CHALMERS | GÖTEBORGS UNIVERSITET

Multilevel Monte Carlo med tillämpning på elliptiska PDE med stokastiska komponenter

Kandidatarbete inom civilingenjörsutbildningen vid Chalmers

Dimitri Sergejev
Kevin Larsson
Mario Iñiguez Ordoñez

Institutionen för matematiska vetenskaper
Chalmers tekniska högskola
Göteborgs universitet
Göteborg 2016

Multilevel Monte Carlo med tillämpning på elliptiska PDE med stokastiska komponenter

Kandidatarbete i matematik inom civilingenjörsprogrammet Teknisk fysik vid Chalmers

Kevin Larsson

Kandidatarbete i matematik inom civilingenjörsprogrammet Teknisk matematik vid Chalmers

Mario Iñiguez Ordoñez Dimitri Sergejev

Handledare: Annika Lang
Andreas Petersson
Examinatorer: Maria Roginskaya
Marina Axelson-Fisk

Institutionen för matematiska vetenskaper
Chalmers tekniska högskola
Göteborgs universitet
Göteborg 2016

Populärvetenskaplig presentation

Ett väntevärde är inom sannolikhetsläran en egenskap som är av stort intresse vid analysen av fenomen kännetecknade av slumpen. Dessa fenomen kan beskrivas inom matematiken genom modeller som tillämpas inom bland annat riskanalys. Exempelvis kan vi utifrån tidigare observationer av jordbävningar uppskatta hur ofta de kommer att förekomma framöver. Alternativt så kan vi med sådana modeller lista ut ungefär var vi kan förvänta oss sprickor längs en vattenledning orsakade av rost. Även andra mer komplicerade modeller med tillhörande slump, såsom värmeledning, kan analyseras.

Väntevärdet av en modell påverkad av slumpen ger oss en insikt i vilket resultat vi kan förvänta oss. Att uppskatta väntevärdet följer en enkel men effektiv idé; Vi uppskattar väntevärdet som medelvärdet av modellens lösningar/resultat. När vi tillämpar modellen, kanske som en ekvation, och sedan löser den med ett beräkningsprogram i datorn approximerar vi väntevärdet genom att ta medelvärdet av resultaten vi erhåller. En svårighet som kan förekomma vid tillämpningen är att det, för att få en bra uppskattning på det sökta väntevärdet, kan krävas väldigt många beräkningar. För vår tillämpade modell och beräkningen av medelvärdet/väntevärdet kan detta betyda en väldigt lång beräkningstid för att uppnå bra noggrannhet. Därför är det av intresse att hitta varianter av metodiken som används vid beräkningen så att samma noggrannhet kan uppnås med kortare beräkningstid.

En sådan metod är vad som används och utvärderas i detta projekt. Metoden som utvärderas är multilevel Monte Carlo som bygger på att dela upp storheten som uppskattas i flera delar där varje del kallas för en nivå. Metoden utvärderas i förhållande till den ovan beskrivna metoden Monte Carlo som är vanligare.

Uppdelningen i delar innebär att vi först tar fram grova uppskattningar från en modell och beräknar väntevärdet för dessa lösningar. Dessa olika delar kallar vi för nivåer där en högre nivå innebär en bättre approximation. För de första, låga, nivåerna går det snabbt att beräkna väntevärdet men i gengäld ger dem inte nödvändigtvis ett korrekt svar. Därefter görs samma process med noggrannare approximationer som ska korrigera det tidigare uppskattade värdet. Detta betyder att vi lägger till allt fler nivåer. Om det nya resultatet inte heller anses tillräckligt noggrant så upprepas förfarandet med ännu bättre approximationer i så många steg som krävs för att resultatet ska bli tillräckligt bra.

I detta projekt är modellen en så kallad partiell differentialekvation, vilka i olika former ofta dyker upp för att modellera fenomen inom naturvetenskapen. Lösningarna till denna kommer vara påverkade av slump och det är till dessa vi vill beräkna väntevärdet. Inledningsvis i projektet gås det igenom nödvändig teori som behövs för implementationen av metoden samt det som krävs för lösandet av den nämnda partiella differentialekvationen.

Vid diskussion och resultat finner vi att den utvärderade metoden, MLMC, presterar bättre ju fler nivåer vi tar med vid beräkningen av väntevärdet, vilket utifrån den teoretiska bakgrunden är att förvänta sig. För modellen som vi arbetar med i projektet är dock mängden nivåer som behövs för att få en bra approximation inte så många. MLMC har därför inte utnyttjats till sin fulla potential. Möjliga alternativ för att få ut mer av metoden diskuteras. Vi kommer fram till att MLMC inte utnyttjas fullt ut men att vi ändå lyckats visa på de fördelar den kan ge vid högre nivåer.

Sammandrag

Rapporten syftar till att påvisa effektiviteten hos metoden multilevel Monte Carlo (MLMC) som är en variant av mer klassiska Monte Carlo-metoder (MC). Detta görs genom att jämföra tidskomplexiteten i förhållande till medelkvadratfelet för estimatorer som bygger på dessa två metoder. Teorin som behövs för att implementera metoden, som bygger på väntevärdet för en stokastisk variabels linearitet, beskrivs ingående. Vidare presenteras hur såväl MLMC som en mer klassisk MC-metod använts för att skatta medelvärdet av ett stokastiskt fält som är lösningen till en given partiell differentialekvation. Resultatet visar att MLMC är att föredra då högre precision krävs för det specifika problem som metoderna implementeras på. För att tydligare åskådliggöra fördelarna med MLMC så konstateras dock att ett annat problem hade varit lämpligt att välja för implementationen.

Abstract

This report aims to show the efficency of multilevel Monte Carlo (MLMC). First, the underlying theory and the implementation of the MLMC method is presented. A comparison between MLMC and a classic Monte Carlo-method is then made by computing the time required to approximate a solution for an elliptic PDE including a stochastic component with a certain mean square error. Results show that MLMC is preferable when a high accuracy is desired. However, it is also stated that a more computationally complex problem would be preferable to clearly illustrate the advantages of MLMC.

Innehåll

Förord

Vi vill till en början tacka våra handledare Annika Lang och Andreas Petersson för deras engagemang och kunskap som har lett oss genom den matematiska djungeln.

Under projektets gång har en dagbok skrivits på veckobasis där arbetet som gjorts under den gångna veckan redogjorts för. Alla tre gruppmedlemmar har också fört individuella tidsloggar över den tid de lagt på projektet.

Alla gruppmedlemmar har i en viss utsträckning bidragit till samtliga delar av projektet. Arbetet har dock delats upp i olika områden för att effektivisera arbetet och då har olika gruppmedlemmar tagit mer ansvar över vissa områden. Kevin har fokuserat på implementationen av MLMC och har tagit ansvar för att utföra alla simuleringar som krävts för att få fram resultat under arbetets gång. Dessutom har han också ansvarat för rapportens formatering i LaTeX. Mario har ansvarat för att ta fram och undersöka teorin bakom lösandet av PDE:n och den nödvändiga teorin bakom det stokastiska fältet. Dimitri har ansvarat för att undersöka teorin och implementationen av Monte Carlo-metoden och även arbetat med en implementation av en finit differens-metod för lösningen av PDE:n vilken vi valde att inte ta med i rapporten på grund av tidsbrist.

Nedan redogörs huvudförfattarna för de olika avsnitten. Trots att det är någon som har huvudansvaret för ett avsnitt så har de andra bidragit med revideringar och idéer. Om ett underavsnitt inte finns med i denna sammanfattning antas författaren av huvudavsnittet skrivit även detta.

Populärvetenskaplig presentation - Mario
Sammandrag/Abstract - Kevin/Dimitri
1 Dimitri
2
- 2.1 Dimitri
- 2.2 Kevin
- 2.3 Mario
- 2.4 Mario

3
- 3.1 Dimitri/Kevin
- 3.2 Dimitri/Kevin
- 3.3 Dimitri
- 3.4 Kevin
- 3.5 Kevin
- 3.6 Dimitri

4 Kevin
- 4.1 Dimitri

5 Samtliga
6 Samtliga
A Samtliga
B Enligt specifikation i filer

1 Inledning

Fluiddynamik, derivatinstrument och forskning inom artificiell intelligens är tre till synes orelaterade områden. En gemensam faktor för dessa områden är dock att de kan modelleras och studeras med hjälp av den klass av beräkningsmetoder som kallas *Monte Carlo* (MC)[1]. Monte Carlo-metoder är en klass av beräkningsalgoritmer som använder sig av upprepad simulering av ett stokastiskt fenomen för att sedan kunna dra slutsatser om vissa egenskaper, såsom väntevärde.

Trots att MC-metoder kan användas för att lösa en stor uppsättning av matematiska problem undkommer de inte sina begränsningar. Den största begränsningen anses vara tidskomplexiteten, då tiden för att simulera ett stort antal scenarion ofta kan bli väldigt lång[2, s. 1-2]. För att tackla denna problematik har olika varianter av standard MC-metoden utvecklats som är menade att minska tidskomplexiteten utan att offra noggrannheten i lösningen. Ett vanligt exempel på en sådan teknik är ”control variates”[3]. En relativt ny variant på MC-metod kallas för *multilevel Monte Carlo* (MLMC). Denna har visat sig vara lovande i en rad olika tillämpningar, så som vid uppskattning av väntevärdet av lösningen till stokastiska partiella differentialekvationer[2][3].

1.1 Syfte

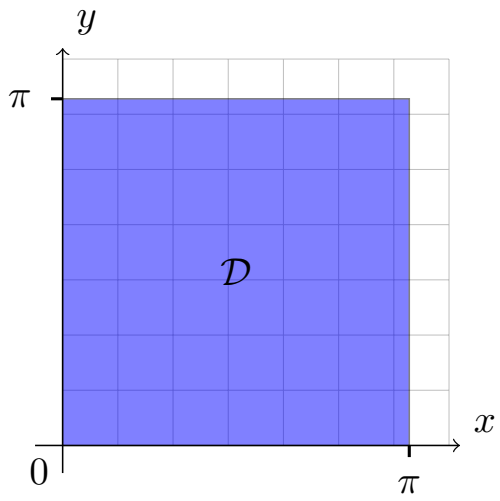
Syftet med detta arbete är att jämföra tidskomplexiteten och noggrannheten hos en klassisk Monte Carlo-metod (MC) och den relativt nya multilevel Monte Carlo-metoden (MLMC).

1.2 Problemformulering

Arbetet begränsas till att tillämpa MC och MLMC för att uppskatta väntevärdet av lösningar till den elliptiska PDE:n, även kallad *Poissons ekvation,*

$$\begin{aligned} &\Delta u(\mathbf{x}) = f(\mathbf{x}), \quad \mathbf{x} \in \mathcal{D}, \\ &u(\mathbf{x})|_{\mathbf{x}\in\partial\mathcal{D}} = 0, \end{aligned} \tag{1.1}$$

där högerledet f är ett givet stokastiskt fält. Definitionsområdet $\mathcal{D} = [0, \pi] \times [0, \pi]$ visas i Figur 1.1. Randvillkoren och fältet väljs sådana att lösningen av PDE:n underlättas. På så sätt kan fokuset falla på MC- och MLMC-implementationen och de relevanta egenskaper projektet syftar till att undersöka.



Figur 1.1: Definitionsområdet $\mathcal{D}$ på vilket lösningen till (1.1) betraktas.

1.3 Översikt av genomförande

Nedan listas de olika stegen i projektet.

1. En analytisk lösning till (1.1) härleds med hjälp av en spektralmetod.
2. En skattning av väntevärdet av lösningarna till (1.1) utförs med en Monte Carlo-metod.
3. En skattning av väntevärdet av lösningarna genomförs också med en multilevel Monte Carlo-metod.
4. MC och MLMC-implementationerna jämförs genom att betrakta tiderna som algoritmerna tar för att skatta en lösning med ett visst fel.

