# Projektbeskrivelse til Unge Forskere Senior

Læs hele teksten, før du udfylder skemaet.

Når du deltager i Unge Forskere, kan du enten være alene om dit projekt eller i en gruppe på op til 3 personer. Det er på baggrund af denne projektbeskrivelse, juryen udtager projekter til semifinalen.

I skemaet skal du beskrive projektet og idéen så detaljeret som muligt.
Felter markeret med * skal udfyldes.
Det er ikke sikkert projektet er helt færdigt, når du tilmelder dig konkurrencen. Det er helt okay, det vigtige er, at du beskriver, hvor du er i projektforløbet lige nu.

DU MÅ SKRIVE LIGE SÅ MEGET TEKST I DET ENKELTE FELT, SOM DU HAR BRUG FOR. SKRIV DIT SVAR MELLEM SPØRGSMÅLENE TIL DE ENKELTE PUNKTER OG DE VEJLEDENDE KOMMENTARER.

Der er ingen krav til, hvilket emne projektet undersøger, så længe du bruger den naturvidenskabelige eller ingeniørens arbejdsmetode i arbejdet.
Læs bedømmelseskriterierne her: https://ungeforskere.dk/konkurrencen/kriterier-senior

LÆS SKEMAET GRUNDIGT IGENNEM, OG HUSK AT INDHOLDET SKAL VÆRE LET AT FORSTÅ FOR JURYEN, DER INTET KENDER TIL PROJEKTET PÅ FORHÅND

Når skemaet er udfyldt, gemmes det som PDF og uploades på **deltag-ungeforskere.dk**
Er du i tvivl om noget, er du altid velkommen til at kontakte os på **info@ungeforskere.dk**

God fornøjelse!

***Navn og titel**

Navn(e): Elvin Chen

Projektets titel: Numerisk modellering af solsystemet

***Kategori** (sæt ét kryds):

Life Science _____ Physical Science ___x___ Technology _____

***Vigtig baggrundsviden**
Dette afsnit skal indeholde svar på:

1) Hvordan blev du opmærksom på det problem du vil undersøge/har undersøgt? (kort)

   Jeg har længe været interesseret i brugen af datalogiske metoder til at undersøge den virkelige verden. Da jeg tidligere havde læst, at bevægelsesbanerne i et gravitationelt system som solsystemet ikke kan løses analytisk, blev jeg interesseret i numeriske metoder, der anvendes til at skabe tilnærmede løsninger til sådanne systemer.

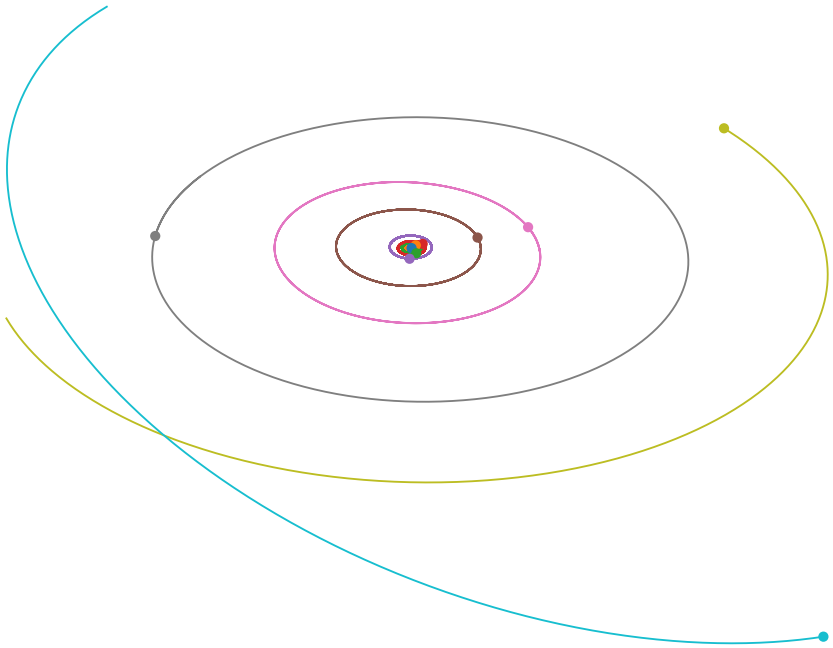
2) Hvorfor synes du, det er fedt at undersøge lige netop dette problem? (kort)

   Numeriske algoritmer kan anvendes til at løse mange fysikrelaterede problem. Der er derfor fedt at få indsigt i hvordan sådanne algoritmer kan implementeres i praksis. Digitale simulationer kan anvendes til at modellere fysiske fænomener, der ellers er upraktiske at udføre som et eksperiment. Man er derfor i stand til at skabe sit eget digitale univers, hvori alt kan varieres af brugeren. Derudover er det fedt at arbejde med lige præcis astronomiske systemer og planetbaner, da det er et område, hvor historisk data og målinger er tilgængeligt. Simulationerne kan derfor sammenlignes med virkeligheden.

3) Hvilken baggrundsviden har du inden for emnet? Beskriv centrale teorier og generel viden, som dit projekt bygger på. (VIGTIGT: angiv kilder, hvor det er relevant)

# Numerisk modellering af solsystemet

Fag: Fysik A, Programmering C



Elvin Chen | 3.B

2020

Vejledere: Morten Hesselberg Grove, Jonas Camillus Jeppesen

## Resume

Solsystemet kan modelleres som et gravitationelt $n$-legeme system, hvori alle legemer interagerer gensidigt. Med udgangspunkt i Newtons gravitationslov samt beskrivelsen af solsystemet som et hamiltonsk system, bliver fysikken bag kredsløbsbaner og simulationsmetoder undersøgt. De gældende hamiltonske bevægelsesligninger opstilles for gravitation, hvorefter egnetheden af forskellige numeriske løsningsalgoritmer undersøges. Simple metoder som Eulers metode er ikke i stand til at bevare energien i et fysisk system, mens en 3. ordens symplektisk metode udviser en høj grad af numerisk stabilitet til gravitationssimulationer. Ved hjælp af programmering i Python og planetdata fra NASA skabes en numerisk solsystemsmodel ud fra et sæt initialbetingelser. Modellen udviser god præcision med en ubetydelig variation i simuleret energiniveau. Med et tidsskridt på 1 dag er den numeriske model i stand til at simulere Mars' position over 30 år med en maksimal afvigelse på 3400 km i forhold til data fra NASA, hvilket i runde tal svarer til planetens radius. Hvis tidsskridtet sænkes, er modellen endvidere i stand til at simulere Månens varierende kredsløb til en nøjagtighed på 2000 km eller under. Lignende metoder kan anvendes til at undersøge store astronomiske strukturer, modellering i ingeniørvidenskab, eller til planlægning af rummisioner.

# Indhold

Kapitel 1

# Indledning

Stjernehimlen har længe været en kilde af forundring blandt menneskene. Mellem de statiske stjerner ndes enkelte planeter, hvis tilsyneladende vandring har skabt astronomisk interesse siden oldtiden. De videnskabelige beskrivelser af planeternes bevægelse har derfor hele tiden udviklet sig i takt med ny viden. De første præcise beskrivelser blev formuleret af Johannes Kepler med sine velkendte ellipsebaner. På baggrund af dette var Newton i stand til at udlede hans generelle teori om tyngdekraften. Det viser sig umiddelbart, at det ikke er muligt at skabe en udelukkende analytisk beskrivelse af bevægelsesbanerne forårsaget af tyngdekraften mellem tre eller ere legemer. Man er derfor nødsaget til at gøre brug af tilnærmede numeriske metoder til modellering af disse fænomener. Målet med projektet er at undersøge hvordan sådanne numeriske algoritmer kan implementeres og anvendes til simulering af et astronomisk system bestående af ere legemer med fælles tiltrækning, det såkaldte n-legeme problem. Til dette formål bliver der først redegjort for den bagvedliggende teori om tyngdekraften mellem ere legemer og banebevægelser i et gravitationalt system, herunder gives en kort redegørelse for Hamiltons formulering af klassisk mekanik og numeriske metoder til løsning af di erentialligninger. Der bliver derefter udarbejdet en model af legemernes bevægelses ved at opstille Hamiltons bevægelsesligninger for n-legeme systemet, hvilket sammenkobles med en diskurs vedrørende valget af en passende numerisk algoritme til løsning af sådanne systemer. Derudover omtales valget af enheder og koordinatsystem, samt datakilder til brug ved planetariske simulationer. På baggrund af den redegjorde og udledte teori skabes et computerprogram, som er i stand til at simulere en simpel version af solsystemet. Denne løsning vurderes ved at undersøge energibevarelse og ved sammenligning med eksisterende astronomisk data. Til sidst diskuteres anvendelsesmulighederne for sådanne metoder inden for naturvidenskab og ingeniørvidenskab.

Kapitel 2

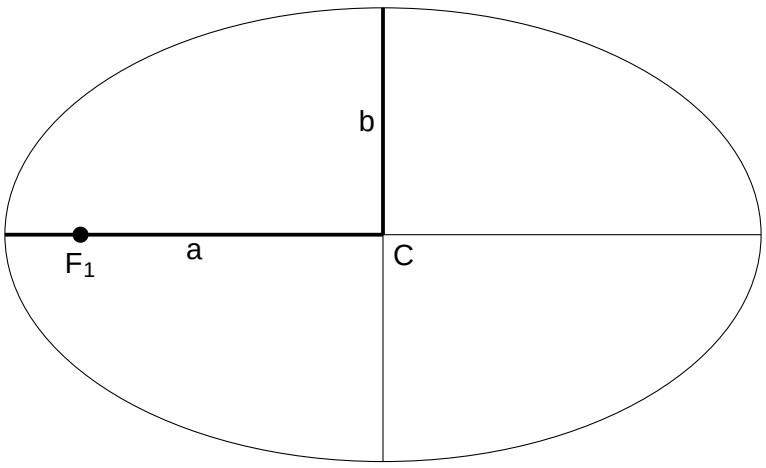
# Baggrund og teori

## 2.1 Keplers love

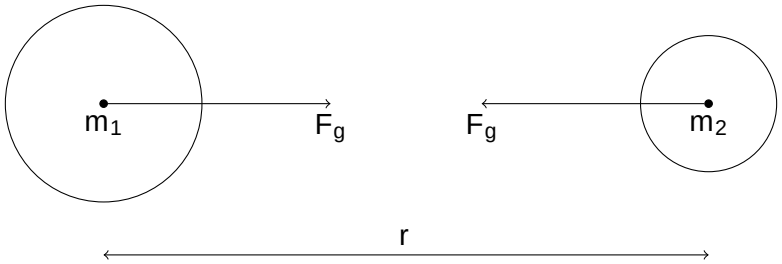
For at forstå den historiske baggrund bag problemet med tyngdekraften og planetbanerne er det nødvendigt at give en kort gennemgang af de vigtigste gennembrud i historien. Det var Johannes Kepler, der i starten af 1600-tallet fandt frem til en ny beskrivelse af planetbanerne gennem analysen af Tycho Brahes nøjagtige observationsdata. I første omgang var Keplers mål at eftervise Kopernikus' teori, men minimale afvigelser mellem teori og målinger af Mars' position drev Kepler til at fortsætte sin forskning. Det førte til hans tre love om planetbaner (Bate, Mueller og White, 2015, ss. 1 2):

- Alle planeter kredser i ellipsebaner med Solen i et brændpunkt.
- Linjen mellem planeten og Solen overstryger lige store arealer efter lige store tidsintervaller.
- Kredsløbsperioden opløftet i anden er direkte proportional med den halve storakse opløftet i tredje. $T^2 / a^3$

Et eksempel på en ideel ellipsebane beskrevet af Kepler er afbildet på gur 2.1. Selvom Keplers kredsløb kun gælder i tilfælde af to legemer, er det ofte en tilstrækkelig approximation af virkelige kredsløb over en kort tidsperiode.




Figur 2.1: Vigtige parametre i en ellipsebane.




Figur 2.2: Tyngdekraften mellem to legemer.

## 2.2 Newtons tyngdelov

Kepler skabte en præcis beskrivelse af hvordan planeterne bevæger sig, men det var først nær slutningen af 1600-tallet, at Isaac Newton med sit magnum opus, hans *Principia* , var i stand til at udlede den matematiske baggrund bag tyngdekraften. Newtons gravitationslov siger, at tyngdekraften mellem to objekter er proportional med produktet af objekternes masse, og omvendt proportional med den kvadrerede afstand. Dette kan udtrykkes med ligningen,

$$F_g = G \frac{m_1 m_2}{r^2} \tag{2.1}$$

hvor $F_g$ er tyngdekraften, G er gravitationskonstanten, r er afstanden mellem de to legemer og $m_1$;$m_2$ er masserne af de to legemer (Holck, Kraaer og Lund, 2009). Tyngdekraften virker på hver af de to legemer og peger imod det andet legeme som vist på gur 2.2. Ligning (2.1) beskriver længden af tyngdekraftvektoren. Alternativt kan tyngdekraften på $m_1$ beskrives ved hjælp af tyngdeloven på vektorform,

$$\vec{F_g} = G\frac{m_1 m_2}{|\vec{r}_{2;1}|^2} \frac{\vec{r}_{2;1}}{|\vec{r}_{2;1}|} = \frac{m_1 m_2}{|\vec{r}_{2;1}|^3} \vec{r}_{2;1} \tag{2.2}$$

hvor $\frac{\vec{r}_{2;1}}{|\vec{r}_{2;1}|}$ svarer til enhedsvektoren, der peger fra $m_2$ mod $m_1$ (Bate, Mueller og White, 2015).

Med udledelsen af sin tyngdelov var Newton i stand til at bevise Keplers love for to legemer i kredsløb om hinanden. I *Principia* behandlede Newton et gravitationelt system bestående af tre legemer matematisk, men var ikke i stand til at udlede en analytisk beskrivelse. Derudover beskriver han, at Solen, alle planeter og alle måner i solsystemet tiltrækker hinanden gensidigt (Newton, 1864). Problemet er i dag kendt som n-legemeproblemet.

## 2.3 Hamiltonsk mekanik

En af de mest grundlæggende fysiske love er energibevarelse. I et gravitationelt n-legeme system er dette gældende for den totale mekaniske energi i systemet. En reformulering af newtonsk mekanik er den såkaldte hamiltonske mekanik, som bygger på Hamiltons bevægelsesligninger. Den hamiltonske mekanik er særlig egnet til beskrivelsen af systemer, hvor den mekaniske energi er bevaret. Det vil vise sig senere i rapporten, at egenskaberne beskrevet af den hamiltonske mekanik er af speciel interesse til netop numerisk modellering af fysik.

