Numerische Integration, angewandt bei der Inferenz in Zustandsraummodellen

Weiterentwicklung von Quadratur-Algorithmen und Vergleich mit populären simulationsbasierten Methoden

---

Dissertation
zur Erlangung des Grades eines Doktors der
wirtschaftlichen Staatswissenschaften
(Dr. rer. pol)
des Fachbereichs Rechts- und Wirtschaftswissenschaften
der Johannes Gutenberg-Universität Mainz
vorgelegt von

Herrn Dipl. Volkswirt Constantin Weiser

in Mainz

---

im Jahre 2013

Diese Arbeit wurde als Dissertation am Fachbereich Rechts- und Wirtschaftswissenschaften der Johannes Gutenberg-Universität Mainz eingereicht (D77).

Erstgutachter: .....
Zweitgutachter: .....
Tag der mündlichen Prüfung: 09. Dezember 2013

Inhaltsverzeichnis

Abbildungsverzeichnis **3**

Abkürzungsverzeichnis **5**

1. Einleitung **7**

2. Zustandsraummodelle **9**
2.1. Einführendes Beispiel 9
2.2. Allgemeine Zustandsraummodelle 10
2.3. Inferenz 12
2.3.1. Bayessche Statistik 12
2.3.2. Sequentielles bayessches Filtern 14
2.3.3. Parameterschätzung 17

3. Forschungsstand **19**
3.1. Sequentielle Monte-Carlo-Methoden (sMC) 19
3.1.1. Einführung in die sMC-Methoden 19
3.1.2. Partikel-Filter: Algorithmen 20
3.1.3. Vor- und Nachteile der Partikel-Filter 23
3.2. Sequentielle numerische Integration (sNI) 24
3.2.1. Historische Entwicklung 24
3.2.2. Algorithmus nach Kitagawa (1987) 24

4. Erweiterungen der sNI **27**
4.1. Schwachstellen des ursprünglichen Algorithmus 27
4.1.1. Rechenaufwand 27
4.1.2. Effizienz 27
4.1.3. A-Priori-Kenntnisse 28
4.2. Erweiterungen - Überblick 28
4.3. Kombinationstechnik (dünne Gitter) 29
4.4. Reskalierung mehrdimensionaler Gitter 33
4.5. Dynamisches Gitter 38
4.5.1. Hilfsfilter 40
4.5.2. Iteratives Verfahren 41

4.5.3. weitere Varianten 43

5. Vergleich von sMC und sNI **47**
5.1. Untersuchungsdesign 47
5.2. Konvergenzverhalten 48
5.2.1. lineare/normalverteilte Zustandsraummodelle 49
5.2.2. Nichlineares Zustandsraummodell 50
5.2.3. Nichtlineare und nicht-normale Zustandsraumodelle 52

6. Implementierung **57**
6.1. Sofware-Paket `nife` 57
6.2. Anwendungbeispiel: stochastisches Volatilitäts-Modell 58

7. Schlussbemerkungen **63**
7.1. Zusammenfassung 63
7.2. Offene Forschungsfragen 64
7.3. Fazit 65

Anhang **67**

A. Kalman-Filter **69**

B. Einführung in die numerische Integration **71**

C. Dokumentation - `nife` **77**

Literatur **87**

Abbildungsverzeichnis

2.1. Algorithmus des sequentielles bayesschen Filterns 15

3.1. Algorithmus des Bootstrap Filters 21
3.2. Illustration des Boostrap- und Auxiliary Particle Filter 22
3.3. Algorithmus des „Auxiliary Particle Filter“ 23
3.4. Algorithmus der sNI nach Kitagawa (1987) 26
3.5. Illustration der statischen Gitter bei Kitagawa (1987) 26

4.1. Level-Matrix für reguläres 2D-dünnes Gitter 30
4.2. Dünnes Gitter und die enthaltenen vollen Gittern (Gauß-Hermite) 31
4.3. Dünnes Gitter und die enthaltenen vollen Gittern (Trapezregel) 32
4.4. Effekt dünner Gitter 34
4.5. Schlecht angepasste Gitter 34
4.6. Methoden zur Reskalierung (Illustration) 36
4.7. Vergleich nicht-reskalierter und reskalierter Gitter 38
4.8. Algorithmus für sNI mit dynamischen Gitter 39
4.9. Illustration dynamisches Gitter 40
4.10. Algorithmus für sNI mit dynamischem Gitter und Hilfsfilter 41
4.11. Algorithmus für sNI mit iterativen dynamischen Gitter. 43
4.12. Adaptives (links) und reguläres Gitter (rechts). 44

5.1. Konvergenzverhalten 1-2D (Anzahl Stützstellen vs. RMSE) 50
5.2. Konvergenzverhalten 3-4D (Anzahl Stützstellen vs. RMSE) 51
5.3. Konvergenzverhalten 1-2D (DAUER vs. RMSE) 52
5.4. Konvergenzverhalten 3-4D (DAUER vs. RMSE) 53
5.5. Unterschied in der Komplexität der Algorithmen 54
5.6. Konvergenzverhalten im SVM (1D, normalverteiltes Rauschen) 55
5.7. Konvergenzverhalten im SVM (1D, t-verteiltes Rauschen) 56

6.1. Beispiel eines `StSp.Obj` für die Nutzung im `NIFilter` 59
6.2. Beispiel eines `StSp.Obj` für die Nutzung im `BootstrapFilter` 60
6.3. Beispiel eines `StSp.Obj` für die Nutzung im `AParticleFilter` 61
6.4. Kurs und Tagesrendite der Commerzbank-Aktie 61
6.5. StSp.Obj für das stochastische Volatilitäts Modell 62

6.6. Gefilterte Volatilität der Commerzbank-Aktie 62

A.1. Algorithmus des Kalman-Filters 69

B.1. Trapezregel (n = 4) 72
B.2. Stützstellen der Trapezregel für Level 1 bis 5 73
B.3. Stützstellen der ursprünglichen Gauss-Hermite-Quadratur 75
B.4. Stützstellen der genesteten Variante der Gauß-Hermite-Quadratur 75
B.5. Volles 2-dimensionales Gitter 76

Abkürzungsverzeichnis

APF Auxilirary Particle Filter

BF Bootstrap-Filter

GHe Gauß-Hermite-Quadratur

sMC sequentielle Monte-Carlo-Methode

sNI sequentielle numerische Integration

1. Einleitung

Viele für die Wirtschaftswissenschaften interessante Phänomene sind zeitlichen Entwicklungen unterworfen. So verändern sich im Zeitablauf z.B. Kurse von Wertpapieren, die Wirtschaftsleistung von Nationen oder die Arbeitsleistung des Arbeiters.

Die Methoden der Zeitreihenanalyse versuchen nun gerade diese Dynamiken zu beschreiben und daraus Rückschlüsse zu ziehen auf die dahinterliegenden Regelmäßigkeiten. Diese Erkenntnisse helfen dann wiederum Erwartungen bzgl. der Zukunft zu bilden, die wir in der Regel benötigen, um rationale Entscheidungen zu treffen.

Eine besondere Stellung nehmen die Zustandsraummodelle unter den Methoden der Zeitreihenanalyse ein, da diese durch ihre allgemeine Formulierung viele bekannte Zeitreihenmodelle, wie die Gruppe der VARIMAX- oder ARCH-Modelle, als Sonderfälle implizieren und darüber hinaus eine sehr vielfältige Formulierung dynamischer Systeme zulässt. Dazu zählen auch zeitvariante Regressionen, Modelle zu Beschreibungen von Strukturbrüchen, nicht-lineare Feedback-Systeme und viele mehr (vgl. z.B. Hamilton (1994); Pole, West und Harrison (1994); Basdevant (2003)).

Der Flexibilität des Modellrahmens steht entgegen, dass sich mathematische Probleme bei der Berechnung der Modelle ergeben können. Denn nur für eine kleine Untergruppe der Zustandsraummodelle existieren analytische Lösungen. Alle anderen Modelle können nur mit Näherungsverfahren gelöst werden. Wobei in der gegenwärtigen Literatur simulationsbasierte Lösungsansätze, sogenannte Partikel-Filter, die Diskussionen dominieren.

Vor dem Siegeszug der Partikel-Filter wurde, insbesondere für einfache, niedrigdimensionale Zustandsraummodelle, die numerische Integration als Verfahren zur Bestimmung von Näherungslösungen verwendet. Allerdings wurde seit den frühen 1990er-Jahren kaum mehr an diesem Lösungsansatz geforscht.

Unabhängig davon gab es deutliche Weiterentwicklungen in der Numerik, der Disziplin, die sich unter anderem mit an der Methode der numerische Integration selbst forscht.

Diese Situation ist der Ausgangspunkt der Arbeit. Sie greift zum einen den Algorithmus von Kitagawa (1987) auf und passt diesen entsprechend der Weiterentwicklungen im Bereich der numerischen Integration an. Die angepassten Algorithmen werden dann mit Blick auf deren Leistungsfähigkeit mit den Partikelfiltern verglichen.

Die Arbeit gliedert sich wie folgt: In Kapitel 2 werden, von einem einfachen Beispiel ausgehend, die Klasse der Zustandsraummodelle und der formale Lösungsansatz beschrieben.

Kapitel 3 fasst schematisch die in der Literatur diskutierten Näherungsverfahren zusammen.

Im folgenden Kapitel werden die Weiterentwicklungen der Algorithmen eingeführt.

In Kapitel 5 wird die sequentielle Monte-Carlo-Methode mit der sequentielle Numerischen Integration bzgl. deren Effizienz in Form einer Simulationsstudie verglichen.

In Kapitel 6 wird die Implementierung beschrieben und an einem Beispiel illustriert.

Im letzten Kapitel wird die Arbeit zusammengefasst und offene Forschungsfragen erörtert.

2. Zustandsraummodelle

In Abschnitt 2.1 wird, ausgehend von einem einführenden Beispiel, die grundlegende Struktur von Zustandsraummodellen beschrieben. Darauf aufbauend wird die Gruppe der linearen und normal-verteilten Zustandsraummodelle vorgestellt und zu einer weitestgehend restriktionslosen Formulierung erweitert (Abschnitt 2.2).

In Abschnitt 2.3 wird die formale Lösung für die Inferenz in Zustandsraummodellen beschrieben und das damit verbundene mathematische Problem aufgezeigt.

Diese Einführung zielt nicht darauf ab, die praktische Anwendung von Zustandsraummodellen zu beschreiben, was die Behandlung von Fragen zur Spezifikation, Modellprognose, Tests und vieles andere bedeuten würde. Hierfür sei auf die entsprechende Literatur verwiesen, wie z.B. der Arbeiten von Hamilton (1994) oder Petris, Campagnoli und Petrone (2009).

2.1. Einführendes Beispiel

Dieses Beispiel dient als einfache Illustration eines Zustandsraummodells, an dem zugleich die Notation vorgestellt wird.

Für dieses Beispiel wird zwischen dem Wert eines Wertpapiers und dessen Preis differenziert. Wobei der Preis in der Form des Börsenkurses direkt beobachtbar und der wahre Wert nicht beobachtbar ist.

Die Beziehung zwischen dem Wert und dem Preis kann wie in Gleichung 2.1 beschrieben werden

$$y_t = 1 \cdot x_t + \varepsilon_t, \tag{2.1}$$

wobei y_t den Preis bezeichnet, der für die Zeitpunkte $t = 1, \ldots, T$ beobachtet wurde. Der Preis entspricht dem wahren Wert x_t zuzüglich einem Fehlerterm ε_t, der als Messfehler oder Rauschen bezeichnet wird. Die Gleichung 2.1 wird als Beobachtungsgleichung (engl.: observation-equation) bezeichnet. Sie definiert in jedem Zustandsraummodell den funktionalen Zusammenhang zwischen dem beobachtbaren Phänomen (y_t), dem unbeobachtbaren Zustand (x_t) und dem Rauschen (ε_t).

Ein Zustandsraummodell umfasst neben der Beobachtungsgleichung auch eine sogenannte Zustandsgleichung, die die Evolution des Zustands modelliert.

$$x_t = 1 \cdot x_{t-1} + \omega_t \tag{2.2}$$

In Gleichung 2.2 ist eine einfache Spezifikation einer Zustandsgleichung gegeben. Gemäß dieser wird der wahre Wert des Wertpapiers zum Zeitpunkt t durch den wahren Wert der Vorperiode (x_{t-1}) und einer Veränderung bzw. Innovation (ω_t) bestimmt. Eine solche Dynamik bezeichnet man als Random-Walk.

Wird für die unbeobachtbaren Terme ε_t und ω_t eine Verteilungsannahme getroffen, ist man in der Lage, Wahrscheinlichkeitsaussagen bzgl. der wahren Werte $x_{1:T}$ zu treffen, obwohl lediglich der ungenaue Preis beobachtet wird[1].

An dieser Stelle gehen wir davon aus, dass sowohl der Messfehler in Gleichung 2.1 als auch die Innovation in Gleichung 2.2 normalverteilt ist, mit den konstanten Parametern $\mu = 0$ und $\sigma^2 = \sigma_\varepsilon^2$ bzw. $\sigma^2 = \sigma_\omega^2$.

Mit den Gleichungen 2.1 und 2.2, den Verteilungsannahmen und den sogenannten Hyperparametern $\theta = \begin{pmatrix} \sigma_\varepsilon^2, & \sigma_\omega^2 \end{pmatrix}$ ist dieses Zustandsraummodell vollständig beschrieben.

2.2. Allgemeine Zustandsraummodelle

Das im Beispiel vorgestellte Modell wird wegen seiner Struktur als „Random Walk + Noise“ - Modell bezeichnet. Es ist eine spezielle Variante aus der Gruppe der linearen und normalverteilten Zustandsraummodellen.

Modelle dieser Art werden als linear bezeichnet, weil die Beobachtungs- und Zustandsgleichung intrinsisch lineare Funktionen sind. Sie werden ferner als normalverteilt bezeichnet, wegen der Verteilungsannahme bzgl. des das Rauschens und der Innovationen.

Eine sehr allgemeine Formulierung der linearen und normal-verteilten Zustandsraummodelle, die auch als dynamisch lineare Modelle bezeichnet werden, ist in Gleichung 2.3 und 2.4 gegeben[2].

$$y_t = F_t x_t + \varepsilon_t \qquad \text{mit } \varepsilon_t \sim N(0, E_t) \tag{2.3}$$

$$x_t = G_t x_{t-1} + \omega_t \qquad \text{mit } \omega_t \sim N(0, W_t) \tag{2.4}$$

Mit den Dimensionen:

[1] Notation: $x_{1:T}$ steht für die Serie von Zuständen für den Zeitraum $t = 1, ..., T$, oder allgemein $z_{a:b} = \{z_i | i = a, ..., b\}$.

[2] Notation: Vektoren und Skalare werden mit Kleinbuchstaben, Matrizen mit Großbuchstaben bezeichnet

- y_t bzw. $\varepsilon_t \rightarrow (m \times 1)$
- x_t bzw. $\omega_t \rightarrow (p \times 1)$
- $F_t \rightarrow (m \times p)$
- $E_t \rightarrow (m \times m)$
- G_t bzw. $W_t \rightarrow (p \times p)$

In dieser Form bezeichnen $N(0, E_t)$ und $N(0, W_t)$ multivariate Normalverteilungen mit Mittelwert Null und entsprechenden Kovarianzmatrizen. Es soll gelten, dass beide Verteilungen untereinander und über die Zeit unabhängig voneinander sind, d.h. folgende Relationen sollen gelten:

$$\begin{aligned} p(\varepsilon_t|\omega_t) = p(\varepsilon_t) \quad \text{bzw.} \quad p(\omega_t|\varepsilon_t) = p(\omega_t) \\ p(\varepsilon_t|\varepsilon_s) = p(\varepsilon_t) \quad \text{bzw.} \quad p(\omega_t|\omega_s) = p(\omega_t) \; \forall t \neq s \end{aligned}$$

Der Parametervektor ergibt sich für dieses Modell als $\theta = (F_{1:T}, E_{1:T}, G_{1:T}, W_{1:T})$.

Diese Formulierung verallgemeinert das anfängliche Beispiel dahingehend, dass man sowohl eine vektorielle Beobachtung als auch einen vektoriellen Zustand zulässt und zudem die Koeffizienten (-vektoren und -matrizen) weitestgehend frei wählen kann.

Für weiterführende Information bzgl. linearer und normalverteilter Zustandsraummodelle sei auf die Arbeiten von Hamilton (1994), Pole, West und Harrison (1994) und Petris, Campagnoli und Petrone (2009) verwiesen.

Modelle dieser Gruppe zeichnen sich dadurch aus, dass sie über eine analytische Lösung verfügen, den sogenannten Kalman-Filter.

Jedoch ist in vielen Anwendungsszenarien die Einschränkung auf lineare und normalverteilte Modelle kritisch zu bewerten, da die in der Regel komplexen ökonomischen Phänomene nicht-lineare Beziehungen implizieren. Exemplarisch seien hier diskrete Auswahlmodelle oder segmentierte Modelle, wie „Threshold“-Modelle, angeführt. Auch die Annahme von Normalverteilung ist häufig kritisch. So werden zur Modellierung gerade finanzwissenschaftlicher Phänomene häufig leptokurdische Verteilungen propagiert.

Löst man nun die Beschränkung auf lineare und normal-verteilte Modelle, gelangt man zu einer sehr allgemeinen Formulierung, wie in den Gleichung 2.5 und 2.6 gegeben.

$$y_t = f_t(x_t, \varepsilon_t, \theta) \tag{2.5}$$

$$x_t = g_t(x_{t-1}, \omega_t, \theta) \tag{2.6}$$

Hier bezeichnen $f_t(\cdot)$ bzw. $g_t(\cdot)$ lineare und/oder nicht-lineare Funktionen, die möglicherweise über die Zeit variieren. Zudem kann das Rauschen wie auch die Innovationen beliebigen Verteilungen[3] folgen. Auch in dieser Spezifikation hängen die Beobachtungs- und Übergangsgleichung von einem Parametervektor θ ab.

In der Regel besitzen solche unrestringierten Modelle keine analytische Lösung, sodass Näherungsverfahren bei der Inferenz angewendet werden müssen.

2.3. Inferenz

Bei der Verwendung von Zustandsraummodellen kann der Anwender insbesondere an zwei Aspekten interessiert sein: (1) an einer Aussage über die unbeobachtbaren Zustände $x_{1:T}$ oder (2) die Schätzung des in der Regel unbekannten Parametervektors θ. Wobei, wie später gezeigt wird, der zweite Fall ebenfalls eine Aussage über die Zustände $x_{1:T}$ impliziert.

Prinzipiell können Aussagen über die Zustände nur Wahrscheinlichkeitsaussagen sein, da die Zustände selbst zu keiner Zeit beobachtbar sind und man nur indirekt über die Beobachtungen und die Modellstruktur Rückschlüsse auf die Zustände ziehen kann.

Die Eigenschaft der Zustandsraummodelle begründet die Verwendung der bayesschen Statistik, die sich auf diese Art der Inferenz beschränkt.

In den kommenden Abschnitten wird kurz das Prinzip der bayesschen Statistik vorgestellt und die Anpassungen für die Verwendung in Zustandraummodellen angegeben. Der Abschnitt zur Inferenz schließt dann mit einer kurzen Ausführung zur Parameterschätzung im Zustandsraummodell.

2.3.1. Bayessche Statistik

Die bayessche Statistik unterscheidet sich von der klassischen (frequentistischen) Statistik unter anderem dadurch, dass sie nicht die Wahrscheinlichkeiten als ein Maß für die Unsicherheit in einem experimentellen Kontext interpretiert, sondern vielmehr die Wahrscheinlichkeit als ein Maß der Plausibilität einer Aussage interpretiert.

Grundlage für die bayessche Statistik ist das Bayes-Theorem, wie in Gleichung 2.7 dargestellt[4].

$$p(A|B) = \frac{p(A) \cdot p(B|A)}{p(B)} \tag{2.7}$$

[3] Diese Arbeit beschränkt sich auf Verteilungen, für die eine (multivariate) Dichtefunktion definiert ist bzw. aus denen Zufallszahlen gezogen werden können.

[4] Für eine detailierte Einführung in die Bayes-Statistik sei z.B. auf das Lehrbuch Greenberg (2013) verwiesen.

Wobei $p(A|B)$ die A-Posterori-Wahrscheinlichkeitsverteilung bezeichnet, also die Wahrscheinlichkeit dafür, dass das unbeobachtbare Ereignis A gültig ist, gegeben dass Ereignis B beobachtet wurde. Sie berechnet sich aus der A-Priori-Wahrscheinlichkeit ($p(A)$) des Ereignisses A, also der Wahrscheinlichkeit, dass Ereignis A überhaupt eintrifft, der Likelihood $p(B|A)$, der Eintrittswahrscheinlichkeit des beobachteten Ereignisses B unter der Annahme, dass A gültig ist und der ebenfalls als Likelihood - jedoch nur im Kontext der frequentistischen Statistik - bezeichneten Wahrscheinlichkeit $p(B)$, die die unbedingte Eintrittswahrscheinlichkeit der Beobachtung angibt.

Das Prinzip der bayesschen Statistik lässt sich in zwei Varianten auf die Inferenz in Zustandsraummodellen übertragen. Zum einen ist eine gleichzeitige (simultane) Betrachtung der gesamten Beobachtungsreihe möglich, sodass die A-Posterori-Verteilung wie folgt bezeichnet werden könnte: $p(x_1, x_2, \ldots, x_T|y_1, y_2, \ldots, y_T)$ oder kurz $p(x_{1:T}|y_{1:T})$. Methoden, die in dieser Art Aussagen über die Zustände herleiten, werden als „offline“-Methoden bezeichnet. Charakteristisch für diese Methoden ist, dass nach dem Verfügbarwerden einer neuen Beobachtung alle Berechnungen erneut durchgeführt werden müssen, was für manche Anwendungen einen unverhältnismäßigen Rechenaufwand bedeutet.

In dieser Arbeit wird dieses Vorgehen nicht weiter betrachtet. Weiterführende Informationen zu diesen Methoden findet man z.B. in der Übersichtsarbeit von Roberts und Rosenthal (2004).

Eine andere Sichtweise ergibt sich durch die vorhin getroffene Annahme, dass das Rauschen und die Innovationen unabhängig über die Zeit verteilt sind. Dann kann die A-Posterori-Verteilung $p(x_{1:T}|y_{1:T})$ umformuliert werden zu $p(x_1|y_{1:1}) \cdot p(x_2|y_{1:2}) \cdot \ldots \cdot p(x_T|y_{1:T}) = \Pi_{t=1}^{T} p(x_t|y_{1:t})$, d.h. die gesamte A-Posterori-Verteilung lässt sich durch das Produkt der sequentiellen A-Posterori-Verteilungen ersetzen. Methoden, die sich diese Sichtweise zunutze machen, werden als „online“-Methoden bezeichnet, da sie die Zustände sequentiell betrachten und jederzeit fortgesetzt werden können, wenn eine neue Beobachtung verfügbar wird.

Diese Arbeit beschränkt sich aus zwei Gründen auf diese sequentielle Sichtweise. (1) Methoden dieser Art sind flexibel einsetzbar. So sind mit dieser auch Echtzeitanwendungen möglich, wie sie z.B. in der Finanzmarktstatistik oder dem Risikomanagement anzutreffen sind. (2) Die Komplexität der Berechnungen ist geringer, dadurch werden mehrere Lösungsansätze anwendbar.

Im Folgenden wird die Umsetzung der bayesschen Statistik für diese sequentielle Sichtweise detailliert beschrieben, mit deren Hilfe Aussagen über die Zustände $x_{1:T}$ in einem allgemeinen Zustandsraummodell möglich sind.

2.3.2. Sequentielles bayessches Filtern

Bei der sequentiellen Sichtweise ist man an der A-Posterori-Verteilung $p(x_t|y_{1:t})$ für $t = 1, ..., T$ interessiert. Diese ergibt sich mit Hilfe des Bayes-Theorems als

$$p(x_t|y_{1:t}) = \frac{p(x_t) \cdot p(y_t|x_t)}{p(y_t)}. \tag{2.8}$$

Durch die rekursive Struktur der Zustandsraummodelle, die sich aus der Zustandsgleichung ergibt (siehe z.B. Gleichung 2.4 oder 2.6), folgt, dass die A-Priori-Verteilung $p(x_t)$ selbst von dem Zustand der Vorperiode und Aussagen über diesen von den Beobachtungen $y_{1:t-1}$ abhängig ist.

Aus diesem Grund ist es zweckmäßig, die Notation wie in Gleichung 2.9 anzupassen[5].

$$p(x_t|y_{1:t}, \theta) = \frac{p(x_t|y_{1:t-1}, \theta) \cdot p(y_t|x_t, \theta)}{p(y_t|\theta)} \tag{2.9}$$

$$\text{mit} \quad p(y_t|\theta) = \int_X p(x_t|y_{1:t-1}, \theta) \cdot p(y_t|x_t, \theta)\, dx_t \tag{2.10}$$

Im Kontext der Zustandsraummodellierung wird diese A-Posterori-Verteilung als Filterverteilung bezeichnet.

Die A-Priori-Verteilung ($p(x_t|y_{1:t-1}, \theta)$) ergibt sich wie in Gleichung 2.11 beschrieben.

$$p(x_t|y_{1:t-1}, \theta) = \int_X p(x_t|x_{t-1}, \theta) \cdot p(x_{t-1}|y_{1:t-1}, \theta)\, dx_{t-1} \tag{2.11}$$

Hier bezeichnet $p(x_t|x_{t-1}, \theta)$ die Übergangswahrscheinlichkeit von einem Zustand x_{t-1} der Vorperiode zu dem aktuellen Zustand (x_t). Die Übergangswahrscheinlichkeit ergibt sich aus der jeweiligen Spezifikation der Zustandsgleichung des Zustandsraummodells und der Verteilungsannahmen bzgl. der Innovationen.

In der Formulierung des Vorhersageschritts ist eine zweite Rekursion zu erkennen, die sich aus der sequentiellen Anwendung des Bayes-Theorems ergibt. Demzufolge ist für die Berechnung der A-Priori-Verteilung für den Zeitpunkt t die A-Posterori-Verteilung für den Zeitpunkt $t - 1$ notwendig.

Für eine Rekursion dieser Art muss ein Anfangspunkt definiert werden, der ebenfalls A-Priori-Verteilung genannt und mit $p_0(x_t)$ bezeichnet wird. Die Wahl dieser Verteilung wird an dieser Stelle nicht weiter thematisiert und es sei auf die entsprechende Literatur

[5] Bei dieser Formulierung wurde zusätzlich die Abhängigkeit von dem Parametervektor θ berücksichtigt, was in Abschnitt 2.3.3 wieder aufgegriffen wird.

verwiesen. Es sei lediglich darauf hingewiesen, dass der Einfluss der gewählten A-Priori-Verteilung ($p_0(x_t)$) mit fortschreitender Zeit abnimmt und daher die Wahl der A-Priori-Verteilung $p_0(x_t)$ in Zustandraummodellen weniger kritisch ist als in Querschnittsanalysen.

Mit dem sequentiellen bayesschen Filtern wird rekursiv die A-Posterori-Verteilung für den Zustand zum Zeitpunkt t bestimmt. Häufig ist man jedoch nicht an der Wahrscheinlichkeitsverteilung selbst interessiert, sondern an statistischen Aussagen über diese, wie z.B. Erwartungswert oder Varianz. Exemplarisch ist die Gleichung 2.12 für den Erwartungswert und 2.13 für die Varianz einer eindimensionalen A-Posterori-Verteilung angegeben.

$$E(x_t) = \int_X x_t \cdot p(x_t|y_{1:t}, \theta)\, dx_t \tag{2.12}$$

$$V(x_t) = \int_X (x_t - E(x_t))^2 \cdot p(x_t|y_{1:t}, \theta)\, dx_t \tag{2.13}$$

Bezüglich komplexerer Auswertungen, wie z.B. Hypothesentest, Regionen hoher Dichten („high-density-regions“, analog zu den Konfidenzintervallen in der klassischen Statistik) oder auch Modellselektionen, sei auf die entsprechende Literatur, wie der Einführung von Marin und Robert (2007), verwiesen.

Zusammenfassend ergibt sich für das seqeuentielle bayessche Filtern der Algorithmus wie in Abbildung 2.1 beschrieben.

1. `definiere die A-Priori-Verteilung` $p_0(x_t)$
2. `für t = 1, ..., T`

   `(Vorhersageschritt:)`

   `falls` $t = 1$: $p(x_1|y_0, \theta) = \int_X p(x_t|x_{t-1}, \theta) \cdot p_0(x_{t-1})\, dx_{t-1}$

   `falls` $t > 1$: $p(x_t|y_{1:t-1}, \theta) = \int_X p(x_t|x_{t-1}, \theta) \cdot p(x_{t-1}|y_{1:t})\, dx_{t-1}$

   `(Filterschritt:)`

   $p(x_t|y_{1:t}) = \frac{p(x_t|y_{1:t-1},\theta)\cdot p(y_t|x_t,\theta)}{p(y_t|\theta)}$

   `(Auswertung der A-Posterori-Verteilung)`

Abbildung 2.1.: Algorithmus des sequentielles bayesschen Filterns

Für die praktische Umsetzung müssen die bisher nur abstrakt mit $p(x_t|x_{t-1}, \theta)$ und $p(y_t|x_t, \theta)$ bezeichneten Verteilungen konkretisiert werden. Wobei sich die Übergangswahrscheinlichkeit $p(x_t|x_{t-1}, \theta)$ aus der Zustandsgleichung und die Likelihood $p(y_t|x_t, \theta)$ aus der Beob-

achtungsgleichung ergibt. Exemplarisch für das „Random Walk + Noise“-Modell aus Abschnitt 2.1 ergeben sich die folgenden beiden Wahrscheinlichkeitsdichten, wobei $\phi(\cdot)$ die Wahrscheinlichkeitsdichte der Standardnormalverteilung bezeichnet:

$$\begin{aligned} p(x_t|x_{t-1},\theta) &= \phi\left(\frac{x_t - 1\cdot x_{t-1}}{\sigma_\omega}\right) \\ p(y_t|x_t,\theta) &= \phi\left(\frac{y_t - 1\cdot x_t}{\sigma_\varepsilon}\right) \end{aligned}$$

Für die Berechnungen der A-Posterori-Verteilungen bzw. deren Statistiken ist das Lösen von Integralen nötig (siehe z.B. die Gleichungen 2.10, 2.11, 2.12 und 2.13). Für ein allgemeines Zustandsraummodell sind diese Integrale nicht in analytischer Form gegeben. Lediglich für einige spezielle Spezifikationen sind analytische Lösungen verfügbar. So sind zum Beispiel die linearen und normal-verteilten Zustandsraummodelle, wie sie in Abschnitt 2.2 beschrieben wurden, analytisch mit dem sogenannten Kalman-Filter berechenbar[6].

Alle Modelle, für die es keine analytische Lösung gibt, können nur näherungsweise durch Approximationsverfahren berechnet werden.

Die Tabelle 2.1 fasst verschiedene Lösungsansätze zusammen.

Typisierung:	**analytisch**	**approximativ-exakt**	**exakt-approximativ**
Beschreibung:	exakte Berechnung möglich	das approximierte Modell wird analytisch exakt berechnet	das exakte Modell wird näherungsweise berechnet
Beispiele:	Kalman-Filter (KF) (Kalman, 1960)	Extended-KF (Athans et al.,1968) Unscented-KF (Julier und Uhlmann, 1997)	Boostrap-Filter (Gordon et al.,1993) Quadrature-Filter (Kitagawa, 1987)

Tabelle 2.1.: Gegenüberstellung der Lösungsansätze

In dieser Arbeit wird der „exakt-approximativ“-Ansatz verfolgt, der relativ universell einsetzbar ist und dessen Approximationsgüte, zumindest theoretisch, beliebig skalierbar ist. Die Methoden des „approximativ-exakt“-Ansatzes sind schnell zu berechnen, allerdings können diese Methoden, je nach Modellspezifikation, zu einer sehr schlechten Approximationsgüte führen (Lopes und Tsay, 2011).

[6]Eine Beschreibung des Kalman-Filters erfolg im Anhang A.

2.3.3. Parameterschätzung

Neben der funktionalen Form determiniert der Parametervektor θ das Zustandsraummodell. In der Regel ist dieser Vektor unbekannt und muss geschätzt werden.

Zwei Konzepte zur Schätzung des Parametervektors im Zustandsraummodell haben sich in der Literatur etabliert. Zum einen der Maximum-Likelihood-Schätzer, zum anderen das endogene Modellieren der unbekannten Parameter.

Ausgangspunkt ist die sogenannte Likelihood[7], die Wahrscheinlichkeit der Beobachtungen bei gegeben Parametervektor. Sie ergibt sich gemäß Gleichung 2.14, wobei $p(y_t|\theta)$ gerade dem Nenner in Gleichung 2.9 entspricht.

$$log\mathcal{L}(\theta) = \sum_{t=1}^{T} log(p(y_t|\theta)) \tag{2.14}$$

Für die Maximum-Likelihood-Schätzung wird der Parametervektor gesucht, der die logarithmierten Likelihood maximiert.

Die konkrete Berechnung der $p(y_t|\theta)$ für $t = 1, ..., T$ unterscheidet sich bei den verschiedenen Lösungsansätzen (analytisch, sequentielle Monte-Carlo, sequentielle numerische Integration) teilweise erheblich. Aus diesem Grund wird die Berechnung an den entsprechenden Stellen konkretisiert.

Eine andere Möglichkeit der Bestimmung der unbekannten Parameter stellt die endogene Modellierung der Hyperparameter dar. In der entsprechenden Literatur wird dies als „parameter-learning“ (Lopes und Tsay, 2011) oder „self-organized state-space model“ (Kitagawa, 1998) bezeichnet. Kerngedanke ist, die unbekannten und auch unbeobachtbaren Parameter als zusätzliche Zustände in den Zustandsvektor (x_t) aufzunehmen und die Methode des sequentiellen bayesschen Filterns auch auf diese anzuwenden. Als Resultat ergeben sich mit zunehmender Zahl von Beobachtungen immer präzisere Schätzungen der Parameter.

Im Rahmen dieser Arbeit wird die Parameterschätzung nicht weiter vertieft. Insbesondere für die Variante des „parameter-learning“ sei auf die entsprechende Literatur verwiesen. Man muss jedoch hervorheben, dass gerade die Parameterbestimmung ein kritischer Punkt bei der praktischen Anwendung von Zustandsraummodellen darstellt. Für nicht analytisch lösbare Zustandraummodelle ergeben sich, bedingt durch die approximierenden Berechnungsverfahren, zum Teil erhebliche Probleme, die in der aktuellen Literatur vermehrt diskutiert werden und sich verschiedene Lösungsansätze herauskristallisieren (siehe z.B. Malik und Pitt (2011); Ionides u. a. (2011)). Kern des Problems ist, dass die Likelihood-Funktion im Fall der simulationsbasierten Näherungsverfahren nicht stetig und somit nicht hinreichend glatt in ihren Paramtern ist. Dies hat zur Folge, dass die Likelihood nur

[7] Aus rechentechnischen Gründen wird in der Regel die logarithmierte Likelihood verwendet.

schwer numerisch maximiert werden kann und somit der Maximum-Likelihood-Schätzer praktisch nicht anwendbar ist. Wie in Kapitel 4 ausgeführt wird, liegt dieses Problem nicht vor, wenn die numerische Integration bei der Näherungslösung verwendet wird.

3. Forschungsstand

Wie im vorherigen Kapitel angedeutet, existieren nur für wenige spezielle Zustandsraummodelle analytische Lösungen. Die Inferenz in allen anderen Modellen kann nur näherungsweise berechnet werden, wobei diese Arbeit sich auf die Betrachtung der „exakt-approximativen“ Lösungsansätze beschränkt (siehe Tabelle 2.1).

In diesem Kapitel wird der aktuelle Forschungsstand beschrieben. In Abschnitt 3.1 werden die simulationsbasierten Methoden vorgestellt, welche die aktuellen Diskussionen dominieren. Als zweiter Ansatz wird in Abschnitt 3.2 ein quadratur-basiertes Verfahren vorgestellt. Im weiteren Verlauf der Arbeit bilden die simulationsbasierten Methoden stets die Referenzklasse für die im Kapitel 4 weiterentwickelten quadraturbasierten Verfahren.

3.1. Sequentielle Monte-Carlo-Methoden (sMC)

3.1.1. Einführung in die sMC-Methoden

Im Rahmen des sequentiellen bayesschen Filterns werden rekursiv Wahrscheinlichkeitsdichten für die unbeobachtbaren Zustände berechnet (vgl. das Ablaufschema in Abbildung 2.1). Die Berechnung ist in vielen Fällen nicht analytisch möglich und führt fast immer zu Wahrscheinlichkeitsdichten, für die es keine (einfache) funktionale Beschreibung gibt.

Einen recht universellen Lösungsansatz bieten die Monte-Carlo-Methoden.

Monte-Carlo-Methoden, auch als simulationsbasierte Methoden bezeichnet, zielen darauf ab, ein stochastisches Phänomen durch eine Vielzahl von zufällig generierten Realisationen, den sogenannten Ziehungen, zu approximieren. Diese Ziehungen werden gemäß eines Modells generiert, über dieses man durch Auswertung der Ziehungen Aussagen treffen kann[1].

Im Kontext des sequentiellen bayesschen Filterns gilt es, die entsprechenden Wahrscheinlichkeitsdichten rekursiv zu approximieren. Das Modell, welches der Generierung der Ziehungen, den sogenannten Partikeln, zugrunde liegt, ist das entsprechende Zustandsraummodell, wobei der stochastische Charakter durch den Innovations- und Rauschterm ge-

[1] Für eine detaillierte Einführung in die Monte-Carlo-Methoden sei auf Robert und Casella (2004) verwiesen. Die gleichen Autoren bieten auch eine sehr angewandte Einführung an (Robert und Casella, 2010).

geben ist. Wegen der sequentiellen Struktur spricht man von sequentiellen Monte-Carlo-Methoden (sMC), diese Methoden werden auch als Partikel-Filter bezeichnet.

Seit den 1990er-Jahren haben diese Methoden an Popularität gewonnen, sodass sie schon seit Mitte der 1990er Jahre als „state of the art“ bezeichnet werden konnten. Dies hängt wahrscheinlich mit der relativ einfachen Implementierung zusammen. Zudem sind die Methoden universell einsetzbar.

Die genannten Vorzüge werden dadurch „erkauft“, dass diese Approximationsmethoden ein in der Regel schlechtes Konvergenzverhalten aufweisen, dazu mehr in Kapitel 5.

Gordon, Salmond und Smith (1993) haben erstmals einen allgemein anwendbaren Algorithmus für einen Partikel-Filter vorgestellt, der als „Bootstrap Filter“ (BF) bezeichnet wird. Pitt und Shephard (1999) haben eine im Ansatz modifizierte Variante des Algorithmus vorgeschlagen, den „Auxilliary Particle Filter“ (APF). Diese beiden Algorithmen bilden bis heute den Ausgangspunkt für die Weiterentwicklungen im Bereich der sequentiellen Monte-Carlo-Methoden.

Im Folgenden werden die Algorithmen von Gordon, Salmond und Smith sowie Pitt und Shephard vorgestellt, die später, in Kapitel 5, mit der Methode der sequentiellen numerischen Integration verglichen werden. Dieses Kapitel schließt mit einer Gegenüberstellung der in der Literatur diskutierten Vor- und Nachteilen der sequentiellen Monte-Carlo Methode.

3.1.2. Partikel-Filter: Algorithmen

Eine Einführung in die Monte-Carlo-Methode wird an dieser Stelle nicht gegeben, hierfür sei auf die Arbeiten von z.B. Robert und Casella (2004, insb. Kapitel 3) verwiesen. Eine umfassende theoretische Diskussion der Methode der Partikel-Filter bietet Doucet u. a. (2001) und eine für den Anwender konzipierten Überblick bietet Lopes und Tsay (2011). Und zuletzt liefert Chen (2003) einen sehr umfassenden Überblick über die Vielzahl von Varianten des Partikel-Filters.

In diesem Abschnitt werden die beiden grundlegenden Algorithmen, der des „Bootstrap Filter“ und der des „Auxiliary Particle Filter“ vorgestellt und miteinander verglichen.

„Bootstrap Filter“

Der von Gordon, Salmond und Smith (1993) vorgeschlagene Partikelfilter kommt der Struktur der Zustandsraummodelle sehr nahe, da er die sequentielle Struktur unverändert übernimmt (vgl. Abbildung 2.1 mit dem Ablaufschema des sequentiellen bayesschen Filterns).

Abbildung 3.1 fasst den Algorithmus zusammen.

Dieser beginnt mit der Generierung von N Partikel aus der A-Priori-Verteilung p_o. Danach werden für jeden Zeitpunkt $t = 1, ..., T$ ...

(1) die Partikel gemäß der Zustandsgleichung fortgeschrieben (engl. propagation). Dafür werden gemäß der getroffenen Verteilungsannahme zufällig Innovationen gezogen und auf die Partikel angewendet. Die resultierenden neuen Partikel sind gemäß der A-Priori-Verteilung $p(x_t|y_{1:t-1})$ verteilt.

(2) Die fortgeschriebenen Partikel werden dann gemäß ihrer Eintrittswahrscheinlichkeit (Likelihood, $p(y_t|x_t)$) mit Zurücklegen gezogen (engl. resampling). Die resultierende Menge von Partikeln repräsentiert die A-Posterori-Verteilung des Zustandsraummodells zum Zeitpunkt t, $p(x_t|y_{1:t})$.

Die Partikel der A-Posterori-Verteilung können dann zur Analyse genutzt werden, indem Momente wie z.B. Mittelwert und Varianz oder andere Eigenschaften, wie die „High-Density-Regions“, bestimmt werden.

Die logarithmierte Likelihood des gesamten Modells ergibt sich gemäß Gleichung 3.1

$$log\mathcal{L}(\theta) = \sum_{t=1}^{T} log\left(\frac{1}{N}\sum_{t=1}^{N} w_{i,t}\right) \tag{3.1}$$

1. N Partikel aus der A-Priori-Verteilung $\{x_{i,o} \sim p_0(x)\}_{i=1,...,N}$
2. für alle $t = 1, ..., T$
   (a) fortschreiben der Partikel $\{\tilde{x}_{i,t} \sim p(x_{i,t}|x_{i,t-1}, \theta)\}_{i=1,...,N}$
   (b) Resampling der Partikel $(\tilde{x}_{i,t})$ mit individuellen Gewichten $(w_{i,t} = p(y_t|\tilde{x}_{i,t}, \theta)) \rightarrow \{x_{i,t}\}_{i=1,...,N}$
   (c) Inferenz anhand der A-Posterori-Partikel $\{x_{i,t}\}_{i=1,...,N}$

Abbildung 3.1.: Algorithmus des Bootstrap Filters

Abbildung 3.2 illustriert in der linken Hälfte das Ablaufschema des BF. Ausgangspunkt (oben) ist die A-Posterori-Verteilung der Vorperiode $(t-1)$, nach der Fortschreibung erfolgt das Berechnen der Gewichte und abschließend das Resampling. Auffällig ist, dass die meiste Wahrscheinlichkeitsmasse $w^{(i)}$ sich auf nur wenige Partikel (hier 3-4 Partikel) verteilt. Daraus ergibt sich, dass die A-Posteriori-Verteilung durch die Wiederholung von lediglich 4 Partikeln abgebildet wird. Damit verbunden sind negative Eigenschaften, wie die relativ schlechte Approximationsgüte.

Der in der Folge diskutierte „Auxiliary Particle Filter“ greift genau dieses Problem auf.

„Auxiliary Particle Filter“

Der „Auxiliary Particle Filter“ (Pitt und Shephard (1999)) umgeht das Problem des Bootstrap Filters, indem es einen zusätzlichen Resampling-Schritt einfügt. In Abbildung 3.2

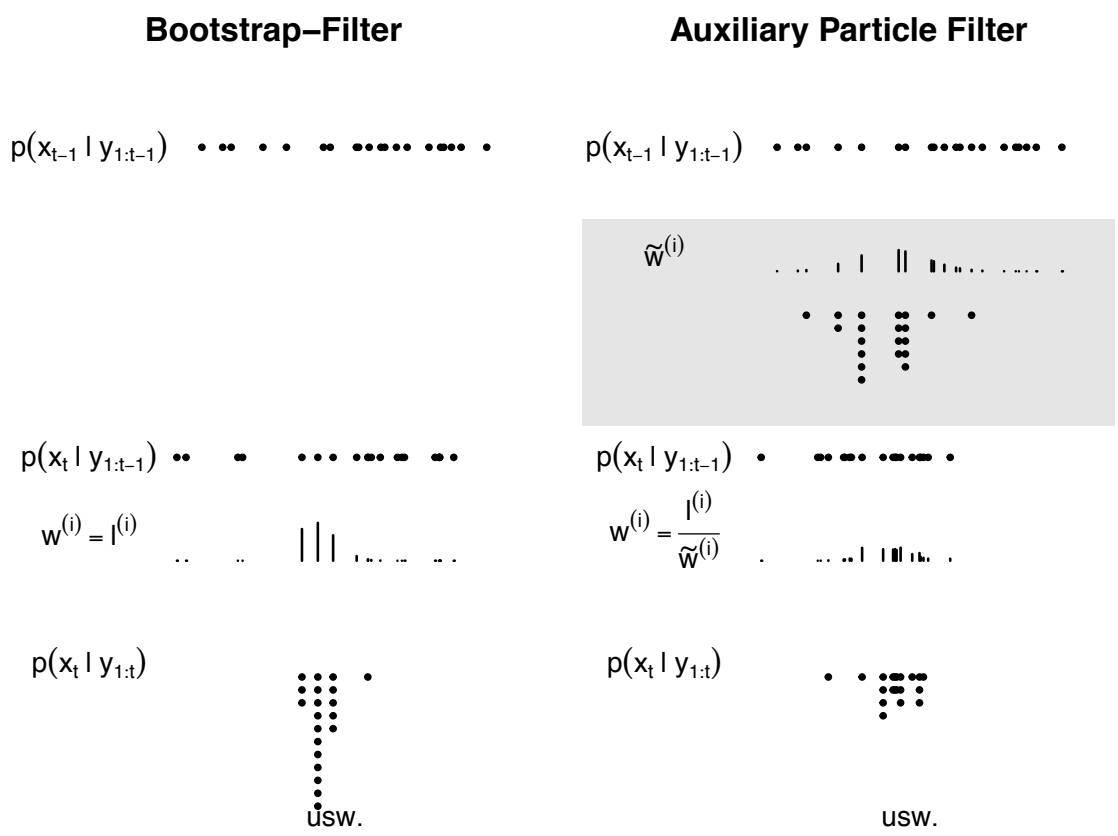

Abbildung 3.2.: Illustration des Boostrap- und Auxiliary Particle Filter

(rechts) ist das angepasste Vorgehen illustriert. Ausgehend von dem identischen Satz von A-Posterori-Partikeln wird nun im APF eine Auswahl von Partikeln getroffen, welche mit Blick auf die spätere Likelihood-Funktion ($p(y_t|x_t, \theta)$) von besonderer Bedeutung sind. Diese Auswahl wird durch einen Resampling-Schritt erzeugt. Das Vorgehen sichert, dass die A-Priori- Partikel ($p(x_t|y_{1:t-1})$) sich in der Region konzentrieren, aus der die Partikel der A-Posterori-Verteilung gezogen werden.

In der Abbildung ist zu erkennen, dass die A-Posterori-Verteilung des APF mit einer größeren Zahl von eindeutigen Partikeln abgebildet wird (hier 10 Partikel), was in der Regel mit eine besseren Approximationsgüte einhergeht.

In Abbildung 3.3 wird nochmals der Algorithmus formal zusammengefasst.

Die logarithmierten Likelihood beim APF ergibt analog zum Bootstrap Filter als:

$$log\mathcal{L} = \sum_{t=1}^{T} log\left(\frac{1}{N}\sum_{i=1}^{N}\tilde{w}_{i,t}\right)$$

1. N Partikel aus der A-Priori Verteilung $\{x_{i,o} \sim p_0(x)\}_{i=1,...,N}$
2. für alle $t = 1, ..., T$

   (a-1) Resampling der Partikel $(x_{i,t-1})$ mit indiviuduellen Gewichten $(\tilde{w}_{i,t} = p(y_t|g(x_{i,t-1})) \rightarrow \{\tilde{x}_{i,t-1}\}_{i=1,...,N}$

   (a) fortschreiben der Partikel $\{\tilde{x}_{i,t} \sim p(x_{i,t}|\tilde{x}_{i,t-1}, \theta)\}_{i=1,...,N}$

   (b) Resampling der Partikel $(\tilde{x}_{i,t})$ mit individuellen Gewichten $(w_{i,t} = p(y_t|\tilde{x}_{i,t}, \theta)/\tilde{w}_{i,t}) \rightarrow \{x_{i,t}\}_{i=1,...,N}$

   (c) Inferenz anhand der A-Posterori-Partikel $\{x_{i,t}\}_{i=1,...,N}$

Abbildung 3.3.: Algorithmus des „Auxiliary Particle Filter“

3.1.3. Vor- und Nachteile der Partikel-Filter

Wie schon zu Beginn des Kapitels angesprochen, handelt es sich bei dem Partikel-Filter um einen sehr universellen Lösungsansatz.

Aus Anwendungssicht liegt sein größter Vorteil in der einfachen Implementierung und der weiten Einsetzbarkeit. Zudem ist dieser Ansatz auch auf hoch dimensionale Probleme anwendbar (Kitagawa, 1996).

Aus theoretischer Sicht hat er die positive Eigenschaft der „fasst sicheren Konvergenz“ Crisan und Doucet (2002).

Während sich die Vorteile der Partikel-Filter überwiegend auf die Flexibiliät beziehen, die sie dem Anwender bieten, beziehen sich die Nachteile überwiegend auf den Approximationsfehler und dessen Konvergenzverhalten. So konvergiert der Approximationsfehler der Partikelfilter mit der Ordnung $\mathcal{O}(N^{-1/2})$ gegen Null. Anschaulich bedeutet dies, dass man die Anzahl der Partikel verhundertfachen muss, um die Genauigkeit des Ergebnisses um eine Dezimalstelle zu erhöhen. Somit sind Partikel-Filter praktisch ungeeignet, wenn man an hoch präzisen Ergebnissen interessiert ist (Bolviken und Storvik, 2001).

Neben dem Konvergenzverhalten ist das Fehlerniveau beim Partikel-Filter relativ hoch und zudem abhängig von der Zahl der Dimensionen (Rebeschini und Handel, 2013). Das bedeutet, dass Partikel-Filter zwar relativ einfach für hoch dimensionale Probleme erweiterbar sind, allerdings verschlechtert sich mit wachsender Zahl von Dimensionen die Approximationsgüte.

Als ein weiterer Nachteil der Partikel-Filter kann angesehen werden, dass die log. Likelihood nicht stetig in ihren Argumenten ist und dadurch nicht bzw. nur schlecht geeignet ist, in einem Maximum-Likelihood-Schätzer genutzt zu werden (Malik und Pitt, 2011).

3.2. Sequentielle numerische Integration (sNI)[2]

In diesem Kapitel wird kurz die historische Entwicklung der numerischen Integration im Bereich der Zustandsraummodellierung skizziert. In der Folge wird der ursprüngliche Algorithmus von Kitagawa (1987) vorgestellt.

3.2.1. Historische Entwicklung

In den 1970er und 1980er Jahren wurde die numerische Integration für den Einsatz in der bayesschen Statistik diskutiert (z.B. Reilly, 1976; Naylor und Smith, 1982). Die damaligen Algorithmen waren in der Regel für statische eindimensionale Probleme ausgelegt und verwendeten Newton-Cotes-Quadraturen oder in selteneren Fällen Gauss-Quadraturen. Durch den statischen Charakter der Modelle war nur eine einmalige Berechnung der A-Posterori-Verteilung nötig.

Im Kontext des sequentiellen bayesschen Filterns hat Kitagawa (1987) einen sehr allgemeinen Algorithmus zur sequentiellen numerischen Integration vorgestellt. Als Quadraturregel nutzt er die Trapezregel auf einem statischen Gitter. Dieser Ansatz wurde für niedrig dimensionale Probleme (1-2 Dimensionen) von verschiedenen Autoren angewandt, z.B. bei Fridman und Harris (1998) für die Schätzung eines eindimensionalen stochastichen Volatilitäts-Modells oder bei Bolviken und Storvik (2001), wo der Algorithmus den Partikelfiltern gegenübergestellt wird. Gerade diese beiden Arbeiten schlagen auch kleinere Erweiterungen vor, analysieren deren Effekt nicht systematisch.

Die numerische Quadratur in dieser Form hat den entscheidenden Nachteil, dass sie praktisch nicht auf höherdimensionale Probleme ($d > 3$) verallgemeinert werden konnte (Kitagawa, 1987, Seite 1040), weshalb seit den 1990er Jahren kaum mehr Publikationen in diesem Bereich erschienen sind. Dieses Problem stellt sich sowohl bei den statischen als auch bei den dynamischen Modellen, wie den Zustandsraummodellen.

In der heutigen Literatur sind Quadraturen in hybriden Filtern zu finden. Diese arbeiten teilweise analytisch und teilweise mit numerischer Integration. Ein Beispiel hiefür ist der „Quadrature Kalman Filter“ von Arasaratnam, Haykin und Elliott (2007).

3.2.2. Algorithmus nach Kitagawa (1987)

Wie in Abschnitt 2.3.2 beschrieben, erfolgt die rekursive Berechnung der Vorhersagedichte (A-Prirori-Verteilung) und der Filterdichte (A-Posterori-Verteilung) unter zur Hilfenahme des Bayes-Theorems, wie in den beiden Gleichungen 3.2 und 3.3 angegeben (hier um die Approximation durch die numerische Quadratur ergänzt).

[2] Im Anhang B sind kurz die grundlegenden Konzepte der numerischen Integration zusammengefasst, die als Grundlage für diesen Abschnitt und das folgende Kapitel benötigt werden.

$$
\begin{aligned}
p(x_t|y_{1:t-1},\theta) &= \int_X p(x_t|x_{t-1},\theta)\cdot p(x_{t-1}|y_{1:t-1},\theta)\,dx_{t-1} \\
&\approx \sum_{i=1}^{n}(w_i\cdot p(x_t|x_{t-1}^{(i)})\cdot p(x_{t-1}^{(i)}|y_{1:t-1}))
\end{aligned}
\tag{3.2}
$$

$$
p(x_t|y_{1:t},\theta) \quad = \quad \frac{p(x_t|y_{1:t-1},\theta)\cdot p(y_t|x_t,\theta)}{p(y_t|\theta)} \tag{3.3}
$$

$$
\begin{aligned}
p(y_t|\theta) &= \int_X p(x_t|y_{1:t-1},\theta)\cdot p(y_t|x_t,\theta)\,dx_t \\
&\approx \sum_{i=1}^{n}(w_i\cdot p(x_t^{(i)}|y_{1:t-1},\theta)\cdot p(y_t|x_t^{(i)}))
\end{aligned}
\tag{3.4}
$$

Im Unterschied zur sMC-Methode werden bei der sNI die Integrale nicht implizit durch die Partikel-Repräsentation gelöst, sondern explizit mit der Methode der numerischen Integration berechnet.

Eine technisch einfache, jedoch allgemein einsetzbare Umsetzung schlägt Kitagawa (1987) vor. Der Algorithmus beginnt chronologisch mit der Wahl der Stützstellen. Da er die äquidistante Trapezregel für die numerische Integration nutzt, müssen also konkret das Intervall $[x_1, x_n]$ sowie die Anzahl der Subintervalle festgelegt werden. Das Gitter ist bei Kitagawa statisch, d.h. alle Dichten werden im Zeitablauf mit dem selben Satz von Stützstellen approximiert.

Das eigentliche sequentielle bayessche Filtern beginnt damit, dass die A-Priori-Verteilung bestimmt und an den Stützstellen evaluiert wird.

Danach folgt die rekursive Berechnung der Vorhersagedichte und der Filterdichte gemäß den Gleichungen 3.2 und 3.3.

Abbildung 3.4 fasst den genauen Algorithmus zusammen und Abbildung 3.5 illustriert das Prinzip des statischen Gitters.

Die Likelihood des gesamten Modells berechnet sich durch: $log\mathcal{L} = \sum_{t=1}^{T} log(C_t)$

Dieser Algorithmus wurde unter anderem von Kitagawa (1987) und Fridman und Harris (1998) zur Berechnung von eindimensionalen, nicht normalverteilten und/oder nicht linearen Zustandsraummodellen verwendet. Kitagawa schlägt auch eine Erweiterung für mehrdimensionale Zustandsraummodelle vor, jedoch wirkt in diesem Algorithmus der „Fluch der Dimensionen“ besonders stark.

1. wähle die Stützstellen $x_1, \ldots, x_n$ und Gewichte $w_1, ..., w_n$
2. evaluiere die A-Priori-Dichte an den Stützstellen

   $f_0^{(i)} = p_0(x_i) \quad \forall i = 1, ..., n$

3. für alle $t = 1, ..., T$:

   (a) berechne die Dichte der Vorhersagewahrscheinlichkeit
   für alle $j = 1, ..., n$: $p_t^{(j)} = \sum_{i=1}^{n} (w_i \cdot p(x_j|x_i) \cdot f_{t-1}^{(i)})$
   (b) berechne die A-Posterori-Dichte
   für alle $j = 1, ..., n$: $\hat{f}_t^{(j)} = p_t^{(j)} \cdot p(y_t|x_j)$
   berechne die Normalisierungskonstante
   $C_t = \sum_{i=1}^{n} (w_i \cdot \hat{f}_t^{(i)})$
   für alle $j = 1, ..., n$: $f_t^{(j)} = \frac{\hat{f}_t^{(j)}}{C_t}$
   (c) Auswertung A-Posterori-Dichte (optional)

Abbildung 3.4.: Algorithmus der sNI nach Kitagawa (1987)

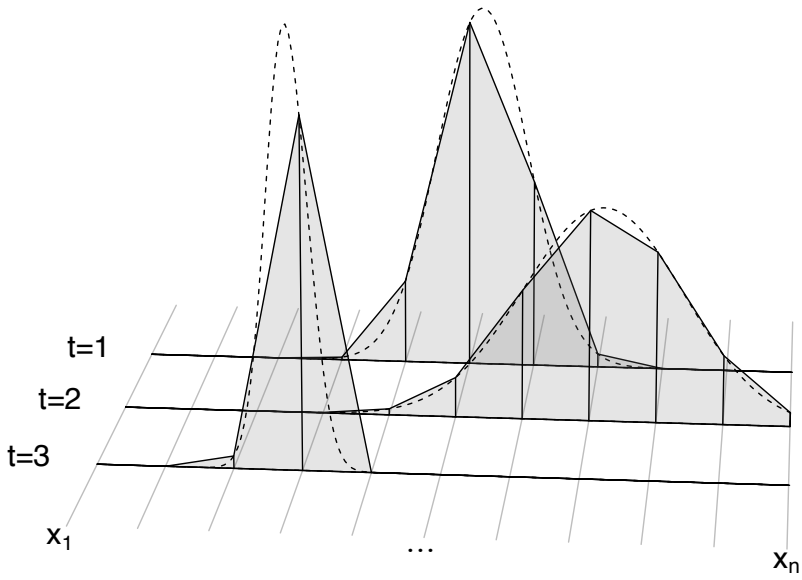

Abbildung 3.5.: Illustration der statischen Gitter bei Kitagawa (1987). *Dargestellt sind die wahren, nicht normalisierten A-Posterori-Dichten (gestrichelte Funktion, analog zu $\hat{f}_t$ in Abbildung 3.4) und die Approximation der Normalisierungskonstante durch die Trapezregel (grau schraffierte Flächen, analog zu C_t in Abbildung 3.4) für die Zeitpunkte t=1,2 und 3.*

4. Erweiterungen der sNI

4.1. Schwachstellen des ursprünglichen Algorithmus

Der Algorithmus von Kitagawa findet in einer Fülle von Veröffentlichungen bis heute Anwendung. Dennoch wurden bereits bei seiner Publikation 1987 in Kommentaren zu diesem Artikel Probleme diskutiert (unter anderem Kohn und Ansley (1987); Martin und Raftery (1987)).

An dieser Stelle werden nun kurz drei Probleme illustriert, für deren Behebung im weiteren Verlauf des Kapitels Erweiterungen vorgeschlagen werden.

4.1.1. Rechenaufwand

Im ursprünglichen Algorithmus wird für die Erweiterung vom eindimensionalen zum d-dimensionalen Zustandsraummodell die Produktregel angewendet[1].

In den Anwendungsbeispielen gibt Kitagawa an, 400 Stützstellen im 1D-Fall zu nutzen. Für ein auf d Dimensionen erweitertes Modell ergeben sich somit 400^d Stützstellen, was schon bei einer geringen Anzahl von Dimensionen ($d = 2$ oder $d = 3$) zu einem erheblichen Rechenaufwand führt und Modelle mit $d > 3$ praktisch nicht mehr berechenbar sind.

4.1.2. Effizienz

Der von Kitagawa vorgestellte Algorithmus ist im Allgemeinen ineffizient.

Als effizient bzw. zu einem anderen Algorithmus relativ effizient wird ein Algorithmus in diesem Zusammenhang bezeichnet, wenn er mit gegebener Anzahl von Stützstellen eine maximale bzw. bessere Approximationsgüte erreicht bzw. eine vorgegebene Approximationsgüte mit einer minimalen bzw. geringeren Anzahl von Stützstellen erreicht.

Zum einen ist die Approximationsgüte im Allgemeinen dadurch steigerbar, dass man die Trapezregel durch eine Gauß-Quadratur ersetzt. Wobei im konkreten Fall auch andere Quadraturen effizienter sein können. Diesbezüglich wurde der Algorithmus z.B. von Fridman und Harris (1998) angepasst.

[1] Wegen der Produktregel vgl. Abschnitt B im Anhang.

Zum anderen bedingt das statische Gitter die Ineffizienz. In Abbildung 3.5 ist dies bereits erkennbar. Die Probleme sind insbesondere bei Approximationen bei den Zeitpunkten $t = 2$ und $t = 3$ ausgeprägt.

Bei $t = 2$ wird die zu integrierende Funktion am rechten Ende abgeschnitten. Durch eine Verschiebung der Stützstellen nach rechts würde die Approximationsgüte gesteigert werden.

Im Zeitpunkt $t = 3$ ist die Funktion an vielen Stützstellen nahe Null. An diesen Stellen ist der Beitrag zum Integral nur gering. So könnte man die Zahl der Stützstellen in diesem Fall stark reduzieren, ohne die Approximationsgüte (merklich) zu beeinflussen.

4.1.3. A-Priori-Kenntnisse

Ein weiterer Nachteil des statischen Gitters liegt in der Notwendigkeit, vor Beginn der Berechnungen, die Stützstellen festzulegen. In der Regel sind zudem mehrere Versuche nötig, um eine adäquate Wahl der Gitterpunkte zu treffen. Aus genau diesem Grund ist aus praktischen Gesichtspunkten die Verwendung statischer Gitter mit der Restriktion auf Modelle mit stationärem Zustandsraum verbunden, was ebenfalls als ein Schwachpunkt angesehen werden kann. Denn für stationäre Modelle liegen die zu approximierenden Wahrscheinlichkeitsdichten in einem festen Korridor, was bei nicht stationären Modellen nicht gegeben sein muss.

4.2. Erweiterungen - Überblick

Die im vorherigen Kapitel aufgeführten Probleme bei der ursprünglichen sNI-Methode können in zwei Gruppen aufgeteilt werden: (1) Probleme der Quadratur. Dazu gehört der Fluch der Dimensionen und die Wahl einer effizienten Quadraturregel. (2) Probleme auf Grund der Verwendung des statischen Gitters, was sich in der Zahl der nötigen Stützstellen und der Approximationsgüte niederschlägt.

Die im weiteren Verlauf des Kapitels eingeführten Erweiterungen teilen sich ebenfalls in die beiden Bereiche Quadratur und Gitter ein. Konkret zählen dazu:

(1) Verwendung dynamischer Gitter. Dies setzt zum einen bei der Quadratur an, indem das potenziell mehrdimensionale Gitter reskaliert wird und zum anderen muss das dynamische Gitter in den Algorithmus eingebunden werden.

(2) Die Kombinationstechnik ist ein alternativer Ansatz zur Konstruktion mehrdimensionaler Quadraturen aus eindimensionalen Quadraturen. Wobei die Zahl der Stützstellen nicht so schnell mit der Zahl der Dimensionen steigt wie bei der Produktregel.

Das weitere Kapitel gliedert sich wie folgt: In Abschnitt 4.3 wird die Kombinationstechnik vorgestellt. In den Abschnitten 4.4 und 4.5 wird zuerst die Reskalierung von Gittern und

dann die algorithmische Umsetzung dynamischer Gitter beschrieben. Neben der Präsentation der Erweiterung wird in den entsprechenden Abschnitten der Effekt der Erweiterungen und potentielle Schwierigkeiten diskutiert.

4.3. Kombinationstechnik (dünne Gitter)

Wie in den Ausführungen zur numerischen Integration beschrieben (vgl. Abschnitt B im Anhang), wächst die Zahl der Gitterpunkte im vollen Gitter exponentiell mit der Zahl der Dimensionen. Dieser Nachteil kann mit einem alternativen Konstruktionsansatz in Anlehnung an Smolyak (1963) gedämpft werden, sodass die numerische Quadratur auch auf höherdimensionale Probleme angewandt werden kann.

Seit den frühen 1990er Jahren wird dieser Ansatz intensiv beforscht und wird auf immer mehr Anwendungsgebiete übertragen. Im Bereich der Statistik und Ökonometrie finden sich bisher nur wenige Anwendungsbeispiele, unter anderem die von Heiss und Winschel (2008) und Heiss (2008). In anderen Bereichen, wie z.B. den Ingenieurwissenschaften oder dem wissenschaftlichen Rechnen, hat dieser Ansatz bisher eine größere Verbreitung erfahren. Bungartz und Griebel (2004) geben hierzu eine Reihe von Anwendungsbeispielen und diskutieren die Methode ausführlich.

Die Grundidee der dünnen Gitter besteht darin, das relativ feine und rechenaufwendige volle Gitter durch eine Kombination von groben, deutlich weniger rechenaufwendigen Gittern zu ersetzen. Zum einen reduziert das Vorgehen, abhängig von der Anzahl der Dimension, den Rechenaufwand erheblich. Zum anderen wurde gezeigt, dass die Approximationsgüte der dünnen Gitter nur wenig schlechter ist als die des entsprechenden vollen Gitters (Griebel, Schneider und Zenger, 1990).

Ausgangspunkt für die formale Konstruktion sind wiederum eindimensionale Quadraturregeln. Die d-dimensionale Quadratur auf dem dünnen Gitter ergibt sich dann gemäß Gleichung 4.1 als gewichtete Summe von voll-Gitter Quadraturen.

$$Q_k^d f = \sum_{k \leq |\mathbf{l}|_1 \leq k+d-1} \underbrace{(-1)^{k+d-|\mathbf{l}|_1-1} \begin{pmatrix} d-1 \\ |\mathbf{l}|_1 - l \end{pmatrix}}_{\text{Gewicht}} \underbrace{(Q_{l_1}^1 \otimes \ldots \otimes Q_{l_d}^1)}_{\text{volles Gitter}} f \tag{4.1}$$

Hier bezeichnet k das Level des dünnen Gitters, d die Anzahl der Dimensionen. $\mathbf{l}$ ist ein Vektor, der die Level der eindimensionalen Quadraturen enthält, d.h. $\mathbf{l} = (\ l_1 \quad l_2 \quad \ldots \quad l_d\ )$ mit $\mathbf{l} \in \mathbb{N}$. $|\cdot|_1$ steht für l^1-Norm, die sich für den Vektor $\mathbf{l}$ gemäß $|\mathbf{l}|_1 = \sum_{i=1}^{d} l_i$ berechnet. Die Indizierung der Summe ist derart zu lesen, dass für alle Vektoren $\mathbf{l}$, die die Bedingung erfüllen, die entsprechenden Quadraturen gewichtet aufzusummieren sind.

Das Vorgehen lässt sich anhand der Abbildung 4.1 für ein zweidimensionales Integral illustrieren. Für die Quadratur $Q_3^2 f$ erfüllen alle umrandeten Level-Kombinationen die

Bedingung des Summenindex. Die Gewichte in Gleichung 4.1 nehmen entweder den Wert $+1$ oder -1 an und sind ebenfalls in der Abbildung abzulesen. Demzufolge besteht das dünne Gitter aus insgesamt fünf vollen Gittern, von denen drei mit positiven Gewichten (Level-Kombinationen: (1,3), (2,2), (3,1)) und zwei mit negativen (Level-Kombinationen: (1,2), (2,1)) eingehen.

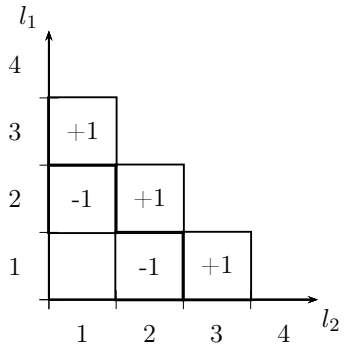

Abbildung 4.1.: Level-Matrix für reguläres 2D-dünnes Gitter

Bisher wurde im Rahmen der Kombinationstechnik das Ergebnis der Quadratur $((Q^1_{l_1} \otimes \ldots \otimes Q^1_{l_d})f)$ gewichtet aufsummiert, d.h. die Approximation des Integrals in Form von skalaren Werten. Für die Umsetzung am Computer ist es jedoch zweckmäßiger, die Stützstellen und Gewichte des dünnen Gitters in einem ersten Schritt zu bestimmen und im zweiten Schritt erst die Funktionswerte an den Stützstellen zu berechnen.

Die Stützstellen des dünnen Gitters ergeben sich als Vereinigungsmenge aller enthaltenen Stützstellen der groben vollen Gitter. Die Gewichte der groben Gitter werden unter Berücksichtigung des positiven bzw. negativen Faktors (+1 bzw. -1) entsprechend aggregiert. Abbildung 4.2 zeigt sowohl die groben vollen Gitter als auch das resultierende dünne Gitter. Die zugrunde liegenden eindimensionalen Quadraturen sind jeweils Gauß-Hermite Quadraturen mit $n = l$ Stützstellen.

Man kann erkennen, dass der Punkt im Ursprung (0,0) sowohl in der Level-Kombination (1,3) als auch (3,1) vorkommt, alle anderen Punkte kommen lediglich einmal vor.

In Abbildung 4.3 ist das dünne Gitter zum Level $l = 3$ für eindimensionale Trapezregeln mit $n = 2^{l-1}$ Stützstellen dargestellt. Wie bereits in Abschnitt B beschrieben, handelt es sich um eine genestete Quadraturregel. Bei der Konstruktion der dünnen Gitter führt es dazu, dass mehrere Stützstellen aufeinander fallen. Bei der Aggregation müssen dann die Gewichte entsprechend zusammengefasst werden. Wie in diesem Beispiel kann dies dazu

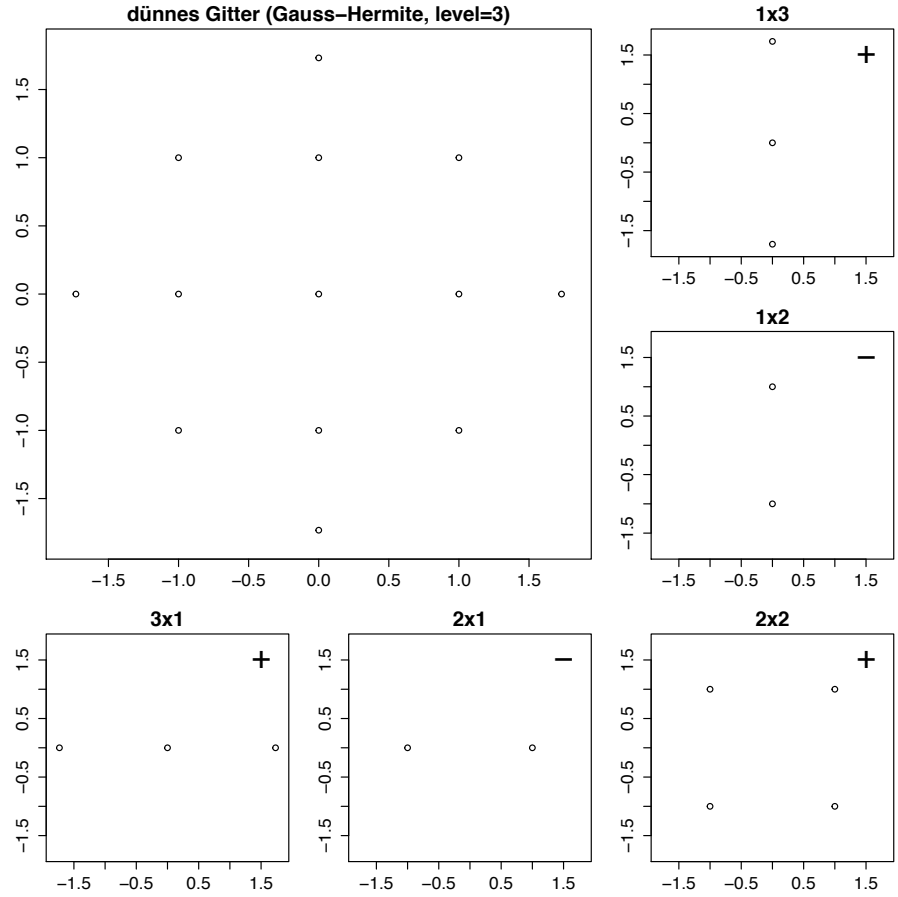

Abbildung 4.2.: Dünnes Gitter und die enthaltenen vollen Gittern (Gauß-Hermite)

führen, dass Stützstellen aufeinander liegen oder sich gegenseitig eliminieren, was beides die Anzahl der Funktionsauswertungen reduziert.

Daraus folgt, dass für praktische Anwendungsfälle genestete Quadraturen bei der Dünn-Gitter-Konstruktion bevorzugt einzusetzen sind, weil bei diesen die Anzahl der Stützstellen mit steigendem Level langsamer ansteigt als bei nicht-genesteten Quadraturen, was zusätzlich den Fluch der Dimensionen dämpft.

In der folgenden Tabelle sind exemplarisch die Anzahl der Gitterpunkte für verschiedendimensionale Gitter mit jeweils vergleichbarer Genauigkeit angegeben. Das Maß für die Genauigkeit ist der polynomiale Exaktheitsgrad (p.E.), d.h. die Ordnung des Polynoms, das gerade noch exakt integriert wird mit dieser Regel. Es ist zu erkennen, dass mit einer steigenden Anzahl von Dimensionen der Vorteil der dünnen Gitter steigt, zudem kann man erkennen, dass mit steigender Genauigkeit der Vorteil der genesteten Quadraturregeln steigt.

Wie in Tabelle 4.1 zu erkennen ist, ist die benötigte Anzahl von Stützstellen im dünnen Gitter geringer als im vollen, wenn man den polynomialen Exaktheitsgrad konstant hält. Im Kontext des sequentiellen bayesschen Filterns werden typischerweise kein Polynome bzw. polynom-ähnliche Funktionen integriert, sondern Funktionen von Verteilungen, wie

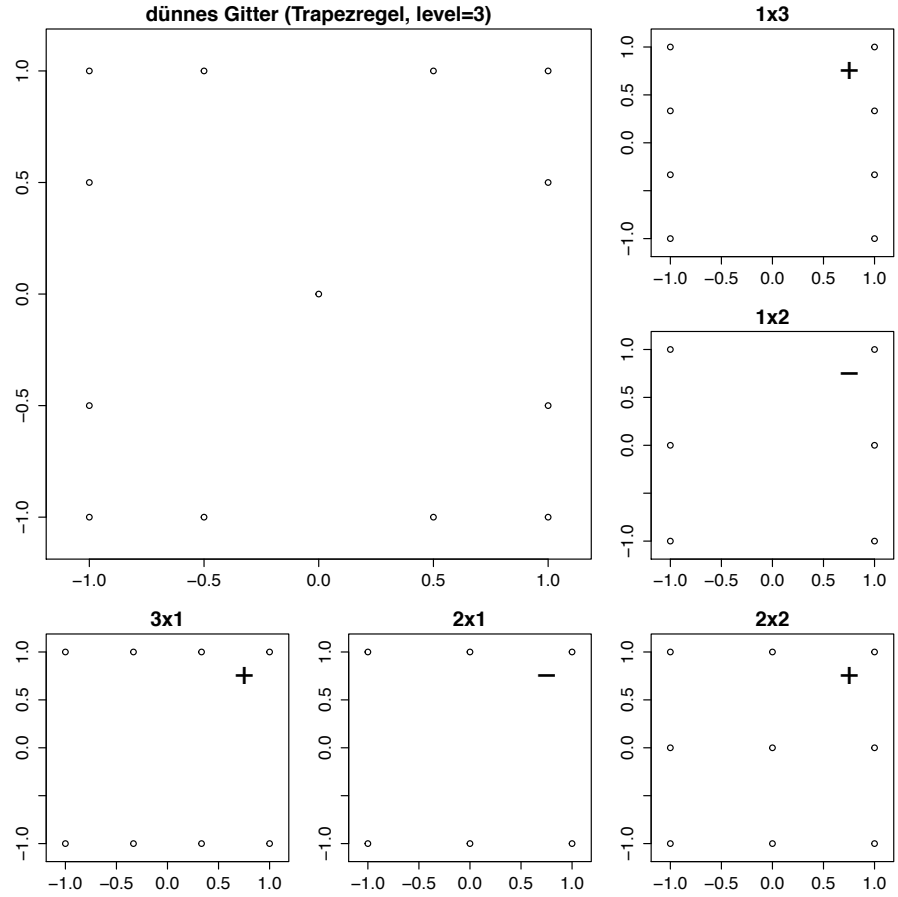

Abbildung 4.3.: Dünnes Gitter und die enthaltenen vollen Gittern (Trapezregel)

sie in den Gleichungen 3.2 und 3.4 zu finden sind.

Aus diesem Grund wird nun, anhand eines typischen Einsatzszenarios, der Effekt der dünnen Gitter auf die Effizienz der Approximation untersucht. Hierfür wird analog zu Gleichung 3.4 eine 5-dimensionale A-Posterori-Verteilung mit vollen und dünnen GHe-Gittern integriert[2]. Zudem wird analog zu Gleichung 3.2 eine Faltung zweier 5-dimensionaler Standard-Normalverteilungen berechnet.

Abbildung 4.4 fast das Resultat der Untersuchung zusammen. Im linken Graphen ist das konkrete Konvergenzverhalten der Approximation eines Integrals analog zu Gleichung 3.4 dargestellt. Das Bild stützt die Erwartungen gemäß Tabelle 4.1, d.h. bei gleicher Approximationsgüte benötigt das dünne Gitter merklich weniger Stützstellen als die Approximation mit einem vollen Gitter.

Anders verhält es sich im rechten Graphen. Bei der Faltung schneidet das dünne Gitter bis zu einem gewissen Punkt merklich schlechter ab als das volle Gitter. Der genauen Klärung des Phänomens wird an dieser Stelle nicht weiter nachgegangen[3].

[2]Im folgenden Abschnitt wird das Untersuchungsdesign detailiert beschrieben

[3]In Gesprächen mit Herrn Prof. Dr. Thomas Gerstner (Goethe-Universität Frankfurt am Main), Herrn Prof. Dr. Michael Griebel (Universität Bonn) und Herrn Prof. Dr. Christoph Zenger (Technische

	volle Gitter		dünne Gitter		
Dim.	**Trapez-regel**	**Gauß-Hermite**	**genestete Trapez-regel**	**Gauß-Hermite**	**genestete Gauß-Hermite**
p.E. ≈ 3					
d=1	4	2	-	-	-
d=3	64	8	50	7	7
d=5	1 024	32	352	11	11
d=10	1 048 576	1 024	27 904	21	21
p.E. ≈ 5					
d=1	6	3	-	-	-
d=3	216	27	123	25	19
d=5	7 776	243	1 032	61	51
d=10	60 466 176	59 049	109 824	221	201
p.E. ≈ 15					
d=1	16	8	-	-	-
d=3	4 096	512	297	1 233	381
d=5	1 048 576	32 768	2 882	13 073	3 793
d=10	1 099 511 627 776	1 073 741 824	394 624	581 385	185 085

Tabelle 4.1.: Anzahl der Stützstellen

Da bei der sequentiellen numerischen Integration beide Formen von Integralen Anwendung finden, bleibt zu überprüfen, welches Konvergenzverhalten sich bei der sNI-Methode ergibt. Dieser Frage wird unter anderem im kommenden Kapitel nachgegangen.

4.4. Reskalierung mehrdimensionaler Gitter

Bei mehrdimensionalen Quadraturen kann die Genauigkeit der Approximation erhöht werden, indem man mit den Stützstellen den relevanten Bereich möglichst passend abdeckt. Dies gilt unabhängig davon, ob nun volle oder dünne Gitter verwendet werden.

Abbildung 4.5 zeigt zwei schlecht angepasste Gitter. Im linken Beispiel ist die durch Konturlinien angedeutete Dichte nicht vollständig abgedeckt. In diesem Fall ist die Güte der Approximation steigerbar, indem man das Gitter verschiebt.

Im rechten Beispiel ist zwar die Dichte vollständig abgedeckt, allerdings ist eine große Anzahl an Gitterpunkten in einer Region, in der die zu integrierende Funktion sehr kleine

Universität München) konkretisierte sich die Vermutung, dass das Prinzip der dünnen Gitter nicht mit dem Konzept der Faltung kompatibel ist. Vorerst wird es jedoch eine offene Forschungsfrage bleiben.

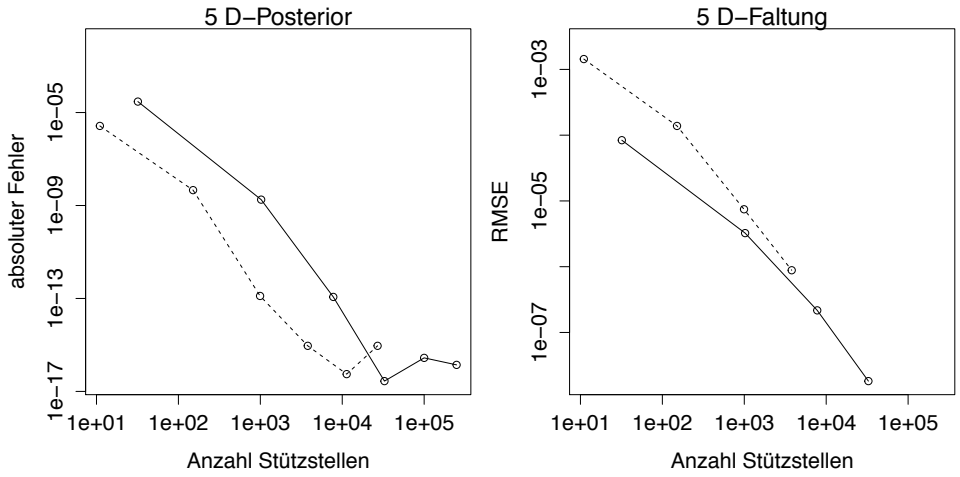

Abbildung 4.4.: Effekt dünner Gitter. *Volle Gitter (durchgezogene Linie) und dünne Gitter (gestrichelte Linie) bei der Approximation 5-dimensionaler Integrale. Links: Integration der A-Posterori-Verteilung, rechts: Faltung zweier 5D-Standard-Normalverteilungen*

Werte (≈ 0) annimmt. Der Rechenaufwand kann erheblich reduziert werden, indem dass Gitter auf den relevanten Bereich beschränkt wird.

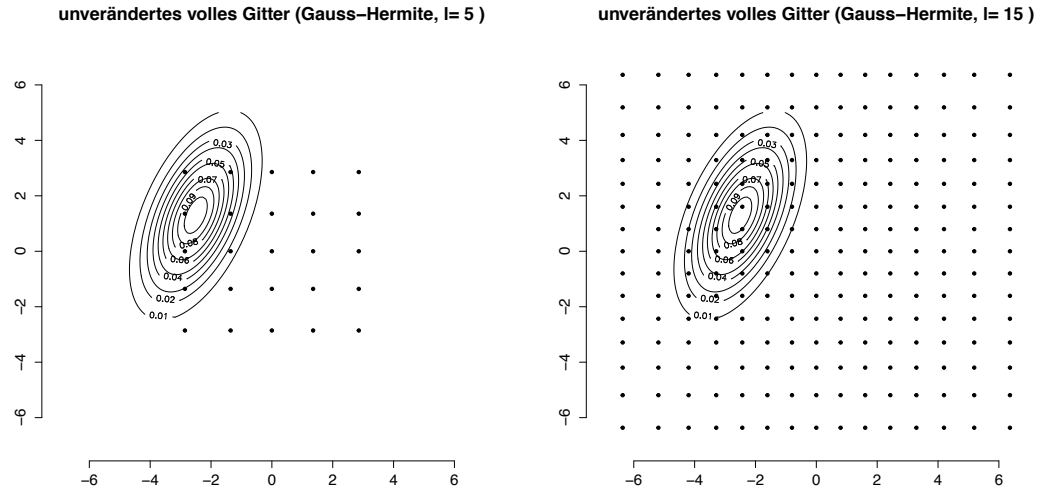

Abbildung 4.5.: Schlecht angepasste Gitter

In Anlehnung an Jäckel (2005) wird das Gitter verschoben und reskaliert, um die Effizienz der Quadratur zu steigern. Im Folgenden werden drei unterschiedliche Ansätze für das Reskalieren vorgestellt. Sie ergeben sich als mehrdimensionale Erweiterung des Reskalierungs-Ansatzes in Abschnitt B (Anhang).

(1) Im linken Graphen der Abbildung 4.6 ist der einfachste Ansatz abgebildet. Hier wurde lediglich das Gitter so verschoben, dass der Mittelpunkt des Gitters auf dem Schwerpunkt der zu integrierenden Funktion liegt. Zudem wurde das Gitter so gestreckt/gestaucht,

dass es den relevanten Bereich abdeckt. Hierzu wurden die Koordinaten der $i = 1, ..., n$ Stützstellen in jeder Dimension ($j = 1, ..., d$) wie folgt manipuliert:

$$\begin{aligned} \tilde{x}_{1,i} &= (x_{1,i} \cdot \sigma_{11}) + m_1 \\ &\dots \\ \tilde{x}_{d,i} &= (x_{d,i} \cdot \sigma_{dd}) + m_d \end{aligned}$$

$$\tilde{w}_i = w_i \cdot \Pi_{j=1}^{d} \sigma_{jj}$$

Hier bezeichnet m_j den Mittelwert in der j-ten Dimension und σ_{jj} die entsprechende Standardabweichung, die als Diagonalelement der Kovarianzmatrix entnommen werden kann.

Diese Art der Reskalierung ist auch für eindimensionale Quadraturen geeignet. In diesem Fall entspricht das Vorgehen dem im Anhang beschriebenen Reskalieren der Gauss-Hermite-Quadratur nach Liu und Pierce (1994).

Im zweiten und dritten Ansatz wird nicht nur das Gitter gestaucht bzw. gestreckt, sondern auch entsprechend der Ausrichtung gedreht (mittlerer Graph in Abbildung 4.6) bzw. geschert (rechter Graph). Dies steigert nochmals die Effizienz, indem zum einen die Abdeckung verbessert wird. So werden nach der Drehung bzw. Scherung nicht mehr die unbedeutenden Stützstellen links oben und recht unten (linker Graph) ausgewertet. Zum anderen wird der Zustandsraum durch die Transformation orthogonal, d.h. die Kovarianzmatrix des transformierten Gitters entspricht der Einheitsmatrix (I) und somit am ehesten der multivariaten Standardnormalverteilung, für die diese Quadratur optimal ist.

(2) Im mittleren Graphen der Abbildung 4.6 wurde ebenfalls das Gitter verschoben. Zudem wurde es gedreht. Hierfür wurden die Koordinaten wie folgt manipuliert:

$$\begin{pmatrix} \tilde{x}_{1,i} \\ \vdots \\ \tilde{x}_{d,i} \end{pmatrix} = \begin{pmatrix} x_{1,i} \\ \vdots \\ x_{d,i} \end{pmatrix} \cdot \mathbf{A} + \begin{pmatrix} m_1 \\ \vdots \\ m_2 \end{pmatrix}$$

$$\text{mit } \mathbf{A} = \mathbf{S} \cdot I\sqrt{\mathbf{\Lambda}}$$

$$\tilde{w}_i = w_i \cdot \Pi_{j=1}^{d} \sqrt{\Lambda_{jj}}$$

Wobei $\mathbf{S}$ das Eigensystem der Kovarianzmatrix der zu integrierenden Funktion und $I\sqrt{\mathbf{\Lambda}}$ eine Diagonalmatrix ist mit den Quadratwurzeln der Eigenwerte auf der Hauptdiagonalen.

Es gilt bei der Eigenwertzerlegung $\mathbf{C} = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^{'}$, wobei $\mathbf{C}$ die Kovarianzmatrix der zu integrierenden Dichtefunktion bezeichnet.

(3) Im rechten Graphen wurde eine ähnliche Transformation durchgeführt. Jedoch wird hier die Matrix $\mathbf{A}$ durch die Matrix $\mathbf{B}$ ersetzt, die sich aus der Cholesky-Zerlegung der Kovarianzmatrix $\mathbf{C}$ ergibt.

$$\begin{pmatrix} \tilde{x}_{1,i} \\ \vdots \\ \tilde{x}_{d,i} \end{pmatrix} = \begin{pmatrix} x_{1,i} \\ \vdots \\ x_{d,i} \end{pmatrix} \cdot \mathbf{B} + \begin{pmatrix} m_1 \\ \vdots \\ m_2 \end{pmatrix}$$

$$\text{es soll gelten } \mathbf{C} = \mathbf{B} \cdot \mathbf{B}^{'}$$

$$\tilde{w}_i = w_i \cdot \Pi_{j=1}^{d} \sqrt{\mathbf{B}_{jj}}$$

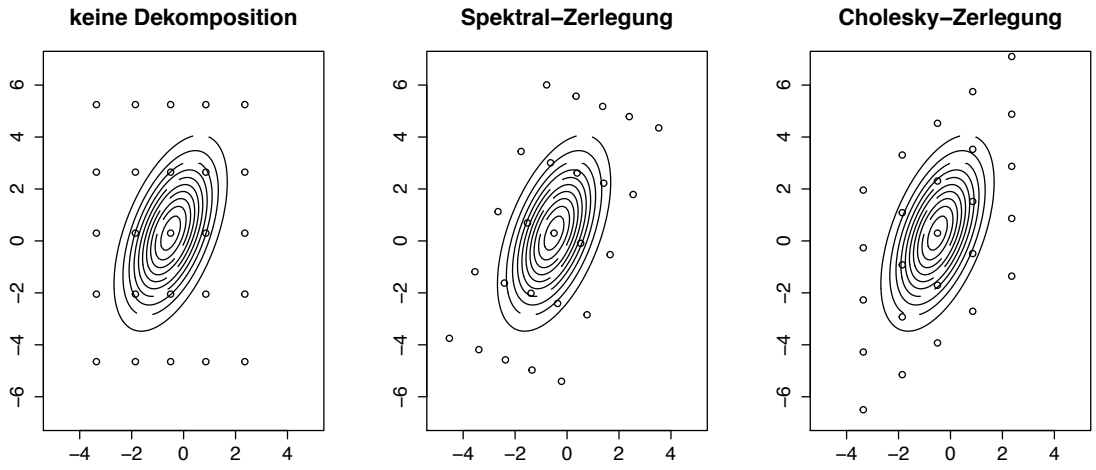

Abbildung 4.6.: Methoden zur Reskalierung (Illustration)

Wird in der Anwendung eine multivariate Normalverteilung integriert, sollte Variante 2 gewählt werden, da diese zu Stützstellen und Gewichten führt, die die Dichte einer Normalverteilung exakt integrieren. Für allgemeine Anwendungen kann ex ante keine der Varianten (2) und (3) als generell überlegen herausgestellt werden. Aus Effizienzüberlegungen sollte jedoch Variante (1) vermieden werden. Allerdings ist die Manipulation gemäß Variante (1) am einfachsten und am wenigsten rechenaufwendig.

Aus dem Vergleich der Abbildungen 4.5 und 4.6 kann schon intuitiv auf eine erhöhte Effizienz bei der Anwendung der Reskalierung geschlossen werden. Die Abbildung 4.7 zeigt an einem konkreten Integrationsproblem die Steigerung der Effizienz durch die Nutzung der Reskalierung vom Type 2 (Spektral-Zerlegung).

Das zu berechnende d-dimensionale Integral ist gegeben durch:

$$\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \phi^d((x_1, \ldots, x_d), 0, 1) * \phi^1((-\sum_{i=1}^{d} x_i), 0, 1)\, dx_1 \ldots dx_d$$

Hier bezeichnet $\phi^x((z), \mu, \sigma^2)$ die Dichte einer x-dimensionalen Normalverteilung mit Erwartungswert μ und Varianz σ^2 an der Stelle z. Das Integral ist in Anlehnung an den Filterschritt (vgl. 3.4) formuliert. Wobei $\phi^d(\cdot)$ die A-Priori-Verteilung bezeichnet und $\phi^1(\cdot)$ die Likelihoodfunktion darstellt. Der wahre Wert des Integrals ist ϕ^1(0,0,d+1).

Für diesen Vergleich wurde das d-dimensionale Integral für $d = 1, ..., 5$ mit einem vollen bzw. dünnen Gitter für verschiedene Level l approximiert. Als zugrundeliegende eindimensionale Quadraturregel wurde die Gauß-Hermite-Quadratur verwendet. Die für die Reskalierung notwendigen Parameter m und C wurden durch eine Monte-Carlo-Simulation bestimmt und als Gütemaß wird der absolute Fehler der Approximation ausgewiesen.

Qualitativ lassen sich aus Abbildung 4.7 unter anderem drei Folgerungen ableiten:

(1) Das Ausgangsniveau des Approximationsfehlers sinkt durch die Reskalierung.

(2) Die Konvergenzgeschwindigkeit steigt mit Verwendung reskalierter Gitter.

(3) Das dünne Gitter ist nur sinnvoll in Kombination mit der Reskalierung einzusetzen, denn ein nicht reskaliertes dünnes Gitter hat schlechtere Approximationseigenschaften als die entsprechenden vollen Gitter.

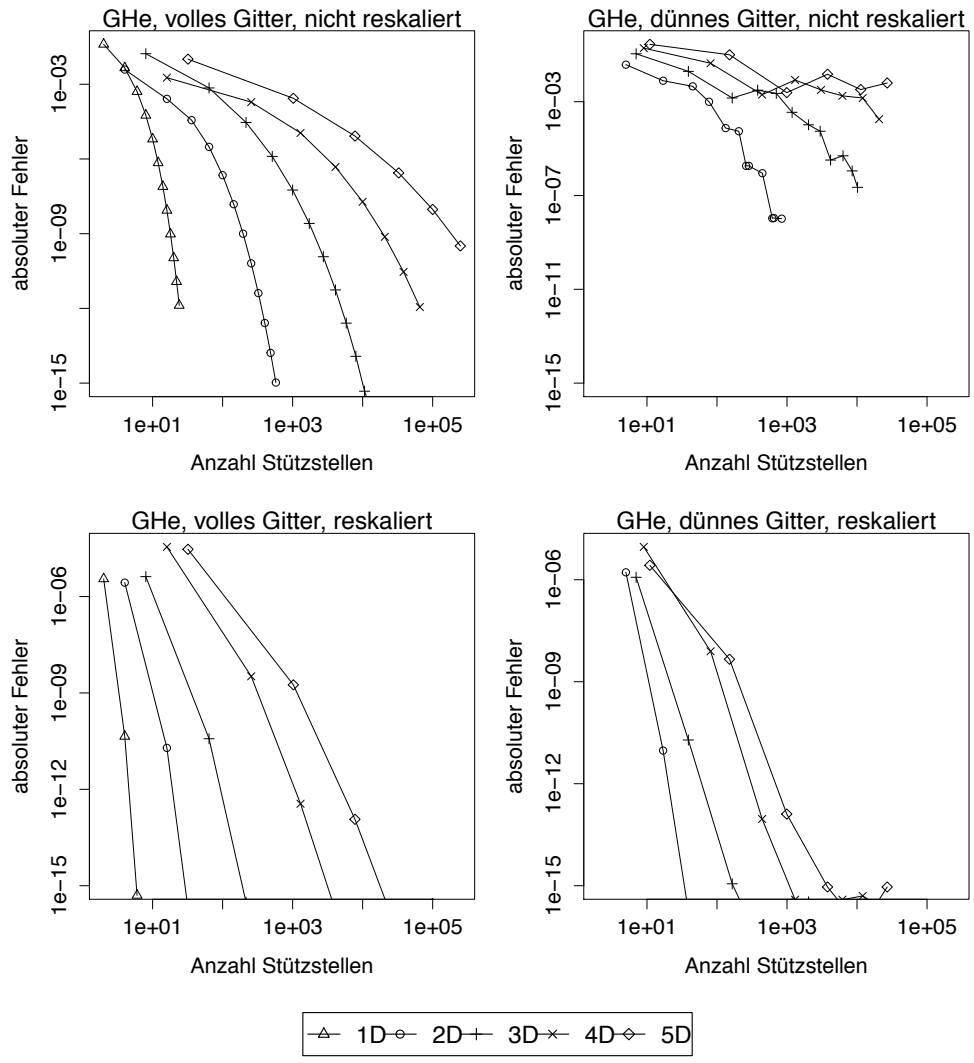

Abbildung 4.7.: Vergleich nicht-reskalierter und reskalierter Gitter

4.5. Dynamisches Gitter

Ein Teilziel der Arbeit ist die Implementierung eines dynamischen Gitters in den Algorithmus der sequentiellen numerischen Integration. Als dynamisch wird ein Gitter bezeichnet, dass sich im Zeitablauf selbstständig an die zu integrierende Dichte so anpasst, dass diese möglichst effizient integriert wird.

Wie in Abbildung 3.4 sowie den Gleichungen 3.2 und 3.3 beschrieben, ergibt sich der rekursive Charakter des bayesschen Filterns durch die Abfolge von Vorhersage- und Filterschritten. Durch diese Rekursion ist es nur zwischen einem Filterschritt (Zeitpunkt t) und dem direkt darauf folgenden Vorhersageschritt (zum Zeitpunkt t für $t+1$) möglich, das Gitter zu verändern.

Währenddessen ist zwischen Vorhersageschritt (Zeitpunkt t) und Filterschritt (Zeitpunkt $t+1$) keine Anpassung am Gitter möglich, da für die Evaluation der A-Posterori-Verteilung

an den Stützstellen die A-Priori-Dichte für die selben Stützstellen benötigt wird.

Zudem ist bei den Gleichungen 3.2 und 3.3 zu erkennen, dass die entsprechenden Integrale sich stets auf die (normalisierte bzw. nicht-normalisierte) A-Posterori-Verteilungen (analog zu f_{t-1} und $\hat{f}_t$) beziehen und somit das dynamische Gitter sich an dieser ausrichten sollte.

Für die Manipulation des Gitters sei auf die vorgestellten Methoden aus Abschnitt 4.4 verwiesen.

Abbildung 4.8[4] zeigt den angepassten Algorithmus für das dynamische Gitter. Für die praktische Umsetzung ergibt sich nun noch die Schwierigkeit, wie das Gitter für die A-Posterori-Verteilung angepasst werden kann, bevor diese Verteilung überhaupt bekannt ist. In den folgenden beiden Abschnitten werden zwei Lösungsansätze vorgestellt, der Hilfsfilter und das iterative Verfahren.

1. speichere initiale Stütztstellen und Gewichte optimiert
für Integration einer Dichte mit $\mu = 0$ und $\sigma = 1$
$\rightarrow (x_i, w_i) \, \forall i = 1, ..n$
2a. reskaliere die initialen Stützstellen und Gewichte

für die Dichte f_1 : $\quad (x_i, w_i) \rightarrow (\tilde{x}_0^{(i)}, \tilde{w}_0^{(i)}) \, \forall i = 1, ..., n$

2b. evaluiere die A-Priori-Dichte an den Stützstellen

$f_0^{(i)} = p_0(\tilde{x}_0^{(i)}) \quad \forall i = 1, ..., n$

3. für alle $t = 1, ..., T$:

(a) reskaliere die initialen Stützstellen und Gewichte
für die Dichte f_t : $\quad (x_i, w_i) \rightarrow (\tilde{x}_t^{(i)}, \tilde{w}_t^{(i)}) \, \forall i = 1, ..., n$
(b) berechne die Dichte der Vorhersagewahrscheinlichkeit
für alle $j = 1, ..., n$: $p_t^{(j)} = \sum_{i=1}^{n} (\tilde{w}_{t-1}^{(i)} \cdot p(\tilde{x}_t^{(j)} | \tilde{x}_{t-1}^{(i)}) \cdot f_{t-1}^{(i)})$
(c) berechne die A-Posterori-Dichte
für alle $j = 1, ..., n$: $\hat{f}_t^{(j)} = p_t^{(j)} \cdot p(y_t | \tilde{x}_t^{(j)})$
berechne die Normalisierungskonstanten
$C = \sum_{i=1}^{n} (\tilde{w}_t^{(i)} \cdot \hat{f}_t^{(i)})$
für alle $j = 1, ..., n$: $f_t^{(j)} = \frac{\hat{f}_t^{(j)}}{C}$
(d) Inferenz anhand der A-Posterori-Dichte (optional)

Abbildung 4.8.: Algorithmus für sNI mit dynamischen Gitter

Um das Prinzip der dynamischen Gitter zu illustrieren, gehen wir an dieser Stelle davon aus, dass wir ex ante die zu integrierenden A-Posterori-Verteilungen kennen. Abbildung

[4]Notation: $(\tilde{x}_t^{(i)}, \tilde{w}_t^{(i)})$ Stützstellen bzw. Gewichte mit einer Tilde „$\sim$“ überschrieben sind reskaliert, das Superskript (z.B. (i)) kennzeichnet die Stützstelle, das Subskript (t) den Zeitpunkt, für den die Stützstelle angepasst wurde

4.9 zeigt das Prinzip. Es ist ersichtlich, dass durch die Anwendung der dynamischen Gitter viele der besprochenen Unzulänglichkeiten des statischen Gitters beseitigt werden. So passt das dynamische Gitter seine Lage und Ausdehnung an, was das Problem des Abschneidens eliminiert und durch das Reskalieren der Gitter wird die zu integrierende Funktion stets adäquat abgebildet; zudem entfällt die Evaluation nicht-relevanter Stützstellen.

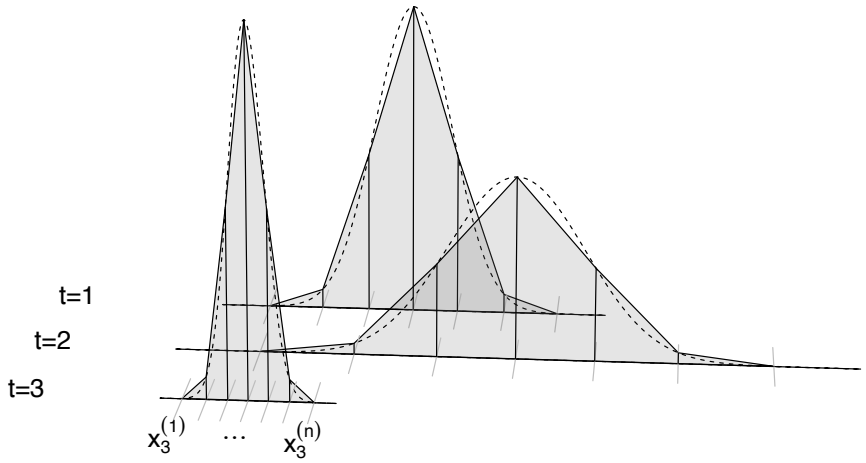

Abbildung 4.9.: Illustration dynamisches Gitter. *Dargestellt sind die wahren nicht normalisierten A-Posterori-Dichten (gestrichelte Linie) und die Approximation der Normalisierungskonstante durch die Trapezregel (grau schraffierte Flächen) für die Zeitpunkte t=1, 2 und 3. Die Stützstellen werden zu jedem Zeitpunkt so angepasst, dass sie optimal die entsprechende Dichte approximieren.*

4.5.1. Hilfsfilter

Der kritische Punkt der dynamischen Gitter besteht darin, dass man ex ante nicht weiß, wie das Gitter zu reskalieren ist, um in der Folge die A-Posterori-Verteilung gut abzubilden. Die Unkenntnis bzgl. der relevanten Region im Zustandsraum ist ein inhärentes Problem der sequentiellen numerischen Integration.

Andere Filter wie z.B. die in Kapitel 3.1 vorgestellten Partikel-Filter haben dieses Problem nicht. Diese ziehen ihre Partikel per Konstruktion aus der relevanten Region. Auch analytische Filter, wie der Kalman-Filter oder für nicht-lineare/nicht normale Zustandsraummodelle nutzbare „unscented Kalman-Filter“, kennen das Problem der „relevanten Region“ nicht.

Ein erster Vorschlag ist daher, einen der genannten Filter hilfsweise zu nutzen, um die relevante Region für die sNI abzuschätzen. Damit können die positiven Eigenschaften sowohl der sNI als auch die des Hilfsfilters genutzt werden.

Beispielsweise kann ein einfacher Bootstrap Filter genutzt werden mit einer relativ geringen Anzahl von Partikeln. Ein solcher Bootstrap Filter ist in der Regel leicht zu implementieren und der Rechenaufwand ist relativ gering. Jedoch sind die Ergebnisse relativ ungenau. Zudem kann dieser Filter nicht zusammen mit der Maximum-Likelihood-Methode genutzt werden (Malik und Pitt, 2011). Durch die Kombination mit dem sNI kann die Genauigkeit stark erhöht werden und die Maximum-Likelihood-Methode wird dadurch anwendbar.

Bei der rechentechnischen Umsetzung wird stets zuerst die A-Posterori-Verteilung mit dem Hilfsfilter berechnet, um die nötigen Informationen für die Reskalierung im sNI-Algorithmus zur Verfügung zu stellen. Abbildung 4.10 zeigt die entsprechenden Anpassungen im Algorithmus.

1. ...
2a-1.berechne mit dem Hilfsfilter die

A-Posterori-Dichte f_1 : $\rightarrow \mu_{1,Hilf}$, $\sigma^2_{1,Hilf}$

2a-2. reskaliere die initialen Stützstellen und Gewichte

für die Dichte f_1 gemäß $\mu_{1,Hilf}$, $\sigma^2_{1,Hilf}$: $(x_i, w_i) \rightarrow (\tilde{x}_0^{(i)}, \tilde{w}_0^{(i)}) \, \forall i = 1, ..., n$

2b. ...
3. für alle $t = 1, ..., T$:

(a-1) berechne mit dem Hilfsfilter die
A-Posterori-Dichte f_t : $\rightarrow \mu_{t,Hilf}$, $\sigma^2_{t,Hilf}$
(a-2) reskaliere die initialen Stützstellen und Gewichte
für die Dichte f_t gemäß $\mu_{t,Hilf}$, $\sigma^2_{t,Hilf}$: $(x_i, w_i) \rightarrow$
$(\tilde{x}_t^{(i)}, \tilde{w}_t^{(i)}) \, \forall i = 1, ..., n$
(b) - (d)...

Abbildung 4.10.: Algorithmus für sNI mit dynamischem Gitter und Hilfsfilter

4.5.2. Iteratives Verfahren

Während das Vorgehen mit dem Hilfsfilter verschiedene Filtertypen miteinander verbindet, basiert das nun vorgeschlagene iterative Verfahren nur auf der sNI. Das vorgeschlagene Vorgehen ist an Smith u. a. (1987, Abschnitt 2: An iterative quadrature strategy) angelehnt und umfasst die folgenden Arbeitsschritte für die Zeitpunkte $t = 2, ...T$.

(1) Die A-Posterori-Verteilung vom Zeitpunkt $t - 1$ wird gemäß der Zustandsgleichung fortgeschrieben und das Gitter wird für diese Vorhersagedichte angepasst. (2) Mit diesem Gitter wird eine vorläufige Berechnung der A-Posterori-Verteilung durchgeführt und deren Mittelwert und Varianz bestimmt. (3) Das Gitter wird entsprechend des berechneten

Mittelwerts und der Varianz reskaliert. Schritt (2) und (3) werden so oft ausgeführt, bis sich ein stabiles Ergebnis ergibt.

Für den Zeitpunkt $t = 1$ kann sich das Verfahren auf die A-Priori-Verteilung beziehen, es können aber auch andere Startwerte gewählt werden. So kann auch hier ein Hilfsfilter genutzt werden.

Generell ist zu beachten, dass bei diesem Verfahren, wie bei jeder Suchheuristik, nicht in allen Fällen das Auffinden der richtigen Lösung garantiert ist. Insbesondere in den Fällen, bei denen sich die Vorhersagedichte stark von der A-Posteriori-Verteilung unterscheidet, können sich Probleme bei der Berechnung ergeben. Dies ist regelmäßig der Fall, wenn die Varianz des Rauschterms sehr klein ist relativ zur Varianz des Innovationsterms.

Sind solche Probleme nicht zu erwarten, so führt im Allgemeinen dieses Vorgehen zu den genauesten Ergebnissen, wie später in den Simulationsstudien zu sehen sein wird.

Der Algorithmus für das iterative Verfahren ist in Abbildung 4.11 zusammengefasst.

1. speichere initiale Stützstellen und Gewichte optimiert
für Integration einer Dichte mit $\mu = 0$ und $\sigma = 1$
$\rightarrow (x_i, w_i) \, \forall i = 1, ..n$
2a. reskaliere die initialen Stützstellen und Gewichte

für die A-Priori Dichte f_0 : $\quad (x_i, w_i) \rightarrow (\tilde{x}_0^{(i)}, \tilde{w}_0^{(i)}) \, \forall i = 1, ..., n$

2b. evaluiere die A-Priori Dichte an den Stützstellen

$f_0^{(i)} = p_0(\tilde{x}_0^{(i)}) \quad \forall i = 1, ..., n$

3. für alle $t = 1, ..., T$:

$iter = 0$ (Anzahl der Iterationen)
wiederhole die Schritte (a)-(d),
bis ein Abbruchkriterium erfüllt ist
(a)

$iter = iter + 1$ (zählen der Iterationen)
falls $iter = 1$
reskaliere die initialen Stützstellen und Gewichte
für die Vorhersagedichte μ_{t-1}, σ_{t-1}^2: $(x_i, w_i) \rightarrow$
$(\tilde{x}_t^{(i)}, \tilde{w}_t^{(i)}) \, \forall i = 1, ..., n$
falls $iter > 1$
reskaliere die initialen Stützstellen und Gewichte
gemäß der vorläufigen A-Posterori Verteilung
μ_t, σ_t^2: $(x_i, w_i) \rightarrow (\tilde{x}_t^{(i)}, \tilde{w}_t^{(i)}) \, \forall i = 1, ..., n$

(b) berechne die Dichte der Vorhersagewahrscheinlichkeit:
für alle $j = 1, ..., n$: $p_t^{(j)} = \sum_{i=1}^{n} (\tilde{w}_{t-1}^{(i)} \cdot p(\tilde{x}_t^{(j)} | \tilde{x}_{t-1}^{(i)}) \cdot f_{t-1}^{(i)})$
(c) berechne die A-Posterori Dichte
für alle $j = 1, ..., n$: $\hat{f}_t^{(j)} = p_t^{(j)} \cdot p(y_t | \tilde{x}_t^{(j)})$
berechne die Normalisierungskonstanten:
$C = \sum_{i=1}^{n} (\tilde{w}_t^{(i)} \cdot \hat{f}_t^{(i)})$
für alle $j = 1, ..., n$: $f_t^{(j)} = \frac{\hat{f}_t^{(j)}}{C}$
(d) berechne μ_t und σ_t^2 der A-Posterori Verteilung

führe keine weitere Iteration durch falls:
- die Approximation der A-Posterori Verteilung sich nicht signifikant verändert
- die maximale Anzahl an Iterationen erreicht ist

Abbildung 4.11.: Algorithmus für sNI mit iterativen dynamischen Gitter.

4.5.3. weitere Varianten

Neben den zwei explizit vorgestellten Ausgestaltungsformen des dynamischen Gitters sind auch weitere Varianten denkbar, die an dieser Stelle nur schemenhaft skizziert werden sollen. Diese Auflistung kann als Ausgangspunkt für weitere Forschungsarbeiten in diesem Bereich dienen.

Mischform

Um die iterative Suche nach einem möglichst passenden Gitter zu beschleunigen, kann man als Ausgangspunkt der Suche statt der Vorhersagedichte die A-Posterori-Dichte eines Hilfsfilters nutzen.

Automatische Quadratur

Das iterative Verfahren kann auch um eine schrittweise Verfeinerung des Gitters erweitert werden. Somit lässt sich die Suche nach der optimalen Reskalierung mit der Suche nach der optimalen Anzahl von Stützstellen kombinieren. Bei der Implementierung würde man für einen Zeitpunkt (t) die Zahl der Stützstellen stetig verfeinern. Beim Wechsel zum nächsten Zeitpunkt $(t+1)$ müsste man die Zahl der Stützstellen zuerst wieder reduzieren, um dann die Verfeinerung erneut zu starten.

Um die Effizienz des Verfahrens zu steigern, sollten je nach Ausgestaltung genestete Quadraturregeln genutzt werden.

Adaptive Gitter

Die bisher diskutieren Ansätze verwenden stets sogenannte reguläre Gitter. Reguläre Gitter sind Gitter, die vollständig durch eine Quadraturregel und ein Genauigkeitslevel definiert sind. Sie zeichnen sich in der Regel durch (1) eine gewisse Regelmäßigkeit und (2) dadurch aus, dass sie „unabhängig“ von der zu integrierenden Funktion erzeugt werden.

Einen anderen Ansatz verfolgen adaptive Gitter, die mit dem Konzept der hierarchischen Basen eng verbunden sind[5]. Beim adaptiven Gitter werden die Stützstellen mit Blick auf die Funktion gewählt. Abbildung 4.12 illustriert den Unterschied zwischen einem adaptiven und einem regulären Gitter, wobei die Zahl der Stützstellen so gewählt wurde, dass der Approximationsfehler für beide Quadraturen gleich ist.

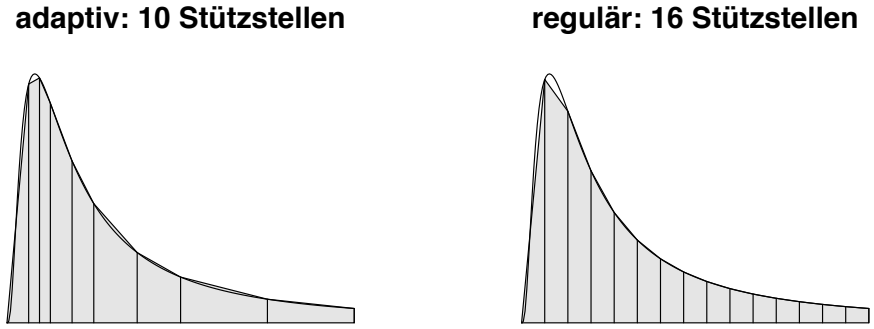

Abbildung 4.12.: Adaptives (links) und reguläres Gitter (rechts).

[5]Eine ausführliche Beschreibung liefert z.B. Pflüger (2010).

Im Kontext der sNI kann dieses Vorgehen genutzt werden, um die iterative Suche abzubilden. So kann man mit einem groben Gitter, das die Vorhersagedichte abbildet, durch adaptive Verfeinerung sich der Approximation der A-Posterori-Verteilung annähern. Dieser Ansatz hat einen gewissen Charme, da bei der mehrdimensionalen Erweiterung per Konstruktion sich die effizienten dünnen Gitter ergeben und zugleich eine automatische Quadratur umsetzbar ist.

Auf Grund der Komplexität wird der Ansatz an dieser Stelle nicht weiter verfolgt.

5. Vergleich von sMC und sNI

In diesem Kapitel werden die beiden Approximationsverfahren (sMC und sNI) im Rahmen einer Simulationsstudie verglichen. Ziel ist, die beiden Methoden bzgl. ihrer Effizienz zu charakterisieren und den Effekt der Weiterentwicklungen aus dem vorherigen Kapitel zu bewerten.

Das Kapitel ist wie folgt aufgebaut: In Abschnitt 5.1 wird das allgemeine Untersuchungsdesign vorgestellt. In den darauf folgenden Abschnitten werden die Simulationsergebnisse ausgewertet, wobei in Abschnitt 5.2 die Studie mit Blick auf das Konvergenzverhalten ausgewertet wird.

5.1. Untersuchungsdesign

Für Approximationsverfahren wie die hier diskutierten sMC-Methode oder sNI ist es von praktischer Bedeutung, deren Eigenschaften zu kennen, um deren Einsatzfähigkeit abzuschätzen. Zu den relevanten Eigenschaften gehören Konvergenzverhalten, Fehlerabschätzung und der mit dem Verfahren verbundene Rechenaufwand. Alle diese Eigenschaften lassen sich in einer Simulationsstudie untersuchen.

Für die folgende Studie wurden wiederholt künstliche Zeitreihen generiert, deren Hyperparameter man vorgibt. Diese Zeitreihen werden dann mit den Approximationsverfahren gefiltert und die gefilterten Zeitreihen mit einer Referenzzeitreihe verglichen. Im Idealfall sollte die Referenzzeitreihe die korrekten Werte annehmen, wobei sich zwei Herangehensweisen dazu eignen: (1) Man wählt in der Simulationsstudie ein Zustandsraummodell, für das eine analytische Lösung existiert oder (2) man nimmt ersatzweise das Ergebnis einer sehr genauen Approximation, z.B. eines Partikel-Filters mit extrem hoher Zahl von Partikeln oder einer Quadratur mit sehr hoher Zahl von Stützstellen.

Im Rahmen dieser Simulationsstudie wurden intensiv die Konvergenzeigenschaften für lineare und normalverteilte Zustandsraummodelle untersucht, für die es eine analytische Lösung gibt. Gerade weil es eine analytische Lösung gibt, sind in diesem Fall die Approximationsverfahren jedoch nicht von praktischer Bedeutung. Darüber hinaus wurden auch Modelle ohne analytische Lösung untersucht, um die Ergebnisse, welche für die inearen und normalverteilten Modelle gelten, zu stützen.

Als Maß der Approximationsgüte wurde der „Root-Mean-Squared-Error“ (RMSE) gewählt, der sich gemäß Gleichung 5.1 berechnet.

$$RMSE = \sqrt{\frac{1}{D \cdot T} \sum_{d=1}^{D} \left(\sum_{t=1}^{T} (E(x_{d,t}) - E(\tilde{x}_{d,t}))^2 \right)} \tag{5.1}$$

Hier bezeichnet D die Anzahl der Dimensionen im Zustandsraum, T die Länge der Zeitreihe und $x_{d,t}$ bzw. $\tilde{x}_{d,t}$ den wahren bzw. den approximativ berechneten Erwartungswert des Zustands in Dimension d zum Zeitpunkt t.

Um dem stochastischen Charakter der Studie, welcher durch das zufällig Erzeugen der Zeitreihen verursacht wird, zu verringern, wurden in jeder Gruppe mehrere Zeitreihen erzeugt und gefiltert. Als Ergebnis wurde dann der gemittelte $RMSE$ ausgewiesen.

Ebenso stochastisch ist das Ergebnis der Partikel-Filter. Um auch diesem zu begegnen, wurden innerhalb jeder Modellgruppe viele Male die Zeitreihe mit den Partikel-Filtern gefiltert und das mittlere Ergebnis ausgewiesen.

Es wurden ein- bis vierdimensionale lineare und normalverteilte Zustandsraummodelle untersucht. Darüber hinaus eindimensionale „nicht lineare“ sowie „nicht-lineare und nicht normalveteilte“ Zustandsraummodelle. Bei den nicht linearen Zustandsraummodellen beschränkt sich die Untersuchung auf eindimensionale Probleme, da dort mit vertretbarem Rechenaufwand sehr exakte Approximationen realisierbar sind, die als Referenzzeitreihe benötigt werden.

Die Simulationsstudie wurde in R (Version 3.0.1, 64 Bit) implementiert, wobei die im Anhang C dokumentierten Routinen genutzt wurden. Die Berechnungen erfolgten auf einer Linux-Workstation (Ubuntu, Kernel 3.8.0-29-generic x86_64 GNU/Linux) mit zwei Intel(R) Xeon(R) Prozessoren vom Typ W3680 (3.33 GHz, insgesamt 12 Kerne) und 12 GB Arbeitsspeicher.

5.2. Konvergenzverhalten

In diesem Abschnitt wird das Konvergenzverhalten der Approximationsverfahren untersucht. Hierbei wird zum einen die Anzahl der Stützstellen bzw. Partikel gegen den RMSE abgetragen, zum anderen wird die Rechendauer gegen den RMSE abgetragen. Im Abschnitt 5.2.1 werden die Ergebnisse für lineare und normalverteilte Zustandsraummodelle illustriert. In den Abschnitten 5.2.2 und 5.2.3 werden nichtlineare, aber normale bzw. nichtlineare und nicht normalverteilte Modelle untersucht.

5.2.1. lineare/normalverteilte Zustandsraummodelle

Die hier untersuchten ein- bis vierdimensionale Zustandsraummodelle gehören zu der Gruppe der linearen und normalverteilten Zustandsraummodelle. Die konkrete Spezifikation entspricht einer zeitvarianten Regression wie in Gleichung 5.2 und 5.3 beschrieben:

$$y_t = X_t^{'}\beta_t + \varepsilon_t \quad \varepsilon_t \sim N(0,1) \tag{5.2}$$

$$\beta_t = 0.8\beta_{t-1} + \omega_t \quad \omega_t \sim N(0,1) \tag{5.3}$$

Wobei $X_t^{'}$ ein Zeilenvektor und β_t ein Spaltenvektor jeweils der Länge d mit $d = 1, \ldots, 4$. Die künstlichen Datensätze werden gemäß diesem Modell für $t = 1, \ldots, 10$ generiert, wobei im eindimensionalen Fall der Datenvektor X_t nur eine Konstante enthält und für jede weitere Dimension Werte aus einer Gleichverteilung hinzugefügt werden. In den Filterprozess gehen dann die Variablen y_t und X_t ein.

Für diese Form von Zustandsraummodellen existiert eine analytische Lösung wie im Anhang A beschrieben. Diese analytische Lösung bildet die Referenzzeitreihe.

Die Abbildungen 5.1 und 5.2 fassen die Ergebnisse der Simulationsstudie zusammen. In den Graphen wird jeweils die Anzahl der Stützstellen gegen den Approximationsfehler (RMSE) abgetragen, wobei die Achsen der Graphen jeweils log-skaliert sind.

Es ist erkennbar, dass für ein- bis dreidimensionale Modelle die sNI bessere Konvergenzeigenschaften hat als die Partikel-Filter. Bei den vierdimensionale Modellen liegen sowohl die Partikelfilter, als auch die sNI gleich auf. Die beiden vorgestellten Partikel-Filter liefern sehr ähnliche Resultate. In der Gruppe der sNI-Verfahren schneiden die dynamischen Modelle stets besser ab als die statischen. Unter den dynamischen Modellen dominiert das iterative Verfahren. Die dynamischen dünnen Gitter haben nur einen relativ geringen Effekt auf die Effizienz der Approximation.

Die Abbildungen 5.3 und 5.4 stellen ebenfalls das Ergebnis dar, allerdings wird hier die Dauer der Berechnung gegen den Approximationsfehler (RMSE) abgetragen. Da die Verfahren unterschiedlich rechenaufwendig sind, ergeben sich leicht unterschiedliche Resultate.

Wie zuvor haben die sNI-Verfahren in ein- bis dreidimensionalen Modellen bessere Konvergenzeigenschaften. Bei vierdimensionalen Modellen dominieren die sMC-Methoden. Die Konvergenzgeschwindigkeit ist in der Gruppe der sNI-Verfahren ähnlich; es existieren jedoch zum Teil erhebliche Unterschiede im Ausgangsniveau der Dauer.

In der Abbildung 5.5 werden für den 3D-Fall die Anzahl der Stützstellen gegen die Dauer abgetragen. Die bestätigt die vorherige Äußerung, dass die Algorithmen sich in der Rechenkomplexität stark unterscheiden.

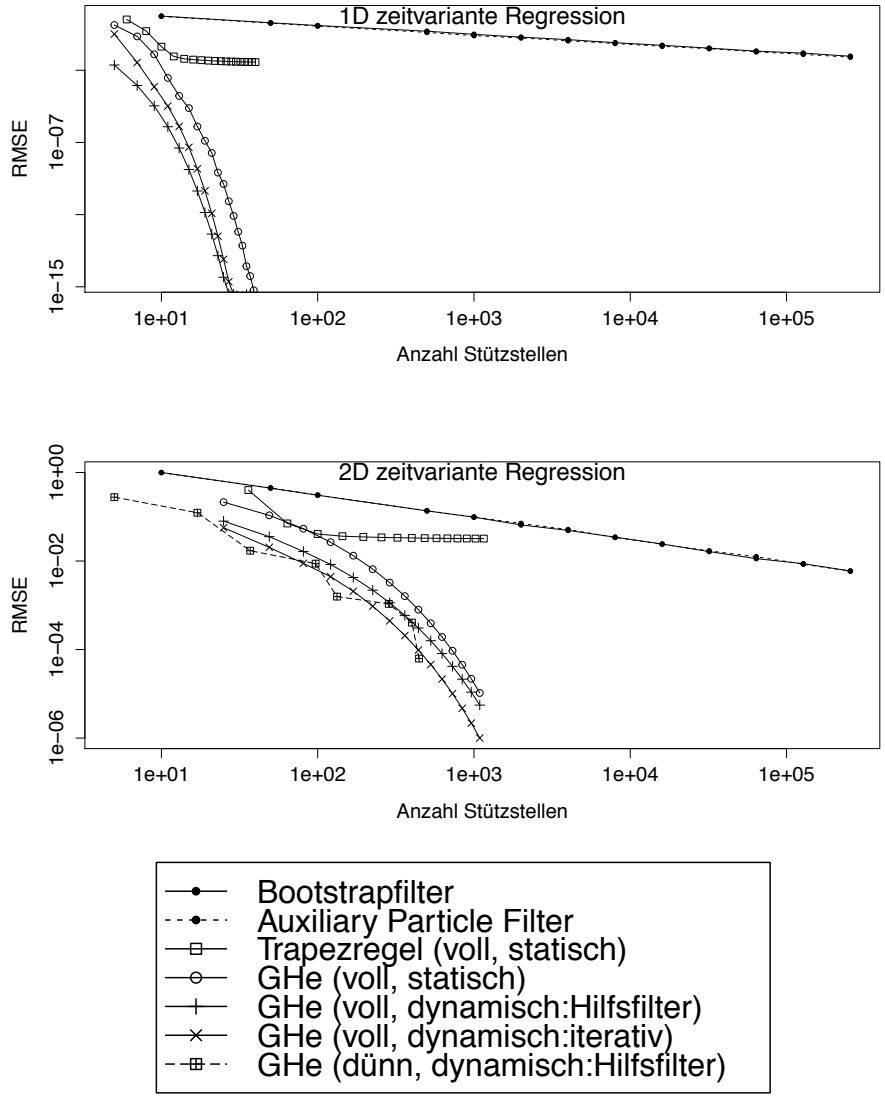

Abbildung 5.1.: Konvergenzverhalten 1-2D (Anzahl Stützstellen vs. RMSE)

Zum einen gibt es einen Unterschied in der Steigung. Während die Zahl der Funktionsauswertung bei den Bootstrap-Filtern mit $\mathcal{O}(N)$ steigt, steigt der Rechenaufwand für die sNI mit $\mathcal{O}(N^2)$. Zudem unterscheiden sich die Niveaus der sNI. Die statischen Methoden sind am schnellsten, die Hilfsfilter liegen im Mittelfeld und am rechenaufwändigsten ist das iterative Verfahren.

5.2.2. Nichlineares Zustandsraummodell

Als Beispiel für ein nichtlineares Zustandsraummodell wird das eindimensionale stochastische Volatilitäts-Modell (SVM) verwendet wie in Gleichungen 5.4 und 5.5 beschrieben[1].

[1]In Abschnitt 6.2, wird diese Klasse von Modellen detaillierter besprochen.

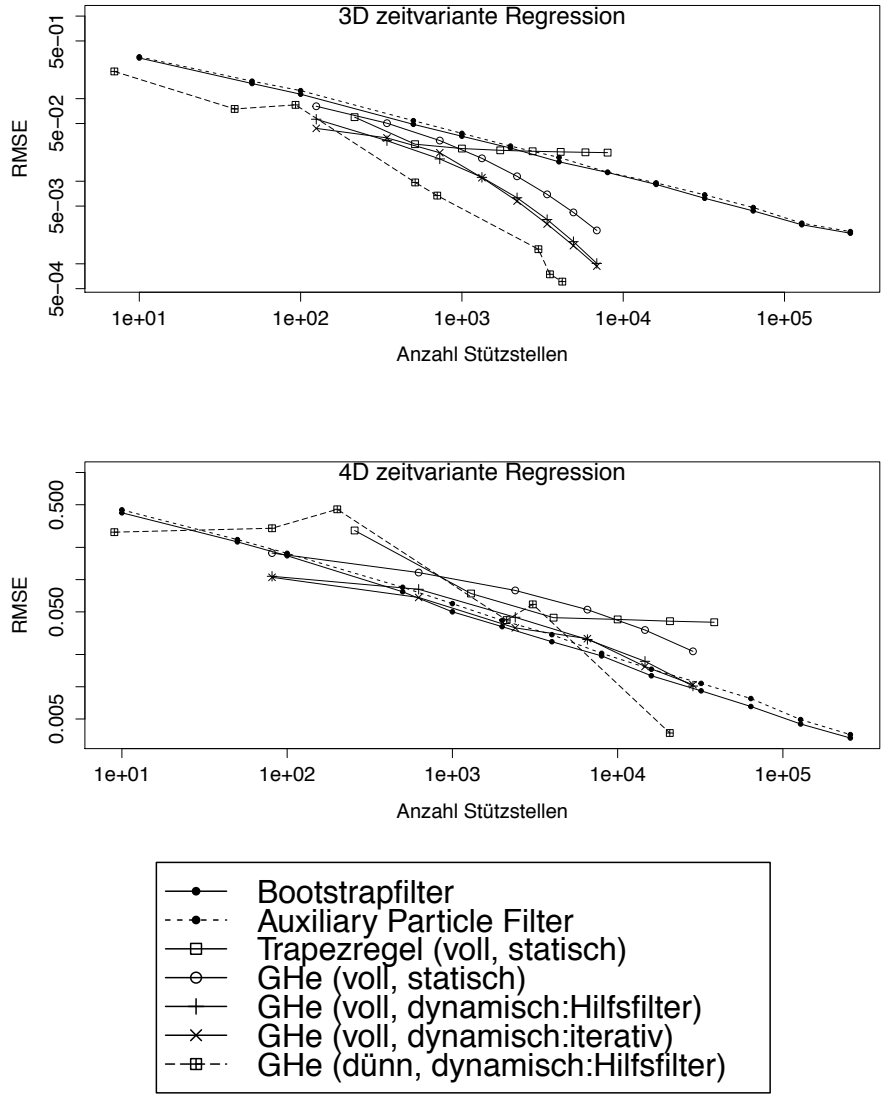

Abbildung 5.2.: Konvergenzverhalten 3-4D (Anzahl Stützstellen vs. RMSE)

$$y_t = \bar{y} + exp(x_t/2) \cdot \varepsilon_t \quad \varepsilon_t \sim N(0,1) \tag{5.4}$$

$$x_t = \alpha + 0.9x_{t-1} + \omega_t \quad \omega_t \sim N(0,1) \tag{5.5}$$

Für diese Gruppe von Zustandsraummodellen exisitiert keine analytische Lösung. Als Ersatz dafür wurde eine Approximation mit einem statischen Gitter mit ca. 10^5 Stützstellen berechnet.

Das Resultat der Simulationsstudie ist in Abbildung 5.6 zusammengefasst. Wie auch im eindimensionalen linearen Modell haben die sNI-Verfahren das bessere Konvergenzverhalten als der Partikel-Filter. Auch hier schneiden die dynamischen Modelle besser ab als die statischen. Somit widerlegen diese Resultate nicht die Erkenntnisse aus den linearen

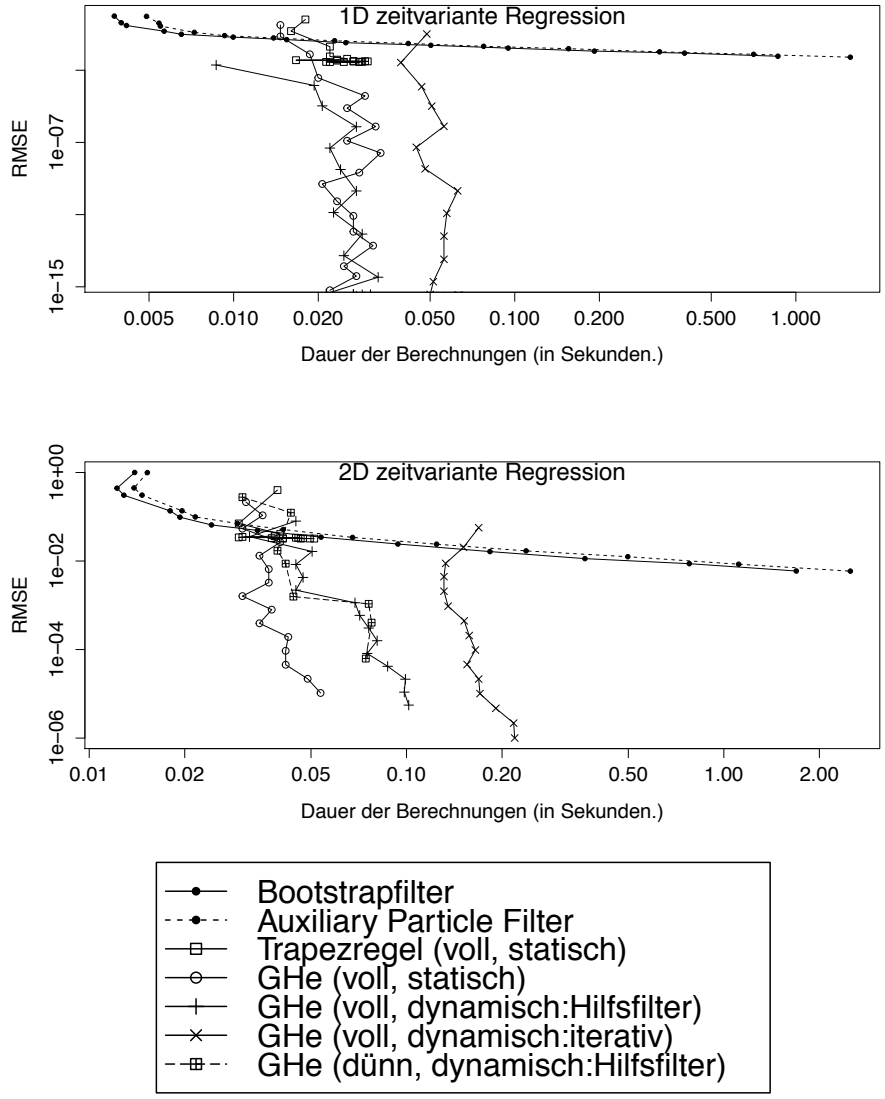

Abbildung 5.3.: Konvergenzverhalten 1-2D (DAUER vs. RMSE)

Modellen.

5.2.3. Nichtlineare und nicht-normale Zustandsraumodelle

An dieser Stelle wird das Modell aus dem vorherigen Abschnitt wieder aufgenommen. Allerdings wird das normalverteilte Rauschen in Gleichung 5.4 durch ein t-verteiltes Rauschen ersetzt. Wichtig ist, dass die t-Verteilung so angepasst wird, dass deren Varianz gleich eins ist.

Qualitativ verändert sich das Ergebnis der Simulationsstudie durch die Änderung des Rauschterms nicht, wie in Abbildung 5.7 zu sehen ist.

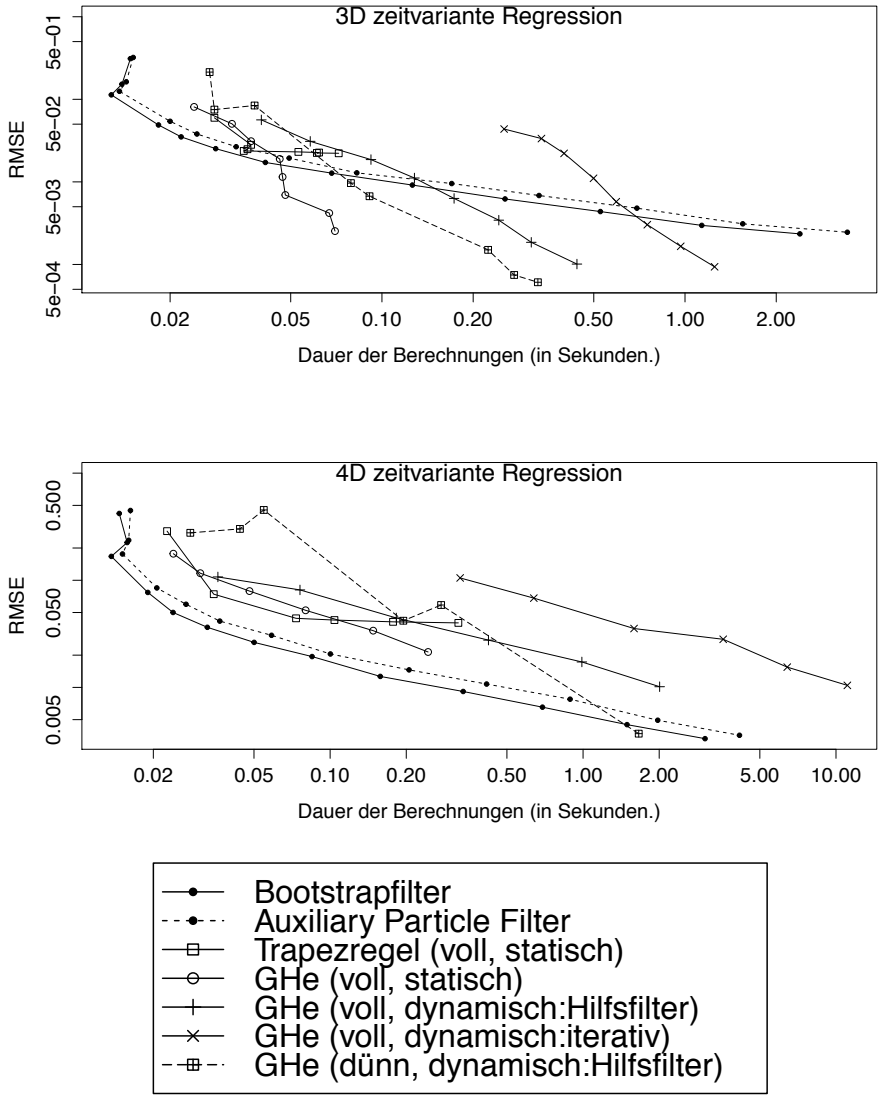

Abbildung 5.4.: Konvergenzverhalten 3-4D (DAUER vs. RMSE)

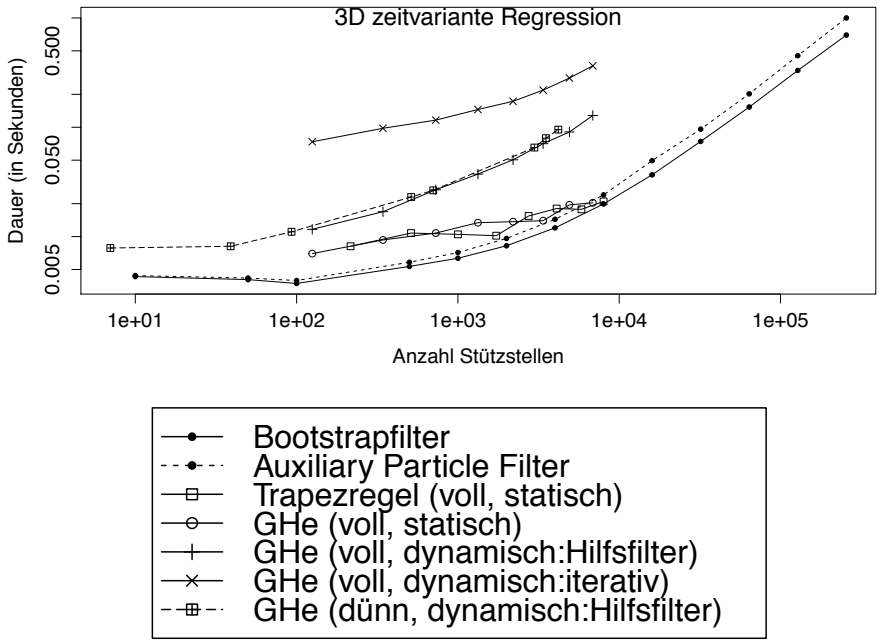

Abbildung 5.5.: Unterschied in der Komplexität der Algorithmen

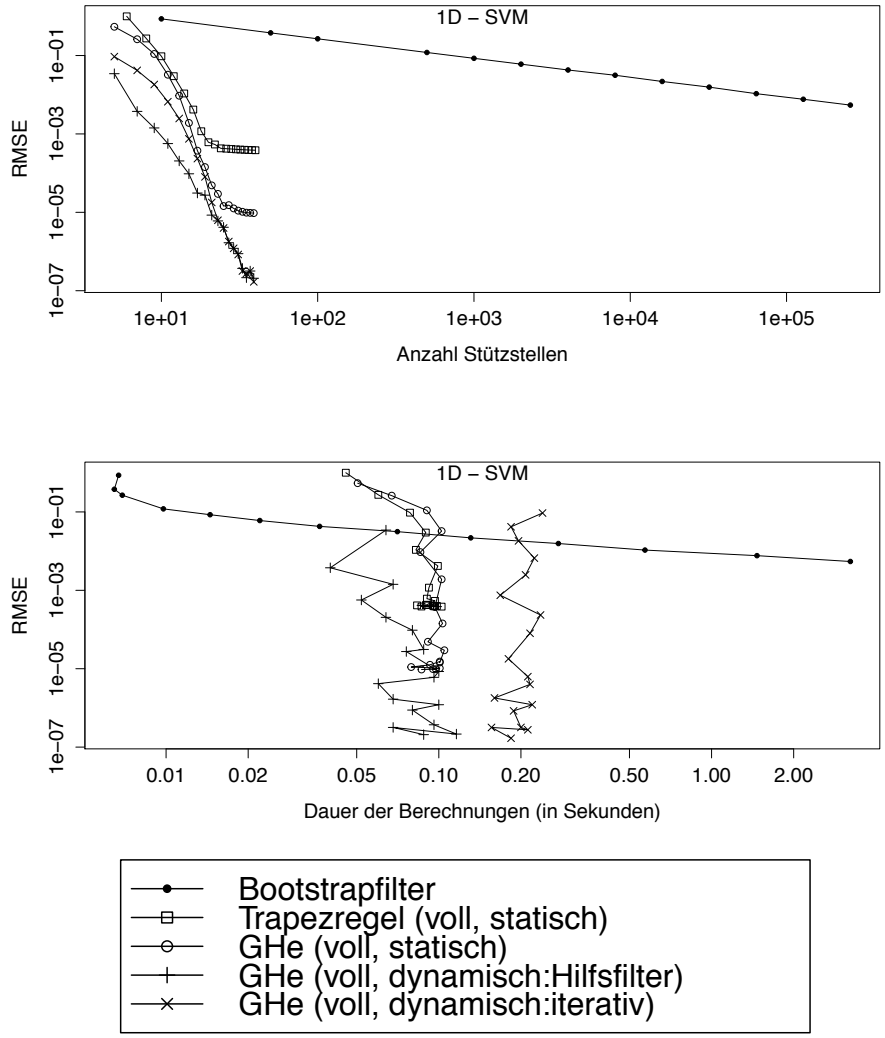

Abbildung 5.6.: Konvergenzverhalten im SVM (1D, normalverteiltes Rauschen)

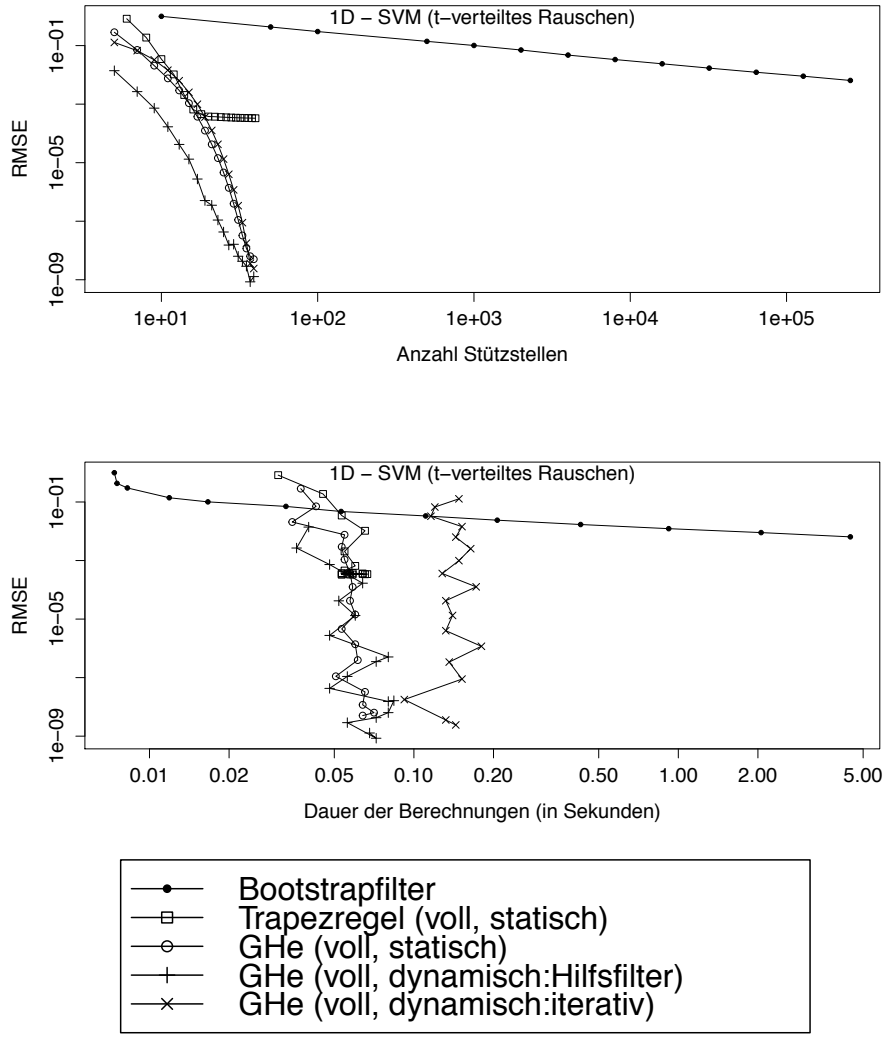

Abbildung 5.7.: Konvergenzverhalten im SVM (1D, t-verteiltes Rauschen)

6. Implementierung

Die in dieser Arbeit vorgestellten Algorithmen wurden in der Programmiersprache R umgesetzt und in dem Paket `nife` zusammengeführt. In diesem Kapitel wird das Paket vorgestellt und dessen Nutzung an einem Anwendungsbeispiel illustriert.

6.1. Sofware-Paket `nife`

Zum Zeitpunkt der Drucklegung dieser Arbeit liegt das Software-Paket `nife` in der Version 0.8 vor. Es umfasst alle in dieser Arbeit vorgestellten Algorithmen zur numerischen Integration und Inferenz von Zustandsraummodellen. Letzteres schließt sowohl die Partikel-Filter (BF und APF), als auch die Filter auf Basis der sNI mit ein.

Das Paket kann über die URL `http://www.staff.uni-mainz.de/weiserc/code/nife_0.8.0.tar.gz` bezogen werden. In der aktuellen Version ist das Paket ausschließlich für UNIX-basierte Systeme konzipiert, da es UNIX-spezifische Methoden für des parallele Rechnen nutzt.

Das Paket `nife` greift auf das frei verfügbare Paket `mvtnorm` zurück. Falls es nicht bereits vorhanden ist, kann es mit dem Befehl `install.packages(pkgs = "mvtnorm", dependencies = TRUE)` installiert werden.

Nachdem dieses Paket verfügbar ist, kann mit dem folgenden Befehl das Paket `nife` installiert werden:

```
install.packages(pkgs = "PFAD/nife_0.8.0.tar.gz", dependencies = TRUE,
type = "source", repos = NULL)
```
[1]

. `PFAD` bezeichnet den Dateipfad zur heruntergeladenen Datei. Nach der Installation muss das Paket für jede R-Sitzung mit dem Befehl `library(nife)` eingebunden werden.

Die in dem Paket hinterlegten Funktionen gliedern sich in zwei Gruppen. Eine umfasst Funktionen der numerischen Integration, eine zweite die Funktionen für die Berechnung von Zustandsraummodellen. Die folgende Tabelle führt die wichtigsten Funktionen auf. Eine detaillierte Beschreibung der Befehle (in englischer Sprache) ist im Anhang C abgedruckt.

[1] Gegebenenfalls müssen die Befehle auf die individuellen Gegebenheiten angepasst werden. Die Dokumentation des Befehls `install.package` gibt hier Hilfestellung.

Numerische Integration	
`Grid1D`	generiert Stützstellen und Gewichte für 1D-Quadraturen
`fGridnD`	konstruiert volles n-dimensionales Gitter mit der Produktregel
`sGridnD`	konstruiert dünnes n-dimensionales Gitter mit der Kombinationstechnik
`MoveGrid`	reskaliert ein übergebenes Gitter gemäß einer der Methoden aus Abschnitt 4.4

Berechnung von Zustandsraummodellen	
`NIFilter`	Filter auf der Basis von sNI; umfasst die in der Arbeit diskutierten Varianten
`BootstrapFilter`	BF gemäß Gordon, Salmond und Smith (1993)
`AParticleFilter`	APF nach Pitt und Shephard (1999)
`KalmanFilter`	analytischer Kalman-Filter für lineare und normalverteilte Zustandsraummodelle

Die Filter-Funktionen `NIFilter`, `BootstrapFilter` und `AParticleFilter` sind so ausgelegt, dass sie für möglichst allgemeine Zustandsraummodelle eingesetzt werden können. Die konkrete Spezifikation des Zustandsraummodells muss hierfür in einer standardisierten Form als sogenanntes `StSp.Obj` (State-Space Object) den Routinen übergeben werden.

Ein Zustandsraummodell wird durch die A-Priori-Verteilung, die Zustandsgleichung, die Beobachtungsgleichung und die Hyperparameter beschrieben. Genau diese Komponenten werden auch im `StSp.Obj` abgebildet, wobei sich je nach Filter die Formulierungen unterscheiden. Die drei Abbildungen 6.1 bis 6.3 zeigen exemplarisch ein konkretes `StSp.Obj`. Es wird jeweils für die drei Filter ein „Random Walk + Noise“-Modell beschrieben (siehe Beispiel in Abschnitt 2.1).

Je nach verwendetem Filter unterscheiden sich die Anforderungen an das `StSp.Obj`. So wird z.B. bei den Partikelfiltern eine Funktion `rprior` mit einem Parameter `N` erwartet, die N-Ziehungen aus der A-Priori Verteilung zurück gibt. Im Gegensatz dazu erwartet der `NIFilter` eine Funktion `dprior` im `StSp.Obj`, welche die Dichte der A-Priori Verteilung für die Stützstellen `x` zurückgibt.

Das folgende Anwendungsbeispiel illustriert detailliert die Nutzung des `NIFilters`.

6.2. Anwendungbeispiel: stochastisches Volatilitäts-Modell

Als ein Anwendungsbeispiel wird nun das stochastische Volatilitäts-Modell betrachtet. Mit diesem Modell kann ähnlich wie bei ARCH-Modellen (bzw. GARCH-Modellen und

`StSp.Obj` für einen `NIFilter`

```
StSp.Obj = list(
    dprior = function(x) {
        return(dnorm(x, 0, 10))
    },
    dtrans = function(x, old.x) {
        return(dnorm(x-old.x, 0, 1))
    },
    dlike  = function(y,x,t) {
        return(dnorm(y-x, 0, 1))
    }
)
```

Abbildung 6.1.: Beispiel eines `StSp.Obj` für die Nutzung im `NIFilter`

deren Erweiterungen) die sich dynamisch entwickelnde Volatilität von Zeitreihen modelliert werden. Eine Variante des Modells ist in den Gleichungen 6.1 und 6.2 gegeben.

$$y_t = \bar{y} + exp(x_t/2) \cdot \varepsilon_t \quad \varepsilon_t \sim N(0,1) \tag{6.1}$$

$$x_t = \alpha + \beta x_{t-1} + \omega_t \quad \omega_t \sim N(0, \sigma_\omega^2) \tag{6.2}$$

In dieser einfachen Ausgestaltung wird angenommen, dass die Zeitreihe einem weißen Rauschen um eine Konstante folgt, wobei die Varianz des Rauschens als unbeobachtbarer Zustand (x_t) modelliert wird. Der Zustand selbst entwickelt sich gemäß einem stationären $AR(1)$-Prozess. Das Modell ist nicht linear, da x_t, durch die natürliche Exponentialfunktion transformiert, multiplikativ in die Beobachtungsgleichung eingeht.

Modelle dieser Art wurden intensiv z.B. in Ghysels, Harvey und Renault (1996) diskutiert. Eingesetzt werden diese Modelle häufig im Bereich der Finanzwissenschaft und dort z.B. in der Risiko- und Portfoliobewertung.

Da aber wegen der Struktur des Modells keine analytische Lösung existiert, wurden verschiedene Näherungslösungen angewandt. Einen Überblick verschafft die Arbeit von Broto und Ruiz (2004), in der die populärsten Lösungsansätze verglichen werden. Im Zentrum stehen hier nicht die gefilterten Zustände selbst, sondern vielmehr die Schätzung der unbekannten Parameter (hier: $\alpha, \beta, \sigma_\omega^2$ und $\bar{y}$).

Der Ansatz der numerischen Integration ermöglicht eine direkte Auswertung der Likelihood und somit die Anwendung der Maximum-Likelihood-Methode für die Schätzung der unbekannten Parameter, was diesen Ansatz von z.B. den Partikel-Filtern unterschei-

`StSp.Obj` für einen `BootstrapFilter`

```
StSp.Obj = list(
    rprior = function(N) {
        return(rnorm(N, 0, 10))
    },
    rtrans = function(old.x) {
        return(old.x + rnorm(length(old.x), 0, 1))
    },
    dlike  = function(y,x,t) {
        return(dnorm(y-x, 0, 1))
    }
)
```

Abbildung 6.2.: Beispiel eines `StSp.Obj` für die Nutzung im `BootstrapFilter`

det.

Im Folgenden werden exemplarisch für die Aktie der Commerzbank sowohl die Parameter geschätzt als auch die gefilterte Volatilität bestimmt.

Ausgangspunkt sind die beobachteten Tagesrenditen der Commerzbank Aktie, wie in Abbildung 6.4 dargestellt.

Das für die Funktion `NIFilter` spezifizierte `StSp.Obj` hat die Form wie in Abbildung 6.5 beschrieben.

Die Parameter `alpha` (α), `beta` (β), `sd_w` ($\sqrt{\sigma_\omega^2}$) und `y.bar` ($\bar{y}$) sind ex ante nicht bekannt und wurden mit Hilfe der Maximum-Likelihood-Methode bestimmt[2]. Dabei maximierten die folgenden Werte die Likelihood des Modells:

α	0.10694
β	0.80235
σ_ω^2	0.67625
$\bar{y}$	-0.00117

Die mit diesen Parametern gefilterte Zeitreihe $exp(x_t/2)$ wird in Abbildung 6.6 dargestellt.

[2] Als Optimierungsroutine wurde die Funktion `maxBHHH` aus dem Paket `maxLik` verwendet. Die numerische Maximierung konvergierte nach 11 Iterationen bei einer log. Likelihood von -444.182.

`StSp.Obj` für einen `AParticleFilter`

```
StSp.Obj = list(
    rprior = function(N) {
        return(rnorm(N, 0, 10))
    },
    dprop  = function(y, x, t) {
        return(dnorm(y-x,0,sqrt(2)))
    },
    rtrans = function(old.x) {
        return(old.x + rnorm(length(old.x), 0, 1))
    }
    dlike  = function(y,x,t) {
        return(dnorm(y-x, 0, 1))
    }
)
```

Abbildung 6.3.: Beispiel eines `StSp.Obj` für die Nutzung im `AParticleFilter`

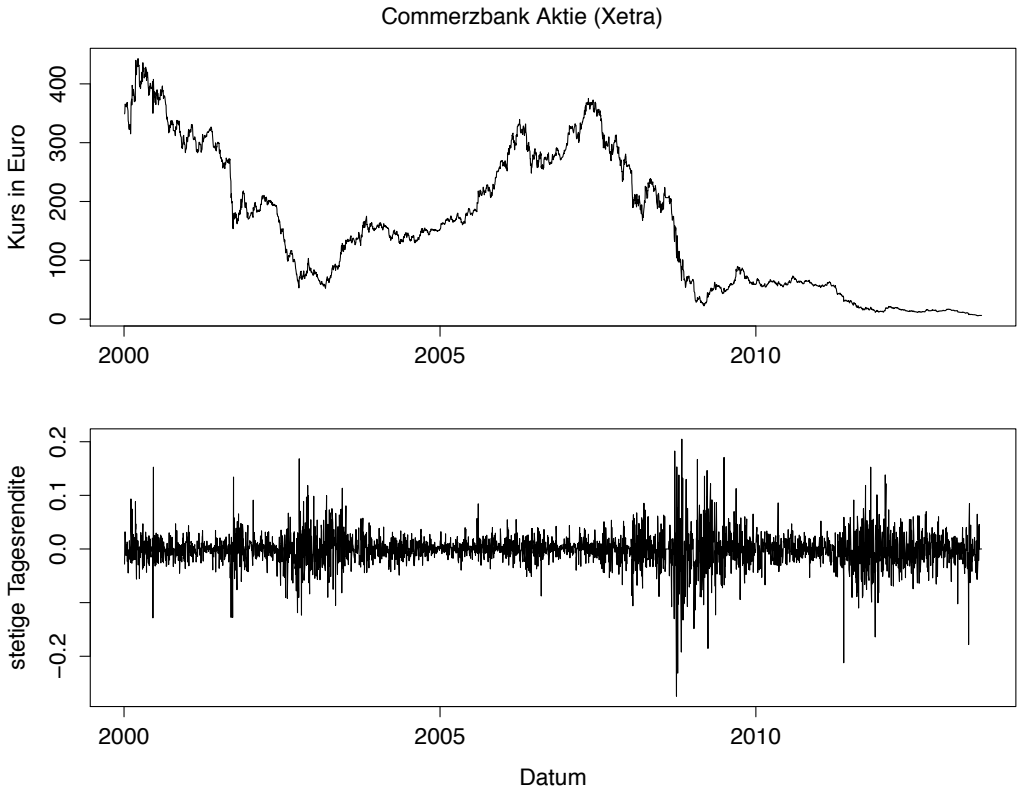

Abbildung 6.4.: Kurs und Tagesrendite der Commerzbank-Aktie

```
StSp.Obj = list(
    dprior = function(x) {
        return(dnorm(x, 0, 4))
    },
    dtrans = function(x, old.x) {
        return(dnorm(x-(alpha + beta*old.x), 0, sd_w))
    },
    dlike  = function(y,x,t) {
        return(dnorm(y-y.bar, 0, exp(x/2)))
    }
)
```

Abbildung 6.5.: StSp.Obj für das stochastische Volatilitäts Modell

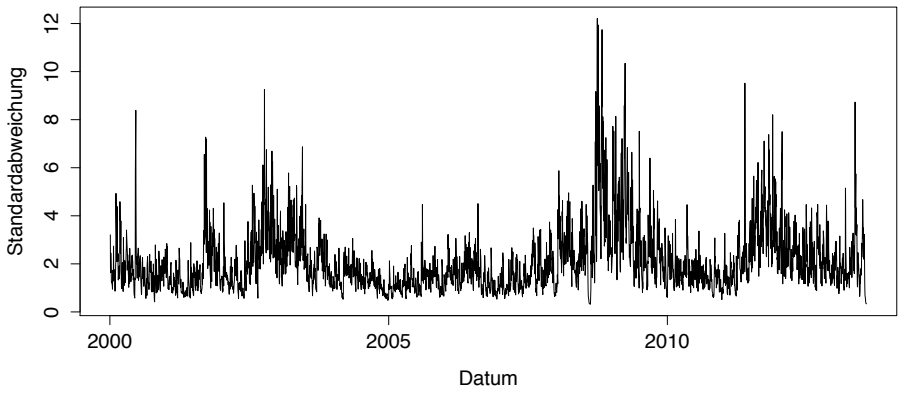

Abbildung 6.6.: Gefilterte Volatilität der Commerzbank-Aktie

7. Schlussbemerkungen

7.1. Zusammenfassung

Zu Beginn der Arbeit wurde das Prinzip der Zustandsraummodelle kurz vorgestellt. Da lediglich für eine begrenzte Gruppe von konkreten Spezifikationen analytische Lösungen existieren, müssen bei der Berechnung allgemeiner Zustandsraummodelle Näherungsverfahren eingesetzt werden. Im Kapitel 3 wurde diesbezüglich der aktuelle Forschungsstand skizziert.

In der wissenschaftlichen Diskussion dominieren gegenwärtig sogenannte Partikel-Filter. Vor Aufkommen dieser Filter wurden Methoden diskutiert, die die numerische Integration zur näherungsweisen Berechnung der Integrale verwenden. Wobei seit den frühen 1990er Jahre keine nennenswerte Beiträge mehr veröffentlicht wurden. Grund hierfür war vor allem die praktisch sehr rechenaufwendige Übertragung der Algorithmen auf mehrdimensionale Zustandsräume.

Ziel dieser Arbeit war es, die Entwicklungen im Bereich der numerischen Integration auf die letztgenannten Methoden zu übertragen und die Leistungsfähigkeit dieser neuen Algorithmen zu untersuchen, wobei insbesondere die Übertragbarkeit auf höherdimensionale Modelle im Fokus stand.

So wurden in Kapitel 4 drei Erweiterungen vorgestellt. (1) Die Kombinationstechnik, die eine numerische Integration höherdimensionaler Funktionen ermöglicht. (2) Das Reskalieren mehrdimensionaler Gitter und (3) die Umsetzung dynamischer Gitter, die auf der Reskalierung der Gitter beruht.

In Kapitel 5 wurden die Algorithmen der sNI mit den vorgestellten Partikelfilter bzgl. ihrer Effizienz verglichen.

Resultat des Vergleichs ist eine Überlegenheit der sNI-Algorithmen für niedrigdimensionale Zustandsraummodelle ($d = 1, ..., 3$). Bei höherdimensionalen Modellen schneiden jedoch sMC-basierte Algorithmen besser ab. Dieses Ergebnis zeigt, dass das in Abbildung 4.4 (rechts) illustrierte Konvergenzproblem bei der Faltung nicht ausreichend durch die Konvergenzvorteile bei der Integration der A-Posterori-Verteilung (links) kompensiert werden kann.

In Kapitel 6 wurde kurz das Programmpaket `nife` vorgestellt, welches die vorgestellten

Algorithmen zusammenfasst. Die Anwendung des Pakets wurde abschließend in einem kurzen Beispiel illustriert.

7.2. Offene Forschungsfragen

Neben den hier diskutierten Punkten ergeben sich eine Reihe von fortgeschrittenen Forschungsfragen, die an dieser Stelle überblicksartig angedeutet werden sollen.

Wahl der Quadraturregel
Wie in der Arbeit beschrieben, ist die verwendete Gauss-Hermite-Quadratur ausgelegt, um Funktionen der Form Polynom $\times$ Normalverteilung zu integrieren. Im Allgemeinen unterscheidet sich jedoch die zu integrierende Funktion davon. So werden im Kontext dieser Arbeit stets Funktionen der Form Verteilung$\times$Verteilung integriert, wobei die Verteilungen in der Regel keine Normalverteilungen sind. Das ist der Ausgangspunkt für die Frage, welche Quadraturregel für ein allgemeines Zustandsraummodell am besten geeignet ist.

Faltung und dünne Gitter
Die Simulationsstudie in Kapitel 5 hat gezeigt, dass die Anwendung der dünnen Gitter es in diesem Kontext nicht ermöglicht, die sNI auf höher-dimensionale Probleme zu verallgemeinern, wozu der dünn-Gitter-Ansatz in anderen Anwendungen durchaus in der Lage war (wie z.B. in Heiss und Winschel (2008)). Die Vermutung steht im Raum, dass die Idee der dünnen Gitter im Widerspruch zum Konzept der Faltung steht. Aus dieser Vermutung ergeben sich zwei Fragestellungen. (1) Lässt sich dieser vermutete Widerspruch formal zeigen und gibt es gebenenfalls Anpassungsmöglichkeiten? Und (2) existieren alternative Möglichkeiten zur Berechnung der A-Priori-Verteilung $p(x_t|y_{1:t-1})$ - wie die Fast-Fourier-Transformationen - die eine Anwendung des sNI-Ansatzes für höher-dimensionale Zustandsraummodelle ermöglicht?

Auswertung der A-Posterori-Verteilungen
In dieser Arbeit stand das sequentielle bayessche Filtern im Zentrum. Bei der Anwendung von Zustandsraummodellen kann neben dem Filtern auch eine systematische Auswertung der gefilterten Zustände von Interesse sein. So könnte ein Anwender z.B. an High-Density-Regions, Hypothesentests oder Prognosen interessiert sein. Auch für diese Fragen benötigt man im allgemeinen Fall Näherungsverfahren. Hier stellt sich die Frage, inwieweit hier die Methoden der sNI bzw. der numerischen Integration allgemein anwendbar sind. Auch hier existieren Arbeiten zu niedrig-dimensionalen Zustandsraummodellen. Deren Erweiterung auf höher-dimensionale Modelle ist jedoch nicht erprobt.

7.3. Fazit

Diese Arbeit entspringt einem Forschungsgebiet, welches seit den frühen 1990er Jahren kaum mehr in der wissenschaftlichen Diskussion zu finden ist. Dennoch ist es für die praktische Arbeit von Relevanz, was Publikationen aus dem angewandten Bereich der Wirtschaftswissenschaften zeigen.

Da auf dem Gebiet der numerischen Integration erhebliche Fortschritte gemacht wurden, erschien es sinnvoll, diese aufzugreifen und mit dem historischen Ansatz zu verbinden. Als Resultat ergaben sich praktisch nutzbare und leistungsfähigere Algorithmen. Doch das Ziel, auch höher-dimensionale Modelle ($4 \leq d \leq 10$) dadurch berechenbar zu machen, ist nicht erreicht.

Die bei der Bearbeitung aufgetauchten Fragen geben Anlass zur Vermutung, dass eine weitere Leistungssteigerung der Algorithmen möglich ist. Zentrales Momemt ist hier die Faltung bei der Zustandsfortschreibung, für die verbesserter Lösungsansatz benötigt wird.

Anhang

A. Kalman-Filter

Für die rekursive Berechnung der Wahrscheinlichkeitsdichten im linearen und normalverteilten Zustandsraummodell hat Kalman (1960) eine analytische Lösung hergeleitet, der Kalman-Filter. Dank der Beschränkung auf lineare und normalverteilte Modelle sind die notwendigen Integrale analytisch gegeben. Die resultierenden Wahrscheinlichkeitsdichten sind wiederum normalverteilt und können somit vollständig durch den Mittelwertvektor und die Varianz-/Kovarianzmatrizen beschrieben werden.

Für ein allgemeines Zustandsraummodell gemäß der folgenden Gleichungen:

$$\begin{aligned} y_t &= F_t x_t + \varepsilon_t \qquad \text{mit } \varepsilon_t \sim N(0, E_t) \\ x_t &= G_t x_{t-1} + \omega_t \qquad \text{mit } \omega_t \sim N(0, W_t) \text{ und } x_0 \sim N(m_0, C_0) \end{aligned}$$

ergibt sich die rekursive Lösung nach dem Ablaufschema in Abbildung A.1[1].

1. `definiere` m_0 `und` C_0 `der A-Priori-Verteilung` $N(m_0, C_0)$
2. `für t = 1, ..., T`
   a) `Übergangsdichte:` $p(x_t|x_{t-1}) \sim N(a_t, R_t)$ `mit ...`
      $a_t = G_t m_{t-1}$
      $R_t = G_t C_{t-1} G_t' + W_t$
   b) `Vorhersagedichte:` $p(y_t|x_{t-1}) \sim N(f_t, Q_t)$ `mit ...`
      $f_t = F_t' a_t$
      $Q_t = F_t' R_t F_t + E_t$
   c) `A-Posterori-Dichte:` $p(x_t|y_t) \sim N(m_t, C_t)$ `mit ...`
      $e_t = y_t - f_t$
      $A_t = R_t F_t Q_t^{-1}$
      $m_t = a_t + A_t e_t$
      $C_t = R_t - A_t A_t' Q_t$

Abbildung A.1.: Algorithmus des Kalman-Filters

Für lineare und normalverteilte Zustandsraummodelle ist diese Lösung exakt, zudem ist

[1] Für eine ausführliche Herleitung sei auf Pole, West und Harrison (1994), Kapitel 3, verwiesen.

die Berechnung selbst bei höherdimensionalen Modellen mit geringem Rechenaufwand verbunden.

B. Einführung in die numerische Integration

In diesem Kapitel werden kurz die elementaren Konzepte und Begriffe der numerischen Integration, die auch als Quadratur bezeichnet wird, für ein- und mehrdimensionale Funktionen beschrieben. Für eine detaillierte Einführung sei z.B. auf Davis und Rabinowitz (1984) verwiesen.

Eindimensionale Integrale

Ausgangspunkt ist die eindimensionale Funktion $f(x)$, für die das bestimmte Integral

$$I = \int_a^b f(x)\,dx, \quad \text{mit } -\infty \leq a \leq b \leq \infty$$

entweder nicht in geschlossener Form gegeben oder dessen Berechnung sehr aufwendig ist. Dieses Integral wird bei der numerischen Integration durch eine gewichtete Summe approximiert,

$$I \approx Q_l f := \sum_{i=1}^{n} w_i \cdot f(x_i)$$

wobei die Stützstellen und Gewichte $(x_i, w_i) \quad \forall i = 1, ..., n$ sich entsprechend der gewählten Quadraturregel (Q) für ein Genauigkeitslevel l ergeben. Für das Genauigkeitslevel l soll gelten, dass mit steigendem l die Genauigkeit der Quadraturregel sich erhöht (oder zumindest nicht sinkt). In der Regel geht mit steigendem l eine steigende Anzahl von Stützstellen einher.

Im Rahmen dieser Arbeit werden lediglich drei Quadraturregeln Verwendung finden: die Trapezregel, die Gauß-Hermite-Quadratur und die genestete Variante der Gauß-Hermite-Quadratur nach Genz und Keister (1996). Im konkreten Anwendungsfall können andere Quadraturregeln effizienter sein. Die getroffene Auswahl bildet daher lediglich einen eher allgemeinen Lösungsvorschlag.

Trapezregel

Die Trapezregel gehört zur Gruppe der wiederholten Newton-Cotes-Formeln. Die Stützstellen x_i werden in der Regel so gewählt, dass sie das zu integrierende Intervall $[a, b]$ in $(n-1) > 1$ gleich breite Subintervalle unterteilt. Für den Fall äquidistanter Subintervalle mit der Breite $h = \frac{b-a}{(n-1)}$ ergeben sich die Stützstellen und Gewichte wie folgt:

$$\begin{aligned} x_i &= a + (i-1)h \qquad \forall i \in \{1, ..., n\} \\ w_i &= \{\frac{h}{2}, h, h, ..., h, h, \frac{h}{2}\} \end{aligned}$$

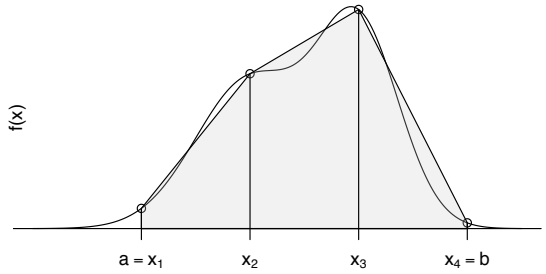

Abbildung B.1.: Trapezregel (n = 4)

Abbildung B.1 illustriert die Trapezregel für das Intervall $[a, b]$ mit $n = 4$ Stützstellen. Hier wird die zu integrierende Funktion durch eine abschnittsweise lineare Funktion approximiert und diese Approximation exakt integriert. Der Flächeninhalt des grau schraffierten Polygons wird somit als Näherungswert für das wahre Integral angenommen. Die Genauigkeit der Approximation kann durch eine Erhöhung der Anzahl der Stützstellen gesteigert werden.

Häufig wird bei der Trapezregel die Anzahl der Stützstellen so gewählt, dass sie sich für ein vorgegebenes Genauigkeitslevel l zu $n_l = 2^{l-1} + 1$ ergeben. Abbildung B.2 zeigt die Stützstellen der ersten 5 Level für die eindimensionale Trapezregel. Man kann erkennen, dass die Stützstellen eines Levels in allen folgenden Leveln auch vorkommen und lediglich neue Punkte hinzugefügt werden. Man bezeichnet dies als genestete Quadraturregel.

Im Kontext dieser Arbeit werden Dichtefunktionen integriert, die in der Regel für das Intervall $[-\infty, \infty]$ definiert sind. Solchen Dichtefunktionen ist gemein, dass sie für $x \to -\infty$ bzw. $x \to \infty$ gegen Null abfallen und der Großteil der Wahrscheinlichkeitsmasse sich auf einen kompakten Träger verteilt. Möchte man solche Funktionen mit der Trapezregel integrieren, wählt man das zu integrierende Intervall $[a, b]$ so, dass außerhalb des Intervalls

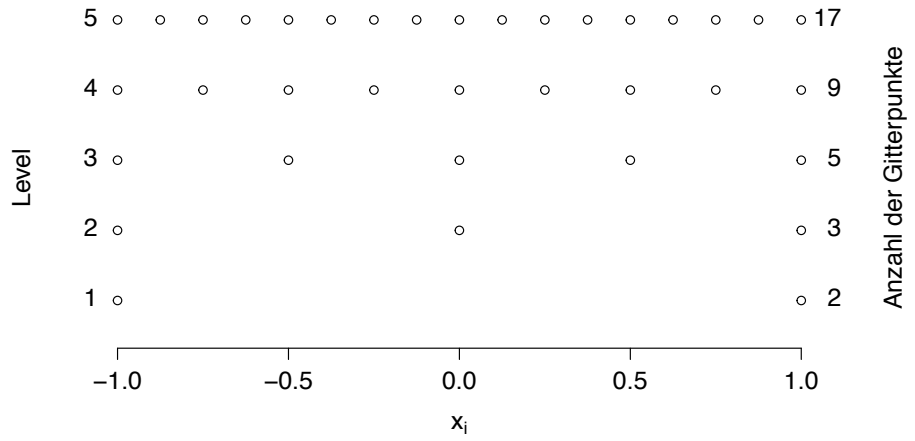

Abbildung B.2.: Stützstellen der Trapezregel für Level 1 bis 5

nur eine vernachlässigbar kleine Wahrscheinlichkeitsmasse liegt. Der Approximationsfehler für das Integral besteht dann zum einen aus dem Fehler der Trapezregel und zum anderen aus den gewählten Abschneidegrenzen a und b, da es die Dichtefunktion für den Bereich $x < a$ und $x > b$ nicht berücksichtigt.

Ein Vorzug der Trapezregel liegt in ihrer allgemeinen Verwendbarkeit. Ein Nachteil ist ihre relativ geringe Genauigkeit. So integriert diese Regel mit n Stützstellen lediglich ein Polynom $(n-1)$-ten Grades exakt. Eine deutlich bessere Genauigkeit erzielen die im Folgenden besprochenen Gauß-Hermite-Quadraturen.

Gauß-Hermite-Quadratur

Die Gauß-Hermite-Quadratur (GHe) gehört zur Gruppe der Gauß-Quadraturen, die sich durch ihre hohe Genauigkeit auszeichnen.

Während bei der Trapezregel relativ wenig Aufmerksamkeit der Wahl der Stützstellen geschenkt wird, werden bei den Gauß-Quadraturen die Stützstellen so gewählt, dass die Quadratur eine maximale Genauigkeit erreicht, gegeben eine feste Anzahl von Stützstellen.

Die Gauß-Hermite-Quadratur ist eine Variante der Gauß-Quadratur, welche für die Approximation unbeschränkter Integrale $\int_{-\infty}^{\infty} f(x)\,dx$ ausgelegt ist. Ihre Stützstellen sind die Nullstellen der Hermite-Polynome und sind gemeinsam mit den Gewichten tabelliert, z.B. in Büchern, wie das von Abramowitz und Stegun (1965), oder in Softwarebibliotheken (Galassi und Gough, 2009).

In Anlehnung an Liu und Pierce (1994) werden die Stützstellen und Gewichte gemäß Gleichung B.1 und B.2 so reskaliert, dass sie die Standardnormalverteilung exakt integrieren.

$$\hat{x}_i = x_i \cdot \sqrt{2} \tag{B.1}$$
$$\hat{w}_i = w_i \cdot e^{x_i^2} \cdot \sqrt{2} \tag{B.2}$$

Wobei x_i und w_i den tabellierten Stützstellen und Gewichten entspricht. Für eine Gauß-Hermite-Quadratur mit drei Stützstellen ergeben sich somit exemplarisch die Koordinaten und Gewichte aus Tabelle B.1.

i	x_i	w_i	$\hat{x}_i$	$\hat{w}_i$
1	-1.22474	0.29541	-1.73205	1.87232
2	0	1.18164	0	1.67109
3	1.22474	0.29541	1.73205	1.87232

Tabelle B.1.: Stützstellen und Gewichte für Gauß-Hermite-Quadratur

Möchte man eine Dichte mit allgemeinem Erwartungswert μ und Varianz σ^2 integrieren, sollten die Stützstellen und Gewichte entsprechend angepasst werden, um eine möglichst genaue Approximation zu erhalten (Liu und Pierce, 1994). Die Anpassung ergibt sich gemäß der Gleichungen B.3 und B.4.

$$\tilde{x}_i = (\hat{x}_i \cdot \sqrt{\sigma^2}) + \mu \tag{B.3}$$
$$\tilde{w}_i = \hat{w}_i \cdot \sqrt{\sigma^2} \tag{B.4}$$

Die so angepasste Gauß-Hermite-Quadratur ist in der Lage, eine eindimensionale Funktion, die einer mit einem Polynom $(2n-1)$-ten Grades multiplizierten Normalverteilung entspricht, exakt zu integrieren. Mit dieser Klasse von Funktionen ist es möglich, allgemeine Dichtefunktionen - auch anderer Verteilungsfamilien - zu approximieren und kann ebenfalls z.B. für die Berechnung von Momenten, wie Erwartungswert und Varianz, eingesetzt werden.

Nicht in allen Fällen gleicht die zu integrierende Funktion dem Muster „Polynom × Normalverteilung“. Daher muss im Einzelfall eventuell nach einer passenderen Regel gesucht werden.

Die Stützstellen der Gauß-Hermite-Quadratur sind für verschiedene Level l nicht genestet, wie in Abbildung B.3 zu sehen ist. Allerdings haben Genz und Keister (1996) eine genestete Variante vorgeschlagen. Abbildung B.4 zeigt die Stützstellen der ersten fünf Level. Der Art der Konstruktion der genesteten Quadraturregel ist geschuldet, dass das Verhältnis von Level zur Anzahl der Stützstellen unregelmäßig ist. Wobei bei der Gauß-Hermite-Quadratur in der Regel gilt: $l = n$.

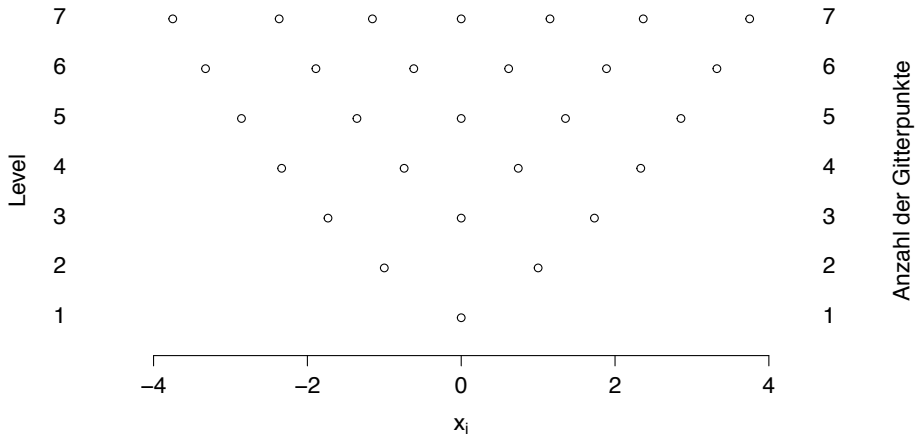

Abbildung B.3.: Stützstellen der ursprünglichen Gauss-Hermite-Quadratur

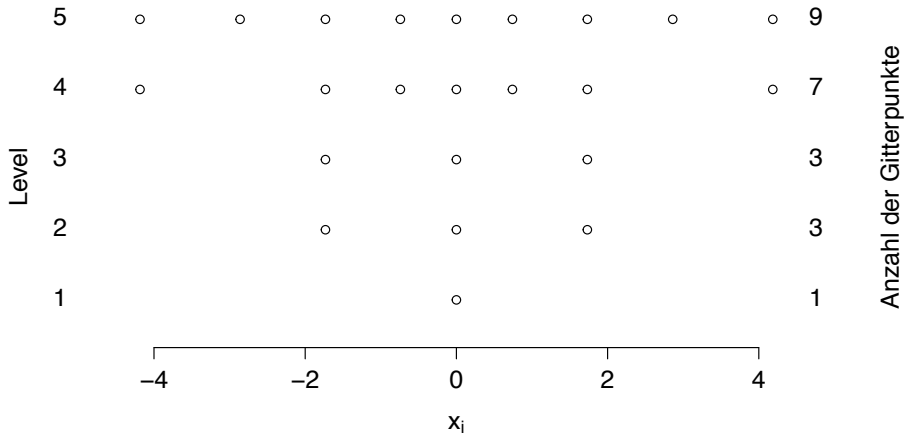

Abbildung B.4.: Stützstellen der genesteten Variante der Gauß-Hermite-Quadratur

Mehrdimensionale Integrale - Produktregel (volle Gitter)

Bei den in dieser Arbeit besprochenen Anwendungen sind die zu berechnenden Integrale Integrale mehrdimensionaler Funktionen. Ausgehend von univariaten Quadraturregeln ergeben sich verschiedene Möglichkeiten für die Erweiterung zu multivariaten Integralen. Eine naheliegende Erweiterung der eindimensionalen Quadraturen auf d-dimensionale Integrale ergibt sich durch die sogenannte Produktregel. Grundlage für diese sind d eindimensionale Quadraturen $(Q^1_{l_1}, Q^1_{l_2}, \ldots, Q^1_{l_d})$, die mit dem kartesischen Produkt zusammengeführt werden.

$$Q^d_l f = (Q^1_{l_1} \otimes \ldots \otimes Q^1_{l_d}) f$$

Hier bezeichnet l den sogenannten Levelvektor, mit $l = \begin{pmatrix} l_1 & \cdots & l_d \end{pmatrix}$.