2 Teori

Här redovisas den teori som krävs för att genomföra de beräkningar som gjorts under projektets gång. Detta inkluderar teori för skattning av väntevärde med MLMC och MC samt teori för lösandet av Poissons ekvation i två dimensioner m.h.a. spektral dekomposition. En uppsättning grundläggande definitioner som används i framställningen finns presenterade i Appendix A.

2.1 Monte Carlo

Som nämndes i introduktionen är Monte Carlo-metoder en klass av beräkningsalgoritmer som bland annat kan användas för att uppskatta väntevärdet $\mathbb{E}[P]$ av ett stokastiskt fält P. MC-estimatorn som tillämpas[3, s. 260] genomför denna uppskattning numeriskt genom att för en given punkt $\mathbf{x}$ beräkna medelvärdet av sekvensen $\{P^{(i)}(\mathbf{x})\}_{i=1}^{N}$ där $P^{(i)}(\mathbf{x})$ $\forall i \in \{1,\ldots,N\}$ och $P(\mathbf{x})$ är stokastiska variabler som är oberoende och identiskt distribuerade.

Definition 2.1 (Monte Carlo-estimator)**.** *Givet ett stokastiskt fält P har vi för en sekvens $\{P^{(i)}(\mathbf{x})\}_{i=1}^{N}, \mathbf{x} \in \mathcal{D}$, där $P^{(i)}(\mathbf{x})$ är oberoende och identiskt distribuerade, Monte Carlo-estimatorn av väntevärdet som*

$$\hat{\mathbb{E}}[P(\mathbf{x})] \equiv \frac{1}{N}\sum_{i=1}^{N} P^{(i)}(\mathbf{x}).$$

Enligt de stora talens lag[4, s. 193] konvergerar estimatorn nästan säkert mot $\mathbb{E}[P(\mathbf{x})]$ då $N \to \infty$.

Teorem 2.1 (De stora talens lag)**.** *Antag att $\{X^{(i)}\}_{i=1}^{N}$ är en sekvens av oberoende, identiskt distribuerade slumpvariabler med ändligt väntevärde $\mathbb{E}[X^{(i)}] = \mu$.*

Då konvergerar medelvärdet av dessa nästan säkert mot μ, det vill säga

$$\frac{1}{N}\sum_{i=1}^{N} X^{(i)} \to \mu,$$

då $N \to \infty$.

När felet i estimatorerna betraktas använder vi oss av *medelkvadratfelet* (MSE), se Definition A.14. För MC-estimatorn givet enligt Definition 2.1 vill vi härav undersöka

$$\mathrm{MSE}\left[\hat{\mathbb{E}}[P(\mathbf{x})]\right] = \mathbb{E}\left[\left(\frac{1}{N}\sum_{i=1}^{N} P^{(i)}(\mathbf{x}) - \mathbb{E}[P(\mathbf{x})]\right)^2\right],$$

där $P^{(i)}(\mathbf{x})$ är oberoende och identiskt distribuerade slumpvariabler. För att analysera detta fel tillämpas den centrala gränsvärdessatsen[4, s. 194].

Teorem 2.2 (Centrala gränsvärdessatsen)**.** *Antag att $\{X^{(i)}\}_{i=1}^{N}$ är en sekvens av oberoende, identiskt distribuerade stokastiska variabler med $\mathbb{E}[X^{(i)}] = \mu$ och ändlig varians $\mathrm{Var}[X^{(i)}] = \sigma^2$. Då har vi att*

$$\sqrt{N}\left[\left(\frac{1}{N}\sum_{i=1}^{N} X^{(i)}\right) - \mu\right] \xrightarrow{D} Z \sim \mathcal{N}(0,\sigma^2)$$

då $N \to \infty$, d.v.s. att den stokastiska variabeln som utgör vänsterledet konvergerar distributionsmässigt mot $Z \sim \mathcal{N}(0,\sigma^2)$ när $N \to \infty$.

En direkt konsekvens av satsen är att distributionen av $\hat{\mathbb{E}}[P(\mathbf{x})]$ för tillräckligt stora N kan approximeras med $\mathcal{N}\left(\mu, \frac{\sigma^2}{N}\right)$ där $\mu = \mathbb{E}[P(\mathbf{x})]$.

Under antagandet att estimatorn inte har någon *bias*, se Definition A.14, gäller det att

$$\text{MSE}\left[\hat{\mathbb{E}}[P(\mathbf{x})]\right] = \text{Var}\left[\hat{\mathbb{E}}[P(\mathbf{x})]\right],$$

vilket innebär att MSE för ett tillräckligt stort antal sampels kan approximeras som $\frac{\sigma^2}{N}$.

2.2 Multilevel Monte Carlo

I detta avsnitt presenteras teori för MLMC-metoden, främst utifrån beskrivningen av metoden i [3].

Låt $\{P_l\}_{l=0}^{L}$ approximera det stokastiska fältet P, där större l innebär en bättre men mer beräkningstung approximation av fältet P. Låt särskilt $P_L \to P$ då $L \to \infty$ gälla. Genom att tillämpa linearitetenav väntevärdet kan vi skriva

$$\mathbb{E}[P_L] = \mathbb{E}[P_0] + \sum_{l=1}^{L} \mathbb{E}[P_l - P_{l-1}],$$

där $\mathbb{E}[P_L]$ i sin tur är en uppskattning till $\mathbb{E}[P]$. Under projektet benämner vi alla begrepp som rör termen $P_l - P_{l-1}, l > 0$, som det aktuella begreppet på nivå l. Exempelvis är begreppen "väntevärdet på nivå l" och $\mathbb{E}[P_l - P_{l-1}]$ ekvivalenta. Då en punktvis tolkning av fältet betraktas i projektet har vi, för varje givet $\mathbf{x} \in \mathcal{D}$, att väntevärdet uppfyller

$$\mathbb{E}[P_L(\mathbf{x})] = \mathbb{E}[P_0(\mathbf{x})] + \sum_{l=1}^{L} \mathbb{E}[P_l(\mathbf{x}) - P_{l-1}(\mathbf{x})]. \tag{2.1}$$

Tillämpa MC-estimatorn i Definition 2.1 separat på varje term i summan (2.1), för att uppskatta väntevärdet av varje nivå. Härav kan vi definiera en MLMC-estimator.

Definition 2.2 (MLMC-estimator)**.** *Låt $\{P_l(\mathbf{x})\}_{l=0}^{L}$, $\mathbf{x} \in \mathcal{D}$, approximera $P(\mathbf{x})$. Då $\{P^{(i,l)}(\mathbf{x})\}_{l=0}^{L}$ är oberoende och identiskt distribuerade definierar vi MLMC-estimatorn $\hat{E}[P_L(\mathbf{x})]$ som*

$$\hat{\mathbb{E}}[P_L(\mathbf{x})] \equiv \frac{1}{N_0}\sum_{i=1}^{N_0} P_0^{(i,0)}(\mathbf{x}) + \sum_{l=1}^{L}\left(\sum_{i=1}^{N_l}\frac{1}{N_l}\left[P_l^{(i,l)}(\mathbf{x}) - P_{l-1}^{(i,l)}(\mathbf{x})\right]\right).$$

MLMC-estimatorn utnyttjar uppdelningen för att undvika att direkt uppskatta värdet av $\mathbb{E}[P_L(\mathbf{x})]$ genom simuleringar av $P_L(\mathbf{x})$ då dessa är väldigt beräkningstunga. I detta fall kan vi utnyttja att de termer som har en lägre beräkningskostnad vid simuleringen, men eventuellt en större varians, kan samplas fler gånger. Att vi kan sampla de lägre nivåerna fler gånger är möjligt eftersom termerna kan samplas oberoende av varandra[3, s. 261].

Beteckna med C_l beräkningskostnaden för ett sampel på nivå l, d.v.s. tiden det tar att beräkna ett sampel av $P_l(\mathbf{x}) - P_{l-1}(\mathbf{x})$. På samma sätt betecknar vi med V_l variansen för sampels på nivå l, d.v.s.

$$V_l(\mathbf{x}) = \text{Var}[P_l(\mathbf{x}) - P_{l-1}(\mathbf{x})].$$

Man kan visa[3, s. 262] att antalet sampels N_l på nivå l optimalt väljs till

$$N_l = \gamma\sqrt{\frac{V_l}{C_l}}, \tag{2.2}$$

där γ är en konstant.

2.3 Lösning av PDE genom spektral dekomposition

Den *spektralmetod* som nyttjas här använder sig av dekomposition av ekvationens komponenter i termer av den betraktade operatorns egenfunktioner.

Definition 2.3 (Egenfunktioner & egenvärden tillhörande en operator)**.** *För en operator A är λ ett egenvärde till A om det finns en nollskild funktion ϕ så att $A\phi = \lambda\phi$. Om detta villkor är uppfyllt kallas ϕ för egenfunktion till A.*

Framöver, när en lösning till Poissons ekvation betraktas arbetas endast med funktioner $\psi \in L^2(\mathcal{D})$. Här är rummet $L^2(\mathcal{D})$ definierat som mängden av alla reellvärda kvadratiskt integrerbara funktioner $\psi(\mathbf{x})$ på $\mathcal{D}$, d.v.s. alla funktioner $\psi(\mathbf{x})$ definierade på $\mathcal{D}$ så att

$$\int_{\mathcal{D}} |\psi(\mathbf{x})|^2 \, \mathrm{d}\mathbf{x} < \infty,$$

se [5, s. 81].

För $L^2(\mathcal{D})$, betrakta mängden av funktioner

$$\{\phi_{n,m} \in L^2(\mathcal{D}) : n, m \in \mathbb{N}\}.$$

Om dessa utgör egenfunktioner till Laplaceoperatorn har vi enligt Definition 2.3

$$\Delta(\phi_{n,m}) = \lambda_{n,m}\phi_{n,m},$$

där $\lambda_{n,m} \in \mathbb{R}$ är de tillhörande egenvärdena. Om mängden $\{\phi_i \in L^2(\mathcal{D}) : i \in \mathbb{N}\}$ utgör en komplett bas i $L^2(\mathcal{D})$ då finns det $\forall u \in L^2(\mathcal{D})$ en unik uppsättning reella konstanter C_i, $i \in \mathbb{N}$ så att

$$u(\mathbf{x}) = \sum_{i=1}^{\infty} C_i \phi_i(\mathbf{x}),$$

se [5, s. 84]. För det undersökta problemet kan vi då ansätta en lösning $u(\mathbf{x})$ som

$$u(\mathbf{x}) = \sum_{n=1}^{\infty} \sum_{m=1}^{\infty} C_{n,m} \, \phi_{n,m}(\mathbf{x}). \tag{2.3}$$

Låter vi nu Laplaceoperatorn verka på u fås

$$\Delta u(\mathbf{x}) = \sum_{n=1}^{\infty} \sum_{m=1}^{\infty} \lambda_{n,m} C_{n,m} \, \phi_{n,m}(\mathbf{x}).$$

Om högerledet f kan utvecklas i samma bas enligt

$$f(\mathbf{x}) = \sum_{n=1}^{\infty} \sum_{m=1}^{\infty} \xi_{n,m} \phi_{n,m}(\mathbf{x}).$$

Då kan Poissons ekvation (1.1) skrivas om som

$$\sum_{n=1}^{\infty} \sum_{m=1}^{\infty} \lambda_{n,m} C_{n,m} \, \phi_{n,m}(\mathbf{x}) = \sum_{n=1}^{\infty} \sum_{m=1}^{\infty} \xi_{n,m} \phi_{n,m}(\mathbf{x}).$$

Här noterar vi att eftersom $\{\phi_{n,m}\}$ utgör en bas så är konstanterna i båda utvecklingarna unika och därför identiska för varje term i respektive summa. Det måste alltså gälla att

$$\lambda_{n,m} C_{n,m} = \xi_{n,m}.$$

Ur detta kan vi lösa ut

$$C_{n,m} = \frac{\xi_{n,m}}{\lambda_{n,m}},$$

vilket vi i sin tur kan sätta in i serieutvecklingen (2.3) av $u(\mathbf{x})$ och därmed erhålla en lösning till Poissons ekvation enligt

$$u(\mathbf{x}) = \sum_{n=1}^{\infty} \sum_{m=1}^{\infty} \frac{\xi_{n,m}}{\lambda_{n,m}} \, \phi_{n,m}(\mathbf{x}). \tag{2.4}$$

2.4 Lösning av Poissons ekvation med homogena Dirichlet-randvilkor

Vi väljer att betrakta det stokastiska fältet f i Poissons ekvation (1.1) definierat som

$$f(x,y) = \sum_{n,m=1}^{\infty} X_{n,m} \sin(nx)\sin(my) \quad \forall (x,y) \in \mathcal{D}, \tag{2.5}$$

där $X_{n,m} \sim \mathcal{N}\left(0, k^2(n^2+m^2)^{-\alpha/2}\right)$ och $k, \alpha < \infty$ är reella parametrar.

