Diplomarbeit
Diplom IA, Fachbereich Mathematik,
Fakultät für Mathematik, Informatik und Informationstechnik
FernUniversität in Hagen, Deutschland

Die Poissonität des Kollisionszählprozesses eines virtuellen Fluids

vorgelegt von

Paul Ruppen

Datum: 27. Mai 2006

Referent: Herr Prof. Dr. Otto Moeschlin

Inhaltsverzeichnis

1 Die Problemstellung **2**

2 Zum Modell eines Systems bewegter Mikrobestandteile **3**

3 Die statistischen Konzepte **10**
3.1 Der Permutationstest 10
3.2 Die Schätzmethoden 15
3.2.1 Das nicht-parametrische Schätzen von Verteilungen durch Kerndichteschätzer 16
3.2.2 Parametrische Schätzung der Wahrscheinlichkeitsverteilung 19
3.3 Die Exponentialverteilung und die Poissonität 21

4 Die Anwendung der statistischen Konzepte auf Zwischenkollisionszeiten **23**

5 Zusammenfassung **31**

6 Literatur **33**

7 Index der Symbole **34**

8 Anhänge **37**

1 Die Problemstellung

Ausgangspunkt ist ein reales Gas beziehungsweise dessen Modellierung durch Ludwig Boltzmann (1844 - 1906) als System bewegter elastischer Kugeln (Fluid) in einem Behältnis B, denen eine Newton Dynamik auferlegt ist, d.h. insbesondere dass beim Stoß zweier Kugeln oder beim Stoß einer Kugel mit der Wand von B stets das Prinzip der Energie- und Impulserhaltung gewahrt sein muss. Zusätzlich wird angenommen, dass beim Zusammenstoß zweier Mikrobestandteile oder bei Reflexionen an der Behälterwand keine Energie in Rotationsenergie der Molekel übergeht.

Ziel hier ist die Untersuchung des Stoßzählprozesses, d.h. des stochastischen Prozesses, der die Stöße der Kugeln zählt. Aufgrund anderer Erfahrungen kann vermutet werden, dass es sich dabei um einen Poisson-Prozess handelt, was sich aber durch die Laborphysik mindestens direkt nicht nachweisen läßt.

Einen Ausweg bietet die Experimentelle Stochastik, wo ein entsprechendes (zweidimensionales) stochastisches, dynamisches Modell auf dem Rechner implementiert wird, um dann interessierende Makro-Größen im statistischen Wortsinne zu schätzen. Die Zwischenstoßzeiten modellierenden Zufallsvariablen sind identisch verteilt gemäß einer durch das Fluid bestimmten Verteilung, die als eine solche mit stetiger Verteilungsfunktion unterstellt wird.

Es ist bekannt, das unabhängige, identisch exponentialverteilte Zufallsvariablen, welche Zwischenereigniszeiten als Werte annehmen, die Poissonität des Zählprozesses dieser Ereignisse implizieren. Entsprechend wird hier versucht nachzuweisen, dass die Zwischenstoßzeiten als Werte unabhängiger, identisch exponentialverteilter Zufallsvariablen betrachtet werden können.

Das Vorgehen kann wie folgt kurz beschrieben werden: Im *Kapitel 2* wird das implementierte Experiment, dessen Quellcode von den Betreuern der Diplomarbeit zur Verfügung gestellt wurde (Pascal-Code, s. *Beilage 8.1*), detailliert beschrieben. Im *Kapitel 3* werden die statistischen Konzepte eingeführt, die in der Folge verwendet werden, um zu überprüfen, ob die Zwischenkollisionszeiten als Werte von unabhängigen und identisch exponentialverteilten Zufallsvariablen betrachtet werden können.

Nach der Entwicklung der Konzepte werden diese im *Kapitel 4* auf computerexperimentell erzeugte Daten zur Anwendung gebracht. Es werden kurz die inhaltlichen Gründe für die Haltbarkeit der identischen Verteiltheit der Stichprobenzufallsvariablen diskutiert. Dann erfolgt eine Überprüfung der Unabhängigkeit der Zufallsvariablen. Schließlich wird für einen konkreten Datensatz eine konkrete Verteilung geschätzt, gemäß der die Zufallsvariablen, als deren Werte die Zwischenkollisionszeiten zu betrachten sind, verteilt sind.

Die vorliegende Arbeit beschränkt sich auf die Analyse der Kollisionen zwischen Mikrobestandteilen. Reflexionen an den Wänden des Behälters

werden nicht untersucht. Für Reflexionen oder Ereignisse, wobei Ereignisse Kollisionen oder Reflexionen sind, könnten dieselben Studien angestellt werden. Die vorliegende Analyse kann als exemplarisch für diese zwei weiteren Fälle betrachtet werden.

Inhaltlich ist die Diplomarbeit im Rahmen der Schrift *Experimental Stochastics in Physics* (Moeschlin, Grycko, 2004) zu sehen. Dort werden die Anwendungen der hier behandelten Schätzungen (Kerndichteschätzer; parametrischer Schätzer) als *Experiment 3.1.* vorgestellt und die Resultate diskutiert (Moeschlin, Grycko, 2004, S. 23 ff.). Das Experiment ist auf der zu Moeschlin, Grycko (2004) mitgelieferten CD selber reproduzierbar. Mit Hilfe der gleichen Methode wie hier wird dort überprüft, ob die Zeitabstände zwischen zwei paarweisen Kollisionen von Mikrobestandteilen Werte exponentialverteilter Zufallsvariablen sind, wobei in der dortigen Darstellung die Konsistenz der Schätzer im Vordergrund steht. Im Unterschied zu dort fällt diese Gewichtung hier weg und es wird hier zusätzlich die Unabhängigkeit der Zufallsvariablen mit dem Permutationstest überprüft.

Ich möchte mich bei Herrn Dr. Prof. Otto Moeschlin für die Betreuung der Diplomarbeit bedanken. Dank gebührt auch Herrn Dr. Eugen Grycko, der die Arbeit vor allem auch bezüglich technischer Details begleitete.

2 Zum Modell eines Systems bewegter Mikrobestandteile

In diesem Kapitel wird das dynamische Modell, das auf dem Computer implementiert wird, dargelegt: den Ausführungen in Moeschlin, Grycko (2004) und Moeschlin et al. (2003) folgend, wird erläutert, wie die Zwischenreflexionszeiten, die Zwischenkollisionszeiten und die Geschwindigkeitsvektoren von Molekeln nach Reflexionen oder Kollisionen zu berechnen sind. Zusätzlich zur angegebenen Literatur wird der von den Betreuern zur Verfügung gestellte Quellcode berücksichtigt (s. *Beilage 8.1*).

Unterstellt wird ein Behälter B, der wie folgt definiert ist:

$$B := \{(x_1, x_2) \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq (R + r)^2\} \text{ für } r, R \in (0, \infty), r < R. \quad (1)$$

Behälter sind damit Kreisflächen in der zweidimensionalen Ebene mit dem Mittelpunkt $(0, 0)$ und mit dem Radius $R+r$ (im Experiment $R = 1[m]$). Im Behälter B kommen N Objekte vor (im Experiment $N = 2000$), die Mikrobestandteile genannt werden und die folgende Eigenschaften haben: Jeder Mikrobestandteil ist eine Kreisscheibe mit einem Radius $r \in (0, R)$, wobei sich die Mikrobestandteile nicht überlappen dürfen. Der Radius r der Mikrobestandteile wird in Hinblick auf das inhaltlichen Gebiet, das experimentell nachgestellt wird (Gas in einem Behälter), im Verhältnis zu R klein gewählt. Im Experiment wird für alle Mikrobestandteile ein identischer Radius von

$$r = \sqrt{\frac{\pi}{500 \cdot N \cdot 2 \cdot \sqrt{3}}}[m] \tag{2}$$

gewählt. Durch die Formel wird garantiert, dass alle Kreisflächen in B Platz haben, ohne sich zu überlappen, und dass zwischen ihnen genügend Zwischenraum ist, so dass Dynamik überhaupt experimentell dargestellt werden kann. Letzteres wird durch die Division durch 500 erreicht.

Für alle Mikrobestandteile wird eine identische Masse von $m \in (0, \infty)$ vorausgesetzt – im Experiment $\frac{39.95}{6.02 \cdot 10^{26}}[Kg] = 1.6611 \cdot 10^{-27}[Kg]$. Dies entspricht der Masse eines Argonatoms (*Formeln und Tafeln*, 1984).

Es wird davon ausgegangen, dass die Mikrobestandteile einer Newtonschen Dynamik folgen: Für jeden Mikrobestandteil $i \in \mathbb{N}_N$ ist der Impulsvektor $u^{(i)} := \left(u_1^{(i)}, u_2^{(i)}\right) \in \mathbb{R}^2$ definiert als das Produkt

$$u^{(i)} := m^{(i)} v^{(i)}, \tag{3}$$

wobei $m^{(i)} \in \mathbb{R}_+ - \{0\}$ die Masse des Mikrobestandteils i ist und $v^{(i)} := \left(v_1^{(i)}, v_2^{(i)}\right) \in \mathbb{R}^2$ der Geschwindigkeitsvektor von i. Die kinetische Energie von i ist definiert als der Wert der Abbildung

$$\begin{aligned} E &: \mathbb{R}^2 \to \mathbb{R}_+ \\ E(u^{(i)}) &:= \frac{1}{2m^{(i)}} \left\langle u^{(i)}, u^{(i)} \right\rangle \end{aligned} \tag{4}$$

Dabei ist $\left\langle u^{(i)}, u^{(i)} \right\rangle$ das Standardskalarprodukt. Bei der Reflexion eines Mikrobestandteils an der Behälterwand bleibt der Betrag des Impulsvektors unverändert. Deshalb bleibt auch die Energie des Mikrobestandteils erhalten. Beim Zusammenstoß zweier Mikrobestandteile bleibt die Summe der Impulse der beteiligten Molekel unverändert, d.h.

$$u^{(i)} + u^{(j)} = \bar{u}^{(i)} + \bar{u}^{(j)}, \tag{5}$$

wobei $u^{(i)}, u^{(j)}$ die Impulsvektoren vor dem Zusammenstoß und $\bar{u}^{(i)}, \bar{u}^{(j)}$ die Impulsvektoren nach dem Zusammenstoß sind. Entsprechend bleibt auch die Gesamtenergie der beteiligten Mikrobestandteile erhalten, d.h.

$$E(u^{(i)}) + E(u^{(j)}) = E(\bar{u}^{(i)}) + E(\bar{u}^{(j)}). \tag{6}$$

Im Experiment werden die Mikrobestandteile auf das Innere von B mit Hilfe eines Zufallsgenerators verteilt: dem Mittelpunkt jedes Mikrobestandteiles i wird ein Punkt $x^{(i)} = (x_1^{(i)}, x_2^{(i)}) \in B' := \{(x_1, x_2) \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq R^2\}$ für obiges R zugewiesen, so dass sich die Mikrobestandteile

nicht überlappen. Es handelt sich um den Ausgangsort des Mikrobestandteils i zur Zeit 0. Dies kann durch $x^{(i)}(0) = (x_1^{(i)}(0), x_2^{(i)}(0))$ ausgedrückt werden. Es sind verschiedene Ausgangsverteilungen wählbar. Es scheint sinnvoll zu sein, für die Mittelpunkte der Mikrobestandteile eine uniforme Ausgangsverteilung auf B' zu wählen. Uniform verteilte Punkte auf B' erhält man mit Hilfe eines Zufallsgenerators, der Zufallszahlen liefert, die als Werte von $U(0,1)-$verteilten unabhängigen Zufallsvariablen betrachtet werden können. Für solche Zufallszahlen wird die Abkürzung „rand_i“ verwendet. Im von den Betreuern zur Verfügung gestellten Quellcode wird die folgende Berechnungsformeln für auf B' uniform verteilte Zahlenpaare geliefert:

$$\begin{aligned} x_1^{(i)}(0) &= R \cdot \sqrt{\text{rand}_1} \cdot \cos(2\pi\, \text{rand}_2) \\ x_2^{(i)}(0) &= R \cdot \sqrt{\text{rand}_1} \cdot \cos(2\pi\, \text{rand}_2) \end{aligned} \tag{7}$$

Ein Beweis der Angemessenheit dieser Formeln folgt der Beweismethode zur Box-Mueller-Methode für die Erzeugung zweidimensional normalverteilter Daten in Moeschlin et al. (1998, S. 65 f.). Alternativ könnte man Punkte $(R\,(2\,\text{rand}_1 - 1)\,, R\,(2\,\text{rand}_2 - 1))$ erzeugen, wobei dann nur die Punkte berücksichtigt werden, die in B' liegen.

Jedem Mikrobestandteil i wird ein zweidimensionaler Geschwindigkeitsvektor $v^{(i)} \in \mathbb{R}^2$ zugeordnet. Es handelt sich um die Ausgangsgeschwindigkeit des Mikrobestandteiles zur Zeit 0. Diese wird symbolisiert durch $v^{(i)}(0) = (v_1^{(i)}(0), v_2^{(i)}(0))$. Die Komponenten des Geschwindigkeitsvektors werden wiederum mit Hilfe eines Zufallsgenerators zugeordnet − wobei im Prinzip verschiedene Verteilungen als Ausgangsverteilungen möglich sind. Im Experiment werden die Komponenten der Geschwindigkeitsvektoren nach $N(0, \sigma^2 I_2)$ verteilt (I_2 = Identitätsmatrix der Dimension 2). Dabei wird die Box-Mueller-Methode (Moeschlin et al., 1997, S. 64) verwendet:

$$\begin{aligned} v_1^i(0) &= \sigma\sqrt{-2\ln(\text{rand}_1)} \cdot \cos(2\pi \cdot \text{rand}_2) \\ v_2^i(0) &= \sigma\sqrt{-2\ln(\text{rand}_1)} \cdot \sin(2\pi \cdot \text{rand}_2) \end{aligned} \tag{8}$$

Die Standardabweichung σ ist mit der Temperatur des Systems, die vom Nutzer frei gewählt werden darf, wie folgt verknüpft:

$$\sigma = \sqrt{\frac{k_B T}{m}} \tag{9}$$

mit T = Temperatur in Kelvin; m = Masse; $k_B = 1.38 \cdot 10^{-23} \frac{[J]}{[K]}$ (Boltzmann-Konstante). Je größer die Temperatur, desto größer ist entsprechend die Streuung der Komponenten der Geschwindigkeitsvektoren.

Damit wird der Zustand des Gesamtsystems zum Zeitpunkt 0 beschrieben durch

$$\left(x^{(1)}(0), ..., x^{(N)}(0), v^{(1)}(0), ..., v^{(N)}(0)\right) \in B'^N \times \mathbb{R}^{2N}. \tag{10}$$

Nach der Zeitdauer t und wenn kein Zusammenstoß und keine Reflexion an der Wand vorliegt, ist das System im folgenden Zustand:

$$\left(x^{(1)}(0)+tv^{(1)}(0),...,x^{(N)}(0)+tv^{N}(0),v^{(1)}(0),...,v^{(N)}(0)\right)= \tag{11}$$
$$\left(x^{(1)}(t),...,x^{(N)}(t),v^{(1)}(t),...,v^{(N)}(t)\right)\in B'^{N}\times\mathbb{R}^{2N}$$

Diese Fortschreibung des Systems wird unterbrochen durch Zusammenstö ße von Mikrobestandteilen oder durch die Reflexion von Mikrobestandteilen an der Wand. Sei ∂B die Behälterwand. Die Zeitdauer bis zur Reflexion eines Mikrobestandteiles i an der Behälterwand ergibt sich allgemein durch die kleinste positive Lösung der Gleichung

$$d(x^{(i)}(0)+tv^{(i)}(0),\partial B)=r, \tag{12}$$

wobei $d(x,y):=\sqrt{(x_1-y_1)^2+(x_2-y_2)^2}$ für $x:=(x_1,x_2)$ und $y:=(y_1,y_2)$ die euklidische Distanz auf $\mathbb{R}^2$ bezeichnet. Im vorliegenden Spezialfall eines kreisrunden, zweidimensionalen Behälters mit dem Radius $R+r$ und Molekel mit dem Radius r erhält man diese Zeitdauer t für ein Molekel i durch die positive Lösung des Gleichungssystems

$$x^{(i)}(0)+tv^{(i)}(0)=R^2. \tag{13}$$

Dies ergibt

$$t=-\frac{\left\langle x^{(i)},v^{(i)}\right\rangle}{\left\|v^{(i)}\right\|^2}\pm\sqrt{\left(\frac{\left\langle x^{(i)},v^{(i)}\right\rangle}{\left\|v^{(i)}\right\|^2}\right)^2-\frac{\left\|x^{(i)}\right\|^2-R^2}{\left\|v^{(i)}\right\|^2}},\ \text{wenn } t>0, \tag{14}$$

mit $\left\|v^{(i)}\right\|^2:=\left(v_1^{(i)}\right)^2+\left(v_2^{(i)}\right)^2$.

Die Zeit t bis zur Kollision zweier Mikrobestandteile i und j erhält man durch die kleinste positive Lösung der Gleichung

$$d(x^{(i)}(0)+tv^{(i)}(0),x^{(j)}(0)+tv^{(j)}(0))=2r. \tag{15}$$

Dies ergibt

$$t=\frac{-b\pm\sqrt{b^2-4ac}}{2a},\ \text{wenn } t>0,\ \text{mit} \tag{16}$$
$$a:=\left(d(v^{(i)},v^{(j)})\right)^2,$$
$$b:=2\left\langle\left(x^{(j)}-x^{(i)}\right),\left(v^{(j)}-v^{(i)}\right)\right\rangle,$$
$$c:=\left(d\left(x^{(i)},x^{(j)}\right)\right)^2-4r^2.$$

Gibt es keine positive, reelle Lösung, bedeutet dies, dass die Mikrobestandteile sich parallel oder voneinander weg bewegen. Damit erfolgt bis zum nächsten Kollisions- oder Reflexionsereignis anderer Mikrobestandteile kein Zusammenstoß.

Der erste Durchlauf des Experimentes kann nun wie folgt beschrieben werden: Das erste Kollisionsereignis zweier Mikrobestandteile nach dem Zeitpunkt 0 findet im Gesamtsystem statt zum Zeitpunkt

$$\bar{t}_1^k := \min(\{t_{ij} \mid i,j \in \mathbb{N}_N, i<j \text{ und } \\ \text{zu } t_{ij} \text{ findet eine Kollision von } i \text{ und } j \text{ statt}\}) \tag{17}$$

– sofern eine Kollision zu Stande kommt und sich nicht vorher eine Reflexion ereignet. Die erste Reflexion im Gesamtsystem findet statt zum Zeitpunkt

$$\bar{t}_1^r := \min(\{t_i \mid i \in \mathbb{N}_N \text{ und zu } t_i \text{ findet eine Reflexion } \\ \text{von } i \text{ an der Behälterwand statt}\}) \tag{18}$$

– sofern sich nicht vorher eine Kollision ereignet. Das erste Ereignis nach 0 im Gesamtsystem findet damit zu

$$\bar{t}_1 := \min\left(\{\bar{t}_1^k, \bar{t}_1^r\}\right) \tag{19}$$

statt.

Ist $\bar{t}_1$ ein Reflexionsereignis, wird der neue Geschwindigkeitsvektor des betroffenen Mikrobestandteils errechnet. Der Betrag der Geschwindigkeit bleibt konstant (Erhaltung von Impuls und Energie). Die Richtung wird durch die Richtung des Aufpralls bestimmt: trifft der Mikrobestandteil im Winkel α auf die Tangente im Reflexionspunkt, so geht er im Winkel $-\alpha$ von dieser weg (s. Moeschlin, Grycko, 2004, S. 14).

Satz 1 *Der Geschwindigkeitsvektor $v^{(i)\prime} \in \mathbb{R}^2$ des Mikrobestandteils i, dessen Mittelpunkt sich bei der Reflexion im Punkt $x^{(i)} \in \mathbb{R}^2$ befindet, ist nach der Reflexion gegeben durch $v^{(i)\prime} = v^{(i)} - \frac{2}{R}\left\langle x^{(i)}, v^{(i)}\right\rangle \frac{x^{(i)}}{R}$*
($v^{(i)} \in \mathbb{R}^2$ = Geschwindigkeitsvektor von i vor der Reflexion, R ist Radius des Teils des Behältnisses, auf den die Mittelpunkte der Molekel fallen können).

Beweis. Der Vektor $v^{(i)}$ lässt sich als Summe eines Vektors $ax^{(i)}$ $(a \in \mathbb{R})$ und einem auf $x^{(i)}$ senkrechten Vektor $y^{(i)} \in \mathbb{R}^2$ darstellen, d.h.

$$v^{(i)} = ax^{(i)} + y^{(i)}$$
$$\left\langle x^{(i)}, y^{(i)}\right\rangle = 0.$$

Mit $y^{(i)} = v^{(i)} - ax^{(i)}$ erhält man daraus:
$\left\langle x^{(i)}, y^{(i)} \right\rangle = \left\langle x^{(i)}, v^{(i)} - ax^{(i)} \right\rangle = \left\langle x^{(i)}, v^{(i)} \right\rangle - a \left\langle x^{(i)}, x^{(i)} \right\rangle = 0 \Longleftrightarrow$
$a = \frac{\left\langle v^{(i)}, x^{(i)} \right\rangle}{\left\| x^{(i)} \right\|^2}$.
Damit ergibt sich $y^{(i)} = v^{(i)} - \frac{\left\langle v^{(i)}, x^{(i)} \right\rangle}{\left\| x^{(i)} \right\|^2} x^{(i)}$.
Da der Mikrobestandteil im Winkel $-\alpha$ von der Tangente im Reflexionspunkt wegfliegt, wenn er im Winkel a auf die Tangente auftrifft, gilt offensichtlich $v'^{(i)} = -v^{(i)} + 2\left(v^{(i)} - \frac{\left\langle x^{(i)}, v^{(i)} \right\rangle}{\left\| x^{(i)} \right\|} \frac{x^{(i)}}{\left\| x^{(i)} \right\|}\right) = v^{(i)} - 2\frac{\left\langle x^{(i)}, v^{(i)} \right\rangle}{\left\| x^{(i)} \right\|} \frac{x^{(i)}}{\left\| x^{(i)} \right\|}$.
Da $R^2 = \left(x_1^{(i)}\right)^2 + \left(x_2^{(i)}\right)^2$, gilt $\left\| x^{(i)} \right\| = R$. Damit erhält man dann: $v'^{(i)} = v^{(i)} - \frac{2}{R} \left\langle x^{(i)}, v^{(i)} \right\rangle \frac{x^{(i)}}{R}$. ■

Erfolgt eine Kollision, so berechnet man die neuen Geschwindigkeitsvektoren der beteiligten Mikrobestandteile mittels des Impulses u. Die Richtung des Impulsaustausches ist gegeben durch den Vektor

$$e := \frac{x^{(i)}(\bar{t}_1) - x^{(j)}(\bar{t}_1)}{\left\| x^{(i)}(\bar{t}_1) - x^{(j)}(\bar{t}_1) \right\|}. \tag{20}$$

Auf Grund der Impulserhaltung gilt für die Impulsvektoren nach der Kollision

$$\bar{u}^{(i)} = u^{(i)} + \xi e \text{ und } \bar{u}^{(j)} = u^{(j)} - \xi e, \text{ mit } \xi \in \mathbb{R}, \tag{21}$$

denn dann gilt $\bar{u}^{(i)} + \bar{u}^{(j)} = u^{(i)} + u^{(j)}$.

