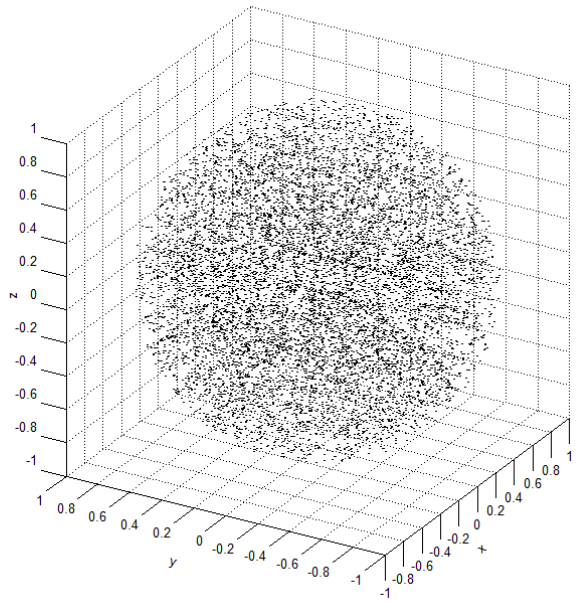
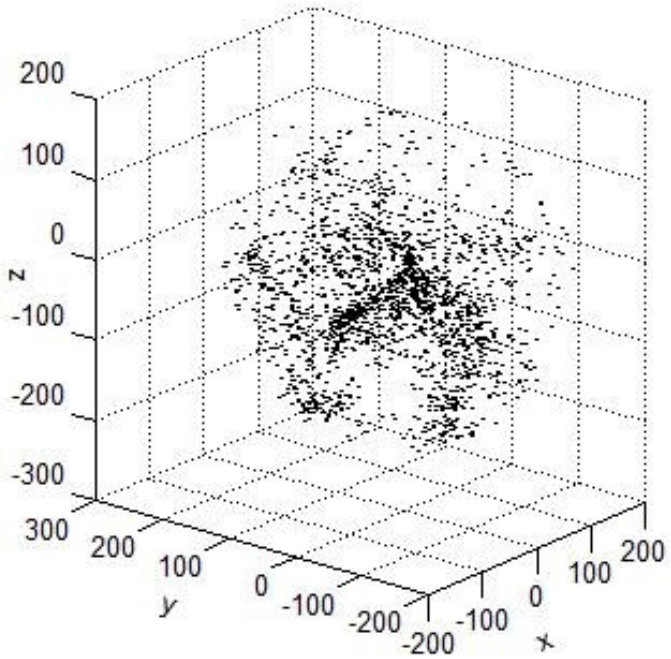




Jævn fordeling af partikler med ingen starthastighed udsat for gravitationskraften. Der er 10000 partikler i simuleringen og langt de fleste partikler findes inde omkring (0, 0, 0).

Indholdsfortegnelse

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.

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, 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 Fortran-kompiler kan man hente på webstedet www.g95.org.

Foruden kompileren kan det også være nyttigt at indlæse andre færdige programbiblioteker. Man kan finde en liste over biblioteker, der virker sammen med g95 på adressen http://g95.org/g95_status.shtml. Bemærk at mange af disse biblioteker skal købes, men de er også kun nødvendige for den avancerede bruger.

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 *g95 mitprogram.f90 -l**bibnavn*** (Husk at skrive henvisningen til biblioteket til sidst i linien.)[1]

Værktøjer

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 kan f.eks. være Notepad++. Det kan du hente på adressen, der er nævnt i fodnote.[2] Hvis det program ikke falder i din smag, er der andre i fodnote[3].