Gleichung B.5 illustriert das Prinzip für ein zweidimensionales Integral (Davis und Rabinowitz, 1984, Kapitel 5.6).

$$I = \int_{a_1}^{b_1} \int_{a_2}^{b_2} f(x_1, x_2)\, dx_1 dx_2 \approx \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} w_{1,i} \cdot w_{2,j} \cdot f(x_{1,i} x_{2,j}) \tag{B.5}$$

Wobei $(x_{1,i}, w_{1,i})$ und $(x_{2,j}, w_{2,j})$ jeweils die Stützstellen und Gewichte einer eindimensionalen Quadratur bezeichnen. Durch die Doppelsumme werden alle Kombinationen von Koordinaten und Gewichten berücksichtigt.

Aus der Kombination der x_1- und x_2-Koordinaten entsteht ein „Gitter“, wie in Abbildung B.5 dargestellt. In diesem Fall wurde in der ersten Dimension (x_1) eine Gauß-Hermite-Quadratur mit 3 Stützstellen und in der x_2-Dimension eine Gauß-Hermite-Quadratur mit 4 Stützstellen verwendet. Gitter, die gemäß der Produktregel konstruiert werden, werden als „volle Gitter“ bezeichnet.

Wie in der Abbildung B.5 zu sehen ist, besteht dieses Gitter aus insgesamt $12 = 3 \cdot 4$ Gitterpunkten. Für ein allgemeines d-dimensionales Gitter mit $n_1, ..., n_d$ Stützstellen in den entsprechenden eindimensionalen Quadraturen ergeben sich $\Pi_{i=1}^{d} n_i$ Stützstellen. Häufig werden in allen Dimensionen die gleiche Anzahl von Stützstellen gewählt ($n = n_1 = n_2 = ... = n_d$), sodass sich die Zahl der Gitterpunkte als n^d ergibt.

An diesem Ausdruck ist zu erkennen, dass die Anzahl der Stützstellen exponentiell mit der Zahl der Dimensionen wächst. Dieses Phänomen wird als „Fluch der Dimensionen“ (engl.: “curse of the dimensionality“) bezeichnet. Für die praktische Arbeit bedeutet dieser Fluch, dass der Rechenaufwand mit der Anzahl der Dimensionen stark steigt und schließlich für höher- und hochdimensionale Probleme praktisch nicht mehr berechenbar ist.

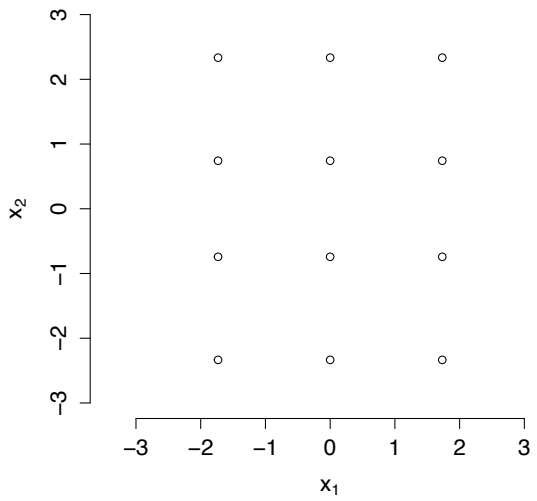

Abbildung B.5.: Volles 2-dimensionales Gitter

C. Dokumentation - `nife`

Im Rahmen der Dissertation wurde das Programmpaket `nife` in der Programmiersprache R entwickelt. In diesem Paket sind die in der Arbeit vorgestellten Algorithmen zusammengefasst.

Auf den folgenden Seiten sind die relevanten Teile der Dokumentation (in englischer Sprache) wiedergegeben.

`NIFilter` *filter for general state-space models*

Description

`NIFilter` computes the filter distributions for a general state-space model

$$y_t = f_t(x_t, e_t)$$

$$x_t = g_t(x_{t-1}, w_t)$$

Usage

```
NIFilter(y, StSp.Obj, Grid, dynamic = NULL, mc.cores = 1)
```

Arguments

`y`	univariate observation vector
`StSp.Obj`	list of functions, determining the state space model; needed elements of the list:
dprior(x)	returns the prior density
dtrans(x, old.x)	returns the transition density
dlike(y,x,t)	returns the likelihood of a scalar obs. for given set of par.vectors (matrix))
`Grid`	a appropriate grid (see: grid1D, fgridnD, sgridnD)
`dynamic`	static grid (NULL, default), exogenous dynamic (sequence of m_t and C_t, for t=1,...,T) or endogenous dynamic (list of options, see example)
`mc.cores`	numbers of cores to be used within the transition step (default=1, no parallelization)

Value

means and covariances of the sequence of posterior distribution; log.likelihood

Examples

```
y <- cumsum(rnorm(30))+rnorm(30)
Grid <- Grid1D("GHe", 10)
Grid[[1]] <- as.matrix(Grid[[1]])
StSp.Obj <- list(dprior = function(x) {dnorm(x, 0, 1)},
                 dlike  = function(x, y, t) {dnorm(y - x, 0, 1)},
                 dtrans = function(x, old.x) {dnorm(x - old.x, 0 , 1)},
                 prop.m = function(m) {m},
                 prop.C = function(C) {C+1})
res.an <- KalmanFilter(y=y, FF=1, m0=0, C0=1, v=1, W=1)
nif.1 <- NIFilter(y, StSp.Obj=StSp.Obj, Grid=Grid, dynamic=NULL)
nif.2 <- NIFilter(y, StSp.Obj=StSp.Obj, Grid=Grid, dynamic=res.an)
```

```
nif.3 <- NIFilter(y, StSp.Obj=StSp.Obj, Grid=Grid, dynamic=list("sa", 10, 1*10^(-5), TRUE))
matplot(cbind(res.an$m, nif.1$m, nif.2$m, nif.3$m), lty=1, type="o", pch=c(20, "1", "2", "3"))
cat("RMSE (static Grid):", sqrt(mean((res.an$m-nif.1$m)^2)), "\n")
cat("RMSE (exogenous dynamic):",sqrt(mean((res.an$m-nif.2$m)^2)), "\n")
cat("RMSE (endogenous dynamic):",sqrt(mean((res.an$m-nif.3$m)^2)), "\n")
```

`AParticleFilter` *Auxiliary Particle Filter for general state-space models*

Description

`AParticleFilter` simulates the filter distribution for a general state-space model

$$y_t = f_t(x_t, e_t)$$
$$x_t = g_t(x_{t-1}, w_t)$$

Usage

```
AParticleFilter(y, StSp.Obj, N)
```

Arguments

`y`	univariate observation vector
`StSp.Obj`	list of functions, determining the state space model (needed elements of the list:
rprior(x)	returns x particles for prior distribution
dprop(old.x)	returns likelihood of propagated particles
rtrans(old.x)	returns particles following the transition
dlike(y,x,t)	returns the likelihood of a scalar obs. for given set of par.vectors (matrix))
`N`	number of particles

Value

means and covariances of the sequence of posterior distribution; log.likelihood

References

M. Pitt, N. Shephard (1999): Filtering via simulation: Auxiliary particle filters

Examples

```
y <- cumsum(rnorm(30))+rnorm(30)
StSp.Obj <- list(rprior=function(N){rnorm(N,0,2)},
                 dprop=function(y, x, t){dnorm(y-x,0,1)},
                 rtrans= function(x){x + rnorm(length(x), 0, 0.1)},
                 dlike=function(y, x, t){dnorm(y-x, 0, 1)})
res.APF <- AParticleFilter(y, StSp.Obj, 10^5)
res.Kalman <- KalmanFilter(y, m0=0, C0=2^2, W=0.1, v=1)
matplot(cbind(res.APF$m, res.Kalman$m), type="o", pch=20)
```

`BootstrapFilter` *Bootstrap filter for general state-space models*

Description

`BootstrapFilter` simulates the filter distribution for a general state-space model

$$y_t = f_t(x_t, e_t)$$

$$x_t = g_t(x_{t-1}, w_t)$$

Usage

```
BootstrapFilter(y, StSp.Obj, N)
```

Arguments

`y`	univariate observation vector
`StSp.Obj`	list of functions, determining the state space model (needed elements of the list:
rprior(x)	returns x particles for prior distribution
rtrans(old.x)	returns particles following the transition
dlike(y,x,t)	returns the likelihood of a scalar obs. for given set of par.vectors (matrix))
`N`	number of particles

Value

means and covariances of the sequence of posterior distribution; log.likelihood

References

N. Gordon, D. Salmond, A. Smith (1993): Novel approach to nonlinear/non-Gaussian Bayesian state estimation

Examples

```
y <- cumsum(rnorm(30))+rnorm(30)
StSp.Obj <- list(rprior=function(N){rnorm(N,0,2)},
                 rtrans= function(x){x + rnorm(length(x), 0, 0.1)},
                 dlike=function(y, x, t){dnorm(y-x, 0, 1)})
res.BF <- BootstrapFilter(y, StSp.Obj, 10^5)
res.Kalman <- KalmanFilter(y, m0=0, C0=2^2, W=0.1, v=1)
matplot(cbind(res.BF$m, res.Kalman$m), type="o", pch=20)
```

`KalmanFilter` *Kalman-Filter for linear and normal state-space models*

Description

`KalmanFilter` Computes the prediction and filtering density via the Kalman recursion for a linear and normal state-space model with a univariate observation and a poptential multivariate state equation.

$$y_t = ff + FF_t * x_t' + e_t; \quad e_t \ N(0, v)$$

$$x_t = gg + GG_t * x_{t-1}' + w_t; \quad w_t \ N(0, W)$$

Usage

```
KalmanFilter(y, ff = 0, FF = NULL, gg = 0, GG = NULL, m0,
  C0, v, W)
```

Arguments

`y`	vector of univariate observations
`ff`	constant in the observation equation
`FF`	coefficients of the observation equation
	`1 x p vector` constant over time (vector) (FF)
	`T x p matrix` time variant (FF_t)
`gg`	constant in the state equation
`GG`	coefficients of the state equation (p x p matrix; constant over time (matrix))
`m0`	prior mean
`C0`	prior covariance matrix
`v`	variance of noise-distribution
`W`	covariance matrix of innovation-distribution (scalar or matrix)

Value

sequence of prior-, prediction- and posterior-densities (`(a, R); (f, Q); (m, C)`) and the log-likelihood of the model

References

Andy Pole, Mike West, Jeff Harrison (1994): Applied Bayesian Forecasting and Time Series Analysis, (p. 43)

Examples

```
y = cumsum(rnorm(20))+rnorm(20) # artificial time series
kf <- KalmanFilter(y, m0=0, C0=5, v=1, W=1)
matplot(cbind(kf$m + 2*sqrt(unlist(kf$C)),kf$m - 2*sqrt(unlist(kf$C))),
    type="l", lty=1, col="gray", ylab="", xlab="")
lines(y, type="p")
lines(kf$m, type="l")
```

`Grid1D` *generates a 1D-grid for a given quadrature rule*

Description

`Grid1D` Generates a 1D-grid, containing nodes and weights for one of the predefined quadrature rules.

Usage

```
Grid1D(type, k = 1)
```

Arguments

`type`	predefined quadrature rule
	`cNC1, cNC2, ..., cNC6` closed Newton-Cotes Formula of degree 1-6 (1=trapezoidal-rule; 2=simpson-rule; ...), interval of integration: [0, 1]
	`oNC0, oNC1, ..., oNC3` open Newton-Cote Formula of degree 0-3 (0=midpoint-rule; ...), interval of integration: [0, 1]
	`GLe, GKr` Gauss-Legendre and Gauss-Kronrod rule for an interval of integration: [-1, 1]
	`nLe` nested Gauss-Legendre rule for an interval of integration: [-1, 1] (Knut Petras (2003). Smolyak cubature of given polynomial degree with few nodes for increasing dimension. Numerische Mathematik 93, 729-753)
	`GLa` Gauss-Laguerre rule for an interval of integration: [0, INF)
	`GHe` Gauss-Hermite rule for an interval of integration: (-INF, INF)
	`nHe` nested Gauss-Hermite rule for an interval of integration: (-INF, INF) (A. Genz and B. D. Keister (1996). Fully symmetric interpolatory rules for multiple integrals over infinite regions with Gaussian weight."Journal of Computational and Applied Mathematics 71, 299-309)
`k`	number of repetitions for compounted Newton-Codes Formulas, number of gridpoints for Gauss-Quadrature

Value

1D-grid of type `NIGrid`

References

Philip J. Davis, Philip Rabinowitz (1984): Methods of Numerical Integration

Examples

```
N <- 5
plot(0,yaxt="n",bty="n",pch="",ylab="degree",xlab="x", ylim=c(1,N), xlim=c(0,1))
for (i in 1:N){
  k <- 2^i
  nw <- Grid1D("cNC1", k )
  points(nw[[1]],rep(i, length(nw[[1]])))
  text(x=-0.03,y=i, labels=k)
}
```

`fGridnD` *generates a multidinensional full grid for a given quadrature rule*

Description

`fGridnD` generates a nD-grid, containing nodes and weights for a given quadrature Rule. The extension from 1D to nD is done with the product rule (kronecker product). The result is the so called "full"grid.

Usage

```
fGridnD(dim, type, k)
```

Arguments

`dim`	number of dimensions
`type`	predefined quadrature rule (see `Grid1D`)
`k`	number of repetitions for compounted Newton-Codes Formulas, number of gridpoints for Gauss-Quadrature

Details

Normally `fGridnD` is used to create symmetric grids. But it is possible to create asymmetric grids (see the example below).

Value

nD-grid

References

Philip J. Davis, Philip Rabinowitz (1984): Methods of Numerical Integration

Examples

```
par(mfrow=c(2,1), mar=c(4,4,2,0.5), oma=c(0,0,0,0))
# symmetric grid
nw <- fGridnD(2, "GHe", 10)
plot(nw[[1]], xlab="", ylab="", main="Symmetric Grid")
# asymmetric grid
nw <- fGridnD(2, c("GHe","cNC1"), c(10,4))
plot(nw[[1]], xlab="GHe-dimension", ylab="cNC1-Dimension", main="Asymmetric Grid")
```

`sGridnD` *generates a multidinensional sparse grid for a given quadrature rule*

Description

`sGridnD` generates a nD-grid, containing nodes and weights for a given quadrature Rule. The extension from 1D to nD is done with the combination technique. The result is the so called sparse grid.

Usage

```
sGridnD(dim, type, k, NClevel = FALSE)
```

Arguments

`dim`	number of dimensions
`type`	predefined quadrature rule (see `Grid1D`)
`k`	max. level
`NClevel`	logical variable denotes either to take the levels as number of grid points (FALSE=default) or to transform in that manner that number of grid points = 2^(levels-1) (TRUE). This is needed to achieve some convergence properties.

Value

nD-grid

References

H.-J. Bungartz, M. Griebel (2004): Sparse grids, Acta Numerica

F. Heiss, V. Winschel (2008): Likelihood approximation by numerical integration on sparse grids, Journal of Econometrics

Examples

```
par(mfrow=c(2,1), mar=c(2,2,3,0.5), oma=c(0,0,0,0))
require(mvtnorm)
# full grid
nw <- fGridnD(2, "cNC1", 4)
plot(nw[[1]], main="full grid")
cat("\n approx. error:", pmvnorm(c(0,0), upper=c(1,1))[1]-sum(dmvnorm(nw[[1]])*nw[[2]]),
    "  with", length(nw[[2]]), "gridpoints")
# sparse grid
nw <- sGridnD(2, "cNC1", 4, TRUE)
plot(nw[[1]], main="sparse grid")
cat("\n approx. error:", pmvnorm(c(0,0), upper=c(1,1))[1]-sum(dmvnorm(nw[[1]])*nw[[2]]),
    "  with", length(nw[[2]]), "gridpoints")
```

MoveGrid *moves, rescales and rotates a multidimensional grid.*

Description

`MoveGrid` manipulates a grid for more efficent numerical integration with respect to a given vector of means and covariance matrix.

Usage

```
MoveGrid(initial.grid, m, C, dec.type = 0)
```

Arguments

`initial.grid`	grid to be manipulated, of type `NIGrid`
`m`	vector of means
`C`	covariance matrix
`dec.type`	type of covariance decomposition
	`dec.type=0` (default) no decomposition, just rescaling
	`dec.type=1` eigenvalue-decomposition (spectral-decomposition), rotates grid
	`dec.type=2` cholesky decomposition, shears grid

Value

manipulated grid of type `NIGrid`

References

Peter Jaeckel (2005): A note on multivariate Gauss-Hermite quadrature

Examples

```
require(mvtnorm)
C = matrix(c(2,0.9,0.9,2),2)
m = c(-.5, .3)
nw <- fGridnD(dim=2,type="GHe",k=5)

# manipulate grid not with repect to the true parameters but an estimate
nw.m <- MoveGrid(initial.grid=nw, m=c(0,0), C= matrix(c(1.9,0.8,0.8,2),2), dec.type=1)
print(sum(dmvnorm(nw.m[[1]], m, C)* nw.m[[2]]))

# plot manipulated grid
xx <- expand.grid(seq(-4,4,length.out=100),seq(-4,4,length.out=100))
y <- dmvnorm(xx, m, C)
y <- matrix(y,100)
par(mfrow=c(3,1))
contour(seq(-4,4,length.out=100),seq(-4,4,length.out=100),y,asp=1)
nw.0 <- MoveGrid(initial.grid=nw, m, C, dec.type=0)
points(nw.0[[1]], col="red")
contour(seq(-4,4,length.out=100),seq(-4,4,length.out=100),y,asp=1)
nw.1 <- MoveGrid(initial.grid=nw, m, C, dec.type=1)
points(nw.1[[1]], col="green")
contour(seq(-4,4,length.out=100),seq(-4,4,length.out=100),y,asp=1)
nw.2 <- MoveGrid(initial.grid=nw, m, C, dec.type=2)
points(nw.2[[1]], col="blue")
```

Literatur

Abramowitz, M. und I. Stegun (1965). *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*. Dover Publications.

Arasaratnam, I., S. Haykin und R.J. Elliott (2007). "Discrete-time nonlinear filtering algorithms using Gauss-Hermite quadrature". In: *Proceedings of the IEEE*.

Athans, M., R.P. Wishner und A. Bertolini (1968). "Suboptimal state estimation for continuous-time nonlinear systems from discrete noisy measurements". In: *Automatic Control, IEEE Transactions on* 13.5, S. 504–514.

Basdevant, O. (2003). "On applications of state-space modelling in macroeconomics".

Bolviken, E. und G. Storvik (2001). "Deterministic and stochastic particle filters in state-space models". In: *Sequential Monte Carlo Methods in Practice*. Hrsg. von Arnaud Doucet, Nando Freitas und Neil Gordon. Springer New York, S. 97–116.

Broto, C. und E. Ruiz (2004). "Estimation methods for stochastic volatility models: a survey". In: *Journal of Economic Surveys* 18.5, S. 613–650.

Bungartz, H.J. und M. Griebel (2004). "Sparse grids". In: *Acta Numerica* 13, S. 1–123.

Chen, Z. (2003). "Bayesian filtering: From Kalman filters to particle filters, and beyond (Manusscript)".

Crisan, D. und A. Doucet (März 2002). "A survey of convergence results on particle filtering methods for practitioners". English. In: *IEEE Transactions on Signal Processing* 50.3, S. 736–746.

Davis, P.J. und P. Rabinowitz (1984). *Methods of numerical integration*. 2nd Editio. Academic Press.

Doucet, A. u. a. (2001). *Sequential Monte Carlo Methods in Practice (Statistics for Engineering and Information Science)*. Springer.

Fridman, M. und L. Harris (1998). "A Maximum Likelihood Approach for Non-Gaussian Stochastic Volatility Models". In: *Journal of Business & Economic Statistics* 16.3, S. 284–291.

Galassi, M. und B. Gough (2009). *GNU scientific library: reference manual*. Hrsg. von Brian Gough. 3rd.

Genz, A. und B.D. Keister (1996). "Fully symmetric interpolatory rules for multiple integrals over infinite regions with Gaussian weight". In: *Journal of Computational and Applied Mathematics* 71, S. 299–309.

Ghysels, E., A.C. Harvey und E. Renault (1996). "Stochastic volatility".

Gordon, N.J., D.J. Salmond und A.F.M. Smith (1993). "Novel approach to nonlinear/non-Gaussian Bayesian state estimation". In: *Radar and Signal Processing IEE Proceedings F* 140.2, S. 107–113.

Greenberg, Edward (2013). *Introduction to Bayesian econometrics*. 2nd. Cambridge University Press.

Griebel, M., M. Schneider und C. Zenger (1990). *A combination technique for the solution of sparse grid problems*. Technische Universität München.

Hamilton, James D. (1994). "State-space models". In: *Handbook of Econometrics*. Hrsg. von Robert F. Engle und Daniel L. McFadden. 1. Aufl. Bd. null. null. Elsevier. Kap. 50.

Heiss, F. (2008). "Sequential numerical integration in nonlinear state space models for microeconometric panel data". In: *Journal of Applied Econometrics* 23.3, S. 373–389.

Heiss, F. und V. Winschel (2008). "Likelihood approximation by numerical integration on sparse grids". In: *Journal of Econometrics* 144.1, S. 62–80.

Ionides, E. u. a. (Juni 2011). "Iterated filtering". EN. In: *The Annals of Statistics* 39.3, S. 1776–1802.

Jäckel, P. (2005). "A note on multivariate Gauss-Hermite quadrature".

Julier, S.J. und J.K. Uhlmann (1997). "A New Extension of the Kalman Filter to Nonlinear Systems". In: *AeroSense'97. International Society for Optics and Photonics*, S. 182–193.

Kalman, R.E. (1960). "A new approach to linear filtering and prediction problems". In: *Journal of basic Engineering* 82.Series D, S. 35–45.

Kitagawa, G. (1987). "Non-Gaussian State-Space Modeling of Nonstationary Time Series". In: *Journal of the American statistical association* 82.400, S. 1032–1041.

– (1996). "Monte Carlo filter and smoother for non-Gaussian nonlinear state space models". In: *Journal of Computational and Graphical Statistics* 5.1, S. 1–25.

– (1998). "A Self-Organizing State-Space Model". In: *Journal of the American Statistical Association* 93.443, S. 1203–1215.

Kohn, Robert und Craig F. Ansley (Nov. 1987). "Non-Gaussian State-Space Modeling of Nonstationary Time Series: Comment". EN. In: *Journal of the American Statistical Association* 82.400, S. 1041–1044.

Liu, Q. und D.A. Pierce (1994). "A note on Gauss-Hermite quadrature". In: *Biometrika* 81.3, S. 624–629.

Lopes, H.F. und R.S. Tsay (2011). "Particle Filters and Bayesian Inference in Financial Econometrics". In: *Journal of Forecasting* 30.1, S. 168–209.

Malik, S. und M. K. Pitt (Dez. 2011). "Particle filters for continuous likelihood evaluation and maximisation". In: *Journal of Econometrics* 165.2, S. 190–209.

Marin, J.M. und C. P. Robert (2007). *Bayesian Core: A Practical Approach to Computational Bayesian Statistics*. Springer.

Martin, R. Douglas und Adrian E. Raftery (Nov. 1987). "Non-Gaussian State-Space Modeling of Nonstationary Time Series: Comment: Robustness, Computation, and Non-

Euclidean Models". EN. In: *Journal of the American Statistical Association* 82.400, S. 1044–1050.

Naylor, J.C. und A.F.M. Smith (1982). "Applications of a Method for the Efficient Computation of Posterior Distributions". In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 31.3, S. 214–225.

Petris, G., P. Campagnoli und S. Petrone (2009). *Dynamic linear models with R*. Springer.

Pflüger, D. (2010). *Spatially Adaptive Sparse Grids for High-Dimensional Problems Spatially Adaptive Sparse Grids for High-Dimensional Problems Pflüger*. Verlag Dr. Hut.

Pitt, M.K. und N. Shephard (1999). "Filtering via simulation: Auxiliary particle filters". In: *Journal of the American Statistical Association* 94.446, S. 590–599.

Pole, A., M. West und J. Harrison (1994). *Applied Bayesian forecasting and time series analysis*. Chapman & Hall / CRC.

Rebeschini, Patrick und Ramon Van Handel (2013). "Can local particle filters beat the curse of dimensionality?" In: *arXiv preprint arXiv:1301.6585*, S. 1–64. arXiv: `arXiv:1301.6585v1`.

Reilly, P.M. (1976). "The Numerical Computation of Posterior Distributions in Bayesian Statistical Inference". In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 25.3, S. 201–209.

Robert, C. und G. Casella (2010). *Introducing Monte Carlo Methods with R*. Springer.

Robert, C.P. und G. Casella (2004). *Monte Carlo Statistical Methods*. Springer Texts in Statistics. Springer.

Roberts, G.O. und J.S. Rosenthal (2004). "General state space Markov chains and MCMC algorithms". In: *Probability Surveys* 1, S. 20–71.

Smith, A.F.M. u. a. (1987). "Progress with Numerical and Graphical Methods for Practical Bayesian Statistics". In: *Journal of the Royal Statistical Society. Series D (The Statistician)* 36.2, S. 75–82.

Smolyak, SA (1963). "Quadrature and interpolation formulas for tensor products of certain classes of functions". In: *Soviet Mathematics* 4, S. 240–243.