UPPSALA
UNIVERSITET

PROJEKTRAPPORT

Computational Consulting

Marcus Molander, Nazmul Bhuiyan, Martin Wåger

Rapport i Teknisk-vetenskapliga datorberäkningar

Juni 2007

INSTITUTIONEN FÖR INFORMATIONSTEKNOLOGI

Sammanfattning
Projektgruppen har arbetat tillsammans med befintliga deltagare i verksamheten Computational Consulting. Tre olika delprojekt har behandlats: integrering av Brillouinzoner, programmering av Cell BE processorn samt analys av en numerisk metod för lösning av Black-Scholes ekvation. Projekten har varit utmanande och omfattande och även om inga slutgiltiga resultat presenteras så har arbetet lett till att vidare studier för övriga involverade kan fortsätta. Speciellt i delprojektet om Black-Scholes presenteras konkreta delresultat och metoder för att gå vidare.

1 Computational Consulting

1.1 Allmänt om Computational Consulting

1.1.1 Bakgrund

Computational Consulting, förkortat C2, är en tjänst som tillhandahålls av institutionen för informationsteknologi vid Uppsala universitet. Tjänsten är en sorts konsultverksamhet som ämnar vara till nytta och sprida kunskap om beräkningar. Via en webbsida [1] annonseras tjänsten och problemställarna kan kontakta C2. De problem som tas upp är inom området beräkningsvetenskap, t.ex. numerisk analys, programmering och simulering. Samtidigt ges de som arbetar med C2 en möjlighet att få erfarenhet av verkliga problem och att arbeta med dessa i projektform.

Projekten som C2 arbetar med kan komma från lokala företag, men kommer hittills från forskare på institutioner på Uppsala Universitet eller SLU. Omfattningen av ett problem som inkommer till C2 kan vara av varierande storlek. Det varierar från de minsta som kan vara en fråga som kan besvaras i ett mail, till de största som kan vara så omfattande att de får göras om till examensarbete eller en doktorandtjänst. Vidare lämnas inga garantier för att C2 kommer att ta på sig att arbeta med en uppgift, eller att den kommer att färdigställas eller lösas. Detta eftersom tjänsten är ideell och kostnadsfri för problemställaren.

1.1.2 Arbetsform

I gruppen ingår doktorander som arbetar med C2-projektet i form av en doktorandkurs. Dessutom ingår vår grupp, från kursen Teknisk-vetenskapliga datorberäkningar, i C2 där vårt syfte ursprungligen var att bistå med arbetskraft till de projekt som C2 arbetar med. Det utvecklades även till att inbegripa planering av delprojekten i större utsträckning. Hela C2-projektet koordineras av Elisabeth Larsson som är anställd vid institutionen för informationsteknologi, avdelningen för teknisk databehandling, på Uppsala Universitet.

En eller ett par gånger i månaden samlas alla deltagare i C2 som har möjlighet att närvara för ett större möte. På dessa “C2-möten” diskuteras alla pågående projekt, vilka framsteg som gjorts och problem som uppstått sedan föregående möte. Dessutom tar man upp eventuella nyinkomna projekt i gruppen. Man diskuterar också hur och vilka som ska arbeta vidare med problemen. Sedan arbetar personerna var för sig eller i grupper med att lösa de uppgifter man fått.

Under arbetets gång sker också regelbunden kommunikation mellan medlemmarna via mail och mindre sammankomster. Information inhämtas, möten med problemställarna anordnas och eventuellt blandas fler in vid behov då problem uppstår med någon fråga.

1.2 Utvärdering och reflektioner

Författarna av denna rapport har inom kursen Teknisk-vetenskapliga datorberäkningar arbetat tillsammans med C2. Att göra vårt projekt i C2-gruppen har varit lärorikt och intressant men vi har också mötts av många motgångar.

En orsak är att arbetet har krävt att många involverats, vilket har medfört att möten och arbetstillfällen måste planeras noggrant. Problemställarna i de olika delprojekten har ibland varit mycket upptagna och att planera en tid då alla är tillgängliga har ibland tagit oönskat lång tid.

En annan del som vållat problem är att vi kom in i C2 då det precis inkommit två nya projekt. Detta medförde att vi tillbringade en stor del av projekttiden för att komma igång med dessa nya projekt. Till en början hade vi fått intrycket att vi som kom till C2 till största del skulle få uppgifter som hade med programmering att göra. Men det vi har lagt ner mycket tid på är att vara ansvariga för hela projekt. Det har dock varit lärorikt att få inblick i hur mycket arbete som krävs för att anordna möten, sätta sig in i problemen och sedan ha fler uppföljningsmöten, och över huvud taget hur mycket arbete som krävs för att styra stora projekt.

Idén som helhet att ha C2 inom projektkursen tycker vi är bra, men det skulle planeras på ett annorlunda sätt. Det har inte passat så bra mot projektkursen i det nuvarande utförandet eftersom det funnits en tidsbegränsning för när projektkursen ska avslutas. Det hade varit önskvärt att delprojekten redan hade passerat sin inledningsfas när vi kom in i C2, men eftersom C2 arbetar i en konsultliknande form med problem så skulle det antagligen varit svårt att åstadkomma då man inte vet när projekt dyker upp. Vi har inte nått så många slutgiltiga resultat eller lösningar inom delprojekten som man annars skulle ha förväntat sig av ett ”traditionellt” projekt, men vi har däremot arbetat och tagit fram material för att C2 och problemställarna ska kunna arbeta vidare med sina delprojekt.

2 Projekt

Projektgruppen har arbetat med totalt tre olika projekt, varav två har pågått under hela kursen. Dessa är “Brillouinzon-integrering” och “Portning av GROMACS till Cell BE processor”. I slutet av kursen har vi dessutom arbetat med “Undersökning av en lösningsmetod för Black-Scholes ekvation”. Det första projektet har flera personer i C2-gruppen arbetat med, medan de båda sistnämnda har endast projektgruppen arbetat med.

2.1 Brillouinzon-integrering

2.1.1 Sammanfattning

Vid beräkningar av egenskaper i metaller görs integreringar över Brillouinzoner. Dessa beräkningar kräver hög noggrannhet vilket leder till onödigt stor beräkningsmängd eftersom beräkningspunkterna i dagsläget placeras i ett uniformt nät. I projektet har en inledande undersökning gjorts om beräkningspunkterna kan placeras adaptivt istället. Även om inga slutgiltiga resultat har nåtts så har ett flertal metoder uteslutits och diskussion påbörjats för andra metoder.