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, 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
```

1 I eksemplet har man præciseret, at man skriver i Fortran 90. Hvis filtypenavnet f.eks. kaldes f95 skal man overholde Fortran 95-standarden.
2 https://notepad-plus-plus.org/download
3 https://www.webpagefx.com/blog/web-design/12-excellent-free-text-editors-for-coders/

```
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.
```

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.

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.

Der findes mere avancerede programmer til fri download 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 dumpes 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.

Parametre

Der skal være konstrueret en tekstfil 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 eta mindre og så gentage simuleringen.

Deltat fortæller hvor tit, man skal skrive en fil ud med resultater. T_{crit} fortæller hvornår simuleringen skal stoppe. Endelig er ε^2 den såkaldte softeningparameter, der fysisk set sørger for, at stjernerne ikke ramler ind i hinanden. Det gør de nemlig ikke i virkeligheden, men da vi ikke kan arbejde med N = mange milliarder, vil problemet kunne opstå i simuleringen. Hvis afstanden mellem to legemer også bliver lille, vil beregningerne 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. Hvis ens totalenergi ikke stemmer særlig godt overens med en teoretisk værdi, kan man gøre ε^2 mindre. Men man skal huske, at beregningen af bevægelsesligningerne kan blive fejlagtig hvis ε^2 bliver *for* lille.

Forslag til problemstillinger at undersøge

Solsystem

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.

Kuglehobe

Man kan også simulere kuglehobe hvor N i naturen ligger i intervallet 10^4-10^6 stjerner. Programmet tillader dog kun op til 10^5 stjerner. 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].

