# Prosjekt 2

Live Wang Jensen

October 3, 2016

**Abstract**

Vi skal i dette prosjektet se nærmere på bølgefunksjonene til hhv ett og to elektroner i et harmonisk-oscillator potensial. Disse finnes ved å løse en tredimensjonal Schrödingerligning. Ligningen kan skrives om til en egenverdiligning som løses ved hjelp av Jacobis metode. Vi har sett at denne metoden er tidkrevende i forhold til Armadillos innebygde funksjon `eig_sym`. Ved bruk av Jacobis metode fant vi at de tre laveste egenverdiene stemte overens med den gitte løsningen 3, 7, 11 med opp til fire gjeldende siffere etter desimalen ved $n \approx 300$. Sammenligningen av bølgefunksjonene for to elektroner med og uten Coulomb-vekselvirkning viser at bølgefunksjonene *med* vekselvirkning er mer utsmørt i radiell retning, som forventet.

## 1 Introduksjon

Vi skal i dette prosjektet løse Schrödingerligningen for to elektroner i en tredimensjonal harmonisk oscillator brønn, både med og uten Coulomb-vekselvirkning. Vi kommer til å omforme og diskretisere Schrödingerligningen slik at vi ender opp med et egenverdi-problem. Dette kan løses med Jacobis metode. Det utvikles derfor en egen kode som implementerer denne metoden. Vi skal også bruke Armadillos egen funksjon `eig_sym` til å løse samme problemstilling. De to løsningsmetodene kan deretter sammenlignes når det kommer til beregningstid.

## 2 Teori

La oss starte med å se på en såkalt **unitær transformasjon**. En slik transformasjon bevarer ortogonaliteten til egenvektorene. Anta at vi har en ortogonal basis

$$\mathbf{v}_i = \begin{bmatrix} v_{i1} \\ \cdots \\ \cdots \\ v_{in} \end{bmatrix}$$

hvor

$$\mathbf{v}_j^T \mathbf{v}_i = \delta_{ij}$$

En unitær transformasjon på formen

$$\mathbf{w}_i = \mathbf{U}\mathbf{v}_i$$

vil bevare både prikk-produktet og ortogonaliteten. Dette kan vises ved

$$\mathbf{w}_j^T\mathbf{w}_i = (\mathbf{U}\mathbf{v}_j)^T\mathbf{w}_i = \mathbf{v}_j^T\mathbf{U}^T\mathbf{U}\mathbf{v}_i = \mathbf{v}_j^T I\mathbf{v}_i = \mathbf{v}_j^T\mathbf{v}_i = \delta_{ij}$$

Vi skal i det følgende anta at de to elektronene befinner seg i et tre-dimensjonalt harmonisk oscillator-potensial. Coulomb-vekselvirkningen mellom elektronene fører til at de frastøter hverandre, og vi antar sfærisk symmetri. Som kjent består Schrödingerligningen av en angulær del og en radiell del. Den angulære delen kan løses analytisk for en harmonisk oscillator. Den radielle delen må løses numerisk, og vi skal derfor se nærmere på denne. Radialligningen er gitt som

$$-\frac{\hbar^2}{2m}\left(\frac{1}{r^2}\frac{d}{dr}r^2\frac{d}{dr} - \frac{l(l+1)}{r^2}\right)R(r) + V(r)R(r) = ER(r). \tag{1}$$

hvor $V(r)$ er harmonisk oscillator-potensialet (heretter kalt H.O.) gitt ved

$$V(r) = \frac{1}{2}kr^2 \tag{2}$$

hvor $k = m\omega^2$. $E$ er energien til H.O. potensialet, gitt ved

$$E_{nl} = \hbar\left(2n + l + \frac{3}{2}\right) \tag{3}$$

hvor hovedkvantetallet $n = 0, 1, 2, ...$ og banespinnkvantetallet $l = 0, 1, 2, ....$ Her er $\omega$ vinkelfrekvensen.

Siden vi i dette tilfellet bruker kulekoordinater, så må avstanden $r$ fra sentrum befinne seg i intervallet $r = [0, \infty)$. Vi definerer $u(r) = rR(r)$, slik at vi får

$$R(r) = \frac{1}{r}u(r) \tag{4}$$

Setter vi dette inn i radialligningen, ender vi opp med uttrykket

$$-\frac{\hbar^2}{2m}\frac{d^2}{dr^2}u(r) + \left(V(r) + \frac{l(l+1)}{r^2}\frac{\hbar^2}{2m}\right)u(r) = Eu(r) \tag{5}$$