2.1.2 Bakgrund

Integrering över s.k. Brillouinzoner [2][3] är vanliga vid beräkningar av storheter som rör metallers egenskaper. Genom att fouriertransfomera en vågfunktion kan elektronstrukturer representeras i ett vågtalsrum. Detta rum kan indelas i periodiska och symmetriska delområden, Brillouinzoner. Tack vare dessa egenskaper kan man begränsa beräkningarna till endast en Brillouinzon. Inuti området finns en s.k. fermiyta till vilken viktiga egenskaper för metaller kan relateras. Då fermiytan ofta är invecklad och innehåller stora variationer leder detta till att man måste ha mycket hög noggrannhet vid beräkningar, se referens [4]. Den höga noggrannheten kan medföra att beräkningsmängden blir stor. Då beräkningspunkter för integreringen placeras likformigt och de variationer man vill fånga bara finns i vissa områden, medför detta att en stor del av beräkningarna blir överflödiga. Detta leder till att det kan vara

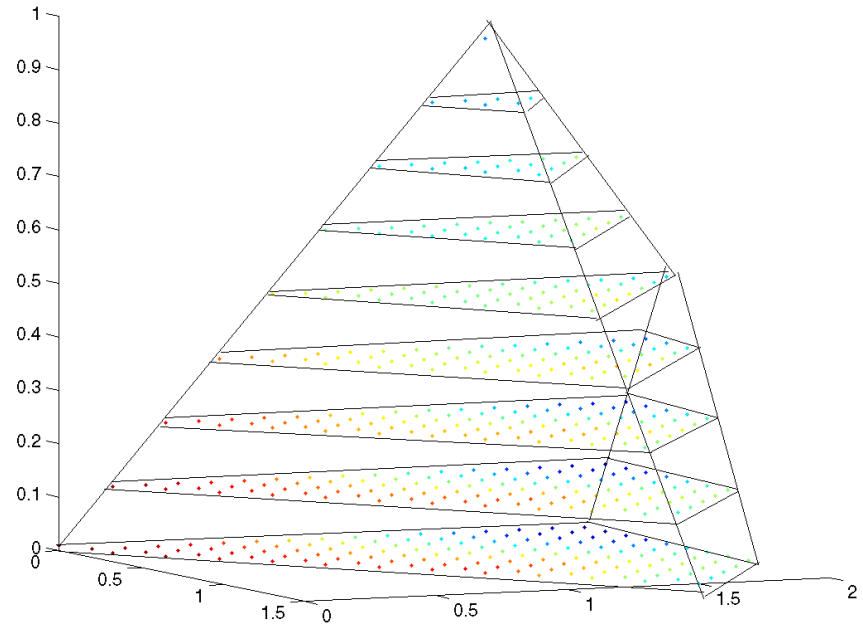
önskvärt med någon form av adaptiv metod för att välja beräkningspunkter och anpassa dessa till områden med större variation.

2.1.3 Problem

Problemställaren är Torbjörn Björkman på fysiska institutionen, teoretisk magnetism, som vill att C2 undersöker om och hur denna integration kan göras adaptiv. Efter en första genomgång framgick ett flertal problem förknippade med integrationen. Bland annat var det svårt att få en bild av arbetsområdet då den ligger i ett fouriertransformerat rum, diskretiserat i så kallade k-punkter. Dessutom ändras ytan (över vilken integreringen skulle ske) beroende på hur beräkningspunkterna placeras. Detta medför svårigheter med en eventuell första uppskattning av integrationsområdet för att sedan lägga till flera punkter i efterhand för noggrannhet.

Vår första uppgift har varit att sätta oss in i den befintliga metoden för problemet, en ej adaptiv metod, innan förslag till förbättringar kan göras. Man börjar med att dela in rummet i ett uniformt nät av k-punkter, sedan beräknar man bandstruktur och energiegenvärden. Slutligen beräknas tillståndstätheten, fördelningen av energiegenvärdena, som integreras för att kunna bestämma fermienergin i elektronstrukturen. Allt detta görs i tre steg med hjälp av ett antal program som vi fått av Torbjörn. Källkoden för dessa program är skriven i fortran77 och finns dokumenterad i referens [5]: str93.f, canon.f och ddns.f. Den förstnämnda sätter upp struktur för k-punkter samt konstanter förknippade med valt problem. Canon.f beräknar bandstrukturen samt egenvärden och egenvektorer för givna k-punkter. Den sista delen, ddns.f, beräknar tillståndstätheten utifrån bandstrukturen given av canon.f. Utöver dessa tre program finns även ett antal datafiler som programmen behöver läsa in.

Till en början var vårt mål att undersöka det sista programmet, ddns.f, och se hur slutberäkningarna av DOS och integreringen av DOS görs och eventuellt om den kunde göras annorlunda. Till vår hjälp skrev vi ett program, ftom.java, som läser in utfilen bandut.d (utfil från canon.f och infil till ddns.f) och skriver k-punkter och energiegenvärden till en fil som kan läsas av MATLAB. Dessa punkter plottades sedan i tre dimensioner, se Figur 1.



Figur 1. - K-punkter plottade i tre dimensioner, med hjälplinjer för bättre 3D-perspektiv.

Det visade sig sedan att detta antagligen inte är rätt ställe att börja, utan att vi måste undersöka en tidigare del av programkedjan om vi vill kunna göra relevanta ändringar för att göra en adaptiv metod. Förslag kom då att vi antagligen måste undersöka det första programmet, str93.f, för att finna hur k-punkter sätts upp och om dessa kan omfördelas för att ge ett icke-uniformt nät av punkter. Möjlighet finns att läsa en infil till str93.f med en uppsättning av eget valda k-punkter. Om dessa kan väljas godtyckligt eller måste vara uniformt fördelade är dock inte utrett än.

2.1.4 Diskussion

Materialet till detta projekt har varit svårt att ta till sig. Dels på grund av att koden till programmen vi fått är gammal, lång och svårförståelig. Koden verkar dock till stor del bestå av att läsa infiler och skriva utfiler vilket bidrar till dess omfattning. Dokumentationen har varit bristfällig, och matematiska formler eller pseudokod har inte funnits att tillgå. Dessutom har själva problemet varit svårt att sätta sig in i eftersom det är mycket avancerat och omfattande med flera delproblem. Dock ser det ljust ut för det fortsatta arbetet med problemet inom C2, eftersom flera metoder har uteslutits och andra har visat sig vara intressanta att undersöka vidare. T.ex. finns det en möjlighet att str93.f kan beskriva punkter i området både i det fouriertransformerade rummet och i ett normalt rum vilket kan leda till ökad förståelse och bidra till en eventuell omfördelning av hur man ska välja beräkningspunkterna.