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 halvmasseradius 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 $100^2 \cdot T = 10^4\, T$.

Beregnede størrelser

Foruden at analysere sted-koordinater for legemer, kan man også overveje at kigge på inertimoment, impulsmoment, form af legemepopulationer, overfladelysstyrke (tænk evt. på de Vaucoleurs-profiler) osv. Det er dog værd at bemærke, at programmering er en langsommelig proces, og hvis man som elev vil arbejde med simuleringer, vil mindre opgaver altid være at foretrække.

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

Galakser/Kuglehobe

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

Keplers 3. lov er:

$$\frac{r^3}{T^2} = \frac{G \cdot M}{4 \cdot \pi^2} \Rightarrow v(r_0) = \sqrt{\frac{G \cdot M(r_0)}{r_0}} = \frac{1}{\sqrt{2}} [\text{computerenheder}].$$

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

$$T_0 = \sqrt{\frac{4 \cdot \pi^2 \cdot r_0^3}{G \cdot M(r_0)}} = \sqrt{8} \cdot \pi\, [\text{computerenheder}].$$

Tilsvarende gælder for energien af en partikel at:

$$E_0 = \frac{-G \cdot M(r_0) \cdot m}{r_0} = \frac{-1}{2 \cdot N} [\text{computerenheder}].$$

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

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

$$T_0 = 1{,}4881 \cdot 10^{14} \cdot \sqrt{8} \cdot \pi \cdot \sqrt{\frac{b^3}{a}}\, s$$

$$E_0 = \frac{-4{,}276 \cdot 10^{50} \cdot a^2}{b \cdot N}\, J\,.$$

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

$$1\, v_{computer} = 207{,}35 \cdot \sqrt{\frac{a}{b}} \frac{km}{s}$$

$$1\, T_{computer} = 4{,}716 \cdot 10^6 \cdot \sqrt{\frac{b^3}{a}}\, yr$$

$$1\, E_{computer} = 8{,}552 \cdot 10^{50} \cdot \frac{a^2}{b}\, J$$

$$1\, l_{computer} = 3{,}086 \cdot 10^{19} \cdot b \cdot m = 1\, kpc \cdot b\,.$$

Formlen for 1 l_{computer} = b kpc tyder på, at vi har regnet rigtigt.

<u>Et stjernesystem</u>
Atter er G = 1 samt M = 1. Vi definerer $r_0 = R$, hvor R er stjernens radius, dvs. $M(r_0)$ = 1. En planet/ asteroides masse defineres til at være en brøkdel c af M, dvs. m = c·M = c. Derefter gentages beregningen fra før, og i computerenheder får man følgende relationer