Et hamiltonsk system kan beskrives af hamiltonfunktion[1], som er systemets samlede energi

[1] Her anvendes den tidsuafhængige hamiltonfunktion, da der ikke tilføres ekstern energi til systemet. Hamiltonfunktionen i dette tilfælde en konstant og er lig systemets energi.

udtrykt som en funktion af position og impuls (Kleitman, 2005):

$$H(q;p) \tag{2.3}$$

Hvor $q$ er systemets generaliserede positioner og $p$ er de generaliserede impulser. Det væsentlige er, at denne funktion giver systemets energi ved en given tilstand beskrevet af legemernes position og impuls. I den tidsuafhængige hamiltonfunktion er $H(q;p)$ konstant, selvom $q$ og $p$ kan ændre sig. Når sådan et system udvikler sig i tid, vil $q$ og $p$ følge en unik kurve i systemets *faserum*, der kan forstås som et slags koordinatsystem, som indeholder alle systemets mulige tilstande. Hver komposant af position og impuls for hver partikel har sin egen *akse* i faserummet (H. Smith, 2009). $H$ forbliver konstant henover løsningskurven beskrevet af Hamiltons bevægelsesligninger (Cohn, udat.), der svarer til Newtons anden lov,

$$\frac{d\vec{p}}{dt} = -\frac{\partial H}{\partial \vec{q}} \; ; \; \frac{d\vec{q}}{dt} = \frac{\partial H}{\partial \vec{p}}. \tag{2.4}$$

Bemærk, at første udtryk svarer til den Newtonske kraft, $\vec{F}$, mens andet udtryk svarer til hastighed $\vec{v}$ (Kleitman, 2005). Hamiltons ligninger er et sæt af første-ordens partielle differentialligninger. Hvis $H$ og startbetingelserne er kendt, eksisterer en unik løsning (Piziak og Mitchell, 2001).

## 2.4 n-legeme problemet

Kort sagt handler n-legeme problemet om at bestemme positionerne af legemerne i et gravitationelt system over tid, baseret på deres startbetingelser. I et system bestående af $n$ legemer med masserne $m_1; m_2; m_3 ::: m_n$ og stedvektorerne, $\vec{r}_1; \vec{r}_2; \vec{r}_3 ::: \vec{r}_n$, kan tyngekraften på $m_i$ betegnes som summen af alle tyngdepåvirkningerne beskrevet af formel (2.2),

$$\vec{F}_g = m_i \vec{a}_i = m_i \ddot{\vec{r}}_i = \sum_{\substack{j=0 \\ j \neq i}}^{n} G \frac{m_i m_j}{|\vec{r}_i - \vec{r}_j|^3} (\vec{r}_i - \vec{r}_j) \tag{2.5}$$

Her betegner $\vec{r}_i - \vec{r}_j$ vektoren fra legeme $j$ til $i$. Denne ligning gælder, hvis alle masser antages som uforanderlige homogene sfæriske legemer, der agerer som punktmasser (Bate, Mueller og White, 2015, s. 7). Ligningen kan omskrives til følgende form.

$$m_i \ddot{\vec{r}}_i = \sum_{\substack{j=0 \\ j \neq i}}^{n} G \frac{m_i m_j (\vec{r}_i - \vec{r}_j)}{|\vec{r}_i - \vec{r}_j|^3} \tag{2.6}$$

Da hver vektor har tre komposanter, beskriver systemet af $3n$ anden ordens differentialligninger det generelle $n$-legeme problem. Den komplette løsning er værdien af $\vec{r}_i$ over tidsintervallet $]-1;1[$ (Aarseth, 2003).

En analytisk løsning ved $n = 2$ blev fundet af Newton, og de resulterende bevægelsesbaner danner kegleudsnit som beskrevet af Keplers love (Bate, Mueller og White, 2015, ss. 11–14). To-legeme problemet i gravitation er derfor også kendt under navnet *Keplers problem*.

Tre-legeme problemet er endnu et specialtilfælde af $n$-legeme problemet. Det var Henri Poincaré, der i 1890 beviste med sit *Mécanique Céleste*, at det generelle problem ikke kan løses analytisk

(Diacu, 1995; Brown m. ., 1892). Dog ndes en række eksempler, såsom de stabile Lagrange-punkter fundet af Lagrange of Euler, hvor eksakte beskrivelser eksisterer (Cartwright, 2017).

Et af de første eksempler på modellering af et større n-legeme system var Holmberg, der i 1941 brugte lamper og fotoceller til at modellere interaktionen mellem galakser numerisk (Holmberg, 1941). Siden er digitale computere brugt til sådanne simulationer.

## 2.5 Numeriske algoritmer

Numerisk analyse i matematik er undersøgelsen af numeriske og tilnærmede løsningsmetoder til matematiske problemer. Sådan en numerisk metode består af en række løsningstrin, der tilsammen udgør en numerisk algoritme. I praksis er numeriske algoritmer er ofte implementeret gennem programmering, og anvendes specielt til løsningen af di erentialligninger, der ikke har en eksakt analytisk løsning (A. Smith og Larsen, 2010). Udover det generelle n-legeme problem ndes mange andre fysiske fænomener, der ikke kan løses analytisk. Eksempler på disse er bevægelsen af væsker og det dobbelte pendul. Ved numerisk løsning af et tidsafhængigt fysisk problem bruges gentagne beregninger af systemets tilstand efter små tidsintervaller, t.

### 2.5.1 Eulers metode

Den mest basale numeriske algoritme til løsning af di erentialligninger er Eulers metode, hvor tangentens hældning til et givent punkt bruges til bestemmelsen af det næste punkt. Ved gentagne beregninger estimeres løsningen. Hvis $f(y) = y^0$ giver tangentens hældning, kan Eulers metode opskrives på følgende måde.

$$y_{n+1} = y_n + f(y_n) \quad h \tag{2.7}$$

Hvor h er skridtlængden. Skridtlængden i en numerisk algoritme påvirker nøjagtigheden af resultatet. Problemet med Eulers metode er, at numeriske fejl stiger hurtigt, specielt hvis løsningskurven krummer meget (Heefelt, 1990, ss. 29 30). Ved brug til en fysisk simulering betyder det, at virkeligheden og simulationen hurtigt divergerer, specielt ved større h. Eulers metode er en første-ordens metode, mens en variation, den forbedrede Eulers metode, er af anden orden. En højere orden betyder, at den numeriske fejl hurtigere går mod nul, når skridtlængden går mod nul, men metoder af højere orden kræver ofte ere beregninger til hvert skridt. Ved valg at en numerisk metode skal der derfor vægtes mellem regnehastighed og resultatets præcision. Både Eulers metode og den forbedrede Eulers metode hører under gruppen af de såkaldte Runge-Kutta metoder, der er kendetegnet ved at tage et vægtet gennemsnit af forskellige estimerede hældningsværdier. I princippet ndes uendelig mange Runge-Kutta metoder, da hver metode beskrives af en række konstanter, der kan vælges frit. Dog er det ikke alle værdier, der fører til gode resultater. En meget anvendt Runge-Kutta metode er en speci k 4. ordens metode, ofte kaldet RK4, der er i stand til at løse mange komplekse di erentialligninger (Heefelt, 1990, s. 34). I næste kapitel introduceres lignende metoder, der er bedre egnet til løsning af Hamiltons bevægelsesligninger.

Kapitel 3

# Metoder til løsning af planeternes kredsløb

Denne del af dokumentet omhandler hvordan teorien om gravitation kombineres med de numeriske algoritmer til en model af legemers bevægelse i et system styret af tyngdekraften.

## 3.1 Bevægelsesligninger for gravitation

For at opstille hamiltonfunktionen for n-legeme problemet skal systemets samlede kinetiske energi og potentielle energi beskrives af partiklernes position og impuls.

I klassisk mekanik er den kinetiske energi af et legeme givet ved (e.g. Holck, Kraaer og Lund, 2009, s. 310),

$$E_{kin} = \frac{1}{2}mv^2: \tag{3.1}$$

Hvis impulsen er $p = mv$, kan den kinetiske energi formuleres som,

$$E_{kin} = \frac{p^2}{2m}: \tag{3.2}$$

Det er let at vise med substitution, at disse to udtryk er ækvivalente. Ved at summere den kinetiske energi over allen legemer i et system findes et udtryk for den totale kinetiske energi,$T$ (Aarseth, 2003, s. 7).

$$T = \sum_{i=1}^{n} \frac{|\vec{p}_i|^2}{2m_i} = \frac{1}{2}\sum_{i=1}^{n} m_i \vec{v}_i^2 \tag{3.3}$$

Den potentielle energi for to tiltrækkende legemer i et tyngdefelt er (Holck, Kraaer og Lund, 2009, s. 36),

$$E_{pot} = -G\frac{m_1 m_2}{r}: \tag{3.4}$$

Den totale potentielle energi, $U$, findes ved at summere energien mellem alle legemer.

$$U = \sum_{i=1}^{n}\sum_{j>i}^{n} \frac{Gm_i m_j}{|\vec{r}_i - \vec{r}_j|} \tag{3.5}$$

Udtryk (3.3) og (3.5) adderes, hvilket giver et udtryk for hamiltonfunktionen (positionen $\vec{r}$ anvendes i stedet for $\vec{q}$ i den oprindelige formulering).

$$H = T + U = \sum_{i=1}^{n} \frac{|\vec{p}_i|^2}{2m_i} - \sum_{i=1}^{n}\sum_{j>i}^{n} \frac{Gm_i m_j}{|\vec{r}_i - \vec{r}_k|} \tag{3.6}$$

Da tiden, $t$, ikke indgår i udtrykket, er den tidsafledte hamiltonfunktion lig nul. $H$ skal altså være konstant i $n$-legeme systemet. Energibevarelse i simulationer kan derfor dømmes ved at undersøge numeriske afvigelser i $H$. Hamiltons bevægelsesligninger som beskrevet i udtryk (2.4) for legemet $i$ opstilles nu sammen med de newtonske modstykker.

$$\vec{F} = \frac{d\vec{p}_i}{dt} = -\frac{\partial H}{\partial \vec{r}_i}\,; \quad \vec{v} = \frac{d\vec{r}_i}{dt} = \frac{\partial H}{\partial \vec{p}_i} \tag{3.7}$$

Det viser sig, ved partial differentiation, at første udtryk svarer til den velkendte gravitationslov for $n$-legeme problemet som beskrevet i sektion 2.4, mens andet udtryk forkortes betydeligt,

$$\dot{\vec{p}}_i = -Gm_i \sum_{\substack{j=0 \\ j \neq i}}^{n} \frac{m_j(\vec{r}_i - \vec{r}_j)}{(|\vec{r}_i - \vec{r}_k|)^3}\,; \quad \dot{\vec{r}}_i = \frac{\vec{p}}{m_i} \tag{3.8}$$

Sammenhængen kan udledes ved at indse, at alle led i summen, der ikke indeholder $t$ eller $r_i$, er konstante og forsvinder ved differentiation. Udledningen af disse udtryk kan findes i bilag A.

Impulsen, $\vec{p}$ kan erstattes af $m\vec{v}$, da massen antages er være konstant. Ved omskrivning giver det følgende ligninger.

$$m_i\dot{\vec{v}} = -Gm_i \sum_{\substack{j=0 \\ j \neq i}}^{n} \frac{m_j(\vec{r}_i - \vec{r}_j)}{(|\vec{r}_i - \vec{r}_k|)^3}\,; \dot{\vec{r}}_i = \frac{m_i\vec{v}}{m_i} \tag{3.9}$$

Bevægelsesligningerne bliver altså,

$$\boxed{\dot{\vec{v}}_i = -G \sum_{\substack{j=0 \\ j \neq i}}^{n} \frac{m_j(\vec{r}_i - \vec{r}_j)}{(|\vec{r}_i - \vec{r}_j|)^3}\,; \quad \dot{\vec{r}}_i = \vec{v}_i} \tag{3.10}$$

Når bevægelsesligningerne løses numerisk, sker det oftest på denne form. Sidste udtryk er naturligvis lig accelerationen, $\vec{a}$ (Aarseth, 2003, s. 6). Fordi ligningerne er givet i vektorform, er der for hele $n$-legeme systemet $6n$ differentialligninger af første grad, en ligning til hver komposant (Musielak og Quarles, 2014).

I bilag B er det vist, at samme udtryk kan udledes fra Newtons anden lov, og at ligningerne derfor er ækvivalente. Her blev hamiltonsk mekanik anvendt for at vise, at energibevarelse er tæt knyttet til $n$-legeme problemet.

## 3.2 Energibevarelse og symplektiske integratorer

Som beskrevet er hamiltonfunktionens værdi konstant, hvis funktionsudtrykket ikke er tidsafhængigt. I tilfælde af et regelmæssigt periodisk dynamisk system (eksempelvis to-legeme problemet eller et simpelt svingende pendul) danner løsningskurven i systemets faserum en lukket løkke, da systemets tilstand efter en periode, $T$, er lig tilstanden ved $t_0$. En anden egenskab ved det hamiltonske faserum er, at volumen udgjort af en given sæt af startbetingelser forbliver konstant henover den tidsmæssige løsningskurve (Nordkvist og Hjorth, 2005). Dette udsagn svarer til Liouville's sætning (Lützen, 2009). Problemet med de klassiske løsningsmetoder, herunder Eulers

metode og de klassiske Runge-Kutta metoder er, at de ikke er i stand til holde værdien af hamiltonfunktionen eller volumen i faserummet konstant. Den totale energi af sådan en løsning har derfor en tendens til at stige eller falde over tid (Hairer, Lubich og Wanner, 2003, s. 399). I eksemplet med et pendul vil det over tid blive exciteret eller dæmpet, uden nogen modelleret ekstern kraft (Ruth, 1983). Den estimerede løsningskurve i faserummet vil ikke danne en lukket løkke, men i stedet en spiral[1].

En familie af numeriske metoder til løsning af differentialligninger er kendt som de symplektiske metoder. Disse metoder er egnet til numerisk løsning af hamiltonske systemer, da deres løsningskurve danner lukkede løkker i tilfælde af et periodisk system. Symplektiske løsningsmetoder skaber altså løsninger, hvor energiniveauet ikke ændrer sig markant. I tilfælde af periodiske systemer vil energiniveauets variation være periodisk, hvor energien i klassiske metoder er stigende eller faldende. Disse egenskaber betyder, at nogle af disse metoder er symmetriske tidsmæssigt, og at tidsreversibilitet i dynamiske simulationer muliggøres (Hairer, Lubich og Wanner, 2003, s. 408).