Hvor vi har Dirichlet grensebetingelser gitt ved $u(0) = 0$ og $u(\infty) = 0$. I kvantefysikk kan tallene fort bli veldig små, derfor vil det være en fordel om denne ligningen besto av dimensjonsløse variabler. Vi kan oppnå dette ved å introdusere den dimensjonsløse variabelen $\rho = r/\alpha$, hvor $\alpha$ er en fritt valgt konstant med enheten *lengde*. Setter vi $r = \rho\alpha$ inn i ligningen, får vi

$$-\frac{\hbar^2}{2m\alpha^2}\frac{d^2}{d\rho^2}u(\rho) + \left(V(\rho) + \frac{l(l+1)}{\rho^2}\frac{\hbar^2}{2m\alpha^2}\right)u(\rho) = Eu(\rho) \tag{6}$$

Vi skal heretter bruke at banespinnkvantetallet $l = 0$. H.O. potensialet kan skrives som funksjon av $\rho$ slik at $V(\rho) = \frac{1}{2}k\alpha^2\rho^2$. Dette forenkler ligningen vår til

$$-\frac{\hbar^2}{2m\alpha^2}\frac{d^2}{d\rho^2}u(\rho) + \frac{k}{2}\alpha^2\rho^2 u(\rho) = Eu(\rho) \tag{7}$$

Multipliserer så leddet $\frac{2m\alpha^2}{\hbar}$ på hver side av ligningen og får

$$-\frac{d^2}{d\rho^2}u(\rho) + \frac{mk}{\hbar^2}\alpha^4\rho^2 u(\rho) = \frac{2m\alpha^2}{\hbar^2}Eu(\rho) \tag{8}$$

Siden konstanten $\alpha$ kan bestemmes fritt, kan vi sette

$$\frac{mk}{\hbar^2}\alpha^4 = 1$$

som gir oss at

$$\alpha = \left(\frac{\hbar^2}{mk}\right)^{1/4}$$

Vi definerer så at

$$\lambda = \frac{2m\alpha^2}{\hbar^2}E$$

og ender opp med en ligning på formen

$$-\frac{d^2}{d\rho^2}u(\rho) + \rho^2 u(\rho) = \lambda u(\rho) \tag{9}$$

Denne ligningen kan løses numerisk. La oss først ta for oss uttrykket for den andrederiverte som funksjon av $u$

$$u'' = \frac{u(\rho+h) - 2u(\rho) + u(\rho-h)}{h^2} + O(h^2) \tag{10}$$

her er $h$ er steglengden, gitt ved

$$h = \frac{\rho_N - \rho_0}{N}. \tag{11}$$

hvor $N$ er antall mesh-points, $\rho_{min} = \rho_0$ og $\rho_{max} = \rho_N$. Vi vet at $\rho_{min} = 0$, mens $\rho_{max}$ må vi bestemme grensen på selv, da $\rho_{max} = \infty$ vil være litt vanskelig for en datamaskin å håndtere. Verdien til $\rho$ ved et punkt $i$ vil være

$$\rho_i = \rho_0 + ih \quad i = 1, 2, ..., N$$

Vi er nå i stand til å diskretisere Schrödingerligningen ved hjelp av $\rho_i$, slik at

$$-\frac{u(\rho_i+h) - 2u(\rho_i) + u(\rho_i-h)}{h^2} + \rho_i^2 u(\rho_i) = \lambda u(\rho_i),$$

som på en mer kompakt form kan skrives

$$-\frac{u_{i+1}-2u_i+u_{i-1}}{h^2}+\rho_i^2 u_i=\lambda u_i$$

eller

$$-\frac{u_{i+1}-2u_i+u_{i-1}}{h^2}+V_i u_i=\lambda u_i$$

Her er $V_i=\rho_i^2$ er H.O. potensialet vårt. Denne ligningen kan nå skrives som en matriseligning. Vi definerer først diagonalelementene som

$$d_i=\frac{2}{h^2}+V_i$$

Ikke-diagonal elementene er *identiske* og defineres som

$$e_i=-\frac{1}{h^2}$$

Ligningen vår kan nå skrives som

$$d_i u_i+e_{i-1}u_{i-1}+e_{i+1}u_{i+1}=\lambda u_i, \tag{12}$$

hvor $u_i$ er ukjent. Settes dette opp på matriseform ser det ut som følger

$$\begin{bmatrix} d_0 & e_0 & 0 & 0 & \dots & 0 & 0 \\ e_1 & d_1 & e_1 & 0 & \dots & 0 & 0 \\ 0 & e_2 & d_2 & e_2 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & \dots e_{N-1} & d_{N-1} & e_{N-1} \\ 0 & \dots & \dots & \dots & \dots & e_N & d_N \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \dots \\ \dots \\ \dots \\ u_N \end{bmatrix} = \lambda \begin{bmatrix} u_0 \\ u_1 \\ \dots \\ \dots \\ \dots \\ u_N \end{bmatrix} \tag{13}$$

Dette er et **egenverdiproblem** som kan løses med Jacobis metode.

