Teil I

Modellierung und Analyse

Kapitel 2

Modellierung und Analyse diskreter Systeme

Als zentrale Modellklasse werden in der Vorlesung diskrete Systeme betrachtet. Also solche Systeme, die zu diskreten Zeitpunkten ihren Zustand ändern. In den meisten Fällen wird das Verhalten der Systeme teilweise durch Zufallsvariablen beschrieben. Dies bedeutet, dass Zufallsvariablen festlegen, wie viel Zeit zwischen zwei Ereignissen vergeht und welche Entscheidung aus einer Menge von möglichen Entscheidungen getroffen wird. Diese Modelle werden per Simulation analysiert. Dabei wird das dynamische Verhalten nachgebildet und beobachtet. Dazu muss der Modellzustand im Programm dargestellt werden, die auftretenden Ereignisse müssen zeitgerecht ausgeführt werden und das Modell muss beobachtet werden, um auf diese Weise zu Resultaten zu gelangen. Als Alternative zur Simulation steht für eine eingeschränkte Modellklasse diskreter Systeme die analytische Berechnung zur Verfügung. Eine kurze Einführung in die analytische Berechnung von Leistungsgrößen wird in Kapitel 3 gegeben.

Es gibt zahlreiche Lehrbücher zum Thema diskrete Simulation mit recht unterschiedlichen Schwerpunktsetzungen. Allerdings existiert momentan kein aktuelles deutschsprachiges Buch, welches die hier behandelte Thematik abdeckt. Das Kapitel beruht auf einzelnen Kapitel der englischsprachigen Lehrbücher [11] und [5]. Beide Bücher liefern eine recht vollständige Darstellung der Thematik, die durch einige spezifische Informationen über spezielle Werkzeuge und Methoden ergänzt wird. Ein weiteres Lehrbuch über diskrete Simulation, welches Simulation auf Basis des in den Übungen verwendeten Tools Arena einführt ist [9]. Ebenfalls zu empfehlen ist ein Blick in die auf der Winter Simulation Conference präsentierten Arbeiten [3]. Neben aktuellen Forschungsarbeiten sind dort auch zahlreiche Tutorien zur Einführung verfügbar.

Das folgende Kapitel gibt eine Einführung in das Gebiet der diskreten Simulation. Die dabei verwendeten Methoden sind der Informatik, der Statistik und den Anwendungswissenschaften zuzuordnen. Die ersten beiden Abschnitte beschäftigen sich mit den programmtechnischen Voraussetzungen zur Realisierung von Simulatoren und sind damit der Informatik zuzuordnen. Die dann folgenden Abschnitte 2.3 bis 2.5 betrachten die Einbeziehung von Zufall in Simulationsmodelle. Dazu sind Methoden aus der Wahrscheinlichkeitsrechnung und Statistik notwendig. Daran anschließend wird ein kurzer Überblick über Simulationssoftware gegeben und es werden die Grenzen der Simulation aufgezeigt. Die dabei betrachteten Methoden sind der Informatik und dem jeweiligen Anwendungsgebiet zuzurechnen. Der letzte Abschnitt des Kapitels stellt Ansätze zur Modellvalidierung vor. Die dabei verwendete Methodik stammt aus der Informatik und der Statistik.

2.1 Konzepte ereignisdiskreter Simulation

Wir wollen in diesem Abschnitt damit beginnen den grundsätzlichen Ablauf diskreter Simulation kennen zu lernen. Diese Grundlagen sind in [5, Kap. 3] und [11, Kap. 2.1, 2.2] sowie in vielen

anderen Lehrbüchern über Simulation zu finden.

Wie bereits herausgearbeitet, soll mittels Simulation der dynamische Ablauf in einem realen System nachgebildet werden. Da ereignisdiskrete Systeme untersucht werden, ändert sich der Zustand jeweils zu Ereigniszeitpunkten. Dies bedeutet, dass

- der Systemzustand bekannt sein muss und
- der Zeitpunkt und die Auswirkungen der nächsten Zustandsänderung ebenfalls bekannt sein müssen.

Statische Struktur eines Simulators

Der Zustand des Simulators ist durch die Werte der Variablen im Programm gegeben. Die nächste Zustandsänderung erfolgt durch ein Programmsegment, auf das der Programmzeiger zeigt. Damit wird der Zustand des Realsystems im Simulator durch eine entsprechende Datenstruktur repräsentiert, die für relevante Zustandsmerkmale Variablen enthält. Man spricht von der *statischen Struktur* des Modells. Die Abbildung vom realen System erfolgt nach dem in der Informatik üblichen Vorgehen. Das reale System wird zuerst in einem Gedankenmodell strukturiert. Dadurch entstehen Objekte mit Attributen, die jeweils in Zustandsvariablen erfasst werden. Die Struktur des Gedankenmodells dient zur Strukturierung der Zustandsvariablen in entsprechenden Datenstrukturen. Das Ergebnis dieses Prozesses ist dann eine minimale Simulator-Datenstruktur.

Dynamische Struktur eines Simulators

Zustandsänderungen im Simulator erfolgen bei Werteänderungen der Zustandsvariablen. Dies bedeutet, dass neue Werte zugewiesen werden müssen, was nur innerhalb des Simulationsprogramms durch dir Ausführung von Simulator-Code geschehen kann. Damit muss zu jedem Ereigniszeitpunkt ein Stück Simulator-Code abgearbeitet werden und in diesem Code-Segment müssen die Auswirkungen des Ereignisses auf den Systemzustand realisiert sein. Um das Vorgehen zu strukturieren werden im Simulator *Ereignistypen* festgelegt. Ein Ereignistyp ist durch eine Menge von Zustandsänderungen, d.h. Wertzuweisungen an Variablen charakterisiert. Zu jedem Ereignistyp gibt es eine *Ereignisroutine*, die die Zustandsänderungen durch den zugehörigen Ereignistyp im Programm umsetzt.

Um die Simulation durchzuführen fehlt damit nur noch ein Mechanismus zur zeitgerechten Ausführung der Ereignisse. Dieser Aspekt soll nun etwas genauer beschrieben werden. Wenn man den Zeitablauf im Modell betrachtet, so fällt auf, dass die Abarbeitung von Ereignisroutinen Zeit benötigt, da Code ausgeführt wird. Aus Sicht des Modells sind Ereignisse dagegen atomar und finden in Nullzeit zu einem Zeitpunkt statt. Andererseits vergeht im Modell und damit auch in der modellierten Realität Zeit zwischen zwei Ereignissen. Wenn zum Zeit t ein Ereignis und zum Zeitpunkt $t + \Delta$ das nächste Ereignis ausgeführt wird, so vergeht die Zeit Δ. Im Simulator wird in diesem Fall das erste Ereignis ausgeführt und nach Abarbeitung der zugehörigen Ereignisroutine die Ereignisroutine für das nächste Ereignis aufgerufen. Dies bedeutet, dass unabhängig von der Größe von Δ praktische keine Zeit zwischen den Ereignissen vergeht. Wir werden uns diesem Phänomen der unterschiedlichen Zeiten in Simulationsmodellen später noch etwas genauer widmen.

Ein einfaches Beispiel

An Hand eines einfachen Beispiels soll zuerst der grundsätzlich Ablauf einer Simulation näher erläutert werden. Wir betrachten als Beispiel einen Schalter an dem Bedienungen stattfinden. Es existiert genau ein Bediener, der zu einem Zeitpunkt genau einen Kunden bedienen kann. Solange Kunden vorhanden sind, bedient der Bediener sie ohne Unterbrechung. Kunden betreten das System (d.h. kommen am Schalter an) und verlangen nach Bedienung. Falls der Schalter belegt ist, so warten die Kunden in einem Warteraum. Wir nehmen an, dass der Warteraum eine potenziell unendliche Kapazität hat und das Kunden nach der Strategie *first come first served* (FCFS)

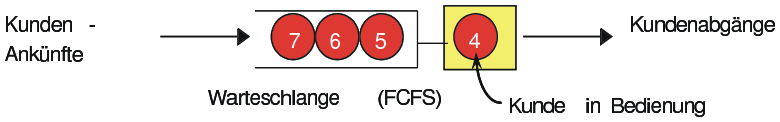

Abbildung 2.1: Station in einem Warteschlangennetz.

bedient werden. Nach Beendigung der Bedienung verlassen die Kunde das System. Die Ankunftszeiten und die Bedienzeiten der Kunden seien bekannt. Sie können entweder aus Messungen an einem realen System oder aus einer stochastischen Verteilung resultieren. Dieser Punkt soll uns erst einmal nicht interessieren, da für den Ablauf der Simulation nur wichtig ist, dass die Werte bekannt sind.

Das beschriebene System stellt eine Station aus einem Warteschlangennetz dar. Warteschlangennetze sind eine weite verbreite Modellklasse zur Modellierung diskreter Systeme. Abbildung 2.1 zeigt die graphische Darstellung unseres Beispiels. Die Darstellung ist so oder ähnlich an vielen Stellen zu finden (siehe auch Abb. 1.11 oder 1.12).

Beginnen wir mit der statischen Struktur des Modells. Es gibt natürlich unterschiedliche Möglichkeiten den Systemzustand zu beschreiben. Je nachdem, welche Details relevant sind, muss eine mehr oder weniger komplexe Datenstruktur erzeugt werden. Wir wollen hier ein Minimalmodell betrachten, um die Abläufe zu verdeutlichen. Deshalb benutzen wir nur die beiden folgenden Variablen zur Modellbeschreibung:

- Die Anzahl der Kunden Q im Warteraum (als integer-Variable kodiert) und
- den Zustand des Bedieners B (der als Boolsche-Variable kodiert ist, 0 = frei, 1 = belegt).

Zusätzlich zu den beiden genannten Variablen gibt es eine Variable t, die in jeder Simulation vorhanden ist und die aktuelle Modellzeit beinhaltet. Wir nehmen an, dass die Simulation zum Zeitpunkt $t = 0$ beginnt, der Zustand (d.h. die Werte von Q und B) zu diesem Zeitpunkt bekannt ist und die Simulation bis zum Zeitpunkt T dauert. $Q(t)$ und $B(t)$ beschreiben den Zustand zum Zeitpunkt t. Der initiale Zustand sei $Q(0) = 0$ und $B(0) = 0$.

Die dynamische Struktur des Modells ist durch zwei Ereignisse charakterisiert, nämlich die Ankunft eines Kunden und das Bedienende eines Kunden.

- Bei der Ankunft eines Kunden:
  Wird B auf 1 gesetzt, falls der Wert 0 ist. Damit startet implizit die Bedienung. Falls B bereits 1 ist, wird Q auf $Q + 1$ gesetzt.
- Bei einem Bedienende:
  Wird B auf 0 gesetzt, falls $Q = 0$ (d.h. kein Kunde wartet), ansonsten wird Q auf $Q - 1$ gesetzt, womit implizit eine neue Bedienung startet.

Es sollte beachtet werden, dass dies nur eine Möglichkeit ist, Ereignisse zu definieren. Zusätzlich zu den Ereignisroutinen muss noch dafür gesorgt werden, dass Ereignisse zeitgerecht ausgeführt werden. Die zugehörigen Mechanismen werden etwas später eingeführt. Weiterhin muss die Simulation das Modellverhalten beobachtbar machen, damit Aussagen über das Systemverhalten möglich sind. Dies geschieht in der Regel dadurch, dass Resultatgrößen zur Auswertung definiert werden. Beispiele für Resultatgrößen sind

- P, die Anzahl der Kunden, die ihre Bedienung im Intervall $[0, T]$ beendet haben
- $\bar{Q} = \int_0^T Q(t)dt/T$, die mittlere Population in der Warteschlange,
- $Q_{\max} = \max_{0 \le t \le T} Q(t)$, die maximale Population in der Warteschlange,
- $\bar{B} = \int_0^T B(t)dt/T$, die mittlere Auslastung des Bedieners,

Kunde	Ankunftszeit	Zwischenankunftszeit	Bedienzeit
1	0.4	0.4	2.0
2	1.6	1.2	0.7
3	2.1	0.5	0.2
4	3.8	1.7	1.1
5	4.0	0.2	3.7
6	5.6	1.6	1.5
7	5.8	0.2	0.4
8	7.2	1.4	1.1

Tabelle 2.1: Ereigniszeiten für die Simulation im Zeitintervall $[0, 10]$.

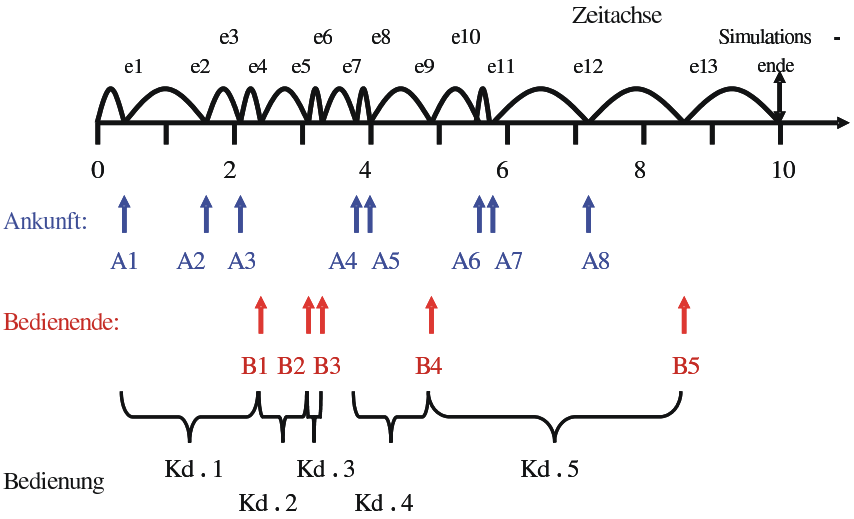

Abbildung 2.2: Ablauf der Beispielsimulation.

- $\bar{V} = \sum_{p=1}^{P} D_p/P$ die mittlere Verweilzeit von Kunden, wobei D_p die Verweilzeit des p-ten Kunden ist und
- $TH = D/T$, der Kundendurchsatz.

Diese Resultatgrößen müssen während der Simulation berechnet werden. Immer dann, wenn sich der Zustand ändert, also zu Ereigniszeitpunkten, müssen die Variablen zur Resultatberechnung modifiziert werden. Bei der nachfolgenden detaillierten Beschreibung des Simulationsablaufs wird die Resultatberechnung exemplarisch für einige Größen vorgestellt.

Tabelle 2.1 beinhaltet die Ankunfts- und Bedienzeiten für einen Beispielablauf im Intervall $[0, 10]$. Ankunftszeiten sind jeweils absolute Zeiten, während Zwischenankunftszeiten und Bedienzeiten Dauern beschreiben. Für einen beliebigen Kunden i gilt offensichtlich

- Bedienende i = Bedienanfang i + Bedienzeit i
- Bedienanfang i = max(Ankunftszeit i, Bedienende $i-1$), wobei Bedienende $0 = 0$ gilt.

Die beschriebenen Ereignisse müssen in der Simulation zeitgerecht ausgeführt werden. Ein zeitgerechter Ablauf erfordert, dass zu den angegebenen Ereigniszeitpunkten die zugehörige Ereignisroutine ausgeführt wird. Abbildung 2.2 zeigt den Ablauf der Beispielsimulation. Es wird nach Ankunft und Bedienende unterschieden. Auf der Zeitachse sieht man das Springen der Simulation von Ereigniszeitpunkt zu Ereigniszeitpunkt. Die unterste Zeile fasst jeweils die zur Bedienung eines Kunden gehörenden Ereignisse zusammen.

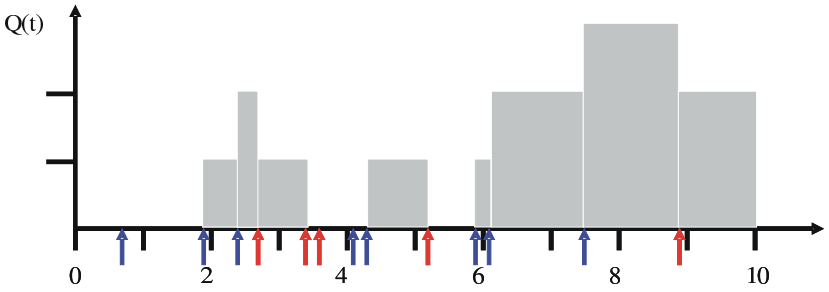

Abbildung 2.3: Verlauf von $Q(t)$ für das Beispielmodell.

Zur Resultatauswertung muss die Simulation beobachtet werden. Dies soll an zwei Beispielen erläutert werden, nämlich der mittleren Population in der Warteschlange und der mittleren Verweilzeit der Kunden. Zur Berechnung der mittleren Population muss $\int_0^T Q(t)dt/T$ ausgewertet werden. Da wir es mit einer ereignisdiskreten Simulation zu tun haben, bleibt $Q(t)$ zwischen den Ereigniszeitpunkten konstant (der Zustand des Systems ändert sich nicht) und springt u.U. zu Ereigniszeitpunkten auf einen anderen Wert. Abbildung 2.3 zeigt den Verlauf von $Q(t)$ für unser Beispiel. Der erste Kunde kommt nicht in den Warteraum, da er direkt bedient wird, der zweite und dritte Kunde müssen dagegen warten usw. Zur Berechnung der mittleren Population muss der Flächeninhalt der grauen Fläche durch die Länge des Beobachtungsintervalls dividiert werden. Zur Bestimmung des Integrals muss damit nur der Flächeninhalt seit der letzten Zustandsänderung in einer Variablen kumuliert werden. Sei Qt diese Variable, die mit 0 initialisiert wird. Bei Ankunft des dritten Kunden wird der Flächeninhalt des Rechtecks beginnend mit der Ankunft des zweiten Kunden berechnet und zu Qt addiert. In unserem Fall beträgt der Abstand zwischen der Ankunft des zweiten und des dritten Kunden 0.5 und die Höhe des Rechtecks ist 1, damit wird 0.5 zu Qt addiert. Beim Bedienende des ersten Kunden ändert sich $Q(t)$ wieder, so dass der zwischenzeitliche Flächeninhalt zu Qt addiert werden muss. Die Breite des neuen Rechtecks ist 0.3 $(= 2.4 - 2.1)$ und die Höhe beträgt 2, so dass 0.6 zu Qt addiert wird. Zur Ermittlung der mittleren Population muss am Ende der Simulation nur noch Qt durch $T = 10$ dividiert werden. Die Ermittlung der mittleren Verweilzeit der Kunden funktioniert anders, da es sich um ein anderes Maß handelt. Abbildung 2.4 zeigt die Messungen der Verweilzeiten. Insgesamt 5 Kunden durchlaufen das System und für jeden Kunden wird eine Verweilzeit gemessen. Der Mittelwert ergibt sich aus der Summe der Kundenverweilzeiten dividiert durch die Anzahl der Kunden, die das System durchlaufen haben. Die Verweilzeit eines Kunden ist ein kundenspezifisches Maß, grundsätzlich müsste für jeden Kunden die Ankunftszeit gespeichert werden und beim Verlassen des Systems müsste von der aktuellen Zeit die Ankunftszeit abgezogen werden, um so an die Verweilzeit zu gelangen. Dies erfordert offensichtlich eine deutlich komplexere Datenstruktur und kann nicht in einer einzelnen Variable gespeichert werden. Allgemein muss also eine Liste von Ankunftszeiten gespeichert werden und jeder Kunde muss eine Identität bekommen, damit beim Verlassen des Systems auf seine Ankunftszeit zugegriffen werden kann. Da die Anzahl der Kunden im System vorab nicht bekannt ist, muss eine dynamische Datenstrukturen verwendet werden. In unserem einfachen Modell können wir uns zu Nutze machen, dass Kunden in der Reihenfolge das System verlassen, in der sie es betreten haben. Dadurch reicht es aus, die Liste der Ankunftszeiten zu speichern und wenn ein Kunde das System verlässt, jeweils die erste noch nicht benutzte Ankunftszeit von der aktuellen Zeit abzuziehen, um so die Verweilzeit zu ermitteln.

