Sammanfattning Kandidatprojekt FUFX02-12-01

Configuration-Interaction Methods and Large-scale Matrix Diagonalization

Pontus Hansson
Joakim Löfgren
Karin Skoglund Keiding
Simon Vajedi

16 maj 2012

1 Introduktion

För att modellera system med många partiklar så behövs approximationer som gör det möjligt att lösa Schödingerekvationen för systemet, då ekvationen generellt inte kan lösas analytiskt för sådana system. En metod som används heter *configuration interaction*, i vilken vågfunktionen som representerar ett mångpartikel-system uttrycks som en linjärkombination av okopplade mångpartikel-bastillstånd [1]. Bastillstånden skall inneha symmetriegenskaper vad gäller utbyte av partiklar och är sammansatta av en linjärkombination av produkttillstånd. Dessa är i sin tur uppbyggda av enpartikeltillstånd, som i detta fall utvecklas i en bas bestående av harmoniska oscillator-tillstånd. Då en sådan utveckling bildar en oändlig serie behöver den trunkeras vid ett visst värde på energin, så att numeriska lösningar blir möjliga. Nu kan ekvationen skrivas på matrisform och egenenergier och egentillstånd fås genom att lösa egenvärdesproblemet för matrisen.

Att finna egenvärdena är dock fortfarande komplicerat, då matrisens dimension växer snabbt för ökande värden på trunkeringsenergin. Det finns dock en familj av metoder som bygger på Lanczos algoritm, som är väl anpassade för att reducera komplexieten hos egenvärdesproblemet för stora, glesa matriser och genom användning av en sådan, lämpligt anpassad, så kan egenvärdena lösas ut på ett effektivt sätt.

Genom att studera energispektrat för de bundna tillstånden och motsvarande egenfunktioner får forskare möjlighet att få reda på mer om system med många partiklar, som exempelvis nukleoner i en atomkärna eller en atomgas fångas i en fälla vid låg temperatur.

Målet med projektet är att efter att ha utfört undersökningar av algoritmer samt matrisernas egenskaper, ta fram rekommendationer inför en framtida implementation av en egenvärdeslösare baserad på Lanczos algoritmen.

2 No-Core Shell Model

No-Core Shell Model (NCSM) är en modell skapad för att studera mångpartikelsystem bestående av starkt växelverkande nukleoner i atomkärnor [2]. Denna kan även hantera partiklar i en extern potential-fälla, vilken modelleras med hjälp av en harmonisk oscillator-potential. Med denna inkluderad ges den totala Hamiltonianen av

$$\hat{H}_{\mathrm{tot},\Omega} = \hat{H}_{\mathrm{rel}} + \hat{H}_{\mathrm{HO}}(\Omega) = \sum_{i=1}^{N}\left[\frac{\hat{p}_i}{2m} + \frac{1}{2}m\Omega^2\hat{r}_i^2\right] + \sum_{i<j}^{N}\left[V_{i,j} - \frac{m\Omega^2}{2N}(\hat{r}_i - \hat{r}_j)^2\right] + \sum_{i<j<k}^{N} W_{i,j,k}, \tag{1}$$

där grupperingen av termerna är enligt en-, två- och trekroppars-operatorer.

Enpartikeltillstånden utvecklas i den harmoniska oscillatorbasen, vilket blir en oändlig serieutveckling som behöver trunkeras vid ett värde $N_{\max}$. För att lösa Schrödingerekvationen för systemet används därmed den trunkerade serien, och ekvationen fås på matrisform. Hamiltonianen diagonaliseras sedan lämpligtvis med en Lanczos metod.

2.1 No Core Shell Model for bosons

För arbetet har vi fått ta del av koden No Core Shell Model for bosons, NCSMb, ännu under utveckling, som är skriven av Simon Tölle vid Bonns universitet i samarbete med C. Forssén, H. T. Johansson, och L. Platter vid Chalmers Tekniska Högskola. Denna kod modellerar bosoner istället för fermioner. Koden har som inparametrar: trunkeringsenergin $N_{\max}$, antalet bosoner N_{boson}, projektionen av det magnetiska momentet M samt energier för två- respektive trekropparsfallet, E_2 och E_3. Med hjälp av dessa beräknar den matriselementen för den resulterande Hamiltonian-matrisen genom att löpa med en for-loop över alla rader i densamma. Genom användning av en hashtabell så kan kopplingar mellan mångpartikeltillstånden snabbt hittas, som ger tillägg till matriselementen.

