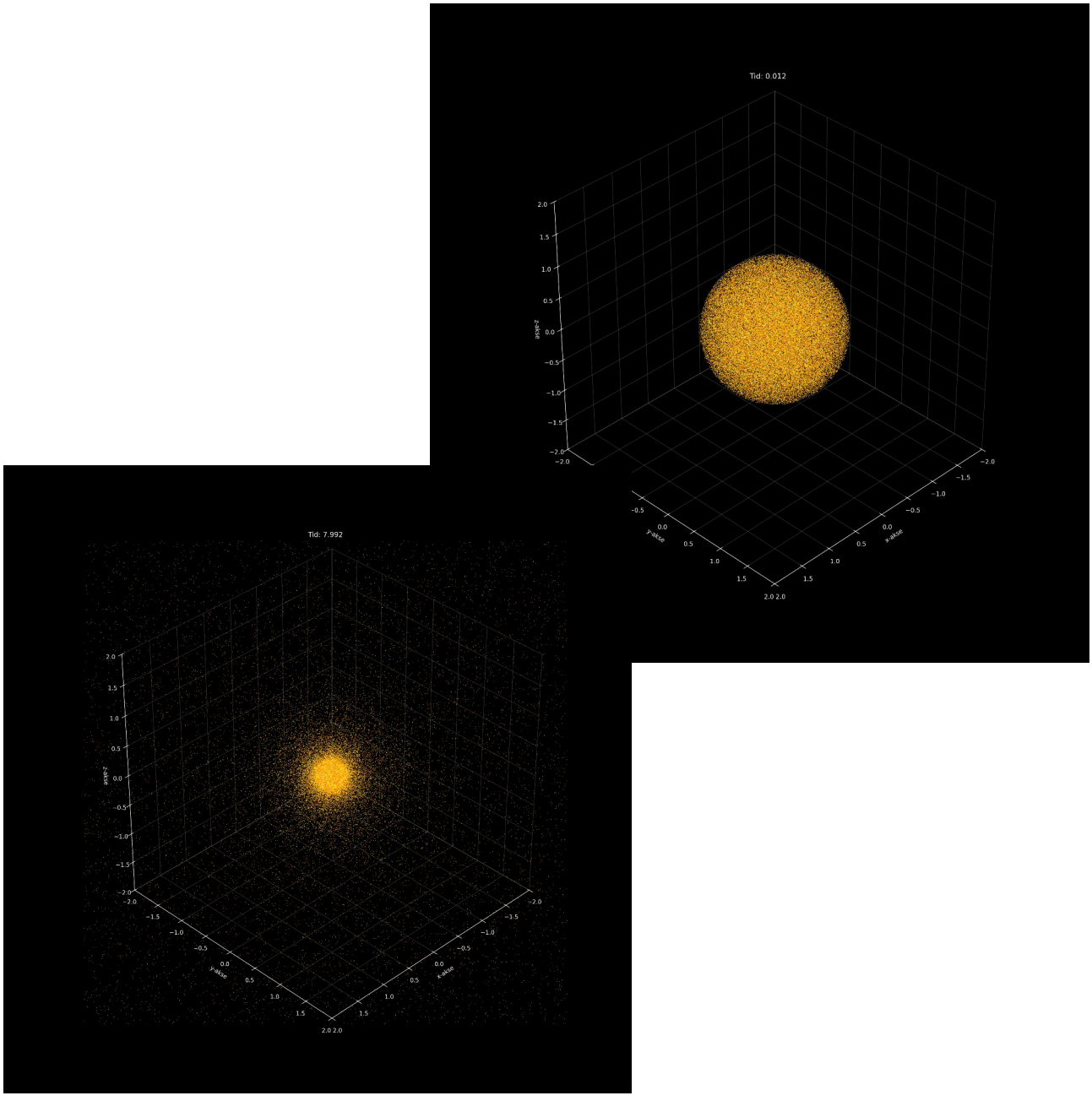
Nbodysimuleringer



Af Michael A. D. Møller

Marts 2018, opdateret april 2026.

Indholdsfortegnelse

1 NBody simuleringer

Denne tekst beskriver relativt kort, hvordan man kan komme i gang med at løse bevægelsesligninger for mange legeme-systemer, herefter kaldet Nbody-systemer. For at kunne bruge de tilhørende eksekverbare programmer, er det nødvendigt at have en Windows pc. Man kan dog også bruge Linux og MacOS, hvis man er villig til selv at kompilere programmerne.

1.1 Fra mennesketekst til maskinkode - kompilering

En programtekst skal oversættes, så computeren kan forstå koden. Derudover skal koden evt. samles med andre programstykker, man tidligere har skrevet. Det hedder at linke objektfiler. Når der i denne tekst skrives kompilering, så menes der også linkning til eksterne filer.

Det lykkes ikke altid at kompilere et program i første omgang, for de mindste skrivefejl i ens program kan give anledning til rigtig mange kompilerfejl, og hvis man benytter andres programmer, så skal man ydermere være meget opmærksom på at få hentet de biblioteker/underprogrammer, som programmet er afhængig af.

Derfor vil jeg ikke foreslå en begynder at gå i gang med kompilering af andres programmer, hvis man arbejder med en kort tidshorisont. Et allerede pc-kompileret program, som Sverre Aarseth har skrevet, kan man downloade fra astro-gym.dk.

Det er dog ofte nødvendigt at benytte egenskrevne programmer til f.eks. at generere startfordelinger for simuleringer, eller til at lave beregninger, men hvis man selv har skrevet et program, er det også meget nemmere at rette de fejl, som en kompiler finder frem til.

En gratis og god Fortran-kompiler kan man hente på webstedet Gfortran.

1.1.1 Biblioteker

Har man en masse rutiner, i form af objektfiler, som man gerne vil samle i et bibliotek til senere brug, kan man lægge dem i et arkiv ved i en kommandoprompt at skrive kommandoen *ar -rcs libbibnavn.a *.o.* Brugeren kan erstatte teksten *bibnavn* med et eget passende valg.

Hvis man vil se, hvad der ligger i et bibliotek, så kan man i en kommandoprompt skrive kommandoen *ar -t libbibnavn.a.*

Har man et bibliotek med rutiner, som man gerne vil linke til ved kompileringen, skal man i kommandoprompten skrive *gfortran mitprogram.f90 -L**c:\StiTilibliotek** -l**bibnavn*** (Husk at skrive henvisningen til biblioteket til sidst i linien.)[1]

1.2 Værktøjer

1.2.1 Redigeringsprogram til programmeringen

Man kan i første omgang bruge Windows eget Notepad.exe til at redigere i fortranprogrammer, men de har ikke ret mange funktioner indbygget. Et gratis program, som er meget bedre end Notepad

1 I eksemplet har man præciseret, at man skriver i Fortran 90. Hvis filtypenavnet f.eks. kaldes f95 skal man overholde Fortran 95-standarden.

kan f.eks. være *Visual Studio Code* eller *Visual Studio*, som også har kunstig intelligens indbygget. Dvs. Visual Studio kan lave langt det meste af programmeringsarbejdet for en! Du kan hente Visual Studio (code) på adressen, der er nævnt i fodnote.[2] Hvis det program ikke falder i din smag, er der andre i fodnote[3].

1.2.2 Graftegning - GNUplot

Et gratis program til at tegne grafer i 2 dimensioner og i 3 dimensioner, kan du finde på adressen: http://gnuplot.sourceforge.net/ GNUplot er interaktivt, men det kan også gemme alle dine indtastede kommandoer, så du hurtigt kan afvikle dem på et senere tidspunkt. Hvis man vil lave rigtig mange grafer hurtigt, er GNUplot en god løsning.

Eksempel på script i GNUplot

Herunder er et eksempel på nogle kodelinier, der kan bruges til at plotte grafer og gemme dem i png-formatet. (Undlad evt at skrive kommentarerne markeret med rødt.)

```
cd 'd:/nbodytest/billeder/samtlige'          #Bemærk at back slash skal skrives som front slash!
set xlabel 'x-akse' tc rgb "black" offset 0,-1 #Sort betyder at akserne bliver usynlige.
set ylabel 'y-akse' tc rgb "black" offset 2,0
set zlabel 'z-akse' tc rgb "black" offset 0,0
set key tc rgb "white"
set xyplane 0
set border lt rgb "black"
set xrange [-40:140] noreverse nowriteback
set yrange [-40:140] noreverse nowriteback
set zrange [-40:40] noreverse nowriteback
#set view equal xyz                                  #Skaler akserne så de bliver ens, hvis
man ønsker det. Her er det ikke ønsket – derfor havelågen foran.
set term jpeg background rgb "black" size 1920,1080      #Sort baggrund.
set view 75,0,1                                          #Output bliver filer i jpeg-format.
do for [i=1:1401] {                                      #Lav 1401 billeder
  set output 'Billede.'.i.'.jpeg'
  splot 'fort.'.i using 4:5:6 with dots lt rgb "white" title 'Galaksekollision '.i
                                                         #Billederne hedder Billede.tal.jpeg
 }
set term wxt                                             #Fremtidig output bliver skærmen.
```

1.2.3 Graftegning - TOPCat

Et program, der er lynhurtig at gå til er TOPCat. Det kræver Java installeret, for at virke. Det kan downloades http://www.star.bris.ac.uk/~mbt/topcat/ Programmet kan hurtigt læse mange filer ind, og så kan man manuelt lave grafer, som gemmes i billedefiler. Programmet er ikke så godt, hvis man ønsker at tegne rigtig mange grafer.

1.2.4 Python

Python er et sprog, som kan næsten alt, dvs. det kan også bruges til at lave grafik, der ser pænere ud en GNUplot. Forsidebilledet er lavet i python. På webstedet hvor denne tekst findes, er der også et pythonprogram til at lave billeder. Python er langsommere end GNUplot, men resultatet er ofte værd at vente på.

---

2 https://visualstudio.microsoft.com/downloads/ d

3 https://www.webpagefx.com/blog/web-design/12-excellent-free-text-editors-for-coders/

1.3 Simuleringsprogrammet

Programmet til at regne bevægelsesligningerne ud for N legemer er et Fortran-77 program, som Sverre J Aarseth har skrevet. Programmet er offentliggjort i ”Galactic Dynamics”. [1] S. J. Aarseth er en af pionererne indenfor Nbody-simuleringer. Programmet har jeg oversat til f90-standarden, så det får et mere læsevenligt look.

Der findes mere avancerede programmer til fri download[4] på Internettet, men ingen er vist så kompakte og overskuelige som programmet, der anvendes her.