Dynamischer Ablauf von Simulatoren

Die Simulation läuft dadurch ab, dass die Ereignisse nacheinander abgearbeitet werden. In unserem Beispiel sind alle Ereigniszeiten als Liste vorgegeben. Das zugehörige Vorgehen bezeichnet man auch als *trace-getriebene* Simulation. Alternativ werden die Zeiten während der Simulation aus vorgegebenen Verteilungen generiert. Damit sind die Zeiten nicht vorab bekannt, sondern werden

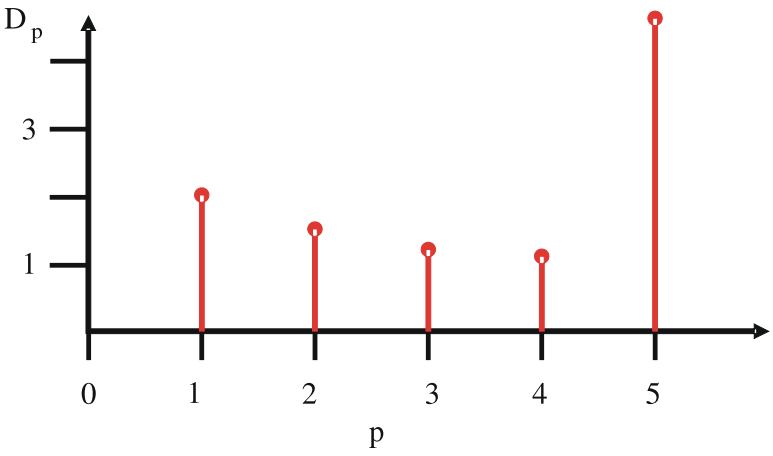

Abbildung 2.4: Kundenverweilzeiten im Beispielmodell.

zur Laufzeit des Simulators erzeugt. Dies kann nur in den Ereignisroutinen erfolgen, da nur an dieser Stelle während der Simulation Programmcode abgearbeitet wird. Die Ereignisroutinen haben damit die folgenden Aufgaben:

1. Modifikation des Systemzustands,
2. Speicherung der Resultatwerte und
3. Einplanung zukünftiger Ereignisse.

Der gesamte Ablauf einer Simulation wird durch die so genannte Simulationshauptroutine gesteuert. In Abbildung 2.5 wird dieser Ablauf dargestellt. Der zentrale Punkt ist die Ausführung der Ereignisroutinen in der Simulationsschleife. In den Ereignisroutinen werden die drei angegebenen Aufgaben erledigt. Vor Beginn der Simulationsschleife müssen die Variablen initialisiert werden und nach Beendigung die Resultatgrößen ausgewertet werden. Neben der Beschreibung der Ereignisroutinen sind die drei in der Abbildung angegebenen Aspekte von Bedeutung, nämlich die Verwaltung zukünftiger Ereignisse in einer adäquaten Datenstruktur, die Generierung von Zufallszahlen und die statistische Auswertung der Simulation. Wir werden uns den beiden letzten Punkten in späteren Abschnitten widmen.

Ereignisverwaltung und Ereignisspeicherung

Ereignisse sind charakterisiert durch ihre Eintrittszeit t_i und einen Ereignistyp tp_i. Da sie konsekutiv nach Eintrittszeit abgearbeitet werden, müssen sie nach Eintrittszeit geordnet in einer Liste gespeichert werden. Bei Erzeugung eines neuen Ereignisses muss dieses an die entsprechende Stelle in der Liste eingeordnet werden. Auslesen des nächsten Elements bedeutet, dass das erste Element der Liste ausgelesen wird. Zusätzlich muss es noch die Möglichkeit geben Ereignisse zu löschen. Dies kommt in dem einfachen Beispiel nicht vor, ist aber bei komplexeren Modellen üblich. Als einfaches Beispiel kann man einen Schalter mit zwei Kundenklassen unterschiedlicher Priorität betrachten. Ein Kunde höherer Priorität unterbricht die Bedienung eines Kunden niedrigerer Priorität. Wenn nun das Ereignis Bedienende für den Kunden niedriger Priorität eingeplant wurde und ein Ereignis Ankunft Kunde mit hoher Priorität eintritt, so wird in der Ereignisroutine für die Ankunft das Bedienende des Kunden höherer Priorität geplant und das bereits geplante Bedienende des Kunden niedriger Priorität gelöscht. Die Situation mit prioritisierten Kunden mag für Bedienschalter wie Supermarktkassen oder ähnliche Systeme künstlich erscheinen, ist aber ein übliches Verhalten, wenn man den Schalter als CPU und die Kunden als Jobs interpretiert. In diesem Fall unterbricht ein Betriebssystem-Job einen Benutzer-Job.

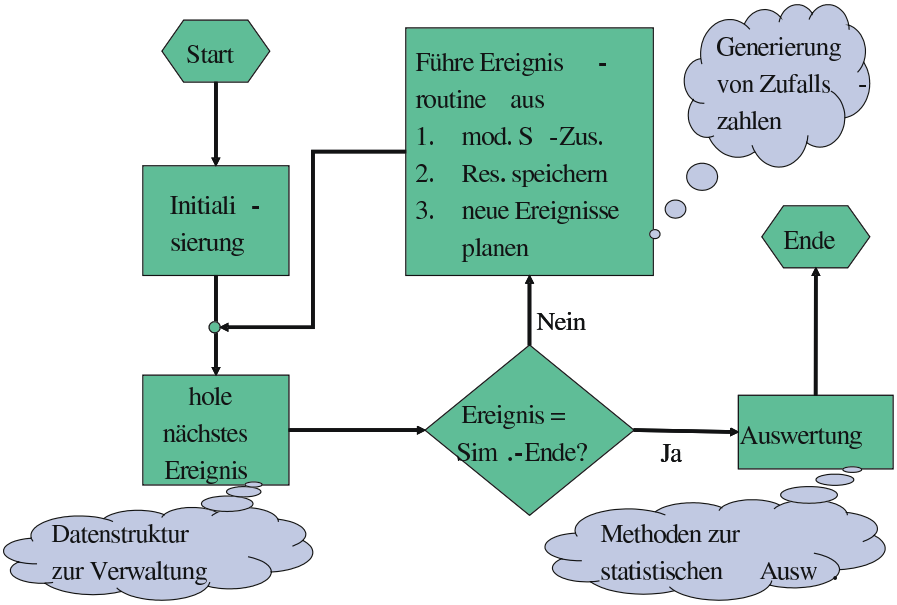

Abbildung 2.5: Ablauf der Simulationshauptroutine.

t_i	t_1	t_2	t_3	t_4	t_5	...
tp_i	tp_1	tp_2	tp_3	tp_4	tp_5	...

Abbildung 2.6: Beispiel für eine Ereignisliste (es gilt $t_i \leq t_{i+1}$).

Die Datenstruktur zur Verwaltung der Ereignisse bezeichnet man als *Ereignisliste*. Der Inhalt der Ereignisliste umfasst die bisher geplante Modellzukunft. Die aktuelle Zeit der Simulation ergibt sich aus der Zeit des nächsten Ereignisses. In der in Abbildung 2.6 gezeigten Situation wäre dies t_1. In diesem Fall können neue Ereignisse nur zu Zeitpunkten $t \geq t_1$ eingefügt werden. Die Simulation kann nur in die Zukunft planen und nicht in der Vergangenheit etwas verändern. Dies entspricht unserem realen Zeitablauf. Nicht definiert ist die Reihenfolge von Ereignissen, die zu gleicher Zeit eintreten (d.h. $t_i = t_j$). Prinzipiell ist damit eine beliebige Reihenfolge möglich. Es gibt Beispiele, bei denen eine Reihenfolge gleichzeitiger Ereignisse vom Modell vorgegeben wird. Z.B. wenn ein Ereignis ein anderes Ereignis zur gleichen Zeit initiiert, so wird das initiierte Ereignis immer später ausgeführt werden. Im allgemeinen Fall existieren solche Abhängigkeiten nicht und gleichzeitige Ereignisse können in beliebiger Reihenfolge angeordnet werden. Die Anordnung kann aber das spätere Modellverhalten verändern. Das Modellverhalten ist für gleichzeitige Ereignisse nicht eindeutig spezifiziert und ist damit indeterministisch. Dieser Indeterminismus ist vom Indeterminismus durch die bewusste Einführung von Stochastik zu unterscheiden. In praktischen Implementierungen wird oftmals eine feste Strategie der Einordnung gleichzeitiger Ereignisse gewählt (z.B. zuerst eingeordnet - zuerst in der Ereignisliste, zuerst eingeordnet - zuletzt in der Ereignisliste).

Dynamischer Ablauf der Beispielsimulation

Wir wollen zu unserem Beispielmodell zurückkehren und das konkrete Modell und einen exemplarischen Ablauf beschreiben. Die folgenden Zustandsvariablen werden benutzt:

- Q Anzahl Kunden in der Warteschlange,
- B Status des Bedieners.

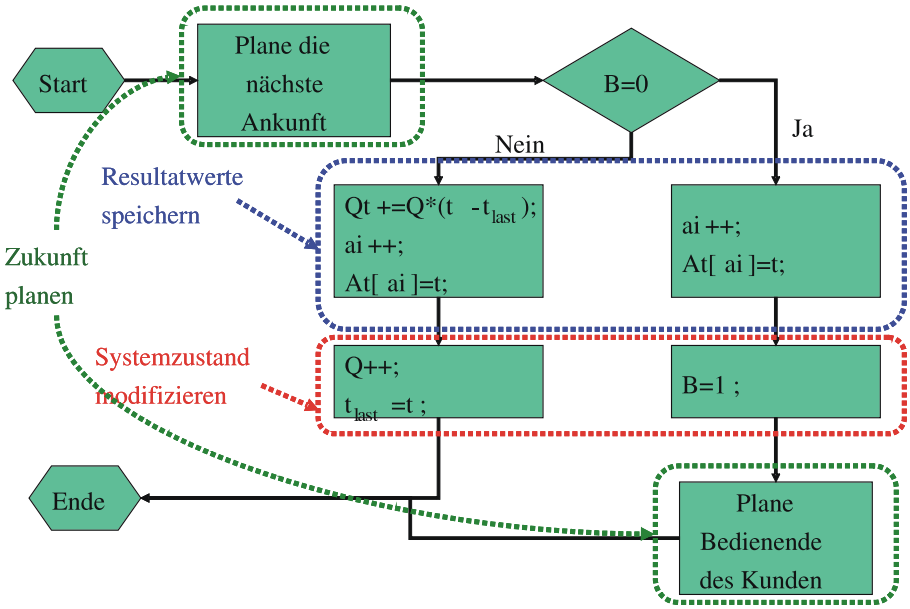

Abbildung 2.7: Ereignisroutine für die Kundenankunft.

Weiterhin werden die folgenden Variablen zur Resultataufzeichnung und -auswertung benötigt.

- Qt der kumulierte Wert der Auftragszahl in der Warteschlange,
- Ft der kumulierte Wert der Verweilzeiten,
- t_{last} der Zeitpunkt der letzten Änderung von Q,
- ai Nummer des letzten Auftrags, der angekommen ist,
- bi Nummer des letzten Auftrags, der das System verlassen hat,
- $At[i]$ Ankunftszeit des i-ten Auftrags.

Die Werte werden jeweils mit 0 initialisiert.

Darüber hinaus dient t zur Darstellung der aktuellen Simulationszeit, diese entspricht der aktuellen Ereigniszeit. Die Ereignisroutine für eine Kundenankunft wird in Abbildung 2.7 gezeigt. Die Beschreibung wird als Flussdiagramm mit der üblichen Semantik gegeben. Die Ereignisroutine umfasst die drei genannten Aufgaben: Planen der Modellzukunft, Modifizieren des Systemzustandes und Auswerten der Resultate. Die jeweiligen Komponenten sind im Flussdiagramm markiert. Das Planen der Modellzukunft besteht einfach darin, dass ein zugehöriges Ereignis in die Ereignisliste eingehängt wird. Die anderen beiden Schritte bestehen aus den Modifikationen der Variablen. In ähnlicher Form lässt sich die Ereignisroutine für das Bedienende formulieren (siehe Abbildung 2.8). Als letzte Ereignisroutine wird eine Routine für das Ereignis Simulationsende benötigt. Diese Routine wird in Abbildung 2.9 gezeigt. In der Ereignisroutine werden die Ergebnisse, nämlich die mittlere Population im Warteraum und die mittlere Verweilzeit der Kunden, ausgegeben. Die Simulation ermittelt die Mittelwerte für Population und Verweilzeit im Intervall $[0, T]$ für eine vorgegebene Sequenz von Ankunfts- und Bedienzeiten. Im allgemeinen Fall würden die Ankunfts- und/oder Bedienzeiten aus einer stochastischen Verteilung generiert und ein Durchlauf liefert eine mögliche Realisierung. Um in diesem Fall Aussagen über das Modellverhalten zu treffen (d.h. alle möglichen Realisierungen des Zufalls) ist eine statistische Auswertung notwendig, auf die in Abschnitt 2.5 eingegangen wird. In unserem Beispiel wird die Simulation nach einer vorgegebenen Zeit abgebrochen. Es wird also für einen festen Zeithorizont eine Simulation durchgeführt. Es gibt

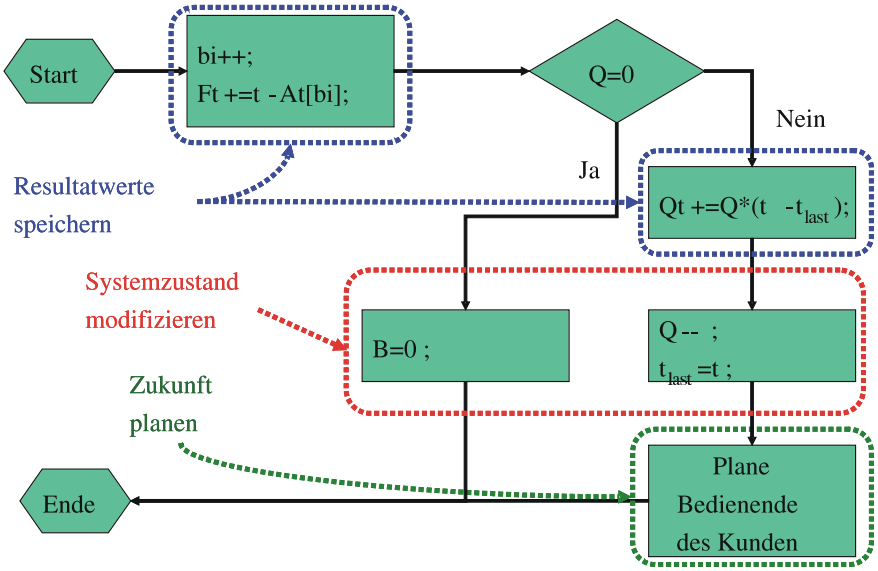

Abbildung 2.8: Ereignisroutine für das Bedienende.

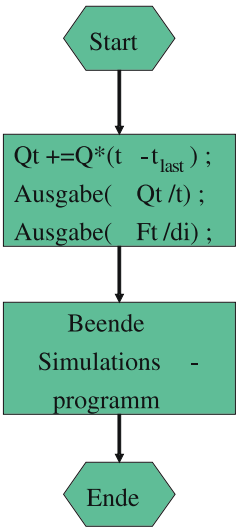

Abbildung 2.9: Ereignisroutine für das Simulationsende.

andere Abbruchbedingungen. So kann abgebrochen werden, wenn ein bestimmtes Ereignis eintritt. Dies wird dadurch realisiert, dass bei Eintreten des Ereignisses ein neues Ereignis Simulationsende zum aktuellen Zeitpunkt eingefügt wird. Weiterhin wird oftmals nach Laufzeit oder Genauigkeit der Ergebnisse abgebrochen. Im ersteren Fall wird das Ereignis Simulationsende dann eingefügt, wenn die vorgegebene CPU-Zeit durch das Simulationsprogramm verbraucht wurde, im zweiten Fall muss die statistische Auswertung mit der Simulation interagieren.

Wir betrachten nun exemplarisch den Ablauf der Simulation für die in Tabelle 2.1 gezeigten Zeiten. Zusätzlich wird das Simulationsende für den Zeitpunkt 10 und eine weitere Ankunft für den Zeitpunkt 10.2 geplant. Die Ereignisse werden als A Ankunft, B Bedienende und E Simulationsende kodiert. Der Ablauf der Simulation wird hier sehr detailliert beschrieben, da er die zentralen Elemente enthält, die in jeder Simulation vorkommen und für das Verständnis diskreter Simulation essentiell sind.

