Institut für Parallele und Verteilte Systeme
Abteilung Simulation großer Systeme
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3590

Quantifizierung von Unsicherheiten auf adaptiven dünnen Gittern mit stückweise polynomiellen Basisfunktionen

Michael Lahnert

Studiengang:	Informatik
Prüfer/in:	Jun.-Prof. Dr. rer. nat. Dirk Pflüger
Betreuer/in:	M.Sc. Fabian Franzelin
Beginn am:	18. November 2013
Beendet am:	20. Mai 2014
CR-Nummer:	G.3, I.6.6

Kurzfassung

Im Zusammenhang mit der Quantifizierung von Unsicherheiten entstehen, bspw. bei der Berechnung des Erwartungswerts, potentiell hochdimensionale Quadraturprobleme. Eine Möglichkeit, um den Fluch der Dimensionalität zumindest teilweise zu überwinden und gleichzeitig mit einer möglichst niedrigen Zahl von Auswertungen eine gute Approximation zu erhalten, stellen dünne Gitter dar.

Bei nicht-intrusiven Verfahren zur Quantifizierung von Unsicherheiten wird das Verhalten eines Systems durch mehrere Simulationsauswertungen mit unterschiedlichen Parameterkombinationen aus dem definierten Wertebereich untersucht, wobei schon ein einzelner Simulationsaufruf einige Rechenzeit in Anspruch nehmen kann. Daher soll die für eine gute Approximation notwendige Zahl der zu berechnenden Parameterkombinationen weiter reduziert werden. Neben der Verwendung von dünnen Gittern wurden im Rahmen dieser Arbeit zusätzlich stückweise polynomielle Basisfunktionen angesetzt, um die Konvergenzordnung der Dünngitterapproximation zu erhöhen. Zusätzlich soll die Zahl der nötigen Auswertungen durch räumlich-adaptive Gitterverfeinerung minimiert werden.

Inhaltsverzeichnis

Abbildungsverzeichnis **7**

Tabellenverzeichnis **9**

Verzeichnis der Listings **9**

Abkürzungsverzeichnis **9**

1. Einleitung **11**

2. Uncertainty Quantification **13**
2.1. Arten von Unsicherheiten . . . 13
2.2. Simulationsumgebung . . . 14
2.3. Ziele des Quantifizierens von Unsicherheiten . . . 16
2.4. Allgemeine Problemstellung . . . 17
2.5. Verfügbare Methoden . . . 17
2.5.1. Intrusive Verfahren . . . 17
2.5.2. Nicht-intrusive Verfahren . . . 18

3. Dünne Gitter **21**
3.1. Hintergrund und Einführung . . . 21
3.1.1. Grundlegendes . . . 21
3.1.2. Übergang zu höheren Dimensionen . . . 23
3.1.3. Randpunkte . . . 25
3.1.4. Polynomielle Basisfunktionen . . . 26
3.2. Operationen . . . 28
3.2.1. Hierarchisierung . . . 28
3.2.2. Evaluation . . . 30
3.2.3. Quadratur . . . 32
3.3. Adaptivität . . . 32

4. Implementierung **35**
4.1. SG++ . . . 35
4.1.1. Allgemeines . . . 35
4.1.2. Neue “UltraPoly”-Klassen . . . 36
Lagrange-Basis . . . 39
Newton-Basis . . . 40
Berechnung der Indizes . . . 41

Hierarchisierung und Dehierarchisierung . . . 42
Evaluation . . . 43
Quadratur . . . 43
Adaptivität . . . 44
4.2. UQLib . . . 44
4.2.1. UQ-Setting . . . 45
4.2.2. Steuerung des Simulationsaufrufs . . . 45

5. Experimente **49**
5.1. Experimente analog [MZ09, Kapitel 4.1] . . . 49
5.1.1. Betragsfunktion . . . 50
5.1.2. Kosinus Hyperbolicus . . . 53
5.1.3. Zwischenfazit . . . 57
5.2. Vergleich der Hierarchisierungsmethoden . . . 57
5.3. Kármán'sche Wirbelstraße . . . 60
5.3.1. Versuchsaufbau . . . 60
5.3.2. Ergebnisse . . . 62
Referenz . . . 62
CSGC . . . 64
ASGC . . . 71
Vergleich der Verfahren . . . 77

6. Zusammenfassung und Ausblick **81**
6.1. Fazit . . . 81
6.2. Ausblick . . . 82

A. Anhang **85**
A.1. Maximaler Fehler pro Verfeinerungsschritt für cosh . . . 85
A.2. Hierarchisierung - benötigte Zeit über Knotenzahl . . . 86
A.3. Varianzberechnung der Wirbelstraße nach MC Quadratur . . . 89
A.3.1. CSGC, stückweise polynomielle Basisfunktionen . . . 89
A.3.2. ASGC, stückweise lineare Basisfunktionen . . . 90
A.3.3. ASGC, stückweise polynomielle Basisfunktionen . . . 91

Literaturverzeichnis **93**

Abbildungsverzeichnis

2.1. Aufbau numerische Simulation ohne Unsicherheiten nach [MK10, S. 4] 15
2.2. Aufbau numerische Simulation mit Unsicherheiten nach [MK10, S. 5] 16

3.1. Level-Index-Baum für randloses dünnes Gitter Level 4 22
3.2. Interpolationsbeispiel 1D, entnommen aus [Pfl10, S. 9] 23
3.3. Teilraumschema dünnes Gitter: Dimension $d = 2$, Level $l = 3$, kein Rand 24
3.4. Teilraumschema dünnes Gitter: Dimension $d = 2$, Level $l = 4$, regulärer Rand 25
3.5. Teilraumschema dünnes Gitter: Dimension $d = 2$, Level $l = 3$, Trapezrand 26
3.6. Menge potentieller Stützstellen für Knoten $(4, 11)$ 27
3.7. Finden beitragender Basisfunktionen, entnommen aus [Pfl14, S. 123] 31
3.8. Beispiel für adaptive Verfeinerung eines dünnen Gitters, entnommen aus [Pfl10, S. 21] 33

4.1. Aktueller Programmablauf in “PolyBasis”-Klasse 37
4.2. Aktueller Programmablauf in “UltraPoly”-Klassen 38
4.3. Array-Index der zum Knoten gehörenden Basisfunktionskoeffizienten 40
4.4. Array-Index der $\alpha_{l,i}$ im Array 41

5.1. Funktionsplot der Betragsfunktion nach [MZ09, S. 3096] 50
5.2. Betragsfunktion: Entwicklung des maximalen Interpolationsfehlers 51
5.3. Betragsfunktion: Entwicklung Interpolationsfehler über Knotenzahl, linear 52
5.4. Betragsfunktion: Entwicklung Interpolationsfehler über Knotenzahl, quadratisch . . 52
5.5. Betragsfunktion: Entwicklung Interpolationsfehler über Knotenzahl, Polynomgrad 6 53
5.6. Betragsfunktion: Vergleich max Interpolationsfehler, Polynomgrad $p = [1, 2, 6]$ . . . 53
5.7. Funktionsplot der hyperbolischen Kosinusfunktion 54
5.8. $\cosh$: Entwicklung des maximalen Interpolationsfehlers 55
5.9. $\cosh$: Entwicklung mittlerer Quadraturfehler über Knotenzahl, linear 56
5.10. $\cosh$: Entwicklung mittlerer Quadraturfehler über Knotenzahl, quadratisch 56
5.11. $\cosh$: Entwicklung mittlerer Quadraturfehler über Knotenzahl, Polynomgrad 6 57
5.12. Vergleich von direkter und indirekter Hierarchisierung, Polynomgrad 2 59
5.13. Vergleich von direkter und indirekter Hierarchisierung, Polynomgrad 10 59
5.14. Geometrie der Kármán’schen Wirbelstraße 60
5.15. Wirbelstraße: Erwartungswert: MC, QMC 63
5.16. Wirbelstraße: Erwartungswertdifferenz: QMC zu MC 63
5.17. Wirbelstraße: Varianz: MC, QMC 64
5.18. Wirbelstraße: Erwartungswert CSGC Level $l = [4, 5, 6]$, linear, analytische Quadratur 65
5.19. Wirbelstraße: Erwartungswertdiff.: CSGC Level $l = [4, 5, 6]$, linear, analytisch zu MC 65
5.20. Wirbelstraße: Varianz: CSGC Level $l = [4, 5, 6]$, linear, analytische Quadratur 66

5.21. Wirbelstraße: Varianzdifferenz: CSGC Level $l = 6$, linear, analytisch zu MC 66
5.22. Wirbelstraße: Erwartungswert CSGC Level $l = [4, 5, 6]$, linear, MC Quadratur 67
5.23. Wirbelstraße: Erwartungswertdiff.: CSGC Level $l = [4, 5, 6]$, linear, MC Quad. zu MC 67
5.24. Wirbelstraße: Varianz: CSGC Level $l = [4, 5, 6]$, linear, MC Quadratur 68
5.25. Wirbelstraße: Varianzdifferenz: CSGC Level $l = [4, 5, 6]$, linear, MC Quadratur zu MC 68
5.26. Wirbelstraße: Erwartungswert: CSGC Level $l = [4, 5, 6]$, polynomiell, analytisch . . . 69
5.27. Wirbelstraße: Erwartungswertdiff.: CSGC Level $l = [5, 6]$, poly., analytisch zu MC . . 69
5.28. Wirbelstraße: Varianz: CSGC Level $l = [4, 5, 6]$, polynomiell, analytische Quadratur . 70
5.29. Wirbelstraße: Varianzdifferenz: CSGC Level $l = 6$, polynomiell, analytisch zu MC . . 70
5.30. Wirbelstraße: Erwartungswert: CSGC Level $l = [4, 5, 6]$, polynomiell, MC Quadratur 71
5.31. Wirbelstraße: Erwartungswertdiff.: CSGC Level $l = [4, 5, 6]$, poly., MC Quad. zu MC . 71
5.32. Wirbelstraße: Erwartungswert: ASGC, linear, analytische Quadratur 72
5.33. Wirbelstraße: Erwartungswertdifferenz: ASGC, linear, analytische Quadratur zu MC 72
5.34. Wirbelstraße: Varianz: ASGC, linear, analytische Quadratur 73
5.35. Wirbelstraße: Varianzdifferenz: ASGC, linear, analytische Quadratur zu MC 73
5.36. Wirbelstraße: Erwartungswert: ASGC, linear, MC Quadratur 74
5.37. Wirbelstraße: Erwartungswertdifferenz: ASGC, linear, MC Quadratur zu MC 74
5.38. Wirbelstraße: Erwartungswert: ASGC, polynomiell, analytische Quadratur 75
5.39. Wirbelstraße: Erwartungswertdifferenz: ASGC, polynomiell, analytisch zu MC 75
5.40. Wirbelstraße: Varianz: ASGC, polynomiell, analytische Quadratur 76
5.41. Wirbelstraße: Varianzdifferenz: ASGC, polynomiell, analytisch zu MC 76
5.42. Wirbelstraße: Erwartungswert: ASGC, polynomiell, MC Quadratur 77
5.43. Wirbelstraße: Erwartungswertdifferenz: ASGC, polynomiell, MC Quadratur zu MC . 77
5.44. Vergleich der Erwartungswertdifferenz der geeignetsten Verfahren 78
5.45. Vergleich der Varianzdifferenz der geeignetsten Verfahren 79

A.1. $\cosh$: Entwicklung max. Interpolationsfehler über Knotenzahl, linear 85
A.2. $\cosh$: Entwicklung max. Interpolationsfehler über Knotenzahl, quadratisch 86
A.3. $\cosh$: Entwicklung max. Interpolationsfehler über Knotenzahl, Polynomgrad 6 86
A.4. Benötigte Zeit über Knotenzahl, direkte Hierarchisierung, Polynomgrad 2 87
A.5. Benötigte Zeit über Knotenzahl, rekursive Hierarchisierung, Polynomgrad 2 87
A.6. Benötigte Zeit über Knotenzahl, direkte Hierarchisierung, Polynomgrad 10 88
A.7. Benötigte Zeit über Knotenzahl, rekursive Hierarchisierung, Polynomgrad 10 88
A.8. Wirbelstraße: Varianz: CSGC Level $l = [4, 5, 6]$, polynomiell, MC Quadratur 89
A.9. Wirbelstraße: Varianzdifferenz: CSGC Level $l = [4, 5, 6]$, polynomiell, MC Quad. zu MC 89
A.10. Wirbelstraße: Varianz: ASGC, linear, MC Quadratur 90
A.11. Wirbelstraße: Varianzdifferenz: ASGC, linear, MC Quadratur zu MC 90
A.12. Wirbelstraße: Varianz: ASGC, polynomiell, MC Quadratur 91
A.13. Wirbelstraße: Varianzdifferenz: ASGC, polynomiell, MC Quadratur zu MC 91

Tabellenverzeichnis

4.1. Lagrange-Basis: Speicherschema der Koeffizienten der polynomiellen Basisfunktionen 39

5.1. Vergleich der zur Hierarchisierung benötigten Zeit für Polynomgrad $p = 2$ 58
5.2. Vergleich der zur Hierarchisierung benötigten Zeit für Polynomgrad $p = 10$ 58

Verzeichnis der Listings

4.1. Mapping von $(p_{\max}, l, i)$ auf Arrayposition für $\alpha_{l,i}$ 42

Abkürzungsverzeichnis

ASGC	Adaptive Sparse Grids Collocation
CSGC	Conventional Sparse Grids Collocation
gPC	verallgemeinerte polynomielle Chaosentwicklung
JNI	Java Native Interface
MC	Monte Carlo
o. B. d. A.	ohne Beschränkung der Allgemeinheit
QMC	Quasi Monte Carlo
QoI	Quantity of Interest
SGCM	Sparse Grid Collocation Method
SWIG	Simplified Wrapper and Interface Generator
UQ	Uncertainty Quantification

1. Einleitung

Beschreibungen von Systemen sind Unsicherheiten unterworfen, die aus den verschiedensten Gründen auftreten. Dies können ungenaue Messungen sein, bspw. durch Rauschen oder schlecht kalibrierte Messwerkzeuge. Denkbar sind auch Ergebnisse aus vereinfachten Modellierungen - etwa die Vernachlässigung von Fertigungstoleranzen, die Reduktion eines dreidimensionalen Modells auf zwei Dimensionen oder die Vernachlässigung von ausgewählten Einflüssen beim Entwurf des Modells. Auch komplexe Prozesse, die nicht oder nur sehr schwer deterministisch zu modellieren sind, wie bspw. die Landung eines Flugzeugs, können Unsicherheit in eine Simulation einbringen.

Man möchte trotz dieser und weiterer unsicherer Einflüsse auf das untersuchte System aber nicht auf die Vorteile des Einsatzes von Simulationen verzichten. Das bedeutet, dass man bspw. trotz nicht vollständiger Systemkenntnis Aussagen über das Verhalten des Systems an einem oder mehreren kritischen Punkt(-en) treffen können möchte oder darüber, wie wahrscheinlich ein System innerhalb eines bestimmten Betriebszustands bleibt. Dazu ist es nötig, Unsicherheiten und ihre Auswirkungen auf das Verhalten des Gesamtsystems zunächst zu beschreiben und anschließend zu quantifizieren.

Kapitel 2 – Uncertainty Quantification definiert zunächst verschiedene Facetten von Unsicherheit. Anschließend werden der grundlegende Aufbau einer modellbasierten Uncertainty Quantification-Simulationsumgebung (UQ) sowie einige klassische Motivationen für das Quantifizieren von Unsicherheiten skizziert. Daraus wird die Problemstellung formal spezifiziert sowie gängige Verfahren zur Quantifizierung von Unsicherheiten vorgestellt.

Kapitel 3 – Dünne Gitter führt das Konzept der dünnen Gitter mithilfe eines Interpolationsbeispiels ein, stellt verschiedene Arten von dünnen Gittern vor und erläutert die grundlegenden Operationen.

Kapitel 4 – Implementierung stellt zunächst das verwendete Dünngitter-Framework SG++ unter besonderer Berücksichtigung der im Rahmen der Arbeit erstellten Basis mit stückweise polynomiellen Basisfunktionen vor. Anschließend wird die für die numerischen Experimente verwendete Bibliothek für UQ eingeführt.

Kapitel 5 – Experimente illustriert mithilfe numerischer Experimente die Wirksamkeit der implementierten Methoden.

Kapitel 6 – Zusammenfassung und Ausblick fasst die erhaltenen Ergebnisse zusammen und schlägt weitere Ansätze vor.

2. Uncertainty Quantification

2.1. Arten von Unsicherheiten

Nach [WHR+05, S. 8] hat eine Unsicherheit Eigenschaften auf drei verschiedenen Ebenen:

1. Den *Ort*, d. h. an welcher Stelle im System sich die Unsicherheit zeigt.
2. Das *Maß*, d. h. wo die Unsicherheit auf einer Skala von "deterministisches Wissen" bis hin zu "völliger Ahnungslosigkeit" einzuordnen ist.
3. Schließlich das *Naturell*, d. h. entsteht die Unsicherheit durch fehlendes Wissen oder durch einen inhärent variablen Prozess.

Das bedeutet, jede einzelne Unsicherheit hat einen Ort, ein Maß und ein Naturell.

Der Ort (1.) der Unsicherheit im System lässt sich nach [WHR+05, S. 9ff] weiter unterscheiden in

- *Kontextunsicherheit*
  Kontextunsicherheit beschreibt die Frage, wo die Systemgrenze genau verläuft, d. h. welche Teile der realen Welt sich innerhalb bzw. außerhalb des Systems befinden sowie die Frage der Vollständigkeit der Darstellung.
- *Modellfehler*
  Der Modellfehler teilt sich in zwei Teile; einerseits eine *strukturelle* Unsicherheit über die Form des Modells an sich und andererseits eine *technische* Unsicherheit, die sich aus der Implementierung des Modells im Rechner ergibt.
- *Eingabefehler*
  Die Unsicherheit der Eingabe betrifft zum einen Unsicherheiten über äußere Antriebe, die das Verhalten des Systems beeinflussen und zum anderen Unsicherheiten über technische Daten des Systems. Zweitere dienen zur Beurteilung der Fähigkeiten eines Referenzsystems. Die Unsicherheit entsteht dabei typischerweise aus einem Wissensdefizit.
- *Parameterunsicherheit*
  Zwischen der Parameterunsicherheit und der strukturellen Unsicherheit des Modells besteht ein Zusammenhang. Es existiert eine Abbildung, die als Funktion der verfügbaren Kalibrierungsdaten und der im Kalibrierungsdatensatz enthaltenen Information auf die optimale Kombination von Modellunsicherheit und Parameterzahl abbildet. Das bedeutet, dass durch Erhöhung der Modellkomplexität und/oder der Parameterzahl zusätzliche Unsicherheit entstehen kann.

- *Unsicherheit der Modellvorhersage*
  Die Unsicherheit der Modellvorhersage beschreibt die Differenz zwischen dem vom Modell vorhergesagten und dem tatsächlichen Wert. Damit ist sie die akkumulierte Unsicherheit, die sich aus den übrigen Unsicherheiten zusammensetzt, die durch das Modell hindurch propagiert wurden.

Das Naturell (3.) der Unsicherheiten unterscheidet zwei Extreme [WHR+05, S. 13]:

- epistemische Fehler
  Diese beschreiben systematische Unsicherheiten. Es existiert eine genaue Lösung, diese ist aufgrund von fehlendem Wissen respektive fehlenden Daten allerdings nicht messbar.
- aleatorische Fehler
  Diese beschreiben stochastische Unsicherheiten. Es existiert aufgrund von intrinsischen Schwankungen beim modellierten (Teil-)Prozess keine genaue Lösung für das Problem.

Diese Unterscheidung ist deshalb nützlich, weil der epistemische Teil durch nicht-physikalische Zufallsvariablen substituiert werden kann. Diese bilden Informationen ab, die durch das Sammeln zusätzlicher Daten oder durch bessere Messmethoden entstehen können und definieren dadurch Korrelationen zwischen verschiedenen Komponenten eines Problems mit gemeinsamen Unsicherheiten. Die Unterscheidung, ob eine Unsicherheit epistemisch oder aleatorisch ist, hängt also nicht nur vom aktuellen Stand der Technik und den verfügbaren Daten ab, sondern auch und v. a. von der praktischen Anforderung an die Simulation, die Modellkomplexität auf ein sinnvolles Maß zu beschränken [KD09, S. 106].

Le Maître und Knio definieren Unsicherheiten ähnlich aber deutlich gröber. Sie orientieren sich dabei insbesondere in der Terminologie deutlich stärker an dem konkreten Einsatzzweck einer modellbasierten numerischen Simulation und kategorisieren Unsicherheiten nur bezüglich des Orts [MK10, S. 1f].

Im Rahmen dieser Arbeit soll v. a. der Einfluss von Eingabeunsicherheiten von beliebigem Maß und Naturell untersucht werden.

2.2. Simulationsumgebung

Eine idealisierte Simulationsumgebung ohne Unsicherheiten lässt sich in drei Schritten zusammensetzen [MK10, S. 3f]:

1. *Fallspezifikation*
   In diesem Schritt werden die Eingabeparameter definiert. Dazu sind alle inneren und äußeren Einflüsse in Form der Geometrie zu spezifizieren, insbesondere auch das Rechengebiet. Rand- und/oder Anfangsbedingungen sowie äußere Antriebe werden definiert. Schließlich werden die Systemeigenschaften ebenso durch physikalische Konstanten beschrieben wie Kalibrierungs- oder Modellierungsdaten, um nicht explizit im Modell aufgelöste Phänomene darzustellen.

2. *Simulation*
   Die Simulation benötigt zunächst einen ersten Diskretisierungsansatz, meist in Form einer Gitterspezifikation und entsprechenden Parametrisierung. Soweit nötig, werden zusätzliche Parameter für die Integration nach der Zeit eingeführt. Daran anschließend wird ein numerisches Lösungsverfahren definiert. Für dieses Lösungsverfahren soll gelten, dass

   a) das ursprüngliche mathematische Modell eine eindeutige Lösung hat,

   b) die Diskretisierung, gesetzt den Fall, sie ist korrekt spezifiziert, ebenfalls eine eindeutige Lösung hat, die gleichzeitig gegen die echte Lösung konvergiert und

   c) dabei hinreichend kleine Diskretisierungsfehler erreichbar sind.

3. *Analyse*
   Im letzten Schritt müssen die aus der Simulation erhaltenen Daten zur Analyse aufbereitet werden.

In Abb. 2.1 ist der beschriebene dreistufige Prozess in Form eines Flussdiagramms visualisiert.

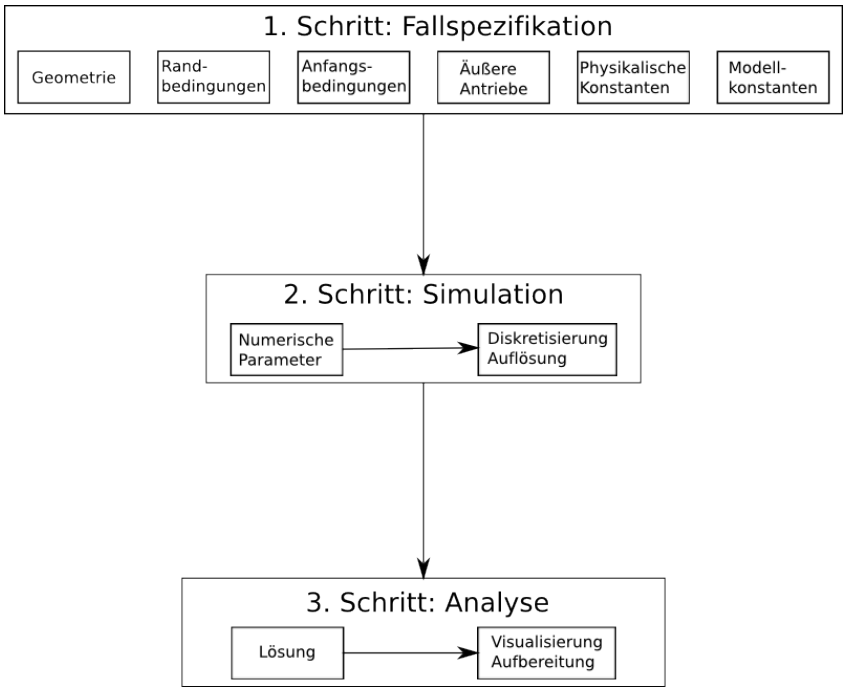

Abbildung 2.1.: Flussdiagramm zum Aufbau einer numerischen Simulation ohne Unsicherheiten nach [MK10, S. 4]

Da in der Praxis häufig Unsicherheiten in den Eingangsdaten vorhanden sind oder sich das modellierte System wegen intrinsischer Schwankungen nicht immer gleich verhält, muss das bestehende Modell erweitert werden, um den Einfluss unsicherer Daten auf das Simulationsergebnis zu untersuchen. Damit wird die Unsicherheit aus den Eingangsdaten mit dem Simulationsergebnis verknüpft, so dass

sich deren Einfluss auf das Ergebnis sowohl charakterisieren als auch quantifizieren lässt [MK10, S. 4f]. Ein Beispiel für eine solche Erweiterung zeigt Abb. 2.2.

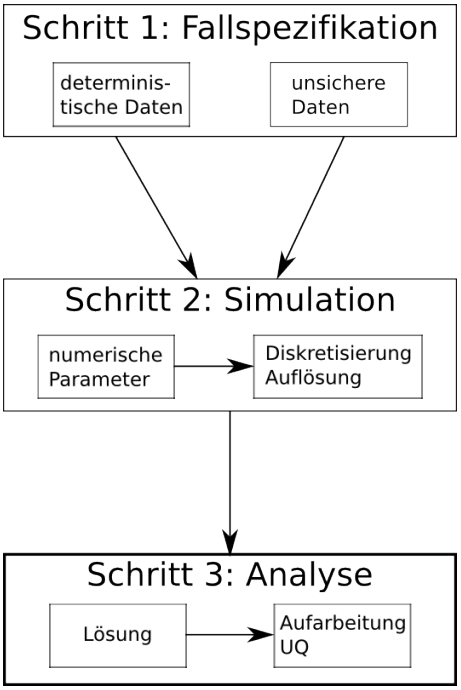

Abbildung 2.2.: Flussdiagramm zum Aufbau einer numerischen Simulation mit Unsicherheiten nach [MK10, S. 5]

2.3. Ziele des Quantifizierens von Unsicherheiten

Eine solche Erweiterung ist nützlich, weil sie u. a. folgende Ergebnisse liefern kann [MK10, S. 5f]:

- *Verifikation der Simulation*
  Durch den Vergleich der Simulationsergebnisse mit realen Versuchsergebnissen kann die Gültigkeit der Simulation überprüft und validiert werden. Dabei ist zu beachten, dass auch die Versuchsergebnisse Unsicherheiten unterliegen, bspw. durch nicht-optimale Systeme oder Messungenauigkeiten. Bei der Validierung muss daher sowohl die rechnerische als auch die experimentelle Ungenauigkeit berücksichtigt werden.

- *Varianzanalyse*
  Der Ausschlag der Systemreaktion um Mittel- oder Sollwert liefert wichtige Informationen zu Entwurf und Optimierung und gleichzeitig Unterstützung bei Entscheidungen. Auf diesem Weg lassen sich die Robustheit der Vorhersage und die Kontrollierbarkeit des Systems beschreiben sowie ein Maß für die Verlässlichkeit der Vorhersagen ableiten.

- *Risikoanalyse*
  Auf Basis der Wahrscheinlichkeitsdichten der Eingangsdaten soll die Wahrscheinlichkeit be-

stimmt werden, dass ein System bestimmte Betriebszustände verlässt oder Schwellwerte an kritischen Punkten überschritten werden. Gleichzeitig können diese Wahrscheinlichkeiten zur Abschätzung von Risiken oder zur Bewertung der Zuverlässigkeit verwendet werden.

- *Sensitivitätsanalyse*
  Wann immer ein System mehreren Unsicherheiten verschiedenen Ursprungs unterliegt, ist ihr jeweiliger Einfluss auf das Verhalten des Gesamtsystems eine entscheidende Frage, um dominante Unsicherheiten überwachen oder eindämmen zu können.

2.4. Allgemeine Problemstellung

Innerhalb des definierten Framework kann die Propagierung der Unsicherheit aus den Eingabeparametern auf das Simulationsergebnis als Bestimmung der funktionalen Abhängigkeit der Lösung von den Zufallsvariablen, die die Eingangsdaten parametrisieren, betrachtet werden [MK10, S. 7]. Formal lässt sich die Problemstellung der Vorwärtspropagation von Unsicherheiten wie folgt formulieren [FDPS14, S. 4]:

Es existiere ein System, das vom Modell $\mathcal{M}$ gesteuert wird. Dieses Modell basiert auf einer endlichen Zahl zufälliger Parameter $\xi = \{\xi_1, \xi_2, \ldots, \xi_k\} \in \Gamma$ mit gegebener Wahrscheinlichkeitsdichtefunktion $p_\xi(\xi)$ und gegebener kumulativer Verteilungsfunktion $F_\xi(\xi)$ sowie einer unbekannten Lösung:

(2.1) $u : \Gamma \mapsto \mathbb{R}$

Gesucht ist also eine Funktion $u(\xi)$ unter dem Modell $\mathcal{M}$, um das probabilistische Verhalten, das durch die unsicheren Parameter ξ in das System eingebracht wird, zu beschreiben. Dabei führt die Kenntnis der Wahrscheinlichkeitsdichte von ξ auf die Wahrscheinlichkeitsdichte von u.

2.5. Verfügbare Methoden

Zur Lösung des zuvor skizzierten Problems existiert eine Reihe von Verfahren. Diese lassen sich in zwei große Gruppen unterteilen, die intrusiven und die nicht-intrusiven Verfahren.

2.5.1. Intrusive Verfahren

Intrusive Verfahren benötigen die Aufstellung und Lösung einer stochastischen Formulierung der ursprünglichen Modellgleichungen.

Ein klassischer Ansatz dafür ist die stochastische Galerkin Expansion, wo die Lösung als Spektralsumme der unsicheren Variablen dargestellt wird [Iac11, S. 6]:

$$u(x,t,\xi) = \sum_{i=0}^{\infty} \underbrace{u_i(x,t)}_{\text{deterministisch}} \overbrace{\psi_i(\xi)}^{\text{stochastisch}}$$

Die $\psi_i(\xi)$ sind bspw. hermite'sche Polynome und bilden eine vollständige Menge orthogonaler Basisfunktionen. Anstelle hermite'scher Polynome können u. a. auch Wavelets verwendet werden. Ist $u(x,t,\xi)$ lösbar, lassen sich Erwartungswert und Varianz direkt berechnen. Eine Möglichkeit zur Berechnung findet sich in [Iac11, S. 10ff].

Die Verwendung einer solchen Expansion verwandelt das ursprüngliche stochastische Problem in ein deterministisches Problem. Dadurch können sich bei der Lösung in der Konvergenzordnung immense Gewinne ergeben.

Im Gegenzug muss dafür ein hoher Preis gezahlt werden. Durch die Erweiterung des Modells erhöht sich die Komplexität des Problems. Außerdem muss eine möglicherweise bereits bestehende Implementierung des ursprünglichen Modells geändert werden, weil sich die Modellgleichungen ändern.

2.5.2. Nicht-intrusive Verfahren

Im Gegensatz zu intrusiven Verfahren basieren nicht-intrusive Verfahren auf einer (möglicherweise großen) Zahl einzelner Realisierungen der Zufallsvariablen, um die Reaktion des stochastischen Modells auf die unsicheren Eingangsdaten zu modellieren. D. h. das Problem wird nicht verändert, sondern in unterschiedlichen Realisierungen der Menge der unsicheren Parameter ξ untersucht.

Der klassische nicht-intrusive Ansatz ist das Monte Carlo Sampling (MC). Dabei wird für jedes Element ξ_i aus der Menge der unsicheren Eingangsdaten ξ nach der Wahrscheinlichkeitsdichtefunktion $p_\xi(\xi)$ eine einzelne Realisation gewählt. Der MC Ansatz ist unabhängig von der Dimension des stochastischen Raums und einfach zu implementieren. Zusätzlich konvergiert die Lösung des MC Ansatzes garantiert gegen die echte Lösung [FDPS14, S. 2].

Der Hauptnachteil dieses Ansatzes ist die niedrige Konvergenzordnung von $\mathcal{O}\left(\frac{1}{\sqrt{n}}\right)$, wobei n für die Anzahl der gezogenen Samples steht. Die Konvergenzordnung zeigt die Dimensionsunabhängigkeit des Ansatzes, da die Konvergenzrate lediglich von der Zahl der Realisationen abhängt.

Die Konvergenzordnung kann durch den Einsatz spezieller nicht-zufälliger Sequenzen verbessert werden, sog. Quasi Monte Carlo Verfahren (QMC). Dabei werden korrelierte Punkt-Sequenzen verwendet, die eine höhere Gleichmäßigkeit mitbringen.

Das bedeutet, dass das Gebiet Ω, in dem die Samples generiert werden, durch QMC Verfahren deutlich harmonischer abgedeckt wird als durch das Verwenden von echtem MC. Bei der Verwendung von MC Sampling ist jedes einzelne Sample unabhängig von allen übrigen. Dadurch entsteht an einigen Stellen eine Anhäufung vieler Samples, an anderen Stellen entsteht eine entsprechend geringe Abdeckung. Im Gegensatz dazu sind bei QMC Verfahren die Samples voneinander abhängig, so dass dieser Effekt vermieden wird. Dies zeigt sich bspw. in [Caf98, S. 24]. Dadurch konvergieren diese Verfahren ungefähr mit $\mathcal{O}\left(\frac{(\log(n))^k}{n}\right), k = konst$ [Caf98, S. 2f]. Im Rahmen dieser Arbeit werden zur Generierung von QMC Samples Sobol-Sequenzen verwendet. Eine Einführung zu Sobol-Sequenzen findet sich in [Sob67].