2.2 Portning av GROMACS till Cell BE processor

2.2.1 Sammanfattning

IBM, Toshiba och Sonys nya Cell BE processor kan vara en lämplig kandidat att köra molekyldynamiksimulatorn GROMACS på. C2-gruppen har fått en förfrågan om att testa en portning gjord av en forskargrupp i USA och se om ytterligare prestandaökning går att uppnå. Vi anser att man kan få bättre resultat än det som rapporterats och att vi är på god väg att kunna mer i detalj beskriva hur. Vi hoppas på att via kontakt med artikelförfattarna komma vidare i vår analys.

2.2.2 Bakgrund

2.2.2.1 GROMACS

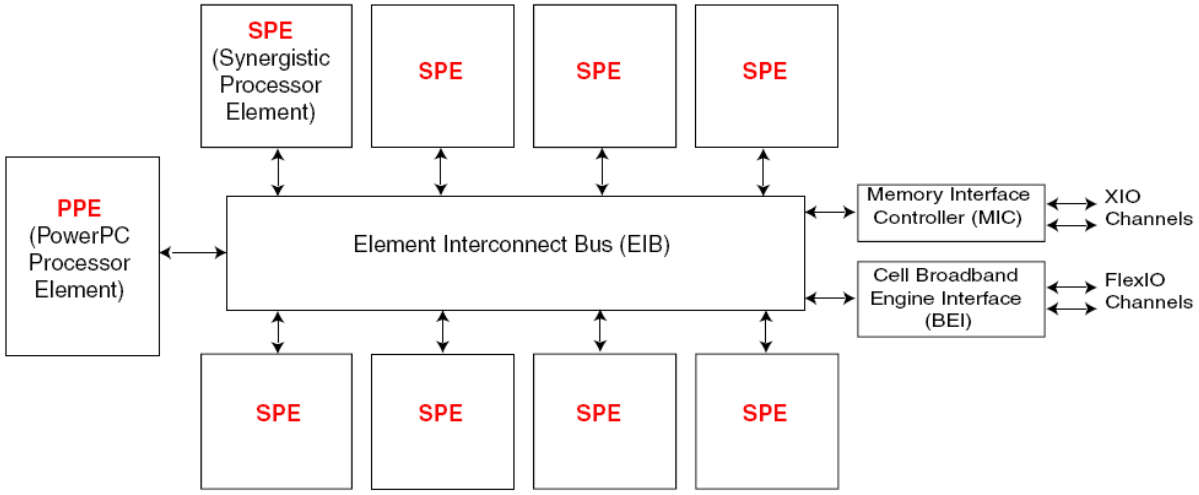
Det andra projektet fick vi från David van der Spoel vid institutionen för cell- och molekylärbiofysik på BMC. Där utvecklar man ett program för beräkning av molekyldynamik kallat GROMACS (Groningen Machine for Chemical Simulations). GROMACS är i dagsläget väldigt optimerat för att köra på en rad olika processorer bland annat Intels Xeon och AMDs Opteron. Programmet utnyttjar om möjligt 64 bitar och SIMD (Single Instruction, Multiple Data) instruktioner som finns i t.ex. SSE/3DNow! processorer. Då mycket av det som redan optimerats för GROMACS verkar passa för IBMs nya Cell BE processor skulle det kunna vara intressant att testa att köra GROMACS på en sådan. Den mest beräkningskrävande delen av programmet är delen där krafter mellan atomer beräknas, och den står för största delen av hela simuleringsberäknandet. David kontaktade C2-gruppen med en fråga om vi kunde testa att köra denna del av programmet på en Cell BE processor.

2.2.2.2 Cell BE

Cell Broadband Engine (Cell BE) är en processor från IBM, Sony och Toshiba och är framtagen för att köra applikationer som kräver mycket beräkningskraft och hög datagenomströmning t.ex. multimedia och 3D-grafik. Cell BE processorn har 9 kärnor, en av

dem är en PowerPC processor och kallas för Power Processor Element (PPE). Dess huvuduppgift är att hantera data och instruktioner åt de åtta andra. Dessa kallas Synergistic Processing Elements (SPE) och är 128 bitars RISC processorer av SIMD modell och de står för huvuddelen av beräknandet. Teoretiskt kan en Cell BE på 3.2 GHz utföra ungefär 200 GFLOPS vilket kan jämföras med en normal PC på 2-10 GFLOPS.

När man designade Cell BE ville man komma förbi tre barriärer som finns på vanliga processorer: energiåtgången, minneshanteringen och frekvensbarriären [6]. Dessa tre problem hänger alla ihop med att moderna processorer körs allt snabbare. Detta resulterar i att dagens processorer blir extremt varma och drar mycket ström. De får också svårare att mata data från RAM till processorn i den takt processorn behöver. På grund av detta måste moderna processorer använda pipelines, dvs. dela upp arbetet i flera steg och man måste därför gissa sig till hur koden kommer att se ut i framtiden. En misslyckad gissning blir kostsam för prestandan.



Figur 2. – Struktur av Cell Broadband Engine processor. Copyright IBM [6].

Cell BE undviker dessa problem genom sin speciella arkitektur. Den delar upp arbetet mellan olika processorer som optimerats för sitt arbete. Man kan då köra på lägre frekvens och samtidigt få bättre resultat. Minneshanteringen löser man genom att varje SPE har ett eget minne kallat Local Store (LS) men ingen cache som behöver laddas. För att ladda data till LS finns det hårdvara både internt i varje SPE och externt i den så kallade Element Interconnect Bus (EIB). EIB distribuerar data asynkront från RAM till LS och gör att man kan undvika väntetider. Cell BE har inte någon lång pipeline, istället kommer prestandan från stora register och specialiseringen mellan PPE och SPE. Dock medför detta att programmen som skrivs för Cell BE måste ta större ansvar för datahantering.

Till Cell BE tillhandahåller IBM tre kompilatorer. En assemblerkompilator, en C/C++ kompilator som följer GNU standard samt en egenutvecklad C/C++ kompilator. Eftersom arkitekturen kräver mer detaljerad styrning av hårdvaran än normalt har man i C/C++ kompilatorerna infört intrinsics. Detta är kommandon som ligger nära assemblerkoden men liknar C kommandon.

För att porta GROMACS har man i detta fall köpt in en Playstation 3 vilken innehåller en Cell BE processor med en PPE och sex SPEs. En PS3 kan användas till annat än spel eftersom man tillåts att installera ett eget operativsystem på den och därmed kan få tillgång att köra

godtyckliga program. Orsaken att den enbart har sex SPEs är att en av dem är reserverade för Sonys egna operativsystem och en är avstängd av tillverkningsskäl.