Den valda mängden av funktioner

$$\{\sin(nx)\sin(my) : n,m \in \mathbb{N}\} \tag{2.6}$$

att arbeta i valdes då de utgör egenfunktioner till Laplaceoperatorn över området $\mathcal{D} = [0,\pi] \times [0,\pi]$ [6, s. 164]. Detta kan ses genom att applicera operatorn på funktionerna

$$\Delta(\sin(nx)\sin(my)) = -(n^2+m^2)\sin(nx)\sin(my) = \lambda_{n,m}\sin(nx)\sin(my),$$

där $\lambda_{n,m} = -(n^2+m^2)$ är de tillhörande egenvärdena till egenfunktionerna i mängden (2.6). Att mängden utgör en bas i vårt område kan visas med följande resultat.

Teorem 2.3. *Antag att $\{\phi_n\}_{n=1}^{\infty}$ är en ortogonal bas för $L^2(a,b)$ och $\{\psi_n\}_{n=1}^{\infty}$ är en ortogonal bas till $L^2(c,d)$ och skriv*

$$\chi_{n,m}(x,y) = \phi_n(x)\psi_m(y).$$

Då är $\{\chi_{n,m}\}_{n,m=1}^{\infty}$ en ortogonal bas för $L^2(\mathcal{D})$, där

$$\mathcal{D} = [a,b] \times [c,d] = \{(x,y) : a \le x \le b, c \le y \le d\}.$$

Se [5, Teorem 4.1] för bevis. Givet att $\{\sin(nx)\}_{n=1}^{\infty}$ utgör en bas i $L^2(0,\pi)$, se [5, Teorem 3.5], utgör mängden (2.6) en bas i $L^2(\mathcal{D})$ enligt Teorem 2.3. Härav kan vi arbeta med fältet enligt teorin som redogjordes i Avsnitt 2.3 om spektral dekomposition. Vidare kan vi också notera att funktionerna satisfierar randvillkoret hos det betraktade problemet. Detta betyder att när man hittar en lösning enligt Avsnitt 2.3 kommer även lösningen att satisfiera randvilkoren.

För existensen av en svag lösning[7, Definition 2.40] enligt (2.4) krävs det att varje realisation av det kopplade stokastiska fältet $f(x,y)$ enligt (2.5) i Poissons ekvation är tillräckligt glatt[7, Teorem 2.42]. För att uppfylla att $f(x,y)$ är glatt väljer vi lämpliga värden på α i variansen av koefficienterna $X_{n,m}$ så att konvergensen är tillräckligt snabb.

2.4.1 Stokastisk konvergens

Konvergensen av högerledet analyseras för varje punkt och vi utgår från Definition A.12 av nästan säker konvergens. Följande lemma[8, Lemma 8] tillämpas för att bestämma vilka villkor på parametern α som ger nästan säker konvergens av summan (2.5) som definierar $f(x,y)$.

Lemma 2.4. *Låt $\{Z_n\}_{n=1}^{\infty}$ vara en sekvens av centrerade stokastiska variabler och definiera*

$$S_N = \sum_{n=1}^{N} Z_n.$$

Sekvensen $\{S_N\}_{N=1}^{\infty}$ konvergerar då nästan säkert mot S om

$$\sum_{n=1}^{\infty} \mathbb{E}\left[Z_n^2\right] < \infty.$$

För att visa att $f(x,y)$ uppfyller detta ordnar vi $(n,m) \in \mathbb{N}^2$ och definierar mängden

$$T = \{\tau_1 = (1,1), \tau_2 = (1,2), ..., \tau_m = (1,m), \tau_{m+1} = (2,1), ..., \tau_{2m} = (2,m), ...\}.$$

Låt nu för ett givet $(x,y) \in \mathbb{R}^2$

$$S_\tau(x,y) = X_{n,m} \sin(nx)\sin(my), \quad \tau \in T.$$

Givet att $X_{n,m} \sim \mathcal{N}\left(0, k^2(m^2+n^2)^{-\alpha/2}\right)$ har vi

$$\begin{aligned}\mathbb{E}[S_\tau] &= \mathbb{E}[X_{n,m}\sin(nx)\sin(my)] \\ &= \mathbb{E}[X_{n,m}]\sin(nx)\sin(my) \\ &= 0,\end{aligned}$$

och $f(x,y)$ är alltså en centrerad stokastisk variabel. Enligt Lemma 2.4 konvergerar sekvensen $\{S_\tau\}_{\tau=1}^{\infty}$ nästan säkert om

$$\sum_{\tau=1}^{\infty} \mathbb{E}\left[S_\tau^2\right] < \infty. \tag{2.7}$$

Lägg märke till att $\mathrm{Var}[S_\tau] = \mathbb{E}[S_\tau^2]$ ty $\mathbb{E}[S_\tau] = 0$. Härav noterar vi att

$$\begin{aligned}\mathrm{Var}[S_\tau] &= \mathrm{Var}[X_{n,m}\sin(nx)\sin(my)] \\ &= \mathrm{Var}[X_{n,m}]\sin^2(nx)\sin^2(my) \\ &= k^2\frac{\sin^2(nx)\sin^2(my)}{(m^2+n^2)^{\alpha/2}}.\end{aligned}$$

Sätt $p = \alpha/2$. Konvergensanalysen av summan (2.7) övergår till att analysera konvergensen av

$$\sum_{k=1}^{\infty}\mathrm{Var}[S_\tau] = \sum_{n,m=1}^{\infty} k^2\frac{\sin^2(nx)\sin^2(my)}{(n^2+m^2)^p}.$$

Tillämpa att de trigonometriska funktionerna är begränsade enligt

$$\sin^2(nx)\sin^2(my) \leq 1 \quad \forall n,m \in \mathbb{N}, \forall (x,y) \in \mathbb{R}^2,$$

varför vi kan uppskatta

$$\mathrm{Var}[S_\tau] \leq \frac{k^2}{(n^2+m^2)^p}.$$

Eftersom $n,m \in \mathbb{N}$ och $p > 0$ gäller

$$\begin{aligned}&(n+m)^p \leq (n^2+m^2)^p \\ \implies &\frac{k^2}{(n^2+m^2)^p} \leq \frac{k^2}{(n+m)^p} \\ \implies &\sum_{n,m=1}^{\infty}\frac{k^2}{(n^2+m^2)^p} \leq \sum_{n,m=1}^{\infty}\frac{k^2}{(n+m)^p}.\end{aligned}$$

Då k^2 är en positiv konstant som är oberoende av m och n, förenklas analysen till att uppskatta konvergensen av dubbelsumman

$$\sum_{n,m=1}^{\infty}\frac{1}{(n+m)^p}. \tag{2.8}$$

Notera att $k = 0$ ger ett trivialfall som inte betraktas vid konvergensanalysen. Definiera $\hat{a}_{n,m} \equiv \frac{1}{(n+m)^p}$. Givet att alla termer i dubbelsumman är positiva kan vi definiera diagonalserien tillhörande (2.8) som

$$\sum_{j=1}^{\infty} c_j,$$

där varje term c_j ges av

$$c_j = \sum_{i=1}^{j} \hat{a}_{j-i+1,i} = \sum_{i=1}^{j} \frac{1}{(j+1)^p} = \frac{j}{(j+1)^p}.$$

Enligt proposition[9, Proposition 7.16] gäller

$$\sum_{n,m=1}^{\infty} \frac{1}{(m+n)^p} = \sum_{j=1}^{\infty} c_j,$$

vilket ger oss att (2.8) kan skrivas som

$$\begin{aligned}\sum_{j=1}^{\infty} c_j = \sum_{j=1}^{\infty} \frac{j}{(j+1)^p} = \{q = j+1\} = \sum_{q=2}^{\infty} \frac{q-1}{q^p} \\ = \sum_{q=2}^{\infty} \frac{q}{q^p} - \sum_{q=2}^{\infty} \frac{1}{q^p} = \sum_{q=2}^{\infty} \frac{1}{q^{p-1}} - \sum_{q=2}^{\infty} \frac{1}{q^p}\end{aligned}$$

där serierna $\sum_{q=2}^{\infty} \frac{1}{q^{p-1}}$ och $\sum_{q=2}^{\infty} \frac{1}{q^p}$ är jämförelseserier som konvergerar för $p > 2$ respektive $p > 1$. Härav konvergerar (2.8) för $p > 2$ och eftersom $p = \alpha/2$ så har vi att den ursprungliga serien (2.7) konvergerar för $\alpha > 4$. Detta implicerar att det stokastiska fältet f definierat i (2.5) konvergerar nästan säkert för $\alpha > 4$, se Definition A.12.

3 Genomförande

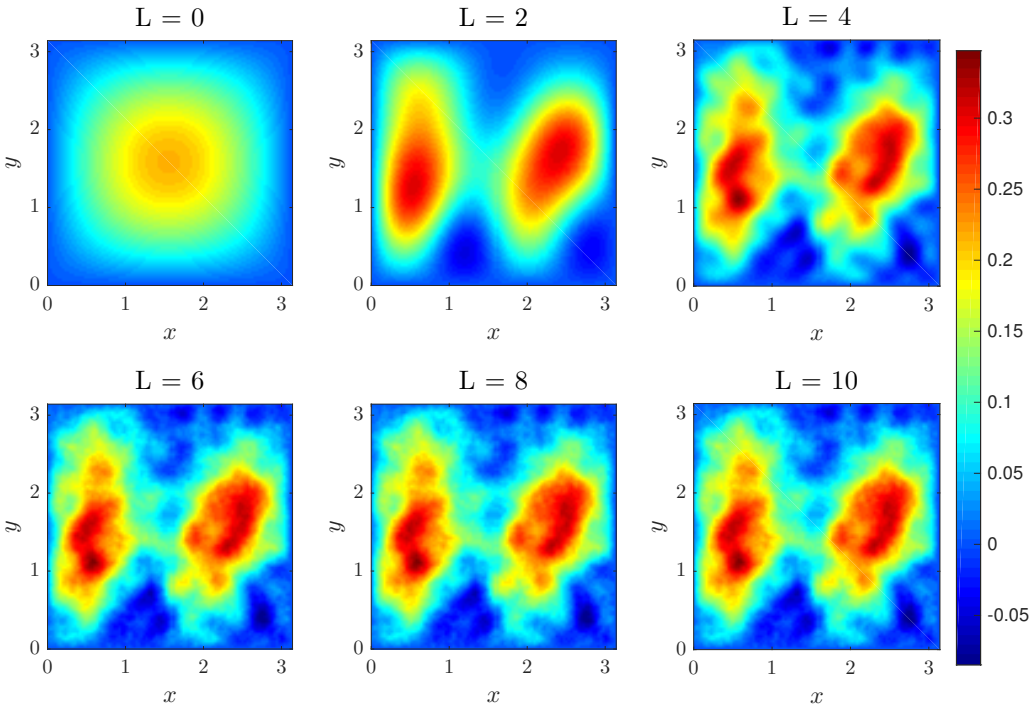
För att jämföra MLMC med vanlig MC löstes Poissons ekvation (1.1) inklusive randvillkor genom spektral dekomposition enligt Avsnitt 2.3 upprepade gånger och väntevärdet av lösningarna u skattades med de båda metoderna. Som högerled användes det stokastiska fältet f definierat i (2.5). Samtliga simuleringar under projektet gjordes i MATLAB där ett urval av kod finns bifogat i Appendix B.