In Abbildung 2.10 werden die Initialisierung der Simulation und die ersten drei Ereignisse dargestellt. Zu Beginn der Simulation werden alle Zustandsvariablen und die Resultatvariablen initialisiert. Im Beispiel werden sämtliche Werte mit 0.0 initialisiert. Die Simulationszeit startet bei 0 und die Ereignisliste enthält die erste Ankunft, die für den Zeitpunkt 0.4 eingeplant wurde und das Simulationsende, das gleich zu Beginn für den Zeitpunkt 10 eingeplant wurde. In der Simulationshautproutine wird nun das erste Ereignis aus der Ereignisliste geholt, wodurch die Zeit t auf den Ereigniszeitpunkt 0.4 gesetzt wird. Das erste Ereignis ist eine Ankunft, die durch Ausführung der entsprechenden Ereignisroutine im Modell realisiert wird. Der zweite Block in Abbildung 2.10 zeigt den Modellzustand nach Abarbeitung der Ereignisroutine. Zum Zeitpunkt 0.4 findet der atomare Zustandswechsel in Nullzeit statt. Der Bediener ist anschließend belegt ($B = 1$), die erste Ankunft bekommt die Nummer 1 ($ai = 1$) und die Ankunftszeit ist ($At = (0.4)$). Die restlichen Variablen werden nicht verändert, da die Warteschlange weiterhin leer ist und bisher auch noch kein Kunde das System verlassen hat. In der Ereignisliste wird durch die Ankunft des ersten Kunden das Bedienende des Kunden und die Ankunft des zweiten Kunden eingeplant. Im nächsten Schritt erfolgt die Ankunft des zweiten Kunden. Da der erste Kunde noch in Bedienung ist, muss der zweite Kunde warten. Dies wird dadurch realisiert, dass Q auf 1 gesetzt wird. Weiterhin muss t_{last} auf 1.6 gesetzt werden, da sich der Zustand des Warteraums zum Zeitpunkt 1.6 geändert hat. Ansonsten sind die Änderungen analog zu den Änderungen bei Ankunft des ersten Kunden. Das nächste Ereignis ist die dritte Ankunft, die ebenfalls die entsprechenden Änderungen an den Modellvariablen durchführt. Bei der dritten Ankunft ändert sich der Zustand des Warteraums. Zur späteren Auswertung des Integrals über die Population im Warteraum muss deshalb der bisher akkumulierte Wert gespeichert werden. Dieser Wert lautet $(2.1 - 1.6) \cdot 1 = 0.5$.

Abbildung 2.11 zeigt die nächsten vier Schritte der Simulation. Das nächste Ereignis ist das Bedienende des ersten Kunden zum Zeitpunkt $t = 2.4$. Der Bediener bleibt danach belegt, da noch zwei weitere Kunden im Warteraum warten und die Bedienung des nächsten Kunden direkt beginnt. Dadurch reduziert sich die Zahl der Kunden im Warteraum auf $Q = 1$. Zur Berechnung der mittleren Füllung des Warteraums wird $(2.4 - 2.1) \cdot 2 = 0.6$ zu Qt addiert. Da der erste Kunde das System verlässt wird $bt = 1$ gesetzt, die Verweilzeit des Kunden ergibt sich aus der aktuellen Zeit minus der Ankunftszeit des Kunden, als $t - At(bt) = 2.4 - 0.4 = 2.0$. Dieser Wert wird zu Ft addiert. Diese Art der Berechnung der Verweilzeit funktioniert nur, weil Kunden in der Reihenfolge das System verlassen, in der sie es betreten haben. Im allgemeinen Fall müsste der Kunde sich seine Ankunftszeit merken. Dies bedeutet, dass für jeden Kunden eine Variable vorhanden sein muss. Dies wird meist dadurch realisiert, dass Kunden Inkarnationen spezieller Datentypen sind. Dieser Aspekt wird später detailliert behandelt. Durch das Bedienende des ersten Kunden wird das Bedienende des zweiten Kunden planbar und für den Zeitpunkt 3.1 in der Ereignisliste vorgemerkt. Nach dem ersten Bedienende folgen zwei weitere Bedienenden, die ähnlich behandelt werden. Nach dem Bedienende des dritten Kunden ist das System leer, d.h. $Q = B = 0$. Die nächste Ankunft ist aber für den Zeitpunkt 3.8 eingeplant. Die Ankunft des vierten Kunden setzt den Wert von B wieder auf 1 und plant die fünfte Ankunft ein. Der Wert Qt ändert sich durch die fünfte Ankunft nicht, da Q im Intervall vor der Ankunft 0 war.

Start der Simulation

Zustandsvariablen	Simulatorzustand
Q=0 B=0 ai=0 bi=0 t_{last}=0.0	t=0.0 EL: 0.4 A; 10 E
Resultatvariablen: Qt=0.0 Ft=0.0 At=()	

Erste Ankunft

Zustandsvariablen	Simulatorzustand
Q=0 B=1 ai=1 bi=0 t_{last}=0.0	t=0.4 EL: 1.6 A; 2.4 B; 10 E
Resultatvariablen: Qt=0.0 Ft=0.0 At=(0.4)	

Zweite Ankunft

Zustandsvariablen	Simulatorzustand
Q=1 B=1 ai=2 bi=0 t_{last}=1.6	t=1.6 EL: 2.1 A; 2.4 B; 10 E
Resultatvariablen: Qt=0.0 Ft=0.0 At=(0.4,1.6)	

Dritte Ankunft

Zustandsvariablen	Simulatorzustand
Q=2 B=1 ai=3 bi=0 t_{last}=2.1	t=2.1 EL: 2.4 B; 3.8 A; 10 E
Resultatvariablen: Qt=0.5 Ft=0.0 At=(0.4,1.6,2.1)	

Abbildung 2.10: Erster Teil des Beispielsimulationslaufs.

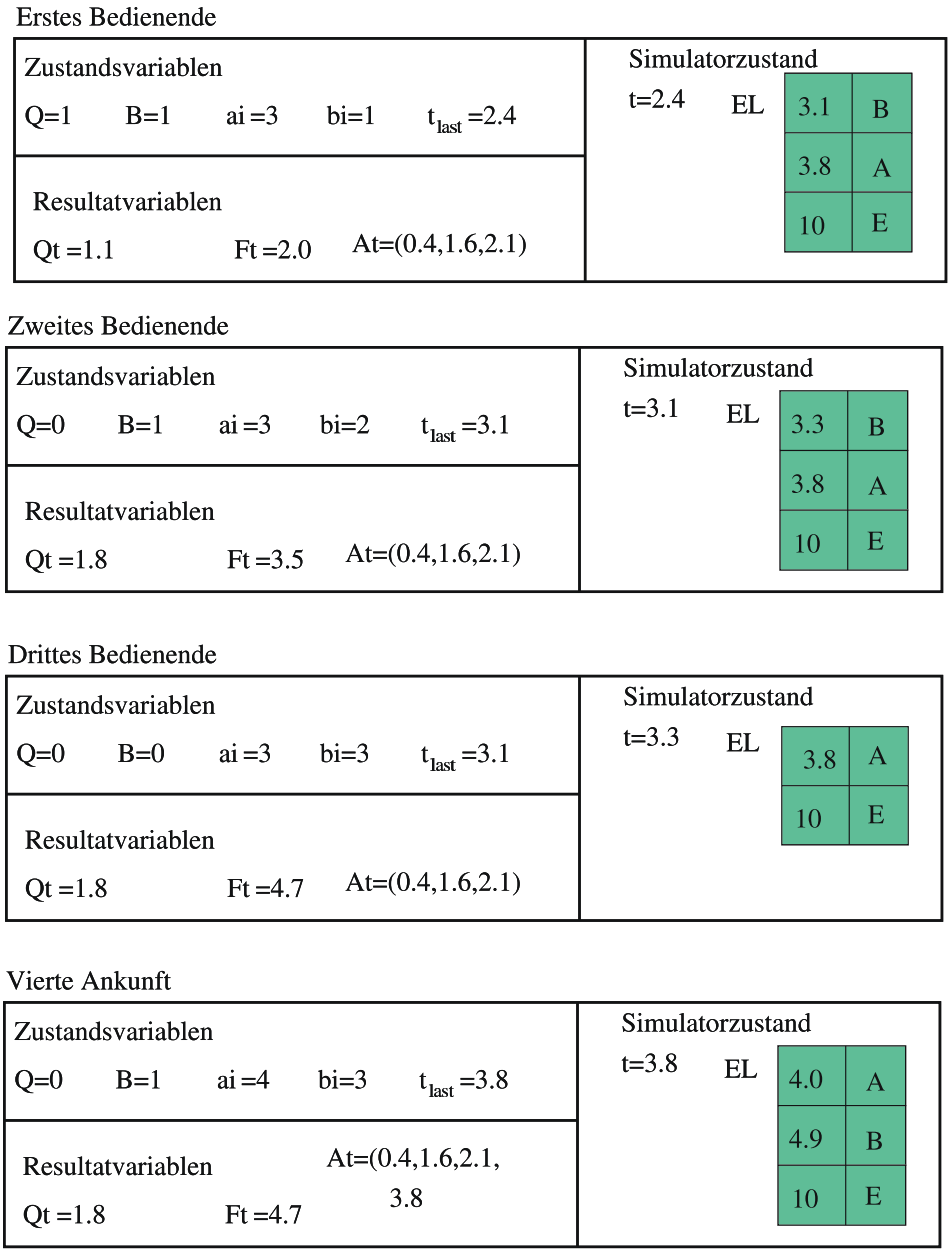

Abbildung 2.11: Zweiter Teil des Beispielsimulationslaufs.

Fünfte Ankunft

Zustandsvariablen	Simulatorzustand
Q=1 B=1 ai =5 bi=3 t_{last} =4.0	t=4.0 EL: 4.9 B; 5.6 A; 10 E
Resultatvariablen Qt =1.8 Ft =4.7 At=(0.4,1.6,2.1, 3.8,4.0)	

Viertes Bedienende

Zustandsvariablen	Simulatorzustand
Q=0 B=1 ai =5 bi=4 t_{last} =4.9	t=4.9 EL: 5.6 A; 8.6 B; 10 E
Resultatvariablen Qt =2.7 Ft =5.8 At=(0.4,1.6,2.1, 3.8,4.0)	

Sechste Ankunft

Zustandsvariablen	Simulatorzustand
Q=1 B=1 ai =6 bi=4 t_{last} =5.6	t=5.6 EL: 5.8 A; 8.6 B; 10 E
Resultatvariablen Qt =2.7 Ft =5.8 At=(0.4,1.6,2.1, 3.8,4.0,5.6)	

Siebte Ankunft

Zustandsvariablen	Simulatorzustand
Q=2 B=1 ai =7 bi=4 t_{last} =5.8	t=5.8 EL: 7.2 A; 8.6 B; 10 E
Resultatvariablen Qt =2.9 Ft =5.8 At=(0.4,1.6,2.1, 5.8,3.8,4.0, 5.6)	

Abbildung 2.12: Dritter Teil der Beispielsimulation.

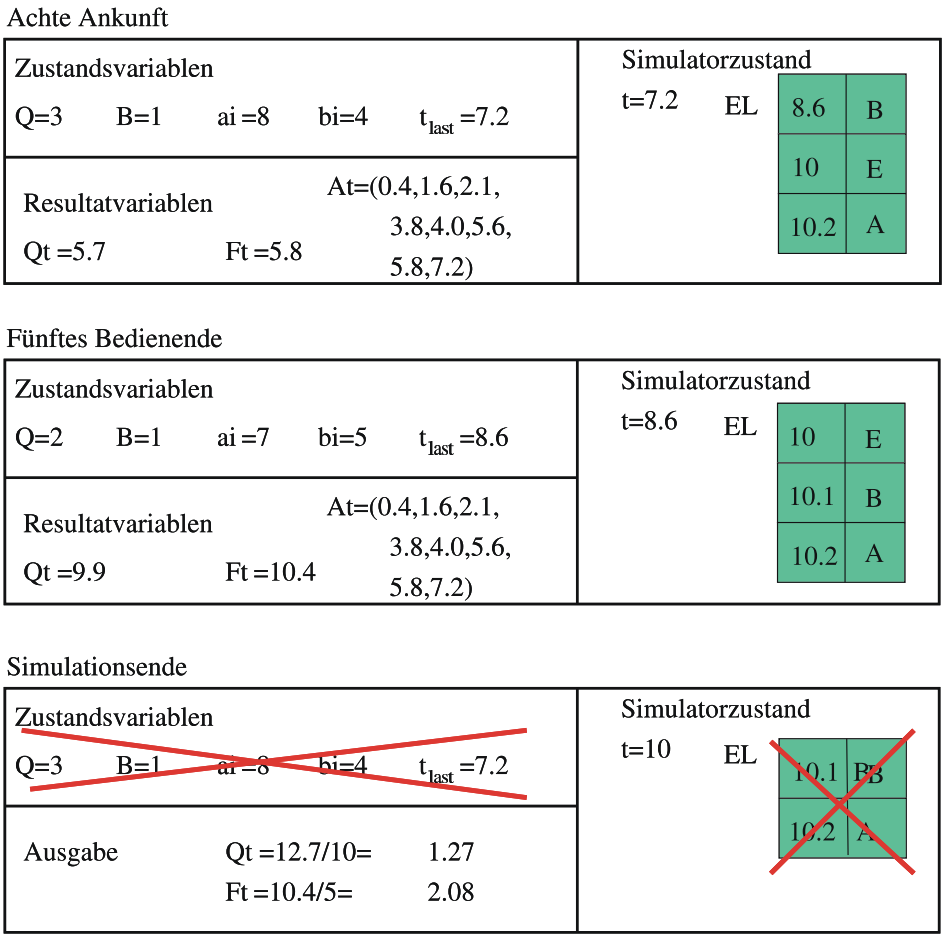

Abbildung 2.13: Letzter Teil der Beispielsimulation.

Die nächsten vier Ereignisse sind in Abbildung 2.12 dargestellt. Das Vorgehen entspricht dem der vorherigen Ereignisse.

Abbildung 2.13 zeigt schließlich die letzten Schritte der Simulation. Zuerst erfolgt die achte Ankunft, die gleichzeitig eine neunte Ankunft für den Zeitpunkt 10.2 einplant. Dieser Zeitpunkt liegt nach dem Ereignis Simulationsende. Dies ist aber nicht vorab feststellbar, deshalb wird das Ereignis einsortiert. Zum Zeitpunkt 10 wird das Ereignis Simulationsende ausgeführt. Dieses Ereignis sorgt dafür, das die Simulation beendet wird und die Auswertung erfolgt. Zur Auswertung wird Ft, die Summe der Verweilzeiten aller Kunden, die das System durchlaufen haben, durch die Anzahl der Kunden, die das System verlassen haben, dividiert. Kunden, die noch im System sind, zählen nicht mit. Zur Berechnung der mittleren Warteraumbelegung wird Qt, die kumulierte Kundenzahl im Warteraum (also die in Abbildung 2.3 gezeigte Fläche) durch die Länge der Simulation dividiert. In diesem Fall zählen Kunden mit, die sich zu Simulationsende noch im Warteraum befinden. Zum Abbruch der Simulation werden die zugehörigen Daten freigegeben, weitere Ereignisse in der Ereignisliste werden also gelöscht und nicht mehr ausgeführt.

Das einfache Beispiel enthält in seiner jetzigen Codierung redundante Information, die aber zum Verständnis hilfreich ist. So gilt $Q = \max(0, bi - ai - 1)$ und $B = \delta(bi > ai)$, wobei $\delta(b) = 1$ falls b true ist und 0 sonst. Weiterhin fällt auf, dass die Ereignisliste maximal 3 Elemente enthält, da immer genau eine Ankunft und ein Bedienende voraus geplant wird. Dazu kommt noch das Simulationsende. Außerdem werden keine Ereignisse aus der Ereignisliste gelöscht. Im Beispiel könnte die Ereignisliste damit sehr einfach realisiert werden. Für komplexere Modelle kann eine Maximalzahl von Elementen in der Ereignisliste nicht garantiert werden und Funktionen zum

Löschen von Ereignissen werden benötigt. Die Variable At speichert alle bisherigen Ankunftszeiten, die Länge wächst mit steigender Simulationszeit und Ankunftszahl. Man kann sich überlegen, dass eigentlich nur die Ankunftszeiten für die Kunden $bi, \ldots, ai$ benötigt werden, da die übrigen Kunden das System bereits verlassen haben. Trotzdem ist schon für das einfache Beispiel die Zahl der Werte, die gespeichert werden müssen nicht vorab bekannt. Damit muss eine dynamische Datenstruktur zur Speicherung der Ankunftszeiten verwendet werden.

Das Vorgehen, das für das einfache Beispiel beschrieben wurde, lässt sich prinzipiell auch auf komplexere Modelle übertragen. Insbesondere ist der Zeitablauf in allen ereignisdiskreten Systemen so wie im Beispiel beschrieben. Unterschiede ergeben sich aber bei den Datenstrukturen, die zur Zustandsdarstellung benötigt werden. Insbesondere reicht es oft nicht aus, Kunden implizit durch eine integer-Variable zu kodieren. Kunden haben oft Eigenschaften, die z.B. Bedienzeiten beeinflussen und für jeden Kunden gespeichert werden müssen. Damit muss für jeden ankommenden Kunden eine neue Variable instantiiert werden. Die Population im Modell ist eine Menge von Kunden, im Simulationsprogramm dargestellt durch eine Liste von Variablen vom Typ Kunden. Verlässt ein Kunde das System, so werden seine Daten nicht mehr benötigt und der belegte Speicherplatz sollte wieder freigegeben werden, da ansonsten die Simulation mit steigender Simulationszeit immer mehr Speicher benötigt. Die dynamische Belegung und Freigabe ist ein zentraler Aspekt der Realisierung von Simulationsprogrammen.

Zeitablauf in der Simulation

Betrachten wir noch einmal den Zeitablauf als zentralen Aspekt der Simulation dynamischer Systeme. Die Zeit läuft diskontinuierlich in Sprüngen ab. Der Zustand des Modells ist nur zu Ereigniszeitpunkten direkt nach Abarbeitung der Ereignisroutine definiert. Ereignisse sind atomar und verbrauchen damit keine Zeit und können auch nur vollständig ausgeführt werden. Es gibt also keinen Zustand während des Auftretens eines Ereignisses. Gleichzeitig ist der Zustand zwischen Ereigniszeitpunkt nicht vollständig definiert. So ist z.B. zwischen zwei Ereignissen der Wert einiger Variablen nicht korrekt gesetzt. Als Beispiel sei auf den Wert von Qt in dem einfachen Beispielmodell verwiesen, der nur zu Ereigniszeitpunkten angepasst wird. Wenn also der Modellzustand zu einem Zeitpunkt ermittelt werden soll, so ist für diesen Zeitpunkt ein Ereignis vorzumerken. Dies kann auch ein spezielles Beobachtungsereignis sein, das dann allerdings in der zugehörigen Ereignisroutine die Variablenwerte anpassen muss. Es ist ferner zu beachten, dass mehrere Ereignisse zu einem Zeitpunkt stattfinden können, so dass ein Modell zu einem Zeitpunkt mehrere Zustände haben kann.