2.2.3 Problem

Vid vårt andra möte med David hade han just fått reda på att det redan gjorts ett försök att porta GROMACS till Cell BE. Portningen gjordes av en grupp forskare i USA. De visade i sin rapport [7] att det går att köra GROMACS ungefär två gånger fortare på en Cell BE än på en motsvarande processor utan SPE (en 3.2 GHz PowerPC processor). De uppnådde detta genom att köra GROMACS i sin helhet på PPE och sedan skriva om en del av den kod som beräknar interaktion med vattenmolekyler så den kördes på SPE. Denna del är den mest använda då de flesta simuleringar sker i vattenlösning. Detta var dock inte riktigt vad David hade väntat sig då han hade hoppats på mycket mer prestanda eftersom Cell BE processorer i dagsläget är mycket dyra. Projektet fick således ett nytt mål, att se om ytterligare förbättringar kan göras.

2.2.4 Resultat

Cell BE är en ganska annorlunda arkitektur. Den ställer större krav på programmeraren än ”vanliga” arkitekturer både då den har flera processorer och då minneshanteringen mellan dessa är annorlunda. Inlärningsprocessen blir därför också längre eftersom det är fler faktorer att tänka på. Det faktum att så få använder Cell BE idag gör att det är svårare att få hjälp och tipps även om IBM har lagt ner stor möda på att underlätta för utvecklare. Detta kombinerat med det skick den källkod vi fick att utvärdera var i gjorde att vi inte lyckades få koden att köra utan problem. Detta medförde att vi inte kunde använda de verktyg som finns för att utvärdera programmet och naturligtvis inte kunde peka på områden som kunde förbättras. Dock har vi kommit en bra bit på vägen och vårt arbete kan fortsättas av C2-gruppen.

2.2.5 Metod

Efter det att vi fått klart för oss att huvuduppgiften redan var löst och vi fått ett nytt mål, dvs. att utvärdera om ytterligare förbättringar kunde göras, bad vi David att ta kontakt med forskargruppen i USA och be om den källkod som dom hade använt sig av. Medan vi väntade på den gick vi igenom den litteratur om Cell BE processorn som finns hos IBM.

Det finns flera olika sätt att utveckla program till Cell BE. Man kan kompilera direkt i det Linux-operativsystem som körs på PPE eller på en annan dator (så kallad korskompilering). Den metod vi valde är den som rekommenderas av IBM, utvecklingsmiljön Eclipse och korskompilering med hjälp av deras Software Development Kit (SDK). Denna SDK installerar kompilatorer för både PPE och SPE samt en emulator som skapar en virtuell Cell BE miljö. Detta SDK installations-skript kräver dock Fedora Core 6 Linux.

Korskompilering har fördelen att den går snabbare än kompilering på en PS3 då den har lite arbetsminne. Eclipse är ett bra verktyg för programutveckling till Cell BE. Det har inbyggt stöd (från SDK) att exekvera kod på externa enheter via SSH och tillåter också extern felsökning i koden. Svårigheterna uppkom dock när vi fick källkoden till GROMACS portningen. Den var mycket bristfälligt kommenterad och stora delar av koden kändes experimentell (dvs. delar var bortkommenterade här och var och vissa andra delar användes ej). Dessutom lyckades vi inte att kompilera koden med den medföljande makefilen (make är ett skriptspråk för automatisk kompilering). Detta medförde att vi fick lägga ganska lång tid på att länka alla delar av källkoden så att den gick att kompilera i Eclipse.

När väl koden kördes visade det sig att den inte stannade eller gav något resultat. Felet lokaliserades till en loop som körs på SPE nr 0. Orsaken till felet antar vi vara antagligen en bugg i koden eller ett fel i de indata som kommer från GROMACS.

2.2.6 Slutsatser

Utan att kunna testa koden blir det svårt att svara på Davids fråga om den går att optimera. Det stora problemet med en Cell är minneshanteringen. Det är nödvändigt att strukturera dataflödet så att SPEerna hela tiden har något att jobba med. Flaskhalsen här är som sagt inte accesstiden, som på en ordinär processor utan hur data lagrats i RAM minnet och hur data skickas fram och tillbaka. Detta problem beskrivs också i den rapport som skrevs om GROMACS portningen, dock menade man att då GROMACS var så stort så skulle det bli en alltför stor operation att skriva om det så att en bättre datastruktur uppnåddes. Vidare i artikeln ser man att deras minneshantering är ganska enkel (t.ex. används bara 3/4 av det data man läser in) och att arbetsfördelningen mellan processorerna inte är perfekt (och om man tittar på den kod jag fått att köra är den väldigt dålig).

Vi anser att man kan få bättre resultat än det som rapporterats och att vi är på god väg att kunna mer i detalj beskriva hur. Den utvecklingsplattform vi byggt upp är ett effektivt hjälpmedel för att utveckla Cell BE kod på. Vi hoppas på att vi via vidare kontakt med artikelförfattarna kan komma vidare i vår analys.

2.3 Undersökning av en lösningsmetod för Black-Scholes ekvation

2.3.1 Sammanfattning

Gruppen har fått i uppgift att undersöka om, och i så fall hur mycket, man kan förbättra noggrannheten vid lösning av Black-Scholes ekvation genom att använda en implementation som bygger på minstakvadratmetoden [8]. Samtidigt ska den ökade minnes- och beräkningsmängden tas hänsyn till som uppkommer som följd av att man löser ett överbestämt system. En analys av noggrannhet, minne och beräkningskostnad har gjorts med hjälp av ytor plottade i tre dimensioner. Samband mellan val av parametrar för stabilt uppförande hos noggrannheten har hittats och förslag till vidare analys av noggrannhet relaterad till kostnad har tagits fram.