3.1 Lösning av Poissons ekvation med trunkerade komponenter

Simuleringar av det stokastiska fältet f gjordes genom trunkeringar av summan som representerar fältet (2.5). Betrakta uppskattningen f_L med trunkeringen 2^L definierad enligt

$$f_L(x,y) = \sum_{n=1}^{2^L}\sum_{m=1}^{2^L} X_{n,m}\sin(nx)\sin(my), \quad X_{n,m} \sim \mathcal{N}\left(0, k^2(n^2+m^2)^{-\alpha/2}\right). \tag{3.1}$$

En visualisering av detta fält kan ses i Figur 3.1 nedan.



Figur 3.1: Illustration av ett sampel av det stokastiska fältet $f_L(x,y)$ definierat i (3.1) vid olika trunkeringar 2^L av summan. Distributionsparametrarna är $k = 1, \alpha = 4.5$.

Tillhörande f_L ansattes en approximation u_L av lösningen u till Poissons ekvation (1.1). Denna approximation $u_L(x,y)$ är då en lösning till ekvationen

$$\Delta u_L(x,y) = f_L(x,y), \quad (x,y) \in \mathcal{D}. \tag{3.2}$$

Genom tillämpning av spektral dekomposition enligt Avsnitt 2.3 kunde således finnas att

$$u_L(x,y) = \sum_{n=1}^{2^L}\sum_{m=1}^{2^L} -\frac{X_{n,m}}{n^2+m^2}\sin(nx)\sin(my). \tag{3.3}$$

3.2 Numerisk lösning av Poissons ekvation

För att lösa Poissons ekvation med trunkerade summor (3.2) numeriskt diskretiserades definitionsmängden $\mathcal{D} = [0, \pi] \times [0, \pi]$. I implementationen representerades varje realisation av vårt fält som ett rutnät av diskreta punkter. Dessa valdes som ekvidistanta i x- och y-led och antalet punkter S längs varje axel angavs som en parameter till våra funktioner. Detta gav ett rutnät med sampelpunkter (x_{s_x}, y_{s_y}) som har index $(s_x, s_y) \in \{1, 2, \ldots, S\} \times \{1, 2, \ldots, S\}$ definierade av

$$\begin{aligned} x_{s_x} &= \frac{s_x - 1}{S - 1}\pi, \\ y_{s_y} &= \frac{s_y - 1}{S - 1}\pi. \end{aligned} \tag{3.4}$$

Där x_{s_x} respektive y_{s_y} är ekvidistanta och $x_1 = y_1 = 0$ samt $x_S = y_S = \pi$. På detta rutnät fann vi sedan en approximation av u_L genom att numeriskt beräkna summan (3.3) som representerar fältet för varje punkt (x_{s_x}, y_{s_y}).

Praktiskt gjordes detta i följande steg:

Algoritm 3.1 Lösning av Poissons ekvation med spektral dekomposition

1: Diskretisera definitionsmängden $\mathcal{D}$ i ett rutnät med $S \times S$ punkter (dessa representerar de olika punkterna (x_{s_x}, y_{s_y}))
2: Beräkna $N \times N$ matrisen $\texttt{nm}(n, m) = n^2 + m^2$ och lagra den
3: Beräkna $S \times N$ matrisen `sinvals` med värden $\texttt{sinvals}(s_x,n) = \sin(n x_{s_x})$
4: Simulera de stokastiska variablerna $X_{n,m} \sim \mathcal{N}\left(0, k^2(n^2 + m^2)^{-\alpha/2}\right)$
5: Beräkna $S \times S$ matrisen `C` som $\texttt{C}(n,m) = -\frac{X_{n,m}}{(n^2+m^2)}$
6: Lösningen beräknas som $u_L(x, y) = \texttt{sinvals} \cdot \texttt{C} \cdot \texttt{sinvals}^T$

I MATLAB integrerades sedan denna lösning i de olika funktioner som implementerar de MC-/MLMC-algoritmer som användes varför ingen specifik referens till kod kan lämnas i detta avsnitt.

3.3 Implementation av Monte Carlo-algoritmen

Monte Carlo-estimatorn enligt Definition 2.1 användes för att uppskatta $\mathbb{E}[u(x, y)]$ med en given trunkering av summan vid 2^L. Tillämpningen av MC på uppskattningen $u_L(x, y)$ gav en ny estimator vilken vi betecknar som $\hat{\mathbb{E}}_L[u(x, y)]$ och som definieras enligt

$$\hat{\mathbb{E}}_L[u(x, y)] \equiv \hat{\mathbb{E}}[u_L(x, y)] = \frac{1}{N} \sum_{i=1}^{N} u_L^{(i)}(x, y). \tag{3.5}$$

Denna beräknades i varje punkt (x_{s_x}, y_{s_y}) i rutnätet. Enligt teorin i Avsnitt 2.1 konvergerar uppskattningen till $\mathbb{E}[u_L(x, y)]$ i alla sampelpunkter (x_{s_x}, y_{s_y}) då $N \to \infty$. Algoritmen som tillämpades för detta syfte är som följer:

Algoritm 3.2 Monte Carlo-implementation

1: Bestäm antal iterationer `N`
2: **for** $i = 1$ to N **do**
3: Lös (1.1) med spektral dekomposition enligt Algoritm 3.1
4: Lagra lösningen $u^{(i)}(x, y)$ för iterationen
5: **end for**
6: Beräkna MC-estimatorn som $\texttt{sum}(u^{(i)}(x, y))/N$

Denna algoritm motsvaras i programkod av MATLAB-funktionerna `MC` och `MC_sample` som presenteras i Appendix B.1 där den senare implementeras av den första.

3.4 Implementation av MLMC-algoritmen

För implementationen av MLMC för skattning av $\mathbb{E}[u(x,y)]$ skapades en MLMC-estimator enligt Definition 2.2. För att göra detta delades den trunkerade lösningen u_L (3.3) upp i flera nivåer l och för dessa beräknades variansen V_l och tidskostnaden C_l enligt Avsnitt 2.2. Detta för att sedan beräkna antalet punkter N_l för varje nivå l enligt den givna formeln i (2.2). Med detta kunde sedan MLMC-estimatorn beräknas som

$$\hat{\mathbb{E}}_L[u(x,y)] = \sum_{i=1}^{N_0} u_0(x,y) + \sum_{l=1}^{L}\sum_{i=1}^{N_l} \Big(u_l(x,y) - u_{l-1}(x,y)\Big). \tag{3.6}$$

3.4.1 Uppdelning av fält i MLMC-nivåer

Vi delade upp $u_L(x,y)$, se (3.3), i flera nivåer $u_0(x,y), u_1(x,y) - u_0(x,y), \ldots, u_L(x,y) - u_{L-1}(x,y)$.

För att underlätta notationen definierades

$$S_l \equiv \left\{(n,m) \in \mathbb{N}^2 : 0 < n \leq 2^l, 2^{l-1} < m \leq 2^l\right\} \cup \left\{(n,m) \in \mathbb{N}^2 : 2^{l-1} < n \leq 2^l, 0 < m \leq 2^l\right\}.$$

Med detta ficks att

$$f_l(x,y) - f_{l-1}(x,y) = \sum_{(n,m)\in S_l} X_{n,m} \sin(nx)\sin(my)$$

och således

$$u_l(x,y) - u_{l-1}(x,y) = \sum_{(n,m)\in S_l} -\frac{X_{n,m}}{n^2+m^2} \sin(nx)\sin(my). \tag{3.7}$$

3.4.2 Variansberäkning för MLMC-nivåer

Några egenskaper av variansen nyttjades för att beräkna $V_l(x,y) = \mathrm{Var}[u_l(x,y) - u_{l-1}(x,y)]$. Detta gjordes i en slumpvald punkt $(x_{s_x}, y_{s_y}) \in \mathcal{D}$ som låg på rutnätet enligt (3.4) vilket då också blev den punkt som MLMC-algoritmen optimerades utefter.

För två oberoende stokastiska variabler X_1, X_2 har vi

$$\mathrm{Var}[X_1 + X_2] = \mathbb{E}\left[(X_1 + X_2 - \mathbb{E}[X_1] - \mathbb{E}[X_2])^2\right].$$

Om $\mathbb{E}[X_1] = \mathbb{E}[X_2] = 0$ fås

$$\begin{aligned}
\mathrm{Var}\left[X_1 + X_2\right] &= \mathbb{E}\left[(X_1 + X_2)^2\right] \\
&= \mathbb{E}\left[X_1^2 + X_2^2 + X_1X_2\right] \\
&= \mathbb{E}\left[X_1^2\right] + \mathbb{E}\left[X_2^2\right] + \mathbb{E}[X_1]\,\mathbb{E}[X_2] \\
&= \mathbb{E}\left[X_1^2\right] - \left(\mathbb{E}[X_1]\right)^2 + \mathbb{E}\left[X_2^2\right] - \left(\mathbb{E}[X_2]\right)^2 \\
&= \mathrm{Var}[X_1] + \mathrm{Var}[X_2].
\end{aligned}$$

Då det för u_l gäller att de stokastiska variablerna $X_{n,m}$ är oberoende och att $\mathbb{E}[X_{n,m}] = 0$

så utnyttjades detta för att härleda

$$\begin{aligned} V_l(x,y) &= \mathrm{Var}\left[u_l(x,y) - u_{l-1}(x,y)\right] \\ &= \mathrm{Var}\left[\sum_{(n,m)\in S_l} -\frac{X_{n,m}}{m^2+n^2}\sin(nx)\sin(my)\right] \\ &= \sum_{(n,m)\in S_l} \mathrm{Var}\left[-\frac{X_{n,m}}{m^2+n^2}\sin(nx)\sin(my)\right] \\ &= \sum_{(n,m)\in S_l} \frac{\sin^2(nx)\sin^2(my)}{(m^2+n^2)^2}\mathrm{Var}[X_{n,m}] \\ &= \sum_{(n,m)\in S_l} \frac{\sin^2(nx)\sin^2(my)}{(m^2+n^2)^{2+\alpha/2}}. \end{aligned}$$

Med detta uttryck räknades sedan variansen ut explicit vilket i programkod representerades av MATLAB-funktionen `levelvar` som presenteras i Appendix B.2.1.

3.4.3 Beräkning av tidskostnad för MLMC-nivåer

Tidskostnaden C_l för att beräkna ett sampel av (3.7) som behövdes i (2.2) för att beräkna N_l togs reda på genom att testa hur lång tid det tog för programmen att beräkna ett stort antal sampels av (3.7). Detta gjordes genom att testa hur många sampels som kunde beräknas under en given tidsperiod och ta den genomsnittliga beräkningstiden. Algoritmen för detta är som följer:

Algoritm 3.3 Beräkning av tidskostnad C_l

1: Starta tid
2: `samples` $= 0$
3: **while** tid $<$ tidsgräns **do**
4: Beräkna ett sampel av $u_l(x,y) - u_{l-1}(x,y)$
5: `samples` $=$ `samples` $+ 1$
6: **end while**
7: $C_l = \frac{\text{Förfluten tid}}{\texttt{samples}}$

Denna implementerades i MATLAB genom funktionen `timecost` som presenteras i Appendix B.2.1.

3.4.4 MLMC-algoritm

Med kännedom om variansen $V_l(x,y)$ och tidskostnaden C_l implementerades en MLMC-algoritm för skattning av $\mathbb{E}[u(x,y)]$. Som parameter användes ett ungefärligt totalt antal iterationer

$$\hat{N}_{\text{tot}} \approx \sum_{l=0}^{L} N_l.$$

Dessa fördelades mellan nivåerna genom att låta N_l vara $\hat{N}_l$ avrundat till närmsta heltal där $\hat{N}_l$ definierades av

$$\hat{N}_l \equiv \hat{N}_{\text{tot}} \frac{\sqrt{\frac{V_l}{C_l}}}{\sum_{l=0}^{L}\sqrt{\frac{V_l}{C_l}}}.$$

Denna fördelning gjordes på grund av att det optimala antalet iterationer enligt (2.2) skalar med l som $\sqrt{\frac{V_l}{C_l}}$. Med detta implementerades en MLMC-algoritm.