Programmet virker ved, at man indlæser en startfordeling med partikelmasser samt deres startpositioner og starthastigheder, og så regner programmet bevægelsesligningerne og totalenergien ud for en, hvorefter resultaterne gemmes i en mængde tekst-filer, som derefter kan indlæses i et grafikprogram eller et regneark. Programmet regner med en justeret gravitationskraft, som har formen $\vec{F}_{ij} = \frac{G \cdot m_i \cdot m_j}{r_{ij}^2 + \epsilon^2} \cdot \vec{e}_{ij}$. Grunden til, at programmet bruger en alternativ gravitationskraft forklares nedenfor.

1.3.1 Inputparametre

Brugeren skal supplere en inputfil i tekstformat med navnet *Startfil*, som indeholder følgende:*N*
eta
deltat
Tcrit
Eps2
m_1 x_1 y_1 z_1 vx_1 vy_1 vz_1
...
...
m_N x_N y_N z_N vx_N vy_N vz_N

Partikeldataene skal være skrevet på Fortran-formatet: format(1x,7(1pe14.6)). Dvs. der skal være 1 mellemrum efterfulgt af 7 tal, der er 14 karakterer lange og med 6 decimaler. Tallene angives i 10-tals eksponentformatet. Nedenfor er et eksempel på de første linier i en startfil:

```
 100
 2.000000E-02
 1.000000E+00
 5.000000E+01
 1.000000E-03
 1.000000E-02 -1.473956E-03 -4.618212E-03  4.161983E-03 -1.635099E-02 -1.733586E-02  1.865468E-01
 1.000000E-02 -1.167595E-02  9.209679E-03  2.078065E-03 -1.156662E+00  8.385236E-01  7.363014E-02
```

Man ser af eksemplet ovenfor at N = 100, *eta* = 0,02, *deltat* = 1,0, *Tcrit* = 50 og *eps2* = 0,001.

N angiver antallet af partikler. η er en parameter, der fortæller hvor præcist, der skal regnes med. Hvis energien f.eks. viser sig ikke at være bevaret i løbet af en simulation, bør man nok overveje at gøre η mindre og så gentage simuleringen.

Δt fortæller hvor tit, man skal skrive en fil ud med resultater. Her er det værd at bemærke, at hvis man vil lave en animation, så kræves der 30 billeder i sekundet. T_{crit} fortæller hvornår simuleringen skal stoppe. Endelig er ε^2 den såkaldte softeningparameter, der fysisk set sørger for, at stjernerne

4 https://people.ast.cam.ac.uk/~sverre/web/pages/nbody.htm

ikke ramler ind i hinanden. Det gør de nemlig ikke i virkeligheden, men da vi ikke kan arbejde med N = mange millioner, vil problemet kunne opstå i simuleringen. Hvis afstanden mellem to legemer også bliver meget lille, vil computerberegningerne af bevægelsesligningerne også gå galt, så det er nødvendigt at partiklerne ikke kommer for tæt på hinanden.

Denne parameter bevirker altså i sig selv, at beregningerne ikke bliver helt korrekte, men de løser nogle endnu værre potentielle beregningsfejl. *Rodionov og Sotnikova*[5] har vist at softeningsparameteren skal være ca. halvdelen af partiklerns middelafstand.

1.4 Middelpartikelafstanden og softeningparameteren

Middelpartikelafstanden, λ, er defineret som

$$\lambda^3 \cdot n = 1 \tag{1}$$

hvor n er antalstætheden. Den kan forstås, som at man inddeler alle partiklerne i kasser med sidelængden λ. Der skal være 1 partikel i hver kasse. Derfor 1-tallet på højresiden i formlen.

1.4.1 Softeningparameteren

En god softeninglængde er det halve af den middelpartikelafstanden.

Hvis vi i en simulering har N partikler er de fordelt på en bestemt måde. F. eks. kan man i en første tilnærmelse placere dem sfærisk symmetrisk med konstant densitet.

I det tilfælde er antalstætheden

$$n = \frac{3 \cdot N}{4 \cdot \pi \cdot R^3} \tag{2}$$

Oftest vælger man i at startfordelingen er spredt udover afstanden R=1 computerlængdeenhed. Dvs. vi kan bruge (1) og (2) til at bestemme middelpartikelafstanden til

$$\lambda = \left(\frac{4 \cdot \pi}{3 \cdot N} \right)^{1/3} \tag{3}$$

Derfor vil man for en sfærisk symmetrisk fordeling med 100000 partikler få $\varepsilon = ½\lambda$ =0,017. Når partiklerne trækker sig sammen formindskes middelpartikelafstanden, så det er en god ide, at vælge en softeningparameter, der er lidt mindre. I simuleringen om kollaps af sfære er parameteren valgt til ε=0,01.

Hvis man arbejder med andre startfordelinger, skal man genoverveje værdien af softening-parameteren.

1.5 Computerenheder vs. SI-enheder

Programmet regner ikke i SI-enheder. Det skyldes hensynet til at lave en hurtig programafvikling. F.eks. er gravitationskonstanten defineret til 1 i programmet. Man skal derefter selv regne om til

5 **Optimal Choice of the Softening Length and Time Step in N-body Simulations.** / Rodionov, S. A.; Sotnikova, N. Ya. In: Astronomy Reports, Vol. 49, 2005, p. 470-476

forståelige enheder. Her er Keplers 3. lov en stor hjælp. Nedenfor vil vi gennemgå et eksempel på, hvordan man regner om mellem enhederne.

Vi definerer totalmassen af galaksen/hoben til M=1. G er i programmet defineret til 1. Lad os definere længdeenheden så helmasseradius, r_0, $M(r_0)=M$ for fordelingen $r_0=1$.

Keplers 3. lov er

$$\frac{r^3}{T^2}=\frac{G\cdot M}{4\cdot\pi^2} \tag{4}$$

Det betyder, at hastigheden ved r_0 er, og hvor [CE] betyder *computerenhed.*

$$v(r_0)=\sqrt{\frac{G\cdot M(r_0)}{r_0}}=1\,[\mathrm{CE}] \tag{5}$$

Hvis man vil finde en tidsenhed, kan man atter manipulere med Keplers 3. lov, og så får man følgende relation:

$$T_0=\sqrt{\frac{4\cdot\pi^2\cdot r_0^3}{G\cdot M(r_0)}}=2\cdot\pi\,[\mathrm{CE}] \tag{6}$$

Tilsvarende gælder for energien af en partikel med gennemsnitsmassen $m=M/N=1/$N, at

$$E_0=\frac{-G\cdot M(r_0)\cdot m}{r_0}=-\frac{1}{N}\,[\mathrm{CE}] \tag{7}$$

1.5.1 Galakser/kuglehobe

Lad nu os betragte en fordeling med $M_{total}(r_{\mathrm{total}})$= a·$10^{10}$·$M_{\mathrm{sol}}$ og r_{total} = b kpc. Sættes disse parametre ind i Keplers 3. lov får man

$$v(r_{total})=\sqrt{\frac{G\cdot M_{total}}{r_{total}}}=\sqrt{\frac{G\cdot a\cdot 10^{10}M_{sol}}{b\,\mathrm{kpc}}}=\sqrt{\frac{a}{b}}\cdot 207{,}4\,\frac{\mathrm{km}}{\mathrm{s}}=1\,[\mathrm{CE}]\Rightarrow$$

$$1\,\mathrm{v}[\mathrm{CE}]=207{,}4\,\frac{\mathrm{km}}{\mathrm{s}}\cdot\sqrt{\frac{a}{b}} \tag{8}$$

Tilsvarende for tidsenheden

$$T_0=\sqrt{\frac{4\cdot\pi^2\cdot(b\,\mathrm{kpc})^3}{G\cdot M_{total}}}=\sqrt{\frac{4\cdot\pi^2\cdot(b\,\mathrm{kpc})^3}{G\cdot a\cdot 10^{10}M_{sol}}}=9{,}349\cdot 10^{14}\,\mathrm{s}\cdot\sqrt{\frac{b^3}{a}}=2\cdot\pi\,[\mathrm{CE}]\Rightarrow$$

$$1\,\mathrm{T}[\mathrm{CE}]=1{,}488\cdot 10^{14}\,\mathrm{s}\cdot\sqrt{\frac{b^3}{a}}=4{,}715\cdot 10^{6}\,\mathrm{yr}\cdot\sqrt{\frac{b^3}{a}} \tag{9}$$

For energien får vi

$$E_0 = \frac{-G\cdot\left(a\cdot 10^{10} M_{sol}\right)^2}{b\,\mathrm{kpc}\cdot N} = \frac{a^2}{b}\cdot 8{,}556\cdot 10^{50}\,\mathrm{J} = \frac{-1}{N}\,[\mathrm{CE}] \Rightarrow$$

$$1\,[\mathrm{ECE}] = 8{,}556\cdot 10^{50}\,\mathrm{J}\cdot\frac{a^2}{b} \tag{10}$$

Og som kontrol kan vi beregne en computerlængdeenheds relation til kpc

$$L\,[\mathrm{lCE}] = 1\,[\mathrm{vCE}]\cdot 1\,[\mathrm{TCE}] = 207{,}4\,\frac{\mathrm{km}}{\mathrm{s}}\cdot\sqrt{\frac{a}{b}}\cdot 1{,}488\cdot 10^{14}\,\mathrm{s}\cdot\sqrt{\frac{b^3}{a}} = b\cdot 3{,}086\cdot 10^{19}\,\mathrm{m} \Rightarrow$$

$$1\,[\mathrm{lCE}] = b\,\mathrm{kpc} \tag{11}$$

1.6 Simuleringens køretid

Hvor lang tid skal en simulering køre? Det kommer jo an på det problem, man vil simulere. Herunder følger et eksempel på at undersøge et kollaps af en gassky til en stjernehob.