Die unbekannte skalare Größe ξ kann auf Grund der vorausgesetzten Energieerhaltung errechnet werden. In diesem Fall muss nämlich gelten

$$E\left(\bar{u}^{(i)}\right) + E\left(\bar{u}^{(j)}\right) = E\left(u^{(i)}\right) + E\left(u^{(j)}\right). \tag{22}$$

Aus (21) und (22) kann ξ berechnet werden:

Satz 2 *Seien $u^{(i)}, \bar{u}^{(i)} \in \mathbb{R}^2$ die Impulsvektoren des Molekels i vor beziehungsweise nach der Kollision, $v^{(i)}, v^{(j)} \in \mathbb{R}^2$ die Geschwindigkeitsvektoren der Molekel i und j vor der Kollision, sei E definiert wie unter (4) und m die Masse jedes der Molekel i und j. Aus $\bar{u}^{(i)} = u^{(i)} + \xi e$, $\bar{u}^{(j)} = u^{(j)} - \xi e$ und $E\left(\bar{u}^{(i)}\right) + E\left(\bar{u}^{(j)}\right) = E\left(u^{(i)}\right) + E\left(u^{(j)}\right)$ folgt $\xi = -m\left\langle v^{(i)} - v^{(j)}, e \right\rangle$ mit $\xi \in \mathbb{R}$.*

Beweis. Durch Einsetzen der Definition von E in (22) und unter Verwendung von (21) erhält man:
$\frac{1}{2m}\left\langle u^{(i)} + \xi e, u^{(i)} + \xi e \right\rangle + \frac{1}{2m}\left\langle u^{(j)} - \xi e, u^{(j)} - \xi e \right\rangle =$
$\frac{1}{2m}\left\langle u^{(i)}, u^{(i)} \right\rangle + \frac{1}{2m}\left\langle u^{(j)}, u^{(j)} \right\rangle$
$\Longleftrightarrow$

$\left\langle u^{(i)} + \xi e, u^{(i)} + \xi e \right\rangle - \left\langle u^{(i)}, u^{(i)} \right\rangle = \left\langle u^{(j)}, u^{(j)} \right\rangle - \left\langle u^{(j)} - \xi e, u^{(j)} - \xi e \right\rangle$
$\iff$ (Linearität: $\langle a + b, u \rangle = \langle a, u \rangle + \langle b, u \rangle$)

$\left\langle u^{(i)}, \xi e \right\rangle + \left\langle \xi e, u^{(i)} \right\rangle + \langle \xi e, \xi e \rangle + \left\langle u^{(i)}, u^{(i)} \right\rangle - \left\langle u^{(i)}, u^{(i)} \right\rangle =$
$\left\langle u^{(j)}, u^{(j)} \right\rangle - \left\langle u^{(j)}, u^{(j)} \right\rangle + \left\langle u^{(j)}, \xi e \right\rangle + \left\langle \xi e, u^{(j)} \right\rangle - \langle \xi e, \xi e \rangle$
$\iff$ (Linearität: $\left\langle \xi e, u^{(i)} \right\rangle = \xi \left\langle e, u^{(i)} \right\rangle$ und $\langle \xi e, \xi e \rangle = \xi^2$,
da $\langle \xi e, \xi e \rangle = \xi^2 \langle e, e \rangle$ und $\langle e, e \rangle = 1$)

$\xi \left(\left\langle u^{(i)}, e \right\rangle + \left\langle e, u^{(i)} \right\rangle \right) + \xi^2 =$
$\xi \left(\left\langle u^{(j)}, e \right\rangle + \left\langle e, u^{(j)} \right\rangle \right) - \xi^2$
$\iff$ (Symmetrie)

$\xi^2 + \xi^2 = \xi 2 \left\langle u^{(j)}, e \right\rangle - \xi 2 \left\langle u^{(i)}, e \right\rangle \iff$

$2\xi = 2 \left\langle u^{(j)}, e \right\rangle - 2 \left\langle u^{(i)}, e \right\rangle \iff$ (Definition (3) von u)

$\xi = -m \left(\left\langle v^{(i)}, e \right\rangle \right) - \left(\left\langle v^{(j)}, e \right\rangle \right) = -m \left\langle v^{(i)} - v^{(j)}, e \right\rangle .$ ■

Aus den Impulsvektoren können die Geschwindigkeitsvektoren berechnet werden. Damit ergibt sich die neue Ausgangslage zum Zeitpunkt $\bar{t}_1$.

Der zweite Durchgang erfolgt auf die für den ersten Durchgang beschriebene Weise: Es wird die Zeitspanne $\bar{t}_2$ bis zum Eintreffen des nächsten Ereignisses (Kollision oder Reflexion) im Gesamtsystem berechnet, wobei für die Systemzeit sz_2 gilt $sz_2 = \bar{t}_1 + \bar{t}_2$. Derart erhält man für n Durchgänge eine endliche Folge $(\bar{t}_i)_{i \in \mathbb{N}_n}$ von Zeitspannen zwischen den Ereignissen und eine Folge von Systemzeiten $(sz_i)_{i \in \mathbb{N}_n}$ mit

$$sz_i = \sum_{j=1}^{i} \bar{t}_j. \tag{23}$$

Umgekehrt kann die Folge $(\bar{t}_i)_{i \in \mathbb{N}_n}$ aus $(sz_i)_{i \in \mathbb{N}_n}$ errechnet werden durch

$$\bar{t}_i = sz_i - sz_{i-1} = \sum_{j=1}^{i} \bar{t}_j - \sum_{j=1}^{i-1} \bar{t}_j (i \in \mathbb{N}_n; sz_0 = 0) \tag{24}$$

Die Folge $(\bar{t}_i)_{i \in \mathbb{N}_n}$ kann in zwei Teilfolgen unterteilt werden:

(i) Statt die Zeitspanne von einem Ereignis (Kollision oder Reflexion) zum nächsten Ereignis zu betrachten, kann im System auch die Zeitspanne $\bar{t}_i^r$ von einer Reflexion zur nächsten Reflexion betrachtet werden. Dies ergibt die Folge $(\bar{t}_i^r)_{i \in \mathbb{N}_m}$ und entsprechend mit

$$sz_i^r := \sum_{j=1}^{i} \bar{t}_j^r \tag{25}$$

die Folge $(sz_i^r)_{i\in\mathbb{N}_m}$.

(ii) Es kann die Zeitspannen von einer Kollision zur nächsten Kollision betrachtet werden. Dies ergibt die Folge $\left(\bar{t}_i^k\right)_{i\in\mathbb{N}_{m'}}$ und entsprechend mit

$$sz_i^k := \sum_{j=1}^{i} \bar{t}_j^k \tag{26}$$

die Folge $(sz_i^r)_{i\in\mathbb{N}_{m'}}$. Dabei gilt $m + m' = n$.

Offenbar kann $(\bar{t}_i)_{i\in\mathbb{N}_n}$ aus $(\bar{t}_i^r)_{i\in\mathbb{N}_m}$ und $\left(\bar{t}_i^k\right)_{i\in\mathbb{N}_{m'}}$ berechnet werden. Umgekehrt kann man $(\bar{t}_i^r)_{i\in\mathbb{N}_m}$ und $\left(\bar{t}_i^k\right)_{i\in\mathbb{N}_{m'}}$ aus $(\bar{t}_i)_{i\in\mathbb{N}_n}$ berechnen, wenn man von jeder Systemzeit sz_i weiß, ob sich zu ihr eine Kollision oder eine Reflexion ereignet hat.

3 Die statistischen Konzepte

In diesem Kapitel werden die statistischen Konzepte und Methoden bereitgestellt, die es erlauben werden, im *Kapitel 4* die Poissonität des Zählprozesses von Kollisionsereignissen zu testen. Wie in der Einleitung erwähnt, wird dazu die Verteilung der Zwischenkollisionszeiten überprüft: es wird versucht nachzuweisen, dass die Zufallsvariablen X_i, als deren Werte die Zwischenkollisionszeiten x_i betrachtet werden können, unabhängig und identisch exponentialverteilt sind. Für die Überprüfung der Unabhängigkeit wird in diesem Kapitel der Permutationstest eingeführt. Zwecks Bestimmung der Verteilung werden nicht-parametrische und parametrische Schätzmethoden verwendet.

3.1 Der Permutationstest

Die Überprüfung auf Unabhängigkeit der Zufallsvariablen $(X_i)_{i\in\mathbb{N}_n}, X_i : \mathbb{R}^n \to \mathbb{R}$, $X_i(x) := x_i$, ist aus zwei Gründen für die Untersuchung nötig. Einerseits setzen die unter 3.2. behandelten Schätzverfahren die Unabhängigkeit voraus. Zweitens kann man die Poissonität des Kollisionszählprozesses bei der gewählten Vorgehensweise nur nachweisen, wenn die Stichprobenvariablen $(X_i)_{i\in\mathbb{N}_n}$ unabhängig sind. Um die Unabhängigkeit nachzuweisen, kann man den Permutationstest verwenden. Dieser wird für Zufallsvariablen gebraucht, welche nach einer Wahrscheinlichkeitsverteilung mit stetiger Verteilungsfunktion verteilt sind, was hiermit vorausgesetzt sei.

Der Permutationstest überprüft, ob die Daten im Datenvektor in einer zufälligen Anordnung − gemäß der Relation „<“ − auftauchen. Dazu wird der Datenvektor $(x_1, ..., x_n)$ in k Vektoren identischer Länge t aufgeteilt − wobei im Allgemeinen ein Rest des Datenvektors der Länge kleiner als t übrigbleibt, der im Test nicht berücksichtigt wird. In Bezug auf die Relation „<“ gibt es $t!$ mögliche Anordnungen eines Vektors der Länge t, wenn alle

Komponenten des Vektors paarweise verschieden sind. Es wird die Anzahl der Vektoren pro Anordnung gezählt. Diese Anzahlen sind als Werte eines Vektors von $t!$ Zufallsvariablen zu betrachten, die unter der Hypothese einer Multinomialen Verteilung mit den Parametern $k := \frac{n-n \bmod t}{t}$ und $p_i = \frac{1}{t!}$ für $i \in \mathbb{N}_{t!}$ folgen, denn die Wahrscheinlichkeiten, spezifische Anordnungen anzutreffen, muss für alle Anordnungen identisch sein. Damit sollten keine Anordnungen mehr als zufällig übervertreten sein. Andererseits sollten die Anzahlen von Anordnungen auch zufällig von $\frac{k}{t!}$ abweichen, da sonst eine Überanpassung an die zu erwartenden Werte vorliegen würde, was der These der Unabhängigkeit widersprechen würde.

Die Anpassung der Anzahlen der vorkommenden Anordnungen wird mit Hilfe eines klassischen χ^2- Anpassungstestes geprüft − wobei auf Grund der vorangegangenen inhaltlichen Überlegungen zweiseitig getestet wird. Nach diesen intuitiven Überlegungen wird der Permutationstest strenger eingeführt − gemäß der Darstellung in *Angewandte Statistik* (5.5).

Es wird das folgende Testexperiment betrachtet:

$$(\mathbb{R}^n, \mathcal{B}^n, \mathcal{W}, \mathcal{W}_1, \mathcal{W}_2), \tag{27}$$

wobei

(a) $\mathcal{B}^n$ die Borelsche $\sigma-$Algebra auf $\mathbb{R}^n$ ist,

(b) $\mathcal{W}$ eine Menge von Wahrscheinlichkeitsverteilungen auf $\mathcal{B}^n$ mit stetigen Verteilungsfunktionen ist,

(c) $\mathcal{W}_1 = \{P^n \mid P$ ist ein Wahrscheinlichkeitsmaß auf $\mathcal{B}$ mit stetiger Verteilungsfunktion$\}$. Damit ist $\mathcal{W}_1$ die Menge aller Produktemaße P^n auf $\mathcal{B}^n$, deren „Faktoren“ P eine stetige Verteilungsfunktion aufweisen. Da P^n Produktemaße sind, ist die gemäß P^n verteilte Stichprobe $(X_1, ..., X_n)$ unabhängig. $\mathcal{W}_1$ ist die Hypothese. Es wird aber auch die Aussage „$P^n \in \mathcal{W}_1$“ „Hypothese“ genannt.

(d) $\mathcal{W}_2 = \mathcal{W} - \mathcal{W}_1$ ist die Alternative. Ebenso wird die Aussage „$Q \in \mathcal{W}_2$“ „Alternative“ genannt.

Sei nun ein Datenvektor $(x_1, ..., x_n) \in \mathbb{R}^n$ gegeben, der als Realisation des Zufallsvektors $(X_1, ..., X_n)$ betrachtet wird (X_i i.i.d, $X_i : \mathbb{R}^n \to \mathbb{R}$, $X_i(x) := x_i$). Der Datenvektor kann in Teilvektoren v_l aufgeteilt werden:

$$v_l := (x_{(l-1)t+1}, ..., x_{(l-1)t+t}) \in \mathbb{R}^{t*} \quad \text{für} \quad l \in \mathbb{N}_k \tag{28}$$

mit

$$k =: \frac{n - (n \bmod t)}{t} \tag{29}$$

und

$$\mathbb{R}^{t*} := \{(x_1, ..., x_n) \in \mathbb{R}^t \mid x_i \neq x_j \text{ für } i \neq j \text{ und } i, j \in \mathbb{N}_t\}. \tag{30}$$

Bleibt ein Rest von Daten, deren Anzahl kleiner als t ist, wird dieser fortgelassen und man betrachtet noch den Vektoren

$$x^{n'} := (x_1, ..., x_{n'}) \tag{31}$$

mit

$$n' := n - (n \bmod t). \tag{32}$$

Das obige Testexperiment (27) wird entsprechend angepasst, indem in (27) n durch n' ersetzt wird. Dafür wird in der Folge die folgende Abkürzung verwendet: $(\mathbb{R}^{n'}, \mathcal{B}^{n'}, \mathcal{W}', \mathcal{W}_1', \mathcal{W}_2')$

Es wird

$$V_l(x^{n'}) := (X_{(l-1)t+1}(x^{n'}), ..., X_{(l-1)t+t}(x^{n'}) = (x_{(l-1)t+1}, ..., x_{(l-1)t+t}) = v_l \tag{33}$$

gesetzt.

Um die Ordnungsstruktur eines Vektors $v \in \mathbb{R}^{t*}$ zu definieren, kann man zuerst die folgende *Indexfunktion* festlegen, die jeder Komponente eines Vektors v deren Position im Vektor zuordnet:

$$\begin{aligned} I_v &: \mathbb{R}^{t*} \times \mathbb{R} \to \mathbb{N} \\ I_v(v_i) &:= I(v, v_i) := i \end{aligned} \tag{34}$$

So gilt zum Beispiel mit $v = (3; 8; 5; 0) : I_v(3) = 1; I_v(8) = 2; I_v(5) = 3; I_v(0) = 4$.

Die Ordnungsstruktur eines Vektors kann nun als Wert der folgenden Abbildung definiert werden:

$$\begin{aligned} O &: \mathbb{R}^{t*} \to \mathbb{N}^t \\ O(v) &:= O((v_1, ..., v_t)) := o^v = (o_1^v, ..., o_t^v) \\ \text{mit } o_i^v &:= I_v(v_j) \text{ genau dann, wenn } i = |\{v_k \mid v_k \leq v_j; k \in \mathbb{N}_t\}|. \\ &(|A| \text{ für die Anzahl der Elemente der Menge } A) \end{aligned} \tag{35}$$

So gilt zum Beispiel mit $v = (5.3; 4; 8; 7; 2)$ die Identität $o_3^v = 1$, da $3 = |\{v_k \mid v_k \leq 5.4; k \in \mathbb{N}_5\}|$ und $I_v(5.4) = 1$. Insgesamt gilt $o^v = (5, 2, 1, 4, 3)$.

In der Praxis erhält man die Ordnungsstruktur eines Vektors v, indem dieser als Spalte hingeschrieben wird. In einer zweiten Spalte wird der Vektor durchnummeriert. Dann ordnet man die $t \times 2-$Matrix aufsteigend der Größe der Komponenten x_i nach $-$ unter Mitnahme der Numerierungsspalte. Die umgeordnete Numerierungsspalte, als Vektor betrachtet, ist die Ordnungsstruktur o von v.

Es gibt offenbar $t!$ Ordnungsstrukturen von Vektoren $v \in \mathbb{R}^{t*}$, wobei $t!$ identisch ist mit der Anzahl der möglichen Permutationen

$$\begin{aligned} s \in \mathcal{S}_t &:= \{s \mid s : \mathbb{N}_t \to \mathbb{N}_t \text{ und } s \text{ ist bijektiv}\} \\ &(\mathcal{S}_t \text{ ist die Menge der Permutationen}) \end{aligned} \tag{36}$$

Jeder solchen Permutation s entspricht eine Ordnungsstruktur, wobei man definieren kann:

$$s \text{ entspricht } O(v) \Longleftrightarrow O(v) = (s(1), ..., s(t)). \tag{37}$$

Für jede Permutation s kann deshalb die Menge der Vektoren mit einer der Permutation entsprechenden Ordnungsstruktur betrachtet werden. Dies führt zur Definition:

$$D_s := \{v \in \mathbb{R}^{t*} \mid O(v) = (s(1), ..., s(t))\} \tag{38}$$

Nun kann eine bijektive Abbildung

$$g : \mathcal{S}_t \to \mathbb{N}_{t!} \tag{39}$$

definiert werden, wobei $\mathcal{S}_t$ wiederum die Menge der Permutationen ist. Eine solche Funktion kann z.B. durch ein rekursives Verfahren, das alle Permutationen auflistet und durchnumeriert, bestimmt werden: den Permutationen ist ihre Nummer in der Liste zuzuordnen − im beiliegenden Programm wird das mit einer rekursive Prozedur bewerkstelligt (Quelle für die Pascal-Prozedur: http://www.schoenleber.org/pascal/ pascal2-03.html [konsultiert 11. September 05], s. *Beilage 8.4.* Für eine weitere Methode s. Knuth, D. 1971, S. 59). Entsprechend kann man dann auch von der $i-$ten Ordnungsstruktur eines Vektors aus $\mathbb{R}^{t*}$ sprechen ($i \in \mathbb{N}_{t!}$), da jedem Vektor aus $\mathbb{R}^{t*}$ eine Ordnungsstruktur und damit eine Permutation entspricht, der ihrerseits eine Zahl aus $\mathbb{N}_{t!}$ zugeordnet ist:

$$v \text{ weist die } i-\text{te Ordnungsstruktur auf genau dann,wenn } g(O(v)) = i. \tag{40}$$

Zudem erhält man mit (39) auch eine Durchnummerierung der $D_s, s \in \mathcal{S}_t$, mit

$$D_i := D_{g(s)}. \tag{41}$$

Man kann feststellen, dass die Menge $\{D_1, ..., D_{t!}\}$ eine Zerlegung von $\mathbb{R}^{t*}$ darstellt. Es gilt offensichtlich der Satz:

Satz 3 *Sei* $(\mathbb{R}^n, \mathcal{B}^n, \mathcal{W})$ *ein statistischer Raum mit* $\mathcal{W} := \{P^t \mid P$ *ist eine Wahrscheinlichkeitsverteilung auf* $\mathcal{B}$ *mit stetiger Verteilungsfunktion*$\}$, *dann gilt für die unter (38) und (41) definierten* $D_i, D_j : P^t(D_i) = P^t(D_j)$ *für* $i, j \in \mathbb{N}_{t!}$ *und* $P^t \in \mathcal{W}$.

Ein Beweis des Satzes ergibt sich, indem man zeigt, dass für Wahrscheinlichkeitsmaße P^t, die auf den Halbring S der halboffenen Intervalle $I = [a_1, b_1) \times ... \times [a_t, b_t) \subset \mathbb{R}^{t*}$ definiert sind, $P^t(I) = P^t(I')$, für alle $I \in S$, gilt, wenn I' durch die Permutation der Faktoren des kartesischen Produkts

$I = [a_1, b_1) \times ... \times [a_t, b_t)$ entsteht. Mittels der beiden Fortsetzungssätze für Maße (*WI*, 2.3.2, 2.3.5) kann das Gewünschte dann gezeigt werden.

Da $\{D_1, ..., D_{t!}\}$ eine Zerlegung von $\mathbb{R}^{t*}$ darstellt und $P^t - f.ü.\ \mathbb{R}^{t*} = \mathbb{R}^t$ − denn P weist gemäß Voraussetzung eine stetige Verteilungsfunktion auf −, gilt mit *Satz 3*

$$P^t(\bigcup_{i=1}^{t!} D_i) = t! P^t(D_i) = P^t(\mathbb{R}^{t*}) = P^t\left(\mathbb{R}^t\right) = 1. \tag{42}$$

Damit gilt dann

$$P^t(D_i) = \frac{1}{t!} \text{ für } i \in \mathbb{N}_{t!}. \tag{43}$$

Als nächstes ist eine Teststatistik zu entwickeln. Dazu werden vorgängig für $i \in \mathbb{N}_{t!}$ und $k = \frac{n'}{t}$ die folgenden Zufallsvariablen definiert:

$$Y_i : \mathbb{R}^{n'} \to \mathbb{N} \tag{44}$$
$$Y_i(x^{n'}) := \sum_{l=1}^{k} 1_{D_i} \circ V_l(x^{n'})$$

Damit ordnet die Zufallsvariable Y_i einer Stichprobenrealisation von $(X_1, ..., X_{n'})$ die Anzahl der Teilvektoren $v_l = (x_{(l-1)t+1}, ..., x_{(l-1)t+t})$ zu, welche die Ordnungsstruktur i aufweisen (s. 40). Im beiliegenden Computerprogramm werden die Werte von Y_i berechnet, indem

$$\left|\{v_l \mid g(O(v_l)) = i; v_l = (x_{(l-1)t+1}, ..., x_{(l-1)t+t}) \in \mathbb{R}^{t*} \text{ mit } l \in \mathbb{N}_k\}\right|$$

bestimmt wird.

Es gilt der Satz:

Satz 4 *Sei $X := (X_1, ..., X_{n'})$ eine einfache Stichprobe, V_l ein Teilvektor von X der Länge t wie unter (33) definiert und k die Anzahl der Teilvektoren V_l von X, $P^{n'}$ eine Wahrscheinlichkeitsverteilung, so dass P eine stetige Verteilungsfunktion aufweist, und Y_i definiert wie unter (44). Dann gilt:* $E_{P^{n'}}(Y_i) = \frac{k}{t!}, \quad i \in \mathbb{N}_{t!}$.

Beweis. *Gemäß der Definition von V_l gilt die Gleichung* $\left(P^{n'}\right)_{V_l} = \left(P^{n'}\right)_{(X_{(l-1)t+1}, ..., X_{(l-1)t+t})} = P^t$ $(l \in \mathbb{N}_k)$, *denn die* $X_{(l-1)t+1}, ..., X_{(l-1)t+t}$ *sind Projektionen und i.i.d. Damit gilt dann:* $E_{P^{n'}}(1_{D_i} \circ V_l) \underset{WI,6.1.11(1)}{=} E_{(P^{n'})_{V_l}}(1_{D_i}) = E_{P^t}(1_{D_i}) \underset{WI,6.1.5(1)}{=} P^t(D_i) \underset{(43)}{=} \frac{1}{t!}$ *für* $i \in \mathbb{N}_{t!}$,

was $E_{P^{n'}}(Y_i) = E_{P^{n'}}\left(\sum_{l=1}^{k} 1_{D_i} \circ V_l\right) \underset{WI,6.1.5(3)}{=} \sum_{l=1}^{k} E_{P^{n'}}(1_{D_i} \circ V_l) = \sum_{l=1}^{k} \frac{1}{t!} = \frac{k}{t!}$, *für* $i \in \mathbb{N}_{t!}$, *ergibt.* ■

Unter der Voraussetzung der Hypothese ist der Zufallsvektor $(Y_i, ..., Y_{t!})$ multinomial verteilt − mit den Parametern k und $p_i = \frac{1}{t!}$ für $i \in \mathbb{N}_{t!}$.

Die Abweichung der Werte des Zufallsvektors $Y = (Y_1, ..., Y_{t!})$ vom Vektor der Erwartungswerte $(E(Y_1), ..., E(Y_{t!}))$ kann man messen durch:

$$S : \mathbb{R}^{n'} \to \mathbb{R}_+ \tag{45}$$

$$S(x^{n'}) = \sum_{i=1}^{t!} \frac{\left(Y_i(x^{n'}) - \frac{k}{t!}\right)^2}{\frac{k}{t!}}$$

Die Zufallsvariable S ist näherungsweise χ^2-verteilt mit $t! - 1$ Freiheitsgraden (Cramér, 1946, S. 416 ff., Agresti, 1990, S. 44 f.).

Um den Test durchzuführen, ist ein Signifikanzniveau α zu bestimmen und bei zweiseitigem Testen sind die $\frac{\alpha}{2}$- und die $1 - \frac{\alpha}{2}$-Fraktile $t_1^{\alpha/2}$ und $t_2^{\alpha/2}$ der $\chi^2_{t!-1}$ Verteilung zu berechnen. Es gilt dann unter der Hypothese

$$P^{n'}\left(\left\{x^{n'} \in \mathbb{R}^{n'} \mid t_1^{\alpha/2} \leq S(x^{n'}) \leq t_2^{\alpha/2}\right\}\right) = 1 - \alpha, \tag{46}$$

wobei $P^{n'} \in \mathcal{W}_1'$. Die These, dass die Stichprobenvariablen X_i stochastisch unabhängig gemäß P verteilt sind, ist zu verwerfen, wenn $S(x^{n'}) \notin [t_1^\alpha, t_2^\alpha]$. Gilt demgegenüber $S(x^{n'}) \in [t_1^\alpha, t_2^\alpha]$, so ist die Hypothese nicht zu verwerfen und die These, dass $P^{n'}$ die gemeinsame Verteilung von $(X_1, ..., X_{n'})$ ist und damit die X_i unabhängig gemäß P verteilt sind, muss nicht abgelehnt werden.

Bei χ^2-Tests wird oft die Faustregel gesetzt, dass die Erwartungswerte größer oder gleich 5 zu sein haben. In manchen Schriften werden allerdings auf Grund von Monte-Carlo-Studien weniger strenge Regeln gesetzt (z.B. Erwartungswert darf nicht in mehr als in 20% der Klassen kleiner als 5 sein; oder bei großer Klassenanzahl muss die Regel überhaupt nicht mehr beachtet werden, siehe Stahel (1995, S. 222) sowie Agresti (1990, S. 49, S. 246 ff); Cramér (1946, S. 420) hingegen verlangt noch einen Erwartungswert von mindestens 10 für alle Klassen). Hier wird die Fassung „alle Erwartungswerte müssen größer oder gleich 5 sein“ der Regel eingehalten. Im vorliegenden Fall ergibt dies $k \geq 5t!$, da $E_{P^{n'}}(Y_i) = \frac{k}{t!}$.

3.2 Die Schätzmethoden

Es wird davon ausgegangen, dass kein Vorwissen über die zu schätzende Verteilung zur Verfügung steht. Für diesen Fall bieten sich die sogenannten nicht parametrischen Verfahren an. Es gibt verschiedene Varianten solcher Verfahren, wie die Schätzung mit Hilfe der empirischen Verteilungsfunktion (*Angewandte Statistik*, 2.2), mit Hilfe eines Histogrammschätzers (*Angewandte Statistik*, 2.3) oder mit Hilfe eines Kerndichteschätzers (*Angewandte Statistik,* 2.4). Diese Verfahren setzen voraus, dass die Zufallsvariablen

X_i unabhängig und identisch verteilt sind. Kriterium für die Güte nicht-parametrischer Schätzungen ist die Konsistenz der Schätzer.

Während die empirische Verteilungsfunktion und der Histogrammschätzer nicht-stetige Funktionen darstellen, sind Kerndichteschätzer stetig. Da Zwischenkollisionszeiten wenigstens theoretisch beliebige positive reelle Werte annehmen können, sind sie als Werte von gemäß einer stetigen Verteilung verteilten Zufallsvariablen zu betrachten. Faktisch sind Werte von empirischen Messprozessen immer rational. Zudem sind diese immer auf eine spezifische Anzahl von Stellen beschränkt. Dies gilt auch bei vom Computer produzierten Zahlen − wenn auch aus anderen Gründen. Bei der Voraussetzung, dass Zwischenkollisionszeiten beliebige positive reelle Werte annehmen können, wird also eine Idealisierung zu Grunde gelegt. Dieser entsprechend ist eine Kerndichteschätzung geeigneter als die anderen Verfahren, um die gesuchte Verteilung zu schätzen. Der Kerndichteschätzer wird nun eingeführt.

3.2.1 Das nicht-parametrische Schätzen von Verteilungen durch Kerndichteschätzer

Die folgenden Ausführungen folgen *Angewandte Statistik* (5.5). Man setzt voraus, das die zu schätzende Dichte f *lebesgue*-stetig ist. Es wird von einem Wahrscheinlichkeitsmaß K über $(\mathbb{R}, \mathcal{B})$ mit einer stetigen Lebesgue-Dichte $k : \mathbb{R} \to \mathbb{R}_+$ ausgegangen. k wird „Kern“ genannt. Das Wahrscheinlichkeitsmaß wird zweifach transformiert, was einerseits einer Verschiebung der entsprechenden Dichte k auf der x-Achse und andererseits einer Streckung von k entspricht. Die Streckung erfolgt durch die $(\mathcal{B}, \mathcal{B})$-messbaren Transformation (Homotetie)

$$\begin{aligned} T_h &: \mathbb{R} \to \mathbb{R} \\ T_h(t) &:= ht, \quad h > 0 \end{aligned} \tag{47}$$

Unter dieser Transformation erhält man auf Grund des Transformationssatzes für Dichten (*WI*, Satz 5.3.2) die Dichte

$$k_h(t) := \frac{1}{h} k\left(\frac{t}{h}\right), t \in \mathbb{R}, \tag{48}$$

des Bildmaßes

$$K_h := K_{T_h} \tag{49}$$

auf $(\mathbb{R}, \mathcal{B})$ unter T_h − denn $T_h^{-1}(t) = \frac{t}{h}$, T_h ist stetig differenzierbar und $\left|\det\left(\frac{d}{dt} T_h(t)\right)\right| = \left|\frac{d}{dt} T_h(t)\right| = h \neq 0$. Damit folgt die Formel für k_h unmittelbar mit Hilfe des Transformationssatzes für Dichten.

Durch $T_h, h > 1$, wird die Dichtefunktion gestreckt, bei $0 < h < 1$ gestaucht, wobei in der Praxis die Stauchung interessiert. Am Beispiel $K =$

$N(0,1)$ (Standardnormalverteilung) und damit $k(t) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}t^2\right)$, ergibt sich mit $h = 0.5$

$$k_{0.5}(t) = \frac{1}{0.5\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{t}{0.5}\right)^2\right) = \frac{1}{0.5\sqrt{2\pi}} \exp\left(-\frac{1}{0.5^2 2}(t)^2\right).$$

k wird in diesem Fall „Gausskern“ genannt. h ist im Falle des Gausskernes mit der Standardabweichung der Normalverteilung identisch.

Als nächstes ist die Transformation zu betrachten, die einer Verschiebung des Kernes auf der x-Achse entspricht. Es wird von der Faltung einer Wahrscheinlichkeitsverteilung P mit dem Einpunktwahrscheinlichkeitsmaß $\delta_{\bar{x}}$ ausgegangen. Die Faltung ist definiert als die Verteilung der $(\mathcal{B}^2, \mathcal{B})-$messbaren Zufallsvariable „Summe“

$$\begin{aligned} T &: \mathbb{R}^2 \to \mathbb{R} \\ T(x,y) &:= x + y \end{aligned} \tag{50}$$

unter dem Produktemaß $P \otimes Q$ auf $(\mathbb{R}^2, \mathcal{B}^2)$. Für die Faltung wird die Abkürzung

$$P * Q := (P \otimes Q)_T \tag{51}$$

verwendet, wobei $(P \otimes Q)_T$ das Bildmaß von $P \otimes Q$ unter der Abbildung T ist. Konkret wird bei der Entwicklung des Kerndichteschätzers die Faltung von P mit dem Einpunktmaß $\delta_{\bar{x}}$ verwendet, wobei $(P \otimes \delta_{\bar{x}})_T = P_S$ mit $S : \mathbb{R} \to \mathbb{R}; S(x) := x + \bar{x}$, wie der folgende Satz zeigt:

Satz 5 *Sei P ein beliebiges Wahrscheinlichkeitsmaß auf $(\mathbb{R}, \mathcal{B})$ und $\delta_{\bar{x}}$ das Einpunktmaß auf $(\mathbb{R}, \mathcal{B})$ für ein $\bar{x} \in \mathbb{R}$. Sei S die $(\mathbb{R}, \mathcal{B})-$messbare Abbildung $S : \mathbb{R} \to \mathbb{R}$; $S(x) := x + \bar{x}$ $(x \in \mathbb{R})$. Dann gilt $(P \otimes \delta_{\bar{x}})_T = P_S$.*

Beweis. Es gilt:
(i) $P_S(-\infty, t) \underset{WI,3.1.14}{=} P(S^{-1}(-\infty, t)) \underset{WI,3.1.6}{=} P(\{x \mid S(x) = x + \bar{x} < t\}) =$
$P(\{x \mid x < t - \bar{x}\}) = P(-\infty, t - \bar{x})$
(ii) $(P * \delta_{\bar{x}})(-\infty, t) \underset{WI,7.5.2}{=} \int_{\mathbb{R}} \delta_{\bar{x}}((-\infty, t - x))dP(x) \underset{WI,2.2.7(1);\ WI,1.2.15}{=}$
$\int_{\mathbb{R}} 1_{(-\infty, t-\bar{x})} dP \underset{WI,4.2.6(1)}{=} P(-\infty, t - \bar{x})$. ■

Die Dichte von $P * \delta_{\bar{x}}$ ist damit $Lebesgue - f.ü.$ identisch mit der Dichte des Bildes von P_S, wobei die Dichte von P_S

$$k_{\bar{x}}(t) := k(t - \bar{x}) \tag{52}$$

ist, da $S^{-1}(t) = t - \bar{x}$, S stetig differenzierbar ist und $\left|\det\left(\frac{d}{dt}(t + \bar{x})\right)\right| = 1 \neq 0$. Damit gilt mit dem Transformationssatz für Dichten unmittelbar, dass die Dichte des Bildes von P unter S durch $k_{\bar{x}}(t) = k(t - \bar{x})$ gegeben ist.

Am Beispiel des Gausskerns ist $N(0,1) * \delta_{\bar{x}} = N(\bar{x}, 1)$, denn mit $k(t) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(t)^2\right)$ und (52) ergibt sich $k_{\bar{x}}(t) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(t-\bar{x})^2\right)$. Dies ist die Dichtefunktion von $N(\bar{x}, 1)$.

Damit sind die Vorarbeiten für die Entwicklung des Kerndichteschätzers geleistet und dieser kann nun eingeführt werden. Sei $(\mathbb{H}_0^\infty, \mathcal{H}_0^\infty, \mathcal{W})$ ein statistischer Raum mit dem Wahrscheinlichkeitsmaß $P_0^\infty \in \mathcal{W}$. Sei $x^n = (x_1, ..., x_n)$ eine Realisation der einfachen nach P_0^n verteilten Stichprobe $X^n : \mathbb{H}_0^\infty \to \mathbb{H}_0^n$; $X^n(x) = x^n$.Es handelt sich um die ersten n Stichprobenvariablen der Stichprobe X^∞, die nach P_0^∞ verteilt ist. Ferner sei

$$Q_n^{x^n} := \mathbb{H}_0 \to [0,1] \tag{53}$$
$$Q_n^{x^n}(B) := \frac{1}{n} \sum_{i=1}^{n} 1_B(x_i), B \in \mathcal{H}_0$$

die *empirische Verteilung* der Stichprobe X^n unter P_0^n.
Man betrachtet nun

$$K_h * Q_n^{x^n} = K_h * \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i} = \frac{1}{n} \sum_{i=1}^{n} (K_h * \delta_{x_i}). \tag{54}$$

Es gilt, dass K_h für $h \to 0$ in Verteilung gegen δ_0 und $Q_n^{x^n}$ für $n \to \infty$ in Verteilung gegen P_0 konvergieren (s. *Angewandte Statistik*, 2.4.5). Damit folgt dann wegen der Stetigkeit von T die Konvergenz in Verteilung von $\left(K_h * Q_n^{x^n}\right)$ gegen $\delta_0 * P_0 = P_0$ für $h \to 0$ und $n \to \infty$. Deshalb eignet sich die Dichtefunktion von $\frac{1}{n} \sum_{i=1}^{n} (K_h * \delta_{x_i})$ als Schätzer für die Dichtefunktion f von P_0.

Die Dichtefunktion von $\frac{1}{n} \sum_{i=1}^{n} (K_h * \delta_{x_i})$ lässt sich mit den obigen Resultaten leicht herleiten: die stetige Lebesgue-Dichte von $K_h * \delta_{x_i}$ ist $k_h(t - x_i)$, für $t \in \mathbb{R}$, (s. *Satz* 5 und (52)) und damit ist die Dichte von $\frac{1}{n} \sum_{i=1}^{n} (K_h * \delta_{x_i})$

$$\frac{1}{n} \sum_{i=1}^{n} k_h(t - x_i) \underset{(48)}{=} \frac{1}{hn} \sum_{i=1}^{n} k(\frac{t - x_i}{h}). \tag{55}$$

$\frac{1}{hn} \sum_{i=1}^{n} k(\frac{t-x_i}{h})$ wird Kerndichteschätzer der Bandbreite h und der Stichprobengröße n von f genannt − wobei x_i, $i \in \mathbb{N}_n$, die Werte von *i.i.d.* Zufallsvariablen X_i sind, die nach P_0 mit der stetigen Dichtefunktion f verteilt sind.

Am Beispiel des Gausskerns wird $\frac{1}{hn} \sum_{i=1}^{n} k(\frac{t-x_i}{h})$ zu

$$\frac{1}{hn\sqrt{2\pi}} \sum_{i=1}^{n} \exp\left(-\frac{1}{2h^2}(t - x_i)^2\right).$$

Es wird also das punktweise arithmetische Mittel der Dichtefunktionen, die an die Stellen $x_i, i \in \mathbb{N}_n$, verschoben werden und die mit h (= Standardabweichung der Normalverteilung) gestaucht werden, gebildet.

Ein Satz von Nadaraja (1965) besagt, dass unter spezifischen Bedingungen Kerndichteschätzer stark konsistente Schätzer der zu schätzenden Dichte f sind. Eine der Bedingungen ist, dass f gleichmäßig stetig ist. In der Folge werden Kerndichteschätzer verwendet, um Dichten von Exponentialverteilungen zu schätzen. Diese sind allerdings nicht stetig. Entsprechend wird hier eine schwächere Konsistenzaussage für Kerndichteschätzer festgehalten, die für lebesgue-integrierbare Dichten gilt:

Sei $(h(n))_{n=1}^{\infty}$ eine Folge positiver Zahlen mit $\lim\limits_{n\to\infty} h(n) = 0$ und $\lim\limits_{n\to\infty} nh(n) = \infty$. Dann folgt mit Devroye, Györfi (1985, chap. 3. sec. 1, Theorem 1, zitiert nach Moeschlin et al., 1998, S. 113)):

$$\lim_{n\to\infty} \int \left| \frac{1}{h(n)n\sqrt{2\pi}} \sum_{i=1}^{n} \exp\left(-\frac{1}{2h(n)^2}(t-x_i)^2 \right) - f(t) \right| d(t) = 0 \; P_0 - f.\ddot{u}., \tag{56}$$

wobei bezüglich des Lebesgue-Maßes integriert wird und f die lebesgue-integrierbare Dichtefunktion der zu schätzenden Verteilung P_0 ist. Der Variationsabstand von $\frac{1}{h(n)n\sqrt{2\pi}} \sum\limits_{i=1}^{n} \exp\left(-\frac{1}{2h(n)^2}(t-x_i)^2\right)$ und f konvergiert also für $n \to \infty$ gegen 0, P_0−f.ü.

Mit Hilfe des Kerndichteschätzers kann eine Dichtefunktion geschätzt werden, auf Grund deren graphischen Form man oft eine Vermutung darüber anstellen kann, zur welcher Verteilungsfamilie diese gehören könnte. Unter einer *Verteilungsfamilie* wird eine Klasse von Verteilungen verstanden, welche dieselbe Funktionalform der Dichtefunktion aufweisen, d.h. deren Dichtefunktionsformeln bis auf die Parameter der Verteilungen identisch sind. So ist die Familie der Normalverteilungen z.B. identisch mit der Menge $\{P_{\mu,\sigma^2} \mid$ die Dichtefunktion von P_{μ,σ^2} ist $f_{\mu,\sigma^2} : \mathbb{R} \to \mathbb{R}_+; f_{\mu,\sigma^2}(t) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(t-\mu)^2\right)$ mit $\mu \in \mathbb{R}, \sigma^2 \in \mathbb{R} - \{0\}\}$. Kann mittels des Kerndichteschätzers eine Verteilungsfamilie vermutet werden, kann man aus dieser in der Folge mit Hilfe der parametrischen Schätzung eine spezifische Verteilung auswählen. Entsprechend wird diese Schätzmethode nun eingeführt.

3.2.2 Parametrische Schätzung der Wahrscheinlichkeitsverteilung

Es wird von einem Schätzexperiment $(\mathbb{R}^{\infty}, \mathcal{B}^{\infty}, \mathcal{W}, Id_{\mathbb{R}_+ - \{0\}})$ ausgegangen (s. *Angewandte Statistik*, 2.1), wobei $\mathcal{W} = \{ (Exp(\lambda))^{\infty} \mid Exp(\lambda)$ ist eine Exponentialverteilung mit dem Parameter $\lambda \in \mathbb{R}_+ - \{0\}\}$. Aus dieser Menge schätzt man die Verteilung, gemäß der die $i.i.d$ Zufallsvariablen $X_i, i \in \mathbb{N}$, verteilt sind, indem der Parameter der Verteilung aus den ersten n Zufallsvariablen der einfachen Stichprobe $(X_1, X_2, ...)$ geschätzt wird. Die Zufallsvariablen X_i sind Projektionen von $\mathbb{R}^{\infty}$ nach $\mathbb{R}$ mit $X_i(x) = x_i$.

Als Schätzstatistik wird

$$S_n : \mathbb{R}^\infty \to \mathbb{R} \tag{57}$$
$$S_n(x) := \frac{1}{\bar{X}_n(x)} := \frac{1}{\frac{1}{n}\sum_{i=1}^{n} X_i(x)}$$

verwendet. $(S_n)_{n\in\mathbb{N}}$ ist eine stark konsistente Folge von Schätzern des Parameters λ einer Exponentialverteilung. Bevor die Konsistenz bewiesen wird, ist diese für Folgen von Punktschätzern zu definieren:

Definition 6 *Sei $(\mathbb{H}, \mathcal{H}, \mathcal{W}, h)$ ein Schätzexperiment, wobei $\mathcal{W} = \{P_y \mid \gamma \in \Gamma\}$ eine parametrisierte Familie von Wahrscheinlichkeitsmaßen auf $(\mathbb{H}, \mathcal{H})$ ist und $h : \Gamma \to \mathbb{R}^m$ eine Abbildung. Für $n \in \mathbb{N}$ sei $T_n : \mathbb{H} \to \mathbb{R}^m$ ein Schätzer für h. Die Schätzerfolge $(T_n)_{n\in\mathbb{N}}$ heißt stark konsistent genau dann, wenn $P_\gamma(\{\omega \in \mathbb{H} \mid \lim_{n\to\infty} T_n(\omega) = h(\gamma)\}) = 1 \qquad (\gamma \in \Gamma)$.*

Es wird nun bewiesen, dass $(S_n)_{n\in\mathbb{N}}$ eine stark konsistente Folge von Schätzern des Parameters λ der Exponentialverteilung ist:

Satz 7 *Sei das Schätzexperiment $(\mathbb{R}^\infty, \mathcal{B}^\infty, \mathcal{W} = \{P_\lambda^\infty \mid \lambda \in \mathbb{R}_+ - \{0\}\}, Id_{\mathbb{R}_+-\{0\}})$ mit den Exponentialverteilungen P_λ sowie die einfache Stichprobe $X = (X_1, X_2, ...)$ gegeben. Dann ist $(S_n)_{n\in\mathbb{N}}$ mit $S_n : \mathbb{R}^\infty \to \mathbb{R}; S_n(x) := \frac{1}{X_n(x)}$ eine stark konsistente Folge von Schätzern von λ $(\lambda \in \mathbb{R}_+ - \{0\})$.*

Beweis. Da gemäß Voraussetzung abzählbar unendlich viele identisch exponential verteilte, unabhängige Zufallsvariablen X_i mit $V_{P_\lambda}(X_i) = \frac{1}{\lambda^2} < \infty$ vorliegen (einfache Stichprobe), besagt ein bekannter Satz von Kolmogorov (*WI*, 8.3.8 und Zusatz 8.3.9), dass für die Folge $(X_i)_{i\in\mathbb{N}}$ das starke Gesetz der großen Zahlen gilt. Damit gilt gemäß Definition dieses Gesetzes
(i) $\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} (X_i - E(X_i)) = 0 \;\; P_\lambda - f.ü.$.
(ii) Für exponentialverteilte Zufallsvariablen X_i gilt $E(X_i) = \frac{1}{\lambda}$.
(iii) Aus (i) und (ii) folgt:
$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} (X_i - E(X_i)) = \lim_{n\to\infty} \left(\bar{X}_n(x) - \frac{1}{n} n E(X_i)\right) =$
$\lim_{n\to\infty} \bar{X}_n(x) - E(X_i) = \lim_{n\to\infty} \bar{X}_n(x) - \frac{1}{\lambda}$, $P_\lambda - f.ü.$ und damit auch
(iv) $\lim_{n\to\infty} \frac{1}{\bar{X}_n(x)} = \frac{1}{\frac{1}{\lambda}} = \lambda \; P_\lambda - f.ü.$. Daraus folgt dann
(v) $P_\lambda(\{x \in \mathbb{H} \mid \lim_{n\to\infty} \frac{1}{\bar{X}_n(x)} = \lambda\}) = 1$ $(\lambda \in \mathbb{R}^+ - \{0\})$. Damit ist
(vi) $\left(\frac{1}{\bar{X}_n(x)}\right)_{n\in\mathbb{N}}$ eine stark konsistente Folge von Schätzern von λ. ■

Da die stetige Verteilungsfunktion der Exponentialverteilung $Exp\,(\lambda) := P_\lambda \in \mathcal{W}$ gegeben ist durch $F : \mathbb{R} \to [0, 1]; \;\; F(t) = (1 - \exp(-\lambda t)) \cdot 1_{[0,\infty)}$, ist die Dichtefunktion von $Exp\,(\lambda)$ gegeben durch $f : \mathbb{R} \to \mathbb{R}_+; \;\; f(t) =$

$\frac{d}{dt}\left(1-\exp(-\lambda t)\cdot 1_{[0,\infty)}\right) = \lambda\exp(-\lambda t)\cdot 1_{[0,\infty)}$. Die geschätzte Dichtefunktion

$$\begin{aligned} \hat{f} &: \mathbb{R} \to \mathbb{R}_+ \\ \hat{f}(t) &= 1_{[0,\infty)}(t)\cdot\hat{\lambda}\exp(-\hat{\lambda}t) \text{ mit } \hat{\lambda} := \frac{1}{\bar{X}_n(x^n)}. \\ &(x^n = (x_1,...,x_n), x_i \text{ sind die Werte der} \\ &\quad \text{ersten } n \text{ Stichprobenvariablen } X_i) \end{aligned} \tag{58}$$

kann in ein kartesisches Koordinatensystem gezeichnet werden und man kann den Kerndichteschätzer darüber legen. Fallen beide Schätzer genügend genau zusammen, kann als bestätigt erachtet werden, dass die Wahl der Verteilungsfamilie, die mittels der Kerndichteschätzung bestimmt wurde, angemessen war. Entsprechend wurde für das parametrische Schätzen ein günstiges Schätzexperiment vorausgesetzt. Zudem ist bestätigt, dass die Stichprobenvariablen exponentialverteilt sind. Zuletzt kann durch die parametrische Schätzung eine konkrete Verteilung aus der Familie der Exponentialverteilungen ausgewählt werden, die bei vielen Daten auf Grund der Konsistenz der Schätzer nahe beim Parameter der „wirklichen“ Verteilung liegt.

3.3 Die Exponentialverteilung und die Poissonität

Zentraler Untersuchungsgegenstand dieser Arbeit ist − unter Auswertung von Daten, die vom im *Kapitel 2* entwickelten virtuellen Experiment produziert werden − die Frage, ob ein Poissonprozess bezüglich der Kollisionsereignisse vorliegt. Es wurde darauf hingewiesen, dass das Vorliegen des Poissonprozesses mittels des Nachweises des Vorliegens unabhängiger und identisch exponentialverteilter Zufallsvariablen, die Zwischenkollisionszeiten als Werte annehmen, erfolgt. In diesem Teilkapitel geht es darum, die entsprechenden Begriffe einzuführen und deren Zusammenhang zu skizzieren. Zuerst wird das Konzept des Poissonprozesses eingeführt, um nachher den erwähnten Zusammenhang aufzuzeigen.

Zu betrachten ist eine Folge $(T_i)_{i\in\mathbb{N}}$ von Zufallsvariablen mit

$$\begin{aligned} T_i &: (\Omega, \mathcal{A}, P) \to (\mathbb{R}_+, \mathcal{B}\cap\mathbb{R}_+) \\ T_i(t) &= t_i. \end{aligned} \tag{59}$$

Die Zeitabstände zwischen Kollisionen sind dabei Werte t_i der T_i. Berücksichtigt werden im konkreten Experiment die ersten M Zufallsvariablen, d.h. es wird die Teilfolge $(T_i)_{i\in\mathbb{N}_M}$ betrachtet, deren Zufallsvariablen die ersten M Zwischenkollisionszeiten t_i als Werte annehmen.

Seien weiter gegeben die Zufallsvariablen

$$S_m := \sum_{i=1}^{m} T_i \text{ für } m \in \mathbb{N} \tag{60}$$

und die Zufallsvariablen

$$N_t := |\{i \in \mathbb{N} \mid S_i \leq t\}|. \tag{61}$$

N_t nimmt damit die Anzahl der Summen $S_i \leq t$ als Wert an. Die Anzahl der Summen ist identisch mit der Anzahl der Kollisionen während der Zeit $[0, t]$. Als nächstes werden für $I = (t_1, t_2]$ die Zufallsvariablen

$$X_I := X_{t_1,t_2} := N_{t_2} - N_{t_1} \text{ mit } 0 \leq t_1 < t_2 \leq \infty \tag{62}$$

definiert. Es handelt sich um die Anzahl der Kollisionen im Zeitintervall $(t_1, t_2]$. Damit wurden die Abkürzungen und Begriffe eingeführt, um den Poissonprozess definieren zu können:

Definition 8 *Die Familie $(N_t \mid t \in \mathbb{R}_+)$ ist ein Poissonprozess der Intensität $\lambda \in (0, \infty)$ genau dann, wenn*
(i) X_{t_1,t_2} poissonverteilt ist mit dem Parameter $\beta := (t_2 - t_1)\lambda$, d.h.
$P(X_{t_1,t_2} = k) = e^{-\beta}\frac{\beta^k}{k!}$) (für alle t_1, t_2 mit $0 \leq t_1 < t_2 \leq \infty$).
(ii) Sind $I_1, ..., I_r$ disjunkte Intervalle, so sind die Zufallsvariablen X_{I_i} unabhängig.
(iii) Die Zufallsvariablen $X_{0,t}$ haben Werte in $\mathbb{N}^0 \cup \{\infty\}$ und $X_{0,t}$ ist als Funktion von t monoton wachsend und rechtsstetig.
(iv) $X_0 = 0$. (siehe Krengel, 1998, S. 244 ff).

Der Erwartungswert der Poissonverteilung mit der Wahrscheinlichkeitsfunktion $\varpi : \mathbb{N}^0 \to [0, 1]; \varpi(x) := e^{-\beta}\frac{\beta^x}{x!}$ ist β (s. *WI.* 6.1.8; $\mathbb{N}^0 := \mathbb{N} \cup \{0\}$). Damit würde beim Vorliegen einer Poissonverteilung gelten:

$$E(N_{t_2} - N_{t_1}) = \beta = (t_2 - t_1)\lambda \tag{63}$$

Deshalb hängt die erwartete Anzahl von Kollisionen nur von der Breite des Intervalls $(t_1, t_2]$ ab. Wo sich dieses Intervall auf der Zeitachse befindet, spielt keine Rolle.

Um das Vorliegen eines Poissonprozesses auszumachen, kann man einige wahrscheinlichkeitstheoretische Theoreme nutzen. Vom Poissonprozess her lässt sich etwas über die Verteilung der Zwischenkollisionszeiten aussagen (Satz und Beweis siehe Krengel, 1998, S. 227):

Satz 9 *Seien $T_i : \Omega \to \mathbb{R}_+$, $i \in \mathbb{N}$, Zufallsvariablen, die Zwischenkollisionszeiten annehmen. Sei $S_m := \sum_{i=1}^{m} T_i$ für $m \in \mathbb{N}$ und die Zufallsvariable $N_t := |\{i \in \mathbb{N} \mid S_i \leq t\}|$. Ist $(N_t \mid t \in \mathbb{R}_+)$ ein Poissonprozess mit Parameter $\lambda > 0$, so sind die Zufallsvariablen T_i unabhängig und identisch exponentialverteilt mit dem Parameter λ.*

Es kann auch eine Art Umkehrung gezeigt werden (Satz und Beweis siehe Krengel, 1998, S. 229):

Satz 10 *Sei $(T_i)_{i\in\mathbb{N}}$ eine Folge von unabhängigen, identisch exponentialverteilten Zufallsvariablen mit dem Parameter $\lambda > 0$, so definiert $N_t := \sup\{k \mid T_1 + ... + T_k \leq t\}$ einen Poissonprozess mit Parameter λ.*

Diese Sätze verdeutlichen einerseits, wie die Parameter des Poissonprozesses, der Poissonverteilung und der Exonentialverteilung zusammenhängen (s. auch Moeschlin, Grycko, 2004, S. 23 f.). Andererseits können die Zufallsvariablen, die Zwischenkollisionszeiten als Werte annehmen, auf Exponentialverteilung und Unabhängigkeit geprüft werden, um das Vorliegen eines Poissonprozesses zu testen.

Damit sind der theoretische Hintergrund und die statistischen Hilfsmittel erarbeitet, so dass nun zu einer Analyse der durch das Computerexperiment erzeugten Daten übergangen werden kann.

4 Die Anwendung der statistischen Konzepte auf Zwischenkollisionszeiten

Die oben entwickelten Konzepte und Verfahren werden nun auf Zwischenkollisionszeiten angewendet, welche durch Durchläufe des im *Kapitel 2* beschriebenen Computerexperimentes erzeugt werden.

Bei den unter *Kapitel 3* eingeführten Verfahren wird die identische Verteiltheit der Stichprobenzufallsvariablen vorausgesetzt. Die Haltbarkeit dieser Voraussetzung fürs beschriebene Experiment soll kurz begründet werden. Alle Mikrobestandteile weisen bis auf die Position und die Geschwindigkeit − und was dadurch impliziert wird (Verschiedenheit von Impulsvektor und Energie) − dieselben Eigenschaften auf (gleicher Radius, gleiche Masse, durch gleiche Gesetze bestimmtes Verhalten bei Reflexionen und Kollisionen). Werden systematisch Molekel vertauscht, indem Molekel i die Position und die Geschwindigkeit der Molekel j übernimmt und umgekehrt, werden deshalb die Zwischenkollisionszeiten nicht verändert. Diese sind nur durch die Position und die Geschwindigkeit der Molekel und nicht durch die Molekel als Individuen bestimmt. Es gibt deshalb keinen Grund, die identische Verteiltheit der betrachteten Zufallsvariablen inhaltlich in Frage zu stellen.

Als nächstes wird die Unabhängigkeitsvoraussetzung geprüft. Es wird dabei auf zwei Arten vorgegangen: Einerseits wird der Permutationstests für $t \in \{2, 3, ..\}$ durchgeführt − solange als $\frac{k}{t!} \geq 5$. Das Signifikanzniveau wurde dabei auf 0.05 belassen, obwohl sich durch das Mehrfachtesten eine Erhöhung der Fehlerwahrscheinlichkeit erster Art ergibt und sich damit unter Umständen tiefere Signifikanzgrenzen rechtfertigen ließen (Bonferroni-Korrektur).

Ein Durchlauf des Computerexperiments mit $50'000$ Ereignissen, davon $36'420$ Zwischenkollisionszeiten, ergibt ein maximales $t = 6$. Entsprechend werden fürs Beispiel alle Permutationstests für $t \in \{2, 3, 4, 5, 6\}$ berechnet.

Es werden die Testwerte (= Werte der Teststatistik) als kleine Ellipsen graphisch in einem Annahmeband ausgegeben. Zudem wird der entsprechende Testwert auch numerisch angegeben. Für die verschiedenen Tests werden zuletzt die 0.025- und 0.975-Quantile angezeigt, die mit Excel Microsoft 2002 berechnet wurden. Für $t \in \{2,3,4,5,6\}$ wurde mit „CHIINV$(0.025, t!-1)$“ und „CHIINV$(0.975, t!-1)$“ gerechnet. Gemäß Excel-Hilfe führt der Befehl CHIINV so lange Iterationsschritte aus, „bis das Ergebnis mit einer Abweichung von höchstens $\pm\, 3E10^{-7}$ feststeht“. Das Ergebnis wird auf drei Stellen gerundet. Für $t \in \{7,8,9\}$ wurde mit NORMINV$(0.025; t!-1;$WURZEL$(2*(t!-1)))$ und NORMINV$(0.975; t!-1;$WURZEL$(2*(t!-1)))$ gerechnet, da die Funktion CHIINV für diese Zahlen eine Fehlermeldung liefert und da bei einer großen Zahl von Freiheitsgraden d die Normalverteilung $N(d, 2d)$ eine Näherung der χ^2_d-Verteilung darstellt (siehe Hartung, 1999, S. 153). Zur Funktion NORMINV werden von Excel Microsoft folgende Angaben gemacht: „NORMINV verwendet zur Berechnung der Funktion ein Iterationsverfahren. Ausgehend von der angegebenen Wahrscheinlichkeit, iteriert NORMINV so lange, bis das Ergebnis mit einer Abweichung von höchstens $\pm 3E10^{-7}$ vorliegt.“

Gemäß Theorie müssen die Teilvektoren v_l Elemente von $\mathbb{R}^{t*} = \{(x_1, ..., x_n) \in \mathbb{R}^t \mid x_i \neq x_j \text{ für } i \neq j \text{ und } i, j \in \mathbb{N}_t\}$ sein. Im Computerexperiment wird mit rationalen Zahlen mit endlich vielen Stellen gerechnet. Entsprechend ist ausnahmsweise ein Zusammenfallen von zwei Werten im Teilvektor nicht auszuschließen. Im beiliegenden Programm wurde dieser Möglichkeit nicht Rechnung getragen, da der Fall eher unwahrscheinlich ist.

In einem Durchlauf des Experimentes ergab sich das folgende Ergebnis (siehe Abbildung 1):

Ergebnisse der Permutationstests auf Unabhängigkeit - für k >= 5t!

k = Anzahl Vektoren der Länge t, t! = Anzahl der Klassen

2!	3!	4!	5!	6!	7!	8!	9!	Anzahl Klassen
5.024	12.832	38.076	151.084	795.2	5235.759	40875.567	364548.72	0.975-Quantile*
	◇	◇	◇	◇				Annahmebereich
0.001	0.831	11.689	90.7	646.588	4842.241	39762.433	361209.28	0.025-Quantile
0.211093	6.66985	23.3321	116.165	731.45				numerischer Wert der Teststatistik

◇ = graphische Darstellung des Wertes der Teststatistik

*Berechnet mit Excel Microsoft 2002 mit "chiinv" für 2! ... 6!, mit "norminv" für 7! .. 9!, gerundet auf drei Stellen

Abbildung 1: Ausgabe für den Permutationstest für $t \in \{2,3,...,6\}$ $k < 5t!$ bisfür den Durchlauf eines Computerexperimentes (Anzahl Ereignisse $50'000$,Anzahl Zwischenkollisionen $36'420$)

Man kann feststellen, dass keiner der Testwerte außerhalb des Annahmebereichs liegt. Die These, dass die Zufallsvariablen unabhängig sind, muss also nicht verworfen werden. Die obigen Ergebnisse haben sich in etlichen

Durchläufen bestätigt, wobei ausnahmsweise Testwerte auch außerhalb des Annahmebandes lagen. Dies ist zu erwarten, da bei mehreren hundert Tests und bei einem Signifikanzniveau von 0.05 durchschnittlich ungefähr 5% der Testwerte außerhalb des Annahmebereichs liegen werden. Die obige Ausgabe wird durch die Routine „SchaetzenUndPermutationstest“ (für den Quellcode s. *Beilage 8.2*) geliefert. Die Zwischenkollisionszeiten, die im Experiment produziert werden, werden jeweils unter „out.txt“ abgespeichert.

Neben der eben behandelten Methode, Permutationstest für $t \in \{2, 3, ...\}$ für den gesamten Datenvektor zu berechnen − solange als $\frac{k}{t!} \geq 5$, wurde ein dynamischeres Experiment durchgeführt: Sei wiederum ein Datenvektor x mit Zwischenkollisionszeiten gegeben. Man startet mit einem Datenvektoren x^{1000} der Länge 1000, der die ersten 1000 Zwischenkollisionszeiten von x enthält. Das Programm berechnet den Permutationstest für ein frei zu wählendes t, fügt 250 Daten hinzu, führt den Permutationstest auf den Datenvektor der Länge 1250 mit t durch, etc. Die jeweiligen Resultate werden dann wiederum graphisch in einem Annahmeband dargestellt. Für einen Durchgang des Computerexperimentes mit $100'000$ Ereignissen und $73'513$ Kollisionen ergaben sich für den jeweils identischen Datensatz und für $t \in \{2, 3, 4, 5, 6\}$ die folgenden Resultate (siehe Abbildungen 2, 3, 4, 5 und 6)

Die obigen Berechnungen wurden für verschiedene Datensätze durchgeführt, und es ergaben sich jeweils vergleichbare Ergebnisse. Man kann diese durch die beiliegende Routine „PermutationstestVariableDatenVektoren“ (für den Quellcode s. *Beilage 8.3*) reproduzieren. Die Routine entnimmt die Daten der Datei „in.txt“. Will man analoge Graphiken und Berechnungen für durch die Routine „SchaetzenUndPermutationstest“ erzeugte Daten produzieren, muss man die Datei „out.txt“ unter dem Namen „in.txt“ im Verzeichnis mit den Routinen abspeichern.

Auch hier liegen die Testwerte im Allgemeinen innerhalb des Annahmebereichs. Die These, dass die Zufallsvariablen unabhängig sind, muss somit nicht verworfen werden. Die Anzahl der Testwerte außerhalb des Annahmebereichs liegt bei $\frac{4+5+2+0+0}{290 \cdot 5} \cdot 100 = 0.758\,62\%$. Bei $290 \cdot 5 = 1450$ Tests könnte dieser Wert als zu tief eingeschätzt werden, da man die Meinung vertreten könnte, es sollten bei einem Signifikanzniveau von 0.05 circa 5% der Testwerte außerhalb des Annahmebands liegen. Allerdings gilt es bei der Interpretation zu berücksichtigen, dass die Testwerte bezüglich eines Durchgangs mit fixem t nicht unabhängig sind: die alten Daten werden bei den folgenden Tests jeweils berücksichtigt. Testwerte liegen deshalb in der Nähe des jeweils vorausgegangenen Testwertes.

In einem nächsten Schritt ist nun die Verteilungsfamilie zu bestimmen. Dazu wird zuerst eine Kerndichteschätzung vorgenommen. Da exponentialverteilte Zufallsvariablen vermutet werden, ist zu beachten, dass die Dichtefunktion der Exponentialverteilung $f(t) = \lambda \exp(-\lambda t) \cdot 1_{[0,\infty)}$ in 0 eine Unstetigkeitsstelle aufweist, denn $f(0) = \lambda > 0$ und $f(t) = 0$ für $x \in (-\infty, 0)$.

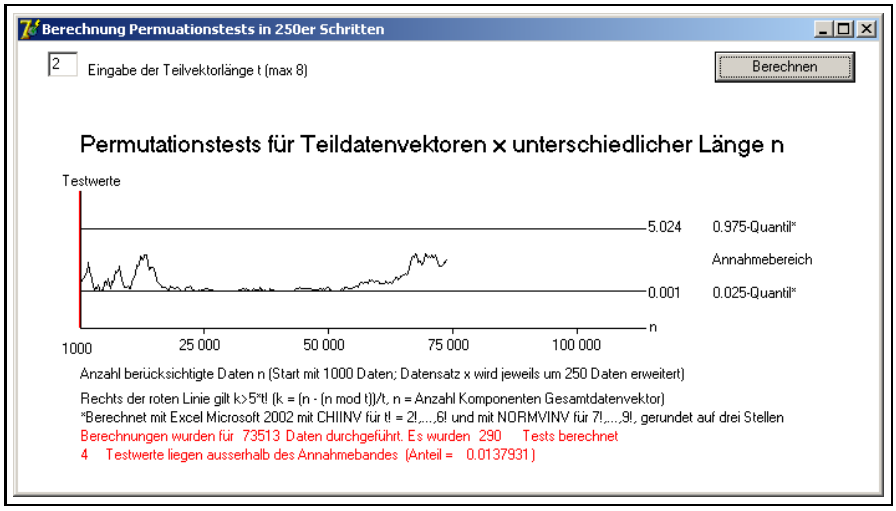

Abbildung 2: Permutationstest für jeweils um 250 Daten erweiterte Datenvektoren (100′000 Ereignisse, 73′513 Zwischenkollisionszeiten) mit Teilvektorelänge von 2

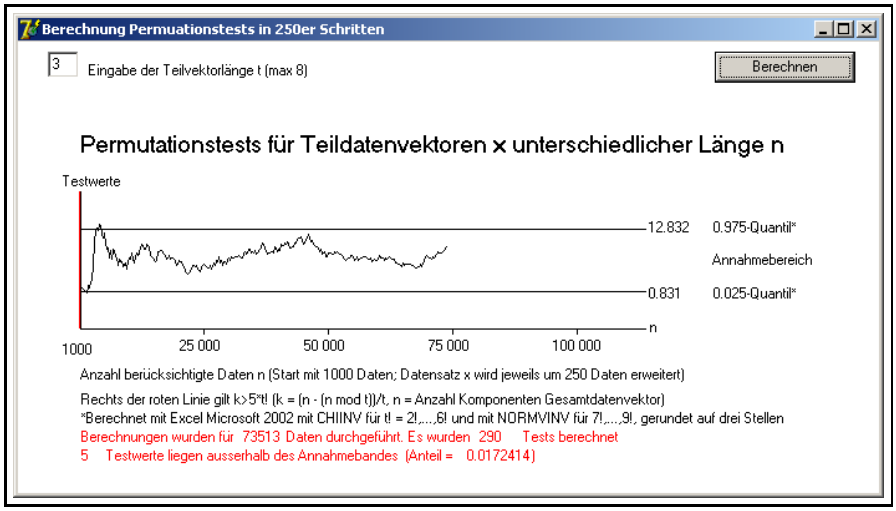

Abbildung 3: Permutationstests für jeweils um 250 Daten erweiterte Datenvektoren (100′000 Ereignisse, 73′513 Zwischenkollisionszeiten) mit Teilvektorlänge von 3

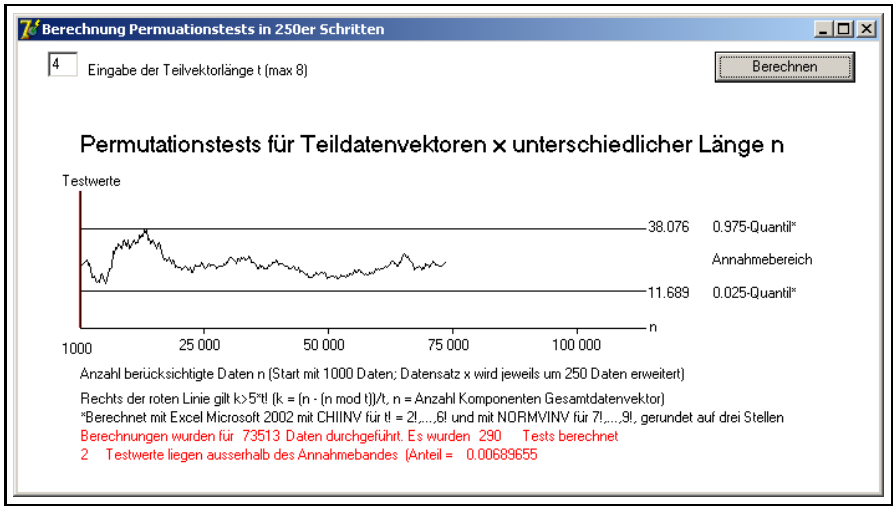

Abbildung 4: Permutationstests für jeweils um 250 Daten erweiterte Datenvektoren (100′000 Ereignisse, 73′513 Zwischenkollisionszeiten) mit Teilvektorlänge von 4

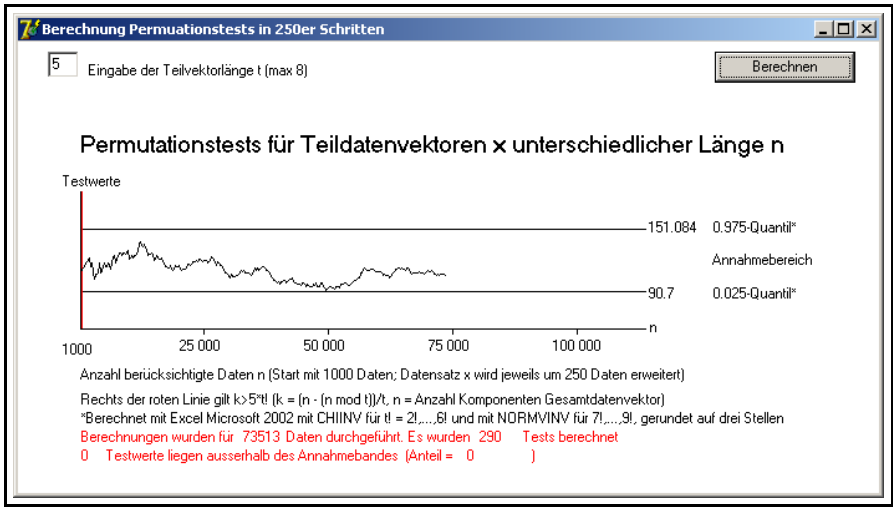

Abbildung 5: Permutationstests für jeweils um 250 Daten erweiterte Datenvektoren (100′000 Ereignisse, 73′513 Zwischenkollisionszeiten) mit Teilvektorlänge von 5

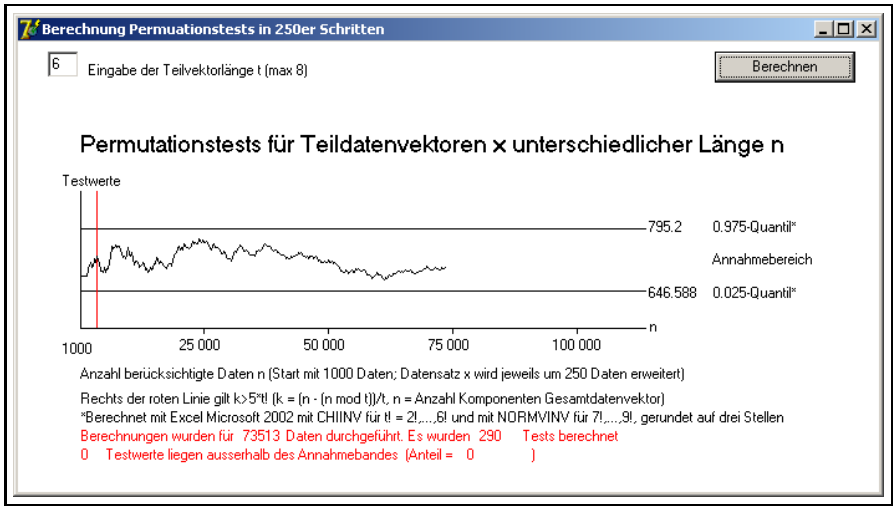

Abbildung 6: Permutationstests für jeweils um 250 Daten erweiterte Datenvektoren (100′000 Ereignisse, 73′513 Zwischenkollisionszeiten) mit Teilvektorlänge von 6

Damit ist eine Bedingung der Anwendung der Kerndichteschätzung nicht gegeben. Da nur positive Werte für die Zwischenkollisionszeiten möglich sind, kann der Wertebereich der Zufallsvariablen jedoch auf $(0, \infty)$ eingeschränkt werden. Auf diesem Bereich ist die Dichtefunktion der Exponentialverteilung stetig. Als Kern wird der Gausskern verwendet. Es wurde im Programm $h(n) = \frac{\bar{x}}{n^{0.4}}$ gewählt, wobei $\bar{x}$ der Mittelwert der Zwischenkollisionszeiten ist. Da $\bar{x} < \infty$, gilt $\lim\limits_{n\to\infty} h(n) = \lim\limits_{n\to\infty} \frac{\bar{x}}{n^{0.4}} = 0$ und $\lim\limits_{n\to\infty} nh(n) = \lim\limits_{n\to\infty} \frac{n\bar{x}}{n^{0.4}} = \infty$. Zudem ist die Dichtefunktion der Exponentialverteilung *lebesgue*-integrierbar. Damit gilt gemäß (56)

$$\lim_{n\to\infty} \int \left| \frac{1}{h(n)\sqrt{2\pi}} \sum_{i=1}^{n} \exp\left(-\frac{1}{2h(n)^2}(t - x_i)^2 \right) - f(t) \right| d\lambda\,(t) = 0$$

$$Exp(\lambda) - f.\ddot{u}.,$$

wobei f die zu schätzende Dichtefunktion der Wahrscheinlichkeitsverteilung $Exp(\lambda)$ ist.

Für einen Durchlauf des Experimentes mit 50′000 Ereignisse, davon 36′425 Zwischenkollisionszeiten, ergab sich die folgende Kerndichteschätzung (siehe Abbildung 7):

Verschiedene Durchläufe führten zu vergleichbaren Ergebnissen. Zu beachten ist, dass die Schätzungen in einer positiven Umgebung von 0 relativ langsam konvergieren, was angesichts der Ausgangslage nicht erstaunt: da keine negativen Zwischenkollisionszeiten vorkommen, fehlen in der Unsstetigkeitsstelle 0 für eine gute Schätzung von λ ungefähr die Hälfte der

Dichtefunktionen, als deren punktweises arithmetisches Mittel der Kerndichteschätzer definiert ist. Damit nimmt der Kerndichteschätzer in 0 näherungsweise den Wert $\frac{\lambda}{2}$ an. Im positiven Bereich ergibt sich eine gute Schätzung der Dichtefunktion $f(t) = \lambda \exp(-\lambda t)$ nur in (a, ∞) für ein $a > 0, a \in \mathbb{R}$, wobei $a \to 0$ für $n \to \infty$ (n Anzahl Zwischenkollisionszeiten). Vergleicht man unter diesem Vorbehalt die Kerndichte mit der Dichtefunktion einer Exponentialverteilung, so erkennt man, dass die Daten offenbar exponentialverteilt sind (siehe Abbildung 8):

Um diese Vermutung zu bestätigen, wird ein weiterer Durchlauf des Computerexperimentes durchgeführt (50′000 Ereignisse, davon 36′420 Kollisionen) und parametrisch aus der Klasse der Exponentialverteilungen mit Hilfe von $\hat{\lambda} := \frac{1}{\frac{1}{36420}\sum_{i=1}^{36420} x_i}[\frac{1}{s}]$ eine Dichtefunktion geschätzt, wobei x_i die Zwischenkollisionszeiten sind. Die parametrisch geschätzte Dichtefunktion $\hat{f}(t) = \hat{\lambda} \exp(-\hat{\lambda} t) \cdot 1_{[0,\infty)}$ wird in dasselbe kartesische Koordinatennetz wie die entsprechende Kerndichteschätzung gezeichnet. Es ergaben sich dabei die folgenden Resultate (siehe Abbildung 9):

Man erkennt optisch, dass die beiden geschätzten Dichtefunktionen bis auf eine kleine Umgebung von 0 sehr gut übereinstimmen. Die Koinzidenz könnte übrigens auch mit Hilfe von statistischen Test überprüft werden: man könnte ein empirisches Histogramm mit der geschätzten Dichtefunktion mit Hilfe eines χ^2-Anpassungstests oder die empirische Verteilungsfunktion mit der geschätzten Verteilungsfunktion mittels des Kolmogorov-Smirnov-Tests vergleichen (s. *Angewandte Statistik*, 5.3. und 5.4.). Eine optische Überprüfung soll hier genügen. Diese bestätigt einerseits die Vermutung, dass die Zwischenkollisionszeiten als Werte exponentialverteilter Zufallsvariablen betrachtet werden können. Zudem konnte mit Hilfe der parametrischen Schätzung eine konkrete Exponentialverteilung bestimmt werden, gemäß der die Zwischenkollisionszeiten bei einer Temperatur von 300 $[K]$ verteilt sind. Die Stichprobenzufallsvariablen sind exponentialverteilt − mit dem geschätzten Parameter $1.058237611 \cdot 10^6 [\frac{1}{s}]$. Das Resultat konnte mittels mehrerer Durchgänge des Experimentes reproduziert werden, wobei die Schätzer von λ natürlich eine Streuung aufweisen. Die obige Ausgabe kann vom Leser durch die beiliegende Routine „SchaetzenUndPermutationstest“ reproduziert werden, wobei die Temperatur und die Anzahl der Ereignisse vom Nutzer frei gewählt werden dürfen. Ungefähr 73% der Ereignisse sind Zwischenkollisionen.

Damit ist bestätigt, dass die Zufallsvariablen, die Zwischenkollisionszeiten als Werte annehmen, unabhängig und identisch exponentialverteilt sind. Daraus folgt mit *Satz 10*, dass der Kollisionszählprozess ein poissonscher ist.

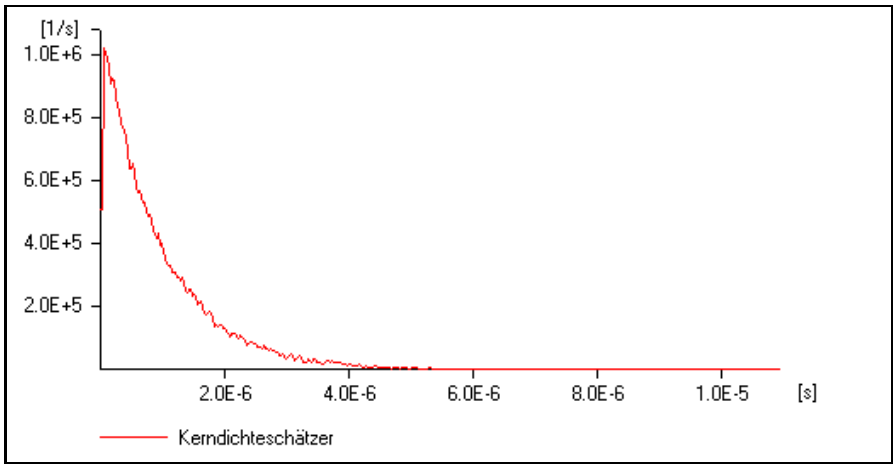

Abbildung 7: Kerndichteschätzer bezüglich der durch einen Durchlaufdes Computerexperimentes produzierten Zwischenkollisionszeiten(50′000 Ereignisse, davon 36′425 Zwischenkollisionszeiten)

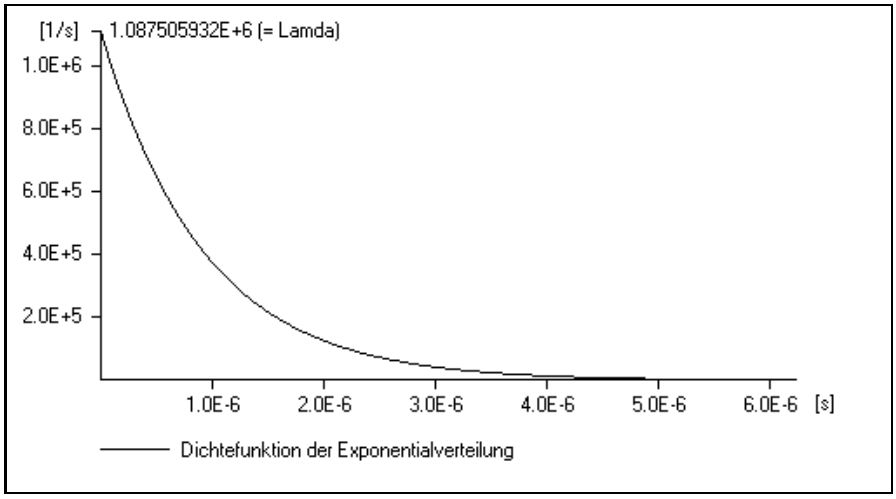

Abbildung 8: Dichtefunktion der Exponentialverteilung $Exp(1.087505932 \cdot 10^6[\frac{1}{s}])$

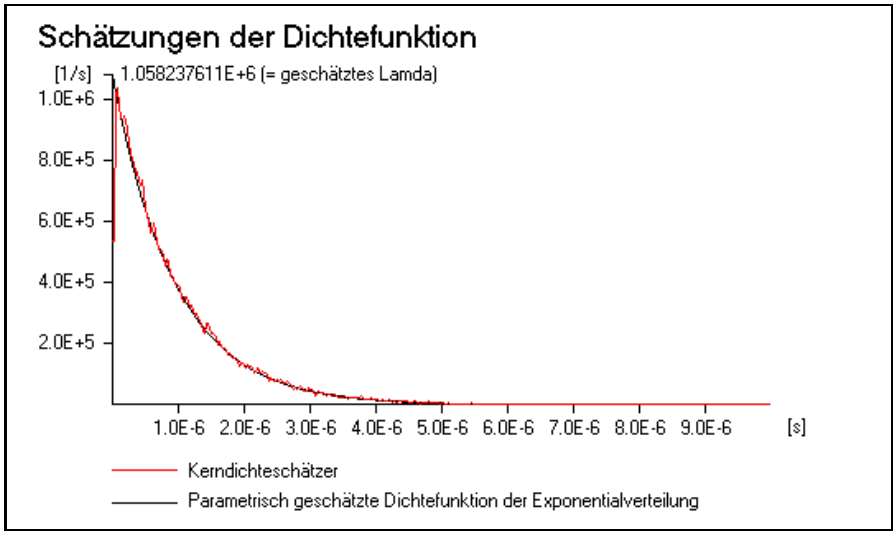

Abbildung 9: Kerndichteschätzer und parametrisch geschätzteDichtefunktion bezüglich der durch einen Durchlaufdes Computerexperimentes produzierten Zwischenkollisionszeiten(50′000 Ereignisse, davon 36′420 Zwischenkollisionszeiten)

5 Zusammenfassung

Die Diplomarbeit ging von der Modellierung eines realen Gases durch Ludwig Boltzmann (1844 - 1906) als System bewegter elastischer Kugeln in einem Behältnis B aus. Den Mikrobestandteilen wird dabei eine Newton-Dynamik auferlegt (Prinzip der Energie- und Impulserhaltung). Zudem wird davon ausgegangen, dass keine Energie in Rotationsenergie der Molekel verwandelt wird. Zu untersuchen war der stochastische Prozess, der die Zusammenstöße jeweils zweier Kugeln zählt.

In der Literatur wird die Vermutung vertreten, dass es sich bei diesem Zählprozess um einen Poisson-Prozess handelt. Dies kann allerdings durch die Laborphysik direkt nicht nachwiesen werden. Deshalb wurde hier versucht, diese Vermutung mittels eines Computerexerimentes und der Experimentellen Stochastik zu bestätigen: es wurde ein von den Betreuern zur Verfügung gestelltes zweidimensionales stochastisches, dynamisches Modell auf dem Rechner implementiert, um dann die interessierenden Makro-Größen mit Hilfe statistischer Methoden zu schätzen.

Da unabhängige, identisch exponentialverteilte Zufallsvariablen, welche Zwischenereigniszeiten als Werte annehmen, die Poissonität des Zählprozesses dieser Ereignisse implizieren, wurde in dieser Arbeit nachgewiesen, dass die Zwischenstoßzeiten als Werte unabhängiger, identisch exponentialverteilter Zufallsvariablen betrachtet werden können.

Nach der Behandlung der benötigten statistischen Konzepte und Me-

thoden wurde dazu kurz mit inhaltlichen Überlegungen dargelegt, dass die Zufallsvariablen als identisch verteilt vorausgesetzt werden können. Die Unabhängigkeit der Zufallsvariablen wurde mit Hilfe des Permutationstestes nachgewiesen. Dabei wurde der Permutationstest auf zwei Arten implementiert: Der gesamte Datenvektor wurde einerseits für verschiedene Teilvektorlängen getestet, andererseits wurden Permutationstests für Teilvektoren durchgeführt, die um jeweils 250 Daten erweitert wurden.

Anschliessend wurde eine Verteilungsfamilie mit Hilfe der Kerndichteschätzung bestimmt. Es zeigte sich, dass der Kerndichteschätzer näherungsweise die Form der Dichtefunktion einer Exponentialverteilung aufweist. Anschliessend wurde aus der Familie der Exponentialverteilung mit parametrischer Schätzung eine spezifische Verteilung geschätzt. Der Kerndichteschätzer und die parametrisch geschätzte Dichtefunktion wurden im selben Koordinatensystem übereinandergelegt und bei der optischen Überprüfung der Übereinstimmung ergab sich, dass die Graphen der beiden Funktionen sehr gut zu einander passen. Damit erwies sich, dass die Zwischenstoßzeiten tatsächlich als Werte exponentialverteilter Zufallsvariablen betrachtet werden können. Somit konnte die Poissonität des Stoßzählprozess durch die gewählten Methoden und Vorgehensweisen nachgewiesen werden.

6 Literatur

Literatur

[1] Agresti, A. (1990), *Categorical Data Analysis*, New York: Wiley.

[2] Cramér, H. (1946), *Mathematical Methods of Statistics*, Princeton: Princeton University Press.

[3] Deutschschweizerische Mathematikkommission et al. (Hrsg.) (1984), *Formeln und Tafeln*, Zürich: Orell Füssli.

[4] Devroye, L, Györfi, L. (1985), *Nonparametric Density Estimation,* New York.

[5] Hartung, J. et al. (1999), *Lehr- und Handbuch der angewandten Statistik*, München: Oldenbourg.

[6] http://www.schoenleber.org/pascal/pascal2-03.html [konsultiert 11. September 2005]; (s. *Beilage* 8.4).

[7] Knuth, Donald E. (1971), *The Art of Computer Programming*, Vol. 2, Seminumerical Algorithms, Reading: Addison-Wesley.

[8] Krengel, U. (1998), *Einführung in die Wahrscheinlichkeitstheorie und Statistik*, Braunschweig: Vieweg.

[9] Moeschlin, O., Grycko, E. (2004), *Experimental Stochastics in Physics*, Hagen: FernUniversität in Hagen.

[10] Moeschlin, O., Grycko, E., Pohl, C., Steinert, F. (1998), *Experimental Stochastics,* Berlin: Springer.

[11] Moeschlin, O., Grycko, E., Poppinga, C. (2001), *Angewandte Statistik*, Hagen: Kurs der FernUniversität Hagen (*Angewandte Statistik*)

[12] Moeschlin, O. et al.(1993), *Wahrscheinlichkeitstheorie I*, Hagen: Kurs der FernUniversität Hagen (*WI*).

[13] Müller, P. H. (Hrsg.) (1991), *Lexikon der Stochastik*, Berlin: Akademieverlag.

[14] Nadaraja, É. A. (1965), On non-parametric estimates of density functions and regression curves. *Theor. Probability and Appl.,* 9, 497-500.

[15] Stahel, W. A., (1995), *Statistische Datenanalyse: Eine Einführung für Naturwissenschaftler*, Braunschweig: Vieweg.

7 Index der Symbole

$:=$	: identisch *per definitionem*.
$x \in A$	: x ist Element der Menge A.
$\{x \mid Q(x)\}$	: die Menge aller x, so dass x ein Q ist (Q ist ein Prädikat).
$[m]$	: Längeneinheit Meter.
$[s]$	: Zeiteinheit Sekunden.
$[K]$	: Temperatureinheit Kelvin.
N	: Anzahl Mikrobestandteile des Fluids.
$r := \sqrt{\frac{\pi}{500 \cdot N \cdot 2 \cdot \sqrt{3}}}[m]$	: Radius der Mikrobestandteile.
$R + r$	: Radius des Behältnisses B.
m	: Masse der einzelnen Mikrobestandteile.
(x, y)	: zweidimensionaler Vektor mit den Komponenten x und y.
$\mathbb{N}$	: die Menge der natürlichen Zahlen.
$\mathbb{Z}$	: die Menge der ganzen Zahlen.
$\mathbb{R}$	: die Menge der reellen Zahlen.
$\mathbb{N}_n := \{x \mid x \in \mathbb{N} \text{ und } x \leq n\}$	: die Menge der ersten n natürlichen Zahlen.
$\mathbb{N}^0 := \{x \mid x \in \mathbb{N} \cup \{0\}\}$	: die Menge der natürlichen Zahlen inklusive 0.
$\mathbb{R}_+ := \{x \mid x \in \mathbb{R} \text{ und } x \geq 0\}$	: die Menge der positiven reellen Zahlen.
$A \times B := \{(x, y) \mid x \in A \text{ und } y \in B\}$	: das kartesische Produkt der Mengen A und B.
$\mathbb{R}^n := \mathbb{R}^{n-1} \times \mathbb{R}$ für $n > 1$ und $\mathbb{R}^1 = \mathbb{R}$	: das n-fache kartesische Produkt der reellen Zahlen mit sich selber.
$B := \{(x_1, x_2) \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq (R + r)^2\}$	: Behältnis mit Mikrobestandteilen.
$B' := \{(x_1, x_2) \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq R^2\}$	: Teilfläche des Behältnisses, auf dem die Mittelpunkte der Mikrobestandteile verteilt werden.
$x^{(i)} := \left(x_1^{(i)}, x_2^{(i)}\right)$	: zweidimensionaler Ortsvektor des Mittelpunktes des i-ten Mikrobestandteils.
$v^{(i)} := \left(v_1^{(i)}, v_2^{(i)}\right)$	: zweidimensionaler Geschwindigkeitsvektor des i-ten Mikrobestandteils.
$\langle x, y \rangle := x_1 y_1 + x_1 y_2$	: Standardskalarprodukt der Vektoren x und y.
$\|x\| := \sqrt{x_1^2 + x_2^2}$	: Betrag des zweidimensionalen Vektors x.
$u^{(i)} := \left(u_1^{(i)}, u_2^{(i)}\right) := mv^{(i)}$	: zweidimensionaler Impulsvektor des i-ten Mikrobestandteils.
$E : \mathbb{R}^2 \to \mathbb{R}_+; E(u^{(i)}) := \frac{1}{2m^{(i)}} \left\langle u^{(i)}, u^{(i)} \right\rangle$	: Energie des Mikrobestandteils i.
$U(0, 1)$	: Stetige Uniforme Verteilung mit $f(x) = 1_{[0,1]}(x)$.

rand_i : Werte von nach $U(0,1)$ verteilter Zufallsvariablen.

$N_t := |\{i \in \mathbb{N} \mid S_i \leq t\}|$: Anzahl der Summen S_i von Zwischenkollisionszeiten, so dass S_i kleiner oder gleich t (= Anzahl der Kollisionen im Zeitraum $[0,t]$).

$I_2 := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$: Identitätsmatrix der Dimension 2.

$N(0, \sigma^2 I_2)$: Zweidimensionale Normalverteilung.

$N(a,b)$: Normalverteilung mit Erwartungswert a und Varianz b.

T : Temperatur in Kelvin.

$k_B = 1.38 \cdot 10^{-23} \frac{[J]}{[K]}$: Boltzmann-Konstante.

$[J]$: Energieeinheit Joule.

∂B : Topologischer Rand der Menge B.

$d(x,y) := \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$: Euklidische Distanz auf $\mathbb{R}^2$ der Punkte x und y.

$\Longleftrightarrow$: genau dann, wenn.

$\bar{t}_i^k$: i-te Zwischenstoßzeit (Zwischenkollisionszeit).

$\bar{t}_i^r$: i-te Zwischenreflexionszeit.

$\bar{t}_i$: i-te Zwischenereigniszeit (Ereignisse sind Kollisionen oder Reflexionen).

$sz_i := \sum_{j=1}^{i} \bar{t}_j$: Summe der ersten i Zwischenereigniszeiten $\bar{t}_j$.

$sz_i^r := \sum_{j=1}^{i} \bar{t}_j^r$: Summe der ersten i Zwischenreflexionszeiten $\bar{t}_j^r$.

$sz_i^k := \sum_{j=1}^{i} \bar{t}_j^k$: Summe der ersten i Zwischenkollisionszeiten $\bar{t}_j^k$.

$(x_i)_{i \in \mathbb{N}_m}$: endliche Folge von reellen Zahlen.

$(X_i)_{i \in \mathbb{N}_n}$: endliche Folge von Zufallsvariablen.

$n \bmod t = b \Longleftrightarrow \frac{n-b}{t} \in \mathbb{Z}$; $n, t, b \in \mathbb{Z}$ und $b < t$: kleinster ganzzahliger Rest bei Division in den ganzen Zahlen.

$n' = n - n \bmod t$: im Permutationstest verwendete Gesamtvektorlänge

$\mathcal{B}^n$: Borelsche $\sigma-$Algebra auf $\mathbb{R}^n$.

$(\mathbb{H}, \mathcal{H}, P)$: Wahrscheinlichkeitsraum mit $\sigma-$Algebra $\mathcal{H}$ auf $\mathbb{H}$ und Wahrscheinlichkeitsverteilung P auf $\mathcal{H}$.

$(\mathbb{R}^n, \mathcal{B}^n, \mathcal{W}, \mathcal{W}_1, \mathcal{W}_2)$: Testexperiment mit Hypothese $\mathcal{W}_1$ und Alternative $\mathcal{W}_2$ (s. Seite 11).

$(\mathbb{R}^{n'}, \mathcal{B}^{n'}, \mathcal{W}', \mathcal{W}_1', \mathcal{W}_2')$: in Testexperiment $(\mathbb{R}^n, \mathcal{B}^n, \mathcal{W}, \mathcal{W}_1, \mathcal{W}_2)$ n durch n' ersetzt.

$\mathbb{R}^{t*}$: Menge der Vektoren der Länge t, die nur ungleiche Komponenten enthalten.

$V_l := (X_{(l-1)t+1}, ..., X_{(l-1)t+t})$: $l-$ter Teilvektor der Länge t des Vektors von Zufallsvariablen $X = (X_1, ..., X_n)$.

$v_l := (x_{(l-1)t+1}, ..., x_{(l-1)t+t})$: l-ter Teilvektor der Länge t des Vektors x.

$\mathcal{S}_n$	: Menge der Permutationen von $\mathbb{N}_n$ nach $\mathbb{N}_n$.
$\|A\|$, für abzählbare Mengen A	: Anzahl der Elemente der abzählbaren Menge A.
$\|x\|$, für reelle Zahlen x	: Betrag der reellen Zahl x.
$I_v(v_j)$	: Position der Komponente v_j im Vektor v.
$O(v)$	: Ordnungsstruktur des Vektors v.
$[a_1, b_1) := \{x \mid a \leq x < b; x \in \mathbb{R}\}, a, b \in \mathbb{R}$	: rechtshalboffenes, reelles Intervall.
D_i	: Menge der Vektoren mit der Ordnungsstruktur i.
$A \cap B$	: Schnittmenge von A und B.
$A \cup B$	: Vereinigungsmenge von A und B.
$\bigcup_{i=1}^{n} A_i = A_1 \cup ... \cup A_n$	: Vereinigungsmenge von n Mengen A_i.
$A - B := \{x \mid x \in A \text{ und } x \notin B\}$	: Differenzmenge von A und B.
$1_A(x) := 1 \Longleftrightarrow x \in A; 0$ sonst	: Indikatorfunktion der Menge A.
$P \otimes Q$	: Produktemaß der Wahrscheinlichkeitsmaße P und Q.
P_T	: Bildmaß von P unter der messbaren Abbildung T.
$P * Q := (P \otimes Q)_T$	: Faltung von P und Q (T = Zufallsvariable „Summe“).
$\delta_{\bar{x}}$	: Punktwahrscheinlichkeitsmaß in Punkt $\bar{x}$.
$P^n := \bigotimes_{i=1}^{n} P$	: n-faches Produktemaß des Wahrscheinlichkeitsmaßes P.
$E_P(X_i)$	: Erwartungswert der Zufallsvariablen X_i unter dem Wahrscheinlichkeitsmaß P.
$V_P(X_i)$	: Varianz der Zufallsvariable X_i unter Wahrscheinlichkeitsmaß P.
$\chi^2_{(k)}$	: Chiquadratverteilung mit k Freiheitsgraden.
$Q_n^{x^n}$	: empirische Verteilung der Stichprobe X^n unter P_0^n.
$(\mathbb{R}^\infty, \mathcal{B}^\infty, \mathcal{W}, h)$	: Schätzexperiment (s. Seite 20).
Id_A	: Identitätsfunktion auf die Menge A.
$Exp(\lambda)$	: Exponentialverteilung mit dem Parameter λ.
$\hat{\lambda}$	: geschätztes λ.
$\bar{X}_n(x) := \frac{1}{n} \sum_{i=1}^{n} X_i(x)$	: Zufallsvariable „Mittelwert“ ($X_i(x) = x_i$).
$T_i(\omega) = t_i$	: Zufallsvariablen „Zwischenkollisionszeiten“
$S_m := \sum_{i=1}^{m} T_i$ für $m \in \mathbb{N}$	: Zufallsvariable „Summe“ von Zwischenkollisionszeiten
$i.i.d$	: identisch und unabhängig verteilt

8 Anhänge

```
PROGRAM vorlage; {Beilage 7.1: Von Betreuern zur Verfügung gestellte Routine}
{$N+}
USES CRT,GRAPH;
CONST N = 2000; (* Anzahl der Mikrobestandteile*)
dim = 2; (* Dimension *)
NA :EXTENDED = 6.02E26; (* Avogadro - Zahl *)
kB :EXTENDED = 1.380662E-23; (* Boltzmann Konstante *)
Rg :EXTENDED = 1.0;
pinf :EXTENDED = 1.0E300; (* numerisches "plus unendlich" *)
f :EXTENDED = 500.0; (* Volumenfaktor *)
eps :EXTENDED = 1.0E-9;
TYPE tvektor = ARRAY[1..dim]OF EXTENDED;
tzustand = RECORD
pos,vel :tvektor;
szeit,radius,mass :EXTENDED;
wand :SHORTINT;
collpart :INTEGER;
aktiv :BOOLEAN;
END;
tfluid = ARRAY[1..N]OF ^tzustand;
VAR fluid :tfluid;
Temp,sigma,Time :EXTENDED;
counter :LONGINT;
PROCEDURE NextReflexion(index :INTEGER; VAR t :EXTENDED);
VAR z :tzustand;
x,xx,xv,vv,norm,diskr,t2 :EXTENDED;
BEGIN
z := fluid[index]^;
t := pinf;
xx := z.pos[1]*z.pos[1] + z.pos[2]*z.pos[2];
xv := z.pos[1]*z.vel[1] + z.pos[2]*z.vel[2];
vv := z.vel[1]*z.vel[1] + z.vel[2]*z.vel[2];
norm := SQRT(xx);
IF ABS(Rg-norm)<(Rg*eps) THEN
BEGIN
t := -2.0*xv/vv;
z.wand := 1;
END
ELSE
BEGIN
diskr := xv*xv + vv*(Rg*Rg-xx);
t := (-xv + SQRT(diskr)) / vv;
z.wand := 1;
END;

fluid[index]^ := z;
END;
PROCEDURE NextMomentumExchange(index1,index2 :INTEGER; VAR t :EXTENDED);
VAR z1,z2 :tzustand;
diskr,xx,xv,vv :EXTENDED;
BEGIN
z1 := fluid[index1]^;
z2 := fluid[index2]^;
xv := (z2.pos[1]-z1.pos[1])*(z2.vel[1]-z1.vel[1]) +
(z2.pos[2]-z1.pos[2])*(z2.vel[2]-z1.vel[2]);
xx := SQR(z2.pos[1]-z1.pos[1]) + SQR(z2.pos[2]-z1.pos[2]);
vv := SQR(z2.vel[1]-z1.vel[1]) + SQR(z2.vel[2]-z1.vel[2]);
diskr := SQR(xv) - vv*(xx-SQR(z1.radius + z2.radius));
IF (xv>=0.0) THEN t := pinf;
IF ((xv<0.0) AND (diskr<=0.0)) THEN t := pinf;
IF ((xv<0.0) AND (diskr>0.0))
THEN t := (-xv-SQRT(diskr)) / vv;
END;
PROCEDURE NewPartner(index :INTEGER);
VAR mintim,sz :EXTENDED;
ap,i,minind :INTEGER;
BEGIN
NextReflexion(index,sz);
```

```
fluid[index]^.collpart := 0;
mintim := sz;
minind := 0;
fluid[index]^.szeit := sz;
FOR i := 1 TO N DO
BEGIN
IF ( (i<>index) AND fluid[i]^.aktiv)THEN
BEGIN
NextMomentumExchange(i,index,sz);
IF sz<mintim THEN
BEGIN
minind := i;
mintim := sz;
END;
END;
END;
IF minind>0THEN
BEGIN
fluid[index]^.collpart := minind;
fluid[index]^.szeit := mintim;
IF fluid[minind]^.szeit>mintim THEN
BEGIN
fluid[minind]^.collpart := index;
fluid[minind]^.szeit := mintim;
END;
END;
END;
PROCEDURE FindePartner(index1,index2 :INTEGER);
BEGIN
fluid[index2]^.aktiv := FALSE;
NewPartner(index1);
fluid[index2]^.aktiv := TRUE;
fluid[index1]^.aktiv := FALSE;
NewPartner(index2);
fluid[index1]^.aktiv := TRUE;
END;


PROCEDURE init;
VAR overlap :BOOLEAN;
xi,rad,phi,dist,volu,rr :EXTENDED;
i,i2 :INTEGER;
BEGIN
volu := PI * Rg * Rg;
rad := SQRT(volu/f/N/2.0/SQRT(3.0));
FOR i := 1 TO N DO
BEGIN
fluid[i]^.radius := rad;
fluid[i]^.mass := 39.95/NA;
fluid[i]^.collpart := 0;
fluid[i]^.szeit := pinf;
fluid[i]^.aktiv := TRUE;
END;
Temp := 300.0;
Time := 0.0;
sigma := SQRT(kB*Temp/fluid[1]^.mass);
FOR i := 1 TO N DO
BEGIN
rad := sigma * SQRT(-2.0*LN(RANDOM));
phi := 2.0 * PI * RANDOM;
fluid[i]^.vel[1] := rad * COS(phi);
fluid[i]^.vel[2] := rad * SIN(phi);
rad := fluid[i]^.radius;
REPEAT
rr := Rg*EXP(LN(RANDOM)/2.0);
phi := 2.0*PI * RANDOM;
fluid[i]^.pos[1] := rr * COS(phi);
fluid[i]^.pos[2] := rr * SIN(phi);
```

```
overlap := FALSE;
FOR i2 := 1 TO i-1 DO
BEGIN
dist := SQR(fluid[i]^.pos[1]-fluid[i2]^.pos[1]);
dist := dist + SQR(fluid[i]^.pos[2]-fluid[i2]^.pos[2]);
dist := SQRT(dist);
IF (dist<1.0001*(rad+fluid[i2]^.radius)) THEN
overlap := TRUE;
END;
UNTIL NOT(overlap);
END;
FOR i := 1 TO N DO
NewPartner(i);
END;


PROCEDURE CarryOutReflexion(index :INTEGER);
VAR z :tzustand;
i :INTEGER;
ev,nv :tvektor;
winkel,phi,cw,vev,R,norm :EXTENDED;
BEGIN
z := fluid[index]^;
CASE z.wand OF
1 :BEGIN
R := Rg;
ev[1] := z.pos[1]/R;
ev[2] := z.pos[2]/R;
vev := z.vel[1]*ev[1] + z.vel[2]*ev[2];
z.vel[1] := z.vel[1] - 2.0 * vev * ev[1];
z.vel[2] := z.vel[2] - 2.0 * vev * ev[2];
END;
END;
fluid[index]^ := z;
NewPartner(index);
FOR i := 1 TO N DO
IF fluid[i]^.collpart=index THEN NewPartner(i);
END;
PROCEDURE ExchangeMomentum(index1,index2:INTEGER);
VAR z1,z2 :tzustand;
diskr,norm,epxv,m1,m2 :EXTENDED;
cp,i,index :INTEGER;
ep :tvektor;
BEGIN
z1 := fluid[index1]^;
z2 := fluid[index2]^;
m1 := z1.mass;
m2 := z2.mass;
norm := 0.0;
FOR index := 1 TO dim DO
norm := norm + SQR(z2.pos[index] - z1.pos[index]);
norm := SQRT(norm);
ep[1] := (z2.pos[1] - z1.pos[1]) / norm;
ep[2] := (z2.pos[2] - z1.pos[2]) / norm;
epxv := 0.0;
FOR index := 1 TO dim DO
epxv := epxv + ep[index] * (z2.vel[index] - z1.vel[index]);
FOR index := 1 TO dim DO
BEGIN
z1.vel[index] := z1.vel[index] + 2.0*m2/(m1+m2) * epxv * ep[index];
z2.vel[index] := z2.vel[index] - 2.0*m1/(m1+m2) * epxv * ep[index];
END;
fluid[index1]^ := z1;
fluid[index2]^ := z2;
FOR i := 1 TO N DO
BEGIN
cp := fluid[i]^.collpart;
IF ((cp=index1) OR (cp=index2)) THEN NewPartner(i);
```

```
END;
FindePartner(index1,index2);
END;
PROCEDURE MoveSystem(t:EXTENDED);
VAR index :INTEGER;
BEGIN
FOR index := 1 TO N DO
fluid[index]^.pos[1] := fluid[index]^.pos[1] + t*fluid[index]^.vel[1] ;
FOR index := 1 TO N DO
fluid[index]^.pos[2] := fluid[index]^.pos[2] + t*fluid[index]^.vel[2];
FOR index := 1 TO N DO
fluid[index]^.szeit := fluid[index]^.szeit - t;
END;


PROCEDURE zeichne_zustand;
VAR index,rint,mx,my,xk,yk,xzero,yzero:INTEGER;
scale,yscale,phi :EXTENDED;
s :STRING;
BEGIN
xzero := 300;
yzero := 200;
SETCOLOR(12);
scale := 170.0/Rg;
CIRCLE(xzero,yzero,170);
rint := ROUND(fluid[1]^.radius*scale);
SETCOLOR(14);
FOR index := 1 TO N DO
BEGIN
mx := xzero + ROUND(fluid[index]^.pos[1] * scale);
my := yzero - ROUND(fluid[index]^.pos[2] * scale);
CIRCLE(mx,my,rint);
END;
END;
PROCEDURE durchlauf;
VAR mt,dt :EXTENDED;
mi,mi2,i :INTEGER;
BEGIN
mt := pinf;
mi := 0;
FOR i := 1 TO N DO
IF mt>fluid[i]^.szeit THEN
BEGIN
mt := fluid[i]^.szeit;
mi := i;
END;
dt := fluid[mi]^.szeit;
Time := Time + dt;
MoveSystem(dt);
IF fluid[mi]^.collpart=0 THEN CarryOutReflexion(mi)
ELSE
BEGIN
mi2 := fluid[mi]^.collpart;
ExchangeMomentum(mi,mi2);
END;
END;
PROCEDURE ig;
VAR modus,treiber :INTEGER;
BEGIN
DETECTGRAPH(modus,treiber);
INITGRAPH(modus,treiber,'c:\bp\bgi');
CLEARVIEWPORT;
END;
PROCEDURE speicher;
VAR i :INTEGER;
BEGIN
FOR i := 1 TO N DO
NEW(fluid[i]);
```

```
END;



BEGIN
RANDOMIZE;
CLRSCR;
speicher;
ig;
init;
Zeichne_Zustand;
counter := 0;
REPEAT
INC(counter);
durchlauf;
IF (counter MOD 1000)=0 THEN
BEGIN
CLEARVIEWPORT;
Zeichne_Zustand;
END;
UNTIL 0>1;
FOR counter := 1 TO N DO
DISPOSE(fluid[counter]);END.
```

```
unit Unit1;  {Beilage 7.2: Quellcode von SchaetzenUndPermutationstest}

interface


uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Menus;

CONST N = 2000;      (* Anzahl der Mikrobestandteile*)
      dim = 2;                (*  Dimension *)
      NA :EXTENDED = 6.02E26;   (* Avogadro - Zahl *)
      kB :EXTENDED = 1.380662E-23;  (* Boltzmann Konstante *)
      Rg :EXTENDED = 1.0;     (* Radius des Behälters *)
      pinf :EXTENDED = 1.0E300;  (* numerisches "plus unendlich" *)
      f :EXTENDED = 500.0;    (* Volumenfaktor *)
      eps :EXTENDED = 1.0E-9;
      ETypKonst = 2;  {1 für Reflexionen, 2 für Kollisionen}
      abstandx=200;
      abstandy=430;
      eckex = 245;
      eckey = 260;

type

  tvektor = ARRAY[1..dim]OF EXTENDED; {dim hier = 2}
     tzustand = RECORD
                pos,vel :tvektor;  {Vektoren für die Position und die
                                    Geschwindigkeit eines Molekels}
                szeit,radius,mass :EXTENDED;  {Zeit zwischen Ereignisssen;
                                     Radius der Molekel; Masse der Molekel}
                wand :SHORTINT;
                collpart :INTEGER;
                aktiv :BOOLEAN;
                END;
     tfluid = ARRAY[1..N]OF ^tzustand;
                                  {Vektor aller Zeiger auf die Molekelrecords}
     tintervallrecord = RECORD   {Dient dazu das i-te Zeitintervall zu speichern
                                  und anzugeben, ob es sich beim i-ten Ereignis
                                  um eine Kollison oder um eine Reflexion
handelt}
                zeitintervall : EXTENDED;
                ereignistyp : shortint; {1 für Kollision; 2 für Reflexion}
                end;

     tZeitintervallstore = Array of ^tintervallrecord;

     extendedvektor=array of extended;
     realmatrix=array of extendedvektor;

     str = STRING [80];

     integervektor = array of integer;

     integermatrix=array of integervektor;



     TForm1 = class(TForm)
      Button1: TButton;
    Label1: TLabel;
    Edit1: TEdit;
    Label2: TLabel;
    Edit2: TEdit;
    Label3: TLabel;
    Label4: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure Edit1Change(Sender: TObject);
```

```
    procedure Edit2Change(Sender: TObject);
    end;

var
  Form1: TForm1;

  fluid :tfluid;
  Temp,sigma,Time,ZWZeit :EXTENDED;
  counter,ii:LONGINT;
  Zeitintervallspeicher:tZeitintervallstore;
  k:shortint;
  x,x1:extended;
  s,temp1,AnzahlRunden1:string;
  i,j,int,inter,t,kk,zaehler,permutationszaehler,weite,AnzahlRunden:integer;
  {inter (globale Variable) reserviert für Ausgabe Ordnungszahl der
Permutationen;
  permutationszaehler (globale Variable) reserviert fürs Zählen der Permuationen
  in der Permutationsprozedur;
  t globale Variable reserviert für Teilvektorlänge;
  kk globale Variable reserviert für Anzahl Teilvektoren;}
  alpha :char;
  w:single;
  schreibe:boolean=false;
  SpeicherFuerTyp,v: extendedvektor;
  liste,liste1 : str;
  a: integermatrix;
  b,b1: realmatrix;
  c: Integervektor;




implementation


{Für ein konkretes Molekel "index" wird der Zeitpunkt des nächsten
Zusammenpralls mit der Wand des Behälters berechnet}
PROCEDURE NextReflexion(index :INTEGER; VAR t :EXTENDED);
VAR z :tzustand;
    xx,xv,vv,norm,diskr :EXTENDED;
BEGIN
z := fluid[index]^;
t := pinf;
xx := z.pos[1]*z.pos[1] + z.pos[2]*z.pos[2];
xv := z.pos[1]*z.vel[1] + z.pos[2]*z.vel[2];
vv := z.vel[1]*z.vel[1] + z.vel[2]*z.vel[2];
norm := SQRT(xx);
IF ABS(Rg-norm)<(Rg*eps) THEN
                         BEGIN
                         t := -2.0*xv/vv;
                         z.wand := 1;
                         END
                         ELSE
                         BEGIN
                         diskr := xv*xv + vv*(Rg*Rg-xx);
                         t := (-xv + SQRT(diskr)) / vv;
                         z.wand := 1;
                         END;
fluid[index]^ := z;
END;

{Für zwei konkrete Molekel "index1" und "index2" wird der Zeitpunkt
bis zur nächsten Kollision berechnet - falls es eine solche gibt. Wenn es eine
solche nicht gibt, ist t = pinf}
PROCEDURE NextMomentumExchange(index1,index2 :INTEGER; VAR t :EXTENDED);
VAR z1,z2 :tzustand;
    diskr,xx,xv,vv :EXTENDED;
```

```
BEGIN
z1 := fluid[index1]^;
z2 := fluid[index2]^;
xv := (z2.pos[1]-z1.pos[1])*(z2.vel[1]-z1.vel[1]) +
      (z2.pos[2]-z1.pos[2])*(z2.vel[2]-z1.vel[2]);
xx := SQR(z2.pos[1]-z1.pos[1]) + SQR(z2.pos[2]-z1.pos[2]);
vv := SQR(z2.vel[1]-z1.vel[1]) + SQR(z2.vel[2]-z1.vel[2]);
diskr := SQR(xv) - vv*(xx-SQR(z1.radius + z2.radius));
IF (xv>=0.0) THEN t := pinf;
IF  ((xv<0.0) AND (diskr<=0.0)) THEN t := pinf;
IF ((xv<0.0) AND (diskr>0.0))
  THEN  t := (-xv-SQRT(diskr)) / vv;
END;

{Sucht nächste Kollision oder nächste Reflexion im Gesamtsystem}
PROCEDURE NewPartner(index :INTEGER);
VAR mintim,sz :EXTENDED;
    i,minind :INTEGER;
BEGIN
NextReflexion(index,sz);
fluid[index]^.collpart := 0;   {Wenn Reflexion stattfindet, dann collpart = 0,
                                sonst nachher auf minind gesetzt}
mintim := sz;
minind := 0;
fluid[index]^.szeit := sz;     {Zeit im Falle einer Reflexion wird zugeordnet}
{Es werden alle Kollisionen berechnet, falls es keine gibt, t = unendlich
Es wird die Kürzeste Kollisionszeit zurückbehalen und der Index des
entsprechenden
Molekels mit mintim identifiziert. Das ist dann der Kollisionpartner, wenn die
Zeit kürzer ist als die Zeit bis zur Reflexion}
FOR i := 1 TO N DO
   BEGIN
   IF ( (i<>index) AND fluid[i]^.aktiv)THEN
     BEGIN
     NextMomentumExchange(i,index,sz);
     IF sz<mintim THEN
       BEGIN
       minind := i;
       mintim := sz;
       END;
     END;
   END;
IF minind>0 THEN
           BEGIN
           fluid[index]^.collpart := minind;
           fluid[index]^.szeit := mintim;
           IF fluid[minind]^.szeit>mintim THEN
              BEGIN
              fluid[minind]^.collpart := index;
              fluid[minind]^.szeit := mintim;
              END;
           END;
END;


PROCEDURE FindePartner(index1,index2 :INTEGER);
BEGIN
fluid[index2]^.aktiv := FALSE;
NewPartner(index1);
fluid[index2]^.aktiv := TRUE;
fluid[index1]^.aktiv := FALSE;
NewPartner(index2);
fluid[index1]^.aktiv := TRUE;
END;



{Initiatlisiert die Records der N Molekel. }
```

```
PROCEDURE init;
VAR overlap :BOOLEAN;
    rad,phi,dist,volu,rr :EXTENDED;
    i,i2 :INTEGER;
BEGIN
volu := PI * Rg * Rg;
rad := SQRT(volu/f/N/2.0/SQRT(3.0));  {Radius der Molekel}
FOR i := 1 TO N DO
   BEGIN
   fluid[i]^.radius := rad;
   fluid[i]^.mass := 39.95/NA;
   fluid[i]^.collpart := 0;
   fluid[i]^.szeit := pinf;
   fluid[i]^.aktiv := TRUE;
   END;

Time := 0.0;
sigma := SQRT(kB*Temp/fluid[1]^.mass);  {die Standardabweichung der
Normalverteilung
wird in Abhängigkeit von der Temperatur festgelegt; laut MaxwellFormel}
FOR i := 1 TO N DO
   BEGIN
   rad := sigma * SQRT(-2.0*LN(RANDOM));   {Initiatilierung der
Geschwindigkeitsvektoren}
   phi := 2.0 * PI * RANDOM;               {Verteilung: N(0,sigma^2}
   fluid[i]^.vel[1] := rad * COS(phi);
   fluid[i]^.vel[2] := rad * SIN(phi);
   rad := fluid[i]^.radius;
   REPEAT
   rr := Rg*EXP(LN(RANDOM)/2.0);   {Initialisierung der Ortsvektoren}
   phi := 2.0*PI * RANDOM;   {Uniforme Verteilung auf den Behälter}
   fluid[i]^.pos[1] := rr * COS(phi);
   fluid[i]^.pos[2] := rr * SIN(phi);
   overlap := FALSE;
   FOR i2 := 1 TO i-1 DO {Wenn die Molekel überlappen, werden sie getrennt}
      BEGIN
      dist := SQR(fluid[i]^.pos[1]-fluid[i2]^.pos[1]);
      dist := dist + SQR(fluid[i]^.pos[2]-fluid[i2]^.pos[2]);
      dist := SQRT(dist);
      IF (dist<1.0001*(rad+fluid[i2]^.radius)) THEN
         overlap := TRUE;
      END;
  UNTIL NOT(overlap);
  END;
FOR i := 1 TO N DO
   NewPartner(i);
END;



{Berechnet neuen Geschwindigkeitsvektor für das Molekel, das reflectiert wird}
PROCEDURE CarryOutReflexion(index :INTEGER);
VAR z :tzustand;
    i :INTEGER;
    ev :tvektor;
    vev,R :EXTENDED;
BEGIN
z := fluid[index]^;
CASE z.wand OF
    1 :BEGIN
       R := Rg;
       ev[1] := z.pos[1]/R;
       ev[2] := z.pos[2]/R;
       vev := z.vel[1]*ev[1] + z.vel[2]*ev[2];
       z.vel[1] := z.vel[1] - 2.0 * vev * ev[1];
       z.vel[2] := z.vel[2] - 2.0 * vev * ev[2];
       END;
     END;
```

```
fluid[index]^ := z;
NewPartner(index);
FOR i := 1 TO N DO
   IF fluid[i]^.collpart=index THEN NewPartner(i);
END;

{Berechnet neuen Geschwindigkeitsvektor für die Molekel, die kollidieren}
PROCEDURE ExchangeMomentum(index1,index2:INTEGER);
VAR z1,z2 :tzustand;
    norm,epxv,m1,m2 :EXTENDED;
    cp,i,index :INTEGER;
    ep :tvektor;
BEGIN
z1 := fluid[index1]^;
z2 := fluid[index2]^;
m1 := z1.mass;
m2 := z2.mass;
norm := 0.0;
FOR index := 1 TO dim DO
   norm := norm + SQR(z2.pos[index] - z1.pos[index]);
norm := SQRT(norm);
ep[1] := (z2.pos[1] - z1.pos[1]) / norm;
ep[2] := (z2.pos[2] - z1.pos[2]) / norm;
epxv := 0.0;
FOR index := 1 TO dim DO
   epxv := epxv + ep[index] * (z2.vel[index] - z1.vel[index]);
FOR index := 1 TO dim DO
       BEGIN
   z1.vel[index] := z1.vel[index] + 2.0*m2/(m1+m2) * epxv * ep[index];
   z2.vel[index] := z2.vel[index] - 2.0*m1/(m1+m2) * epxv * ep[index];
       END;
fluid[index1]^ := z1;
fluid[index2]^ := z2;
FOR i := 1 TO N DO
   BEGIN
   cp := fluid[i]^.collpart;
   IF ((cp=index1) OR (cp=index2)) THEN NewPartner(i);
   END;
FindePartner(index1,index2);
END;

{Setzt alle Ortsvektoren an den neuen Platz, nachdem die kürzeste Zeit bis
zu einer Reflexion oder Kollision bestimmt ist}
PROCEDURE MoveSystem(t:EXTENDED);
VAR index :INTEGER;
BEGIN
FOR index := 1 TO N DO
   fluid[index]^.pos[1] := fluid[index]^.pos[1] + t*fluid[index]^.vel[1] ;
FOR index := 1 TO N DO
   fluid[index]^.pos[2] := fluid[index]^.pos[2] + t*fluid[index]^.vel[2];
FOR index := 1 TO N DO
   fluid[index]^.szeit := fluid[index]^.szeit - t;
END;


{Runde: von einem Ereignis zum nächsten}
PROCEDURE durchlauf;
VAR mt,dt :EXTENDED;
    mi,mi2,i  :INTEGER;
BEGIN
mt := pinf;
mi := 0;
FOR i := 1 TO N DO    {Es wird die kürzeste Zeit gesucht bis Ereignis}
   IF mt>fluid[i]^.szeit THEN
    BEGIN
    mt := fluid[i]^.szeit;
    mi := i;
    END;
```

```
dt := fluid[mi]^.szeit;
new(Zeitintervallspeicher[counter]);  {Erstellen Array of Records für
Zwischenereigniszeiten}
Zeitintervallspeicher[counter]^.zeitintervall:= dt;
if fluid[mi]^.collpart=0 then
     Zeitintervallspeicher[counter]^.ereignistyp:=1   {Reflexion}
     else
     Zeitintervallspeicher[counter]^.ereignistyp:=2;  {Kollision}

Time := Time + dt;  {Systemzeit wird weitergestellt}
MoveSystem(dt);
IF fluid[mi]^.collpart=0 THEN CarryOutReflexion(mi)
  ELSE
  BEGIN
  mi2 := fluid[mi]^.collpart;
  ExchangeMomentum(mi,mi2);
  END;
END;

{Stellt Speicher für den Zustand des Gesamtsystems bereit}
PROCEDURE speicher;
VAR i :INTEGER;
BEGIN
FOR i := 1 TO N DO
   NEW(fluid[i])
END;

{Zaehlt die Anzahl der Ereignisse eines bestimmten Typs; wird k = 1 gewaehlt,
so werden die Anzahl der Zwischenkollisionzeiten gezaehlt, bei 2 Reflexionen}
function ZaehleEreignistyp(k:shortint):integer;
var i,z:integer;
begin
z:=0;
for i:=1 to Anzahlrunden do
if Zeitintervallspeicher[i].ereignistyp=k then z:=z+1;
ZaehleEreignistyp:=z
end;


{Berechnung der Zwischenkollisionszeiten aus den
Zwischenreigniszeiten und gibt diese in Extendedarray aus}
procedure TrenneEreignistypen (k:shortint;var SpeicherFuerTyp:extendedvektor);
var i,u:integer;
begin
i:=1;
u:=1;
repeat
    if Zeitintervallspeicher[i].ereignistyp=k then
        begin
        SpeicherFuerTyp[u]:=Zeitintervallspeicher[i].zeitintervall;
        i:=i+1;
        end
      else
        begin
        zWZeit:=0;
          repeat
          ZWZeit:=Zeitintervallspeicher[i].zeitintervall+ZWZeit;
          i:=i+1;
          until (Zeitintervallspeicher[i].ereignistyp=k) or (i=Anzahlrunden);
        if (Zeitintervallspeicher[i].ereignistyp=k)then
          begin
          ZWZeit:=Zeitintervallspeicher[i].zeitintervall+zwzeit;
          SpeicherFuerTyp[u]:=zwzeit;
          end;
        if not (i = Anzahlrunden) then i:=i+1;
        end;
        u:=u+1;
until i=Anzahlrunden;
```

```
if (Zeitintervallspeicher[Anzahlrunden-1].ereignistyp=k) and
   (Zeitintervallspeicher[Anzahlrunden].ereignistyp=k) then  begin
SpeicherFuerTyp[u]:=Zeitintervallspeicher[Anzahlrunden].zeitintervall;
end
end;


{berechnet geschätztes Lamda der Exponentialverteilung}
function lamda(SpeicherFuerTyp:extendedvektor):extended;
var x3:extended;
j:integer;
begin
x3:=0;
for j:=1 to ZaehleEreignistyp(ETypKonst) do begin
x3:=x3+SpeicherFuerTyp[j] end;
lamda:=1/(x3/ZaehleEreignistyp(ETypKonst))
end;

{Berechnet den Wert der Standardnormaldichte an der Stelle x für
den Erwartungswert m und die Standardabweichung Sigma}
function pdfNormal(x,m,sigma:extended):extended;
begin
pdfnormal:=(1/(sigma*sqrt(2*pi)))*exp((-1/(2*sqr(sigma)))*sqr(x-m));
end;

{Berechnet den Wert der Kerndichtefunktion an der Stelle x}
function kerndichte(x:extended):extended;
var j: integer;
x3,x4: extended;
begin
x3:=0;
x4:=((1/lamda(SpeicherFuerTyp))/exp(0.4*ln(ZaehleEreignistyp(ETypKonst))));
                                    (*(2/exp(0.25*ln(zaehleEreignistyp)));*)
for j:=1 to zaehleEreignistyp(ETypKonst) do begin
x3:= pdfNormal(x,SpeicherfuerTyp[j],x4) + x3
end;
kerndichte:=(1/zaehleEreignistyp(ETypKonst))*x3;
end;


{berechnet Minimum der Zwischenereigniszeiten - je nach Typ}
function min(SpeicherFuerTyp:extendedvektor):extended;
var x3:extended;
j:integer;
begin
x3:=SpeicherFuerTyp[1];
for j:=1 to ZaehleEreignistyp(ETypKonst)-1 do begin
  if x3 > SpeicherFuerTyp[j+1]
    then x3:=SpeicherFuerTyp[j+1]
    end;
min:=x3;
end;

{berechnet Maximum der Zwischenereigniszeiten - je nach Typ}
function max(SpeicherFuerTyp:extendedvektor):extended;
var x3:extended;
j:integer;
begin
x3:=SpeicherFuerTyp[1];
for j:=1 to zaehleEreignistyp(ETypKonst)-1 do begin
  if x3 < SpeicherFuerTyp[j+1]
    then x3:=SpeicherFuerTyp[j+1]
    end;
max:=x3;
end;

{Berechnet Schrittweite für Beschriftung der Achsen}
function SchrittweiteBeschr(x:extended):extended;
```

```
var int:integer;
xx:real;
begin
xx:=x;
if (x >= 1.0) and (x < 10.0) then int:=trunc(x)
 else if (x >= 10.0) then
    begin
      repeat
      x:= x/10;
      until (x >= 1.0) and (x < 10.0);
      int:=trunc(x);
      end

    else
    x:=xx;
      if (x < 1.0)
        then
        repeat
        x:=x*10;
        until (x >=1.0) and (x < 10.0);
        int:=trunc(x);
if int = 1 then
Schrittweitebeschr:=0.2;
if int in [2..4] then
Schrittweitebeschr:=0.5;
if int in [5..9] then
Schrittweitebeschr:=1;
end;

{Berechnet die Anzahl der Hochstellen samt Vorzeichen - Ziel Beschriftung}
function HochzahlBeschriftung (x:extended):integer;
var xx:extended;
j:integer;
begin
j:=0;
xx:=x;
if (x >= 1.0) and (x < 10.0) then HochzahlBeschriftung:=0
 else if (x >= 10.0) then
    begin
      repeat
      x:= x/10;
      j:=j+1;
      until (x >= 1.0) and (x < 10.0);
      HochzahlBeschriftung:=j
      end

    else
    x:=xx;
      if (x < 1.0)
        then
        begin
        repeat
        x:=x*10;
        j:=j+1;
        until (x >=1.0) and (x < 10.0);
        HochzahlBeschriftung:=-j
        end;
end;

{Prozeduren und Routinen für die Berechnung des Permutatonstestes}

{Ordnet n x m Matrix nach einer Spalte i in aufsteigender Ordnung}
function ordnematrix(b:realmatrix;j,n,m:integer):realmatrix;
var b1: realmatrix;
d:extendedvektor;
f:boolean;
i,k:integer;
begin
```

```
setlength(d,m);
setlength(b1,n,m);
b1:=copy(b);
repeat
f:=false;
for i:=0 to n-2 do begin
if b1[i,j]>b1[i+1,j]then
  begin
  for k:=0 to m-1 do begin
  d[k]:=b1[i,k];
  b1[i,k]:=b1[i+1,k];
  b1[i+1,k]:=d[k];
  f:=true
  end;
  end;
  end;
until f=false;
ordnematrix:=b1;
end;

{.....................................................}
{Berechnet Fakultät für Zahlen < = 12}
function fakultaet(n:integer):integer;
var j,i: integer;
  begin
  if n = 0 then fakultaet:= 1 else
  begin
  j:=n;
  for i:=n downto 3 do begin
  j:=(i-1)*j;
  end;
  fakultaet:=j
  end;
end;

{.....................................................}
{Berechnet Fakultäten für Zahlen grösser als 12: Ausgabe extended}
function fakultaetreal(n:integer):extended;
var x: extended;
i:integer;
  begin
  if n = 0 then fakultaetreal:=1 else
  begin
  x:=n/1;
  for i:=n downto 3 do begin
  x:=(i-1)*x;
  end;
  fakultaetreal:=x
  end;
end;

{.....................................................}
{Funktion fürs Potenzieren einer reellen Zahl mit einer pos. ganzen Zahl}
function hoch(x:real; z:integer):real;
var y:real;
  i:integer;
begin
if (z = 0) and (not(x = 0)) then hoch:=1 else
begin
if x=0 then hoch:=0 else
begin
y:=x;
i:=1;
while i < z do begin
 x:=x*y;
 i:=i+1;
 end;
hoch:=x
```

```
end;
end;
end;

{.....................................................}
{vergleicht die Ordnungszahlen einer Stringreihe mit einen Ganzzahlenvektor;
ist die Reihenfolge der Ordnungszahlen mit der Reihenfolge der der
Zahlen identisch, so erfolgt die Ausgabe wahr, sonst die Ausgabe falsch}
function vergleiche (var sult:str; c:integervektor; max:integer):boolean;
var i,m:integer;
c1:integervektor;
begin
setlength(c1,max);
     for i:=0 to max-1 do begin
     c1[i]:=ord(sult[i+1]);
     end;

m:=0;
 {Vergleich der Vektoren}
for i:= 0 to max-1 do begin
if c[i]=c1[i] then m:=m+1
end;
if m = max then
vergleiche:=true
else vergleiche:=false
end;

{.....................................................................}
{Verschiebt die Zahlen in einer Zahlenliste ganzer positiver Zahlen um eins
 nach vorne. Die erste Zahl wird an die letzte Stelle verschoben.}

PROCEDURE rotiere (VAR restliste : str; teil: integer);
  VAR i : integer;
      c : char;
BEGIN
  c := restliste [1];
  FOR i := 1 TO teil DO
    restliste [i] := restliste [i + 1];
  restliste [teil] := c;
  END;

{.....................................................................}
{Berechnet alle Permuationen und ordnet der Permutation, die mit dem Integer
vektor (Ordnungsvektor des Teildatenvektors) identisch ist, die Ordnungszahl zu}
procedure permutiere (var eingabe : str; c:integervektor; zahl: integer; var
inter:integer);
  VAR i: integer;
 BEGIN
  IF (zahl = 1) THEN
  BEGIN
   permutationszaehler:=permutationszaehler+1;
   if vergleiche(eingabe,c,t)=true then inter:=permutationszaehler
   end
    ELSE
  FOR i := 1 TO zahl DO
    BEGIN
      rotiere (eingabe,zahl);
      permutiere (eingabe,c,zahl - 1,inter);
     END;
END;



{Berechnung auf "Berechnen"-Knopf}
procedure TForm1.Button1Click(Sender: TObject);
var y,BreiteEinheitDaten,sprungweite,lam:extended; {Sprungweit für Skalieren
x-Achse)}
```

```
i,j,breite,add,anzahl,BreiteEinheitGraphik,breitetestgraphik:integer;
s:String;
konstanten: extendedvektor;
Testwert:extended;
ss:string;


begin

{Spezifikationen}
if temp1='' then temp:=300 else temp:=strtofloat(temp1);
if AnzahlRunden1='' then AnzahlRunden:=1000 else
AnzahlRunden:=strtoint(AnzahlRunden1);


{Durchführen des Experimentes und Produktion der Daten}
RANDOMIZE;
  speicher;
  init;
  setlength(zeitintervallspeicher,AnzahlRunden+1);
  counter := 0;
  REPEAT   {Experiment wird AnzahlRunden-mal durchgeführt}
    INC(counter);
    durchlauf;
  UNTIL counter = AnzahlRunden;

  FOR ii := 1 TO N DO
   DISPOSE(fluid[ii]);


setlength(SpeicherFuerTyp,ZaehleEreignistyp(ETypKonst)+1);
TrenneEreignistypen(ETypKonst,SpeicherFuerTyp);
lam:=lamda(SpeicherFuerTyp);
Sprungweite:=max(SpeicherFuerTyp)/400;
refresh;


{Zeichnen der parametrisch geschätzten Exponentialverteilung und der Achsen}

y:=0;
form1.Canvas.MoveTo(eckex,eckey);
for j:=0 to 400 do begin   {400 für Breite der Zeichnung in Pixel}
form1.Canvas.LineTo(eckex+j,eckey-trunc(200*exp(-lam*y)));{Höhe Zeichnung 200
Pixel}
y:=y+Sprungweite;
end;
form1.Canvas.MoveTo(eckex,eckey);
form1.canvas.lineto(eckex+400,eckey);

{Zeichnen des Kerndichteschätzers}
y:=0;
form1.Canvas.MoveTo(eckex,eckey-trunc(200/lam*kerndichte(0)));
for j:=1 to 400 do begin
form1.canvas.pen.color:=clRed;
form1.Canvas.LineTo(eckex+j,eckey-(trunc(200/lam*kerndichte(y))));
y:=y+Sprungweite;
end;
form1.canvas.pen.color:=clblack;

{Legende}
form1.Canvas.MoveTo(eckex,eckey+40);
form1.canvas.pen.color:=clRed;
form1.canvas.lineto(eckex+40,eckey+40);
form1.canvas.pen.color:=clblack;
form1.canvas.textout(eckex+40,eckey+34,'  Kerndichteschätzer');
form1.Canvas.TextOut(eckex+40,eckey+52,'  Parametrisch geschätzte Dichtefunktion
der Exponentialverteilung');
form1.Canvas.MoveTo(eckex,eckey+60);
```

```
form1.canvas.pen.color:=clblack;
form1.canvas.lineto(eckex+40,eckey+60);



{x-Koordatenunterteilung und -beschriftung}
y:=max(SpeicherFuerTyp);
BreiteEinheitDaten:= SchrittWeitebeschr(y)*exp(hochzahlbeschriftung(y) *ln(10));
{Breite zwischen Beschriftungen - Einheit Daten}
anzahl:=trunc(y/BreiteEinheitDaten); {Anzahl Beschriftungen}
BreiteEinheitGraphik:=trunc(400/y*BreiteEinheitDaten);
{Breite zwischen Beschriftungen in Pixels}
{Zeichnung und Beschriftung}
add:=BreiteEinheitGraphik + eckex;
y:=BreiteEinheitDaten;
for j:=1 to anzahl do begin
  form1.Canvas.MoveTo(add,eckey+5);
  form1.Canvas.LineTo(add,eckey);
  s:=floattostrf(y,ffExponent,2,1);
  form1.Canvas.textout(add-15,eckey+8,s);
  add:=add+BreiteEinheitGraphik;
  y:=y+BreiteEinheitDaten;
  end;
form1.Canvas.TextOut(eckex+410,eckey+7,'[s]');


{y-Koordatenbeschriftung}
y:=lamda(SpeicherFuerTyp);
BreiteEinheitDaten:=SchrittWeitebeschr(y)*exp(hochzahlbeschriftung(y)*ln(10));
anzahl:=trunc(y/BreiteEinheitDaten); {Anzahl Beschriftungen}
BreiteEinheitGraphik:=trunc(200/y*BreiteEinheitDaten);
add:=-BreiteEinheitGraphik + eckey;
y:=BreiteEinheitDaten;
for j:=1 to anzahl do begin
  form1.Canvas.MoveTo(eckex-5,add);
  form1.Canvas.LineTo(eckex,add);
  ss:=floattostrf(y,ffExponent,2,1);
  form1.Canvas.textout(eckex-45,add-7,ss);
  add:=add-BreiteEinheitGraphik;
  y:=y+BreiteEinheitDaten;
  end;

{Angabe von Lamda auf der Graphik}
y:=lamda(SpeicherFuerTyp);
form1.Canvas.MoveTo(eckex-5,eckey-200);
  form1.Canvas.LineTo(eckex,eckey-200);
  ss:=floattostrf(y,ffExponent,10,1);
  form1.Canvas.textout(eckex-35,eckey-7-200,'[1/s]');
  form1.Canvas.textout(eckex+5,eckey-7-200,ss);
  form1.canvas.textout(eckex+90,eckey-7-200,'(= geschätztes Lamda)');



{Berechnung und graphische Darstellung der Permutationstest}


BreiteTestgraphik:=0;  {Initialisierung für Ausgabe Quantile Graphik}
setlength(konstanten,17);
{Grenzen des Annahmebereichs; Mit Excel Microsoft 2002 Berechnet,
auf drei Stellen gerundet}
konstanten[1]:= 0.001; {uG: 2!=2; 0.025-Quantil;
Chi2Verteilung}
konstanten[9]:= 5.024; {oG: 2!=2; 0.975-Quantil;
Chi2Verteilung}
konstanten[2]:= 0.831; {uG: 3!=6; 0.025-Quantil; Chi2verteilung mit 6-1 df}
konstanten[10]:= 12.832; {oG: 3!=6; 0.975-Qantil Chi2verteilung mit 6-1 df}
```

```
konstanten[3]:= 11.689; {uG: 4!=24; 0.025-Qantil Chi2verteilung mit 24-1 df}
konstanten[11]:= 38.076;{oG: 4!=24; 0.975-Qantil Chi2verteilung mit 24-1 df}
konstanten[4]:= 90.7;{uG: 5!=125; 0.025-Qantil Chi2verteilung mit 125-1 df}
konstanten[12]:= 151.084;{oG: 5!=125; 0.975-Qantil Chi2verteilung mit 125-1 df}
konstanten[5]:= 646.588;{uG: 6!=125; 0.025-Qantil Chi2verteilung mit 125-1 df}
konstanten[13]:= 795.2;{oG: 6!=125; 0.975-Qantil Chi2verteilung mit 125-1 df}
konstanten[6]:= 4842.241; {uG: 7!=125; 0.025-Qantil Chi2verteilung mit 125-1 df;
Näherung Normalverteilng}
konstanten[14]:= 5235.759; {oG: 7!=125; 0.975-Qantil Chi2verteilung mit 125-1
df;Näherung Normalverteilng}
konstanten[7]:= 39762.433; {uG: 8!=125; 0.025-Qantil Chi2verteilung mit 8!-1
df;Näherung Normalverteilng}
konstanten[15]:= 40875.567; {oG: 8!=125; 0.975-Qantil Chi2verteilung mit 8!-1
df;Näherung Normalverteilng}
konstanten[8]:= 361209.28; {uG: 9!=125; 0.025-Qantil Chi2verteilung mit 9!-1
df;Näherung Normalverteilng}
konstanten[16]:= 364548.72;  {oG: 9!=125; 0.975-Qantil Chi2verteilung mit 9!-1
df;Näherung Normalverteilng}


{Zeichnen und Beschriftung der Darstellung des Annahmebereichs}
{Zeichnen des grossen Recktecks}
with form1.canvas do begin
  moveto(abstandx,abstandy);
  lineto(abstandx+455,abstandy);
  MoveTo(abstandx,abstandy+50);
  lineto(abstandx+455,abstandy+50);
  MoveTo(abstandx,abstandy-30);
  lineto(abstandx,abstandy+80);
  breite:=57;
  {Zeichnen der vertikalen Linien}
  for i:=1 to 7 do begin
    moveto(abstandx+breite,abstandy-30);
    lineto(abstandx+breite,abstandy+80);
    breite:=breite+57
    end;
  {Beschriften der Segmente obere Reihe}
  breite:=0;
  for i:=9 to 16 do begin
    ss:=floattostr(konstanten[i]);
    textout(abstandx+breite+6, abstandy - 16,ss);
    ss:=floattostr(i+1-8);
    textout(abstandx+breite+6,abstandy - 29, ss);
    ss:='!';
    textout(abstandx+breite+13,abstandy - 29, ss);
    breite:=breite +57;
     end;
  {Beschriftung der Zeilen}
  font.Size:=14;
  textout(abstandx,abstandy-80,'Ergebnisse der Permutationstests auf
Unabhängigkeit  - für k >= 5t!');
  textout(eckex-45,eckey-235,'Schätzungen der Dichtefunktion');
  textout(24,abstandy-80,'Anzahl Ereignisse');
  font.Size:=8;
  textout(abstandx,abstandy-50,'k = Anzahl Vektoren der Länge t, t! = Anzahl der
Klassen');
  textout(abstandx+breite+8, abstandy - 29,'Anzahl Klassen');
  textout(abstandx+breite+8, abstandy - 16,'0.975-Quantile*');
  textout(abstandx+breite+8, abstandy +18,'Annahmebereich');
  textout(abstandx+breite+8, abstandy +50+3,'0.025-Quantile');
  textout(abstandx+breite+8, abstandy +50+15,'numerischer Wert');
  textout(abstandx+breite+8, abstandy +50+27,'der Teststatistik');
  ellipse(abstandx-3, abstandy+50+45,abstandx+3,abstandy+50+51);
  textout(abstandx+5, abstandy+50+40,' = graphische Darstellung des Wertes der
Teststatistik');
  textout(abstandx-4, abstandy+50+55,'*Berechnet mit Excel Microsoft 2002 mit
"chiinv" für 2! ... 6!, mit "norminv" für 7! .. 9!, gerundet auf drei Stellen');
```

```
  {Beschriften der Segmente untere Reihe}
  breite:=0;
  for i:=1 to 8 do begin
    ss:=floattostr(konstanten[i]);
    textout(abstandx+breite+6, abstandy +50+ 3,ss);
    breite:=breite +57;
     end;

  end;


  {Schlaufe für verschieden lange Teilvektoren t}
  t:=2;
  kk:=zaehleEreignistyp(ETypKonst)div t;
  while (kk >= 5*fakultaet(t)) do begin
   {Bestimmen von k in Abhängigkeit von t}
   kk:=zaehleEreignistyp(ETypKonst)div t;

  {Schreibe Liste der Länge t - für die Erstellung der Permutationen}
   alpha:='  '; {= Zeichen mit Ordnungszahl 1; damit sind bezüglich dieser
   Festlegung Permutationen der Länge 255 möglich}
   liste[1]:=alpha;
   for i:=2 to t do begin
   liste[i]:=succ(alpha);
   alpha:=succ(alpha);
   end;


  {Initialisierung des Vektors, der die identischen Permuationen zählt}
   setlength(a,fakultaet(t),2);
   for i:=0 to fakultaet(t)-1 do begin
   a[i,0]:=i+1;
   a[i,1]:=0
   end;

{Berechnung der Ordnungszahl der Permutationen,
  Zählen der Anzahl Vektoren mit spezifischer Permutation}
   zaehler:=1;
   permutationszaehler:=0;
   while zaehler < kk+1 do begin
    {Lese Daten in erste Spalte der tX2-Matrix und Durchnummerieren der
    Daten in der zweiten Spalte}
    setlength(b,t,2);
    for i:=0 to t-1 do begin
    b[i,0]:=SpeicherFuerTyp[(zaehler-1)*t+i+1];   {Matrix mit Daten}
    b[i,1]:=i+1  {Durchnummeriert in der zweiten Spalte}
    end;

    {Ordnen der Zeilen der ersten Spalte nach - d.h. den eingelesenen Daten
nach}
    b1:=ordnematrix(b,0,t,2);

    {Erstelle Integervektor für die zweite Spalte von b1}
    setlength(c,t);
    for i:=0 to t-1 do begin
    c[i]:=round(b1[i,1])
    end;

    {Berechne der einer Permutation zugeordneten Ordnungszahl und Setzen des
    Zählers der Permuationen}

    for i:=1 to t do begin
    liste1[i]:=liste[i] end;

    permutiere(liste1,c,t,inter);
    a[inter-1,1]:=a[inter-1,1]+1;
    permutationszaehler:=0;
```

```
    zaehler:=zaehler+1;

  end; {Ende der while-Schlaufe}

{Berechnung und Ausgabe der Werte der Chi2verteilten Teststatistik}
      testwert:=0;
      for i:=0 to fakultaet(t)-1 do begin
      testwert:=sqr(a[i,1]-(kk/fakultaet(t)))/(kk/fakultaet(t))+testwert;
      end;
      ss:=floattostrf(testwert,ffgeneral,6,3);

testwert:=((testwert-konstanten[t-1])*50/(konstanten[t+7]-konstanten[t-1]));

form1.canvas.ellipse(abstandx-3+breitetestgraphik,Abstandy+50-trunc(testwert)-3,
abstandx+3+breitetestgraphik,Abstandy+50-trunc(testwert)+3);
      form1.canvas.TextOut(abstandx+6+breitetestgraphik, abstandy+65,ss);
      BreiteTestGraphik:=BreiteTestGraphik+57;


   t:=t+1;

end; {Ende der while-Schlauf}

{Ausgabe Anzahlen Ereignistypen}
with form1.canvas do  begin
 moveto(80,abstandy-30);
 lineto(80,abstandy+80);
 moveto(180,abstandy-30);
 lineto(180,abstandy+80);
 moveto(130,abstandy-30);
 lineto(130,abstandy+80);
 moveto(24,abstandy);
 lineto(180,abstandy);
 moveto(24,abstandy+40);
 lineto(180,abstandy+40);
 moveto(24,abstandy+80);
 lineto(180,abstandy+80);
 textout(24,abstandy+15,'Kollisionen');
 textout(24,abstandy+55,'Reflexionen');
 textout(90,abstandy-15,'Anzahl');
 textout(90+50,abstandy-15,'Anteil');
 ss:=floattostr(ZaehleEreignistyp(ETypKonst));
 textout(90,abstandy+15,ss);
 ss:=floattostr(ZaehleEreignistyp(ETypKonst)/AnzahlRunden);
 textout(90+50,abstandy+15,ss);
 ss:=floattostr(AnzahlRunden-ZaehleEreignistyp(ETypKonst));
 textout(90,abstandy+55,ss);
 ss:=floattostr((AnzahlRunden-ZaehleEreignistyp(ETypKonst))/AnzahlRunden);
 textout(90+50,abstandy+55,ss);

{Ausgabe der Daten in Datei out.txt}
for i:=1 to  ZaehleEreignistyp(ETypKonst) do begin
  writeln(SpeicherFuerTyp[i]) end;

end;

end; {TForm1.Button1Click Ende}



{Spezifikationen}
procedure TForm1.Edit1Change(Sender: TObject);
begin
temp1:=edit1.text;
end;

procedure TForm1.Edit2Change(Sender: TObject);
begin
```

```
AnzahlRunden1:=edit2.text;
end;

{.....................................................................}
{Programmbeginn}
begin   {Das Programm selber}
 {Festlegung der Ausgabedatei}
  assignfile(output, 'out.txt');
  rewrite(output);
  {$R *.dfm}

{Programmende}
end.
```

```
unit permutationstest; {Tests für immer längere Datenvektoren}

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;

  const abstandx=50;
  abstandy=150;

type
  str = STRING [80];
  integervektor = array of integer;
  integermatrix=array of integervektor;
  einlesereallliste=^realliste;
  extendedvektor=array of extended;
  realmatrix=array of extendedvektor;
    realliste=record
      next:einlesereallliste;
      dat:real
      end;
  TForm1 = class(TForm)
    Button1: TButton;
    Edit1: TEdit;
    Label1: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure Edit1Change(Sender: TObject);
    end;

var
  Form1: TForm1;

i,j,k,int,t,zaehler,permutationszaehler,n,Gesamtanzahl,Anzahldaten,vektorlaenge1
:integer;
  {int (globale Variable) reserviert für Ausgabe Ordnungszahl der Permutationen;
  permutationszaehler (globale Variable) reserviert fürs Zählen der Permuationen
  in der Permutationsprozedur;
  t globale Variable reserviert für Teilvektorlänge;
  k globale Variable reserviert für Anzahl Teilvektoren;}
  vektorlaenge,liste,liste1 : str;
  alpha :char;
  w:single;
  x,x1:extended;
  schreibe:boolean=false;
  l1,o1:einlesereallliste;
  a: integermatrix;
  b,b1: realmatrix;
  c: Integervektor;
  SpeicherFuerTyp:extendedvektor;


  PROCEDURE rotiere (VAR restliste : str; teil : integer);
  PROCEDURE permutiere (var eingabe : str; c:integervektor; zahl: integer; var
int:integer);
  function ordnematrix(b:realmatrix;j,n,m:integer):realmatrix;
  function fakultaet(n:integer):integer;
  function fakultaetreal(n:integer):extended;
  function vergleiche (var sult:str; c:integervektor; max:integer):boolean;




implementation
{$R *.dfm}
{.....................................................}
{Ordnet n x m Matrix nach einer Spalte i in aufsteigender Ordnung}
function ordnematrix(b:realmatrix;j,n,m:integer):realmatrix;
```

```
var b1: realmatrix;
d:extendedvektor;
f:boolean;
i,k:integer;
begin
setlength(d,m);
setlength(b1,n,m);
b1:=copy(b);
repeat
f:=false;
for i:=0 to n-2 do begin
if b1[i,j]>b1[i+1,j]then
  begin
  for k:=0 to m-1 do begin
  d[k]:=b1[i,k];
  b1[i,k]:=b1[i+1,k];
  b1[i+1,k]:=d[k];
  f:=true
  end;
  end;
  end;
until f=false;
ordnematrix:=b1;
end;

{.....................................................}
{Berechnet Fakultät für Zahlen < = 12}
function fakultaet(n:integer):integer;
var j,i: integer;
  begin
  if n = 0 then fakultaet:= 1 else
  begin
  j:=n;
  for i:=n downto 3 do begin
  j:=(i-1)*j;
  {writeln(j);}
  end;
  fakultaet:=j
  end;
end;

{.....................................................}
{Berechnet Fakultäten für Zahlen grösser als 12: Ausgabe extended}
function fakultaetreal(n:integer):extended;
var x: extended;
i:integer;
  begin
  if n = 0 then fakultaetreal:=1 else
  begin
  x:=n/1;
  for i:=n downto 3 do begin
  x:=(i-1)*x;
  {writeln(j);}
  end;
  fakultaetreal:=x
  end;
end;

{.....................................................}
{Funktion fürs Potenzieren einer reellen Zahl mit einer pos. ganzen Zahl}
function hoch(x:real; z:integer):real;
var y:real;
  i:integer;
begin
if (z = 0) and (not(x = 0)) then hoch:=1 else
begin
if x=0 then hoch:=0 else
begin
```

```
y:=x;
i:=1;
while i < z do begin
 x:=x*y;
 i:=i+1;
 end;
hoch:=x
end;
end;
end;

{.....................................................}
{vergleicht die Ordnungszahlen einer Stringreihe mit einen Ganzzahlenvektor;
ist die Reihenfolge der Ordnungszahlen mit der Reihenfolge der der
Zahlen identisch, so erfolgt die Ausgabe wahr, sonst die Ausgabe falsch}
function vergleiche (var sult:str; c:integervektor; max:integer):boolean;
var i,m:integer;
c1:integervektor;
begin
setlength(c1,max);
     for i:=0 to max-1 do begin
     c1[i]:=ord(sult[i+1]);
     end;

m:=0;
 {Vergleich der Vektoren}
for i:= 0 to max-1 do begin
if c[i]=c1[i] then m:=m+1
end;
if m = max then
vergleiche:=true
else vergleiche:=false
end;

{.....................................................................................}
{Verschiebt die Zahlen in einer Zahlenliste ganzer positiver Zahlen um eins
 nach vorne. Die erste Zahl wird an die letzte Stelle verschoben.}

PROCEDURE rotiere (VAR restliste : str; teil: integer);
  VAR i : integer;
      c : char;
BEGIN
  c := restliste [1];
  FOR i := 1 TO teil DO
    restliste [i] := restliste [i + 1];
  restliste [teil] := c;
  END;

{.....................................................................................}
{Berechnet alle Permuationen und ordnet der Permutation, die mit dem Integer
vektor (Ordnungsvektor des Teildatenvektors) identisch ist, die Ordnungszahl zu}
procedure permutiere (var eingabe : str; c:integervektor; zahl: integer; var
int:integer);
  VAR i: integer;
 BEGIN
  IF (zahl = 1) THEN
  BEGIN
   permutationszaehler:=permutationszaehler+1;
   if vergleiche(eingabe,c,t)=true then int:=permutationszaehler
   end
    ELSE
  FOR i := 1 TO zahl DO
    BEGIN
      rotiere (eingabe,zahl);
      permutiere (eingabe,c,zahl - 1,int);
     END;

END;
```

```
{.....................................................................}

{.....................................................................}
{Bestimmt Minumum einer Menge von 2 Elementen - kleiner gleich}
function min(x,y:extended):extended;
begin
if x < y then min:=x else min:=y;
end;

{.....................................................................}
{Bestimmt den Wert der Gammafunktion für ganz- und halbzahlige Argumente}
function gamma(x:extended):extended;
var y:extended;
i:integer;
begin
if x=trunc(x) then begin
gamma:=fakultaetreal(trunc(x)-1);
end
else
begin
y:=sqrt(pi);
for i:=1 to trunc(x) do begin
y:=(((2*i)-1)/2)*y;
end;
gamma:=y;
end;
end;

{.....................................................................}


procedure TForm1.Edit1Change(Sender: TObject);
begin
vektorlaenge:=edit1.text;
end;

{.....................................................................}
{onClick Prozedur für Ausgabe der Permutationstestresultaten auf Form1}
procedure TForm1.Button1Click(Sender: TObject);
var i,breite,BreiteTestgraphik,AusserhalbAB: integer;
konstanten: extendedvektor;
Testwert,Testwert1:extended;
s:string;

begin
refresh;

if vektorlaenge ='' then t:=2 else t:=(trunc(strtofloat(vektorlaenge)));


BreiteTestgraphik:=0;  {Initialisierung für Ausgabe Testwerte Graphik}


setlength(konstanten,17);
{Grenzen des Annahmebereichs; Mit Excel Microsoft 2002 Berechnet,
auf drei Stellen gerundet}
konstanten[1]:= 0.001; {uG: 2!=2; 0.025-Quantil;
Chi2Verteilung}
konstanten[9]:= 5.024; {oG: 2!=2; 0.975-Quantil;
Chi2Verteilung}
konstanten[2]:= 0.831; {uG: 3!=6; 0.025-Quantil; Chi2verteilung mit 6-1 df}
konstanten[10]:= 12.832; {oG: 3!=6; 0.975-Qantil Chi2verteilung mit 6-1 df}
konstanten[3]:= 11.689; {uG: 4!=24; 0.025-Qantil Chi2verteilung mit 24-1 df}
konstanten[11]:= 38.076;{oG: 4!=24; 0.975-Qantil Chi2verteilung mit 24-1 df}
konstanten[4]:= 90.7;{uG: 5!=125; 0.025-Qantil Chi2verteilung mit 125-1 df}
konstanten[12]:= 151.084;{oG: 5!=125; 0.975-Qantil Chi2verteilung mit 125-1 df}
konstanten[5]:= 646.588;{uG: 6!=125; 0.025-Qantil Chi2verteilung mit 125-1 df}
```

```
konstanten[13]:= 795.2;{oG: 6!=125; 0.975-Qantil Chi2verteilung mit 125-1 df}
konstanten[6]:= 4842.241; {uG: 7!=125; 0.025-Qantil Chi2verteilung mit 125-1 df;
Näherung Normalverteilng}
konstanten[14]:= 5235.759; {oG: 7!=125; 0.975-Qantil Chi2verteilung mit 125-1
df;Näherung Normalverteilng}
konstanten[7]:= 39762.433; {uG: 8!=125; 0.025-Qantil Chi2verteilung mit 8!-1
df;Näherung Normalverteilng}
konstanten[15]:= 40875.567; {oG: 8!=125; 0.975-Qantil Chi2verteilung mit 8!-1
df;Näherung Normalverteilng}
konstanten[8]:= 361209.28; {uG: 9!=125; 0.025-Qantil Chi2verteilung mit 9!-1
df;Näherung Normalverteilng}
konstanten[16]:= 364548.72;  {oG: 9!=125; 0.975-Qantil Chi2verteilung mit 9!-1
df;Näherung Normalverteilng}


{Zeichnen und Beschriftung der Darstellung des Annahmebereichs}
{Zeichnen des grossen Recktecks}
with form1.canvas do begin
  moveto(abstandx,abstandy);
  lineto(abstandx+455,abstandy);
  MoveTo(abstandx,abstandy+50);
  lineto(abstandx+455,abstandy+50);
  MoveTo(abstandx,abstandy-30);
  lineto(abstandx,abstandy+80);
  moveto(abstandx,abstandy+80);
  lineto(abstandx+455,abstandy+80);
  textout(abstandx-15,abstandy+90,'1000');
  moveto(abstandx+99,abstandy+80);
  lineto(abstandx+99,abstandy+85);
  textout(abstandx-20+99,abstandy+87,'25 000');
  moveto(abstandx+199,abstandy+80);
  lineto(abstandx+199,abstandy+85);
  textout(abstandx-20+199,abstandy+87,'50 000');
  moveto(abstandx+299,abstandy+80);
  lineto(abstandx+299,abstandy+85);
  textout(abstandx-20+299,abstandy+87,'75 000');
  moveto(abstandx+399,abstandy+80);
  lineto(abstandx+399,abstandy+85);
  textout(abstandx-20+399,abstandy+87,'100 000');
  {Angebe Grenze für k >=5t!}
  moveto(abstandx + (5*fakultaet(t) div 250)-1,abstandy-30);
  pen.Color:=clRed;
  lineto(abstandx+ (5*fakultaet(t) div 250)-1,abstandy+80);
  pen.Color:=clblack;



  {Beschriften der Segmente obere Reihe}
    s:=floattostr(konstanten[t+7]);
    textout(abstandx+450+6, abstandy - 8,s);
    s:=floattostr(t);
    (*textout(abstandx+450+6,abstandy - 29, s);
    s:='!';
    textout(abstandx+450+13,abstandy - 29, s);*)
    textout(abstandx-15,abstandy-45,'Testwerte');


  {Beschriftung der Zeilen}
  font.Size:=14;
  textout(abstandx,abstandy-80,'Permutationstests für Teildatenvektoren x
unterschiedlicher Länge n');
  font.Size:=8;
  (*textout(abstandx,abstandy-50,'k = (n - (n mod t)) / t (= Anzahl Teilvektoren
der Länge t); t! = Anzahl der Klassen');*)
  (*textout(abstandx+500+8, abstandy - 29,'Anzahl Klassen');*)
  textout(abstandx+500+8, abstandy - 8,'0.975-Quantil*');
  textout(abstandx+457,abstandy+72,'n');
  textout(abstandx+500+8, abstandy +18,'Annahmebereich');
```

```
  textout(abstandx+500+8, abstandy +45,'0.025-Quantil*');
  textout(abstandx, abstandy +110,'Anzahl berücksichtigte Daten n (Start mit
1000 Daten; Datensatz x wird jeweils um 250 Daten erweitert)');
  textout(abstandx,abstandy+130,'Rechts der roten Linie gilt k>5*t! (k = (n - (n
mod t))/t, n = Anzahl Komponenten Gesamtdatenvektor)');
  textout(abstandx,abstandy+145,'*Berechnet mit Excel Microsoft 2002 mit CHIINV
für t! = 2!,...,6! und mit NORMVINV für 7!,...,9!, gerundet auf drei Stellen');


  {Beschriften der Segmente untere Reihe}
    s:=floattostr(konstanten[t-1]);
    textout(abstandx+450+6, abstandy +45,s);


  end;



  {Programmbeginn}
  {Festlegung der Ausgabedatei}
  (*assignfile(output, 'out.txt');
  rewrite(output);*)


  {Festlegen der Eingabedatei}
  assignfile(input, 'in.txt');
  reset (input);

  {Einlesen in dynamische Datenstruktur und Zählen der Daten}
     l1:=nil;
     i:=0;
       while not eof do
             begin
             new(o1);
             read(o1^.dat);
             o1^.next:=l1;
             l1:=o1;
             i:=i+1;
             end;
       Gesamtanzahl:=i-1;
       setlength(SpeicherFuerTyp,i+1);
       o1:=o1^.next;
       for j:=i-1 downto 1 do begin
       SpeicherFuerTyp[j]:=o1^.dat;
       (*writeln(SpeicherFuerTyp[j]);*)
       o1:=o1^.next
       end;
       dispose(o1);
       dispose(l1);
             {Schlaufe für verschieden lange TeilDatenvektoren}
  Anzahldaten:=1000;
  AusserhalbAB:=0;
  while anzahldaten < Gesamtanzahl do begin

  {Bestimmen von k in Abhängigkeit von t}
   k:=anzahldaten div t;
   {Schreibe Liste der Länge t - für die Erstellung der Permutationen}
   alpha:='  '; {= Zeichen mit Ordnungszahl 1; damit sind bezüglich dieser
   Festlegung Permutationen der Länge 255 möglich}
   liste[1]:=alpha;
   for i:=2 to t do begin
   liste[i]:=succ(alpha);
   alpha:=succ(alpha);
   end;


  {Initialisierung des Vektors, der die identischen Permuationen zählt}
   setlength(a,fakultaet(t),2);
```

```
   for i:=0 to fakultaet(t)-1 do begin
   a[i,0]:=i+1;
   a[i,1]:=0
   end;

{Berechnung der Ordnungszahl der Permutationen,
  Zählen der Anzahl Vektoren mit spezifischer Permutation}
   zaehler:=1;
   permutationszaehler:=0;
   while zaehler < k+1 do begin
    {Lese Daten in Vektor der Länge t und Durchnummerieren der
    Daten im Vektor}
    setlength(b,t,2);
    for i:=0 to t-1 do begin
    b[i,0]:=SpeicherFuerTyp[(zaehler-1)*t+i+1];   {Matrix mit Daten}
    b[i,1]:=i+1  {Durchnummeriert in der zweiten Spalte}
    end;

    {Ordnen der Zeilen der ersten Spalte nach - d.h. den eingelesenen Daten
nach}
    b1:=ordnematrix(b,0,t,2);

    {Erstelle Integervektor für die zweite Spalte von b1}
    setlength(c,t);
    for i:=0 to t-1 do begin
    c[i]:=round(b1[i,1])
    end;

    {Berechne der einer Permutation zugeordneten Ordnungszahl und Setzen des
    Zählers der Permuationen}

    for i:=1 to t do begin
    liste1[i]:=liste[i] end;

    permutiere(liste1,c,t,int);
    a[int-1,1]:=a[int-1,1]+1;
    permutationszaehler:=0;
    zaehler:=zaehler+1;

  end; {Ende der while-Schlaufe}

{Berechnung und Ausgabe der Werte der Chi2verteilten Teststatistik}

     testwert:=0;
      for i:=0 to fakultaet(t)-1 do begin
      testwert:=sqr(a[i,1]-(k/fakultaet(t)))/(k/fakultaet(t))+testwert;
      testwert1:=testwert;
      end;
      s:=floattostrf(testwert,ffgeneral,6,3);

testwert:=((testwert-konstanten[t-1])*50/(konstanten[t+7]-konstanten[t-1]));
      if Anzahldaten=1000 then
form1.canvas.moveto(abstandx,abstandy+50-trunc(testwert))
      else form1.Canvas.lineto(abstandx+(Anzahldaten div 250),
Abstandy+50-trunc(testwert));

      if (testwert1 < konstanten[t-1]) or  (testwert1 > konstanten[t+7]) then
      AusserhalbAB:=AusserhalbAB+1;

 Anzahldaten:=Anzahldaten+250;
 end; {Ende für Schlaufe für TeilDatenVektoren}

 if Anzahldaten > Gesamtanzahl - 250 then begin
 form1.Canvas.font.Color:=clred;
 form1.Canvas.TextOut(abstandx,abstandy+160,'Berechnungen wurden für
 Daten durchgeführt. Es wurden              Tests berechnet');
 form1.canvas.TextOut(abstandx+130,abstandy+160,inttostr(Gesamtanzahl));
```

```
form1.canvas.TextOut(abstandx+320,abstandy+160,inttostr(trunc((Gesamtanzahl-1000
)/250)));
 form1.canvas.Textout(abstandx,abstandy+175,inttostr(AusserhalbAB));
 form1.canvas.TextOut(abstandx+20,abstandy+175,'Testwerte liegen ausserhalb des
Annahmebandes  (Anteil =');
 form1.canvas.TextOut(abstandx+362,abstandy+175,')');

form1.canvas.TextOut(abstandx+310,abstandy+175,floattostrf(AusserhalbAB/trunc((G
esamtanzahl-1000)/250),ffgeneral,6,3));
 form1.Canvas.font.Color:=clblack;
 end;



end;



begin  {Eigentliches Programm - leer}


{Ende eigentliches Programm}
end.
```

```
PROGRAM permutations;  {Beilage 7.4: Pascalroutine aus
http://www.schoenleber.org/pascal/pascal2-03.html [konsultiert 11. September
05]}

  USES crt;

  TYPE str = STRING [80];

  VAR list : str;
      max  : integer;

PROCEDURE rotate (VAR _rlist : str; _len : integer);
  VAR i : integer;
      c : char;
BEGIN
  c := _rlist [1];
  FOR i := 1 TO _len DO
    _rlist [i] := _rlist [i + 1];
  _rlist [_len] := c;
END;

PROCEDURE permute (VAR result : str; len : integer);
  VAR i : integer;
BEGIN
  IF (len = 1) THEN
  BEGIN
    FOR i := 1 TO max DO
      write (result [i]:5);
    writeln;
  END
  ELSE
    FOR i := 1 TO len DO
    BEGIN
      rotate (result,len);
      permute (result,len - 1);
    END;
END;
```