3 Numeriska metoder

Att lösa ut egenvektorer och egenvärden, ett så kallat egenpar, för stora matriser kan vara en tidsödande sysselsättning. Konventionella egenvärdeslösare beräknar ofta alla egenpar och skapar dessutom nya matriser av samma storlek som den ursprungliga och förstör i processen eventuell gles struktur. För matriser av större dimensioner blir detta snabbt mer eller mindre omöjligt på grund av begränsningar i tid och minne, och samtidigt är det ytterst sällan som alla egenpar är intressanta.

Lanczos-algoritmen löser båda dessa problem då den istället för att förändra en matris skapar en ny men betydligt mindre matris med egenpar som väl approximerar de minsta och största egenparen för den ursprungliga matrisen. Denna algoritm bygger på att en ortonormal bas Q skapas med linjärkombinationer av så kallade Krylov-vektorer. Det resulterande rummet som spänns av denna bas kallas Krylov-rummet och den ursprungliga matrisen A kan sedan projiceras ner på detta rum till en matris T. Det visar sig att för relativt små dimensioner på detta rum, då den projicerade matrisen är betydligt mindre än den ursprungliga, erhålls fortfarande bra approximationer till de extrema egenparen vid egenvärdeslösning av denna mindre mer lätthanterliga matris. Sambandet man då får är,

$$AQ = QT \tag{2}$$

Algoritm 1 nedan beskriver tillvägagångssättet för metoden, där α_i är T-matrisens i :te diagonala element, och β_i är matrisens i :te element närmast över och under diagonalen.

I praktiken kompliceras läget av att man jobbar med ändlig precision. Då kommer små avrundningsfel att ackumuleras och på sikt förstöra ortogonaliteten mellan basvektorerna som byggs av Lanczos-algoritmen. Detta resulterar vanligtvis i felaktiga egenvärden men kan förhindras - genom att introducera någon form av reortogonaliseringsmetod i algoritmen.

Algorithm 1 Lanczos algoritm

1: $q_0 = 0$
2: $\beta_0 = 0$
3: $x_0 =$ slumpad startvektor, skild från noll
4: $q_1 = x_0 / \|x_0\|_2$
5: **for** $i = 1 \rightarrow k$ **do**
6: $\quad x_i = Aq_i$
7: $\quad \alpha_i = q_i^T x_i$
8: $\quad x_i = x_i - \beta_i q_{i-1} - \alpha_i q_i$
9: $\quad \beta_i = ||u_i||_2$
10: $\quad q_{i+1} = x_i / \beta_i$
11: **end for**

4 Undersökningar av Lanczos och Block Lanczos

Vid undersökningarna av hur framgångsrika de olika varianterna av Lanczos är så har inriktningen främst varit på matriser som är så stora att matrismultiplikationerna dominerar beräkningarna. Följaktligen inriktar sig gruppens rekommendationer kring hur denna tiden ska minimeras. Det visar sig att användandet av block av vektorer, beroende på antalet sökta egenpar, kan minska antalet matrismultiplikationer avsevärt men till en kostnad av att varje multiplikation tar lite längre tid att utföra.

Hur mycket längre är intimt förknippat med hur bra blocken samverkar med cache-linjerna, där extra fördelaktliga blockbredder är $p = 1$, 2, 4 och 8. Mätningarna visar att om ett eller två egenpar söks, så rekommenderas även blockbredderna $p = 1$ respektive $p = 2$. För upp till tio egenvärden är en blockbredd på $p = 4$ bäst och därefter fungerar blockbredd $p = 8$ lika bra som $p = 4$. Dessa resultat verkar gälla oavsett storlek på matrisen[1].

Utöver blockbredd spelar även kvalitéen på startvektorerna för Lanczos en stor roll för konvergensen. Hur approximationerna av egenvektorerna används, om en approximation finns i varje startvektor eller om de linjärkombinerades, visade sig spela mindre roll, så länge komponenter av andra vektorer hölls utanför. En startapproximation är därmed lika bra som blocket i genomsnitt, vilket specifikt skulle betyda att det är bättre med en vektor som är en god approximation, än två vektorer varav den enda är god och den andra dålig, även om en blockbredd $p = 2$ egentligen är att föredra.

