっていれば、

```
% ssh -l lecxxx an03.cfca.nao.ac.jp xterm
```

(パスワードを聞かれるので入力)

で、xterm の画面がでます。デフォルトの設定では bash になっているので、

```
% csh
% source /home/makinojn/work/nemo_cvs/nemo_start
```

を実行して下さい。 この後、 bash を使いたいなら、

```
% bash
```

です。これで、 nemo が使える環境になっているはずです。

5.1 わかっている問題

mkkd95 が動作しません。サンプルの出力を
/home/makinojn/work/sandbox/testkd
に用意したので、コピーして使って下さい。