Im Zusammenhang mit Simulationen kann allerdings bereits ein einziger Simulationsaufruf viel Rechenzeit in Anspruch nehmen. Um Aufrufe einzusparen, bietet es sich daher für glatte Abbildungen $u(\xi)$ an, nicht nur die Menge der einzelnen Auswertungspunkte $\{f(\xi_j)\},\ j = \{1, \ldots, n\}$ zur Approximation der Lösung zu verwenden, sondern zusätzlich zwischen den einzelnen Auswertungspunkten zu approximieren.

Eine Möglichkeit dazu liefert die nicht-intrusive verallgemeinerte polynomielle Chaosentwicklung (gPC), vorgestellt in [XK02] und [XK03]. Dabei werden Parameter-Verteilungen mithilfe des Askey Schemas optimierten orthogonalen Basen zugeordnet. Die Parameter werden bspw. durch Orthogonalprojektion berechnet, so dass ein potentiell hochdimensionales Quadraturproblem entsteht, das numerisch gelöst werden muss, bspw. durch Sampling oder Ansätze mit dünnen Gittern. Dabei ist zu beachten, dass die verwendeten Basisfunktionen global definiert sind, so dass bei scharfen Übergängen, d. h. bei Sprungstellen in der Zielfunktion, das Gibbs'sche Phänomen zu beobachten ist. Das bedeutet, dass in der Nähe dieser Sprünge in der Approximation Überschwingungen auftreten, die sich auch durch das Hinzunehmen endlich vieler weiterer Stützstellen nicht eliminieren lassen. Die Methode lässt sich erweitern, um die beschriebenen Effekte zu mildern, überblicksweise werden entsprechende Konzepte z. B. in [MZ09, S. 3085f] vorgestellt.

Im Rahmen dieser Arbeit soll die stochastische Kollokation mithilfe von dünnen Gittern erfolgen, der sog. "Conventional Sparse Grid Collocation" (CSGC) und der "Adaptive Sparse Grid Collocation" (ASGC). Dünne Gitter sind konstruktionsbedingt geeignet, den "Fluch der Dimensionalität" zumindest teilweise zu überwinden und leiden wegen der Verwendung lokaler Basisfunktionen nicht unter dem Gibbs'schen Phänomen. In einer Reihe von Benchmark-Versuchen mit bekannten analytischen Lösungen hat ein solcher Ansatz vergleichbare oder bessere Ergebnisse gezeigt als gPC - Ansätze [EB09].

Im Folgenden sollen zunächst dünne Gitter eingeführt und anschließend das verwendete Dünngitter-Framework SG++ vorgestellt werden.

3. Dünne Gitter

3.1. Hintergrund und Einführung

3.1.1. Grundlegendes

Die vorgeschlagene Methode zur Quantifizierung von Unsicherheiten führt zu einem hochdimensionalen (Quadratur-)Problem, weil jeder zusätzliche unsichere Parameter die Dimension des Problems um eins erhöht. Dieser sog. "Fluch der Dimensionalität", d. h. der exponentielle Zusammenhang zwischen nötigem Rechenaufwand und der Dimension des Problems, kann mithilfe von dünnen Gittern teilweise überwunden werden.

Bei einem Vollgitter mit Dimension $d = 1$ und m Gitterpunkten bedeutet die Hinzunahme von d weiteren Dimensionen unter Beibehaltung der Maschenweite h, dass m^d Gitterpunkte entstehen. Bei einem dünnen Gitter werden dagegen solche Gitterpunkte eines Vollgitters ausgelassen, die zur Gesamtlösung nur einen geringen Beitrag leisten, so dass die Approximation nur um einen logarithmischen Faktor schlechter ist als die Vollgitterlösung [Pfl10, S. 5].

Im Rahmen dieser Arbeit werden Gitter auf dem sog. Unit Hypercube definiert, d. h. es wird o. B. d. A. das Gebiet $\Omega = [0, 1]^d$ mit dem vorgestellten Dünngitter-Ansatz diskretisiert.

Die hier gegebene Einführung beschränkt sich auf die wichtigsten Grundlagen und orientiert sich dabei an [Pfl10, Kap. 2]. Für eine detailliertere Einführung wird an die vorgenannte Stelle sowie [Feu10, Kap. 2] verwiesen.

Im 1D unterscheiden sich dünne Gitter nicht von vollen Gittern. Der Rahmen dieser Arbeit beschränkt sich auf dyadische Gittertypen, d. h. für die Vollgitter-Maschenweite gilt für Level $l \geq 0 : \ h_l := 2^{-l}$ [Feu10, S. 5].

Als einführendes Beispiel wird die Interpolation im 1D für Gebiete $\Omega = [0, 1]$ gewählt. Für die Gebiete gilt zunächst, dass die Funktion, die interpoliert werden soll, an den Randpunkten den Wert 0 hat, d. h. $\partial\Omega = 0$.

Für dünne Gitter im 1D ergibt sich eine hierarchische Struktur:
Beginnend bei Level 1 und einem Gitterpunkt bei 0.5 führt die Erhöhung des Levels l um 1 zu 2^l zusätzlichen Gitterpunkten, die, weil dyadische Gitter verwendet werden, jeweils genau in der Mitte der Maschen von Level l angeordnet sind.

Um die Knoten zu identifizieren, wird ein (Level, Index)-Tupel verwendet. Dabei wird der Index von 1 beginnend auf jedem Level hochgezählt. Da alle Knoten des Levels $l - 1$ auch Knoten von Level

l sind, ist die Namensgebung noch nicht eindeutig. Um dies zu beheben, wird das (Level, Index)-Tupel verwendet, das den minimalen Level hat. Das bedeutet, dass alle Indizes ungerade sind. Das Namensschema ist in folgender Abb. 3.1 anschaulich dargestellt.

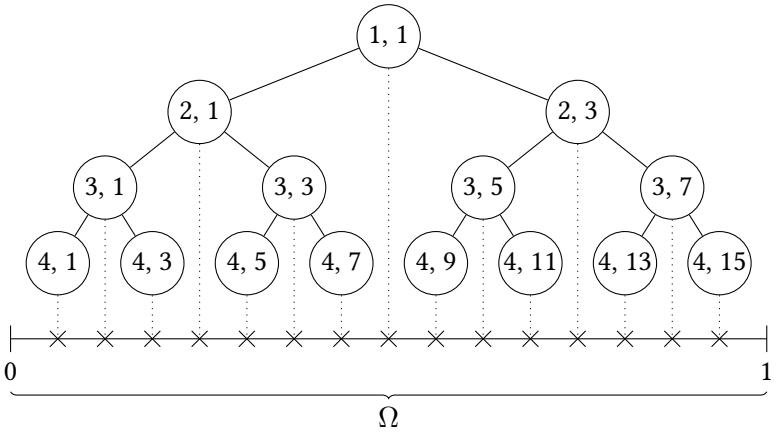

Abbildung 3.1.: Level-Index-Baum für ein randloses dünnes Gitter mit Level 4. Jedes Kreuz markiert die Position eines Gitterpunkts

Dieses Schema führt zur Indexmenge I_l

$$I_l := i \in \mathbb{N};\ 1 \leq i \leq 2^l - 1,\ i = 1 \mod 2. \tag{3.1}$$

Für das Interpolationsbeispiel wurden somit Stützstellen definiert, an denen die Funktion ausgewertet wird. Nun fehlen noch geeignete Basisfunktionen, die die Funktion zwischen den Schnittstellen interpolieren sollen. Zur Einführung wird die hierarchische Hut-Funktion angesetzt:

$$\begin{aligned} \varphi(x) &= \max(1 - |x|, 0) \\ \varphi_{l,i}(x) &:= \varphi(2^l x - i) \end{aligned} \tag{3.2}$$

Daraus ergeben sich hierarchische Unterräume W_l auf jedem Level l mit nicht-überlappenden Basisfunktionen, wenn man nur die Indizes $i \in I_l$ berücksichtigt.

$$W_l := \text{span}\{\varphi_{l,i}(x)\} \text{ mit } i \in I_l \tag{3.3}$$

Der Raum der stückweise linearen Basisfunktionen V_N ergibt sich aus der direkten Summe über alle Unterräume W_l mit $1 \leq l \leq N$.

Um eine Interpolation $f_N(x)$ in V_N durchzuführen, muss die zu einer Stützstelle gehörende Basisfunktion mit dem hierarchischen Überschuss $v_{l,i}$ zwischen dem realen Funktionswert $u(x_{l,i})$ und dem Wert der bisherigen Dünngitterapproximation $f_{N-1}(x_{l,i})$ skaliert werden.

$$\begin{aligned} v_{l,i} &= u(x_{l,i}) - f_{N-1}(x_{l,i}) \\ \rightsquigarrow f_N(x) &= \sum_{l \leq n, i \in I_l} v_{l,i} \cdot \varphi_{l,i}(x) \end{aligned} \tag{3.4}$$

Das bisherige Ergebnis der Einführung nach [Pfl10, Kap. 2] zeigt Abb. 3.2 auf der nächsten Seite.

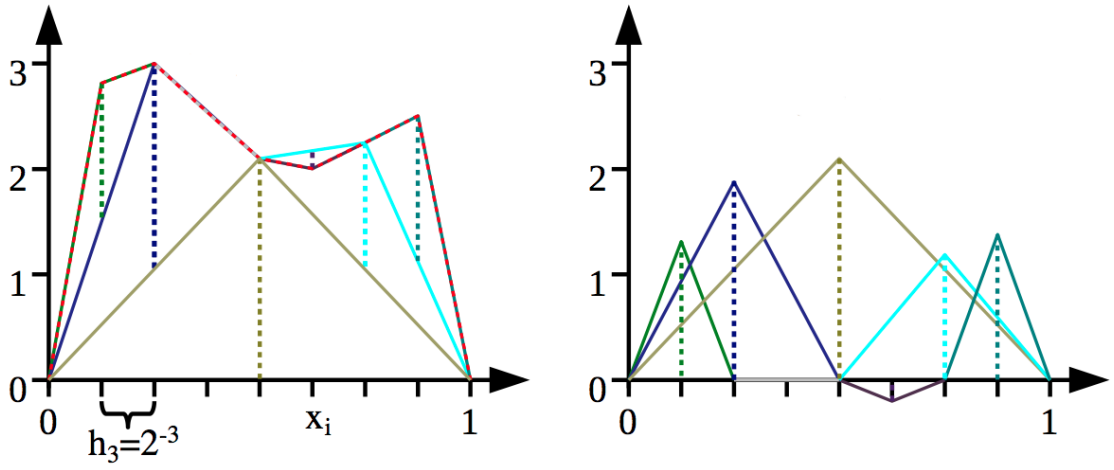

Abbildung 3.2.: Interpolationsbeispiel 1D, entnommen aus [Pfl10, S. 9]. Links sieht man in rot das Resultat $f_N(x)$ aus Gleichung (3.4). Rechts sind die hierarchisch angeordneten und skalierten Hut-Funktionen $v_{l,i} \cdot \varphi_{l,i}$ einzeln dargestellt. Die hierarchischen Überschüsse $v_{l,i}$ sind in Form der gepunkteten Linien visualisiert.

3.1.2. Übergang zu höheren Dimensionen

Der Übergang aus dem Eindimensionalen in die Mehrdimensionalität gelingt mithilfe eines Tensorproduktansatzes:

$$\varphi_{\vec{l},\vec{i}} := \prod_{j=1}^{d} \varphi_{l_j,i_j}(x_j) \tag{3.5}$$

Auf die erneute Herleitung von $W_{\vec{l}}, V_N$ und $f_N(\vec{x})$ wird an dieser Stelle verzichtet und stattdessen auf [Pfl10, S. 10] verwiesen.

Dieser Ansatz alleine genügt allerdings nicht, um mehrdimensionale dünne Gitter zu erhalten: Im 1D unterscheiden sich dünne und volle Gitter nicht, so dass der in Gleichung (3.5) beschriebene Ansatz mehrdimensionale Vollgitter liefert. Für die so erzeugten Vollgitter gilt, dass die Vollgitterapproximation $g_N(x) \in V_N$ für eine hinreichend glatte zu interpolierende Funktion $u(x)$ eine asymptotische Fehlerschranke aufweist:

$$\|u(\vec{x}) - g_N(\vec{x})\|_{L_2} \in \mathcal{O}(h_N^2) \tag{3.6}$$

Dazu sind $\mathcal{O}(h_N^{-d}) = \mathcal{O}(2^{Nd})$ Funktionsauswertungen nötig [Pfl10, S. 10].

Der Übergang vom vollen zum dünnen Gitter gelingt nun durch das Auslassen derjenigen Unterräume, deren Beitrag zur Vollgitter-Lösung am geringsten ist [BG04]:

$$
\begin{aligned}
V_N^{(1)} &:= \bigoplus_{|\vec{l}|\leq N+d-1} W_{\vec{l}} \\
\rightsquigarrow f_N(\vec{x}) &= \sum_{|\vec{l}|\leq N+d-1,\, \vec{i}\in I_{\vec{l}}} v_{\vec{l},\vec{i}}\varphi_{\vec{l},\vec{i}}(\vec{x})
\end{aligned} \tag{3.7}
$$

Die Bedingung $\left|\vec{l}\right| \leq N + d - 1$ wird mit der Hamming-Distanz vermessen. Daraus ergibt sich für 2D und Level $l = 3$ das in Abb. 3.3 dargestellte Teilraumschema. Die Teilräume sind das jeweilige Ergebnis des kartesischen Kreuzprodukts der am äußeren Rand dargestellten Basisfunktionen.

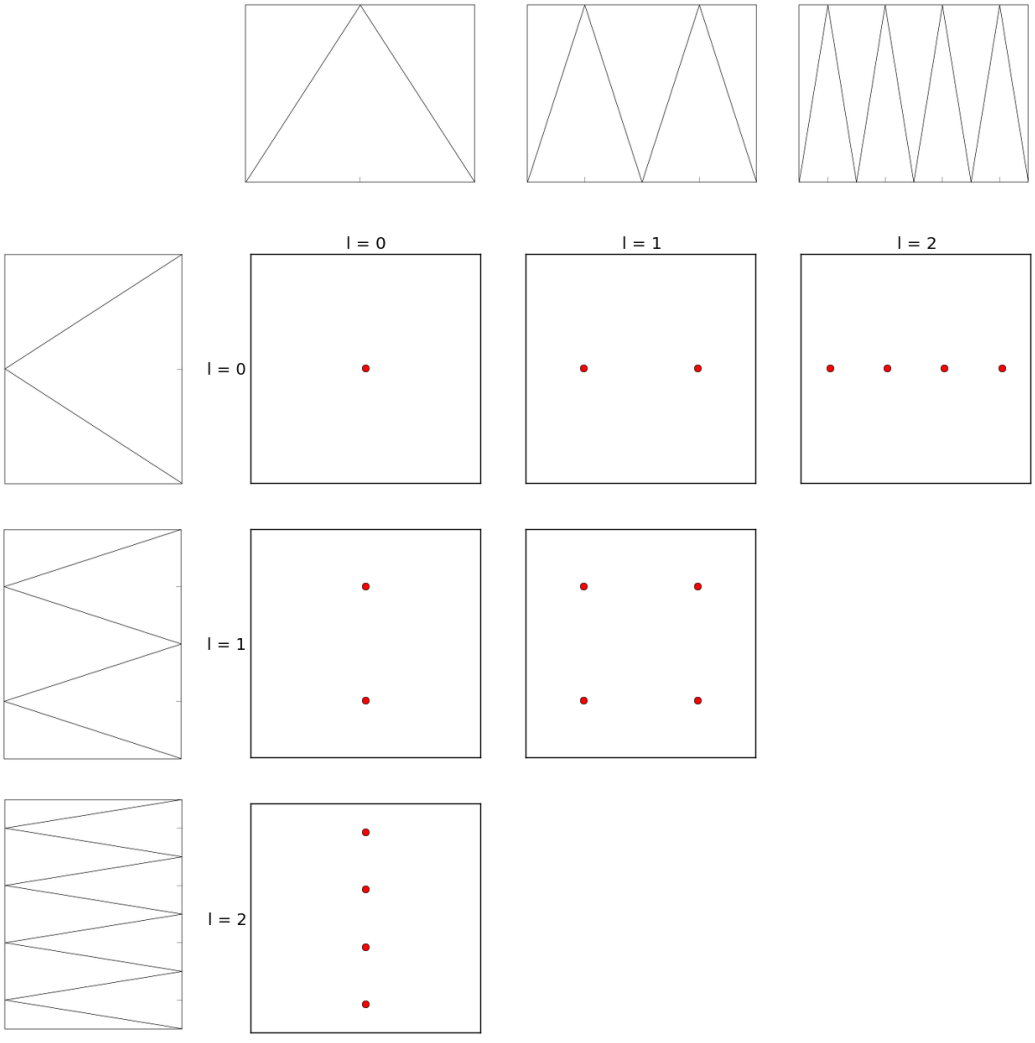

Abbildung 3.3.: Teilraumschema dünnes Gitter: Dimension $d = 2$, Level $l = 3$, kein Rand

Dadurch reduziert sich die asymptotische Genauigkeit von $\mathcal{O}(h_N^2)$ auf

(3.8) $\mathcal{O}(h_N^2(\log(h_N^{-1}))^{d-1})$,

gleichzeitig reduziert sich die Anzahl der benötigten Gitterpunkte von $\mathcal{O}(h_N^{-d})$ auf

(3.9) $\mathcal{O}(h_N^{-1}(\log(h_N^{-1}))^{d-1})$.

Das bedeutet, dass die Zahl der benötigten Gitterpunkte deutlich abfällt ohne signifikanten Verlust der Approximationsgenauigkeit.

3.1.3. Randpunkte

Um den bisherigen Ansatz auf Funktionen zu erweitern, auf deren Rand gilt, dass $\partial\Omega \neq 0$, wird ein zusätzlicher Level $l = 0$ mit zwei überlappenden Basisfunktionen $\varphi_{0,0}$ und $\varphi_{0,1}$ eingeführt. Dadurch erweitert sich das Teilraumschema wie in Abb. 3.4 dargestellt.

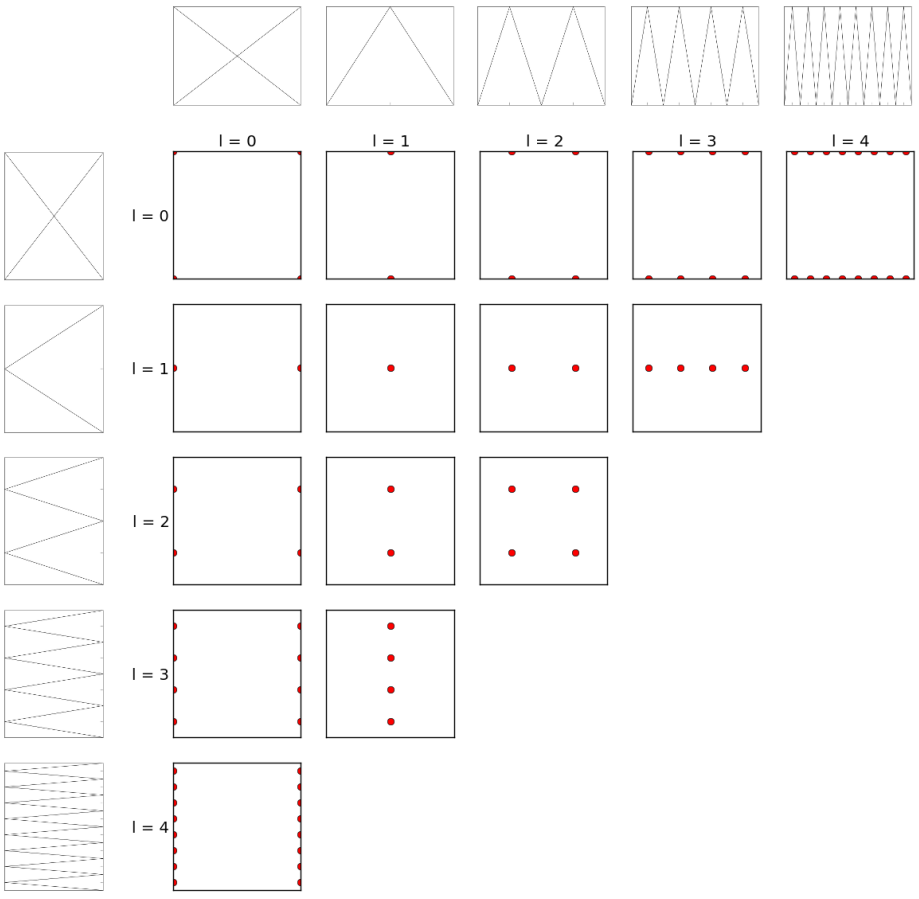

Abbildung 3.4.: Teilraumschema dünnes Gitter: Dimension $d = 2$, Level $l = 4$, regulärer Rand

Dabei entsteht das Problem, dass nicht nur einige Punkte auf dem Rand liegen, sondern fast alle. Im 2D liegen bereits doppelt so viele Punkte auf dem Rand wie auf den Hauptachsen des Gitters. In höheren Dimensionen vergrößert sich diese Problematik, weil Randpunkte eines dünnen Gitters der Dimension d nichts anderes sind als ein dünnes Gitter der Dimension $d-1$ [Feu10, S. 10].

Um dieses Problem zu lindern, erzwingt man am Rand die gleiche Diskretisierung wie auf den Hauptachsen. Diese sog. Trapezrand-Gitter kann man bspw. dadurch konstruieren, dass man die Basisfunktionen des vormaligen Level 0 nicht als 0ten Level hinzufügt, sondern mit der Basis aus Level 1 überlagert. Für Trapezrand-Gitter ergibt sich damit das in Abb. 3.5 dargestellte Teilraumschema.

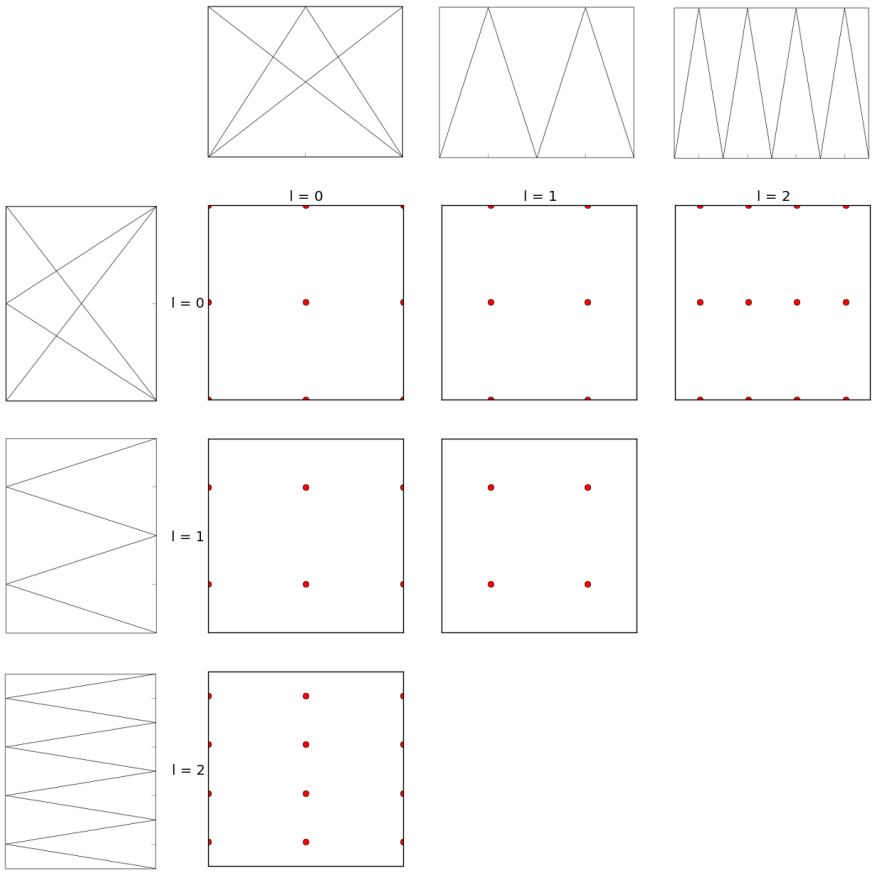

Abbildung 3.5.: Teilraumschema dünnes Gitter: Dimension $d = 2$, Level $l = 3$, Trapezrand

3.1.4. Polynomielle Basisfunktionen

Gibt man den Approximationsfehler ε nicht in Abhängigkeit der Maschenweite, sondern in Abhängigkeit der Zahl der Gitterpunkte m an, liegt er nach [Bun98, S. 43] bei

$$\varepsilon_{L_2}(m) = \mathcal{O}(m^{-2} \cdot |\log_2(m)|^{3(d-1)}). \tag{3.10}$$

Für hinreichend glatte Funktionen lässt sich die Konvergenzgeschwindigkeit erhöhen, indem man statt stückweise linearen Basisfunktionen Basisfunktionen mit mehr Stützstellen zulässt und bspw. stückweise polynomielle Basisfunktionen ansetzt. Dazu wird eine Lagrange-Basis über k Stützstellen verwendet, mit

$$k = \min\{\text{gewünschter Polynomgrad: } p+1,\ \text{max mögliche Stützstellen aus Gitterlevel: } l+2\}. \tag{3.11}$$

Die zweite Einschränkung erklärt sich wegen des hierarchischen Ansatzes. Der maximale Polynomgrad p ist auf Level l durch $p \leq l + 1$ beschränkt, weil nur der Knoten selbst und seine hierarchischen Vorgänger als Stützstellen in Frage kommen. Dies veranschaulicht Abb. 3.6 beispielhaft für den Knoten (4, 11).

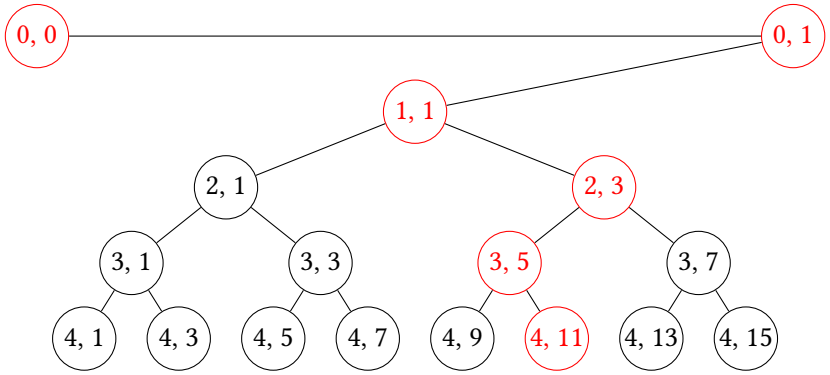

Abbildung 3.6.: Menge potentieller Stützstellen für Knoten $(4, 11)$

Als Stützstellen für den Ansatz der Lagrange-Basis existieren damit zwei Gitterpunkte auf Level $l = 0$ und für jedes weitere Level bis $l = l_{\max}$ genau ein Gitterpunkt. Die Funktionswerte der Stützstellen werden wie folgt angesetzt:

$$f(x_{l,i}) = \begin{cases} 1 & \text{wenn } l = l_{\max} \\ 0 & \text{wenn } l < l_{\max} \end{cases} \tag{3.12}$$

Dabei ist zu beachten, dass es keine Rolle spielt, ob das Gitter einen Rand hat oder nicht, weil auf Level 0 keine polynomielle Basisfunktion von höherem Polynomgrad als 1 möglich ist, was der stückweise linearen Basis entspricht. Somit ist an den Stützstellen am Rand für alle Level $l > 0$ grundsätzlich 0 anzusetzen, was sich mit randlosen Gittern verträgt, weil dort per definitionem gilt, dass $\partial\Omega = 0$.

Wählt man einen Polynomgrad, der nicht dem maximal möglichen entspricht, sind nicht die p hierarchischen Vorgänger des Knotens nach Level geordnet als Stützstellen zu verwenden, sondern die p räumlich nächstgelegenen Gitterpunkte, um möglichst lokal zu approximieren. Nicht maximal möglicher Polynomgrad bedeutet anschaulich, dass in dem in Abb. 3.6 dargestellten Fall nicht alle roten Knoten als Stützstelle verwendet werden.

Dadurch kann der Approximationsfehler für hinreichend glatte Funktionen weiter reduziert werden, nach [Bun98, S. 78] auf:

$$\varepsilon_{L_2}^{(p)}(m) = \mathcal{O}(m^{-p+1} \cdot |\log_2(m)|^{(p+2)(d-1)}) \tag{3.13}$$

3.2. Operationen

3.2.1. Hierarchisierung

Die hierarchischen Überschüsse, die zum Skalieren der Basisfunktionen verwendet werden, werden bei der *Hierarchisierung* aus den Funktionswerten errechnet.

Das dünne Gitter wird nach dem *unidirektionalen Prinzip* hierarchisiert, d. h., die Tensorprodukt-Struktur des dünnen Gitters wird ausgenutzt, indem ein linearer Operator für d Dimensionen auf mehrere eindimensionale lineare Operatoren heruntergebrochen wird. Der eindimensionale Operator wird nacheinander für alle Dimensionen angewandt und rechnet dabei in jedem Schritt mit den aktualisierten Werten aus den vorherigen Schritten [Pfl10, S. 28].

Zur Durchführung der Hierarchisierungsoperation werden im Folgenden zwei Verfahren vorgeschlagen. Stückweise polynomielle Basisfunktionen ab Polynomgrad $p = 2$ können auf beide Arten hierarchisiert werden, während stückweise lineare Basisfunktionen wegen fehlender Vorgänger-Polynomgrade nicht nach Variante 2 hierarchisierbar sind.

Variante 1: *Direkte Hierarchisierung* als Differenz zwischen bisherigem Dünngitter-Interpolant $f_{N-1}(x_{l,i})$ und dem Funktionswert $u(x_{l,i})$ am Knoten.

$$v^{(p)}(x_{l,i}) = u(x_{l,i}) - f_{N-1}^{(p)}(x_{l,i}) \tag{3.14}$$

Variante 2: *Rekursive Hierarchisierung* aus den Überschüssen des vorherigen Polynomgrads $p-1$ des hierarchischen Vorgängerknotens $v^{(p-1)}(x_m)$ und des aktuellen Knotens $v^{(p-1)}(x_n)$ sowie einem zusätzlichen Faktor $\alpha(x_0, \ldots, x_{p+1})$, der von den relativen Positionen der Vorgängerknoten des aktuellen Knotens $x_n = x_{l,i}$ abhängt, vgl. [Bun98, S. 58].

$$v^{(p)}(x_{l,i}) = v^{(p-1)}(x_m) - \alpha(x_0, \ldots, x_{p+1}) \cdot v^{(p-1)}(x_n) \tag{3.15}$$

Dabei stehen die Knoten $x_0, \ldots, x_{p+1}$ für die $p+1$ hierarchischen Vorgänger inklusive des Knotens selbst, geordnet nach ihrer Position entlang der Zahlengerade, so dass

$$x_0 < x_1 < \ldots < x_{p+1}. \tag{3.16}$$

Der aktuelle Knoten wird mit x_n bezeichnet und sein hierarchischer Vorgänger mit x_m. Die α-Faktoren berechnen sich damit wie folgt:

$$\alpha_{l,i} = \alpha(x_0, \ldots, x_{p+1}) := \frac{x_n - x_m}{x_{p+1} - x_n} \prod_{\substack{k=0 \\ k \neq m,\, k \neq n}}^{p} \frac{x_m - x_k}{x_n - x_k} \tag{3.17}$$

Für stückweise polynomielle Basisfunktionen ist zu beachten, dass das Gitter für jeden Polynomgrad p von $p = 1$ bis hin zu $p = p_{\max}$ erneut hierarchisiert werden muss. Dabei entfallen in jedem Schritt jeweils die $(p-1)$ ersten Level, weil diese zu wenig Stützstellen für Basisfunktionen von Polynomgrad p bieten. Der erste Hierarchisierungsschritt einer Hierarchisierung für stückweise polynomielle Basisfunktionen ist eine Hierarchisierung für stückweise lineare Basisfunktionen.

Alternativ zu dem in [Bun98, S. 58f] vorgeschlagenen Beweis lassen sich die $\alpha_{l,i}$ auch als dividierte Differenzen und damit als Newton-Basis interpretieren. Das ist nützlich, weil nach dem bisher vorgeschlagenen Beweis zwar die Korrektheit der $\alpha_{l,i}$ bewiesen ist, aber nur wenige Eigenschaften bekannt sind, während für eine Newton-Basis sehr viele Eigenschaften bekannt sind.

BEWEIS $\alpha_{l,i}$ sind eine Newton-Basis
Es sei $p_{i-k,\dots,i}$ das Interpolationspolynom in den Stützstellen $x_{i-k}, x_{i-k+1}, \dots, x_i$ und $\Phi_{i-k,\dots,i}$ das zugehörige Newton'sche Basispolynom mit

$$\Phi_{i-k,\dots,i} = \prod_{j=i-k}^{i} (x - x_j). \tag{3.18}$$

Dann gilt

$$p_{i-k,\dots,i} := p_{i-k,\dots,i-1} + [x_{i-k}, \dots, x_i]\Phi_{i-k,\dots,i-1} \tag{3.19}$$

und

$$p_{i-k,\dots,i} = p_{i-k+1,\dots,i} + [x_{i-k}, \dots, x_i]\Phi_{i-k+1,\dots,i}. \tag{3.20}$$

Die bisherige Interpolation $f_N^{(p)}(x_m)$ (d. h. der Wert der Interpolation von Grad p an der Stelle x_m) ohne den Knoten x_m selbst lautet in der Schreibweise von Gleichung (3.19):

$$f_N^{(p)}(x_m) = p_{m-p-1,\dots,m-1}(x_m) \tag{3.21}$$

Der Überschuss am Punkt x_m für Grad p berechnet sich nach Gleichung (3.14) auf Seite 28 wie folgt:

$$\begin{aligned} v^{(p)}(x_m) &= u(x_m) - f_N^{(p)}(x_m) \\ &= p_{m-p-1,m-p,\dots,m}(x_m) - p_{m-p-1,\dots,m-1}(x_m) \end{aligned} \tag{3.22}$$

Somit folgt nach Gleichung (3.19)

$$v^{(p)}(x_m) = [x_{m-p-1}, \dots, x_m]\Phi_{m-p-1,\dots,m-1}(x_m) \tag{3.23}$$

Damit können alle nötigen Parameter für die Rekursionsformel der dividierten Differenzen hergeleitet und entsprechend eingesetzt werden.

$$\begin{aligned} f_N^{(p)}(x_m) &= p_{m-p-1,m-p,\dots,m-1}(x_m) \\ &\overset{(3.20)}{=} \underbrace{p_{m-p,\dots,m-1}(x_m)}_{f_N^{(p-1)}(x_m)} + \underbrace{[x_{m-p-1}, \dots, x_{m-1}]}_{\frac{v^{(p-1)}(x_{m-1})}{\Phi_{m-p-1,\dots,m-2}(x_{m-1})}} \Phi_{m-p,\dots,m-1}(x_m) \\ &\overset{(3.21)}{\overset{(3.23)}{=}} f_N^{(p-1)}(x_m) + v^{(p-1)}(x_{m-1}) \underbrace{\frac{\Phi_{m-p,\dots,m-1}(x_m)}{\Phi_{m-p-1,\dots,m-2}(x_{m-1})}}_{:=\alpha_{m,p}} \\ &= f_N^{(p-1)}(x_m) + v^{(p-1)}(x_{m-1}) \cdot \alpha_{m,p} \end{aligned} \tag{3.24}$$

q. e. d.