### 3.2.1 Störmer Verlet metoden

Störmer-Verlet metoden er et af de simpleste eksemper på en symplektisk metode, der er egnet til løsning af bevægelsen i et dynamisk hamiltonsystem. Det er faktisk tilfældet, at denne metode er blevet opfundet og genopdaget flere gange historisk. Varianter er beskrevet af C. Störmer i 1907 til partikelfysik, J. F. Encke omkring 1860 til planetbaner, J. B. Delambre i 1792 til astronomi og endda Newton i sin Principia til at bevise Keplers anden lov (Hairer, Lubich og Wanner, 2003, s. 402). I dag bruges metoden ofte til molekylærdynamik under navnet Verlet metoden, hvoraf to varianter eksisterer, den klassiske og hastighedsvarianten velocity Verlet (Nikolic, 2002). Den sidstnævne bruger eksplicit legemernes hastighed under udregningerne. Hvis $\dot{r} = v$ og $\dot{v} = a(r)$ kan metoden opskrives på følgende form (adapteret fra Hairer, Lubich og Wanner, 2003, s. 407):

$$v_{n+\frac{1}{2}} = v_n + \frac{1}{2}ha(r_n) \tag{3.11}$$

$$r_{n+1} = r_n + hv_{n+\frac{1}{2}} \tag{3.12}$$

$$v_{n+1} = v_{n+\frac{1}{2}} + \frac{1}{2}ha(r_{n+1}) \tag{3.13}$$

Ligningerne viser, at denne metode bruger en estimeret hastighed et halvt skridt frem i tiden til at beregne positionen og den nye hastighed. Det står i kontrast til Eulers metode og de klassiske Runge-Kutta metoder, der kun bruger hele skridt i udregningerne. Verlet metoden er en anden ordens metode, og den numeriske fejl vil derfor være mindre end Euler og andre metoder af første orden. Fordi metoden er symplektisk, er den i stand til at bevare energiniveauet i et dynamisk system.

### 3.2.2 Ruths metode af 3. orden

Læg mærke til, at Verlet-metoden anvender koefficienterne $\frac{1}{2}$ og (implicit) 1 sammen med skridtlængden $h$. Ved at generalisere disse koefficienter kan en gruppe af symplektiske metoder, hvoraf Verlet indgår, beskrives med følgende algoritme (adapteret fra Ruth, 1983 og Koto og Song, 2014).

[1] Da faserummets struktur kan beskrives på mange måder, herunder som en cylinder eller $n$-dimensionel mangfoldighed, bruges ordet spiral løst.

1. For hver af koe cienterne $c_i = c_1; c_2 : : :$ med de tilsvarende $d_i = d_1; d_2 : : :$

   (a) Bestem accelerationen, $a(r_n)$, i den nuværende kon guration.

   (b) Opdater hastighed med den nye acceleration, $v_{n+1} = v_n + c_i h a(r_n)$.

   (c) Opdater position med den nye hastighed, $r_{n+1} = r_n + d_i h v_{n+1}$.

Denne gruppe af metoder minder til dels om de klassiske Runge-Kutta metoder, da mellemregninger med koe cienter anvendes. Der anvendes dog seperate koe enter for hastighed og position. Gruppen af metoder går derfor under navnet symplektiske opdelte Runge-Kutta metoder (Symplectic Partitioned Runge Kutta eller SPRK) (Koto og Song, 2014). Der ndes mange mulige kombinationer af c og d, som kan bruges til løsning af Hamiltons ligninger. Ruth, 1983 giver følgende koe cienter til en symplektisk metode af 3. orden.

$$c_1 = \frac{7}{24} \quad c_2 = \frac{3}{4} \quad c_3 = \frac{1}{24} \tag{3.14}$$

$$d_1 = \frac{2}{3} \quad d_2 = \frac{2}{3} \quad d_3 = 1 \tag{3.15}$$

I resten af denne rapport kaldes denne 3. ordens SPRK med Ruths koe cienter bare Ruths metode eller variationer heraf. Udover de nævnte symplektiske metoder til løsning af Hamiltons ligninger, er der i dag opfundet mange andre algoritmer af højere orden, hvoraf nogle bruger adaptive skridtlængde eller er optimeret til parallelle supercomputere (Montenbruck, 1992). Af hensyn til opgavens længde tages der udgangspunkt i Euler, Verlet og Ruth.

## 3.3 Koordinatsystem og enheder

Til beskrivelsen af objekternes position og hastighed i det planetariske system, er det nødvendigt at vælge et passende koordinatsystem. Fordi udregningen sker gennem vektorer, vælges et tre-dimensionalt kartesisk koordinatsystem, hvor mellemregninger ikke er nødvendige under vektoroperationer. Origos placering kan vælges arbitrært, men denne skal ikke være accelerende. Massemidtpunktet af et gravitionel system uden ekstern kraftpåvirkning opfylder denne betingelse (Aarseth, 2003, s. 7) grundet Newtons første lov. Hvis hele det planetariske system bevæger sig (set udefra) vil massemidtpunktet være stationær set fra den valgte referenceramme. Selvom Solen overordnet lader til at være i solsystemets centrum, kredser den også rundt omkring systemets massemidtpunkt[2]. Koordinatsystemets $xy$-plan sættes til planet, som Jordens kredsløb be nder sig i ved referencetidspunktet j2000. Koordinatsystemets $x$-akse peger fra origo mod Jordens position ved referencetidspunktet og $y$-aksen be nder sig 90° i kredsløbets retning. Ved højrehåndsreglen er $z$-aksen vinkelret med $xy$-planet og peger i nordpolens retning (JPL, 2019a).

I dagligdagsfysik er det almindeligt at bruge SI-enhederne. Når man bevæger ud til den planetariske skala er meter og sekund ikke længere betydelig. I planetær astronomi anvendes den astronomiske enhed, AU, som længdeenheden, hvorfor denne vælges til dette projekt. En AU er de neret til præcis 149 597 870 700 m af den Internationale Astronomiske Union. Derudover sættes tidsenheden til dage, dag = 86 400 s (IAU, 2012).

Fordi massen og gravitationskonstanten in-legeme problemet altid opstår sammen i formen $GM$, som har enheden $(\frac{\text{distance}^3}{\text{tid}^2})$, er det ikke direkte nødvendigt at vælge en enhed for massen.

[2] I december 2019 er massemidtpunktet af hele solsystemet omtrent 1,8 solradier fra Solens centrum.

| Legeme | GM, $\frac{au^3}{dage^2}$ |
|---|---|
| Sun | 0,295912208285591100E-03 |
| Mercury | 0,491248045036476000E-10 |
| Venus | 0,724345233264412000E-09 |
| Earth | 0,888769244512563400E-09 |
| Moon | 0,109318945074237400E-10 |
| Mars | 0,954954869555077000E-10 |
| Jupiter | 0,282534584083387000E-06 |
| Saturn | 0,845970607324503000E-07 |
| Uranus | 0,129202482578296000E-07 |
| Neptune | 0,152435734788511000E-07 |
| Pluto | 0,217844105197418000E-11 |

Tabel 3.1: JPL's bestemte $GM$ -værdier for udvalgte legemer.

Det viser sig, at $GM$ for solsystemets planeter er kendt til en højere præcision end masserne alene, da $GM$ kan bestemmes fra observationer eller data fra forbi yvende rumsonder, mens præcisionen af $m$ hviler på den eksperimentelt bestemte $G$ (Lewis, 1997, s. 54). Der vælges derfor arbitrært SI-enheden for masse, kg.

## 3.4 Astronomisk data

Til at de nere systemets fysiske parametre, herunder planeternes $GM$ og deres startbetingelser, anvendes højpræcisionsdata o entliggjort af NASA's Jet Propulsion Laboratory, oprindeligt skabt til at understøtte interplanetariske rummisioner (W. Folkner, 2014). Det er ikke muligt direkte at måle en planets parametre som masse og $GM$ . JPL's planetdata er derfor udledt ved at tilpasse en detaljeret numerisk solsystemsmodel til de bedste tilgængelige observationsdata indsamlet fra starten af 1900-tallet frem til i dag. Disse data stammer bl.a. fra afstandsmåling til månen med lasere, telemetri fra satellitter i kredsløb, forbi yvende rumsondere og astrometri (W. M. Folkner m. ., 2014, ss. 24, 29, 39). Månens kredsløbsdata er beskrevet i modellen med en usikkerhed på under en meter, mens klippeplaneternes position er beskrevet med en usikkerhed på under en kilometer. Da information om gasplaneternes udelukkende stammer fra astrometriske observationer og enkelte forbi yvninger (Voyager 1 og 2), er usikkerheden af disse i størrelsesordenen titusinde kilometer (W. M. Folkner m. ., 2014, s. 1). JPL's bestemte $GM$ -værdier med enheden $\frac{au^3}{dage^2}$ er gengivet i gur 3.1.

Datapunkter genereret fra deres system, kaldet Horizons, udgør et af de mest præcise o entlig tilgængelige efemerider for solsystemets planeter, og afvigelsen fra virkeligheden i tidsintervallet 1910-2013, hvor der har været observationer, er derfor relativ lille (JPL's model går ca. 13000 år tilbage i tiden og 17000 år frem i tiden). For at veri cere modellen, der implementeres i denne rapport vil der derfor blive sammenlignet med JPL's planetdata. Datapunkternes format kan kon gureres i Horizons, hvoraf koordinatsystemet og enhederne beskrevet i sidste afsnit er nogle af valgmulighederne (JPL, 2019a).

Kapitel 4

# Implementering af numerisk solsystemsmodel

I dette kapitel bliver den fysiske, matematiske og datalogiske teori fra de tidligere dele af rapporten implementeret i praksis gennem brug af programmering.

## 4.1 Strukturering og planlægning

Før et programmeringsprojekt er første trin at planlægge strukturen af programmet for at skabe oversigt og forhindre uorden. Det kan både være valgt af programmeringssprog, strukturering af ler eller design af algoritme. Endvidere kan der undersøges, om der eksisterer relevante biblioteker eller pakker, der gør arbejdet lettere. For at holde fokus på det algoritmiske, bruges Python som programmeringssproget. Det er et sprog med meget abstraktion fra selve maskinen, og koden bliver derfor mere kortfattet. Endvidere bruges en objekt-orienteret struktur til opbygningen af programmet, hvilket er kendetegnet ved eksistensen af generelle klasser, der kan indeholde parametre eller funktioner (Hansen, 2013). Eksempelvis skabes en klasse for konceptet legeme, der de nerer parametre som position og hastighed. Ved skabelsen af en instans, en slags kopi, af klassen, tildeles egenskaberne en reel værdi. Objekt-oriented programmering er altså brugbart, når mange variationer af samme objekt er til stede.

I den teoretiske del af rapporten blev vektorregning brugt til beskrivelse af position og hastighed. Helt generelt består en vektor i rummet af tre skalarer[1]. Et kendt Python-bibliotek er numpy som er meget populært til videnskabelig beregning. numpy indeholder datastrukturer til beskrivelsen af n-dimensionelle vektorer og matricer, samt nyttige værktøjer til bearbejdning af disse. numpy anvendes derfor i projektet.

For at visualisere resultat anvendes Python-biblioteket matplotlib , som både kan skabe to- dimensionelle og tre-dimensionelle diagrammer.

## 4.2 Legeme-klassen

Den mest grundlæggende klasse i programmet skal indkode konceptet legeme. De vigtigste egenskaber, som legemet skal indeholde, er information vedrørende dets fysiske parametre og tilstand. I første omgang de neres GM , position, hastighed og acceleration i en klasse døbt body:

```
1 class body :
2     def __init__(self, GM, position, velocity, acceleration, name="body"):
3         self.GM = GM
```

[1] I teorien kan vektorer skabes ved at de nere tre seperate variabler, en for hver komposant. Dette frarådes dog kraftigt fra erfaring.

```
4           self.pos = np.array(position, dtype=np.float64)
5           self.vel = np.array(velocity, dtype=np.float64)
6           self.acc = np.array(acceleration, dtype=np.float64)
7           self.name = name
```

For at mindske den numeriske fejl forårsaget af afrundinger i computeren bruges 64-bit ydende komma tal (float64 ), der har en præcision svarende til ca. 16 betydende cifre i decimalsystemet, og som kan repræsentere tal i størrelsesordenen $10^{308}$ (Kahan, 1996). Funktionen np.array skaber enn-dimensional numpyvektor, hvor n er dimensionen af de givne initialværdier. Denne klasse kan derfor både bruges til at repræsentere legemer i 1, 2 eller 3-dimensionalt rum (eller højere, hvis ønsket).

## 4.3 Fysik og numerisk løsningsmetode

Der de neres nu en klasse, der repræsenterer det system, som indeholder legemerne. Det er altså her, at interaktionerne foregår. For at holde alle de fysiske ligninger og implementationen af de numeriske algoritmer samlet, bliver disse placeret i systemets klasse i stedet for legemerne. Denne klasse døbes nbody_system, og initialiseres med en liste over legemerne.

```
1 class nbody_system:
2     def __init__(self, body_list, G=1):
3         self.body_list = body_list
4         self.n = len(body_list)
5         self.G = G
```

Variablen G i programmet bruges udelukkende til omregninger, ikke til selve simulationen. Standardværdien af denne sættes til 1.

Under denne klasse skabes nu en metode, døbt update_acc, som er i stand til at opdatere alle legemernes accelerationsværdi ved summation som beskrevet af den udledte bevægelsesligning, (3.10).

```
1  def update_acc(self):
2      for i in range(self.n):
3          body_i = self.body_list[i]        # forces on body i
4          body_i.last_acc = np.copy(body_i.acc)       # save previous acceleration
5          body_i.acc *= 0  # reset current acceleration and sum:
6          for j in range(self.n):
7              if j != i:
8                  # every other body
9                  body_j = self.body_list[j]
10                 # calculate acceleration
11                 body_i.acc -= body_j.GM * (body_i.pos - body_j.pos) /
                       (np.linalg.norm(body_i.pos - body_j.pos)**3)
```

Den første for-løkke (linje 2) itererer over alle legemer i systemet. For hvert legeme bruges endnu en for-løkke (linje 6) til at iterere over alle andre legemer i systemet, hvilket svarer til

summationstegnet i udtrykket,

$$\vec{a_i} = \dot{\vec{v}_i} = \sum_{\substack{j=0 \\ j \neq i}}^{n} \frac{Gm_j(\vec{r_i} - \vec{r_j})}{(|\vec{r_i} - \vec{r_j}|)^3}. \tag{4.1}$$

Linje 11 i kodestumpen svarer direkte til udtrykket i summen, da accelerationsbidraget forårsaget af objekt $j$ beregnes som produktet af $j$'s GM-værdi og vektoren fra $j$ til $i$ divideret med afstanden opløftet i tredje potens. Bemærk minus-tegnet foran lighedstegnet i linje 11, hvilket svarer til minustegnet foran summen.