## 2.1 To elektroner som vekselvirker

Vi skal nå ta for oss tilfellet hvor de to elektronene *vekselvirker* med hverandre via den frastøtende Coulombkraften. For et enkelt elektron kan radialligningen skrives

$$-\frac{\hbar^2}{2m}\frac{d^2}{dr^2}u(r)+\frac{1}{2}kr^2u(r)=E^{(1)}u(r) \tag{14}$$

hvor $E^{(1)}$ står for energien når vi kun har ett elektron. Dersom vi har *to* elektroner *uten* Coulombvekselvirkning, vil ligningen se slik ut

$$\left(-\frac{\hbar^2}{2m}\frac{d^2}{dr_1^2}-\frac{\hbar^2}{2m}\frac{d^2}{dr_2^2}+\frac{1}{2}kr_1^2+\frac{1}{2}kr_2^2\right)u(r_1,r_2)=E^{(2)}u(r_1,r_2) \tag{15}$$

hvor bølgefunksjonen $u$ nå er funksjon av *begge* elektronenes posisjon, og energien $E^{(2)}$ avhenger av begge elektronenes energi. Dersom vi innfører den

relative avstanden $\mathbf{r} = \mathbf{r}_1 - \mathbf{r}_2$ og massesenteret $\mathbf{R} = 1/2(\mathbf{r}_1 + \mathbf{r}_2)$, kan vi skrive om ligningen til

$$\left(-\frac{\hbar^2}{m}\frac{d^2}{dr^2} - \frac{\hbar^2}{4m}\frac{d^2}{dR^2} + \frac{1}{4}kr^2 + kR^2\right)u(r,R) = E^{(2)}u(r,R) \tag{16}$$

Vi kan dele bølgefunksjonen i en $r$-avhengig del og en $R$-avhengig del, slik at $u(r,R) = \psi(r)\phi(R)$. Den totale energien er da summen av den relative energien og massesenter-energien, $E^{(2)} = E_r + E_R$. Coulombkreftene som virker mellom de to elektronene er gitt som

$$V(r_1, r_2) = \frac{\beta e^2}{|\mathbf{r}_1 - \mathbf{r}_2|} = \frac{\beta e^2}{r} \tag{17}$$

hvor $\beta e^2 = 1.44$ eVnm. Setter vi dette inn i den $r$-avhengige delen av radialligningen, får vi at

$$\left(-\frac{\hbar^2}{m}\frac{d^2}{dr^2} + \frac{1}{4}kr^2 + \frac{\beta e^2}{r}\right)\psi(r) = E_r\psi(r).$$

Vi innfører nå en dimensjonsløs variabel $\rho = r/\alpha$, hvor $\alpha$ igjen har enheten lengde. Dersom vi utfører de samme trinnene som for tilfellet *uten* Coulombvekselvirkning, ender vi til slutt opp med

$$-\frac{d^2}{d\rho^2}\psi(\rho) + \frac{1}{4}\frac{mk}{\hbar^2}\alpha^4\rho^2\psi(\rho) + \frac{m\alpha\beta e^2}{\rho\hbar^2}\psi(\rho) = \frac{m\alpha^2}{\hbar^2}E_r\psi(\rho).$$

Vi definerer nå "vinkelfrekvensen"

$$\omega_r^2 = \frac{1}{4}\frac{mk}{\hbar^2}\alpha^4$$

og finner konstanten $\alpha$ ved å kreve at

$$\frac{m\alpha\beta e^2}{\hbar^2} = 1 \quad \Rightarrow \quad \alpha = \frac{\hbar^2}{m\beta e^2}$$

I tillegg definerer vi

$$\lambda = \frac{m\alpha^2}{\hbar^2}E$$

slik at vi kan skrive Schrödingers ligning som

$$-\frac{d^2}{d\rho^2}\psi(\rho) + \omega_r^2\rho^2\psi(\rho) + \frac{1}{\rho} = \lambda\psi(\rho)$$

Vi kan se på $\omega_r$ som en størrelse som sier noe om *styrken* på oscillatorpotensialet.

# 3 Løsningsmetoder

## 3.1 Jacobis metode

Jeg skal her ta for meg hovedtrekkene i Jacobis metode, som også kan finnes i kursets lærebok [3]. La oss ta for oss en $n \times n$ matrise $S$, som har den egenskapen at den utfører en ortogonal transformasjon på elementene med en gitt vinkel $\theta$

$$S = \begin{bmatrix} 1 & 0 & ... & 0 & 0 & ... & 0 & 0 \\ 0 & 1 & ... & 0 & 0 & ... & 0 & 0 \\ ... & ... & ... & ... & ... & ... & 0 & ... \\ 0 & 0 & ... & cos\theta & 0 & ... & 0 & sin\theta \\ 0 & 0 & ... & 0 & 1 & ... & 0 & 0 \\ ... & ... & ... & ... & ... & ... & 0 & ... \\ 0 & 0 & ... & 0 & 0 & ... & 1 & 0 \\ 0 & 0 & ... & -sin\theta & ... & ... & 0 & cos\theta \end{bmatrix}$$

hvor $S^T = S^{-1}$. Matriseelementene som *ikke* er lik null, er gitt ved

$$s_{kk} = s_{ll} = cos\theta, \quad s_{kl} = -s_{lk} = -sin\theta, \quad s_{ii} = -s_{ii} = 1, \quad i \neq k, \quad i \neq l$$

