Reinhold Kainhofer
reinhold.kainhofer@kfunigraz.ac.at

Die numerische Simulation von Transportgleichungen mittels Quasi-Monte Carlo Methoden

Grundlagen und Simulation am Beispiel der Boltzmann-Gleichung

Diplomarbeit aus
Technische Mathematik
Sudienzweig Technomathematik

durchgeführt am

Institut für Mathematik A
Technische Universität Graz
bei O.Univ.Prof. Dr. Robert F. Tichy

Graz, im Juli 2000

Ich versichere, diese Arbeit selbstständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubter Hilfsmittel bedient zu haben.

Zusammenfassung

Diese Diplomarbeit beschäftigt sich mit dem Thema der Quasi-Monte Carlo Simulation von Transportgleichungen, im speziellen der Boltzmann-Gleichung. Quasi-Monte Carlo Methoden benutzen möglichst gleichverteilte, dafür aber deterministische Folgen, sogenannte Folgen kleiner Diskrepanz, und stellen damit ein deterministisches Pedant zu Monte Carlo Verfahren dar.

Es wurden alle Schemata (Bird's DSMC-B, Nanbu's DSMC-N, Lécot's LD Schemata, sowie Lécot's QMC1 und QMC3 Schemata) implementiert und verglichen. Es zeigt sich auch hier, dass Quasi-Monte Carlo Verfahren einen geringeren Fehler bei der Simulation liefern als Monte Carlo Methoden.

Außerdem wird eine bei weitem bessere Diskrepanzschranke der Halton-Folge, die von Atanassov bewiesen wurde, erläutert und dieses Ergebnis auch numerisch an einigen Beispielen untersucht.

Inhaltsverzeichnis

1 Quasi-Monte Carlo (QMC) Simulation **1**
- 1.1 Beispiel . . . 2
- 1.2 Diskrepanz . . . 3
- 1.3 Fehlerschranken . . . 7

2 Folgen kleiner Diskrepanz **9**
- 2.1 Die $(n\alpha)$ Folgen . . . 10
- 2.2 Digitale Folgen (VAN DER CORPUT, HALTON) . . . 11
- 2.3 (t,s)-Folgen . . . 25
- 2.4 Existenz von (t, s)-Folgen . . . 31
- 2.5 Erzeugung von (t, m, s)-Netzen und (t, s)-Folgen . . . 31

3 Die Boltzmann-Gleichung **38**
- 3.1 Teilchenkollisionen . . . 39
- 3.2 Physikalische "Herleitung" . . . 40
- 3.3 Liouville-Gleichung und BBGKY-Hierarchie . . . 42
- 3.4 Einschränkungen und Verallgemeinerungen . . . 44
- 3.5 Existenz und Eindeutigkeit der Lösung . . . 44
- 3.6 Lösungen der Boltzmann-Gleichung . . . 47
- 3.7 Schwache Variante der Boltzmann-Gleichung . . . 47
- 3.8 Master-Gleichung von KAC . . . 49

4 Simulation der Boltzmann-Gleichung **50**
- 4.1 Entkopplung . . . 51
- 4.2 Notation . . . 51
- 4.3 Die Anfangswerte . . . 52
- 4.4 Schema von BELOTSERKOWSKIY-YANITSKIY . . . 53
- 4.5 BIRD's DSMC-B und NTC Methoden . . . 53
- 4.6 NANBU's DSMC-N Schema . . . 54
- 4.7 LÉCOT's LD Methode . . . 59
- 4.8 LÉCOT's QMC1 Schema . . . 62

5 Numerische Resultate **72**
- 5.1 Die Simulation . . . 72
- 5.2 Überblick über alle Simulationsmethoden . . . 73
- 5.3 Zahl der Teilchenkollisionen . . . 84
- 5.4 3-dimensionale vs. 1-dimensionale Simulation . . . 84
- 5.5 Bedeutung des Umordnens der Teilchen . . . 85
- 5.6 MC vs. QMC Simulation . . . 86

5.7 Direkte Vergleiche von Halton- und $(0, s)$-Folgen 86

Source-Code **90**

Abbildungsverzeichnis

1.1 Fehler bei der Simulation der WLG in einer und in 3 Dimensionen 3

5.1 Fehler-Vergleich aller Schemata, $N = 29282$, $\Delta t = 0.015$ 74
5.2 Die wichtigsten Schemata mit Krook / Wu's Lösung, $\Delta t = 0.015$ 76
5.3 Zeitaufwand der Schemata mit Krook / Wu's Lösung, $N = 2662$, $\Delta t = 0.015$. 79
5.4 Fehlerexponent für Krook / Wu's Lösung, $\Delta t = 0.015$ 81
5.5 Fluktuationen der verschiedenen Schemata für die Boltzmann-Verteilung und $\Delta t = 0.015$ 82
5.6 Fehlerexponent der verschiedenen Schemata für die Boltzmann-Verteilung und $\Delta t = 0.015$ 83
5.7 Zahl der Teilchenkollisionen der wichtigsten Schemata, $N = 2662$ 84
5.8 Verbesserung des Fehlers durch Umordnen, QMC1, $N = 29282$ 85
5.9 Verbesserung des Fehlers durch Umordnen, QMC3, $N = 29282$ 85
5.10 Vergleich der Simulation mit Halton- und $(0, s)$-Folge, $N = 29282$ 88

Tabellenverzeichnis

5.1 Alle implementierten Schemata 73
5.2 Fehler der diversen Methoden bei $M = 100$, $T = 1.5$ und Krook / Wu's Lösung 77
5.3 Fehlerexponent bei $M = 100$, $T = 1.5$ und Krook / Wu's Lösung 77
5.4 Fehler für verschiedene Δt, Krook / Wu's Lösung, $N = 29282$ 78
5.5 Zeitaufwand der diversen Methoden bei $M = 100$, $T = 1.5$ und Krook / Wu's Lösung 79
5.6 Fluktuationen für die Boltzmann-Verteilung und $\Delta t = 0.015$ 80
5.7 Verbesserung durch QMC Methoden beim QMC1 Schema, Krook / Wu's Lösung, $N = 2662$ 87
5.8 Vergleich der QMC und MC Methoden beim QMC3 Schema, Krook / Wu's Lösung, $N = 2662$ 87
5.9 Fehler durch (Q)MC Integration der Testfunktion, $N = 10000$ (Quasi-) Zufallszahlen 89
5.10 Fehler durch (Q)MC Integration der ausgewählter Testfunktionen, $N = 10000$ (Quasi-) Zufallszahlen 89

Vorwort

Trotz der meist besseren Ergebnisse und einer absoluten oberen Fehlerschranke werden Quasi-Monte Carlo Methoden nach wie vor nur sehr selten zur Simulation benutzt. Nachdem in den 60er Jahren des 20. Jahrhunderts die ersten Folgen kleiner Diskrepanz bekannt wurden, gibt es mittlerweile unzählige Konstruktionen, die alle entweder auf die Halton-Folge oder auf (t, s)-Folgen zurückgeführt werden können.

In dieser Diplomarbeit möchte ich nach einer kurzen Übersicht im ersten Kapitel über die die theoretischen Grundlagen der Folgen kleiner Diskrepanz im zweiten Kapitel die bekanntesten Folgen kleiner Diskrepanz vorstellen und Diskrepanzabschätzungen geben. Zudem werde ich eine vor kurzem erst von Atanassov vorgestellte und bewiesene Diskrepanzabschätzung der Halton-Folge anführen, welche die Bevorzugung vieler Mathematiker der (t, s)-Folgen gegenüber Halton-Folgen keineswegs zu rechtfertigen scheint.

Im dritten Kapitel schließlich werde ich näher auf die physikalischen und mathematischen Hintergründe der Transportgleichung, auch nach ihrem "Entdecker" als Boltzmann-Gleichung bezeichnet, näher eingehen. Diese Gleichung beschreibt unter anderem das Verhalten verdünnter Gase, berücksichtigt aber keine relativistischen oder quantenmechanischen Effekte. Das vierte Kapitel schließlich ist der Vorstellung der verschiedenen bekannten direkten Simulationsschemata gewidmet, die im abschließenden fünften Kapitel auch numerisch implementiert und verglichen werden. Zudem werde ich im letzten Kapitel auch direkte Vergleiche von Halton- und (t, s)-Folgen durchführen, motiviert durch die neue Diskrepanzschranke von Atanassov für Halton-Folgen.

An dieser Stelle möchte ich mich bei Prof. Tichy für die Betreuung dieser Arbeit bedanken, ebenso bei Ch. Lécot für einige Hinweise zur QMC Simulation der Boltzmann-Gleichung.

Meinen Eltern, die während meines Studiums immer für mich gesorgt und mich unterstützt haben, möchte ich besonders danken.

Ebenso gilt mein Dank all den vielen, die auf verschiedene Weise zum Entstehen der Arbeit beigetragen haben.

Graz, 1. August 2000

Reinhold Kainhofer

Kapitel 1

Quasi-Monte Carlo (QMC) Simulation

Inhaltsangabe

1.1 Beispiel **2**
1.2 Diskrepanz **3**
1.3 Fehlerschranken **7**

Oft werden Quasi-Monte Carlo Verfahren (im Folgenden kurz als QMC Verfahren bezeichnet) einfach als analytische Versionen des Monte Carlo Verfahrens beschrieben. In gewissem Sinne stimmt dies auch: Prinzipiell wird zuerst eine Diskretisierung der zu simulierenden Gleichung durchgeführt, wobei dann die Koeffizienten c_i in der diskretisierten Gleichung als Übergangswahrscheinlichkeiten interpretiert werden und je nachdem, ob eine Zufallszahl x_j im Intervall $\left[\sum_{k=0}^{i} c_k, \sum_{k=0}^{i+1} c_k\right)$ liegt, wird dann der entsprechende Schritt durchgeführt.

Bei Monte Carlo Verfahren, wo wirkliche (Pseudo-) Zufallszahlen verwendet werden, kann gezeigt werden, dass die so erhaltene Lösung zur exakten Lösung konvergiert, und zwar mit einer Konvergenzordnung von $O(\frac{1}{N^s})$. Allerdings kann für den Fehler auch nur ein Erwartungswert angegeben werden, und keine obere Schranke.

Quasi-Monte Carlo Verfahren verwenden anstatt der Zufallszahlen analytische Folgen mit möglichst guter Gleichverteilung, sodass auch für den Fehler eine obere Schranke angegeben werden kann. konnte LÉCOT an einigen Beispielen ([Léc89a], [Léc89b], [Léc91] etc.) zeigen, dass Simulation mittels Folgen kleiner Diskrepanz zu geringeren Fehlern führt als bei Monte Carlo oder anderen numerischen Verfahren. Die dabei verwendeten Folgen sollten natürlich so gut wie möglich gleichverteilt sein, es treten allerdings starke Korrelationen auf. Im Unterschied zu Monte Carlo Methoden werden nun die simulierten Teilchen bzw. die Dirac-Maße zur Näherung der gesuchten Lösung nach jedem Zeitschritt umgeordnet ([Léc89a]), wodurch diese Korrelationen gebrochen werden können.

Einen weiteren Vorteil haben QMC Verfahren: Bei Verwendung derselben Folge und derselben Parameter erhält man immer dieselbe Lösung, also Reproduzierbarkeit. Im Extremfall kann es lediglich passieren, dass eine schlechte Folge gewählt wurde, was allerdings durch die obere Fehlerschranke wieder relativiert wird.

Wo nicht anders angegeben, stammen die Definitionen und Sätze dieses und des nächsten Kapitels sämtlich aus [Nie78] oder [Nie92].

1.1 Beispiel

Als ein einfaches Beispiel einer Quasi-Monte Carlo Simulation soll nun die s-dimensionale Diffusionsgleichung mit einsperiodischen Randbedingungen kurz behandelt werden ([Rad95] und [Kai99]):

$$\partial_t u(x_1, \ldots x_s, t) = \sum_{i=1}^{s} \partial_{x_i x_i} u(x_1, \ldots, x_s, t) \tag{1.1a}$$

$$u(x_1, \ldots, x_s, 0) = u_0(x, y) \tag{1.1b}$$

$$u(\ldots, x_i + 1, \ldots, t) = u(x, y, t) \quad \text{für } i = 1, \ldots, s \tag{1.1c}$$

Wie aus der numerischen Mathematik bekannt, kann diese Gleichung durch einen einfachen Euler-Vorwärtsschritt diskretisiert werden

mit Ortsschritt Δx, Zeitschritt Δt und $\lambda := \frac{\Delta t}{\Delta x^2}$. Im Fall $n = 2$ sieht dieses Schema etwa folgendermaßen aus:

$$u_{ij}^{(n+1)} = (1 - 4\lambda) u_{ij}^{(n)} + \lambda \left(u_{i-1,j}^{(n)} + u_{i+1,j}^{(n)} + u_{i,j-1}^{(n)} + u_{i,j+1}^{(n)} \right) \tag{1.2a}$$

$$u_{ij}^{(0)} = u_0 \left(i\Delta x, j\Delta y \right) \tag{1.2b}$$

$$u_{i+M,j}^{(n)} = u_{i,j}^{(n)}, u_{i,j+M}^{(n)} = u_{ij}^{(n)} \text{ (Einsperiodizität)} \tag{1.2c}$$

Unter der Courant-Friedrich-Levi-Bedingung $\lambda \leq \frac{1}{4}$ ist dieses Schema stabil. Dieses Schema kann gleich simuliert werden, indem die $u_{ij}^{(n+1)}$ gemäß obiger Formel berechnet werden.

Zur Monte Carlo und QMC Simulation allerdings wird als Anfangsbedingung an jeden Gitterpunkt eine bestimmte Anzahl von "Teilchen" gesetzt, die dann entsprechend obiger Diskretisierung bewegt werden. Dabei werden die Vorfaktoren der $u^{(n)}$ als die Übergangswahrscheinlichkeiten für das Teilchen interpretiert und jedes einzelne Teilchen dementsprechend bewegt (x ist hier die benutzte Zufallszahl für das jeweilige Teilchen):

$0 \leq x_{ij} < \lambda$	nach links
$\lambda \leq x_{ij} < 2\lambda$	nach rechts
$2\lambda \leq x_{ij} < 3\lambda$	nach unten
$3\lambda \leq x_{ij} < 4\lambda$	nach oben
$4\lambda \leq x_{ij} < 1$	Teilchen bleibt an der Stelle (i, j)

Bei der Monte Carlo Simulation wird wirklich für jedes Teilchen eine Pseudo-Zufallszahl benutzt, während bei der QMC Simulation die Elemente einer bestimmten Folge kleiner Diskrepanz benutzt werden.

Abbildung 1.1 zeigt einen Vergleich der Ergebnisse der ein- und der dreidimensionalen Wärmeleitungsgleichung. Während in einer Dimension noch eine deutliche Verbesserung durch QMC Verfahren gegenüber der MC Simulation erkennbar ist, existiert in drei Dimensionen (die Simulation geschieht nach wie vor nach oben beschriebenem Schema durch eine eindimensionale Folge) praktisch kein Unterschied im Fehler mehr erkennbar.

Allgemein kann meist eine diskretisierte Gleichung wie (1.2a) geschrieben werden in der Form [LC96a]

$$g^{(n)}(\mathbf{r}) = \frac{1}{N} \sum_{j=0}^{N-1} \left((1 - p_j) \delta(\mathbf{r} - \mathbf{x_j^{(n)}}) + Q_j(\mathbf{r}) \right) \tag{1.3}$$

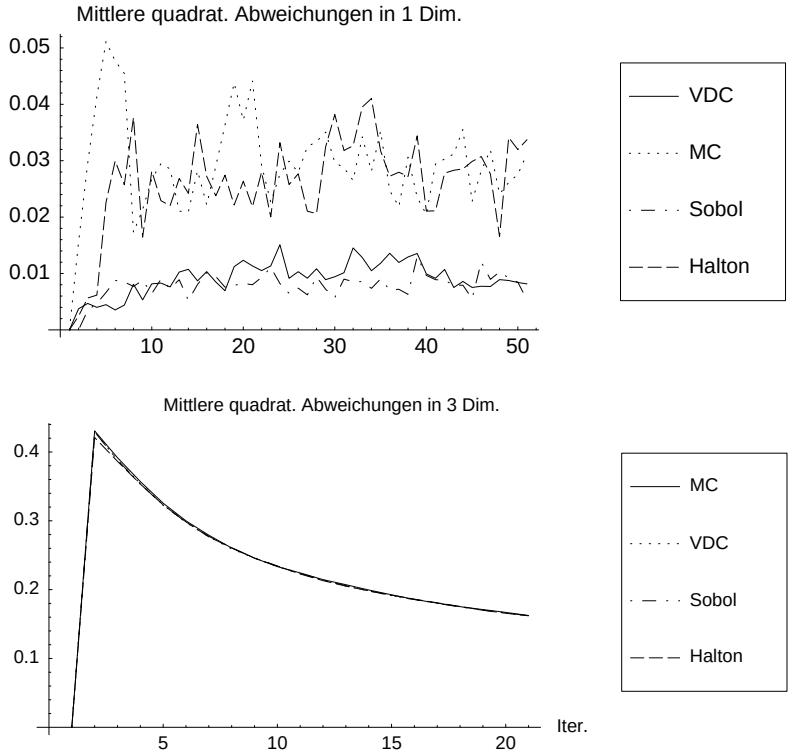

Abbildung 1.1: Fehler bei der Simulation der WLG in einer und in 3 Dimensionen

wobei Q_j eine Summe von Wahrscheinlichkeitsmaßen $Q_{j,j'}$ ist. p_j stellt dabei die Wahrscheinlichkeit einer Kollision des j-ten Teilchens dar. Ist eine (Quasi) Zufallszahl $y_j > p_j$, so bleibt $\mathbf{r}_j$ unverändert, ansonsten wird ein Kollisionspartner entsprechend den $Q_{j,j'}$ und der Wert von $\mathbf{r}_j$ nach der Kollision bestimmt.

1.2 Diskrepanz

Wie schon erwähnt ist für die verwendeten Folgen nicht so sehr eine zufällige Verteilung nötig, als vielmehr eine möglichst gute Gleichverteilung. Als Maß dafür dient die sogenannte Diskrepanz, welche beschreibt, welcher maximale Fehler bei der Näherung des Maßes einer Familie von Mengen durch die Zählfunktion entsteht.

Für eine beliebige Teilmenge $B \in \bar{I}^s$ des Einheitsintervalls in s Dimensionen und eine Menge P von N Punkten $\mathbf{x}_1, \ldots \mathbf{x}_N$ in $\bar{I}^s$ sei

$$A(B;P) := \sum_{i=1}^{N} \chi_B(\mathbf{r}_n) \tag{1.4}$$

wobei χ_B die charakteristische Funktion von B darstellt. $A(B;P)$ zählt also, wie viele Punkte von P sich in B befinden und $\frac{A(B;P)}{N}$ ist damit ein Schätzer für das Lebesque-Maß von B.

Definition 1.1 (Diskrepanz). Für eine allgemeine Familie $\mathfrak{B}$ von Lebesque-messbaren Teilmengen von $\bar{I}^s$ ist die Diskrepanz von P bezüglich $\mathfrak{B}$ definiert durch:

$$D_N(\mathfrak{B};P) = \sup_{B \in \mathfrak{B}} \left| \frac{A(B;P)}{N} - \lambda_s(B) \right| \tag{1.5}$$

Durch spezielle Wahl der Familie $\mathfrak{B}$ als Familie aller halboffenen Intervalle, bzw. der halboffenen Intervalle, die den Ursprung beinhalten, erhalten wir die beiden wichtigsten Typen der Diskrepanz.

Definition 1.2 ((extreme) Diskrepanz). Die (extreme) Diskrepanz $D_N(P)$ der Punktmenge P ist definiert durch $D_N(P) = D_N(\mathfrak{I}; P)$, wobei $\mathfrak{I}$ die Familie aller Teilintervalle von $\bar{I}^s$ der Form $\prod_{i=1}^s [u_i, v_i)$ darstellt.

Definition 1.3 (Sterndiskrepanz). Die Sterndiskrepanz $D_N^*(P)$ der Punktmenge P ist definiert durch $D_N^*(P) = D_N(\mathfrak{I}^*; P)$, wobei $\mathfrak{I}^*$ die Familie aller Teilintervalle von $\bar{I}^s$ der Form $\prod_{i=1}^s [0, v_i)$ darstellt, jener Teilmenge von $\mathfrak{I}$ mit Intervallen, die auch den Ursprung beinhalten.

Für Punktfolgen ist D_N definiert als die Diskrepanz der ersten N Elemente der Folge. Da wir nur Teilmengen von $\bar{I}^s$ betrachten, folgt sofort, dass auch $D_N(P) \leq 1$ und $D_N^*(P) \leq 1$.

Wie sich bereits erahnen lässt, sind obige beiden Konzepte der Diskrepanz nicht ganz unabhängig voneinander. Vielmehr hängen diese beiden Größen durch folgende Ungleichung zusammen.

Proposition 1.1. *Für jede Menge P von Punkten aus $\bar{I}^s$ gilt*

$$D_N^*(P) \leq D_N(P) \leq 2^s D_N^*(P) \tag{1.6}$$

Beweis. Die erste Ungleichung $D_N^*(P) \leq D_N(P)$ ist klar, da $\mathfrak{I}^*$ nur eine Teilfamilie von $\mathfrak{I}$ darstellt und die Diskrepanz als Supremum über diese Familie definiert ist.

Die zweite Ungleichung folgt aus der Tatsache, dass sich jedes Intervall aus $\mathfrak{I}$ als Summe und Differenz von 2^s Intervallen aus $\mathfrak{I}^*$ darstellen lässt (Inklusions-Exklusionsprinzip). In einer Dimension etwa gilt $[u, v) = [0, v) - [0, u)$ und in zwei Dimensionen $[u_1, v_1) \times [u_2, v_2) = [0, v_1) \times [0, v_2) - [0, v_1) \times [0, u_2) - [0, u_1) \times [0, v_2) + [0, u_1) \times [0, u_2)$. Nach genau demselben Schema addieren bzw. subtrahieren sich auch die $A(\prod [u_i, v_i))$ und die $\lambda_s(\prod [u_i, v_i)\,; P)$. Nimmt man nun den Absolutbetrag und schätzt diesen mittels Dreiecksungleichung ab (auch beim Supremum), so erhält man obige Ungleichung. □

Da wir die Diskrepanz immer benutzen werden, um obere Grenzen für den Fehler zu bestimmen, ist es damit also nicht so wichtig, ob wir die Diskrepanz oder die Sterndiskrepanz betrachten, da sich die Schranke nur maximal um einen numerischen Faktor unterscheidet. Da allerdings die Sterndiskrepanz leichter zu bestimmen ist, werden wir zumeist diese benutzen.

Eine erste Beobachtung zeigt gleich, dass es für die Diskrepanz eine untere Schranke gibt:

Proposition 1.2. *Für Punkte $\mathbf{x}_1, \ldots \mathbf{x}_N \in \mathbb{R}^s$ gilt*

$$D_N(\mathbf{x}_1, \ldots \mathbf{x}_N) \geq \frac{1}{N} \tag{1.7}$$

Beweis. Für $\mathbf{r}_1 = (x_1^{(1)}, \ldots x_1^{(s)})$ und $\varepsilon > 0$ sei $I_\varepsilon = \left[x_1^{(1)}, x_1^{(1)} + \varepsilon\right) \times \ldots \times \left[x_1^{(s)}, x_1^{(s)} + \varepsilon\right)$. Bei hinreichend kleinem ε befindet sich nur mehr x_1 in diesem Intervall und damit gilt

$$D_N(\mathbf{x}_1, \ldots \mathbf{x}_N) \geq \left| \frac{A(I_\varepsilon; \mathbf{x}_1, \ldots \mathbf{x}_N)}{N} - \lambda(I_\varepsilon) \right| = \frac{1}{N} - \varepsilon^s$$

Durch den Grenzübergang $\varepsilon \to 0$ erhalten wir Gleichung (1.7). □

Bemerkung 1.1. Für $s = 1$ gibt es eine solche Folge, die (1.7) mit Gleichheit erfüllt: $x_n = \frac{n}{N}$, $1 \leq n \leq N$.

Eine derartige Folge existiert allerdings für $s \geq 1$ nicht mehr. Vielmehr konnte Roth für diesen Fall eine andere untere Schranke für die Diskrepanz angeben (nach [DT97]).

Satz 1.3 (Roth, o.B.). *Für $s \geq 2$ ist die Diskrepanz $D_N(\mathbf{r}_n$ von N Punkten in $\mathbb{R}^s$ nach unten beschränkt durch*

$$D_N(\mathbf{r}_n) \geq D_N^*(\mathbf{r}_n) \geq \frac{1}{x^{4s}} \frac{1}{\left((k-1)\ln 2\right)^{\frac{s-1}{2}}} \frac{\ln^{\frac{k-1}{2}} N}{N} \tag{1.8}$$

Für $s = 1$ lässt sich leicht eine explizite Formel für die Diskrepanz angeben, davor allerdings noch ein Lemma, welches zeigt, dass die Diskrepanzen D_N und D_N^* stetige Funktionen sind:

Lemma 1.4. *Falls die Punkte $\mathbf{x}_1, \ldots \mathbf{x}_N, y_1, \ldots y_N \in [0,1]$ die Bedingung*

$$|x_n - y_n| \leq \varepsilon$$

für $1 \leq n \leq N$ erfüllen, so gilt

$$|D_N^*(\mathbf{x}_1, \ldots \mathbf{x}_N) - D_N^*(y_1, \ldots y_N)| \leq \varepsilon \tag{1.9}$$

$$|D_N(\mathbf{x}_1, \ldots \mathbf{x}_N) - D_N(y_1, \ldots y_N)| \leq 2\varepsilon \tag{1.10}$$

Beweis. Bezeichne $P = \{\mathbf{x}_1, \ldots \mathbf{x}_N\}$ und $Q = \{y_1, \ldots y_N\}$ und $J = [0, u) \subseteq [0,1)$ ein Intervall.

Wenn nun $y_n \in J$, dann ist $x_n \in J_1 := [0, u + \varepsilon) \cap [0,1]$ und daher

$$\frac{A(J;Q)}{N} - \lambda_1(J) \leq \frac{A(J_1;P)}{N} - \lambda_1(J_1) + \varepsilon \leq D_N^*(P) + \varepsilon$$

Wenn nun $x_n \in J_2 := [0, u - \varepsilon)$, dann ist $y_n \in J$ und daher

$$\frac{A(J;Q)}{N} - \lambda_1(J) \geq \frac{A(J_2;P)}{N} - \lambda_1(J_2) - \varepsilon \geq -D_N^*(P) - \varepsilon$$

Damit gilt also $D_N^*(Q) \leq D_N^*(P)_\varepsilon$. Durch Vertauschung von P und Q erhält man $D_N^*(P) \leq D_N^*(Q) + \varepsilon$ und damit $|D_N^*(P) - D_N^*(Q)| \leq \varepsilon$. Die zweite Gleichung folgt analog. □

Satz 1.5 (Niederreiter, de Clerck). *Für $0 \leq x_1 \leq x_2 \leq \ldots \leq x_N \leq 1$ gilt:*

$$D_N^*(\mathbf{x}_1, \ldots \mathbf{x}_N) = \frac{1}{2N} + \max_{1 \leq n \leq N} \left| x_n - \frac{2n-1}{2N} \right| \tag{1.11}$$

$$D_N(\mathbf{x}_1, \ldots \mathbf{x}_N) = \frac{1}{N} + \max_{1 \leq n \leq N} \left(\frac{n}{N} - x_n \right) - \min_{1 \leq n \leq N} \left(\frac{n}{N} - x_n \right) \tag{1.12}$$

Beweis. Aufgrund der Stetigkeit von D_N und D_N^* können wir $0 < x_1 < x_2 < \ldots < x_N < 1$ annehmen. Es bezeichne wieder $P = \{\mathbf{x}_1, \ldots \mathbf{x}_N\}$ und $x_0 = 1$ und $x_{N+1} = 1$. Damit lässt sich

nun die Sterndiskrepanz leicht berechnen:

$$\begin{aligned} D_N^*(P) &= \max_{0\le n\le N} \sup_{x_n<u\le x_{n+1}} \left| \frac{A([0,u)\,;P)}{N} - u \right| \\ &= \max_{0\le n\le N} \sup_{x_n<u\le x_{n+1}} \left| \frac{n}{N} - u \right| \\ &= \max_{0\le n\le N} \max\left(\left| \frac{n}{N} - x_n \right|, \left| \frac{n}{N} - x_{n+1} \right| \right) \\ f &= \max_{1\le n\le N} \max\left(\left| \frac{n}{N} - x_n \right|, \left| \frac{n-1}{N} - x_n \right| \right) \\ &= \frac{1}{2N} + \max_{1\le n\le N} \left| x_n - \frac{2n-1}{2N} \right| \end{aligned}$$

Ebenso für die Diskrepanz:

$$\begin{aligned} D_N(P) &= \max_{0\le i\le j\le N} \sup_{\substack{x_i<u\le x_{i+1} \\ x_j<v\le x_{j+1} \\ u<v}} \left| \frac{A([u,v)\,;P)}{N} - (v-u) \right| \\ &= \max_{0\le i\le j\le N} \sup_{\substack{x_i<u\le x_{i+1} \\ x_j<v\le x_{j+1} \\ u<v}} \left| \frac{j-i}{N} - (v-u) \right| \\ &= \max_{0\le i\le j\le N} \max\left(\left| \frac{j-i}{N} - (x_{j+1}-x_i) \right|, \left| \frac{j-i}{N} - (x_j - x_{i+1}) \right| \right) \end{aligned}$$

Mit der Abkürzung $r_n := n/N - x_n$ für $0 \le n \le N+1$ wird dies zu

$$\begin{aligned} D_N(P) &= \max_{0\le i\le j\le N} \max\left(\left| r_{j+1} - r_i - \frac{1}{N} \right|, \left| r_j - r_{i+1} + \frac{1}{N} \right| \right) \\ &= \max_{\substack{0\le i\le N \\ 1\le j\le N+1}} \left| \frac{1}{N} + r_i - r_j \right| \end{aligned}$$

Das Maximum ist auch ohne Betragsstriche sicherlich positiv (Wäre es negativ, wäre $\frac{1}{N} - r_i + r_j$ betragsmäßig um $\frac{2}{N}$ größer). Damit kann nun obiges Maximum auf die drei Summanden aufgeteilt werden:

$$D_N(P) = \frac{1}{N} + \max_{0\le i\le N} r_i - \min_{1\le j\le N+1} r_j$$

Damit erhält man genau Gleichung (1.12). □

Bemerkung 1.2. Proposition 1.2 folgt auch direkt aus diesem Satz.

Bemerkung 1.3. Wenn wir die Minima und Maxima Beiträge betrachten, so fällt auf, dass die Menge $\left\{ \frac{n}{N} : 1 \le n \le N \right\}$ minimale Diskrepanz $D_N(\mathbf{x}_1, \ldots \mathbf{x}_N) = \frac{1}{N}$ besitzt, wogegen die Menge $\left\{ \frac{2n-1}{2N} : 1 \le n \le N \right\}$ minimale Sterndiskrepanz $D_N^*(\mathbf{x}_1, \ldots \mathbf{x}_N) = \frac{1}{2N}$ aufweist.

Bemerkung 1.4. Diese unteren Diskrepanzschranken können nur von endlichen Punktmengen, nicht aber von unendlichen Punktfolgen für alle N angenommen werde.

Wenn wir allerdings Integrale bzw. Lösungen von Differentialgleichungen durch QMC Verfahren nähern, dann benötigen wir hin und wieder auch die Diskrepanz einer Punktmenge bezüglich einer Funktion $f(x)$, die angibt, wie gut eine gegebene Punktmenge die Funktion f annähert:

Definition 1.4. d[Sterndiskrepanz bzgl. einer Funktion] Die Sterndiskrepanz $D_N^*(P,f)$ der Menge $P = \{\mathbf{x}_1, \ldots \mathbf{x}_N \in \mathbb{R}\}$ bezüglich der Funktion $f(x)$ ist definiert durch

$$D_N^*(P,f) = \sup_{r>0} \left| \frac{1}{N} \sum_{1 \le i \le N} \chi_{[0,r)}(x_i) - \int_{[0,r)} f(x)dx \right| \tag{1.13}$$

1.3 Fehlerschranken

Obwohl neben der Diskrepanz auch noch andere Maße für die Gleichverteilung einer Menge bzw. Folge angegeben wurden, wie etwa die Nonuniformity von SOBOL [Sob60] oder Diaphony von Zinterhof [Zin76], hat die Diskrepanz eine besondere Bedeutung erlangt aufgrund einer von Koksma angegebenen Fehlerabschätzung, die den Fehler bei einer QMC Näherung nach oben abschätzt durch ein Produkt aus der Diskrepanz der verwendeten Folge und der Variation der zu nähernden Funktion.

Dazu wird allerdings das Konzept der Variation einer Funktion benötigt. Als eine Partition von $\bar{I}^s$ wird eine Menge von s endlichen Folgen $\eta_i^{(0)}, \eta_i^{(1)}, \ldots \eta_i^{(m_i)}$, $i \le 1 \le s$ mit $0 = \eta_i^{(0)} \le \eta_i^{(1)} \le \ldots \le \eta_i^{(m_i)}$ bezeichnet. Für eine solche Partition definiert man nun den Operator Δ_i für jedes i mit $1 \le m_i \le m_i$ durch

$$\Delta_i f(\mathbf{x}_1, \ldots \mathbf{x}_{i-1}, \eta_i^{(j)}, \mathbf{x}_{i+1}, \ldots \mathbf{x}_s) = \\ f(\mathbf{x}_1, \ldots \mathbf{x}_{i-1}, \eta_i^{(j+1)}, \mathbf{x}_{i+1}, \ldots \mathbf{x}_s) - f(\mathbf{x}_1, \ldots \mathbf{x}_{i-1}, \eta_i^{(j)}, \mathbf{x}_{i+1}, \ldots \mathbf{x}_s)$$

Und damit $\Delta_{i_1 \ldots i_m} := \Delta_{i_1} \Delta_{i_2} \ldots \Delta_{i_m}$.

Definition 1.5 (Variation im Sinn von Vitali). Die Variation im Sinne von Vitali für eine Funktion $f : [0,1] \to \mathbb{R}$ ist definiert durch

$$V^{(s)}(f) = \sup_{P \in \mathfrak{P}} \sum_{j_1=0}^{m_1-1} \cdots \sum_{j_s=0}^{m_s-1} \left| \Delta_{1,\ldots s} f\left(\eta_1^{(j_1)}, \ldots \eta_s^{(j_s)}\right) \right| \tag{1.14}$$

wobei das Supremum über alle Partitionen von $\bar{I}^s$ läuft.
Wenn $V^{(k)}(f)$ endlich ist, so wird f von beschränkter Variation im Sinn von Vitali genannt.

Bemerkung 1.5. Die $\eta_{i_k}^{(j)}$ stellen ein Netz auf $\bar{I}^s$ dar, und die Variation im Sinne von Vitali berechnet sich durch alternierende Summen über benachbarte Gitterpunkte und entsprechende Absolutbeträge.

Bemerkung 1.6. Wenn f stetige partielle Ableitungen in $\bar{I}^s$ besitzt, dann kann die Variation im Sinn von Vitali berechnet werden durch

$$V^{(s)}(f) = \int \cdots \int_{\bar{I}^s} \left| \frac{\partial^s f}{\partial u_1 \ldots \partial u_s} \right| du_1 \ldots du_s$$

In einer Dimension können wir damit schon das klassische Ergebnis von Koksma formulieren:

Satz 1.6 (Koksma). *Wenn f von beschränkter Variation auf $[0,1]$ ist, dann gilt für alle $\mathbf{x}_1, \ldots \mathbf{x}_N \in [0,1]$*

$$\left| \frac{1}{N} \sum_{n=1}^{N} f(x_n) - \int_0^1 f(u)du \right| \leq V(f) D_N^*(\mathbf{x}_1, \ldots \mathbf{x}_N) \tag{1.15}$$

Beweis. Ausgehend von $0 = x_0 \leq x_1 \leq \ldots \leq x_N \leq x_{N+1} = 1$ können wir die linke Seite von (1.15) durch partielle Summation bzw. Integration umformen zu

$$\begin{aligned} \frac{1}{N} \sum_{n=1}^{N} f(x_n) - \int_0^1 f(u)du &= -\sum_{n=0}^{N} \frac{n}{N} \big(f(x_{n+1}) - f(x_n) \big) + \int_0^1 u df(u) \\ &= \sum_{n=1}^{N} \int_{x_n}^{x_{n+1}} \left(u - \frac{n}{N} \right) df(u) \end{aligned}$$

Für $0 \leq n \leq N$ gilt $\left| u - \frac{n}{N} \right| \leq D_N^*(\mathbf{x}_1, \ldots \mathbf{x}_N)$ für $x_n \leq u \leq x_{n+1}$ und damit

$$\begin{aligned} &\leq D_N^*(\mathbf{x}_1, \ldots \mathbf{x}_N) \sum_{n=1}^{N} \int_{x_n}^{x_{n+1}} df(u) \\ &\leq D_N^*(\mathbf{x}_1, \ldots \mathbf{x}_N) V(f). \end{aligned}$$

□

Bemerkung 1.7. Obige Ungleichung (1.15) ist deshalb von solcher Wichtigkeit, weil damit der Fehler in Form eines Produktes geschrieben werden kann, wobei der eine Faktor nur von der Funktion f abhängt und der andere nur von der verwendeten Folge bzw. Punktmenge.

Diese Ungleichung lässt sich verallgemeinern auf beliebige Dimensionen, allerdings ist dafür das Konzept der Variation zu modifizieren:

Definition 1.6 (Variation im Sinn von Hardy und Krause). Für $1 \leq k \leq s$ und $1 \leq i_1 \leq \ldots \leq i_k$ bezeichne $V^{(k)}(f; i_1, \ldots i_k)$ die Variation im Sinne von Vitali der Restriktion von f auf die k-dimensionale Fläche

$$\big\{ (u_1, \ldots u_s) \in \bar{I}^s : u_j = 1 \text{ für } j \neq i_1, \ldots i_k \big\}$$

Dann wird

$$V(f) = \sum_{k=1}^{s} \sum_{1 \leq i_1 \leq \ldots \leq i_k \leq s} V^{(k)}(f; i_1, \ldots i_k) \tag{1.16}$$

die Variation von f auf $\bar{I}^s$ im Sinn von Hardy und Krause genannt.

Mit diesen Definitionen kann nun die bekannte und wichtige Koksma-Hlawka-Ungleichung als Verallgemeinerung von (1.15) auf s Dimensionen formuliert werden (ohne Beweis).

Satz 1.7 (Koksma-Hlawka-Ungleichung, o.B.). *Ist f von beschränkter Variation $V(f)$ auf $\bar{I}^s$ im Sinne von* Hardy *und* Krause*, so gilt für alle $\mathbf{x}_1, \ldots \mathbf{x}_N \in \bar{I}^s$*

$$\left| \frac{1}{N} \sum_{n=1}^{N} f(\mathbf{r}_n) - \int_{\bar{I}^s} f(\mathbf{u}) d\mathbf{u} \right| \leq V(f) D_N^*(\mathbf{x}_1, \ldots \mathbf{x}_N) \tag{1.17}$$

Kapitel 2

Folgen kleiner Diskrepanz

Inhaltsangabe

2.1 Die $(n\alpha)$ Folgen 10
2.2 Digitale Folgen (VAN DER CORPUT, HALTON) 11
2.2.1 Die Van der Corput - Folge 11
2.2.2 Die Halton - Folge 12
2.2.3 ATANASSOV's Diskrepanzschranke 14
2.2.4 Die Hammersley-Menge 24
2.3 (t,s)-Folgen 25
2.3.1 Diskrepanz von (t, m, s)-Netzen 26
2.3.2 Diskrepanz von (t, s)-Folgen 29
2.4 Existenz von (t, s)-Folgen 31
2.5 Erzeugung von (t, m, s)-Netzen und (t, s)-Folgen 31
2.5.1 Ein allgemeines Prinzip für (t, m, s)-Netze 32
2.5.2 Ein allgemeines Prinzip für (t, s)-Folgen 33
2.5.3 Spezielle Konstruktionen von (t, s)-Folgen 34
2.5.4 Faure-Folgen 36

Die untere Schranke der Ordnung $O(\frac{1}{N})$ für die Diskrepanz einer Punktmenge kann nur von endlichen Mengen, nicht aber von Folgen für alle N erfüllt werden. Im Gegenteil: Die Diskrepanz einer Folge ist für unendlich viele N von höherer Ordnung als $\frac{1}{N}$, ein Phänomen, das "Irregularities of distribution" genannt wird.

Für $s = 1$ wurde von Schmitt gezeigt, dass eine Konstante c existiert, sodass für jede Folge S mit Elementen aus I für unendlich viele N gilt (Für endliche Punktmengen gilt dies auch für $s = 2$):

$$D_N(S) \geq c\frac{\ln^s N}{N} \tag{2.1}$$

Eine Schranke dieser Ordnung wurde nur in einer Dimension bewiesen, für höhere Dimension existiert lediglich die Vermutung, dass eine Konstante c existiert, sodass für jede Folge S mit Elementen aus $\bar{I}^s$ für unendlich viele N gilt

$$D_N(S) \geq c_s\frac{\ln^s N}{N} \tag{2.2}$$

Es existieren Folgen, die eine Diskrepanz von dieser Ordnung haben, womit dann diese Ordnung die beste mögliche wäre.

Folgen $\mathbf{r}_n$, die für alle N eine kleine Diskrepanz $D_N(\mathbf{r}_n)$ aufweisen, werden *Folgen kleiner Diskrepanz* genannt. Im Allgemeinen sind damit nur Folgen gemeint, die eine obere Schranke für die Diskrepanz besitzen von der Form:

$$D_N(\mathbf{r}_n) = O\left(\frac{\ln^s N}{N}\right) \tag{2.3}$$

Es gibt verschiedene Konstruktionen, um Folgen kleiner Diskrepanz zu erzeugen, die ich in diesem Abschnitt besprechen möchte

2.1 Die $(n\alpha)$ Folgen

Nimmt man alle Vielfachen einer irrationalen Zahl $\alpha \in \mathbb{R}_+$, so ist die Folge $S(\alpha) := (n\alpha - [n\alpha])$ gleichverteilt ([DT97]). Eine Folge kleiner Diskrepanz ist sie allerdings nur unter einer einschränkenden Voraussetzung, welche die Kettenbruchentwicklung von α betrifft:

Es sei $\alpha = [a_0; a_1, a_2, \ldots]$ die Kettenbruchentwicklung von α, also

$$\alpha = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \ldots}}$$

wobei $a_i \in \mathbb{N}^+$. Es seien weiters $q_0, q_1, \ldots$ die Nenner dieser Konvergenten zu α. Dann existiert folgende Schranke für die Diskrepanz (Für einen Beweis siehe [DT97] oder [Nie92]):

Satz 2.1. *Sei α irrational und $N \geq 1$. Dann kann N dargestellt werden in der Form:*

$$N = \sum_{i=1}^{l(N)} c_i q_i \tag{2.4}$$

wobei $l(N)$ jene natürliche Zahl ist, für die $q_{l(N)} \leq N < q_{l(N)+1}$ gilt, und die Koeffizienten $0 \leq c_i \leq a_{i+1}$ für $0 \leq i \leq l(n)$ erfüllen. Damit gilt dann folgende Diskrepanzschranke:

$$D_N\big(S(\alpha)\big) < \frac{1}{N} \sum_{\substack{i=0 \\ c_i \neq 0}}^{l(n)} (c_i + 1) \leq \frac{1}{N} \sum_{i=1}^{l(N)+1} a_i \tag{2.5}$$

Korollar 2.2. *Wenn die Zahl α so gewählt wird, dass $\frac{1}{m}\sum_{i=1}^{m} a_i$ für alle m endlich ist, dann gilt*

$$D_N\big(S(\alpha)\big) = O\left(\frac{\ln N}{N}\right). \tag{2.6}$$

Damit ist also $S(z)$ eine Folge kleiner Diskrepanz wenn die Koeffizienten a_i der Partialbruchentwicklung "klein" sind.

In höheren Dimensionen können analoge mehrdimensionale Folgen definiert werden, wobei die verwendeten Irrationalzahlen linear unabhängig über $\mathbb{Q}$ sind. Für $s \geq 2$ ist allerdings kein $\alpha \in \mathbb{R}^s$ bekannt, für das $D_N\big(s(\alpha)\big) = O\left(\frac{(1+\ln N)^{s+1}}{N}\right)$ gilt. Wenn die Koordinaten von $\alpha \in \mathbb{R}^s$ algebraische Zahlen sind, die obige Bedingung der linearen Unabhängigkeit erfüllen, dann gilt nach einem Resultat von NIEDERREITER $D_N(S(\alpha)) = O(N^{-1+\varepsilon})$ für alle $\varepsilon > 0$.

2.2 Digitale Folgen (Van der Corput, Halton)

Eine weitere Klasse von Folgen kleiner Diskrepanz kann erhalten werden mit Hilfe der sogenannten Radikalinvers-Funktion Φ_p.

Jede ganze Zahl $n \in \mathbb{N}_0$ kann eindeutig in Basis $p \geq 2$ dargestellt werden in der Form

$$n = \sum_{j=0}^{\infty} a_j(n) p^j \tag{2.7}$$

wobei die $a_j(n) \in Z_p = \{0, 1, \ldots p-1\}$ und $a_j(n) = 0$ für hinreichend große j gilt, die Summe also eigentlich endlich ist.

Definition 2.1 (Radikal-Inversfunktion). Für $p \geq 2$ ist die Radikal-Inversfunktion $\Phi_p(n)$ in Basis p definiert durch

$$\Phi_p(n) = \sum_{j=0}^{\infty} a_j(n) p^{-j-1} \tag{2.8}$$

wobei a_j die Ziffern der Darstellung (2.7) von n in Basis p bezeichnen.

$\Phi_p(n)$ ist also anschaulich gesagt die Spiegelung der Darstellung von n in Basis p am Komma, womit natürlich $\Phi_p(n) < 1$ gilt. Bei Folgen kleiner Diskrepanz beginnt im Folgenden n immer bei 0.

2.2.1 Die Van der Corput - Folge

Mit Hilfe dieser Radikal-Inversfunktion Φ_p kann nun die (eindimensionale) Van der Corput-Folge definiert werden:

Definition 2.2 (Van der Corput-Folge). Für eine natürliche Zahl $p \geq 2$ wird die durch $x_n = \Phi_p(n) \in I$ für $n \geq 0$ definierte Folge die Van der Corput-Folge S_p genannt.

Die Van der Corput-Folge ist, wie später noch bewiesen wird, eine Folge kleiner Diskrepanz. Speziell ist

$$\limsup_{n\to\infty} \frac{N D_N^*(S_p)}{\ln N} = \limsup_{n\to\infty} \frac{N D_N(S_p)}{\ln N} = \begin{cases} \frac{p^2}{4(p+1)\ln p} & \text{für } p = 2m \\ \frac{p-1}{4\ln p} & \text{für } p = 2m+1 \end{cases}$$

womit S_3 asymptotisch die beste Van der Corput-Folge darstellt. Allerdings geht diese von also der Basis p abhängige Konstante mit steigendem p ebenfalls gegen ∞.

Verbesserungen dieser Konstante sind möglich, indem sogenannte *verallgemeinerte Van der Corput-Folgen in Basis p* betrachtet werden:

$$x_n = \sum_{j=1}^{\infty} \sigma\big(a_j(n)\big) p^{-j-1} \tag{2.9}$$

wobei σ eine Permutation von Z_b darstellt. Als bisher bester bekannter Wert konnte von Faure [Fau93] obige Konstante für $p = 36$ und eine spezielle Permutation σ von Z_{36} herabgesetzt werden bis auf

$$\limsup_{N\to\infty} \frac{N D_N(\tilde{S}_{36})}{\ln N} = \frac{23}{35 \ln 6} = 0.366$$

Für die Sterndiskrepanz gilt nach [Fau81] für eine bestimmte Folge in Basis 12 sogar ein Wert von 0.223.

2.2.2 Die Halton - Folge

Die Erweiterung der Van der Corput-Folge auf mehrere Dimensionen wird Halton-Folge genannt:

Definition 2.3 (Halton-Folge). Für eine Dimension $s \geq 2$ seien $p_1, \ldots p_s$ natürliche Zahlen ≥ 2. Dann wird die Halton-Folge in den Basen $p_1, \ldots p_s$ definiert durch:

$$\mathbf{r}_n = (\Phi_{p_1}(n), \ldots \Phi_{p_s}(n)) \in \bar{I}^s \quad \text{für } n \geq 0$$

Für die Halton-Folge kann eine relativ einfache Schranke für die Diskrepanz angegeben werden, die auch ein Kriterium für die Wahl der Basen $p_1, \ldots p_s$ darstellen wird.

Satz 2.3 (HALTON, 1960, [Hal60]). *Wenn S die Halton-Folge in den paarweise relativ primen Basen $p_1, \ldots p_s \geq 2$ bezeichnet, dann kann die Diskrepanz für alle $N \geq 1$ abgeschätzt werden durch:*

$$D_N^*(S) < \frac{s}{N} + \frac{1}{N} \prod_{i=1}^{s} \left(\frac{p_i - 1}{2 \ln p_i} \ln N + \frac{p_i + 1}{2} \right) \tag{2.10}$$

Beweis. Bei fixem $N \geq 1$ bezeichne $D(J) = A(J; S_N) - N\lambda_s(J)$ für ein Intervall $J \subseteq \bar{I}^s$, wobei S_N die Menge der ersten N Elemente der Halton-Folge darstellt.

Weiters gelten folgende Bezeichnungen für $1 \leq i \leq s$ und $e \in \mathbb{N}_0$:

$$\mathfrak{E}_i(e) = \left\{ \left[0, a p_i^{-e}\right) : a \in \mathbb{N}_0, 0 < a \leq p_i^e \right\}$$
$$\mathfrak{F}_i(e) = \left\{ \left[c p_i^{-f}, (c+1) p_i^{-f}\right) : c, f \in \mathbb{N}_0, 0 \leq f \leq e, 0 \leq c < p_i^f \right\}$$

Außerdem sei die Menge der s-dimensionalen derartigen Intervalle folgendermaßen bezeichnet:

$$\mathfrak{E}(e_1, \ldots e_s) = \left\{ \prod_{i=1}^{s} E_i : E_i \in \mathfrak{E}_i(e_i) \cup \mathfrak{F}_i(e_i) \right\}$$

Zum Beweis des Satzes benötigt man eine Hilfsabschätzung, die mittels vollständiger Induktion zu beweisen ist:

$$|D(E)| \leq \prod_{\substack{i=1 \\ E_i \notin \mathfrak{F}_i(e_i)}}^{s} \left(\frac{1}{2} (p_i - 1) e_i + 1 \right) \quad \text{für alle } E \in \mathfrak{E}(e_1, \ldots e_s) \tag{2.11}$$

wobei ein leeres Produkt gleich 1 sein soll. Die Induktion wird über die Anzahl k der Faktoren in obigem Produkt durchgeführt, also über die Anzahl der Indices, für die $E_i \notin \mathfrak{F}_i(e_i)$.

Induktionsbasis: $k = 0$ bedeutet, dass das Intervall E dargestellt werden kann als

$$\prod_{i=1}^{s} \left[c_i p_i^{-f_i}, (c_i + 1) p_i^{-f_i} \right) . \tag{2.12}$$

Damit sind allerdings für die Komponenten $\Phi_{p_i}(n)$ von $\mathbf{r}_n \in E$ jeweils die ersten f_i Ziffern nach dem Komma und damit die hintersten Ziffern von n in Basis p_i festgelegt. Dies bedeutet, dass n in einer fix vorgegebenen Restklasse $(\mathrm{mod} p_i)$ liegen muss, damit $\Phi_{p_i}(n) \in E_i$. Da die Basen für die einzelnen Dimensionen relativ prim sind, muss nach den Chinesischen Restsatz n damit in einer fixen Restklasse $\left(\mathrm{mod} m = p_1^{f_1} \ldots p_s^{f_s}\right)$ liegen, damit $\mathbf{r}_n \in E$. Das bedeutet, dass von m auf einander folgenden Zahlen nur eine in E liegen kann. Mit $N = mq + r$, $r < m$

liegen damit nur q oder $q+1$ Punkte von S_N in E. Da außerdem $\lambda_s(E) = p_1^{-f_1} \ldots p_s^{-f_s} = \frac{1}{m}$, folgt in beiden Fällen obige Ungleichung:

$$|D(E)| = \left| q - N\frac{1}{m} \right| = \left| q - \frac{mq+r}{m} \right| = \left| \frac{r}{m} \right| < 1$$

oder

$$|D(E)| = \left| q+1 - N\frac{1}{m} \right| = \left| q+1 - \frac{mq+r}{m} \right| = \left| 1 - \frac{r}{m} \right| \leq 1$$

Induktionsschritt: Obige Ungleichung gelte für $k-1$ und soll damit für k bewiesen werden. Ohne Beschränkung der Allgemeinheit kann $E_i \notin \mathfrak{F}_i(e_i)$ für $1 \leq i \leq k$ angenommen werden. (Anderenfalls werden einfach die Komponenten dementsprechend vertauscht. Die Eigenschaften der Folge ändern sich dadurch nicht.) Damit hat also E_1 die Form $\left[0, ap_1^{-e_1}\right)$ mit $0 < a \leq p_1^{e_1}$. Aus der Ziffernentwicklung von $ap_1^{-e_1} = \sum_{j=1}^{e_1} d_j p_1^{-j}$ mit $d_j < p_1$ in Basis p_1 ist ersichtlich, dass E_1 als disjunkte Vereinigung von d_1 Intervallen aus $\mathfrak{F}_1(e_1)$ der Länge p_1^{-1} und d_2 Intervallen aus $\mathfrak{F}_1(e_1)$ der Länge p_1^{-2} usw. dargestellt werden kann. Mit $d = \sum_{j=1}^{e_1} d_j$ also

$$E_1 = \bigcup_{r=1}^{d} F_r \quad \text{mit } F_r \in \mathfrak{F}_1(e_1)$$

und damit wird E als Vereinigung von Mengen geschrieben, für die die Induktionsannahme gilt:

$$E = \bigcup_{r=1}^{d} (F_r \times E_2 \times \ldots \times E_s)$$

Mit Hilfe der Dreiecksungleichung und der Induktionsannahme folgt

$$|D(E)| \leq \sum_{r=1}^{d} |D\left(F_r \times E_2 \times \ldots \times E_s\right)| \leq d \prod_{i=1}^{k} \left(\frac{1}{2}(p_i - 1)e_i + 1 \right) \tag{2.13}$$

In einem zweiten Schritt wird nun $\left[0, ap_1^{-e_1}\right)$ dargestellt als $[0,1) \setminus \left[ap_1^{-e_1}, 1\right)$. Analog wie $\left[0, ap_1^{-e_1}\right)$ kann $\left[ap_1^{-e_1}, 1\right)$ dargestellt werden durch insgesamt $(p_1-1)e_1 - d + 1$ Intervalle aus $\mathfrak{F}_1(e_1)$. Da $[0,1) \in \mathfrak{F}_1(e_1)$, folgt:

$$\begin{aligned} |D(E)| &\leq |D([0,1) \times E_2 \times \ldots \times E_s)| + \left|D(\left[ap_1^{-e_1}, 1\right) \times E_2 \times \ldots \times E_s)\right| \\ &\leq \prod_{i=2}^{k} \left(\frac{1}{2}(p_i - 1)e_i + 1 \right) + ((p_1-1)e_1 - d + 1) \prod_{i=2}^{k} \left(\frac{1}{2}(p_i - 1)e_i + 1 \right) \\ &= ((p_1-1)e_1 - d + 2) \prod_{i=2}^{k} \left(\frac{1}{2}(p_i - 1)e_i + 1 \right) \end{aligned} \tag{2.14}$$

Nach Addition der Gleichungen (2.13) und (2.14) und Division durch 2 ist damit der Induktionsschritt beendet.

Für ein allgemeines Intervall $J = \prod_{i=1}^{s} [0, u_i) \in \mathfrak{I}^*$ sei e_i für $1 \leq i \leq s$ definiert als die kleinste natürliche Zahl mit $p_i^{e_i} \geq N$ und a_i als die kleinste natürliche Zahl mit $a_i p_i^{-e_i} \geq u_i$. Damit wird nun ein neues Intervall $E = \prod_{i=1}^{s} \left[0, a_i p_i^{-e_i}\right) \in \mathfrak{E}(e_1, \ldots e_s)$ definiert, für

das $A(J;S_N) = A(E;S_N)$ ist. Dies gilt deshalb, da die i-te Koordinaten aller $\mathbf{r}_n \in S_N$ Rationalzahlen mit Nenner $p_i^{e_i}$ sind. Mit diesen Definitionen gilt somit

$$\begin{aligned}|D(J)| = |A(J;S_N) - N\lambda_s(J)| &\leq N\left(\lambda_s(E) - \lambda_s(J)\right) + |D(E)| \\ &\leq N\sum_{i=1}^{s} p_i^{-e_i} + |D(E)| \leq s + \prod_{i=1}^{s}\left(\frac{1}{2}(p_i-1)e_i + 1\right)\end{aligned}$$

Aus $p_i^{e_i-1} < N$ folgt $e_i < 1 + \frac{\ln N}{\ln p_i}$ für $1 \leq i \leq s$ und damit

$$|D(J)| \leq s + \prod_{i=1}^{s}\left(\frac{p_i-1}{2\ln p_i}\ln N + \frac{p_i+1}{2}\right)$$

Die rechte Seite ist somit von J unabhängig und gilt daher auch für das Supremum über alle $J \in \mathfrak{I}^*$. Mit $D_N^*(S) = \sup_J\left(\frac{D(J)}{N}\right)$ folgt also sofort:

$$D_N^*(S) \leq \frac{s}{N} + \frac{1}{N}\prod_{i=1}^{s}\left(\frac{p_i-1}{2\ln p_i}\ln N + \frac{p_i+1}{2}\right)$$

□

Obige Schranke für die Diskrepanz kann man auch umformen zu

$$D_N^*(S) \leq A\left(p_1,\ldots p_s\right)\frac{\ln^s N}{N} + O\left(\frac{\ln^{s-1} N}{N}\right) \tag{2.15}$$

wobei der Koeffizient des führenden Terms $A\left(p_1,\ldots p_s\right) = \prod_{i=1}^{s}\frac{p_i-1}{2\ln p_i}$ lautet. Dieser ist minimal, wenn man für die Basen möglichst kleine Zahlen, also am besten die ersten s Primzahlen benutzt. Mit dieser Abschätzung ist also die Halton-Folge - und damit mit $s = 1$ auch die Van der Corput-Folge - eine Folge kleiner Diskrepanz.