$v(r_0)$ = 1 [computerenhed].
$T_0 = 2\pi$ [computerenhed].
E_0 = -c [computerenhed].

I SI-enheder defineres stjernen til at veje M = a·M_{Sol}, r_0 = b·R_{Sol}, $M(r_0)$ = a·M_{Sol}.
Det giver følgende relationer

$$v(r_0) = 436{,}81 \sqrt{\frac{a}{b}}\,\text{km/s}\,.$$

$$T_0 = 10004 \sqrt{\frac{b^3}{a}}\,\text{s}\,.$$

$$E_0 = -3{,}795 \cdot 10^{41} \frac{a^2 c}{b}\,\text{J}\,.$$

Dermed bliver sammenhængen mellem SI-enheder og computerenheder i dette eksempel som følger

$$1[v_{computer}] = 436{,}81 \sqrt{\frac{a}{b}}\,\text{km/s}\,.$$

$$1[T_{computer}] = 1592 \sqrt{\frac{b^3}{a}}\,\text{s}\,.$$

$$1[E_{computer}] = 3{,}795 \cdot 10^{41} \frac{a^2}{b}\,\text{J}\,.$$

Øvelse: Kontroller ovenstående beregninger.

I de næste sider er der to eksempler på rapporter over simuleringer, hvor man har antaget en startfordeling, der har konstant densitet, men hvor partiklerne alle laver jævn cirkelbevægelse.

De to eksempler kan læses uafhængigt af hinanden - derfor vil man opleve at noget af teksten går igen i de to eksempler.

Eksempel 1: Galakse-simulering

Sfærisk symmetrisk fordeling, konstant densitet og jævn cirkelbevægelse

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 compilere g95, som kan hentes på http://g95.org. Foruden Libreoffice er programmet TOPCAT[4] 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.

Startbetingelser

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:

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

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

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:

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

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

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 $(vx, vy, vz)=(-y, x, 0)$.

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 $\frac{a^3}{T^2}=\frac{G\cdot(M+m)}{4\cdot\pi^2}$.