Dersom vi utfører en similær transformasjon på en matrise $A$, får vi

$$B = S^T A S$$

hvor $A$ og $B$ er similære matriser. Dette resulterer i ligningene

$$b_{ii} = a_{ii} \quad i \neq k, i \neq l$$
$$b_{ik} = a_{ik}cos\theta - a_{il}sin\theta \quad i \neq k, i \neq l$$
$$b_{il} = a_{il}cos\theta + a_{ik}sin\theta \quad i \neq k, i \neq l$$
$$b_{kk} = a_{kk}cos^2\theta - 2a_{kl}cos\theta sin\theta + a_{ll}sin^2\theta$$
$$b_{ll} = a_{ll}cos^2\theta + 2a_{kl}cos\theta sin\theta + a_{kk}sin^2\theta$$
$$b_{kl} = (a_{kk} - a_{ll})cos\theta sin\theta + a_{kl}(cos^2\theta - sin^2\theta)$$

hvor vinkelen $\theta$ er virklårlig bestemt. Trikset nå er å velge $\theta$ slik at alle ikke-diagonal elementer $b_{kl}$ blir null. Vi kan finne en algoritme som gjør dette for oss. Dersom vi velger en grense $\epsilon$ nær null, kan vi systematisk kan minske absoluttverdien til disse elementene (i matrisen vår $A$) gitt ved

$$off(A) = \sqrt{\sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} a_{ij}^2}$$

I en $2 \times 2$ matrise ville en slik similær transformasjon tatt seg ut på følgende måte:

$$\begin{bmatrix} b_{kk} & 0 \\ 0 & b_{ll} \end{bmatrix} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} a_{kk} & a_{kl} \\ a_{lk} & a_{ll} \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$$

hvor $c = cos\theta$ og $s = sin\theta$. Vi krever at elementene på ikke-diagonalen skal være $b_{kl} = b_{lk} = 0$, som fører til at

$$a_{kl}(c^2 - s^2) + (a_{kk} - a_{ll})cs = b_{kl} = 0$$

Dersom $a_{kl} = 0$ allerede, ser vi at $\cos\theta = 1$ og $sin\theta = 0$. Vi har tidligere vist at Frobenius-normen alltid er bevart under en ortogonal transformasjon. Definisjonen lyder som følger

$$||A||_F = \sqrt{\sum_{i=1}^{n} \sum j = 1^n |a_{ij}|^2} \tag{18}$$

Bruker vi dette på en $2 \times 2$-matrise, får vi at

$$2a_{kl}^2 + a_{kk}^2 + a_{ll}^2 = b_{kk}^2 + b_{ll}^2$$

som fører til at

$$off(B)^2 = ||B||_F^2 - \sum_{i=1}^{n} b_{ii}^2 = off(A)^2 - 2a_{kl}^2$$

siden

$$||B||_F^2 - \sum_{i=1}^{n} b_{ii}^2 = ||A||_F^2 - \sum_{i=1}^{2} a_{ii}^2 + (a_{kk}^2 + a_{ll}^2 - b_{kk}^2 - b_{ll}^2).$$

Dette gjør at ikke-diagonal elementene i $A$ vil gå mer og mer mot null etter hver similær transformasjon som utføres på $A$. Vi kan definere størrelsene

$$tan\theta = t = s/c$$

og

$$cot2\theta = \tau = \frac{a_{ll} - a_{kk}}{2a_{kl}}$$

som gir oss to verdier for $t$:

$$t^2 + 2\tau t - 1 = 0 \quad \Rightarrow t = -\tau \pm \sqrt{1 + \tau^2}$$

og at

$$c = \frac{1}{\sqrt{1 + t^2}} \quad \text{og} \quad s = tc$$

Dersom vi velger den *minste* absoluttverdien av de to løsningene vi har for $t$, sikrer vi oss at $|\theta| \leq \pi/4$, som fører til at forskjellen mellom matrisene $B$ og $A$ blir minimal, siden

$$||B - A||_F^2 = 4(1-c)\sum_{i=1, i\neq k,l}^{n}(a_{ik}^2 + a_{il}^2) + \frac{2a_{kl}^2}{c^2}.$$