Ved et kollaps udveksler partiklerne energi med hinanden – vi siger de *relakserer.* (Relax: Falde til ro.) Partiklerne skal lave flere passager, før man ender med en stabil fordeling. For at sikre os at simuleringen kører så længe, at stjernerne er faldet helt til ro, kan vi bruge fritfalds-formlen[6], som angiver den tid det ca. tager for en stjerne at flyve en diameter. Der skal flyves nogle ture frem og tilbage, før systemet er helt faldet til ro, så vi lader simuleringen køre $5T_{\mathrm{ff}}$.

$$T_{ff} = \sqrt{\frac{3\cdot\pi}{32\cdot G\cdot\varrho}} \tag{12}$$

1.6.1 Eksempel. Sfærisk symmetrisk og konstant densitetsmodel

Lad os antage, at en simulering er sfærisk symmetrisk med N partikler, og M=1=R_{start}. Så er densiteten

$$\varrho = \frac{3\cdot M}{4\cdot\pi\cdot R^3} = \frac{3}{4\cdot\pi} \tag{13}$$

Indsætter vi (13) i (12) får vi

$$t_{ff} = \frac{\pi}{\sqrt{8}}\,[\mathrm{TCE}] \tag{14}$$

Så simuleringen skal i dette tilfælde køre 5,6 computertider. Man kan efterfølgende lave en visuel inspektion, for at forvisse sig om, at der ikke sker væsentlige ændringer i strukturen efter dette tidsrum.

1.7 Stjernernes massefordeling

Lad os antage, at stjernernes massefordeling til at være inspireret af en Maxwell-Boltzmann-fordeling (MB). Det giver en fordeling af forholdsvist mange små stjerner, og færre store stjerner.

6 Mo et al, *Galaxy Formation and Evolution*, 3rd ed, Cambridge University Press 2014.

Her er det værd at bemærke, at store stjerner er væk i kuglehobe, fordi hobene er meget ældre end levetiden for de tungeste stjerner. I første omgang inkluderer vi dem, og når vi har fået et stabilt system, kan vi evt. fjerne dem, før simuleringen får lov at fortsætte.

Fordelingen har følgende fordele:

i. Den er nem at bruge i et computerprogram, da den kan genereres af 2 normalfordelte tilfældige tal.
ii. Den giver større hyppighed af lette partikler/stjerner end tungere, hvilket passer med virkeligheden.
iii. Resultater fra gasfysikken er tidligere med held brugt i stjerners dynamik. (Tænk f.eks. på den isoterme stjernefordeling.)

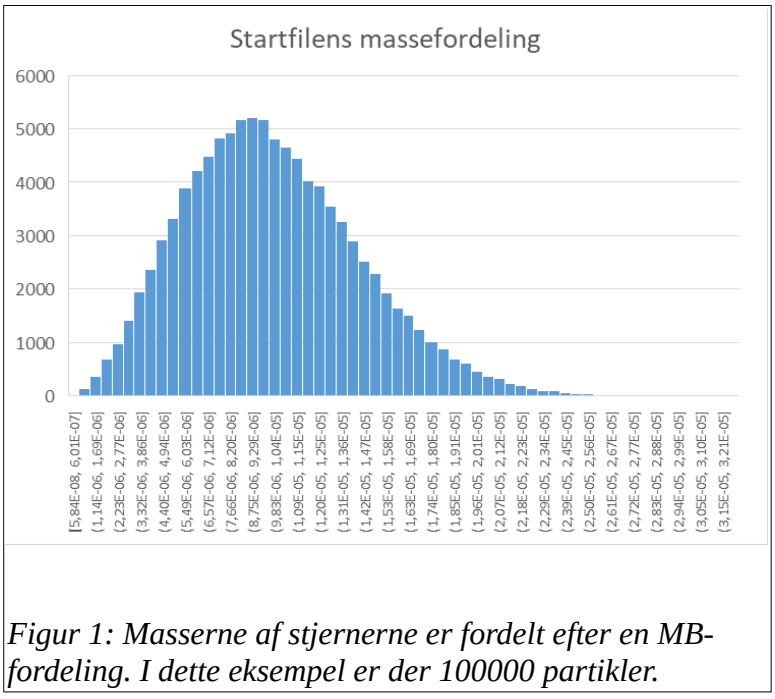
Fordelingsfunktionen er

$$f(M)=\frac{4\cdot a^{1,5}}{\sqrt{\pi}}\cdot M^2\cdot\exp\left(-a\cdot M^2\right) \tag{15}$$

Vi sætter sætter hele systemets masse til 1 computerenhed. Så får vi en fordelingsfunktion, der ser ud som følger

$$f(M)=\frac{32}{\pi^2}\cdot M^2\cdot\exp\left(\frac{-4\cdot M^2}{\pi}\right) \tag{16}$$

Vi kan beregne antal partikler med en bestemt masse som $N(M) = N_0 \cdot f(M) \cdot \Delta M$. I figur 1 kan man se hvordan masserne af stjernerne ser ud i en fordeling med 10^5 partikler. Gennemsnitsmassen af en partikel er 10^{-5} computermasser.



Figur 1: Masserne af stjernerne er fordelt efter en MB-fordeling. I dette eksempel er der 100000 partikler.

1.8 Masse-lysstyrkefordeling

Der findes forskellige[7] funktioner til at angive sammenhængen mellem en stjernes masse og dens lys. Jeg har antaget en simpel funktion på formen

$$\frac{L}{L_{Sol}} = \left(\frac{M}{M_{sol}} \right)^{3,5} \tag{17}$$

Hvis man ikke er tilfreds med ovenstående antagelse, kan man læse mere om forbedringer i fodnoten.

1.9 Hastighedsfordelingen for partiklerne

Man skal også vælge en måde at tildele hastigheder til partiklerne, med mindre man vil undersøge kollapset af en stillestående sky uden partikelhastigheder, hvilket dog nok er en kende urealistisk. Vi sætter totalmassen til 1. Her forelås at man som minium tilføjer normalfordelte hastigheder. Med 3 uafhængige normalfordelte hastigheder, viser det sig igen at en Maxwellfordeling beskriver farten af de enkelte partikler, men her er vi ikke interesseret i farten, men i hastigheden. Dog har vi brug for middelhastigheden og spredningen på hastighederne, når gausfordelingen skal bruges til at at vælge hastigheder.

Hvis massemidtpunktet af skyen står stille er middelværdien 0. Spredningen kan beregnes ud fra formlen

$$E_{kin} = \frac{1}{2} \cdot M_{total} \cdot \left(\sigma_x^2 + \sigma_y^2 + \sigma_z^2 \right) \tag{18}$$

Formlen gælder fordi middelværdien af hastighederne er 0, og fordi spredning og middelværdi er givet ved formlen

$$Var(X) = \sigma^2 = \frac{1}{N} \cdot \sum_1^N \left(x_i - \mu \right)^2 = \frac{1}{N} \cdot \sum_1^N x_i^2 = < x^2 > \tag{19}$$

I det sfærisk symmetriske tilfælde er $\sigma_i = \sigma_j$. Dvs.

$$E_{kin} = \frac{1}{2} \cdot M_{total} \cdot 3 \cdot \sigma^2 \Leftrightarrow \sigma = \sqrt{\frac{2 \cdot E_{kin}}{3 \cdot M_{tot}}} = \sqrt{\frac{-2 \cdot Q \cdot E_{pot}}{3 \cdot M_{tot}}} \tag{20}$$

Ved indsættelse får vi $\sigma = (-0{,}2 \cdot E_{pot}/3)^{0,5}$. Dvs. vi skal have fundet et udtryk for den potentielle energi af hele systemet. Det afhænger af den fordeling af masse for hele systemet, man vælger. Herunder er vist et eksempel for en sfærisk symmetrisk massefordeling, med R=1 og M=1.

Densiteten er givet ved (13). Energien på en masseskal *dm* fra den masse, *m(r)*, der ligger inden for dens radius *r* er givet ved

$$dE_{pot} = \frac{-G \cdot m(r) \cdot dm}{r} \tag{21}$$

7 https://en.wikipedia.org/wiki/Mass%E2%80%93luminosity_relation

Her er dm=$4\cdot\pi\cdot r^{2}\cdot dr\cdot\varrho$ og $m(r)=4\cdot\pi\cdot r^3/3\cdot\varrho$. Vi indsætter i (21) og integrerer op

$$E_{pot}=-G\cdot\int_0^R\frac{4\cdot\pi}{3}\cdot\frac{1}{r}\cdot\varrho\cdot r^3\cdot dm=-G\cdot\int_0^R\frac{4\cdot\pi}{3}\cdot\varrho\cdot r^2\cdot 4\cdot\pi\cdot r^2\cdot\varrho\,dr=\frac{-16\cdot\pi^2}{3}\cdot G\cdot\left(\frac{3\cdot M}{4\cdot\pi\cdot R^3}\right)^2\cdot\int_0^R r^4\cdot dr\Leftrightarrow$$

$$E_{pot}=\frac{-3\cdot G\cdot M^2}{5\cdot R} \tag{22}$$

Programmet *StjernerVelDisp.f90* udnytter ovenstående formler til at konstruere en sfærisk symmetrisk startfordeling, som input til *nbody100k.f90*.

2 Radialplots

Når man observerer en galakse eller en kuglehob, ser man en 2 dimensionel projektion af hob/galaksens lys, mens man i en nbody-simulering har en 3d-fordeling. Derfor skal man projicere 3d-fordelingen ind på en flade. Det gøres i programmet *nbody_analysis.f90*.

Programmet fitter derefter projektionen til hhv. en Sersic-profil og en King-profil. King-profiler passer godt til kuglehobe, mens Sersic-profiler er mere generelle, og kan tilpasse både kuglehobe og galakser.

2.1 Kingprofil-fit

King-fittene er fittet ud fra enten[8] en modificeret King-funktion

$$\Sigma(R)=\frac{k_0}{\left(1+\left(\frac{x}{r_c}\right)^2\right)^{\alpha}} \tag{23}$$

Eller[9] fra

$$\Sigma(R)=k\cdot\left(\frac{1}{\sqrt{1+\left(\frac{R}{r_c}\right)^2}}-\frac{1}{\sqrt{1+\left(\frac{r_t}{r_c}\right)^2}}\right)^2 \tag{24}$$

Ovenfor er kerneradius, r_c, og maksiumradius, r_t. Man ser, at hvis $R=r_t$, er $\Sigma(r_t)=0$, og $\Sigma(r_c)\approx 0{,}5$, fordi $r_t>>r_c$.

2.2 Sersic-profilfit

Sersic-fittene er fittet ud fra funktionen

$$\Sigma(R)=s_0\cdot\exp\left(-\left(2\cdot n-\frac{1}{3}\right)\cdot\left(\left(\frac{r}{R_e}\right)^{1/n}-1\right)\right) \tag{25}$$

8 *nbody_analysisStortRfit.f90*
9 *nbody_analysis.f90*

Hvis indekset n=4 får man en klassisk *de Vaucoleur-profil*, som ofte passer godt til E-galakser.

Det foreslås kun at fitte over kerneområdet i *nbody_analysis.f90*, dvs. for R<1, men brugeren spørges om R_{max}, så hvis resultatet indbyder til det, kan man forsøge sig med en større R.