BEMERKUNG Der vorgestellte Beweis orientiert sich sehr deutlich an [Bun98, Lemma 4.3]. Da die $\alpha_{l,i}$-Koeffizienten eine Newton-Basis sind, kann die Gleichheit alternativ auch direkt aus der rekursiven Formel der dividierten Differenzen gezeigt werden.

Die rekursive Formel für dividierte Differenzen lautet

$$[x_{m-p-1,\ldots,x_m}] = \frac{[x_{m-p},\ldots,x_m] - [x_{m-p-1},\ldots,x_{m-2}]}{x_m - x_{m-p-1}}. \tag{3.25}$$

Zusätzlich werden folgende Abkürzungen eingeführt:

$$\begin{aligned} d_m^{(p)} &:= [x_{m-p-1,\ldots,x_m}] \\ \Phi_m^{(p)} &:= \Phi_{m-p-1,\ldots,m-1}(x_m) \end{aligned} \tag{3.26}$$

Beim Übertragen Schreibweise von Gleichung (3.26) auf Gleichung (3.23) zur Berechnung der hierarchischen Überschüsse, ergibt sich

$$v^{(p)}(x_m) = d_m^{(p)} \cdot \Phi_m^{(p)}. \tag{3.27}$$

Nun werden für $d_m^{(p)}$ die rekursive Formel der dividierten Differenzen aus Gleichung (3.25) angesetzt, für die Ergebnisse die umgeformte Gleichung (3.23) eingesetzt sowie der Nenner $\Phi_m^{(p-1)}$ zugeordnet.

$$\begin{aligned} v^{(p)}(x_m) &= d_m^{(p)} \cdot \Phi_m^{(p)} \\ &\overset{(3.25)}{=} \left(d_m^{(p-1)} - d_{m-1}^{(p-1)}\right) \frac{\overbrace{\Phi_m^{(p-1)}}}{\Phi_m^{(p)}} \\ &\overset{(3.23)}{=} \left(\frac{v_m^{(p-1)}}{\Phi_m^{(p-1)}} - \frac{v_{m-1}^{(p-1)}}{\Phi_{m-1}^{(p-1)}}\right) \Phi_m^{(p-1)} \\ &= v_m^{(p-1)} - v_{m-1}^{(p-1)} \cdot \frac{\Phi_m^{(p-1)}}{\Phi_{m-1}^{(p-1)}} \end{aligned} \tag{3.28}$$

3.2.2. Evaluation

Zur Evaluation der Dünngitterapproximation für einen beliebigen Punkt $\vec{x}$ innerhalb des Unit-Hypercube $[0,1]^d$ müssen in jeder Dimension d die beitragenden Basisfunktionen gefunden und an der Stelle $\vec{x}_d$ ausgewertet werden. Die Ergebnisse der Auswertung werden mit dem zur Basisfunktion gehörenden hierarchischen Überschuss skaliert und summiert.

Das Finden der beitragenden Basisfunktionen kann aufgrund der levelweise nicht-überlappenden Basisfunktionen über einfache Indexberechnungen erfolgen. Die Basisfunktionen der Level $l = 0, 1$ sind grundsätzlich beteiligt, weil sie das gesamte Gebiet Ω überspannen. Für alle weiteren Level gilt,

dass die Komponente entweder größer, kleiner oder gleich dem Mittelpunkt des vorherigen Levels ist. Dann ist die nächste beitragende Basisfunktion:

$$\text{(3.29)}\quad \text{nächste beitragende Basisfunktion} = \begin{cases} \text{linker Kindknoten} & \text{wenn } \vec{x}_d < 0.5 \cdot x_{l,i} \\ \text{rechter Kindknoten} & \text{wenn } \vec{x}_d > 0.5 \cdot x_{l,i} \\ \text{beide leisten keinen Beitrag} & \text{wenn } \vec{x}_d = 0.5 \cdot x_{l,i} \end{cases}$$

Falls Fall 3 eintritt, liegt die Komponente auf einem Gitterpunkt, an dem genau interpoliert wird. Damit ist definitionsgemäß in allen Leveln l mit $l > l_{\text{aktuell}}$ der Wert der jeweiligen Basisfunktion $\varphi_{l,i} = 0$.

Für ein zweidimensionales Gitter mit Level $l = 3$ ist das Finden der beitragenden Basisfunktionen in Abb. 3.7 dargestellt.

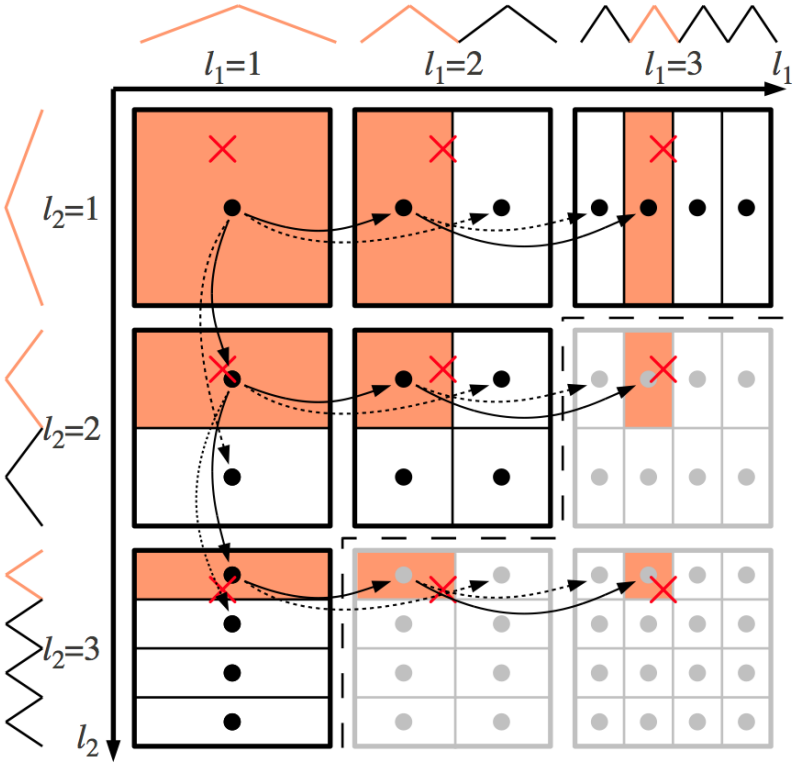

Abbildung 3.7.: Das Finden beitragender Basisfunktionen für ein randloses Gitter von Dimension $d = 2$ und Level $l = 3$, entnommen aus [Pfl14, S. 123]. Das rote Kreuz stellt den Punkt dar, für den die beitragenden Basisfunktionen gesucht werden sollen, die Pfeile die in Frage kommenden Basisfunktionen der Kindknoten.

3.2.3. Quadratur

Zur Quadratur wird der folgende Ansatz herangezogen:

$$\int f_N(x)dx = \sum_{j=1}^{|V|} \left(v_j \cdot \left(\prod_{k=1}^{|D|} \int \varphi_{l'(j,k),i'(j,k)}(x)dx \right) \right) \tag{3.30}$$

mit l', i' als Funktionen, die zu Knoten j in Dimension k das zugehörige Level bzw. den zugehörigen Index berechnen. Für stückweise lineare Basisfunktionen, d. h. Hut-Funktionen, berechnet sich das Integral sehr leicht, weil hier lediglich Dreiecksflächen zu berechnen sind.

$$\int \varphi_{i,j}(x)dx = 2^{\|l\|_1} \overset{1D}{=} 2^{-l} \tag{3.31}$$

Für stückweise polynomielle Basisfunktionen gibt es zwei Möglichkeiten. Die erste ist der klassische Ansatz, die vorhandenen Basispolynome zu integrieren. Dies ist problemlos direkt möglich, da

$$\int a_n \cdot x^n dx = \frac{a_n}{n+1} \cdot x^{n+1} + c. \tag{3.32}$$

Zusätzlich sind die Integralgrenzen bekannt, so dass sich das Integral leicht aufstellen und berechnen lässt. Alternativ lässt sich die Dreiecksfläche einer Hut-Basisfunktion auf eine stückweise polynomielle Basisfunktion skalieren, wie in [Rö11, S. 22f] beschrieben. Das bietet den Vorteil, dass keine Stammfunktionen berechnet werden müssen. Stattdessen müssen die Skalierungskoeffizienten berechnet und gespeichert werden.

Die berechneten Integrale jeder Dimension werden anschließend, wie in Formel (3.30) dargestellt, miteinander multipliziert, mit dem hierarchischen Überschuss am Knoten skaliert und schließlich summiert.

3.3. Adaptivität

Oftmals ist eine Funktion lokal nicht hinreichend glatt oder zeigt lokal unterschiedliche Verhaltensweisen, bspw. in Form von Sprung- oder Knickstellen oder lokalen Oszillationen. In solchen Fällen bietet es sich an, auf die Erhöhung des Gitterlevels zu verzichten. Damit würde man global mehr Gitterpunkte investieren und so in den bereits gut aufgelösten Bereichen entweder nur geringe oder gar keine zusätzlichen Gewinne bei der Interpolationsqualität erzielen. Im Gegenzug erhöhen sich die Kosten durch zusätzlich benötigte Funktionsauswertungen und zusätzlich benötigten Speicherbedarf möglicherweise deutlich. Stattdessen ist es wesentlich geschickter, die Auflösung des Gitters nur lokal zu erhöhen und das Gitter damit adaptiv zu verfeinern. Ein Beispiel für räumlich-adaptive Verfeinerung zeigt Abb. 3.8 auf der nächsten Seite.

Zu beachten ist, dass konstruktionsbedingt nicht alle Gitterpunkte beliebig verfeinert werden können. Vielmehr muss bei der Verfeinerung die Dünngitter-Struktur erhalten bleiben und anschließend, so weit nötig, rekursiv durch das Hinzufügen weiterer Punkte wiederhergestellt werden, weil viele Algorithmen implizit annehmen, dass alle hierarchischen Vorgänger eines Knotens existieren. Ein Beispiel

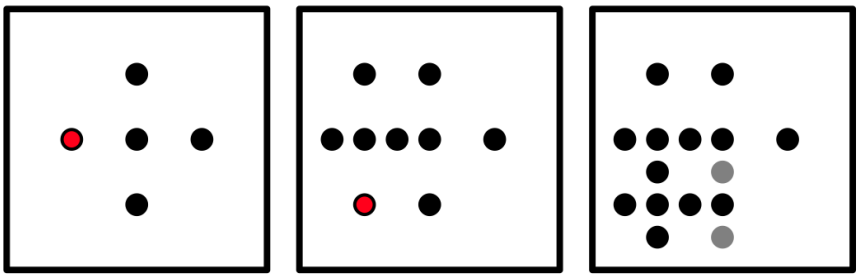

Abbildung 3.8.: Beispiel für die adaptive Verfeinerung eines dünnen Gitters, entnommen aus [Pfl10, S. 21]. Dabei wird der rote Knoten jeweils verfeinert und in jeder Dimension werden beide Kindknoten erzeugt. Da viele Gitteralgorithmen implizit annehmen oder sogar darauf beruhen, dass alle hierarchischen Vorgänger eines Knotens existieren (wie bspw. der in Abschnitt 3.2.1 vorgestellte rekursive Hierarchisierungsalgorithmus), müssen diese rekursiv erzeugt werden.

für einen solchen Algorithmus stellt der in Abschnitt 3.2.1 vorgestellte rekursive Hierarchisierungsalgorithmus dar. Daher können in jedem Verfeinerungsschritt nur aus der Menge verfeinerungsfähiger Knoten die geeignetsten Knoten ausgewählt werden. Die Kandidatenmenge besteht aus allen Knoten, die im Dünngitter-Baum in einer beliebigen Dimension Blattknoten sind und ist damit eine Teilmenge der Menge der Gitterknoten. Im Verfeinerungsschritt werden dann in jeder Dimension alle noch nicht vorhandenen Kindknoten erzeugt, ebenso wie deren noch nicht vorhandene hierarchische Vorgänger. Die Erzeugung dieser hierarchischen Vorgängerknoten erfolgt rekursiv.

Zur Auswahl der bestmöglichen Punkte aus der Kandidatenmenge bieten sich verschiedene Kriterien an, die alle darauf basieren, den Beitrag der potentiellen Kindknoten zur Approximationsgenauigkeit zu qualifizieren. Damit erhält man ein Ranking, das die geeignetsten Punkte für eine Verfeinerung zeigt. Das sind genau die Punkte, deren lokaler Beitrag nach der gegebenen Metrik maximal ist. Um das Ranking zu erstellen, wäre es offensichtlich optimal, alle potentiellen Kandidaten auszuwerten und deren jeweils gelieferten Beitrag festzustellen. Diese naive Herangehensweise ist insbesondere im Zusammenhang mit UQ nicht umsetzbar, wenn eine einzelne Simulation mehrere Minuten oder ggf. sogar mehr Zeit in Anspruch nehmen kann. Daher werden die Knoten, die besonders vielversprechend erscheinen, verfeinert [FDPS14, S. 7].

Im Zusammenhang mit UQ werden nach [FDPS14, S. 8] drei Metriken vorgeschlagen:

1. *Verfeinerung nach dem Betrag der hierarchischen Überschüsse*
   Dies ist der klassische Ansatz für Interpolation mit dünnen Gittern, da der hierarchische Überschuss auf Level l den Interpolationsfehler am Knoten auf Level $l-1$ angibt. Verfeinert wird der Knoten, dessen hierarchischer Überschuss den größten Betrag hat.

2. *Verfeinerung nach dem Erwartungswert*
   Es wird mithilfe der Dünngitter-Approximation der Erwartungswert für eine bestimmte unsi-

chere Zielgröße (Quantity of Interest (QoI)) bestimmt. Anschließend wird festgestellt, welcher Knoten den größten Beitrag leistet, indem jeder Knoten je einmal nicht berücksichtigt wird.

$$\begin{aligned} \left\| \mathbb{E}(f_N(x)) - \mathbb{E}(f_{N\setminus\{(l,i)\}}(x)) \right\|_\infty &= \|v_{l,i}\| \cdot \int \varphi_{l,i}(x) \cdot p(x) dx \\ \text{(nur für Gleichverteilung)} &= \|v_{l,i}\| \cdot p(x_{l,i}) \cdot \int \varphi_{l,i}(x) dx \\ \text{(nur für Gleichverteilung und Hut-Basis)} &= \|v_{l,i}\| \cdot p(x_{l,i}) \cdot 2^{\|l\|_1} \end{aligned} \tag{3.33}$$

Verfeinert wird derjenige Knoten, dessen Beitrag zum Erwartungswert maximal ist.

3. *Verfeinerung nach der Varianz*
   Nach dem in [MZ09] und [FDPS14] vorgeschlagenen Ansatz für ein Verfeinerungskriterium nach der Varianz, wird die lokale Varianz an einem Knoten durch die hierarchischen Überschüsse des Quadrats der Dünngitter-Funktion abgeschätzt. D. h. die Funktionswerte der Stützstellen werden quadriert und die dadurch entstehende Funktion anschließend mit einem dünnen Gitter interpoliert. Auf die dabei entstehenden hierarchischen Überschüsse $v'_{l,i}$ wird das Überschusskriterium angewandt, d. h. verfeinert wird der Knoten, dessen hierarchischer Überschuss $v'_{l,i}$ maximal ist.

Geeignete Methoden zur Berechnung des Erwartungswerts und der Varianz mithilfe von dünnen Gittern finden sich in [Lei13, S. 40ff].

Im folgenden Kapitel sollen die beiden zur Quantifizierung von Unsicherheiten benutzten Frameworks und deren Erweiterung im Rahmen dieser Arbeit vorgestellt werden.

4. Implementierung

4.1. SG++

4.1.1. Allgemeines

SG++[1] ist ein größtenteils in der Programmiersprache C++ geschriebenes freies und offenes Framework für Berechnungen mit dünnen Gittern. Neben einer Bibliothek für C++ werden durch das Java Native Interface (JNI)[2] eine Version der Bibliothek für Java und durch den Simplified Wrapper and Interface Generator (SWIG)[3] eine für Python bereitgestellt. Zusätzlich ist eine Integration in MATLAB[4] möglich.

Bei der Konzeption und Entwicklung wurden zwei große Ziele verfolgt:

1. Flexibilität
2. Effizienz

Im Zielkonflikt zwischen Flexibilität und Effizienz hat Flexibilität Vorrang vor Effizienz. Das Entwicklungsziel bestand und besteht in der Bereitstellung eines effizienten allgemeinen Frameworks für dünne Gitter im Gegensatz zu der Bereitstellung einer höchsteffizienten "Single Purpose" - Anwendung, die zu jeder Problemstellung entsprechend umfangreich modifiziert werden müsste. Dies wird durch die Trennung von grundlegenden und massiv optimierten Basis-Algorithmen für dünne Gitter auf der einen Seite von anwendungsspezifischem Überbau auf der anderen Seite realisiert. Zusätzlich wurde darauf geachtet, Datenstrukturen und Algorithmen zu trennen [Pfl10, S. 41f].

Im Rahmen dieser Arbeit wurde SG++ auf Seite der Basis-Algorithmen um zwei Klassen mit stückweise polynomiellen Basisfunktionen erweitert. Die Klassen basieren auf der bereits bestehenden Klasse "PolyBasis", basierend auf der Arbeit von [Rö11], und erweitern diese um einige Features, wie bspw. die in Abschnitt 3.2.1 vorgestellte rekursive Hierarchisierung nach [Bun98], Quadratur für Polynome mit Polynomgrad $p > 3$ und die Möglichkeit, Gitter mit Randpunkten zu verwenden. Dazu wurden zwei Klassen, "UltraPolyBasis" und "UltraPolyBoundaryBasis", mitsamt zugehöriger Gittertypen analog der bereits vorhandenen Schemata für stückweise lineare Basen (analog den Klassen "LinearBasis" und "LinearBoundaryBasis") aus der bestehenden Klasse "PolyBasis" erzeugt, um eine möglichst gute Kompatibilität mit dem bestehenden Framework zu erzielen und die Integration so einfach wie möglich zu gestalten.

[1] http://www5.in.tum.de/SGpp/releases/index.html
[2] http://docs.oracle.com/javase/7/docs/technotes/guides/jni/index.html
[3] http://www.swig.org/
[4] http://www.mathworks.de/products/matlab/

Das ist aus Entwicklersicht ebenso wünschenswert wie aus Anwendersicht:
Entwicklerseitig profitiert man davon, dass durch das Hinzufügen der implementierten, ähnlich strukturierten Klassen die modulare Struktur erhalten bleibt und zu großen Teilen wiederverwendet werden kann. Zusätzlich wird die Einarbeitungszeit klein gehalten und die Wartbarkeit des Programmcodes erleichtert, indem auf die Einführung zusätzlicher Strukturen außerhalb der bestehenden so weit wie möglich verzichtet wird.
Anwender profitieren davon, dass die durch die stückweise polynomiellen Basisfunktionen eingeführte zusätzliche Komplexität, verglichen mit stückweise linearen Basisfunktionen, für sie weitgehend transparent, d. h. nicht oder nur in geringem Umfang wahrnehmbar ist, weil sich die Bedienung kaum ändert.

Dank des Tensorprodukt-Ansatzes und der Verwendung des unidirektionalen Prinzips genügt es, die Funktionen im 1D zu implementieren. Die konkrete Umsetzung des unidirektionalen Prinzips ist in SG++ bereits generisch vorhanden und kann aufgrund der Übernahme der Aufbau-Schemata der stückweise linearen Basen ohne weitere Anpassung verwendet werden.

4.1.2. Neue “UltraPoly”-Klassen

Der Programmablauf der bisherigen Klasse “PolyBasis” ist in Abb. 4.1 auf Seite 37 in Form eines Flussdiagramms dargestellt.

Bei näherer Betrachtung fällt auf, dass viele Teile in ähnlicher Form erhalten werden können, einiges jedoch aufgrund konzeptueller Entscheidungen weniger oder nicht geeignet ist. In der “PolyBasis” Klasse wird grundsätzlich eine Lagrange-Basis erstellt. Bei direkter Hierarchisierung ist das ein sinnvoller Ansatz, da bei diesem Verfahren in jedem Hierarchisierungsschritt Lagrange-Basisfunktionen ausgewertet werden müssen. Für die “UltraPoly”-Klassen ist dies dagegen kein sinnvoller Ansatz, da hier rekursive Hierarchisierung verwendet werden soll, für die keine Auswertungsoperation, stattdessen aber eine Newton’sche Basis in Form der α-Koeffizienten benötigt wird.

Ähnlich verhält es sich bei der Quadratur-Operation. Für die Implementierung der “UltraPoly”-Klassen wurde das in Gleichung (3.32) skizzierte Verfahren gewählt, die stückweise polynomiellen Basisfunktionen direkt zu integrieren. Diese Berechnungen können entfallen, wenn keine Quadratur-Operation durchgeführt werden soll.

Deshalb wurden zwei Flags eingeführt, die beim Erstellen der Klasseninstanz signalisieren, welche Operation durchgeführt werden soll.

Dadurch entsteht in den “UltraPoly”-Klassen der in Abb. 4.2 auf Seite 38 in Form eines Flussdiagramms visualisierte Programmablauf.

Im Folgenden werden einzelne Komponenten der neu implementierten Klassen näher betrachtet.

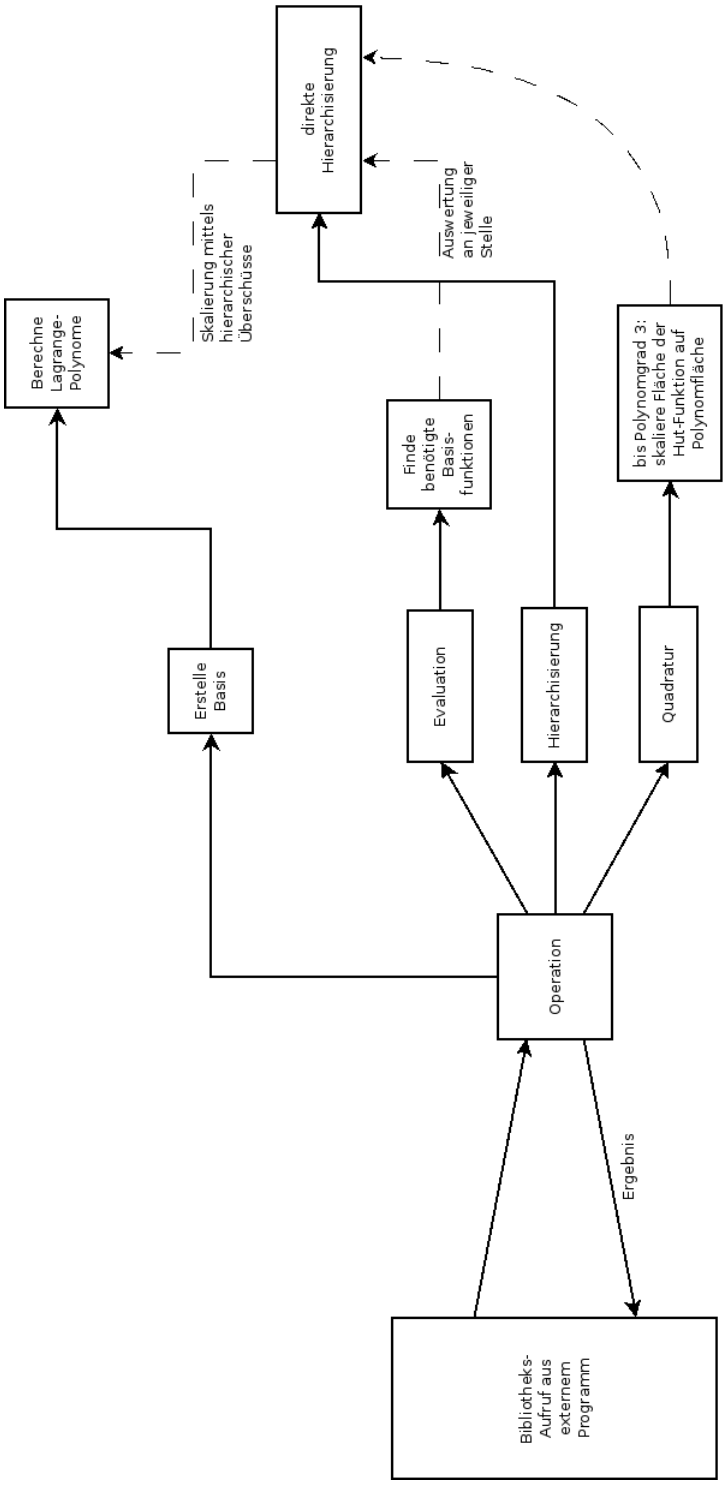

Abbildung 4.1.: Aktueller Programmablauf in “PolyBasis”-Klasse

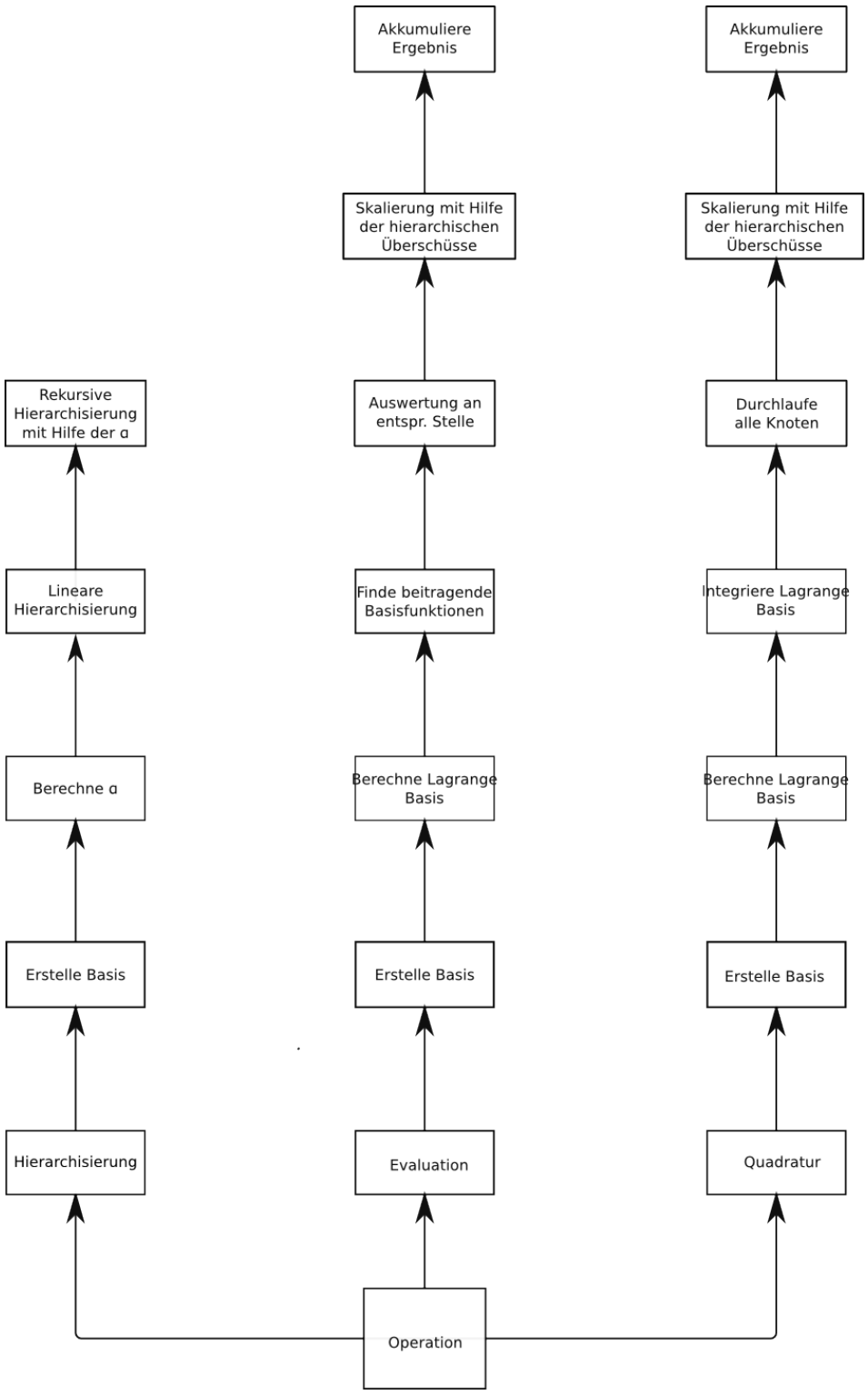

Abbildung 4.2.: Aktueller Programmablauf in “UltraPoly”-Klassen

Lagrange-Basis

Die Basis wird in Tiefensuche nach dem klassischen Lagrange-Ansatz mit $(p+1)$ Stützstellen (x_i, y_i) für die Evaluation- und Quadratur-Operation erstellt.

$$\begin{aligned} L(x) &:= \sum_{i=1}^{p+1} y_i \cdot l_i(x) \\ \text{mit } l_i(x) &:= \prod_{\substack{0 \le m \le p+1 \\ m \neq i}} \frac{x - x_m}{x_i - x_m} \end{aligned} \tag{4.1}$$

Durch den Tiefensuche-Ansatz sind konstruktionsbedingt alle Stützstellen in Form der hierarchischen Vorgänger eines Knotens bei jedem Funktionsaufruf bereits vorhanden, so dass diese nicht mehr explizit gesucht werden müssen. Wie in Gleichung (3.12) definiert, sind die y-Komponenten aller Stützstellen $\{(x_i, y_i)\}$ außer der Stützstelle an der aktuellen Knotenposition (x_n, y_n) null, so dass $\{y_i | i \neq n\} = 0$ und $y_n = 1$. Damit vereinfacht sich der allgemeine Lagrange-Ansatz aus Gleichung (4.1) dramatisch zu

$$L_{l,i}(x) = \prod_{\substack{0 \le m \le p+1 \\ m \neq n}} \frac{x - x_m}{x_n - x_m}. \tag{4.2}$$

Der naive Ansatz wäre nun, die Basisfunktionen nach gegebenem Polynomgrad $p_{\max}$ und gegebenem maximalem Level l nach dem gegebenem Schema zu berechnen. Das kann allerdings insbesondere in adaptiven Fällen dazu führen, dass zu viele Basisfunktionen berechnet werden. Es genügt stattdessen, alle Basisfunktionen bis zu $p_{\max}$ zu berechnen und die erhaltenen Basisfunktionen anschließend zu skalieren. Um die zu skalierende Basisfunktion zu finden, wird das (Level, Index)-Tupel in Abhängigkeit des maximalen Polynomgrads umgerechnet. Diese Umrechnung wird im Anschluss an die Vorstellung der $\alpha_{l,i}$ Koeffizienten vorgestellt.

Aus dem vorgestellten Lagrange Ansatz ergibt sich an jedem Knoten mit Level $l \le p - 1$ eine polynomielle Basisfunktion der Form

$$\varphi_{l,i}(x) = a_p x^p + a_{p-1} x^{p-1} + \ldots + a_1 x + a_0. \tag{4.3}$$

Die Koeffizienten $a_0, \ldots, a_p$ werden in einem Array gespeichert. Dabei wird die Koeffizientenmenge für einen Knoten wie in Tabelle 4.1 dargestellt abgelegt

Tabelle 4.1.: Lagrange-Basis: Speicherschema der Koeffizienten der polynomiellen Basisfunktionen

…	$x_{l,i}$	$int_{l,i}(x)$	a_0	…	a_p	…

Koeffizienten zu Stützstelle $x_{l,i}$

Dabei gilt

$$(4.4)\quad int_{l,i}(x) := \begin{cases} \int_{x_{l,i}-2^{-l}}^{x_{l,i}+2^{-l}} \varphi_{l,i}(x)\, dx & \text{für Quadratur-Operation, Level } l \neq 0 \\ \frac{1}{2} & \text{für Quadratur-Operation, Level } l = 0 \\ 0 & \text{für Evaluation-Operation.} \end{cases}$$

Der Wert des Integrals wird also nur berechnet, wenn er benötigt wird. Das gilt auch für die beiden Level $l = 0$ Einträge, die nur dann gespeichert werden, wenn ein Gitter mit Randpunkten verwendet wird.

Diese Gruppe von Feldern wird für alle Knoten nach der in Abb. 4.3 dargestellten Reihenfolge konkateniert.