Den numeriske løsningsalgoritme implementeres nu som endnu en metode under `body_system`-klassen. For sammenligningens skyld implementeres både Eulers metode, Verlet metoden og Ruths 3. ordens symplektiske metode. Koden til den sidstnævnte er vist her, da den indkoder koncepter fra de første to. Koden til Eulers metode og Verlet metoden kan findes i bilag E.

```
1  def ruth3(self, dt=0.005):
2      # 3rd order symplectic integrator
3      c = [7/24, 3/4, -1/24]
4      d = [2/3, -2/3, 1]
5
6      for idx in range(3):
7          # update acceleration for new pos
8          self.update_acc()
9
10         for i in range(self.n):
11             body_i = self.body_list[i]
12             # update pos and vel using coeffecients
13             body_i.vel += body_i.acc * c[idx] * dt
14             body_i.pos += body_i.vel * d[idx] * dt
```

Linje 3-4 specificerer konstanterne beregnet af Ruth, 1983. Resten af kodestumpen følger algoritmen fra sektion 3.2.2. For-løkkens indhold udføres tre gange: en gang for hvert par af $c_i$ og $d_i$. I for-løkken bliver alle legemers acceleration opdateret baseret på de nyeste positioner, hvorefter nye værdier for hastighed og position beregnes. Denne metode skal altså beregne accelerationen tre gange for hvert tidsskridt.

Til sidst skabes en metode, som bestemmer den totale energi i systemet. Værdien fra denne funktion svarer til værdien af hamiltonfunktionen ved systemets givne tilstand.

```
1  def total_energy(self):
2      E_total = 0
3      for i in range(self.n):
4          body_i = self.body_list[i]
5          #kinetic energy
6          E_total += 1/2 * body_i.GM * np.linalg.norm(body_i.vel)**2
7
8          for j in range(i+1, self.n):
9              #potential energy
10             body_j = self.body_list[j]
```

```
11                E_total -= body_i.GM * body_j.GM/(np.linalg.norm(body_i.pos -
                      body_j.pos))
12
13        return  E_total/self.G
```

Hamiltonfunktionen for n-legeme problemet indeholder to summationstegn som udtrykt i (3.6). For at mindske antallet af for-løkker i programmet, er de to summer i udtrykket kombineret, da begge går fra 1 til n. Symbolsk svarer kodestumpen til en omskrevet[2] version af hamiltonfunktionen, hvilket er vist her som et udtryk for den totale energi, E.

$$E = \frac{\sum_{i=1}^{n} \frac{1}{2} G m_i |\vec{v}_i|^2 \quad \sum_{j>i}^{n} \frac{G m_i G m_j}{|\vec{r}_i - \vec{r}_k|}}{G} \tag{4.2}$$

Første for-løkke i linje 3 svarer til første sum i udtryk (4.2), mens anden for-løkke i linje 8 svarer til anden sum, hvori den potentielle energi beregnes. Bemærk, at værdien af E_total frem til linje 12 faktisk er E G, da Gm bruges til beregningerne af kinetisk og potentiel energi i stedet for m. Dette er også vist i udtryk (4.2). For at beregne selve energien, bliver værdien divideret med G i linje 13.

Udover metoderne til selve simulationen skabes også metoder til datalogning og datadatavisualisering. Udvalgte dele kan findes i bilag E.

## 4.4 Afprøvning

Under udviklingen af et program kan løbende afprøvninger anvendes til at verificere den implementerede kode. Der opstilles derfor et to-dimensionalt to-legeme problem med følgende arbitrære startværdier.

$$G = 1 \tag{4.3}$$

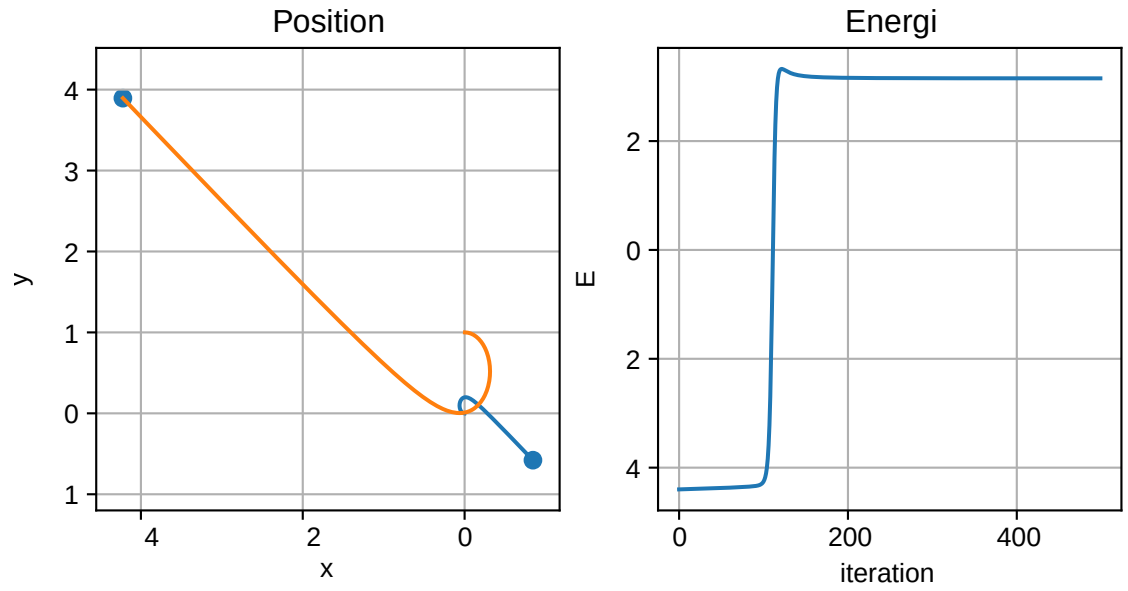
$$Gm_1 = 5 \quad \vec{r}_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \vec{v}_1 = \begin{pmatrix} \frac{1}{5} \\ 0 \end{pmatrix} \tag{4.4}$$

$$Gm_2 = 1 \quad \vec{r}_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \vec{v}_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{4.5}$$

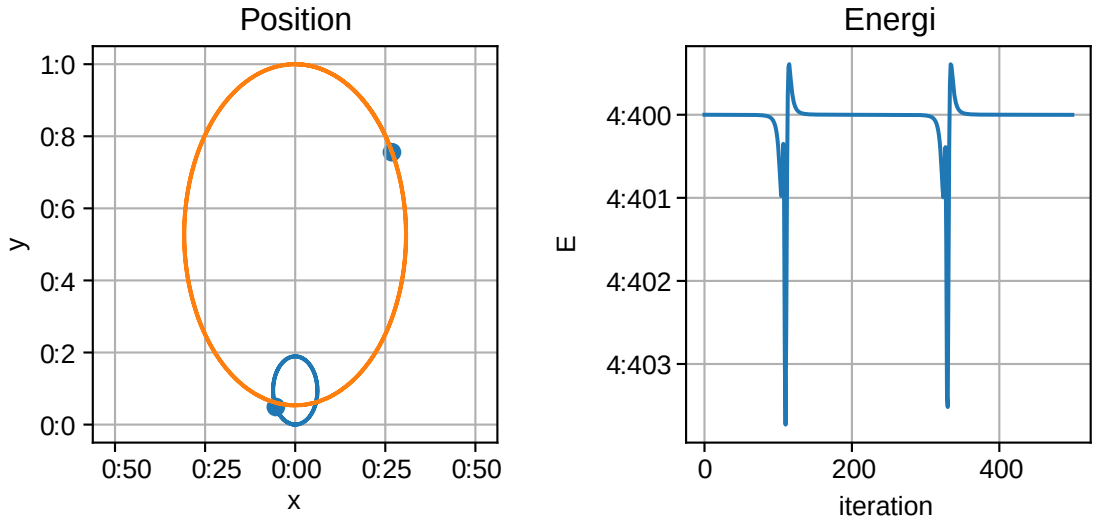
De valgte hastigheder og masser sikrer, systemets massemidtpunkt forbliver konstant, da systemets samlede impuls er nul. Tidsskridtet $t = 0{,}005$ anvendes og simulationen forløber i 500 iterationer. Ved hver iteration gemmes positionen af begge legemer, samt systemets samlede mekaniske energi med total_energy -metoden. Resultatet fra Eulers metode er vist i figur 4.1.

Legeme 1 er repræsenteret af den blå kurve, mens legeme 2 er orange. På energidiagrammet ses det, at den mekaniske energi i systemet ikke er konstant som forventet. Derudover viser positionsdiagrammet, at de to legemer bliver slynget bort fra hinanden kort tid efter de mødes nær origo. Dette skyldes, at Eulers metode ikke er i stand til at bevare energien i et hamiltonsk system. Simulationsparametrene og startbetingelser er til dels valgt for at illustrere netop dette. Metodens fejl kan mindskes ved at bruge en mindre skridtlængde, men som vist på figur C.1 i Bilag C, er fejlen stadig markant med skridtlængden $t = 0{,}0005$. Hvis Verlet-metoden

[2] Fordi hamiltonfunktionen er de neret som en funktion af impuls og position, kan denne omskrivning strengt taget ikke længere kaldes hamiltonfunktionen.




Figur 4.1: Resultatet af Eulers metode til to-legeme problemet $t = 0;005$. Bemærk skalering af akserne: De to legemer burde kredse omkring et punkt nær origo.




Figur 4.2: Resultatet af Ruths 3. ordens metode til to-legeme problemet $t = 0;005$.

i stedet anvendes, forbedres simulationens stabilitet drastisk. Resultatet kan ndes på gur C.2 i bilaget. Energiniveauet i systemet stiger ikke længere ukontrolleret, og den periodiske kredsløbsbevægelse er stabil. Kredsløbets position ændrer sig dog langsomt. Hvis Ruths 3. ordens metode anvendes, er denne variation næsten elimineret som vist på gur 4.2. På samme måde som Verlet metoden forbliver energiniveauet i systemet relativ konstant, mens den periodiske variation på E 0;005 enheder er mindre end Verlet ( E 0;25 enheder). Når eccentriciteten af kredsløbet er lav, er variationen i energiniveau ubetydelig, hvilket er illustreret på gur C.3 i bilaget, hvor energivariationen med Ruths metode er E 8 10 $^{8}$.

Til brug i den større solsystemsmodel anvendes Ruths metode derfor.

## 4.5 Astronomisk data og startbetingelser

Som omtalt tidligere, bruges data fra NASA JPL's Horizons system. For at afprøve den udviklede simulationsprogram skal startbetingelserne på en bestemt dato udvælges. Her vælges datoen d. 1. januar 1970, som ligger inden for tidsperioden hvori Horizons-modellens observationsdata blev indsamlet. Følgende datapunkt fra Horizons beskriver Jordens tilstand på den førnævnte dato (JPL, 2019b):

```
1 2440587.500000000 = A.D. 1970-Jan-01 00:00:00.0000 TDB
2  X =-1.762267229040138E-01 Y = 9.684335265498731E-01 Z = 3.860769680717466E-06
3  VX=-1.719568902488065E-02 VY=-3.210508485900838E-03 VZ= 2.480944449126105E-07
4  LT= 5.685056457215108E-03 RG= 9.843370120168281E-01 RR=-8.007839608577742E-05
```

Første linje beskriver tidspunktet. Det relevante for dette projekt er X, Y og Z, der angiver positionen samt VX, VY og VZ, der angiver hastighedens komposanter. Positionens enhed er i AU, mens hastigheden er givet i AU per dag. Værdierne på den sidste linje er henholdsvis afstand til referencepunktet i lysdage, afstand i AU og afstandsændringen, men disse tre bliver ikke nødvendige til dette projekt.

For at gøre det lettere anvende data fra Horizons, bruges en Python-bibliotek kaldet astroquery som giver adgang til astronomisk data.Astroquery indeholder en API til efterspørgsel af data fra JPL horizons, og den resulterende data kan direkte bruges i Python til beregninger og visualisering. Figur 4.3 viser et 3d-diagram af solsystemets planetbaner over tidsintervallet 1970-2000 baseret på data fra JPL Horizons. Positionerne og hastighederne for Solen, de resterende planeter, Pluto og Jordens måne d. 1. januar 1970 er givet i bilag D.

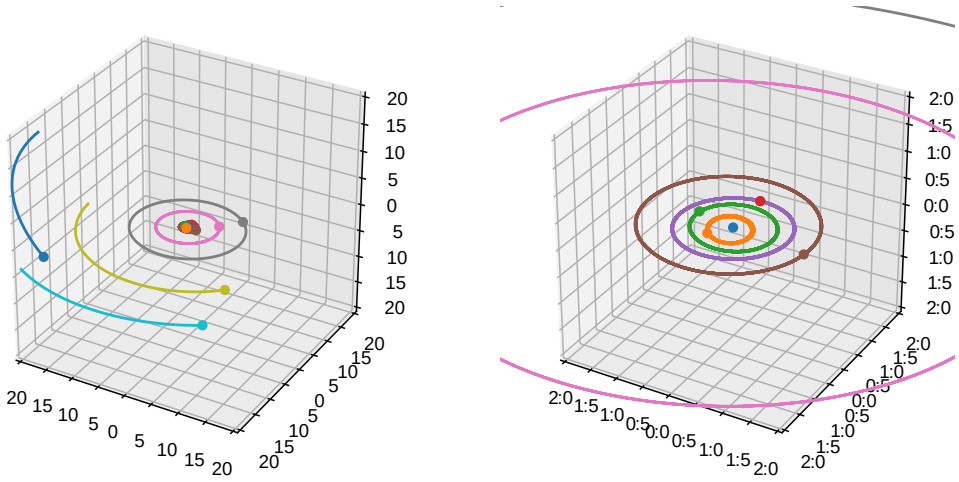
Fordi alle værdierne er angivet i samme enhedssystem (AU, dag), er det ikke nødvendigt at omregne disse ved overførsel til simulationsprogrammet. Den newtonske gravitationskonstant har værdien (NIST, 2019),

$$G = 6;674\,30 \quad 0;000\,15 \quad 10^{-11}\,\frac{m^3}{kg\,s^2}: \tag{4.6}$$

Der omregnes til enhederne AU, dag og kg.

$$G = 6;674\,30 \quad 10^{-11}\,\frac{m^3}{kg\,s^2} \quad \frac{(86\,400\,\frac{s}{dag})^2}{(149\,597\,870\,700\,\frac{m}{AU})^3} \tag{4.7}$$

$$G = 1;488\,185\,2 \quad 10^{-34}\,\frac{AU^3}{kg\,dag^2} \tag{4.8}$$



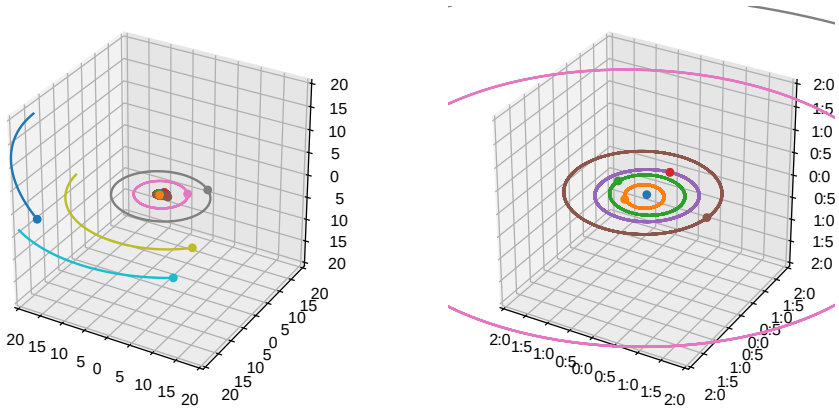

Figur 4.3: 3D visualisering af Solen, planeterne og Plutos position fra 1970 til 2000 (Data fra Horizons). Akserne er i AU.