3 Forslag til problemstillinger at undersøge

3.1 Et stjernesystem

En opgave for eleven kunne være at konstruere en startfordeling af legemer i et færdigt stjernesystem. For eksempel kan eleven simulere vores solsystem, og så kan opgaven være at tilegne passende masser og hastigheder samt afstande for kloderne, så planetsystemet bliver stabilt. Her kan ε^2 i første omgang svare til en asteroide-radius.

3.2 Kuglehobe

Man kan også simulere kuglehobe hvor N i naturen ligger i intervallet 10^4-10^6 stjerner. Programmet fungerer dog kun op til ca. 10^5 stjerner på en alm. hurtig pc. Vælger man flere partikler går regnetiden voldsomt op.

Hvis man skal simulere systemer med flere end 10^5 stjerner, må man bundte stjernerne i småhobe, dvs. masserne af det samlede antal partikler skal svare til hobens masse. Hvis man vil simulere kuglehobe, kan det anbefales at læse om Plummer- eller King-modeller, som beskriver hvordan massefordelingen skal være i en kuglehob. [2], [3].

3.3 Galakser

I de vedlagte programmer findes programmet Jaffe.exe. Det simulerer en lysfordeling, der svarer til en E0-galakse. Hvis masse-lysstyrkeforholdet er konstant, vil fordelingen også passe med massefordelingen. Jaffe.exe tager altså ikke højde for variationen af masse-lysstyrkeforholdet i galakserne. Jaffefordelingen er i øvrigt stabil, så skal man simulere kollisioner, vil denne model være egnet til at lave startgalakser. Softeningparameterens værdi ved Jaffefordelingen er $\varepsilon^2 = (3{,}7 \cdot r_0 \cdot N^{-1/3})^2$, hvor r_0 er *halv*masseradius af fordelingen. Den er i Jaffe.exe defineret til 1. Halvmasseradius betyder, at hvis man akkumulerer fordelingens masse, så vil man finde halvdelen af fordelingens masse indenfor halvmasseradius.

Hvis man er universitetsstuderende, kan man overveje at lade Jeans-ligningerne lave en stjernefordeling. Se [1] for mere information om Jeans-ligningerne.

Regnetiden i simuleringsprogrammet går som $O(N^2)$. Det betyder, at hvis det tager tidsrummet T at afvikle en simulering med N partikler, så tager det altså i omegnen af $4T$ at afvikle en simulering med $2N$ partikler. At køre med $100N$ partikler kræver altså $100^2 \cdot T = 10^4 T$.

3.4 Analyse af resultaterne

Foruden at analysere sted-koordinater for legemer, kan man også overveje at kigge på inertimoment, impulsmoment, form af legemepopulationer, overfladelysstyrke (tænk evt. på *King*-profiler eller *Sersic/de Vaucoleurs*-profiler) osv. I programarkivet, der hører til denne note, findes et kombineret Fortran/python-program, *nbody_analysis.f90*, som kan hjælpe med analyse og grafer af

akseforhold og radialplots. Programmet virker kun, hvis man har Python installeret med bibliotekerne *numpy, scipy, os* og *matplotlib* installeret.

4 Eksempel 1: Dannelse af kuglehob

I denne simulering vil vi forsøge at gendanne strukturen af en kuglehob. Det sker ved at lave en sfærisk symmetrisk startfordeling med konstant densitet og E_{kin}=0,1|E_{pot}|. Hvorfor lige 0,1? Fordi testkørsler med værdier mindre og større end 0,1 ikke giver særligt gode slutresultater. Den kinetiske energi er givet til partiklerne ved normalfordelite tilfældigt valgte hastigheder.

Dvs. i dette tillfælde får vi fra (22) at E_{pot} = -0,60 og derfor er E_{kin}=0,06. Den mekaniske energi er dermed -0,54, og det bør man sammenligne med simuleringens energiberegninger, som gemmens i filen *Energi.txt*. Hvis energien afviger, er der noget galt. (Sandsynligvis med størrelsen af η.)

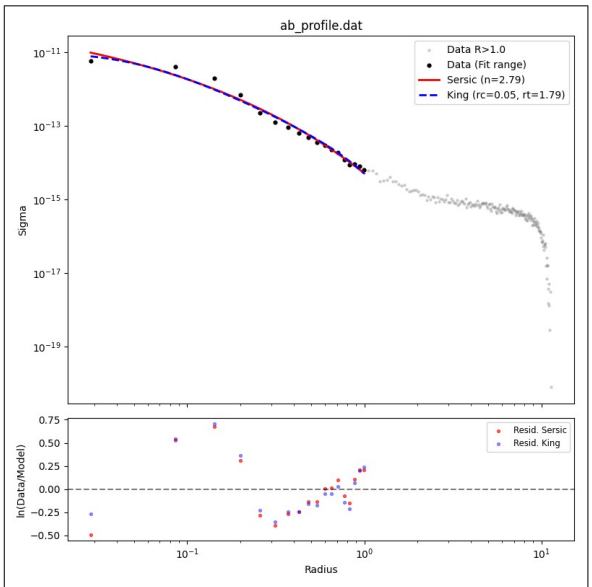
Ved brug af (3) og ε=0,5·λ fås softeningparameteren til 0,017 for N=100 000, men jeg vælger for en sikkerhedsskyld parameteren til at være 0,01. Energiparameteren vælges til η =0,015.

På notens forsidebillede kan man se startfordelingen og resultatet til T=7,99.

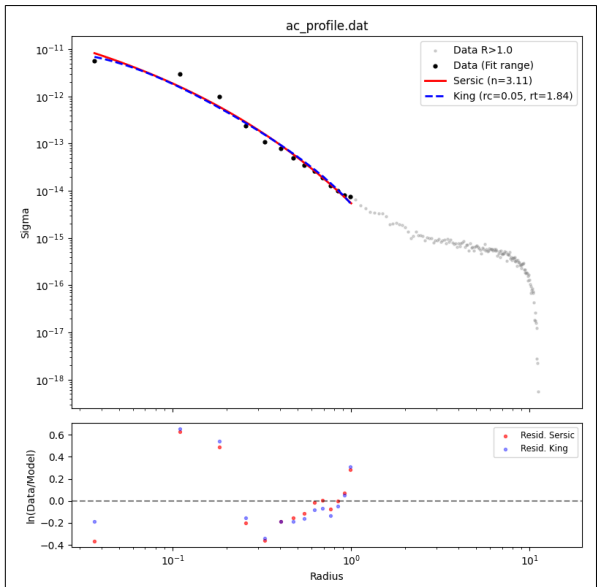
4.1 Analyse af lysfordelingen

Ved anvendelse af programmet *nbody_analysis.f90* får man følgende resultater for formen og lysfordelingen af den færdige hob.

Akseforholdene for den 3 dimensionelle fordeling er *b/a*=0,9768 og *c/a*=0,9719, dvs. fordelingen er forblevet sfærisk symmetrisk, hvilket giver god mening, da starthastighedsfordelingen også var sfærisk symmetrisk. Radialplottene er vist herunder.



Figur 3: AB-Planet. Q=0,1, ε=0,01, R<1.

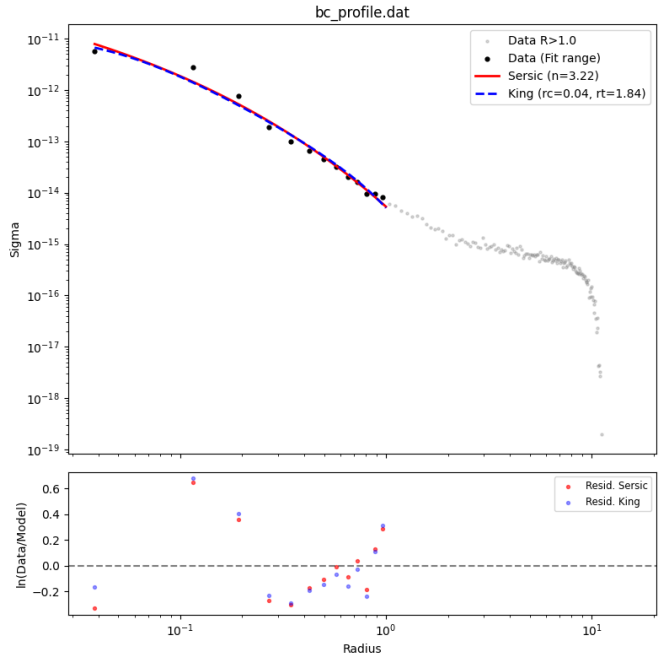


Figur 2: AC-planet. Q=0,1, ε=0,01, R<1.

Der er fittet ud til den oprindelige radius, da fittene bliver dårlige, når partikeltætheden falder drastisk, og ved R=1, er lysstyrken allerede faldet med en faktor 1000 ift. kernen.

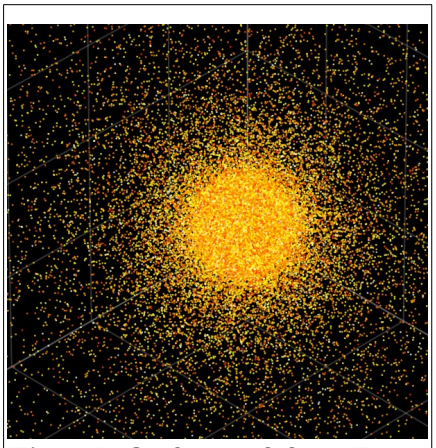
Koncentrationen, *c*, er et mål for tætheden af en kuglehobs lysfordeling, når den er fittet med en King-profil. Man kan beregne koncentrationen som

$$c = \log_{10}\left(\frac{r_c}{r_e}\right) \tag{26}$$



Figur 4: BC-planet.Q=0,1, ε=0,01, R<1.

	AB	**BC**	**AC**	**Gns**
n	2,79	3,22	3,11	3,04
c	1,57	1,62	1,60	1,60



Figur 5: Q=0,1, ε=0,01. MB-masse og hastighedsfordeling.

Herunder er en tabel over nogle tilfældige kuglehobe i Mælkevejen.

Tabel 1: Et tilfældigt udpluk af kuglehobe i Mælkevejen.