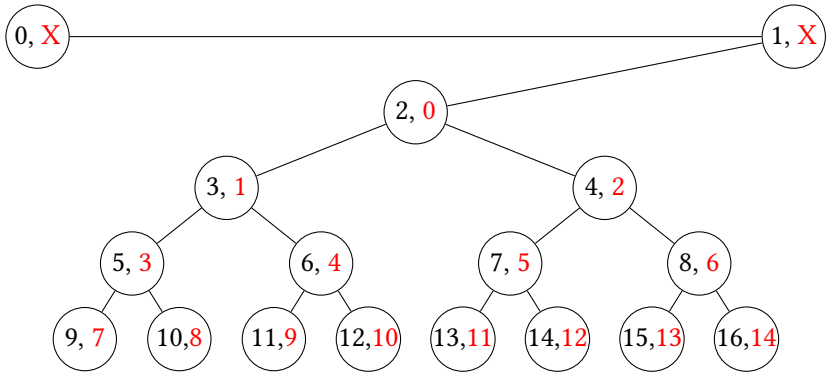

Abbildung 4.3.: Array-Index der zum Knoten gehörenden Basisfunktionskoeffizienten nach Tabelle 4.1. Schwarz für ein Gitter mit Rand, blau für ein randloses Gitter

Newton-Basis

Die zur rekursiven Hierarchisierung benötigte Newton-Basis wird genau wie die im vorigen Abschnitt beschriebene Lagrange-Basis durch Tiefensuche erstellt. Ebenso analog wird die Erstellung weiterer Koeffizienten gestoppt, wenn gilt, dass Level $l = p_{\max}$.

Die Koeffizienten werden nach der in Abschnitt 3.2.1 vorgestellten Berechnungsvorschrift berechnet.

$$(4.5)\quad \begin{aligned} \alpha_{0,0} &:= 1 \\ \alpha_{0,1} &:= 1 \\ \alpha_{1,1} &:= 1 \\ \alpha_{l,i} &:= \frac{x_n - x_m}{x_{p+1} - x_n} \cdot \prod_{\substack{k=0 \\ k \neq m, k \neq n}}^{p} \frac{x_m - x_k}{x_n - x_k} \end{aligned}$$

Dabei liegen die Stützstellen x_i als im Sinne ihrer Position in $\mathbb{R}$ geordnete Liste vor. Der aktuelle Knoten wird mit x_m und sein hierarchischer Vorgänger mit x_n bezeichnet.

Für den Knoten $(2,1)$ ergibt sich damit folgende Beispielrechnung für $\alpha_{2,1}$:

Hierarchische Vorgänger von $(2,1)$: $[(0,0),(0,1),(1,1)] \rightsquigarrow [0, \underbrace{0.25}_{x_m}, \overbrace{0.5}^{x_n}, 1]$

$$\begin{aligned}
\alpha_{2,1} &= \frac{x_n - x_m}{x_{p+1} - x_n} \cdot \prod_{k=0, k\neq m,n}^{p} \frac{x_m - x_k}{x_n - x_k} \\
&= \frac{0.5 - 0.25}{1 - 0.5} \cdot \frac{0.25 - 0}{0.5 - 0} \\
&= \left(\frac{0.25}{0.5}\right)^2 \\
&= 0.25
\end{aligned}$$

Eine Übersicht weiterer Werte liefern die Bilder in [Bun98, S. 60].

Aus Symmetriegründen genügt es, sich bei Berechnung und Speicherung der $\alpha_{l,i}$-Koeffizienten auf den linken Teilbaum ab Knotenlevel 2 zu beschränken. Die berechneten $\alpha_{l,i}$-Koeffizienten werden analog den Koeffizienten der Basispolynome der Lagrange-Basis in einem Array abgelegt. Das verwendete Schema zur Bestimmung der Speicherposition der einzelnen $\alpha_{l,i}$-Koeffizienten zeigt Abb. 4.4.

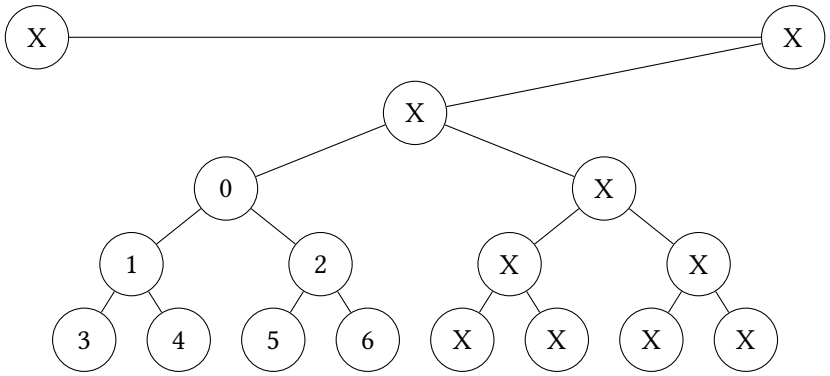

Abbildung 4.4.: Speicherposition der $\alpha_{l,i}$ im Array. Die Werte für Knoten, die ein X enthalten ergeben sich entweder aus der Symmetrie (Level $l \geq 2$) oder sie werden nicht benötigt (Level $l = 0, 1$).

Berechnung der Indizes

Durch den gewählten Ansatz, die Basen nicht durch das maximale Gitterlevel $l_{\max}$ sondern durch den maximalen Polynomgrad $p_{\max}$ zu beschränken, kann es vorkommen, dass für höhere Level zunächst keine entsprechenden Einträge vorhanden sind und entsprechend Einträge niedrigerer Level skaliert werden müssen. Der Code in Listing 4.1 auf Seite 42 zeigt am Beispiel der $\alpha_{l,i}$-Koeffizienten, wie das Mapping von maximalem Polynomgrad $p_{\max}$, Level l und Index i auf den entsprechenden Index im Array umgesetzt ist.

Für die Indizes der Lagrange-Basen funktioniert das Mapping ähnlich, da es aufgrund der Binärbaumstruktur genügt, den Anfang um einen Level nach oben zu verschieben.

```
// Funktion, die in Abhaengigkeit von Polynomgrad p_degree_ui32 fuer Knoten
// (p_level_ui32, p_index_ui32) den zugehoerigen alpha-Koeffizienten sucht und
// zurueck gibt
double getAlpha
 (unsigned int p_degree_ui32,
  unsigned int p_level_ui32,
  unsigned int p_index_ui32)
{
  // falls Level > Polynomgrad muss skaliert werden
  if (p_level_ui32 > p_degree_ui32)
  {
    p_level_ui32 = p_degree_ui32;
    p_index_ui32 = p_index_ui32 % (1 << p_degree_ui32)
  }

  // falls Knoten in rechtem Teilbaum liegt:
  // Abbildung auf symmetrischen Knoten im linken Teilbaum
  if (p_index_ui32 > (unsigned int)(1 << (p_level_ui32 - 1)))
  {
    p_index_ui32 = (1 << p_level_ui32) - p_index_ui32;
  }

  // nun sind Level und Index ''normiert'', d. h. es existiert fuer das
  // gegebene (Level, Index)-Tupel ein Eintrag im alpha-Vektor
  //
  // Nun Mapping: (2, 1) => 0, (3, 1) => 1, (3, 3) => 2, (4, 1) => 3 usw.
  unsigned int l_deg_ui32;
  if (p_degree_ui32 - 1 < p_level_ui32)
  {
    l_deg_ui32 = p_degree_ui32 - 1;
  }
  else
  {
    l_deg_ui32 = p_level_ui32 - 1;
  }
  unsigned int l_idMask_ui32 = (1 << l_deg_ui32) - 1;
  unsigned int l_provID_ui32 =
    (((p_index_ui32 & l_idMask_ui32) >> 1) | (1 << (l_deg_ui32 - 1))) - 1;
  return m_alphas_f64[l_provID_ui32];
}
```

Listing 4.1: Mapping des Tupels $(p_{\max}, l, i)$ auf die zugehörige Arrayposition als C++ - Code am Beispiel der $\alpha_{l,i}$-Koeffizienten

Hierarchisierung und Dehierarchisierung

Wie bereits erwähnt, soll das dünne Gitter in den "UltraPoly"-Klassen rekursiv hierarchisiert werden. Stückweise polynomielle Basisfunktionen müssen von $p = 1$ bis $p = p_{\max}$ für jeden Polynomgrad separat hierarchisiert werden. Begonnen wird mit der linearen Hierarchisierung. Dann werden für jeden Polynomgrad nacheinander die hierarchischen Überschüsse nach der in Gleichung (3.15) gegebenen Formel errechnet.

Mit dieser Formel ist die Dehierarchisierung analog zu implementieren:

$$v^{(p-1)}(x_m) = v^{(p)}(x_{l,i}) + \alpha(x_0, \ldots, x_{p+1}) \cdot v^{(p-1)}(x_n) \tag{4.6}$$

Zu beachten ist, dass die Reihenfolge der Dehierarchisierungsschritte genau umgekehrt zu den Hierarchisierungsschritten ist. Während bei der Hierarchisierung aufsteigend von $p = 1$ bis $p = p_{\max}$ hierarchisiert wird, läuft die Dehierarchisierung entsprechend dem Polynomgrad nach absteigend ab.

In beiden Fällen wird der hierarchische Überschuss des Vorgängerknotens in dem Polynomgrad des vorherigen (De-)Hierarchisierungsschritts benötigt. Dieser wurde wegen der Traversierung des Dünngitterbaums in Tiefensuche bereits überschrieben, so dass der Überschussvektor kopiert werden muss.

Evaluation

Zur effizienten Auswertung der Polynombasis wird das Hornerschema verwendet.

$$(\ldots(a_p x + a_{p-1})\, x + \ldots)\, x + a_0 \tag{4.7}$$

Dadurch werden im Vergleich zur klassischen Darstellung $a_0 + a_1 x + \ldots + a_p x^p$ Multiplikationen oder Potenz-Operationen eingespart, je nach dem wie die x^i-Ausdrücke algorithmisch aufgelöst werden.

Die Vorgehensweise zum Finden der beitragenden Basisfunktionen zu einer bestimmten Evaluationsstelle unterscheidet sich für stückweise polynomielle Basisfunktionen nicht von der für stückweise lineare Basisfunktionen. Somit kann die grundsätzliche algorithmische Struktur für die Auswertung der Dünngitter-Funktion $f_N(x)$ beibehalten werden. Ersetzt wird einzig die Auswertung der beitragenden Basisfunktionen $\varphi_{l,i}(x)$.

Die Skalierung der Basisfunktionen wurde aus der bereits bestehenden Klasse “PolyBasis” übernommen.

Quadratur

Zur Berechnung der Quadratur einer einzelnen Basisfunktion $\varphi_{l,i}(x)$ wurde, wie bereits erwähnt, das in Gleichung (3.32) skizzierte Verfahren zur Berechnung der Stammfunktion der Lagrange-Basispolynome umgesetzt. Zur Auswertung von deren Wert an den Integralgrenzen wurde analog zur Evaluations-Operation das Hornerschema angewandt.

Zur Berechnung des Integrals über die Dünngitter-Approximation $f_N(x)$ wurde der Ansatz aus Gleichung (3.30) direkt implementiert. Dabei ist zu berücksichtigen, dass auch der Wert des Integrals skaliert werden muss, wenn der Level l des aktuellen Gitterknotens $x_{l,i}$ gleich oder größer als Polynomgrad p ist, so dass keine Lagrange-Basisfunktion mehr vorhanden ist. Der Skalierungsfaktor beträgt

$$\frac{1}{2^{l-p+1}}. \tag{4.8}$$

D. h. nichts anderes, als dass sich der Wert des Integrals einer Stammfunktion $\varphi_{l,i}(x)$ mit jedem Skalierungsschritt halbiert:

$$\begin{aligned} \int_{\Gamma} u(x)\,dx &\approx \int_{\Gamma} f_N(x)\,dx \\ &= \sum_{j=1}^{m} \left(v_j \cdot \left(\prod_{k=1}^{|D|} \int \varphi_{l'(j,k),i'(j,k)}(x)\,dx \right) \right) \end{aligned} \tag{4.9}$$

Daraus ergibt sich:

$$\int_{x_{l,i}-2^{-l}}^{x_{l,i}+2^{-l}} \varphi_{l,i}(x)\,dx = \begin{cases} \int \varphi_{l,i}(x)\,dx & \text{wenn } l < p \\ \frac{1}{2^{(l-p+1)}} \int_{x_{l',i'}-2^{-l'}}^{x_{l',i'}+2^{-l'}} \varphi_{l',i'}(x)\,dx & \text{sonst} \end{cases} \tag{4.10}$$

mit $l' = p - 1$

$$\rightsquigarrow \int_{x_{l',i'}-2^{-l'}}^{x_{l',i'}+2^{-l'}} \varphi_{l',i'}(x)\,dx = \int_{x_{(p-1),i'}-2^{-(p-1)}}^{x_{(p-1),i'}+2^{-(p-1)}} \varphi_{(p-1),i'}(x)\,dx$$

Adaptivität

Auch bei der adaptiven Verfeinerung konnte das bereits bestehende Konzept übernommen werden.

Zu betonen ist aber, dass bei der Verwendung adaptiver Gitterverfeinerung das Gitter in jedem adaptiven Verfeinerungsschritt komplett neu hierarchisiert werden muss. Wegen der geltenden Berechnungsformel für den Überschuss an einem Knoten $v_{l,i}$ müssten die Überschüsse aller potentiellen Vaterknoten in allen Dimensionen und Polynomgraden bekannt sein. Diese Hinterlegung ist weder in der Programmstruktur von SG++ vorgesehen noch besonders speichereffizient, weil es insbesondere im Hochdimensionalen ab einer entsprechenden Gittergröße deutlich mehr Blattknoten geben wird, die nicht verfeinert werden, als umgekehrt.

4.2. UQLib

Die UQLib ist eine Python-Bibliothek, die ein breites Funktions-Spektrum zur Berechnung von UQ-Problemen mithilfe nicht-intrusiver Methoden abdeckt. Neben der im Rahmen dieser Arbeit untersuchten Sparse Grid Collocation Method (SGCM), d. h. ASGC und CSGC, können auch MC und QMC Verfahren verwendet werden.

4.2.1. UQ-Setting

Zunächst muss die UQ-Umgebung definiert werden. Zu definieren sind u. a. die folgenden Parameter:

- Die *unsichere Parametermenge* sowie deren *Wahrscheinlichkeitsdichte*. Dabei werden für jeden unsicheren Parameter Name, Wertebereich und Verteilung festgelegt.
- Regeln für die *Diskretisierung der Zeit*. Neben Start- und Endzeit können auch die Schrittweite sowie Regeln zur Verfeinerung der Schrittweite definiert werden.
- Die *Transformation* um die Samples, die im Unit-Hypercube generiert werden, auf die Wertebereiche der Elemente der unsicheren Parameter zu transformieren. Im Rahmen der vorliegenden Arbeit wird ausschließlich die lineare Transformation verwendet. Eine Alternative, die inverse Rosenblatt Transformation, wird in [Lei13, S. 24] vorgestellt.
- *Parameter für einzelne Simulationsaufrufe*. Diese Parameter beinhalten v. a. die Definition der Steuerfunktionen für den bibliotheksgesteuerten Simulationsaufruf.
- Die *zu interpolierende QoI*. In diesem Schritt wird definiert, welche Zielgröße für die Interpolation, d. h. insbesondere für die adaptive Verfeinerung, maßgeblich ist.
- Die *statistischen Auswertungen*, die aus den Simulationsergebnissen gewonnen werden sollen. D. h. abhängig von der gewünschten Untersuchung, die durchgeführt werden soll (Varianzanalyse, Sensitivitätsanalyse, …), werden die Ergebnisse weiter analysiert und aufbereitet.

Zusätzlich sind methodenspezifische Parameter festzulegen. Für *MC und QMC Verfahren* bedeutet das, die Zahl der zu generierenden Samples festzulegen. Für QMC Verfahren kann außerdem der Sequenzgenerator ausgewählt werden. Für *SGCM Verfahren* bestehen umfangreichere Konfigurationsmöglichkeiten. Es können u. a. folgende Parameter angepasst werden:

- das initiale Gitter, d. h. der Gittertyp (Art des Rands, initiales Gitterlevel, Art der Basisfunktionen),
- die Adaptivität, d. h. Verfeinerungskriterien, Abbruchbedingungen, verschiedene Arten der Bildung der Kandidatenmenge,
- verschiedene Verfahren zur Berechnung von Erwartungswert und Varianz u. a. m..

4.2.2. Steuerung des Simulationsaufrufs

Ein Simulationsaufruf wird als fünfschrittiger Prozess modelliert.

1. *Generierung eines Samples*
   Im Unit-Hypercube wird ein Sample ξ_{unit} generiert. Das bedeutet im Fall von
   - MC eine randomisierte Generierung nach der gegebenen Verteilung,
   - QMC die Erzeugung eines Samples nach der gegebenen deterministischen Sequenz und
   - SGCM die Bestimmung der Samples durch die Knoten des dünnen Gitters.

2. *Transformation*
   Die generierten Samples werden gemäß der spezifizierten Transformation in den Raum der unsicheren Parameter transformiert.

$$\begin{aligned} &trans : [0,1]^d \mapsto \Gamma \\ &p' = trans(\xi_{unit}) \end{aligned} \tag{4.11}$$

3. *Preprocessing*
   Um eine Simulation durchzuführen, müssen eventuell weitere Parameter festgelegt werden, die von den stochastischen Größen abhängen. So kann bspw. in Abhängigkeit eines unsicheren Parameters oder mehrerer unsicherer Parameter eine feinere Diskretisierung der Zeit notwendig sein.
   Zusätzlich besteht durch diesen Schritt die Möglichkeit, die Parameter der Simulation von den unsicheren Parameter zu entkoppeln. Damit kann bspw. ein Werkstoff oder ein äußerer Antrieb für den Anwender durch andere Kennzahlen definiert werden, als dies in der Simulation der Fall ist. Als triviales Beispiel für eine solche "Übersetzung" kann bspw. ein Moment dienen, das alternativ durch Kraft und Hebelarm definiert werden kann. Auch unterschiedliche Skalen sind denkbar, wie bspw. eine Umrechnung zwischen britischen und Standard-Einheiten.

$$p' = g(p) \tag{4.12}$$

4. *Simulation*
   In diesem Schritt wird die Simulation mit den in den vorherigen Schritten ermittelten Parametern gestartet. Die Parameterübergabe kann dabei etwa über eine Steuerdatei oder Kommandozeilen-Parameter erfolgen. Die Ergebnisse der Simulation werden im Anschluss an den nächsten Schritt übergeben.

$$S(p') \mapsto \{Res\} \tag{4.13}$$

5. *Postprocessing*
   Im Normalfall stimmen die Ergebnisse der Simulation nicht mit den Zielgrößen überein, die als QoI definiert sind. Daher müssen die Ergebnisse entsprechend aufbereitet werden, um sie im Folgenden weiterverarbeiten zu können.

$$\begin{array}{rcrcl} & & & t_0 & \rightarrow & t_n \\ \{Res\} & \rightarrow & \{QoI_1: & [& \dots &] \\ & & QoI_2: & [& \dots &] \\ & & \vdots & & & \\ & & QoI_k: & [& \dots &]\} \end{array} \tag{4.14}$$

   Die so definierte Menge Res erfüllt noch nicht die Vorgabe von Gleichung (2.1). Stattdessen ist die Menge Res aus $\mathbb{R}^{((t_n - t_0)\cdot t_s)\times|\{QoI\}|}$. Daher muss für jedes Element der Menge $\{QoI\}$ zu jedem diskretisierten Zeitschritt eine Abbildung $u_{QoI,t}(x)$ nach Gleichung (2.1) definiert werden.

Die gewonnenen Ergebnisse aus einem Simulationsaufruf stellen je einen Punkt der in Gleichung (2.1) definierten Abbildung $u_{QoI,t}(x)$ dar. Zur Berechnung der Quadratur von $u_{QoI,t}(x)$ kann dann entweder mit MC oder QMC approximiert werden oder aber man approximiert zunächst die Zielfunktion selbst, um anschließend die Quadratur der Approximation zu bestimmen (gPC, SGCM).

5. Experimente

In diesem Kapitel soll mithilfe von numerischen Experimenten die Wirksamkeit der implementierten Maßnahmen untersucht werden. Zum Vergleich der Approximationsqualität wird die im Rahmen dieser Arbeit entstandene Implementierung der stückweise polynomiellen Basisfunktionen mit MC und stückweise lineare Basisfunktionen verglichen.

Begonnen wird mit zwei analytischen, nicht-polynomiellen Funktionen mit bekanntem Verhalten aus $\mathbb{R}^2 \to \mathbb{R}$:

1. Der in [MZ09, S. 3096] definierten Betragsfunktion und
2. einer zweidimensionalen Projektion des Kosinus Hyperbolicus von $[-1, 1]^2$ auf $[0, 1]^2$.

Anschließend werden direkte und rekursive Hierarchisierung für die in Abschnitt 5.1.1 definierte Betragsfunktion miteinander verglichen und schließlich analog [Lei13] die Kármán'sche Wirbelstraße untersucht.

5.1. Experimente analog [MZ09, Kapitel 4.1]

In diesem Abschnitt wird das in [MZ09, S. 3096f] beschriebene Approximationsexperiment nachgestellt und zusätzlich um dünne Gitter mit stückweise polynomiellen Ansatzfunktionen von Grad $p = 2$ und $p = 6$ erweitert. Diese Werte begründen sich damit, dass $p = 2$ die erste und damit einfachste polynomielle Ansatzfunktion darstellt und dass $p = 6$ der maximale Polynomgrad des initialen Gitterlevels $l = 5$ ist.

Dabei werden als Referenz $n = 1000$ gleichverteilte Samples aus $[0, 1]^2$ gezogen, an denen die Funktion ausgewertet wird. Diese Ergebnisse werden mit denen der SGCM verglichen.

Für den Vergleich der SGCM werden, ausgehend vom initialen Gitterlevel $l = 5$ mit Trapezrand, für jeden Polynomgrad CSGC und ASGC gegenüber gestellt. Dazu wurde im Fall von CSGC der Gitterlevel l Schritt für Schritt erhöht bis zu $l = 13$. Im Fall von ASGC wurden verschiedene Schwellwerte ε als Abbruchbedingung für das Verfeinerungskriterium des Betrags der Überschüsse definiert. Dabei werden in jedem Schritt aus der Menge der verfeinerbaren Gitterknoten all die Knoten verfeinert, deren hierarchischer Überschuss betragsmäßig oberhalb des definierten Schwellwertes ε liegt.

Die folgenden Messungen wurden für jede Methode und jeden Polynomgrad durchgeführt, wobei die Ergebnisse jeweils mit den generierten MC Referenzwerten verglichen wurden:

- Messung des maximalen Approximationsfehlers über die verschiedenen Schwellwerte
- Messung des maximalen Approximationsfehlers über die Knotenzahl für CSGC und für ASGC mit unterschiedlichen Schwellwerten ε nach jedem Verfeinerungsschritt

5.1.1. Betragsfunktion

Approximiert wird die Funktion

$$u_1(x, y) = \frac{1}{|0.3 - x^2 - y^2| + 0.1}, \tag{5.1}$$

dargestellt in Abb. 5.1, mit x, y als jeweils in $[0, 1]$ gleichverteilten unsicheren Parametern. Die Funktion hat eine lokale Unstetigkeit in Form einer Knickstelle entlang eines Viertelkreises mit Radius 0.3 um den Koordinatenursprung in Punkt $(0, 0)$ herum. Diese Unstetigkeit führt dazu, dass globale Polynomansätze wie gPC am in Abschnitt 2.5.2 beschriebenen Gibbs'schen Phänomen leiden und damit nicht geeignet sind.

Als Abbruchkriterium für die Verfeinerung werden die Schwellwerte $\varepsilon = [1, 0.1, 0.01, 0.001]$ verwendet.

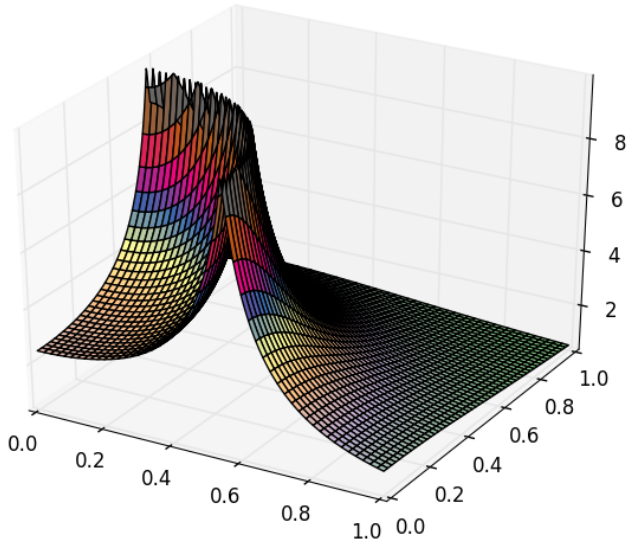

Abbildung 5.1.: Funktionsplot der Betragsfunktion nach [MZ09, S. 3096]

Zunächst wird jeweils der *maximale Interpolationsfehler* gemessen, d. h.

$$\max(|f_N(x_i) - u_1(x_i)|, x_i \in \{\text{Referenzsamples}\}). \tag{5.2}$$

Dabei konnten die Ergebnisse aus [MZ09, S. 3097] nachgestellt werden, so dass dieses Experiment auch als Test für die Korrektheit des Codes betrachtet werden kann.

Abb. 5.2 zeigt den maximalen Fehler am Ende der adaptiven Verfeinerung für die jeweiligen Schwellwerte. Man erkennt, dass die stückweise polynomiellen Basisfunktionen einen größeren maximalen Fehler haben, als der entsprechende Ansatz mit stückweise linearen Basisfunktionen. Aufgrund der Unstetigkeit in Form des Viertelkreises kommt dieses Ergebnis nicht unerwartet, da bei der

Einführung der polynomiellen Basisfunktionen die Bedingung, eine hinreichend glatte Funktion zu approximieren, explizit gestellt wurde.

Abb. 5.3, Abb. 5.4 und Abb. 5.5 auf Seite 52f zeigen die Entwicklung des maximalen Fehlers nach jedem Verfeinerungsschritt für die betrachteten Polynomgrade $p = [1, 2, 6]$. Diese werden jeweils mit dem maximalen Fehler aus der CSGC Methode für ein reguläres dünnes Gitter mit Trapezrand auf den Levels $l = [5, \ldots, 13]$ für den entsprechenden Polynomgrad verglichen. Man sieht zweierlei: Erstens, dass sich für jeden Polynomgrad ein deutlicher Vorteil von ASGC gegenüber CSGC einstellt, d. h. Adaptivität im betrachteten Experiment einen deutlichen Nutzen hat, und zweitens, dass auch in diesen Darstellungen kein Hinweis darauf zu finden ist, dass sich im gegebenen Beispiel der höhere Aufwand in Form des Einsatzes stückweise polynomieller Basisfunktionen lohnt. Im Gegenteil, der maximale Fehler der stückweise polynomiellen Basisfunktionen liegt ab ca. 10000 Gitterpunkten auch für Polynomgrad $p = 2$ höher und damit schlechter als der maximale Fehler der stückweise linearen Basisfunktionen. Für Polynomgrad $p = 6$ zeigt sich ein noch schlechteres Fehlerverhalten. Dies zeigt auch der Verlauf des maximalen Interpolationsfehlers nach jedem Verfeinerungsschritt für die Basisfunktionen der jeweiligen Polynomgrade für den Schwellwert $\varepsilon = 0.001$ in Abb. 5.6 auf Seite 53.

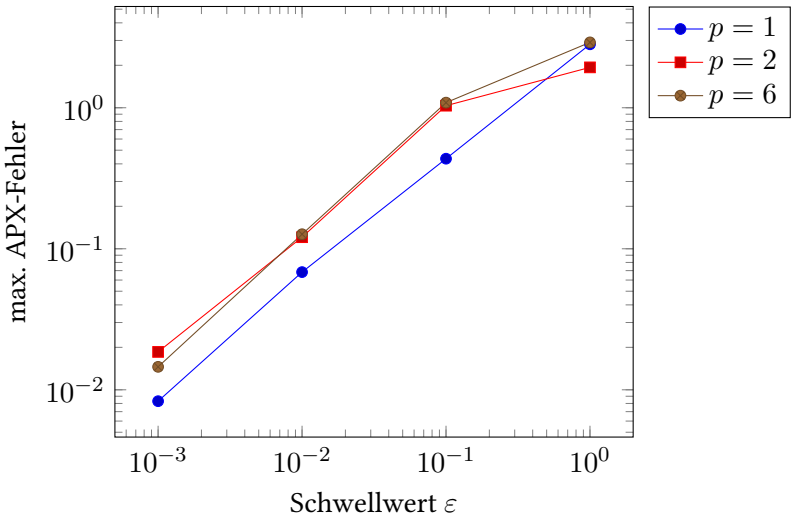

Abbildung 5.2.: Betragsfunktion: Entwicklung des maximalen Interpolationsfehlers über 1000 zufällige Punkte aus $[0, 1]^2$ für verschiedene Polynomgrade p und Adaptivitätsschwellwerte ε der Dünngitter-Basisfunktionen

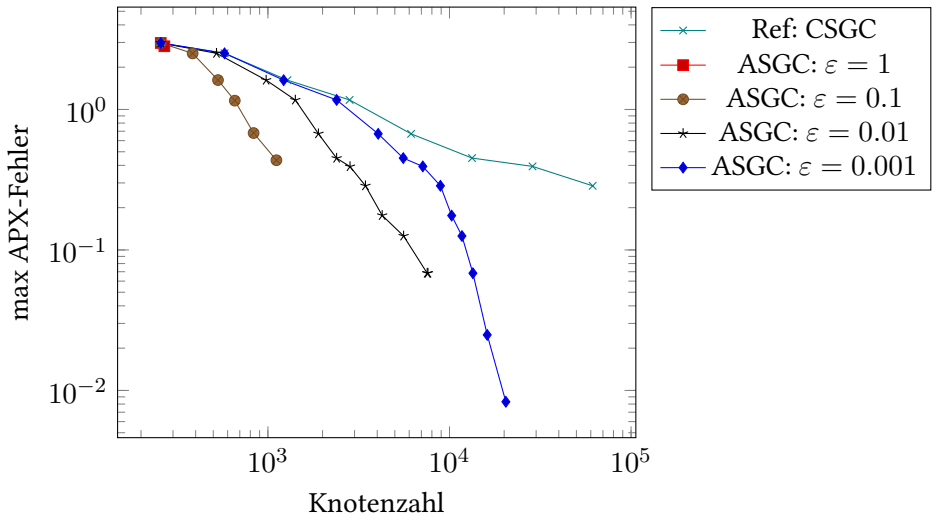

Abbildung 5.3.: Betragsfunktion: Entwicklung des Interpolationsfehlers über die Knotenzahl für stückweise lineare Basisfunktionen

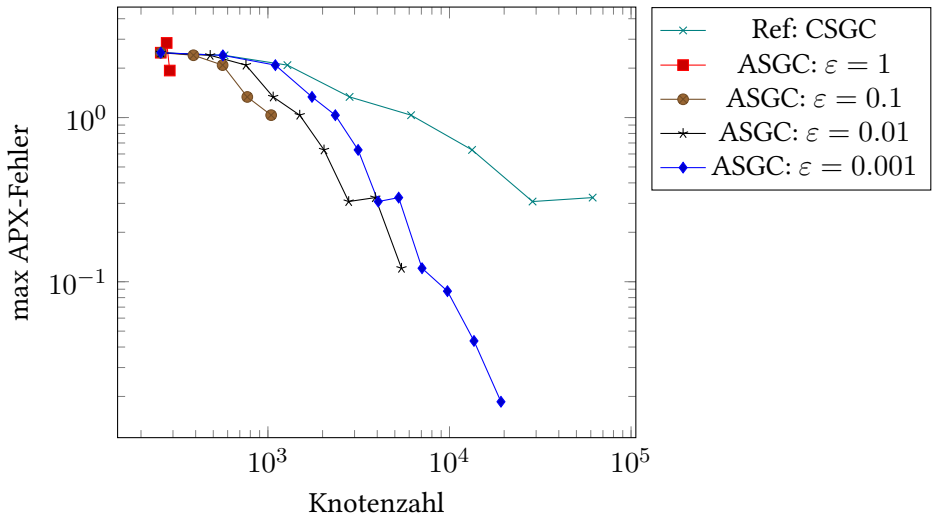

Abbildung 5.4.: Betragsfunktion: Entwicklung des Interpolationsfehlers über die Knotenzahl für stückweise polynomielle Basisfunktionen mit Grad 2

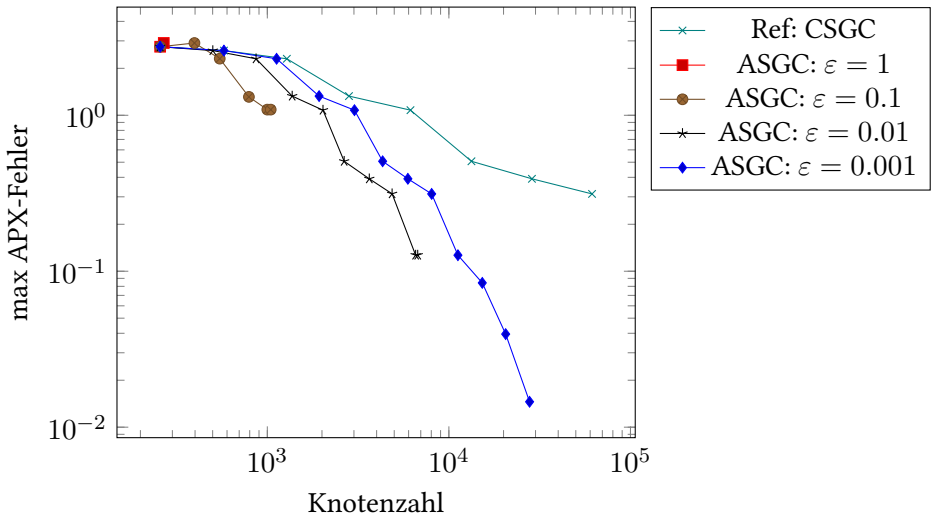

Abbildung 5.5.: Betragsfunktion: Entwicklung des Interpolationsfehlers über die Knotenzahl für stückweise polynomielle Basisfunktionen mit Grad 6

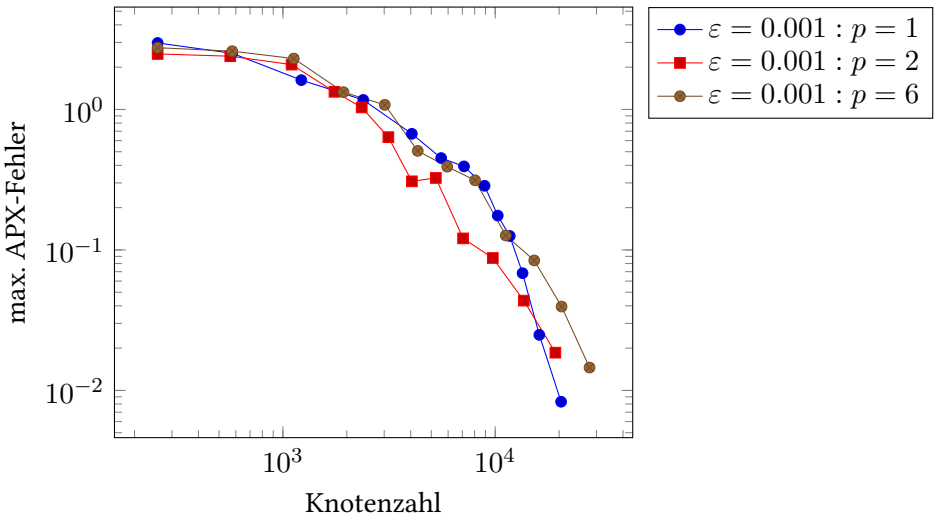

Abbildung 5.6.: Betragsfunktion: Entwicklung des maximalen Interpolationsfehlers über die Knotenzahl für verschiedene Polynomgrade p und Adaptivitätsschwellwert $\varepsilon = 0.001$ der Dünngitter-Basisfunktionen

5.1.2. Kosinus Hyperbolicus

Im vorherigen Abschnitt wurde die Wirksamkeit räumlich adaptiver Verfeinerung mit ASGC bei lokalen Unstetigkeiten gegenüber globaler Verfeinerung mit CSGC gezeigt. Die erhoffte höhere Konvergenzordnung durch den Ansatz stückweise polynomieller Basisfunktionen konnte aufgrund dieser

Unstetigkeit allerdings nicht beobachtet werden. Stattdessen lag deren maximaler Fehler sogar überhalb des Fehlers stückweise linearer Basisfunktionen. Aufgrund der Verletzung der Glattheitsannahme von u_1 kam das Ergebnis aber nicht überraschend.