2.2.3 Atanassov's Diskrepanzschranke

Die Konstante $A\left(p_1,\ldots p_s\right) = \prod_{i=1}^{s}\frac{p_i-1}{2\ln p_i}$ aus Gleichung (2.15) strebt für hohe Dimensionen $s \to \infty$ ebenfalls gegen ∞, da die p_i paarweise relativ prime und damit verschiedene natürliche Zahlen darstellen. Dies hat zur Folge, dass diese Abschätzung für hochdimensionale QMC Simulationen praktisch unbrauchbar ist. Erst vor kurzem gelang es Atanassov in [Ata00] diese Schranke für die oben definierte Halton Folge in den Basen $p_1,\ldots p_s$ derart zu verbessern, dass

$$\lim_{s\to\infty}\tilde{A}_s\left(p_1,\ldots p_s\right) = 0 \tag{2.16}$$

gilt. Unter spezieller Wahl der Basen p_i, bzw für eine leicht modifizierte Halton-Folge uneingeschränkt konnte er eine weitere Verbesserung erzielen. Der Kürze wegen werde ich im Folgenden für ein s-Tupel $a^{(1)},\ldots a^{(s)}$ auch manchmal nur $\mathbf{a}$ schreiben.

Satz 2.4 (Atanassov,2000, [Ata00]). *Wenn S die Halton-Folge in den paarweise relativ primen Basen $p_1,\ldots p_s \geq 2$ bezeichnet, dann gilt für die Diskrepanz für $N \geq 1$:*

$$D_N^*(S) < \tilde{A}_s\frac{\ln^s N}{N} + O\left(\frac{\ln^{s-1} N}{N}\right) \tag{2.17}$$

mit einer von N unabhängigen Konstante

$$\tilde{A}_s\left(p_1,\ldots p_s\right) = \frac{2}{s!}\prod_{i=1}^{s}\frac{p_i-1}{2\ln p_i} \tag{2.18}$$

Bemerkung 2.1. Im Beweis wird von einer speziellen Form der Entwicklung in Basis p_i Gebrauch gemacht: Und zwar wird der Punkt $\mathbf{z} = \left(z^{(1)}, \ldots z^{(s)}\right)$ zerlegt in der Form

$$z^{(i)} = \sum_{j=0}^{\infty} a_j^{(i)} p_i^{-j}, \left|a_j^{(i)}\right| \leq \frac{p_i}{2}, \left|a_j^{(i)}\right| + \left|a_{j+1}^{(i)}\right| \leq p_i - 1, a_0^{(i)} = 0 \text{ oder } 1 \tag{2.19}$$

Diese spezielle Entwicklung kann aus der üblichen Entwicklung in Basis p_i (also $z^{(i)} = \sum_{j=0}^{\infty} \tilde{a}_j^{(i)} p_i^{-j}, 0 \leq a_j^{(i)} < p_i$) erhalten werden, indem alle Koeffizienten $\tilde{a}_k^{(i)} > \frac{p_i}{2}$ durch $p_i - (p_i - \tilde{a}_k^{(i)})$ ersetzt werden, bis alle Koeffizienten die Bedingung erfüllen. Ersterer Term wirkt natürlich für den Koeffizienten von p_i^{-j+1} als inkrementierend, während der zweite, das neue $a_j^{(i)}$, nun kleiner gleich $\frac{p_i}{2}$ ist. Dies wird solange durchgeführt, bis die $a_j^{(i)}$ obige Bedingung erfüllen.

Beweis. Es sei das Intervall $J = \prod_{i=1}^{s} \left[0, z^{(i)}\right) \subseteq I^s$ beliebig gewählt und $N \geq 1$ hinreichend groß. Der Punkt $\mathbf{z} = \left(z^{(1)}, \ldots z^{(s)}\right)$ sei zerlegt wie oben beschrieben. Es sei weiters $n_i = \left\lfloor \frac{\ln N}{\ln p_i} \right\rfloor + 1$ gewählt und somit $p_i^{n_i} = p_i^{\left\lfloor \frac{\ln N}{\ln p_i} \right\rfloor + 1} > p_i^{\frac{\ln N}{\ln p_i}} = e^{\frac{\ln N}{\ln p_i} \ln p_i} = N$.

Schließlich sollen noch die Teilsummen bis zum n_i-ten Summanden obiger Entwicklung durch $z_k^{(i)} = \sum_{j=0}^{k-1} a_j^{(i)} p_i^{-j}$ für $1 \leq k \leq n_i - 1$, $z_0^{(i)} = 0$, $z_{n_i}^{(i)} = z^{(i)}$ bezeichnet werden.

Wegen der Teleskopsumme $z^{(i)} = \sum_{j=0}^{n_i - 1} \left(z_{j+1}^{(i)} - z_j^{(i)}\right)$ kann das Intervall J analog dargestellt werden als

$$J = \sum_{j_1=0}^{n_1-1} \ldots \sum_{j_s=0}^{n_s-1} \varepsilon(j_1, \ldots j_s) I(j_1, \ldots j_s) \tag{2.20}$$

mit den Bezeichnungen

$$\varepsilon(j_1, \ldots j_s) = \prod_{i=1}^{s} \varepsilon_{j_i}^{(i)}$$

$$\varepsilon_{j_i}^{(i)} = \begin{cases} 1, \text{ falls } z_{j_i}^{(i)} \leq z_{j_i+1}^{(i)} \\ -1, \text{ falls } z_{j_i}^{(i)} > z_{j_i+1}^{(i)} \end{cases}$$

und $I(j_1, \ldots j_s) = \prod_{i=1}^{s} I_{j_i}^{(i)}$, $I_{j_i}^{(i)} = \left[\min\left(z_{j_i}^{(i)}, z_{j_i+1}^{(i)}\right), \max\left(z_{j_i}^{(i)}, z_{j_i+1}^{(i)}\right)\right)$. Zudem bedeute Subtraktion für Intervalle $A - B := A \backslash B$.

Im Folgenden sei

$$B = \left\{ (j_1, \ldots j_s) \in \mathbb{N}^s : j_i \leq n_i - 1, \prod_{i=1}^{s} p_i^{j_i+1} \leq N \right\} \tag{2.21}$$

$$B_k = \left\{ (j_1, \ldots j_s) \in \mathbb{N}^s : j_i \leq n_i - 1, \prod_{i=1}^{k} p_i^{j_i+1} \leq N, \prod_{i=1}^{k+1} p_i^{j_i+1} > N \right\} \tag{2.22}$$

Für die Elemente von B gilt sogar $j_i < n_1 - 1$, da $p_i^{n_1} > N$. Die einzigen Intervalle $I_{j_i}^{(i)}$, die sich nicht in der Form $\left[x_i p_i^{j_i}, x_i p_i^{j_i} + \tilde{x}_i p_i^{j_i - 1}\right)$ oder $\left[x_i p_i^{j_i} - \tilde{x}_i p_i^{j_i - 1}, x_i p_i^{j_i}\right)$ schreiben lassen,

sind jene mit $j_i = n_i - 1$. Diese Indices befinden sich allerdings nicht in B und damit nicht im ersten, sondern im zweiten Term in folgender Umformung:

$$A_N(J) - N\mu(J) = \sum_{j_1=0}^{n_1-1} \cdots \sum_{j_s=0}^{n_s-1} \varepsilon(\mathbf{j}) \left(A_N\left(I(\mathbf{j})\right) - N\mu\left(I(\mathbf{j})\right)\right) =$$

$$\sum_{\mathbf{j}\in B} \varepsilon(\mathbf{j}) \left(A_N(I(\mathbf{j})) - N\mu(I(\mathbf{j}))\right) + \sum_{k=0}^{s-1} \sum_{\mathbf{j}\in B_k} \varepsilon(\mathbf{j}) \left(A_N(I(\mathbf{j})) - N\mu(I(\mathbf{j}))\right) \quad (2.23)$$

Wie sich im folgenden zeigen wird, ist der erste Term von Ordnung $O(\ln^s N)$ während die Ordnung des zweiten Term nur $O(\ln^{s-1} N)$ beträgt und damit für unsere Abschätzung relevant ist.

Die Anzahl der Punkte $(j_1, \ldots j_k)$, die $\prod_{i=1}^{k} p_i^{j_i+1} \leq N$ erfüllen ist gleichbedeutend mit der Anzahl der Punkte, welche die lineare Ungleichung $\sum_{i=1}^{k} (j_i + 1) \ln p_i \leq \ln N$ erfüllen. Letzteres ist genau die Anzahl der Punkte im von den Achsenebenen (eine Komponente $= 0$) und der Hyperebene $\sum_{i=1}^{k} (j_i + 1) \ln p_i = \ln N$ eingeschlossenen Simplex (bzw. verallgemeinerter "Pyramide") mit Volumen

$$\frac{1}{k!} \frac{\ln^k N}{\ln p_1 \ldots \ln p_k} + O\left(\ln^{k-1} N\right) \quad (2.24)$$

Dieses Volumen ist demnach (da außerdem $j_i + 1 > 0$ gilt) auch eine obere Schranke für die Punktezahl.

Die Anzahl der Punkte in B_k ist sicherlich geringer als ohne die zusätzliche Bedingung $\prod_{i=1}^{k+1} p_i^{j_i+1} > N$ und damit ist die Anzahl der k-Tupel $(j_1, \ldots j_k)$, für die $(j_1, \ldots j_k, \ldots j_s) \in B_k$ gelten kann, durch diese Abschätzung beschränkt. Über die Indices $k+1, \ldots s$ wird nun summiert.

Es sei durch $(j_1, \ldots j_k)$ ein derartiges k-Tupel gewählt, und durch Summation über die restlichen Indices $j_{k+1}, \ldots j_s$ gilt:

$$\sum_{(j_{k+1}, \ldots j_s)} \varepsilon(\mathbf{j}) \left(A_N(I(\mathbf{j})) - N\mu(I(\mathbf{j}))\right) = \varepsilon \left(A_N(\pi) - N\mu(\pi)\right), \varepsilon = \pm 1 \quad (2.25)$$

mit einem Intervall π der Form $\pi = \prod_{i=1}^{k} I_{j_i}^{(i)} \times \pi_{k+1} \times \prod_{k+2}^{s} \left[0, z^{(i)}\right)$. Der Index $k+1$ nimmt dabei aufgrund der zusätzlichen Bedingung eine Sonderstellung ein. Das Intervall π_{k+1} ist ein Teilintervall von $\left[c p_{k+1}^{-r+1}, (c+1) p_{k+1}^{-r+1}\right)$ wobei c eine natürliche Zahl und r die kleinste natürliche Zahl mit $\prod_{i=1}^{k} p_i^{j_i+1} p_{k+1}^{r+1} > N$ bezeichnet (für ein kleineres r ist $(j_1, \ldots j_k, r, j_{k+2}, \ldots j_s) \in B_{k+i}$, $i \geq 1$).

Für dieses π gilt nun $|A_N(\pi) - N\mu(\pi)| \leq p_1 \ldots p_k p_{k+1}^2$, also Beschränktheit, weshalb lediglich die Anzahl der k-Tupel, die ja von Ordnung $O(\ln^k N)$ ist, relevant ist. Da $k \leq s - 1$, ist der gesamte zweite Term maximal von Ordnung $O(\ln^{s-1} N)$.

Für den ersten Term sei $\mathbf{j} \in B$ gewählt. Da wie oben erwähnt $j_i < n_i - 1$ ist, sind die I_{j_I} Intervalle der Form $\left[z_{j_i}^{(i)}, z_{j_i}^{(i)} + a_k^{(i)} p_i^{-k}\right)$ für $a_k^{(i)} > 0$, analog für $a_k^{(i)} < 0$, und $I(\mathbf{j}) = \prod_{i=1}^{s} I_{j_i}^{(i)}$ kann demnach in $\prod_{i=1}^{s} \left|a_{j_i}^{(i)}\right|$ elementare Intervalle geteilt werden. Für ein elementares Intervall

$\tilde{J}$ gilt aber $\left|A_N(\tilde{J}) - N\mu(\tilde{J})\right| \leq 1$ und damit die Abschätzung

$$\left|\sum_{\mathbf{j}\in B} \varepsilon(\mathbf{j})\left(A_N(I(\mathbf{j})) - N\mu(I(\mathbf{j}))\right)\right| \leq \sum_{\mathbf{j}\in B} \left|A_N(I(\mathbf{j})) - N\mu(I(\mathbf{j}))\right| \leq$$
$$\sum_{\mathbf{j}\in B} 1 \cdot \prod_{i=1}^{s} \left|a_{j_i}^{(i)}\right| = \prod_{i=1}^{s} \left|a_{j_i}^{(i)}\right| \cdot \left(\frac{1}{s!}\frac{\ln^s N}{\ln p_1 \ldots \ln p_s} + O\left(\ln^{s-1} N\right)\right) \quad (2.26)$$

Die von ATANASSOV in [Ata00] gegebene Abschätzung

$$\prod_{i=1}^{s} \left|a_{j_i}^{(i)}\right| \leq \prod_{i=1}^{s} \frac{p_i - 1}{2} \quad (2.27)$$

ist allerdings nur bedingt richtig, und zwar nur, wenn alle p_i ungerade sind. Für ungerade p_i gilt schon wegen der Definition der $a_{j_i}^{(i)}$ und der Ganzzahligkeit, dass $\left|a_{j_i}^{(i)}\right| \leq \frac{p_i-1}{2}$. Für gerades p_i (und nur eine einzige Basis kann gerade sein), gilt allerdings $a_{j_i}^{(i)} \leq \frac{p_i}{2}$. Dass diese Schranke nicht verbessert werden kann, ist leicht einsichtig, da die Wahl $a_{j_i}^{(i)} = \frac{p_i-1}{2}$ für ungerade p_i und $a_{j_i}^{(i)} = \frac{p_i}{2}$ für gerade p_i für mindestens ein $\mathbf{j}$ zulässig ist. Im Extremfall $p_i = 2$ entspricht diese korrigierte Abschätzung einem zusätzlichen Faktor $f = 2$, mit steigendem geraden p_i sinkt dieser Faktor, bis er asymptotisch irrelevant wird. Für die folgende Abschätzung muss aber $f = 2$ benutzt werden.

Insgesamt ergibt sich also

$$|A_N(J) - N\mu(J)| \leq \prod_{i=1}^{s} \left|a_{j_i}^{(i)}\right| \left(\frac{1}{s!}\frac{\ln^s N}{\ln p_1 \ldots \ln p_s} + O(\ln^{s-1} N)\right) + O(\ln^{s-1} N) \leq$$
$$2\prod_{i=1}^{s} \frac{p_i - 1}{2} \frac{1}{s!}\frac{\ln^s N}{\ln p_1 \ldots \ln p_s} + O(\ln^{s-1} N) = 2\frac{1}{s!}\prod_{i=1}^{s} \frac{p_i - 1}{2\ln p_i} \ln^s N + O(\ln^{s-1} N) \quad (2.28)$$

Da die rechte Seite nicht von der Wahl von J abhängt, gilt diese Abschätzung auch für das Supremum über alle J, womit der Satz bewiesen ist. □

Bemerkung 2.2. Der zusätzliche Faktor 2 in der Abschätzung beeinträchtigt aufgrund seiner Konstanz natürlich nicht das Verhalten der Abschätzung für $s \to \infty$.

Bemerkung 2.3. Werden für die b_i die ersten s Primzahlen gewählt, so ist

$$\lim_{s\to\infty} \tilde{A}_s\left(p_1, \ldots b_s\right) = 0 \quad (2.29)$$

Beweis. Es gilt

$$c_s = \frac{1}{s!}\prod_{i=1}^{s} \frac{p_i}{\ln p_i} = \prod_{i=1}^{s} \frac{1}{i}\left(1 - \frac{1}{p_i}\right)\frac{p_i}{\ln p_i} \leq \prod_{i=1}^{s}\left(1 - \frac{1}{p_i}\right) \quad (2.30)$$

Letztere Abschätzung gilt aufgrund der Ungleichung $\pi(x)\ln x > x$ für die Anzahl $\pi(x)$ der Primzahlen kleiner gleich x. Letzteres Produkt strebt außerdem für $s \to \infty$ gegen 0. □

Für eine weitere Verbesserung obiger Schranke wird folgende Eigenschaft benötigt, die Atanassov mit R bezeichnet.

Definition 2.4 (R-Eigenschaft). Es seien $p_1, \ldots p_s$ verschiedene Primzahlen und g_i, $i = 1, \ldots s$ so gewählt, dass g_i jeweils eine primitive Wurzel $(\mathrm{mod} p_i)$ darstellt. Damit sind a_{ij} definiert durch $g_i^{a_{ij}} \equiv p_j \,(\mathrm{mod} p_i)$ für $i \neq j$ und $a_{ii} = 0$. Wenn das lineare Gleichungssystem

$$a_{i1}x_1 + \ldots a_{is}x_s + (p_1 - 1)y_i = b_i, i = 1, \ldots s \tag{2.31}$$

für alle $(b_1, \ldots b_s)$ zumindest 1 ganzzahlige Lösung hat, so besitzen die Primzahlen $p_1, \ldots p_s$ die R-Eigenschaft.

Satz 2.5 (Diskrepanz der Halton-Folge, Atanassov [Ata00]). *Es sollen die Primzahlen $p_1, \ldots p_s$ die R-Eigenschaft besitzen. Dann gilt für die Diskrepanz der Halton-Folge S in den Basen $p_1, \ldots p_s$*

$$D_N^*(S) < \tilde{A}_s(p_1, \ldots p_s)\frac{\ln^s N}{N} + O\left(\frac{\ln^{s-1} N}{N}\right)$$

mit

$$\tilde{A}_s(p_1, \ldots p_s) = \frac{1}{s!}\left(1 + \sum_{i=1}^{s} \ln p_i\right) \prod_{i=1}^{s} \frac{p_i(2 + \ln p_i)}{(p_1 - 1)\ln p_i} \tag{2.32}$$

Für den Beweis benötigen wir die folgenden Notationen

$$\Delta_p(x, y) = \begin{cases} 1 & \text{wenn } x \equiv y \,(\mathrm{mod} p) \\ 0 & \text{sonst} \end{cases} \tag{2.33}$$

$$M(p_1, \ldots p_s) = \{(j_1, \ldots j_s) \in \mathbb{Z}^s : 0 \leq j_i \leq p_i - 1, (j_1, \ldots j_s) \neq \mathbf{0}\} \tag{2.34}$$

$$R(j_1, \ldots j_s) = r_1(j_1) \ldots r_j(j_s) \tag{2.35}$$

$$r_i(j_i) = \begin{cases} 1 & \text{für } j = 0 \\ 2j & \text{für } j \leq \frac{1}{2}(p_1 - 1) \\ 2(p_i - j) & \text{für } j > \frac{1}{2}(p_1 - 1) \end{cases} \tag{2.36}$$

$$S_N(j_1, \ldots j_s, S) = \sum_{n=0}^{N-1} exp\left(1\pi i \sum_{k=1}^{s} \frac{j_k x_n^{(k)}}{p_k}\right) \tag{2.37}$$

$$\langle x \rangle = \min\{|n - x| : n \in \mathbb{Z}\} \in I \ldots \quad \text{Abstand zur nächsten ganzen Zahl} \tag{2.38}$$

Weiters werden folgende beiden Lemmata benötigt:

Lemma 2.6. *Es seien $p_1, \ldots p_s$ verschiedene Primzahlen und $\sigma = (\mathbf{x}_n)_{n=1}^{N-1}$ eine Folge in $\mathbb{Z}^s$ mit*

$$\delta_N(b_1, \ldots b_s) = \left|\left\{\mathbf{x}_n \in \sigma : \forall i \exists l_i \in \{0, \ldots b_1 - 1\} : x_n^{(i)} \equiv l_i \,(\mathrm{mod} p_i)\right\}\right| \tag{2.39}$$

Dann gilt

$$\sup_{b_1, \ldots b_s \in \mathbb{N}} \left|\delta_N(b_1, \ldots b_s) - N\frac{b_1 \ldots b_s}{p_1 \ldots p_s}\right| \leq \sum_{\mathbf{j} \in M(p_1, \ldots p_s)} \frac{|S_N(j_1, \ldots j_s, \sigma)|}{R(j_1, \ldots j_s)} \tag{2.40}$$

Beweis. Unter Verwendung von $\frac{1}{p_i}\sum_{j=1}^{p_i-1} e^{2\pi i j\frac{x-y}{p_i}} = \Delta_{p_i}(x,y)$ kann

$$\delta_N(b_1,\ldots b_N) = \sum_{n=0}^{N-1}\prod_{k=1}^{s}\sum_{l_k=0}^{b_k-1}\Delta_{p_k}(x_n^{(k)},l_k)$$

dargestellt werden als

$$\delta_N(b_1,\ldots b_s) - \frac{Nb_1\ldots b_s}{p_1\ldots p_s} = \sum_{\mathbf{v}\in M(p_1,\ldots p_s)} S_N(j_1,\ldots j_s,\sigma)\prod_{k=1}^{s}\frac{1}{p_k}\sum_{l_k=0}^{b_k-1} e^{-2\pi i j_k\frac{l_k}{p_k}}$$

Für jeden der Faktoren $\sum_{l_k=0}^{b_k-1}\frac{\exp(-2\pi i k j_k\frac{l_k}{p_k})}{p_k}$ gilt nun für $j_k\neq 0$ (für $j_k=0$ ist folgende Behauptung trivialerweise wahr):

$$\begin{aligned}\frac{1}{p_k}\left|\sum_{l_k=0}^{b_k-1}\exp(-2\pi i k\frac{j_k}{p_k})^{l_k}\right| &= \frac{1}{p_k}\left|\frac{1-\exp(-2\pi i k\frac{j_k}{p_k})^{b_k}}{1-\exp(-2\pi i k\frac{j_k}{p_k})}\right| \\ \leq \frac{2}{p_k\left|1-\exp(-2\pi i k\frac{j_k}{p_k})\right|} &= \frac{2}{p_k 2\left|\sin(-\pi k\frac{j_k}{p_k})\right|\left|\cos(-\pi k\frac{j_k}{p_k})+i\sin(-\pi k\frac{j_k}{p_k})\right|} \\ = \frac{1}{p_k\left|\sin(-\pi\left\langle k\frac{j_k}{p_k}\right\rangle)\right|} &\leq \frac{1}{p_k 2\left\langle k\frac{j_k}{p_k}\right\rangle} = \frac{1}{r(j_k)}\end{aligned}$$

und damit gilt die Behauptung. □

Lemma 2.7. *Seien $p_1,\ldots p_s$ verschiedene Primzahlen und als $g_1,\ldots g_s$ jeweils primitive Wurzeln $(\mathrm{mod} p_i)$ gewählt. Dann gilt:*

$$\begin{aligned}S = \frac{1}{p_i-1}\sum_{m_1=1}^{p_1-1}\cdots\frac{1}{p_s-1}\sum_{m_s=1}^{p_s-1}\sum_{\mathbf{j}\in M(p_1,\ldots p_s)}\frac{1}{2R(\mathbf{j})\left\langle\sum_{i=1}^{s}\frac{j_i m_i}{p_i}\right\rangle} \\ \leq \prod_{i=1}^{s}\frac{(2+\ln p_i)}{p_i-1}\left(1+\sum_{i=1}^{s}\ln p_i\right) \qquad (2.41)\end{aligned}$$

Beweis. Es sei $P = p_1\ldots p_s$. In einem ersten Schritt wird gezeigt, dass

$$\frac{1}{\left\langle\sum_{i=1}^{s}\frac{j_i m_i}{p_i}\right\rangle} = P\sum_{r=1}^{P-1}\frac{1}{\min(r,P-r)}\Delta_p\left(\sum_{i=1}^{s}\frac{j_i m_i P}{p_i},r\right)$$

Da nur für eine einzige Zahl $1\leq r\leq P-1$ für fixe j_i, m_i und p_i gilt, dass $\sum_{i=1}^{s}\frac{j_i m_i}{p_i}\equiv r\,(\mathrm{mod}P)$, wirkt das Δ_P auf der rechten Seite wie ein Kronecker-Delta und liefert nur für einen einzigen Wert von r den Wert 1. Für die linke Seite wird benutzt, dass

$$\left\langle\frac{a}{b}\right\rangle = \frac{1}{b}\min\left(a\,(\mathrm{mod}b)\,,b-a\,(\mathrm{mod}b)\right)\,,$$

sodass diese umgeschrieben werden kann zu:

$$\left\langle\frac{\sum_{i=1}^{s}\frac{j_i m_i P}{p_i}}{P}\right\rangle^{-1} = \frac{P}{\min\left(\sum_{i=1}^{s}\frac{j_i m_i P}{p_i}\,(\mathrm{mod}P)\,,P-\sum_{i=1}^{s}\frac{j_i m_i P}{p_i}\,(\mathrm{mod}P)\right)} \qquad (2.42)$$

Durch das Δ_P wird nun gleichzeitig das $(\mathrm{mod} P)$ behandelt und nur jenes r ausgewählt, das zu dieser Summe äquivalent ist. Damit gilt diese Gleichung.

Sind andererseits r und j_i fixiert, so beschreibt Δ_P eine lineare Kongruenz, welche eine eindeutige Lösung besitzt für $j_i \neq 0$, sodass aus den Summen $\sum_{m_i=1}^{p_i-1}$ nur jeweils ein Term von Bedeutung ist.

Damit gilt nun unter Verwendung von $\sum_{i=1}^{m-1} \frac{1}{\min(i,m-i)}\, 2 \sum_{i=1}^{\lfloor \frac{m}{2} \rfloor} \frac{1}{i} \leq 2\,(\ln m + 1)$:

$$\frac{1}{p_1 - 1} \sum_{m_1=1}^{p_1-1} \cdots \frac{1}{p_s - 1} \sum_{m_1=1}^{p_s-1} \frac{P}{R(\mathbf{j})} \sum_{r=1}^{P-1} \frac{\Delta_P\left(\sum_{i=1}^{s} \frac{j_i m_i P}{p_i}, r\right)}{2\min(r, P-r)} =$$

$$\prod_{i=1}^{s} \frac{p_i}{p_i - 1} \frac{1}{R(\mathbf{j})} \sum_{r=1}^{P-1} \frac{\Delta_P\left(\sum_{i=1}^{s} \frac{j_i \tilde{m}_i P}{p_i}, r\right)}{2\min(r, P-r)} =$$

$$\frac{1}{R(\mathbf{j})} \prod_{i=1}^{s} \frac{p_i}{p_i - 1} \sum_{r=1}^{P-1} \frac{1}{2\min(r, P-r)} = \frac{1}{R(\mathbf{j})} \prod_{i=1}^{s} \frac{p_i}{p_i - 1} \sum_{r=1}^{P-1} (\ln P + 1) \tag{2.43}$$

und durch Summation über alle j:

$$S \leq \sum_{\mathbf{j} \in M(p_1, \ldots p_s)} \frac{1 + \ln P}{R(\mathbf{j})} \prod_{i=1}^{s} \frac{p_i}{p_i - 1} =$$

$$\left(1 + \sum_{i=1}^{s} \ln p_i\right) \prod_{i=1}^{s} \frac{p_i}{p_i + 1} \prod_{i=1}^{s} \left(1 + \sum_{j_i=1}^{p_i-1} \frac{1}{2\min(j_i, p_i - j_i)}\right)$$

$$\leq \left(1 + \sum_{i=1}^{s} \ln p_i\right) \prod_{i=1}^{s} \frac{p_i}{p_i - 1} (2 + \ln p_i) = \left(1 + \sum_{i=1}^{s} \ln p_i\right) \prod_{i=1}^{s} \frac{p_i\,(2 + \ln p_i)}{p_i - 1} \tag{2.44}$$

□

Beweis des Satzes 2.5. Ausgehend von der Zerlegung $z^{(i)} = \sum_{j=0}^{\infty} a_j^{(i)} p_i^j$, $0 \leq a_j^{(i)} \leq p_i - 1$, also der üblichen Darstellung in Basis p_I wird ganz analog zum Beweis von Satz 2.4 vorgegangen (die $\varepsilon(j_1, \ldots j_s)$ sind nun überflüssig, das Intervall wird als disjunkte Vereinigung dargestellt), der Fehler wieder in die beiden Terme zerlegt, wobei der zweite analog von Ordnung $O\left(\ln^{s-1} N\right)$ ist.

Für den ersten Term $\sum_{\mathbf{j} \in B(N)} (A_N(I(\mathbf{j})) - N\mu(I(\mathbf{j})))$ wird wieder ein beliebiges, aber festes $\mathbf{j} \in B(N)$ betrachtet. Das Intervall $I(j_1, \ldots j_s)$ ist dann von der Form

$$I(j_1, \ldots j_s) = \prod_{i=1}^{s} \left[c_i p_i^{-j_i}, c_i p_i^{-j_i} + b_i p_i^{-j_i - 1}\right)$$

mit $b_i = a_{j_i}^{(i)}$. Es sei nun $P_i(j_1, \ldots j_s) := \prod_{k \neq i} p_i^{j_k}$ definiert sowie eine Folge $\omega = \{\mathbf{x}_n\}_{n=0}^{\infty} \subset \mathbb{Z}^s$ durch

$$x_n^{(i)} = d_i + n P_i(j_1, \ldots j_s)$$

Dabei seien d_i jene ganzen Zahlen, für die gilt, dass das erste Element von σ, das in $G = \prod_{i=1}^{s} \left[c_i p_i^{-j_i}, (c_i + 1) p^{-j_i}\right)$ liegt, auch im Teilintervall

$$\prod_{i=1}^{s} \left[c_i p_i^{-j_i} + d_i p_i^{-j_i - 1}, c_i p_i^{-j_i} + (d_i + 1) p_i^{-j_i - 1}\right)$$

liegt. Wenn nun K die Anzahl der Elemente in G unter den ersten N Elementen von σ bezeichnet, gilt $N\mu(I(j_1,\ldots j_s)) = K\frac{b_1\cdot\ldots\cdot b_s}{p_1\cdot\ldots\cdot p_s}$ und weiters

$$A_N\left(I(j_1,\ldots j_s)\right) = \delta_K(b_1,\ldots b_s,\omega) \tag{2.45}$$

Damit gilt

$$\Delta_N(\mathbf{j}) = A_N\left(I(\mathbf{j})\right) - N\mu\left(I(\mathbf{j})\right) = \delta_K(b_1,\ldots b_d s,\omega) - K\frac{b_1\ldots b_s}{p_1\ldots p_s} \tag{2.46}$$

und mit Lemma 2.6

$$\begin{aligned}|\Delta_N(\mathbf{j})| &< \sum_{\mathbf{r}\in M(\mathbf{p})} \frac{|S_K(\mathbf{r},\omega)|}{R(\mathbf{r})} < \sum_{\mathbf{r}\in M(\mathbf{p})} \frac{1}{R(\mathbf{r})}\left|\sum_{n=0}^{N-1} e^{2\pi i\sum_{k=1}^{s}\frac{j_k(d_i+nP_i(\mathbf{j}))}{p_k}}\right| \\ &< \sum_{\mathbf{r}\in M(\mathbf{p})} \frac{1}{2R(\mathbf{r})\left\langle\sum_{i=1}^{s}\frac{r_i}{p_i}P_i(\mathbf{j})\right\rangle}\end{aligned} \tag{2.47}$$

Letztere Ungleichung gilt wegen der Argumentation bei Lemma 2.6. Insgesamt ergibt sich damit für die erste Summe

$$\sum_{\mathbf{j}\in B(N)}\left(A_N(I(\mathbf{j})) - N\mu(I(\mathbf{j}))\right) \le \sum_{\mathbf{j}\in B(N)}\sum_{\mathbf{r}\in M(\mathbf{p})} \frac{1}{2R(\mathbf{r})\left\langle\sum_{i=1}^{s}\frac{r_i}{p_i}P_i(\mathbf{j})\right\rangle} \tag{2.48}$$

Die Summe über alle $\mathbf{j} \in B(N)$ wird nun weiter aufstrukturiert in Summen über $m_1 = 1\ldots p_i - 1$ für $i = 1,\ldots s$, wobei $P_i(\mathbf{j}) \equiv m_i\,(\mathrm{mod} p_i)$ gewählt wird. Damit ist $m_i = 0$ nicht von Bedeutung, da dann $\frac{r_iP_i(\mathbf{j})}{p_i}$ ganzzahlig und damit irrelevant ist. Wegen der Division durch p_i kann außerdem $P_i(\mathbf{j})$ durch m_i ersetzt werden (jedes Vielfache von p_i trägt nur mit einer ganzen Zahl bei, die wegen dem $\langle\cdot\rangle$ wegfällt). Die Anzahl aller $\mathbf{j}$ mit $m_i \equiv P_i(\mathbf{j})$ für $i = 1,\ldots s$ wird nun mit $\zeta_N(m_1,\ldots m_s)$ bezeichnet. Somit lässt sich also schreiben

$$\left|\sum_{\mathbf{j}\in B(N)}\Delta_N(\mathbf{j})\right| \le \sum_{m_1=1}^{p_1-1}\cdots\sum_{m_s=1}^{p_s-1}\nu_N(m_1,\ldots m_s)\sum_{\mathbf{r}\in M(\mathbf{p})}\frac{1}{2R(\mathbf{r})\left\langle\sum_{i=1}^{s}\frac{m_ir_i}{p_i}\right\rangle} \tag{2.49}$$

Es seien nun g_i als primitive Wurzeln $(\mathrm{mod} p_i)$ gewählt, sie erfüllen also $g_i^{p_i-1} = 1$. Dazu seien b_i und a_{ij} folgendermaßen gewählt:

$$b_i \text{ sodass } m_i = g_i^{b_i} \text{ für } 1 \le i \le s \text{ gilt} \tag{2.50}$$

$$a_{ij} \text{ sodass } g_i^{a_{ij}} \equiv p_j\,(\mathrm{mod} p_i) \text{ für } i \ne j \text{ und } a_{ii} = 0 \text{ gilt} \tag{2.51}$$

Nun gilt:

$$\begin{aligned} m_i \equiv P_i(j_1,\ldots j_s)\,(\mathrm{mod} p_i) \quad &\Longleftrightarrow\ (j_1,\ldots j_s) \text{ löst das System (2.31),} \\ &\quad \text{also } a_{i1}j_1 + \ldots + a_{is}j_s + (p_i-1)y_i = b_i \end{aligned}$$

Beweis: "⇒": Es seien $m_i \equiv P_i(j_1,\ldots j_s) = p_1^{j_1}\ldots p_{i-1}^{j_{i-1}}p_{i+1}^{j_{i+1}}\ldots p_s^{j_s}\,(\mathrm{mod} p_i)$ bzw. gleichbedeutend

$$g_i^{b_i} \equiv g_i^{a_{i1}j_1}\ldots g_i^{a_{i(i-1)}j_{i-1}}g_i^{0j_i}g_i^{a_{i(i+1)}j_{i+1}}\ldots g_i^{a_{is}j_s} = g_s^{\sum_{k=1}^{s}a_{ik}j_k - b_i}\,(\mathrm{mod} p_i)\ . \tag{2.52}$$

Dies bedeutet aber

$$np_i = g_i^{b_i}\left(1 - g_i^{\sum_{k=1}^{s} a_{ik}j_k - b_i}\right) \tag{2.53}$$

und da g_i eine primitive Wurzel $(\mathrm{mod} p_i)$ ist

$$\tilde{n}p_i = 1 - g_i^{\sum_{k=1}^{s} a_{ik}j_k - b_i} \tag{2.54}$$

Da für den Wert obiger Gleichung nur 0 in Betracht kommt (da $1 \leq g_i^x < p_i\,(\mathrm{mod} p_i)$), gilt $1 = g_i^{\sum_{k=1}^{s} a_{ik}j_k - b_i}$ bzw. $\sum_{k-1}^{s} a_{ik}j_k - b_i = \tilde{y}(p_i - 1)$, weil $g_i$eine primitive Wurzel $(\mathrm{mod} p_i)$ ist. Dies ist aber genau das gewünschte Gleichungssystem.

"$\Leftarrow$": Obige Argumentation gilt natürlich auch in die andere Richtung, womit die Behauptung gezeigt wäre. qed.

Da sich die Lösungen eines inhomogenen linearen Gleichungssystems nur um Lösungen des homogenen Systems unterscheiden, beinhaltet jede der Mengen

$$\{(j_1, \ldots j_s) \in \mathbb{Z}^s : m_i(p_1 - 1)\ldots(p_s - 1) \leq j_i < (m_i + 1)(p_1 - 1)\ldots(p_s - 1)\} \tag{2.55}$$

für alle $(m_1, \ldots m_s)$ dieselbe Anzahl von Lösungen. Die Gesamtzahl von $\mathbf{j} \in B(N)$ beträgt (analog zu Satz 2.4) $\frac{1}{s!}\frac{\ln^s N}{\ln p_1 \ldots \ln p_s} + O\left(\ln^{s-1} N\right)$ und damit gilt für ν:

$$\nu_N(m_1, \ldots m_s) = \frac{\ln^s N}{s! \ln p_1 \ldots \ln p_s}\frac{1}{(p_1 - 1)\ldots(p_s - 1)} + O\left(\ln^{s-1} N\right) \tag{2.56}$$

Damit gilt nun insgesamt mit Lemma 2.7

$$A_N(J) - N\mu(J) \leq \tag{2.57}$$

$$\leq \sum_{m_1=1}^{p_1-1} \cdots \sum_{m_s=1}^{p_s-1} \frac{\ln^s N}{s! \ln p_1 \ldots \ln p_s} \prod_{i=1}^{s} \frac{1}{p_i - 1} \sum_{\mathbf{r} \in M(\mathbf{p})} \frac{\left\langle \sum_{i=1}^{s} \frac{m_i r_i}{p_i} \right\rangle^{-1}}{2R(\mathbf{r})} + O\left(\ln^{s-1} N\right) \tag{2.58}$$

$$= \frac{\ln^s N}{s! \ln p_1 \ldots \ln p_s} \frac{1}{p_1 - 1} \sum_{m_1=1}^{p_1-1} \cdots \frac{1}{p_s - 1} \sum_{m_s=1}^{p_s-1} \sum_{\mathbf{r} \in M(\mathbf{p})} \frac{\left\langle \sum_{i=1}^{s} \frac{m_i r_i}{p_i} \right\rangle^{-1}}{2R(\mathbf{r})} + O\left(\ln^{s-1} N\right) \tag{2.59}$$

$$\leq \left(\frac{1}{s!} \prod_{i=1}^{s} \frac{(2 + \ln p_i)p_i}{(p_i - 1)\ln p_i}\left(1 + \sum_{i=1}^{s} \ln p_i\right)\right) \ln^s N + O\left(\ln^{s-1} N\right) . \tag{2.60}$$

□

Da die Eigenschaft R natürlich nicht für jede Wahl der Primzahlen $p_1, \ldots p_s$ gegeben ist (laut Atanassov allerdings für $s \leq 10$ immer und für $1 \leq s \leq 100$ genau 68 mal wenn die ersten s Primzahlen als Basen benutzt werden), ist eine allgemeine Konstruktion gesucht, die für jede Wahl von $p_1, \ldots p_s$ obige Diskrepanz-Schranke (2.32) aufweist. Folgende modifizierte Halton-Folge erfüllt dies:

Definition 2.5 (modifizierte Halton-Folge, Atanassov[Ata00]). Es seien verschiedene Primzahlen $p_1, \ldots p_s$ als Basen gewählt, $g_1, \ldots g_s$ jeweils primitive Wurzeln $(\mathrm{mod} p_i)$ und

$a_j(n)$ die Koeffizienten der Entwicklung von n in Basis p_j. Die Halton Folge $S = (\mathbf{x}_n)_{n\in\mathbb{N}}$ in den Basen $p_1, \ldots p_s$ mit den Modifikatoren $k_1, \ldots k_s$ ist dann definiert durch

$$x_n^{(i)} = \sum_{j=1}^{\infty} b_{ji} p_i^{-j-1} \tag{2.61}$$

$$b_{ji} = a_j(n) g_i^{k_j} \,(\mathrm{mod} p_i) \tag{2.62}$$

Für diese modifizierten Halton-Folge kann nun bedingungslos obige Schranke aus Gleichung (2.32) gezeigt werden:

Satz 2.8 (Diskrepanz der modif. Halton-Folge, Atanassov). *Es seien $p_1, \ldots p_s$ verschiedene Primzahlen und g_i jeweils eine primitive Wurzel $(\mathrm{mod} p_i)$ für $i = 1, \ldots s$. Weiters seien die $(k_1, \ldots k_s)$ nach folgendem Lemma 2.9 gewählt. Für die modifizierte Halton Folge $\tilde{S}$ in den Basen $p_1, \ldots p - s$ mit den Modifikatoren $k_1, \ldots k_s$ gilt dann:*

$$ND_N(\tilde{S}) < \frac{2^s}{s!} \prod_{i=1}^{s} \frac{p_i(2+\ln p)}{(p_i-1)\ln p_i} \left(1 + \sum_{i=1}^{s} \ln p_i\right) \ln^s N + O\left(\ln^{s-1} N\right) \tag{2.63}$$

Lemma 2.9 (Atanassov). *Es seien p_i, g_i, a_{ij} für $i, j = 1, \ldots s$ wie für die Eigenschaft R aus Definition 2.4 gewählt. Dann existieren natürliche Zahlen $k_1, \ldots k_s$, sodass folgendes System von diophantischen Gleichungen*

$$a_{1i}x_1 + \ldots + k_i x_i + \ldots + a_{is}x_s + (p_i - 1)y_i = b_i,\ i = 1, \ldots s \tag{2.64}$$

für jede Wahl von $b_1, \ldots b_s$ ganzzahlige Lösungen $(x_1, \ldots x_s, y_1, \ldots y_s)$ besitzt.

Beweis. Obige Gleichung kann auch in Matrixform geschrieben werden

$$\mathbf{Ax} + (\mathbf{p} - \mathbf{1})\mathbf{y})_\mathbf{i} = \mathbf{b}$$

Nun können die k_i so gewählt werden, dass $\det A = 1$ und damit A^{-1} ganzzahlig ist:

Der Beweis läuft über Induktion nach der Dimension der Matrix und unter Verwendung des Entwicklungssatzes der Determinante von Laplace: $\det A = k_1 A_{11} - a_{21}A_{21} + \ldots + (-1)^s a_{s1}A_{s1}$. Die Matrix für den Minor $A_1 1$ hat dabei dieselbe Form wie A, nur mit einer um 1 kleineren Dimension. Nach Induktionsvoraussetzung ist damit $A_{11} = 1$ und es kann $k_1 = 1 - (-a_{21}A_{21} + \ldots + (-1)^s a_{s1}A_{s1})$ gewählt werden.

Als Induktionsanfang ist die Matrix $A = (k_1)$ zu betrachten, für die trivialerweise $k_1 = 1$ gesetzt werden kann.

Damit gilt nun

$$\mathbf{x} = -\mathbf{A}^{-1}((\mathbf{p} - 1)\mathbf{y}) + \mathbf{A}^{-1}\mathbf{b}$$

wobei wegen der Ganzzahligkeit von $\mathbf{A}^{-1}$ bei beliebiger ganzzahliger Wahl von $y_1, \ldots y_s$ und $b_1, \ldots b_s$ die Ganzzahligkeit der $x_1, \ldots x_s$ gewährleistet ist. □

Beweisskizze für Satz 2.8. Dieser Beweis verläuft völlig analog zum Beweis von Satz 2.5. Lediglich die benutzte Folge $\omega(j_1, \ldots j_s)$ muss abgeändert werden zu

$$x_n^{(i)} = g_i^{k_i j_i} P_i(j_1, \ldots j_s) n, \tag{2.65}$$

damit die Gleichung (2.46) erfüllt bleibt. Dadurch ändert sich weiters die Bedingung $m_i \equiv P_i(j_1, \ldots j_s) \,(\mathrm{mod} p_i)$ zu

$$m_i \equiv p_1^{j_1} \ldots p_{i-1}^{j_{i-1}} g_i^{k_i j_i} p_{i+1}^{j_{i+1}} \ldots p_s^{j_s} \,(\mathrm{mod} p_i) \;, \tag{2.66}$$

wodurch nicht mehr das System (2.31) äquivalent dazu ist, sondern das System (2.64). Da jedoch die k_i nach Voraussetzung derart gewählt sind, dass sie (2.64) erfüllen, verläuft der restliche Beweis wörtlich wie bei Satz 2.5. □

In Kapitel 5 werde ich einige direkte Vergleiche von Halton- und den im nächsten Abschnitt behandelten (t,s)-Folgen durchführen. Diese Experimente deuten auch darauf hin (bzw. bestätigen die hier angeführten Ergebnisse), dass die Halton-Folgen keineswegs schlechtere Ergebnisse als die (t,s)-Folgen liefern. Da für die bisher beste bekannte Abschätzung für die Halton-Folge allerdings die Konstante A_s mit $s \to \infty$ gegen ∞ strebte, ging man allgemein wegen der besseren bekannten Abschätzung davon aus, dass (t,s)-Folgen auch besser verteilt und damit bessere Ergebnisse liefern würden.

2.2.4 Die Hammersley-Menge

Um obige Schranke (2.10) noch weiter zu verbessern, kann man eine Koordinate einer s-dimensionalen Folge durch $\frac{n}{N}$ belegen, womit man dann im wesentlichen (zumindest bei Verwendung der Halton-Folge) die Diskrepanz der $s-1$-dimensionalen restlichen Folge erhält. Allerdings wird damit die Punktfolge zu einer endlichen Punktmenge, da die Mächtigkeit N der Menge zuvor festgelegt werden muss. Dies birgt auch einige Nachteile in sich, da damit die gesamte Punktmenge bei anderer Wahl von N neu berechnet werden muss. Allerdings ist nur eine Koordinate davon betroffen, die zudem relativ einfach zu berechnen ist, sodass sich der Aufwand in Grenzen hält. Dennoch muss man diese Tatsache bedenken, wenn man mehrere Simulationen mit derselben Punktfolge oder -menge durchführen möchte.

Lemma 2.10. *Für $s \geq 2$ sei S eine $(s-1)$-dimensionale Punktfolge $\mathbf{r}_0, \mathbf{r}_1, \ldots$ in $\bar{I}^s$. Bezeichnet man nun die Menge $\left\{\left(\frac{n}{N}, \mathbf{r}_n\right)\right) \in \bar{I}^s, 0 \leq n \leq N-1\right\}$ mit P, so gilt für die Sterndiskrepanz von P:*

$$N D_N^*(P) \leq \max_{1 \leq M \leq N} M D_M^*(S) + 1 \tag{2.67}$$

Beweis. Es sei $J = \prod_{i=1}^{s} [0, u_i) \subseteq \bar{I}^s$ beliebig gewählt. Dann gilt $(\frac{n}{N}, \mathbf{r}_n) \in J$, wenn $n < u_1 N$ und $(\mathbf{r}_n) \in \prod_{i=2}^{s} [0, u_i) = J'$ gilt. Die Diskrepanz von P lässt sich nun abschätzen durch

$$|A(J;P) - N\lambda_s(J)| \leq \left|A(J;P) - M\lambda_{s-1}(J')\right| + \left|M\lambda_{s-1}(J') - N\lambda_s(J)\right|$$

wobei M die kleinste ganze Zahl mit $Nu_1 \leq M$ bezeichnet. Weiters gilt $A(J;P) = A(J';S_M)$. Da u_1 in I gleichverteilt sein sollte, wird M alle möglichen Werte annehmen, und man kann den ersten Term abschätzen durch

$$\left|A(J;P) - M\lambda_{s-1}(J')\right| \leq \max_{0 \leq M < N} M D_M^*(S)$$

Der zweite Term kann durch 1 abgeschätzt werden:

$$\left|M\lambda_{s-1}(J') - N\lambda_s(J)\right| < \left|(Nu_1 + 1)\lambda_{s-1}(J') - N\lambda_s(J)\right| = \lambda_s(J) \leq 1$$

Damit ist obiges Lemma bewiesen. □

Die bekannteste derartige Folge ist die sogenannte Hammersley-Menge, die aus der Halton-Folge durch Hinzufügen von $\frac{n}{N}$ als zusätzliche Koordinate entsteht:

Definition 2.6 (Hammersley-Menge). Für die Dimension $s \geq 2$, die Anzahl $N \geq 1$ von Elementen und die Basen $p_1, \ldots p_{s-1} \geq 2$ ist die N-elementige Hammersley-Punktmenge in den Basen $p_1, \ldots p_{s-1}$ gegeben durch

$$\mathbf{r}_n = \left(\frac{n}{N}, \Phi_{p_1}(n), \ldots \Phi_{p_{s-1}}\right) \in \bar{I}^s \quad \text{für } 0 \leq n \leq N-1 \tag{2.68}$$

Aus Lemma 2.10 und Theorem 2.3 folgt sofort die Abschätzung für die Diskrepanz der Hammersley-Menge:

Satz 2.11. *Wenn P die N-elementige Hammersley-Menge in den paarweise relativ primen Basen $p_1, \ldots p_{s-1}$ darstellt, so gilt*

$$D_N^*(P) < \frac{s}{N} + \frac{1}{N} \prod_{i=1}^{s-1} \left(\frac{p_i - 1}{s \ln p_i} \ln N + \frac{p_i + 1}{2}\right) \tag{2.69}$$

$$= \left(\frac{\ln^{s-1} N}{N}\right) \prod_{i=1}^{s-1} \frac{p_i - 1}{2 \ln p_i} + O\left(\frac{\ln^{s-2} N}{N}\right) \tag{2.70}$$

Mit ATANASSOV's Abschätzung aus Satz 2.4 gilt sogar:

$$D_N^*(P) \leq \left(\frac{\ln^{s-1} N}{N}\right) \frac{1}{(s-1)!} \prod_{i=1}^{s-1} \frac{p_i - 1}{2 \ln p_i} + O\left(\frac{\ln^{s-2} N}{N}\right) \tag{2.71}$$

2.3 (t,s)-Folgen

Trotz der schon recht guten Eigenschaften (und der einfachen rekursiven Erstellung) der Halton-Folgen, werden in der Praxis für die Simulation meist andere Folgen benutzt. Dies vor allem deshalb, da die bis vor kurzem bekannten Diskrepanzschranken der Halton-Folge für hohe Dimensionen alles andere als befriedigend waren:

- Die Konstante A_s in der Diskrepanz-Formel (2.10) ging mit s überexponentiell gegen ∞ und war daher außer für kleine Dimensionen s praktisch unbrauchbar. Dasselbe gilt für die Hammersley-Menge

Durch die Konstruktion von Atanassov (bzw. auch schon durch die verbesserte Diskrepanzschranke der einfachen Halton-Folge) tritt nun dieses Problem auch bei der Halton-Folge nicht mehr auf. Dieses Problem tritt bei den sogenannten (s, t, m)-Netzen und den daraus abgeleiteten (s, t)-Folgen ebenfalls nicht mehr auf, die von NIEDERREITER ([Nie87] und [Nie88a]) erstmals definiert wurden als eine Verallgemeinerung von Konstruktionen von FAURE [Fau82] und SOBOL [Sob67]. Diese Netze besitzen die besten zur Zeit bekannten Eigenschaften bezüglich der Diskrepanz-Schranke und damit der Gleichverteilung.

Definition 2.7. Für fixe Dimension $s \geq 1$ und $p \geq 2$ ist ein *elementares Intervall E in Basis p* ein Teilintervall von $\bar{I}^s$ der Form

$$E = \prod_{i=1}^{s} \left[a_i p^{-d_i}, (a_i + 1) p^{-d_i}\right) \tag{2.72}$$

mit $a_i, d_i \in \mathbb{Z}$, $0 \leq d_i$ und $0 \leq a_i < p^{-d_i}$ für $1 \leq i \leq s$.

Definition 2.8 ((s,t,m)-Netz). Es seien $0 \leq t \leq m$ natürliche Zahlen. Ein *(t,m,s)-Netz in Basis p* ist eine Menge P von p^m Punkten $\in \bar{I}^s$ sodass $A(E;P) = p^t$ für alle elementaren Intervalle E in Basis p der Länge $\lambda_s(E) = p^{t-m}$.

Definition 2.9 ((t,s)-Folge). Sei $t \in \mathbb{N}_0$. Eine Folge $\mathbf{r}_0, \mathbf{r}_1, \ldots$ von Punkten in $\bar{I}^s$ wird eine *(s,t)-Folge in Basis p* genannt, wenn für alle $k \in \mathbb{N}_0$ und $m > t$ die Punktmenge, die aus den $\mathbf{r}_n$ besteht mit $kp^m \leq n < (k+1)p^m$ ein (t,m,s)-Netz in Basis p darstellt.

Bemerkung 2.4. Nach dieser Definition stellt auch die Van der Corput-Folge in Basis b eine $(0,1)$-Folge in Basis p dar: Die Auswahl $kp^m \leq n < (k+1)p^m$ für die Indices der Punkte der betrachteten Teilmengen bedeutet in der Entwicklung von n in Basis p, dass die letzten m Ziffern beliebige Werte annehmen können (und alle möglichen Werte auch annehmen, da n alle Zahlen $< N$ durchläuft), während die führenden Ziffern durch die Entwicklung von k festgelegt sind. Nach der Invertierung um das Komma durch die Funktion Φ_p bedeutet dies, dass die ersten m Ziffern nach dem Komma beliebig sind, während die restlichen Ziffern fixiert sind. Da die ersten m Ziffern nach dem Komma auch alle möglichen Werte annehmen, folgt, dass für jedes Intervall der Form $[a_i p^{-m}, (a_i+1)p^{-m})$ genau 1 Punkt $\mathbf{r}_j$ in diesem Intervall liegt. Damit ist aber die Van der Corput-Folge eine $(0,1)$-Folge.

Bemerkung 2.5. Stellt P ein (t,m,s)-Netz in Basis p dar, dann folgt aus $A(E_1;P) = p^t$ für $\lambda_s(E_1) = p^t$, dass für eine Union E von b^u disjunkten elementaren Intervallen mit $\lambda_s(E) = p^{t-m}$, dass $A(E;P) = p^u p^t = p^m \lambda_s(E)$. Damit stellt jedes (s,t,m)-Netz P in Basis p auch für jedes $t \leq u \leq\leq m$ ein (u,m,s)-Netz in Basis p und dementsprechend auch jede (t,s)-Folge für $u \leq t$ eine (u,s)-Folge dar. Kleinere Werte für t bedeuten also größere Regelmäßigkeit.

2.3.1 Diskrepanz von (t,m,s)-Netzen

Für die Diskrepanz lassen sich auch für (t,s)-Folgen einige Abschätzungen angeben. Die folgende allgemeine Abschätzung für $p \geq 3$ möchte ich beweisen, einige Verbesserungen für spezielle Werte von s nur erwähnen, die entsprechenden Beweise können in [Nie92] oder [DT97] nachgelesen werden.

Satz 2.12 ([Nie87] oder [Nie92]). *Für ein (t,m,s)-Netz P in Basis $p \geq 3$ gilt für die Sterndiskrepanz*

$$ND_N^*(P) \leq p^t \sum_{t=0}^{s-1} \binom{s-1}{i} \binom{m-t}{i} \left\lfloor \frac{p}{2} \right\rfloor^i \tag{2.73}$$

Für den Beweis allerdings benötigt man noch ein Hilfslemma über affine Transformationen auf das Netz:

Lemma 2.13 (o.B.). *Sei P eine (t,m,s)-Netz in Basis p und E ein elementares Intervall in Basis b mit $\lambda_s(E) = p^{-u}$, wobei $0 \leq u \leq m-t$, und $T : E \to \bar{I}^s$ eine affine Transformation. Dann werden die Punkte von P, die in E liegen, durch T in ein $(t,m-u,s)$-Netz transformiert.*

Beweis von Theorem (2.12). Im Folgenden sei die rechte Seite obiger Formel (2.73) bezeichnet durch $\Delta_p(t,m,s)$. Bei fixem $t \geq 0$ soll nun diese Ungleichung durch doppelte Induktion über $s \leq 1$ und $m \geq t$ bewiesen werden.