<u>Hastigheden</u>

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

<u>Tiden</u>

Vi manipulerer med Keplers 3. lov, og får:

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

<u>Energien</u>

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 = $a\cdot10^{10}\cdot M_{sol}$ og r_0 = b kpc. Sættes disse parametre ind i Keplers 3. lov i stedet for ovenstående parametre får man:

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

$$T_0 = 1{,}4881 \cdot 10^{14} \cdot \sqrt{8} \cdot \pi \cdot \sqrt{\frac{b^3}{a}}\, s$$

$$E_0 = \frac{4{,}276 \cdot 10^{50} \cdot a^2}{b \cdot N} J.$$

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

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

$$1\mathrm{T}_{computer} = 6{,}669 \cdot 10^6 \cdot \sqrt{\frac{b^3}{a}}\, yr$$

$$1\, E_{computer} = 4{,}276 \cdot 10^{50} \cdot \frac{a^2}{b} J$$

$$1\mathrm{l}_{computer} = 19{,}385 \cdot 10^{19} \cdot b \cdot m.$$

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.

Totalenergien

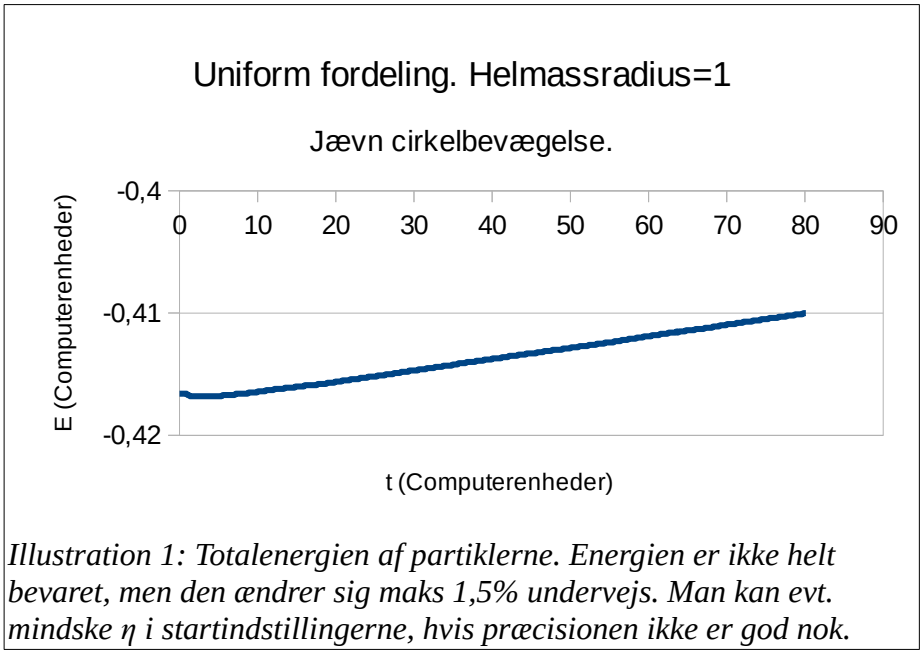
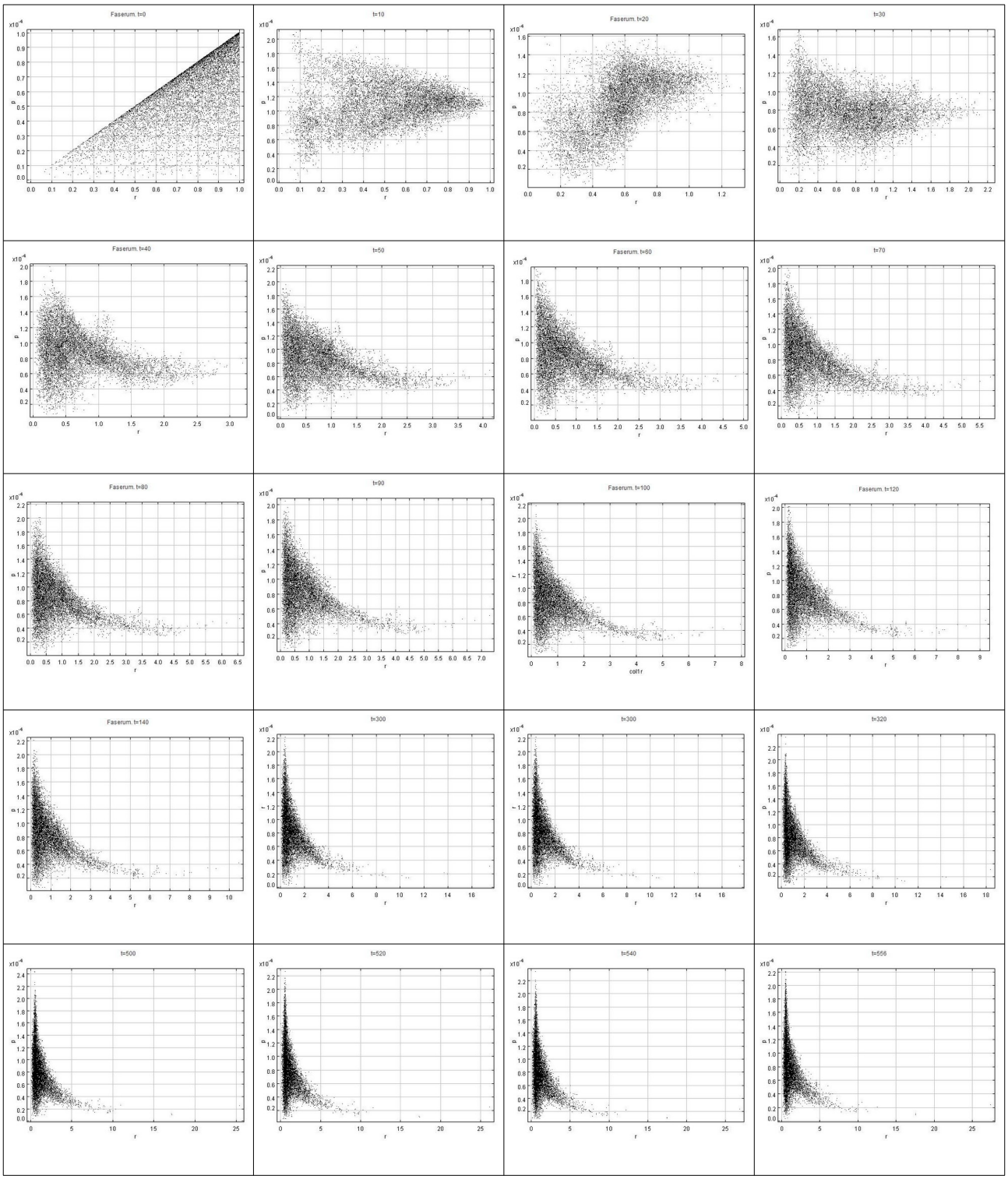


Illustration 1: Totalenergien af partiklerne. Energien er ikke helt bevaret, men den ændrer sig maks 1,5% undervejs. Man kan evt. mindske η i startindstillingerne, hvis præcisionen ikke er god nok.

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

Faserumsplot

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



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.

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.

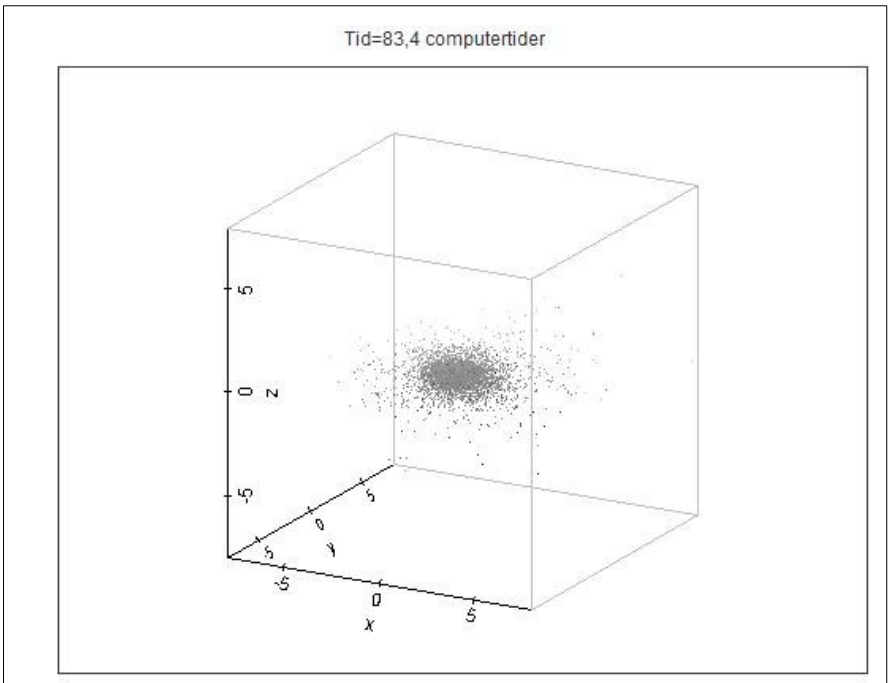
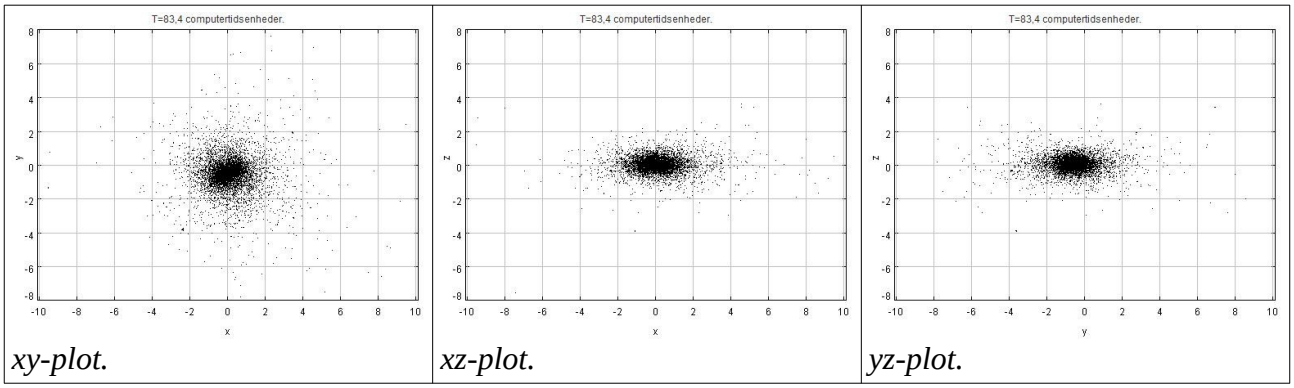


Illustration 2: Slutbilledet.



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.

Bilag

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
```

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
```

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
```

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.