Um die Glattheitsannahme zu erfüllen, wird das bestehende Setting übernommen und die zu approximierende Funktion durch einen Kosinus Hyperbolicus ersetzt,

$$u_2(x, y) = \cosh(2x - 1) \cdot \cosh(2y - 1), \tag{5.3}$$

dargestellt in Abb. 5.7, wiederum mit x, y als jeweils gleichverteilten, unsicheren Parametern auf $[0, 1]$.

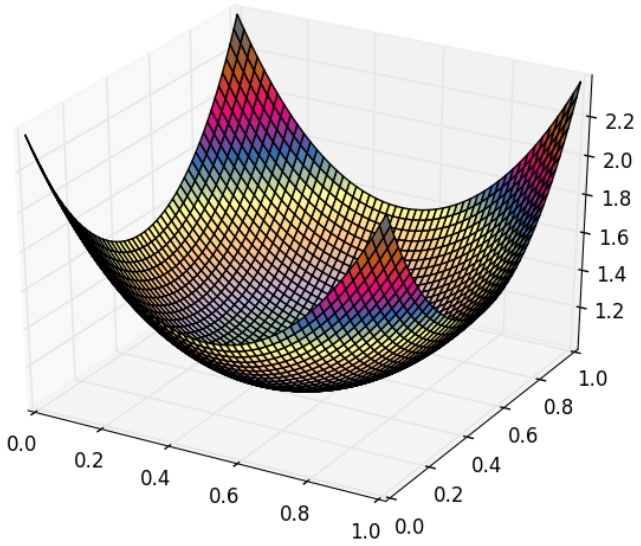

Abbildung 5.7.: Funktionsplot der hyperbolischen Kosinusfunktion

Als Abbruchkriterium der adaptiven Verfeinerung dienen die Schwellwerte $\varepsilon = [1, 0.1, 1 \cdot 10^{-3}, 1 \cdot 10^{-5}, 1 \cdot 10^{-7}]$. Zusätzlich wird in diesem Kontext die Genauigkeit der Berechnung des Erwartungswerts untersucht mithilfe der analytischen Referenzlösung von

$$\int_0^1 \int_0^1 u_2 \, dx \, dy = \frac{1}{2}(\cosh(2) - 1). \tag{5.4}$$

Wegen der Gleichverteilung von x, y ist dies ein reiner Test der Quadraturqualität der Dünngitterapproximation von u_2, vgl. Gleichung (3.33).

Dabei stellt sich, wie Abb. 5.8 auf Seite 55 zeigt, der aus Abschnitt 3.1.4 erwartete deutliche Gewinn bei der Konvergenzordnung für stückweise polynomielle Basisfunktionen ein. Zu beachten ist, dass zwar nur ein geringer Unterschied zwischen stückweise linearen Basisfunktionen und stückweise polynomiellen Basisfunktionen von Polynomgrad $p = 2$ zu liegen scheint. Die im Anhang abgedruckten Fehlerplots nach jedem Verfeinerungsschritt je Schwellwert zeigen, dass diese vergleichbare Genauigkeit mit deutlich weniger Gitterpunkten erreicht werden kann.

Durch die Homogenität der verwendeten cosh-Funktion ist Adaptivität in diesem Szenario allerdings ohne Nutzen. Dies zeigt die Kongruenz der Kurven für ASGC und CSGC.

An dieser Stelle soll, wie eingangs erwähnt, zusätzlich die Berechnung des Erwartungswerts als

$$\mathbb{E} = \int f_N(x) p(x)\, dx = \int f_N(x)\, dx \text{ (wegen Gleichverteilung)} \tag{5.5}$$

mithilfe der in Gleichung (5.4) gegebenen analytischen Lösung untersucht werden.

Dabei zeigen Abb. 5.9 bis Abb. 5.11 auf Seite 56f grundsätzlich eine gute Approximationsqualität für die Berechnung der Quadratur, bei der - analog zur Evaluationsoperation - die stückweise polynomiellen Basisfunktionen bessere Ergebnisse als die stückweise linearen Basisfunktionen liefern, bei gleichzeitig weniger benötigten Gitterpunkten.

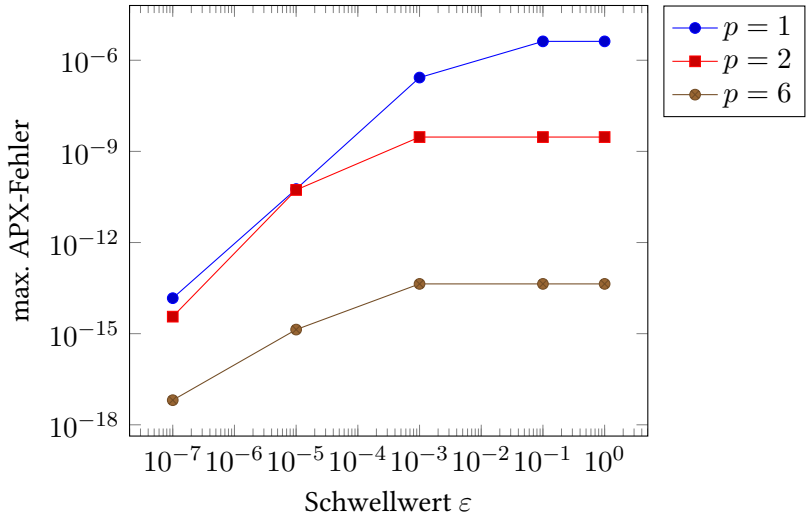

Abbildung 5.8.: cosh: Entwicklung des maximalen Interpolationsfehlers über 1000 zufällige Punkte aus $[0, 1]^2$ für verschiedene Polynomgrade und Adaptivitätsschwellwerte der Dünngitter-Basisfunktionen

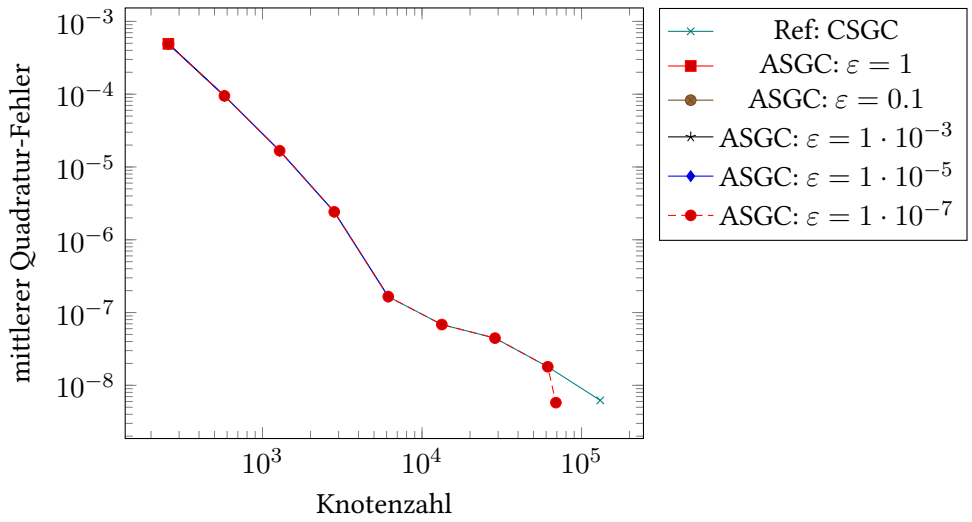

Abbildung 5.9.: cosh: Entwicklung des mittleren Quadraturfehlers über die Knotenzahl für stückweise lineare Basisfunktionen

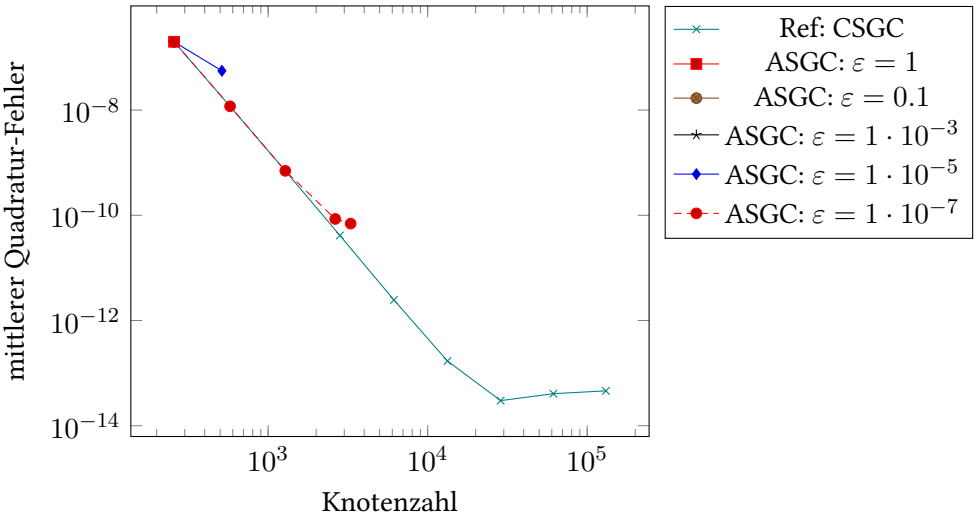

Abbildung 5.10.: cosh: Entwicklung des mittleren Quadraturfehlers über die Knotenzahl für stückweise polynomielle Basisfunktionen mit Grad 2

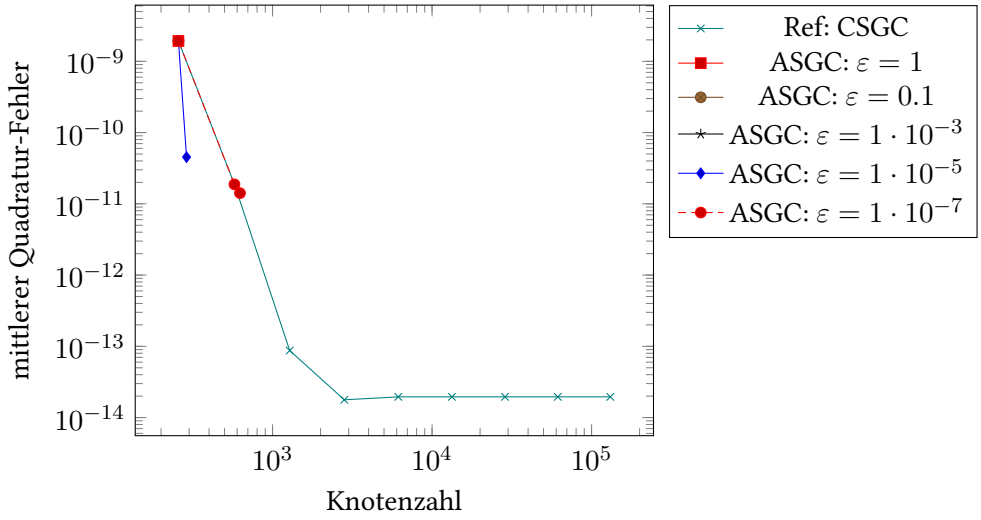

Abbildung 5.11.: cosh: Entwicklung des mittleren Quadraturfehlers über die Knotenzahl für stückweise polynomielle Basisfunktionen mit Grad 6

5.1.3. Zwischenfazit

Man sieht anhand dieser Beispiele, die nicht aus dem UQ-Zusammenhang sind, dass für unterschiedliche Szenarien unterschiedliche Approximationsmethoden unterschiedlich gute Ergebnisse liefern. Das bedeutet, dass der Nutzen der implementierten Verfahren nicht allgemein angegeben werden kann im Sinne von "stückweise polynomiell ist immer besser oder schlechter als stückweise linear" oder "Adaptivität ist grundsätzlich nützlich oder nicht zielführend". Stattdessen muss in Abhängigkeit der Zielfunktion $u(x)$ situativ entschieden werden, welche Methode die geeignetste ist.

Vor dem Übergang zu einem größeren UQ-Problem sollen die beiden Hierarchisierungsmethoden miteinander verglichen werden.

5.2. Vergleich der Hierarchisierungsmethoden

Um die Hierarchisierungsmethoden miteinander zu vergleichen, wurde die Betragsfunktion

$$u_1(x, y) = \frac{1}{|0.3 - x^2 - y^2| + 0.1} \tag{5.6}$$

aus Abschnitt 5.1.1 für Polynomgrad $p = 2$ und $p = 10$ mit einem regulären dünnen Gitter ohne Randpunkte auf den Levels $l = [10, \ldots, 15]$ sowohl direkt als auch rekursiv hierarchisiert und die jeweils benötigte Zeit gemessen.

Die Ergebnisse zeigen Tabelle 5.1 auf Seite 58 für Polynomgrad $p = 2$ und Tabelle 5.2 auf Seite 58 für Polynomgrad $p = 10$. Außerdem sind die Resultate in Abb. 5.12 und Abb. 5.13 auf Seite 59 auf einer zeit-logarithmischen Skala geplottet. Dabei erkennt man für beide Hierarchisierungsformen

einen exponentiellen Zusammenhang zwischen Gitterlevel und zur Hierarchisierung benötigter Zeit. Zusätzlich finden sich im Anhang Plots der benötigten Zeit für die Hierarchisierung über der Knotenzahl auf einer doppelt-logarithmischen Skala. Dabei kann man den linearen Zusammenhang zwischen Knotenzahl und Zeit erkennen.

Offenkundig ist die direkte Hierarchisierung deutlich effizienter als die rekursive Hierarchisierung, unabhängig von Polynomgrad oder Level. Insbesondere zeigt die direkte Hierarchisierung das erwartete Verhalten, für die doppelte Anzahl Knoten doppelt so viel Zeit für die Hierarchisierung zu benötigen. Das ist bei der rekursiven Hierarchisierung nicht der Fall.

Tabelle 5.1.: Vergleich der zur Hierarchisierung benötigten Zeit für Polynomgrad $p = 2$ für direkte und rekursive Hierarchisierung

Hierarchisierungsmethode	Level	benötigte Zeit
direkt	10	0.023 s
rekursiv	10	0.177 s
direkt	11	0.047 s
rekursiv	11	1.044 s
direkt	12	0.110 s
rekursiv	12	11.283 s
direkt	13	0.254 s
rekursiv	13	52.391 s
direkt	14	0.535 s
rekursiv	14	3 min 41.106 s
direkt	15	1.164 s
rekursiv	15	15 min 42.843 s

Tabelle 5.2.: Vergleich der zur Hierarchisierung benötigten Zeit für Polynomgrad $p = 10$ für direkte und rekursive Hierarchisierung

Hierarchisierungsmethode	Level	benötigte Zeit
direkt	10	0.039 s
rekursiv	10	0.256 s
direkt	11	0.077 s
rekursiv	11	1.348 s
direkt	12	0.143 s
rekursiv	12	12.923 s
direkt	13	0.315 s
rekursiv	13	1 min 00.552 s
direkt	14	0.682 s
rekursiv	14	4 min 17.515 s
direkt	15	1.490 s
rekursiv	15	18 min 16.895 s

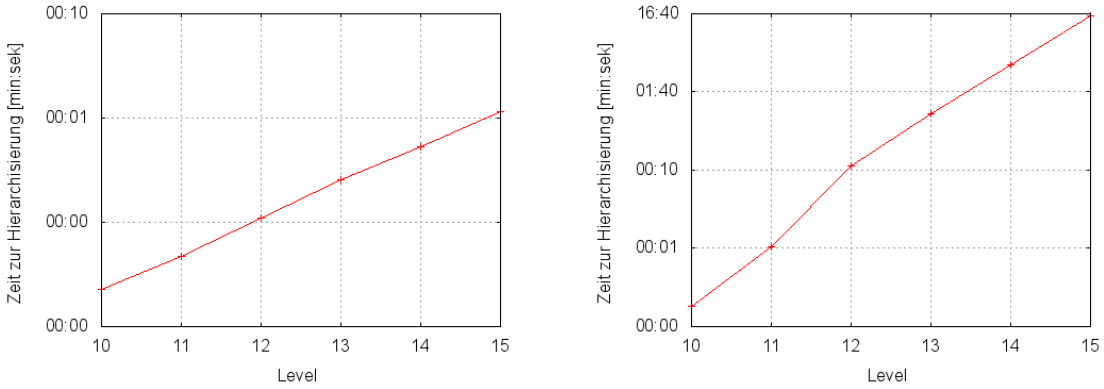

Abbildung 5.12.: Plot der benötigten Zeit zur Hierarchisierung von u_1 mit stückweise polynomiellen Basisfunktionen von Polynomgrad $p = 2$ über das Gitterlevel. Links die direkte Hierarchisierung, rechts die rekursive.

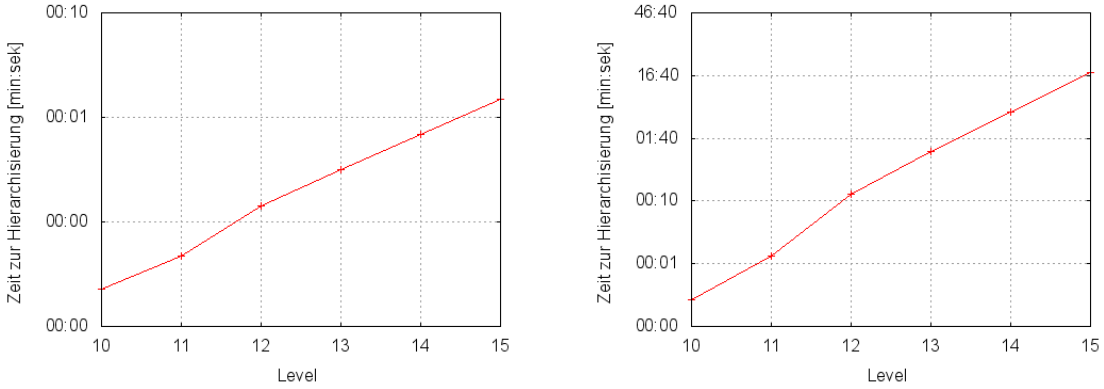

Abbildung 5.13.: Plot der benötigten Zeit zur Hierarchisierung von u_1 mit stückweise polynomiellen Basisfunktionen von Polynomgrad $p = 10$ über das Gitterlevel. Links die direkte Hierarchisierung, rechts die rekursive.

Das schlechte Abschneiden der rekursiven Hierarchisierung liegt u. a. daran, dass in der rekursiven Hierarchisierung der Dünngitter-Baum nicht nur einmal, sondern p-fach durchlaufen wird. Dabei müssen die $\alpha_{l,i}$-Koeffizienten des rechten Teilbaums, die aus Symmetriegründen nur implizit gespeichert sind, jeweils rekonstruiert werden. Dazu sind viele if - Konstrukte nötig, deren Auswertung Zeit kostet.

Zusätzlich muss in jedem Durchlauf der Zugriff auf die Speicherstrukutur der Gitterknoten öfter benutzt werden. Einmal für den Knoten selbst und einmal für seinen direkten hierarchischen Vorgänger, um den Index des zum Knoten gehörenden hierarchischen Überschusses im Überschussvektor zu finden. Zusätzlich muss der Überschussvektor, wie in Abschnitt 4.1.2 beschrieben, kopiert werden.

5.3. Kármán'sche Wirbelstraße

Im Rahmen einer Vorlesung an der betreuenden Abteilung wurde von Studenten ein Strömungslöser implementiert, der verwendet wird, um das in [Lei13, Kap. 3.2] beschriebene Experiment nachzustellen. Dabei zeigte sich, dass die inverse Rosenblatt Transformation nicht die gewünschten Ergebnisse liefert. Das liegt daran, dass in $[0,1]^d$ interpoliert wird, die Berechnung aber in Γ erfolgt und dazwischen eine nicht-lineare Transformation liegt, die zu Problemen führt, die bei einer linearen Transformation nicht auftreten. Daher wurde im Rahmen dieser Arbeit lediglich linear transformiert. Um zu besseren Ergebnissen als [Lei13] zu kommen, wurden drei Maßnahmen ergriffen:

1. Die Einführung von räumlicher Adaptivität,
2. die Erhöhung des maximalen Levels für CSGC auf $l_{\max} = 6$, d. h. bis zu $m = 2561$ Gitterknoten sowie
3. der Ansatz stückweise polynomieller Basisfunktionen.

5.3.1. Versuchsaufbau

Simuliert wird eine Kármán'sche Wirbelstraße, in die von links eine Flüssigkeit einströmt. Diese trifft nach 0.7 Längeneinheiten auf ein in der Mitte des Strömungskanals platziertes, schräg stehendes Hindernis. Dabei entsteht hinter dem Hindernis eine Turbulenz, die sich bis zum Ende des Strömungskanals fortpflanzt. Die Länge des Strömungskanals beträgt 10 Längeneinheiten und die Breite 2 Längeneinheiten.

In jedem Simulationsaufruf werden 500 Zeitschritte zwischen $t = 0$ und $t = 50$ berechnet und in jedem Zeitschritt der Druck an der Messstelle, 6 Längeneinheiten von der Einströmungsöffnung entfernt, über die gesamte Breite des Kanals gemessen und dessen arithmetisches Mittel gebildet (P_{y60}). Die räumliche Diskretisierung des Kanals erfolgt über quadratische Gitterzellen mit Seitenlänge 0.1 Längeneinheiten.

Die Geometrie ist mitsamt einigen exemplarischen Gitterzellen in Abb. 5.14 dargestellt.

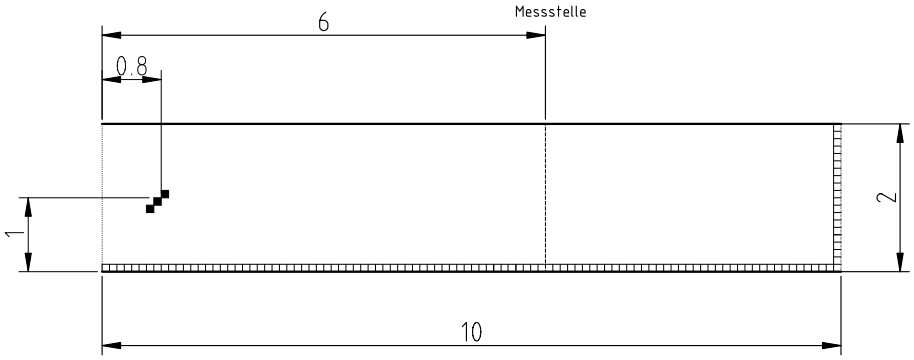

Abbildung 5.14.: Geometrie der Kármán'schen Wirbelstraße. Die Interpolationszellen sind stellvertretend als Hinweis auf die Größe in der Spalte am rechten Rand der Geometrie und der untersten Zeile angedeutet.

Verglichen mit [Lei13] wird also eine leicht modifizierte Versuchsgeometrie mit kleinerem Hindernis verwendet.

Als unsichere Parameter werden die beiden vektoriellen Komponenten der Einströmgeschwindigkeit u und v, die Reynolds-Zahl sowie die Prandtl-Zahl angenommen, so dass sich ein vierdimensionales UQ-Problem ergibt. Die Reynolds-Zahl ist eine dimensionslose Größe, die den Zusammenhang zwischen Trägheits- und Zähigkeitskräften in einem Fluid oder Gas beschreibt. Die Prandtl-Zahl ist ebenso wie die Reynolds-Zahl eine dimensionslose Größe und beschreibt das Verhältnis von kinematischer Viskosität zur Temperaturleitfähigkeit in Gasen oder Fluiden.

Im durchgeführten UQ-Experiment werden die Parameter in den folgenden Intervallen normalverteilt variiert:

- Reynolds-Zahl: $(8000, 12000)$
- $\vec{x}$ Komponente der Einströmgeschwindigkeit: $(2.0, 5.0)$ $\left[\frac{\text{Längeneinheiten}}{\text{Zeitschritt}}\right]$
- $\vec{y}$ Komponente der Einströmgeschwindigkeit: $(-0.5, 0.5)$ $\left[\frac{\text{Längeneinheiten}}{\text{Zeitschritt}}\right]$
- Prandtl-Zahl: $(7, 13)$

Als Normalverteilung wird die Funktion "truncnorm" aus dem stats-Modul der Python Erweiterung scipy[1] auf dem Intervall $[a, b]$ mit Erwartungswert $\mu = b - \frac{b-a}{2}$ und Standardabweichung $\sigma = \frac{b-a}{6}$ verwendet. Die Funktion schneidet die Bereiche außerhalb des definierten Intervalls ab und verlegt den dort im Vergleich zu einer Standard-Normalverteilung abgeschnittenen Flächenanteil ins Intervall, so dass das Integral über die "truncnorm"-Funktion über das Intervall $[a, b]$ den Wert 1 ergibt. Dies ist nötig, um zu verhindern, dass das Ergebnis der Dünngitter-Quadratur mit jeder Dimension kleiner skaliert und damit verfälscht wird.

Die Transformationen aus dem Wertebereich der Parameter auf $[0, 1]$ und umgekehrt erfolgen linear. Als Richtgrößen für verschiedene SGCM-Verfahren wurden 5000 MC Samples und 2500 QMC Samples nach der Sobol - Sequenz generiert.

Diese werden mit CSGC für stückweise polynomielle sowie CSGC für stückweise lineare Basisfunktionen von Gitterlevel $l = 4$ bis $l = 6$ ohne Randpunkte verglichen. Dabei wurden die stückweise polynomiellen Basisfunktionen unter Ausnutzung des maximal möglichen Polynomgrads angesetzt, d. h. $p = l + 1$. Außerdem werden Gitter mit Level $l = 1$ ohne Randpunkte für ASGC angesetzt. Diese Gitter werden jeweils mit einer der drei in Abschnitt 3.3 vorgeschlagenen Verfeinerungskriterien einmal für stückweise lineare und einmal für stückweise polynomielle Basisfunktionen von Polynomgrad $p = 10$ verfeinert. Dabei wird so lange verfeinert, bis für die Zahl der Dünngitter-Knoten m vor dem nächsten Verfeinerungsschritt gilt $m > 1000$.

[1] http://docs.scipy.org/doc/scipy-0.13.0/reference/generated/scipy.stats.truncnorm.html

5.3.2. Ergebnisse

Zur Bewertung der Ergebnisse werden die Simulationsergebnisse nach den einzelnen Verfahren sowie den angesetzten Basisfunktionen gruppiert. D. h. es werden fünf Gruppen gebildet

- QMC,
- CSGC mit stückweise linearen und stückweise polynomiellen Basisfunktionen sowie
- ASGC mit stückweise linearen und stückweise polynomiellen Basisfunktionen.

Die Ergebnisse der einzelnen Gruppen für Erwartungswert und Varianz werden jeweils zusammen mit dem Ergebnis der MC Lösung geplottet. Für die geeigneten Approximationen der Gruppe wird anschließend die Differenz zwischen dem Ergebnis eines Elements aus der Gruppe und dem Ergebnis der MC Methode gezeichnet, um die Ergebnisse genauer zu analysieren. Abschließend werden die geeignetsten Ergebnisse aus allen Gruppen miteinander verglichen.

Der Erwartungswert der Dünngitter-Approximation wird berechnet als

$$\mathbb{E}(x) = \int_{\Gamma} f_N(x) \cdot p(x)\, dx. \tag{5.7}$$

Wegen der Verwendung einer Normalverteilung kann eine Zielfunktion entstehen, die einer Gauss-Kurve ähnelt. Nach [Pfl10, Kap. 4.2] können bei der Approximation von Gauss-Kurven mit dünnen Gittern Überschwingungen auftreten, die dazu führen, dass das Ergebnis der Quadraturberechnung bis hin zu negativen Werten verfälscht werden kann. Dieses Phänomen kann insbesondere bei der Verwendung stückweise polynomieller Basisfunktionen auftreten. Daher wurden die Ergebnisse für Erwartungswert und Varianz für die beiden Dünngitter-Verfahren CSGC und ASGC auf zwei verschiedene Arten berechnet:

1. Analytisch durch die direkte Berechnung der Quadratur der Dünngitterapproximation $f_N(x)$ der Zielfunktion $u(x)$ nach Gleichung (5.7) und
2. näherungsweise mithilfe der MC Quadratur

   $$\mathbb{E}(x) = \int_{\Gamma} f_N(x) \cdot p(x)\, dx \approx \frac{1}{n} \sum_{j=1}^{n} f_N(x_j) \cdot p(x_j). \tag{5.8}$$

   Dazu wurden $n = 40000$ Samples aus Γ gezogen und in die Dünngitterapproximation $f_N(x)$ eingesetzt.

Referenz

Vor Betrachtung der Ergebnisse aus SGCM werden zunächst die beiden Vergleichsverfahren verglichen, MC und QMC. In Abb. 5.15 auf Seite 63 sieht man, dass QMC für den Erwartungswert den gleichen Referenzwert liefert wie MC. Abb. 5.16 auf Seite 63 zeigt das noch deutlicher: Der Betrag der Differenz liegt konstant unter 30 und die relative Abweichung damit unter 2%.

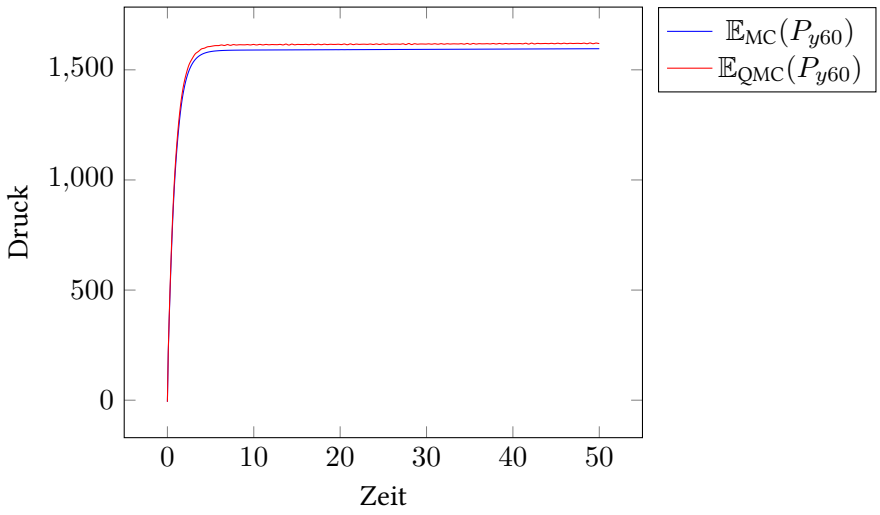

Abbildung 5.15.: Wirbelstraße: Erwartungswert: MC, QMC

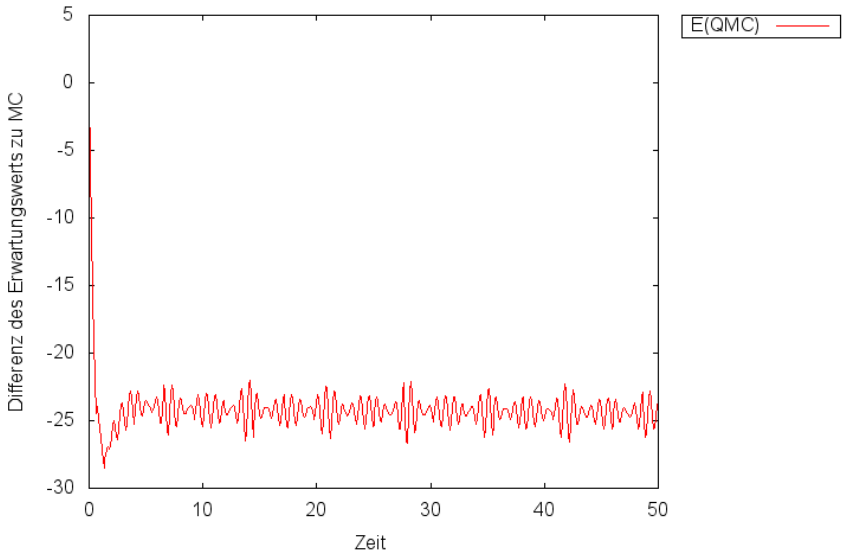

Abbildung 5.16.: Wirbelstraße: Erwartungswertdifferenz: QMC zu MC

Betrachtet man dagegen die Varianz, s. Abb. 5.17 auf Seite 64, ergibt sich zunächst kein klares Bild, weil QMC und MC ein qualitativ gleiches Verhalten auf quantitativ unterschiedlichen Niveaus zeigen. Da die im Folgenden vorgestellten Werte von SGCM gegen den Wert der MC Quadratur streben, wird im Weiteren die MC Quadratur als Referenzlösung betrachtet. Der Wert der Varianz liegt damit für QMC konstant über $1 \cdot 10^5$ oberhalb der MC Referenzlösung, was einem relativen Fehler von über 0.5 entspricht.