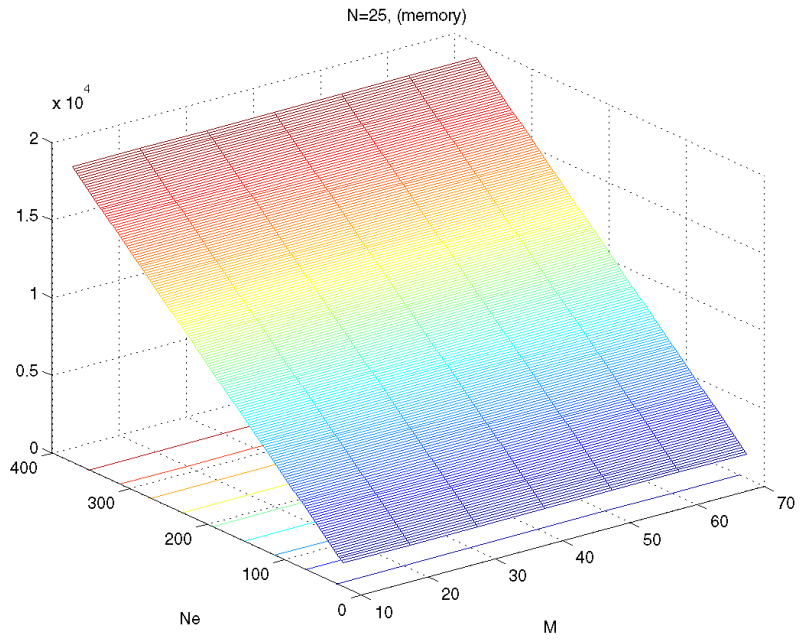
2.3.2 Bakgrund

Vid beräkning av värdet på optioner använder man t.ex. Black-Scholes ekvation. I den ursprungliga metodformuleringen löses ekvationen med lika många villkor som variabler vilket ger ett kvadratiskt ekvationssystem. Problemställaren Elisabeth Larsson undersöker nu om man kan tjäna på att istället använda ett överbestämt ekvationssystem och lösa detta med minstakvadratmetoden. Förbättringen som ska undersökas är om man kan få en högre noggrannhet i förhållande till minnesåtgång och beräkningskostnad.

2.3.3 Problem

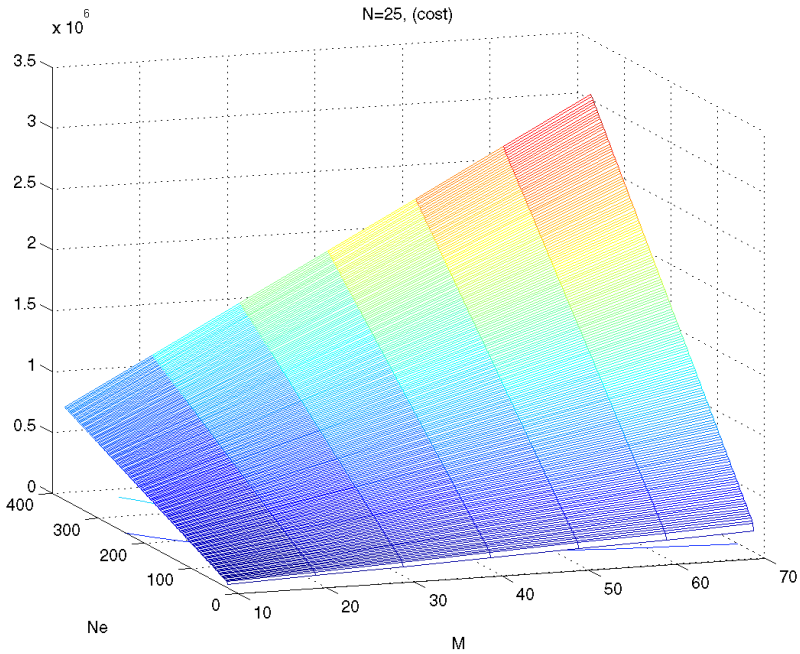
För att avgöra den nya föreslagna minstakvadratmetodens egenskaper behöver vi veta noggrannhet, minnesanvändning och beräkningskostnad för en given uppsättning med inparametrar. Felet mäts i maxnorm eller alternativt i finansiell norm. Dessa normer ges som utparametrar av MATLAB-programmet BSLSMain1D.m som Elisabeth Larsson har skrivit och givit till projektgruppen. Formler för minnesåtgång och beräkningskostnad har också givits. Minne och kostnad beräknas från M, N, Nb och Ne. M är antalet tidssteg, N antalet centrumpunkter, Nb antalet randpunkter och Ne antalet minstakvadratpunkter. För formler se appendix. Dessa ingår även som inparametrar till programmet för att bestämma felet.

Målet är att för en given noggrannhet, där felet mäts i maxnorm eller finansiell norm, jämföra vilken parameteruppsättning som är mest lönsam med avseende på minne och kostnad. För att göra detta görs först en serie körningar med olika inparametrar, där man varierar M, Ne och N. För ett givet N varieras M och Ne. Fel, minne eller kostnad kan sedan plottas som en yta mot M och Ne.



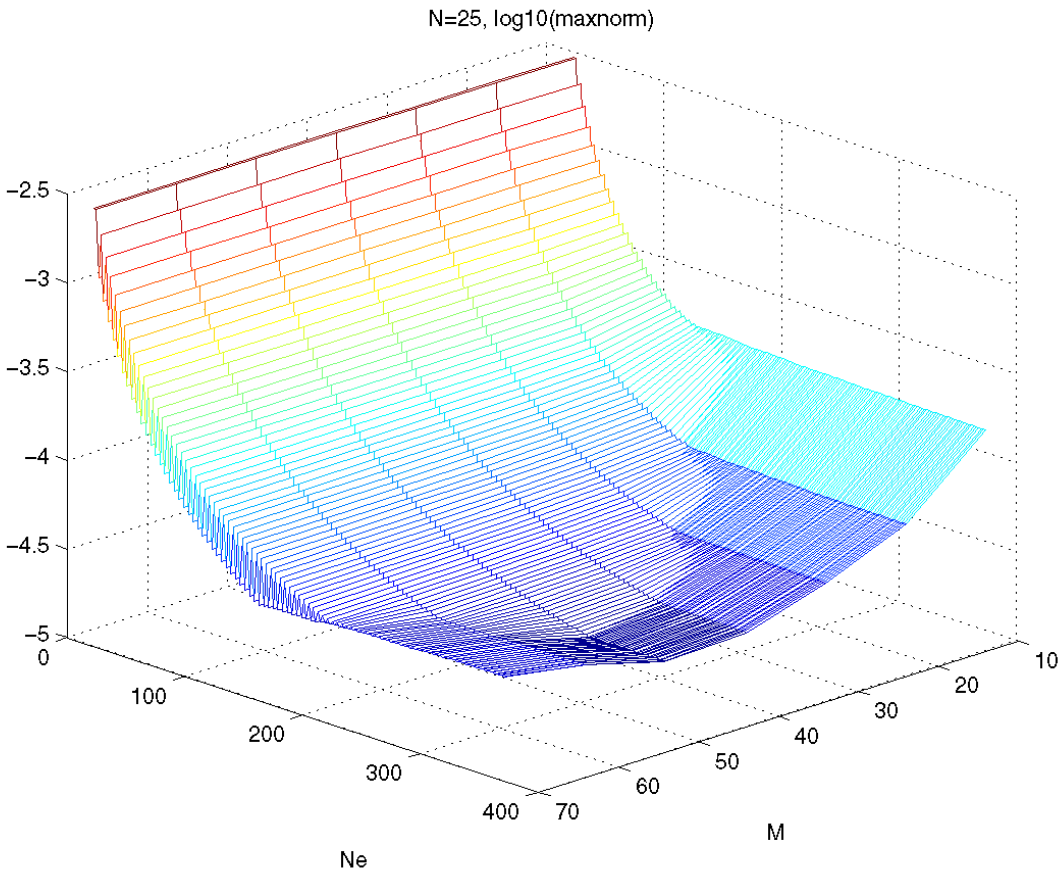
Figur 3. - Typisk minnesanvändning, alltid oberoende av M.