Navn (NGC/Messier)	**Koncentration (*c*)**	**Sersic-indeks (*n*)**	**Bemærkning**
Omega Centauri	1,31	~1,1	Mælkevejens største; meget lav koncentration.
M13 (Hercules)	1,5	~1,0	Den klassiske nordlige kuglehob.
M22	1,38	~0,9	En af de tætteste på Jorden; ret "løs".
47 Tucanae	2,05	~2,2	Meget koncentreret og massiv.
M15	2,50 (post-CC)	~3,5	**Core-collapsed**; ekstremt høj tæthed i midten.
M80	1,73	~1,6	En ret kompakt og stjernerig hob.
M4	1,59	~1,2	En "gennemsnitlig" hob tæt på Scorpius.
NGC 6397	2,50 (post-CC)	~3,8	Den næsttætteste hob; også kernekollapset.
Simulering med Q=0,1 MB-hastighedsfordelingog massefordeling. *ε*=0,01.	**1,60**	**3,04**	**Simuleringens koncentration passer til M4, M13, M80.**

Vi mangler at få oversat computerens enheder til rigtige enheder, så lad os starte med længde- og tidsskaler.

4.2 Fordelingens startradius

Vi skal finde størrelsen af nebulaen før kollapset.

En nebula, der kollapser til en kugle er beskrevet i *Elmegreen*[10]. Massen er ca. 5 MM_{sol} og partikeldensiteten er ca. $n=10^{12}$ m^{-3}. Skyen har N_0 partikler, (*H* og *He*) dvs. dens startvolumen kan bestemmes ved $n \cdot V_0 = N_0$.

I denne simulering vil jeg dog antage, at hver *computer*partikel svarer til 10 stjerner, så jeg vælger en mindre startmasse M_0=0,5 MM_{sol}. Dvs. hver partikel vejer 5M_{sol}.

Vi antog, at nebulaen er sfærisk symmetrisk, dvs. dens startradius er $R=\left(\frac{3\cdot N_0}{4\cdot\pi\cdot n}\right)^{1/3}$.

Lad os antage, at skyen består af X=75 %, Y=25 % og Z=0. Vi antager dermed, at der ikke er støv. Vi skal finde N_0.

$$N_H=\frac{X\cdot M_0}{m_H}\wedge N_{He}=\frac{Y\cdot M_0}{m_{He}}\wedge N_0=N_H+N_{He}. \quad \text{Det giver} \quad N_0=\left(\frac{X}{m_H}+\frac{Y}{m_{He}}\right)\cdot M_0.$$

Dermed kan vi bestemme nebulaens radius til

$$R=\left(\frac{3\cdot M_0}{4\cdot\pi\cdot n}\cdot\left(\frac{X}{m_H}+\frac{Y}{m_{He}}\right)\right)^{1/3} \tag{27}$$

Ved indsættelse af talværdier får vi $R=4{,}88\cdot10^{16}\,\mathrm{m}=1{,}58\cdot10^{-3}\,\mathrm{kpc}$.

Vi får nogle gange brug for *halvmasseradius* r_0, som er den radius hvor halvdelen af massen ligger. Dvs.

$$\frac{1}{2}\cdot M_0=\frac{4}{3}\cdot\pi\cdot\varrho\cdot r_0^3 \tag{28}$$

Vi kan bortsubstituere densiteten, da den jo er konstant, da vi antog en konstant antalsdensitet, *n*. Så får vi

$$r_0=\frac{R}{2^{1/3}} \tag{29}$$

4.3 Enhedsomregning

Vi bruger formlerne (8)-(11), hvor vi kan indsætte *a* og *b*. Vi har allerede, at b=1,58·10^{-3}.
$M=0{,}5$ MM_{sol} , dvs. a=5·10^{-5}.

Vi indsætter i formlerne:

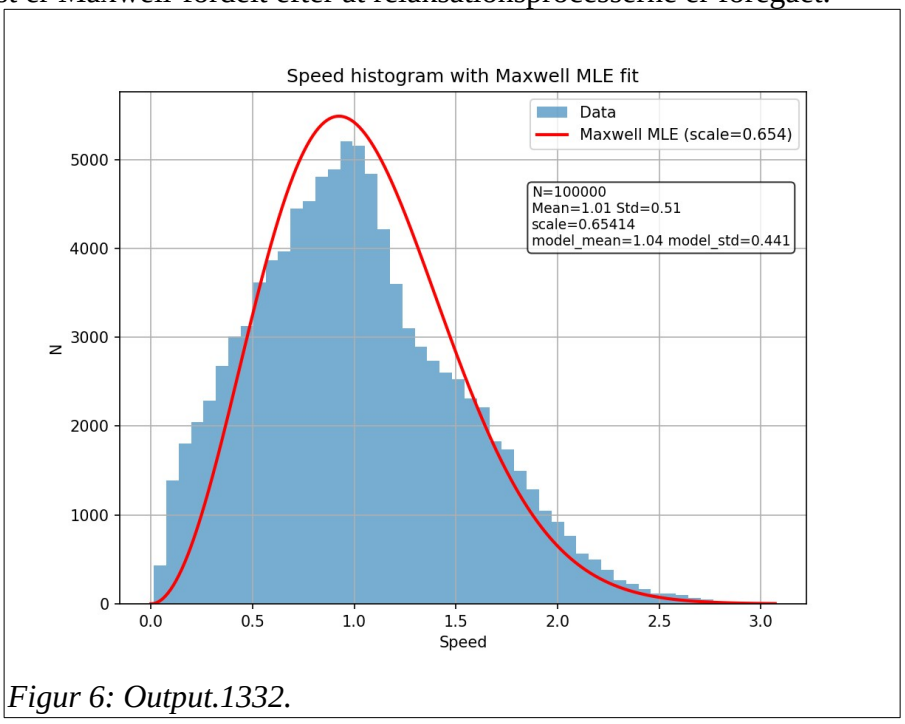
10 https://iopscience.iop.org/article/10.3847/1538-4357/836/1/80/pdf

$$
\begin{aligned}
1\,v_{computer} &= 207{,}4\cdot\sqrt{\frac{a}{b}}\,\frac{\mathrm{km}}{\mathrm{s}} = 3{,}69\cdot10^{6}\,\frac{\mathrm{km}}{\mathrm{s}} \\
1\,T_{computer} &= 4{,}715\cdot10^{6}\cdot\sqrt{\frac{b^{3}}{a}}\,\mathrm{yr} = 41{,}88\,\mathrm{kyr} \\
1\,E_{computer} &= 8{,}556\cdot10^{50}\cdot\frac{a^{2}}{b}\,\mathrm{J} = 1{,}354\cdot10^{45}\,\mathrm{J} \\
1\,l_{computer} &= 3{,}086\cdot10^{19}\cdot b\cdot 1\,\mathrm{m} = 1\,\mathrm{kpc}\cdot b\,.
\end{aligned}
\tag{30}
$$

Derfor kan vi omregne computerens enheder til SI-enheder, når simuleringen er færdig.

4.4 Hastighedsfordeling

Herunder er vist en graf over hastighederne efter simuleringen har kørt. Bemærk at hastigheden kun tilnærmelsesvist er Maxwell-fordelt efter at relaksationsprocesserne er foregået.



Figur 6: Output.1332.

4.5 Energiudvikling

Herunder ses en graf, der viser hvorden den mekaniske energi har udviklet sig over hele kørslen.

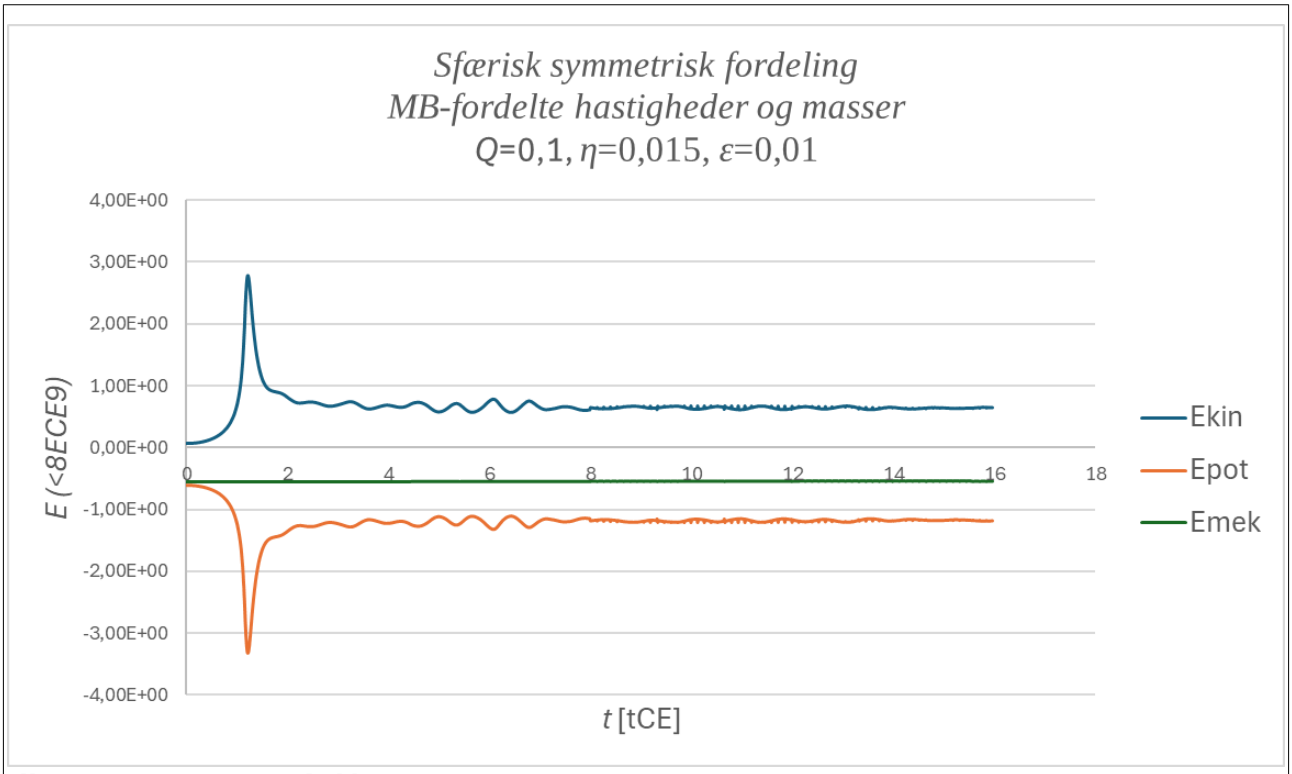


Illustration 1: Energiudvikling.