Denne fremgangsmåten har blitt implementert i koden `Jacobi.cpp`, som finner egenverdiene og egenvektorene til matrisen $A$. Dette gjøres for ulike verdier av $n$, som representerer størrelsen på matrisen vår. I tillegg beregnes CPU-tiden for de ulike verdiene av $n$, og resultatene skrives til slutt til filen *Jacobi.txt*. Figur 1 viser hovedtrekkene i metoden. Siden målet vårt er å få alle ikke-diagonal elementene til å bli null, setter vi en toleranse *epsilon* $= 10^{-8}$. Funksjonen `maxoffdiag` finner absoluttverdien til det **største** ikke-diagonal elementet i matrisen A. Dersom denne verdien er større enn tolerasen vi har satt, og vi samtidig **ikke** har nådd iterasjonsgrensen vår (her satt til $10^6$ iterasjoner), anvender vi Jacobis metode på $A$. Variabelen *iterations* fungerer som en teller som holder oversikt over hvor mange iterasjoner, eller rotasjoner, som er blitt utført på $A$. Slik fortsetter while-løkken frem til elementene på ikke-diagonalen har blitt mindre enn toleransen vår, eller til iterasjonsgrensen er nådd. CPU-tiden til denne prosessen måles enkelt ved hjelp av funksjonen `clock()`, hentet fra biblioteket `time.h`.

```
//start timer
start = clock();

while (max_off > epsilon && iterations < max_number_iterations) {
    max_off = maxoffdiag(A, &k, &l, n-1);
    jacobi_method(A, R, k, l, n-1); //rotate one more time
    iterations++;
}

//stop timer
finish = clock();
CPU_time = ((double) (finish - start)/CLOCKS_PER_SEC );
```

Figure 1: Figuren viser hovedtrekkene i hvordan Jacobis metode kan implementeres i C++, og hvordan CPU-tiden til denne prosessen kan beregnes.

### 3.1.1 To elektroner som vekselvirker

Jacobis metode kan også brukes i det tilfellet hvor to elektroner vekselvirker med hverandre via Coulombkrefter. Vi skal se på hva som skjer når vi har $\omega_r = 0.01$, $\omega_r = 0.5$, $\omega_r = 1$ og $\omega_r = 5$, når vi befinner oss i grunntilstanden med $l = 0$. Vi skal her se bort fra massesenter-energien $E_R$. Fremgangsmåten for dette oppsettet er tilnærmet likt som i forrige seksjon, så lenge vi husker å endre potensialet fra $\rho^2$ til $\omega_r^2\rho^2 + 1/\rho$. Diagonalelementene i matrisen $A$ vil da være på formen $d_i = 2/h^2 + \omega_R^2\rho_i^2 + 1/\rho_i$.

### 3.2 Bruk av Armadillo

En annen metode å løse egenverdiproblemet på, er å ved hjelp av Armadillos innebygde funksjon `eig_sym`. Denne funksjonen finner enkelt egenverdiene og egenvektorene vi er ute etter. Funksjonen `eig_sym(eigenvalues, R, A)` tar inn matrisen $A$ og finner dens egenverdier og egenvektorer. Egenverdiene lagres i vektoren *eigenvalues*, mens egenvektorene lagres som kolonner i matrisen $R$. Siden vi nå har to ulike løsningsmetoder, kan vi enkelt sammenligne resultatene for å se om de stemmer overens med hverandre.

## 4 Resultater

### 4.1 Ett elektron

Jacobis metode ble implementert i programmet `Jacobi.cpp`. Dette programmet inneholder også Armadillos løsning på problemet, slik at vi enkelt kunne sammenligne resultatene fra de to løsningsmetodene. Vi kjenner allerede de tre laveste egenverdiene fra oppgaveteksten [5], nemlig 3,7,11. Tabell 1 og 2 viser resultatene fra Jacobis metode og Armadillo-metoden for ulike verdier av $n$. Her er $\rho_{max} = 5.0$.

Table 1: De tre minste egenverdiene for ulike verdier av $n$, sammen med beregningstiden og antall iterasjoner. Verdiene ble funnet ved å kjøre programmet `Jacobi.cpp`.

| **n** | **Egenverdier**$_{Jacobi}$ | **CPU [s]** | **Iterasjoner** |
|---|---|---|---|
| 10 | (2.919484, 6.583038, 9.936657) | 0.00022600 | 105 |
| 50 | (2.996871, 6.984342, 10.96193) | 0.091970 | 3 852 |
| 100 | (2.999219, 6.996094, 10.99066) | 1.0987 | 16 217 |
| 200 | (2.999805, 6.999026, 10.99781) | 17.236 | 66 065 |
| 300 | (2.999913, 6.999568, 10.99914) | 90.121 | 149 879 |
| 310 | (2.999919, 6.999596, 10.99921) | 118.81 | 160 171 |

Table 2: Tabellen viser de tre laveste egenverdiene samt beregningstiden ved bruk av Armadillo-metoden. Verdiene ble funnet ved å kjøre programmet `Jacobi.cpp`.