Im Folgenden werden die Ergebnisse der SGCM präsentiert. Diese sind so geordnet, dass zunächst die Ergebnisse für reguläre dünne Gitter, d. h. der CSGC Methode, gezeigt werden und anschlie-

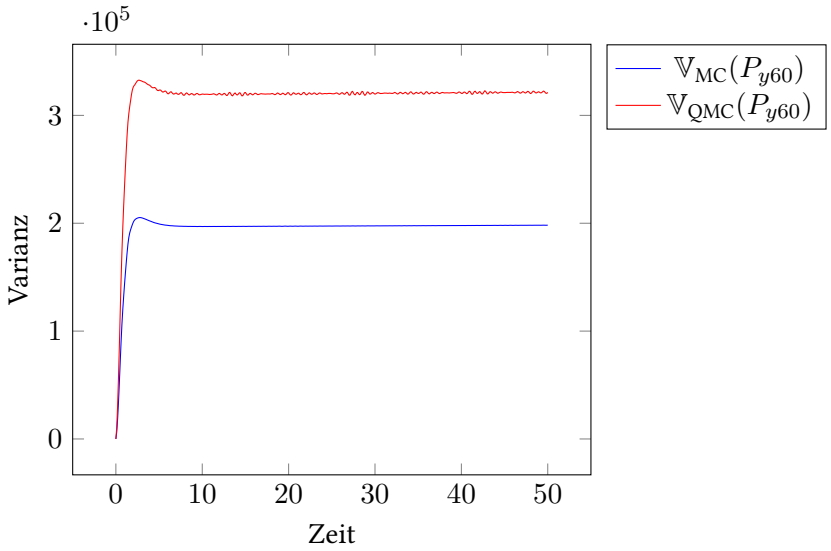

Abbildung 5.17.: Wirbelstraße: Varianz: MC, QMC

ßend die von ASGC. Dabei wird jeweils mit linearen Basisfunktionen und analytischer Quadratur des Dünngitter-Interpolanten begonnen, bevor die Ergebnisse der MC Quadratur gezeigt werden. Anschließend werden in der gleichen Reihenfolge die Ergebnisse der stückweise polynomiellen Basisfunktionen vorgestellt.

CSGC

CSGC liefert, wie Abb. 5.18 auf Seite 65 zeigt, bei analytischer Quadratur und linearen Basisfunktionen eine gute Approximation für den Erwartungswert.

Nach Abb. 5.19 auf Seite 65 liegt die Abweichung zwischen -40 und 5, d. h. der relative Fehler liegt unter 5%. Dass die Abweichung für Level $l = 5$ geringer ist als für Level $l = 6$, entsteht wahrscheinlich durch günstige Auslöschung von Fehlern auf Level $l = 5$.

Für die Varianz ergibt sich ein ähnliches Bild wie für den Erwartungswert. Nach Abb. 5.20 auf Seite 66 zeigt das CSGC Verfahren für Level $l = 6$ und damit $m = 2561$ Gitterknoten keine große Abweichung mehr von der MC Referenzlösung.

Mithilfe von Abb. 5.21 auf Seite 66 lässt sich die Abweichung genauer quantifizieren, auf eine relative Abweichung von der MC Referenzlösung von ca. 0.02.

Verwendet man anstelle der analytischen Quadratur eine MC Quadratur, sollte sich am Niveau der Ergebnisse nichts substanziell ändern, es sei denn, es wird eine Zielfunktion $u(x)$ mit Gauss'scher Form approximiert. In diesen Fällen sollte sich eine Besserung gegenüber der analytischen Quadratur einstellen, weil die MC Quadratur von den zu Beginn des Abschnitts beschriebenen Überschwingungen nicht betroffen ist. Verfahrensbedingt oszillieren die Ergebnisse durch die niedrige Konvergenzgeschwindigkeit der MC Quadratur jedoch deutlich stärker als mit analytischer Quadratur.

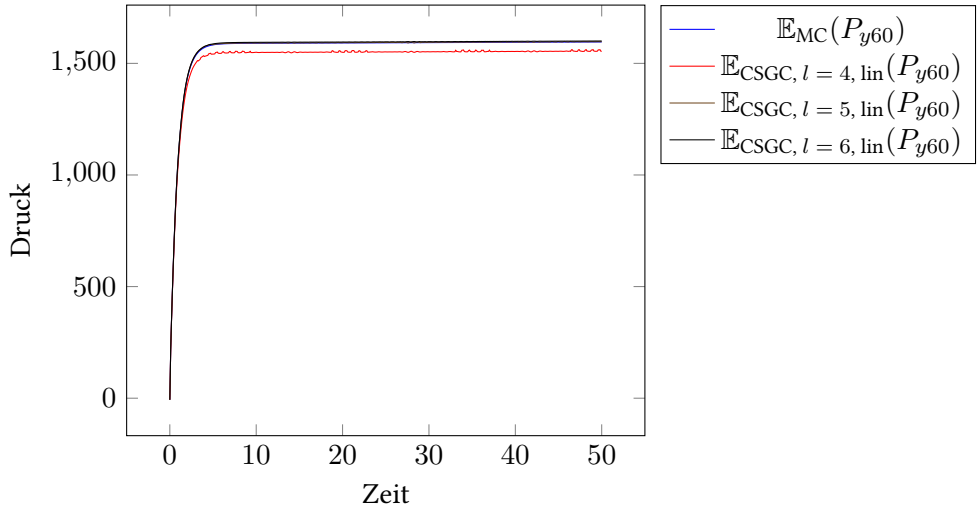

Abbildung 5.18.: Wirbelstraße: Erwartungswert: CSGC Level $l = [4, 5, 6]$, keine Randpunkte, stückweise lineare Basisfunktionen, analytische Quadratur

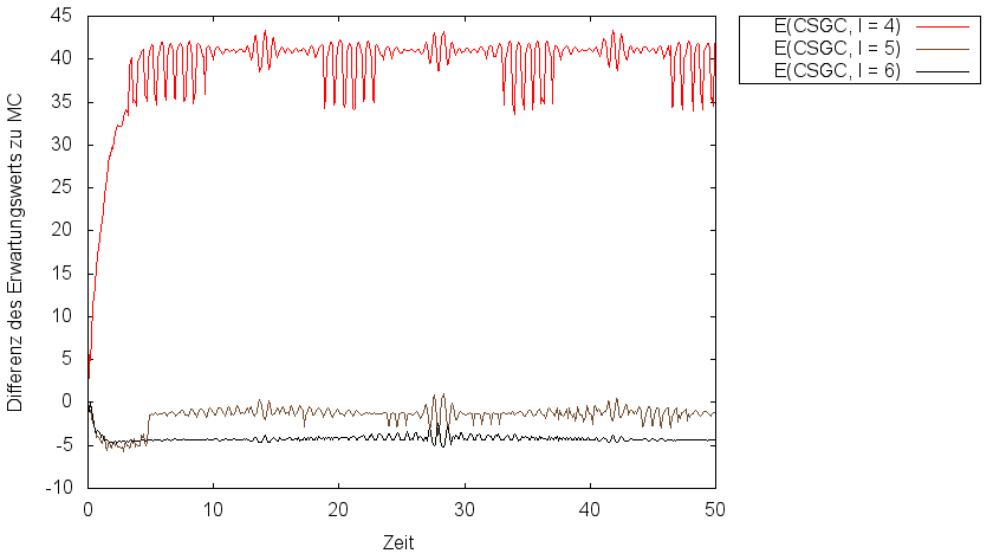

Abbildung 5.19.: Wirbelstraße: Erwartungswertdifferenz: CSGC Level $l = [4, 5, 6]$, keine Randpunkte, stückweise lineare Basisfunktionen, analytische Quadratur zu MC

Abb. 5.22 auf Seite 67 zeigt für die Berechnung des Erwartungswerts mit MC Quadratur ein sehr ähnliches Bild wie die Verwendung analytischer Quadratur.

Nach Abb. 5.23 auf Seite 67 und Abb. 5.19 trifft das Ergebnis der analytischen Lösung das Ergebnis der MC Referenzlösung besser als die MC Quadratur.

Wie Abb. 5.24 auf Seite 68 und Abb. 5.25 auf Seite 68 zeigen, lassen sich auf Grundlage des vorhandenen Datenmaterials keine sinnvollen Aussagen über die Varianzapproximation machen, weil das

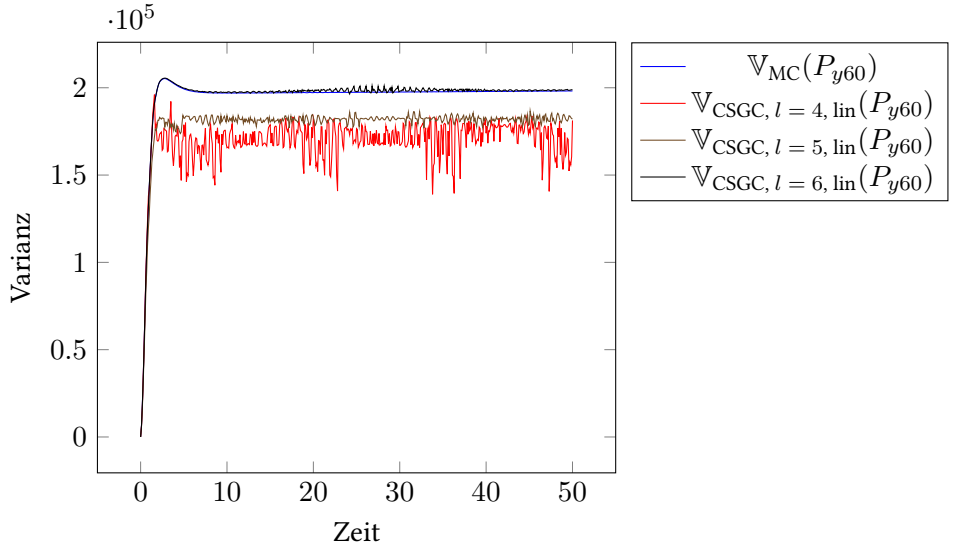

Abbildung 5.20.: Wirbelstraße: Varianz: CSGC Level $l = [4, 5, 6]$, keine Randpunkte, stückweise lineare Basisfunktionen, analytische Quadratur

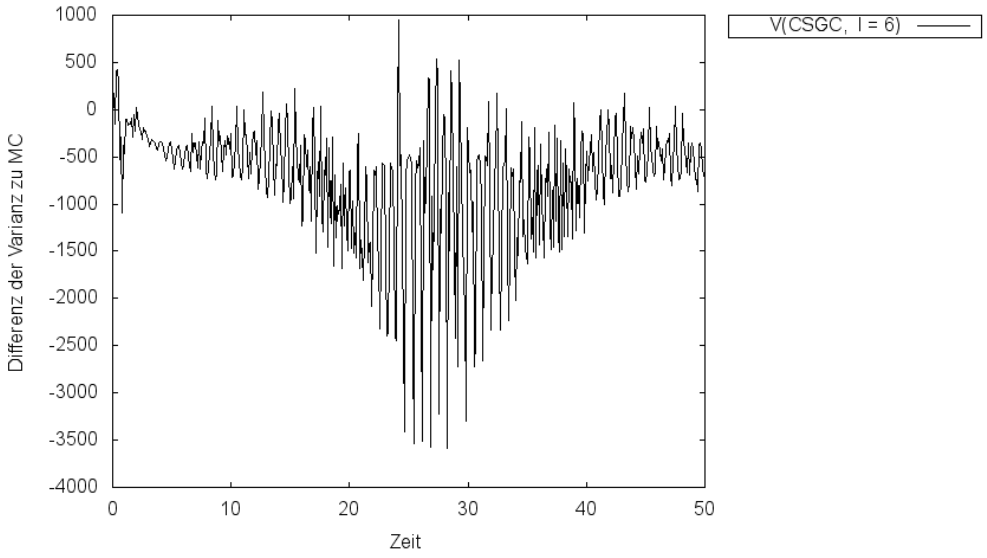

Abbildung 5.21.: Wirbelstraße: Varianzdifferenz: CSGC Level $l = 6$, keine Randpunkte, stückweise lineare Basisfunktionen, analytische Quadratur zu MC

Rauschen der Daten zu stark ist. Da sich dieses Phänomen für alle weiteren gezeigten Verfahren in der Varianzbeschreibung einstellt, finden sich die Plots der Varianzberechnung mithilfe von MC Quadratur für alle übrigen Verfahren im Anhang.

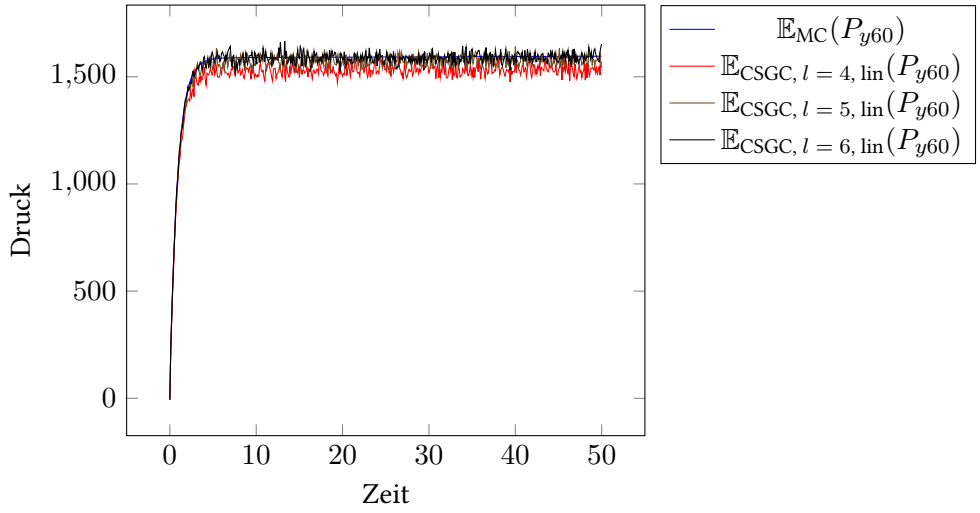

Abbildung 5.22.: Wirbelstraße: Erwartungswert: CSGC Level $l = [4, 5, 6]$, keine Randpunkte, stückweise lineare Basisfunktionen, MC Quadratur

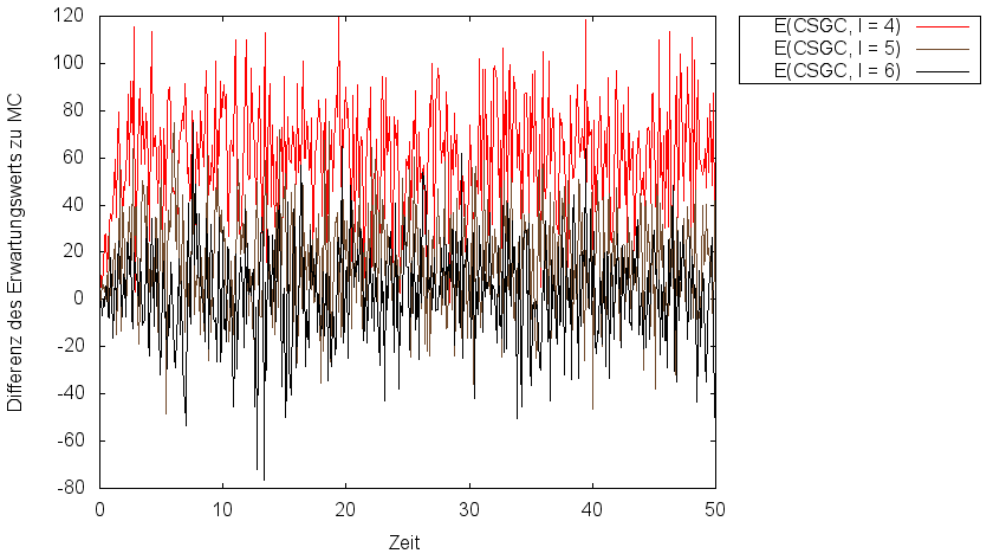

Abbildung 5.23.: Wirbelstraße: Erwartungswertdifferenz: CSGC Level $l = [4, 5, 6]$, keine Randpunkte, stückweise lineare Basisfunktionen, MC Quadratur zu MC

Setzt man anstelle der stückweise linearen Basisfunktionen stückweise polynomielle Basisfunktionen an, ergibt sich nach Abb. 5.26 auf Seite 69 für die Level $l = [4, 5]$ ein schlechteres Ergebnis. Auf Level $l = 6$ ergibt sich hingegen ein ähnliches Fehlerbild wie im stückweise linearen Fall.

Untersucht man den Fehler genauer, zeigt sich nach Abb. 5.27 auf Seite 69 für Level $l = 5$ ein eindeutig schlechteres Verhalten als im stückweise linearen Fall, für $l = 6$ liegen die stückweise polynomiellen Basisfunktionen dagegen ungefähr gleichauf.

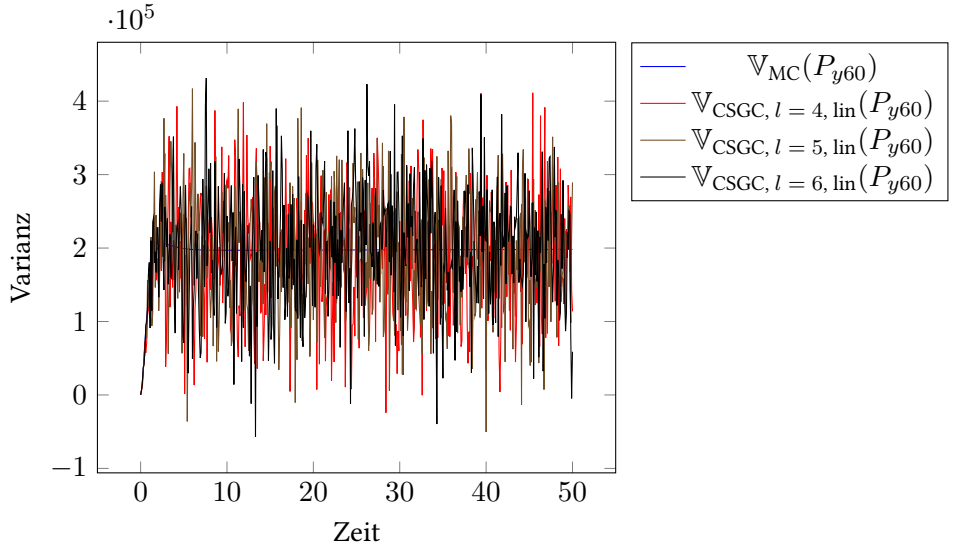

Abbildung 5.24.: Wirbelstraße: Varianz: CSGC Level $l = [4, 5, 6]$, keine Randpunkte, stückweise lineare Basisfunktionen, MC Quadratur

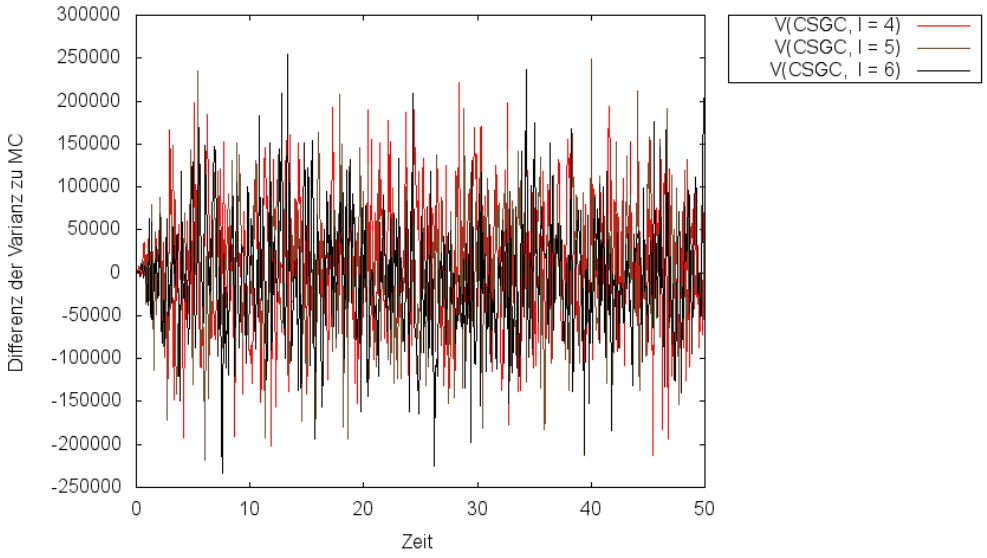

Abbildung 5.25.: Wirbelstraße: Varianzdifferenz: CSGC Level $l = [4, 5, 6]$, keine Randpunkte, stückweise lineare Basisfunktionen, MC Quadratur zu MC

Nach Abb. 5.28 auf Seite 70 zeigt die Approximation auf Level $l = 5$ eine negative Varianz. Möglicherweise zeigt sich hier das eingangs angesprochene Problem für die Interpolation von Gauss-Funktionen, so dass sich an dieser Stelle bei Verwendung der direkten Quadraturberechnung von $f_N(x)$ die negative Varianz ergibt. Andere Ursachen sind allerdings ebenfalls denkbar, wie bspw. Oszillationen in der Zielfunktion, so dass auf Level 5 korrigierende negative Überschüsse entstehen o. ä.. Zudem ist zu erkennen, dass die Approximation auf Level $l = 6$ die Referenzlösung sehr gut trifft.

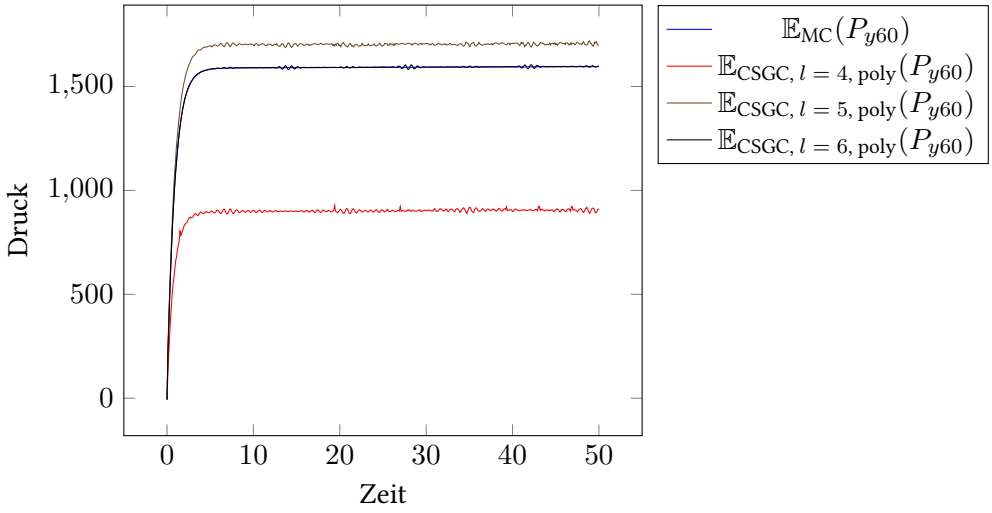

Abbildung 5.26.: Wirbelstraße: Erwartungswert CSGC Level $l = [4, 5, 6]$, keine Randpunkte, stückweise polynomielle Basisfunktionen, analytische Quadratur

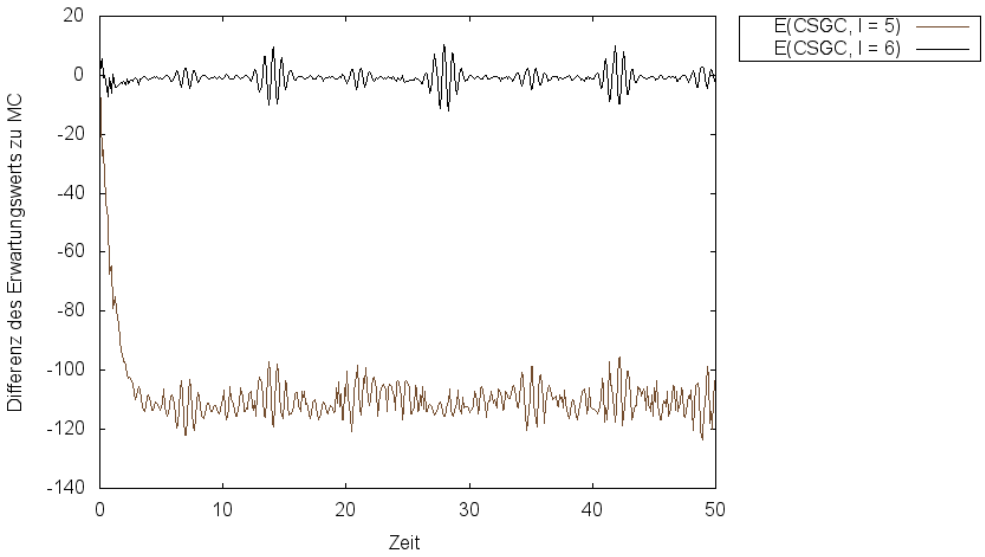

Abbildung 5.27.: Wirbelstraße: Erwartungswertdifferenz: CSGC Level $l = [5, 6]$, keine Randpunkte, stückweise polynomielle Basisfunktionen, analytische Quadratur zu MC

In Abb. 5.29 auf Seite 70 zeigt sich jedoch, dass der stückweise polynomielle Ansatz bei der Approximation der Varianz keinen Gewinn gegenüber stückweise linearen Basisfunktionen bringt, im Gegenteil. Der relative Fehler für die Varianz-Approximation liegt zwischen 5 und 10%.

Nach Abb. 5.30 auf Seite 71 bringt der Einsatz der MC Quadratur Methode verglichen mit der analytischen Quadratur für stückweise polynomielle Basisfunktionen für die Level $l = [4, 5]$ bei der Berechnung des Erwartungswerts deutliche Gewinne bei der Approximationsgenauigkeit.

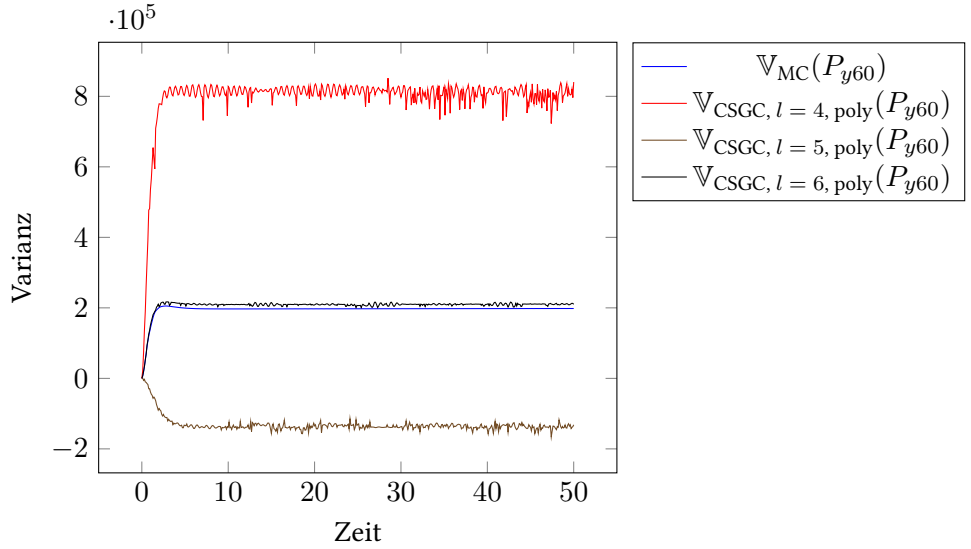

Abbildung 5.28.: Wirbelstraße: Varianz: CSGC Level $l = [4, 5, 6]$, keine Randpunkte, stückweise polynomielle Basisfunktionen, analytische Quadratur

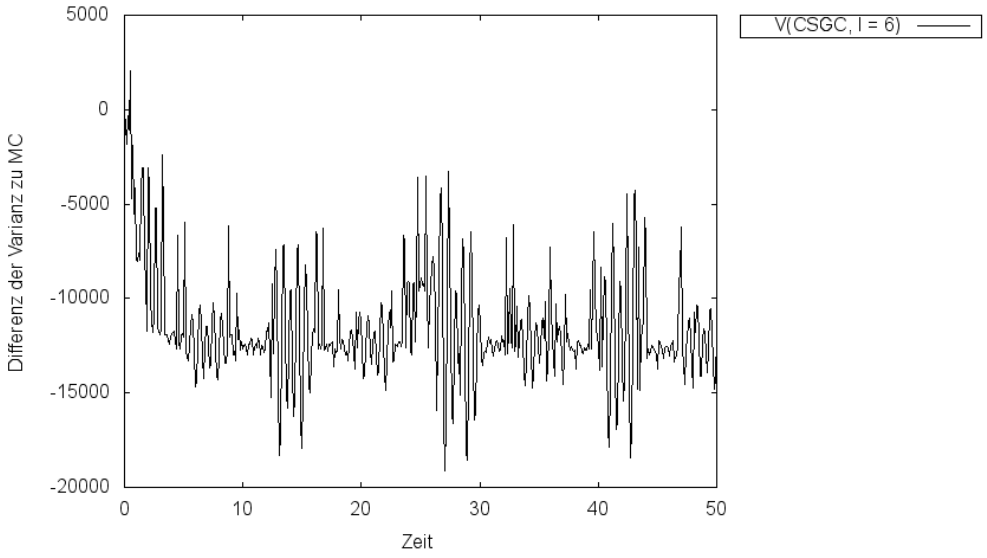

Abbildung 5.29.: Wirbelstraße: Varianzdifferenz: CSGC Level $l = 6$, keine Randpunkte, stückweise polynomielle Basisfunktionen, analytische Quadratur zu MC

Diese liegen, wie auch Abb. 5.31 auf Seite 71 zeigt, im Gegensatz zur analytischen Quadratur auf gleichem Niveau wie Level $l = 6$, der wiederum eine gute Approximation für die Referenzlösung des Erwartungswerts darstellt.

Außerdem tritt bei Verwendung der MC Quadratur auf Level $l = 5$ keine negative Varianz mehr auf, vgl. Abb. A.8 im Anhang.

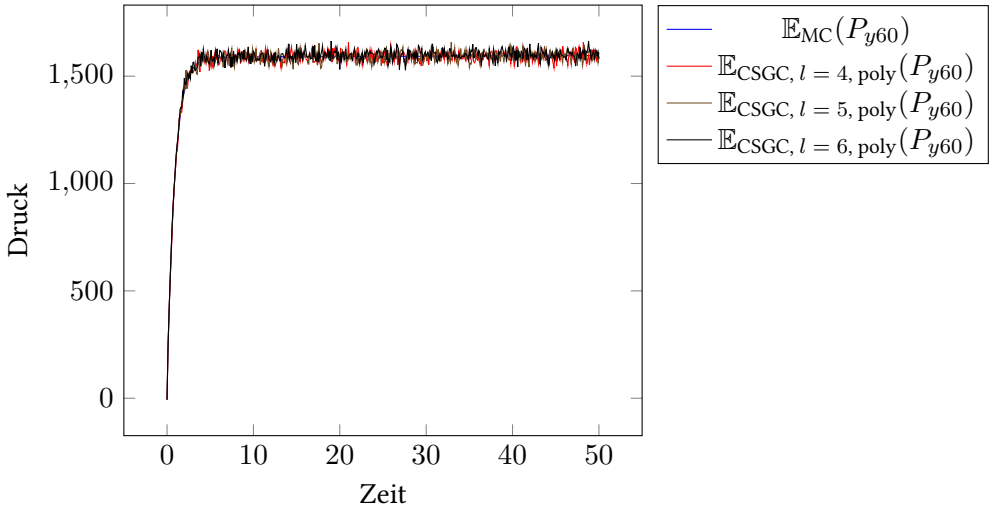

Abbildung 5.30.: Wirbelstraße: Erwartungswert: CSGC Level $l = [4, 5, 6]$, keine Randpunkte, stückweise polynomielle Basisfunktionen, MC Quadratur

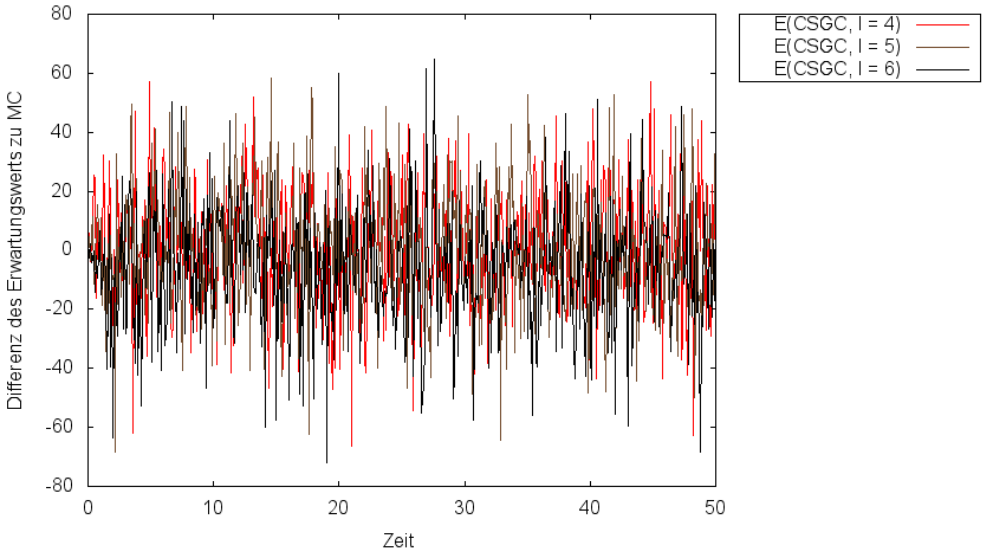

Abbildung 5.31.: Wirbelstraße: Erwartungswertdifferenz: CSGC Level $l = [4, 5, 6]$, keine Randpunkte, stückweise polynomielle Basisfunktionen, MC Quadratur zu MC

ASGC

Beim Einsatz von adaptiver Verfeinerung ergibt sich mit stückweise linearen Basisfunktionen und analytischer Quadratur nach Abb. 5.32 auf Seite 72 für den Erwartungswert eine sehr gute Approximation.