5 Eksempel 2: Galakse-simulering

Sfærisk symmetrisk fordeling, konstant densitet og jævn cirkelbevægelse

5.1 Indledning

Simuleringen er foretaget ved hjælp af Sverre Aarseths program, som findes bagest i bogen ”Galactic Dynamics” af James Binney og Scott Tremaine. Princeton Series in Astrophysics, Princeton University Press, 1987. Vejledning i brug af programmet kan f.eks. ses på www.astro-gym.dk under punktet Nbody.

Man kan compilere sine programmet ved hjælp af compileren gfortran. Foruden Libreoffice er programmet TOPCAT[11] anvendt til at tegne grafer.

Herunder er grundindstillingerne til NBody-programmet anført.

N=10000.	Antal partikler i simuleringen.
η=0,02.	Præcision af beregningerne – hvis E ikke er bevaret, skal man ændre denne værdi.
dt=0,15.	Tidsstep i computerenheder.
ε^2=0,0005.	Softeningparameter. (For at undgå overflow ved kollisioner.)
M=1,0.	Alle partikler har fået samme masse.

Simuleringen har kørt tiden $T_{simulering} = 556 \cdot dt = 83{,}4$. (I computertider.)
Startfordelingen har en *hel*masseradius r_0 = 1. Densiteten er antaget konstant, og der er lagt jævn cirkelbevægelse ind fra starten. Koordinatsystemet er anbragt, så fordelingen har z-aksen som rotationsakse.

11 http://www.star.bristol.ac.uk/~mbt/topcat/

5.2 Startbetingelser

5.2.1 Stedkoordinaterne

Densiteten $\rho=\frac{M}{V}=\frac{3\cdot M}{4\cdot\pi\cdot r_0^3}=\frac{3\cdot N\cdot m}{4\cdot\pi\cdot r_0^3}$. Der er konstrueret 100 kugelskaller rundt om centrum, og antallet af partikler i hver skal er dermed givet ved:

$$dV=4\cdot\pi\cdot r^2\cdot dr\Leftrightarrow dM=\rho\cdot dV=4\cdot\pi\cdot r^2\cdot\rho\cdot dr\Leftrightarrow dN=\frac{dM}{m}=\frac{4\cdot\pi\cdot r^2\cdot\rho\cdot dr}{m}=\frac{4\cdot\pi\cdot r^2\cdot dr\cdot 3\cdot N\cdot m}{m\cdot 4\cdot\pi\cdot r_0^3}\Leftrightarrow$$

$$dN=\frac{3\cdot N\cdot r^2\cdot dr}{r_0^3}.$$

Selve fordelingen er nemmest at generere, når man anvender polære koordinater. Her skal man huske at et overfladeelement $dS=\sin(\theta)\,d\theta\,d\phi$. Dvs. man kan generere tilfældige dφ-værdier i intervallet [0, 2π[, mens $-dcos(\theta)$ skal genereres jævnt i intervallet [0, π[. Radius r vokser i spring af dr = r_0/100.

Stedkoordinaterne kan dermed skrives ved hjælp af formlerne:

$$x=r\cdot\sin(\theta)\cdot\cos(\phi)$$
$$y=r\cdot\sin(\theta)\cdot\sin(\phi)$$
$$z=r\cdot\cos(\theta).$$

5.2.2 Hastighedskoordinaterne

Når man vælger jævn cirkelbevægelse *og* samme fart for samtlige partikler skal alle partikler have samme omløbstid, T. Specielt kan man betragte en partikel, der ligger i maksiumstartafstanden, r_0, og som roterer omkring z-aksen nede i xy-planet. For den partikel gælder følgende:

$$v_{rot}(r_0)=\frac{2\cdot\pi\cdot r_0}{T}\wedge F=\frac{m\cdot v_{rot}^2}{r_0}=\frac{G\cdot(M-m)\cdot m}{r_0^2}\Leftrightarrow$$

$$\frac{m\cdot\left(\frac{2\cdot\pi\cdot r_0}{T}\right)^2}{r_0}=\frac{G\cdot(M-m)\cdot m}{r_0^2}\Leftrightarrow\frac{4\cdot\pi^2\cdot r_0^3}{T^2}=G\cdot(M-m)\Leftrightarrow T=\frac{2\cdot\pi\cdot r_0^{3/2}}{G\cdot(M-m)^{\frac{1}{2}}}\approx\frac{2\cdot\pi\cdot r_0^{3/2}}{G\cdot M^{\frac{1}{2}}}=2\cdot\pi.$$

Ved den sidste beregning har vi anvendt, at computeren regner med G=1, og vi havde valgt totalmassen M = 1 ved radius r = r_0 = 1. Da T var valgt til at være konstant, gælder værdien af T for alle partikler.

For en vilkårlig partikel - altså også en partikel, der ikke bevæger sig i xy-planet - i afstanden r_z fra z-aksen er rotationsfarten altså $v(r)=\frac{2\cdot\pi\cdot r_z}{T}=\frac{2\cdot\pi\cdot r_z}{2\cdot\pi}=r_z$.

Til slut skal en enhedsvektor for v_i findes. Hastighederne er vinkelret på stedvektorerne, når der er tale om cirkelbevægelse, og derfor må $\vec{e}_v=\frac{\hat{\vec{r}}_z}{r_z}$. Altså kan man for enhver partikel skrive hastighedsvektoren som $v(\vec{r}_z)=r_z\cdot\vec{e}_v=\frac{r_z\cdot\hat{\vec{r}}_z}{r_z}=\hat{\vec{r}}_z$. På koordinatform skrives $(vx, vy, vz)=(-y, x, 0)$.

5.3 Enhedsomregninger

For at simuleringerne kan køre hurtigst muligt, regner Nbody-programmet i naturlige enheder, hvor G = M = r_0 = 1. Disse enheder skal omregnes til mere forståelige enheder, når man skal fortolke resultaterne. Hertil bruges Keplers 3. lov $\frac{a^3}{T^2}=\frac{G\cdot(M+m)}{4\cdot\pi^2}$.

Hastigheden

Vi fandt før, at $T=2\cdot\pi$. Specielt gælder $v(r_0)=\sqrt{\frac{G\cdot(M-m)}{r_0}}=1$ [computerenheder.]

Tiden

Vi manipulerer med Keplers 3. lov, og får:

$$T_0=\sqrt{\frac{4\cdot\pi^2\cdot r_0^3}{G\cdot(M+m)}}\approx 2\cdot\pi\,[\text{computerenheder}],\ \text{da } m \ll M.$$

Energien

Her benytter vi formlen for den potentielle energi af en partikel:

$$E_0=\frac{-G\cdot M(r_0)\cdot m}{r_0}=-\frac{1}{N}\,[\text{computerenheder}].$$

Eksempel: Roterende galaksesky

Lad os betragte en galakse med M = a·10^{10}·M_{sol} og r_0 = b kpc. Sættes disse parametre ind i Keplers 3. lov i stedet for ovenstående parametre får man:

$$v_0=146{,}6\cdot\sqrt{\frac{a}{b}}\,\frac{km}{s}$$

$$T_0=1{,}4881\cdot10^{14}\cdot\sqrt{8}\cdot\pi\cdot\sqrt{\frac{b^3}{a}}\,s$$

$$E_0=\frac{4{,}276\cdot10^{50}\cdot a^2}{b\cdot N}\,J.$$

Vi kan altså nu finde den direkte sammenhæng mellem computerenheder og SI-enheder:

$$1v_{computer}=146{,}6\cdot\sqrt{\frac{a}{b}}\,\frac{km}{s}$$

$$1T_{computer}=6{,}669\cdot10^{6}\cdot\sqrt{\frac{b^3}{a}}\,yr$$

$$1\,E_{computer}=4{,}276\cdot10^{50}\cdot\frac{a^2}{b}\,J$$

$$1l_{computer}=19{,}385\cdot10^{19}\cdot b\cdot m.$$

Hvis a = 20 har vi en galakse af Mælkevejens masse. Hvis vi antager at galaksen er lavet af en masse dværggalakser eller store kuglehobe af m = M/N, hvor N = 10000, svarer hver partikel i simuleringen altså til en masse på 10^7 M_{Sol}. Lad os antage at galaksen er lavet af en sky, der oprindeligt havde en radius på 10 kpc (altså en kugle med ca. samme størrelse, som den nuværende skives udbredelse), kan vi nu omregne computerens enheder til menneskeenheder, og vi kan sammenligne simuleringens output med rigtige målinger.

Nedenfor følger resultaterne fra simuleringen. Først følger en serie faserumsplots, der illustrerer partikelradierne og deres impulser. Radierne er målt i forhold til det oprindelige massemidtpunkt og altså ikke den aktuelle fordelings massemidtpunkt. Impulserne er beregnet i det samme koordinatsystem som radierne.

Derefter følger nogle billeder af slutkonfigurationen, og akseforhold samt afstande bestemmes.