Induktionsbasis 1: Für $s = 1$ und ein beliebiges $m \geq t$ beachte man, dass ein beliebiges Intervall $J = [0,u) \in \mathfrak{I}^*$ aufgespalten werden kann in die Intervalle $J_h = \left[hp^{t-m}, (h+1)p^{t-m}\right)$ für $1 \leq h < k = \left\lfloor up^{m-t} \right\rfloor$ und das Intervall $J_k = \left[kp^{t-m}, u\right)$. Erstere Intervalle liefern nach

der Definition der (t,m,s)-Netze keinen Beitrag zur Diskrepanz, also $D(J) = D(J_k)$. Da für dieses Intervall allerdings $0 \leq \lambda_s J - kp^m \leq p^t$ und $0 \leq A(J_k) \leq p^t$ gilt, folgt

$$|D(J)| = |A(J_k) - \lambda_b(J_k)p^m| \leq p^t$$

und damit sofort die Induktionsbasis für alle $m \geq t$.

Induktionsschritt 1: Es gelte also Abschätzung (2.73) für $s-1$ und alle $m \geq t$. Mittels vollständiger Induktion nach m wird der Induktionsschritt bewiesen:

Induktionsbasis 2: Für $m = t$ is die Abschätzung trivial.

Induktionsschritt 2: Es wird also ein $(t, m+1, s)$-Netz betrachtet, wobei die Abschätzung für m gilt. Es muss dann für $N = p^{m+1}$ und jedes Intervall $J = \prod_{i=1}^{s} \left[0, u^{(i)}\right) \subseteq \bar{I}^s$ gezeigt werden, dass $|D(J)| \leq \Delta_p(t, m+1, s)$.

Fall 1: $u^{(s)} = 1$. In diesem Fall brauchen wir nur die ersten $s-1$ Koordinaten der Punkte des Netzes betrachten, die ein $(t, m+1, s-1)$-Netz darstellen, für das die Induktionsannahme gilt. Also

$$|D(J)| \leq \Delta_p(t, m+1, s-1) \leq \Delta_p(t, m+1, s)$$

Fall 2: $u^{(s)} < 1$. Es sei nun $l = \lfloor pu_s \rfloor$ mit $0 \leq l \leq p-1$.

Fall 2.1: $0 \leq l \leq \lfloor p/2 \rfloor$. In diesem Fall wird das Intervall J unterteilt in disjunkte Teilintervalle J_h, die sich nur in der letzten Koordinate unterscheiden: $J_h^{(s)} = \left[\frac{h}{p}, \frac{h+1}{p}\right)$ für $0 \leq h \leq l-1$ und $J_l^{(s)} = \left[\frac{l}{p}, u_s\right)$. Es gilt damit $D(J) = \sum_{h=0}^{l} D(J_h)$.

Für $h = l$ lässt sich dies abschätzen, indem $E_l = [0,1)^{s-1} \times [l/p, (l+1)/p)$ und eine affine Transformation $T_l : E_l \mapsto \bar{I}^s$ betrachtet wird. Dadurch werden die Punkte von P, die in E_l liegen, zu einem (t, m, s)-Netz P_2 in Basis p transformiert, für das $D(J_l; P) = D(T_l(J_l); P_2)$ gilt. Nach der Induktionsannahme gilt damit $|D(J_l; P)| \leq \Delta_p(t, m, s)$.

Für $0 \leq h < l$ transformiert die Projektion $T : I^s \mapsto I^{s-1}$ die Punkte von P, die in $E_h = [0,1)^{s-1} \times [h/p, (h+1)/p)$ liegen, in ein $(t, m, s-1)$-Netz $P_3^{(h)}$ in Basis p. Für diese gilt wieder $D(J_h; P) = D(T(J_h); P_3^{(h)})$ und mit der Induktionsannahme damit $|D(J_h; P)| \leq \Delta_p(t, m, s-1)$ für $0 \leq h < l$.

Insgesamt ergibt dies also

$$\begin{aligned} |D(J)| &\leq \sum_{h=0}^{l} |D(J_h)| \leq \Delta_p(t,m,s) + l\Delta_p(t,m,s-1) \\ &\leq \Delta_p(t,m,s) + \left\lfloor \frac{p}{2} \right\rfloor \Delta_p(t,m,s-1) = \Delta_p(t,m+1,s) \end{aligned}$$

wie sich nun leicht durch algebraische Umformungen der Summen in Δ_p zeigen lässt.

Fall 2.2: Als letzter Fall schließlich betrachten wir $\lfloor p/2 \rfloor \leq l < p$. J stellen wir nun dar als Komplement von $M = \prod_{i=1}^{s-1} [0, u_i) \times [u_s, 1)$ in $L = \prod_{i=1}^{s-1} [0, u_i) \times [0, 1)$ und unterteilen M entsprechend der letzten Komponente in disjunkte Intervalle:

$$M_l = \prod_{i=1}^{s-1} [0, u_i) \times \left[u_s, \frac{l+1}{p}\right)$$

$$M_h = \prod_{i=1}^{s-1} [0, u_i) \times \left[\frac{h}{p}, \frac{h+1}{p}\right) \quad \text{für } l < h < p$$

Damit ergibt sich:

$$D(J;P) = D(L;P) - D(M_l;P) - \sum_{h=l+1}^{p-1} D(M_h;P)$$

Der erste Summand wird nach Fall 1, die restlichen gleich wie die Terme für J_h und J_l in Fall 2.1 abgeschätzt:

$$|D(J;P)| \leq \Delta_p(t,m+1,s-1) + \Delta_p(t,m,s) + \left((p-1) - \left(\left\lfloor \frac{p}{2} \right\rfloor + 1\right)\right) \Delta_p(t,m,s-1)$$

was nach einigen algebraischen Umformungen der Summen mit Binomialkoeffizienten ebenfalls wieder die gewünschte Ungleichung liefert:

$$|D(J;P)| \leq \Delta_p(t,m+1,s) \tag{2.74}$$

Durch Bildung des Supremums ergibt sich dann das gewünschte

$$ND_N^*(P) \leq p^t \sum_{i=0}^{s-1} \binom{s-1}{i} \binom{m-t}{i} \left\lfloor \frac{p}{2} \right\rfloor^i$$

□

Satz 2.14. *Die Sterndiskrepanz eines (t,m,s)-Netzes P in gerader Basis p erfüllt*

$$ND_N^*(P) \leq p^t \sum_{i=0}^{s-1} \binom{m-t}{i} \left(\frac{p}{2}\right)^i + \left(\frac{p}{2}-1\right) p^t \sum_{i=0} s-2 \binom{m-t+i+1}{i} \left(\frac{p}{2}\right)^i$$

Satz 2.15. *Die Sterndiskrepanz eines (t,m,s)-Netzes in Basis p erfüllt folgende Beziehungen für spezielle Werte von s:*

$$ND_N^*(P) \leq \left\lfloor \frac{p-1}{2}(m-t) + \frac{3}{2} \right\rfloor p^t \quad \textit{für } s = 2 \tag{2.75a}$$

$$ND_N^*(P) \leq \left\lfloor \left(\frac{p-1}{2}\right)^2 (m-t)^2 + \frac{p-1}{2}(m-t) + \frac{9}{4} \right\rfloor b^t \tag{2.75b}$$

Ausgedrückt in Potenzen von $(m-t)$ hat diese Abschätzung die Form:

$$D_N(P) \leq \frac{1}{(s-1)!} p^t \left\lfloor \frac{p}{2} \right\rfloor^{s-1} (m-t)^{s-1} + O\left((m-t)^{s-2}\right) \tag{2.76}$$

Oder in der Anzahl $N = p^m$ der Punkte ausgedrückt

$$D_N(P) \leq \frac{1}{(s-1)!} p^t \left(\frac{\lfloor p/2 \rfloor}{\ln p}\right)^{s-1} \ln^{s-1} N + O\left(\ln^{s-2} N\right) \tag{2.77}$$

sodass also (t,m,s)-Netze wirklich eine Diskrepanz der gewünschten Form (2.3) besitzen.

2.3.2 Diskrepanz von (t, s)-Folgen

Da die Definition von (t, s)-Folgen auf derjenigen der (t, m, s)-Netze aufbaut, ist es naheliegend, dass sich die Diskrepanz für (t, s)-Folgen auch aus einer Abschätzung $ND_N^*(t, m, s) \leq \Delta_p(t, m, s)$ der Diskrepanz der (t, m, s)-Netze abschätzen lässt. Dafür kann etwa Satz 2.12 benutzt werden.

Lemma 2.16. *Für die Diskrepanz $D_N^*(P)$ der ersten $N \geq p^t$ Elemente einer (t, s)-Folge P gilt:*

$$D_N^*(P) \leq \frac{p-1}{2} \sum_{m=t}^{k} \Delta_p(t, m, s) + \frac{1}{2}\Delta_p(t, k+1, s) + \frac{1}{2}\max(p^t, \Delta_p(t, r, s)) \tag{2.78}$$

wobei k die kleinste natürliche Zahl mit $p^k \leq N < p^{k+1}$ und p^r die größte Potenz von p darstellt, welche N teilt. Für $r < t$ soll $\Delta_p(t, r, s) = 0$ gelten.

Beweis. N hat die Entwicklung $N = \sum_{m=0}^{k} a_m p^m$ in Basis p, wobei $a_m \in Z_p$, $a_k \neq 0$ und $k > t$. Die (t, s)-Folge $\mathbf{r}_1, \mathbf{r}_2, \ldots$ teilen wir nun auf in Teilmengen P_m:

$$P_k = \left\{ \mathbf{r}_n : 0 \leq n < a_k p^k \right\}$$

$$P_m = \left\{ \mathbf{r}_n : \sum_{h=m}^{k} \leq n < \sum_{h=M+1}^{k} \right\} \quad \text{für } 0 \leq m < k$$

Diese Mengen enthalten jeweils $a_m p^m$ Punkte, sind also nicht leer, wenn $a_m \neq 0$, und da wir eine (t, s)-Folge betrachten, können sie entsprechend dem Wert des Koeffizienten von p^m in n aufgeteilt werden in a_m (t, m, s)-Netze in Basis p, wenn $m \geq t$. Für $m < t$ ist p^m eine Schranke für die Diskrepanz von P_m:

$$ND_N^*(t, m, s) \leq \sum_{m=t}^{k} a_m \Delta_p(t, m, s) + \sum_{m=0}^{t-1} a_m p^m$$

In einem zweiten Schritt betrachten wir nun die Punkte $\mathbf{r}_n$ von P mit $N < n \leq p^{k+1}$, also eine Menge von $p^{k+1} - N = \sum_{m=0}^{k+1} c_m p^m$ Punkten. Auf dieselbe Weise wie oben erhalten wir auch hier die entsprechende Abschätzung der Diskrepanz, in der nur a_m durch c_m ersetzt ist und die

erste Summe bis $k+1$ läuft. Die ersten p^{k+1} Punkte von P stellen nach Definition eines (t, s)-Netzes ein $(t, k+1, s)$-Netz dar, womit unter Zuhilfenahme der Dreiecksungleichung die Diskrepanz der ersten N Punkte der Folge auch abgeschätzt werden kann durch:

$$ND_N^*(t, m, s) \leq \Delta_p(t, m+1, s) + \sum_{m=t}^{i+1} c_m \Delta_p(t, m, s) + \sum_{m=0}^{t-1} c_m p^m$$

Für $m < r$ gilt auch $a_r = c_r = 0$ nach Definition von r, weiters ist $a_k + c_k = b$ und damit auch $a_m + c_m = p - 1$ für $r < m \leq k$ (da sich die durch a_m und c_m repräsentierten Zahlen zu p^{k+1} summieren müssen. Durch Summation obiger beider Abschätzungen und Division durch 2 ergibt sich:

$$ND_N^*(P) \leq \frac{1}{2} \sum_{m=t}^{k} (a_m + c_m)\Delta_p(t, m, s) + \frac{1}{2}\Delta_p(t, k+1, s) + \frac{1}{2} \sum_{m=1}^{t-1} (a_m + c_m) p^m$$

Fall 1: $r < t$

$$\begin{aligned} ND_N^*(P) &\le \frac{p-1}{2}\sum_{m=t}^{k}\Delta_p(t,m,s) \\ &\quad + \frac{1}{2}\Delta_p(t,k+1,s) + \frac{1}{2}\left(p^{r+1} + \sum_{m=r+1}^{t-1}(p-1)p^m\right) \\ &= \frac{p-1}{2}\sum_{m=t}^{k}\Delta_p(t,m,s) + \frac{1}{2}\Delta_p(t,k+1,s) + \frac{1}{2}p^t \end{aligned}$$

Fall 2: $r \ge t$

$$ND_N^*(P) \le \frac{p}{2}\Delta_p(t,r,s) + \frac{p-1}{2}\sum_{m=r+1}^{k}\Delta_p(t,m,s) + \frac{1}{2}\Delta_p(t,k+1,s)$$

womit sich auch das Maximum in der Abschätzung ergibt.

□

Mit dieser Formel können nun ausgehend von den Diskrepanzschranken des vorigen Abschnittes auch Abschätzungen für (t,s)-Netze angegeben werden.

Satz 2.17. *Die Sterndiskrepanz $D_N^*(P)$ der ersten $N \ge p^t$ Elemente einer (t,s)-Folge in Basis $p \ge 3$ erfüllt*

$$\begin{aligned} ND_N^*(P) \le \frac{p-1}{2}b^t\sum_{i=1}^{s}\binom{s-1}{i-1}\binom{k+1-t}{i}\left\lfloor\frac{p}{2}\right\rfloor^{i-1} \\ + \frac{1}{2}p^t\sum_{i=0}^{s-1}\binom{s-1}{i}\left(\binom{k+1-t}{i}+\binom{k-t}{i}\right)\left\lfloor\frac{p}{2}\right\rfloor^{i} \end{aligned} \tag{2.79}$$

Die Sterndiskrepanz $D_N^*(P)$ der ersten $N \ge p^t$ Elemente einer (t,s)-Folge in gerader Basis p erfüllt

$$\begin{aligned} ND_N^*(P) \le (p-1)p^{t-1}\sum_{i=1}^{s}\binom{k+1-t}{i}\left(\frac{p}{2}\right)^i \\ + \binom{p-1}{2}p^{t-1}\sum_{i=1}^{s-1}\binom{k+i+1-t}{i}\left(\frac{p}{2}\right)^i \\ + \frac{1}{2}p^t\sum_{i=0}^{s-1}\left(\binom{k+1-t}{i}+\binom{k-t}{i}\right)\left(\frac{p}{2}\right)^i \\ + \frac{p-2}{4}p^t\sum_{i=0}^{s-2}\left(\binom{k+i+2-t}{i}+\binom{k+i+1-t}{i}\right)\left(\frac{p}{2}\right)^i \end{aligned} \tag{2.80}$$

wobei k jeweils die größte natürliche Zahl mit $p^k \le N$ darstellt.

Beweis. Durch Verwendung der Abschätzungen (2.73) und (2.14) im vorigem Satz. □

Asymptotisch ergeben diese (und einige weiter in [Nie92] und [Nie87] angegebene) Schranken kombiniert folgende Diskrepanzschranke:

Satz 2.18. *Die Sterndiskrepanz $D_N^*(P)$ der ersten N Elemente einer (t,s)-Folge P in Basis p erfüllt:*

$$ND_N^*(P) \leq C(s,p)p^t \ln^s N + O\left(p^t \ln^{s-1} N\right) \quad \text{für } N \geq 2 \tag{2.81}$$

mit

$$C(s,p) = \begin{cases} \frac{1}{s}\left(\frac{p-1}{2\ln p}\right)^s & \text{für } s = 2 \text{ oder } p = 2, s = 3,4 \\ \frac{1}{s!}\frac{p-1}{s\lfloor p/2 \rfloor}\left(\frac{\lfloor p/2 \rfloor}{\ln p}\right)^s & \text{sonst} \end{cases}$$

2.4 Existenz von (t,s)-Folgen

Nach all diesen theoretischen Abschätzungen der Diskrepanz stellt sich natürlich die Frage, ob derartige (t,s)-Folgen überhaupt existieren, bzw. falls ja, für welche Werte von t und s.

Da niedrige t höhere Regelmäßigkeit bedeuten, interessiert

natürlich vor allem die Existenz von $(0,m,s)$-Netzen. Bereits NIEDERREITER zeigte aufbauend auf Arbeiten von SOBOL und FAURE mit Hilfe von orthogonalen latin squares folgenden Satz mit dem zugehörigen relevanten Korollar:

Satz 2.19. *Für $m \geq 2$ kann ein $(0,m,s)$-Netz in Basis p nur existieren, wenn $s \leq M(p) + 2$, wobei $M(p)$ die maximale Anzahl von paarweise orthogonalen latin squares in Basis p bezeichnet.*

Korollar 2.20. *Da nach [Nie92] $M(p) \leq p-1$ gilt, kann für $m \geq 2$ ein $(0,m,s)$-Netz in Basis p nur existieren, wenn $s \leq p+1$.*

Ebenso wie die Hammersley-Menge durch Hinzufügen einer Dimension in der Form $\frac{n}{N}$ zur Halton-Folge entsteht, kann aus einer (t,s)-Folge leicht für beliebiges $m \geq t$ ein $(t,m,s+1)$-Netz erzeugt werden:

Satz 2.21. *Wenn eine (t,s)-Folge in Basis p existiert, so existiert für jedes $m \geq t$ ein $(t,m,s+1)$-Netz in Basis p.*

Beweis. Für ein festes $m \geq t$ sei $\mathbf{y}_n = \left(\frac{n-1}{p^m}, x_n^{(1)}, \ldots x_n^{(s)}\right)$ für $1 \leq n \leq p^m$. Diese Menge von p^m Punkten stellt ein $(t,m,s+1)$-Netz dar:

Es sei $E = \prod_{i=1}^{s+1}\left[a^{(i)}p^{-d_i}, (a^{(i)}+1)p^{-d_i}\right)$ ein $(s+1)$-dimensionales elementares Intervall in Basis p mit $\lambda_{s+1}(E) = p^{t-m}$, und daher gilt $\sum_{i=1}^{s+1} d_i = m - t$. Da die $\mathbf{r}_n$ eine (t,s)-Folge in Basis p darstellen, sind die Punkte $\mathbf{r}_n$ mit $y_n^{(1)} \in E^{(1)}$ ein $(t, m-d_1, s)$-Netz in Basis p. Daher enthält das Intervall $E' = \prod_{i=1}^{s+1} E^{(i)}$ genau p^t Punkte der Folge, weil dieses Intervall ein s-dimensionales elementares Intervall in Basis p mit $\lambda_s(E') = p^{t-m+d_1}$ darstellt. Folglich enthält E genau p^t Punkte $\mathbf{y}_n$. □

Eine analoge Aussage zur Existenz von $(0,m,s)$-Netze gilt auch für $(0,s)$-Folgen:

Satz 2.22. *Eine $(0,s)$-Folge in Basis p kann nur für $s \leq p$ existieren.*

Beweis. Folgt gemäß der Definition von (s,t)-Folgen mit den letzten beiden Sätzen. □

2.5 Erzeugung von (t,m,s)-Netzen und (t,s)-Folgen

Zuerst möchte ich ein Prinzip zur Erzeugung von (t,m,s)-Netzen in einer Basis p beschreiben, und danach erst die eigentlich relevante Erzeugung von (t,s)-Folgen in Basis p, die natürlich entsprechend ihrer Definition auf dem Prinzip der Netze beruht.

2.5.1 Ein allgemeines Prinzip für (t, m, s)-Netze

Es seien die natürlichen Zahlen $m \geq 1$, die Dimension $s \geq 1$ und die Basis $p \geq 2$ gegeben. Zur Erzeugung eines (t, m, s)-Netzes benötigt man nun

- Einen kommutativen Ring R mit Einheit und $\mathrm{card}(R) = p$
- Bijektionen $\psi_r : Z_p \to R$ für $0 \leq r \leq m-1$
- Bijektionen $\eta_{ij} : R \to Z_p$ für $1 \leq i \leq s$ und $1 \leq j \leq m$
- Elemente (auch Richtungszahlen genannt) $c_{jr}^{(i)} \in R$ für $1 \leq i \leq s$, $1 \leq j \leq m$ und $0 \leq r \leq m-1$

Für $0 \leq n \leq p^m - 1$ seien $a_r(n)$ wieder die Koeffizienten in der Ziffernentwicklung von n in Basis p. Mit den Koeffizienten

$$y_{nj}^{(i)} = \eta_{ij}\left(\sum_{r=0}^{m-1} c_{rj}^{(i)} \psi_r\left(a_r(n)\right)\right) \in Z_p \quad \text{für } 0 \leq n < p^m,\ 1 \leq i \leq s,\ 1 \leq j \leq m \tag{2.82}$$

definiert man nun die einzelnen Koordinaten

$$x_n^{(i)} = \sum_{j=1}^{m} y_{nj}^{(i)} p^{-j} \quad \text{für } 0 \leq n < p^m \text{ und } 1 \leq i \leq s \tag{2.83}$$

und damit $\mathbf{r}_n = \left(x_n^{(1)}, \ldots x_n^{(s)}\right) \in \bar{I}^s$ für $0 \leq n \leq p^m - 1$.

Bemerkung 2.6. In einer Dimension erhält man mit den speziellen Werten von $\psi_r = \mathrm{id}$, $\eta_{ij} = \mathrm{id}$ und $c_{jr}^{(1)} = \delta_{j-1,r}$ genau die Van der Corput-Folge in Basis p.

Die derart definierte Punktmenge ist allerdings noch nicht notwendigerweise ein (t, m, s)-Netz. Folgender Satz gibt aber eine hinreichende Bedingung dafür:

Satz 2.23. *Wenn die Zahl t, $0 \leq t \leq m$ derartig gewählt ist, dass für alle Zahlen $d_1, \ldots d_s \geq 0$mit $\sum_{i=1}^{s} d_i = m - t$ und alle $f_j^{(i)} \in R$, $1 \leq j \leq d_i$, $1 \leq i \leq s$ das Gleichungssystem*

$$\sum_{r=0}^{m-1} c_{jr}^{(i)} z_r = f_j^{(i)} \quad \textit{für } 1 \leq j \leq d_i,\ 1 \leq i \leq s$$

von $m - t$ Gleichungen in $\mathbf{z}$ über R genau p^t Lösungen hat, dann ist die Menge der $\mathbf{r}_n$ ein (t, m, s)-Netz in Basis p.

Beweisskizze. Für ein elementares Intervall $E = \prod_{i=1}^{s} \left[a_i p^{t_i}, (a_i + 1)p^{t_i}\right)$ in Basis p mit dem Maß $\lambda_s(E) = p^{t-m}$ gilt offensichtlich $\sum_{i=1}^{s} d_i = m - t$. Betrachtet man nun die Ziffernentwicklung der Zahlen $a_i = \sum_{j=1}^{d_i} a_{ij} p^{d_i - j}$ in Basis p mit $a_{ij} \in Z_p$ betrachtet, so ist klar, dass $\mathbf{r}_n \in E$ genau dann, wenn $y_{nj}^{(i)} = a_{ij}$ für $1 \leq j \leq d_i$ und $1 \leq i \leq s$. Dies ist jedoch äquivalent zu

$$\sum_{r=0}^{m-1} c_{jr}^{(i)} \psi_r(a_r(n)) = \eta_{-1}(a_{ij}) \quad \text{für } 1 \leq j \leq d_i,\ 1 \leq i \leq s$$

Dieses Gleichungssystem hat nach Voraussetzung genau p^t Lösungen für ψ_r, wobei jede dieser Lösungen dann genau einem Punkt $\mathbf{r}_k$ entspricht. Damit ist also $A(E; P) = p^t$. □

2.5.2 Ein allgemeines Prinzip für (t, s)-Folgen

Ein allgemeines Prinzip zur Erzeugung von (t, s)-Folgen in Basis p ähnelt sehr stark dem oben angegebenen für Netze, LÉCOT dass jeweils die Beschränkung nach oben durch m wegfällt:

Es seien die Dimension $s \geq 1$ und die Basis $p \geq 2$ gegeben. Zur Erzeugung eines (t, m, s)-Netzes benötigt man nun

- Einen kommutativen Ring R mit Einheit und $\text{card}(R) = p$
- Bijektionen $\psi_r : Z_p \to R$ für $0 \leq r$
- Bijektionen $\eta_{ij} : R \to Z_p$ für $1 \leq i \leq s$ und $1 \leq j$
- Elemente (auch Richtungszahlen genannt) $c_{jr}^{(i)} \in R$ für $1 \leq i \leq s$, $1 \leq j$ und $0 \leq r$

Für $0 \leq n$ seien $a_r(n)$ wieder die Koeffizienten in der Ziffernentwicklung von n in Basis p. Mit den Koeffizienten

$$y_{nj}^{(i)} = \eta_{ij}\left(\sum_{r=0}^{\infty} c_{rj}^{(i)} \psi_r\left(a_r(n)\right)\right) \in Z_p \quad \text{für } 0 \leq n,\ 1 \leq i \leq s,\ 1 \leq j \tag{2.84}$$

definiert man nun die einzelnen Koordinaten

$$x_n^{(i)} = \sum_{j=1}^{\infty} y_{nj}^{(i)} p^{-j} \quad \text{für } 0 \leq n \text{ und } 1 \leq i \leq s \tag{2.85}$$

und damit $\mathbf{r}_n = \left(x_n^{(1)}, \ldots x_n^{(s)}\right) \in \bar{I}^s$ für $0 \leq n$.

Um zu gewährleisten, dass $\mathbf{r}_n \in I^s$ und nicht nur in $\bar{I}^s$ liegt, ist weiters folgende Bedingung nötig:

- Für jedes $n \geq 0$ und $1 \leq i \leq s$ gilt $y_{nj}^{(i)} < p - 1$ für unendlich viele j.

Hinreichend für letztere Bedingung ist bereits, dass $\eta_{ij}(0) = 0$ für $1 \leq i \leq s$ und hinreichend große j, sowie $c_{jr}^{(i)}$ für $1 \leq i \leq s$, $r \geq 0$ und genügend große j. Mit dieser stärkeren Bedingung sind alle auftretenden Summen auch endlich.

Satz 2.23 gilt in ähnlicher Form auch für (t, s)-Folgen:

Satz 2.24. *Die natürliche Zahl $t \geq 0$ erfülle folgende Bedingung: Für alle $m > t$ und $d_1, \ldots d_s \geq 0$ mit $\sum_{i=1}^{s} d_i = m - t$ und alle $f_j \in r$, $1 \leq j \leq d_i$, $1 \leq i \leq s$ besitzt das lineare Gleichungssystem:*

$$\sum_{r=0}^{m-1} c_{jr}^{(i)} z_r = f_j^{(i)} \quad \textit{für } 1 \leq j \leq d_i \textit{ und } 1 \leq i \leq s$$

über R genau p^t Lösungen. Dann ist die Folge der $\mathbf{r}_n$ eine (t, s)-Folge in Basis p.

Beweis. Der Beweis läuft analog zum Beweis von Satz 2.23 durch Betrachtung der Mengen der Punkten $\mathbf{r}_n$ mit $kp^m \leq n < (k+1)p^m$. □

Für Primpotenzen $p = q^l$ lässt sich eine einfachere Aussage machen. Als Ring wird dabei ein endlicher Körper $R = F_p$ mit p Elementen benutzt.

Definition 2.10. Für ein System $C = \left\{\mathbf{c}_j^{(i)} \in V : 1 \le i \le s, 1 \le j \le m\right\}$ von Vektoren eines endlich dimensionalen Vektorraumes V sei $\sigma(C)$ die größte ganze Zahl d, sodass jedes System $\left\{\mathbf{c}_j^{(i)} : 1 \le j \le d_i, 1 \le i \le s\right\}$ mit $0 \le d_i \le m$ für $1 \le i \le s$ und $\sum_{i=1}^{s} d_i = d$ linear unabhängig in V ist.

Lemma 2.25 (NIEDERREITER [Nie87], o.B.). *$p = q^l$ sei eine Primzahlpotenz, F_p der benutzte endliche Körper mit p Elementen und $t \ge 0$. Wenn für alle $m > t$ das System $C^{(m)}$, das die Vektoren*

$$\mathbf{c}_j^{(i)} = \left(c_{j0}^{(i)}, \ldots c_{j,m-1}^{(i)}\right) \in F_p^m \quad \textit{für } 1 \le i \le s,\ 1 \le j \le m$$

enthält, die Bedingung $\sigma(C^{(m)}) \ge m - t$ erfüllt, dann stellt die oben definierte Folge der $\mathbf{r}_n$ eine (t, s)-Folge in Basis $p = q^l$ dar.

2.5.3 Spezielle Konstruktionen von (t, s)-Folgen

Aufgrund der oben gegebenen allgemeinen Konstruktionsprinzipien können nun spezielle Konstruktionen durch geeignete Wahl der Elemente $c_{jr}^{(i)} \in R$ angegeben werden.

Die Bijektionen ψ_r und η_{ij} können beliebig gewählt werden, wobei es am einfachsten ist, dafür die Identität zu benutzen.

Eine spezielle Konstruktion von NIEDERREITER [Nie87] basiert auf den formalen Laurent-Reihen. Als Ring wird dabei der Körper $F_p(x^{-1})$ der formalen Laurent-Reihen über F_p in der Variable x^{-1} benutzt. Die Elemente von $F_p(x^{-1})$ haben also die Form

$$L = \sum_{k=w}^{\infty} t_k x^{-k}$$

wobei w eine beliebige ganze Zahl ist und alle $t_k \in F_p$. Für eine Laurent-Reihe ist $\nu(L)$ definiert als der kleinste Index k mit $t_k \ne 0$ für $L \ne 0$, und durch $\nu(0) = -\infty$.

Zur Konstruktion wird eine Basis $p = q^l$ betrachtet, die eine Primzahlpotenz darstellt. Für eine Dimension $s \ge 1$ seien weiters $q_1, \ldots q_s \in F_p[x]$ verschiedene monische (führender Koeffizient beträgt 1) irreduzible Polynome über F_p mit $e_i = \deg(q_i)$. Man betrachtet die Entwicklungen

$$\frac{x^k}{q_i(x)^j} = \sum_{r=0}^{\infty} a^{(i)}(j,k,r) x^{-r-1} \in F_p(x^{-1}) \quad \text{für } 1 \le i \le s \tag{2.86}$$

und konstruiert damit mit natürlichen Zahlen $Q = Q(i,j)$ und $k = k(i,j)$, $0 \le k < e_i$ die Richtungszahlen

$$c_{jr}^{(i)} = a^{(i)}(Q+1, k, r) \in F_p \quad \text{für } 1 \le i \le s,\ j \ge 1, r \ge 0 \tag{2.87}$$

mit der zusätzlichen Bedingung $j - 1 = Qe_i + k$. Die Bijektionen η_{ij} sind so gewählt, dass $\eta_{ij}(0) = 0$ für $1 \le i \le s$ und genügend große j. Damit sind die Richtungszahlen $c_{jr}^{(i)}$ für hinreichend große j gleich 0, und die letzte Bedingung, welche gewährleistet, dass die $\mathbf{r}_n$ nicht nur in $\bar{I}^s$, sondern sogar in I^s liegen, für die (t, s)-Folgen ist damit erfüllt.

Für die so erhaltene (t, s)-Folge lässt sich nun auch ein Wert für t angeben:

Satz 2.26 (NIEDERREITER, [Nie92], o.B.). *Die Konstruktion mit den in (2.87) angegebenen Richtungszahlen $c_{jr}^{(i)}$ liefert ein (t, s)-Netz mit $t = \sum_{i=1}^{s}(e_i - 1)$*

Anmerkung zum Beweis. Der Beweis dieses Satzes baut einfach auf Satz 2.25 auf und benutzt weiters die Eindeutigkeit der Partialbruchzerlegung. □

Minimales t wird damit also erreicht, indem die ersten s irreduziblen Polynome über F_p benutzt werden.

Eine zweite Konstruktion von $(0,s)$-Folgen in einer Basis p, die eine Primzahlpotenz q^l darstellt, basiert auf Hyperableitungen und wurde ebenfalls von NIEDERREITER [Nie88b] vorgeschlagen. Für eine natürliche Zahl $k \geq 0$ ist die k-te Hyperableitung der F_p-lineare Operator $H^{(k)}$ auf dem Polynomring $F_p[x]$, definiert durch $H^{(k)}(x^r) = \binom{r}{k} x^{r-k}$ für $r \geq k$ und $H^{(k)}(x^r) = 0$ für $0 \leq r < k$.

Satz 2.27 (NIEDERREITER [Nie88b], o.B.). *Für $s \leq p$ seien $b_1, \ldots b_s$ verschiedene Elemente aus F_p. Für $1 \leq i \leq s$ sei $g_i \in F_p[x]$ mit $g_i(b_i) \neq 0$. Es seien dann die Richtungszahlen definiert durch*

$$c_{jr}^{(i)} = [H^{(j-1)}(x^r g_i(s))](b_i) \quad \textit{für } 1 \leq i \leq s,\ j \geq 1 \textit{ und } r \geq 0$$

Wenn weiters die Bijektionen η_{ij} so gewählt sind, dass $\eta_{ij}(0) = 0$ für $1 \leq i \leq s$ und genügend große j, dann stellen die $\mathbf{r}_n$ eine $(0,s)$-Folge in der Basis $p = q^l$ dar.

Letztere Konstruktion hat unter anderem den Vorteil, dass bei spezieller Wahl von $g_i(x) = 1$ - wie LÉCOT in [Léc91] beschreibt - die $y_{nj}^{(i)}$ rekursiv berechnet werden können. Die $c_{jr}^{(i)}$ eingesetzt in Gleichung (2.84) ergeben

$$y_{n,(j+1)}^{(i)} = \sum_{k=j}^{\infty} \binom{k}{j} b_i^{k-j} a_k(n) \in F_p \quad \text{für } 1 \leq i \leq s, j \geq 0$$

Dabei stellen die $a_k(n)$ die Koeffizienten der Ziffernentwicklung von n in Basis p dar:

$$n = \sum_{k=0}^{\infty} a_k(n) p^k \quad \text{mit } 0 \leq a_k(n) < p \tag{2.88}$$

Es sei nun $\lambda \in \mathbb{N}$ fix gewählt und $\nu = p^\lambda$. Die (t,s)-Folge $\mathbf{Y}$ wird nun in Punktmengen $\mathbf{Y}^{(m)} = \{\mathbf{y}_n : m\nu \leq n < (m+1)\nu\}$ zerlegt, die eine nach der anderen rekursiv erzeugt werden können.

Für $m = 0$ sei $1 \leq q < p$ und $1 \leq l < \lambda$. Für n mit $qp^l \leq n < (q+1)p^l$ stimmen die Ziffern in Basis p von n und $n - qp^l$ bis auf die vorderste überein: $a_i(n) = a_i(n - qp^l)$ für $0 \leq i < l$ und $a_l(n) = q$, $a_l(n - qp^l) = 0$. Daher können die Ziffern von $y_n^{(i)}$ in Basis p aber dargestellt werden durch:

$$\begin{aligned}
y_{n,(j+1)}^{(i)} &= \sum_{k=j}^{\infty} \binom{k}{j} b_i^{k-j} a_k(n) = \sum_{k=j}^{l} \binom{k}{j} b_i^{k-j} a_k(n) = \\
&= \sum_{k=j}^{l-1} \binom{k}{j} b_i^{k-j} a_k(n - qp^l) + \binom{l}{j} b_i^{l-j} a_l(n) = \\
&= \sum_{k=j}^{\infty} \binom{k}{j} b_i^{k-j} a_k(n - qp^l) + \binom{l}{j} b_i^{l-j} q = \\
&= y_{n-qp^l,j+1}^{(i)} + \binom{l}{j} b_i^{l-j} q \quad \text{für } 0 \leq j \leq l
\end{aligned} \tag{2.89}$$

Für $j > l+1$ ist $y_{nj}^{(i)}$ ohnehin gleich 0.

Damit erhält man nun eine rekursive Folge von Mengen mit steigender Mächtigkeit p^l, $1 \leq l < \lambda$:

$$\left(y_{p^l,j}^{(i)}, \ldots y_{p^{l+1}-1,j}^{(i)}\right) \text{ mit } 1 \leq i \leq s,\ 1 \leq j \leq l \text{ für } 1 \leq l+1 < \lambda \tag{2.90}$$

Damit sind alle Koeffizienten für $\mathbf{Y}^{(0)}$ gegeben. Diese können aber auch noch weiter verwendet werden, um $\mathbf{Y}^{(m)}$ rekursiv zu berechnen:

Wir betrachten wieder ein n mit $m\nu < n \leq (m+1)\nu$. Es sei n' dergestalt, dass $p^{n'} < m \leq p^{n'+1}$. Damit gilt nun wieder für die Ziffern der Entwicklung von n und $n - m\nu$ in Basis p:

$$\begin{aligned} a_i(n) &= a_i(n - m\nu) && \text{für } 0 \leq i < \lambda \\ a_i(n - m\nu) &= 0 && \text{für } i \geq \lambda \\ a_i(n) &= a_{i-\lambda}(m) && \text{für } \lambda \leq i < n' + \lambda \end{aligned}$$

Damit gilt also wieder folgende Rekursionsbeziehung (eine leere Summe soll dabei den üblichen Wert von 0 besitzen):

$$\begin{aligned} y_{n,(j+1)}^{(i)} &= \sum_{k=j}^{n'+\lambda} \binom{k}{j} b_i^{k-j} a_k(n) = \\ &= \sum_{k=j}^{\lambda-1} \binom{k}{j} b_k^{k-j} a_k(n - m\nu) + \sum_{\max(\lambda,j)}^{n'+\lambda} \binom{k}{j} b_i^{k-j} a_{k-\lambda}(m) = \\ &= y_{n-m\nu,j+1}^{(i)} + \sum_{\max(\lambda,j)}^{n'+\lambda} \binom{k}{j} b_i^{k-j} a_{k-\lambda}(m) \end{aligned} \tag{2.91}$$

Damit kann die Berechnung der $\mathbf{Y}^{(m)}$ erfolgen, indem die $y_{n,j}^{(i)}$ für $1 \leq i \leq s$, $1 \leq n \leq \nu$, $1 \leq j \leq \lambda$ gespeichert werden und Gleichung (2.91) zur Berechnung für die Indices mit $n+m\nu$ benutzt wird. Aus diesen werden schließlich mit Gleichung (2.85) die Elemente der (t,s)-Folge erstellt.

Der anfangs gewählte Parameter $\lambda \in \mathbb{N}$ legt bei diesem Algorithmus das Verhältnis zwischen Speicheraufwand (die $y_{n,j}^{(i)}$ für $1 \leq j \leq \lambda$, $1 \leq n \leq \nu = b^\lambda$ müssen zwischengespeichert werden) und Rechenaufwand (die Summe in (2.91), die bei der Berechnung jedes Koeffizienten benutzt wird, beginnt bei λ) fest.

Im Anhang A ist eine Klasse zur Erzeugung von (t,s)-Folgen mittels des hier angegebenen Algorithmus angegeben. Dieser wurde auch für alle numerischen Experimente benutzt.

2.5.4 Faure-Folgen

Die zuletzt angegebene Konstruktion ist eigentlich eine direkte Verallgemeinerung der sogenannten Faure-Folgen auf allgemeine Basen, die 1982 von Henri FAURE [Fau82] angegeben wurden als eine Verallgemeinerung von SOBOL's binären Folgen (also in Basis 2) aus dem Jahr 1967. Als eine Verallgemeinerung dieser Folgen gab NIEDERREITER schließlich die Definition der (t,s)-Folgen. Die Faure-Folgen sind in der Sprache von (t,s)-Folgen ausgedrückt $(0,s)$-Folgen in Basis $p \geq s$, wobei p die kleinste Primzahl darstellt, die diese Bedingung erfüllt.

Definition 2.11 (Faure-Folge). p sei eine Primzahl mit $p \geq s \geq 2$. Die Faure-Folge in Basis q ist definiert als

$$S_p^f(n) = \left(\Phi_p(n), C^1\Phi_p(n), \ldots C^{p-1}\Phi_p(n)\right) \tag{2.92}$$

wobei Φ_p die Radikalinversfunktion darstellt und $C^f\Phi_p(n) = \sum_{j=0}^{\infty} b_j^{(k)}(n)p^{-j-1}$ mit $b_j^{(k)}(n) = \sum_{r\geq j}\binom{r}{j} f^{r-j}a_r(n) \,(\mathrm{mod} p)$ mit der Ziffernentwicklung $a_r(n)$ von n in Basis p.

Diese Folge ist ebenfalls eine Folge kleiner Diskrepanz, zudem konnte FAURE [Fau98] auch für $s = 2$zeigen, dass die Diskrepanz dieser Folge exakt von Ordnung $O\left(\frac{\ln^2 N}{N}\right)$ ist.

Kapitel 3

Die Boltzmann-Gleichung

Inhaltsangabe

3.1 Teilchenkollisionen 39
3.2 Physikalische "Herleitung" 40
3.3 Liouville-Gleichung und BBGKY-Hierarchie 42
3.4 Einschränkungen und Verallgemeinerungen 44
3.5 Existenz und Eindeutigkeit der Lösung 44
3.6 Lösungen der Boltzmann-Gleichung 47
3.7 Schwache Variante der Boltzmann-Gleichung 47
3.8 Master-Gleichung von Kac 49

Als die Transportgleichung schlechthin möchte ich im Folgenden die Boltzmann-Gleichung behandeln. Sie beschreibt das Verhalten verdünnter monoatomarer Gase, wobei allerdings nicht makroskopische Parameter betrachtet werden, sondern bei der Herleitung von N wechselwirkenden Teilchen (durch Dirac-Maße modelliert) ausgegangen wird. N hat dabei eine Größenordnung von etwa 10^{23}, was zur Simulation allerdings meist zu etwa 10^6 reduziert wird.

In den ersten paar eher nicht-mathematisch gehaltenen Abschnitten dieses Kapitels möchte ich kurz den physikalischen Hintergrund der Boltzmann-Gleichung darlegen. Dabei werde ich anstatt mathematische Rigorosität anzuwenden eher die Plausibilität des Lesers ansprechen.

Die Dichte $f(\mathbf{r}, \mathbf{v}, t)$ des betrachteten Gases sei von Ort $\mathbf{r}$, Geschwindigkeit $\mathbf{v}$ und der Zeit t abhängig. Unsere Aufgabe besteht nun darin, eine Gleichung - die Boltzmann-Gleichung - herzuleiten, die $f(\mathbf{r}, \mathbf{v}, t) : \mathbb{R}^3 \times \mathbb{R}^3 \times [0, T) \to \mathbb{R}_0^+, T \in \mathbb{R}_+$ bestimmt:

$$\frac{\partial f}{\partial t}(\mathbf{r}, \mathbf{v}, t) + \mathbf{v} \cdot \nabla_{\mathbf{r}} f(\mathbf{r}, \mathbf{v}, t) = J(f, f) \tag{3.1}$$

mit dem bilinearen Operator (Kollisionsterm) $J : \tilde{C} \times \tilde{C} \to \tilde{D}$

$$J : (f, g) \mapsto \int_{\mathbb{R}^3} \int_{S^2} \sigma(\vartheta, |\mathbf{v} - \mathbf{w}|) \left(f\left(\mathbf{r}, \mathbf{v}', t\right) f\left(\mathbf{r}, \mathbf{w}', t\right) - f(\mathbf{r}, \mathbf{v}, t) f(\mathbf{r}, \mathbf{w}, t) \right) d\Omega d\mathbf{w} \tag{3.2}$$

wobei $\tilde{C}$ und $\tilde{D}$ Funktionenräume sind, S^2 die 2-dimensionale Sphäre und ϑ den Winkel zwischen $\mathbf{v}$ und $\mathbf{v}'$ darstellt. Der Term $Df = v \cdot \nabla_r f$ wird auch Divergenzterm genannt.

Für das Hard Sphere Modell hat $J(f,f)$ die folgende Form mit einer Konstanten C:

$$J(f,f) = C \int_{\mathbb{R}^3} \int_{S^2_+} \mathbf{n}\cdot(\mathbf{v}-\mathbf{w}) \left(f\left(\mathbf{r},\mathbf{v}',t\right) f\left(\mathbf{r},\mathbf{w}',t\right) - f\left(\mathbf{r},\mathbf{v},t\right) f\left(\mathbf{r},\mathbf{w},t\right)\right) d\Omega d\mathbf{n}$$

Dabei stellt $S^2_+ = S^2_{\mathbf{v},\mathbf{w}} = \{\mathbf{n} \in S^s : \mathbf{n}\cdot(\mathbf{v}-\mathbf{w} > 0\}$ lediglich die halbe Sphäre dar (nur dieser Winkelbereich ist physikalisch für die Streuung zulässig), sowie $\mathbf{v}' = \mathbf{v} - \mathbf{n}\cdot(\mathbf{v}-\mathbf{w})\mathbf{n}$ und $\mathbf{w}' = \mathbf{w} + \mathbf{n}\cdot(\mathbf{v}-\mathbf{w})\mathbf{n}$.

Wir werden im Folgenden immer von einem Anfangswert $f_0(\mathbf{r},\mathbf{v}) = f(\mathbf{r},\mathbf{v},0)$ ausgehen. Zudem werde ich die Abkürzungen

$$\begin{aligned} f &:= f\left(\mathbf{r},\mathbf{v},t\right) & f' &:= f\left(\mathbf{r},\mathbf{v}',t\right) \\ f_1 &:= f(\mathbf{r},\mathbf{w},t) & f_1' &:= f(\mathbf{r},\mathbf{w}',t) \\ \mathbf{g} &= \mathbf{v}-\mathbf{w} & f &= |\mathbf{v}-\mathbf{w}| \end{aligned}$$

benutzen.

Um zur Boltzmann-Gleichung zu gelangen, gibt es verschiedene Wege, von denen ich zwei hier demonstrieren möchte:

- Zum einen eine "Herleitung", die auf physikalischen Argumenten beruht, wobei natürlich einige Annahmen gemacht werden müssen. Auf diese Weise leitete BOLTZMANN ursprünglich die Gleichung her.
- Zum anderen kann man von der Liouville-Gleichung ausgehen, welche die N-Teilchendichte als Funktion der Variablen (r_1,...,r_N, v_1,...,v_N) beschreibt. Durch Integration gelangt man dann zu einem System von geschachtelten Gleichungen für die normalisierten i-Teilchen Wahrscheinlichkeitsdichten F_i, der sogenannten BBGKY-Hierarchie. In einem letzten Schritt ist es dann möglich, aus der BBGKY-Hierarchie zu einer geschlossenen Gleichung für F_1 und damit auch zur Boltzmann-Gleichung zu gelangen.

Wie allerdings schon BOLTZMANN zeigte, ist die Gültigkeit der Gleichung für F_1 eingeschränkt durch folgende Forderungen [Har71]:

1. Die Dichte des Gases ist derartig gering, dass keine mehrfachen Kollisionen auftreten.
2. Die räumliche Abhängigkeit der Gaseigenschaften ist so langsam, dass die Kollisionen als lokal im physikalischen Raum angenommen werden können.
3. Das Potential der Teilchen aufeinander ist von genügend kleiner Reichweite.

3.1 Teilchenkollisionen

Die Kollisionen der Teilchen erfolgen vollkommen elastisch und unabhängig vom spezifischen Potential zwischen ihnen unter den Bedingungen der Impuls- und Energieerhaltung:

$$\mathbf{v} + \mathbf{w} = \mathbf{v}' + \mathbf{w}' \tag{3.3}$$

$$|\mathbf{v}|^2 + |\mathbf{w}|^2 = \left|\mathbf{v}'\right|^2 + \left|\mathbf{w}'\right|^2 \tag{3.4}$$

Dies sind 4 Bedingungen für 6 Unbekannte, sodass $\mathbf{v}'$ und $\mathbf{w}'$ durch zwei Parameter dargestellt werden können. Am einfachsten wird dafür die Richtung α zwischen $\mathbf{v}$ und $\mathbf{v}'$ benutzt ([Gra58]): $(\mathbf{v}' - \mathbf{v}) = k\alpha$, $|\alpha| = 1$, wobei k später noch bestimmt wird.

Damit ergibt die Energieerhaltung $|\mathbf{v}|^2 + |\mathbf{w}|^2 = |\mathbf{v} + k\alpha|^2 + |\mathbf{w} - k\alpha|^2$ und somit $0 = k\alpha\mathbf{v} - 2k\alpha\mathbf{w} + 2|\alpha k|^2 = 2k(\alpha(\mathbf{v} - \mathbf{w}) + k)$. Dies ergibt somit

$$\mathbf{v}' = \mathbf{v} + k\alpha = \mathbf{v} + \alpha(\alpha(\mathbf{v} - \mathbf{w})) \tag{3.5}$$

$$\mathbf{w}' = \mathbf{w} - \alpha(\alpha(\mathbf{v} - \mathbf{w})) \tag{3.6}$$

für die Geschwindigkeiten der Teilchen nach der Kollision. Da jedoch α und $-\alpha$ zu exakt demselben Ergebnis führen, muss man sich beschränken auf $\alpha(\mathbf{v} - \mathbf{w}) \geq 0$.

Für die inverse Transformation erhält man exakt dieselbe Form

$$\mathbf{v} = \mathbf{v}' + k\alpha = \mathbf{v}' + \alpha\left(\alpha\left(\mathbf{v}' - \mathbf{w}'\right)\right) \tag{3.7}$$

$$\mathbf{w} = \mathbf{w}' - \alpha\left(\alpha\left(\mathbf{v}' - \mathbf{w}'\right)\right) \tag{3.8}$$

allerdings mit entgegengesetztem α, um $\alpha(\mathbf{v}' - \mathbf{w}') \geq 0$ zu erfüllen.

Die Jacobi-Determinante der linearen Transformation $(\mathbf{v}, \mathbf{w}) \mapsto (\mathbf{v}' - \mathbf{w}')$ beträgt außerdem

$$\frac{\partial(\mathbf{v}', \mathbf{w}')}{\partial(\mathbf{v}, \mathbf{w})} = 1 \tag{3.9}$$

3.2 Physikalische "Herleitung"

BOLTZMANN leitete die nach ihm benannte Gleichung rein aufgrund von physikalischen Argumenten her, die ich [Bir94] folgend hier wiedergeben möchte. Da dies, wie ZWEIFEL[Zwe84] bereits feststellt, eher ein Plausibilitätsargument ist, sei in diesem Abschnitt die Existenz aller betrachteten Integrale vorausgesetzt, ebenso wie die Gültigkeit aller betrachteten Umformungen.

Wie schon oben erwähnt, werden im Laufe dieses Plausibilitätsargumentes aufgrund von physikalischen Tatsachen einige implizite Annahmen gemacht [Zwe84]:

1. Die einzelnen Moleküle haben keine inneren Freiheitsgrade, was sich dadurch auswirkt, dass alle Kollisionen vollkommen elastisch sind.

2. Die Dichte des Gases ist derartig gering, dass nur Kollisionen von zwei Molekülen (binäre Kollisionen) betrachtet werden müssen.

3. Die Geschwindigkeit $\mathbf{v}$ eines Moleküls ist nicht mit dem Ort $\mathbf{r}$ korreliert.

Wenn wir das Gleichgewicht der Gasmoleküle in einem beliebigen Teil $V \times \Pi \in \mathbb{R}^6$ des Phasenraumes (also des kartesischen Produkts aus Konfigurationsraum und Impulsraum) betrachten, so ist die Änderung der Teilchen im betrachteten Element, dargestellt durch das Integral der Teilchendichte $n_c f(\mathbf{r}, \mathbf{v}, t)$ über die Zelle, gleich der Differenz zwischen der Anzahl der "erzeugten" und der "vernichteten" Teilchen:

$$\frac{\partial}{\partial t} \int_{V \times \Pi} n_c f(\mathbf{r}, \mathbf{v}, t)\, d\mathbf{r} d\mathbf{v} = P - D$$

Für die rechte Seite sind 2 Prozesse von Interesse: Zum einen die Bewegung der Teilchen, die dazu führt, dass sich Teilchen aus V entfernen bzw. hineinwandern. Zum anderen ändern Teilchen durch Kollisionen ihre Geschwindigkeiten $(\mathbf{v}, \mathbf{w}) \mapsto (\mathbf{v}', \mathbf{w}')$.

Die Änderung der Teilchenzahl durch Bewegung der Teilchen in einem Element $d\mathbf{v}d\mathbf{r}$ des Phasenraumes kann durch das Oberflächenintegral

$$B_{d\mathbf{v}d\mathbf{r}} = \int_{\partial S} n_c f\left(\mathbf{r}, \mathbf{v}, t\right) \mathbf{v} \cdot d\mathbf{S} \tag{3.10}$$

ausgedrückt werden, wobei $d\mathbf{S}$ das Flächenelement der Oberfläche des Phasenraumelements darstellt.

Mit Hilfe des GAUSS'schen Satzes[1] (Divergenzsatz) kann Gleichung (3.10) umgeschrieben werden:

$$B = \int_{V\times\Pi} B_{d\mathbf{v}d\mathbf{r}} = \int_{V\times\Pi}\int_{d\mathbf{v}d\mathbf{r}} \nabla(n_c f\left(\mathbf{r}, \mathbf{v}, t\right) \mathbf{v}) d\mathbf{v}d\mathbf{r} = \int_{V\times\Pi} n\mathbf{v}\frac{\partial}{\partial\mathbf{r}} f\left(\mathbf{r}, \mathbf{v}, t\right) d\mathbf{r}d\mathbf{v} \tag{3.11}$$

Letztere Identität gilt, da n_c und $\mathbf{v}$ über das Phasenraumelement $d\mathbf{r}d\mathbf{v}$ als konstant gelten.

Analog wirkt eine äußere Kraft auf $\mathbf{v}$ gleich, wie die Geschwindigkeit auf $\mathbf{r}$. Daher ist bei einer äußeren Kraft $\mathbf{F}$ noch ein zusätzlicher Term zu berücksichtigen:

$$K = \int_{V\times\Pi} n_c\mathbf{F}\frac{\partial}{\partial\mathbf{v}} f\left(\mathbf{r}, \mathbf{v}, t\right) d\mathbf{r}d\mathbf{v} \tag{3.12}$$

Für die Behandlung der Streuung müssen die Geschwindigkeiten sowohl vor ($\mathbf{v}$, $\mathbf{w}$) als auch nach ($\mathbf{v'}$, $\mathbf{w'}$) der Streuung betrachtet werden.

Die Streurate für feste Geschwindigkeiten $(\mathbf{v}, \mathbf{w}) \mapsto (\mathbf{v'}, \mathbf{w'})$ für ein Teilchen mit Geschwindigkeit $\mathbf{v}$ wird bestimmt, indem ein Testteilchen mit Geschwindigkeit $\mathbf{v}$ durch fixe "Feldteilchen" mit Geschwindigkeit $\mathbf{w}$ fliegt. Die Wahrscheinlichkeit einer Kollision beträgt dabei $W = |\mathbf{v} - \mathbf{w}|\, \sigma d\Omega$, während $N_{\mathbf{w}} = n_c f_1(\mathbf{r}, \mathbf{v}, t) d\mathbf{w}$ Teilchen mit $\mathbf{w}$ im Einheitsvolumes des physikalischen Raumes liegen.

Die Gesamtzahl der Streuungen $(\mathbf{v}, \mathbf{v'}) \mapsto (\mathbf{v'}, \mathbf{w'})$ ergibt sich nun durch Integration über den betrachteten Phasenraumbereich:

$$S_1 = \int_{V\times\Pi} d\mathbf{v}d\mathbf{r} \int_{\mathbb{R}^3\times S^2} W N_{\mathbf{w}} n f\left(\mathbf{r}, \mathbf{v}, t\right) = \int_{V\times\Pi} d\mathbf{v}d\mathbf{r} \int_{\mathbb{R}^3\times S^2} |\mathbf{v} - \mathbf{w}|\, n_c^2 f_1 f d\Omega d\mathbf{w} \tag{3.13}$$

Umgekehrt können natürlich auch Teilchen in den Bereich $V\times\Pi$ gestreut werden, dich sich davor nicht in $V\times\Pi$ befanden. Es ergibt sich hierfür analog zu Gleichung (3.13) ein Term ([Bir94]):

$$S_2 = \int_{V\times\Pi} d\mathbf{v}d\mathbf{r} \int_{\mathbb{R}^3\times S^2} |\mathbf{v} - \mathbf{w}|\, n^2 f'_1 f' d\Omega d\mathbf{w} \tag{3.14}$$

mit den Bezeichnungen $f' = f\left(\mathbf{r}, \mathbf{v'}, t\right)$ und $f'_1 = f(\mathbf{r}, \mathbf{v'_1}, t)$.

Die gesamte Änderung durch Teilchenbewegung und Streuung ist damit:

$$P - D = -B - F - S_1 + S_2$$

[1] Es sei V ein Volumen in $\mathbb{R}^3$ mit Begrenzung ∂V und $\mathbf{F}$ ein Vektorfeld in $\mathbb{R}^3$. Dann ist

$$\int_V \nabla \cdot \mathbf{F} dV = \int_{\partial V} \mathbf{F} d\mathbf{a}$$

Eingesetzt in Gleichung (3.10) ergibt dies:

$$\int_{V\times\Pi} d\mathbf{r}d\mathbf{v}\left(n_c\frac{\partial f}{\partial t}+n_c\mathbf{v}\cdot\frac{\partial f}{\partial \mathbf{r}}+n_c\mathbf{F}\cdot\frac{\partial f}{\partial \mathbf{v}}+\right.$$
$$\left.\int\int|\mathbf{v}-\mathbf{w}|\,\sigma n_c^2\left(f_1 f-f' f_1'\right)d\Omega d\mathbf{w}\right)=0 \quad (3.15)$$

Da $V\times\Pi$ beliebig gewählt werden kann, muss der Integrand gleich 0 sein, und damit gilt

$$\frac{\partial f}{\partial t}+\mathbf{v}\cdot\frac{\partial f}{\partial \mathbf{r}}+\mathbf{F}\cdot\frac{\partial f}{\partial \mathbf{v}}=\int_{\mathbb{R}^3}\int_{S^2}|\mathbf{v}-\mathbf{w}|\,\sigma n_c\left({f'}_1 f'-f f_1\right)d\Omega d\mathbf{w}, \quad (3.16)$$

also die Boltzmann-Gleichung, für die Dichte $f(\mathbf{r},\mathbf{v},t)$ der Teilchenverteilung im Phasenraum.