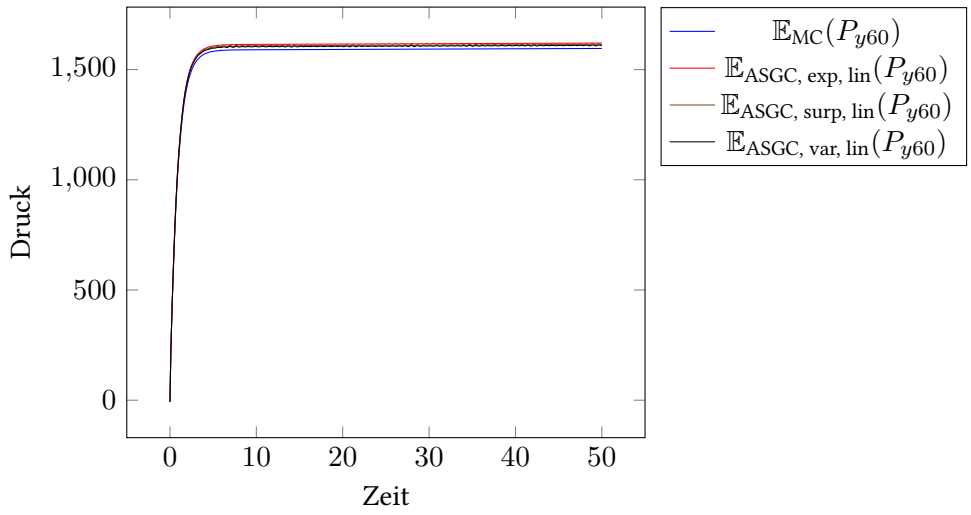

Abbildung 5.32.: Wirbelstraße: Erwartungswert: ASGC, keine Randpunkte, stückweise lineare Basisfunktionen, analytische Quadratur

Dies illustriert auch Abb. 5.33. Der Fehler bei der Approximation des Erwartungswerts liegt bei ca. 0.01 bis 0.02.

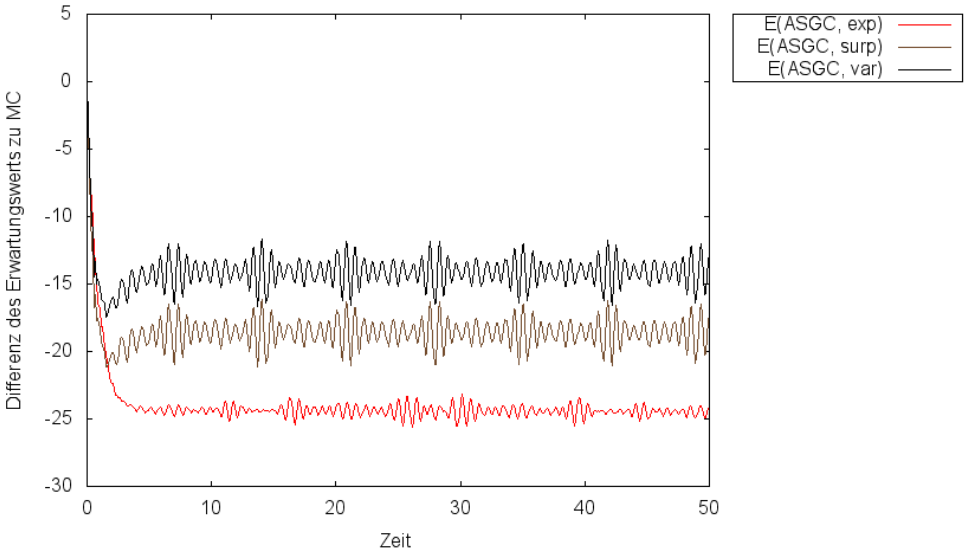

Abbildung 5.33.: Wirbelstraße: Erwartungswertdifferenz: ASGC, lineare Basisfunktionen, analytische Quadratur zu MC

Bezüglich der Varianz ist die Approximation akzeptabel, wie Abb. 5.34 auf Seite 73 und Abb. 5.35 auf Seite 73 zeigen. Der relative Fehler liegt bei ca. 25%.

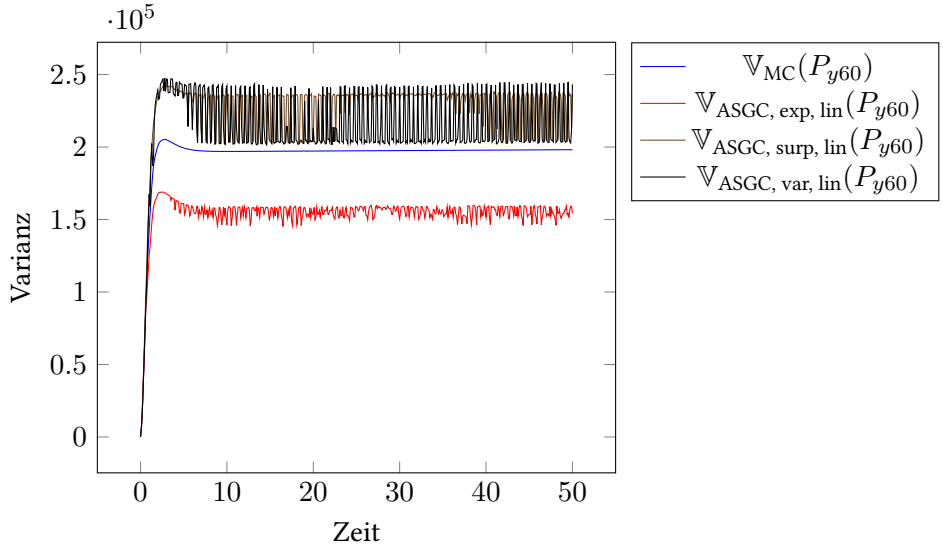

Abbildung 5.34.: Wirbelstraße: Varianz: ASGC, keine Randpunkte, stückweise lineare Basisfunktionen, analytische Quadratur

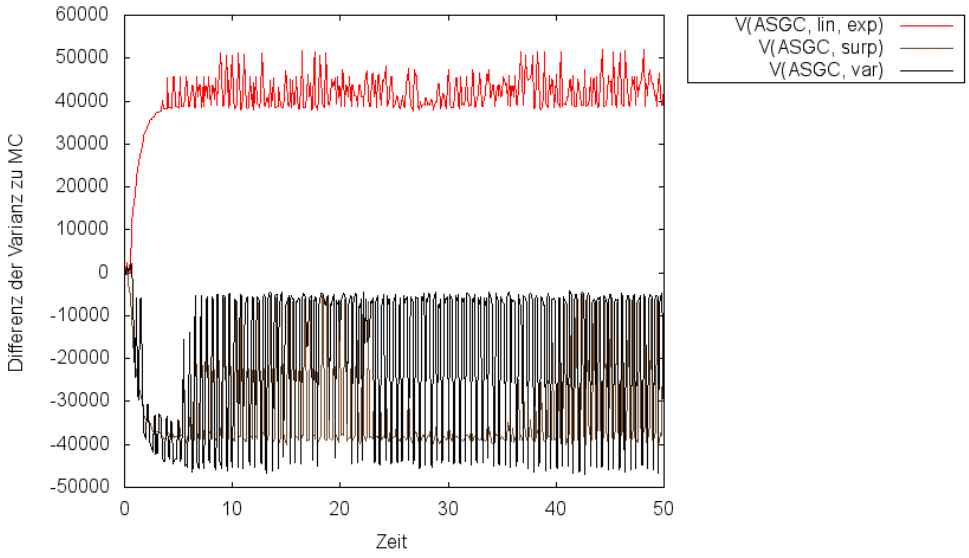

Abbildung 5.35.: Wirbelstraße: Varianzdifferenz: ASGC, keine Randpunkte, lineare Basisfunktionen, analytische Quadratur zu MC

Abb. 5.36 auf Seite 74 und Abb. 5.37 auf Seite 74 zeigen bei der Berechnung des Erwartungswerts mit MC Quadratur für das Verfeinerungskriterium nach dem Erwartungswert eine gute Konvergenz. Für das Verfeinerungskriterum nach der Varianz und nach dem Betrag der Überschüsse liegt man dagegen um 3 bis 5% über der MC Referenzlösung.

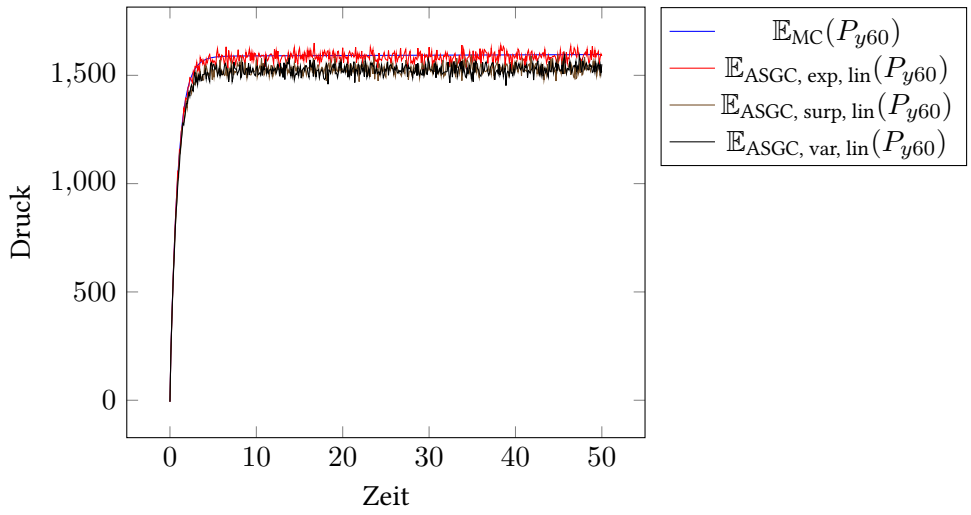

Abbildung 5.36.: Wirbelstraße: Erwartungswert: ASGC, keine Randpunkte, stückweise lineare Basisfunktionen, MC Quadratur

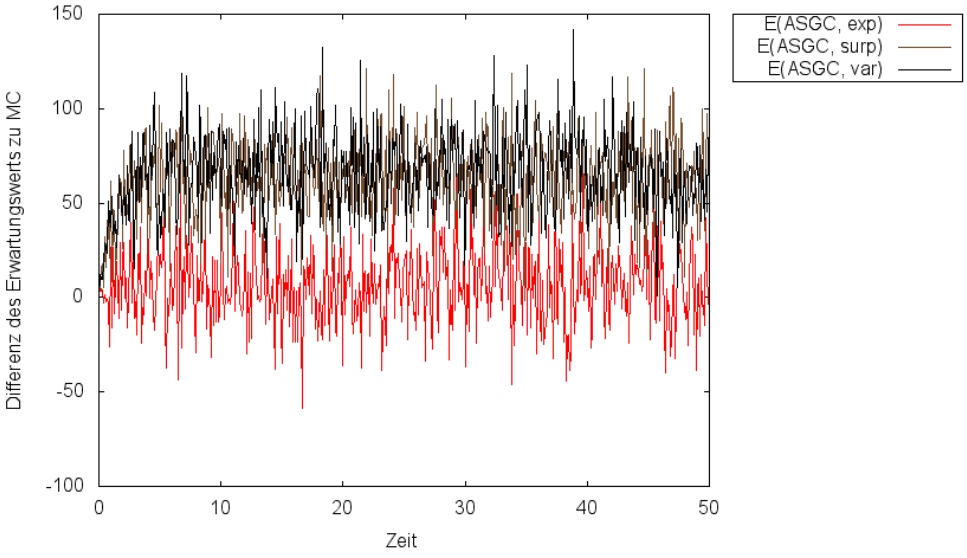

Abbildung 5.37.: Wirbelstraße: Erwartungswertdifferenz: ASGC, lineare Basisfunktionen, MC Quadratur zu MC

Beim Übergang zu stückweise polynomiellen Basisfunktionen zeigt sich bei der Approximation des Erwartungswerts mit analytischer Quadratur nach Abb. 5.38 auf Seite 75 und 5.39 auf Seite 75 eine Verschlechterung gegenüber stückweise linearen Basisfunktionen. Der relative Fehler liegt bei Verwendung des Verfeinerungskriteriums des Betrags der hierarchischen Überschüsse bei ca. 10% und im Falle der beiden Kriterien mit statistischem Hintergrund mit $3 - 5\%$ deutlich niedriger.

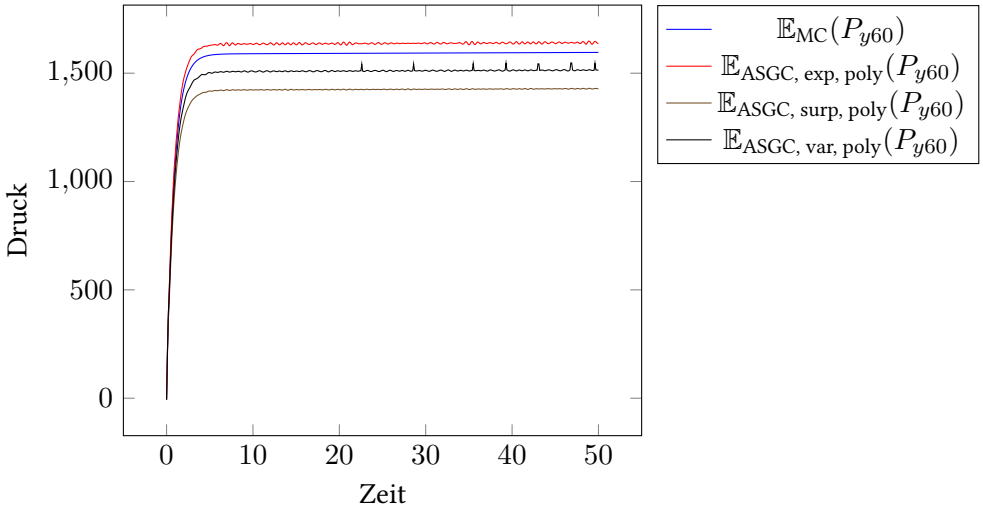

Abbildung 5.38.: Wirbelstraße: Erwartungswert: ASGC, keine Randpunkte, stückweise polynomielle Basisfunktionen, analytische Quadratur

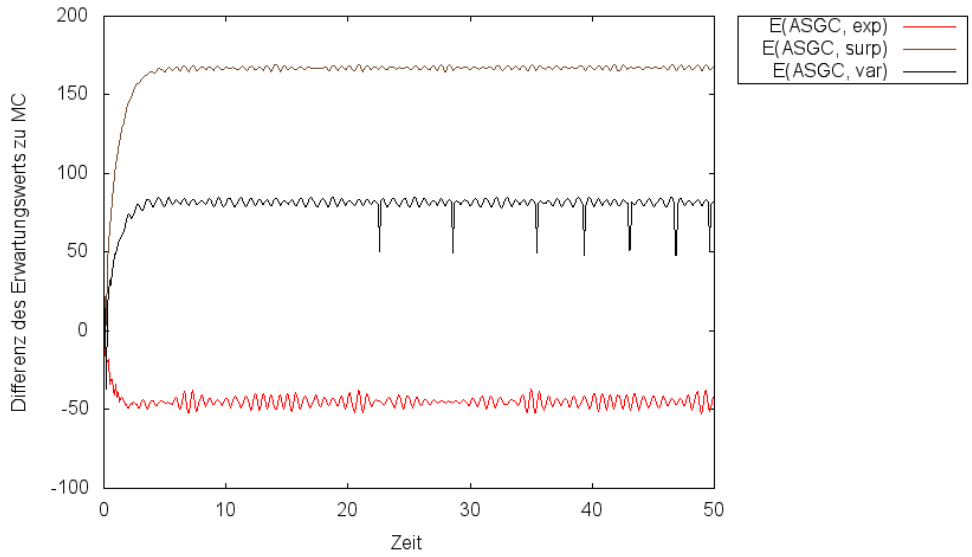

Abbildung 5.39.: Wirbelstraße: Erwartungswertdifferenz: ASGC, polynomielle Basisfunktionen, analytische Quadratur zu MC

Auch die Approximation der Varianz funktioniert weniger gut als mit stückweise linearen Basisfunktionen. Wie Abb. 5.40 auf Seite 76 zeigt, führt das Verfeinerungskriterium der Beträge der hierarchischen Überschüsse zu völlig falschen Ergebnissen. In Abb. 5.41 auf Seite 76 sieht man außerdem, dass im konkreten Fall für stückweise polynomielle Basisfunktionen die in Abschnitt 3.3 beschriebene Abschätzung der lokalen Varianz nicht gut funktioniert.

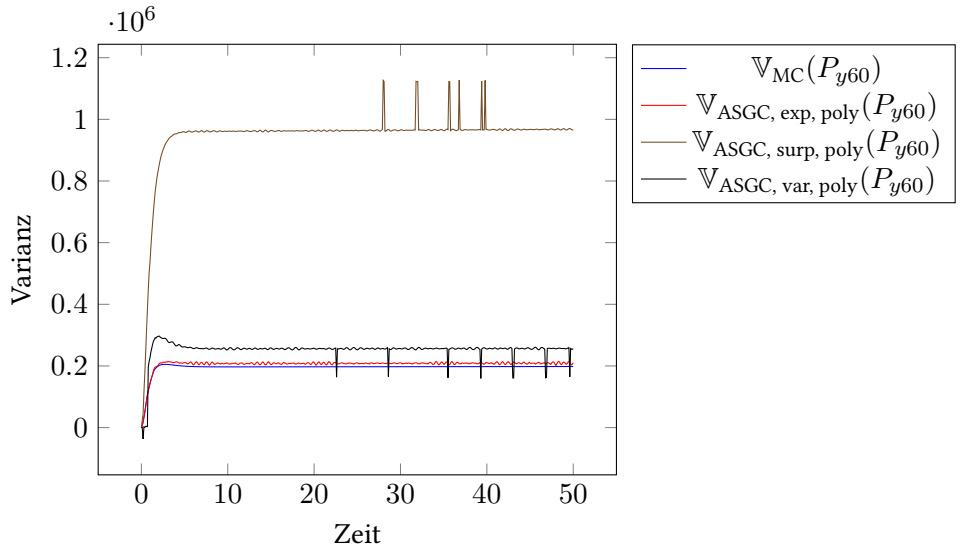

Abbildung 5.40.: Wirbelstraße: Varianz: ASGC, keine Randpunkte, stückweise polynomielle Basisfunktionen, analytische Quadratur

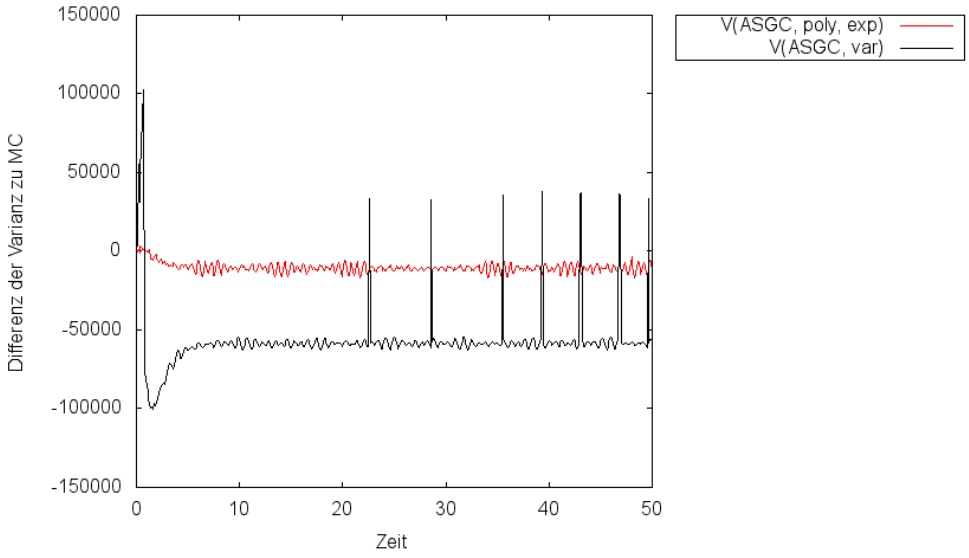

Abbildung 5.41.: Wirbelstraße: Varianzdifferenz: ASGC, polynomielle Basisfunktionen, analytische Quadratur zu MC

Abb. 5.42 auf Seite 77 und Abb. 5.43 auf Seite 77 zeigen für die MC Quadratur mit stückweise polynomiellen Basisfunktionen bei der Approximation des Erwartungswerts ein sehr ähnliches Verhalten, wie es bei der MC Quadratur im Rahmen von ASGC mit stückweise linearen Basisfunktionen in Abb. 5.36 auf Seite 74 dargestellt ist.

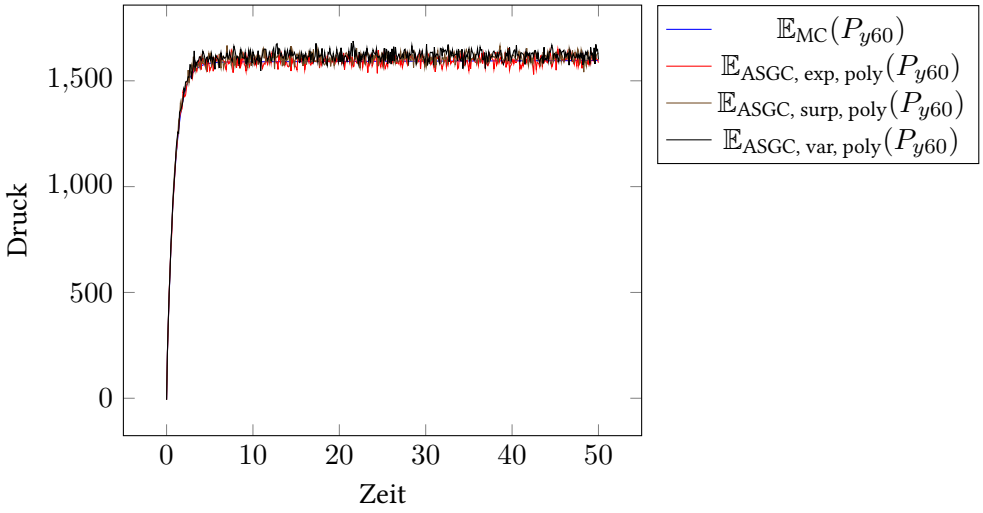

Abbildung 5.42.: Wirbelstraße: Erwartungswert: ASGC, keine Randpunkte, stückweise polynomielle Basisfunktionen, MC Quadratur

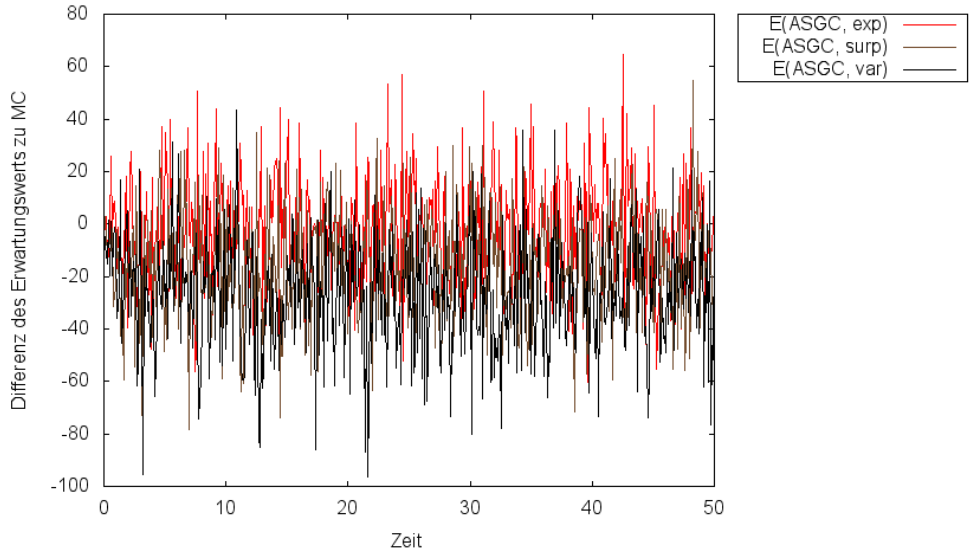

Abbildung 5.43.: Wirbelstraße: Erwartungswertdifferenz: ASGC, polynomielle Basisfunktionen, MC Quadratur zu MC

Vergleich der Verfahren

Vergleicht man die geeignesten Verfahren miteinander, stellen sich nach Abb. 5.44 auf Seite 78 für die Berechnung des Erwartungswerts stückweise lineare Basisfunktionen mit adaptiver Verfeinerung neben QMC als am besten geeignet heraus. Der Abstand zwischen den jeweils bestgeeigneten Approximationsverfahren, also ASGC nach dem Erwartungswertkriterium mit stückweise linea-

ren Basisfunktionen und ASGC nach dem Erwartungswertkriterium mit stückweise polynomiellen Basisfunktionen, beträgt im vorliegenden Fall Faktor 10.

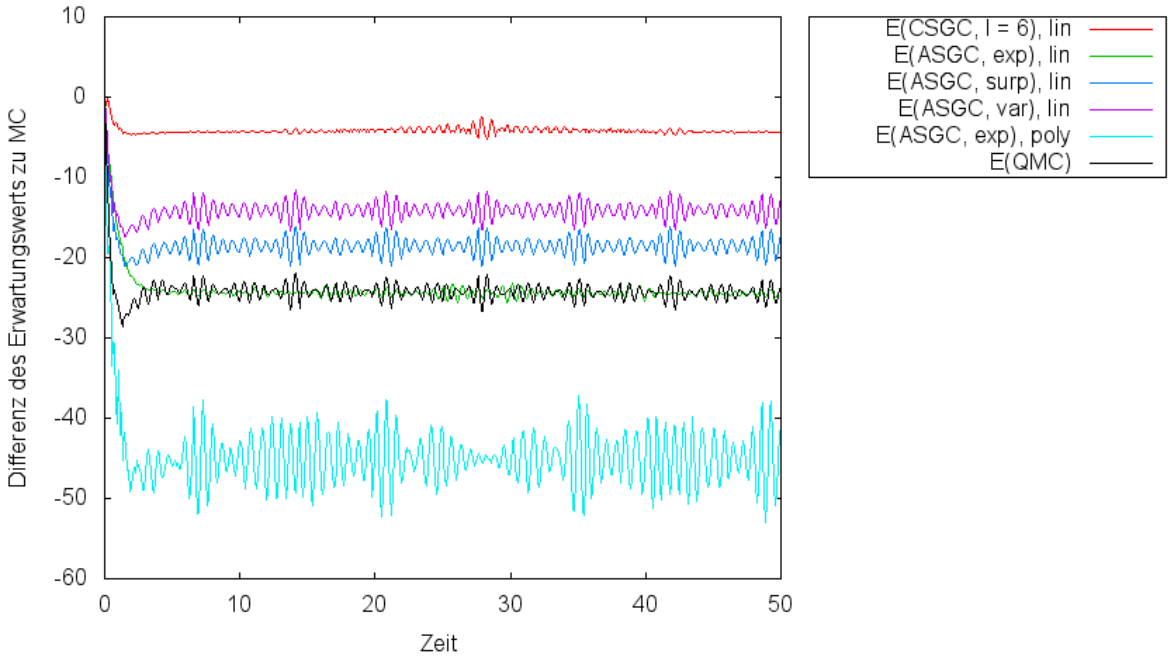

Abbildung 5.44.: Vergleich der Erwartungswertdifferenz der geeignetsten Verfahren

Für die Varianz setzt sich das Bild, dass stückweise lineare Basisfunktionen im vorliegenden Kontext bessere Ergebnisse liefern als stückweise polynomielle Basisfunktionen, fort. In diesem Fall bietet die Adaptivität allerdings keine Vorteile, vielmehr stellt die CSGC - Approximation auf Level $l = 6$ die geeignetste Approximation dar, wie Abb. 5.45 auf Seite 79 zeigt.

Das Zustandekommen der negativen Varianz bei CSGC auf Level $l = 5$ und dem Ansatz stückweise polynomieller Basisfunktionen bleibt rätselhaft. Durch Änderung des Quadraturverfahren von analytischer Quadratur zu einer MC Quadratur ist die Varianz korrekterweise positiv, wenngleich die Ergebnisse hier massiv oszillieren und eigentlich noch deutlich mehr Samples zu ziehen wären, um diese zu stabilisieren. Die Unklarheit darüber, woher die negative Varianz kommt, liegt v. a. daran, dass in keinem anderen Verfahren negative Varianzen aufgetreten sind. Weder auf anderen Level für stückweise polynomielle Basisfunktionen bei CSGC noch für stückweise lineare Basisfunktionen bei CSGC noch bei ASGC. Um dies weiter zu untersuchen, können nun zwei Schritte unternommen werden. Erstens die Analyse einer Visualisierung der Dünngitterinterpolation, bspw. mithilfe der im Rahmen von [Sch13] entstandenen Visualisierungssoftware, ob die Zielfunktion tatsächlich die Form einer Gauss-Kurve hat. Zweitens die Betrachtung der Entwicklung des Betrags der Überschüsse. Für glatte Funktionen wird nach Gleichung (3.8) erwartet, dass der Betrag der Überschüsse exponentiell abfällt. Ob dies tatsächlich passiert oder ob die Zielfunktion oszilliert oder Sprünge aufweist, so dass die Überschüsse wieder anwachsen, bleibt zu analysieren.

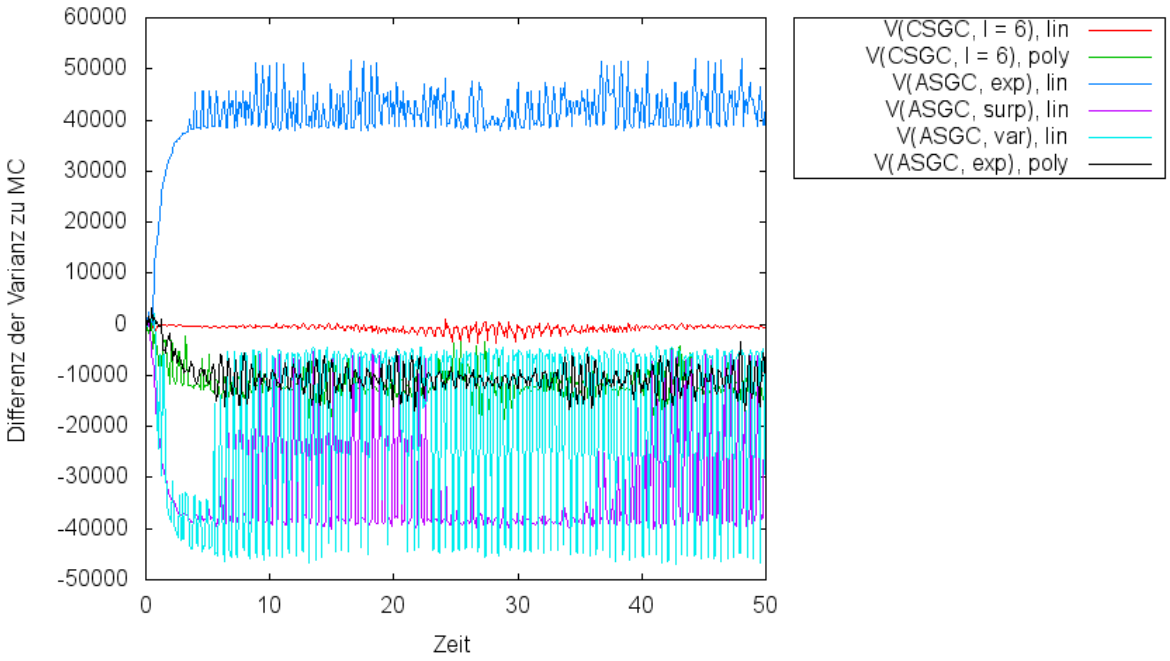

Abbildung 5.45.: Vergleich der Varianzdifferenz der geeignetsten Verfahren

6. Zusammenfassung und Ausblick

6.1. Fazit

Im Rahmen dieser Arbeit wurde eine Klasse für stückweise polynomielle Basisfunktionen im Rahmen des Dünngitter-Frameworks SG++ implementiert, deren Funktionalität anhand der im vorigen Kapitel vorgestellten Beispiele getestet wurde.

Dabei lag der Fokus auf der Feststellung einer höheren Konvergenzordnung gegenüber stückweise linearen Basisfunktionen durch den Einsatz stückweise polynomieller Basisfunktionen, der Wirksamkeit von räumlicher Adaptivität zur Verringerung der Zahl der nötigen Auswertungen für eine gute Approximation sowie rekursiver Hierarchisierung.

Die rekursive Hierarchisierung hat sich als nicht zielführend erwiesen. Anstelle des erhofften Zeitgewinns hat sich, wie das Hierarchisierungsexperiment in Abschnitt 5.2 gezeigt hat, eine massive Verschlechterung der Performance eingestellt, und zwar sowohl bei niedrigen als auch bei hohen Polynomgraden. Zusätzlich gegen die rekursive Hierarchisierung spricht, dass sich die zur Hierarchisierung benötigte Zeit, im Gegensatz zur direkten Hierarchisierung, um deutlich mehr als Faktor 2 erhöht, wenn ein zusätzliches Level hinzugefügt wird und damit etwa doppelt so viele Knoten zu hierarchisieren sind.

Die erhoffte höhere Konvergenzordnung für stückweise polynomielle Basisfunktionen bei hinreichend glatten Funktionen hat sich dagegen, den Erwartungen entsprechend, eingestellt, wie das Beispiel des Kosinus Hyperbolicus in Abschnitt 5.1.2 gezeigt hat. Erwartungsgemäß nicht gezeigt werden konnte dagegen, dass sich der globale Einsatz stückweise polynomieller Basisfunktionen lohnt, wenn die Zielfunktion lokale Unstetigkeiten aufweist, wie in Abschnitt 5.1.1 am Beispiel einer Betragsfunktion.