5.4 Totalenergien

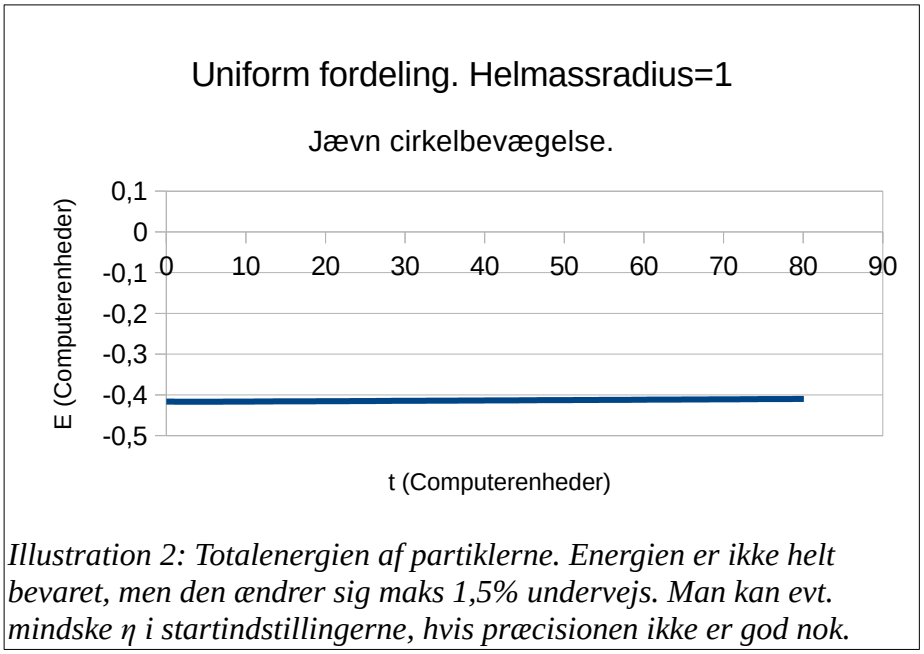
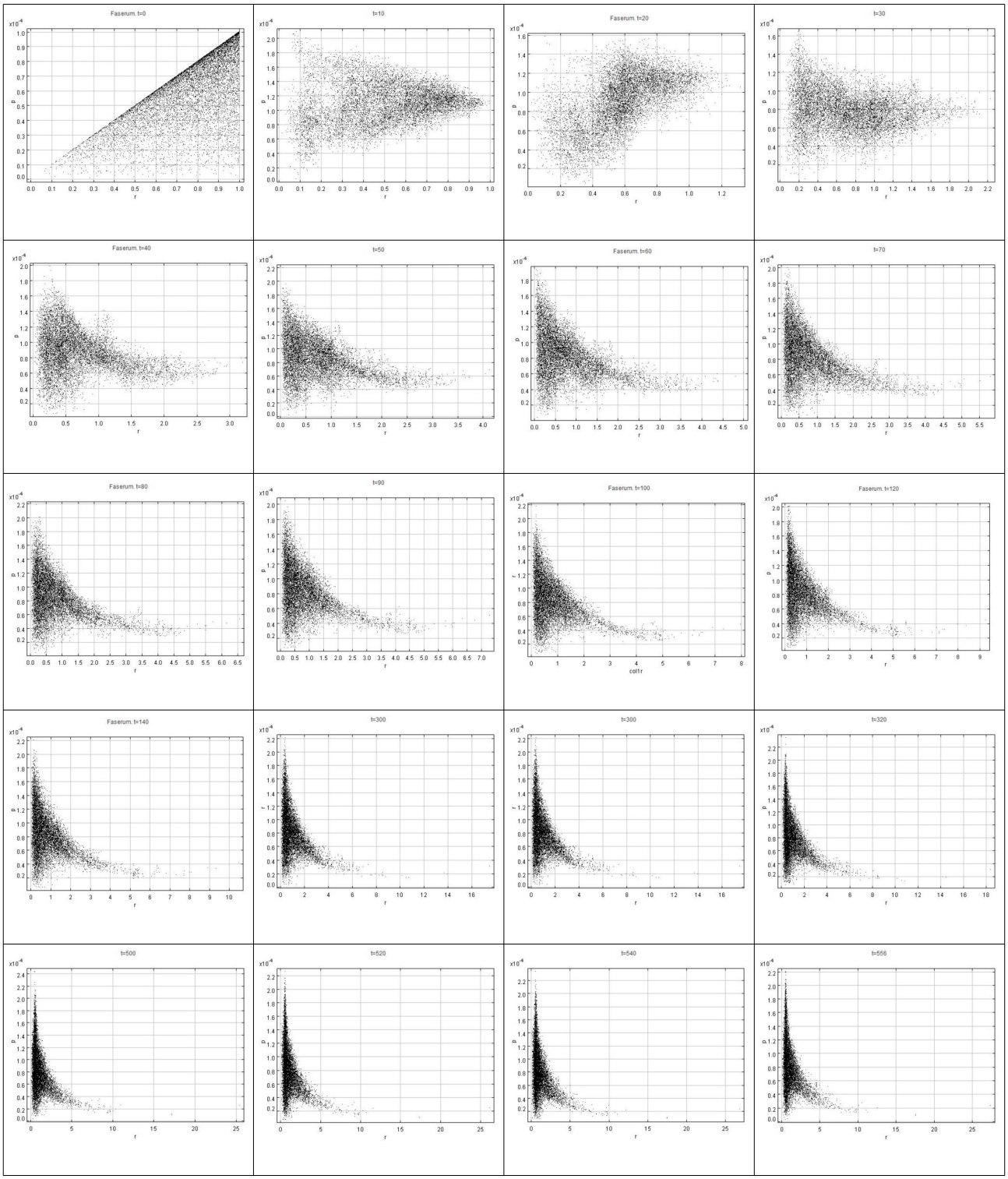


Illustration 2: Totalenergien af partiklerne. Energien er ikke helt bevaret, men den ændrer sig maks 1,5% undervejs. Man kan evt. mindske η i startindstillingerne, hvis præcisionen ikke er god nok.

Hvis a = 20 og b = 10 bliver $E = -6{,}8 \cdot 10^{51}\, J$.

5.5 Faserumsplot

Bemærk at der bør stå dt i overskrifterne. Den rigtige computertid er 0,15 x det tal, der står i overskrifterne.



5.5.1 Analyse af faserumsplot

Man ser, at da $\vec{v_{xy}} = \hat{\vec{r}_{xy}}$, og $v_{i,z} = 0$ ligger maksimalhastighederne af partiklerne langs en graf med hældningen 1, og resten ligger under. I løbet af simuleringen ser man, at afstandene falder, mens

hastighederne stiger. Vi får altså en gravitationel sammentrækning (*sic)*, mens de enkelte partiklers hastigheder vokser – Eftersom $v(r)=\sqrt{\frac{G\cdot M(r)}{r}}$, kan man se, at når den samlede masse ikke stiger, mens r falder, så vil v(r) vokse.

5.6 Positionsgrafer

På de følgende grafer kan man se billeder af slutkonfigurationen. T = 83,4 computerenheder, og hvis a = 20 og b = 10 giver det i menneskeenheder et forløbet tidsrum på 47,2 Myr.

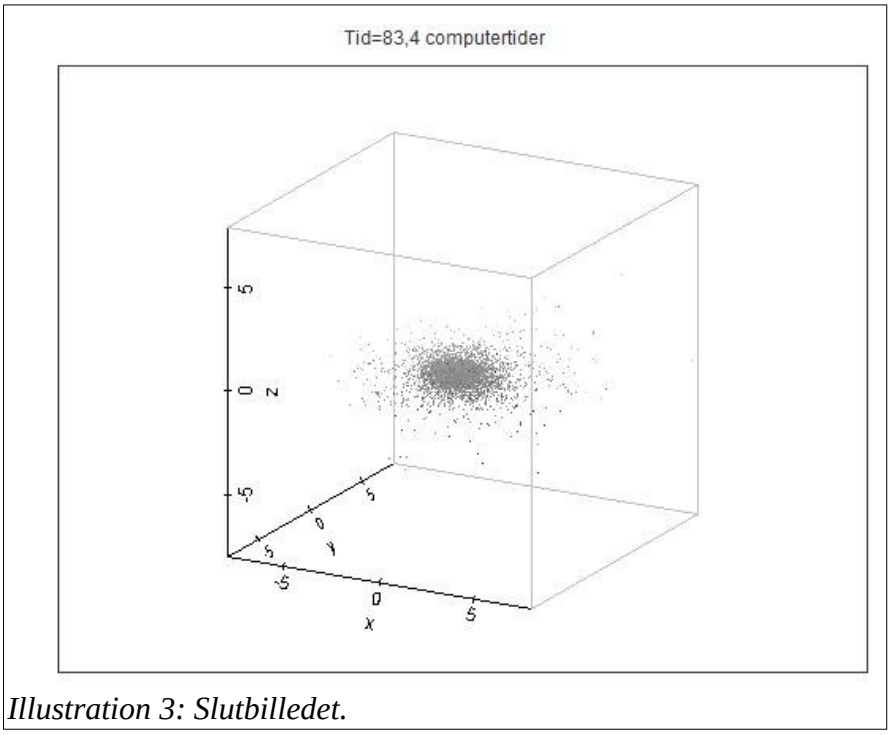
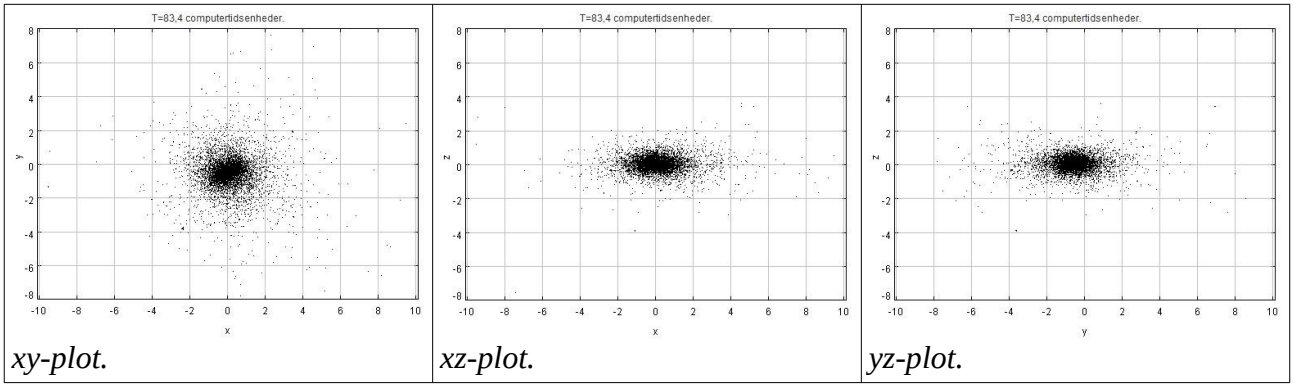


Illustration 3: Slutbilledet.



xy-plot. *xz-plot.* *yz-plot.*

Man kan af graferne se, at fordelingen er ret cirkulær i xy-planet, hvilket nok ikke overrasker, da der er et pænt impulsmoment til at bibeholde den oprindelige form. Langs z-aksen er fordelingen klappet sammen, og det kan forklares ved, at gravitationen trækker partiklerne ind mod centrum, og der er ikke noget stort impulsmoment ved store z-værdier, som kan forhindre sammentrækningen.

Ved at aflæse på forstørrelser af billederne, kan akseforholdene bestemmes. Se nedenfor.

	xy-planet	**xz-planet**	**yz-planet**
$\mathbf{a_{kerne}}$	0,94	1,69	1,41
$\mathbf{b_{kerne}}$	0,47	0,658	0,66
$\mathbf{a_{skive}}$	1,88	3,29	3,29
$\mathbf{b_{skive}}$	1,88	1,13	1,13
$\frac{\mathbf{b}}{\mathbf{a}_{\mathbf{kerne}}}$	0,5	0,39	0,47
$\frac{\mathbf{b}}{\mathbf{a}_{\mathbf{skive}}}$	1	0,34	0,34

Omregningsfaktoren for længde er $1l_{computer} = 19{,}385 \cdot 10^{19} \cdot 10 \cdot m = 1{,}9385 \cdot 10^{21}\, m = 72{,}8\, kpc.$ Dermed bliver vores model-galakses skiveradius 240kpc. Dette er jo 24 gange så stor som Mælkevejens faktiske radius og dermed kan modellen ikke forklare Mælkevejens dannelse. Man kan diskutere om sammentrækningen er færdig, men faseplottene er jo ret konstante, hvilket antyder, at fordelingen ér faldet til ro.