Algoritm 3.4 MLMC-implementation

1: Bestäm ungefärligt totalt iterationer $\hat{N}_{\text{tot}}$
2: Sätt den sammanlagda estimatorn $\mathbb{E}_L[u(x,y)] = 0$
3: **for** $l = 0$ till L **do**
4: Beräkna N_l sampels av $u_l(x,y) - u_{l-1}(x,y)$ och ta medelvärdet
5: Addera medelvärdet till $\mathbb{E}_L[u(x,y)]$
6: **end for**

De huvudsakliga MATLAB-funktionerna som implementerades för detta är `MLMC` och `MLMC_level` vilka presenteras i Appendix B.2. För att beräkna fördelningen av sampels mellan nivåer användes också `lfrac` i Appendix B.2.1.

3.5 MSE-jämförelse mellan MC och MLMC

För jämförelsen av hur effektivt MC respektive MLMC var för att skatta $\mathbb{E}[u(x,y)]$ till lösningen $u(x,y)$ av Poissons ekvation (1.1) med högerledet $f(x,y)$ enligt (2.5) betraktades vilket MSE, se Definition A.14, som uppnåddes i förhållande till tidskomplexiteten.

MSE betraktades i samma givna punkt (x_{s_x}, y_{s_y}) på rutnätet som användes till att beräkna variansen. Detta gjordes med givna distributionsparametrar k, α och ett fixt antal sampels S längs varje axel. Dessutom sattes en nivå L som gav trunkeringsnivån 2^L för MC-metoden och maximal trunkeringsnivå 2^L för MLMC. Med detta kunde sedan MSE beräknas med MC respektive MLMC och tiden för beräkningen mätas, vilket gjordes enligt Algoritm 3.2 respektive 3.4. Algoritmerna för detta presenteras nedanför.

Algoritm 3.5 Beräkna MSE för MC

1: **for** `pwr` = 1 to `maxpwr` **do**
2: Starta tidsmätning
3: Beräkna `samples` stycken MC-estimatorer med $N = 2^{pwr}$
4: Stoppa tidsmätning och dela med `samples` för att få den genomsnittliga tiden per estimatorberäkning
5: Beräkna MSE för medelfältet och jämför med uppmätt tidsåtgång
6: **end for**

Algoritm 3.6 Beräkna MSE för MLMC

1: Beräkna $\hat{N}_l / \hat{N}_{\text{tot}}$
2: **for** `pwr` = 1 to `maxpwr` **do**
3: Starta tidsmätning
4: Beräkna `samples` stycken MLMC-estimatorer med $\hat{N}_{\text{tot}} = 2^{pwr}$
5: Stoppa tidsmätning och dela med `samples` för att få den genomsnittliga tiden per estimatorberäkning
6: Beräkna MSE för estimatorn och jämför med uppmätt tidsåtgång
7: **end for**

Dessa algoritmer implementerades i MATLAB genom funktionerna `MSE_MC` respektive `MSE_MLMC` vilka presenteras i Appendix B.1 respektive B.2.

3.6 Konvergenstakt av MC/MLMC

För att jämföra MC- och MLMC-algoritmen utfördes ett antal simuleringar där nivån L varierades. Utifrån detta ficks en körningstid T för en viss högsta nivå L och ett MSE δ förknippat med L.

Det var då av intresse att finna en funktion $g(\delta) = T$, ty konvergenstakten av funktionen med avseende på MSE kunde analyseras med hjälp av denna. Uppsättningar $(\delta_i, T_i)_{i=1}^N$ av beräknade värden och en modellfunktion $g(\delta; \beta)$ användes där β är en parametervektor som anpassades så att de uppmätta värdena stämde så väl överens med modellfunktionen som möjligt. Under antagandet att $g(\delta; \beta) = a\delta^b, \beta = [a, b]$ var det möjligt att istället betrakta ett linjärt problem ty

$$T = g(\delta; \beta) \implies \ln T = \ln(g(\delta; \beta)) = b \ln \delta + \ln a.$$

För att finna det β som passade simuleringsresultaten bäst i minsta kvadrat-mening var det alltså av intresse att minimera summan

$$S = \sum_{i=1}^{N} r_i^2, \tag{3.8}$$

där $r_i = \ln T_i - \ln g(\delta_i; \beta)$.

För att göra detta i praktiken användes MATLABs inbyggda implementation av minsta kvadratmetoden.

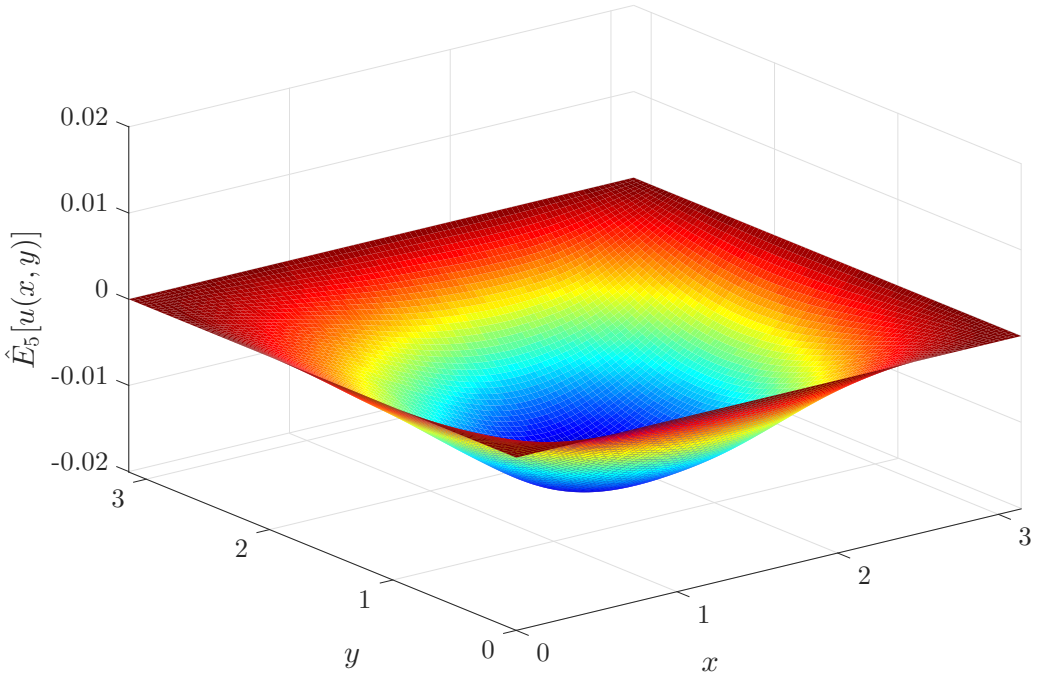
En rimlighetsbedömning av kurvpassningen gjordes genom att betrakta RMSE-felet som för ett antal observationer $(\delta_i, T_i)_{i=1}^N$ och modellfunktionen $g(\delta; \beta)$ definieras som

$$\text{RMSE} = \sqrt{\frac{S}{N}},$$

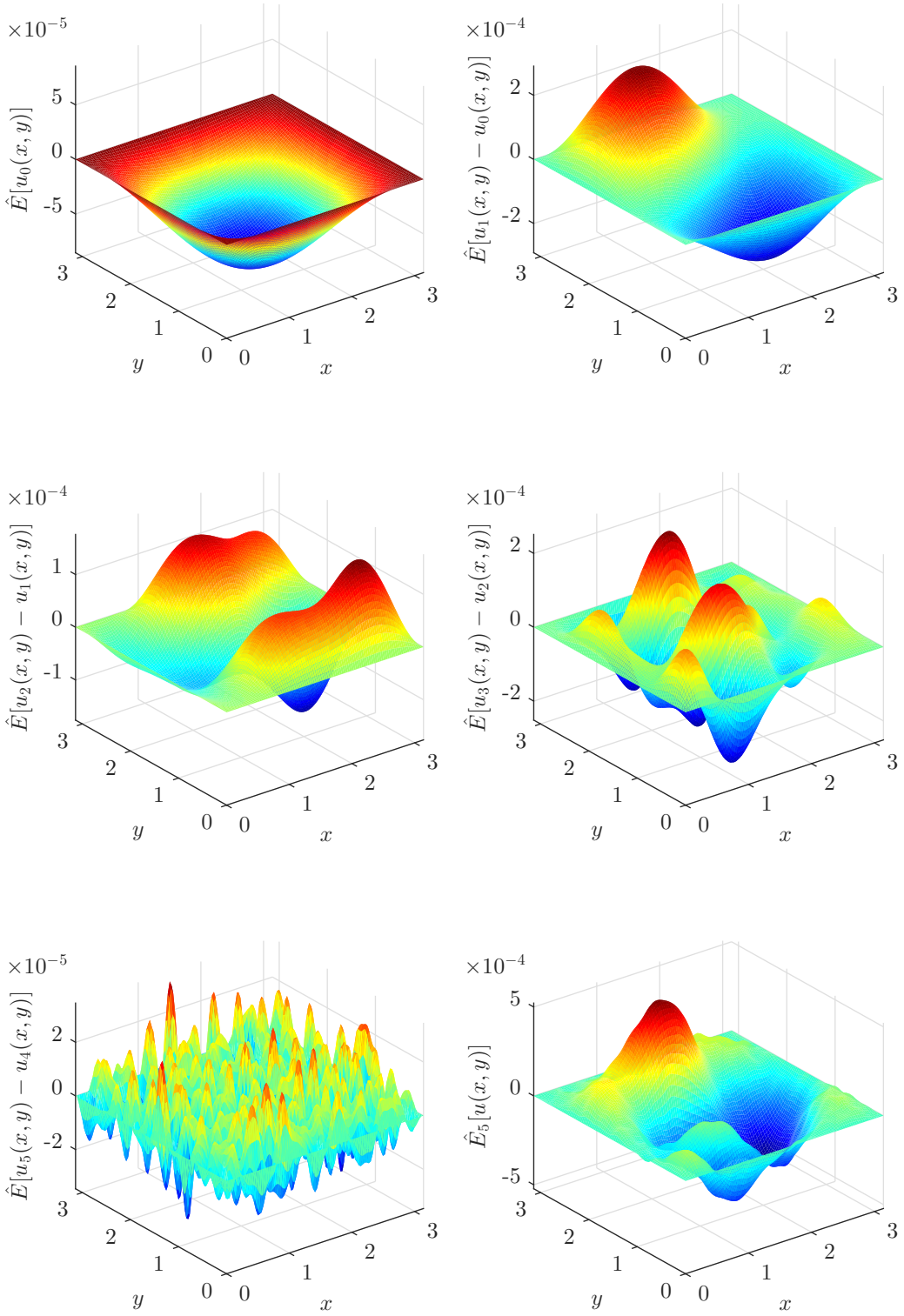
där summan S är definierad enligt (3.8). Detta mått kunde betraktas som en slags medelresidual, vilket säger hur mycket observationerna avviker från den passade linjen. Ett lägre RMSE tyder på en bättre passning. Utifrån kurvpassningen gavs ett antal optimala värden på a, b för både MC och MLMC på olika nivåer L.

4 Resultat

Två oberoende realisationer av MC- respektive MLMC-implementationen som användes presenteras i Figur 4.1, respektive Figur 4.2. Vi ser om vi jämför de totala estimatorerna i dessa figurer att MC-grafen ser jämnare ut än MLMC vilket också konstateras som ett generellt resultat då många sampels görs.