| **n** | **Egenverdier**$_{armadillo}$ | **CPU**$_{armadillo}[s]$ |
|---|---|---|
| 10 | (2.919484, 6.583038, 9.936657) | 8.0000E-05 |
| 50 | (2.996871, 6.984342, 10.96193) | 0.00062900 |
| 100 | (2.999219, 6.996094, 10.99066) | 0.0034070 |
| 200 | (2.999805, 6.999026, 10.99781) | 0.013732 |
| 300 | (2.999913, 6.999568, 10.99914) | 0.032447 |
| 310 | (2.999919, 6.999596, 10.99921) | 0.035778 |

## 4.2 To elektroner

Ved å gjøre en liten endring i potensialet i programmet `Jacobi.cpp`, kunne vi gå fra ett elektron til **to** elektroner som vekselvirker. Det nye programmet er kalt `jacobi_interaction.cpp`, og tabell 3 viser de tre laveste egenverdiene når vi har to elektroner som vekselvirker. Her er $n = 200$ og $\rho_{max} = 5$ konstant, mens verdien til $\omega_R$ varierer.

Table 3: I tabellen vises de numerisk beregnede egenverdiene til grunntilstanden for ulike verdier av oscillatorrekvensen $\omega_R$. Verdiene ble funnet ved hjelp av programmet `jacobi_interaction.cpp`. Her er $n = 200$ og $\rho_{max} = 5$.

| $\omega_R$ | **Egenverdi (numerisk)** |
|---|---|
| 0.01 | 0.8407035 |
| 0.5 | 2.230821 |
| 1 | 4.057058 |
| 5 | 17.42822 |

Figur 2 viser et plot *sannsynlighetsfordelingen*, dvs $|u(\rho)|^2$, til de tre laveste bølgefunksjonene $u(\rho)$ for de to elektronene, som funksjon av den relative koordinaten $r$. I dette tilfellet er $\omega_R = 0.01$, og $\rho_{max} = 5$. Her er det **ingen** Coulomb-vekselvirkning, og bølgefunksjonene er normert. Programmet `2e.cpp` bruker Armadillos funksjon `eig_sym` til å finne egenverdiene for ulike verdier av $n$, og skriver resultatene til en .txt fil. Programmet `2e_plot.py` leser inn verdiene fra .txt filen og plotter resultatene.

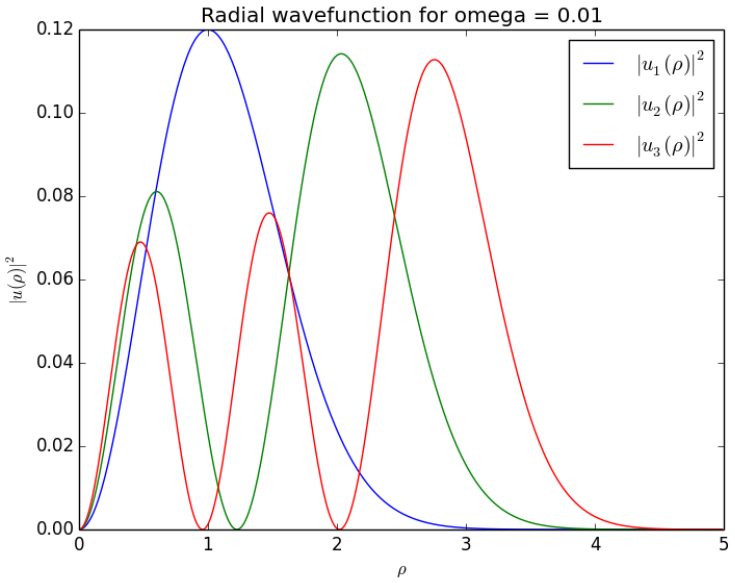


Figure 2: Plottet viser sannsynlighetsfordelingen til de to elektronene når vi ikke har noen Coulomb-vekselvirkning. $\omega_R$ er satt til 0.01, mens $\rho_{max}$ er 5. Plottet ble produsert ved hjelp av programmene `2e.cpp` og `2e_plot.py`.

Ved å gjøre en liten ending i uttrykket for potensialet i programmet `2e.cpp`, kunne vi plotte sannsynlghetsfordelingen til de tre laveste bølgefunksjonene *med* Coulomb-vekselvirkning, som vist i figur 4. Dette ble gjort for ulike verdier av $\omega_R$, nemlig 0.01, 0.5, 1 og 5. Bølgefunksjonene er normert.