Mht akseforholdene, så er kernens forhold for Mælkevejen 0,85, mens den for M31 er 0,54. Denne model har altså en ret flad struktur i forhold til spiralgalakser.

5.7 Bilag

5.7.1 Startfordeling

```
! Et program til at danne en fordeling af stjerner (Det kan udnyttes til at regne ud fra Jeans-ligningerne.)
! Lige nu spreder den bare en masse partikler jævnt ud i rummet, og de står alle stille i begyndelsen.

module Definitioner
 implicit none
 integer, parameter :: dbl=selected_real_kind(p=8,r=99) ! 8 dec. ønsket. 99 i 10-tals eksponent.
 integer,parameter :: fillaengde=30
end module Definitioner

Program Stjerner
 use Definitioner
 implicit none

 ! Initialisering
 real(KIND=dbl), allocatable, dimension(:,:) :: pos,vel
 real(KIND=dbl) :: dr,r,rr,r0,pi,theta,cth,fi,ran
 real(KIND=dbl),allocatable,dimension(:) :: masse ! En skalar er eg. nok, men i fremtiden kan m(i) f.eks. varieres.
 integer :: err,i,ii,j,N,k,n_r,M ! N er antallet af partikler.
 character(len=fillaengde) :: Udskriftsfil
 Udskriftsfil="Stjernefordeling.txt"
 Udskriftsfil=TRIM(Udskriftsfil)
 ! Indlæs antal partikler.
 print *,"Indtast antal partikler. (Indtast et heltal!) :"
 read *, N

 ! Arrays allokeres nu, og for en sikkerheds skyld sættes alt til 0_dbl.
 allocate (masse(1:N),pos(1:3,1:N),vel(1:3,1:N), STAT=err)
 if (err.ne.0) STOP "Fejl i Allokering"
 masse=0._dbl
 M=1._dbl
 pos(1:3,1:N)=0._dbl
 vel(1:3,1:N)=0._dbl
 do i=1,N
  masse(i)=M/real(N)
 enddo
 ! Beregn stedkoordinater.
 r0=1._dbl
 pi=3.14159265355
 dr=0.01*r0
 r=0._dbl
 ii=0
 k=INT(r0/dr)
 do j=0,k-1,1
  n_r=INT(3.*N/r0**3*r**2*dr)
  do i=1,n_r
  ! dS=sin(theta)*dfi*dtheta=-d(cos(theta))*dfi. Dvs. cos(theta) skal variere jævnt.
  cth=rand(234689*i*j)*2-1
  theta=acos(cth)
  fi=2*pi*rand(i*j*9082618)
  rr=rand(8797234*i*j)*dr+r
  pos(1,i+ii)=rr*sin(theta)*cos(fi)
  pos(2,i+ii)=rr*sin(theta)*sin(fi)
  pos(3,i+ii)=rr*cos(theta)
  vel(1,i+ii)=-pos(2,i+ii)
  vel(2,i+ii)=pos(1,i+ii)
  enddo
  ii=ii+n_r
  r=r+dr
 enddo
 ! Gem sted- og hastighedskoordinater.
 Open(Unit=1, File=Udskriftsfil, status='new',action='Write',Form='formatted')
```

```
 do i=1,N
!  write(Unit=1,fmt='(7(E14.6,2X))') masse(i),pos(1,i),pos(2,i),pos(3,i),vel(1,i),vel(2,i),vel(3,i)
 write(Unit=1,fmt='(1x,7(1pe14.6))') masse(i),pos(1,i),pos(2,i),pos(3,i),vel(1,i),vel(2,i),vel(3,i)

 enddo
 Close(1)

End Program Stjerner
```

5.7.2 Jaffe/Osipkov-Merrit-model

Vælg anisotropiparameter til 0, og en stabil sfærisk symmetrisk galakse dannes. Hvis anisotropiparmenteren vokser stiger ustabiliteten af galaksen.

```
      Program Jaffe
      integer j,N,ni,nj,idum1,idum2
      real x,y,z,vx,vy,vz,r,ra,theta,fi,m,Mtot,pos(3,260000)
      real vel(3,260000),pi,r0,dr,sr,st,sigr,sig0,G,epsvect
      real Mold,Mnew,sth,sfi,f1,f2,f3,my,fortegn,rr,vr,vth,vfi
      print*,'Indtast et stort heltal'
      read*,idum1
      print*,'Indtast antal partikler'
      read*,N
      print*,'Indtast totalmasse'
      read*,Mtot
      print*,'Indtast anisotropiparameteren'
      read*,ra
      print*,'Indtast eps^2'
      read*,epsvect
      m=Mtot/real(N)
      Mold=0.
      pi=3.1415926536
      G=1.
      r0=1.
      sig0=G*Mtot/r0
      print*,'Indtast dr'
      read*,dr
      r=0.
      nj=0
      idum1=-abs(idum1)
      idum2=idum1-764
      idum3=idum1-9689
 10   r=r+dr
      ni=int((Mnew(r,Mtot,r0)-Mold)/m+0.5)
      if (ni.eq.0) goto 10
      Mold=Mnew(r,Mtot,r0)
      sr=sigr(sig0,r,r0,ra)
      st=sqrt(sr**2/(1.+(r/ra)**2))
      do 20 j=1,ni
 1     my=ran1(idum1)
      my=fortegn(idum2)*my
      if (my.eq.-1.) goto 1
      theta=acos(my)
      fi=2.*pi*ran1(idum1)
      rr=r-ran1(idum1)*dr
      x=rr*sin(theta)*cos(fi)
      y=rr*sin(theta)*sin(fi)
      z=rr*cos(theta)
      vr=gasdev(idum3)*sr
      vth=gasdev(idum3)*st
      vfi=gasdev(idum3)*st
      vx=vr*sin(theta)*cos(fi)+vth*cos(theta)*cos(fi)-vfi*sin(fi)
      vy=vr*sin(theta)*sin(fi)+vth*cos(theta)*sin(fi)+vfi*cos(fi)
      vz=vr*cos(theta)-vth*sin(theta)
      pos(1,nj+j)=x
```

```
      pos(2,nj+j)=y
      pos(3,nj+j)=z
      vel(1,nj+j)=vx
      vel(2,nj+j)=vy
      vel(3,nj+j)=vz
 20    continue
      nj=nj+ni
      if ((Mold.lt.Mtot).and.(nj.lt.N)) goto 10
      open(unit=25,file='treebi',status='new')
      write(25,40) N
      write(25,41) 0.02
      write(25,41) 1.
      write(25,41) 50.
      write(25,41) epsvect
      do 30 i=1,N
       write(25,41) m,pos(1,i),pos(2,i),pos(3,i),vel(1,i),vel(2,i),vel
     + (3,i)
 30    continue
 40    format(1x,5(1i6))
 41    format(1x,7(1pe14.6))
      close(25)
      end
      real function Mnew(r,Mtot,r0)
      real r,Mtot,r0
       Mnew=Mtot*r/r0/(1.+r/r0)
      end
      real function sigr(sig0,r,r0,ra)
      real sig0,r,r0,ra,l,x,dum,dum1,dum2
      l=r0/ra
      x=r/r0
      dum=sig0/(1.+(l*x)**2)
      dum1=.5-2.*x-(9.+1.5*l*l)*x*x-(6.+l*l)*x**3
      dum2=-x*x*(1.+x)**2*(6.+l*l)*alog(x/(1.+x))
      sigr=sqrt(dum*abs(dum1+dum2))
      return
      end
      real function fortegn(idum)
      integer idum
      real tilftal
      tilftal=ran0(idum)
      if (tilftal.lt. .5) then
       fortegn=1.
      else
       fortegn=-1.
      endif
      return
      end
```

5.7.3 Faserumskoordinater

```
module Definitioner
 implicit none
 integer, parameter :: dbl=selected_real_kind(p=8,r=99) ! 8 dec. ønsket. 99 i 10-tals eksponent.
 integer,parameter :: fillaengde=30
end module Definitioner
Program Faserum
! Et program der beregner faserumsfunktionen.
 Use definitioner
 Implicit none
 ! Initialisering
 character(len=fillaengde) :: Filnavn,Fasefil,ekst
 integer :: Filnr,partikelnr,ios,err
 integer :: i,j,k,N,nstart, nslut
 real(KIND=dbl) :: step
 real(KIND=dbl),allocatable,dimension(:) :: m,r,p
 real(KIND=dbl),allocatable,dimension(:,:) :: pos,vel
```

```
 j=1
 k=0
 print *,'Skriv Datafilnavnet'
 read *,Filnavn
 print *,'Skriv startnummeret paa filen'
 read *,nstart
 print *,'Skriv slutnummeret paa filen'
 read *,nslut
 print *,'Skriv antallet af partikler'
 read *,N
 allocate (m(1:N),pos(1:3,1:N),vel(1:3,1:N),r(1:N),p(1:N),STAT=err)
 do i=nstart,nslut
  Write(ekst,'(i3)') i
  ekst=trim(adjustl(ekst))
  open(unit=1,file=trim(filnavn)//'.'//trim(ekst),status='old',action='read',form='formatted')
   do j=1,N
    read(unit=1,fmt=*,iostat=ios) partikelnr,m(j),step,pos(1:3,j),vel(1:3,j)
    if (ios<0) exit
    k=k+1
   enddo
  close(1)
  open(unit=2,file='Fasefil'//'.'//trim(ekst),status='new',action='write')
  do j=1,k
   r(j)=(pos(1,j)**2+pos(2,j)**2+pos(3,j)**2)**0.5
   p(j)=m(j)*(vel(1,j)**2+vel(2,j)**2+vel(3,j)**2)**0.5
   write(unit=2,fmt='(2(E14.6,2X))'),r(j),p(j)
  enddo
  k=0
  close(2)
 enddo
end program Faserum
```

6 Referencer

1. James Binney & Scott Tremaine: ”Galactic Dynamics”, Princeton University Press, 1987.
2. https://en.wikipedia.org/wiki/Plummer_model
3. http://www.ast.cam.ac.uk/~vasily/Lectures/SDSG/sdsg_7_clusters.pdf
4. Nørgaard et al: *Universets melodi*, Gyldendal Uddannelse, 2001.