Ur Figur 3 ser vi att minnesanvändningen är oberoende av tidssteget M och att den ökar linjärt med Ne.



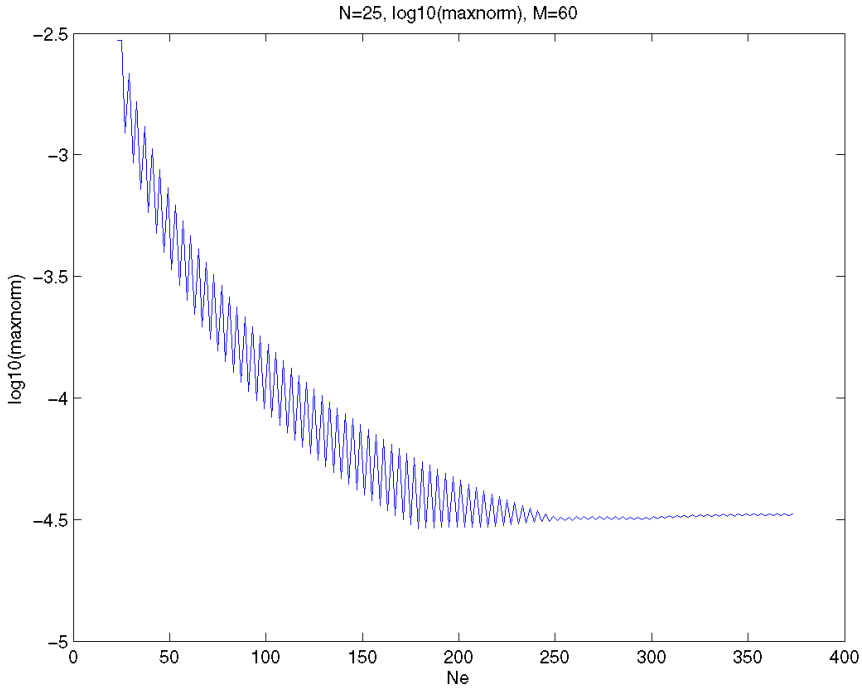
Figur 4. - Typisk beräkningskostnad med avseende på M och Ne.

Figur 4 visar att kostnaden för beräkningen ökar både med avseende på tidssteget M och antal minstakvadratpunkter Ne. Figur 3,4 och 5 visar värden för N=25 och M mellan 10 och 70 samt Ne mellan 23 och 373.



Figur 5. - Exempel på maxnorm, med avseende på M och Ne.

Figur 5 visar maxnormen logaritmerad. Det vi ser är att maxnormen, felet, minskar när vi introducerar fler minstakvadratpunkter Ne. Beroende på vilket N vi väljer fås olika lutningar och metoden konvergerar vid olika värden, men formen är i stora drag densamma. Lösningens noggrannhet påverkas dock inte speciellt mycket av tidssteget M. Endast för korta tidssteg, M mellan 10 och ungefär 60, förekommer variationer i maxnormen. För större värden på M förblir felet oförändrat, därför kan man med fördel välja ett litet M för att hålla beräkningskostnaden låg. Den finansiella normen har en liknande form.



Figur 6. - Figur 5 i genomskärning längs M=60.

Figur 6 visar figur 5 i genomskärning i planet M=60, dvs. efter att tidssteget M har nått ett stabilt värde. Den taggiga formen på kurvan uppstår pga. att beräkningspunkterna placeras likformigt inom beräkningsområdet, och således kan punkterna ibland placeras på viktiga detaljer och ibland vid sidan av dessa detaljer. Här syns att för Ne=23, dvs Ne=N-Nb vilket motsvarar ett kvadratiskt ekvationssystem utan extra minstakvadratpunkter, ligger felet i närheten av 10^-2.5 och minskar sedan för att konvergera runt Ne=250 vid ett fel som bara är ungefär 10^-4.5. Detta visar att införandet av extra minstakvadratpunkter ger en tydlig förbättring av noggrannheten, dock återstår att diskutera till vilket pris i minne och kostnad denna förbättring görs.

Då man studerar graferna för maxnormen ser man generellt att för låga Ne har man stora variationer i felet, medan dessa variationer minskar när man infört tillräckligt många extra minstakvadratpunkter. För vidare studier av hur man ska välja Ne för att vara säker på att passera det oroliga området där Ne är litet vill vi gärna ha en generell regel för hur vi ska välja Ne, gärna på formen Ne=k*N. Efter att ha studerat grafer för N mellan 10 och 35 fann vi att k bör väljas till ungefär 7, dvs. att om man väljer Ne sju gånger större än N så kan man vara säker på att inte befinna sig i det värsta av det oroliga området.

Efter att ha fått ett förslag av Elisabeth att modifiera området, göra området större, över vilka punkterna placeras så förväntades ett förbättrat resultat. Det resultat vi fick av denna modifiering var att noggrannheten minskade något (felet blev större), dock konvergerade noggrannheten snabbare från det oroliga till det stabila området. Ett k på 5 räckte här för att helt passera det oroliga området.

2.3.4 Diskussion

Något som är oturlig för detta projekt, är att det inkom i ett sent skede av projektarbetet. Att arbeta med detta projekt har däremot varit givande och intressant då given källkod har varit fullt fungerande och utförligt kommenterad. Något som problemställaren hade önskat var att

jämföra den ursprungliga implementationen, kvadratiskt ekvationssystem, med den nuvarande, överbestämt med minstakvadratmetoden, i termer av beräknings- och minneskostnad för att erhålla samma feltolerans. På den korta tid då vi arbetat med detta har vi ej hunnit utföra denna analys, men ett flertal idéer om hur man ska gå vidare har framkommit.

För det första vill man studera för vilka kombinationer av parametrar man får en given maxnorm. Dessa uppsättningar av parametrar leder i sin tur till att jämföra vilken uppsättning som är mest effektiv i fråga om minnesanvändning och beräkningskostnad. När man gjort denna analys bör man kunna säga om och i så fall hur mycket man tjänar på att göra minstakvadratmetoden. Detta ger i sin tur upphov till ett problem med hur man ska avgöra hur man ska bestämma dessa parameteruppsättningar för ett givet fel.