Ved at initiere en række instanser af `body` objektet med de givne positionsdata, hastighedsdata og GM-værdier, og ved initialisering af en instans af `nbody_system` med den førnævnte værdi af G, skabes en model af solsystemet. Den væsentlige del af koden er vist her:

```
1  body_list = []
2  input_file = csv.DictReader(open("initial.csv", encoding="utf8"))
3  for b in input_file:
4      pos = [float(x) for x in [b["x"], b["y"], b["z"]]]
5      vel = [float(x) for x in [b["vx"], b["vy"], b["vz"]]]
6      GM = float(b["GM"])
7      name = b["name"]
8      body_list.append(body(GM, pos, vel, [0, 0, 0], name))
9
10 system = nbody_system(body_list, 1.4881852e-34)
11 years = 10
12 step_perday = 4
13 stepsize = 1/step_perday
14 for i in range(int(365 * years)):
15     for j in range(step_perday):
16         system.ruth3(stepsize)
17
18 system.show_system_3d_multi(1)
```

Linje 2 indlæser en csv- l med initialbetingelserne angivet i bilag D og GM -værdierne angivet i tabel 3.1. For hvert legeme skaber linje 4-8 en instans af `body` med starttilstanden. n-legeme systemet skabes på linje 10, mens linje 13-16 løser systemet numerisk over det angivne tidsinterval med Ruths metode. Til sidst vises systemet med et 3D-plot på linje 18 (koden kan ndes i bilag

E). Denne kodestump (med de nødvendige pakker) giver resultatet som vist i gur 4.4. I næste kapitel vil denne numeriske løsning blive undersøgt.



Figur 4.4: Legemernes placering fra den numeriske løsning (1970 til 2000). Akserne er i AU.

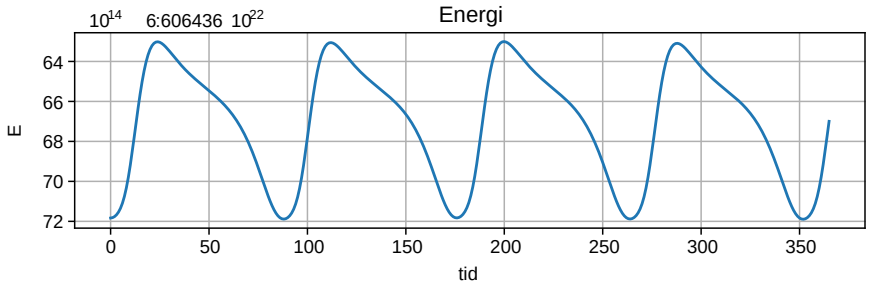
Kapitel 5

# Resultater

Data genereret fra simulationen bliver nu undersøgt for at vurdere nøjagtigheden af den numeriske model.

## 5.1 Energibevarelse

Figur 5.1 viser det numeriske solsystems mekaniske energi over en simuleret tidsperiode på 365 dage med tidsskridtet $t = 1\,\text{dag}$. Bemærk, at y-aksen er skaleret for at synliggøre de numeriske variationer. Svingningerne i dette tilfælde er 8 størrelsesordener mindre end systemets samlede mekaniske energi, og vil derfor være usynlige på en normal y-akse. Enheden for energi i guren er $\frac{\text{kg AU}^2}{\text{dag}^2}$.

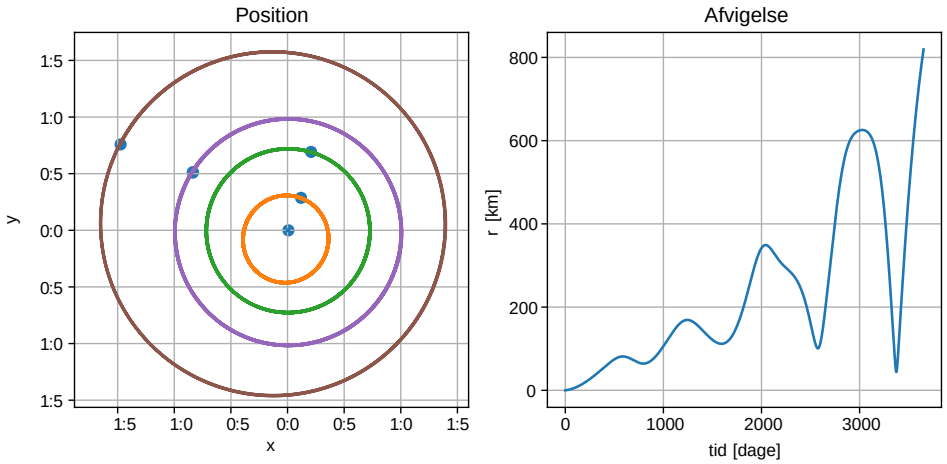



Figur 5.1: Det beregnede energiniveau i det numeriske solsystem

Over en simuleret tidsperiode på 10 år er der ikke synlige ændringer i systemets energiniveau. Den mekaniske energi over 10 år ndes på gur F.1 i bilag F. På gur 5.1 har svingningene en periode på omkring 88 dage. Dette skyldes formentlig Merkurs eccentriske kredsløb, der skaber næsten in nitimale oscillerende afvigelser med en periode lig planetens kredsløbstid på 88 dage (Williams, 2018). Ved at eliminere Merkur fra solsystemet falder afvigelserne med ca. 2 størrelsesordener som vist på gur F.2 i bilaget. Overordnet er afvigelserne så små, at energien i systemet kan beskrives som konstant. Energibevarelse er altså gældende i den implementeret model. For sammenligningens skyld indeholder gur F.3 i bilaget resultatet ved brug af Eulers metode over 10 år med samme parametre, hvor energiniveauet er stigende og planetbanerne særdeles ustabile.

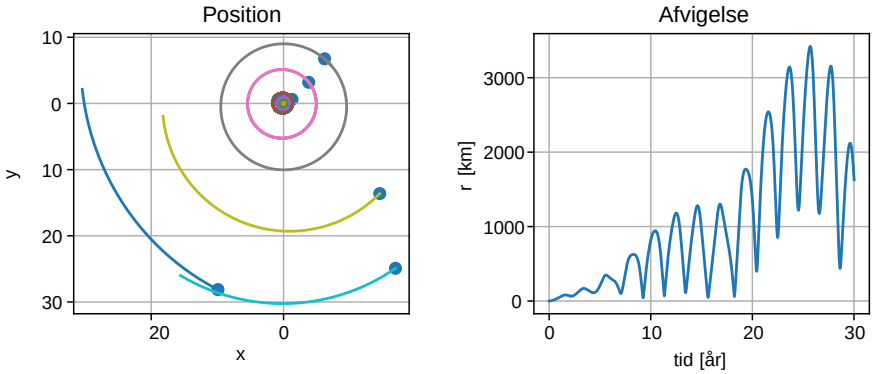
## 5.2 Kredsløbsbaner

Blikket vendes nu mod selve kredsløbsbanerne og nøjagtigheden af de simulerede planetpositioner. Til dette sammenlignes Mars' simulerede position med dataene givet af JPL Horizons. Hvis $r_{jpl}$ er værdien fr Horizons og $r_{model}$ er værdien fra denne model, anvendes afstanden, $|r_{jpl} - r_{model}|$, til at bedømme afvigelsen.




Figur 5.2: (Venstre) Simulerede planetbaner for klippeplaneterne projekteret på xy-planen. (Højre) Afvigelse mellem projektets numeriske model og Horizons data over 10 år ($t = 1$ dag).

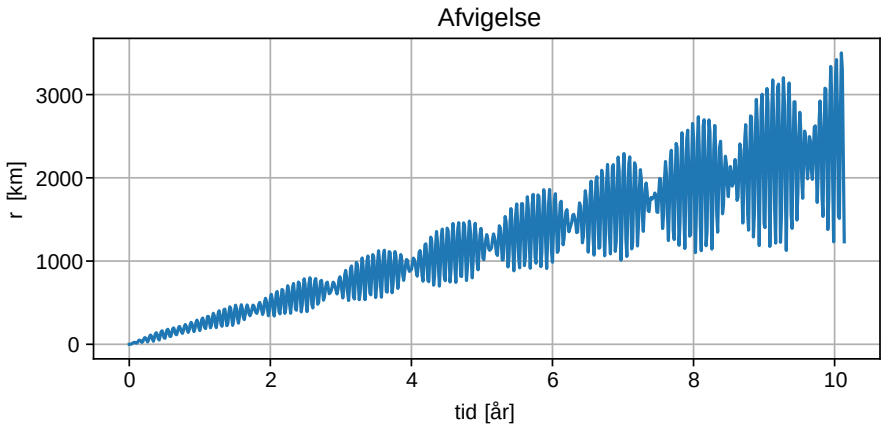
Figur 5.2 viser den absolute afvigelse mellem den numeriske model baseret på Ruths metode og dataene fra JPL horizons. Vedt = 0 er afvigelsen nødvendigvis 0, da initialbetingelserne stammer fra Horizons. Efter 10 år er afvigelsen ca800 km, hvilket svarer til lige under $\frac{1}{4}$ af Mars' radius på 3389,5 km (Williams, 2018). Hvis simulationen fortsættes i en tidsperiode svarende til 30 år, er den maksimale afvigelse fra Horizons data ca. 3400 km som vist i gur 5.3. I Mars' tilfælde følger denne forholdsvis simple model altså JPL's data udarbejdet på baggrund af senere observationer og ere af solsystemets legemer.




Figur 5.3: (Venstre) Simulerede baner. (Højre) Den modellerede Mars' afvigelse fra Horizons data over 30 år ( $t = 1$ dag).

Et tilsvarende diagram for Jordens afvigelse er på gur 5.4. Den maksimale afvigelse efter 10 år

er 3500 km, og er ligeledes inden for Jordens radius. Her er skaber Månens kredsløb oscillationer i datapunkterne.

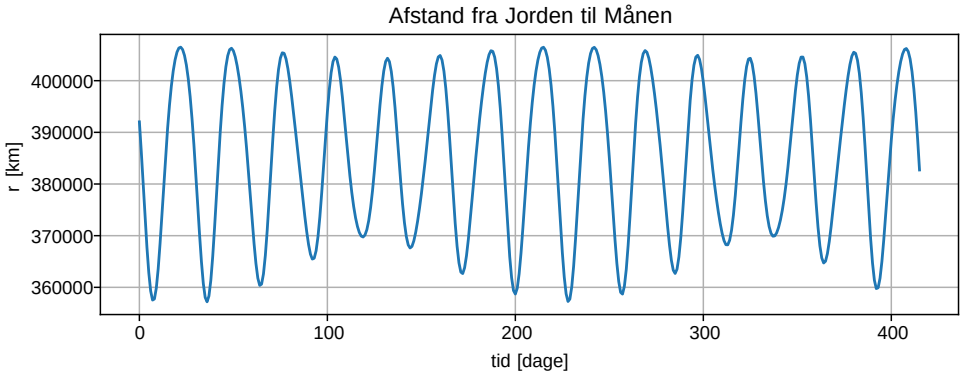



Figur 5.4: Afvigelse mellem simuleret og Horizons data for Jorden over 10 år. t = 1 dag.

For at demonstrere andre planeters ind ydelse på Mars' simulerede kredsløbsbane elimineres Neptun fra solsystemet. Neptun er repræsenteret af den turkise bane på gur 5.3. Neptun be nder sig en betydelig distance fra Mars, og udgør kun 0,005 % af Solens masse. Den resulterende afvigelse fra Horizons data er vist på gur F.4 i bilaget, hvor afvigelsen efter 10 år er ca. 15000 km. Det viser, at alle legemer i n-legeme systemet påvirker hinanden.

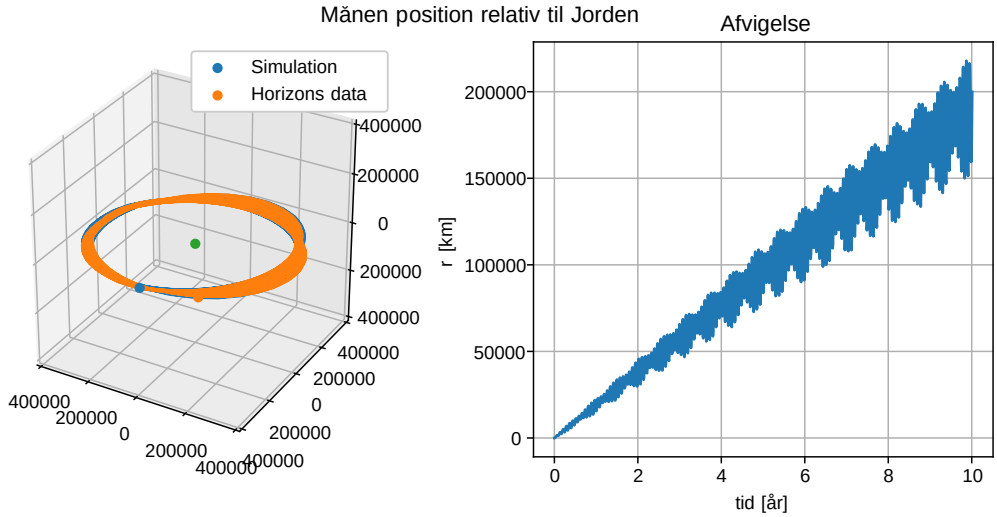
## 5.3 Jord-måne systemet

I den numeriske simulation indgår Jordens måne som den eneste satellit. Månens simulerede positionsdata anvendes derfor til at undersøge algoritmens evne til at simulere på tværs af størrelsesordener. I første omgang plottes den simulerede afstand fra Jorden til Månen som en funktion af tid, hvilket nes på gur 5.5.




Figur 5.5: Simuleret afstand fra Jorden til Månen over 410 dage ( t = 1 dag).