Den Ablauf eines Simulators über ein Zeitintervall $[0, T]$ bezeichnet man als *Trajektorie*. Für ein Modell mit vorgegebenen Ereigniszeiten ist die Trajektorie dann eindeutig definiert, wenn keine gleichzeitigen Ereignisse vorkommen. Bei gleichzeitigen Ereignissen, die nicht kausal abhängig sind, gibt es mehrere Trajektorien. In stochastischen Simulatoren, bei den Zeiten oder Entscheidungen durch Zufallsvariablen definiert sind, können meist unendliche viele Trajektorien auftreten. Entsprechend erfordert die Auswertung stochastischer Simulationen Aussagen über alle möglichen Trajektorien auf Basis der Beobachtung einiger Trajektorien. Diese Beobachtungslage erlaubt natürlich nur statistische Aussagen.

Zentraler Aspekt jeder Simulation sind die unterschiedlichen Zeitbegriffe der Simulation. Wir unterscheiden die folgenden Zeitbegriffe.

- Die *Objektzeit* ist die Zeit, in der das reale System, zumindest hypothetisch, abläuft.
- Die *Modell-* oder *Simulationszeit* wird im Simulationsprogramm manipuliert indem Ereignisse abgearbeitet werden und damit t gesetzt wird. Die Modellzeit imitiert die Objektzeit und ist damit bis auf Translation (z.B. durch Skalierung, Start bei 0 etc.) identisch zur Objektzeit.
- Das Simulationsprogramm läuft in *Realzeit* ab. Diese Zeit ist für die Analyse natürlich belanglos.

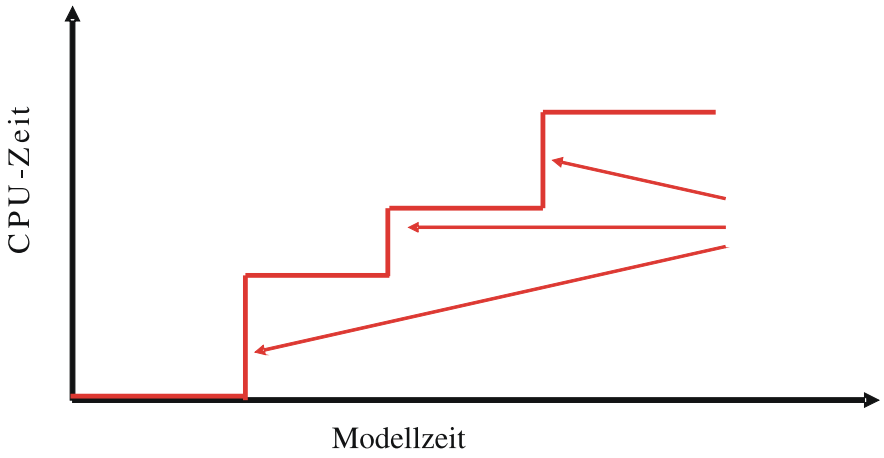

Abbildung 2.14: Zusammenhang zwischen CPU-Zeit und Modellzeit.

- Zur Ausführung benötigt das Simulationsprogramme *Ausführungszeit* oder *CPU-Zeit*. Die CPU-Zeit ist neben dem Speicherbedarf der zentrale Teil des Ressourcenbedarfs. Simulationsprogramme zur Abbildung realer Systeme sind oft notorische Langläufer. In den meisten Fällen ist auf heutigen Rechnern der CPU-Zeitbedarf ein größeres Problem als der Speicherbedarf.

Abbildung 2.14 stellt noch einmal den Zusammenhang zwischen CPU-Zeit und Modellzeit dar. Immer dann, wenn Modellzeit verstreicht, verstreicht keine CPU-Zeit und umgekehrt. Dies liegt daran, dass endliche Modellzeitintervalle zwischen Ereignissen in verschwindender CPU-Zeit übersprungen werden und verschwindende Modellzeitintervalle zu Ereigniszeitpunkten endliche CPU-Zeit benötigen, da Ereignisroutinen abgearbeitet werden. Damit ist der CPU-Zeitbedarf eines Simulators von der Anzahl der Ereignisse und der Komplexität der Ereignisroutinen und nicht allein von der Modellzeit abhängig. Die Modellzeit ist in den meisten Fällen festgelegt, da eine gewisse Anzahl Beobachtungen notwendig ist oder das System für ein vorgegebenes Zeitintervall analysiert werden soll. Eine Erhöhung der Effizienz von Simulatoren muss damit an der Ereigniszahl und an der Komplexität der einzelnen Ereignisse ansetzen. Dies bedeutet, dass am Abstraktionsniveau der Modellierung und damit meist ursächlich am Abstraktionsniveau des mentalen Modells angesetzt werden muss. Ein höheres Abstraktionsniveau führt zu weniger Ereignissen und zu weniger komplexen Ereignissen. Die Kunst der Modellbildung, die die Laufzeit des resultierenden Simulators maßgeblich beeinflusst, ist die Erstellung eines möglichst abstrakten Modells, das alle für die Ergebnisermittlung wesentlichen Systemaspekte abbildet.

2.2 Spezifikation von Simulatoren

Nachdem im letzten Abschnitt das grundsätzliche Vorgehen der ereignisdiskreten Simulation eingeführt wurde, soll in diesem Abschnitt die Umsetzung der vorgestellten Konzepte in ein Simulationsprogramm behandelt werden. Das vorgestellte Prinzip der ereignisdiskreten Simulation ist strukturiert und verständlich, die konkrete Umsetzung in ein Simulationsprogramm erfordert aber Abstraktion und ist damit komplex. Grundsätzlich ist zu beantworten, welche sprachlichen Konstrukte notwendig sind, um Simulatoren zu implementieren. Wir betrachten hier eine programmiersprachliche Realisierung und orientieren uns an gängigen Programmiersprachen. Später, in Abschnitt 2.6, werden andere Arten der Spezifikation, wie graphische oder menübasierte Techniken, vorgestellt und spezielle Simulationssprachen eingeführt. Die abstrakteren graphischen Spezifikationsansätze dienen in den meisten Fällen als Eingabeschnittstelle für eine Simulationssprache. D.h. die graphische Spezifikation wird automatisch in ein Simulationsprogramm in einer gängige Programmiersprache oder speziellen Simulationssprache transformiert und dieses Programm

wird dann für die Zielarchitektur übersetzt. Aus diesem Grund macht es Sinn, die Realisierung von Simulationsprogrammen in Programmiersprachen und die Syntax und Semantik notwendiger Konstrukte einzuführen. Darüber hinaus soll kurz erläutert werden, welche Voraussetzungen eine Programmiersprache erfüllen muss, um die Konstrukte zu realisieren.

Anforderungen an Programmiersprachen

Die folgenden Anforderungen an Programmiersprachen durch spezielle, simulationsspezifische Konstrukte wurden herausgearbeitet:

1. Höhere rekursive Datenstrukturen zur Speicherung von homogenen aber auch inhomogenen Variablen oder Objekte. Dazu notwendig sind Funktionen zum
   (a) Einfügen nach Schlüssel, am Anfang oder am Ende,
   (b) Herauslesen am Anfang, am Ende oder nach Schlüssel,
   (c) Löschen nach Schlüssel.
2. Die Verfügbarkeit eines Kalenders zukünftiger Ereignisse geordnet nach Eintretenszeit. Funktionen zum Einfügen von Elementen (nach Schlüssel), zum Herauslesen des ersten Elements und zum Löschen eines Elements nach Schlüssel werden benötigt. Dies ist im Prinzip ein Spezialfall von 1, der aber hier gesondert behandelt wird, da die effiziente Realisierung der Zugriffe auf die Ereignisliste von zentraler Bedeutung für die Effizienz der Simulationsprogramme ist.
3. Die Verfügbarkeit von Methoden zur dynamischen Belegung und Freigabe von Speicherplatz, um damit Variablen und Objekte zur Laufzeit zu erzeugen und zu zerstören. Diese Anforderung ist zentral, da in komplexen Simulationsprogrammen eine Vielzahl von Objekten generiert und nur temporär benötigt wird.
4. Schließlich müssen Methoden zur Generierung von Zufallszahlen vorhanden sein und statistische Methoden zur Simulationsauswertung unterstützt werden.

Im Licht der genannten Anforderungen betrachten wir gängige Programmiersprachen wie C, C++ und Java. Auch heute noch werden viele Simulationsprogramme in diesen Sprachen oder auch in Fortran realisiert (siehe auch [11, Kap. 1]), auch wenn dies aus Sicht des Software-Entwurfs nicht befriedigend ist.

Höhere Datenstrukturen, wie sie für die Simulation benötigt werden, sind heute in praktisch allen modernen Programmiersprachen vorhanden. Sie werden z.B. als *structure*, *record* oder *class* bezeichnet. Als Beispiel soll die Definition einer Datenstruktur zur Beschreibung eines Kunden mit den Attributen *Name* (als Zeichenkette kodiert), *Ankunftszeit* (als reelle Zahl kodiert) und *Priorität* (als ganze Zahl kodiert) dienen.

```
        In C:                In Java:
typedef struct {        class Kunde {
     char *name ;             char[] name ;
    float azeit ;             float azeit ;
     int prio ;               int prio ;
                         // Methodendefinitonen
   } Kunde ;             }
```

Die wesentlichen Unterschiede der beiden Ansätze liegen darin, dass für Java, als objektorientierte Sprache, die Methoden zur Datenmanipulation und -ausgabe direkt mit der Datenstruktur spezifiziert werden. In C müssen Datenmanipulation und -ausgabe durch Funktionen realisiert werden, die unabhängig von der Datenstruktur spezifiziert werden. Weitere Unterschiede ergeben sich bei der Freispeicherverwaltung.

Ebenso bieten die meisten modernen Programmiersprachen die Funktionalität zur dynamischen Speicherverwaltung. Dazu sind Funktionen zur Allokation eines Speicherbereichs zur Laufzeit notwendig. Dies geschieht in der Regel durch Aufruf einer entsprechenden Funktion oder des Konstruktors eines Objekts. Damit der allokierte Speicher, wenn er nicht mehr benötigt wird, wieder benutzt werden kann, muss er freigegeben werden. Dies geschieht entweder explizit durch Aufruf einer Funktion zur Freigabe oder implizit per *garbage collection*. Letzteres bedeutet, dass von Zeit zu Zeit oder bei Speichermangel automatisch eine Funktion aufgerufen wird, die überprüft, auf welche allokierten Speicherbereiche keine Referenzen mehr existieren. Diese Überprüfung kann relativ aufwändig sein, verhindert aber, dass Speicher zu früh freigegeben wird. Grundsätzlich ist die dynamische Speicherbelegung eine der fehleranfälligsten Teile eines Programms. So kann Speicher nicht oder nicht in ausreichender Größe allokiert worden sein, der allokierte Bereich kann nicht korrekt initialisiert worden sein, die Speicherfreigabe kann zu früh oder gar nicht erfolgen. Diese Probleme sind nur sehr eingeschränkt statisch durch einen Compiler überprüfbar und führen oft zu Fehlern, die zur Laufzeit nur schwer lokalisierbar sind, da sie unter Umständen nicht lokale Effekte haben. Deshalb ist eine weitgehende Benutzerunterstützung notwendig, wie sie z.B. bei der Entwicklung moderner objekt-orientierter Programmiersprachen als grundlegende Anforderung diente. Gleichzeitig muss die Speicherverwaltung aber auch effizient realisiert sein, da sie die Effizienz des Simulationsprogramms stark beeinflusst.

Als einfaches Beispiel sollen die verfügbaren Funktionen bzw. Methoden zur Freispeicherverwaltung in C und Java kurz vorgestellt werden. Wir beginnen mit der Sprache C, die ein sehr niedriges Abstraktionsniveau besitzt. In C gibt es unter anderem die folgenden Funktionen zur Speicherallokation und -freigabe:

```
void *malloc (size_t size) ; // Allokiert size Bytes, ohne Initialisierung
void *calloc (size_t nelem, size_t elsize) ;
          // Allokiert nelem * elsize Bytes und initialisiert diese mit 0
void *realloc (void *prt, size_t size) ;
          // Der Speicherplatz, auf den prt zeigt, wird auf Länge size verlängert/verkürzt
void free (void *prt) ; // Speicherfreigabe des Bereichs, auf den prt zeigt
```

Die Zuweisung des allokierten Speichers an eine Variable erfolgt durch Umformung in den gewünschten Typ, also

```
Kunde *meine_kunden = (Kunde *) calloc(10, sizeof(Kunde)) ;
```

allokiert Speicherplatz für 10 Kunden. Der Zugriff auf den i-ten Kunden erfolgt durch `meine_kunden[i]`. Die Freigabe erfolgt durch `free(meine_kunden)`. C erlaubt eine sehr flexible Handhabung der Speicherverwaltung, da die Sprache nicht streng getypt ist und Adressarithmetik möglich ist. Gleichzeitig bedeutet dies aber auch, dass die Konstrukte extrem fehleranfällig sind und Fehler sehr schwer zu lokalisieren sind.

In Java erfolgt die Allokation durch Aufruf von new, wodurch implizit ein Konstruktor der Klasse aufgerufen wird. Es kann mehrere Konstruktoren geben, die durch unterschiedliche Parameterlisten unterschieden werden. Die folgenden Beispiele zeigen zwei mögliche Konstruktoren für den Kunden.

```
          Kunde {                Kunde (int my_prio, float my_azeit, char[] my_name) {
            prio = 1 ;                 prio = my_prio ;
          azeit = 0.0 ;                azeit = my_azeit ;
      name = "Mustermann" ;            name = my_name ;
    } // ein Konstruktor         } // ein anderer Konstruktor
```

Konstruktoren haben jeweils den Namen der zugehörigen Klasse. C++ kann als objektorientierte Sprache ähnlich wie Java verwendet werden.

Zusammenfassend kann man sagen, dass die Freispeicherverwaltung in C (und auch in C++) sehr effizient und flexibel, aber auch extrem fehleranfällig ist. Die Freispeicherverwaltung in Java ist deutlich weniger effizient, insbesondere das garbage collection ist aufwändig, und auch weniger flexibel. Durch die vorgenommenen Einschränkungen werden viele Fehlermöglichkeiten zwar nicht beseitigt, aber doch deutlich eingeschränkt. Beispiele sind der Zugriff auf nicht allokierten Speicher oder Speicherlecks.

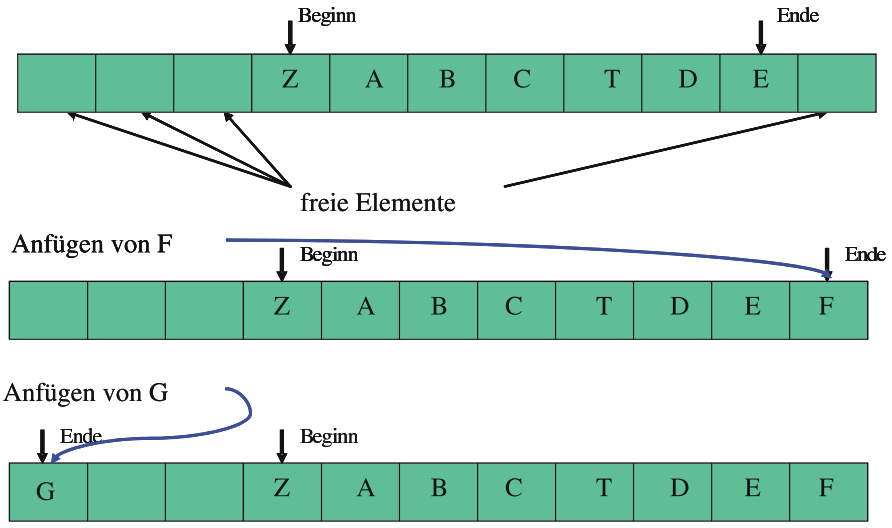

Abbildung 2.15: Ereignisliste als Array realisiert.

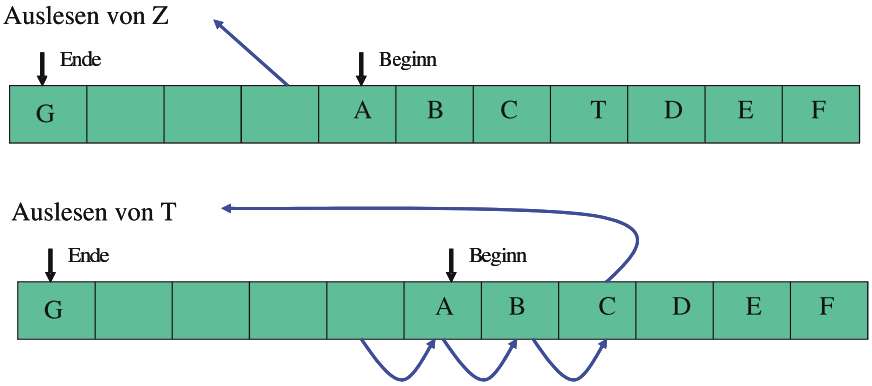

Abbildung 2.16: Ausfügen eines Elements aus dem Array.

Da in der Simulation meistens sehr viele temporäre Objekte benötigt werden, muss auf eine effiziente Realisierung geachtet werden. Gleichzeitig sollte die Allokation und Freigabe von Speicherplatz nicht manuell erfolgen, sondern automatisch durch das System vorgenommen werden. Es sollten also Simulationssprachen oder Werkzeuge benutzt werden, die eine entsprechende Unterstützung bieten.

Datenstrukturen für die Ereignisliste

Als zentrale Datenstruktur der Simulation soll die Ereignisliste näher betrachtet und verschiedene Realisierungsalternativen verglichen werden. Die einfachste Form ist die Realisierung als Array fester Länge. Zwei Variablen dienen dazu das erste und das letzte Element der Ereignisliste zu referenzieren. Sei m die Länge des Arrays und n die Anzahl der Elemente in der Ereignisliste. Abbildung 2.15 zeigt die Struktur dieser Realisierung und das Anfügen von Elementen am Ende. Das Auslesen des ersten oder letzten Elements ist unproblematisch, da nur ein Zugriff erfolgt und ein Variablenwert verändert wird. Der Aufwand für das Aus- oder Einfügen des ersten oder letzten Elements liegt damit in $O(1)$. Problematischer ist das Aus- oder Einfügen eines Elements in der Liste. Durch binäres Suchen kann das gesuchte Element bzw. die Stelle an der das Element eingefügt wird in $O(\lg n)$ gefunden werden. Anschließend müssen allerdings im Mittel $n/2$ Elemente verschoben werden (siehe Abbildung 2.16). Damit beträgt der Aufwand für das Ein- und Ausfügen $O(n)$.

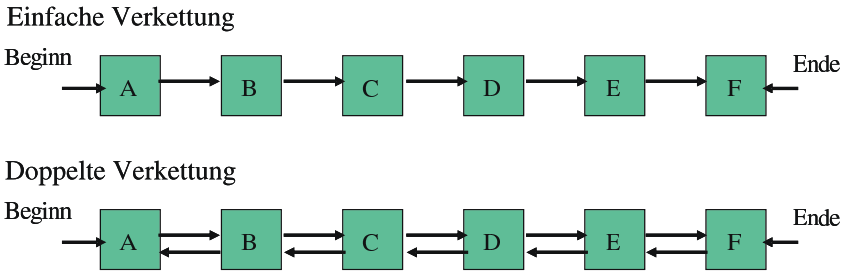

Abbildung 2.17: Ereignisliste als Listenstruktur.

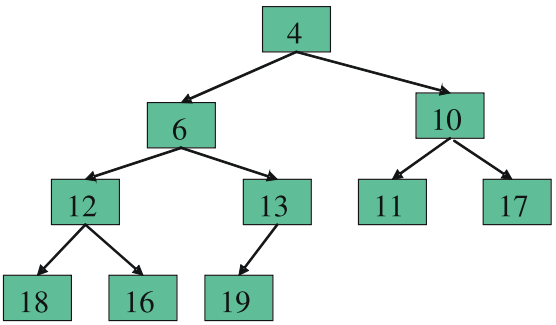

Abbildung 2.18: Ereignisliste als Heap kodiert.

Da man davon ausgehen kann, dass in einer Simulation ungefähr gleich viele Operationen neue Elemente einfügen und das erste Element auslesen und relativ wenige Operationen Elemente aus der Liste ausfügen, werden im Mittel die Hälfte der Operationen mit Aufwand $O(1)$ durchgeführt und der Rest in $O(n)$. Ein weiterer Nachteil der Array-Variante besteht darin, dass nicht mehr als m Elemente in der Ereignisliste sein dürfen. Da die Anzahl der Elemente in der Ereignisliste aber bei den meisten Simulationsprogrammen a priori unbekannt ist, muss ein relativ großer Wert für m gewählt werden und falls $n >$m auftritt, muss entweder das Programm mit Laufzeitfehler abgebrochen werden oder das Array zur Laufzeit vergrößert werden.

Als natürliche Alternative zur einem Array fester Länge bietet sich eine Listenstruktur zur Speicherung der Ereignisliste an. Diese kann als einfach oder doppelt verkettete Liste realisiert werden (siehe Abbildung 2.17). Dabei besteht insbesondere die Möglichkeit auch inhomogene Elemente zu speichern, auch wenn dies für die Ereignisliste nicht notwendig ist, da Ereignisse in der Regel durch einen einfachen Typ referenziert werden. Die Listen werden durch einen Zeiger auf das erste und das letzte Element komplettiert. Der Aufwand für das Auslesen des ersten Elements ist wieder in $O(1)$. Das Einfügen oder Löschen eines Elements erfordert dagegen einen Aufwand in $O(n)$, da nun zwar die eigentliche Einfüge- oder Löschoperation in $O(1)$ durchgeführt wird, das Suchen aber einen linearen Durchlauf durch die Liste notwendig macht. Entscheidender Vorteil der Listenrealisierung ist der Speicherplatzbedarf von $O(n)$, so dass die Länge der Ereignisliste nur durch den physikalisch verfügbaren Speicherplatz beschränkt ist.

In der Informatik gibt es neben der linearen Liste weitere Datenstrukturen, die ein effizienteres Suchen ermöglichen. Ein Beispiel sind die so genannten Heaps. Ein Heap kann als Binärbaum dargestellt werden, dessen Wurzel das minimale Element beinhaltet und in dem kein Vorgänger kleiner als sein Nachfolger ist. Üblicherweise werden Heaps in Arrays gespeichert, was wir hier aber nicht näher betrachten wollen. Die unterste Ebene eines Heaps wird von links gefüllt. Abbildung 2.18 zeigt ein Beispiel für einen Heap, bei dem für die Elemente nur die Werte von t und nicht die Ereignistypen angegeben sind. Auf einem Heap sind die benötigten Operationen wie im Folgenden beschrieben zu realisieren.

- Entfernen der Wurzel: Der Wurzelknoten wird entfernt und der letzte Knoten der unteren

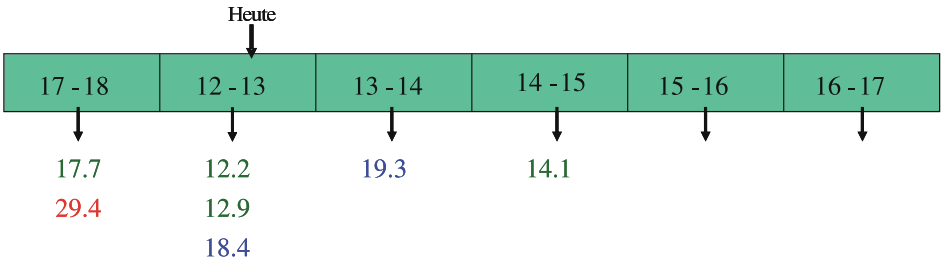

Abbildung 2.19: Ereignisliste als Kalender-Warteschlange.

Ebene wird als neue Wurzel eingesetzt. Anschließend wird die Wurzel so lange mit dem kleinsten Nachfolger vertauscht, bis kein kleinerer Nachfolger mehr existiert.

- Einfügen eines Elements: Das neue Element wird als letztes Element der untersten Ebene eingehängt und anschließend solange mit seinem Vorgänger vertauscht, bis dieser kleiner ist.

Die beiden Operationen können in einem ausgeglichenen Heap mit logarithmischen Aufwand ausgeführt werden. Das Löschen eines beliebigen Schlüssels erfordert dagegen einen linearen Suchaufwand. Es zeigt sich, dass für viele Verteilungen der Ereigniszeiten der Heap mit großer Wahrscheinlichkeit fast ausgeglichen bleibt, so dass aufwändige Operationen zum Ausgleichen nur selten notwendig wären.

Als letzte Datenstruktur für die Ereignisliste soll kurz die Kalender-Warteschlange vorgestellt werden, die sich speziell als Datenstruktur für Ereignislisten eignet. Die Idee entspricht der eines Kalenders, der in Intervalle fester Größe eingeteilt ist. Ein Beispiel ist in Abbildung 2.19 zu sehen. Es liegt also eine Anzahl von Zellen fester Breite vor. Im Beispiel ist die Zeit des nächsten Ereignisses 12.2. Da die Zellenbreite 1 ist, befinden sich alle Ereignisse mit Zeitstempeln aus dem Intervall $[12, 13)$ in der aktuellen Zelle. Ereignisse aus dem Intervall $[13, 14)$ sind in der nachfolgenden Zelle usw. Da im Beispiel 6 Zellen vorhanden sind, umfasst die letzte Zelle das Intervall $[17, 18)$. Tritt nun ein Ereignis mit einem neuen Zeitstempel auf, der größer gleich 18 ist, so wird der Kalender zyklisch benutzt. Man kann dies damit vergleichen, dass in einen Kalender des aktuellen Jahres Ereignisse, die in einem der nächsten Jahre eintreten, für den entsprechenden Tag vorgemerkt werden. Formal sei $T_{\max}$ die rechte Grenze der letzten Zelle, T_{min} die linke Grenze der aktuellen Zelle und Δ die Zellenbreite, dann wird t $(\geq t_{jetzt} \geq T_{min})$ in die Zelle $\lceil((t - T_{min}) \mod (T_{max} - T_{min})/\Delta\rceil$ eingeordnet, wobei die Zellen konsekutiv beginnend mit der aktuellen Zelle nummeriert werden. Innerhalb einer Zelle sind die Elemente linear geordnet. Wenn alle Elemente des aktuellen Intervalls aus einer Zelle ausgelesen wurden, wird solange zur nächsten Zelle übergegangen, bis das kleinste Element in der Zelle einen Zeitstempel im aktuellen Intervall hat. Beim Übergang werden die Grenzen der letzten Zelle jeweils um Δ vergrößert. Im Beispiel würden also die Elemente 12.2 und 12.9 ausgelesen, dann würde zur nächsten Zelle gewechselt und dabei die Grenzen der alten Zelle auf 18 und 19 gesetzt. In der nächsten Zelle befindet sich nur das Element 19.3, das nicht zum aktuellen Intervall gehört. Deshalb werden die Grenzen auf 19 und 20 gesetzt und in der nächsten Zelle nachgeschaut. Dort befindet sich das Element 14.1, welches als nächstes Ereignis abgearbeitet wird.

Es ist einsichtig, dass die Effizienz der Zugriffsoperationen von der Menge der Elemente in einer Zelle und der Anzahl der Zellen ohne Elemente im aktuellen Intervall abhängt. Da diese von der Struktur der Ereigniszeiten abhängen, werden die Parameter der Datenstruktur dynamisch angepasst. Die Anzahl Zellen wird dynamisch angepasst, indem die Zahl verdoppelt wird, wenn die Anzahl der Elemente in der Datenstruktur doppelt so groß wie die Zahl der Zellen ist und sie wird halbiert, wenn die Anzahl der Elemente halb so groß wie die Anzahl der Zellen ist. Der Aufwand der Verdopplung oder Halbierung ist linear in der Anzahl Elemente in der Datenstruktur. Die Zellenbreite Δ sollte so gewählt werden, dass ungefähr 75% aller Einfügeoperationen im aktuellen Jahr stattfinden und damit keinen Überlauf erzeugen. Die Zellenbreite wird dann angepasst, wenn die Zellenzahl modifiziert wird.

Die gesamte Datenstruktur und die zugehörigen Operationen sind relativ komplex, so dass eine Aufwandsanalyse schwierig ist. Insbesondere macht natürlich eine worst case Analyse wenig Sinn. Zahlreiche Experimente zeigen aber, dass sich unter realistischen Bedingungen linearer Aufwand für das Auslesen des nächsten Elements und das Einfügen beliebiger Elemente erreichen lässt.

Gleichwohl zeigen empirische Studien auch, dass die Vorteile komplexerer Datenstrukturen erst bei relativ großen Ereigniszahlen in der Ereignisliste zum Tragen kommen. Für viele, selbst komplexe Simulationsprogramme ist deshalb eine lineare Liste ausreichend.

Operationen zur Realisierung von Simulationsprogrammen

Zur Formulierung der folgenden Simulationsabläufe soll eine Pseudoprogrammiersprache benutzt werden, die sich an gängige Programmiersprachen anlehnt, diesen aber nicht entspricht. Für diese Programmiersprache definieren wir jetzt einige Basisfunktionen zur Simulation, die ähnlich in vielen realen Simulationssprachen existieren. Die Funktion

```
atime = ziehe_zz (Verteilung, Parameter) ;
```

generiert ein Zufallszahl aus einer gegebenen Verteilung mit bekannten Parametern. Die konkrete Realisierung der Funktion wird in Abschnitt 2.3 beschrieben. Weiterhin realisiert

```
plane (Ereignistyp, Ereigniszeit) ;
```

das Einhängen eines Ereignisses vom Typ `Ereignistyp` für den Zeitpunkt `Ereigniszeit` in die Ereignisliste. Dabei spielt es keine Rolle, wie die Ereignisliste realisiert ist. Die Funktion

```
ereignis = erstes_ereignis() ;
```

liefert das nächste Ereignis aus der Ereignisliste und

```
zeit = t ;
```

liefert die Zeit des nächsten Ereignisses.

Struktur von Simulatoren

Mit den vorgestellten Konstrukten können ereignisdiskrete Simulatoren auf dem bisher verwendeten Niveau realisiert werden. Es müssen dazu

- die Datenstrukturen zur Abbildung der statischen Struktur aufgebaut werden und
- die Ereignisroutinen für alle Ereignisse definiert werden.

Den zugehörigen Ansatz bezeichnet man als *event scheduling*. Das Vorgehen beschreibt Modellverhalten auf einem sehr niedrigen Abstraktionsniveau, da

- sämtliche Ereignisse und deren (globalen) Auswirkungen erkannt und kodiert werden müssen und damit
- flache aber stark vernetzte und damit unübersichtliche Modelle entstehen.

Dies bedingt, dass event scheduling für komplexe Probleme sehr aufwändig und damit fehleranfällig ist. Insbesondere entspricht die unstrukturierte Darstellung weder den Prinzipien des modernen Softwareentwurfs noch der üblicherweise verwendeten Methodik zum Entwurf komplexer technischer Systeme. In beiden Fällen wird versucht, durch die Einführung von Struktur die Komplexität beherrschbar zu machen. Damit stellt sich natürlich die Frage, ob dies nicht auch in der Simulation möglich ist.

Da die Basis der Modellierung das mentale oder semi-formale Modell ist, mit dem die Modellierung beginnt, müssen einfachere Simulationsmodelle näher am mentalen Modell sein. In unserer Vorstellung sind komplexe Zusammenhänge strukturiert, sonst wären sie nicht verständlich oder überschaubar. Die Strukturierung wird dadurch unterstützt, dass in praktisch allen realen Systemen Ereignisse im Wesentlichen *lokale* Auswirkungen haben, die nur einzelne Zustandsvariablen betreffen. Damit besteht die Kunst einer übersichtlicheren Modellierung darin, Ereignisse mit ähnlichen lokalen Auswirkungen geeignet zusammenzufassen. Eine solche Zusammenfassung ist natürlich abhängig vom zu untersuchenden System und der Zielstellung der Modellierung, kann

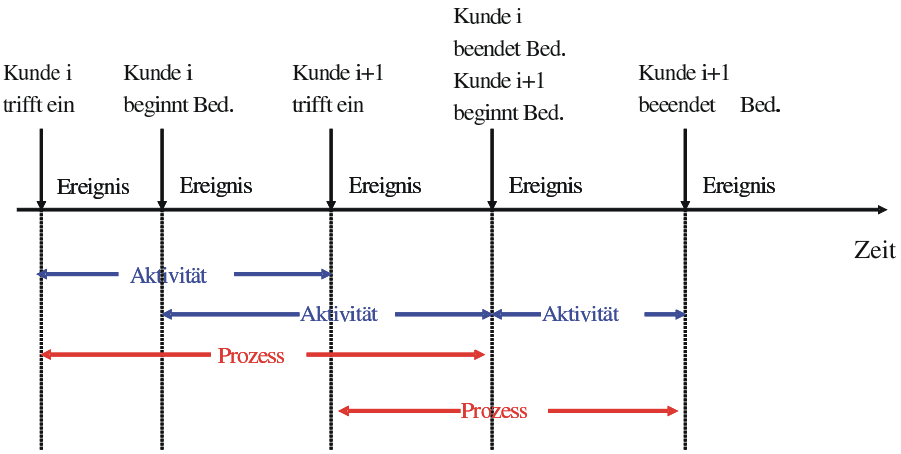

Abbildung 2.20: Unterschiedliche Sichtweisen der Zeitsteuerung.

aber durch geeignete Strukturierungsmechanismen unterstützt werden. In der Literatur existieren zahlreiche Strukturierungsmethoden aber relativ wenig Systematik. Wir betrachten die drei folgenden Ansätze, die recht weit verbreitet sind:

- event scheduling
- activity scanning
- process interaction

Die drei Ansätze entsprechen drei Sichten auf ein System und definieren den zugehörigen konzeptuellen Rahmen der Zeitsteuerung mit steigender Abstraktion. Die Ansätze sollen an einfachen Beispielen eingeführt werden. Dazu betrachten wir zuerst die Abläufe an dem einfachen Schalter. Abbildung 2.20 zeigt einen Ausschnitt aus dem Modellverhalten. Oberhalb der Zeitachse sind die einzelnen Ereignisse eingetragen und der Ablauf in der bisher verwendeten event scheduling-Sichtweise beschrieben. Eine abstraktere Darstellung wird dadurch erreicht, dass zusammengehörige Ereignisse zu *Aktivitäten* zusammengefasst werden. Ein typische Sichtweise wäre die Definition der Aktivitäten Bedienung und Ankunft. Durch die Ankunftsaktivität trifft jeweils ein neuer Kunde ein und die Zeit zwischen dem Eintreffen von Kunden ist gerade die Zwischenankunftszeit. Die Aktivität Bedienung beschreibt den Ablauf vom Beginn der Bedienung bis zu deren Ende. Wie man in Abbildung 2.20 sehen kann, laufen die Aktivitäten Ankunft und Bedienung parallel ab. Eine höhere Abstraktionsstufe erreicht man, wenn man die Ereignisse *handelnder Individuen* zusammenfasst. Im Beispiel sollen Kunden gewählt werden. Wir betrachten also das Leben eines Kunden im Modell. Dieses ist charakterisiert durch drei Ereignisse, nämlich seine Ankunft, seinen Bedienbeginn und sein Bedienende. Man kann schon bei diesem einfachen Beispiel deutliche Unterschiede zwischen den Strukturierungsansätzen erkennen. So gibt es beim event scheduling eine endliche Menge von Ereignissen, beim activity scanning eine endliche Menge von Aktivitäten, während beim process interaction potenziell unendlich viele Kunden gleichzeitig im System sein können. Auch wenn sich im Beispiel die Definition von Aktivitäten und Prozessen sehr natürlich ergab, ist diese keineswegs eindeutig. In einem komplexen Modell gibt es eine Vielzahl von Möglichkeiten Aktivitäten oder Prozesse zu definieren. Je nach Auswahl wird das resultierende Modell übersichtlicher oder komplexer. Letztendlich ist die Strukturierung vom mentalen Modell abhängig und damit auch ein Stück weit subjektiv. Trotzdem gibt es Regeln, wie strukturiert werden sollte.

Bevor die einzelnen Modellierungsansätze an einem etwas komplexeren Modell erläutert werden, sollen die zentralen Begriffe festgelegt werden:

- Ein *Ereignis* ist die Veränderung der Zustandsvariablen zu bestimmten Zeitpunkten.

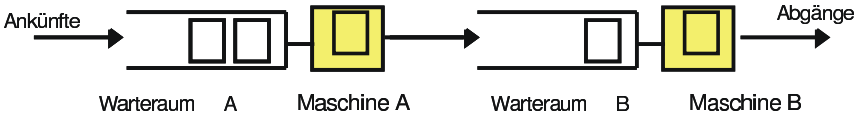

Abbildung 2.21: Bespielmodell mit zwei Stationen.

- Eine *Aktivität* ist eine Menge von Operationen während eines Zeitintervalls.
- Ein *Prozess* ist eine Folge von Aktivitäten eines Objekts über eine Zeitspanne.

Ein zentraler Freiheitsgrad gerade bei der prozessorientierten Modellierung ist die Festlegung aktiver und passiver Elemente. Man kann dies schon am einfachen Schalterbeispiel erläutern. In der bisherigen Sichtweise waren Kunden aktiv und Bediener passiv. Kunden bewegen sich zwischen Bedienern und belegen diese für ihre Bedienzeit. Man kann auch eine andere Sichtweise haben und Bediener als aktiv und Kunden als passiv interpretieren. Dann übernimmt und übergibt ein Bediener einen Kunden und bedient diesen für eine Zeit. Als dritte Möglichkeit können sowohl Kunden als auch Bediener aktiv sein. In diesem Fall müssen sich beide auf eine Bedienung und deren Dauer einigen. Die am Beispiel beschriebenen Möglichkeiten bestehen in allen warteschlangenorientierten Modellen. Man bezeichnet sie als *kunden-* (bzw. *material-*) und *ressourcen-orientierte* Sichtweise. Je nach Wahl der aktiven Elemente lassen sich manche Verhaltensweisen besser modellieren und andere unter Umständen nur mit Schwierigkeiten oder gar nicht modellieren. Gleiches gilt für die Möglichkeiten der Systembeobachtung und Auswertung. Wir werden später noch einmal zu den Vor- und Nachteilen der beiden bzw. drei Ansätze zurückkommen.

Die eingeführten Konzepte der Modellbeschreibung sollen nun an einem etwas komplexeren, aber immer noch sehr elementaren Beispiel detailliert beschrieben werden. Das Beispiel (siehe Abbildung 2.21) besteht aus 2 Stationen, die als Maschinen interpretiert werden können. Werkstücke oder Aufträge (dies waren im vorherigen Beispiel die Kunden) betreten das Modell indem sie den Warteraum der ersten Maschine betreten. Nachdem sie an der ersten Maschine bearbeitet wurden, gehen sie zu Maschine 2 über, werden dort bearbeitet und verlassen danach das System. Die Aufträge werden an den Maschinen nach FCFS bearbeitet und die Puffer besitzen eine potenziell unendliche Kapazität. Der Zeitabstand T_I zwischen zwei Ankünften ist eine Zufallsvariable mit unabhängiger Verteilung. Die Bearbeitungszeiten an den Maschinen T_A und T_B seien ebenfalls Zufallsvariablen mit unabhängigen Verteilungen. Das einfache Modell kann zur Ermittlung unterschiedlicher Ergebnisse simuliert werden. Als Beispiele seien hier genannt:

- Durchlaufzeit von Aufträgen (kundenorientiert)
- Anzahl Aufträge im Warteraum (ressourcenorientiert)
- Auslastung der Maschinen (kostenorientiert)

Modellierung nach dem event scheduling-Ansatz

Die statische Struktur ist charakterisiert durch zwei Maschinen A und B, die als permanent existierende Objekte vorhanden sind. Der Zustand der Maschinen ist *frei* oder *belegt*. Die Warteräume vor jeder Maschine werden als FIFO-Warteschlangen (First In First Out) realisiert. Die Warteschlangen müssen Aufträge aufnehmen. Wir wollen keine detaillierte Definition eines Auftrags geben, man kann aber annehmen, dass Aufträge durch eine entsprechende Datenstruktur beschrieben sind. Im Gegensatz zum einfachen Schalterbeispiel reicht es nicht mehr aus, nur die Anzahl Aufträge im System zu zählen, sondern für jeden Auftrag wird ein neues Auftragsobjekt erzeugt. Aufträge sind damit temporäre Objekte, von denen sich potenziell unendlich viele im Modell befinden können.

Die Dynamik des Modells wird durch die folgenden Ereignisse realisiert:

1. Ankunft an Maschine A (a_A)

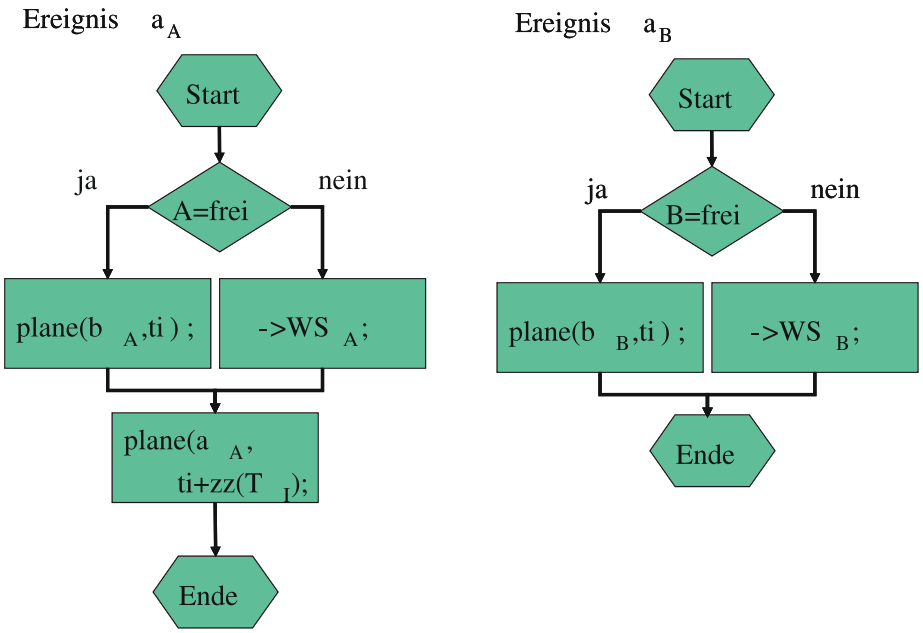

Abbildung 2.22: Ereignsroutine für die Auftragsankunft an Maschine A und B.

2. Ankunft an Maschine B (a_B)
3. Bearbeitungsbeginn an Maschine A (b_A)
4. Bearbeitungsbeginn an Maschine B (b_B)
5. Bearbeitungsende an Maschine A (e_a)
6. Bearbeitungsende an Maschine B (e_B)

Für die einzelnen Ereignisse werden die Ereignisroutinen wieder mit Hilfe von Flussdiagrammen spezifiziert. In den Diagrammen werden die folgenden Abkürzungen verwendet. ti beschreibt die aktuelle Simulationszeit, $-> WS_A$ und $WS_A ->$ beschreiben das Ein- und Ausfügen von Aufträgen in die/aus der Warteschlange vor Maschine A (analog für Maschine B) und $zz(T)$ generiert eine Zufallszahl gemäß T (T beschreibt die Verteilungsfunktion). Abbildung 2.22 zeigt die Ereignisroutinen für die Auftragsankünfte an den beiden Maschinen. Es wird jeweils zuerst getestet, ob die Maschine frei ist. Falls die Maschine frei ist, so kann direkt ein Ereignis Bedienbeginn für die aktuelle Zeit geplant werden. Bei belegter Maschine wird der ankommende Auftrag in die jeweilige Warteschlange einsortiert. In der Ankunftsroutine für Maschine A muss anschließend noch die nächste Ankunft für den Zeitpunkt $ti + zz(T_I)$ geplant werden. Eine solche Planung ist für Maschine B nicht notwendig, da dort Ankünfte durch das Bedienende an Maschine A initiiert werden.

Die restlichen Ereignisroutinen sind in Abbildung 2.23 dargestellt. Beim Bedienbeginn wird jeweils der Zustand der Maschine auf *belegt* gesetzt und anschließend das Ende der Bedienung geplant. Beim Bedienende wird der Zustand der Maschine auf frei gesetzt und dann getestet, ob die Warteschlange noch weitere Aufträge beinhaltet. Falls dies der Fall ist, so wird der nächste Auftrag aus der Warteschlange geholt und sein Bedienbeginn für den aktuellen Zeitpunkt geplant. Bei Maschine A muss zusätzlich noch die Ankunft des Auftrags, der gerade seine Bedienung beendet hat, an Maschine B für den aktuellen Zeitpunkt geplant werden. Es ist dabei zu beachten, dass der Ansatz so nu rfunktioniert, wenn ausgeschlossen werdne kann, dass Bedienende und Ankunft gleichzeitig stattfinden können, da ansonsten zwei gleichzeitige Bedienungen initiiert werden könnten (Ablauf nachvollziehen!).

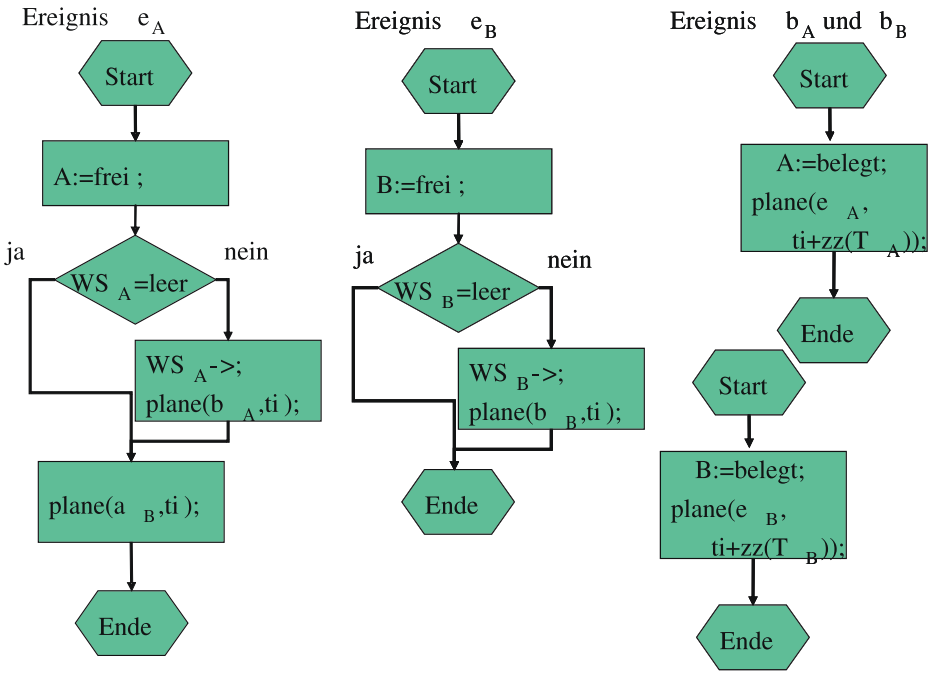

Abbildung 2.23: Restliche Ereignisroutinen für das Beispielmodell.

Offensichtlich beschreibt die vorgenommene Strukturierung der Ereignisse nur eine mögliche Realisierung. Es wäre genauso gut möglich, die Ereignisse Bedienbeginn an Maschine A und B mit in den Ereignissen Ankunft und Bedienende zu kodieren, wie es für den einfachen Schalter gemacht wurde.

Zum Start der Simulation müssen die Variablen initialisiert werden. Naheliegend ist es $ti := 0$ zu wählen, die Warteschlangen leer zu initialisieren und die Maschinen auf frei zu setzen. Wenn zu Beginn die erste Ankunft an Maschine A für den Zeitpunkt $zz(T_I)$ eingeplant wird, dann läuft die Simulation von da an weiter. Zur Beendigung der Simulation kann entweder ein Ereignis Simulationsende zu Beginn für einen vorgegebenen Zeitpunkt in die Ereignisliste per $plane(Simulationsend, T_{Ende})$ eingehängt werden oder die Simulation kann durch andere Bedingungen abgebrochen werden. Z.B. könnte nach einer bestimmten Anzahl Bedienungen abgebrochen werden. In diesem Fall müsste in der Ereignisroutine für das Bedienende an Maschine B nachgehalten werden, wie viele Aufträge bereits ihre Bedienung beendet haben und falls die gewünschte Anzahl erreicht wurde, muss für den aktuellen Zeitpunkt ein Ereignis Simulationsende eingeplant werden.

Damit die Simulation nicht spurlos abläuft, müssen Daten aufgezeichnet und am Ende ausgewertet werden. Wie bereits erwähnt, ist dies Teil der Aufgaben der Ereignisroutinen. Dieser Teil wurde aus Übersichtlichkeitsgründen in den Beispielroutinen weggelassen. Es soll kurz das Konzept zur Ergebnisermittlung skizziert werden. Man unterscheidet dabei zwischen auftrags- und stationsspezifischen Maßen. Auftragsspezifische Maße wie Verweilzeiten oder Wartezeiten müssen in der Auftragsdatenstruktur gespeichert werden. Dazu sind entsprechende Variablen vorzusehen. Dies soll kurz am Beispiel der Wartezeiten erläutert werden. Die Wartezeit jedes Auftrags soll für jeden Auftrag und beide Maschinen ermittelt werden. Dazu sind zwei reelle Variablen t_in und t_wait notwendig, die für jeden Auftrag mit 0 initialisiert werden. Bei Ankunft des Auftrags (d.h. in der zugehörigen Ereignisroutine) wird die Ankunftszeit t_in auf die aktuelle Zeit ti gesetzt. Bei Bedienbeginn wird zu t_wait die Differenz $ti - t\_in$ (d.h. die Wartezeit) addiert. Bei Bedienbeginn an Maschine B muss anschließend noch der Wert von t_wait zur Auswertung gespeichert werden. Dies geschieht durch Aufruf der entsprechenden Funktion oder Methode. Wie die einzelnen Werte gespeichert werden hängt vom Ziel der Auswertung ab (z.B. nur Mittelwertermittlung, Ermittlung

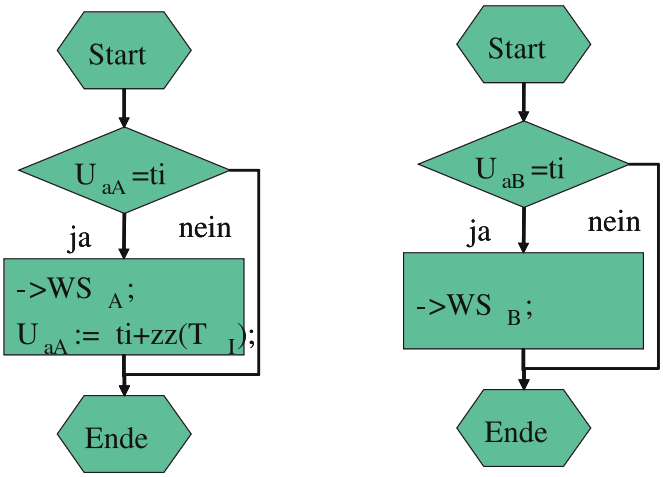

Abbildung 2.24: Aktivitätsroutinen für Auftragsankünfte.

von Quantilen etc.). Bei stationsspezifischen Maßen kann man davon ausgehen, dass die Station für einen spezifischen Zeitraum in einem Zustand ist (z.B. frei-belegt, mit x Aufträgen). Der Zustandswert und der Zeitraum müssen gespeichert werden. Für eine Mittelwertbestimmung kann der bereits für den einfachen Schalter beschriebene Ansatz verwendet werden.

Modellierung nach dem activity scanning-Ansatz

Das nächst höhere Abstraktionsniveau ist der activity scanning-Ansatz. Es wird von einer Menge von Aktivitäten ausgegangen, die im Gegensatz zu Ereignissen wissen, wann sie beginnen oder enden können. Aktivitäten entscheiden also lokal, ob sie eintreten oder nicht. In der Simulation werden nach jeder Zustandsänderung, d.h. nach Ausführung einer Aktivität, alle Aktivitäten angestoßen und jede entscheidet lokal für sich, ob sie eintreten kann oder nicht. Die Implementierungsidee besteht in einer Menge lokaler Aktivitätsuhren. Bei Aktivierung testet eine Aktivität, ob die eigene Uhr abgelaufen ist. Im Unterschied zum event scheduling, bei dem alle Folgeereignisse von einem Ereignis generiert werden müssen, werden beim activity scanning nur die unmittelbaren Folgen des Ereignisses beschrieben, da Folgeereignisse selbst testen, ob sie ausgeführt werden müssen.

Bei der Realisierung stehen die Uhren einer Aktivität auf der lokalen Aktivitätszeit und die jeweils nächste lokale Aktivitätszeit tritt ein. Wie beim event scheduling ist bei identischen Aktivitätszeiten keine Reihenfolge vorgegeben und das Modellverhalten hängt von der Reihenfolge ab, in der Aktivitäten angestoßen werden. Der Vorteil des activity scanning ist eine kompaktere Beschreibung und damit ein höheres Abstraktionsniveau. Gleichzeitig ergibt sich auch ein Nachteil, nämlich oft eine mangelnde Effizienz. In Modellen mit vielen Aktivitäten werden Aktivitäten oft angestoßen, obwohl sie nicht ausführbar sind. Im Programm bedeutet dies in der Regel jeweils einen Funktions- und Methodenaufruf, der Zeit benötigt, auch wenn der eigentliche Test auf Ausführbarkeit sehr effizient realisiert ist.

Wir betrachten nun das einfache Beispiel im activity scanning Ansatz. U_{xY} wird zur Bezeichnung der Uhr für Aktivitätsroutine x_Y verwendet. Die lokale Uhr wird in der zugehörigen Aktivitätsroutine gesetzt und gelesen. Ein globaler Prozess springt jeweils zur nächsten Uhrzeit. Ähnlich zu den Ereignissen wird das Modellverhalten durch 6 Aktivitäten und die zugehörigen Aktivitätsroutinen beschrieben. Auch in den Aktivitätsroutinen wird die Aufzeichnung von Resultaten nicht betrachtet. Diese funktioniert genauso wie im event scheduling-Ansatz. Abbildung 2.24 zeigt die Aktivitätsroutinen für die Auftragsankunft an Maschine A und B. Die einzige Bedingung für das Eintreten ist, dass die lokale Uhr auf der aktuellen Zeit steht. Ansonsten sind die Aktivitätsroutinen recht ähnlich zu den Ereignisroutinen, was an den relativ einfachen Ereignisstrukturen liegt, die im Wesentlichen nur lokale Änderungen hervorrufen.

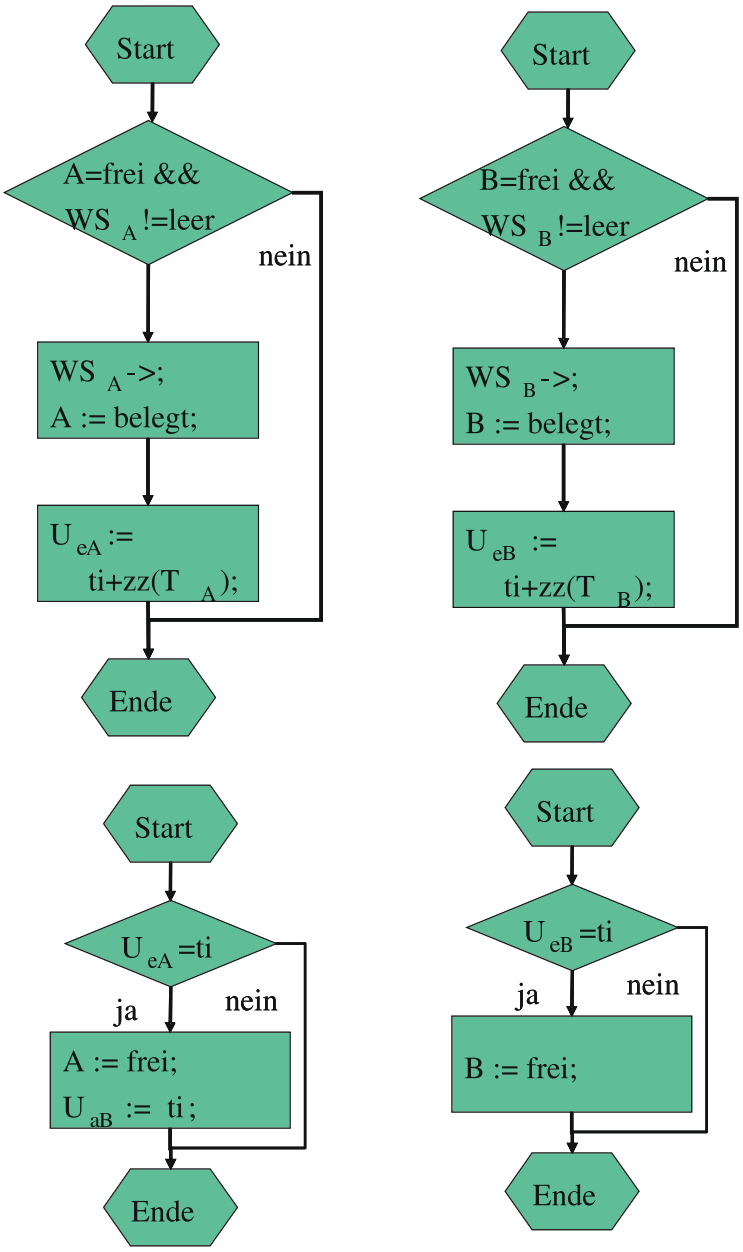

Abbildung 2.25: Restliche Aktivitätsroutinen für das Beispiel.

Abbildung 2.25 zeigt die restlichen Aktivitätsroutinen. Bei den Aktivitätsroutinen für den Bedienbeginn wurde auf die explizite Angabe der lokalen Uhren verzichtet, da das Abprüfen, ob die Maschine frei ist und die Warteschlange nicht leer ist ausreicht. Die Synchronisation zwischen den Aktivitäten findet zum Teil dadurch statt, dass eine Aktivität die Uhr einer anderen Aktivität setzt.

Da das Problem des vielfachen Aufrufs von Aktivitätsroutinen gerade bei komplexeren Modellen auftaucht, ist man bestrebt den Ansatz effizienter zu realisieren. Eine solche effizientere Realisierung ist der folgende 3 Phasen-Ansatz, der zwischen B- und C-Aktivitäten unterscheidet. B-Aktivitäten sind solche, deren Vorbedingungen immer erfüllt sind, bei denen also nur die lokale Uhrzeit eine Rolle spielt. Im Beispiel wären dies Auftragsankunft und Bedienende. B-Aktivitäten werden in einer Liste nach Eintretenszeit geordnet gespeichert. Die restlichen Aktivitäten werden als C-Aktivitäten bezeichnet. Die Simulation läuft dann wie folgt ab:

1. Zeitfortschreibung
2. Ausführung der B-Aktivitäten, deren Uhren auf der aktuellen Zeit stehen (dazu muss maximal ein Test durchgeführt werden, bei dem keine Aktivität ausgeführt wird).
3. Test und evtl. Ausführung der C-Aktivitäten.

Durch diese Unterscheidung werden B-Aktivitäten praktisch zu Ereignissen und die Unterschiede zwischen event scheduling und activity scanning verschwimmen. Im Gegensatz zum event scheduling-Ansatz, der von mehreren Simulationssprachen unterstützt wird, gibt es kaum Sprachen, die activity scanning unterstützen. Trotzdem beschreibt der Ansatz einen wichtigen Abstraktionsschritt.

Modellierung nach dem process interaction-Ansatz

Die bisher vorgestellte Abstraktionsebene ist für viele Anwendungen zu niedrig und entspricht auch nicht dem in der Informatik heute üblichen Abstraktionsniveau. Es gibt dort und in vielen anderen Ingenieursdisziplinen den Trend zum Denken und Strukturieren in Prozessen. Dieser Ansatz wird in der Simulation seit langem verwendet, was auch daran deutlich wird, dass die Simulationssprache Simula67 als erste objektorientierte Sprache mit einem Koroutinenkonzepz vor fast 40 Jahren entwickelt wurde.

Das Strukturieren in Prozessen soll nun am Beispiel entwickelt werden. Man hat dabei zwei grundsätzliche Möglichkeiten Prozesse zu definieren. Die erste Möglichkeit ist die Sichtweise der Aufträge als Prozesse. Das Leben eines Auftrages im Modell ist gekennzeichnet durch die folgenden 6 Ereignisse.
Jeder Auftrag

- trifft an Maschine A ein und stellt sich an,
- kommt dran,
- ist fertig und geht weg,
- trifft an Maschine B ein und stellt sich an,
- kommt dran,
- ist fertig und geht weg.

Damit überhaupt Aufträge in das System gelangen, müssen diese generiert werden. Dies kann einmal durch die Aufträge selbst geschehen oder durch einen separaten Prozess. Hier soll die zweite Möglichkeit betrachtet werden, auch wenn später meistens die erste Möglichkeit verwendet werden wird. Die Prozess Umwelt zur Auftragsgenerierung ist durch die folgende Sequenz realisiert

- schickt einen Auftrag in das System,

- schickt später einen weiteren Auftrag in das System,
- ...

Die Strukturierung in Aufträge ist offensichtlich nur eine Möglichkeit der Strukturierung, nämlich genau die, die dem bisherigen Ansatz entspricht, der auch beim event scheduling oder activity scanning eingesetzt wurde. Alternativ können Maschinen und nicht Aufträge als Prozesse angesehen werden. Auch in diesem Fall wird neben den beiden Prozessen für die Maschinen ein Prozess Umwelt benötigt, der die Maschinen mit Aufträgen versorgt. Der Ablauf eines Prozesses zur Modellierung einer Maschine ist durch die folgende Sequenz beschrieben. Die Maschine

- beginnt mit einer Bedienung,
- beendet diese später,
- beginnt die nächste Bedienung,
- ...

Jeder der Schritte der Maschinen und der Umwelt kann als Ereignis angesehen werden und sowohl Maschine als auch Umwelt können eine potenziell unendliche Anzahl von Ereignissen ausführen. Im Gegensatz zur Strukturierung nach Auftragsprozessen, in dem eine potenziell unendliche Zahl von Prozessen generiert und prinzipiell auch gleichzeitig im System sein kann, benötigt der zweite Ansatz nur 3 Prozesse. Aufträge sind *temporäre Prozesse*, da sie während der Simulation generiert werden und das System auch wieder verlassen. Umwelt und Maschinen sind *permanente Prozesse*, die zu Anfang generiert werden und während der gesamten Simulationszeit im System sind.

Nach dieser kurzen informellen Einführung soll der Ansatz weiter formalisiert werden. Ziel ist es, "Objekt-Leben" als Strukturierungsmittel zu nutzen, womit die Einzelereignisse Unterstukturen werden. Damit ist ein dynamisches System beschrieben durch eine Menge dynamischer Objekte (Prozesse), die sich entsprechend festgelegter Vorschriften (Prozessmuster) verhalten. Es liegt also ein *System paralleler Prozesse* vor. Dieser Ansatz liegt der Sichtweise der Informatik sehr nahe, da diese sich an verschiedenen Stellen mit parallelen Prozessen beschäftigt und diese als Strukturierungsmittel einsetzt. Die Sichtweise kann aber auch auf natürliche Weise auf viele andere Gebiete übertragen werden.

Prozesse laufen unabhängig, müssen aber von Zeit zu Zeit interagieren. Im Beispiel tritt eine solche Interaktion beim "Bearbeitetwerden" im Auftrags-Prozess und beim "Bearbeiten" im Maschinen-Prozess auf. Also *Prozesse müssen sich synchronisieren*, d.h. gewisse Aktionen gleichzeitig ausführen oder nacheinander ausführen. Um ein prozessorientiertes Modell zu beschreiben müssen die folgenden Bedingungen erfüllt sein.

1. Prozessmuster müssen notiert werden.
2. Prozesse bestimmter Muster müssen gestartet werden.
3. Prozesse müssen interagieren können.

Ein Prozess läuft nach folgendem Muster ab

- Operation(en) ausführen
- Warten (d.h. nichts tun, Zeit verstreichen lassen)
- Operation(en) ausführen
- ...

Für das Beispiel Auftrag existiert folgende Teilsequenz

- Stell dich an

- Warte bis du dran bist (1)
- Warte bis Bedienzeit abgelaufen (2)

(1) und (2) beschreiben mögliche Wartebedingungen und sie beschreiben abstrakt gesehen sogar alle möglichen Wartebedingungen:

(1) Es wird auf eine Bedingung gewartet. Der Prozess kann erst fortfahren, wenn diese Bedingung erfüllt ist.

(2) Es wird bis zu einem Zeitpunkt gewartet. Der Prozess kann fortfahren, wenn die Simulationszeit bis zum Ende der Wartezeit fortgeschritten ist.

Um diese beiden Wartebedingungen auszudrücken definieren wir die beiden folgenden Befehle für unsere Pseudoprogrammiersprache.

- wait_until (Boolsche-Bedingung)
  - Formuliert Bedingung (1).
  - Der Prozess wird inaktiv und bleibt so lange inaktiv, bis die Bedingung erfüllt ist.
  - Die Bedingung wird durch Zustandsänderungen erfüllt. Damit kann sie nur durch Ereignisse in anderen Prozessen erfüllt werden.
- pause_for (t)
  - Formuliert Bedingung (2).
  - Der Prozess wird für die Dauer t inaktiv.
  - Nach Ablauf der Zeit wird er von selbst wieder aktiv.

In beiden Fällen fährt der Prozess, nachdem er wieder aktiv wird, dort fort wo er gewartet hat, also nach der wait_until- oder pause_for-Anweisung.

Mit den beiden Anweisungen können die Prozesse spezifiziert werden. Wie bisher werden dazu Flussdiagramme verwendet. Abbildung 2.26 zeigt das Prozessmuster für einen Auftrag. Jeder Auftrag durchläuft beide Maschinen und fasst alle dazu notwendigen Ereignisse zusammen. Die Maschinen werden in diesem Ansatz durch zwei Boolsche-Variablen A und B und durch zwei Warteschlangen WS_A und WS_B realisiert. Wenn ein Auftrag an einer belegten Maschine eintrifft, so ordnet er sich selbst in die Warteschlange ein und wartet anschließend bis er der erste in der Warteschlange ist (1. WS) und die Maschine frei ist. Diese Bedingung kann nur erfüllt werden, wenn alle Aufträge vor ihm ihre Bedienung abgeschlossen haben und die Maschine nach Bedienende auf frei gesetzt haben. Das Vorrücken in der Warteschlange geschieht automatisch, was mit einer entsprechenden Datenstruktur (z.B. Liste) einfach realisiert werden kann. Um eine Bedienung auszuführen belegt ein Prozess die Maschine, pausiert für seine Bedienzeit und gibt anschließend die Maschine wieder frei. Wie schon in den vorherigen Beispielen wurden die Operationen zur Simulationsbeobachtung weggelassen. Genauso wie in den anderen Beispielen muss die Systembeobachtung als Teil des Prozessmusters notiert werden.

Abbildung 2.27 zeigt den Prozess Umwelt, der eine sehr einfache Struktur hat. Im Gegensatz zu den Auftragsprozessen gibt es im Umweltprozess kein Ende, sondern nur einen Start, da der Prozess permanent läuft. Zum Starten der Simulation wird ein Umwelt Prozess generiert, der dann die Auftragsprozesse generiert, die sich mit Hilfe der Boolschen-Variablen und der Warteschlangen an den Maschinen synchronisieren.

Nach der Darstellung des Beispiels als prozessorientiertes Modell mit aktiven Aufträgen, soll nun als Alternative die prozessorientierte Sichtweise mit aktiven Maschinen vorgestellt werden. Die Abbildungen 2.28 und 2.29 zeigen die drei benötigten Prozesse zur Modellierung der Umwelt und der beiden Maschinen. Alle Prozesse laufen in dieser Darstellung permanent. Aufträge werden durch den Umweltprozess erzeugt und in den Puffer der ersten Maschine abgelegt. Falls

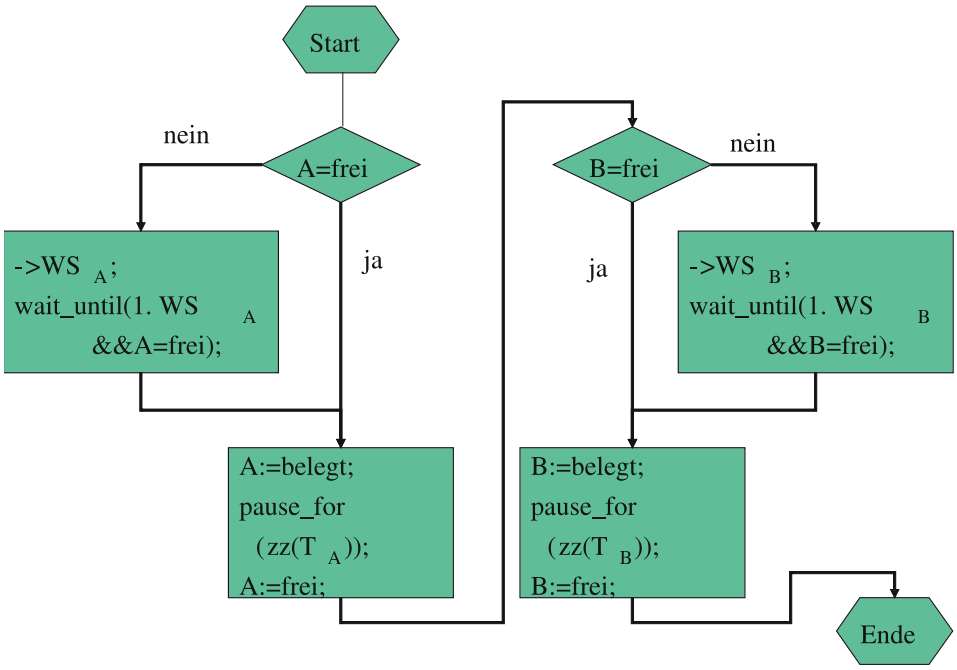

Abbildung 2.26: Prozessmuster eines Auftrags.

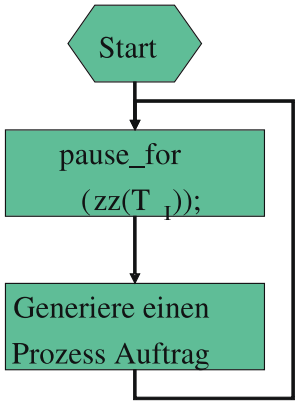

Abbildung 2.27: Umweltprozess für die auftragsorientierte Modellierung.

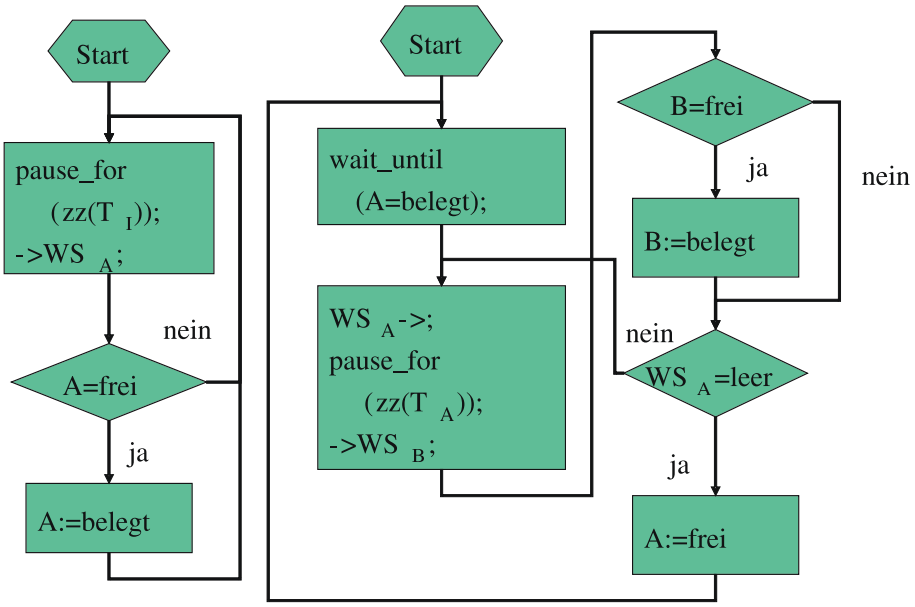

Abbildung 2.28: Umweltprozess und Prozess für Maschine A in der maschinenorientierten Sicht.