Vidare har även undersökts hur approximationer till startvektorer kan skapas, såsom att minska precisionen för matrisen, gå från ett mindre modellrum eller avbryta Lanczos och starta om med vektorer som ännu inte konvergerat.

För att råda bot på ortogonalitetsförluster som uppstår på grund av ändlig precision vid datorberäkningar så har ett flertal olika stödalgoritmer studerats. Bland dessa visar sig det bästa alternativet vara en så kallad partiell reortogonalisering vilket bygger på att ortogonalitetsförluster simuleras med en differensekvation och när förlusterna överskrider en viss tillåten nivå gör programmet en reortogonalisering med hjälp av en Gram-Schmidt-metod.

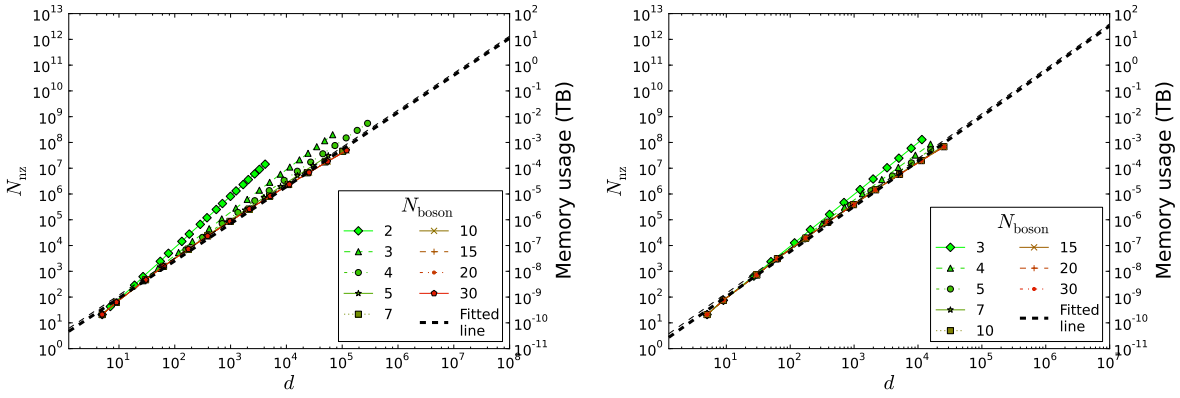
[1] Matriserna som undersöktes var beräknade från NCSMb.

5 Undersökningar av matrisegenskaper

Genom att analysera egenskaper hos matrisen kan information fås exempelvis om hur dimensionen på matrisen växer och därmed hur mycket lagringsutrymme som en matris behöver. Med hjälp av detta kan sedan Lanczos-algoritmen anpassas, vilket kommer att utgöra grunden i en framtida implementering av en egenvärdeslösare.

En undersökning av hur matrisens dimension växer med olika värden på $N_{\max}$ och N_{boson} antydde att den framförallt växer med trunkeringsenergin $N_{\max}$ och endast påverkas av N_{boson} upp till $2N_{\text{boson}} \approx N_{\max}$.

Ytterligare en intressant aspekt är att undersöka hur antalet nollskilda element i matrisen varierar med $N_{\max}$ och N_{boson}, då man baserat på detta kan förutsäga lagringsutrymmet som behövs för ett givet parameterval. Med antagandet att antalet nollskilda element växer med dimensionen som $N_{\text{nz}} = d^x$ för ett okänt $1 \leq x \leq 2$, är det tydligt i figur 1 att relationen mellan $\log_{10} N_{\text{nz}}$ och $\log_{10} d$ i princip ges av en rät linje. Med en linjär anpassning fann vi $x = 3/2$ respektive $x = 7/4$ för två- och trekropparskrafter. Från detta resultat och information om hur stort minne varje element upptar, kan vi även få fram en relation mellan lagringsutrymme och d, vilket även detta är givet i figur 1, skalat i terabyte (TB). Som ett exempel kan noteras att en matris av storlek 1 TB för tvåkropparskrafter har en dimension på ungefär $d = 10^7$, vilket är betydligt större än en dimension på $d = 10^6$ som gäller för trekropparskrafter.



(a) Tvåkropparskrafter.

(b) Två- och trekropparskrafter.

Figur 1. *Antalet nollskilda element och lagringsutrymme uttryckt i TB, för olika dimensioner d på matrisen.*

Fördelningen av de nollskilda elementen över matrisen undersöktes sedan, en aspekt som är relevant då man vill bestämma en optimerad beräkningsstrategi. Då matriserna snabbt växer med $N_{\max}$ bör implementationen vara parallell för att fördela arbetet över flera datorkärnor.

Vidare undersöktes det genomsnittliga antalet uppdateringar per matriselement under beräkning av matrisen, och både i två- och trekropparsfallet är mindre än 1 de flesta värden på N_{boson} och med ökande dimension. Detta motiverade oss till att implementera en hashtabell i NCSMb, som kräver mycket mindre storlek och minnesplats än ett fält (engelska *array*) med storleken en rad, för tillfällig lagring av elementen på en rad. Beräkningstiden minskades.

Absolutbeloppet på matriselementens värde beräknades även och sorterades enligt storleksordning, och vi fann att det fanns tre distinkta grupperingar vid 10^{-15}, 10^{-2} samt 10^{3}. Den första gruppen existerar troligtvis på grund av avrundningsfel, då numerisk imprecision har gjort att element ej kancellerat ordentligt. Elementen av storleksordning 10^{3} befinner sig på och omkring diagonalen.

En begränsning under arbetets gång var att dimensionerna på matriserna som genererades var begränsade av minne och tid, och därför har mätningarna utförts på mindre dimensioner. Dock är den generella formen på matriserna giltig även uppåt i dimension, och därmed bör resultaten generellt att vara representativa även för större matriser.

6 Rekommendationer för en framtida implementation

Sammanfattningsvis innehåller detta stycke ett antal rekommendationer som kan användas för implementeringen av en egenvärdeslösare baserad på Lanczos algoritm. Däribland finns även en rekommendation av en tillgänglig programvara som kan användas för att skapa en implementation direkt.

- Egenvärdeslösaren ska vara baserad på Block Lanczos. Om de glesa matrismultiplikationerna kostar mycket, så skall blockbredden sättas till $p = 4$ för en litet antal önskade egenvärden, <10, eller $p = 8$ om fler än 10 egenvärden är sökta.
- Partiell reortogonalisering är att rekommendera då denna minimerar antal ortogonaliseringar och antalet gånger som funktionen behöver kalla på matrisen.
- Vid brist på minne skall algoritmen utföra en omstart med ett nytt startblock av Ritzvektorer som ännu inte konvergerat, eventuellt som en linjärkombination av dem om de är fler än vad som ryms i p.
- Om enbart egenvärdena och inte egenvektorerna sökes, det vill säga energispektrumet för Hamiltonianen, så är det möjligt att använda en matris med enkel precision för att beräkna egenvektorerna med en precision av ett par decimaler, och sedan få noggrannare egenvärden genom att använda matrisen i full precision en gång.
- Approximativa egenvektorer kan beräknas från ett mindre modellrum, projiceras på ett större rum och då användas som startvektorer till en Lanczos-körning, vilket ger en något snabbare konvergens. Detta visar sig troligen mer attraktivt vid större modellrum än de här undersökta.
- Av den tillgängliga programvaran så visade sig BLZPACK vara den som bäst matchade kravspecifikationerna. Denna är baserad på en Block Lanczos rekursion, använder modifierad partiell reortogonalisering samt kan köras med parallellt med MPI.
- Vid en parallell implementation kommer kraven på fördelningen av matriselementen att bero på om matrisen delas upp i lika stora delar, eller delar med olika storlek. För att använda cachen så effektivt som möjligt så kan eventuellt matrisens rader och kolumner omorganiseras.

Referenser

[1] Sherrill, C. D., Schaefer III, H. F. (1999) *The Configuration Interaction Method: Advances in Highly Correlated Approaches.* San Diego: Academic Press. pp. 143-269. (Advances in Quantum Chemistry, vol. 34, pp. 143-269).

[2] Navrátil, P., Quaglioni, S., Stetcu, I., Barrett, B. R. (2009) Recent developments in no-core shell-model calculations. *J. Phys. G: Nucl. Part. Phys.*, vol. 36, no. 8.