Huvudsakligen har vi två olika idéer om hur detta kan lösas. Antingen söker man efter det lägsta Ne-värde där man för högre Ne inte har någon maxnorm som överstigen given gräns. Detta skulle kunna göras med en iterativ metod där man börjar på det största Ne-värdet och sedan minskar Ne tills man får en maxnorm där man passerat gränsen. Det andra alternativet är att man skulle kunna införa ett polynom som beskriver en kurva, anpassat till alla punkter i maxnormen. Utifrån denna anpassade kurva skulle man sedan snabbt kunna ta fram gränsen där Ne ger den sökta maxnormen. I båda fallen förutsätter det att vi har låst tidssteget M till ett värde innan sökandet efter Ne börjar, detta för att kunna söka i en tvådimensionell kurva istället för i de tredimensionella ytorna. Med stöd av denna information skulle sedan problemställaren kunna ta ställning till om den nyare implementationen är lönsam.

3 Tack

Vi vill börja med att tacka Elisabeth Larsson vars hjälp och engagemang har varit ovärderlig för vårt arbete inom C2 och alla delprojekten. Vi vill också tacka Torbjörn Björkman för samarbetet i Brillouinzon integreringsprojektet, David van der Spoel för samarbetet inom Cell-projektet, samt alla medlemmar i C2-gruppen för den hjälp de bidragit med under arbetets gång.

4 Appendix

4.1 Brillouinzon-integrering

4.1.1 Kod

ftom.java

```
import java.io.*;
import java.util.*;
public class ftom{
    public static void main(String[] arg) throws IOException{
              //Reading bandut.d to memory
              FileReader fr = new FileReader("bandut.d");
              BufferedReader br = new BufferedReader(fr);
              double [][] values = new double[8][505];
              //505 must be changed for other problem sizes.
              int i=0,posts=0,v=0;
              String s;
              s=br.readLine();
              while(s!=null){
                  s=s.trim(); //Cut away leading/trailing spaces
                  if(s.startsWith("EIGENVALUES")){
                            //Reads coordinates
                            values[0][posts]=Double.valueOf(s.substring(38,46));
                            values[1][posts]=Double.valueOf(s.substring(48,56));
                            values[2][posts]=Double.valueOf(s.substring(58,66));
                            v=Integer.valueOf( s.substring(71,74).trim() );
                            if(v!=posts+1){
                                System.err.println("Desync!");
                                System.exit(0);
                            }
                            posts++;
                  }
                  if(s.startsWith("D")){
                            //Reads D-values
                            j=2;k=12;
                            values[3][posts-1]=Double.valueOf(s.substring(j+0*k,j+1*k));
                            values[4][posts-1]=Double.valueOf(s.substring(j+1*k,j+2*k));
                            values[5][posts-1]=Double.valueOf(s.substring(j+2*k,j+3*k));
                            values[6][posts-1]=Double.valueOf(s.substring(j+3*k,j+4*k));
                            values[7][posts-1]=Double.valueOf(s.substring(j+4*k,j+5*k));
                  }
                  s=br.readLine();
              }
              System.out.println("Posts="+posts);
              //Done reading bandut.d
              //Saving bandut.m
              FileWriter fw = new FileWriter("bandut.m");
              BufferedWriter bw = new BufferedWriter(fw);
              bw.write("coords=[");
              for(i=0;i<505;i++){
                  bw.write(values[0][i]+" "+values[1][i]+" "+values[2][i]);
                  bw.newLine();
              }
              bw.write("];");
              bw.newLine();
              bw.write("values=[");
              for(i=0;i<505;i++){
                  bw.write(values[3][i]+" "+values[4][i]+" "+values[5][i]+" "+values[6][i]+"
"+values[7][i]);
                  bw.newLine();
              }
              bw.write("];");
              bw.close();
              //Done saving bandut.m
    }//main
}
```

4.2 Undersökning av en lösningsmetod för Black-Scholes ekvation

4.2.1 Formler

Följande formler är hämtade från referens [8].

Parameter	Meaning
M	The number of timesteps
N	The total number of center points
N_b	The number of boundary points outof N
N_e	The number of least squares points

Formler för minnesanvändning

Structure	Storage
Q	$N_e(N - N_b)$
R	$(N - N_b + 1)(N - N_b)/2$ if sparse storage
$A_{b;\mu}$	N_b^2 stored as LU
$C_{e;\mu} = A_{e;\mu} - \beta_0^n B_{e;\mu}$	$N_e N_b$
$A_{b;\lambda}$	$N_b(N - N_b)$
$A_{e;\lambda}$	$N_e(N - N_b)$
$A_{e;\mu}$	$N_e N_b$
3 vectors approximately	length $N - N_b$

Formler för beräkningskostnader

Operation	Cost
LU-factorization	$2(N - N_b)^3/3$
Solve using L and U	$M\left(2(N - Nb)^2\right)$

Operation	Cost
QR-factorization	$2(N - N_b)^2(N_e - (N - N_b)/3)$
Computation of f	$2M N_e N$
Multiplication with $C_{e;\mu}$	$2M N_e N_b$
Solve of least squares system	$M\left(2N_e(N - N_b) + (N - Nb)^2\right)$
Multiplication with $A_{b\lambda}$	$2M N_b(N - N_b)$
Subtraction	$M(N - N_b)$

4.2.2 Kod

run.m

Detta MATLAB-skript används för att beräkna minnesåtgång, beräkningskostnad, maxnorm samt finansiell norm för en uppsättning parametrar. Efteråt kan run2.m användas för att plotta resultaten.

```
%clear;
close all hidden;

phi='mq';
ep=1;
M=20;
N=40;
%rg_c=[0 4];
rg_c=[-0.5 4.5]; %ny
rg_e=[0 4];
Nb=2;
Nvector=10:5:50;
Mvector=10:10:70;

%Avkommentera "for N=Nvector" för att göra flera körningar med
%olika N.
%Annars görs en körning med givet N.
%Notera att ett antal kommenteringar måste ändras för att
%automatiskt spara ner resultaten till .mat-filer

%tic;
%for N=Nvector
  NNb=N-Nb;
  Nevector=NNb:2:(15*N-Nb);
  memmatris=zeros(size(Mvector,2),size(Nevector,2));
  timmatris=zeros(size(Mvector,2),size(Nevector,2));
  maxmatris=zeros(size(Mvector,2),size(Nevector,2));
  finmatris=zeros(size(Mvector,2),size(Nevector,2));
  [k,l]=size(memmatris);
  k*l

  %tic;
  k=1;
  j=1;
  for Ne=Nevector
    memcost=Ne*NNb + (NNb+1)*NNb/2 + Nb*Nb + Ne*Nb;
    memcost=memcost+ Nb*NNb + Ne*NNb + Ne*Nb + 3*NNb;

    i=1;
    for M=Mvector
      if Ne==NNb
        timcost=2*NNb^3/3 + M*(2*NNb^2);
      else
        timcost=2*NNb^2*(Ne-NNb/3) + M*(2*Ne*NNb+NNb^2);
      end
      timcost=timcost+ 2*M*Ne*N + 2*M*Ne*Nb + 2*M*Nb*NNb + M*NNb;

      [maxnrm,finnrm]=BSLSMain1D(phi,ep,M,N,rg_c,Ne,rg_e);
      memmatris(i,j)=memcost;
      timmatris(i,j)=timcost;
      maxmatris(i,j)=maxnrm;
      finmatris(i,j)=finnrm;

      if mod(k,100)==1
        k %Skriver ut vart hundrade beräkningssteg, så man vet hur
          %långt man kommit.
      end
      i=i+1;
      k=k+1;
    end
    j=j+1;
  end
  %t=toc
  %Använd följande två rader för att automatiskt spara resultat för
  %senare plottning med run2.m
  %s=sprintf('N%dc',N);
```

```
  %save(s,'memmatris','timmatris','maxmatris','finmatris','N','Nevector','Mvector');
%end
%tid=toc
val=3;
```

run2.m

Detta MATLAB-skript plottar resultat efter run.m eller efter att man laddat tidigare körningar från .mat-filer.

```
close all hidden;
%Denna fil plottar ytor av resultaten från run.m eller efter
%att en sparad körning har laddats. Använd då t.ex. "load N25b.mat"
%före körning av run2.m

a=memmatris;
b=timmatris;
c=maxmatris;
d=finmatris;
Mlog=log10(Mvector);
Nelog=log10(Nevector);

%val=3; %Måste väljas manuellt, 3 rekommenderas för loggning av värden.

if val==1 % 1= Ingen modifiering
  meshc(Mvector,Nevector,a')
  title(sprintf('N=%d, (memory)',N));
  xlabel('M');
  ylabel('Ne');
  figure;
  meshc(Mvector,Nevector,b')
  title(sprintf('N=%d, (cost)',N));
  xlabel('M');
  ylabel('Ne');
  figure;
  mesh(Mvector,Nevector,c')
  title(sprintf('N=%d, (maxnorm)',N));
  xlabel('M');
  ylabel('Ne');
  figure;
  mesh(Mvector,Nevector,d')
  title(sprintf('N=%d, (finnorm)',N));
  xlabel('M');
  ylabel('Ne');
elseif val==2 % 2= log10 på axlarna
  meshc(Mlog,Nelog,a')
  xlabel('M');
  ylabel('Ne');
  figure;
  meshc(Mlog,Nelog,b')
  xlabel('M');
  ylabel('Ne');
  figure;
  meshc(Mlog,Nelog,c')
  xlabel('M');
  ylabel('Ne');
  figure;
  meshc(Mlog,Nelog,d')
  xlabel('M');
  ylabel('Ne');
elseif val==3 % 3= log10 på värdena
  meshc(Mvector,Nevector,log10(a'))
  title(sprintf('N=%d, log10(memory)',N));
  xlabel('M');
  ylabel('Ne');
  figure;
  meshc(Mvector,Nevector,log10(b'))
  title(sprintf('N=%d, log10(cost)',N));
  xlabel('M');
  ylabel('Ne');
  figure;
  mesh(Mvector,Nevector,log10(c'))
  title(sprintf('N=%d, log10(maxnorm)',N));
  xlabel('M');
  ylabel('Ne');
```

```
  figure;
  mesh(Mvector,Nevector,log10(d'))
  title(sprintf('N=%d, log10(finnorm)',N));
  xlabel('M');
  ylabel('Ne');
elseif val==4 % 4= log10 både på axlar och värden
  meshc(Mlog,Nelog,log10(a'))
  xlabel('M');
  ylabel('Ne');
  figure;
  meshc(Mlog,Nelog,log10(b'))
  xlabel('M');
  ylabel('Ne');
  figure;
  meshc(Mlog,Nelog,log10(c'))
  xlabel('M');
  ylabel('Ne');
  figure;
  meshc(Mlog,Nelog,log10(d'))
  xlabel('M');
  ylabel('Ne');
end


%Följande beräkningar kan användas för att normalisera matriserna
%till värden mellan 0 och 1 ifall man vill undersöka att
%addera olika normer med varandra.
%Ej färdigt, matriserna bör antagligen loggas (log10) först.

%amax=max(max(a'));
%amin=min(min(a'));
%anorm=(a-amin)./(amax-amin);
%bmax=max(max(b'));
%bmin=min(min(b'));
%bnorm=(b-bmin)./(bmax-bmin);
%cmax=max(max(c'));
%cmin=min(min(c'));
%cnorm=(c-cmin)./(cmax-cmin);
%dmax=max(max(d'));
%dmin=min(min(d'));
%dnorm=(d-dmin)./(dmax-dmin);
%meshc(Mvector,Nevector,cnorm'.*bnorm')
%figure;
%meshc(Mvector,Nevector,cnorm' +bnorm')
```

5 Referenslista

[1] *UU/IT/Computational Consulting (C2),* Uppsala: Uppsala Universitet, 2007-05-31 http://www.it.uu.se/research/tdb/c2/

[2] P. E. Blöchl, O. Jepsen, O. K. Andersen, *Improved tetrahedron method for Brillouin-zone integrations*, Physical Review B, Volume 49 Number 23, The American Physical Society, 1994.

[3] M. Methfessel, A. T. Paxton, *High-precision sampling for Brillouin-zone integration in metals*, Physical Review B, Volume 40 Number 6, The American Physical Society, 1989.

[4] T. Björkman, *Adaptive Brillouin zone integration*, UppsalaUniversitet, 2007.

[5] H. L. Skriver, *The LMTO Method*, Springer-Verlag, 1984.

[6] *Cell Broadband Engine Programming Tutorial*, Verison 2.1, IBM, 2007.

[7] S. Olivier, J. Prins, J. Derby, K. Vu, *Porting the GROMACS Molecular Dynamics Code to the Cell Processor*, 1-4244-0910-1/07/$20.00 2007 IEEE.

[8] E. Larsson, ”Artikel om minstakvadratmetoden för Black-Scholes ekvation”, Uppsala Universitet, (manuskript).