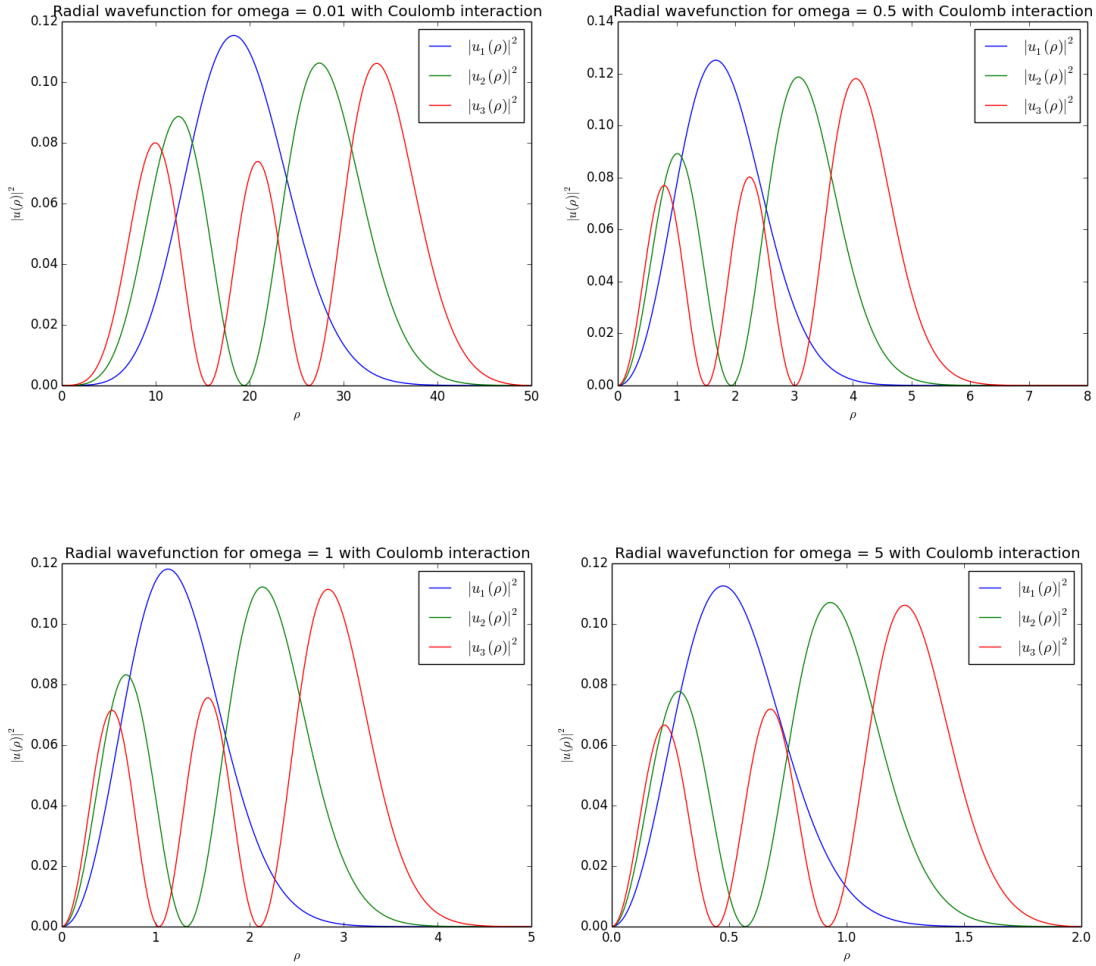


Figure 4: Sannsynlighetsfordelingen for de tre laveste bølgefunksjonene som funksjon av $\rho$. $\omega_R$ har verdien 0.01, 0.5, 1 og 5, mens $\rho_{max}$ går fra å være hhv 50, 8, 5 og 2. Plottene ble produsert ved hjelp av programmene `2e.cpp` og `2e_plot.py`.

# 5 Diskusjon

Vi ser av tabell 1 at implementasjonen av Jacobis metode ga oss resultater som stemte godt overens med de kjente egenverdiene 3, 7, 11. Likevel var dette en tidkrevende prosess, særlig for høye $n$-verdier. Når $n = 310$, var CPU-tiden på nesten 2 minutter. Til sammenligning brukte Armadillo-metoden kun 0.04

sekunder for samme $n$-verdi. De to løsningsmetodene kom frem til de samme resultatene, men på grunn av den betraktelige forskjellen i beregningstid, valgte jeg å bruke Armadillo-metoden i resten av prosjektet.

I tillegg ser vi av tabell 1 at vi trengte $n \approx 300$ grid-points for at de tre laveste egenverdiene skulle stemme med fire gjeldende siffere etter desimalen. Vi satte da $\rho_{max} = 5.0$.

Antallet iterasjoner, eller rotasjoner, som trengtes før alle ikke-diagonal elementene ble tilnærmet lik null varierte med antall grid-points $n$. I tilfellet hvor $n = 300$ trengte vi 149 879 iterasjoner. Vi kan finne en funksjon som forteller oss om antall iterasjoner som funksjon av størrelsen $n$ på matrisen. En liten regresjonsanalyse av resultatene i tabell 1 gir oss en funksjon på formen

$$iterasjoner = 1.69n^2 - 7.9n + 29.8$$

For å gjøre en matrise $A^{n\times n}$ diagonal, bruker Jacobis metode altså omkring $1.7n^2$ iterasjoner (eller rotasjoner). Vanligvis vil en slik algoritme bruke mellom $3n^2$ og $5n^2$ iterasjoner på en generell matrise uten null-elemetner. I vårt tilfelle var matrisen tridiagonal, slik at vi ikke trengte fullt så mange iterasjoner. Vi har likevel fått erfare at Jacobis metode er ganske tidkrevende i forhold til andre metoder som baserer seg på tridiagonale matriser. Dette kommer av at vi ved en rotasjon kan risikere at et tidligere null element endres til et ikke-null element. Dette vil helt klart sinke prosessen, men er altså et vanlig problem i Jacobis metode [3].

Det ble også utført noen enhetstester underveis for å sikre oss et fungerende program og at vi fikk ut riktige resultater. Disse testene kan finnes i programmet `Jacobi.cpp`. Den første testen gikk ut på å sjekke om funksjonen `maxoffdiag` returnerte de riktige elementene. Funksjonens oppgave var å finne absoluttverdien til det største ikke-diagonal elementet. Ved hjelp av en kjent 5×5-matrise så vi at funksjonen ga oss riktig svar ut. I den andre testen så vi nærmere på funksjonen `jacobi_method`, som brukte Jacobis metode til å finne egenverdiene til en gitt matrise. Vi testet funksjonen på en kjent 3×3-matrise, hvor egenverdiene allerede var kjent ved hjelp av MATLAB sin funksjon `eig()`. Det viste seg at `jacobi_method` returnerte egenverdier som lå ganske nær egenverdiene funnet i MATLAB. Testen kunne muligens blitt gjort på en større matrise for mer nøyaktige resultater.

Dersom vi sammenligner plottet av de tre laveste bølgefunksjonene med og uten vekselvirkning (figur 2 og første plot i figur 4) for en gitt $\omega_R$=0.01, ser vi at bølgefunksjonene *uten* vekselvirkning ligger mye tettere enn bølgefunksjonene *med* vekselvirkning. Hvis vi tar en kikk på plottene i figur 4 av de tre laveste bølgefunksjonene for to vekselvirkende elektroner, ser vi at jo mindre $\omega_R$ var, jo større ble $\rho_{max}$. Store deler av denne oppgaven besto i å justere $\rho_{max}$ som funksjon av $\omega_R$. Dette ble gjort ved å observere plottene av sannsynlighetsfordelingen. Parameteren $\omega_R$ angir nemlig styrken på oscillator-potensialet.

Dersom vi øker $\omega_R$, vil vi kunne se at de tre laveste tilstandene presses tettere sammen. Vi kunne også se at når vi skrudde på vekselvirkningen, lå tilstandene mer spredt og hadde derfør en større utstrekning $\rho_{max}$. Oppførselen er derfor i tråd med det vi kan forvente av et slikt fysisk system, da vekselvirkningen er en frastøtende kraft.

# 6 Konklusjon

Vi har i dette prosjektet har vi sett nærmere på hvordan vi kan finne bølgefunksjonene til hhv ett -og to elektroner i et harmonisk oscillator-potensiale. Problemstillingen kunne omformes til et egenverdi-problem, som kunne løses med Jacobis metode. I tillegg kunne vi løse samme problem ved hjelp av Armadillos innebygde funksjon `eig_sym`. Både Jacobis metode og Armadillo metoden kom frem til de samme egenverdiene. Etter å ha målt de to metodenes CPU-tid, fant vi ut at Armadillo-metoden var overlegen i forhold til Jacobis metode.

Vi har sett at vi enkelt kan gå fra et system med ett elektron til to elektroner. Dette ble gjort ved å endre den potensielle energien fra $\rho^2$ til $\omega_r^2\rho^2 + 1/\rho$. Vi fikk da se hvilken effekt dette hadde på bølgefunksjonene. Coulomb-vekselvirkningen mellom elektronene resulterte i at bølgefunksjonene strakte seg lenger ut i radiell retning, som forventet.

# 7 Vedlegg

Alle koder og resultater som er brukt i rapporten finnes på Github-adressen: https://github.com/livewj/Project2

# References


[1] Kursets offisielle Github-side *FYS3150 - Computational Physics* https://github.com/CompPhysics/ComputationalPhysics, 03.09.2016

[2] Slides fra kursets offisielle nettside "Matrices and linear algebra": http://compphysics.github.io/ComputationalPhysics/doc/web/course, 24.09.16 "How to write a scientific project, with examples": http://compphysics.github.io/ComputationalPhysics/doc/pub/projectwriting/html/projectwriting.html, 24.09.17

[3] M. Hjort-Jensen: Computational physics, lecture notes 2015. Fysisk institutt, UiO, 2016.

[4] Armadillo C++ linear algebra library: http://arma.sourceforge.net/docs.html, 01.10.16

[5] Oppgavetekst: Project 2, Fysisk institutt, UiO, 01.10.16