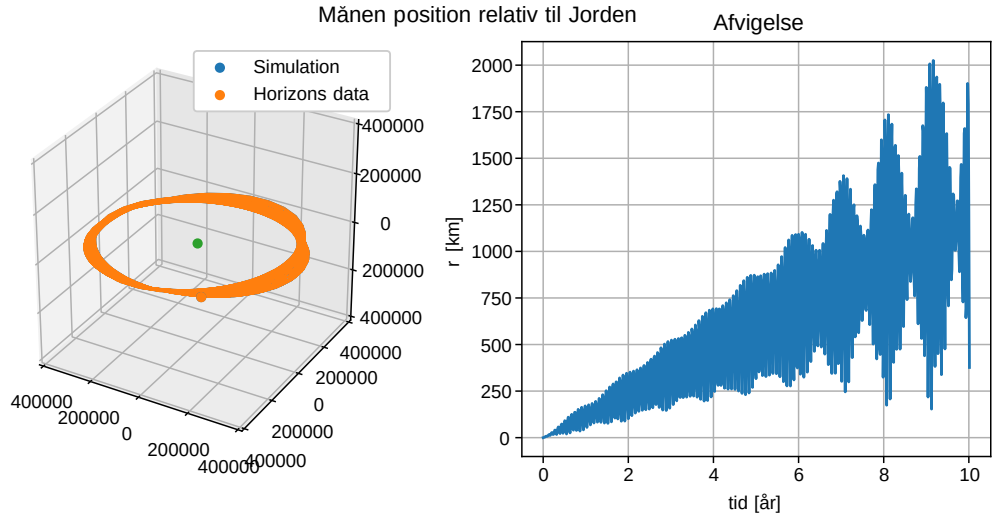
Fordi Månens kredsløb er elliptisk, svinger afstanden mellem Jorden og Månen. Denne svingning kan bruges til at estimere Månens omløbsperiode i simulationen. Over 10 perioder (Mellem 11




Figur 5.6: Månens position relativ til Jorden ( t = 1 dag).

toppunkter i gur 5.5) er gennemsnitsperioden 27;5 dage Det er relativ tæt på den virkelige værdi, 27,32 dage (Williams, 2018).

Månens positionsdata relativ til Jorden vurderes nu ved sammenligning med Horizons data. Her er det tydeligt, at den numeriske fejl over en periode på 10 år er markant. Figur 5.6 viser modellens løsning i forhold til data fra Horizons, hvor afvigelsen efter 10 år stiger til 200000 km.




Figur 5.7: Månens position relativ til Jorden ( t = 0 ;25 dage).

Det viser sig, at en stor del af afvigelsen skyldes tidsskridtet på 1 dag. Dette tidsskridt er tilstrækkelig for de este planeter på solsystemsskala, men da Jord-Måne systemet er mindre i skala, kræves et mindre tidsskridt. Hvis t = 0 ;25 dage vælges i stedet, er afvigelsen efter 10 år kun 2000 km som vist på gur 5.7, hvor det ikke længere er muligt at skelne mellem de gra ske repræsentationer af de endelige positioner. Sådan en ændring i tidsskridt kræver selvfølgelig mere regnetid[1]. Ved det mindre tidsskridt er afvigelsen faldet med en faktor 100, hvilket viser,

[1] Med et tidsskridt på 1 dag over en periode på 10 år tog simulationen 15 s på en standard stationær-computer.

at fejlen i en 3. ordens metode falder hurtigt ved kortere skridtlængde. Bemærk, at simulationen og Horizons data begge viser, at Månens kredsløb ikke er statisk, men varierer over tid. Denne præcession skyldes primært interaktionerne mellem Jorden, Månen og Solen, hvilket skaber et resulterende kredsløb, som ikke udelukkende kan beskrives med Keplers love (Mignard, 1981).

## 5.4 Vurdering og diskussion

Sammenligningen med JPL's model baseret på observationer viser, at den numeriske models resultater følger virkeligheden nøje. Algoritmen er i stand til at bevare energiniveauet i systemet, og afvigelserne for Mars og Jorden er inden for få tusinde kilometer over en tidsperiode på 30 år. De tilstedeværende afvigelser kan forklares af to årsager. For det første kan en del af afvigelserne skyldes andre legemer eller fysiske e ekter, som eksisterer i det virkelige solsystem. Det kan være tyngdepåvirkningerne fra mindre legemer. Usikkerheden her vil dog være minimal, da Solen og planeterne samlet udgør over 99,99 % of solsystemets masse (Berger, 2002). Derudover er det teoretisk set muligt, at en lille brøkdel af variationen stammer fra målbare relativistiske e ekter i tilfælde af Merkur, som har et præcesserende kredsløb, der ikke helt kan beskrives udelukkende med klassisk mekanik. Denne e ekt er dog minimal i forhold til de simulerede afvigelser (Morrison og Ward, 1975). Den største kilde af usikkerhed skyldes den numeriske fejl i simulationen, specielt ved større tidsskridt, hvilket blev illustreret i eksemplet med Jord-Måne systemet. Hvis dette ønskes minimeret, kan et mindre tidsskridt eller en mere avanceret numerisk metode anvendes. Omvendt er det også muligt, at de anvendte initialbetingelser fra NASA betyder, at simulationen præcision er kunstigt forhøjet, da der netop sammenlignes med data fra NASA. Hvis det er tilfældet, viser resultaterne dog alligevel, at der er enighed mellem den implementerede metode og et kompleks etableret datasæt. Det samme kan ikke siges om Eulers metode.

Den anvendte numeriske metode egner sig specielt til løsning af hamiltonske systemer. Udover gravitation kan algoritmen let bruges til andre problemer ved at modi cere beregningen af acceleration på systemets partikler. Eksempler kan  ndes inden for partikelfysik, molekylærdynamik eller fysiske systemer bestående af penduler og fjedre. Det forskningsområde, som beskæftiger sig med disse numeriske simulationer går under navnet beregningsmæssig fysik. Fordelen ved at bruge computersimulationer er, at de muliggør forskning af videnskabelige fænomener, hvor fysiske forsøg er upraktiske eller endda umulige i solsystemets tilfælde. Ved at skabe en digital model, er det muligt at kontrollere alle parametrene, og drage konklusioner om den fysiske verden. Et lille kvalitativt eksempel fra denne rapport er, at eksistensen af Neptun har en vis ind ydelse på resten af planeterne. Simulationer afn-legeme systemer kan blandt andet anvendes til at undersøge formationen af astronomiske strukturer som galakser eller stjernehob.

Inden for ingeniørvidenskab kan lignende numeriske metoder anvendes til at analysere og afprøve fysiske konstruktioner før prototypefasen. Et eksempel kan være simulationer af aerodynamikken af et  y. Tæt knyttet til dette projekt er forskning af rummet, hvor lignende metoder anvendes til at planlægge og simulere kursen for en rumsonde. Fra opsendelsen og ud til destinationen skal rumsonders kurs planlægges nøje for sikre en vellykket mission. Numeriske simulationer kan i disse tilfælde anvendes for at veri cere mulige metoder.

Simulationstiden er proportional med det samlede antal skridt og omvendt proportional med skridtlængden

Kapitel 6

# Konklusion

Teorien om gravitation og kredsløbsbaner har sine moderne rødder i Keplers love og Newton gravitationslov. I et system bestående af ere fælles tiltrækkende legemer, beskriver Newtons ligninger de resulterende kræfter på hver partikel og den resulterende acceleration. Fordi sådan et n-legeme system fundamentalt bevarer den samlede mekaniske energi, er problemet specielt egnet til beskrivelse ved brug af hamiltonsk mekanik. Hamiltonfunktionen beskriver systemets mekaniske energi, hvilket forbliver konstant. Ud fra hamiltonfunktionen er det muligt at udlede en række bevægelsesligninger, der modellerer hvert legemes bevægelse i n-legeme systemet. Bevægelsesligningerne for tre eller ere legemer kan ikke løses analytisk, men kan i stedet analyseres gennem brug af numeriske algoritmer, der estimerer systemets ændring over en række nitte tidsintervaller. Den mest grundlæggende algoritme, Eulers metode, er simpel og ligetil, men her divergerer estimatet ofte fra den virkelige løsning grundet numeriske afvigelser. De såkaldte Runge-Kutta metoder af højere orden er bedre, men for at bevare Hamiltonfunktionens værdi og derved systemets mekaniske energi skal såkaldte symplektiske metoder anvendes, da de er udviklet specielt til løsning af Hamiltons ligninger. Ved at implementere bevægelsesligningerne for gravitation og numeriske algoritmer ved hjælp af programmering, samt ved anvendelse af planetdata fra NASA's Jet Propulsion Laboratory blev en simulation af solsystemets største legemer skabt. Simulationens gyldighed blev undersøgt ved at analysere det simulerede energiniveau over tid, hvilket stemte overens med bevarelsessætningerne. Variationen af energiniveauet viste sig at være 8 størrelsesordener mindre end den samlede energi, med et tidsmæssigt skridtlængde på 1 dag over en periode på 10 år. Sammenligning af Mars' position med NASA's data gav ligeledes fornuftige resultater over en periode på 30 år, med en maksimal absolut afvigelse på 3400 km (svarende til Mars' radius). Simulationen af Månens position krævede et kortere tidsinterval på 0,25 dage før enighed med NASA's datasæt opstod. Projektet har demonstreret, at fysiske love og numeriske algoritmer kan kombineres med programmering for at skabe digitale simulationer af den virkelige verden. Disse simulationer kan blandt andet anvendes til undersøgelse af fysiske fænomener eller under planlægning af interplanetariske rummisioner.

# Bibliogra


Bate, Roger R., Donald D. Mueller og Jerry E. White (2015). Fundamentals of astrodynamics Dover Publications.

Berger, Wolfgang H. (2002).The Jovian Planets. url : http://earthguide.ucsd.edu/virtualmuseum/ita/08_1.shtml (senest bes. 19.12.2019).

Brown, Ernest W m. . (1892). Poincaré's Mécanique Céleste . I: Bulletin of the New York Mathematical Society 1.9, ss. 206 214.

Cartwright, Jon (dec. 2017). Physicists Discover a Whopping 13 New Solutions to Three-Body Problem. url : https://www.sciencemag.org/news/2013/03/physicists-discover-whopping-13-new-solutions-three-body-problem (senest bes. 15.12.2019).

Cohn, Henry (udat.). Why symplectic geometry is the natural setting for classical mechanics url : https://math.mit.edu/~cohn/Thoughts/symplectic.html.

Diacu, Florin (1995). The solution of the n-body problem Tekn. rap. url : https://www.math.uvic.ca/faculty/diacu/diacuNbody.pdf.

Folkner, William (2014). JPL Planetary and Lunar Ephemerides url : https://ssd.jpl.nasa.gov/?planet_eph_export (senest bes. 16.12.2019).

Folkner, William M m. . (2014). The planetary and lunar ephemerides DE430 and DE431 . I: Interplanetary Network Progress Report 196, ss. 1 81.

Hairer, Ernst, Christian Lubich og Gerhard Wanner (2003). Geometric numerical integration illustrated by the Störmer Verlet method . I: Acta numerica 12, ss. 399 450.url : http://www.math.kit.edu/ianm3/lehre/geonumint2009s/media/gni_by_stoermer-verlet.pdf.

Hansen, Henning Bernhard (2013). objektorienteret programmering . I: Den Store Danske, Gyldendal. url : http://denstoredanske.dk/index.php?sideId=134119 (senest bes. 16.12.2019).

Heefelt, Mogens Brun. (1990).Numeriske algoritmer: et emnehæfte til datalogisk matematik Matematiklærerforeningen.

Holck, Per, Jens Kraaer og Birgitte Merci. Lund (2009). Orbit A htx . Systime.

Holmberg, Erik (1941). On the Clustering Tendencies among the Nebulae. II. a Study of Encounters Between Laboratory Models of Stellar Systems by a New Integration Procedure. I: The Astrophysical Journal 94, s. 385.

IAU (2012). On the re-de nition of the astronomical unit of length. url : https://www.iau.org/static/resolutions/IAU2012_English.pd .

JPL (2019a). HORIZONS User Manual. url : https://ssd.jpl.nasa.gov/?horizons_doc (senest bes. 16.12.2019).

(2019b). HORIZONS Web-Interface. url : https://ssd.jpl.nasa.gov/horizons.cgi (senest bes. 17.12.2019).

Kahan, William (1996). Lecture notes on the status of IEEE standard 754 for binary oating-point arithmetic . I: url : https://people.eecs.berkeley.edu/~wkahan/ieee754status/IEEE754.PDF.

Kleitman, Daniel J. (2005). 16.3 The Hamiltonian. url : http://math.mit.edu/classes/18.013A/HTML/chapter16/section03.html.

Koto, Toshiyuki og Eunjee Song (2014). Stability of Explicit Symplectic Partitioned Runge-Kutta Methods . I: Journal of information and communication convergence engineering12.1, ss. 39 45. doi : http://dx.doi.org/10.6109/jicce.2014.12.1.039.

Lewis, John (1997). Physics and Chemistry of the Solar System. Elsevier Science. url : https://books.google.dk/books?id=ApJCRO-_mSUC.

Lützen, Jesper (2009). Liouvilles sætning . I: Den Store Danske, Gyldendal. url : http://denstoredanske.dk/index.php?sideId=117314 (senest bes. 16.12.2019).

Mignard, F (1981). The lunar orbit revisited. III . I: Moon and Planets 24, ss. 189 207.

Montenbruck, Oliver (1992). Numerical integration methods for orbital motion . I: Celestial Mechanics and Dynamical Astronomy 53.1, ss. 59 69.

Morrison, LV og CG Ward (1975). An analysis of the transits of Mercury: 1677 1973 . I: Monthly Notices of the Royal Astronomical Society 173.1, ss. 183 206.

Musielak, ZE og B Quarles (2014). The three-body problem . I: Reports on Progress in Physics 77.6, s. 065901.

Newton, Sir Isaac (1864). Mathematical Principles of Natural Philosophy. Overs. af Andrew Motte. url : https://archive.org/details/newtonspmathema00newtrich.

Nikolic, Branislav K. (2002). Verlet Method. url : http://www.physics.udel.edu/~bnikolic/teaching/phys660/numerical_ode/node5.html.

NIST (2019). The NIST Reference on Constants, Units, and Uncertainty. url : https://physics.nist.gov/cgi-bin/cuu/Value?bg (senest bes. 18.12.2019).

Nordkvist, Nikolaj og Poul G Hjorth (2005). Classical Mechanics and Symplectic Integration . I: url : https://orbit.dtu.dk/en/publications/classical-mechanics-and-symplectic-integration.

Piziak, R og JJ Mitchell (2001). Hamiltonian formalism and the state of a physical system . I: Computers & Mathematics with Applications 42.6-7, ss. 793 805. doi : https://doi.org/10.1016/S0898-1221(01)00199-7.

Ruth, Ronald D (1983). A canonical integration technique . I: IEEE Trans. Nucl. Sci. 30.CERN-LEP-TH-83-14, ss. 2669 2671.

Smith, Anders og Mogens Esrom Larsen (2010). numerisk analyse . I: Den Store Danske, Gyldendal. url : http://denstoredanske.dk/index.php?sideId=133401 (senest bes. 14.12.2019).

Smith, Henrik (2009). Faserum . I: Den Store Danske, Gyldendal. url : http://denstoredanske.dk/index.php?sideId=74580 (senest bes. 14.12.2019).