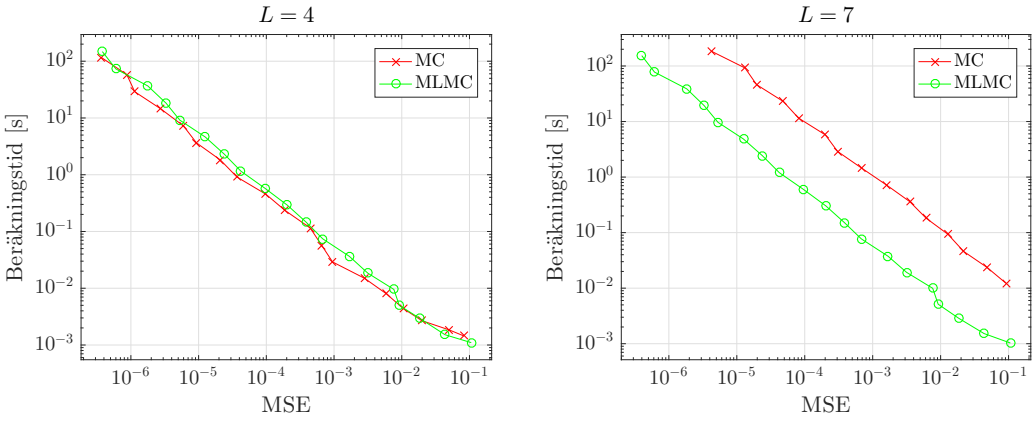
Figur 4.1: Ett sampel av MC-estimatorn $\hat{\mathbb{E}}_L[u(x,y)]$ enligt (3.5) med $L = 5$, $k = 1$, $\alpha = 4.5$ och $S = 100$.



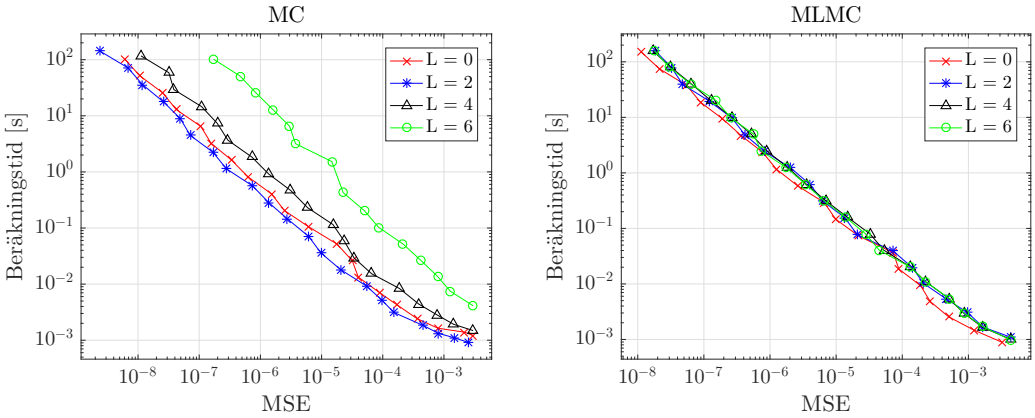
Figur 4.2: Sampels av $\hat{\mathbb{E}}[u_l(x,y) - u_{l-1}(x,y)]$ för flera nivåer l och den totala MLMC-estimatorn $\hat{\mathbb{E}}_L[u(x,y)]$ enligt (3.6) med $L = 5$, $k = 1$, $\alpha = 4.5$ och $S = 100$.

Resultatet från olika MSE-beräkningar presenteras bland annat i Figur 4.3 där vi har det resulterande skattade medelfältet från två olika simuleringar med MC respektive MLMC då $k = 10$, $\alpha = 8$ och L är 4 respektive 7 i de två olika fallen. Här ser vi att för $L = 7$ presterar MLMC-metoden bättre i termer av MSE i förhållande till MC medan för fallet $L = 4$ så presterar metoderna likvärdigt. Detta är ett genomgående mönster för olika simuleringar att för högre L så presterar MLMC bättre i förhållande till MC.

Om vi jämför MSE för olika nivåer L då MLMC implementeras så kan det konstateras att beräkningstiden för givet L är approximativt konstant. För MC gäller dock att beräkningstiden ökar med nivån L. Denna skillnad åskådliggörs i Figur 4.4 där vi ser att MLMC-graferna överlappar varandra medan MC-graferna visar på att $L = 2$ ger bäst resultat och $L = 6$ sämst.



Figur 4.3: Genomsnittlig beräkningstid för 100 MC- respektive MLMC-estimatorer för $L = 4$ respektive $L = 7$ plottat mot uppnått MSE beräknat i punkten $(1.3465, 1.7771)$ utifrån vilken $V_l(x, y)$ också beräknats för att optimera fördelningen av sampels mellan nivåer i MLMC. Övriga parametrar är $S = 100$, $k = 10$ och $\alpha = 8$. Vi ser att i fallet $L = 7$ så är beräkningstiden kortare då MLMC används i förhållande till när MC används. För $L = 4$ är metoderna istället approximativt likvärdiga.



Figur 4.4: Genomsnittlig beräkningstid för 100 MC- respektive MLMC-estimatorer för olika L. MSE är optimerat med avseende på punkten $(1.3465, 1.7771)$ då N_l beräknats. Övriga parametrar är $k = 1, \alpha = 4.5, S = 100$. Notera att för MLMC så överlappar de uppmätta värdena varandra medan det är tydligt att $L = 2$ presterar bäst och $L = 6$ presterar sämst för MC.

4.1 Konvergenstakt av MC/MLMC

Resultat för kurvpassningen enligt Avsnitt 3.6 presenteras i Tabell 4.1. En visualisering av anpassningen för $L = 7$ för såväl MC som MLMC presenteras i Figur 4.5 tillsammans med de uppmätta datapunkterna som anpassningen är gjord utifrån.

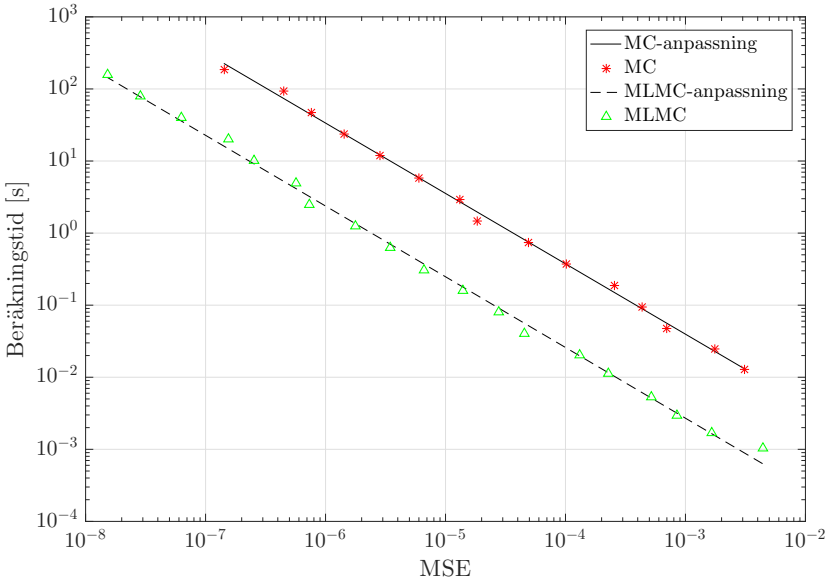
L	b	$\ln a$	RMSE
0	-0.90626	-12.914	0.40648
1	-0.92728	-13.205	0.35814
2	-0.91387	-13.384	0.38686
3	-0.92824	-13.287	0.36190
4	-0.94949	-12.769	0.31614
5	-0.95911	-12.144	0.18984
6	-1.08550	-12.097	0.25635
7	-0.97633	-9.9731	0.14137

(a) MC

L	b	$\ln a$	RMSE
0	-0.99309	-13.183	0.2116
1	-0.98909	-13.354	0.15577
2	-0.97523	-12.628	0.22161
3	-0.97309	-12.67	0.15753
4	-0.98539	-12.705	0.14686
5	-0.98273	-12.718	0.18991
6	-0.98749	-12.755	0.19656
7	-0.98123	-12.692	0.2019

(b) MLMC

Tabell 4.1: Tabell över anpassade värden på parametrarna a, b samt RMSE utifrån MC-/MLMC-simulering för olika nivåer L där $k = 1, \alpha = 4.5, S = 100$ hållits konstant.



Figur 4.5: Visualisering av anpassning av funktion till datapunkter för MC och MLMC där $L = 7, k = 1, \alpha = 4.5, S = 100$.

5 Diskussion

I detta avsnitt behandlas resultaten som redogörs i Avsnitt 3. Beroendet mellan tidskomplexiteten och MSE för MC-/MLMC-metoden diskuteras. Vi reflekterar också över varianter av vårt uppställda problem som skulle kunna visa effektiviteten hos MLMC på ett bättre sätt.

5.1 Skillnader mellan estimatorerna

Om vi ser på samplet av MC-estimatorn i Figur 3.2 respektive samplet av MLMC-estimatorn i sista grafen i Figur 3.3 så noterar vi att MC-estimatorn är jämnare än MLMC-estimatorn. Att detta är fallet generellt förklaras naturligt genom MLMC-metodens utformning som innebär att för högre nivåer l tas färre sampels än för lägre och en större slump spelar in där i förhållande till motsvarande nivå i MC. Således blir det förväntade felet av termen $u_l - u_{l-1}$ större för höga l i MLMC än för MC medan felet blir högre för lägre l i MC. I och med att det förväntade felet är det vi antas se i graferna så förklarar detta också varför MC-grafen är mjukare än MLMC då felet till större del uppkommer för lägre l i MC-approximationerna där termerna är av högst storleksordning.

5.2 Felkällor

Det finns ett antal felkällor som skulle kunna påverka slutresultatet. En sådan är att vi endast granskar MSE i en punkt på fältet när vi beräknar det optimala antalet iterationer på varje MLMC-nivå. Eftersom punkten valdes slumpmässigt så kan det spela en roll i slutresultatet. Till exempel ser vi att MSE kommer vara identiskt lika med noll på randen men inte i annan godtycklig punkt i fältet.

Kostnaden för att beräkna ett sampel på en MLMC-nivå beräknas i vårt fall numeriskt innan själva MLMC-metoden implementeras, det vill säga att vi beräknar kostnaderna för ett antal provkörningar av metoden och sedan antar att denna kostnad kommer att vara densamma vid varje körning framöver. Detta kan eventuellt leda till ett suboptimalt val av antalet iterationer på varje nivå.

5.3 Samband mellan beräkningstid och MSE

Resultaten i Tabell 4.1 indikerar att den asymptotiska tidskomplexiteten av MC och MLMC-implementationen verkar vara proportionell mot δ^{-1}, ty $b \approx -1$ för alla L. Skillnaden mellan metodernas konvergenstakt verkar främst vara en konstant som varierar beroende på L. För exemplet på körningar för olika nivåer L som syns i Figur 4.3 ser vi att för $L = 4$ presterar MC och MLMC ungefär likvärdigt. För $L = 7$ syns istället att MLMC presterar bättre än MC enligt ett konstant förhållande.

Utifrån resultaten tycks MLMC prestera bättre för höga nivåer, vilket är i enlighet med det vi väntar oss. För vårt problem märker vi dock att det krävs relativt få nivåer för att få en hög precision. Detta tyder på att vårt problem inte är komplext nog för att visa på alla fördelar med MLMC.

5.4 Val av problem

Fördelarna med MLMC för skattning av väntevärde hade kunnat åskådliggöras tydligare om vi hade arbetat med andra konstruktioner av det korrelerade stokastiska högerledet.