3.3 Liouville-Gleichung und BBGKY-Hierarchie

Es sei $F_N:\left(\mathbb{R}^3\right)^N\times[0,T)\to\mathbb{R}_0^+$ die Dichtefunktion der Positionen aller Teilchen im Raum, wobei $[0,T)\subset\mathbb{R}_+$ das betrachtete Zeitintervall darstellt. Natürlich kann durch $F_N|_{C^N\times\mathbb{R}}=0$ auch ein unerlaubter Raum Bereich $C\subset\mathbb{R}^3$ existieren.

Definition 3.1. Die R-Teilchendichte (Randverteilung) für $1\le R<N$ ergibt sich auf F_N durch Integration:

$$F_R(\mathbf{r}_1,\ldots\mathbf{r}_R)=\int d\mathbf{r}_{R+1}\cdots d\mathbf{r}_N F_N(\mathbf{r}_1,\ldots\mathbf{r}_R,\ldots\mathbf{r}_N) \quad (3.17)$$

Von besonderem Interesse ist natürlich die 1-Teilchendichte F_1, die durch $f_N(\mathbf{r}_1,t)=NF_1(\mathbf{r}_1,t)$ propertional zur gesuchten physikalische Dichte des Gases ist. Da wir N ununterscheidbare Teilchen betrachten, ist F_N und damit F_R invariant gegen Vertauschung der Argumente:

$$F_k(\mathbf{r}_1,\ldots\mathbf{r}_i,\ldots\mathbf{r}_j,\ldots\mathbf{r}_N)=F_k(\mathbf{r}_1,\ldots\mathbf{r}_j,\ldots\mathbf{r}_i,\ldots\mathbf{r}_N)$$
$$\text{für } 1\le i<j\le k \text{ und } 1<k\le N$$

Als Ausgangspunkt wird nach [Har71] die Liouville-Gleichung aus der statistischen Physik benutzt:

$$\frac{\partial F_N}{\partial t}+\sum_{i=1}^{N}\frac{\mathbf{p}_i}{m}\frac{\partial F_N}{\partial \mathbf{r}_i}+\sum_{i=1}^{N}\dot{\mathbf{p}}_i\frac{\partial F_N}{\partial \mathbf{p}_i}=0 \quad (3.18)$$

Dabei bezeichnet $\mathbf{r}_i$ die Koordinaten der einzelnen Teilchen, $\mathbf{p}_i$ stellt den kanonischen Impuls dar, hier $\mathbf{p}_i=m\dot{\mathbf{r}}_i$. Unter Vernachlässigung von äußeren Kräften wirkt zwischen dem i-ten und dem j-ten Teilchen nur eine Kraft, die durch ein Potential der Form $\Phi_{ij}(|\mathbf{r}_i-\mathbf{r}_j|)$ verursacht wird. Damit ist also

$$\mathbf{F_i}=\dot{\mathbf{p}}_i=-\sum_{j=1}^{N}{}'\frac{\partial\Phi_{ij}}{\partial \mathbf{r}_i}$$

Aus Gleichung (3.18) kann durch Integration über $\mathbf{r}_{R+1},\ldots\mathbf{r}_N$ eine entsprechende Gleichung für F_R bestimmt werden.

Der erste Term ergibt:

$$\int d\mathbf{r}_{R+1}\cdots d\mathbf{r}_N \frac{\partial F_N}{\partial t} = \frac{\partial}{\partial t}\int d\mathbf{r}_{R+1}\cdots d\mathbf{r}_N F_N = \frac{\partial}{\partial t}F_R \tag{3.19}$$

da $\mathbf{r}_i$ und t unabhängige Variablen darstellen.

Für den zweiten Term ergibt sich

$$\int d\mathbf{r}_{R+1}\ldots d\mathbf{r}_N \sum_{i=1}^{N}\frac{\mathbf{p}_i}{m}\frac{\partial F_N}{\partial \mathbf{r}_i} = \sum_{i=1}^{R}\frac{\mathbf{p}_i}{m}\frac{\partial}{\partial \mathbf{r}_i}\int d\mathbf{r}_{R+1}\ldots d\mathbf{r}_N F_N + \sum_{i=R+1}^{N}\frac{\mathbf{p}_i}{m}\int d\mathbf{r}_{R+1}\ldots d\mathbf{r}_N \frac{\partial F_N}{\partial \mathbf{r}_i} \tag{3.20}$$

Da F_N jedoch eine Wahrscheinlichkeitsdichte darstellt, verschwindet diese Funktion im Unendlichen ($\lim_{x_i\to\pm\infty} F_N(x_1,\ldots x_i,\ldots x_N) = 0$). Dadurch fällt die zweite Summe weg.

Analog folgt für den letzten Term:

$$-\int d\mathbf{r}_{R+1}\ldots d\mathbf{r}_N \sum_{i=1}^{N}\frac{\partial \Phi_{ij}}{\partial \mathbf{r}_i}\frac{\partial F_N}{\partial \mathbf{p}_i} = -\sum_{i,j=1}^{N}\frac{\partial \Phi_{ij}}{\partial \mathbf{r}_i}\frac{\partial F_R}{\partial \mathbf{p}_i} - \int d\mathbf{r}_{R+1}\ldots d\mathbf{r}_N \left(\sum_{\substack{1\le i\le R\\ R+1\le j\le N}}\frac{\partial \Phi_{ij}}{\partial \mathbf{r}_i}\frac{\partial F_N}{\partial \mathbf{p}_i} + \sum_{\substack{R+1\le i\le N\\ 1\le q\le N}}\frac{\partial \Phi_{ij}}{\partial \mathbf{r}_i}\frac{\partial F_N}{\partial \mathbf{p}_i}\right) \tag{3.21}$$

Die letzte Summe verschwindet wie jene im zweiten Term, und wegen der Symmetrie von F_N wird Gleichung (3.21) zu

$$-(N-R)\int d\mathbf{r}_{R+1}\sum_{i=1}^{R}\frac{\partial \Phi_{i,R+1}}{\partial \mathbf{r}_i}\frac{\partial F_{R+1}}{\partial \mathbf{p}_i}(\mathbf{r}_1,\ldots \mathbf{x}_{R+1},t) \tag{3.22}$$

Insgesamt ergibt sich also für F_R die Gleichung [Har71], oder besser die Hierarchie von Gleichungen, da in der Gleichung für F_R auch F_{R+1} vorkommt:

$$\frac{\partial F_R}{\partial t} + \sum_{i=1}^{R}\frac{\mathbf{p}_i}{m}\frac{\partial F_R}{\partial \mathbf{r}_i} - \sum_{i,j=1}\frac{\partial \Phi_{ij}}{\partial \mathbf{r}_i}\frac{\partial F_r}{\partial \mathbf{p}_i} = (N-R)\int d\mathbf{r}_{R+1}\sum_{i=1}^{R}\frac{\partial \Phi_{i,R+1}}{\partial \mathbf{r}_i}\frac{\partial F_{R+1}}{\partial \mathbf{p}_i} \tag{3.23}$$

Diese Hierarchie wird nach Bogoliubov, Born, H.S. Green, Kirkwood und Yvon, die alle unabhängig die Gleichung (3.23) herleiteten, auch die BBGKY-Hierarchie genannt.

Unter den oben angegebenen vereinfachenden Annahmen 1 - 3 kann diese Hierarchie nach Grad [Gra58] und einigen anderen Autoren zur Boltzmann-Gleichung vereinfacht werden. Von besonderer Bedeutung für die Entkopplung der ersten und zweiten Gleichung der Hierarchie ist dabei der sogenannte Stoßzahlansatz von Boltzmann:

$$F_2(\mathbf{r}_1,\mathbf{r}_2,t) = F_1(\mathbf{r}_1,t)F_1(\mathbf{r}_2,t) \tag{3.24}$$

der im Boltzmann-Gas Limes, kurz BGL ($N\to\infty$, $m\to 0$, $\sigma\to 0$, $N\sigma^2 = \text{const}$, $Nm = \text{const}$), exakt gilt [Har71].

3.4 Einschränkungen und Verallgemeinerungen

Wie bei der Herleitung bereits mehrfach erwähnt, wurden einige Annahmen und Vereinfachungen getroffen, um zu dieser Form der Boltzmann-Gleichung zu gelangen. UHLENBECK zeigt in [Uhl73] einige dieser Schwachpunkte auf und erwähnt mögliche Lösungsansätze:

Die Boltzmann-Gleichung in der oben angegebenen Form wurde nur für monoatomare Gase hergeleitet. Bereits BOLTZMANN versuchte, die Gleichung zu verallgemeinern, doch schon Lorentz zeigte die grundlegenden Schwierigkeiten dabei auf [Uhl73]. Zur Beschreibung der inneren Freiheitsgrade der Moleküle, die ja bei der Herleitung als nicht existent angenommen wurden, für polyatomare Gase allerdings sehr wohl existieren, ist die Quantentheorie notwendig.

Die Einschränkung auf binäre Kollisionen ist außerdem nur gültig, falls $d \ll \lambda$, wobei λ die freie Weglänge und d die Reichweite des intermolekularen Potentials darstellt. Es konnte bisher keine Verallgemeinerung gefunden werden, die auch bei höheren Dichten noch gültig bleibt. Die Boltzmann-Gleichung ist nämlich nicht der erste Term in der Virialentwicklung. Es bestehen sogar Zweifel [Uhl73], ob es überhaupt eine geschlossene Gleichung für $f(\mathbf{r}, \mathbf{v}, t)$ für höhere Dichten gibt.

Die Boltzmann-Gleichung lässt die statistische Interpretation der Thermodynamik völlig außer acht, kann also die Fluktuationen im Gas nicht beschreiben. $f(\mathbf{r}, \mathbf{v}, t)$ sollte eine stochastische Variable darstellen, nicht eine deterministische, wie es in der Boltzmann-Gleichung der Fall ist.

Die Boltzmann-Gleichung ist eine rein klassische Gleichung, welche die Quantentheorie völlig außer Acht lässt. Diese ist aber nötig, um einige Feinheiten beschreiben zu können:

1. Wie oben schon erwähnt, können bei einem polyatomaren Gas die inneren Freiheitsgrade der Moleküle nur durch die Quantentheorie beschrieben werden.

2. Aber auch schon bei monoatomaren Molekülen kann die Quantentheorie nicht außer Acht gelassen werden. Falls nämlich die de Broglie-Wellenlänge $\Lambda = \frac{h}{\sqrt{mkT}}$ in etwa dieselbe Größenordnung wie die Größe der Moleküle hat, sind große Abweichungen von der klassischen Theorie zu erwarten. Es muss dann zur Beschreibung des Wirkungsquerschnitts die quantenmechanische Streutheorie benutzt werden. Diese Abweichungen wurden speziell bei Helium bei geringen Temperaturen experimentell untersucht.

3. Für Bosonen und Fermionen sollte im thermodynamischen Gleichgewicht die Bose- bzw. Fermi-Verteilung erhalten werden anstatt der Maxwell-Boltzmann- Verteilung. Mit anderen Worten, es muss bei Fermionen auch das Pauli-Prinzip berücksichtigt werden, dass es zwei Teilchen verbietet, sich in exakt demselben Zustand zu befinden.

3.5 Existenz und Eindeutigkeit der Lösung

Da die Boltzmann-Gleichung eine nichtlineare Integrodifferentialgleichung darstellt, ist die Frage nach der Existenz und Eindeutigkeit der Lösung natürlich alles andere als trivial zu beantworten.

Als einfachstes Modell könnte man das räumlich homogene, unendlich ausgedehnte Gas mit dem Hard Sphere Modell für das intermolekulare Potential bezeichnen. Für diesen Fall

konnte CARLEMAN [Car33] einen Existenz- und Eindeutigkeitssatz des Anfangswertproblems beweisen:

Satz 3.1 (Carleman [Car33], o.B.). *Wenn der Anfangswert $f_0(|\mathbf{v}|)$ für die Dichtefunktion $f(\mathbf{r},\mathbf{v},t) = f(|\mathbf{v}|,t)$ integrierbar, nicht-negativ und von oben durch $A_1(1-|\mathbf{v}|)^{-\alpha}$, $A_1 > 0$, $\alpha \geq 6$ konstant, beschränkt ist, so existiert genau eine Lösung $f(|\mathbf{v}|,t)$ der Boltzmann-Gleichung mit $f(|\mathbf{v}|,0) = f_0(|\mathbf{v}|)$. Diese ist beschränkt durch*

$$f(|\mathbf{v}|,t) \leq A_1 \left(1-|\mathbf{v}|\right)^{-\alpha}$$

Im weiteren Verlauf der Arbeit werde ich genau diesen Fall betrachten, also von räumlicher Homogenität, unendlicher Ausdehnung (der Divergenzterm auf der linken Seite der Boltzmann-Gleichung fällt dadurch weg) und dem Hard Sphere Modell ausgehen.

Zuvor allerdings möchte ich noch exemplarisch einige weitere Existenzsätze anführen. Für ein abgeschnittenes Maxwell Potential wurden ähnliche Ergebnisse von WILD, MORGENSTERN [Mor54] und GRAD [Gra58] bewiesen. WILD etwa schlägt in [Wil51] eine Lösungsmethode vor, wo die Boltzmann-Gleichung durch Integration in eine reine Integralgleichung umgewandelt wird, die dann rekursiv in sich selbst substituiert wird. Dies führt zu einer unendlichen Reihe, deren Terme immer verschachteltere Integrale sind. Für ein Maxwell-Gas zeigte WILD, dass diese Summe konvergiert und somit eine Lösung der Boltzmann-Gleichung darstellt. Leichter zu behandeln ist die linearisierte Boltzmann-Gleichung, von der ausgehend eine Verallgemeinerung auf die volle Boltzmann-Gleichung unter der Annahme nur geringer Abweichungen von der Linearität möglich ist. Das Randwertproblem der linearisierten Boltzmann-Gleichung benötigt allerdings auch schon verschärfte Voraussetzungen (siehe [Cer88]).

Als ein weiteres Beispiel für einen solchen Eindeutigkeitssatz möchte ich einen Satz von BELLOMO und TOSCANI angeben, der eine Anfangsverteilung f_0 eines unendlich ausgedehnten Gases beschreibt, die im Unendlichen mit $\frac{1}{|\mathbf{r}|^p}$ gegen 0 geht. Als Ergebnis wird das intuitiv klare und physikalisch relativ uninteressante Ergebnis erhalten, dass die Dichte f mit $t \to \infty$ gegen 0 geht, sich die Teilchen also quasi in der Unendlichkeit verlieren.

Der Operator $J(f,f)$ aus Definition (3.2) kann auch geschrieben werden als:

$$J(f,f) = J_1(f,f) - f\left(\mathbf{r},\mathbf{v},t\right) J_2(f) \tag{3.25}$$

$$\text{mit} \quad J_1 = \int \sigma(\vartheta, \mathbf{v}-\mathbf{v}') f\left(\mathbf{r},\mathbf{v}',t\right) f(\mathbf{r},\mathbf{w}',t) d\Omega dw \tag{3.26}$$

$$J_2 = \int \sigma(\vartheta, \mathbf{v}-\mathbf{v}') f(\mathbf{r},\mathbf{w},t) d\Omega dw \tag{3.27}$$

Die Boltzmann-Gleichung (3.1) mit dem Operator $J(f,f)$ aus Gleichung (3.25) kann über t integriert werden, was zu der Integralgleichung führt:

$$f\left(\mathbf{r},\mathbf{v},t\right) = f_0(\mathbf{r}-\mathbf{v}t,\mathbf{v}) + \int_0^t J_1(\mathbf{r}-\mathbf{v}(t-s),\mathbf{v},s)ds - \int_0^t f(\mathbf{r}-\mathbf{v}(t-s),\mathbf{v},t) J_2(\mathbf{r}-\mathbf{v}(t-s),\mathbf{v},s)ds \tag{3.28}$$

Es sei $C_t(\mathbb{R}^3 \oplus \mathbb{R}^3 \oplus \mathbb{R})$ der Raum aller stetigen Funktionen $f\left(\mathbf{r},\mathbf{v},t\right)$, $f : [0,T] \times \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}$, die mit $\mathbf{r}$ und $\mathbf{v}$ im Unendlichen gegen 0 gehen, ausgestattet mit der Norm

$$\|f\| = \sup_{t\in[0,T]} \sup_{\substack{\mathbf{r}\in\mathbb{R}^3 \\ \mathbf{v}\in\mathbb{R}^3}} \left|f\left(\mathbf{r},\mathbf{v},t\right)\right|$$

Definition 3.2 (milde Lösung der Boltzmann-Gleichung). Eine Funktion $f(\mathbf{r},\mathbf{v},t) \in C_T$ heißt *milde Lösung* der Boltzmann-Gleichung, wenn $f \geq 0$ f.ü. gilt und f die Integralgleichung (3.28) erfüllt.

Der Raum S_β mit $\beta > 0$ sei definiert als der Abschluss der reellwertigen Funktionen mit kompaktem Träger, gemeinsam mit der Norm

$$\|f\|_\beta = \int_{\mathbb{R}^3} \max_{\mathbf{r}} \left| e^{\beta|\mathbf{r}|^2} f(\mathbf{r},\mathbf{v}) \right| dv \tag{3.29}$$

Weiters bezeichne $S_\beta^+ = \{f \in S_\beta : f(\mathbf{r},\mathbf{v}) \geq 0 \quad \text{f.ü.}\}$, und L_j^1 den Raum der Funktionen $\mathbb{R}^3 \to \mathbb{R}$ mit der Norm $\|f\|_j = \int_{\mathbb{R}^3} |\mathbf{r}|^j |f(\mathbf{r})|\, d\mathbf{r}$.

Mit diesen Bezeichnungen lässt sich nun der Existenz- und Eindeutigkeitssatz von Illner und Shinbrot formulieren:

Satz 3.2 (Existenz- und Eindeutigkeit des AWP, [IS84]). *Es gelte die Anfangsbedingung $f_0(\mathbf{r},\mathbf{v}) \in S_\beta^+$. Wenn für die Funktion $\psi(\mathbf{v}) = \sup_{\mathbf{r}} \left(e^{|\mathbf{r}|^2} f_0(\mathbf{r},\mathbf{v}) \right)$ in $L_0^1 \cap L_1^1$ liegt und $q := 16\pi C \sqrt{\frac{\pi}{\beta}} \|\psi\|_0 < 1$ gilt, so hat das Anfangswertproblem*

$$\begin{aligned} \frac{d}{dt} f(\mathbf{r}+\mathbf{v}t,\mathbf{v},t) &= J(f,f)(\mathbf{r}+\mathbf{v}t,t) \\ f(\mathbf{r},\mathbf{v},t) &= f_0(\mathbf{r},\mathbf{v}) \end{aligned} \tag{3.30}$$

eine eindeutige globale Lösung $\hat{f} \in C_\beta^+$.

Beweisskizze. Die Boltzmann-Gleichung wird zerlegt in das sogenannte *getrennte Boltzmann-System*

$$\begin{aligned} \frac{d}{dt}\tilde{l} + \tilde{l}\tilde{R}(u) &= \tilde{J}(l,l) \\ \frac{d}{dt}\tilde{u} + \tilde{u}\tilde{R}(l) &= \tilde{J}(u,u) \\ u(0) &= f_0 = l(0) \end{aligned} \tag{3.31}$$

mit $R(f)(\mathbf{r},\mathbf{v},t) := \pi C \int_{\mathbb{R}^3} |\mathbf{v}-\mathbf{w}|\, f(\mathbf{r},\mathbf{w},t)\, d\mathbf{w}$ und der Notation, dass die Tilde $\tilde{}$ eine Auswertung an der Stelle $(\mathbf{r}+\mathbf{v}t,\mathbf{v},t)$ bedeutet. Wenn $u = l$ gilt, so ist u eine milde Lösung der Boltzmann-Gleichung.

Unter der Bedingung $q < 2$ und den restlichen Voraussetzungen des Satzes wird gezeigt, dass dieses System eine globale Lösung (l,u) besitzt. Mit der Verschärfung $q < 1$ folgt schließlich $l = u$, für die Eindeutigkeit wird ganz analog $l' = u$ und damit $l' = l$ (oder $u' = u$) gezeigt, wobei (l',u') eine weitere Lösung von Gleichung (3.31) darstellen. □

Bemerkung 3.1. Ohne die Voraussetzung $\psi \in L_0^1 \cap L_1^1$, dafür aber $q \in (\frac{1}{2},1)$ existiert ebenfalls eine eindeutige Lösung $\hat{f}$, allerdings gilt nicht zwingend $\hat{f} \geq 0$ f.ü..

Aufbauend auf Satz 3.2 konnten Bellomo und Toscani in [BT85] außerdem eine Abschätzung der Lösung unter einigen zusätzlichen Annahmen angeben: Der Wirkungsquerschnitt $\sigma(\vartheta,\mathbf{v}-\mathbf{w})$ lasse sich faktorisiert darstellen als $\sigma(\vartheta,\mathbf{v}-\mathbf{w}) = \beta_s |\mathbf{v}-\mathbf{w}|^{1-\frac{4}{s}}$. Für $s = 4$ beschreibt dies Maxwell-Moleküle, $s < 4$ bedeutet weiche, $s > 4$ harte Streuung und der Limes $s \to \infty$ beschreibt das bereits erwähnte Hard Sphere Modell, wo die Teilchen als vollkommen elastisch stoßende Kugeln beschrieben werden.

Satz 3.3 (Bellomo und Toscani [BT85], o.B.). *Die Funktion $\beta_s(\vartheta)$ mit $s > 4$ sei regulär und erfülle $\int_0^{\pi/2} \frac{\beta_s(\vartheta)}{\sin\vartheta\cos\vartheta} d\vartheta = F < \infty$. Die Anfangsbedingung $f_0(\mathbf{r}, \mathbf{v})$ sei stetig, gehe im Unendlichen gegen 0 und erfülle*

$$0 \leq f_0 \leq \alpha_{pq} e^{-q^2 v^2} \frac{1}{(1-r^2)^{p/2}}$$

wobei $p > 1$ und $\alpha_{pq} = \left(\frac{(p-1)}{64\pi^2}\frac{F}{q^2}\right)\left(2q^2\right)^{(s-4)/(4s)}$ eine Konstante darstellt. Dann besitzt das Cauchy'sche Anfangswertproblem eine eindeutige milde Lösung $\forall T > 0 : f(\mathbf{r}, \mathbf{v}, t) \in C_t$ mit

$$f(\mathbf{r}, \mathbf{v}, t) \leq 2\alpha_{pq} \frac{e^{-q^2 v^2}}{\left(1 + |\mathbf{r} - \mathbf{v}t|^2\right)^{p/2}}$$

Korollar 3.4. *Es gibt Anfangswerte f_0 mit $\|f_0\| > 0$, sodass $\lim_{t\to\infty} \|f\| = 0$.*

3.6 Lösungen der Boltzmann-Gleichung

Aufgrund der Komplexität der Boltzmann-Gleichung ist abgesehen von Näherungslösungen und Gleichgewichtslösungen[2] nur eine spezielle Lösung von Krook und Wu [KW77] bekannt für den Fall eines homogenen Gases mit isotroper Streuung.

Mit dem Wirkungsquerschnitt $\sigma(\vartheta, g) = \frac{k}{g} = \frac{3}{2\pi}\frac{1}{g}$ und der Bezeichnung $K(t) = 1 - \frac{2}{5}e^{-t}$ ist

$$f(v,t) = \frac{4}{K(t)^{\frac{5}{2}}\sqrt{2\pi}} \left(5K(t) - 3 + \frac{1-K(t)}{K(t)} v^2\right) e^{-\frac{v^2}{2K(t)}} \tag{3.33}$$

eine Lösung der Boltzmann-Gleichung. Diese Lösung konvergiert wie gefordert mit $t \to \infty$ gegen die Gleichgewichtslösung, unterscheidet sich aber auch für $t < \infty$ nicht allzu sehr von ihr.

Da sie jedoch die einzige bekannte nichtstationäre Lösung ist, wird sie im Kapitel 5 benutzt werden, um die diversen Simulationsschemata zu vergleichen.

3.7 Schwache Variante der Boltzmann-Gleichung

Im speziellen Fall eines homogenen isotropen Gases, den ich hier als Vereinfachung benutzen möchte, hängt f per definitionem nur vom Betrag von $\mathbf{v}$, nicht aber von $\mathbf{v}$ selbst ab. Außerdem bedeutet Homogenität, dass f von $\mathbf{r}$ überhaupt nicht abhängt:

$$f(\mathbf{r}, \mathbf{v}, t) = f(|\mathbf{v}|, t) \tag{3.34}$$

Außerdem soll hier nur isotrope Streuung betrachtet werden, σ soll also nicht von ϑ, sondern nur von $|\mathbf{v} - \mathbf{w}|$ abhängen.

[2] Für $t \to \infty$ geht die Verteilungsfunktion von v^2 - also die Lösung der Gleichung - gegen die Boltzmann-Verteilung

$$\frac{1}{2\pi^{3/2}} e^{-\frac{v^2}{2}} \tag{3.32}$$

welche die stationäre Lösung der Boltzmann-Gleichung darstellt.

Satz 3.5 (schwache Version der Boltzmann-Gleichung). *Sei f eine reguläre Funktion $\mathbb{R}_+ \times [0,T] \to \mathbb{R}_+$, welche die Boltzmann-Gleichung 3.1 erfüllt, und sei $B(\mathbb{R}_+)$ der Raum aller beschränkten messbaren Funktionen $\mathbb{R}_+ \to \mathbb{R}_+$, die auf ganz $\mathbb{R}_+$ definiert sind. Dann erfüllt f auch:*

$$\frac{d}{dt}\int_{\mathbb{R}_+} \varphi(v)v^2 f(v,t)dv = \frac{n_c}{(4\pi)^2}\int_{\mathbb{R}^6\times S^2} \left(\varphi(|\mathbf{v}'|) - \varphi(|\mathbf{v}|)\right)|\mathbf{v}-\mathbf{w}| \\ \sigma(\vartheta, |\mathbf{v}-\mathbf{w}|) f(|\mathbf{v}|,t) f(|\mathbf{w}|,t) d\mathbf{v}\, d\mathbf{w}\, d\nu, \varphi \in B(\mathbb{R}_+), t \in \mathbb{R}_+ \quad (3.35)$$

Beweis. Gleichung (3.1) wird mit einer Testfunktion $\varphi \in B(\mathbb{R})$ multipliziert und dann über $\mathbf{v}$ integriert. Dies ergibt (auf der linken Seite, die nur $|\mathbf{v}|$ abhängt, wird die Integration über die Polarwinkel φ und ϑ gleich ausgeführt):

$$4\pi\frac{\partial}{\partial t}\int_{\mathbb{R}_+} \varphi(v)v^2 f(v,t)dv = \frac{n_c}{4\pi}\int_{\mathbb{R}^3\times S^2_{v,w}\times R^3} \varphi(|\mathbf{v}|) \\ \left(f(v',t)f(w',t) - f(v,t)f(w,t)\right)|\mathbf{v}-\mathbf{w}|\,\sigma(\vartheta,|\mathbf{v}-\mathbf{w}|)d\mathbf{w}\, d\mathbf{n}\, d\mathbf{v} \quad (3.36)$$

Im Term mit $f(v',t)f(w',t)$ unter dem Integral wird nun die Variablentransformation $T_{\mathbf{n}} : (\mathbf{v},\mathbf{w}) \mapsto (\mathbf{v}',\mathbf{w}')$ mit $\det T_{\mathbf{n}} = 1$ durchgeführt. $\mathbf{v}'$ und $\mathbf{w}'$ gehen dabei in $\mathbf{v}$ und $\mathbf{w}$ über.

$$\frac{d}{dt}\int_{R_+} \varphi(v)v^2 f(v,t)dv = \frac{n_c}{4\pi}\int \left(\varphi(|\mathbf{v}'|) - \varphi(|\mathbf{v}|)\right) f(|\mathbf{v}|,t) f(|\mathbf{w}|,t)\,|\mathbf{v}-\mathbf{w}| \\ \sigma(\vartheta,|\mathbf{v}-\mathbf{w}|)d\mathbf{v}\, d\mathbf{w}\, d\mathbf{n}$$

Die folgenden Umformungen beziehen sich nur auf den Fall $\sigma(\vartheta,|\mathbf{v}-\mathbf{w}|) = const.$ und $n_c = 1$, können aber auf den allgemeinen Fall erweitert werden und liefern dann genau obiges Ergebnis.

Eine neuerliche Transformation für $\mathbf{w} \neq \mathbf{w}$

$$T_{\mathbf{v},\mathbf{w}} : \mathbf{n} \in S^2_{\mathbf{v},\mathbf{w}} \mapsto \nu = \frac{1}{|\mathbf{v}-\mathbf{w}|}\left(\mathbf{v}-\mathbf{w}-2\mathbf{n}\cdot(\mathbf{v}-\mathbf{w})\mathbf{n}\right) \in S^2$$

ergibt nun

$$\frac{d}{dt}\int_{R_+} \varphi(v)v^2 f(v,t)dv = \frac{k}{(2\pi)^2}\int_{\mathbb{R}^6\times S^2} \left(\varphi(|\mathbf{v}'|) - \varphi(|\mathbf{v}|)\right) f(|\mathbf{v}|)f(|\mathbf{w}|)d\mathbf{v}\, d\mathbf{w}\, d\nu \quad (3.37)$$

wobei nun $\mathbf{v}' = \frac{1}{2}\left(\mathbf{v}+\mathbf{w}+|\mathbf{v}-\mathbf{w}|\,\mathbf{v}\right)$ ist. In einer letzten Transformation sei nun

$$v = |\mathbf{v}| \qquad w = |\mathbf{w}| \quad (3.38)$$

$$a = \frac{1}{2} + \frac{(\mathbf{v}+\mathbf{w})\cdot\nu}{2\,|\mathbf{v}+\mathbf{w}|} \qquad b = \frac{1}{2} + \frac{\mathbf{v}\cdot\mathbf{w}}{2\,|\mathbf{v}|\,|\mathbf{w}|} \quad (3.39)$$

Mit $\mathbf{v}' = [v,w;a,b] := \frac{1}{\sqrt{2}}\sqrt{v^2+w^2+\sqrt{(v^2+w^2)^2 - 4(2b-1)^2v^2w^2(2a-1)}}$ gilt nun die schwache Version der Boltzmann-Gleichung:

$$\frac{d}{dt}\int_{\mathbb{R}_+} \varphi(v)v^2 f(v,t)dv = 4k\pi\int_{\mathbb{R}^2_+\times I^2} \left(\varphi(v') - \varphi(v)\right) v^2 f(v,t)f(w,t)dv\, dw\, da\, db \quad (3.40)$$

□

3.8 Master-Gleichung von KAC

KAC's Zugang zur Boltzmann-Gleichung beruht auf sogenannten Master-Gleichungen. Dies sind Bilanzgleichungen für die zeitliche Änderung einer Wahrscheinlichkeitsverteilung $f(A,t)$. Master-Gleichungen haben nach [ABC71] die Form

$$\frac{\partial}{\partial t} f(A,t) = \int dA \left(W(A|A') f(A',t) - W(A'|A) f(A,t) \right) \tag{3.41}$$

wobei die W die Übergangswahrscheinlichkeiten darstellen, und

können aus der Liouville-Gleichung abgeleitet werden. Aus Master-Gleichungen können nicht nur die Erwartungswerte von Größen, sondern auch der Zusammenhang der Schwankungen mit den Mittelwerten berechnet werden. Unter anderem kann bestimmt werden, wie Schwankungen durch äußere Einflüsse abklingen.

In [Kac59] leitet KAC unter einigen vereinfachenden Schritten aus seiner allgemeinen Master-Gleichung, die etwa von BELOTSERKOWSKIY und YANITSKIY in [BY75] simuliert wird, die vereinfachte Gleichung

$$\frac{\partial f(x,t)}{\partial t} = \frac{\nu}{2\pi} \int_{\mathbb{R}} dy \int_0^{2\pi} (f(x\cos\vartheta - y\sin\vartheta, t) f(-x\sin\vartheta + y\cos\vartheta, t) - f(x,t) f(y,t))\, d\vartheta \tag{3.42}$$

her als Beschreibung eines eindimensionalen räumlich homogenen Gases in der Boltzmann-Theorie. Die gravierendste physikalische Annahme ist die Aufgabe der Impulserhaltung bei der Herleitung. Massen- und Energieerhaltung ist allerdings gegeben.

LÉCOT und COULIBALY [LC96a] simulierten diese Gleichung mittels einer Quasi-Monte Carlo Methode mit dem Ergebnis, dass sie eine bessere Konvergenzrate sowie einen kleineren Fehlerexponent als NANBU's oder BIRD's Schema besitzt.

Kapitel 4

Simulation der Boltzmann-Gleichung

Inhaltsangabe

4.1 Entkopplung **51**
4.2 Notation **51**
4.3 Die Anfangswerte **52**
4.4 Schema von Belotserkowskiy-Yanitskiy **53**
4.4.1 Modifikationen des BY-Schemas 53
4.5 Bird's DSMC-B und NTC Methoden **53**
4.5.1 TC-Methode 54
4.5.2 NTC Methode 54
4.6 Nanbu's DSMC-N Schema **54**
4.7 Lécot's LD Methode **59**
4.7.1 Beweis von Satz 4.3 60
4.8 Lécot's QMC1 Schema **62**
4.8.1 Die benutzte $(0,5)$-Folge 64
4.8.2 Beweis des Konvergenzsatzes 4.7 65
4.8.3 3-dimensionale Variante der QMC Methode 68

Die zahlreichen bekannten Schemata zur Simulation der Boltzmann-Gleichung von Bird [Bir94], Belotserkowskiy - Yanitskiy [BY75], Deshepande[Des78], Koura[Kou70], Nanbu [Nan80], Lécot ([Léc89a], [Léc89b] und [Léc91]) und einigen anderen lassen sich grob einteilen in zwei Gruppen (nach [Nan83]):

Zum einen sind dies jene Mathoden, die direkt aus der Boltzmann-Gleichung abgeleitet sind. Diese Gruppe umfasst im wesentlichen Nanbu's stochastisches direktes Monte-Carlo Simulations (DSMC-N, manchmal aus SS genannt) Schema [Nan80], Babovsky's Modifikation DSMC-N* [Bab86] dessen und die LD-Methode von Lécot [Léc89a], welche lediglich eine Quasi-Monte Carlo Version des DSMC-N Schemas von Nanbu darstellt.

Die zweite Gruppe von Schemata leitet sich aus der sogenannten Master- Gleichung von Kac ab, die sich für $N \to \infty$ und der Annahme, dass zu $t = 0$ die Teilchen-Geschwindigkeiten unabhängig sind, zur Boltzmann-Gleichung vereinfacht. Sie umfasst das Schema von Belotserkowskiy-Yanitskiy (B-Y) als exaktes Simulationsschema, sowie die Schemata von Koura und Deshepande als direkte Modifikationen. Auch das zur Simulation der Boltzmann-Gleichung am öftesten benutzte Schema von Bird (ein direct simulation Monte Carlo Schema,

kurz DSMC-B Schema) passt nach [Nan83] als eine Modifikation des B-Y-Schemas in diese Gruppe.

BIRD hatte sein Schema allerdings nur unter Zuhilfenahme der zu Grunde liegenden Physik der Teilchenkollisionen erstellt, nicht aber unter Referenz zur Boltzmann-Gleichung oder zur Master-Gleichung von KAC. Auf Grund einiger Vereinfachungen ist bis heute nicht klar, ob letzteres Schema wirklich die exakte Lösung der Boltzmann-Gleichung liefert (siehe [Nan83]). Dass sie zumindest sehr gute Ergebnisse liefert, ist allerdings unbestritten.

Alle hier betrachteten Schemata simulieren die Dichtefunktion f direkt durch Molekül- bzw. Teilchenbewegung. Daher stammt auch die generische Bezeichnung DSMC (Direct Simulation Monte Carlo) für Monte Carlo-Verfahren, die mittels Teilchenbewegung arbeiten. Es gab in den frühen 60er-Jahren noch andere Methoden (siehe [Bir94]), welche sich jedoch nicht durchsetzen konnten.

4.1 Entkopplung

Allen Methoden zur Simulation der Boltzmann-Gleichung gemeinsam ist, dass die Kollision der Teilchen getrennt und unabhängig von der Bewegung der Teilchen zwischen den Kollisionen behandelt wird. Physikalische Argumente hierfür brachte bereits BIRD [Bir76], NANBU [Nan80] zeigte jedoch, dass sich dieses Prinzip auch direkt aus der Boltzmann-Gleichung ergibt.

Für kleine Δt gilt durch lineare Näherung für die Zeitableitung:

$$f(\mathbf{r},\mathbf{v},t) = f(\mathbf{r},\mathbf{v},0) + \left.\frac{\partial f(\mathbf{r},\mathbf{v},t)}{\partial t}\right|_{t=0} t + O\left(\Delta t^2\right) \tag{4.1}$$

Unter Verwendung der Boltzmann-Gleichung (3.1) ergibt dies:

$$\begin{aligned} f(\mathbf{r},\mathbf{v},\Delta t) &= f(\mathbf{r},\mathbf{v},0) + (-Df(\mathbf{r},\mathbf{v},0) + Jf(\mathbf{r},\mathbf{v},0)) + O\left(\Delta t^2\right) \\ &= (1 - \Delta tD + \Delta tJ)\, f(\mathbf{r},\mathbf{v},0) + O\left(\Delta t^2\right) \\ &= (1 + \Delta tJ)\,(1 - \Delta tD)\, f(\mathbf{r},\mathbf{v},0) + O\left(\Delta t^2\right) \\ &= (1 - \Delta tD)\,(1 + \Delta tJ)\, f(\mathbf{r},\mathbf{v},0) + O\left(\Delta t^2\right) \end{aligned} \tag{4.2}$$

Diese Gleichung zeigt, dass die Simulation tatsächlich in zwei Schritten erfolgen kann. Zunächst wird mittels

$$H(\mathbf{c},\mathbf{r}) := (1 + \Delta tJ) f(\mathbf{r},\mathbf{v},0) \tag{4.3}$$

die Kollision der Teilchen simuliert, und anschließend werden diese nach

$$f(\mathbf{r},\mathbf{v},\Delta t) = (1 + \Delta tJ) H(\mathbf{c},\mathbf{r}) + O\left(\Delta t^2\right) \tag{4.4}$$

verschoben.

In der von uns betrachteten Situation wird das Gas in Zellen unterteilt, für die jeweils die Wahrscheinlichkeitsdichte konstant angenommen werden kann, d.h. $f(\mathbf{r},\mathbf{v},t)$ ist unabhängig von $\mathbf{r}$ innerhalb der Zelle.

4.2 Notation

Um die Formeln in den nächsten Abschnitten etwas kompakter schreiben zu können, benötigen wir einige Abkürzungen, die sich aus der benutzten Mechanik der Teilchenkollisionen ergeben.

Für $(v, w, x^{(4)}, x^{(5)}) \in \mathbb{R}_+^2 \times I^2$ sei:

$$[v, w; a, b] := \frac{\sqrt{v^2 + w^2 + \sqrt{(v^2+w^2)^2 - 4(2a-1)^2 v^2 w^2}(2b-1)}}{\sqrt{2}} \tag{4.5}$$

$$[v, w; a] := \sqrt{v^2 + w^2 - 2(2a-1)v^2 w^2} \tag{4.6}$$

Weiters soll - wenn nicht anders angegeben - $\chi_{i,j}$ die charakteristische Funktion des Intervalls $[\frac{i-1}{N}, \frac{i}{N}) \times [\frac{j-1}{N}, \frac{j}{N})$ für $1 \leq i \leq N$ und $c_{i,j}^{(n)}(x^{(3)}, x^{(4)})$ die charakteristische Funktion von $\left\{(x^{(3)}, x^{(4)}) \in I^2 : x^{(3)} \leq \Delta t q([v_i^{(n)}, v_j^{(n)}; x^{(4)}])\right\}$ darstellen. Falls q in der entsprechenden Methode definiert ist, sei außerdem χ_i die charakteristische Funktion von $[q, 1) \times \left[\frac{i-1}{N}, \frac{i}{N}\right)$.

Zur Simulation wird eine Folge $\mathbf{X} = \left\{x_n \in I^5 : n \geq 1\right\}$ betrachtet, wobei für jeden Zeitschritt $\mathbf{X}^{(n)} = \{x_\rho : nN < \rho \leq (n+1)N\}$, $n \geq 0$ bezeichnet.

4.3 Die Anfangswerte

Um die Anfangsbedingung $f_0(\mathbf{v}, \mathbf{r}) = f_0(v) = f(v, 0)$ zu nähern, benötigen wir eine mit der Dichte $f_0(v)$ verteilte Menge $V^{(0)}$ von Geschwindigkeiten. Aus der Wahrscheinlichkeitstheorie ist bekannt, dass durch die Quantilfunktion Q_0, welche die Inverse der Verteilungsfunktion $F_0(v) = \int_0^v dw w^2 f(w)$ darstellt, eine gleichverteilte Zufallsvariable in eine Zufallsvariable mit Dichte f_0 überführt wird.

Damit genügt es nun, eine Menge X_0 von N auf I gleichverteilten Zufallszahlen mit $Q_0 = F_0^{-1}$ zu transformieren, um zur Anfangsverteilung zu gelangen:

$$V^{(0)} = Q_0(X_0) = F_0^{-1}(X_0) \tag{4.7}$$

Anstelle von Zufallszahlen x_i kann aber auch eine N-elementige Punktmenge mit kleiner Diskrepanz benutzt werden. Nach Abschnitt 1.2 (Bemerkung 1.3) ist $U = \left\{\frac{2i-1}{2N} : 1 \leq i \leq N\right\}$ jene Menge mit kleinster Sterndiskrepanz $D^*(U) = \frac{1}{2N}$. Definieren wir nun $V^{(0)} = F_0^{-1}(U)$, so gilt

$$\begin{aligned} D*(V^{(0)}, v^2 f_0(v)) &= \sup_{r>0} \left| \frac{1}{N} \sum_{1 \leq i \leq N} \chi_{[0,r)}\left(F_0^{-1}(u_i)\right) - \int_{[0,r)} v^2 f_0(v) dv \right| \\ &= \sup_{r>0} \left| \frac{1}{N} \sum_{1 \leq i \leq N} \chi_{[0,F_0(r))}(u_i) - F_0(r) \right| \\ &= \sup_{J=[0,F=F(r))} \left| \frac{1}{N} \sum_{1 \leq i \leq N} \chi_J(u_i) - \lambda(J) \right| = D_N^*(U) \end{aligned} \tag{4.8}$$

Dies stellt den kleinstmöglichen Wert für die Diskrepanz der Anfangsverteilung dar.

Im folgenden werde ich die Anfangswerte $V_{MC}^{(0)} = Q_0(X_0)$ als Monte-Carlo (MC) Anfangswerte, und die Menge $V_{QMC}^{(0)} = Q_0(U)$ als QMC-Anfangswerte bezeichnen.

4.4 Schema von Belotserkowskiy-Yanitskiy

Dieses Schema ist aus dem stochastischen Prozess abgeleitet, der zu Kac's Master Gleichung äquivalent ist. Diese wird asymptotisch zur Boltzmann-Gleichung für $N \to \infty$ und der Voraussetzung, dass die N Geschwindigkeiten zum Zeitpunkt $t = 0$ unabhängig sind.

Die Kollisionswahrscheinlichkeit zweiter Teilchen (i, j) im Zeitintervall $(t, t + \Delta t)$ ist

$$P_{ij} = \frac{n_c \Delta t}{N} g_{ij} \sigma_T(g_{ij}) \tag{4.9}$$

Es werden nun nacheinander einzelne Kollisionen betrachtet, wobei diese einen Poisson-Prozess mit Kollisionsrate $\rho(t) = \frac{n_c}{N} \sum_{i,j} g_{ij} \sigma_T(g_{ij})$ darstellen, die Zeit T_k für eine Kollision also verteilt sind nach:

$$\begin{aligned} P[T_1 \leq \tau] &= 1 - e^{-\rho(t)\tau} \\ &\vdots \\ P[T_k \leq \tau] &= 1 - e^{-\rho(t+\tau_1+\ldots+\tau_{k-1})\tau} \end{aligned} \tag{4.10}$$

wobei die τ_j Realisationen der Zufallsvariablen T_j darstellen.

Es werden nun so lange Kollisionen simuliert, bis $\tau_1 + \ldots + \tau_k \leq \Delta t < \tau_1 + \ldots + \tau_{k+1}$. Dabei wird zuerst ein Paar (i, j) nach der Verteilung (4.12) gesampelt, sowie τ_k nach 4.10. Die Kollision an sich läuft nach Abschnitt 3.1 ab, wobei in unserem homogenen isotropen Fall die Geschwindigkeiten nach der Kollision durch $[v, w; a, b]$ und $[w, v; a, 1 - b]$ nach Gleichung (4.5) und Gleichung (4.6) gegeben sind (mit (Quasi-)Zufallszahlen a und b). Die beiden Geschwindigkeiten werden nun sofort ersetzt, sodass auch die Reihenfolge der Kollisionen in einem Zeitschritt das Ergebnis beeinflusst. Abschließend muss nach jeder Kollision noch $\rho(t + \tau_1 + \ldots + \tau_k)$ aktualisiert werden.

Im Spezialfall der benutzten Maxwell-Moleküle allerdings ist $\rho(t) = const.$, sodass alle T_j dieselbe Verteilung aufweisen und sich der durch die Berechnung der $\rho(t + \tau_1 + \ldots + \tau_k)$ ergebende Aufwand der Ordnung N^2 reduziert.

4.4.1 Modifikationen des BY-Schemas

Die Modifikation von Koura [Kou70] geht in allen Fällen von einem während $(t, t + \Delta t)$ konstanten $\rho(t)$ aus, ebenso wie Deshepande [Des78], wo allerdings die Zahl der Kollisionen in jedem Zeitschritt zu Beginn nach einer Poisson-Verteilung gesampelt wird, anstatt die Zeit für jede Kollision zu sampeln.

4.5 Bird's DSMC-B und NTC Methoden

Wie schon mehrmals erwähnt, simuliert Bird nicht die Boltzmann-Gleichung selbst, sondern den physikalischen Gasfluss, der von der Boltzmann-Gleichung beschrieben wird. Aus diesem Grund möchte ich hier die "direct simulation Monte Carlo" (kurz DSMC-B, auch "Time Counter", kurz TC, Methode genannt) und die "No Time Counter" (NTC) Methode nur kurz beschreiben, ohne näher drauf einzugehen.

4.5.1 TC-Methode

Bei dieser in [Bir76] beschriebenen Methode wird nach [Nan83] jeder Teilchenkollision eine gewisse Zeit

$$T_k = \frac{2}{N n_c g_{ij} \sigma_T(g_{ij})} \tag{4.11}$$

zugeordnet, wobei das Kollisionspaar (i, j) aus der Wahrscheinlichkeitsverteilung

$$\frac{P_{ij}}{\sum_{i',j'} P_{i'j'}} = \frac{g_{ij}\sigma_T\left(g_{ij}\right)}{\sum_{i',j'} g_{i'j'}\sigma_T\left(g_{i'j'}\right)} \tag{4.12}$$

bestimmt wird. Es werden nun so lange Teilchenkollisionen simuliert, bis die Realisierungen τ_k der Zufallsvariablen T_k die Bedingung

$$\tau_1 + \cdots + \tau_X > \Delta t \tag{4.13}$$

erfüllen. Die Teilchenkollisionen werden dabei gleich wie beim B-Y Schema realisiert, und die Geschwindigkeiten sofort ersetzt. Die Geschwindigkeit zur Zeit $\tau_1 + \cdots + \tau_X$ werden schließlich als Geschwindigkeiten zum Zeitpunkt Δt verwendet.

Anstatt für die Auswahl von (i, j) die Gleichung (4.12) zu benutzen, führte BIRD die sogenannte "Acceptance-Rejection" Methode ein: Es sei g^* eine Geschwindigkeit größer als alle relativen Geschwindigkeiten g_{ij}. Ein zufällig gewähltes Paar (i', j') wird dann zur Kollision benutzt, wenn für eine gleichverteilte Zufallszahl $\tilde{x} < \frac{g_{i'j'}\sigma_T(g_{i'j'})}{g^*\sigma_T(g^*)}$ gilt, ansonsten wird solange ein anderes Paar zufällig gewählt, bis schließlich ein Paar selektiert wird.

Wie NANBU in [Nan83] argumentiert, gibt es Fälle, in denen dieses Schema sicherlich nicht exakt gilt.

4.5.2 NTC Methode

In einer Modifikation [Bir94] der TC Methode wird der Zeitschritt für die jeweilige Kollision gar nicht mehr benutzt, sondern es werden gleich zu Beginn jedes Zeitschrittes Δt genau $\frac{1}{2} N \bar{N} F_N \left(\sigma_T c_r\right)_{max} \frac{\Delta t}{V_C}$ Paare ausgewählt, die mit einer Wahrscheinlichkeit von $\frac{\sigma_T c_r}{\left(\sigma_T c_r\right)_{max}}$ eine Kollision erfahren.

BIRD's TC-Methode ist insofern eine Modifikation des B-Y Schemas, als nur die Verteilungsfunktion der T_k derartig modifiziert wird, dass $\rho\left(t + \sum \tau_i\right)$ gar nicht benutzt wird und daher auch nicht berechnet werden muss (was einen Aufwand von N^2 hervorrufen würde.

4.6 NANBU's DSMC-N Schema

NANBU's stochastisches Simulationsschema (DSMC-N oder SS-Schema) ist deshalb interessant, da es sich als einziges direkt und exakt aus der Boltzmann-Gleichung ableitet, und somit auch hervorragend geeignet ist, als Grundlage für ein Quasi-Monte Carlo Verfahren für diese Gleichung zu dienen.

Nanbu's Idee bestand darin, den Wechselwirkungsterm $J(f, f)$ direkt zu simulieren. Ausgangspunkt ist wie üblich das Anfangswertproblem (3.1) ($\Lambda \subset \mathbb{R}^3$ ist ein beschränkter Raumbereich, auf den das Gas eingeschränkt ist) mit der Wandbedingung

$$f\left(\mathbf{r}, \mathbf{v} - 2\left(\mathbf{n}(\mathbf{r})\cdot\mathbf{v}\right)\cdot\mathbf{n}(\mathbf{r}), t + 0\right) = f(\mathbf{r}, \mathbf{v}, t - 0) \tag{4.14}$$

am Rand $\partial\Lambda$. Diese Randbedingung beschreibt sogenannte spekulare Reflexion am Rand $\partial\Lambda$, wobei angenommen werden muss, dass die innere Normale $\mathbf{n}(\mathbf{r})$ für fast alle $\mathbf{r} \in \partial\Lambda$ existiert. Da wir in unseren Simulationen allerdings immer unendlich ausgedehntes Gas annehmen ist die Bedingung (4.14) für uns nicht von Bedeutung. BABOVSKY und ILLNER [BI89] allerdings konnten für diese Randbedingung die Konvergenz von NANBU's Schema zeigen.

Die Funktion $\sigma(\vartheta, |\mathbf{v} - \mathbf{w}|)$ im $J(f, f)$-Term (3.2) wird auch Boltzmann-Kern genannt, und der Vorfaktor des Integrals ist reziprok proportional zur mittleren freuen Weglänge.

Die Bewegung der Teilchen zwischen den Kollisionen geschieht mathematisch durch einen freien Fluss Φ_t: Ein Teilchen mit $(\mathbf{r}, \mathbf{v})$ bewegt sich entlang $(\mathbf{r}_t, \mathbf{v}_t) = \Phi_t(\mathbf{r}, \mathbf{v})$, bis es an eine Wand stößt und sich dadurch seine Geschwindigkeit gemäß Gleichung (4.14) ändert. Damit ist Φ_t i.A. für alle $(\mathbf{r}, \mathbf{v})$ definiert. (Es gibt allerdings nach [BI89] spezielle Formen von $\partial\Lambda$ wo dies nicht gilt.)

Für $f : \Lambda \times \mathbb{R}^3 \to \mathbb{R}$ sei eine Norm definiert durch $\|f\|_\alpha = \operatorname{ess\,sup}_{\mathbf{r},\mathbf{v}} \left| e^{\alpha \mathbf{v}^2} f(\mathbf{r}, \mathbf{v}) \right|$ und damit B_α als der Raum der messbaren Funktionen f mit $\|f\|_\alpha < \infty$. Durch diesen Fluss wird auch ein Operator M_t induziert auf B_α durch

$$M_t f(\mathbf{r}, \mathbf{v}) = f(\Phi_{-t}(\mathbf{r}, \mathbf{v}))$$

In dieser Schreibweise wird der Euler-Schritt (4.3) zu

$$H(\mathbf{y}, \mathbf{w}, t + \Delta t) = f(\Phi_{-\Delta t}(\mathbf{y}, \mathbf{w}) + \Delta t J(f, f)(\Phi_{-\Delta t}(\mathbf{y}, \mathbf{w})) \tag{4.15}$$

und die Bewegung der Teilchen erfolgt einfach als

$$f(\mathbf{y}, \vec{w}, t + \Delta t) = H(\Phi_{-\Delta t}(\mathbf{y}, \mathbf{w}), t + \Delta t) \tag{4.16}$$

Bevor nun die Kollisionen betrachtet werden können, wird nun auch der Raum Λ diskretisiert, indem Zellen eingeführt werden, in denen $f_z(\cdot, t)$ nicht von $\mathbf{r}$ und $\mathbf{v}$ abhängt, also homogen ist. Damit ist natürlich auch $H(\mathbf{y}, \mathbf{w}, t + \Delta t)$ homogen, eine Eigenschaft die aber für $f(\cdot, t + \Delta t)$ durch $\Phi_{-\Delta t}$ wieder zerstört wird. Für den nächsten Zeitschritt muss damit jede Zelle erneut homogenisiert werden.

Die schwache Version der Boltzmann-Gleichung mit einem allgemeinen Wirkungsquerschnitt $\sigma(\vartheta, |\mathbf{v} - \mathbf{w}|)$ ergibt sich analog zu (3.35) und mit der Zeitdiskretisierung (4.3) zu ([BI89]):

$$\int_{\mathbb{R}} \varphi(v) f_{j+1}(v) dv = \iint_{\mathbb{R}^2} K_{v,w} \varphi f_j(v) f_j(w) dv \, dw \tag{4.17}$$

mit dem sogenannten Übergangskern

$$K_{v,w}\varphi = \left(1 - \Delta t \int_{S^2} \sigma\left(\vartheta, |\mathbf{v} - \mathbf{w}|\right) d\omega\right) \varphi(v) + \Delta t \int_{S^2} \sigma\left(\vartheta, |\mathbf{v} - \mathbf{w}|\right) \varphi(v') d\omega \tag{4.18}$$

Besonders wichtig ist hier, dass $K_{\mathbf{v},\mathbf{w}}\varphi$ nicht von f_j abhängt, weshalb obige Gleichung als Grundlage des stochastischen Spiels dienen kann.

Definition 4.1 (Schwache Konvergenz). Eine Folge von Punktmaßen

$$f^{(n)} = \left(\frac{1}{N} \sum_{i=1}^{N} \delta(\mathbf{v}_i^N)\right)_{N=1,2,\ldots} \tag{4.19}$$

konvergiert schwach gegen eine Funktion f, wenn für alle beschränkten stetigen Funktionen φ gilt: $\frac{1}{N} \sum_{i=1}^{N} \varphi(\mathbf{v}_i^N) \xrightarrow{N \to \infty} \int \varphi(\mathbf{v}) f(\mathbf{v}) d\mathbf{v}$.

Definition 4.2 (konvergentes Simulationsschema). Eine Folge $\left(S^N(j)\right)_{N\in\mathbb{N}}$ von Abbildungen $S^N(j) : \mathbb{R}^{3N} \to \mathbb{R}^{3N}$ wird ein konvergentes Simulationsschema für die Maße $f_j(\mathbf{v})d\mathbf{v}$, $j = 0, 1, 2, \ldots$ genannt, wenn aus

$$\frac{1}{N}\sum_{i=1}^{N} \delta(\mathbf{v}_i^N) \to_w f_0(\mathbf{v})d\mathbf{v}$$

folgt, dass

$$\frac{1}{N}\sum_{i=1}^{N} \delta(S^N(j)\left((\mathbf{v}_i^N)\right) \to_w f_j(\mathbf{v})d\mathbf{v}$$

für alle $j = 1, 2, \ldots$.

Bemerkung 4.1. Diese Bedingung sagt nichts anderes aus, als dass für eine Näherung $\mathbf{v}_i^N$ der Anfangsbedingung f_0 auch die durch $S^N(j)$ um j (Zeit-) Schritte weiterbewegten Samples eine Näherung für die entsprechende "exakte" Lösung $f_j(\mathbf{v})$ darstellen.

Für die Formulierung des Schemas (und auch im Beweis des Konvergenzsatzes in [BI89]) wird noch die Hilfsfunktion $\Phi_{\mathbf{v},\mathbf{w}} : B^1 \to S^2_+$ benötigt, wobei B^1 den Kreis mit Fläche 1 im $\mathbb{R}^2$ bezeichnet:

Lemma 4.1 (aus [BI89]). $\forall \mathbf{v}, \mathbf{w} \exists$ *stetige Abbildung* $\Phi_{\mathbf{v},\mathbf{w}} : B^1 \to S^2_+$, *sodass* $K_{\mathbf{v},\mathbf{w}}\varphi = \iint\limits_{B^1} \varphi\left(T_{\mathbf{v},\mathbf{w}} \circ \Phi_{\mathbf{v},\mathbf{w}}(\mathbf{r})\right) d\mathbf{r}\ \forall\varphi$. *Es bezeichne dann* $\psi(\mathbf{r}, \mathbf{v}, \mathbf{w}) := T_{\mathbf{v},\mathbf{w}} \circ \Phi_{\mathbf{v},\mathbf{w}}(\mathbf{r})$.

Bemerkung 4.2. $T_{\mathbf{v},\mathbf{w}}$ ist die in Abschnitt 3.1 definierte Transformation $(\mathbf{v}, \mathbf{w}) \mapsto (\mathbf{v}', \mathbf{w}')$.

Wenn nun $\frac{1}{N}\sum_{i=1}^{N} \delta(\mathbf{v}_i^N)$ schwach gegen $f_j(\mathbf{v})d\mathbf{v}$ konvergiert, so wird nach NANBU [Nan80] bzw. BABOVSKY und ILLNER [BI89], die das DSMC-N Schema für ihren Konvergenzbeweis in der hier dargestellten Form formulierten, eine neue Folge von Maßen $\frac{1}{N}\sum_{i=1}^{N} \delta(\tilde{\mathbf{v}}_i^N)$ berechnet bzw. gesampelt, die schwach gegen $f_{j+1}(\mathbf{v})d\mathbf{v}$ konvergiert. Dazu wird eine Folge $(a_i)_{1\le i\le N}$ von N gleichverteilten Zufallszahlen in $[0, T)$ gesampelt, die die Kollisionspartner $\lfloor Na_i \rfloor + 1$ der einzelnen Teilchen darstellen, sowie eine Folge $(z_i)_{1\le i\le N}$ von N gleichverteilten Zufallszahlen in B^1, welche als Parameter für die Kollisionen dienen. Damit werden nun die neuen Geschwindigkeiten

$$S^N(1)(\mathbf{v}_1, \ldots \mathbf{v}_N) := \left(\psi(z_i, \mathbf{v}_i^N, \mathbf{v}_{\lfloor Na_i\rfloor+1}^N)\right)_{1\le i\le N} \tag{4.20}$$

gemäß Lemma 4.1 gesampelt.

Unter folgenden Voraussetzungen zeigten BABOVSKY und ILLNER[BI89], dass das DSMC-N Schema konvergiert:

(DSMC-N 1.) Der Übergangskern σ ist beschränkt: $\int \sigma(\vartheta, |\mathbf{v} - \mathbf{w}|)d\Omega \le C_1$.

(DSMC-N 2.) Für alle Intervalle $[0, T]$ gibt es Zahlen $\alpha_2 \ge \alpha_1 > 0$, sodass $\|f_0\|_{\alpha_1} < \infty$ und die Boltzmann-Gleichung eine eindeutige milde Lösung $f(\cdot, t)$ auf $[0, T]$ mit $\sup_{t\in[0,T]} \|f(\cdot, t)\|_{\alpha_1}$ besitzt.

(DSMC-N 3.) Es existieren zwei Konstanten $\alpha_3 > \alpha_1$ und $B > 0$, sodass

$$\operatorname*{ess\,sup}_{\mathbf{r},\mathbf{v},t} |f(\mathbf{r} + \Delta\mathbf{r}, \mathbf{v}, t) - f(\mathbf{r}, \mathbf{v}, t)|\, e^{\alpha_3 v^2} \le B\,|\Delta\mathbf{r}|$$

für alle $\Delta\mathbf{r}$.

(DSMC-N 4.) Die Funktion $\psi(\mathbf{r}, \mathbf{v}, \mathbf{w})$ sei fast überall stetig.

Satz 4.2 (Konvergenz des DSMC-N Schemas, Babovsky, Illner [BI89], o.B.). *Es seien* $\mu_j^N = \frac{1}{N}\sum_{i=1}^N \delta(\mathbf{r}_i^j, \mathbf{v}_i^j)$ *die durch das DSMC-N Schema erhaltenen Maße. Weiters gelte* $\mu_0^N \to_w f_0(\cdot)\, d\mathbf{r}\, d\mathbf{v}$, *also* $D(\mu_0^N, f_0 d\mathbf{r}\, d\mathbf{v}) \xrightarrow{N\to\infty} 0$, *sowie die Bedingungen (DSMC-N 1) bis (DSMC-N 4). Dann existieren für alle Folgen von Zellgrößen* $\Delta r_n \to 0$ *und Zeitschritten* $\Delta t_n \to 0$ *Teilchenzahlen* $N(v) \to \infty$, *sodass*

$$\lim_{n\to\infty} D\left(\mu_j^{N(n)}, f(\mathbf{r}, \mathbf{v}, j\Delta t_n)\right) d\mathbf{r}d\mathbf{v} = 0 \tag{4.21}$$

fast sicher bezüglich der Zufallszahlen a_n *und* z_n *für alle* $j = 1, \ldots \lfloor T/\Delta t_n \rfloor + 1$ *gilt.*

Analog zu obiger Form kann das DSMC-N Schema auch als LD-Schema (vgl. Lécot's LD Methode im nächsten Abschnitt. Dort werden lediglich Folgen kleiner Diskrepanz anstatt wie hier Pseudo-Zufallszahlen benutzt.) formuliert werden:

Ausgehend von der schwachen Gleichung 3.35

$$\frac{\partial}{\partial t}\int_{\mathbb{R}_+} \varphi(v) v^2 f(v,t) dv = \frac{n_c}{(4\pi)^2}\int_{\mathbb{R}^6\times S^2} \left(\varphi(|\mathbf{v}|) - \varphi(|\mathbf{v}|)\right) |\mathbf{v}-\mathbf{w}| \\ \sigma(\vartheta, |\mathbf{v}-\mathbf{w}|) f(|\mathbf{v}|, t) f(|\mathbf{w}|, t) d\mathbf{v} d\mathbf{w} d\nu,\ \varphi \in B(\mathbb{R}_+), t \in \mathbb{R}_+ \tag{4.22}$$

wird einerseits der Euler-Schritt für die Zeitableitung durchgeführt, und andererseits für f eine Summe von δ-Maßen eingesetzt, welche letztendlich f approximieren sollen (hier für den vereinfachten Fall eines konstanten Wirkungsquerschnitts $g\sigma(g) = k = \frac{q}{4\pi\Delta t}$):

$$f^{(n+1)} = \frac{1}{N}\sum_{i=1}^N \delta\left(v - v_i^{(n+1)}\right) \tag{4.23}$$

$$\int_{\mathbb{R}_+} \varphi(v) g^{(n)}(dv) = \frac{1}{N}\left((1-q)\sum_{i=1}^N \varphi(v_i^{(n-1)}) + \frac{q}{N}\sum_{i,j=1}^N \int_{I^2} \varphi(\left[a, b, ; v_j^{(n-1)}, v_i^{(n-1)}\right]) da db\right) \tag{4.24}$$

In einem letzten Schritt wird daraus nun der Operator K aus Gleichung (4.31) definiert, gemäß dem schließlich die Teilchen bewegt werden:

$$K^{(n)}\varphi(x_1, x_2, x_3, x_4) = \sum_{i=1}^N \varphi(v_i^{(n+1)})\chi_i(x_3, x_4) + \\ \sum_{i=1}^N\sum_{j=1}^N \varphi\left(\left[v_j^{(n-1)}, v_i^{(n-1)}; x_1, x_2\right]\right)\chi_{j,i}(x_3, x_4) \tag{4.25}$$

Speziell läuft die Simulation durch das DSMC-N Schema mit folgenden Schritten ab:

(DSMC-N 1.) Die alten Geschwindigkeiten $\mathbf{v}_j^N$ werden zwischengespeichert für die folgenden Schritte.

(DSMC-N 2.) In jedem Zeitschritt wird jedem Teilchen eine Bernoulli-verteilte Zufallszahl zugeordnet, die bestimmt, ob dieses Teilchen mit einem anderen kollidiert:

$$P[X_i = 1] = P_i \tag{4.26}$$

$$P[X_i = 0] = 1 - P_i \tag{4.27}$$

$$P_i = \sum_{j=1}^{N}{}' \frac{n_c \Delta t}{N} g_{ij} \sigma_T(g_{ij}) \tag{4.28}$$

g_{ij} sind dabei die relativen Geschwindigkeiten zum Zeitpunkt t, berücksichtigen alle im Gegensatz zum BY-Schema keine anderen Kollisionen in diesem Zeitschritt.

(DSMC-N 3.) Nach Gleichung (4.28) wird der Kollisionspartner j bestimmt, falls $X_i = 1$. Ansonsten wird das nächste Teilchen i betrachtet.

(DSMC-N 4.) Die Kollision der Teilchen wird nach Abschnitt 3.1 simuliert, wobei α aufgrund der Isotropie gleichverteilt auf der Halbkugel S_+ gesampelt wird. Mit der hier benutzten schwachen Boltzmann-Gleichung (3.35) ergibt dies einfach:

$$\mathbf{v}_i' = [\mathbf{v}_i, \mathbf{v}_j; a, b] \tag{4.29}$$

mit auf I gleichverteilten Zufallszahlen a und b.

Im diesem Schritt wird (gemäß Gleichung (4.20)) die Geschwindigkeit $\mathbf{v}_j$ nicht verändert, die Gleichung also als rein mathematisches Übergangsgesetz ohne den physikalischen Sachverhalt der Teilchenkollisionen gesehen, bei denen ja beide beteiligten Teilchen ihre Geschwindigkeiten wegen der Energie- und Impulserhaltung ändern.

(DSMC-N 5.) Die Schritte 2 bis 4 werden nun für jedes Teilchen i wiederholt.

Im Gegensatz zu den anderen Simulationsschemata wird hier nicht die Geschwindigkeit des Kollisionspartners geändert, wodurch Energie- und Impulserhaltung nicht mehr exakt gelten. Im Mittel allerdings werden diese nach wie vor erfüllt, wie NANBU bereits in [Nan80] zeigte.

Zur Bestimmung in Punkt 2, ob ein Teilchen i kollidiert, müssen alle $g_{ij}\sigma_t(g_{ij})$ berechnet werden, was einen Aufwand der Ordnung N^2 mit sich bringt. BABOVSKY [Bab86] zeigte, dass dieser Aufwand zu N reduziert werden kann, indem die benutzten Intervalle auf $I = [0, 1]$ geeignet umgeordnet werden. Anstatt alle Intervalle der Länge P_{ij} als $\left[\sum_{j'=1}^{j-1} P_{ij'}, \sum_{j'=1}^{j} P_{ij'}\right)$ von 0 beginnend sequentiell anzuordnen, schlägt BABOVSKY Intervalle der Form

$$\left[\frac{j}{N} - P_{ij}, \frac{j}{N}\right) \tag{4.30}$$

oder gleichwertig $\left[\frac{j}{N}, \frac{j}{N} + P_{ij}\right)$ vor. Da die Zufallsvariablen a_i auf I gleichverteilt sind, ist die Anordnung der Intervalle gleichgültig, solange Überlappungen ausgeschlossen sind. Letzteres ist unter der Voraussetzung $\frac{n\Delta t}{N} g_{ij}\sigma(g_{ij}) < \frac{1}{N}$ für alle $j \leq N$, also sicher für hinreichend kleine Δt, der Fall. Dieses Schema wird DSMC-N* Schema (oder SS* Schema) genannt und hat dieselben Eigenschaften und Ergebnisse wie Nanbu's DSMC-N Schema.

4.7 LÉCOT's LD Methode

Die von LÉCOT in [Léc89a] vorgeschlagene LD Methode stellt lediglich eine Quasi-Monte Carlo Variante von NANBU's DSMC-N Schema dar. Er betrachtet allerdings nur den Fall, in dem der Wirkungsquerschnitt die Form $\sigma(\vartheta, |\mathbf{v}-\mathbf{w}|) = \frac{k}{g}$ mit $k = \frac{3}{2\pi}$ beträgt und definiert $q = 4k\pi\Delta t < 1$. Außerdem sei $n_c = 1$. Unter den Voraussetzungen von räumlicher Homogenität (kann durch Einführung der beim DSMC-N Schema diskutierten Zellen immer erreicht werden), der Isotropie und Verwendung von Maxwell-Molekülen ersetzt LÉCOT die 3 benutzten Zufallszahlen pro Teilchen i durch eine 4-dimensionale Folge Z_n kleiner Diskrepanz. Die zusätzliche Komponente wird benutzt, um die Teilchen i nicht sequentiell, sondern in (quasi-)zufälliger Reihenfolge zu durchlaufen. Dies benötigt natürlich noch eine zusätzliche Bedingung an Z_n:

(LD 1.) Jedes Intervall $I^3 \times \left[\frac{i-1}{N}, \frac{i}{N}\right)$, $1 \le i \le N$ enthält genau ein Element aus Z_n.

Die Simulation wird wieder gemäß dem Operator

$$K^{(n)}\varphi(x_1,x_2,x_3,x_4) = \sum_{i=1}^{N} \varphi(v_i^{(n+1)})\chi_i(x_3,x_4) + \sum_{i=1}^{N}\sum_{j=1}^{N} \varphi\left(\left[v_j^{(n-1)}, v_i^{(n-1)}; x_1, x_2\right]\right)\chi_{j,i}(x_3,x_4) \quad (4.31)$$

durchgeführt. Hier stellt $\chi_{i,j}$ im Gegensatz zu Abschnitt 4.2 die charakteristische Funktion von $\left[q\frac{j-1}{N}, q\frac{j}{N}\right) \times \left[\frac{i-1}{N}, \frac{i}{N}\right)$ und χ_i die charakteristische Funktion von $[q,1) \times \left[\frac{i-1}{N}, \frac{i}{N}\right)$ dar.

Um etwaige Korrelationen zu brechen und auch, um für $N \to \infty$ eine stetige Funktion $f^{(n)}$ zu erhalten, wird am Ende jedes Schrittes die Menge der Teilchengeschwindigkeiten $\mathbf{V}^{(n)} = \left\{\mathbf{v}_i^N \in \mathbb{R} : 1 \le i \le N\right\}$ aufsteigend umgeordnet. Dies ist der wichtigste Schritt in einer Quasi-Monte Carlo Simulation der Boltzmann-Gleichung.

LÉCOT zeigte auch eine Diskrepanzschranke für die so erhaltenen Lösungen der Boltzmann-Gleichung:

Satz 4.3 (LÉCOT, [Léc89a]). *Die Lösung $f(\mathbf{r},\mathbf{v},t)$ der Boltzmann-Gleichung sei zweimal stetig differenzierbar bezüglich t, und es sei $v^2\frac{\partial^2 f}{\partial t^2}(v,t) \in L^1(\mathbb{R}_+ \times (0,T))$. Wenn die Diskrepanzen der für die einzelnen Zeitschritte benutzten Punktmengen $\mathbf{Z^n}$ durch ein D_N beschränkt sind, so gilt für $1 \le n \le M$:*

$$D_N^*(\mathbf{V}^{(n)}, f) \le e^{12k\pi t^{(n)}} D_N^*(\mathbf{V}^{(0)}, f) + \Delta t \int_{\left[0,t^{(n)}\right]\times\mathbb{R}_+} e^{12k\pi(t^{(n)}-t)} \left|\frac{\partial^2 f}{\partial t^2}(\mathbf{v},t)\right| v^2 dv\, dt + \frac{13}{3k\pi} e^{12k\pi t^{(n)}} \frac{D_N^{\frac{1}{4}}}{\Delta t} \quad (4.32)$$

Lécot's Experimente zeigten allerdings, dass der Fehler trotz des Δt im Nenner nicht gegen ∞ geht. Falls für alle Zeitschritte dieselben Punktmengen benutzt werden, zeigte er eine weitere Abschätzung:

Satz 4.4 (LÉCOT, [Léc89a], o.B.). *Wenn für alle Zeitschritte dieselbe Punktmenge*

$$\mathbf{Z}^{(n)} = \mathbf{Z}^* = \{(a_l, b_l, c_l, d_l) : 1 \le l \le N\}$$

benutzt wird und $\Delta t \leq \frac{1}{4k\pi} \min\{c_l : 1 \leq l \leq N, c_l > 0\}$ *gilt, dann ist*

$$D_N^*(\mathbf{V}^{(n)}, f) \leq c^{12k\pi t^{(n)}} D_N^*(\mathbf{V}^{(0)}, t) + \Delta t \int_{[0,t^{(n)}]\times\mathbb{R}_+} e^{12k\pi(t^{(n)}-t)} \left|\frac{\partial^2 f}{\partial t^2}(v,t)\right| v^2 dv\ dt + \frac{1}{3} e^{12k\pi t^{(n)}} \quad (4.33)$$

4.7.1 Beweis von Satz 4.3

Um die Diskrepanz

$$D_N^*(\mathbf{V}^{(n)}, f) = \sup_{r>0} \left|d_N^{(n)}\right| \quad (4.34)$$

$$d_N^{(n)}(r) = \frac{1}{N} \sum_{1\leq i\leq N} \chi_{[0,r)}\left(v_i^{(n)}\right) - \int_{\mathbb{R}_+} \chi_{[0,r)}(v) v^2 f(v, t^{(n)}) dv \quad (4.35)$$

abzuschätzen, werden folgende Terme benötigt:

$$e_N^{(n)}(r) = \int_{\mathbb{R}_+^2\times I^2} \left[\chi_{[0,r)}(v') - \chi_{[0,r)}(v)\right] f^{(n)}(dv) f^{(n)}(dw) da\ db - \int_{\mathbb{R}_+^2\times I^2} \left[\chi_{[0,r)}(v') - \chi_{[0,r)}(v)\right] v^2 f(v, n\Delta t) w^2 f(w, \Delta t) dv\ dw\ da\ db \quad (4.36)$$

$$\varepsilon^{(n)}(r) = \int_{\left[t^{(n)}, t^{(n+1)}\right]\times\mathbb{R}_+} \chi_{[0,r)}(v) \left(\frac{\partial f}{\partial t}(v, t^{(n)}) - \frac{\partial f}{\partial t}(v,t)\right) v^2 dv\ dt \quad (4.37)$$

$$\delta_N^{(n)}(r) = \frac{1}{N} \sum_{1\leq l\leq N} K^{(n+1)} \chi_{[0,r)}(z_{nN+l}) - \int_{I^n} K^{(n+1)} \chi_{[0,r)}(z) dz \quad (4.38)$$

$e_N^{(n)}(r)$ bezeichnet her den Fehler, der dadurch entsteht, dass $v^2 f(v,t)$ durch die diskrete Summe $\frac{1}{N}\sum_{i=1}^N \delta(v - v_i^{(n)})$ ersetzt wird. $\varepsilon^{(n)}(r)$ wird benötigt, um den Fehler zu bezeichnen, der durch den Euler-Vorwärtschritt in der Zeitableitung entsteht, und $\delta_N^{(n)}(r)$ gibt den Fehler durch die Näherung des Integrals über $K^{(n+1)}$ an.

Es gilt damit

$$d_N^{(n)}(r) = d_N^{(n-1)}(r) + q e_N^{(n-1)}(r) + \varepsilon^{(n-1)}(r) + \delta_N^{(n-1)}(r) \quad (4.39)$$

Lemma 4.5. *Es gilt* $\left|e_N^{(n)}(r)\right| \leq 3 D_N^*(\mathbf{V}^{(n)}, f)$.

Beweis. Gleichung (4.36) lässt sich aufspalten in drei Terme

$$e_N^{(n)}(r) = \frac{1}{N} \sum_{j=1}^N \int_{I^2} d_N(x_1, x_2; v_j^{(n)}) dx_1 dx_2 + \int_{R_+^2\times I^2} d_N(x_1, x_2; v) v^2 f(v, t^{(n)}) dv\ da\ db - d_N^{(n)}(r)$$

mit

$$d_N(x_1,x_2;v)=\frac{1}{N}\sum_{j=1}^{N}\varphi_r\left(\left[x_1,x_2;v_j^{(n)},v\right]\right)-\int_{R_+}\varphi_r\left([x_1,x_2;w,v]\right)w^2f(w,t^{(n)})dw$$

Die Abbildung $w\mapsto\chi_{[0,r)}\left([w,v;a,b]\right)$ kann außerdem verstanden werden als eine charakteristische Funktion $w\mapsto\chi_{[0,s(a,b,v))}(w)$ mit $s(a,b,v)\in\bar{\mathbb{R}}_+$. Damit können alle drei Terme unter Verwendung von $\int v^2f(v,t)dv=1$ und $\int da\;db=1$ durch das Supremum über alle $[0,r)$, also durch $D_N^*({}^{(n)}V,f)$ abgeschätzt werden:

$$\left|e_N^{(n)}(r)\right|\le\frac{1}{N}\sum\left|\int D_N^*da\;db\right|+\left|\int D_N^*v^2f(v,t^{(n)})dv\;da\;db\right|+D_N^*=3D_N^*$$

□

Weiters gilt durch partielle Integration über t:

$$\left|\varepsilon^{(n)}(r)\right|\le\Delta t\int_{\left[t^{(n)},t^{(n+1)}\right]\times\mathbb{R}_+}\left|\frac{\partial^2f}{\partial t^2}(v,t)\right|v^2dv\;dt \tag{4.40}$$

Zur Abschätzung der $\delta_N^{(n)}(r)$ gibt Lécot in [Léc89a] folgende Schranke an:

Lemma 4.6 (o.B.).

$$\left|\delta_N^{(n)}(r)\right|\le 52D_N\left(\mathbf{Z}^{(n+1)}\right)^{1/4} \tag{4.41}$$

Aus Gleichung (4.39) ergibt sich durch Summation:

$$d_N^{(n)}-d_N^{(0)}=\sum_{i=0}^{n-1}\left(d_N^{(i+1)}-d_N^{(i)}\right)=\sum_{i=0}^{n-1}\left(qe_N^{(i)}+\varepsilon^{(i)}+\delta_N^{(i)}\right) \tag{4.42}$$

und damit

$$D_N^*(\mathbf{V}^{(n)},f)=\sup_{r>0}\left|d_N^{(n)}(r)\right|\le D_N^*(\mathbf{V}^{(0)},f)+3q\sum_{i=0}^{n-1}D_N^*\left(\mathbf{V}^{(i)},f\right)+$$
$$\Delta t\int_{\left[0,t^{(n)}\right]\times\mathbb{R}_+}\left|\frac{\partial^2f}{\partial t^2}(v,t)\right|v^2dv\;dt+\sum_{i=0}^{n-1}52D_N\left(\mathbf{V}^{(i)}\right)^{1/4} \tag{4.43}$$

Setzt man nun diese Ungleichung rekursiv in sich selbst ein, und benutzt

$$(1+3q)^n=(1+12\pi k\Delta t)^n=\left(1+\frac{12\pi kt^{(n)}}{n}\right)^n\le e^{12\pi kt^{(n)}}$$

so ergibt sich:

$$D_N^*(\mathbf{V}^{(n)},f)\le e^{12k\pi t^{(n)}}D_N^*(\mathbf{V}^{(0)},f)+$$
$$\Delta t\int_{\left[0,t^{(n)}\right]\times\mathbb{R}_+}e^{12k\pi\left(t^{(n)}-t\right)}\left|\frac{\partial^2f}{\partial t^2}(v,t)\right|v^2dv\;dt+\frac{n52D_N^{1/4}}{12k\pi t^{(n)}}e^{12k\pi t^{(n)}} \tag{4.44}$$

□

4.8 LÉCOT's QMC1 Schema

Im Gegensatz zum LD Schema, welches auf NANBU's DSMC-N Schema beruht, führte LÉCOT [Léc91] ein weiteres Schema ein, das zwar nach wie vor auf NANBU's Zugang beruht, aber in einigen Teilen auch an BIRD's DSMC-B Methode erinnert. Ich möchte es im Folgenden QMC1 Schema nennen.

Im Gegensatz zur LD Methode wird bei der QMC1 Methode ein allgemeiner Wirkungsquerschnitt $\sigma(|\mathbf{v}-\mathbf{w}|)$ betrachtet, das Gas sei aber nach wie vor homogen (in der Zelle, also keine große Einschränkung) und die Streuung erfolge isotrop.

Wenn $f(v,t)$ eine Lösung der Boltzmann-Gleichung ist, so erfüllt sie auch die schwache Gleichung

$$\frac{\partial}{\partial t}\int_{\mathbb{R}_+}\varphi(v)v^2 f(v,t)dv = \frac{n_C c}{(4\pi)^2}\int_{\mathbb{R}^6\times S^2}\left(\varphi(|\mathbf{v}'|)-\varphi(|\mathbf{v}|)\right)|\mathbf{v}-\mathbf{w}| \\ \sigma(|\mathbf{v}-\mathbf{w}|)f(|\mathbf{v}|,t)f(|\mathbf{w}|,t)d\mathbf{v}d\mathbf{w}d\nu \quad (4.45)$$

mit

$$\mathbf{v}' = \frac{1}{2}\left(\mathbf{v}+\mathbf{w}+|\mathbf{v}-\mathbf{w}|\,\nu\right) \quad (4.46)$$

$$\chi = \pi - 2\vartheta \qquad \text{der Rückstoßwinkel} \quad (4.47)$$

$$\nu = \frac{1}{|\mathbf{v}-\mathbf{w}|}\left(\mathbf{v}-\mathbf{w}-2\mathbf{n}\cdot(\mathbf{v}-\mathbf{w})\,\mathbf{n}\right) = \frac{\mathbf{v}'-\mathbf{w}'}{|\mathbf{v}'-\mathbf{w}'|} \quad (4.48)$$

Der totale Wirkungsquerschnitt σ_T der isotropen Streuung

wird nach [Bir94] am besten durch das "variable hard sphere" Modell beschrieben:

$$\sigma_t(g) = \left(\frac{4(\eta-2)}{\eta-1}\frac{kT_{ref}}{m_r}\right)^{\frac{2}{\eta-1}} g^{-\frac{4}{\eta-1}}\sigma_{ref} \quad (4.49)$$

wobei $5 \le \eta \le \infty$ einen Parameter darstellt, der die Härte der Streuung beschreibt ($\eta = 5$ bedeutet Maxwell-Moleküle, $\eta = \infty$ beschreibt das Hard Sphere model). σ_{ref} ist dabei außerdem ein bekannter Wert des Wirkungsquerschnitts zur Temperatur T_{ref}, und $g \in \mathbb{R}_+$ ist wie üblich $|\mathbf{v}-\mathbf{w}|$. Damit wird nun eine neue (positiv und monoton geforderte) Funktion

$$q(g) = n_c g\sigma(g) \quad (4.50)$$

definiert, sowie $Q := \sup_{g\in\mathbb{R}_+} q(g) < \infty$. Um letzte Bedingung zu erfüllen, ist es unter Umständen (etwa im hard sphere model) nötig, den Wirkungsquerschnitt "abzuschneiden". In dieser Schreibweise betrachtete die LD Methode den Spezialfall $q(g) = k$.

Analog zu Abschnitt 3.7 ergibt sich nun durch die Variablentransformationen

$$v = |\mathbf{v}| \qquad w = |\mathbf{w}| \quad (4.51)$$

$$x^{(4)} = \frac{1}{2}\left(1+\frac{\mathbf{v}\cdot\mathbf{w}}{|\mathbf{v}|\,|\mathbf{w}|}\right) \qquad x^{(5)} = \frac{1}{2}\left(1+\frac{(\mathbf{v}+\mathbf{w})\cdot\nu}{|\mathbf{v}+\mathbf{w}|}\right) \quad (4.52)$$

die schwache Gleichung (mit $v' = [v,w;x^{(4)},x^{(5)}]$ und $g = [v,w;x^{(4)}]$)

$$\frac{d}{dt}\int_{\mathbb{R}_+}\varphi(v)v^2 f(v,t)dv = \int_{\mathbb{R}_+^2\times I^2}\left(\varphi(v')-\varphi(v)\right)q(g) \\ v^2 f(v,t)w^2 f(w,t)dv\,dw\,dx^{(4)}\,dx^{(5)} \quad (4.53)$$

Da isotrope Streuung betrachtet und die Bewegung der Teilchen völlig vernachlässigt wird, wird für jedes Teilchen nur der Absolutbetrag der Geschwindigkeit gespeichert, und dann g_{ij} nach dem Cosinussatz und der gleichverteilten Zufallsvariable $x^{(4)}$ durch $[v, w; x^{(4)}]$ gesampelt.

Unter der Voraussetzung $Q\Delta t < 1$ soll nun mittels einer Folge $\left(\mathbf{x}^{(n)}\right)_{n\in\mathbb{N}} \subset I^5$ die Boltzmann-Gleichung simuliert werden. Als erster Schritt wird die Zeitableitung auf der linken Seite der Gleichung behandelt, indem durch einen einfachen Euler-Schritt mit Zeitordnung $O(\Delta t)$ ein neues Maß $g^{(n)}$ auf $\mathbb{R}_+$ definiert wird durch:

$$\int_{\mathbb{R}_+} \varphi(v) g^{(n)}(dv) = \int_{\mathbb{R}_+} \varphi(v) f^{(n)}(dv) + \Delta t \int_{\mathbb{R}_+^2 \times I^2} \left(\varphi(v') - \varphi(v)\right) q(g) f^{(n)}(dv) f^{(n)}(dw) dx^{(4)} dx^{(5)} \quad (4.54)$$

Dieses $g^{(n)}$ approximiert offensichtlich in einem gewissen Sinn die neue Dichte. In obige Definition von $g^{(n)}$ kann äquivalent $f^{(n)}(v) = \frac{1}{N}\sum_{i=1}^{N} \delta(v - v_i^{(n)})$ eingesetzt werden:

$$\int_{\mathbb{R}_+} \varphi(v) g^{(n)}(dv) = \frac{1}{N}\sum_{i=1}^{N} \left(1 - \frac{\Delta t}{N}\sum_{j=1}^{N} \int_I q\left([v_i^{(n)}, v_j^{(n)}; x^{(4)}]\right) dx^{(4)}\right) \varphi(v_i^{(n)}) + \frac{\Delta t}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} \int_{I^2} q\left([v_i^{(n)}, v_j^{(n)}; x^{(4)}]\right) \varphi\left([v_i^{(n)}, v_j^{(n)}; x^{(4)}, x^{(5)}]\right) dx^{(4)} dx^{(5)} \quad (4.55)$$

Ausgehend von dieser Gleichung wird der Operator

$$L^{(n)}\varphi(\mathbf{x}) = \sum_{i=1}^{N}\sum_{j=1}^{N} \chi_{i,j}(x^{(1)}, x^{(2)}) \Big((1 - c_{i,j}^{(n)}(x^{(3)}, x^{(4)}))\varphi(v_i^{(n)}) + c_{i,j}^{(n)}(x^{(3)}, x^{(4)})\varphi([v_i^{(n)}, v_j^{(n)}; x^{(4)}, x^{(5)}])\Big) \quad (4.56)$$

für $\mathbf{x} \in I^5$ definiert, wobei $c_{i,j}^{(n)}$ die charakteristische Funktion von

$$\left\{\left(x^{(3)}, x^{(4)}\right) : x^{(3)} < \Delta t q\left([v_i^{(n)}, v_j^{(n)}; x^{(4)}]\right)\right\}$$

darstellt. Da $\int_{I^5} L^{(n)}\varphi(\mathbf{x}) d\mathbf{x} = \int_{\mathbb{R}_+} \varphi(v) g^{(n)}(dv)$ gilt, wird $f^{(n+1)}$ definiert durch

$$\int_{R_+} \varphi(v) f^{(n+1)}(dv) = \frac{1}{N}\sum_{l=1}^{N} L^{(n)}\varphi(\mathbf{x}_{nN+l}) \quad (4.57)$$

Die Simulation wird demnach entsprechend dem Operator (4.56) durchgeführt, der interpretiert wird, wie schon in Abschnitt 1.1 angedeutet:

Für jede Kollision werden $x^{(1)}$ und $x^{(2)}$ benutzt, um die beiden kollidierenden Teilchen i und j zu bestimmen:

$$i(l) = \left\lfloor N x_{nN+l}^{(1)} \right\rfloor \qquad j(l) = \left\lfloor N x_{nN+l}^{(2)} \right\rfloor \quad (4.58)$$

Die dritte Zufallsvariable $x^{(3)}$ bestimmt gemäß

$$x_{nN+l}^{(3)} < \Delta t q\left([v_i(l)^{(n)}, v_j(l)^{(n)}; x_{nN+l}^{(4)}]\right) \quad (4.59)$$

ob die beiden Teilchen tatsächlich kollidieren, und durch $x^{(4)}_{nN+l}$ und $x^{(5)}_{nN+l}$ werden die Geschwindigkeiten der Teilchen nach der Kollision bestimmt.

$$v^{(n+1)}_{i(l)} = \left[v^{(n)}_{i(l)}, v^{(n)}_{j(l)}; x^{(4)}_{nN+l}, x^{(4)}_{nN+l}\right] \qquad v^{(n+1)}_{j(l)} = \left[v^{(n)}_{j(l)}, v^{(n)}_{i(l)}; x^{(4)}_{nN+l}, 1 - x^{(4)}_{nN+l}\right] \tag{4.60}$$

Da in diesem Schema in jedem einzelnen Schritt zwei Geschwindigkeiten geändert werden, brauchen in jedem Zeitschritt nur halb so viele Kollisionen wie beim LD Schema betrachtet werden. Um allerdings zu gewährleisten, dass sowohl das i-te mit dem j-ten Teilchen, als auch umgekehrt, kollidieren würden, werden an die Folge $\mathbf{X}^{(n)}$ noch einige Forderungen gestellt:

(QMC1 1.) $i(l)$ und $j(l)$ sind Bijektionen von $\mathbf{N} = \{0, \cdot s, N-1\}$ auf $\mathbf{N}$

(QMC1 2.) Mit $\mathbf{x} = (x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}, x^{(5)}) \in \mathbf{X}^{(n)}$ soll auch gelten, dass

$$\tilde{\mathbf{x}} = (x^{(2)}, x^{(1)}, x^{(3)}, x^{(4)}, 1 - x^{(5)}) \in \mathbf{X}^{(n)}$$

Die erste Forderung stellt sicher, dass jedes Teilchen nur einmal betrachtet wird (einmal als Teilchen i und einmal als Teilchen j), während die zweite Bedingung die Symmetrie gewährleistet, sodass es tatsächlich genügt, nur $\frac{N}{2}$ Kollisionen zu betrachten. Damit kommt jedes Teilchen auch nur einmal als i oder j vor. Aus der zweiten Forderung folgt auch sofort, dass die Energie bei jeder Kollision erhalten bleibt.

(QMC1 3.) Am Ende jedes Zeitschrittes werden schließlich noch alle Geschwindigkeiten der Größe nach geordnet.

Die so erhaltenen Geschwindigkeiten in jedem Zeitschritt $\mathbf{V}^{(n)}$ konvergieren nun gegen die die tatsächliche Verteilung von $v^2 f(v, ({}^{(n)}t))$:

Satz 4.7 (Lécot [Léc91]). *Sei* $\mathbf{S} \subset \mathbf{N}^2$, *sowie*

$$\lim_{N\to\infty} D^*_N(V^{(0)}, v^2 f_0) = 0 \tag{4.61}$$

$$\mathbf{X}^+_{nN} = \left\{ \left(\mathbf{x}_p, \frac{p-1}{nN} \right) : 1 \le p \le nN \right\} \subset I^6 \tag{4.62}$$

$$\lim_{\substack{(N,M)\in\mathbf{S} \\ N,M\to\infty}} \max_{1\le n\le M} nD_{nN}(\mathbf{X}^+_{nN})^{1/6} = 0 \tag{4.63}$$

Dann konvergiert $\mathbf{V}^{(n)}$ *gegen* $v^2 f_n$*:*

$$\lim_{\substack{(N,M)\in\mathbf{S} \\ N,M\to\infty}} \max_{1\le n\le M} D_N(\mathbf{V}^{(n)}, v^2 f_n) = 0 \tag{4.64}$$

4.8.1 Die benutzte $(0,5)$-Folge

Die Forderungen (QMC1 1) und (QMC1 2) sind natürlich nicht automatisch von einem $(0,5)$-Netz erfüllt.

Daher konstruierte Lécot aus einer $(0,5)$-Folge eine neue Folge, die die geforderten Bedingungen erfüllt:

Definition 4.3 (Lécot). Es seien $N = 2p^\lambda$ die Anzahl der simulierten Teilchen (der in jedem Zeitschritt benötigten quasi-Zufallszahlen), $\nu = p^\lambda$, sowie $\mathbf{Y}$ eine $(0,5)$-Folge in Basis p. Dann wird eine neue Folge $\mathbf{X}$ folgendermaßen definiert:

Für $2m\nu \le q < (2m+1)\nu$ sei (mit der Bezeichnung $q^* = q - m\nu$)

$$\mathbf{x}_q = \left(\frac{1}{2}\left(y_{q^*}^{(1)} + \frac{\left\lfloor \nu y_{q^*}^{(1)} \right\rfloor + 1}{\nu} \right), \frac{1}{2}\left(y_{q^*}^{(2)} + \frac{\left\lfloor \nu y_{q^*}^{(2)} \right\rfloor}{\nu} \right), y_{q^*}^{(3)}, y_{q^*}^{(4)}, \frac{1}{2}\left(y_{q^*}^{(5)} + \frac{\left\lfloor \nu y_{q^*}^{(5)} \right\rfloor}{\nu} \right) \right)$$

und für $(2m+1)\nu \le q < 2(m+1)\nu$ sei (mit der Bezeichnung $q^* = q - (m+1)\nu$)

$$\mathbf{x}_q = \left(\frac{1}{2}\left(y_{q^*}^{(2)} + \frac{\left\lfloor \nu y_{q^*}^{(2)} \right\rfloor}{\nu} \right), \frac{1}{2}\left(y_{q^*}^{(1)} + \frac{\left\lfloor \nu y_{q^*}^{(1)} \right\rfloor + 1}{\nu} \right), y_{q^*}^{(3)}, y_{q^*}^{(4)}, 1 - \frac{1}{2}\left(y_{q^*}^{(5)} + \frac{\left\lfloor \nu y_{q^*}^{(5)} \right\rfloor}{\nu} \right) \right)$$

Lécot zeigte für diese Folge die folgende Diskrepanzschranke:

Lemma 4.8 (Lécot [Léc91], o.B.). *Die Sterndiskrepanz $D_{nN}^*(\mathbf{X})$ erfüllt*

$$D_{nN}^*(\mathbf{X}) \le \frac{3}{2} D_{n\nu}^*(\mathbf{Y}) + \frac{2}{N^2} \qquad \text{für } n \ge 1$$

Die Folge $\mathbf{X}$ erfüllt einerseits obige Bedingungen, andererseits befinden sich keine Punkte von $\mathbf{X}$ im Intervall $\left[\frac{i-1}{N}, \frac{i}{N}\right) \times I^3$, $1 \le i \le N$. Das bedeutet, dass kein Teilchen mit sich selbst kollidieren kann.

4.8.2 Beweis des Konvergenzsatzes 4.7

Der Beweis von Satz 4.7 läuft im Prinzip analog zu Abschnitt 4.7.1. $d_N^{(n)}(r)$, $e_N^{(n)}(r)$, $\varepsilon^{(n)}(r)$ und $\delta_N^{(n)}(r)$ seien wie in Abschnitt 4.7.1 definiert (nur das Integral in $\delta_N^{(n)}$ läuft über I^5).

Mit der Definition

$$\Delta_N^{(n)}(r) = \sum_{m=0}^{n-1} \delta_N^{(m)}(r) \tag{4.65}$$

folgt durch Summation aus Gleichung (4.39)

$$d_N^{(n)}(r) = d_N^{(0)}(r) + \Delta t \sum_{m=0}^{n-1} e_N^{(m)}(r) + \sum_{m=0}^{n-1} \varepsilon^{(m)}(r) + \Delta_N^{(n)}(r) \tag{4.66}$$

Der Term $\varepsilon^{(n)}(r)$ wird wieder durch Gleichung (4.40) abgeschätzt, und für $e_N^{(n)}(r)$ ergibt sich analog zu Lemma 4.5:

$$\left| e_N^{(n)}(r) \right| \le 11 Q D_N^*(\mathbf{V}^{(n)}, v^2 f_n) \tag{4.67}$$

Aus Gleichung (4.66) folgt mit den Abkürzungen

$$K = \Delta t 11Q \tag{4.68}$$

$$I_m = \int_{\left[t^{(n)},t^{(n+1)}\right)\times\mathbb{R}} v^2 \left|\frac{\partial^2 f}{\partial t^2}(v,t)\right| dv\ dt \tag{4.69}$$

$$D_N^{(m)} = D_N^*(\mathbf{V}^{(m)}, v^2 f_m) \tag{4.70}$$

$$G = \max_{1\le m\le n} \sup_{r>0} \left|\Delta_N^{(m)}(r)\right| \tag{4.71}$$

sofort über das Supremum und dann rekursiv in sich eingesetzt:

$$\begin{aligned}
D_N^*(\mathbf{V}^{(n)}, v^2 f_n) &\le D_N^{(0)} + K\sum_{m=0}^{n-1} D_N^{(m)} + \Delta t \sum_{m=0}^{n-1} I_m + G \\
&\le (1+K)^n D_N^{(0)} + (1+K)^{n-1} G + \Delta t \sum_{i=0}^{n-1} (1+K)^{n-1-i} I_i \\
&\le e^{10Qn\Delta t} D_N^{(0)} + e^{11Q((n-1)\Delta t)} G + \Delta t \sum_{i=0}^{n-1} e^{11Q(n\Delta t-(i+1)\Delta t)} I_i \\
&\le e^{10Qt^{(n)}} D_N^{(0)} + e^{11Qt^{(n-1)}} G + \\
&\qquad \Delta t \sum_{i=0}^{n-1} \int_{\left[t^{(i)},t^{(i+1)}\right)} e^{11Q\left(t^{(n)}-t^{(i+1)}\right)} \left|\frac{\partial^2 f}{\partial t^2}(v,t)\right| v^2 dv\ dt \\
&= e^{10Qt^{(n)}} D_N^{(0)} + e^{11Qt^{(n-1)}} G + \Delta t \int_{\left[0,t^{(n+1)}\right)} e^{11Q\left(t^{(n)}-t\right)} \left|\frac{\partial^2 f}{\partial t^2}(v,t)\right| v^2 dv\ dt
\end{aligned} \tag{4.72}$$

wobei $(1+K)^n \le e^{nK}$, sowie $t^{(i)} \le t \le t^{(i+1)}$ unter dem Integral benutzt wurden.

Der erste Term geht nach Voraussetzung für $N \to \infty$ und fixed n gegen 0. Ebenso geht im zweiten Term $\Delta t = \frac{T}{M}$, $T = const.$ mit $M \to \infty$ gegen 0.

Es bleibt somit nur noch G bzw. $\Delta_N^{(m)}(r)$ abzuschätzen.

Die Bedingung $\left[v, w; x^{(4)}, x^{(5)}\right] < r$, $x^{(4)}, x^{(5)} \in [0,1]$ lässt sich leicht umformen in

$$x^{(5)} < g_{v,w}^*\left(x^{(4)}\right) = \begin{cases} \min\left(\max\left(\frac{2r^2-v^2-w^2}{2\sqrt{(v^2+w^2)^2-4(2x^{(4)}-1)^2v^2w^2}} + \frac{1}{2}, 0\right), 1\right) & \\ \qquad \text{wenn } (v \ne w) \text{ oder } (v = w \ne 0, x^{(4)} \ne 0, 1) & \\ \varphi_r(v) \qquad \text{wenn } v = w \ne 0,\ x^{(4)} = 0, 1 & \\ 1 \qquad \text{für } v = w = 0 & \end{cases} \tag{4.73}$$

$g_{v,w}^*$ ist stetig und erfüllt $v < v', w < w' \Rightarrow g_{v,w}^* > g_{v',w'}^*$.

Da die $v_i^{(n)}$ immer geordnet werden, gilt $v_i^{(m)} < r \iff i \le \sum_{j=1}^N \varphi_r(v_j^{(n)})$. Es sei außerdem $\chi_m^{(n)}(r) = \chi_{\left[\frac{m-1}{n}, \frac{m}{n}\right)}(r)$ definiert sowie Mengen $\mathbf{X}_{nN}^+$, $\mathbf{E}_N^{(n)}(r)$, $\mathbf{F}_N^{(n)}(r)$ und $\mathbf{G}_N^{(n)}(r)$:

$$\mathbf{X}_{nN}^+ = \left\{\left(\mathbf{x}_p, \frac{p-1}{nN}\right) : 1 \le p \le nN\right\} \subset I^6 \tag{4.74}$$

$$\mathbf{E}_N^{(n)}(r) = \left\{\mathbf{x} \in I^6 : x^{(1)} < \frac{1}{N}\sum_{i=1}^N \sum_{m=1}^{n-1} \varphi_r(v_i^{(m)}) \chi_{m+1}^{(n)}(x^{(6)})\right\} \tag{4.75}$$

$$\mathbf{F}_N^{(n)}(r) = \left\{ \mathbf{x} \in I^6 : x^{(3)} < \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{m=1}^{n-1} \chi_{i,j}\left(x^{(1)}, x^{(2)}\right) \Delta t \right.$$
$$\left. q\left([v_i^{(m)}, v_j^{(m)}; x^{(4)}]\right) \chi_{m+1}^{(n)}\left(x^{(6)}\right) \right\} \quad (4.76)$$

$$\mathbf{G}_N^{(n)}(r) = \left\{ \mathbf{x} \in I^6 : x^{(3)} < \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{m=1}^{n-1} \chi_{i,j}\left(x^{(1)}, x^{(2)}\right) \right.$$
$$\left. g^*_{v_i^{(m)}, v_j^{(m)}}\left(x^{(4)}\right) \chi_{m+1}^{(n)}\left(x^{(6)}\right) \right\} \quad (4.77)$$

Da der Operator $L^{(n)}$ sich aus lauter charakteristischen Funktionen zusammensetzt, ergibt sich durch Einsetzen und Benutzung obiger Mengen und Zusammenhänge:

$$\Delta_N^{(n)}(r) = n \left(\frac{A\left(\mathbf{E}_N^{(n)} \cap \left(\mathbf{F}_N^{(n)}\right)^C, \mathbf{X}_{nN}^+\right)}{nN} - \left| \mathbf{E}_N^{(n)} \cap \left(\mathbf{F}_N^{(n)}\right)^C \right| + \right.$$
$$\left. \frac{A\left(\mathbf{F}_N^{(n)} \cap \mathbf{G}_N^{(n)}, \mathbf{X}_{nN}^+\right)}{nN} - \left| \mathbf{F}_N^{(n)} \cap \mathbf{G}_N^{(n)} \right| \right) \quad (4.78)$$

Mittels eines Lemmas von NIEDERREITER und WILLS wird dies nun weiter abgeschätzt: Für $E \subset \bar{I}^s$ und $\varepsilon > 0$ seien

$$E_\varepsilon = \left\{ \mathbf{x} \in \bar{I}^s : \exists \mathbf{y} \in E, |\mathbf{x} - \mathbf{y}| < \varepsilon \right\} \quad (4.79a)$$
$$E_{-\varepsilon} = \left\{ \mathbf{x} \in \bar{I}^s : \forall \mathbf{y} \in \bar{I}^s / E, |\mathbf{x} - \mathbf{y}| \geq \varepsilon \right\} \quad (4.79b)$$

E_ε beinhaltet damit alle Punkte, die von E maximal den Abstand ε haben, während $E_{-\varepsilon}$ nur jene Punkte von E enthält, die vom Komplement von E einen Abstand $\geq \varepsilon$ besitzen. Dementsprechend stellt E_ε / E salopp ausgedrückt den ε-Streifen auf der äußeren Seite von E dar, $E/E_{-\varepsilon}$ den Streifen auf der inneren Seite des Randes von E.

Lemma 4.9 (Niederreiter, Wills). *Wenn $E \subset \bar{I}^s$ messbar ist mit*

$$\exists K > 0 \forall \varepsilon > 0 : \max\left(|E_\varepsilon / E|, |E / E_{-\varepsilon}|\right) \leq K\varepsilon,$$

dann gilt für eine beliebige P-elementige Menge $\mathbf{Z} \subset \bar{I}^s$

$$\left| \frac{A(E; \mathbf{Z})}{P} - |E| \right| \leq \left(4K\sqrt{s} + 2K + 1\right) \sqrt[s]{D_P(\mathbf{Z})}$$

Man definiert dann $K(E) = \inf\{K > 0 : \forall \varepsilon > 0 \max\left(|E_\varepsilon / E|, |E/E_{-\varepsilon}|\right) \leq K\varepsilon\}$.

Lemma 4.10. *Wenn eine Funktion $g : \bar{I}^s \to [0,1]$ von beschränkter Variation $V(g)$ im Sinne von* HARDY *und* KRAUSE *ist und $E = \left\{\left(x^{(0)}, \mathbf{x}'\right) \in I^{s+1} : x^{(0)} < g(\mathbf{x}')\right\}$, dann gilt*

$$K(E) \leq \frac{6^s - 2^s + 2}{2} s V(g) + 1 \quad (4.80)$$

Mittels einiger weiterer Hilfslemmata konnte LÉCOT schließlich in [Léc91] folgende Abschätzungen für die Konstanten $K(\cdot)$ obiger Mengen angeben. $\mathbf{X}^*_{nN}$ bezeichne hierbei die die Menge $\mathbf{X}^+_{nN}$ nach Streichung der Koordinate $x^{(5)}$.

Lemma 4.11. *Für die Menge* $\mathbf{E}^{(n)}_N(r)$ *aus Definition 4.75 gilt*

$$K\left(\mathbf{E}^{(n)}_N(r)\right) \leq 3 + 4Qt^{(n-1)} + 4((4\sqrt{5}+2)(90+9Qt^{(n-1)})+1) \\ (n-1)D_{(n-1)N}\left(\mathbf{X}^*_{(n-1)N}\right)^{1/5} \quad (4.81)$$

Lemma 4.12. *Für die Menge* $\mathbf{F}^{(n)}_N(r)$ *aus Definition 4.76 gilt*

$$K\left(\mathbf{F}^{(n)}_N(r)\right) \leq 90 + 9Qt^{(n-1)} \quad (4.82)$$

Lemma 4.13. *Für die Menge* $\mathbf{G}^{(n)}_N(r)$ *aus Definition 4.77 gilt*

$$K\left(\mathbf{G}^{(n)}_N(r)\right) \leq 99 + 18Qt^{(n-1)} + 18((4\sqrt{5}+2)(90+9Qt^{(n-1)})+1) \\ (n-1)D_{(n-1)N}\left(\mathbf{X}^*_{(n-1)N}\right)^{1/5} \quad (4.83)$$

Unter Verwendung von Lemma 4.9 ergibt sich mit diesen Abschätzungen durch einfaches Einsetzen:

Lemma 4.14. *Es gilt:*

$$\left|\Delta^{(n)}_N(r)\right| \leq \left((4\cdot\sqrt{6}+2)\left(282+40Qt^{(n-1)}+22\left((4\sqrt{5}+1)(90+9Qt^{(n-1)})+1\right)\right.\right. \\ \left.\left.(n-1)D_{(n-1)N}\left(\mathbf{X}^*_{(n-1)N}\right)^{1/5}\right)+2\right) nD_{nN}\left(\mathbf{X}^+_{nN}\right)^{1/6} \quad (4.84)$$

Bemerkung 4.3. Für ein fixes n gehen in diesem Term die beiden auftretenden Diskrepanzen $D_{(n-1)N}\left(\mathbf{X}^*_{(n-1)N}\right)^{1/5}$ und $nD_{nN}\left(\mathbf{X}^+_{nN}\right)^{1/6}$ mit $N \to \infty$ nach Voraussetzung gegen 0.

Damit geht auch der letzte Term in Gleichung (4.72) gegen 0, womit der Satz bewiesen ist. □

4.8.3 3-dimensionale Variante der QMC Methode

LÉCOT und KHETTABI stellten in [LK99] auch eine Variante de QMC1-Schemas vor, wo nicht die Absolutbeträge, sondern auch die Richtung der Geschwindigkeiten berechnet und gespeichert wird. Wiederum wird ein 1-komponentiges homogenes Gas mit isotroper Streuung betrachtet und analog zu Abschnitt 3.7 eine schwache Gleichung hergeleitet. Mit $\gamma(\mathbf{g}) = 4\pi n_c\,|\mathbf{g}|\,\sigma(|\mathbf{g}|)$, $\|\gamma\|_\infty < \infty$ wird zuerst die Variablentransformation

$$\nu(x^{(8)}, x^{(9)}) = \begin{pmatrix} 2x^{(8)}-1 \\ 2\sqrt{x^{(8)}(1-x^{(8)})}\cos(2\pi x^{(9)}) \\ 2\sqrt{x^{(8)}(1-x^{(8)})}\sin(2\pi x^{(9)}) \end{pmatrix} \quad (4.85)$$

auf sphärische Polarkoordinaten durchgeführt. Diese Transformation liefert - wenn (x_8, x_9) auf $[0,1)$ gleichverteilt ist - eine auf S^2 gleichverteilte Zufallsvariable.

Mit $\mathbf{v}' = \left[\mathbf{v}, \mathbf{w}; x^{(8)}, x^{(9)}\right] = \frac{1}{2}\left(\mathbf{v} + \mathbf{w} + |\mathbf{v} - \mathbf{w}|\, \nu(x^{(8)}, x^{(9)})\right)$ ergibt sich die schwache Variante

$$\frac{d}{dt}\int_{\mathbb{R}^3} \varphi(\mathbf{v}) v^2 f(\mathbf{v}, t) d\mathbf{v} = \int_{\mathbb{R}^6 \times I^2} \left(\varphi(\mathbf{v}') - \varphi(\mathbf{v})\right) \gamma(\mathbf{g}) \\ v^2 f(\mathbf{v}, t) w^2 f(\mathbf{w}, t) d\mathbf{v}\, d\mathbf{w}\, dx^{(8)} dx^{(9)} \quad (4.86)$$

Diese Gleichung benutzt im Gegensatz zu den bei den QMC1, LD und DSMC-N Methoden benutzten schwachen Gleichungen 3-dimensionale Geschwindigkeiten $\mathbf{v}_j^{(n)}$ zur Beschreibung der Dichtefunktion.

Für die Simulation wird eine Basis b gewählt und eine natürliche Zahl m, die aufgeteilt wird in $m = d_1 + d_2 + d_3$. Die $N = 2b^m$ benutzten Teilchen werden nun durch einen 3-dimensionalen Index $\mathbf{j}$ mit $0 \le j_i \le b^{d_i}$, $i = 1, 2$ und $0 \le j_3 \le 2b^{d_3}$ beschrieben und nach jedem Zeitschritt entsprechend den Komponenten ihrer Geschwindigkeit umgeordnet, sodass gilt:

$$\begin{aligned} j_1 < k_1 &\Rightarrow \left(v_{\mathbf{j}}^{(n)}\right)_1 \le \left(v_{\mathbf{k}}^{(n)}\right)_1 \\ j_1 = k_1, j_2 < k_2 &\Rightarrow \left(v_{\mathbf{j}}^{(n)}\right)_2 \le \left(v_{\mathbf{k}}^{(n)}\right)_2 \\ j_1 = k_1, j_2 = k_2, j_3 < k_3 &\Rightarrow \left(v_{\mathbf{j}}^{(n)}\right)_3 \le \left(v_{\mathbf{k}}^{(n)}\right)_3 \end{aligned} \quad (4.87)$$

Durch den Vorwärts-Euler-Schritt erhält man völlig analog zu Gleichung (4.55) die Näherung $g^{(n+1)}$ für f_{n+1} mit der Bezeichnung $D = \left\{0, \dots b^{d_1}\right\} \times \left\{0, \dots b^{d_2}\right\} \times \left\{0, \dots 2b^{d_3}\right\}$:

$$\int_{\mathbb{R}^3} \varphi(\mathbf{v}) g^{(n+1)}(\mathbf{v}) = \frac{1}{N}\sum_{\mathbf{j}\in D}\left(1 - \frac{\Delta t}{N}\sum_{\mathbf{k}} \gamma\left(\left|\mathbf{v}_{\mathbf{j}}^{(n)} - \mathbf{v}_{\mathbf{k}}^{(n)}\right|\right)\right)\varphi(\mathbf{v}_{\mathbf{j}}^{(n)}) + \\ \frac{\Delta t}{N^2}\sum_{\mathbf{j},\mathbf{k}\in D}\int_{I^2} \varphi\left(\left[\mathbf{v}_{\mathbf{j}}^{(n)}, \mathbf{v}_{\mathbf{k}}^{(n)}; x_8, x_9\right]\right) dx_8 dx_9 \gamma\left(\left|\mathbf{v}_{\mathbf{j}}^{(n)} - \mathbf{v}_{\mathbf{k}}^{(n)}\right|\right) \varphi\left(\left[\mathbf{v}_{\mathbf{j}}^{(n)}, \mathbf{v}_{\mathbf{k}}^{(n)}; x_8, x_9\right]\right) \quad (4.88)$$

Simuliert wird nun der Operator

$$\Phi^{(n)}(\mathbf{x}) = \sum_{\mathbf{j},\mathbf{k}\in D} \chi_{\mathbf{j},\mathbf{k}}(\tilde{\mathbf{x}}, \tilde{\tilde{\mathbf{x}}})\left((1 - c_{\mathbf{j},\mathbf{k}}^{(n)}(x_7))\varphi\left(\mathbf{v}_{\mathbf{j}}^{(n)}\right) + c_{\mathbf{j},\mathbf{k}}^{(n)}(x_7)\varphi\left(\left[\mathbf{v}_{\mathbf{j}}^{(n)}, \mathbf{v}_{\mathbf{k}}^{(n)}; x_8, x_9\right]\right)\right) \quad (4.89)$$

mit $\mathbf{x} = (\tilde{\mathbf{x}}, \tilde{\tilde{\mathbf{x}}}, x_7, x_8, x_9)$, $\tilde{\mathbf{x}}, \tilde{\tilde{\mathbf{x}}} \in I^3$ und

$$J_{\mathbf{j}} = \left[\frac{j_1}{b^{d_1}}, \frac{j_1 + 1}{b^{d_1}}\right) \times \left[\frac{j_2}{b^{d_2}}, \frac{j_2 + 1}{b^{d_2}}\right) \times \left[\frac{j_3}{b^{d_3}}, \frac{j_3 + 1}{b^{d_3}}\right) \quad (4.90)$$

$$c_{\mathbf{j},\mathbf{k}}^{(n)} \dots \qquad \text{char. Funktion von } \left[0, \Delta t\gamma\left(\left|\mathbf{v}_{\mathbf{j}}^{(n)} - \mathbf{v}_{\mathbf{k}}^{(n)}\right|\right)\right) \quad (4.91)$$

$$\chi_{\mathbf{j},\mathbf{k}} \dots \qquad \text{char. Funktion von } J_{\mathbf{j},\mathbf{k}} = J_{\mathbf{j}} \times J_{\mathbf{k}} \quad (4.92)$$

Die Näherung für f_{n+1} ist damit definiert durch

$$\int_{\mathbb{R}^3} \varphi(\mathbf{v}) f^{(n+1)}(\mathbf{v}) = \frac{1}{N}\sum_{0\le l<N} \Phi^{(n)}(\mathbf{y}_{nN+l}) \quad (4.93)$$

Dabei bezeichnet die Folge $\mathbf{Y}$ eine symmetrisierte $(0, 9)$-Folge (oder allgemeiner eine Symmetrisierung einer Folge, wobei $\tilde{\mathbf{x}}$ und $\tilde{\tilde{\mathbf{x}}}$ $(0, 3)$-Folgen in Basis b darstellen). Die Simulation läuft nun nach folgendem Schema ab. In jedem Zeitschritt n werden $b^m = \frac{N}{2}$ Teilchenpaare auf die folgende Weise behandelt ($nb^m \le l < (n+1)b^m$):

(QMC3 1.) Das 9-dimensionale Element $\mathbf{x}_{nb^m+l}$ der Folge kleiner Diskrepanz wird symmetrisiert durch

$$\mathbf{y}_{nb^m+l} = (x_1, x_2, \tilde{x}_3, x_4, x_5, \tilde{x}_6, x_7, x_8, x_9) \tag{4.94}$$

$$\mathbf{y}_{(n+1)b^m+l} = \left(x_4, x_5, \tilde{x}_6, x_1, x_2, \tilde{x}_3, x_7, 1 - x_8, x_9 \pm \frac{1}{2}\right) \tag{4.95}$$

wobei $\tilde{x}_3 = \frac{1}{2}\left(x_3 + \frac{\lfloor b^{d_3} \rfloor}{b^{d_3}}\right)$ und $\tilde{x}_6 = \frac{1}{2}\left(x_6 + \frac{\lfloor b^{d_3} x_6 \rfloor + 1}{b^{d_3}}\right)$ bedeuten, sowie

$$x \pm \frac{1}{2} = \begin{cases} x + \frac{1}{2} & \text{für } 0 \leq x < \frac{1}{2} \\ x - \frac{1}{2} & \text{für } \frac{1}{2} \leq x < 1 \end{cases} \tag{4.96}$$

Dadurch wird einerseits wieder sichergestellt, dass wenn Teilchen $\mathbf{j}$ den Kollisionspartner $\mathbf{k}$ hat, dies auch umgekehrt gilt. Andererseits muss dann natürlich auch die Richtung der Geschwindigkeitsänderung (beschrieben durch $\nu(x_8, x_9)$) übereinstimmen, weshalb die letzten beiden Koordinaten ebenfalls "symmetrisiert" werden.

(QMC3 2.) Die 3-dimensionalen Indices der betrachteten Teilchen sind nun definiert durch

$$\mathbf{j}^{(n)}(l) = \left(\left\lfloor b^{d_1} y_{nb^m+l,1} \right\rfloor, \left\lfloor b^{d_2} y_{nb^m+l,2} \right\rfloor, 2\left\lfloor b^{d_3} y_{nb^m+l,3} \right\rfloor\right) \tag{4.97}$$

$$\mathbf{k}^{(n)}(l) = \left(\left\lfloor b^{d_1} y_{nb^m+l,4} \right\rfloor, 2\left\lfloor b^{d_2} y_{nb^m+l,5} \right\rfloor, \left\lfloor b^{d_3} y_{nb^m+l,6} \right\rfloor + 1\right) \tag{4.98}$$

Zur Abkürzung sei $v^{(n)}_{\mathbf{j},\mathbf{k}} := \left|\mathbf{v}^{(n)}_{\mathbf{j}} - \mathbf{v}^{(n)}_{\mathbf{k}}\right|$.

(QMC3 3.) Die beiden freien Parameter für den Kollisionsprozess, (x_8, x_9) definieren die Richtung der Geschwindigkeitsänderung:

$$\nu^{(n)}(l) = \nu(y_{nb^m+l,8}, y_{nb^m+l,9}) \tag{4.99}$$

(QMC3 4.) Die Teilchen $\mathbf{j}$ und $\mathbf{k}$ kollidieren, wenn $x_{nb^m+l,7} < \Delta t \gamma(v^{(n)}_{\mathbf{j},\mathbf{k}})$. Anderenfalls wird für das nächste Paar $(l \to l+1)$ wieder bei Punkt 1 begonnen.

(QMC3 5.) Die Geschwindigkeiten nach der Kollision betragen

$$\mathbf{v}^{(n+1)}_{\mathbf{j}(l)} = \frac{1}{2}\left(\mathbf{v}_{\mathbf{j}}(l)^{(n)} + \mathbf{v}_{\mathbf{k}}(l)^{(n)} + v^{(n)}_{\mathbf{j},\mathbf{k}} \nu^{(n)}(l)\right) \tag{4.100}$$

$$\mathbf{v}^{(n+1)}_{\mathbf{k}(l)} = \frac{1}{2}\left(\mathbf{v}_{\mathbf{j}}(l)^{(n)} + \mathbf{v}_{\mathbf{k}}(l)^{(n)} - v^{(n)}_{\mathbf{j},\mathbf{k}} \nu^{(n)}(l)\right) \tag{4.101}$$

Bemerkung 4.4. Die Symmetrisierung bewirkt, dass in Schritt 5 gleich beide Geschwindigkeiten ersetzt werden können, und $\mathbf{y}_{(n+1)b^m+l}$ nicht gesondert betrachtet werden braucht.

Bemerkung 4.5. Die Forderung, dass $\tilde{\mathbf{x}}$ und $\tilde{\tilde{\mathbf{x}}}$ $(0,3)$-Folgen darstellen, bewirkt, dass jedes Teilchen genau einmal betrachtet wird.

Bemerkung 4.6. Da die $\mathbf{v}^{(n+1)}$ aus den $\mathbf{v}^{(n)}$ berechnet werden, müsste $\mathbf{V}^{(n)}$ zwischengespeichert werden. Da aber jedes Teilchen nur einmal betrachtet wird, und $\mathbf{v}^{(n)}_{\mathbf{j}}$ für die anderen Kollisionen irrelevant ist, können die Geschwindigkeiten gleich ersetzt werden.

Bemerkung 4.7. LÉCOT und KHETTABI führen in [LK99] keinen Konvergenzbeweis und keine Diskrepanzabschätzung an. Aufgrund ihrer Ergebnisse (vgl. dazu auch das Kapitel 5 dieser Arbeit), welche für den von ihnen betrachteten Fall der Lösung von KROOK und WU durchwegs besser sind als die anderen bekannten Simulationsmethoden, kann allerdings stark vermutet werden, dass dieses Verfahren konvergiert.

Bemerkung 4.8. LÉCOT's QMC Schemata verwenden $(0, s)$-Folgen in Basis b, wobei folgende Eigenschaft ausgenutzt wird:

Wenn $(x_1, \ldots x_s)$ eine $(0, s)$-Folge ist, so erhält man durch Streichung von t beliebigen Koordinaten eine $(0, s-t)$- Folge, also wieder eine $(s-t)$-dimensionale Folge kleiner Diskrepanz.

Prinzipiell könnten alle Folgen mit dieser und der in Bemerkung 4.5 beschriebenen Eigenschaft benutzt werden.

Kapitel 5

Numerische Resultate

Inhaltsangabe

5.1 Die Simulation 72
5.2 Überblick über alle Simulationsmethoden 73
5.3 Zahl der Teilchenkollisionen 84
5.4 3-dimensionale vs. 1-dimensionale Simulation 84
5.5 Bedeutung des Umordnens der Teilchen 85
5.6 MC vs. QMC Simulation 86
5.7 Direkte Vergleiche von Halton- und $(0, s)$-Folgen 86

In den folgenden Abschnitten möchte ich anhand einiger numerischer Experimente die in den bisherigen Kapiteln beschriebenen Ergebnisse verdeutlichen. Zum einen geht es dabei natürlich um den Vergleich herkömmlicher Simulationsmethoden mit QMC Verfahren mittels des dabei erhaltenen Fehlers (im Vergleich zur bekannten Lösung von KROOK und WU bzw. zur stationären Boltzmann-Verteilung). Zum anderen möchte ich die Bedeutung des Umordnens bei QMC-Verfahren demonstrieren, womit die durch die Deterministik der Folgen geringer Diskrepanz unweigerlich auftretenden Korrelationen gebrochen werden.

In einem letzten Abschnitt soll schließlich aufgrund ATANASSOV's neuer Diskrepanzschranke der Halton-Folge dies auch numerisch anhand der (Q)MC Integration einiger ausgewählter Funktionen betrachtet werden.

5.1 Die Simulation

Durchgeführt wurden sämtliche Berechnungen auf einem PC mit einem IBM Cyrix 686 Prozessor und 64MB RAM (300MB Swap space) unter RedHat Linux 6.2. Implementiert wurden alle Folgen und Simulations-Schemata objektorientiert in C++, wobei alle Folgen auf der Klasse `SimulationSeqsBase` und alle Simulationsschemata auf der Klasse `BoltzmannSimBase` aufbauen. Diese beiden Basisklassen stellen das Grundgerüst zur Verfügung, sodass in der entsprechenden Implementierung der Folge oder des Simulationsschemas nur mehr auf die Unterschiede eingegangen werden braucht.

Zur graphischen Verarbeitung der Daten wurde Mathematica 4.0 benutzt.

5.2 Überblick über alle Simulationsmethoden

Es wurden insgesamt 19 mehr oder weniger verschiedene Simulationsschemata bzw. deren leichte Abwandlungen getestet (Tabelle 5.1).

Abkürzung	Beschreibung
none	Anfangsverteilung in jedem Schritt (keine Teilchenbewegung)
bir	Bird's DSMC-B Schema
by	Belotserkovskiy - Yanitskiy's Schema
dsh	Deshepande's Schema
nan	Nanbu's DSMC-N Schema
bab	Babovsky's Modifikation des DSMC-N Schemas
ldh	Lécot's LD Schema (Halton Folge)
lduns	LD Schema ohne Umordnen
ldhb	Lécot's LD Schema (Halton Folge) mit Babovsky's Modifikation
ldhm	Lécot's LD Schema (Hammersley Menge)
ldts	Lécot's LD Schema mit $(0,4)$-Folge
qmc1	Lécot's QMC1 Schema mit $(0,5)$-Folge
qmc1hal	QMC1 mit Halton-Folge
qmc1uns	QMC1 Schema ohne Umordnen
qmc1mc	QMC1 mit Zufallszahlen statt Folgen kleiner Diskr.
qmc3	Lécot und Khettabi's QMC3 Schema mit $(0,9)$-Folge
qmc3hal	QMC3 mit Halton-Folge
qmc3uns	QMC3 Schema ohne Umordnen
qmc3mc	QMC3 mit Zufallszahlen statt Folgen kleiner Diskr.
qmc3vabs	QMC3 Schema, Umordnen entsprechend $\lvert\mathbf{v}_i\rvert$

Tabelle 5.1: Alle implementierten Schemata

Es stellt sich natürlich die Frage, in wie weit, bzw. ob überhaupt QMC Schemata geringere Fehler liefern als herkömmliche (MC) Schemata. Da für die Boltzmann-Gleichung praktische keine exakten Lösungen bekannt sind, außer der Lösung von Krook und Wu[1], die sich aber von der Gleichgewichtslösung nur sehr wenig unterscheidet, können die folgenden Ergebnisse natürlich nur Hinweise darauf sein.

Abbildung 5.1 zeigt den Vergleich aller Schemata für Krook und Wu's Lösung bei $29282 = 2{\cdot}11^4$ Teilchen und einem Zeitschritt $\Delta t = \frac{T}{M} = \frac{1.5}{100}$. Dabei wird der bei der Simulation auftretende Fehler

$$\tilde{D}_N^*\left(\mathbf{V}^{(n)}, f^{(n)}\right) = \max_i \left| \frac{1}{N}\sum_{j=1}^{N} \chi_{\left[0,\mathbf{v}_i^{(n)}\right)}\left(\mathbf{v}_j^{(n)}\right) - \int_{\mathbb{R}^3} \chi_{\left[0,\mathbf{v}_i^{(n)}\right)}(\mathbf{v})\, f^{(n)}(\mathbf{v})\, d\mathbf{v} \right| \tag{5.1}$$

[1]Für ein homogenes Gas mit isotroper Streuung und einem Wirkungsquerschnitt $\sigma(\vartheta, g) = \frac{k}{g} = \frac{3}{2\pi}\frac{1}{g}$ ist $f(v,t) = \frac{4}{K(t)^{5/2}\sqrt{2\pi}}\left(5K(t) - 3 + \frac{1-K(t)}{K(t)}v^2\right)e^{-\frac{v^2}{2K(t)}}$ mit $K(t) = 1 - \frac{2}{5}e^{-t}$ eine Lösung der Boltzmann-Gleichung zur Anfangsbedingung $f_0(v) = \frac{20\pi}{9}\left(\frac{5}{5\pi}\right)^{3/2} v^2 \exp\left(-\frac{5}{6}v^2\right)$

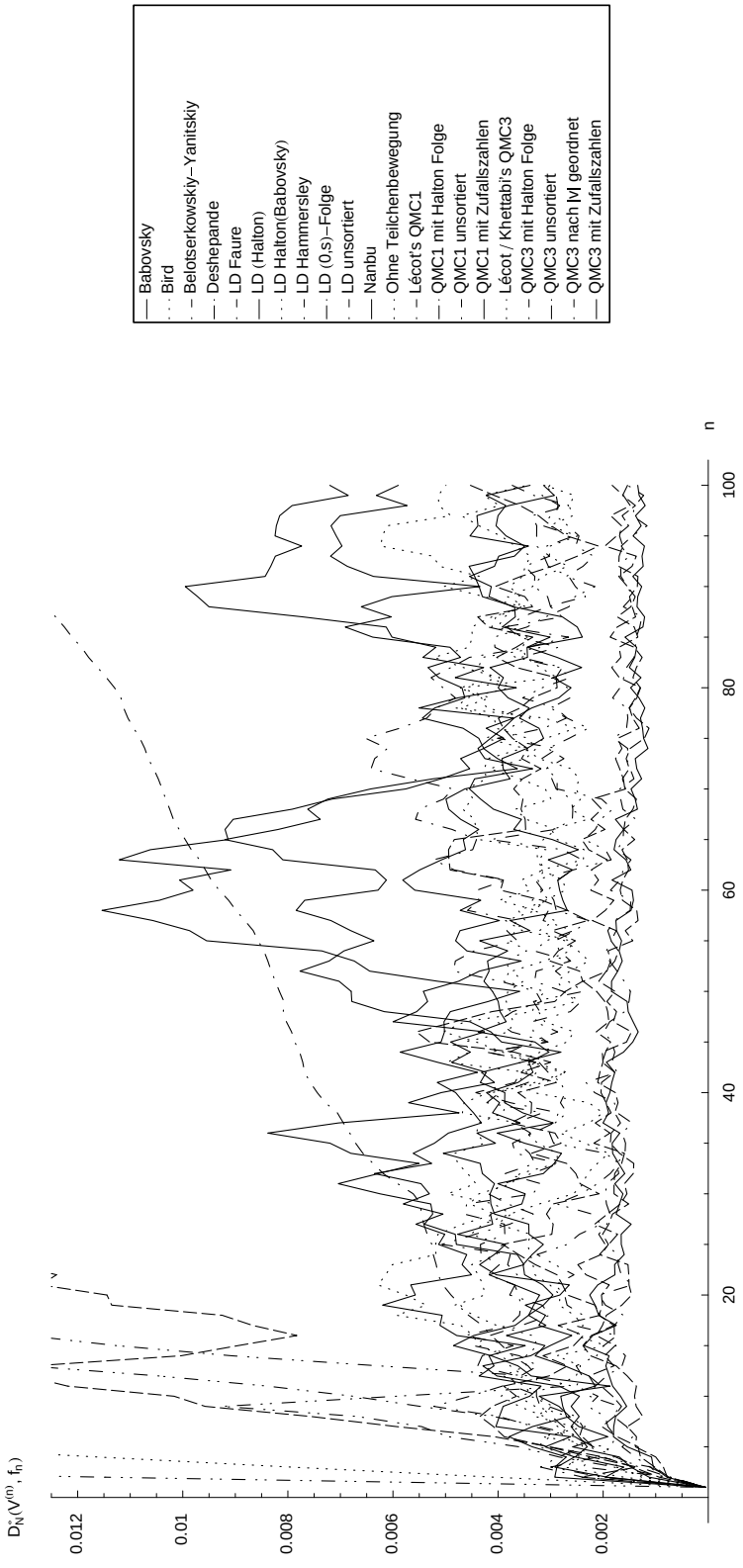

Abbildung 5.1: Fehler-Vergleich aller Schemata, $N = 29282$, $\Delta t = 0.015$

betrachtet, da die Berechnung der Diskrepanz

$$D_N^*\left(\mathbf{V}^{(n)}, f^{(n)}\right) = \sup_{\mathbf{w}\in\mathbb{R}^3} \left| \frac{1}{N}\sum_{j=1}^{N} \chi_{[0,\mathbf{w})}\left(\mathbf{v}_j^{(n)}\right) - \int_{\mathbb{R}^3} \chi_{[0,\mathbf{w})}\left(\tilde{\mathbf{v}}\right) f^{(n)}\left(\tilde{\mathbf{v}}\right) d\tilde{\mathbf{v}} \right| \tag{5.2}$$

zu aufwendig wäre.

Sehr schön zu sehen ist hier, dass die QMC Methoden (LD, QMC1 und QMC3) doch um einiges bessere Lösungen liefern als herkömmliche Verfahren (Bird, B-Y, Nanbu). Als Vergleich ist auch die Diskrepanz der Anfangsverteilung $D^*(V^{(0)}, f_n)$ angegeben, wenn die Teilchen also überhaupt nicht bewegt werden.

Während bei $242 = 2 \cdot 11^2$ Teilchen die herkömmlichen Methoden (Bird, Nanbu) noch einen Fehler aufweisen, der dem ohne jeglicher Teilchenbewegung entspricht, weisen die QMC Methoden bereits eine signifikante Verbesserung auf (Abbildung 5.2). Mit steigender Teilchenzahl sinkt natürlich der Fehler aller Methoden erwartungsgemäß stark, wobei allerdings der Fehler der QMC Methoden nach wie vor nur etwa die Hälfte dessen beträgt, was herkömmliche Methoden liefern (Tabelle 5.2).

Ähnlich sieht die Situation aus, wenn man die Abhängigkeit des Fehlers von Δt (bzw. bei festem T von M) betrachtet (Tabelle 5.4). Während es für einige Schemata (Bird, B-Y, Deshepande, QMC3) keine Fehlerabschätzung gibt, bewies LÉCOT in [Léc89a] für das LD Schema (und damit auch für Nanbu's Schema, das dafür ja die Grundlage bildet) die Diskrepanzschranke (4.32). Wie dort schon erwähnt, spielt das Δt im Nenner im Grunde keine Rolle bei den numerischen Ergebnissen, sodass diese Abschätzung für kleine Δt um einige Größenordnungen zu hoch ist. Numerisch zeigt sich viel mehr folgende Abhängigkeit von Δt: Wird bei den QMC Methoden ein Zeitschritt zehnmal so klein gewählt, so halbiert sich von einigen Ausnahmen abgesehen der Fehler in etwa. Für die QMC1 Simulation enthält die Diskrepanz-Schranke aus dem Beweis von Satz 4.7 Δt linear, was sich allerdings numerisch nicht bestätigt.

Interessant im Zusammenhang mit dem geringeren Fehler der QMC Methoden ist natürlich der Rechenaufwand bzw. die zur Simulation benötigte Zeit. Tabelle 5.5 und Abbildung 5.3 zeigt den Vergleich für die verschiedenen Methoden mit $N = 242, 686, 2662, 6750$ und 29282. Die Tatsache, dass die Methoden, die eine $(0, s)$-Folge benutzen, beträchtlich mehr Zeit benötigen als Methoden, welche die Halton-Folge oder Zufallszahlen verwenden, rührt daher, dass bei der Implementation zwar die optimierte Konstruktion aus Satz 2.27 von LÉCOT benutzt wurde, aber dennoch jeder Koeffizient einzeln berechnet werden muss. Auf Vektorcomputern allerdings würde dieser für einen PC doch beträchtliche Nachteil eher unerheblich werden.

Bei der QMC3 Methode fällt zusätzlich ins Gewicht, dass durch die komplexere Sortierbedingung aus Gleichung (4.87) der Sortieralgorithmus verschachtelt angewendet wird. Für das Umordnen wird allgemein ein 3-Median QuickSort Algorithmus angewendet (für Mengen mit mehr als 4 Elementen) mit nachfolgendem InsertionSort. Bei der QMC3 Methode werden dabei natürlich sehr viele redundante Sortierungen durchgeführt, die durch das anschließende Umordnen entsprechend der 2. bzw. 3. Koordinate ohnehin wieder hinfällig werden. Insofern läege auch hier noch einiges Potential, um die Differenz zwischen QMC3uns und QMC3, die ja nur durch das Umordnen bei der QMC3 Methode hervorgerufen ist, noch stark zu verringern.

Da die QMC3 Methode allerdings 3-dimensionale Kollisionen und dafür eine 9-dimensionale Folge benutzt, und in jedem Zeitschritt jedes Teilchen betrachtet werden muss, bleibt der Aufwand trotzdem beträchtlich größer als bei anderen Schemata.

Auch die Implementierung der Faure-Folge wurde nicht optimiert, sondern einfach der Definition folgend durchgeführt.

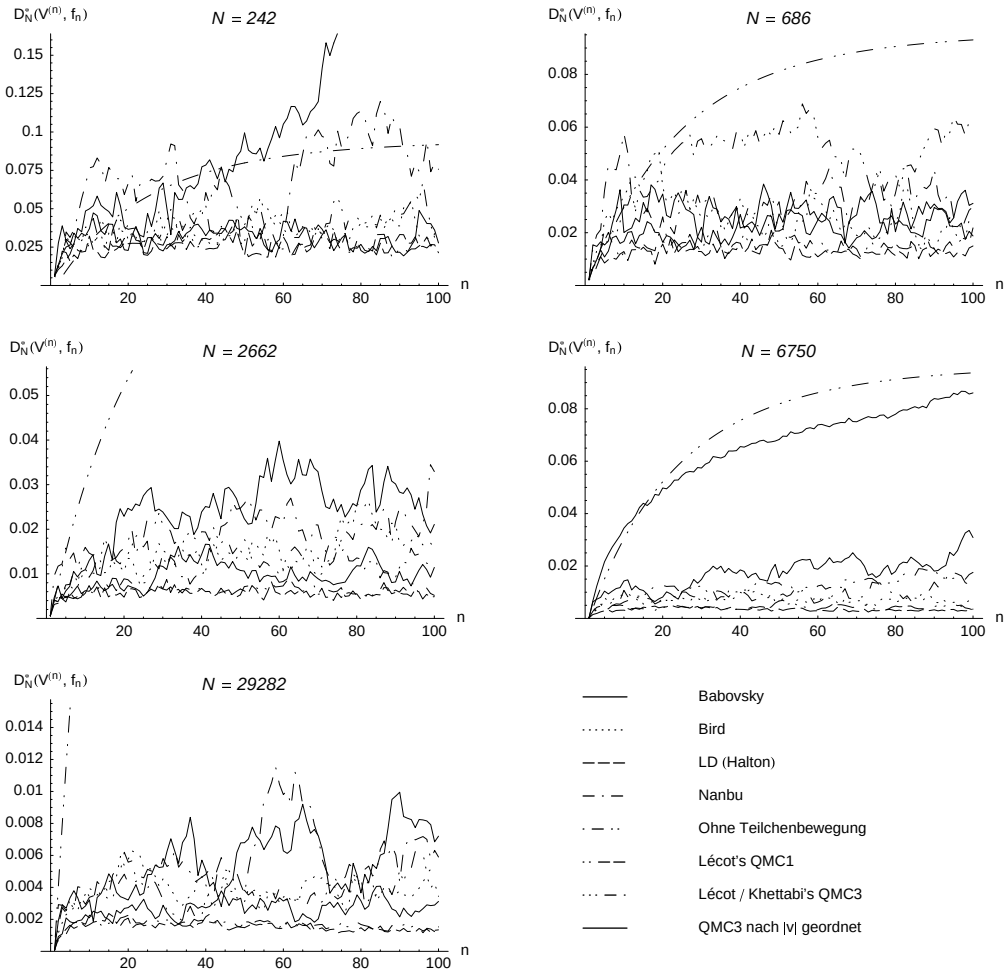

Abbildung 5.2: Die wichtigsten Schemata mit Krook / Wu's Lösung, $\Delta t = 0.015$

	242	686	2662	6750	29282
Ohne Teilchenbewegung	0.091733	0.093071	0.093612	0.093726	0.093783
Bird	0.029561	0.026208	0.015773	0.007603	0.00344
Belotserkowskiy-Yanitskiy	0.04006	0.025362	0.011107	0.00545	0.003731
Deshepande	0.048734	0.03996	0.013726	0.005512	0.004526
Nanbu	0.075704	0.029877	0.032945	0.014643	0.005898
Babovsky	0.238841	0.031121	0.02108	0.030856	0.007202
LD (Halton)	0.021308	0.014976	0.004878	0.003564	0.001342
LD unsortiert	0.202734	0.043431	0.127896	0.099233	0.139348
LD Halton(Babovsky)	0.029422	0.014525	0.012803	0.033172	0.003113
LD Hammersley	0.201585	0.087273	0.040565	0.0248	0.014326
LD (0,s)-Folge	0.036404	0.015499	0.00854	0.231281	0.001829
Lécot's QMC1	0.027208	0.021825	0.00517	0.00324	0.001533
QMC1 mit Halton Folge	0.018252	0.010453	0.007622	0.003201	0.001536
QMC1 unsortiert	0.13136	0.124563	0.045781	0.042343	0.038563
QMC1 mit Zufallszahlen	0.046757	0.032473	0.013898	0.009184	0.00339
Lécot / Khettabi's QMC3	0.034534	0.06221	0.012058	0.017568	0.004991
QMC3 mit Halton Folge	0.055602	0.017389	0.009928	0.006779	0.003015
QMC3 unsortiert	0.043573	0.286141	0.242699	0.118548	0.110167
QMC3 mit Zufallszahlen	0.076109	0.023875	0.009949	0.007958	0.003135
QMC3 nach $\|v\|$ sortiert	0.027707	0.018763	0.011432	0.086067	0.00312

Tabelle 5.2: Fehler der diversen Methoden bei $M = 100$, $T = 1.5$ und Krook / Wu's Lösung

	242	686	2662	6750	29282
Ohne Teilchenbewegung	0.435216	0.363564	0.300323	0.268493	0.230125
Bird	0.641527	0.557611	0.526124	0.553368	0.551525
Belotserkowskiy-Yanitskiy	0.586157	0.562635	0.570594	0.591127	0.543629
Deshepande	0.550449	0.493024	0.54375	0.589844	0.524848
Nanbu	0.470205	0.537549	0.432736	0.479035	0.499103
Babovsky	0.260881	0.531303	0.489351	0.3945	0.479682
LD (Halton)	0.701169	0.643299	0.674925	0.639297	0.64305
LD unsortiert	0.290741	0.48027	0.260756	0.262017	0.191622
LD Halton(Babovsky)	0.642385	0.647981	0.552576	0.386292	0.561237
LD Hammersley	0.291777	0.373413	0.406354	0.419279	0.412814
LD (0,s)-Folge	0.603592	0.638043	0.603917	0.166051	0.612946
Lécot's QMC1	0.656638	0.585633	0.667553	0.650106	0.630112
QMC1 mit Halton Folge	0.729373	0.698354	0.618336	0.65148	0.629922
QMC1 unsortiert	0.369801	0.318938	0.391017	0.358608	0.316534
QMC1 mit Zufallszahlen	0.557993	0.524791	0.542171	0.531942	0.552949
Lécot / Khettabi's QMC3	0.613199	0.425248	0.560178	0.45838	0.515339
QMC3 mit Halton Folge	0.526429	0.620425	0.584822	0.566378	0.564347
QMC3 unsortiert	0.570842	0.191593	0.179531	0.241847	0.214469
QMC3 mit Zufallszahlen	0.469233	0.571887	0.584554	0.548193	0.560552
QMC3 nach $\|v\|$ sortiert	0.653327	0.60878	0.566937	0.278161	0.561018

Tabelle 5.3: Fehlerexponent bei $M = 100$, $T = 1.5$ und Krook / Wu's Lösung

	5	10	20	50	100	200	500
ΔT	0.3	0.15	0.075	0.03	0.015	0.0075	0.003
Ohne Teilchenbewegung	0.091436	0.092858	0.093407	0.093694	0.093783	0.093827	0.093853
Bird	0.004078	0.003636	0.004464	0.004272	0.00344	0.004407	0.004243
Belotserkowskiy-Yanitskiy	0.005145	0.003241	0.002901	0.003277	0.003731	0.003417	0.002981
Deshepande	0.003011	0.004381	0.002605	0.004224	0.004526	0.003951	0.003582
Nanbu	0.194769	0.008987	0.008065	0.006621	0.005898	0.00293	0.007559
Babovsky	0.022464	0.009874	0.01031	0.009615	0.007202	0.009045	0.0042
LD (Halton)	0.352129	0.052066	0.119499	0.063676	0.001342	0.001082	0.001156
LD unsortiert	0.246543	0.053708	0.025109	0.054597	0.139348	0.13669	0.173147
LD Halton(Babovsky)	0.019331	0.003955	0.021217	0.01165	0.003113	0.00856	0.006184
LD Hammersley	0.492513	0.017801	0.004173	0.006455	0.014326	0.016539	0.028689
LD (0,s)-Folge	0.262401	0.122545	0.166182	0.208825	0.001829	0.001995	0.002364
Lécot's QMC1	0.008164	0.003515	0.001817	0.001343	0.001533	0.001465	0.00187
QMC1 mit Halton Folge	0.025437	0.013804	0.005381	0.002309	0.001536	0.001363	0.001235
QMC1 unsortiert	0.022586	0.032759	0.091806	0.044461	0.038563	0.060492	0.124197
QMC1 mit Zufallszahlen	0.01241	0.003147	0.004656	0.006032	0.00339	0.00536	0.004172
Lécot / Khettabi's QMC3	0.008822	0.006554	0.002657	0.004286	0.004991	0.003793	0.004757
QMC3 mit Halton Folge	0.012505	0.005479	0.004587	0.003098	0.003015	0.003139	0.002303
QMC3 unsortiert	0.094621	0.095513	0.035135	0.070481	0.110167	0.05203	0.014171
QMC3 mit Zufallszahlen	0.015869	0.002948	0.003769	0.005287	0.003135	0.003258	0.003827
QMC3 nach $\|v\|$ sortiert	0.008121	0.004391	0.00618	0.003474	0.00312	0.003086	0.003164

Tabelle 5.4: Fehler für verschiedene Δt, Krook / Wu's Lösung, $N = 29282$

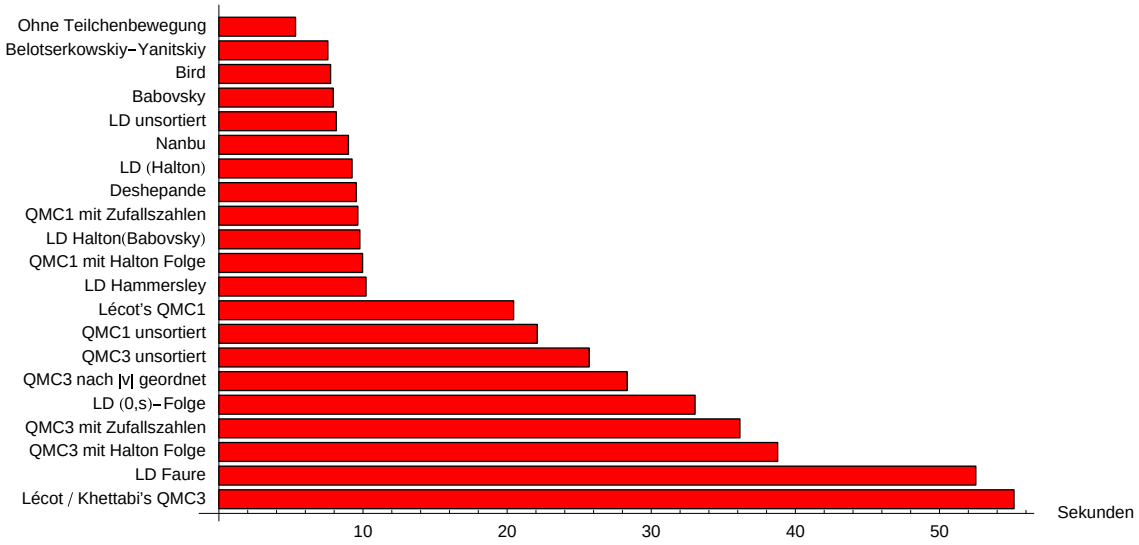

Abbildung 5.3: Zeitaufwand der Schemata mit Krook / Wu's Lösung, $N = 2662$, $\Delta t = 0.015$

	242	686	2662	6750	29282
Ohne Teilchenbewegung	0.38	1.38	5.32	12.94	51.18
Bird	0.59	1.65	7.75	26.44	274.02
Belotserkowskiy-Yanitskiy	0.35	1.65	7.56	27.07	268.86
Deshepande	0.32	1.58	9.53	27.13	267.05
Nanbu	0.55	1.88	8.99	18.29	70.02
Babovsky	0.53	1.98	7.94	19.08	73.06
LD (Halton)	0.61	2.26	9.24	24.66	101.15
LD unsortiert	0.59	2.03	8.15	20.82	81.65
LD Halton(Babovsky)	0.79	2.13	9.78	24.15	105.51
LD Hammersley	0.98	2.29	10.21	24.11	97.3
LD (0,s)-Folge	2.03	9.32	33.04	73.17	462.19
Lécot's QMC1	1.51	7.5	20.45	53.94	263.58
QMC1 mit Halton Folge	0.66	2.38	9.97	23.72	104.04
QMC1 unsortiert	1.6	10.62	22.09	53.59	244.85
QMC1 mit Zufallszahlen	0.74	2.3	9.64	23.41	96.5
Lécot / Khettabi's QMC3	2.44	13.15	55.17	191.35	2990.73
QMC3 mit Halton Folge	1.22	5.73	38.77	148.53	2722.47
QMC3 unsortiert	1.99	8.36	25.69	62.08	304.31
QMC3 mit Zufallszahlen	1.29	5.36	36.15	149.91	2825.02
QMC3 nach $\|v\|$ sortiert	2.22	9.27	28.33	65.89	356.51

Tabelle 5.5: Zeitaufwand der diversen Methoden bei $M = 100$, $T = 1.5$ und Krook / Wu's Lösung

Die oben beschriebene Verbesserung durch QMC Methoden zeigt sich auch deutlich im Fehlerexponenten $p_N^{(n)}$, der definiert ist als:

$$p_N^{(n)} = -\frac{\ln\left(\tilde{D}_N^*\left(\mathbf{V}^{(n)}, f^{(n)}\right)\right)}{\ln N} \tag{5.3}$$

Abbildung 5.4 und Tabelle 5.3 zeigen diesen Fehlerexponent für $\Delta t = 0.015$ und Krook/Wu's Lösung der Boltzmann-Gleichung.

Anhand der Gleichgewichts- (Boltzmann-) Verteilung[2] können die Fluktuationen der einzelnen Schemata untersucht werden. Auch hier zeigt sich, dass QMC Verfahren im Allgemeinen besser abschneiden als herkömmliche (Abbildungen 5.5 und 5.6 sowie Tabelle 5.6). Dies ist natürlich auf die guten Gleichverteilungseigenschaften der Folgen kleiner Diskrepanz zurückzuführen.

	242	686	2662	6750	29282
Ohne Teilchenbewegung	0.002067	0.00073	0.000189	0.000075	0.000018
Bird	0.029132	0.033451	0.015653	0.005037	0.003966
Belotserkowskiy-Yanitskiy	0.045223	0.026245	0.011798	0.005451	0.003737
Deshepande	0.049099	0.036942	0.015009	0.00883	0.004707
Nanbu	0.077069	0.027613	0.033491	0.017337	0.005921
Babovsky	0.242466	0.033703	0.023659	0.030156	0.006999
LD (Halton)	0.019981	0.0142	0.004722	0.003244	0.001295
LD unsortiert	0.209732	0.041801	0.131701	0.088507	0.134628
LD Halton(Babovsky)	0.019981	0.0142	0.004722	0.003244	0.001295
LD Hammersley	0.205402	0.088877	0.034317	0.0212	0.015566
LD (0,s)-Folge	0.033974	0.015383	0.008492	0.24945	0.001634
Lécot's QMC1	0.026779	0.021747	0.005536	0.003206	0.001455
QMC1 mit Halton Folge	0.017977	0.010772	0.007115	0.00285	0.001767
QMC1 unsortiert	0.118639	0.122217	0.046506	0.046311	0.029988
QMC1 mit Zufallszahlen	0.047724	0.032317	0.014136	0.009241	0.003446
Lécot / Khettabi's QMC3	0.026539	0.077077	0.01612	0.01491	0.004203
QMC3 mit Halton Folge	0.034634	0.018088	0.010144	0.006746	0.003288
QMC3 unsortiert	0.049275	0.287256	0.234978	0.057689	0.104901
QMC3 mit Zufallszahlen	0.067622	0.016107	0.0139	0.004981	0.004124
QMC3 nach $\|v\|$ sortiert	0.044143	0.034103	0.012496	0.098738	0.00407

Tabelle 5.6: Fluktuationen für die Boltzmann-Verteilung und $\Delta t = 0.015$

[2]Die stationäre Lösung der Boltzmann-Gleichung ist die Boltzmann-Verteilung (3.32):

$$f(v,t) = \frac{1}{2\pi^{3/2}} e^{-\frac{v^2}{2}} \tag{5.4}$$

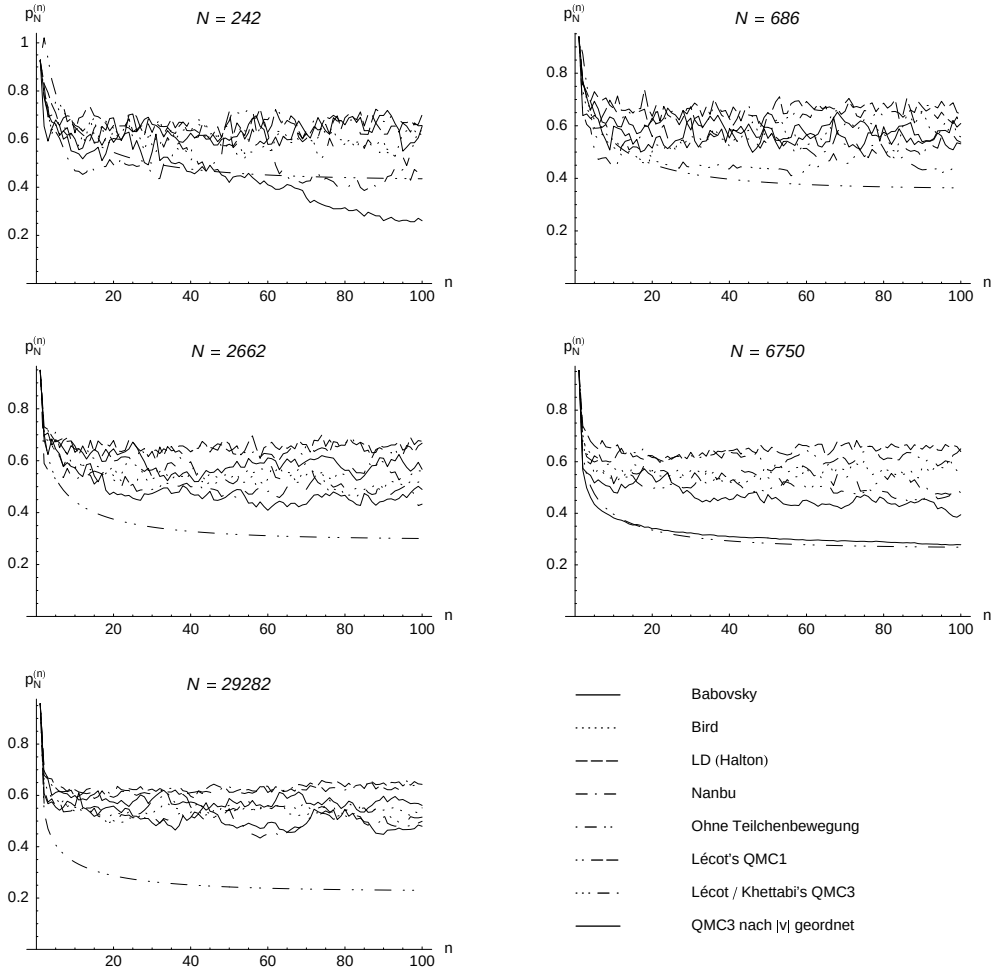

Abbildung 5.4: Fehlerexponent für Krook / Wu's Lösung, $\Delta t = 0.015$

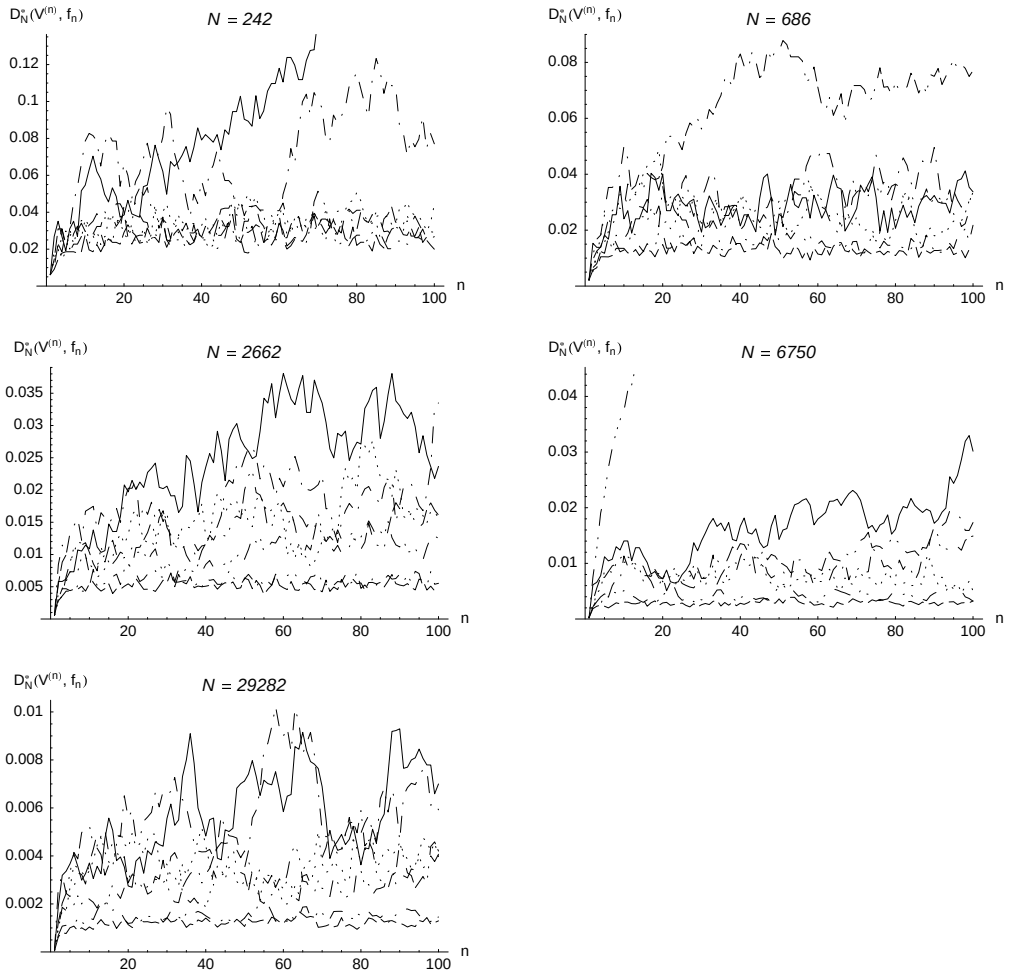

Abbildung 5.5: Fluktuationen der verschiedenen Schemata für die Boltzmann-Verteilung und $\Delta t = 0.015$

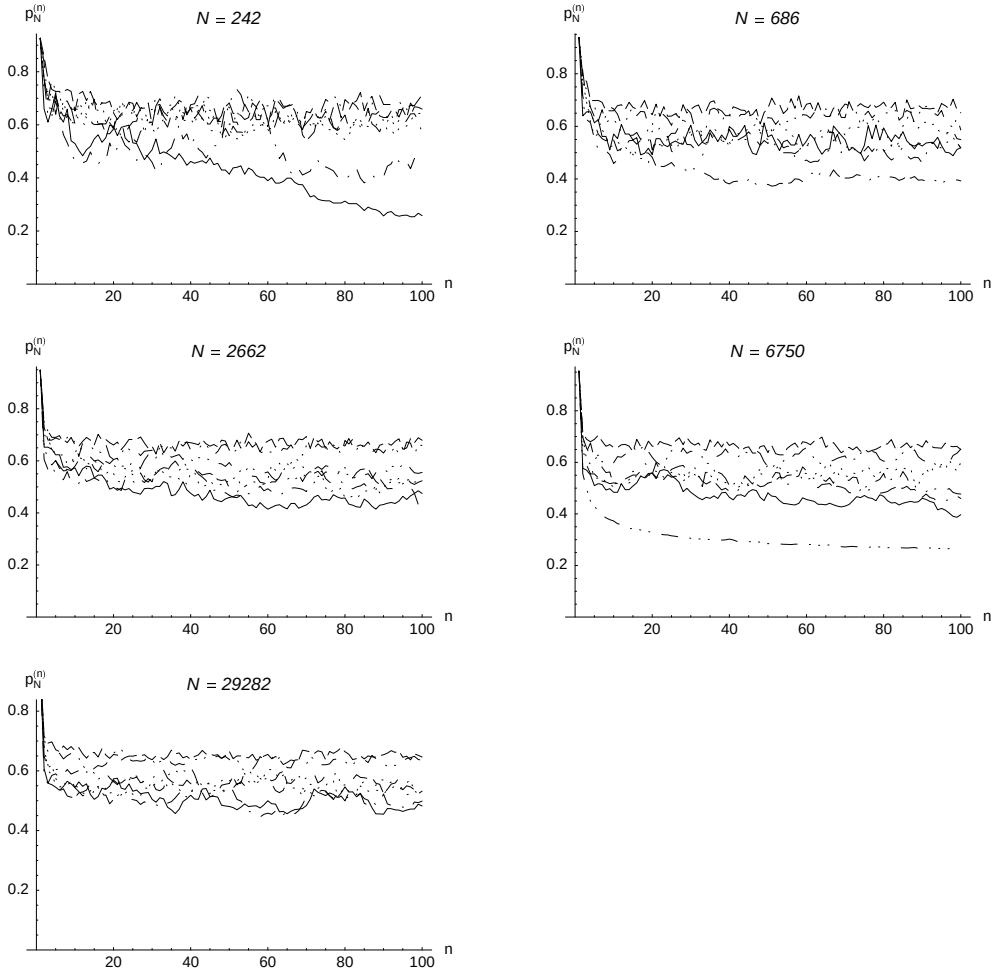

Abbildung 5.6: Fehlerexponent der verschiedenen Schemata für die Boltzmann-Verteilung und $\Delta t = 0.015$

5.3 Zahl der Teilchenkollisionen

Wie Nanbu in [Nan83] beschreibt, soll die Zahl X der Kollisionen pro Zeitschritt (die ja prinzipiell eine Zufallsvariable darstellt sofern Zufallszahlen benutzt werden) im Mittel

$$E(X) = \rho(t)\Delta t, \quad \rho(t) = \frac{n_c}{N} \sum_{\substack{i,j=1 \\ i<j}}^{N} g_{ij}\sigma_T\left(g_{ij}\right)$$

betragen. In den von uns betrachteten Fällen der Boltzmann-Verteilung von Krook / Wu's Lösung, die beide Maxwell'sche Moleküle benutzen, ist $\rho(t) = const.$ und damit $E(X) = const.$ Nur Methoden, die bei einer Kollision nicht beide Geschwindigkeiten, sondern lediglich die eines einzigen Teilchens ändern (DSMC-N, Babovsky, LD, ...), sollten doppelt so viele Zusammenstöße aufweisen, damit gleich viele Teilchen ihre Geschwindigkeit ändern.

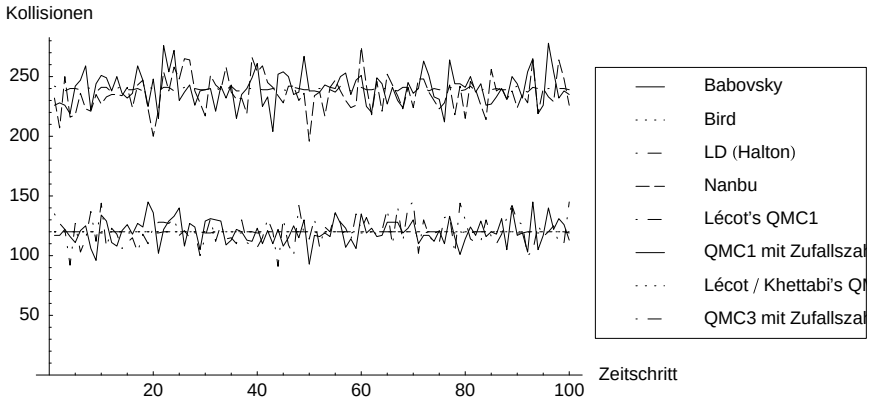

Abbildung 5.7: Zahl der Teilchenkollisionen der wichtigsten Schemata, $N = 2662$

Abbildung 5.7 zeigt den zeitlichen Verlauf der Zusammenstöße der wichtigsten Schemata bei 2662 Teilchen. Sehr schön zu sehen ist hier, dass Methoden, die für jedes Teilchen eine (Pseudo-)Zufallsvariable benutzen (also DSMC-N, Babovsky, QMC1mc, QMC3mc), wesentlich stärkere Schwankungen in der Kollisionszahl aufweisen als Methoden, die Folgen kleiner Diskrepanz benutzen (LD, QMC1, QMC3). Dies ist natürlich bedingt durch die hohe Gleichverteilung der Folgen kleiner Diskrepanz.

5.4 3-dimensionale vs. 1-dimensionale Simulation

Der Vorteil der Methoden, die 3-dimensionale Geschwindigkeiten zur Simulation benutzen, liegt vor allem darin, dass diese relativ leicht verallgemeinert werden können auf den Fall nichtisotroper Streuung ($\sigma(\vartheta, \mathbf{g}_{ij})$ nicht unabhängig von *theta* und der Richtung von $\mathbf{g}_{ij}$). Ein Konvergenz-Beweis oder eine Diskrepanzschranke ist dann aber relativ schwierig zu finden.

Theoretisch sollte es keine großen Unterschiede zwischen den Methoden geben, die 3-dimensionale Geschwindigkeiten benutzen[3], und jenen, die nur mit dem Absolutbetrag der Geschwindigkeit arbeiten[4], da alle dieselbe Teilchenkinetik benutzen.

[3] Bird, B-Y, Deshepande und die QMC3 Methoden
[4] Nanbu, Babovsky, die LD und QMC1 Methoden

Auch experimentell lassen sich in dieser Hinsicht keine gröberen Unterschiede feststellen, die nicht auf die Methode selbst zurückgeführt werden können. Lediglich die QMC3 Methode weist im Vergleich zum QMC1 Schema eine Verschlechterung des Fehlers um etwa einen Faktor 2 auf. Dies kann aber zurückgeführt werden auf die eher ungünstige Wahl der Umsortierung nach Gleichung (4.87).

5.5 Bedeutung des Umordnens der Teilchen

Anhand der QMC Schemata, die ja als wesentlichen Bestandteil das Umordnen der simulierten Teilchen beinhalten, kann sehr schön die numerische Bedeutung dieses Schrittes demonstriert werden. Da die Teilchenzahl bei diesen Schemata typischerweise eine Potenz der benutzten Basis (sofern $(0, s)$-Folgen, bzw. eigentlich Netze für jeden Zeitschritt verwendet werden) oder ein Vielfaches davon betragen, würden natürlich sehr starke Korrelationen für jedes Teilchen zwischen den einzelnen Zeitschritten auftreten. Diese sollen durch die Umordnung gebrochen werden.

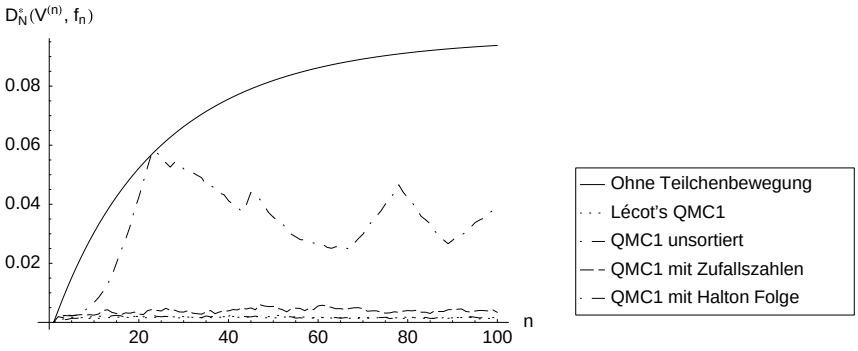

Abbildung 5.8: Verbesserung des Fehlers durch Umordnen, QMC1, $N = 29282$

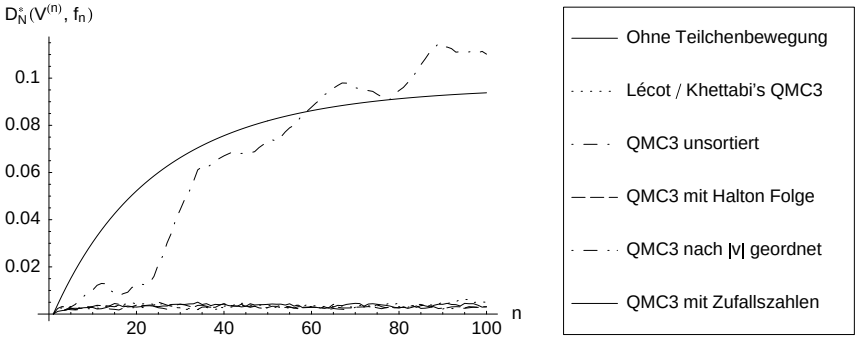

Abbildung 5.9: Verbesserung des Fehlers durch Umordnen, QMC3, $N = 29282$

Die Abbildungen 5.8 und 5.9 zeigen einen Vergleich des Fehlers für die QMC1 und QMC3 Methoden, wobei einmal die Teilchen nicht umsortiert werden, und einmal Zufallszahlen statt $(0, s)$-Folgen benutzt werden. Sehr deutlich wird klar, dass die Ergebnisse ohne Umsortieren praktisch unbrauchbar sind. Außerdem liefern die Methoden mit Zufallszahlen doch signifikant schlechtere Ergebnisse (Tabelle 5.7).

Insbesondere am Beispiel der QMC3 Methode lässt sich die Signifikanz der richtigen Umsortier-Bedingung erkennen. Während die von LÉCOT und KHETTABI in [LK99] vorgeschlagene Bedingung für das Umordnen aus Gleichung (4.87) erkennbar schlechtere Ergebnisse als QMC1 liefert, kann durch ein Umsortieren entsprechend dem Absolutbetrag der Geschwindigkeit das Ergebnis in einigen Fällen stark verbessert werden auf in etwa denselben Wert wie bei QMC1 (siehe auch Tabelle 5.8). Dies ist allerdings nicht immer der Fall, wie etwa in Abbildung 5.2 für $N = 6750$, $M = 100$ zu erkennen ist.

5.6 MC vs. QMC Simulation

Wie in den letzten Abschnitten immer wieder betont, liefern die Quasi-Monte Carlo Schemata im Allgemeinen bessere Ergebnisse als jene Simulations-Schemata, die Zufallszahlen benutzen. Allerdings können natürlich aufgrund der hohen Gleichverteilung der Folgen kleiner Diskrepanz starke Korrelationen mit den Simulations-Schemata auftreten. Speziell im Fall des LD Schemas, wobei aber BABOVSKY's Anordnung $\left[\frac{j}{N} - P_{ij}, \frac{j}{N}\right)$ der Intervalle nach Gleichung (4.30) benutzt wird, tritt für 686 Teilchen und der Verwendung der Halton-Folge in den Basen $(2, 3, 7, 5)$ dieses Problem auf. Es fallen dann entweder alle Quasi-Zufallszahlen in die entsprechenden Intervalle, oder gar keine. Damit sieht die Zahl der Kollisionen in den einzelnen Zeitschritten folgendermaßen aus: 342, 0, 0, 343, 0, 0, 1, 342, 0, 0, 0, 0, 0, 0, 0, 343, 0, 0, 1, 342, 0, 0, 0, 0, 343, 0, 0, 1, 342, 0, 0, 343, 0, 0, 0, 0, 0, 0, 0, 1, 342, 0, 0, 343, 0, 0, 0, 0, 1, 342, 0, 0, 343, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 343, 0, 0, 1, 342, 0, 0, 0, 0, 343, 0, 0, 1, 342, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 342, 0, 0, 343, 0, 0, 0, 0, 1, 342, 0, ...

5.7 Direkte Vergleiche von Halton- und $(0, s)$-Folgen

Da die direkte Berechnung der Diskrepanz für höherdimensionale Folgen nicht möglich ist, können die Diskrepanzschranken aus Kapitel 2 lediglich exemplarisch an ausgewählten Testfunktionen verifiziert, aber natürlich niemals bewiesen werden. Zum einen möchte ich hier nochmals die Simulation der Boltzmann-Gleichung zum direkten Vergleich von Halton- und $(0, s)$-Folgen heranziehen. Zum anderen werde ich anhand der in [Rad95] erwähnten Funktionen der Form

$$f(\mathbf{x}) = \prod_{k=1}^{s} \frac{\left|4x^{(k)} - 2\right| + a_k}{1 + a_k} \tag{5.5}$$

die Ergebnisse für bis zu 50 dimensionale Folgen vergleichen.

Bei der Simulation der Boltzmann-Gleichung wurde sowohl bei Nanbu's DSMC-N Schema, als auch beim QMC1 und beim QMC3 Schema die Simulation jeweils mit Pseudo-Zufallszahlen, mit einer $(0, s)$-Folge und mit der Halton-Folge durchgeführt. Während Lécot's LD Schema als QMC Variante von Nanbu's DSMC-N Methode definiert wurde und beliebige Folgen kleiner Diskrepanz benutzt, sind die QMC1 une QMC3 Schemata speziell auf (t, s) - Folgen ausgelegt, die garantieren, dass jedes Teilchen genau einmal betrachtet wird.

Vergleicht man die Ergebnisse (siehe Tabelle 5.4), so lässt sich im Grunde kein Unterschied zwischen der Verwendung der Halton-Folge und der $(0, s)$-Folge erkennen (Abbildung 5.10).

Bei Testfunktionen der Form (5.5) gilt allgemein die Abschätzung

$$\prod_{i=1}^{s} \frac{a_i}{1 + a_i} \leq f(\mathbf{x}) = \prod_{i=1}^{s} \frac{\left|4x_i - 2\right| + a_i}{1 + a_i} \leq \prod_{i=1}^{s} \frac{2 + a_i}{1 + a_i} \,.$$

	5	10	20	50	100	200	500
Lécot's QMC1	0.012167	0.008502	0.008643	0.008466	0.00517	0.006823	0.005652
QMC1 mit Halton Folge	0.021095	0.016063	0.010514	0.006165	0.007622	0.006274	0.005766
QMC1 mit Zufallszahlen	0.014068	0.013952	0.014715	0.014206	0.013898	0.017176	0.012053
QMC1 unsortiert	0.072607	0.094463	0.060102	0.06249	0.045781	0.026213	0.114155

Tabelle 5.7: Verbesserung durch QMC Methoden beim QMC1 Schema, Krook / Wu's Lösung, $N = 2662$

	5	10	20	50	100	200	500
Lécot / Khettabi's QMC3	0.015456	0.009991	0.011525	0.014087	0.012058	0.008904	0.019582
QMC3 mit Halton Folge	0.015036	0.014592	0.010112	0.01294	0.009928	0.012848	0.011485
QMC3 mit Zufallszahlen	0.023327	0.012513	0.023556	0.011101	0.009949	0.019386	0.011823
QMC3 nach $\|v\|$ sortiert	0.011173	0.01406	0.008583	0.008834	0.011432	0.009509	0.010982
QMC3 unsortiert	0.071517	0.047514	0.051145	0.016386	0.242699	0.037491	0.139235

Tabelle 5.8: Vergleich der QMC und MC Methoden beim QMC3 Schema, Krook / Wu's Lösung, $N = 2662$

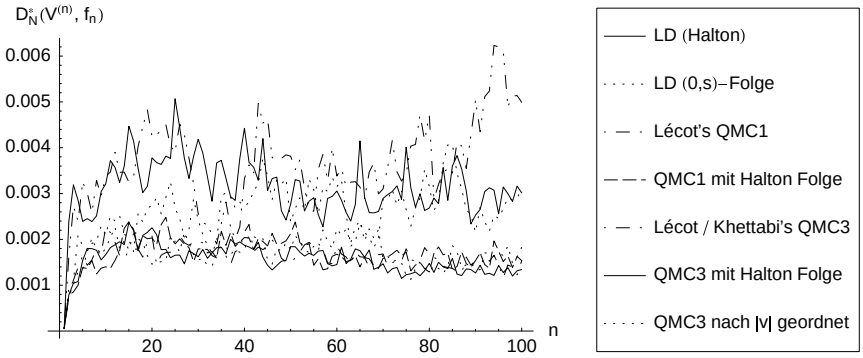

Abbildung 5.10: Vergleich der Simulation mit Halton- und $(0, s)$-Folge, $N = 29282$

Außerdem gilt $\int_{\bar{I}^s} f(\mathbf{x})d\mathbf{x} = 1$ für jede Wahl der a_i.

Im Folgenden werden nur die beiden Fälle $a_i = 0.01$ und $a_i = i^2$ betrachtet. Für $a_i = 0.01$ strebt die obere Schranke mit s exponentiell gegen ∞:

$$f(\mathbf{x}) \leq \prod_{i=1}^{s} \frac{2 + 0.01}{1 + 0.01} = 1.99^s$$

Hingegen gilt für $a_i = i^s$

$$f(\mathbf{x}) \leq \prod_{i=1}^{s} \frac{2 + i^s}{1 + i^2} \xrightarrow{s\to\infty} \tilde{X} < \infty$$

Für den ersten Fall kann also sehr leicht ein beträchtlicher Fehler entstehen, während er im zweiten Fall für beliebige Dimensionen s beschränkt ist. Tabelle 5.9 zeigt den Vergleich der Fehler $\sum_{i=1}^{N} f(\mathbf{x}^{(i)}) - \int_{\bar{I}^s} f(\mathbf{x})d\mathbf{x}$ dieser beiden Fälle für steigende Dimensionen s unter Verwendung der Halton sowie der $(0, s)$-Folge.

Es liefert mit steigendem s zunehmend die Halton-Folge bessere Ergebnisse als $(0, s)$-Folgen.

Als weitere Testfunktionen wurden folgende Funktionen gewählt:

- $g(\mathbf{x}) = \prod_{i=1}^{s} \sin(2\pi n_i x_i)$ mit $-1 \leq g(\mathbf{x}) \leq 1$ und $\int_{\bar{I}^s} g(\mathbf{x})d\mathbf{x} = 0$
- $h(\mathbf{x}) = e^{-\mathbf{x}^2}$ mit $\int_{\bar{I}^s} h(\mathbf{x})d\mathbf{x} = 0.7468241328^s$

Die entsprechenden Ergebnisse sind in der Tabelle 5.10 dargestellt.

Auch hier zeigt sich eine bessere Performance der Halton-Folge gegenüber der benutzten $(0, s)$-Folge. Allerdings sind diese Ergebnisse natuerlich nur punktuelle Daten, sodass sie lediglich Hinweise darauf geben können, dass die Halton-Folge der $(0, s)$-Folge keineswegs nachsteht.

Zusammenfassend lässt sich damit sagen, dass die Halton-Folge keineswegs so schlechte Ergebnisse liefert, wie bisher stillschweigend angenommen aufgrund der schlechten bekannten Diskrepanzabschätzung. ATANASSOV's neue Diskrepanzabschätzung löste dieses Problem theoretisch, die numerischen Experimente in diesem Abschnitt scheinen dieses Ergebnis auch zu bestätigen.

	$a_i = 0.05$				$a_i = i^2$			
	Halton	Atanassov	(t,s)-Folge	MC	Halton	Atanassov	(t,s)-Folge	MC
1000	−186.94	−186.936	−832.88	515.81	0.030534	0.028249	1.13212	30.1517
5000	−2444.93	−2444.93	−4285.11	−198.793	−0.328966	−0.239891	6.65979	20.4776
10000	903.695	903.695	−8298.29	−306.171	−0.173687	0.050744	8.44931	51.9149
50000	−9417.09	−9417.09	−19723.1	−5483.72	−0.109098	−0.038789	14.2814	34.6007
100000	−14981.	−14981.	−33973.8	3910.39	−0.159641	−0.138066	11.712	19.8693
200000	−18877.1	−18876.9	−63751.6	−5328.91	−0.365416	0.040656	16.8611	−3.33942
300000	−33698.4	−33696.1	−83483.2	−14624.2	−0.029827	−0.091732	−5.77092	−158.156
400000	−10248.	−10247.9	−88239.5	−19684.1	−0.374003	−0.157472	3.77402	−207.526
499000	−10756.7	−10756.7	−105836.	−8102.15	−0.447281	−0.206761	4.61365	−132.585

Tabelle 5.9: Fehler durch (Q)MC Integration der Testfunktion, $N = 10000$ (Quasi-) Zufallszahlen

	$g(\mathbf{x})$				$h(\mathbf{x})$			
	Halton	Atanassov	(t,s)-Folge	MC	Halton	Atanassov	(t,s)-Folge	MC
1	0.	0.	0.	0.	−0.000708	−0.000783	0.0045	−0.000732
1000	−0.007638	−0.007638	−0.002823	0.002788	0.10658	0.10825	0.008848	−0.050186
5000	0.050213	0.050213	0.004138	−0.003599	0.215783	0.215742	−0.10877	0.054134
10000	0.028653	0.028657	0.014721	−0.008116	0.270354	0.270302	−0.138864	0.156867
50000	0.046409	0.046409	0.112089	0.003552	0.378528	0.378608	−0.555445	0.592333
100000	−0.035151	−0.035151	0.103624	0.062128	0.341933	0.342307	−1.10646	0.84282
200000	−0.00089	−0.000889	0.072918	0.034575	0.278002	0.278243	−1.92202	0.74386
300000	−0.01269	−0.01269	0.116595	−0.079523	0.45064	0.450814	−2.26856	2.17072
400000	−0.069152	−0.069152	0.116985	−0.210129	1.06455	1.06444	−2.77342	2.27072
499000	−0.076274	−0.076274	0.058882	−0.106872	0.905093	0.90618	−3.13993	1.57809

Tabelle 5.10: Fehler durch (Q)MC Integration der ausgewählter Testfunktionen, $N = 10000$ (Quasi-) Zufallszahlen

Source-Code

Main Program

```
/***************************************************************************
                          main.cpp  -  description
    copyright            : (C) 1999 by Reinhold Kainhofer
      email                : reinhold@wolfram.com
 ** ************************************************************************/

#include <stdio.h>
#include <math.h>
#include <iostream.h>
#include "BoltzmannSimulation.h"
#include "SimSeqs.h"

// whether all the (Q)MC numbers should be created before simulation
// starts or just each one when it is needed
//#define PRECREATE_NUMBERS

// whether the values should be read in or just default
//should be used (for debugging -> less work if you
//don't have to type in everything every time)
#define INTERACTIVE
void IntroData(double &DeltaT, long &iter, char*file, char*outfile,
      long &fstep, long&methods, long&base, long*d){   double T=1.5;
  cout<<"SIMULATION der Boltzmanngleichung\n";
  cout<<"Diplomarbeit in Technischer Mathematik (Stzw: Technomathematik )\n";
  cout<<"R. Kainhofer, TU Graz, 9613474\n\n\n";
  cout<<"Die Anzahl der Teilchen wird bestimmt durch die Anzahl der"<<
      " vorgegebenen Geschwindigkeiten.\n";
#ifdef INTERACTIVE
  cout<<"Bis zu welchem Zeitpunkt ömchten Sie die Gleichung simulieren ["<<
        T<<"]: T = ";
      cin>>T;
  cout<<"Wie viele Iterationen sollen durchgefuehrt werden ["<<
        iter<<"]: iter = ";
      cin>>iter;
  cout<<"Datei mit den Anfangsbedingungen ["<<file<<"]: filename = ";
      cin>>file;
  cout<<"In welche Datei sollen die Daten ausgegeben werden ["<<
      outfile<<"]: outfile = ";
      cin>>outfile;
  cout<<"Step for the Faure-Sequence ["<<fstep<<"]: fstep = ";
      cin>>fstep;
  cout<<"\n\n Welche Methoden sollen üdurchgefhrt werden:\n";
  cout<<BM_NONE<<"... keine Teilchenbewegung\n";
  cout<<BM_BIRD<<"... Birds DSMC-B\n";
  cout<<BM_BY<<"... Belotserkovskiy-Yanitskiy\n";
  cout<<BM_DESHEPANDE<<"... Deshepande\n";
  cout<<BM_NANBU<<"... Nanbu\n";
  cout<<BM_BABOVSKY<<"... Babovsky's Modifikation von DSMC-N\n";
  cout<<BM_LDH<<"... Lecot's LD (Halton-Folge)\n";
  cout<<BM_LDUNS<<"... LD ohne Umordnen\n";
  cout<<BM_LDHB<<"... LD mit Babobsky\n";
  cout<<BM_LDHM<<"... LD mit Hammersley-Menge\n";
  cout<<BM_LDF<<"... LD mit Faure Folge\n";

  cout<<BM_LDTS<<"... LD mit (t,s)-Folge\n";
  cout<<BM_QMC1<<"... Lecot's QMC1\n";
  cout<<BM_QMC1HALTON<<"... QMC1 mit Halton-Folge\n";
  cout<<BM_QMC1UNS<<"... QMC1 ohne Umordnen\n";
  cout<<BM_QMC1RAND<<"... QMC1 mit Zufallszahlen\n";
  cout<<BM_QMC3<<"... Lecot's QMC3 mit (0,9)-Folge\n";
  cout<<BM_QMC3HALTON<<"... QMC3 mit Halton-Folge\n";
  cout<<BM_QMC3UNS<<"... QMC3 ohne Umordnen\n";
  cout<<BM_QMC3MC<<"... QMC3 mit Zufallszahlen\n";
  cout<<BM_QMC3VABS<<"... QMC3 mit Vabs sortiert\n";
  cout<<"methods [-1=alle]: ";
      cin>>methods;

  cout<<"Base for the LD Sequences in the QMC and QMC3 methods ["<<
      base<<"]: base = ";
      cin>>base;
  cout<<"Exponents for the QMC3-method (partnr=base^(d1+d2+d3)):\n";
  cout<<"d1 = ["<<d[0]<<"] = ";
```

```
      cin>>d[0];
   cout<<"d2 = ["<<d[1]<<"] = ";
      cin>>d[1];
   cout<<"d3 = ["<<d[2]<<"] = ";
      cin>>d[2];
#endif

   cout<<"\n\n";
   DeltaT=T/iter;
   cout<<"DeltaT = "<<DeltaT<<"      M = "<<iter<<"\n";
}

void MainSimulation(BoltzmannSimBase* sim, SimulationSeqsBase* seq=NULL,
     bool sort=false) {
   sim->SetSequence(seq, true, sort);
   sim->DoSimulation();
   free(sim);
}

int main(){
   char file[150]="indata/Krook_p242", outfile[150]="outdata/outKrk242";
   double DeltaT=.05;
   long iter=100, fstep=0/*3*/, methods=8192;//0x0400;//0xFFFF;
   long dd[3]={1,1,0};
   long bases[9]={2,3,7,5,11,13,17,19,23};
   long base=11;
   BoltzmannSimBase* sim;

   IntroData(DeltaT, iter, file, outfile,fstep, methods, base, &dd[0]);

   if (methods & BM_NONE)
     MainSimulation( (BoltzmannSimBase*)new NoBoltzmannSim(file, outfile,
         iter, DeltaT, fstep), NULL, false);

   if (methods & BM_BIRD)
     MainSimulation( (BoltzmannSimBase*)new Bird(file, outfile,
         iter, DeltaT, fstep), NULL, false);
   if (methods & BM_BY)
     MainSimulation( (BoltzmannSimBase*)new BY(file, outfile,
         iter, DeltaT, fstep), NULL, false);
   if (methods & BM_DESHEPANDE)
     MainSimulation( (BoltzmannSimBase*)new Deshepande(file, outfile,
         iter, DeltaT, fstep), NULL, false);

   if (methods & BM_NANBU)
     MainSimulation( (BoltzmannSimBase*)new Nanbu(file, outfile,
         iter, DeltaT, fstep), NULL, false);
   if (methods & BM_BABOVSKY)
     MainSimulation( (BoltzmannSimBase*)new Babovsky(file, outfile,
         iter, DeltaT, fstep), NULL, false);

   if (methods & BM_LDH)
     MainSimulation( (BoltzmannSimBase*)new LD(file, outfile,
         iter, DeltaT, fstep), NULL, true);
   if (methods & BM_LDUNS)
     MainSimulation( (BoltzmannSimBase*)new LD(file, outfile,
         iter, DeltaT, fstep, "lduns", "LD unsorted"), NULL, false);
   if (methods & BM_LDHB) {
     sim=(BoltzmannSimBase*)new LD(file, outfile,
         iter, DeltaT, fstep, "ldhb", "LD Halton(Babovsky)");
     ((Nanbu*)sim)->babov=true;
     sim->DoSimulation();
     free(sim);
   }

   if (methods & BM_LDHM) {
     sim=(BoltzmannSimBase*)new LD(file, outfile,
         iter, DeltaT, fstep, "ldhm", "LD Hammersley");
     sim->ReadV();
     // The hammersley set needs to get the particle number,
     // not the whole number of needed QR numbers
     sim->SetSequence((SimulationSeqsBase*) new Hammersley(&bases[0],
         4, sim->GetPartNr(), BGENAU, GENAU, 3, "ldhm", "LD Hammersley"),
         true, true);
     sim->DoSimulation();
     free(sim);
   }
   if (methods & BM_LDF) {
     sim=(BoltzmannSimBase*)new LD(file, outfile,
         iter, DeltaT, fstep, "ldf", "LD Faure");
     sim->ReadV();
     sim->SetSequence((SimulationSeqsBase*) new Faure(&bases[0],
         4, sim->GetPartNr()*iter, BGENAU, GENAU, 2, "ldf", "LD Faure"),
         true,true);
     sim->DoSimulation();
     free(sim);
   }
   if (methods & BM_LDTS) {
     sim=(BoltzmannSimBase*)new LD(file, outfile,
         iter, DeltaT, fstep, "ldts", "LD (0,s)-Folge");
     sim->ReadV();
     bases[2]=4;
     sim->SetSequence((SimulationSeqsBase*) new Netz(&bases[0],
         base, 4, sim->GetPartNr()*iter, 3, "ldts", "LD (0,s)-Folge"),
         true, true);
```

```
    sim->DoSimulation();
    free(sim);
  }

  if (methods & BM_QMC1)
    MainSimulation( (BoltzmannSimBase*)new QMC1(file, outfile,
        iter, DeltaT, fstep, base), NULL, true);
  if (methods & BM_QMC1HALTON) {
    sim=(BoltzmannSimBase*)new QMC1(file, outfile,
        iter, DeltaT, fstep, base, "qmc1hal", "QMC1 mit Halton Folge");
    sim->ReadV();
    bases[0]=2;
    bases[1]=3;
    bases[2]=5;
    bases[3]=7;
    bases[4]=11;
    sim->SetSequence((SimulationSeqsBase*) new Halton(&bases[0],
        5, sim->GetPartNr()*iter, BGENAU, GENAU, "qmc1hal", "QMC1 mit Halton Folge"),
        true, true);
    sim->DoSimulation();
    free(sim);
  }

  if (methods & BM_QMC1RAND)
    MainSimulation( (BoltzmannSimBase*)new QMC1(file, outfile,
        iter, DeltaT, fstep, base,"qmc1mc", "QMC1 random"),
        (SimulationSeqsBase*) new MonteCarlo("qmc1mc",
        "QMC1 mit Zufallszahlen"), true);
  if (methods & BM_QMC1UNS)
    MainSimulation( (BoltzmannSimBase*)new QMC1(file, outfile,
        iter, DeltaT, fstep, base, "qmc1uns", "QMC1 unsorted"),
        NULL, false);
  if (methods & BM_QMC3)
    MainSimulation( (BoltzmannSimBase*)new QMC3(file, outfile,
        iter, DeltaT, fstep, dd, base), NULL, true);
  if (methods & BM_QMC3HALTON) {
    sim=(BoltzmannSimBase*)new QMC3(file, outfile,
        iter, DeltaT, fstep, dd, base, "qmc3hal", "QMC3 mit Halton Folge");
    sim->ReadV();
    bases[0]=2;
    bases[1]=3;
    bases[2]=5;
    bases[3]=7;
    bases[4]=11;
    bases[5]=13;
    bases[6]=17;
    bases[7]=19;
    bases[8]=23;
    sim->SetSequence((SimulationSeqsBase*) new Halton(&bases[0],
        9, sim->GetPartNr()*iter, BGENAU, GENAU, "qmc3hal", "QMC3 mit Halton Folge"),
        true, true);
    sim->DoSimulation();
    free(sim);
  }
  if (methods & BM_QMC3UNS)
    MainSimulation( (BoltzmannSimBase*)new QMC3(file, outfile,
        iter, DeltaT, fstep, dd, base, "qmc3uns", "QMC3 unsorted"),
        NULL, false);
  if (methods & BM_QMC3MC)
    MainSimulation( (BoltzmannSimBase*)new QMC3(file, outfile,
        iter, DeltaT, fstep, dd, base, "qmc3mc", "QMC3 random"),
        (SimulationSeqsBase*) new MonteCarlo("qmc3mc",
        "QMC3 mit Zufallszahlen"), true);
  if (methods & BM_QMC3VABS)
    MainSimulation( (BoltzmannSimBase*)new QMC3SortVabs(file, outfile,
        iter, DeltaT, fstep, dd, base), NULL, true);
  return 0;
}
```

Die Simulationsmethoden (Deklaration)

```
/***************************************************************************
                          BoltzmannSimulation - description
                             -------------------
    copyright            : (C) 1999 by Reinhold Kainhofer
    email                : reinhold@kainhofer.com
 ***************************************************************************/


#ifndef BMSIM_H
#define BMSIM_H
#include <stdlib.h>
#include "SimSeqs.h"

#define HOMOGENEOUS        // sigma does not depend on x
#define SIGMA_G_CONSTANT   // g*sigma(g)=const (save some calculations)
#define ERRORCHECK         // check if all the indices are in bounds
#define TS_LAMBDA 4        // parameter lambda for the construction of (t,s)
#define MAXWELL            // use maxwellian molecules (save calculations)

// various constants for the calculations / equations
#define Pi 3.141592654
#define kconst 3/(2*Pi)
```

```
#define qconst 6
#define ncell 1


#define BM_NONE          0x1
#define BM_BIRD          0x2
#define BM_BY            0x4
#define BM_DESHEPANDE 0x8
#define BM_NANBU         0x10
#define BM_BABOVSKY      0x20
#define BM_LDH           0x40
#define BM_LDUNS         0x80
#define BM_LDHB          0x100
#define BM_LDHM          0x200
#define BM_LDF           0x400
#define BM_LDTS          0x800
#define BM_QMC1          0x1000
#define BM_QMC1HALTON 0x2000
#define BM_QMC1UNS       0x4000
#define BM_QMC1RAND      0x8000
#define BM_QMC3          0x10000
#define BM_QMC3HALTON 0x20000
#define BM_QMC3UNS       0x40000
#define BM_QMC3MC        0x80000
#define BM_QMC3VABS      0x100000



/****************************************
Class Hierarchy is:

  BoltzmannSimBase
  |---TimeCounter
  |   |--Bird
  |   |--BY
  |      |-Deshepande
  |---Nanbu
      |--LD

****************************************/

/**This is the base class for all the Boltzmann simulation methods
   (DSMC, SS,BY, LD, QMC etc)
  *@author Reinhold Kainhofer
*/

class BoltzmannSimBase {
protected:
//public:
  SimulationSeqsBase* sequence;
  char* file;
  char* outfile;
//FILE* tempfile;
  time_t starttime; // time this scheme startet -> for profiling
  long iter;     // number of iterations = T / DeltaT
  double DeltaT;// time for each time step
  long step;     // index of the current time step
  long partnr;   // number of particles simulated
  double* V;     // 3-dim velocities
  double* Vabs; // abs. values of the velocities
  double* Vold; // temp buffer for vel.
  bool dim3;     // whether scheme uses 3-dim vel. or abs.val.
  double*buffer;// will keep the modified el. of the ld seq.
  double*buff; // will keep the el. of the ld seq.
  int bufferlen;
  char*name, *ext;
  bool sort, DelSequence;
  long base;
  long collisions;    // nr of collisions in this time step
  long nrSoFar;       // how many ld number have already been used
public:
  BoltzmannSimBase(char* fl, char*outfl, long it, double DeltT,
      long stp, char*ex="", char*nm="");
  virtual ~BoltzmannSimBase();
  virtual bool DoSimulation();
  virtual bool Simulate();
/**/  virtual bool InitSequence();
  virtual bool InitBuffer();
  virtual bool ReadV();




  virtual bool QuickSort(long l, long r);
  virtual bool QuickSort(long l, long r, long pos);
  virtual bool InsertionSort(long lo0, long hi0);
  virtual bool InsertionSort(long lo0, long hi0, long pos);
  virtual long SortV();
/**/  virtual bool PreProcess(){return true;};
/**/  virtual bool PostProcess(){return true;};
/**/  virtual bool SimulateStep(/*long&nr*/){return false;};
/**/  virtual double DoCollision(/*long nr, */double t){return 0;};
  virtual bool SetSequence(SimulationSeqsBase*sim, bool delSeq, bool srt);
  virtual bool SetSequence(SimulationSeqsBase*sim, bool delSeq) {
      return SetSequence(sim, delSeq, sort);}
```

```
  double G(long i, long j);
  long SampleHomogeneousVelocities3D(double*V, long i, long j,
      double buff2, double buff3, double*Vold, double vdist=-1);

  void GetPair(long&i, long&j, long pair);
  void SampleHomogeneousVelocities(double*V, long i, long j,
      double rand1, double rand2, double* oldV);
  void SamplePairEqualP(long&i,long&j, double randnr, long partnr);
    /** q(k) gives the ... q(k)=4*pi*k for some constant k, for now... */
  double q(double g);
    /** helper function, calculate [a,b;x4,x5] */
  double bracket (double v, double w, double x4, double x5);
    /** helper function, calculate [a,b;x4] */
  double bracket(double v, double w, double x4);
  virtual long GetPartNr(){return partnr;}
};

class NoBoltzmannSim:public BoltzmannSimBase {
public:
  NoBoltzmannSim(char* fl, char*outfl, long it, double DeltT, long stp,
      char*ex="none", char*nm="No Simulation (initial distribution)"):
      BoltzmannSimBase(fl, outfl, it, DeltT, stp, ex, nm){}
  virtual bool SimulateStep(/*long&nr*/){return true;}
};



class TimeCounter:public BoltzmannSimBase {
protected:
//public:
  double rho;
public:
  TimeCounter(char* fl, char*outfl, long it, double DeltT, long stp,
      char*ex="", char*nm=""):
      BoltzmannSimBase(fl, outfl, it, DeltT, stp, ex, nm){rho=0;};
  virtual bool SimulateStep(/*long&nr*/);
  double CalculateRho(double* V,long partnr, double n);
  double RecalcRhoTerm(long i, long j, double*Vjold, double*VVold,
      double*VV, double rho);
  double RecalcRho(long i, long j, double*Viold, double*Vjold,
      double*VVold, double*VV, double rho, long partnr);
};

class Bird:public TimeCounter {
protected:
//public:
  double gmax;
public:
  Bird(char* fl, char*outfl, long it, double DeltT, long stp,
      char*ex="bir", char*nm="Bird"):
      TimeCounter(fl, outfl, it, DeltT, stp, ex, nm){
    gmax=0;bufferlen=4;dim3=true;
  };
  void SampleVelocitiesBird(long i, long j, double buffer2,
      double buffer3, double* oldV);
  virtual bool PreProcess();
  virtual void SamplePair(long i, long j, double gij,
      double*V, double*Vabs, long partnr, double gmax);
  virtual double DoCollision(/*long nr, */double t);
  double CalculateGMax(double*Vabs, long partnr);
};

class BY:public TimeCounter {
public:
  BY(char* fl, char*outfl, long it, double DeltT, long stp,
      char*ex="by", char*nm="Belotserkowskiy-Yanitskiy"):
      TimeCounter(fl, outfl, it, DeltT, stp, ex, nm){
      dim3=true;
      bufferlen=4;
    };
  virtual bool PreProcess();
  virtual double DoCollision(/*long nr, */double t);
};

class Deshepande:public BY {
private:
  long oldk;
  double oldval, lnval;
public:
  Deshepande(char* fl, char*outfl, long it, double DeltT, long stp,
      char*ex="dsh", char*nm="Deshepande"):
      BY(fl, outfl, it, DeltT, stp, ex, nm) {oldk=0; oldval=0; lnval=0; };
  virtual bool SimulateStep(/*long&nr*/);
  double RecalcRho(long i, long j, double*Viold, double*Vjold,
      double*VVold, double*VV, double rho, long partnr);
};

class Nanbu:public BoltzmannSimBase {
public:
//  double*Vold;
  bool babov;
public:
  Nanbu(char* fl, char*outfl, long it, double DeltT, long stp,
      char*ex="nan", char*nm="Nanbu"):
      BoltzmannSimBase(fl, outfl, it, DeltT, stp, ex, nm){
```

```
      babov=false;
      bufferlen=4;
    };
  virtual bool PreProcess();
  virtual bool PostProcess();
  virtual bool SimulateStep(/*long&nr*/);
};
class Babovsky:public Nanbu {
public:
  Babovsky(char* fl, char*outfl, long it, double DeltT, long stp,
      char*ex="bab", char*nm="Babovsky"):
      Nanbu(fl, outfl, it, DeltT, stp, ex, nm){babov=true;};
};

class LD:public Nanbu {
public:
  LD(char*fl, char*outfl, long it, double DeltT, long stp,
      char*ex="ldh", char*nm="LD (Halton)"):
      Nanbu(fl, outfl, it, DeltT, stp, ex, nm) {sort=true;babov=false;};
  virtual bool InitSequence();
};

class QMC1:public BoltzmannSimBase {
protected:
//  double*Vold;
public:
  QMC1(char* fl, char*outfl, long it, double DeltT, long stp, long bs,
      char*ex="qmc1", char*nm="Lecot QMC1"):
      BoltzmannSimBase(fl, outfl, it, DeltT, stp, ex, nm){
      bufferlen=5;
      sort=true;
      base=bs;
    };
  virtual bool SimulateStep(/*long&nr*/);
  virtual bool InitSequence();
};

class QMC3:public BoltzmannSimBase {
protected:
  long d[3], dim[3];
//  long base;
public:

  QMC3(char*fl, char*outfl, long it, double DeltT, long stp,long* D,long bs,

      char*ex="qmc3", char*nm="Lecot's QMC3");
  long index3d(long i1,long i2,long i3) {
    if (d!=NULL) return i3 + 2*i2*dim[2] + 2*i1*dim[2]*dim[1];
    else return 0;
  }
  virtual bool SimulateStep(/*long&nr*/);
  virtual bool InitSequence();
/*  virtual bool PreProcess();
  virtual bool PostProcess();*/
  virtual long SortV();
};

class QMC3SortVabs:public QMC3 {
public:
  QMC3SortVabs(char*fl, char*outfl, long it, double DeltT, long stp,
      long* D,long bs, char*ex="qmc3vabs", char*nm="QMC3 (sorted Vabs)"):
      QMC3(fl, outfl, it, DeltT, stp,D,bs,ex,nm){};
  virtual long SortV();
};


#endif
```

Die Simulationsmethoden (Implementation)

```
// ***************************************************************************
//                          BoltzmannSimulation.cpp  -  description
//                             -------------------
//    copyright            : (C) 1999 by Reinhold Kainhofer
//    email                : reinhold@wolfram.com
// ***************************************************************************/

#include <stdlib.h>
#include <stdio.h>
#include <iostream.h>
#include <math.h>
#include <string.h>
#include <time.h>
#include "BoltzmannSimulation.h"
#include "SimSeqs.h"


/*****************************************
 *     General functions for the BEq         *
 *****************************************/

bool ErrorCheck(long i, long j, long partnr) {
  bool noerror=true;
```

```
  if (i<0) {
    cout<<"error: i="<<i<<"       ";
    noerror=false;
  } else if (i>=partnr) {
    cout<<"error: i="<<i<<"       ";
    noerror=false;
  }

  if (j<0) {
    cout<<"error: j="<<j<<"       ";
    noerror=false;
  } else if (j>=partnr) {
    cout<<"error: j="<<j<<"       ";
    noerror=false;
  }
  return noerror;
}

long factorial(long k) {
  long res=1;
  for (long j=2; j<=k; j++) res*=j;
  return res;
}

long SampleXFromPoisson(double randnr, double lambd, long &oldk, double &oldval, double &lnval) {
  double ll=log(lambd);

  double tmp=-lambd;
  double where=exp(tmp);
  long k=0;

  while (where<randnr) {
    k++;
    tmp+=ll;
    tmp-=log(k);
    where+=exp(tmp);
  }
  return k-1;
}


#ifdef SIGMA_G_CONSTANT
#define gsigma(g) kconst*4*Pi
#else
double gsigma(double theta,double g) {return kconst*4*Pi;}
double gsigma(double g) {return kconst*4*Pi;}
#endif



// ************************************************************************
// BoltzmannSimBase: base class for all simulation schemes of the BEq.
// ************************************************************************

BoltzmannSimBase::BoltzmannSimBase(char* fl, char*outfl, long it,
    double DeltT, long stp, char*ex, char*nm) {
  sequence=NULL;
  file=fl;
  outfile=outfl;
  iter=it;
  DeltaT=DeltT;
  step=stp;
  partnr=0;
  V=NULL;
  Vabs=NULL;
  Vold=NULL;
  buffer=NULL;
  dim3=false;
  bufferlen=1;
  name=nm;
  ext=ex;
  sort=false;
  DelSequence=true;
}

bool BoltzmannSimBase::SetSequence(SimulationSeqsBase*sim, bool delSeq,
    bool srt) {
  if (sim!=NULL) {
    if (sequence!=NULL) {free(sequence);sequence=NULL;}
    sequence=sim;
    DelSequence=delSeq;
  }
  sort=srt;
  return true;
}


bool BoltzmannSimBase::DoSimulation(){
  bool res=false;

  if((Vabs==NULL) &&(!ReadV())) return false;
  if ((sequence==NULL)&&(!InitSequence())) return false;
  if (!InitBuffer()) return false;

  res=Simulate();
```

```
  if (sequence!=NULL && DelSequence) {free(sequence); sequence=NULL;}
  if (V!=NULL) {free(V); V=NULL;}
  if (Vabs!=NULL) {free(Vabs); Vabs=NULL;}

  return res;
}

bool BoltzmannSimBase::InitBuffer() {
  if (buffer!=NULL) {free(buffer);buffer=NULL;}
    // this will keep the actual element of the ld sequence
  buff=(double*)calloc(bufferlen, sizeof(double));
    // this will hold the symmetrized el. of the ld sequence
  buffer=(double*)calloc(bufferlen, sizeof(double));
  return (buffer!=NULL);
}

BoltzmannSimBase::~BoltzmannSimBase(){
  if (sequence!=NULL) {free(sequence);sequence=NULL;}
  if (V!=NULL) {free(V);V=NULL;}
  if (Vabs!=NULL) {free(Vabs); Vabs=NULL;}
  if (buffer!=NULL) {free(buffer); buffer=NULL;}
  if (buff!=NULL) {free(buffer); buffer=NULL;}
}

bool BoltzmannSimBase::Simulate() {
  char fn[300];
  strcat(strcat(strcpy(fn,outfile), "."), sequence->GetExtension());
  FILE* datastr=fopen(fn,"wt");               // The output file
  strcat(fn, ".m");
  FILE* str=fopen(fn,"wt");                // The output file
//  long nr=0;       // counter for the sequences or number of collisions
  long*collnr=(long*)calloc(2*iter, sizeof(long));
  long n, i;

  nrSoFar=0;
  if (!PreProcess()) {
    fclose(str);
    return false;
  }
#ifdef PRECREATE_NUMBERS
starttime=clock();
cout<<"start createNumbers()\n";
sequence->CreateNumbers();
cout<<"end createNumbers(): "<<
    (double)(clock()-starttime)/(double)CLOCKS_PER_SEC<<" seconds\n";
#endif

  if (sort) SortV();
  WriteIntro(str, sequence, iter, partnr, DeltaT);
  fprintf(str, "};\n\n");
  fprintf(datastr, "%li %li\n%f\n", partnr, iter, DeltaT);
  starttime=clock();
  for (n=0; n<iter; n++) {
cout<<n<<"\n";
    collisions=0;
    // write out the v-Distribution before calculation the new one
//    WriteDoubleArray(str,partnr,Vabs,",\n", 1);
    for (i=0; i<partnr; i++) fprintf(datastr, "%f ", Vabs[i]);
    fprintf(datastr, "\n");
    if (!SimulateStep()) {
//      WriteDoubleArray(str, partnr, Vabs, "};\n\n",1);
      for (i=0; i<partnr; i++) fprintf(datastr, "%f ", Vabs[i]);
      fprintf(datastr, "\n");

      cout<<"Error in the "<<sequence->GetName()<<" simulation...\n";
      fprintf(str, "error[\"%s\"]=True; (*Error in Simulate Step *)\n", ext);
      fclose(str);
      return false;
    }
    collnr[n]=collisions;
    if (sort) SortV();
  }

  //write out the final v-distribution
//  WriteDoubleArray(str, partnr, Vabs, "};\n", 1);
  for (i=0; i<partnr; i++) fprintf(datastr, "%f ", Vabs[i]);
  fprintf(datastr, "\n");

  double timeused=(double)(clock()-starttime)/(double)CLOCKS_PER_SEC;
  fprintf(str, "(* the time used for this simulation*)\ntime[\"%s\", %li]=%g;\n",
      ext,partnr,timeused);

  fprintf(str, "(* Number of collisions in each time step *)\n");
  fprintf(str, "collisions[\"%s\", %li]={%li", ext, partnr, collnr[0]);
  for (n=1;n<iter;n++) fprintf(str, ", %li", collnr[n]);
  fprintf(str, "};\n\n");
  fprintf(str, "discr[\"%s\", %li]={\n", ext, partnr);
  if (collnr!=NULL) {free(collnr);collnr=NULL;}
  fclose(str);
  fclose(datastr);

  cout<<"Results of "<<sequence->GetName()<<" method written to: "<<fn<<"\n";
  if (!PostProcess()) return false;
  return true;
```

```
}

bool BoltzmannSimBase::InitSequence() {
  sequence=(SimulationSeqsBase*)new MonteCarlo(ext, name);
  return (sequence!=NULL);
};

bool BoltzmannSimBase::ReadV() {
  long n=0, i;
  FILE* stream;
  char fn[250];
  strcpy(fn, file);
  if (dim3) strcat(fn, "_3D");

  stream=fopen(fn, "rt");
  if (stream==0) return false;
  fscanf(stream, "%li", &n);
//  while (fscanf(stream, "%lg", &nnr)>0) {
//    n++;
//  }
//  fclose(stream);
//  if (dim3) partnr=n/3; else partnr=n;
  partnr=n;
  if (dim3) n*=3;
cout<<"partnr = "<<partnr<<",   n="<<n<<"\n";
  double* VV=(double*) calloc(n, sizeof(double));


//  stream=fopen(fn, "rt");
  for (i=0;i<n;i++) {
    fscanf(stream, "%lg  ", &VV[i]);
  }
  fclose(stream);

  if (V!=NULL) {free(V);V=NULL;}
  if (Vabs!=NULL) {free(Vabs);Vabs=NULL;}
  if (dim3) {
    V=VV;
    Vabs=(double*)calloc(partnr, sizeof(double));
    for (i=0; i<partnr; i++) {

      Vabs[i]=vectabs(V[3*i], V[3*i+1], V[3*i+2]);
    }
  } else {
    Vabs=VV;
    V=NULL;
  }
  return (VV!=NULL);
}

void ExchangeVij(double*V, double*Vabs, long i, long j, long partnr) {
  double tmp=0;
#ifdef ERRORCHECK
ErrorCheck(i,j,partnr);
#endif

  tmp=Vabs[i];
  Vabs[i]=Vabs[j];
  Vabs[j]=tmp;

  if (V!=NULL) {
    for (int k=0;k<3;k++) {
      tmp=V[3*i+k];
      V[3*i+k]=V[3*j+k];
      V[3*j+k]=tmp;
    }
  }
}


#define SORTPARTLEN  4
bool BoltzmannSimBase::QuickSort(long l, long r) {
  long i, j;
  double v;

  if ((r-l)>SORTPARTLEN) {
    i = (r+l)/2;
    if (Vabs[l]>Vabs[i]) ExchangeVij(V, Vabs,l,i, partnr); // Tri-Median Methode!
    if (Vabs[l]>Vabs[r]) ExchangeVij(V, Vabs,l,r, partnr);
    if (Vabs[i]>Vabs[r]) ExchangeVij(V, Vabs,i,r, partnr);

    j = r-1;
    ExchangeVij(V, Vabs,i,j, partnr);
    i = l;
    v = Vabs[j];
    do {
      while(Vabs[++i]<v);
      while(Vabs[--j]>v);
      if (j>=i) ExchangeVij(V, Vabs,i,j, partnr);
    } while (j>=i);
    ExchangeVij(V, Vabs,i,r-1, partnr);
    QuickSort(l,j);
    QuickSort(i+1,r);
  }
  return true;
```

```
}

// If the particles should be sorted according to one of the
// coordinates of V[], give the coordinate as an additional argument
bool BoltzmannSimBase::QuickSort(long l, long r, long pos) {
  long i, j;
  double v;

  if ((r-l)>SORTPARTLEN) {
    i = (r+l)/2;
    if (V[l*3+pos]>V[i*3+pos]) ExchangeVij(V, Vabs,l,i, partnr);
    if (V[l*3+pos]>V[r*3+pos]) ExchangeVij(V, Vabs,l,r, partnr);
    if (V[i*3+pos]>V[r*3+pos]) ExchangeVij(V, Vabs,i,r, partnr);

    j = r-1;
    ExchangeVij(V, Vabs,i,j, partnr);
    i = l;
    v = V[j*3+pos];
    do {
      while(V[(++i)*3+pos]<v);
      while(V[(--j)*3+pos]>v);
      if (j>=i) ExchangeVij(V, Vabs,i,j, partnr);
    } while (j>=i);
    ExchangeVij(V, Vabs,i,r-1, partnr);
    QuickSort(l,j,pos);
    QuickSort(i+1,r,pos);
  }
  return true;
}

bool BoltzmannSimBase::InsertionSort(long lo0, long hi0) {
  int i, j;
  double v, tmp[3];

  for (i=lo0+1;i<=hi0;i++) {
    v = Vabs[i];
    if (dim3) {
      tmp[0]=V[3*i];
      tmp[1]=V[3*i+1];
      tmp[2]=V[3*i+2];
    }
    j=i;
    while ((j>lo0) && (Vabs[j-1]>v)) {
#ifdef ERRORCHECK
ErrorCheck(i,j,partnr);
#endif
      Vabs[j]=Vabs[j-1];
      if (dim3) {
        V[3*j]=V[3*(j-1)];
        V[3*j+1]=V[3*(j-1)+1];
        V[3*j+2]=V[3*(j-1)+2];
      }
      j--;
    }
#ifdef ERRORCHECK
ErrorCheck(i,j,partnr);
#endif
    Vabs[j] = v;
    if (dim3) {
      V[3*j]=tmp[0];
      V[3*j+1]=tmp[1];
      V[3*j+2]=tmp[2];
    }
  }
  return true;
}
bool BoltzmannSimBase::InsertionSort(long lo0, long hi0, long pos) {
  int i, j;
  double v, tmp[3];

  for (i=lo0+1;i<=hi0;i++) {
    v = Vabs[i];
    if (dim3) {
      tmp[0]=V[3*i];
      tmp[1]=V[3*i+1];
      tmp[2]=V[3*i+2];
    }
    j=i;
    while ((j>lo0) && (V[(j-1)*3+pos]>tmp[pos])) {
#ifdef ERRORCHECK
ErrorCheck(i,j,partnr);
#endif
      Vabs[j]=Vabs[j-1];
      if (dim3) {
        V[3*j]=V[3*(j-1)];
        V[3*j+1]=V[3*(j-1)+1];
        V[3*j+2]=V[3*(j-1)+2];
      }
      j--;
    }
    Vabs[j] = v;
#ifdef ERRORCHECK
ErrorCheck(i,j,partnr);
#endif
    if (dim3) {
```

```
      V[3*j]=tmp[0];
      V[3*j+1]=tmp[1];
      V[3*j+2]=tmp[2];
    }
  }
  return true;
}



long BoltzmannSimBase::SortV() {
  QuickSort(0, partnr-1);
  InsertionSort(0, partnr-1);
  return 0;
}



double BoltzmannSimBase::G(long i, long j) {
  if (dim3) {
    return vectabs(V[3*i]-V[3*j], V[3*i+1]-V[3*j+1], V[3*i+2]-V[3*j+2]);
  } else {
    return Vabs[i]+Vabs[j];
  }
}

long BoltzmannSimBase::SampleHomogeneousVelocities3D(double*V,
    long i, long j, double rnd1, double rnd2, double*Vold, double vdist){
  double nu[3];
  double Vdist=0;
  double vi,vj;
  if (vdist>=0) Vdist=vdist;
  else Vdist=vectabs(V[3*j]-V[3*i],V[3*j+1]-V[3*i+1],V[3*j+2]-V[3*i+2]);

  nu[0]=2*rnd1-1;
  nu[1]=2*sqrt(rnd1*(1-rnd1))*cos(2*Pi*rnd2);
  nu[2]=2*sqrt(rnd1*(1-rnd1))*sin(2*Pi*rnd2);
  for (int k=0; k<=2; k++) {
      // temporarily save the old values, since V[i+k] is overwritten
    vi=V[3*i+k];
    vj=V[3*j+k];
    V[3*i+k]=(vi + vj + Vdist*nu[k])/2.;
    V[3*j+k]=(vi + vj - Vdist*nu[k])/2.;
  }
  if (Vabs!=NULL) {
    Vabs[i]=vectabs(V[3*i], V[3*i+1], V[3*i+2]);
    Vabs[j]=vectabs(V[3*j], V[3*j+1], V[3*j+2]);
  } else return -1;

  return 0;
}

void BoltzmannSimBase::GetPair(long&i, long&j, long pair) {
  long pr=pair;
  i=1;
  while (pr>=i) {
    pr-=i;
    i++;
  }
  j=pr;
  return;
}

void BoltzmannSimBase::SampleHomogeneousVelocities(double*V, long i, long j,
    double rand1, double rand2, double* oldV) {
  double vi=oldV[i], vj=oldV[j];
  V[i]=bracket(vi, vj, rand1, rand2);
  V[j]=bracket(vj, vi, rand1, 1-rand2);
  return;
}

void BoltzmannSimBase::SamplePairEqualP(long&i,long&j, double randnr,
    long partnr) {
  long pair=lfloor(randnr*partnr*(partnr-1)/2.);
  GetPair(i,j,pair);
  return;
}

//long fstep;
/** q(k) gives the ... q(k)=4*pi*k for some constant k, for now... */

double BoltzmannSimBase::q(double g) {
  return /*4*Pi**/ncell*gsigma(g);
}

/** helper function, calculate [a,b;x4,x5] */
double BoltzmannSimBase::bracket (double v, double w, double x4, double x5) {
  double res;
  double vv=v*v;
  double ww=w*w;
  double vvww=vv+ww;

  res=(vvww + sqrt(pow(vvww,2) - 4*vv*ww*pow(-1 + 2*x4,2))*(-1 + 2*x5)) / 2;
  if (res>0) {
    res=sqrt(res);
```

```
  } else {
    res=0;
  }

  return res;
}

/** helper function, calculate [a,b;x4] */
double BoltzmannSimBase::bracket(double v, double w, double x4) {
  return sqrt( pow(v+w,2)-4*x4*v*w);
}




/*************************************************************
 *       Time counter - base class for Bird, BY etc.         *
 *************************************************************/
bool TimeCounter::SimulateStep(/*long&nr*/){
  double t=0, tau=0;
  while (t<DeltaT) {
    tau=DoCollision(t);
    if (tau<0) {return false;} else t+=tau;
  }

  return true;
};

double TimeCounter::CalculateRho(double* V,long partnr, double n) {
#ifdef SIGMA_G_CONSTANT
  return gsigma(0)/2*(partnr-1)*n;
#else
  long i,j;
  double rho=0;
  double va=0;
  for (i=0;i<partnr;i++) {
    for (j=i+1;j<partnr;j++) {
      va=vectabs(V[i*3]-V[j*3], V[i*3+1]-V[j*3+1], V[i*3+2]-V[j*3+2]);
      rho+=(gsigma(va));
    }
  }
  return rho*n/partnr;
#endif
}


double TimeCounter::RecalcRhoTerm(long i, long j, double*Vjold,
    double*VVold, double*VV, double rho){
  double newrho=rho;
  double va=vectabs(VVold[i*3]-Vjold[0], VVold[i*3+1]-Vjold[1],
      VVold[i*3+2]-Vjold[2]);
  newrho-=(gsigma(va));
  va=vectabs(VV[i*3]-VV[j*3], VV[i*3+1]-VV[j*3+1], VV[i*3+2]-VV[j*3+2]);
  newrho+=(gsigma(va));
  return newrho;
}

double TimeCounter::RecalcRho(long i, long j, double*Viold, double*Vjold,
    double*VVold, double*VV, double rho, long partnr) {
#ifdef MAXWELL
  return rho;
#else
  double newrho=rho;
  long n;
  for (n=0; n<partnr; n++) newrho=RecalcRhoTerm(n,j,Vjold,VVold, VV, newrho);
  for (n=0; n<j-1; n++) newrho=RecalcRhoTerm(n,i,Viold,VVold, VV, newrho);
  for (n=j+1; n<partnr; n++) newrho=RecalcRhoTerm(n,i,Viold, VVold,VV,newrho);
  return newrho;
#endif
}



/*************************************************************
 *       Bird's Time counter method                          *
 *************************************************************/
void Bird::SampleVelocitiesBird(long i, long j,
    double buffer2, double buffer3, double* oldV) {
   /* ToDo: for Maxwell molecules, we don't need
      this function, so it's not implemented */
  return;
}

double Bird::CalculateGMax(double*Vabs, long partnr) {
  double gm=0;
  long i=0;
  for (i=0;i<partnr;i++) {
    if (gm<Vabs[i]) gm=Vabs[i];
  }
  return 2*gm;
}
bool Bird::PreProcess() {
   #ifndef SIGMA_G_CONSTANT
   gmax=CalculateGMax(Vabs, partnr);
```

```
   #endif
   return true;
};

void Bird::SamplePair(long i, long j, double gij, double*V, double*Vabs,
     long partnr, double gmax) {
   bool found=false;
   double randfrac;
   double randpair;
   while(!found) {
      randfrac=(double)(rand()/RAND_MAX);
      randpair=(double)(rand()/RAND_MAX);
      long pair=(long) (randpair*(partnr-1)*(partnr-2)/2);
      GetPair(i,j,pair);
#ifdef ERRORCHECK
ErrorCheck(i,j,partnr);
#endif

      gij=G(i,j);

      found=(randfrac< (gsigma(gij))/(gsigma(gmax)));
   }
   return;
}


double Bird::DoCollision(double t) {
  long i=0, j=0;
  double tau=0;
  double gij;
  sequence->NextElement(nrSoFar, buffer, bufferlen);
  nrSoFar++;
#ifdef SIGMA_G_CONSTANT
  SamplePairEqualP(i,j,buffer[0],partnr);    // sample a collision pair
  gij=G(i,j);
#else
  SamplePair(i,j,gij, V,Vabs,partnr,gmax);
#endif
#ifdef ERRORCHECK
  ErrorCheck(i,j,partnr);
#endif
    // advance the time counter
  tau=2/(partnr*ncell*gsigma(gij) );

    // calculate the new velocity from the angles
  SampleHomogeneousVelocities3D(V, i,j, buffer[2], buffer[3], Vold);
  collisions++;

#ifndef HOMOGENEOUS
    // update the gmax
  if (Vabs[i]*2>gmax) gmax=2*Vabs[i];
  if (Vabs[j]*2>gmax) gmax=2*Vabs[j];
#endif

  return tau;
};



/*********************************************
 *   BY  -  Belotserkowskiy-Yanitskiy        *
 *********************************************/
bool BY::PreProcess() {
  rho=CalculateRho(V, partnr, ncell);   // will be updated dynamically
  return true;
}

double BY::DoCollision(double t) {
  long i=0, j=0;
  double viold[3], vjold[3];
  sequence->NextElement(nrSoFar, buffer, bufferlen);
//fprintf(tempfile, ",\n {%li, {%g, %g, %g, %g}}", nrSoFar, buffer[0],
//buffer[1], buffer[2], buffer[3]);
  nrSoFar++;

  double tau=-log(1-buffer[0])/rho;

#ifdef SIGMA_G_CONSTANT
  SamplePairEqualP(i,j,buffer[1], partnr);
#ifdef ERRORCHECK
ErrorCheck(i,j,partnr);
#endif

#else
  /* ToDo: if g sigma(g)!=const, the kinematics is more complicated,
     but our experiments don't deal with that case, so this is not
     implemented */
#endif
  collisions++;
  for (int k=0;k<3;k++) {
    viold[k]=V[3*i+k];
    vjold[k]=V[3*j+k];
  }
  SampleHomogeneousVelocities3D(V, i, j, buffer[2], buffer[3], V);
```

```
  rho=RecalcRho(i,j, viold, vjold, V, V, rho, partnr);
  return tau;
}



/*************************************************************
 *       Deshepande Method (modification of BY)              *
 *************************************************************/
bool Deshepande::SimulateStep(/*long&nr*/){
  rho=CalculateRho(V, partnr, ncell);
  long collisionnr=SampleXFromPoisson(rand()/(double)RAND_MAX, rho*DeltaT, oldk, oldval, lnval);
  for (int k=1; k<collisionnr; k++) DoCollision(0);
  return true;
};

double Deshepande::RecalcRho(long i, long j, double* Viold, double * Vjold,
    double* VVold, double *VV, double rho, long partnr) {
  return 0;
}



/*************************************************************
 *       Nanbu's DSMC method                                 *
 *************************************************************/
bool Nanbu::PreProcess() {
  if (4*Pi*kconst*DeltaT>=1){
    cout<<"q=4 Pi k DeltaT is not lower than 1. The method \""<<name<<
        "\" is not necessarily convergent!!!!!\n";
  }

  Vold=(double*) calloc(partnr, sizeof(double));
  return (Vold!=NULL);
}

bool Nanbu::PostProcess() {
  if (Vold!=NULL) {free(Vold);Vold=NULL;}
  return true;
}

bool Nanbu::SimulateStep(/*long&nr*/) {
  long l=0, i=0, j=0;

  double q=4*Pi*kconst*DeltaT;
  double tmp;

  memcpy(&Vold[0], &Vabs[0], (partnr*sizeof(double)));
  for (l=0; l<partnr; l++) {
    sequence->NextElement(nrSoFar, buffer, bufferlen);
    nrSoFar++;
//fprintf(tempfile, ",\n {%li, {%g, %g, %g, %g}}", nrSoFar,
//buffer[0], buffer[1], buffer[2], buffer[3]);

    tmp=buffer[2]*partnr;

    if (!babov) {
      // The DSMC method uses the interval q/N*[j-1,j)
      if (buffer[2]< q) {
        i=lfloor(buffer[3]*partnr);
        j=lfloor(tmp/q);
#ifdef ERRORCHECK
ErrorCheck(i,j,partnr);
#endif
        collisions++;
        Vabs[i]=bracket(Vold[j], Vold[i],buffer[0], buffer[1]);
      }
    } else {
      // The DSMC* method uses 1/N*[j-q,j)
      if (tmp-floor(tmp)<q) {
        i=lfloor(buffer[3]*partnr);
        j=lfloor(tmp);
#ifdef ERRORCHECK
ErrorCheck(i,j,partnr);
#endif
        collisions++;
        Vabs[i]=bracket(Vold[j], Vold[i],buffer[0], buffer[1]);
      }
    }
  }
  return true;

}



/*************************************************************
 *       LD - Lecot's QMC variation of Nanbu's method        *
 *************************************************************/
bool LD::InitSequence() {
  long bases[4]={2,3,7,5};
  return (sequence=(SimulationSeqsBase*)new Halton(&bases[0], 4,
      (long)partnr*iter, BGENAU, GENAU, ext, name))!=NULL;
}
```

```
/*************************************************************
 *       QMC1 - Lecot's QMC method (1-dimensional)           *
 *************************************************************/
bool QMC1::SimulateStep(/*long&nr*/) {
  long l=0, i=0, j=0;
  double vi, vj, nu=(double)partnr/2.;
  for (l=0; l<partnr/2; l++) {
    sequence->NextElement(nrSoFar, buff, bufferlen);
    nrSoFar++;
//fprintf(tempfile, ", {%li,\n {%g, %g, %g, %g, %g}} ", nrSoFar,
//buffer[0], buffer[1], buffer[2], buffer[3], buffer[4]);

      // Symmetrize the sequence so that the requirements (1,18)
      // from Lecot's paper are satisfied
    buffer[0]=(buff[0]+ (floor(nu*buff[0])+1.) / nu )/2.;
    buffer[1]=(buff[1]+ (floor(nu *buff[1])) / nu )/2.;
    buffer[4]=(buff[4]+ (floor(nu*buff[4])+1.) / nu )/2.;
    buffer[2]=buff[2];
    buffer[3]=buff[3];

      // don't add the 1 like in the paper, because here the indices are 0-based
    i=lfloor(partnr*buffer[0]);
    j=lfloor(partnr*buffer[1]);
#ifdef ERRORCHECK
ErrorCheck(i,j,partnr);
#endif
    vi=Vabs[i];
    vj=Vabs[j];
    if (buffer[2]<DeltaT*q(bracket(vi, vj, buffer[3])) ) {
      collisions++;
      Vabs[i]=bracket(vi, vj, buffer[3], buffer[4]);
      Vabs[j]=bracket(vj, vi, buffer[3], 1-buffer[4]);
    }
  }
  return true;
}

bool QMC1::InitSequence() {
  long bases[5]={1,2,3,4,5};
  sequence=(SimulationSeqsBase*)new Netz(&bases[0], /*base:*/base, /*dim*/5,
      (long)partnr*iter, (long)(log(partnr/2.)/log(base)), ext, name);
  return (sequence!=NULL);
}



/*************************************************************
 *       QMC3 - Lecot/Khettabi's 3-dimensional QMC method    *
 *************************************************************/
QMC3::QMC3(char*fl, char*outfl, long it, double DeltT, long stp, long*D,
    long bs, char*ex, char*nm):
    BoltzmannSimBase(fl, outfl, it, DeltT, stp, ex, nm) {
  dim3=true;
  bufferlen=9;
  base=bs;
  for (int i=0; i<3; i++) {
    d[i]=D[i];
    dim[i]=power(base, d[i]);
  }
}

/*bool QMC3::PreProcess() {
  Vold=(double*) calloc(3*partnr, sizeof(double));
  return (Vold!=NULL);
}
bool QMC3::PostProcess() {
  if (Vold!=NULL) {free(Vold);Vold=NULL;}
  return true;
}*/


bool QMC3::SimulateStep(/*long&nr*/) {
  long l, j[3], k[3];
  long jindex, kindex;
  double vdist;

//  memcpy(&Vold[0], &V[0], 3*partnr*sizeof(double));
Vold=V;

  for (l=0; l<partnr/2; l++) {
    sequence->NextElement(nrSoFar, buff, bufferlen);
    nrSoFar++;
//fprintf(tempfile, ", {%li,\n {%g, %g, %g, %g, %g, %g, %g, %g, %g}} ", nr,
//buffer[0], buffer[1], buffer[2], buffer[3], buffer[4], buffer[5],
//buffer[6], buffer[7], buffer[8]);
      // "Symmetrize" the sequence
    buffer[0]=buff[0];
    buffer[1]=buff[1];
    buffer[2]=.5*(buff[2] + floor(dim[2]*buff[2])/(double)dim[2] );
    buffer[3]=buff[3];
    buffer[4]=buff[4];
```

```
    buffer[5]=.5*(buff[5] + (floor(dim[2]*buff[5]) + 1.)/(double)dim[2] );
    buffer[6]=buff[6];
    buffer[7]=buff[7];
    buffer[8]=buff[8];

      // get the indeces of the particles
    j[0]=lfloor(dim[0]*buffer[0]);
    j[1]=lfloor(dim[1]*buffer[1]);
    j[2]=lfloor(dim[2]*buffer[2])*2;
    k[0]=lfloor(dim[0]*buffer[3]);
    k[1]=lfloor(dim[1]*buffer[4]);
    k[2]=lfloor(dim[2]*buffer[5])*2+1;

      // the index of j and k in Vold
    jindex=index3d(j[0], j[1], j[2]);
    kindex=index3d(k[0], k[1], k[2]);
#ifdef SIGMA_G_CONSTANT
      // if x7<DeltaT*q(Delta v)  => particles j and k collide
      // so calculate new velocities
    if (buffer[6]<DeltaT*q(0)) {
      vdist=vectabs( Vold[3*jindex]-Vold[3*kindex],
          Vold[3*jindex+1]-Vold[3*kindex+1],
          Vold[3*jindex+2]-Vold[3*kindex+2]);
#else
    vdist=vectabs( Vold[3*jindex]-Vold[3*kindex],
        Vold[3*jindex+1]-Vold[3*kindex+1],
        Vold[3*jindex+2]-Vold[3*kindex+2]);
      // if x7<DeltaT*q(Delta v)  => particles j and k collide
      // so calculate new velocities
    if (buffer[6]<DeltaT*q(vdist)) {
#endif
        // the nu-vector (difference between v and v')
      SampleHomogeneousVelocities3D(V, jindex, kindex,
          buffer[7], buffer[8], Vold, vdist);
      collisions++;
    }
  }
  Vold=NULL;
  return true;
}

bool QMC3::InitSequence() {
  long bases[9]={1,2,3,4,5,6,7,8,9};
  /* add one to lambda, so that more numbers are precreated*/
  sequence=(SimulationSeqsBase*)new Netz(&bases[0], base, /*dim*/9,
      (long)partnr*iter, /*lambda*/1+ d[0]+d[1]+d[2], ext, name);
  return (sequence!=NULL);
}


long QMC3::SortV() {
  // Sort everything acc. to first coord.
  QuickSort(0, partnr-1, 0);
  InsertionSort(0, partnr-1,0);
  long low, up;

    // now, within k[0]=const. sort according to second coord.
  for (long l=0; l<dim[0]; l++) {
    low=l*dim[1]*dim[2]*2;
    up=(l+1)*dim[1]*dim[2]*2 - 1;
//    QuickSort(low, up, 1);
    InsertionSort(low, up, 1);
      // within k[0], k[1]=const, sort according to third coordinate
    for (long k=0; k<dim[1]; k++) {
//      QuickSort(low +k*dim[2]*2, low + (k+1)*dim[2]*2 - 1, 2);
      InsertionSort(low +k*dim[2]*2, low + (k+1)*dim[2]*2 - 1, 2);
    }
  }

  return 0;
}


/*************************************************************
 *       Lecot's QMC3, sorted according to absolute value    *
 *************************************************************/
long QMC3SortVabs::SortV() {
//  return BoltzmannSimBase::SortV();
  QuickSort(0, partnr-1);
  InsertionSort(0, partnr-1);
  return 0;
}
```

Die Folgen kleiner Diskrepanz (Deklaration)

```
/***************************************************************************
                          SimSeqs.h  -  description
                             -------------------
    begin                : Sun May 28 2000
    copyright            : (C) 2000 by Reinhold Kainhofer
    email                : reinhold@kainhofer.com
 ***************************************************************************/
```

```
/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/


#ifndef SIMSEQS_H
#define SIMSEQS_H
#include <stdlib.h>
#include <stdio.h>

  // constants for the recursive construction of the Halton sequence
#define GENAU    1E-9
#define BGENAU   29
#define MAXLONG 2147483647
#define MAXPREC 30
#define primefile "primes.dat"


class polynomial {
public:
  polynomial(){}
  long Ordnung(){return 0;}
  long coeff(long what) {return 0;}
//  operator polynomial *(polynomial poly1) { return NULL;}
};


class SimulationSeqsBase;
long readPrimes(long* primes, long m);
long NextPrime(long prime);
#define lfloor(f) (long)floor(f)

  // writing the results (arrays) out into a file
void WriteIntro(FILE* str, SimulationSeqsBase* simulation, long iter,
    long partnr, double DeltaT);
void WriteEnd(FILE* str, long dim);
void WriteDoubleArray(FILE* str, long len, double* vals, char* c=",",
    long dim=1);

  // general math functions
long power(long m, long n);
double power(double m, long n);
long binom(long i, long k);
double vectabs(double a1, double a2, double a3);
#define min(x,y) (x<y)?x:y
#define max(x,y) (x>y)?x:y

long LongToCoeff(long nr, long base, long* cfs, long cflen);
double CoeffToDoubleModified(long base, long* cfs, long cflen, long modfact);
double CoeffToDouble(long base, long* cfs, long cflen);
bool IsPrimitiveRoot(long proot, long base);

long CalcDeterminant(long*A, long dim, long i);



/**  This is the base class for all the simulation sequences including
 *   Monte Carlo (just random numbers).
 *@author Reinhold Kainhofer
*/

class SimulationSeqsBase {
protected:
  char* name, *ext;
  /** number of dimensions (also the length of long* bases) */
  long dim, len;
  long* bases;
  double* numbers;
  bool created;

public:
  /** Determines whether the sequence is a Low Discrepancy Sequence or not. */
  bool qmc;
  SimulationSeqsBase(long*b=NULL, long dm=0,long iterations=0,
      long genau=0, double genau1=0., char* ex="",char* nm="");
  virtual ~SimulationSeqsBase();

        /** Returns the dimension of the LD sequence */
  virtual void InitMethod(long*b=NULL, long dm=0,long iterations=0,
      long genau=0, double genau1=0., char* ex="",char* nm="");
  virtual long GetDimension() {return dim;};
  virtual long GetLength() {return len;};
  virtual char* GetName() {return name;};
  virtual char* GetExtension() {return ext;};
        /** Set the bases for the sequence (or change them if they
            were already given in the constructor */
  virtual long SetBases(long*b, long dm, long genau, double genau1);
  virtual long GetBases(long*b, long dm) {
      for (long i=0;i<min(dm, dim);i++) b[i]=bases[i]; return min(dm, dim);}
        /** writes the next element of the sequence in a buffer. */
```

```
  virtual long CalculateNextElement(long nr, double*buffer, long bufflen);
  virtual long NextElement(long nr, double*buffer, long bufflen);
        /** initialize the method specific data */
  virtual long InitData(long genau, double genau1);
  virtual bool CreateNumbers();
        /** Set the name of the funciton and the default extension */
  virtual void SetNames(char* ex="",char* nm="");
  virtual long ExitData() { return 0;}
};



/**This gives the Monte Carlo Method (just pseudo-random numbers)
  *@author Reinhold Kainhofer
  */

class MonteCarlo : public SimulationSeqsBase  {
public:
  MonteCarlo(char* ex="mc",char* nm="Monte Carlo"):
      SimulationSeqsBase(NULL,0,0,0,0){
      qmc=false;
      InitMethod(NULL, 0,0,0,0, ex, nm);
    }
        /** writes the next elements of the sequence in a buffer. */
  virtual long CalculateNextElement(long nr, double*buffer, long bufflen) {
    for (long i=0;i<bufflen;i++) { buffer[i]=(double)rand()/RAND_MAX;};
    return dim;
  }
  virtual long InitData(long genau, double genau1){return dim;};
  virtual long ExitData(){return 0;};
  virtual bool CreateNumbers() {return true;}
};



/** This class implements the Halton Sequence. The VanDerCorput Sequence is
    just a special case with dim=1.
 *@author Reinhold Kainhofer
 */

class Halton : public SimulationSeqsBase  {
public:
  Halton(long*b=NULL, long dm=0,long iterations=0, long genau=BGENAU,
      double genau1=GENAU, char* ex="hal",char* nm="Halton"):
      SimulationSeqsBase(b, dm, iterations, genau, genau1, ex, nm){
      p=NULL; q=NULL; InitMethod(b, dm, iterations, genau, genau1,  ex, nm);}
        /** writes the next element(s) of the sequence in a buffer. */
  virtual long CalculateNextElement(long nr, double*buffer, long bufflen);
  virtual long InitData(long genau, double genau1);
  virtual long ExitData();
protected: // Protected attributes
  long prec;
  double *q,*p;
};



/** This class implements Atanassov's modified Halton sequence
 *@author Reinhold Kainhofer
 */

class Atanassov : public SimulationSeqsBase  {
protected: // Protected attributes
  long* factors;
  long* proots;
  long* modifiers;
public:
  Atanassov(long*b=NULL, long dm=0,long iterations=0, long genau=BGENAU,
      double genau1=GENAU, char* ex="hal",char* nm="Halton", long*prts=NULL,long*mdfs=NULL):
      SimulationSeqsBase(b, dm, iterations, genau, genau1, ex, nm){
      proots=prts; modifiers=mdfs; InitMethod(b, dm, iterations, genau, genau1,  ex, nm);}
        /** writes the next element(s) of the sequence in a buffer. */
  virtual long CalculateNextElement(long nr, double*buffer, long bufflen);
  virtual long InitData(long genau=0, double genau1=0);
  virtual long ExitData();
  bool GeneratePrimitiveRoots(long*bases, long*prts, long dim);
  bool GenerateModifiers(long*bases, long*prts, long*modfs, long dim);
};

/** This class implements the Hammersley Set.
   *@author Reinhold Kainhofer
  */

class Hammersley : public Halton  {
protected:
  int Npos;
public:
     /** npos gibt an, an welcher Stelle das n/iterations eingefuegt werden
         soll. Der Index is 0-basierend, negative Werte sind von rechts
         gezaehlt bzw. der Wert ist modulo dim */
  Hammersley(long*b=NULL, long dm=0,long iterations=0, long genau=0,
      double genau1=0., int npos=1, char* ex="ham",char* nm="Hammersley"):
      Halton(b,dm-1, iterations, genau, genau1,ex,nm) {
      Npos=((npos%dm)+dm)%dm;dim++;};
        /** write the next elements of the sequence in a buffer. */
```

```
  virtual long CalculateNextElement(long nr, double*buffer, long bufflen);
};

/** This class implements the Sobol Sequence.
  *@author Reinhold Kainhofer
  */

class Sobol : public SimulationSeqsBase  {
protected:
  long*V;
public:
  Sobol(long*b=NULL, long dm=0,long iterations=0, long genau=0,
       double genau1=0., char* ex="sob",char* nm="Sobol"):
       SimulationSeqsBase(b, dm, iterations, genau, genau1, ex,nm){
       V=NULL;InitMethod(b, dm, iterations, genau, genau1,ex,nm);}
         /** write the next element(s) of the sequence in a buffer. */
  virtual long CalculateNextElement(long nr, double*buffer, long bufflen);
  virtual long InitData(long genau, double genau1);
  virtual long ExitData();
};


/**Implementation of the Faure low discrepancy sequence
  *@author Reinhold Kainhofer
  */

class Faure : public SimulationSeqsBase  {
protected:
  long r, base;
  long*c;
public:
  long fstep;
public:
  Faure(long*b=NULL, long dm=0,long iterations=0, long genau=0,
       double genau1=0., long fstp=0, char* ex="fau",char* nm="Faure");
  virtual long CalculateNextElement(long nr, double*buffer, long bufflen);
  virtual long InitData(long genau, double genau1);
  virtual long ExitData();
//  double Faure::createNrFromCoeff(long bs, long* cff, long nrofCoeff);
  void cyMultiply(long*cc, long*yy, long*yold, long dm, long bs);
};

/** Implementation of the (0,s) nets,
  * according to an algorithm given by Lecot
  * after the theory of Niederreiter,
  * @author Reinhold Kainhofer
  */
class Netz : public SimulationSeqsBase  {
protected:
  long base, nu, lambda;
  long nprime, ncurr, ncmax;
  long*ncoeff, *ypji;
  long ypjilen;
public:
  Netz(long*b=NULL, long bas=5, long dm=0,long iterations=0,
       long lamb=5, char* ex="ts" ,char* nm="(t,s)-Sequence");
  virtual long CalculateNextElement(long nr, double*buffer, long bufflen);
  virtual long InitData(long genau, double genau1);
  virtual long ExitData();
private:
  long getypji(long p, long j, long i);
  long getypji(long index);
  long setypji(long p, long j, long i, long value);
  long setypji(long index, long value);
  long getindex(long p, long j, long i);
};

/** Implementation of the (0,s) nets,
  * according to an algorithm given by Lecot
  * after the theory of Niederreiter,
  * @author Reinhold Kainhofer
  */
class TSSequence : public SimulationSeqsBase  {
protected:
  polynomial**poly;
  long maxlog;  // holds log(MAXLONG)/log(base)   == Dimension of c-Matrix
  long base, lambda;
  long*c;
public:
  TSSequence(long*b=NULL, long bas=5, long dm=0,long iterations=0,
       long lamb=5, char* ex="tss" ,char* nm="(t,s)-Sequence 1");
  virtual long CalculateNextElement(long nr, double*buffer, long bufflen);
  virtual long InitData(long genau, double genau1);
  virtual long ExitData();
  virtual long InitPolys(long dim,long base);
};

#endif
```

Die Folgen kleiner Diskrepanz (Implementation)

```
/***************************************************************************
                          SimSeqs.cpp  -  description
```

```
                                  -------------------
    begin                : Sun May 28 2000
    copyright            : (C) 2000 by Reinhold Kainhofer
    email                : reinhold@kainhofer.com
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/
//begin{latexinclude}

#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <iostream.h>
#include "SimSeqs.h"
#include "matrix.h"


long binom(long i, long k) {
  if (k>i||k<0) return 0;
  long j, mx=max(k, i-k), res=1;
  for (j=mx+1; j<=i; j++) res*=j;
  for (j=1; j<=i-mx; j++) res/=j;
  return res;
};

long power(long m, long n) {
  long res=1;
  for (long i=1;i<=n;i++) res*=m;
  return res;
}
double power(double m, long n) {
  double res=1;
  for (long i=1;i<=n;i++) res*=m;
  return res;
}

double vectabs(double a1, double a2, double a3) {
  return sqrt(a1*a1+a2*a2+a3*a3);
}

//end{latexinclude}

bool IsPrimitiveRoot(long proot, long base) {
  bool res=true;
  long j=0;
  long tmp=1;
  while (res && (j<base-2)) {
    j++;
    tmp=(tmp*proot % base);
    if (tmp==1) res=false;
  }
  if (res) {
    tmp=(tmp*proot % base);
    if (tmp!=1) res=false;
  }
  return res;
}


double Phi(long base, long x){
  long tmp=x;
  double res=0;
  long j=1, rem=0;

  while (tmp>0) {
      rem=tmp % base;
      tmp/=base;
      res+=(rem*pow(base, -j));
      j++;
  }
  return res;
}


double CoeffToDouble(long base, long* cfs, long cflen) {
  double res=0.;
  for (long i=cflen-1; i>=0; i--) {
    res+=(double)(cfs[i]);
    res/=(double)base;
  }
  return res;
}

double CoeffToDoubleModified(long base, long* cfs, long cflen, long modfact) {
  double res=0.;
  for (long i=cflen-1; i>=0; i--) {
    res+=(double)(cfs[i]);
```

```
    res/=(double)base;
  }
  return res;
}

long LongToCoeff(long nr, long base, long*cfs, long cflen) {
  long nn=nr;
  long maxcoeff=0;

  for (long i=0; i<cflen; i++) cfs[i]=0;
  while(nn>0 && maxcoeff<cflen) {
    cfs[maxcoeff]=nn%base;
    nn=(long)nn/base;
    maxcoeff++;
  }
  return maxcoeff;
}

/** this function is needed for Atanassov's modified Halton sequence.
  * It calculates the matrix A of values a_ij with
  * prts[i]^{a_ij}=bases[j] (mod bases[i])
  *@author: Reinhold Kainhofer */
bool CreateModExponents(long*A, long dim, long*bases, long*prts){
  //ToDo: Debug this routine
  long tmp=0;
  long val=0;
  long bs=0;
  bool noerror=true;
  for (long i=0; i<dim; i++){
    for (long j=0; j<dim; j++) {
      if (i==j) A[i*dim+i]=0;
      else {
        bs=bases[j] % bases[i]; // compare with this
        val=prts[i] % bases[i]; // initial value for a_ij=1
        tmp=1;                  // holds a_ij
        while( (val != bs) && (tmp<=bases[i]) ) {
          tmp++;
          val=(val*prts[i]) % bases[i];
        }
        if (val==bs) A[i*dim+j]=tmp;
        else {
          noerror=false;
          A[i*dim+j]=0;
        }
      }
    }
  }
  return noerror;
}

long expand(long*A, bool*rowsUsed, long dim, long dimleft, long dimtocalc) {
  long fact=-1;
  long res=0;
  for (long i=0; i<dimtocalc; i++) {
    if (!rowsUsed[i]) {
      fact*=(-1);
      rowsUsed[i]=true;

      res+=fact*A[dim*i + dimleft]*expand(A, rowsUsed, dim, dimleft-1, dimtocalc);
      rowsUsed[i]=false;
    }
  }
  return res;
}

// calculate the determinant via the expansion according to the last column.
// This is really slow, but never mind...
// flag the rows used so that we don't need another array for the cofactor.
// This saves a lot of mem...
long CalcDeterminant(long*A, long dim, long dimtocalc) {
  // Calculate the determinant of A_i, which is the submatrix
  // of the first i rows and columns of A
  Matrix mx((int)dimtocalc);
  for (int i=0; i<dimtocalc; i++) {
    for (int j=0; j<dimtocalc; j++) {
      mx.Set(i+1,j+1,(Zahl)A[i*dim+j]);
     }
  }
  long res=(long)rint(mx.Det());
  return res;
}

/** this function calculates the entries k_i for Atanassov's
  * modified halton sequence. The matrix A needs to have a determinant
  * of 1, so this is used to determine k_i.
  *@author: Reinhold Kainhofer */
void MakeDet1(long*A, long dim, long i) {
  // set a_ii =0, so the new a_ii then has to be equal 1-det(A_i),
  // since det(A_{i-1}) == 1 by induction. A_i ist the matrix of the
  // first i rows and columns
//  A[i*dim+i]=0;
  long determ=CalcDeterminant(A, dim,i);
  A[i*dim+i]=1-determ;
  return;
}
```

```
/*****************************************************************
 *           reading in the primes

 *****************************************************************/
long readPrimes(long* primes, long m) {
  long pr=0,i=0;
  FILE* stream=fopen(primefile, "r");
  if (stream==0) return 0;
  for (i=0;i<m;i++) {
    fscanf(stream, " %li ", &pr);
    primes[i]=(long)pr;
  }  return m;
}

long NextPrime(long number) {
  long pr=0;
  if (number<50) {
  // for small number don't bother to look into the file, explicitely check
    pr=number;
    if (pr%2==0) pr++;
    switch (pr) {
      case 1: return 2;
      case 3: return 3;
      case 5: return 5;
      case 7: return 7;
      case 9:
      case 11: return 11;
      case 13: return 13;
      case 15:
      case 17: return 17;
      case 19: return 19;
      case 21:
      case 23: return 23;
      case 25:
      case 27:
      case 29: return 29;
      case 31: return 31;
      case 33:
      case 35:
      case 37: return 37;
      case 39:
      case 41: return 41;
      case 43: return 43;
      case 45:
      case 47: return 47;
      case 49:
      case 51: return 51;
      default: return 2;
    }
  } else {
    FILE* stream=fopen(primefile, "r");
    if (stream==NULL) {
      cout<< "Error reading the primes file \"" << primefile<<"\n";
      return -1;
    }
    while (pr<=number) fscanf(stream, " %li ", &pr);
  }
  return (long)pr;
}



/*****************************************************************
 *     writing out the results (arrays) in Mathematica format
 *****************************************************************/

void WriteIntro(FILE* str, SimulationSeqsBase* simulation, long iter,
    long partnr, double DeltaT) {
  fprintf(str, "SimulationName[\"%s\"] = \"%s\";\n\n",
      simulation->GetExtension(),simulation->GetName() );
  fprintf(str, "Iterations[\"%s\", %li] = %li;\n",
      simulation->GetExtension(),partnr,iter);
  fprintf(str, "ParticleNumber[\"%s\",%li] = %li;\n",
      simulation->GetExtension(),partnr,partnr);
  fprintf(str, "DeltaT[\"%s\",%li] = %f;\n\n", simulation->GetExtension(),partnr,DeltaT);
  fprintf(str, "SimulationData[\"%s\", %li] = {\n",simulation->GetExtension(), partnr);
}

void WriteEnd(FILE* str, long dim) {
    fprintf(str, "\n\n");
}

void WriteDoubleArray(FILE* str, long len, double* vals, char* endchar, long dim) {
  long k;
  if (dim<=1) {
    fprintf(str, "{ %f", vals[0]);
    for (k=1;k<len;k++) {fprintf(str, ", %f", vals[k]);}
    fprintf(str, "}%s\n", endchar);
  } else {
    fprintf(str, "{ ");
    for (k=0;k<len-1;k++) {
      WriteDoubleArray(str, len, &vals[k*power(len, dim-1)],",", dim-1);
    }
    char tmp[30];
```

```
    strcpy(tmp, "}");
    strcat(tmp, endchar);
    strcat(tmp,"\n");
    WriteDoubleArray(str, len, &vals[(len-1)*power(len, dim-1)], tmp, dim-1);
  }
}
```

```
// ****************************************************************************
// SimulationSeqsBase is the base class for all the sequences.
// For every sequence, only the InitData(int, double), SetNames(), ExitData(),
// CalculateNextElement(double*, long) methods need to be overridden.
// ****************************************************************************

SimulationSeqsBase::SimulationSeqsBase(long*b, long dm, long iterations,
    long genau, double genau1, char* ex,char* nm) {
  qmc=true;
  numbers=NULL;
  bases=NULL;
  created=false;
//    InitMethod(b, dm, iterations, genau, genau1);
}
void SimulationSeqsBase::InitMethod(long*b, long dm, long iterations,
    long genau, double genau1, char* ex,char* nm) {
  numbers=NULL;
  SetNames(ex,nm);
  dim=dm;
  if (b!=NULL) {
    bases=(long*)calloc(dm, sizeof(long));
    for (long i=0;i<dm;i++) {bases[i]=b[i];}
  } else {bases=NULL;}
  len=iterations;
  InitData(genau, genau1);
}

SimulationSeqsBase::~SimulationSeqsBase(){
  if (numbers!=NULL) {free(numbers);numbers=NULL;}
  ExitData();
  if (bases!=NULL) {free(bases); bases=NULL;}
}

long SimulationSeqsBase::CalculateNextElement(long nr, double*buffer,
    long bufflen) {return -1;}
long SimulationSeqsBase::NextElement(long nr, double*buffer, long bufflen) {
  if (created) {
    long to=min(dim, bufflen);
    for (long i=0; i<to; i++) {
      buffer[i]=numbers[nr*dim+i];
    }
    return min(dim, bufflen);
  } else return CalculateNextElement(nr, buffer, bufflen);
}

/** Set the bases for the sequence (or change them if they were already given
    in the constructor */
long SimulationSeqsBase::SetBases(long*b, long dm, long genau, double genau1){
  bases=b;
  dim=dm;
  return InitData(genau, genau1);
}

/** Set the name of the funciton and the default extension */
void SimulationSeqsBase::SetNames(char* ex, char* nm){
  name=nm;
  ext=ex;
}

bool SimulationSeqsBase::CreateNumbers() {
  if (numbers!=NULL) {free(numbers);numbers=NULL;}
  numbers=(double*)calloc(dim*len, sizeof(double));
  if (numbers==NULL) return false;
  for (long i=0; i<len; i++) {
    CalculateNextElement(i, &numbers[i*dim], dim);
  }
  ExitData();
  created=true;
  return true;
}

/** initialize the method specific data */
long SimulationSeqsBase::InitData(long genau, double genau1){
  return dim;
}

//long SimulationSeqsBase::ExitData() ;
```

```
// ****************************************************************************
// MonteCarlo implements the MonteCarlo method, which just returns an array
// of pseudo random numbers
// ****************************************************************************



// ****************************************************************************
// Halton implements the Halton sequence, and thus also the VanDerCorput
// sequence, which is just the Halton sequence in one dimension
// ****************************************************************************


long Halton::InitData(long genau, double genau1) {
  ExitData();
  prec=genau;

  p=(double*)calloc(dim*(prec+2), sizeof(double));
  q=(double*)calloc(dim*(prec+2), sizeof(double));
  for (long i=0;i<dim;i++) p[i*(prec+2)]=1;

  for (long j=0;j<=prec;j++) {
    for (long i=0;i<dim;i++) {
      p[i*(prec+2)+j+1]=p[i*(prec+2)+j]/(double)bases[i];
      q[i*(prec+2)+j]=(1.0-p[i*(prec+2)+j])*(1.0-genau1);
    }
  }
  return dim;
}

/** Assuming buffer already holds the old sequence */
long Halton::CalculateNextElement(long nr, double*buffer, long bufflen) {
  long d=min(bufflen,dim);
  for (long i=0;i<d;i++) {
    long k=1;
    while (buffer[i]>=q[i*(prec+2)+k] && (k<prec)) {
      k++;
    }
    if (k==1) {buffer[i]+=(double)(1./bases[i]);}
    else {buffer[i]+=(p[i*(prec+2)+k-1]+p[i*(prec+2)+k]-1);}
      // Something weird happened, because the number is not
      // in the correct range, so just return 0...
    if (buffer[i]<0 || buffer[i]>=1) {buffer[i]=0;}
  }
  return d;
}


long Halton::ExitData() {
  SimulationSeqsBase::ExitData();
  if (p!=NULL) {free(p);p=NULL;}
  if (q!=NULL) {free(q);q=NULL;}
  return 0;

}

// ****************************************************************************
// Atanassov implements the modified Halton sequence introduced by Atanassov.
// If no primitive roots are given, the lowest possible values are taken.
// ****************************************************************************


long Atanassov::InitData(long genau, double genau1) {
  long*tmp;
  long i;
//  ExitData();

   tmp=proots;
    // allocate memory for the primitive roots
  proots=(long*)calloc(dim, sizeof(long));
  if (tmp==NULL) GeneratePrimitiveRoots(bases, proots, dim);
  else { // just copy them over
    for (i=0; i<dim; i++) proots[i]=tmp[i];
  }
    // and the same with the modifiers.
  tmp=modifiers;
    // allocate memory for the primitive roots
  modifiers=(long*)calloc(dim, sizeof(long));
  if (tmp==NULL) GenerateModifiers(bases, proots, modifiers, dim);
  else { // just copy them over
    for (i=0; i<dim; i++) modifiers[i]=tmp[i];
  }
  factors=(long*)calloc(dim, sizeof(long));
  for (i=0; i<dim; i++) factors[i]=power(proots[i], modifiers[i])%bases[i];
  return dim;
}

long Atanassov::CalculateNextElement(long nr, double*buffer, long bufflen) {
  long* cfs=(long*)calloc(MAXPREC, sizeof(long));
  long d=min(bufflen, dim);
  for (long i=0; i<d; i++) {
    LongToCoeff(nr, bases[i], cfs, MAXPREC);
    buffer[i]=CoeffToDoubleModified(bases[i], cfs, MAXPREC, factors[i]);
  }
  free(cfs);
```

```
  return dim;
}


long Atanassov::ExitData() {
  SimulationSeqsBase::ExitData();
  if (factors!=NULL) {free(factors);factors=NULL;}
  if (modifiers!=NULL) {free(modifiers);modifiers=NULL;}
  if (proots!=NULL) {free(proots);proots=NULL;}
  return 0;

}

bool Atanassov::GeneratePrimitiveRoots(long*bases, long*prts, long dim){
  bool res=true;
  bool found;
  long i,j;
  for (i=0; i<dim; i++) {
    found=false;
    j=0;
    while((!found)&&(j<bases[i])) {
      j++;
      found=IsPrimitiveRoot(j, bases[i]);
    }
    prts[i]=j;
    res=(res && (j<bases[i]));
  }
  return res;
}

bool Atanassov::GenerateModifiers(long*bases, long*prts, long*modfs, long dim){
  long*A = (long*)calloc(dim*dim, sizeof(long));
  long i;
  if (CreateModExponents(A, dim, bases, prts)) {
    for (i=0; i<dim; i++) {
       /* Calculate a_ii such that det(A)=1  => the system of
          diophantine equations then has integer solutions
          (see proof in the thesis or in Atanassov's paper)*/
       MakeDet1(A, dim, i);
       modfs[i]=A[i*dim+i];
    }
    free(A);
    return true;
  } else {
    free(A);
    return false;
  }
}


// ************************************************************************
// Hammersley implements the Hammersley sequence, which is the Halton Sequence
// with one more dimension of the form n/N */
// ************************************************************************

/** Assuming buffer already holds the old element */
long Hammersley::CalculateNextElement(long nr, double*buffer, long bufflen) {
  dim--;
  long lngth=min(dim+1, bufflen);
  long i;

    // Reset to 0 so that every timestep uses the same set
  if ((nr%len)==0) {
    for (i=0;i<lngth; i++) {
      buffer[i]=0;
    }
  }

    // Shift the elements to get rid of the additional entry
  for (i=Npos; i<dim; i++) buffer[i]=buffer[i+1];
    // then call the Halton sequence
  long res=Halton::CalculateNextElement(nr, buffer, bufflen-1);
    // and shift the elements back
  for (i=dim-1; i>=Npos; i--) buffer[i+1]=buffer[i];
  buffer[Npos]=(double)((double)(nr%len))/(double)len;
  dim++;
  return res;
}



// ************************************************************************
// Sobol implements the Sobol sequence, which is more or less a permuation
// of the VanDerCorpus sequence in base two
// ************************************************************************

long Sobol::InitData(long genau, double genau1) {  long i,j;
  ExitData();
  long r=lfloor(log(MAXLONG)/log(3));

  V=(long*)calloc(r, sizeof(long));
  long*c=(long*)calloc(r*r, sizeof(long));

  long first=0;
  long last=0;
```

```
  c[0]=1;
  last=0;
  for (i=1;i<r;i++){
    first=i*r;
    c[first]=1;
    c[first+i]=1;
    for (long j=1;j<i;j++) {
      c[first+j]=(c[last+j]+c[last+j-1]) % 2;
    }

    last=first;
  }
// daraus die Richtungszahlen erstellen
  for (i=0;i<r;i++) {
    for (j=0;j<r;j++) {
      V[i]*=2;
      V[i]+=c[i*r+j];
    }
  }
  free(c);c=NULL;
  return r;
}

long Sobol::CalculateNextElement(long nr, double*buffer, long bufflen) {
  long tmp, j, res;

  for (long i=0;i<bufflen;i++) {
    tmp=nr+i;
    j=0;
    res=0;
    // Zerlegen in Basis 2
    // Aus der Darstellung in basis 2 gleich durch xor die neue Zahl erstellen.
    while (tmp>0) {
      if ((tmp%2)==1) {res^=V[j];}
      tmp/=2;
      j++;
    }
    buffer[i]=(double)res/(2.*(double)V[0]);
  }
  return bufflen;
}

long Sobol::ExitData() {
  SimulationSeqsBase::ExitData();
  if (V!=NULL) {free(V);V=NULL;}
  return 0;
}


// ****************************************************************************
// Faure implements the Faure sequence
// ****************************************************************************

Faure::Faure(long*b, long dm,long iterations, long genau, double genau1,
    long fstp, char* ex,char* nm):
    SimulationSeqsBase(b, dm, iterations, genau, genau1, ex, nm){
  r=0;
  base=0;
  c=NULL;
  fstep=fstp;
  InitMethod(b, dm, iterations, genau, genau1, ex, nm);
}


long Faure::InitData(long genau, double genau1) {
  base=NextPrime(dim);
  if (base<0) return -1;
  r=(long)(log(MAXLONG)/log(base));
  c=(long*)calloc(r*r, sizeof(long));
  long first=0;
  long last=0;
  if (base>0) {
    c[0]=1;
    last=0;

    for (long i=1;i<r;i++){
      first=i*r;
      c[first]=1;
      c[first+i]=1;
      for (long j=1;j<i;j++) {
        c[first+j]=(c[last+j]+c[last+j-1]) % base;
      }
      last=first;
    }
  }
  return dim;
}

void Faure::cyMultiply(long*cc, long*yy, long*yold, long dm, long bs){
/* y=c.yold */
  yy[0]=cc[0]*yold[0];
  long first=0;
  for (long i=1; i<dm; i++) {
    yy[i]=0;
    first=(i*dm);
```

```
    for (long j=0;j<=i;j++) {
      yy[i]+=(yold[j]*cc[first+j]);
    }
    yy[i]%=bs;
  }
}

long Faure::CalculateNextElement(long nr, double*buffer, long bufflen) {
  if (base<0) return 0;
  long d=min(dim, bufflen);
  long* a=(long*)calloc(r, sizeof(long));

  long* aold=(long*)calloc(r, sizeof(long));

    // Darstellung von n-1 in Basis base

  long tmp=nr+power(base, fstep);
  long j=0;
  while ((tmp>0) && (j<r)) {
    a[j]=tmp % base;
    tmp/=base;
    j++;
  }

    // daraus x1 erstellen (wie bei van der corput)
  buffer[0]=CoeffToDouble(base, a, r);
  for (long k=1; k<d;k++){
    memcpy(&aold[0], &a[0], r*sizeof(long));
      // mit c multiplizieren ergibt neues y
    cyMultiply(c, a, aold, r,base);
      // sind Koeffizienten von xk in Basis q
    buffer[k]=CoeffToDouble(base, a, r);
  }
  free(a);a=NULL;
  free(aold);aold=NULL;
  return d;
}

long Faure::ExitData() {
  SimulationSeqsBase::ExitData();
  if (c!=NULL) {free(c);c=NULL;}
  return 0;
}


// ****************************************************************************
// Netz implements (0,s)-Sequences using hyperderivatives and Lecot's
// recursive construction
// ****************************************************************************

Netz::Netz(long*b, long bas, long dm,long iterations, long lamb,
    char* ex,char* nm):SimulationSeqsBase(b, dm, iterations,0,0, ex, nm){
  base=bas;
  lambda=lamb;
  nu=power((long)base,(long)lambda);
  ncmax=(long)(8*sizeof(long)-(double)lambda*log(base)/log(2))+1;
  ncoeff=(long*)calloc(ncmax, sizeof(long));
  ncurr=0;
  nprime=0;
  InitMethod(b, dm, iterations, 0,0, ex, nm);
}
long Netz::InitData(long genau, double genau1){
  long p=0, bl=0, ind=0;
  long indexoffset=0;
  long i=0, j=0, l=0, m=0, tmp=0;
  // just store the coefficients in an nuŒ lambda rectangle.
  // Many of them are 0, but much less than half of them, so
  // there is not much to gain by trying to save only the non-zero
  // ones. This would just require an enormous calculation effort...
//  ypjilen=(long)(nu*lambda + 1 - (1-nu)/(1-base) )*dim;
  ypjilen=(long)(nu*lambda*dim);
  ypji=(long*)calloc(ypjilen, sizeof(long));
  for (p=0;p<base;p++) {
    for (i=0; i<dim; i++) {
      setypji(p,0,i,p%base);
    } //i=0..dim
  } //p=0..b

  for (l=1; l<lambda; l++) {
    for (m=1; m<base; m++) {
      bl=power(base,l);
      indexoffset=m*bl*dim*lambda;
      for (p=m*bl; p<(m+1)*bl; p++) {
        ind=getindex(p, 0, 0);
//        indexoffset=1+p+power(base, l)*(-1+l*m)-l*power(base, l-1);
        for (j=0; j<l; j++) {
          for (i=0; i<dim; i++) {
            tmp=ypji[ind-indexoffset]; //getypji(p-m*bl, j,i)
            tmp+=(long)(binom(l,j))*(long)(power(bases[i], l-j))*(long)(m);

            setypji(ind, tmp);
            ind++;
          } //i=0...s (Dimension)
        } //j=0...l
        for (i=0;i<dim; i++) {
```

```
            setypji(ind, m);
            ind++;
          }
        } //p=m*b^l...(m+1)*b^l
      } // m=1...b-1
    } // l=1...lambda
    return 0;
};

long Netz::getindex(long p, long j, long i) {
  return i+dim*(p*lambda+j);
}
long Netz::getypji(long index) {
  if (0<=index<ypjilen) return ypji[index];
  else return 0;
}
long Netz::getypji(long p, long j, long i) {
  long index=getindex(p,j,i);
  if (0<=index<ypjilen) return ypji[index];
  else return 0;
}
long Netz::setypji(long index, long value) {
  long oldval;
  if (0<=index<ypjilen) {
    oldval=ypji[index];
    ypji[index]=value % base;
    return oldval;
  } else return -1;
}

long Netz::setypji(long p, long j, long i, long value) {
  long index=getindex(p,j,i);
  return setypji(index, value);
}
long Netz::CalculateNextElement(long nr, double*buffer, long bufflen){
  long k, ypjinew=0, i=0, j=0;
  long index=0;
  for (i=0; i<dim; i++) buffer[i]=0;
  if (nr>=nu) {
    if ((ncurr*nu>nr)||(nr>=(ncurr+1)*nu)) {
      ncurr=(long)nr/nu;
      nprime=LongToCoeff(ncurr, base, ncoeff, ncmax);
    }
    for (j=0; j<lambda+nprime; j++){
      for (i=0; i<dim; i++) {
        if (j>=lambda) ypjinew=0;
        else ypjinew=getypji(nr-ncurr*nu, j, i);

        for (k=max(j, lambda); k<lambda+nprime; k++) {
          ypjinew+=(long)((long)binom(k,j)*(long)power(bases[i],k-j)*
              (long)ncoeff[k-lambda]);
        } //k=max()...lambda+nprime
        ypjinew=ypjinew % base;
        buffer[i]+=(double)((double)ypjinew/(double)power(base,j+1));
      } //i=1..s
    } //j=1...lambda+nprime+1
  } else {
    index=getindex(nr, 0,0);
    for (j=0; j<lambda; j++) {
      for (i=0; i<dim; i++) {
        buffer[i]+=(double)((double)ypji[index]/(double)power(base, j+1));
        index++;
      }
    }
  }
  return 0;
};

long Netz::ExitData(){
  SimulationSeqsBase::ExitData();
  if (ncoeff!=NULL) {free(ncoeff);ncoeff=NULL;}
  if (ypji!=NULL) {free(ypji);ypji=NULL;}
  return 0;
};


// ****************************************************************************
// TSSequence implements (0,s)-sequences. The construction is based on
// irreducible polynomials.
// ****************************************************************************

TSSequence::TSSequence(long*b, long bas, long dm,long iterations, long lamb,
    char* ex,char* nm):SimulationSeqsBase(b, dm, iterations,0,0, ex, nm){
  base=bas;
  maxlog=1floor(log(MAXLONG)/log(base))+1;
  lambda=lamb;
  c=NULL;
  poly=NULL;
  poly=(polynomial**)calloc(dim, sizeof(polynomial*));
  InitMethod(b, dm, iterations, 0,0, ex, nm);
}
long TSSequence::InitPolys(long dim,long base){
  // ToDo: Find the polynomials that are to be used for the (t,s) Sequence
  return 0;
}
```

```
long TSSequence::InitData(long genau, double genau1){
  long*v;
  long q,u,m;
  long i,j,k,l,r;
  polynomial*b;
  InitPolys(dim, base);

  if (c!=NULL) {free(c);c=NULL;}
  c=(long*)calloc(dim*maxlog*maxlog, sizeof(long));

  for (i=0; i<dim; i++) {
    j=0;
    q=-1;
    u=poly[i]->Ordnung();
    b=poly[i];
    v=(long*)calloc(maxlog+poly[i]->Ordnung()-2 +1, sizeof(long));
    for (j=1; j<maxlog; j++) {
      // (2) aus dem Algorithmus bei Radovic
      if (u==poly[i]->Ordnung()) {
        q++;
// ToDo:          b=b*poly[i];
        m=poly[i]->Ordnung()*(q+1);
        // v_i berechnen:
        for (k=0; k<m-1; k++) {
          v[k]=0;
        }
        v[m-1]=1;
        for (k=m; k<=maxlog+poly[i]->Ordnung()-2; k++) {
          v[k]=0;
          for (l=1; l<=m; l++) v[k]+=poly[i]->coeff(m-1)*v[k-l];
          v[k]=v[k]%base;
        }
      }

      // (3) aus dem algorithmus bei Radovic
      for (r=0; r<maxlog; r++)
        c[maxlog*maxlog*i + (j-1)*maxlog +r] = v[r-u];
      u++;
    }
    free(v);v=NULL;
  }
  return 0;
};

long TSSequence::CalculateNextElement(long nr, double*buffer, long bufflen){
  long*ncoeff=(long*)calloc(maxlog, sizeof(long));
  long maxcoeff=LongToCoeff(nr /* + base^lambda*/, base, ncoeff, maxlog);
  long d=0;
  long Q=0;

  for (long i=0; i<dim; i++) {
    Q=0;
    for (long j=0; j<maxlog; j++) {
      d=0;
      for (long r=0; r<min(maxlog, maxcoeff+1); r++) {
        d+=c[i*maxlog*maxlog + j*maxlog + r] * ncoeff[r];
        d=d%base;
      }
      Q*=base;
      Q+=d;
    }
    buffer[i]=(double)Q/(double)power(base, maxlog);
  }
  free(ncoeff);ncoeff=NULL;
  return dim;
};

long TSSequence::ExitData(){
  SimulationSeqsBase::ExitData();
  if (c!=NULL) {free(c);c=NULL;}
  for (long i=0; i<dim; i++) {
    if (poly[i]!=NULL) free(poly[i]);
  }
  free(poly);poly=NULL;
  return 0;
};
```

Literaturverzeichnis

[ABC71] *ABC Physik.* Verlag Harri Deutsch, 1971.

[Ata00] Atanassov, Emanouil I.: *On the Discrepancy of the Halton Sequences*, 2000. To be published.

[Bab86] Babovsky, Hans: *On a Simulation Scheme for the Boltzmann Equation.* Math. Meth. in the Appl. Sci., 8:223 – 233, 1986.

[BGN+90] Babovsky, Hans, Frank Gropengiesser, Helmut Neunzert, Jens Struckmeier und Bernd Wiesen: *Application of well-distributes sequences to the numerical simulation of the Boltzmann equation.* Journal of Computational and Applied Mathematics, 31:15 – 22, 1990.

[BI89] Babovsky, Hans und Reinhard Illner: *A Convergence Proof for Nanbu's Simulation Method for the Full Boltzmann Equation.* SIAM J. Numer. Anal., 26(1):45 – 65, February 1989.

[Bir76] Bird, Graeme A.: *Molecular gas dynamics.* Clarendon Press, Oxford / Oxford University Press, London, 1976.

[Bir94] Bird, Graeme A.: *Molecular Gas Dynamics and the Direct Simulation of Gas Flows.* Clarendon Press, Oxford, 1994.

[BT85] Bellomo, Nicola und Giuseppe Toscani: *On the Cauchy problem for the nonlinear Boltzmann equation. Global existence, uniqueness and asymptotic stability.* J. Math. Phys, 26(2):334 – 338, February 1985.

[BY75] Belotserkovskiy, O.M. und V.Ye. Yanitskiy. *Zhurn. Vych. Mat. i Mat. Fiz.*, 15:1195 –, 1975.

[Car33] Carleman, T. *Acta Mathematica*, 60:91 –, 1933.

[Cer88] Cercignani, Carlo: *The Boltzmann Equation and Its Applications*, Band 67 der Reihe *Applied mathematical sciences.* Springer-Verlag, 1988. Früher erschienen als "Theory and Application of the Boltzmann Equation", Scottish Academic Press Ltd., 1975.

[CL98] Coulibaly, Ibrahim und Christian Lécot: *Monte Carlo and quasi-Monte Carlo algorithms for a linear integro-differential equation.* In: Niederreiter, Harald, Peter Hellekalek, Gerhard Larcher und Peter Zinterhof (Herausgeber): *Monte Carlo and Quasi-Monte Carlo Methods 1996*, Nummer 127 in *Lecture Notes in Statistics*, Seiten 176 – 188. Springer - Verlag New York, Inc, 1998.

[Des78] Deshepande, S.M. Report 78 FM 4, Dept. Aero. Engng., Indian Institute of Science, 1978.

[DT97] Drmota, Michael und Robert F. Tichy: *Sequences, Discrepancies and Applications.* Nummer 1651 in *Lecture notes in Mathematics.* Springer, 1997.

[Fau81] Faure, Henri: *Discrépance de suites associées à un système de numération (en dimension un).* Bull. Soc. Math. France, 109:143–182, 1981.

[Fau82] Faure, Henri: *Discrépance de suites associées à un système de numération (en dimension s).* Acta Arith., 41:337–351, 1982.

[Fau98] FAURE, HENRI: *Discrepancy lower bounds for special quasi-random sequences.* In: NIEDERREITER, HARALD, PETER HELLEKALEK, GERHARD LARCHER und PETER ZINTERHOF (Herausgeber): *Monte Carlo and Quasi-Monte Carlo Methods 1996*, Nummer 127 in *Lecture Notes in Statistics*, Seiten 232 – 237. Springer - Verlag New York, Inc, 1998.

[Fau93] FAURE, HENRI: *Good permutations for extreme discrepancy.* J. Number Theory, 1992/93.

[Gra58] GRAD, HAROLD: *Principles of the Kinetic Theory of Gases.* In: FLÜGGE, S. (Herausgeber): *Thermodynamik der Gase*, Band XII der Reihe *Handbuch der Physik (Encyclopedia of Physics)*. Springer-Verlag, 1958.

[Hal60] HALTON, JOHN H.: *On the efficiency of certain quasi-random points in evaluating multi-dimensional integrals.* Numer. Math., 2:84 – 90, 1960.

[Har71] HARRIS, STEWART: *An Introduction to the Theory of The Boltzmann Equation.* Holt, Rinehart and Winston, Inc., 1971.

[HJK98] HOOGLAND, JIRI, FRED JAMES und RONALD KLEISS: *Quasi-Monte Carlo, Discrepancies and Error Estimates.* In: NIEDERREITER, HARALD, PETER HELLEKALEK, GERHARD LARCHER und PETER ZINTERHOF (Herausgeber): *Monte Carlo and Quasi-Monte Carlo Methods 1996*, Nummer 127 in *Lecture Notes in Statistics*, Seiten 266 – 276. Springer - Verlag New York, Inc, 1998.

[IN87] ILLNER, REINHARD und HELMUT NEUNZERT: *On Simulation Methods for the Boltzmann Equation.* Transport Theory and Statistical Physics, 16(2&3):141 – 154, 1987.

[IS84] ILLNER, REINHARD und MARVIN SHINBROT: *The Boltzmann Equation: Global Existence for a Rare Gas in an Infinite Vacuum.* Commun. Math. Phys., 95:217 – 226, 1984.

[Kac59] KAC, MARK: *Probability and Related Topics in Physical Sciences*, Band I der Reihe *Lectures in Applied Mathematics.* Interscience Publishers, John Wiley & Sons, Inc., 1959.

[Kai99] KAINHOFER, REINHOLD: *Die Simulation der Wärmeleitungsgleichung*, 1999. EDV-Projekt zum Studium der Technischen Mathematik.

[Kou70] KOURA, KATSUHISA: *ToDo.* Physics of Fluids, 13:1457, 1970.

[KW77] KROOK, MAX und TAI TSUN WU: *Exact solutions of the Boltzmann equation.* The Physics of Fluids, 20(10):1589 – 1595, October 1977.

[LC96a] LÉCOT, CHRISTIAN und IBRAHIM COULIBALY: *Quasi-Monte Carlo methods for kinetic equations.* Prépublication du L.A.M.A. 96-12A, Université de Savoie, Laboratoire de Mathématiques, December 1996.

[LC96b] LÉCOT, CHRISTIAN und IBRAHIM COULIBALY: *A quasi-Monte Carlo scheme using nets for a linear Boltzmann equation.* Prépublication du L.A.M.A. 96-06A, Université de Savoie, Laboratoire de Mathématiques, June 1996.

[Léc89a] LÉCOT, CHRISTIAN: *A Direct Simulation Monte Carlo Scheme and Uniformly Distributed Sequences for Solving the Boltzmann Equation.* Computing, 41:41 – 57, 1989.

[Léc89b] LÉCOT, CHRISTIAN: *Low discrepancy sequences for solving the Boltzmann equation.* Journal of Computational and Applied Mathematics, 25:237 – 249, 1989.

[Léc91] LÉCOT, CHRISTIAN: *A Quasi-Monte Carlo Method for the Boltzmann Equation.* Mathematics of Computation, 56(194):621 – 644, April 1991.

[LK99] LÉCOT, CHRISTIAN und F. EL KHETTABI: *A Quasi-Monte Carlo Simulation of the Boltzmann equation.* Prépublication du LAMA 99-10a, Université de Savoie, Laboratoire de Mathématiques, 1999.

[MMN95] MULLEN, GARY L., ARIJIG MAHALANABIS und HARALD NIEDERREITER: *Tables of (t, m, s)-net and (t, s)-sequence parameters.* In: NIEDERREITER, HARALD und PETER JAN-SHYONG SHINE (Herausgeber): *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, Nummer 106 in *Lecture Notes in Statistics*, Seiten 58 – 86. Springer - Verlag New York, Inc, 1995.

[Mor54] MORGENSTERN, D. *Proc. Nat. Acad. Sci. U.S.A.*, 40:719, 1954.

[Nan80] NANBU, KENICHI: *Direct Simulation Scheme Derived from the Boltzmann Equation. I. Monocomponent Gases.* Journal of the Physical Society of Japan, 49(5):2042 – 2049, November 1980.

[Nan83] NANBU, KENICHI: *Interrelations between Various Direct Simulation Methods for Solving the Boltzmann Equation.* Journal of the Physical Society of Japan, 52(10):3382 – 3388, October 1983.

[Nie78] NIEDERREITER, HARALD: *Quasi-Monte Carlo Methods and Pseudo-Random Numbers.* Bulletin of the American Mathematical Society, 84(6):957 – 1041, November 1978.

[Nie87] NIEDERREITER, HARALD: *Point Sets and Sequences with Small Discrepancy.* Monatsheft für Mathematik, 104:273 – 337, 1987.

[Nie88a] NIEDERREITER, HARALD: *Low discrepancy and low-dispersion sequences.* J. Number Theory, 30:51–70, 1988.

[Nie88b] NIEDERREITER, HARALD: *Quasi-Monte Carlo methods for multidimensional numerical integration.* In: H. BRASS, G. HÄMMERLIN (Herausgeber): *J.Numerical Integration III*, Band 85 der Reihe *International Series of Numerical Mathematics*, Seiten 157–171. Birkhäuser, Basel, 1988.

[Nie92] NIEDERREITER, HARALD: *Random Number Generation and Quasi-Monte Carlo Methods.* Nummer 63 in *CBMS-NSF regional conference series in applied mathematics.* SIAM Philadelphia, 1992.

[Rad95] RADOVIĆ, IGOR: *Quasi-Monte Carlo Methoden zur numerischen Integration: Vergleich verschiedener Punktfolgen kleiner Diskrepanz.* Diplomarbeit, Technische Universität Graz, Mai 1995.

[Sch98] SCHMID, WOLFGANG CH.: *Shift-Nets: a New Class of Binary Digital (t, m, s)-Nets.* In: NIEDERREITER, HARALD, PETER HELLEKALEK, GERHARD LARCHER und PETER ZINTERHOF (Herausgeber): *Monte Carlo and Quasi-Monte Carlo Methods 1996*, Nummer 127 in *Lecture Notes in Statistics*, Seiten 369 – 381. Springer - Verlag New York, Inc, 1998.

[Sob60] SOBOL', ILYA M.: *An accurate error estimate for multidimensional quadrature formulae for the functions of the class S_p.* Dokl. Akad. Nauk SSSR, 132:1041–1044, 1960. Auf Russisch.

[Sob67] SOBOL', ILYA M.: *The distribution of points in a cube and the approximate evaluation of integrals.* USSR Comput. Math. Math. Phys., 4(7):86 – 112, 1967.

[Uhl73] UHLENBECK, GEORGE E.: *The Validity and the Limitations of the Boltzmann-Equation.* In: COHEN, E.G.D. und WALTER THIRRING (Herausgeber): *The Boltzmann Equation, Theory and Applications*, Acta Physica Austriaca, Suppl. X, Seiten 107 – 119. Springer-Verlag, Wien, New York, 1973.

[Wil51] WILD, E.: *On Boltzmann's Equation in the kinetic Theory of Gases.* Proceedings of the Cambridge Philosophical Society, 47:602 – 609, 1951.

[Zin76] ZINTERHOF, PETER: *Über einige Abschätzungen bei der Approximation von Funktionen mit Gleichverteilungsmethoden.* Österr. Akad. Wiss. SB II, 185:121–132, 1976.

[Zwe84] ZWEIFEL, PAUL F.: *The Boltzmann Equation and its Properties.* In: CERCIGNANI, CARLO (Herausgeber): *Kinetic Theories and the Boltzmann Equation*, Nummer 1048 in *Lecture Notes in Mathematics*, Seiten 111 – 175. Springer Verlag, Berlin, Heidelberg, New York, Tokyo, 1984. Lectures given at the 1st 1981 Session of the C.I.M.E held at Motecatini, Italy.