Williams, David R. (2018). Planetary Fact Sheets. url : https://nssdc.gsfc.nasa.gov/planetary/planetfact.html (senest bes. 18.12.2019).

Aarseth, Sverre J. (2003). Gravitational N-Body Simulations: Tools and Algorithms. Cambridge Monographs on Mathematical Physics. Cambridge University Press. doi : 10.1017/CBO9780511535246.

Bilag A

# Udledning af Hamiltons bevægelsesligninger for n-legeme problemet

Hamiltonfunktionen for n-legeme problemet er gengivet her.

$$H = T + U = \sum_{i=1}^{n} \frac{|\vec{p}_i|^2}{2m_i} \quad \sum_{i=1}^{n} \sum_{j>i} \frac{Gm_i m_j}{|\vec{r}_i \quad \vec{r}_k|} \tag{A.1}$$

Bevægelsesligningerne er,

$$\frac{d\vec{p}_i}{dt} = \quad \frac{\partial H}{\partial \vec{r}_i} \; ; \; \frac{d\vec{r}_i}{dt} = \frac{\partial H}{\partial \vec{p}_i} : \tag{A.2}$$

Først di erentieres $H$ med hensyn til $\vec{p}_k$ (1 k n vælges som indeks for at forhindre tvetydige variabler).

$$\frac{d\vec{r}_k}{dt} = \frac{\partial H}{\partial \vec{p}_k} = \frac{\partial}{\partial \vec{p}_k} \sum_{i=1}^{n} \frac{|\vec{p}_i|^2}{2m_i} \quad \sum_{i=1}^{n} \sum_{j>i} \frac{Gm_i m_j}{|\vec{r}_i \quad \vec{r}_k|} \tag{A.3}$$

Bemærk, at hele andet led er konstant med hensyn til $\vec{p}_k$. Leddet kan derfor ignoreres.

$$\frac{d\vec{r}_k}{dt} = \frac{\partial}{\partial \vec{p}_k} \sum_{i=1}^{n} \frac{|\vec{p}_i|^2}{2m_i} \tag{A.4}$$

Hvis summen udvides, ndes $\vec{p}_k$ kun i et af de resulterende led. Alle andre led forsvinder derfor under di erentiationen.

$$\frac{d\vec{r}_k}{dt} = \frac{\partial}{\partial \vec{p}_k} \frac{|\vec{p}_k|^2}{2m_k} \tag{A.5}$$

Da der di erentieres med hensyn til en vektor, er resultatet lig med en vektor, hvor værdien af hver komposant er udtrykket di erentieret med hensyn til den komposant[1].

$$\frac{d\vec{p}_k}{dt} = \frac{\partial H}{\partial \vec{p}_k} = \frac{\partial}{\partial \vec{p}_k} \frac{p_{k;1}^2 + p_{k;2}^2 + p_{k;3}^2}{2m_k} \tag{A.6}$$

Hver komposants resulterende værdi er $\frac{\partial}{\partial p_{k;n}} \frac{p_{k;n}^2}{2m_k} = \frac{p_{k;n}}{m}$. Resultatet er derfor:

$$\frac{d\vec{p}_k}{dt} = \frac{\vec{p}_k}{m_k} \tag{A.7}$$

[1] https://www.cse.huji.ac.il/~csip/tirgul3_derivatives.pdf

Nu bestemmes $\frac{d\vec{p}_k}{dt}$ ved at differentiere $H$ med hensyn til $\vec{r}_k$.

$$\frac{d\vec{p}_k}{dt} = \frac{\partial H}{\partial \vec{r}_k} = \frac{\partial}{\partial \vec{r}_k} \sum_{i=1}^{n} \frac{|\vec{p}_i|^2}{2m_i} \sum_{i=1}^{n} \sum_{j>i}^{n} \frac{Gm_i m_j}{|\vec{r}_i - \vec{r}_j|} \tag{A.8}$$

Da $r_k$ kun findes i anden summation, ignoreres det første led.

$$\frac{d\vec{p}_k}{dt} = \frac{\partial}{\partial \vec{r}_k} \sum_{i=1}^{n} \sum_{j>i}^{n} \frac{Gm_i m_j}{|\vec{r}_i - \vec{r}_j|} \tag{A.9}$$

Minus går ud

$$\frac{d\vec{p}_k}{dt} = \frac{\partial}{\partial \vec{r}_k} \sum_{i=1}^{n} \sum_{j>i}^{n} \frac{Gm_i m_j}{|\vec{r}_i - \vec{r}_j|} \tag{A.10}$$

Fordi udtrykket i summationen beregnes for alle par af $i$ og $j$, indgår $\vec{r}_k$ sammen med alle andre legemer. Summationen omskrives, så kun de led med $\vec{r}_k$ indgår. Bemærk, at $j$ og $i$ ikke kan være lig med hinanden.

$$\frac{d\vec{p}_k}{dt} = \frac{\partial}{\partial \vec{r}_k} \sum_{\substack{i=0 \\ i \neq k}}^{n} \frac{Gm_i m_k}{|\vec{r}_k - \vec{r}_i|} \tag{A.11}$$

Konstanterne trækkes ud og udtrykket differentieres med kædereglen

$$\frac{d\vec{p}_k}{dt} = \sum_{\substack{i=0 \\ i \neq k}}^{n} Gm_k m_i \frac{\partial}{\partial \vec{r}_k} \frac{1}{|\vec{r}_k - \vec{r}_i|} \tag{A.12}$$

$$\frac{d\vec{p}_k}{dt} = \sum_{\substack{i=0 \\ i \neq k}}^{n} Gm_k m_i \frac{\partial}{\partial \vec{r}_k} (|\vec{r}_k - \vec{r}_i|)^{-1} \tag{A.13}$$

$$\frac{d\vec{p}_k}{dt} = \sum_{\substack{i=0 \\ i \neq k}}^{n} Gm_k m_i (|\vec{r}_k - \vec{r}_i|)^{-2} \frac{\partial}{\partial \vec{r}_k} |\vec{r}_k - \vec{r}_i| \tag{A.14}$$

Resultatet af $\frac{\partial}{\partial \vec{r}_k} |\vec{r}_k - \vec{r}_i|$ er en enhedsvektor $\frac{\vec{r}_k - \vec{r}_i}{|\vec{r}_k - \vec{r}_i|}$

$$\frac{d\vec{p}_k}{dt} = \sum_{\substack{i=0 \\ i \neq k}}^{n} \frac{Gm_k m_i (\vec{r}_k - \vec{r}_i)}{|\vec{r}_k - \vec{r}_i|^3} \tag{A.15}$$

Bilag B

# Omskrivning af Newtons anden lov

Den anden-ordens di erentialligning for n-legeme problemet, er gengivet her:

$$m_i \ddot{\vec{r}}_i = \sum_{\substack{j=0 \\ j \neq i}}^{n} G \frac{m_i m_j (\vec{r}_i - \vec{r}_j)}{|\vec{r}_i - \vec{r}_j|^3} \tag{B.1}$$

$Gm_i$ trækkes ud af summen.

$$m_i \ddot{\vec{r}}_i = Gm_i \sum_{\substack{j=0 \\ j \neq i}}^{n} \frac{m_j (\vec{r}_i - \vec{r}_j)}{|\vec{r}_i - \vec{r}_j|^3} \tag{B.2}$$

Der divideres med $m_i$

$$\ddot{\vec{r}}_i = G \sum_{\substack{j=0 \\ j \neq i}}^{n} \frac{m_j (\vec{r}_i - \vec{r}_j)}{(|\vec{r}_i - \vec{r}_j|)^3} \tag{B.3}$$

For at konvertere denne til et par koblede første-ordens di erentialligninger indføres hastighedsvektoren $\vec{v}$ (Piziak og Mitchell, 2001, s. 794),

$$\vec{v} = \dot{\vec{r}} \tag{B.4}$$

$$\dot{\vec{v}} = \ddot{\vec{r}} \tag{B.5}$$

ved simpel substitution fører det til bevægelsesligningerne af første orden,

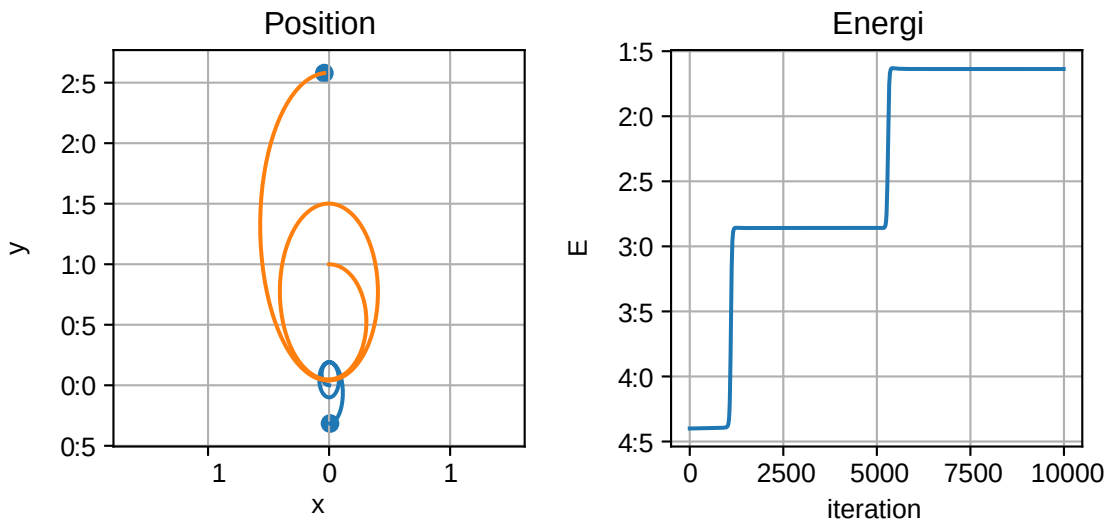
$$\dot{\vec{r}} = \vec{v} \tag{B.6}$$

$$\dot{\vec{v}} = G \sum_{\substack{j=0 \\ j \neq i}}^{n} \frac{m_j (\vec{r}_i - \vec{r}_j)}{(|\vec{r}_i - \vec{r}_j|)^3}. \tag{B.7}$$

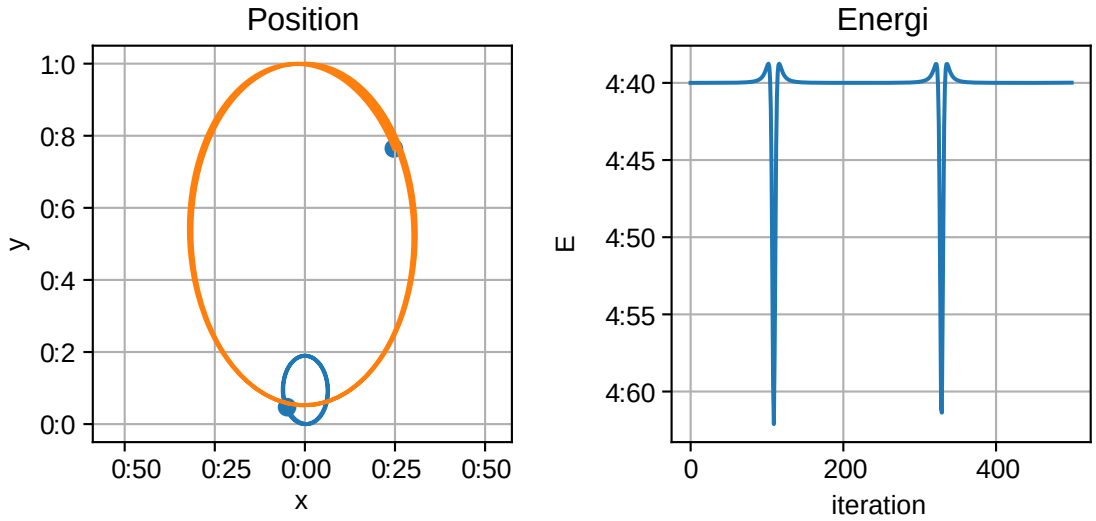
De sidste udtryk er identiske med udtrykkene udledt gennem hamiltonsk mekanik.

Bilag C

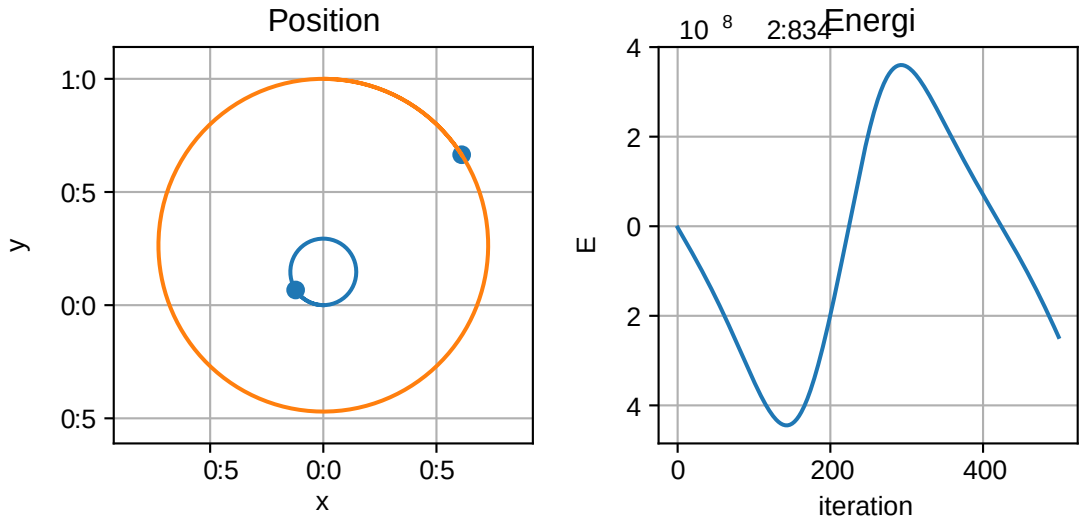
# Simulation af to-legeme problem




Figur C.1: Resultatet af Eulers metode til to-legeme problemet $t = 0;0005$




Figur C.2: Resultatet af Verlet metoden til to-legeme problemet $t = 0;005$.




Figur C.3: Resultatet af Ruths metode til to-legeme problemet med lavere eccentricitet $t = 0;005$.

Bilag D

# NASA Data

| Legeme | x | y | z |
|---|---|---|---|
| | [au] | [au] | [au] |
| Sun | 0.004306786483674715 | 0.001837535134108785 | -6.089583868599627e-05 |
| Mercury | 0.2608031036290285 | 0.1941548012009124 | -0.007920680050178147 |
| Venus | -0.03163833272887579 | -0.7240794841588817 | -0.007846758267002717 |
| Earth | -0.1762267229040138 | 0.9684335265498731 | 3.860769680717466e-06 |
| Moon | -0.1787960981527179 | 0.9679289211958966 | -0.000100778811079838 |
| Mars | 1.330411585966007 | 0.4980293609694482 | -0.0223529272952337 |
| Jupiter | -5.006795142311221 | -2.138375001047115 | 0.1210075675379885 |
| Saturn | 7.242418408520271 | 5.690983176953221 | -0.3873316179327782 |
| Uranus | -18.20862604496443 | -1.932694891138334 | 0.2291418327688023 |
| Neptune | -15.55788654385216 | -26.00847573815253 | 0.893962458830729 |
| Pluto | -30.41812591647754 | 2.124458945607829 | 8.570729686704894 |

Tabel D.1: Legemernes position d. 1. januar 1970.

| Legeme | vx | vy | vz |
|---|---|---|---|
| | [au/dag] | [au/dag] | [au/dag] |
| Sun | -1.819961544921342e-06 | 5.304893809120167e-06 | 1.938237686924352e-08 |
| Mercury | -0.02240357151534085 | 0.02373852851400617 | 0.003995763813333178 |
| Venus | 0.02006306534672056 | -0.001072254244657087 | -0.001173273323954599 |
| Earth | -0.01719568902488065 | -0.003210508485900838 | 2.480944449126105e-07 |
| Moon | -0.01704844265352481 | -0.003765966714598013 | -4.396576331390077e-05 |
| Mars | -0.004366714270603195 | 0.0143061322209902 | 0.0004070441686830017 |
| Jupiter | 0.002875828217850848 | -0.006589043275627507 | -3.732762958497579e-05 |
| Saturn | -0.003748446018618532 | 0.004375007876010112 | 7.246809953213954e-05 |
| Uranus | 0.0003858506265054555 | -0.004094769175075032 | -2.0359868048643e-05 |
| Neptune | 0.002674515405738786 | -0.001593012414617136 | -2.861141150447117e-05 |
| Pluto | 0.0003651488253370403 | -0.003322913885520158 | 0.0002547141809385972 |

Tabel D.2: Legemernes hastighed d. 1. januar 1970.

Bilag E

# Kode

## E.1 body

```
class body:
    def __init__(self, GM, position, velocity, acceleration, name="body"):
        self.GM = GM
        self.mass = GM / constants.G
        self.pos = np.array(position, dtype=np.float64)
        self.vel = np.array(velocity, dtype=np.float64)
        self.acc = np.array(acceleration, dtype=np.float64)
        self.last_acc = np.array(acceleration, dtype=np.float64)
        self.name = name
```

## E.2 nbody_system

```
class nbody_system:
    def __init__(self, body_list, G=1):
        self.body_list = body_list
        self.n = len(body_list)
        self.G = G

        self.time = 0

        # get dimension from pos of first body
        self.dimension = len(self.body_list[0].pos)

        # Data logging
        self.xs = [[] for i in range(self.n)]
        self.ys = [[] for i in range(self.n)]
        self.zs = [[] for i in range(self.n)]
        self.E_data = []
        self.times = []

        self.log_data()

        # update inital gravitational accelerations
        self.update_acc()


    def log_data(self):
        # log position data for each body
        for i in range(self.n):
            self.xs[i].append(self.body_list[i].pos[0])
            self.ys[i].append(self.body_list[i].pos[1])

```

```
31              if self.dimension == 3:
32                  # add z-dimension
33                  self.zs[i].append(self.body_list[i].pos[2])
34
35          # log total energy and time
36          self.E_data.append(self.total_energy())
37          self.times.append(self.time)
38
39
40      def update_acc(self):
41          # calculate gravitational interaction between each body
42          for i in range(self.n):
43              # forces on body i
44              body_i = self.body_list[i]
45              # save previous acceleration
46              body_i.last_acc = np.copy(body_i.acc)
47
48              # reset current acceleration and sum:
49              body_i.acc *= 0
50              for j in range(self.n):
51                  if j != i:
52                      # every other body
53                      body_j = self.body_list[j]
54                      # calculate acceleration
55
56                      body_i.acc -= body_j.GM * (body_i.pos - body_j.pos) /
                        (np.linalg.norm(body_i.pos - body_j.pos)**3)
57
58
59      def euler(self, dt=0.005):
60          # calculate new gravitational acceleration for each body.
61          self.update_acc()
62
63          for i in range(self.n):
64              body_i = self.body_list[i]
65
66              #position full-step
67              body_i.pos += dt * body_i.vel
68
69              # velocity step
70              body_i.vel += dt * body_i.acc
71
72          self.time += dt
73          self.log_data()
74
75
76      def symplectic_euler(self, dt=0.005):
77          self.update_acc()
78
79          for i in range(self.n):
80              body_i = self.body_list[i]
81              body_i.vel += dt * body_i.acc
82              body_i.pos += dt * body_i.vel
83
84          self.time += dt
85          self.log_data()
86
87      def verlet(self, dt=0.005):
88
89          for i in range(self.n):
```

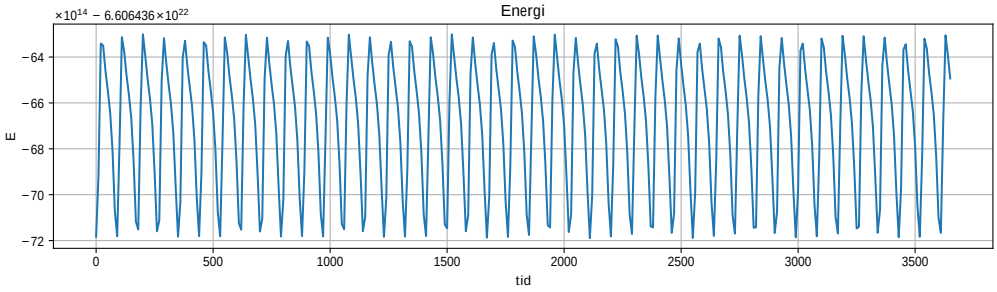
```
90              # velocity half-step
91              body_i = self.body_list[i]
92              body_i.vel += dt/2 * body_i.acc
93
94              #position full-step
95              body_i.pos += dt * body_i.vel
96
97          self.update_acc()
98
99          for i in range(self.n):
100             # final velocity half-step
101             body_i = self.body_list[i]
102             body_i.vel += dt/2 * body_i.acc
103
104         self.time += dt
105         self.log_data()
106
107     def ruth3(self, dt=0.005):
108         # 3rd order symplectic integrator
109         c = [7/24, 3/4, -1/24]
110         d = [2/3, -2/3, 1]
111
112         for idx in range(3):
113             # update acceleration for new pos
114             self.update_acc()
115
116             for i in range(self.n):
117                 body_i = self.body_list[i]
118                 # update pos and vel using coeffecients
119                 body_i.vel += body_i.acc * c[idx] * dt
120                 body_i.pos += body_i.vel * d[idx] * dt
121
122         self.time += dt
123         self.log_data()
124
125     def total_energy(self):
126         E_total = 0
127
128         for i in range(self.n):
129             body_i = self.body_list[i]
130             #kinetic energy
131             E_total += 1/2 * body_i.GM * np.linalg.norm(body_i.vel)**2
132
133             for j in range(i+1, self.n):
134                 body_j = self.body_list[j]
135                 # potential energy
136                 E_total -= body_i.GM * body_j.GM/(np.linalg.norm(body_i.pos - body_j.pos))
137
138         return E_total/self.G
139
140     def show_system_energy(self, plot_samle_interval = 1):
141         fig, ax = plt.subplots(1, 1, figsize=(6,3))
142
143         ax.set_title("Energi")
144         ax.set_xlabel( ' tid ' )
145         ax.set_ylabel( ' E' )
146         ax.grid(True)
147
148         print ("Plotting {} datapoints".format(int(len(self.E_data)/plot_samle_interval)))
149
```

```
150             ax.plot(self.times[::plot_samle_interval], self.E_data[::plot_samle_interval])
151
152             fig.tight_layout()
153             plt.show()
154
155         def show_system_3d(self, plot, plot_samle_interval):
156             for i in range(self.n):
157                 # plot position history
158                 plot.plot(self.xs[i][::plot_samle_interval], self.ys[i][::plot_samle_interval],
                    ↵     self.zs[i][::plot_samle_interval], linewidth=1, alpha=0.6)
159
160                 # Same color as lines
161                 plot.scatter(self.xs[i][-1],self.ys[i][-1], self.zs[i][-1], [0.3])
162
163         def show_system_3d_multi(self, plot_samle_interval = 1):
164
165             fig = plt.figure(figsize=(5.8, 3.0))
166
167             ax = fig.add_subplot(121, autoscale_on=False, projection='3d')
168             ax.set_xlim3d(-20, 20)
169             ax.set_ylim3d(-20,20)
170             ax.set_zlim3d(-20,20)
171
172             ax.set_aspect('equal')
173             ax.grid()
174
175             self.show_system_3d(ax, plot_samle_interval)
176
177             ax2 = fig.add_subplot(122, autoscale_on=False, projection='3d')
178             ax2.set_xlim3d(-2, 2)
179             ax2.set_ylim3d(-2,2)
180             ax2.set_zlim3d(-2,2)
181
182             ax2.set_aspect('equal')
183             ax2.grid()
184             self.show_system_3d(ax2, plot_samle_interval)
185
186             plt.show()
187
188         def show_system_2d_energy(self, plot_samle_interval = 1):
189             fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(6,3))
190
191             ax1.set_title("Position")
192             ax1.set_xlabel('$x$')
193             ax1.set_ylabel('$y$')
194             ax1.grid(True)
195
196
197             ax2.set_title("Energi")
198             ax2.set_xlabel('iteration')
199             ax2.set_ylabel('E')
200             ax2.grid(True)
201
202             for i in range(self.n):
203                 # plot position history
204                 ax1.plot(self.xs[i][::plot_samle_interval], self.ys[i][::plot_samle_interval])
205
206                 # Last positions
207                 ax1.scatter(self.xs[i][-1],self.ys[i][-1])
208
```

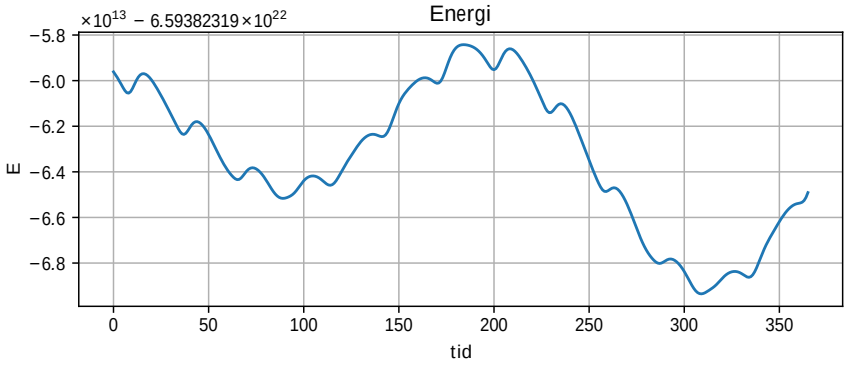
```
209         # plot energy
210         ax2.plot(self.times[::plot_samle_interval], self.E_data[::plot_samle_interval])
211 
212         fig.tight_layout()
213         plt.show()
```

Bilag F

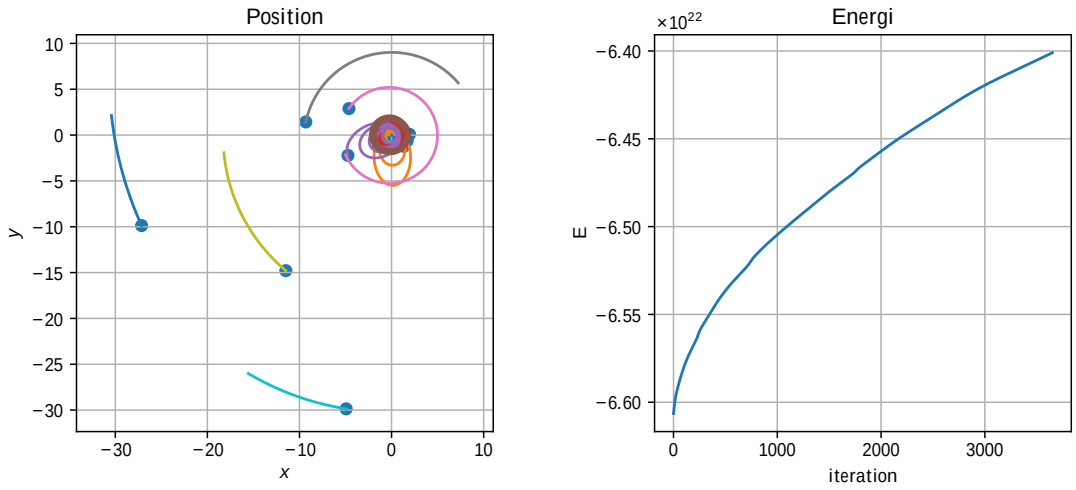
# Figurer



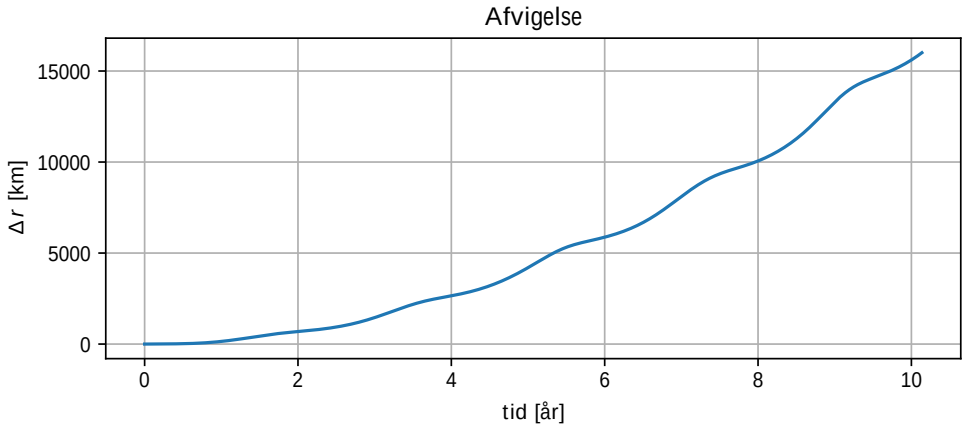

Figur F.1: Den samlede mekaniske energi i det simulerede solsystem over 3650 dage $\Delta t = 1$ dag.




Figur F.2: Den samlede mekaniske energi i det simulerede solsystem uden Merkur over 365 dage $\Delta t = 1$ dag.




Figur F.3: Solsystemet simuleret med Eulers metode over 3650 dage $\Delta t = 1$ dag.




Figur F.4: Afvigelse for Mars, hvis Neptun ikke findes i solsystemet, i forhold til Horizons data over 10 år. $\Delta t = 1$ dag.