Fältet som arbetades med nu ger en teoretisk lösning till Poissons ekvation som för varje punkt $(x, y) \in \mathcal{D}$ uppfyller

$$\mathbb{E}[u(x, y)] = 0.$$

Väntevärdet för de approximativa lösningarna $u_L(x,y)$ för alla trunkeringsnivåer 2^L är identiska med det teoretiska väntevärdet $\mathbb{E}[u(x,y)]$ i varje punkt $(x,y) \in \mathcal{D}$ då

$$\mathbb{E}[u_L(x,y)] = \sum_{n,m=1}^{2^L} -\frac{\mathbb{E}[X_{n,m}]}{n^2+m^2}\sin(nx)\sin(my) = 0.$$

Detta innebär att använda en MC-estimator med $L = 0$, d.v.s. en enda term i summan ovan, skulle ge samma resultat som på varje nivå $L > 0$ givet tillräckligt många sampels. Om slumpvariablerna $X_{n,m}$ i $f(x,y)$ (2.5) valts så att $E[X_{n,m}] \not\equiv 0$ hade vi kunnat ha

$$\mathbb{E}[u(x,y)] \neq \mathbb{E}[u_L(x,y)].$$

Genom att skriva MSE för estimatorerna $\hat{\mathbb{E}}_L[u(x,y)]$ enligt nedan (se Definition A.14 för en allmän estimator Θ för att se varför vi kan skriva såhär) så kan vi se att ytterligare en term (bias) skulle komma in i felet i detta fall.

$$\begin{aligned}\text{MSE}\left[\hat{\mathbb{E}}_L[u(x,y)]\right] &= \text{Var}\left[\hat{\mathbb{E}}_L\left[u(x,y)\right]\right] + \left(\mathbb{E}\left[\hat{\mathbb{E}}_L[u(x,y)]\right] - \mathbb{E}[u(x,y)]\right)^2 \\ &= \text{Var}\left[\hat{\mathbb{E}}_L\left[u(x,y)\right]\right] + \left(\mathbb{E}\left[u_L(x,y)\right] - \mathbb{E}[u(x,y)]\right)^2.\end{aligned}$$

I detta projekt har vi enbart behövt betrakta den första termen eftersom vi haft $\mathbb{E}[u_L(x,y)] = \mathbb{E}[u(x,y)]$ och den andra termen har varit noll. Hade inte detta varit fallet dock hade vi behövt välja L så stort att

$$\left(\mathbb{E}\left[u_L(x,y)\right] - \mathbb{E}[u(x,y)]\right)^2 < \delta$$

för att kunna ha

$$\text{MSE}\left[\hat{\mathbb{E}}_L\left[u(x,y)\right]\right] < \delta.$$

Detta tyder på att den fulla kapaciteten av MLMC inte har utnyttjas. MLMC hade troligen kunnat presterat bättre i förhållande till MC tidsmässigt och dessutom haft en bättre konvergenstakt. Om vi ser på exempelvis Mike Giles skattningar av stokastiska partiella differentialekvationers väntevärde[3, s. 49-55, Figur 7.10, 7.11] så ser vi att han lyckas uppnå sådana resultat.

En relativt enkel modifiering till vårt problem skulle kunnat vara att ha ett väntevärde som avtar enligt samma modell som variansen. D.v.s. att vi hade haft samma stokastiska högerled f som definierat i (2.5) fast med

$$X_{n,m} \sim \mathcal{N}\left(k_1^2(n^2+m^2)^{\alpha_1/2}, k_2^2(n^2+m^2)^{\alpha_2/2}\right)$$

där $k_1, k_2, \alpha_1, \alpha_2 < \infty$ är reella parametrar. Ett ytterligare problem som hade uppstått om vi genomfört denna modifiering hade varit att beräkna bias-termen i MSE. Dessutom hade ytterligare teori för att visa att denna konstruktion av f konvergerar behövts.

Det finns även andra faktorer som tydligare skulle kunna åskådliggöra fördelarna med MLMC framför MC. Till exempel är det möjligt att implementera en MLMC-algoritm som för olika nivåer använder sig av ett finare rutnät. Detta kan bidra till en avsevärd tidsbesparing då antalet beräkningar som utförs i en iteration av Algoritm 3.2 respektive 3.4 till stor del beror på antalet sampelpunkter.

6 Slutsats

Resultaten indikerar att MLMC inte presterar bättre än MC då det bara är av intresse att beräkna en grov uppskattning. Det motsatta verkar gälla för finare uppskattningar, när fler nivåer används, då MLMC tenderar att vara mindre tidskrävande i förhållande till MC.

Metoderna visar på en liknande asymptotisk tidskomplexitet då körningstiden för båda verkar vara inverst proportionell mot MSE. Detta förmodas bero på att problemet inte är tillräckligt komplext för att MLMC skall ge en märkbar förändring i den asymptotiska tidskomplexiteten.

En större skillnad mellan algoritmernas körningstid hade eventuellt kunnat visas med ett mer beräkningstungt problem. Ett problem som dels skulle ha mer rum för approximationer, som minskar beräkningstiden avsevärt, och som har en lösning där väntevärdet av den trunkerade estimatorn och för den teoretiska lösningen till problemet inte är desamma.

Referenser

[1] D. P. Kroese, T. Brereton, T. Taimre och Z. I. Botev, *Why the Monte Carlo method is so important today*. WIREs Comp Stat, 2014, vol. 6, s. 386–392.

[2] K. Cliffe, M. Giles, R. Scheichl och A. L. Teckentrup, "Multilevel Monte Carlo methods and applications to elliptic PDEs with random coefficients", *Computing and Visualization in Science*, vol. 14, nr 1, s. 3–15, 2011.

[3] M. Giles, "Multilevel Monte Carlo methods", *Acta Numerica*, vol. 24, s. 259–328, 2015.

[4] G. Grimmett och D. Stirzaker, *Probability and random processes*. Oxford university press, 2001.

[5] G. B.Folland, *Fourier Analysis and It's Application*. American Mathematical Sociaty, 2009.

[6] J. Kuttler och V. Sigillito, "Eigenvalues of the laplacian in two dimensions", *Siam Review*, vol. 26, nr 2, s. 163–193, 1984.

[7] G. Lord, C. Powell och T. Shardlow, *An Introduction to Computational Stochastic PDEs*. Cambridge University Press, 2014.

[8] V. V. Petrov, "Sums of independent random variables", i. Berlin, Heidelberg: Springer Berlin Heidelberg, 1975, kap. Laws of Large Numbers, s. 256–291.

[9] S. Ghorpade och B. V. Limaye, *A course in multivariable calculus and analysis*. Springer, 2010.

A Grundläggande definitioner

Här redovisas för grundläggande begrepp inom sannolikhetslära och statistik som används i rapporten. Grunden för detta avsnitts konstruktion har varit [4] och [7].

Inom sannolikhetsläran bygger $(\Omega, \mathcal{A}, \mathbb{P})$, Ω ett utfallsrum, $\mathcal{A}$ en sigma-algebra av delmängder till Ω och ett sannolikhetsmått $\mathbb{P}$, ett sannolikhetsrum.

Definition A.1. *En reellvärd stokastisk variabel $X : \Omega \to \mathbb{R}$ är en funktion som avbildar utfallsrummet Ω på $\mathbb{R}$.*

Definition A.2. *Fördelningsfunktionen av en stokastisk variabel är en funktion $F_X : \mathbb{R} \to [0,1]$ definierad av*

$$F_X(x) = \mathbb{P}(X \leq x).$$

Definition A.3. *En stokastisk variabel är kontinuerlig om dess fördelningsfunktion kan uttryckas*

$$F_X(x) = \mathbb{P}(X \leq x) = \int_{-\infty}^{x} f_X(s)\,\mathrm{d}s$$

för någon funktion $f_X : \mathbb{R} \to [0,\infty)$.

Definition A.4. *Funktionen f_X i Definition A.3 definieras som täthetsfunktionen till den stokastiska variabeln X.*

Definition A.5. *För en kontinuerlig stokastisk variabel X med täthetsfunktion $f_X(x)$ definieras väntevärdet $\mu = \mathbb{E}[X]$ som*

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} x f_X(x)\,\mathrm{d}x.$$

Definition A.6. *En stokastisk variabel X med $\mathbb{E}[X] = 0$ sägs vara centrerad.*

Definition A.7. *För två stokastiska variabler $X_1, X_2 \in \mathbb{R}$ definierade på ett sannolikhetsrum $(\Omega, \mathcal{A}, \mathbb{P})$ gäller att de är oberoende om*

$$\mathbb{E}[X_1 X_2] = \mathbb{E}[X_1]\,\mathbb{E}[X_2].$$

Definition A.8. *För en stokastisk variabel $X \in \mathbb{R}$ definieras variansen $\sigma^2 = \mathrm{Var}[X]$ som*

$$\mathrm{Var}[X] = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right].$$

Definition A.9. *För en mängd $D \in \mathbb{R}^d$, är ett stokastiskt fält $P = \{P(\mathbf{x},\omega) : (\mathbf{x},\omega) \in D \times \Omega\}$ en mängd reellvärda variabler definierade på ett sannolikhetsrum $(\Omega, \mathcal{A}, \mathbb{P})$.*

Definition A.10. *För ett fixt $\omega \in \Omega$ är den associerade realisationen av ett stokastiskt fält f en deterministisk funktion $h : \mathcal{D} \to \mathbb{R}$ definierad enligt $h(\mathbf{x}) \equiv f(\mathbf{x}, \omega), \quad \mathbf{x} \in \mathcal{D}$.*

Definition A.11. *För ett kontinuerligt stokastiskt fält P definierar vi väntevärdet som en funktion $\mu : \mathcal{D} \to \mathbb{R}$ givet av $\mu(\mathbf{x}) = \mathbb{E}[P(\mathbf{x})]\ \forall \mathbf{x} \in \mathcal{D}$.*

Definition A.12. *En sekvens av stokastiska variabler $\{X_n(\omega)\}_{n=1}^{\infty}$ definierade på sannolikhetsrummet $(\Omega, \mathcal{A}, \mathbb{P})$ konvergerar nästan säkert mot X för alla ω om*

$$X_n(\omega) \underset{n\to\infty}{\longrightarrow} X(\omega)$$

det vill säga om

$$\{\omega \in \Omega : X_n(\omega) \underset{n\to\infty}{\longrightarrow} X(\omega)\}$$

är en händelse vars sannolikhet är ett.

Definition A.13. *Givet en sekvens av stokastiska variabler $\{X^{(i)}\}_{i=1}^{N}$ med tillhörande fördelningsfunktioner $\{F_X^{(i)}\}_{i=1}^{N}$ säger vi att $X^{(i)}$ konvergerar distributionsmässigt mot den stokastiska variabeln X med fördelningsfunktionen F om $F_X^{(i)} \to F$ då $N \to \infty$. Detta betecknar vi som*

$$X_n \xrightarrow{D} X.$$

Definition A.14. *För en estimator* Θ *till en betraktad parameter* θ *definierar vi MSE som*

$$\begin{aligned}\mathrm{MSE}[\Theta] &\equiv \mathbb{E}\left[(\Theta-\theta)^2\right] \\ &= \mathbb{E}\left[\Theta^2 - 2\Theta\theta + \theta^2\right] \\ &= \mathbb{E}\left[\Theta^2\right] - \left(\mathbb{E}[\Theta]\right)^2 + \left(\mathbb{E}[\Theta]\right)^2 - 2\,\mathbb{E}[\Theta]\theta + \theta^2 \\ &= \mathrm{Var}[\Theta] + \left(\mathbb{E}[\Theta]-\theta\right)^2.\end{aligned}$$

Här kallar vi den andra termen $\left(\mathbb{E}[\Theta]-\theta\right)^2$ *för estimatorns bias.*

B MATLAB-kod

I detta avsnitt visas de mest centrala MATLAB-funktionerna som använts under detta projekt.