An diesem Beispiel konnte dagegen der Nutzen von räumlicher Adaptivität gezeigt werden. Beim Kosinus Hyperbolicus war das nicht der Fall. Das liegt daran, dass durch die Symmetrie und Glattheit der Kosinus Hyperbolicus Funktion ungefähr ein reguläres dünnes Gitter entsteht. D. h. beim Kosinus Hyperbolicus ergibt sich deshalb verglichen mit globaler Verfeinerung kein Gewinn durch den Einsatz von lokaler Verfeinerung, weil sich in diesem Beispiel lokale und globale Verfeinerung kaum unterscheiden.

Mit diesen Erfahrungswerten wurde die Tauglichkeit von adaptiven dünnen Gittern mit stückweise polynomiellen Basisfunktionen im Zusammenhang mit UQ am Beispiel einer Kármán'schen Wirbelstraße in Abschnitt 5.3 untersucht. Im untersuchten Wirbelstraßen-Beispiel konnte der Erwartungswert mit ca. einem Fünftel der Funktionsauswertungen, verglichen mit MC und $n = 5000$ Samples, auf unter

1% Abweichung angenähert werden. Dazu wurde nach dem in Abschnitt 3.3 vorgestellten Erwartungswertkriterium verfeinert und stückweise lineare Basisfunktionen angesetzt. Für die Varianz ergab sich dagegen ein schlechteres Bild. Hier stellte CSGC mit Level $l = 6$ und stückweise linearen Basisfunktionen die beste Approximation dar, für alle übrigen Verfahren ergab sich eine Abweichung von mindestens 5%, oftmals auch deutlich darüber.

Für stückweise polynomielle Basisfunktionen konnten diese Ergebnisse qualitativ reproduziert werden, die Genauigkeit der stückweise linearen Basisfunktionen aber nicht erreicht werden. Das spricht dafür, dass das Wirbelstraßen-Beispiel bezüglich der Glattheit eher dem Beispiel der Betragsfunktion ähnelt als dem des Kosinus Hyperbolicus.

Zusätzlich ergab sich für CSGC auf Level $l = 5$ bei der Berechnung der Varianz mit analytischer Quadratur der Dünngitterapproximation eine negative Varianz. Dieses Problem konnte durch den Wechsel des Quadraturverfahrens von analytischer Quadratur zu MC Quadratur behoben werden.

6.2. Ausblick

Es bietet sich an, für SG++ eine neue Klasse für stückweise polynomielle Basisfunktionen einzuführen, die die Vorzüge der bereits vorhandenen Klasse “PolyBasis” mit denen der im Rahmen der Arbeit implementierten “UltraPoly”-Klassen verbindet. Dabei sollte die direkte Hierarchisierung aus der bestehenden Klasse mit der Quadraturmethode für beliebige Polynomgrade und der Behandlung von Randpunkten aus den implementierten Klassen kombiniert werden.

Darüber hinaus ist zu überlegen, ob die stückweise polynomiellen Basisfunktionen ans Gitter gebunden werden können, da die Basis wie in Abb. 4.1 dargestellt, für jede Operation neu erstellt wird. Dieser redundante Aufwand könnte durch eine Bindung ans Gitter ausbleiben.

Testen ließe sich auch die Einführung einer Klasse mit “variablem Polynomgrad”. Wegen der nichtüberlappenden Basisfunktionen ist es auch auf demselben Level problemlos möglich, Basisfunktionen von verschiedenem Polynomgrad zu haben.
Dabei ist jedoch zu beachten, dass es wegen der Tensorprodukt-Struktur des Dünngitter-Ansatzes keine räumlich-adaptive Anpassung des Polynomgrads analog der Verfeinerung der Gitterknoten geben kann, wo die Effekte der Verfeinerung hauptsächlich lokal sind, wenn man von der rekursiven Erzeugung hierarchischer Vorgängerknoten absieht. Stattdessen hat eine adaptive Veränderung des Polynomgrads für Dimensionen $d > 1$ immer globale Auswirkungen.
D. h., es ist bspw. nicht allgemein möglich, global ein dünnes Gitter mit stückweise polynomiellen Basisfunktionen zur Approximation zu verwenden, um von deren höherer Konvergenzordnung zu profitieren, und gleichzeitig bei der Erkennung von lokalen Unstetigkeiten nur an diesen Stellen lokal stückweise lineare Basisfunktionen anzusetzen. Stattdessen müssen bei diesem Vorgehen Einbußen in den glatten Bereichen hingenommen werden.
Das liegt daran, dass die, um in dem geschilderten Beispiel zu bleiben, an einem Knoten angesetzte stückweise lineare Basis durch den Tensorprodukt-Ansatz nicht nur an einem konkreten Knoten angesetzt wird. Stattdessen ist die “adaptiv erzeugte” lineare Basisfunktion über alle Dimensionen für alle Knoten mit entsprechendem (Level, Index)-Tupel für Hierarchisierung, Evaluation und Quadratur anzusetzen.

Zusätzlich ausprobieren lässt sich, ob sich ein manuelles Verfälschen der Überschüsse auszahlt, indem dem Betrag nach sehr kleine Überschüsse auf den Wert 0 gesetzt werden. Dies würde dazu führen, dass man für einen solchen Fall den Schritt zur Polynomauswertung oder Quadraturberechnung in der jeweiligen Funktion einsparen kann.

Im UQ-Kontext sind v. a. weitere Daten und Erfahrungen zu sammeln. Auf dieser Basis sollten dann die Verfeinerungskriterien verbessert werden. Insbesondere ist zu prüfen, ob bei der adaptiven Verfeinerung die Verteilung angemessen berücksichtigt wird. Bislang ist nur beim Erwartungswertkriterium der Fall, dass die Verteilung überhaupt berücksichtigt wird. Ob die bestehende Methodik ausreicht, um vor allem im Maximum der Normalverteilung neue Gitterpunkte zu erzeugen statt an den Rändern, ist nach Meinung des Autors noch nicht eindeutig klar. Zusätzlich liegt auch bei der Approximation der Varianz noch Potential, insbesondere in Form der Berücksichtigung der Verteilung und einer besseren lokalen Abschätzung der Varianz.

A. Anhang

A.1. Maximaler Fehler pro Verfeinerungsschritt für cosh

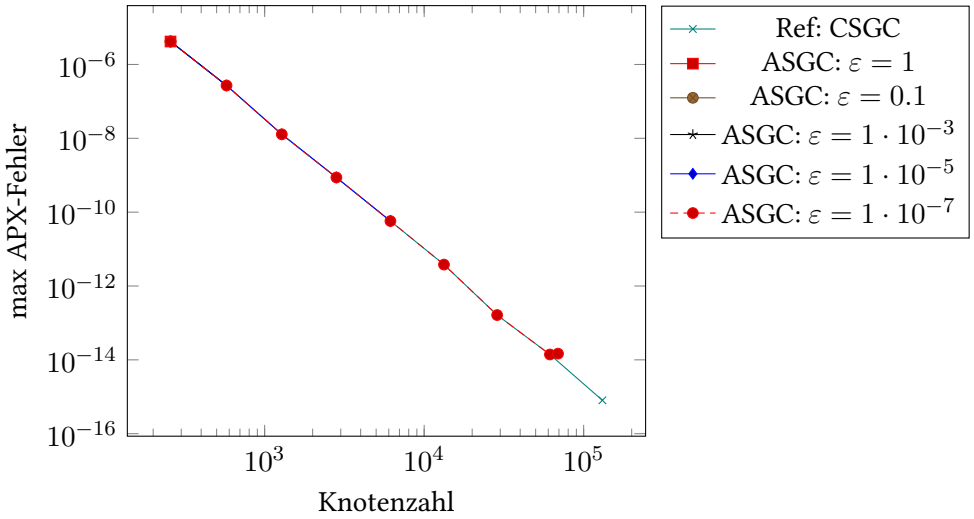

Abbildung A.1.: cosh: Entwicklung des maximalen Interpolationsfehlers über die Knotenzahl für stückweise lineare Basisfunktionen

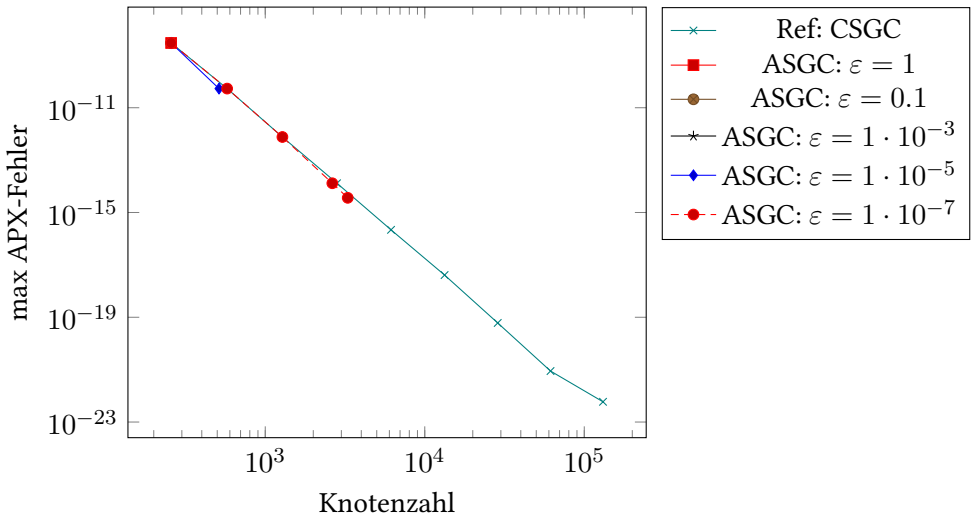

Abbildung A.2.: cosh: Entwicklung des maximalen Interpolationsfehlers über die Knotenzahl für stückweise polynomielle Basisfunktionen mit Grad 2

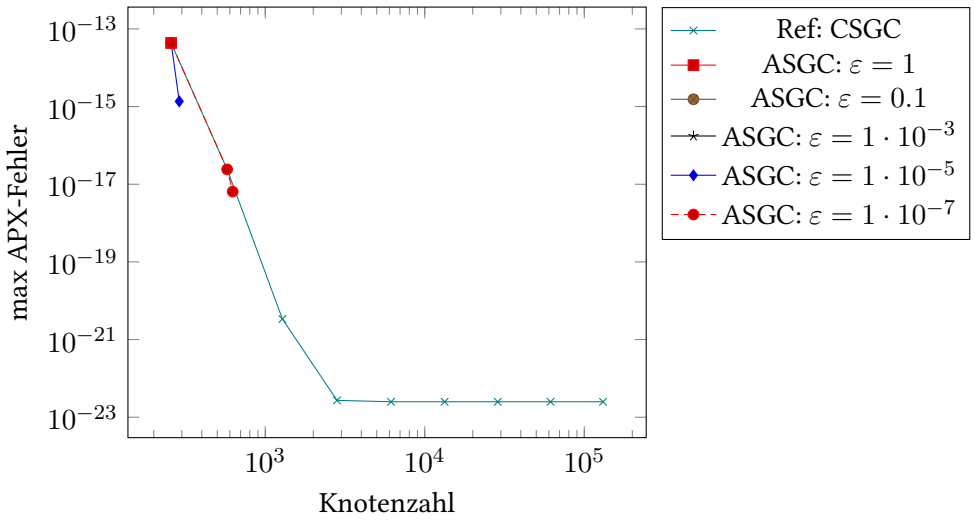

Abbildung A.3.: cosh: Entwicklung des maximalen Interpolationsfehlers über die Knotenzahl für stückweise polynomielle Basisfunktionen mit Grad 6

A.2. Hierarchisierung - benötigte Zeit über Knotenzahl

(A.1) $$u_1(x, y) = \frac{1}{|0.3 - x^2 - y^2| + 0.1}$$

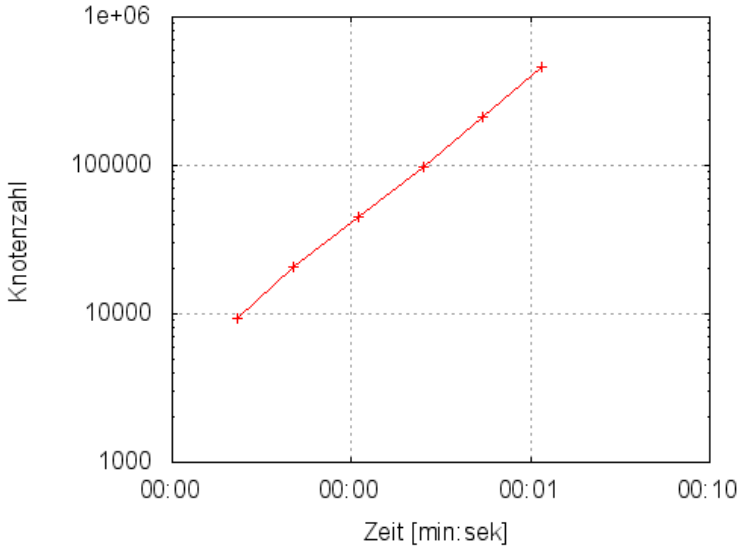

Abbildung A.4.: Plot der benötigten Zeit zur direkten Hierarchisierung von u_1 mit stückweise polynomiellen Basisfunktionen von Polynomgrad $p = 2$ über die Knotenzahl.

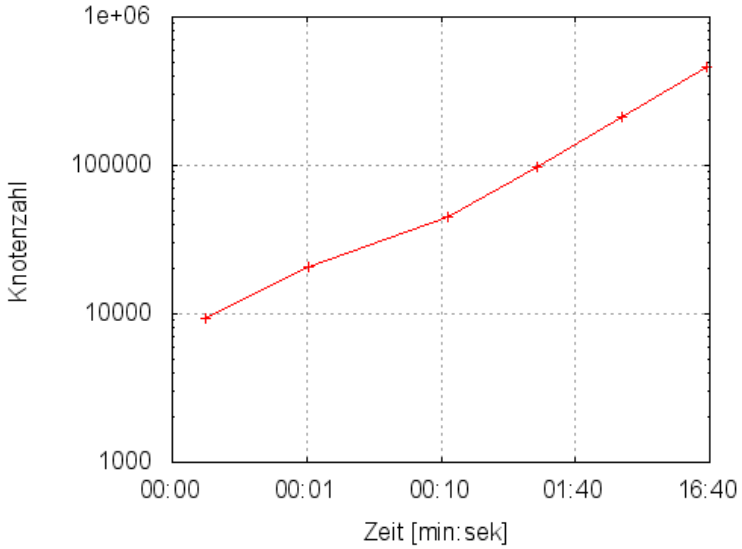

Abbildung A.5.: Plot der benötigten Zeit zur rekursiven Hierarchisierung von u_1 mit stückweise polynomiellen Basisfunktionen von Polynomgrad $p = 2$ über die Knotenzahl.

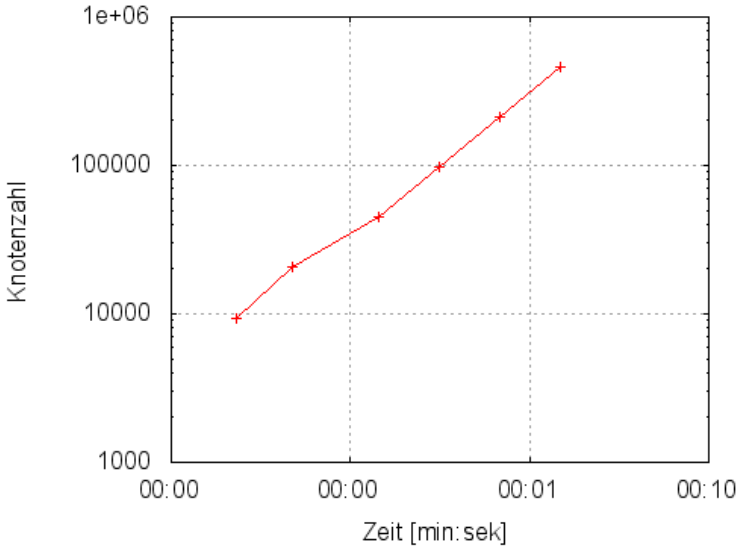

Abbildung A.6.: Plot der benötigten Zeit zur direkten Hierarchisierung von u_1 mit stückweise polynomiellen Basisfunktionen von Polynomgrad $p = 10$ über die Knotenzahl.

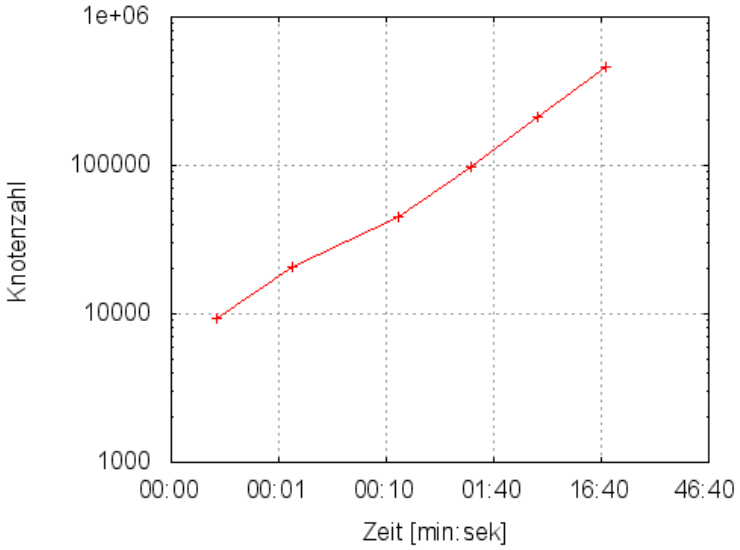

Abbildung A.7.: Plot der benötigten Zeit zur rekursiven Hierarchisierung von u_1 mit stückweise polynomiellen Basisfunktionen von Polynomgrad $p = 10$ über die Knotenzahl.

A.3. Varianzberechnung der Wirbelstraße nach MC Quadratur

A.3.1. CSGC, stückweise polynomielle Basisfunktionen

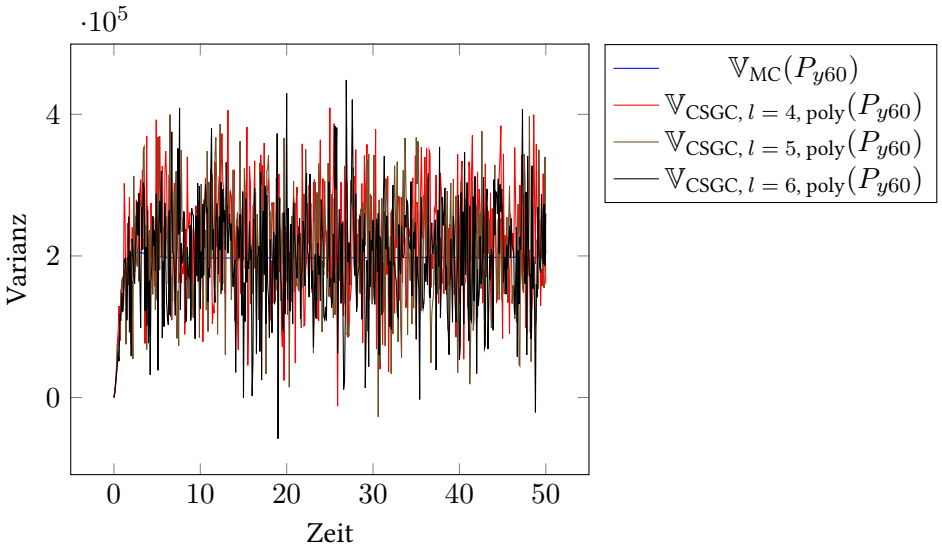

Abbildung A.8.: Wirbelstraße: Varianz: CSGC Level $l = [4, 5, 6]$, keine Randpunkte, stückweise polynomielle Basisfunktionen, MC Quadratur

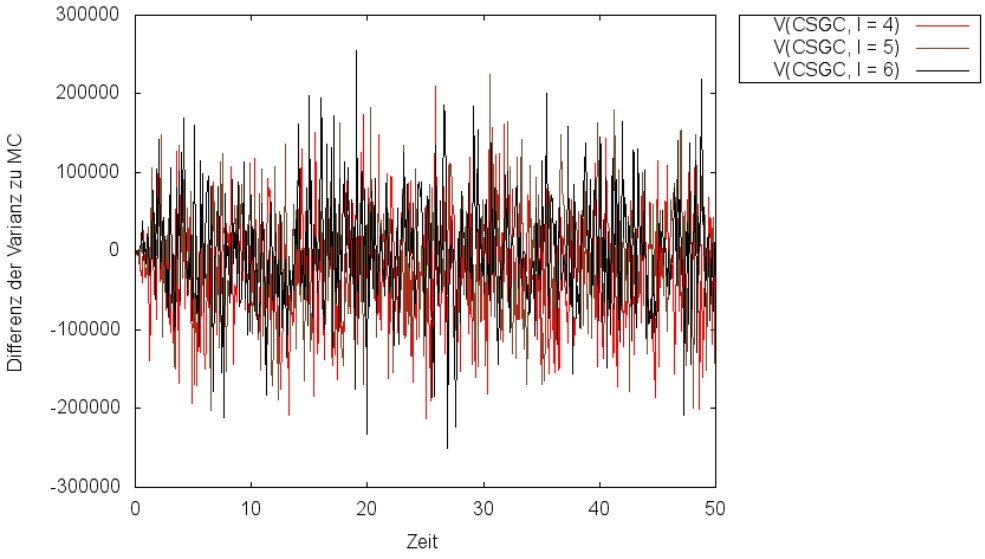

Abbildung A.9.: Wirbelstraße: Varianzdifferenz: CSGC Level $l = [4, 5, 6]$, keine Randpunkte, stückweise polynomielle Basisfunktionen, MC Quadratur zu MC

A.3.2. ASGC, stückweise lineare Basisfunktionen

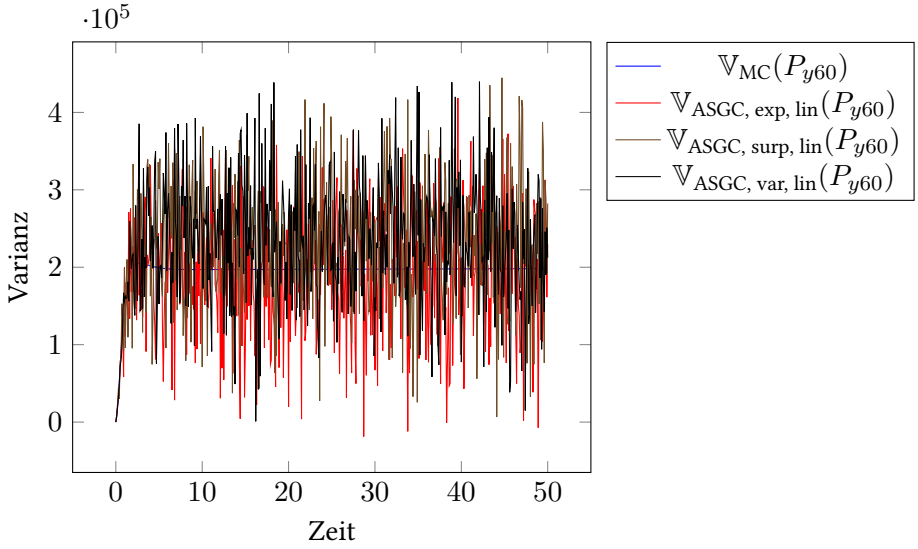

Abbildung A.10.: Wirbelstraße: Varianz: ASGC, keine Randpunkte, stückweise lineare Basisfunktionen, MC Quadratur

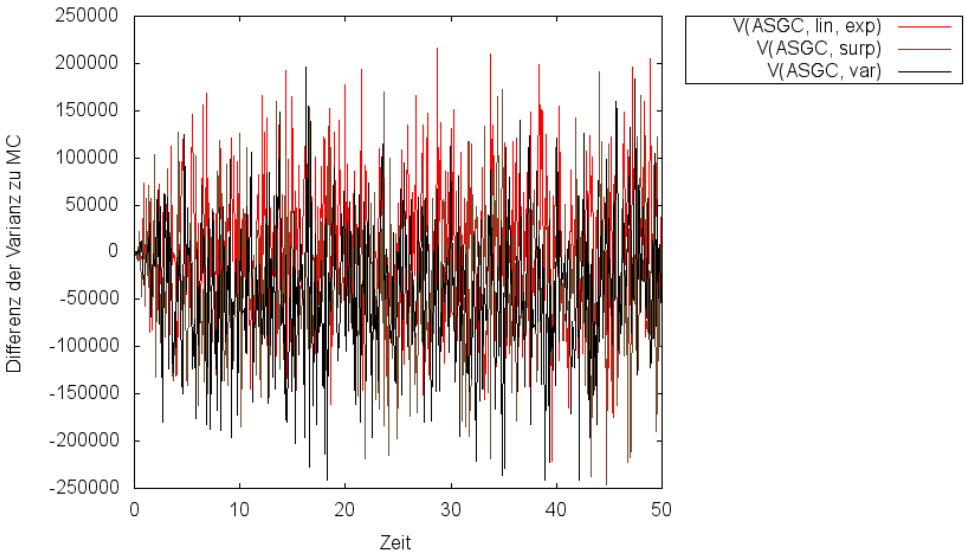

Abbildung A.11.: Wirbelstraße: Varianzdifferenz: ASGC, keine Randpunkte, lineare Basisfunktionen, MC Quadratur zu MC

A.3.3. ASGC, stückweise polynomielle Basisfunktionen

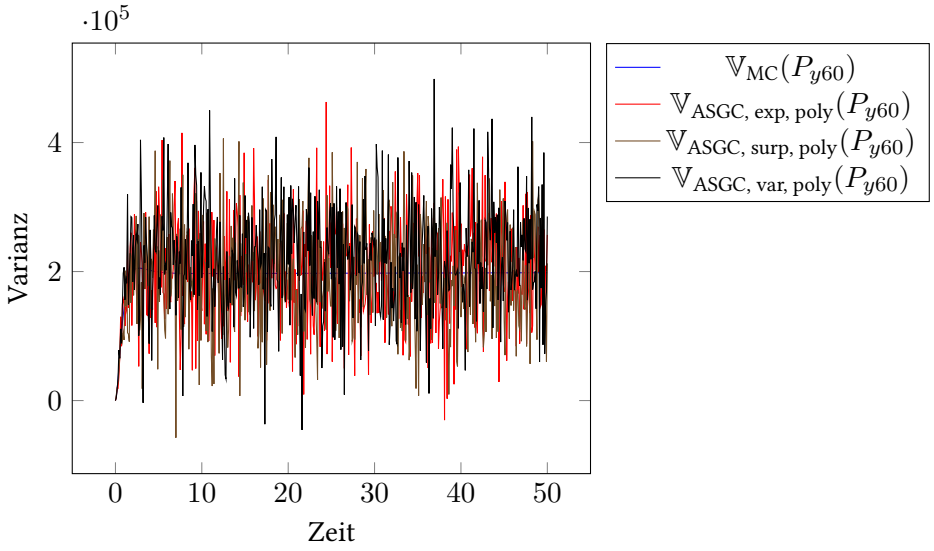

Abbildung A.12.: Wirbelstraße: Varianz: ASGC, keine Randpunkte, stückweise polynomielle Basisfunktionen, MC Quadratur

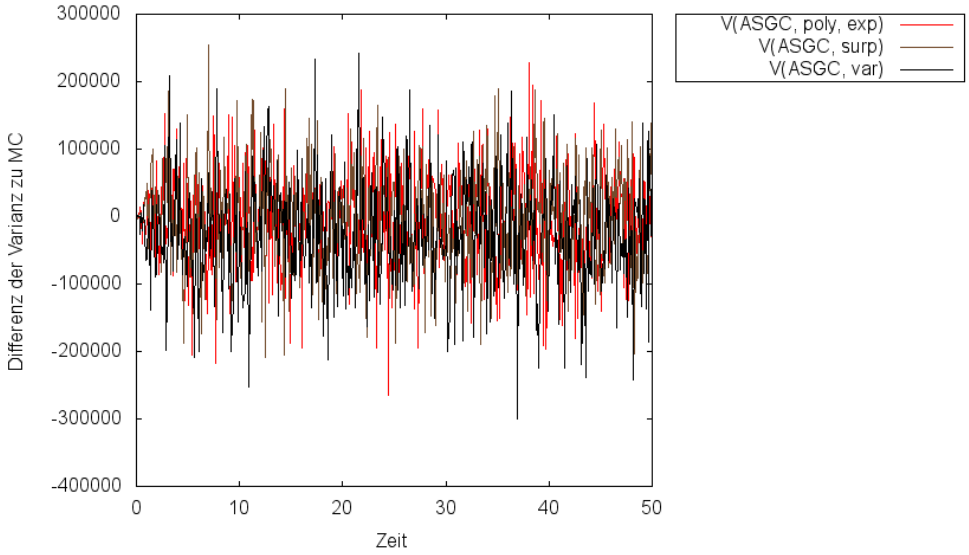

Abbildung A.13.: Wirbelstraße: Varianzdifferenz: ASGC, polynomielle Basisfunktionen, MC Quadratur zu MC

Literaturverzeichnis

[BG04] H.-J. Bungartz, M. Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004. doi:10.1017/S0962492904000182. URL http://journals.cambridge.org/article_S0962492904000182. (Zitiert auf Seite 24)

[Bun98] H.-J. Bungartz. *Finite Elements of Higher Order on Sparse Grids.* Habilitation, Technische Universität München, 1998. (Zitiert auf den Seiten 26, 27, 28, 29, 30, 35 und 41)

[Caf98] R. E. Caflisch. Monte Carlo and quasi-Monte Carlo methods. *Acta Numerica*, 7:1–49, 1998. doi:10.1017/S0962492900002804. URL http://journals.cambridge.org/article_S0962492900002804. (Zitiert auf Seite 18)

[EB09] M. S. Eldred, J. Burkhardt. Comparison of Non-Intrusive Polynomial Chaos and Stochastic Collocation Methods for Uncertainty Quantification. *Proceedings of 47th AIAA Aerospace Sciences Meeting and Exhibit, Paper No. AIAA-2009-0976*, 2009. (Zitiert auf Seite 19)

[FDPS14] F. Franzelin, P. Diehl, D. Pflüger, M. A. Schweitzer. Non-intrusive uncertainty quantification with sparse grids for multivariate peridynamic simulations. *Meshfree Methods for Partial Diffferential Equations, submitted*, 2014. (Zitiert auf den Seiten 17, 18, 33 und 34)

[Feu10] C. Feuersänger. *Sparse Grid Methods for Higher Dimensional Approximation.* Dissertation, Institut für Numerische Simulation, Universität Bonn, 2010. (Zitiert auf den Seiten 21 und 26)

[Iac11] G. Iaccarino. Uncertainty Quantification in Computational Science. Lecture, 2011. (Zitiert auf den Seiten 17 und 18)

[KD09] A. D. Kiureghian, O. Ditlevsen. Aleatory or epistemic? Does it matter? *Structural Safety*, 31:105–112, 2009. (Zitiert auf Seite 14)

[Lei13] J. Leibinger. *Quantifizierung von Unsicherheiten mit dünnen Gittern.* Diplomarbeit, Universität Stuttgart, 2013. (Zitiert auf den Seiten 34, 45, 49, 60 und 61)

[MK10] O. L. Maître, O. Knio. *Spectral Methods for Uncertainty Quantification.* Springer, 2010. (Zitiert auf den Seiten 7, 14, 15, 16 und 17)

[MZ09] X. Ma, N. Zabaras. An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations. *Journal of Computational Physics*, 228(8):3084–3113, 2009. doi:10.1016/j.jcp.2009.01.006. URL http://linkinghub.elsevier.com/retrieve/pii/S002199910900028X. (Zitiert auf den Seiten 6, 7, 19, 34, 49, 50, 51, 53 und 55)

[Pfl10] D. Pflüger. *Spatially Adaptive Sparse Grids for High-Dimensional Problems*. Dissertation, Technische Universität München, 2010. (Zitiert auf den Seiten 7, 21, 22, 23, 28, 33, 35 und 62)

[Pfl14] D. Pflüger. Ausgewählte Kapitel des Wissenschaftlichen Rechnens. Lecture, 2014. (Zitiert auf den Seiten 7 und 31)

[Rö11] K. Röhner. *Polynomial Basis Functions on Adaptive Sparse Grids*. Bachelorarbeit, Technische Universität München, 2011. (Zitiert auf den Seiten 32 und 35)

[Sch13] R. Schwarz. Interaktive Visualisierung von hochdimensionalen Funktionen. Bachelorarbeit: Universität Stuttgart, Institut für Parallele und Verteilte Systeme, Parallele Systeme, 2013. URL http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=BCLR-0081&engl=0. (Zitiert auf Seite 78)

[Sob67] I. M. Sobol. The distribution of points in a cube and the approximate evaluation of integrals. *USSR Comput. Math. Math. Phys.*, 7:86 – 112, 1967. (Zitiert auf Seite 18)

[WHR+05] W. Walker, P. Harremoes, J. Rotmans, J. van der Sluijs, M. van Asselt, P. Janssen, M. K. von Krauss. Defining Uncertainty: A Conceptual Basis for Uncertainty Management in Model-Based Decision Support. *Integrated Assessment*, 4(1), 2005. URL http://journals.sfu.ca/int_assess/index.php/iaj/article/view/122/79. (Zitiert auf den Seiten 13 und 14)

[XK02] D. Xiu, G. E. Karniadakis. The Wiener–Askey Polynomial Chaos for Stochastic Differential Equations. *SIAM J. Sci. Comput.*, 24(2):619–644, 2002. doi:10.1137/S1064827501387826. URL http://dx.doi.org/10.1137/S1064827501387826. (Zitiert auf Seite 19)

[XK03] D. Xiu, G. E. Karniadakis. Modeling uncertainty in flow simulations via generalized polynomial chaos. *Journal of Computational Physics*, 187(1):137 – 167, 2003. doi:http://dx.doi.org/10.1016/S0021-9991(03)00092-5. URL http://www.sciencedirect.com/science/article/pii/S0021999103000925. (Zitiert auf Seite 19)

Alle URLs wurden zuletzt am 19. Mai 2014 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift