FYS3150 Computational Physics 2014

Oblig 2

Løysning av Schrödinger likninga for to elektron i ein tredimensjonal oscillator brønn.

Øyvind Sigmundson Schøyen

27. september 2014

Sammendrag

Skriv eit samandrag av prosjektet her oppe.
https://github.com/Schoyen/FYS3150/tree/master/Oblig2

Innhold

1 Introduksjon

I dette prosjektet er me interesserte i å løyse Schrödinger likninga for to elektron. Likninga er skrive om slik at me kan jobbe med eit ein-lekam problem istadenfor to. Me nyttar lineær algebra for å løyse differensiallikningane som eit sett med lineær likningar. Måten me gjer dette på er ved Jacobirotasjon for å finne eigenvektorar og eigenverdiar. Til slutt vil me plotte bølgjefunksjonen for grunntilstanden til elektrona ved hjelp av eigenvektorane og eigenverdiane.

2 Jacobirotasjon

For å løyse eigenverdi- og eigenvektorproblem vil me nytte Jacobirotasjon. Dette er ein algoritme som, etter ein rekke similaritetsformasjonar, vil gjere alle ikkje-diagonale matriseelement til null. Denne algoritmen er likevel ikkje ein veldig effektiv algoritme då me ved ein rotasjon kan kome i skade for å gjere eit element som tidligare var null til å bli ikkje-null. Numerisk kan det og ta lang tid før elementa vert null. Me vil difor heile tida teste verdiane mot ein toleranse.

2.1 Algoritma

Ein similaritetstransformasjon er gitt ved

$$B = S^T AS$$

kor S er ein ortogonal matrise der $SS^T = SS^{-1} = I$. Matrisa S transformerer A ein vinkel θ i planet medan S^T tek ho tilbake. Me vil då velje θ slik at alle ikkje-diagonale element vert null. Når me gjer dette numerisk må me gjere ein rekke similaritetstransformasjonar for å oppnå dette. Då har me

$$B = S_n^T \dots S_1^T A S_1 \dots S_n.$$

Kvar matrise S og S^T er identitetsmatrisa med unntak av elementa $s_{kk} = s_{ll} = \cos\theta$, $s_{kl} = -s_{lk} = -\sin\theta$ og $s_{ii} = 1$ for $i \neq k$ og $i \neq l$. Produktet $B = S^T AS$ kan då skrivast som

$$\begin{aligned}
b_{ii} &= a_{ii}, \qquad i \neq k,\ i \neq l \\
b_{ik} &= a_{ik}\cos\theta - a_{il}\sin\theta, \qquad i \neq k,\ i \neq l \\
b_{il} &= a_{il}\cos\theta + a_{ik}\sin\theta, \qquad i \neq k,\ i \neq l \\
b_{kk} &= a_{kk}\cos^2\theta - 2a_{kl}\cos\theta\sin\theta + a_{ll}\sin^2\theta \\
b_{ll} &= a_{ll}\cos^2\theta + 2a_{kl}\cos\theta\sin\theta + a_{kk}\sin^2\theta \\
b_{kl} &= (a_{kk} - a_{ll})\cos\theta\sin\theta + a_{kl}(\cos^2\theta - \sin^2\theta).
\end{aligned}$$

Me vil no velje θ slik at alle ikkje-diagonale element b_{kl} i praksis vert null. For kvar iterasjon vil me då teste om summen av alle dei ikkje-diagonale elementa er mindre enn ein toleranse ϵ. Me vil derimot ikkje gjer dette då det er ein tidkrevande prosess. Erstatninga vert då å sjå om det største elementet blant dei ikkje-diagonale elementa er mindre enn ϵ. Dette vil då vere

$$|a_{kl}| = \max_{i \neq j} |a_{ij}| .$$

Me krever at $b_{kl} = b_{lk} = 0$. Det gjer oss likninga

$$a_{kl}(c^2 - s^2) + (a_{kk} - a_{ll})cs = b_{kl} = 0. \tag{1}$$

Me definerer no

$$\tau = \frac{a_{ll} - a_{kk}}{2a_{kl}} \qquad \Rightarrow \qquad a_{ll} - a_{kk} = 2\tau a_{kl}.$$

Setter dette inn i (1) og får

$$\begin{aligned} & a_{kl}c^2 - a_{kl}s^2 - 2\tau a_{kl}cs = 0, \\ \Rightarrow \qquad & a_{kl} - a_{kl}\frac{s^2}{c^2} - 2\tau a_{kl}\frac{s}{c} = 0. \end{aligned}$$

Siden $c = \cos\theta$ og $s = \sin\theta$ vil me få

$$\begin{aligned} & a_{kl} - a_{kl}\tan^2\theta - 2\tau a_{kl}\tan\theta = 0, \\ \Rightarrow \qquad & 1 - t^2 - 2\tau t = 0, \\ \Rightarrow \qquad & t^2 + 2\tau t - 1 = 0, \end{aligned}$$

kor $t = \tan\theta$. Me vil då få

$$\begin{aligned} t &= \frac{-2\tau \pm \sqrt{4\tau^2 - 4(-1)}}{2} \\ &= -\tau \pm \sqrt{1 + \tau^2}. \end{aligned}$$

For å unngå avrungdingsfeil ved $\tau >> 0$ gonger me andregradslikninga med den konjugerte. Det vil gje oss

$$\begin{aligned} t &= \left(-\tau \pm \sqrt{1+\tau^2}\right)\left(\frac{-\tau \pm \sqrt{1+\tau^2}}{-\tau \pm \sqrt{1+\tau^2}}\right) \\ &= \frac{-\tau^2 + (1+\tau^2)}{\tau \pm \sqrt{1+\tau^2}} = \frac{1}{\tau \pm \sqrt{1+\tau^2}}, \end{aligned}$$

som gjer oss

$$t = \frac{1}{\tau \pm \sqrt{1+\tau^2}} \qquad \vee \qquad t = \frac{-1}{-\tau + \sqrt{1+\tau^2}}.$$

Me vil no velje den minste av røttene t slik at c og s henholdsvis går mot ein og null ved

$$c = \frac{1}{\sqrt{1+t^2}}, \qquad s = tc.$$

Då vil me kunne begrense vinkelen $\theta \leq \frac{\pi}{4}$ slik at differansen mellom matrisa B og A vert så liten som mogleg ved formelen

$$|B - A|_F^2 = 4 \underbrace{(1-c)}_{=0 \text{ for } c \to 1} \sum_{i=1, i \neq k,l}^{n} (a_{ik}^2 + a_{il}^2) + \frac{2a_{kl}^2}{c^2}.$$

Me vil fortsette desse operasjonane heilt til $\max(a_{ij})^2 \leq \epsilon$ for $i \neq j$.

2.2 Implementering

Algoritma vert implementert i ein klasse med tre metodar som løyser eigenverdi og eigenvektor problemet.

3 Resultat

Denne seksjonen vil bli delt opp i fire deler. Me vil diskutere metoden i seg sjølv kor me ser på køyretid og stabilitet. Me vil sjå på energinivåa til elektrona med, og uten, Coulomb interaksjon for forskjellige ω_r og $\rho_{\max}$. Eg har valt å gjere dette ved Jacobi. Dette krev at me normaliserer eigenvektorane for å få riktige verdiar. Det vil bli forklart i meir nøyaktighet. Til slutt har eg lagt ved ein feilanalyse på eigenverdiane som funksjon av n_{step}.

3.1 Jacobi

Jacobirotasjon er ein reknetung algoritme. Denne metoda bryt veldig kjapt ned når ein lager ei stor matrise. For vårt formål er difor essensielt at me uttnyttar peikar- og referansemoglegheitane i C++ for å unngå unødvendig tidbruk på lagring. Køyretida til metoden går som $\mathcal{O}(n^3)$ for rotasjon. Mykje av grunnen til at metoda er treig kjem frå det faktum at me står i fare for å gjere eit element som er null til ikkje-null ved ein rotasjon. Me merker fort når matrisa blir stor at tidbruken stig veldig kjapt. Ein samanlikning av `eig_sym` frå Armadillo og Jacobi gjer oss køyretidene i sekund i tabellen under.

	Armadillo	Jacobi
n = 10	0.000218953	6.6139e-05
n = 25	0.000627105	0.00146394
n = 50	0.00260522	0.0173505
n = 75	0.00581946	0.075161
n = 100	0.0109806	0.228011
n = 150	0.02755	1.14266
n = 200	0.0534935	3.50042
n = 250	0.0921532	8.35271