B.1 Monte Carlo

Här presenteras de tre viktigaste MATLAB-funktioner som berör MC-lösningen för vårt problem. Funktionen `MC` som ger en MC-skattning av medelvärdet till lösningen av ekvationen presenteras först. Därefter kommer funktionen `MC_sample` som genererar ett sampel till `MC`-funktionen och sist presenteras funktionen `MSE_MC` som beräknar MSE för vår MC-metod.

```
% Written by Dimitri
%
% Gives an MC solution to the Poisson equation in two dimensions with
% stochastic right hand side.
%
% S:            # of sample points along each axis
% L:            Decides the truncation at 2^L terms for each dimension
% k, alpha:     Random coefficient distribution parameters,
%                   i.e. Var[a_{n, m}] = k^2*(n^2+m^2)^(-alpha/2)
% iter:         # of iterations to take the mean of
%
function E_approx = MC(S, L, iter, k, alpha)
    nm = repmat((1:2^L).^2, [2^L, 1]);
    nm = nm + nm';
    % Values of sin(nx_s) in sample points x_s
    sinvals=sin(repmat(linspace(0,pi,S)',1,2^L).*repmat((1:2^L),S,1));
    u_sum = zeros(S);
    for i = 1:iter  % Create sum of sampels of u_L(x)
        u_sum = u_sum + MC_sample(k, alpha, nm, sinvals);
    end
    E_approx = u_sum / iter;  % Ê_L[u(x)]
end
```

```
% Written by Dimitri
%
% Gives a sample of u_L(x)
%
% k, alpha:     Random coefficient distribution paramters,
%                   i.e. Var[a_{n, m}] = k^2*(n^2+m^2)^(-alpha/2)
% nm:           Matrix with values nm(n, m) = n^2 +m^2
% sinvals:      Matrix storing values of sin(nx_s) in samplepoints x_s
%
function u_L = MC_sample(k, alpha, nm, sinvals)
    % Matrix with values C(n,m) = - X_{n,m}/(n^2+m^2)^alpha
    C = -k*(randn(size(nm))).*(nm.^(-1-alpha/4));
    u_L = sinvals*C*sinvals';  % u_L(x) in sample points
end
```

```
% Written by Kevin and Dimitri
%
% Calculates MC solutions of the Poisson equation in two dimensions
% repeatedly. Decides the MSEof the total solution and takes the time to
% calculate a specific MSE.
%
% S:            amount of points sampled along x and y axis
% L:            highest MLMC level
% k, alpha:     Random coefficient distribution parameters,
%                   i.e. Var[a_{n, m}] = k^2*(n^2+m^2)^(-alpha/2)
% maxpwr:       Exponential deciding maximum N = 2^pwr
% samples:      Number of samples of Ê_L[u(x)]
```

```
%
function [MSE, time] = MSE_MC(S, L, k, alpha, maxpwr, samples, s_x, s_y)
E_hat_L = zeros(S, S, samples);     % Field Ê_L[u(x)]
time = zeros(1, maxpwr);            % Measured time vector
MSE=zeros(maxpwr,1);                % MSE vector
x = (s_x - 1)/(S-1)*pi;
y = (s_y - 1)/(S-1)*pi;

% Creating savepath from current date and time
starttime = datestr(datetime('now','Format','yyyyMMdd_HHmm'));
savepath = sprintf('./data/MSE_MC_%s.mat',starttime);

gcp;    % Initialize parallel pool if it doesn't exist
for pwr = 1:maxpwr
    N = 2^pwr;
    tic;    % Start time
    parfor i = 1:samples
        E_hat_L(:,:,i) = MC(S, L, N, k, alpha);
    end
    time(pwr) = toc / samples;  % Average elapsed time
    MSE(pwr) = mean(E_hat_L(s_x, s_y, :).^2);   % MSE by second moment
    save(savepath,'MSE','time','S','L','alpha','maxpwr','samples',...
        's_x','s_y','x','y');
end
fprintf('Done!\n');
end
```

B.2 Multilevel Monte Carlo

Här presenteras de tre viktigaste MATLAB-funktioner som används för MLMC-lösningen av vårt problem. Först redovisas funktionen `MLMC` som ger en MLMC-skattning av medelvärdet av lösningarna till ekvationen. Därefter presenteras funktionen `MLMC_level` som generar sampel på de olika MLMC-nivåerna och sist redovisas funktionen `MSE_MLMC` som beräknar MSE för MLMC-implementationen av problemet.

```
% Written by Kevin
%
% Gives an MLMC solution of the Poisson equation with stochastic right
% hand side.
%
% N:            Vector for # iterations on each level l, indice l+1 for level l
% k, alpha:     Random coefficient distribution parameters,
%                   i.e. Var[a_{n, m}] = k^2*(n^2+m^2)^(-alpha/2)
% S:            # of points sampled along x and y axis
%
function E_approx = MLMC(S, N, k, alpha)
L = length(N(N~=0))-1;
E_approx = zeros(S,S);
nm = repmat((1:2^L).^2,2^L,1);
nm = nm+nm';    % Matrix with values nm(n, m) = n^2 + m^2
% Values of sin(nx_s) in sample points x_s
sinvals = sin(repmat(linspace(0,pi,S)',1,2^L).*repmat((1:2^L),S,1));
for l=0:L
    sinvals_l = sinvals(:,1:2^l);   % Truncates to match level l
    nm_l = nm(1:2^l, 1:2^l);        % Truncates to match level l
    sum_u = zeros(S,S);
    for iterations=1:N(l+1) % Create sum of N_l l:th level samples
        sum_u = sum_u + MLMC_level(l, k, alpha, nm_l, sinvals_l);
    end
    E_approx = E_approx + sum_u/N(l+1);   % Ê_L[u(x)]
end
end



% Written by Kevin
%
```

```
% Gives a sample of the l:th level solution u_l−u_{l−1}
% The variance is k*(n^2+m^2)^(−alpha/2)
%
% l:            The concerned level
% k, alpha:     Random coefficient distribution parameters,
%                   i.e. Var[a_{n, m}] = k^2*(n^2+m^2)^(−alpha/2)
% nm_l:         Matrix with values nm(m,n)=m^2+n^2
% sinvals_l:    Values of sin(nx) at different points
%
function u_l_diff = MLMC_level(l, k, alpha, nm_l, sinvals_l)
if l~=0
    % Matrix with values C(n,m) = −X_{n,m}/(n^2+m^2)^alpha for (n,m) in S_l
    C = −k*[zeros(2^(l−1)), randn(2^(l−1));
        randn(2^(l−1)), randn(2^(l−1))].*(nm_l.^(−1−alpha/4));
    u_l_diff = sinvals_l*C*sinvals_l';  % u_l(x)−u_{l−1}(x) in sample points
else
    % Handles case l=0 separately
    u_l_diff = sinvals_l*(randn*k*2^(−1−alpha/4))*sinvals_l';
end
end
```

```
% Written by Kevin
%
% Calculates MC solutions of the Poisson equation in two dimensions
% repeatedly. Decides the MSE of the total solution and takes the time to
% calculate a specific MSE.
%
% S:            amount of points sampled along x and y axis
% L:            highest MLMC level
% k, alpha:     Random coefficient distribution parameters,
%                   i.e. Var[a_{n, m}] = k^2*(n^2+m^2)^(−alpha/2)
% maxpwr:       Exponential deciding maximum N_tot = 2^pwr
% samples:      Number of samples of Ê_L[u(x)]
%
function [MSE, time] = MSE_MLMC(S, L, k, alpha, maxpwr, samples, s_x, s_y)
load('./data/cost.mat');    % Load file with cost on each level l
C_l = costl(1:(L+1))';      % From the loaded file
x = (s_x − 1)/(S−1)*pi;
y = (s_y − 1)/(S−1)*pi;
frac_level = lfrac(L, k, alpha, C_l, x, y); % Theroretical N_l normalized
%N_l = zeros(maxpwr,L+1);                    % N_l values matrix
E_hat_l = zeros(S, S, samples);             % Field Ê_L[u(x)]
time = zeros(maxpwr, 1);                    % Measured time vector
MSE = zeros(maxpwr,1);                      % MSE vector

% Creating savepath from current date and time
starttime = datestr(datetime('now','Format','yyyyMMdd_HHmm'));
savepath = sprintf('./data/MSE_MLMC_%s.mat',starttime);

gcp;    % Initialize parallel pool if it doesn't exist
for pwr = 1:maxpwr
    N_l = round(2^pwr*frac_level);
    tic;    %Start time
    for i = 1:samples
        E_hat_l(:,:,i) = MLMC(S, N_l, k, alpha);
    end
    time(pwr) = toc / samples;  %Averaged elapsed time
    MSE(pwr) = mean(E_hat_l(s_x, s_y, :).^2);   % MSE by second moment
    save(savepath,'MSE','time','S','L','alpha','maxpwr','samples',...
        's_x','s_y','x','y','N_l','frac_level');
end
fprintf('Done!\n')
end
```

B.2.1 Sampelberäkning för MLMC

Här redovisas för de funktioner som används för att beräkna N_l, d.v.s. det optimala antalet sampels på varje nivå i MLMC-implementationen. Först är funktionen `lfrac` som ger vilken andel av iterationerna som ska genomföras på respektive nivå. Därefter presenteras `levelvar` som ger variansen på på varje nivå l och sist `timecost` som beräknar tidskomplexiteten för sampels på de olika nivåerna l genom att upprepat kalla på `MLMC_level`.

```
% Written by Kevin
%
% Calculates the optimal fraction of calculations on each level
%
% k, alpha: Random coefficient distribution parameters,
%   i.e. Var[a_{n, m}] = k^2*(n^2+m^2)^(-alpha/2)
% L:        Maximum level l
% costl:    Costs on each level l
%
function N_l_norm = lfrac(L, k, alpha, costl, x, y)
V_l=levelvar(L, k, alpha, x, y);    % Variances on each level l
N_l=sqrt(V_l./costl);               % Scale factor for # of iterations
N_l_norm = N_l / sum(N_l);          % Normalize
end
```

```
% Written by Kevin
%
% Returns a vector of the average variance of u_l-u_{l-1}
% Indice l+1 for level l
%
% L:            Maximum level l
% k, alpha:     Random coefficient distribution parameters,
%                   i.e. Var[a_{n, m}] = k^2*(n^2+m^2)^(-alpha/2)
% S:            Sample points along each axis
% x:            Concerned point x
% y:            Concerned point y
%
function V_l = levelvar(L, k, alpha, x, y)
V_l=zeros(1,L+1);
for l=0:L
    for n=1:2^l
        for m=max(2^(l-1)+1,n+1):2^l
            % Adds non-diagonal terms
            V_l(l+1) = V_l(l+1)+k*(m^2+n^2)^(-2-alpha/2)*...
                ((sin(n*x)*sin(m*y))^2 + (sin(m*x)*sin(n*y))^2);
        end
        if n > 2^(l-1)
            % Adds diagonal terms
            V_l(l+1) = V_l(l+1)+k*(2*n^2)^(-2-alpha/2)*...
                (sin(n*x)*sin(n*y)).^2;
        end
    end
end
end
```

```
% Written by Kevin
%
% Calculates the time cost for MLMC levels up to level L.
%
% S:            amount of points sampled along x and y axis
% L:            highest MLMC level
% k, alpha:     Random coefficient distribution parameters,
%
function [t, iter] = timecost(S, L, k, alpha, tlimit)
t=zeros(L+1,1);

% Creating savepath from current date and time
starttime=datestr(datetime('now','Format','yyyyMMdd_HHmm'));
```

```
savepath=sprintf('./data/timecost_%s.mat',starttime);

nm = repmat((1:2^L).^2,2^L,1);
nm = nm+nm';     % Matrix with values nm(n, m) = n^2 + m^2
% Values of sin(nx_s) in sample points x_s
sinvals = sin(repmat(linspace(0,pi,S)',1,2^L).*repmat((1:2^L),S,1));

for l=0:L
    sinvals_l = sinvals(:,1:2^l);   % Truncates to match level l
    nm_l = nm(1:2^l, 1:2^l);        % Truncates to match level l
    sum_u = zeros(S, S);
    iter = 0;
    tStart = tic;
    while(toc(tStart) < tlimit)
        % Stores the sum of all l:th level solutions
        sum_u = sum_u + MLMC_level(l, k, alpha, nm_l, sinvals_l);
        iter = iter + 1;
    end
    t(l+1) = toc(tStart) / iter;
    save(savepath,'t','iter');
end
```