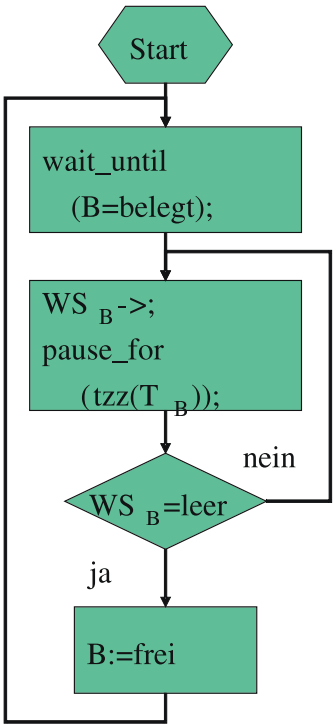

Abbildung 2.29: Prozess für Maschine B.

die erste Maschine frei ist, wird sie durch den Umweltprozess bei Ankunft eines Kunden dadurch aktiviert, dass sie belegt wird. Dadurch wird die wait_until Bedingung der Maschine erfüllt und die Bedienung beginnt. Falls ein Auftrag bedient wurde und noch andere in der Warteschlange vorhanden sind, so beginnt sofort die nächste Bedienung. Die erste Maschine übergibt Aufträge der zweiten Maschine und aktiviert diese, falls nötig. Durch die Erzeugung der drei Prozesse läuft die Simulation an, da der Umwelt-Prozess die Aufträge nacheinander generiert. Aufträge tauchen in dieser Modellierung nur implizit auf, könnten aber durchaus als komplexere Datenstruktur zur Speicherung von Parametern dargestellt werden. Die Warteschlangen vor den Maschinen müssen nur die entsprechenden Datenstrukturen aufnehmen können. Wie schon in den vorangegangenen Beschreibungen wurden die Teile zur Resultatauswertung in den Beschreibungen weggelassen.

Bewertung der material- und maschinenorientierter Sicht

Grundsätzlich können Modelle material- oder maschinenorientiert spezifiziert werden, die Art der Beschreibung hat aber Auswirkungen auf die Ausdrucksmöglichkeiten. In einer maschinenorientierten Sicht können die folgenden Aspekte auf sehr natürliche Weise beschrieben werden.

- Unterschiedliche Scheduling-Strategien, bei denen die Maschine Aufträge nach ihren Eigenschaften auswählt. So könnte der Auftrag mit der geringsten Bearbeitungszeit im zuerst gewählt werden. Diese Strategie nennt shortest job first (SJF). Aufträge könnten in Klassen eingeteilt werden und jede Klasse eine bestimmte Priorität bekommen, wobei Aufträge höherer Priorität vor Aufträgen niedriger Priorität bedient werden. Die Implementierung der Scheduling-Strategie im Auftragsprozess würde sehr komplex, da sich alle auf eine Maschine wartenden Aufträge synchronisieren müssten.
- Maschineorientierte Leistungsmaße sind im Maschinenprozess ebenfalls auf natürlich Weise auswertbar. Beispiele für solche Leistungsgrößen sind die Auslastung der Maschine oder die mittlere Population im Puffer.
- In der Realität fallen Maschinen aus oder bringen auf Grund von temporären Fehlern kurzzeitig eine geringere Leistung. Auch dieses Verhalten ist im Maschinenprozess einfach beschreibbar, kann aber nur sehr aufwändig und unnatürlich im Auftragsprozess beschrieben werden.

Für andere Abläufe ist die materialorientierte Sichtweise die natürliche Beschreibungsform.

- Wenn Aufträge mehrere Maschinen gleichzeitig belegen, so kann dies im Auftrag einfach spezifiziert werden, würde aber auf Maschinenebene eine komplexe Synchronisation erfordern.
- Materialorientierte Leistungsmaße wie Durchlaufzeit sind auf Ebene der Maschine kaum analysierbar, können aber im Auftragsprozess einfach akkumuliert werden.
- Wenn Material synchronisiert wird, so kann dies auf Maschinenebene nur in einfachen Fällen beschrieben werden. Komplexere Synchronisationen erfordern Prozesse für die einzelnen Materialien.

Es zeigt sich deutlich, dass je nach Modellstruktur die eine oder die andere Sichtweise vorzuziehen ist. In vielen komplexen Modellen treten Strukturen auf, die beides, eine material- und eine maschinenorientierte Sichtweise erfordern. Prinzipiell ist eine solche Beschreibung möglich und wird auch von Simulationswerkzeugen unterstützt. Es muss nur darauf geachtet werden, dass die Beschreibung konsistent bleibt. So kann die Bedienzeit vom zu Bedienenden und auch vom Bediener festgelegt. Es ist aber nicht möglich, dass beide jeweils eine eigene Bedienzeit festlegen, da der Bediener genau so lange belegt sein muss, wie der zu Bedienende bedient wird.

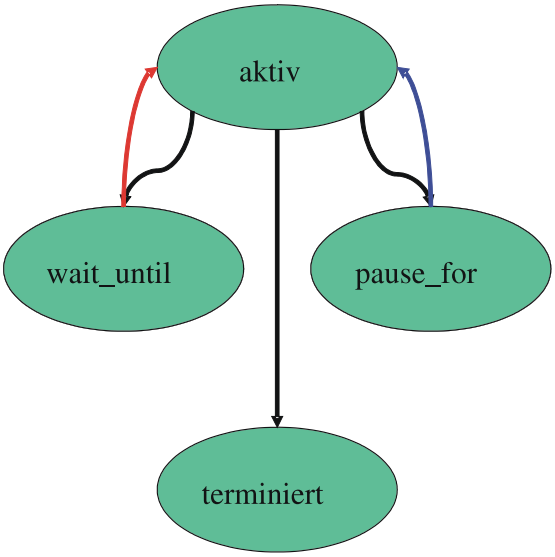

Abbildung 2.30: Prozesszustandsdiagramm der prozessorientierten Simulation.

Realisierung des process interaction-Ansatzes

Der generelle Ablauf prozessorientierter Simulationen und deren Realisierung soll nun näher erläutert werden. Die einzelnen Prozesse können in vier verschiedenen Zuständen sein, wie in Abbildung 2.30 gezeigt wird. Prozesse wechseln zwischen aktiv und einer der beiden Wartebedingungen wait_until oder pause_for. Schließlich terminiert ein Prozess und kann danach nicht wieder aktiv werden. In einem Simulationsmodell können mehrere Prozesse gleichzeitig im Zustand aktiv sein. Da wir davon ausgehen, dass unsere Simulation auf einem Einzelprozessor abläuft, kann von den aktiven Prozessen nur einer zu einer Zeit wirklich bearbeitet werden. Die zeitgerechte Ausführung zeitgleich aktiver Prozesse ist ein zentraler Aspekt der prozessorientierten Simulation. Bevor dieser Punkt näher untersucht wird soll die Frage beantwortet werden, wie Prozesse in einer Programmiersprache realisiert werden. Nahe liegend wäre eine Beschreibung als Prozeduren und Funktionen, wie sie in allen Programmiersprachen vorhanden sind. Prozessmuster und -abläufe lassen sich in Prozeduren grundsätzlich beschreiben. Wenn man allerdings etwas näher darüber nachdenkt, so wird schnell deutlich, dass Prozeduren nicht ausreichen. Das Problem wird in Abbildung 2.31 verdeutlicht. Eine Prozedur würde beim Eintreten einer Wartebedingung verlassen und beim nächsten Aufruf, d.h. beim nächsten Wechsel in den Zustand aktiv, wieder von vorn beginnen. Dies ist offensichtlich nicht das gewünschte Verhalten, sondern bei der nächsten Aktivierung soll nach der Wartebedingung weitergemacht werden. Dieses Verhalten wird durch Koroutinen oder Threads beschrieben, die in manchen Programmiersprachen vorhanden sind. So besitzt Simula67 ein Koroutinenkonzept und Java ein Thread-Konzept. C oder C++ hat standardmäßig kein solches Konzept, es gibt allerdings Zusatzbibliotheken zur Realisierung von Threads und/oder Koroutinen.

Kommen wir nun zur Ausführung gleichzeitiger Ereignisse zurück. Beim event scheduling-Ansatz wurde davon ausgegangen, dass gleichzeitige Ereignisse in beliebiger Reihenfolge ausgeführt werden können. Dies ist im process interaction-Ansatz nicht möglich, wie das folgende Beispiel zeigt. Wir betrachten dazu wieder einen Schalter und einen Kunden. Der Schalter besitzt eine Warteschlange *WS* für Kunden, eine Boolsche Variable belegt und eine Referenz *Kd* auf einen Kunden, der gerade bedient wird. Der Kunde besitzt eine Boolsche Variable *dran*, eine Referenz *Sc* auf den Schalter und eine Referenz *Current* auf sich selbst. Schalter und Kundenprozess sind wie folgt realisiert.

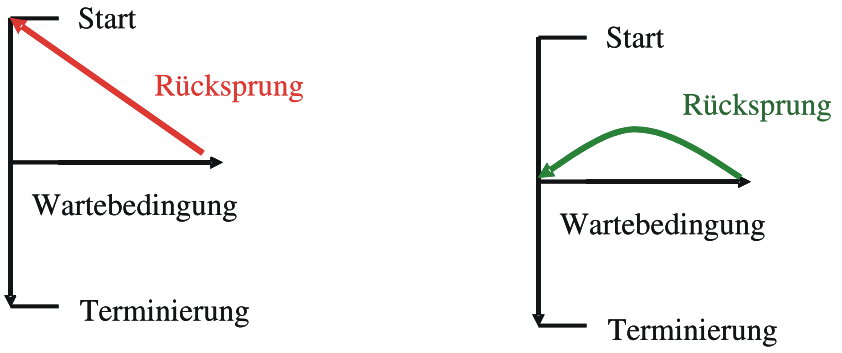

Abbildung 2.31: Verhalten von Prozeduren und Koroutinen.

Schalter		Kunde	
Repeat			
wait_until(WS nicht leer) ;	(S1)	dran := false ;	(K1)
Repeat		Current->Sc.Ws ;	(K2)
Kd := WS.first ;	(S2)	wait_until (dran) ;	(K3)
belegt := true ;	(S3)	pause_for (Bedienzeit) ;	(K4)
wait_until (not belegt) ;	(S4)	Sc.belegt := false ;	(K5)
Until WS leer ;	(S5)	Sc.WS-> ;	(K6)
Until (Simulationsende) ;			

Die folgende Festlegung der Semantik würde dem bisherigen Vorgehen im event scheduling-Ansatz entsprechen.

- Die Reihenfolge der Ereignisse eines Prozesses muss beachtet werden.
- Zeitgleiche Ereignisse in unterschiedlichen Prozessen können beliebig angeordnet werden.

Ein möglicher Ablauf, der mit der Ankunft des ersten Kunden startet wäre der folgende:

K1, K2, S1, S2, S3, S4, K3, K4, K5, S5, S2, S3, S4, K6

Nach Abarbeitung dieser Schritte hätte der Kunde das System verlassen, wäre also terminiert. Der Schalter hätte eine Referenz auf den Kunden, der das System gerade verlassen hat und würde nun warten, dass der Kunde seine Bedienung beendet. Wenn nun nachfolgende Kunden eintreffen, würden sie sich selbst in die Warteschlange einordnen und warten, dass der Schalter ihren Bedienbeginn initiiert. Dies würde aber nicht passieren, da der Schalter auf das Bedienende des Kunden wartet, der bereits weggegangen ist. Offensichtlich liegt eine klassische Deadlock-Situation vor.

Was ist bei dem Ablauf falsch gelaufen? Der Grund liegt darin, dass die Atomizität der Ereignisse nicht beachtet wurde. Wie im event scheduling festgelegt wurde, müssen Ereignisse ganz oder gar nicht ausgeführt werden. Nun existieren im process interaction keine Ereignisse im ursprünglichen Sinne. Trotzdem sind unter Umständen mehrere Zuweisungen notwendig, um den Systemzustand zu ändern. Der Zustand ist undefiniert oder nicht vollständig definiert, solange nicht alle notwendigen Zuweisungen ausgeführt wurden. Im obigen Ablauf hatte der Kunde K6 noch nicht ausgeführt und der Zustand war nicht definiert, als der Schalter S2 ausführte.

Um solche Probleme zu vermeiden, werden in allen prozessorientierten Programmiersprachen die folgenden Vereinbarungen getroffen:

- Ein Prozess läuft solange, bis er eine hemmende Bedingung erreicht oder terminiert.
- Die Auswahl des nächsten Prozesses aus einer Menge aktiver Prozesse bleibt willkürlich und kann das Verhalten des Modells und damit die Semantik beeinflussen.

Kommen wir nun zur Realisierung der beiden Wartebedingungen:

- pause_for (t)
  - Der Prozess pausiert für t Zeiteinheiten und wird danach wieder aktiv.
  - Das Ereignis der Aktivierung kann damit für den Zeitpunkt $ti + t$ eingeplant werden.
  - Die Realisierung erfolgt über eine Ereignisliste, die alle Prozesse enthält, die pausieren.
- wait_until (b)
  - Bedingung b kann über beliebige Zustandsvariablen formuliert werden.
  - Damit kann jede Wertzuweisung jedes wait_until eines Wartenden ändern.
  - Damit muss nach jedem Anhalten eines Prozesses jede Wartebedingung überprüft werden.
  - Dies ist extrem aufwändig und führt zu ineffizienten Simulatoren.

Aus Effizienzgründen ist wait_until in allgemeiner Form in kaum einer Simulationssprache oder einem Simulationswerkzeug realisiert. Trotzdem wird neben dem reinen Warten für einen Zeitraum, ein Warten auf Bedingungen benötigt, da sonst keine wirkliche Synchronisation zwischen Prozessen möglich ist. Es gibt zwei unterschiedliche Möglichkeiten das ineffiziente wait_until zu ersetzen.

Man kann näher zur Implementierung gehen und damit weiter weg vom mentalen Modell. In diesem Fall werden zwei Anweisungen *passivate* und *activate* eingeführt. Der Aufruf von passivate sorgt davor, dass der aufrufende Prozess anhält und damit praktisch in den Zustand von wait_until gerät. Typischerweise wird erst die Bedingung b getestet und falls diese nicht gilt, wird der Prozess passivate aufrufen.
if not b then passivate ;
Im Gegensatz zu wait_until wird der Prozess nicht automatisch aufgeweckt, wenn b gilt. Sondern das Aufwecken muss explizit durch den Aufruf von activate(P) für einen wartenden Prozess P erfolgen. Der Aufruf kann natürlich nur durch einen anderen Prozess erfolgen, idealerweise durch den Prozess, der b erfüllt hat. Diese Art der Beschreibung sorgt für effiziente Simulatoren, bedingt aber auch, dass der Simulator-Code länger und komplexer wird. So muss der aufweckende Prozess eine Referenz auf den aufzuweckenden Prozess besitzen und der eigentlich Vorteil der prozessorientierten Simulation, nämlich die weitgehende Unabhängigkeit und lokale Beschreibung von Prozessen wird teilweise eingeschränkt. Trotzdem sind activate und passivate in vielen Simulationssprachen, z.B. in Simula67 verfügbar.

Die zweite Möglichkeit orientiert sich mehr am mentalen Modell und ist damit weiter weg von der Implementierung. Das Warten auf Bedingungen wird dahingehend eingeschränkt, dass b nicht mehr eine beliebige Bedingung sein darf, sondern nur auf bestimmte Bedingungen gewartet werden kann. Dieser Ansatz ist insbesondere für Szenario-Sprachen von Interesse. So ist in Warteschlangennetzen das Warten auf Bediener üblich. Falls es nur eine endliche Anzahl möglicher Bedingungen gibt, auf die gewartet werden kann, so wird für jede Bedingung eine Warteschlange gebildet, in die Prozesse, die auf die zugehörige Bedingung warten, eingeordnet werden. Wird nun eine Bedingung erfüllt, so braucht nur in der zugehörigen Warteschlange nach Prozessen zur Aktivierung gesucht werden. Dieses Konzept ist effizient und stellt keine zusätzlichen Anforderungen an den Modellierer. Es ist allerdings eine Einschränkung gegenüber dem allgemeinen wait_until.

2.3 Generierung und Bewertung von Zufallszahlen

In der diskreten Simulation spielt die Stochastik eine große Rolle. Es gibt nur wenige Modelle, in denen keine Stochastik auftritt. Man kann sicherlich lange und intensiv darüber diskutieren, ob Zufall oder Stochastik Teil unserer Umwelt ist oder ob es nur ein künstliches Phänomen ist, das vom Menschen aus Unkenntnis eingeführt wurde. Diese Diskussion soll hier aber nicht geführt werden, für Interessierte sei auf die beiden populärwissenschaftlichen Bücher [7, 10] zum Weiterlesen verwiesen.

Unabhängig davon, ob Zufall als gegeben vorausgesetzt wird oder als ein Instrument angesehen wird um reales Verhalten in einem Modell kompakt zu beschreiben, ist unsere Wahrnehmung der Realität in den meisten Fällen nicht deterministisch. So treten Ausfälle technischer Geräte zufällig auf und auch die Reparatur erfordert eine nicht exakt bestimmbare Zeit. Ankünfte von Fahrzeugen an einer Ampel oder von Kunden in einem Supermarkt sind ebenfalls zufällig. Auch viele Vorgänge in der Natur, wie etwa die Einschlagstellen von Blitzen oder der Zerfall von Atomen treten zufällig auf. Gleichzeitig kann Zufall auch benutzt werden, um komplexe deterministische Vorgänge kompakter zu beschreiben.

Im Modell wird Zufall benutzt, um Komplexität zu vermeiden oder wenn Details und Zusammenhänge nicht bekannt sind, so dass eine deterministische Beschreibung nicht möglich ist. Grundlage von Zufallsprozessen sind Wahrscheinlichkeitsrechnung und Statistik. Bevor nun die Realisierung von Zufallsprozessen im Rechner beschrieben wird, sollen in einer sehr kurzen Einführung auf die benötigten Grundlagen eingegangen werden (siehe auch [5, Kap. 5.1] und [11, Kap. 4.2]).

Grundlagen der Wahrscheinlichkeitsrechnung

Das mathematische Modell zur Behandlung zufälliger Ereignisse liefert die Wahrscheinlichkeitsrechnung.

Ein *Zufallsexperiment* ist ein Prozess, dessen Ausgang wir nicht mit Gewissheit vorhersagen können. Die Menge aller möglichen einander ausschließenden Ausgänge bezeichnen wir mit S, die Menge der *Elementarereignisse*. Eine Menge E ist eine *Ereignismenge*, falls gilt

- $\emptyset \in E$ und $S \in E$
- $A \in E \Rightarrow S \setminus A \in E$
- $A_i \in E$ $(i \in \mathcal{I}) \Rightarrow \cup_{i\in\mathcal{I}} A_i \in E$ und $\cap_{i\in\mathcal{I}} A_i \in E$

Ein weiterer zentraler Begriff ist das *Wahrscheinlichkeitsmaß*. Ein Wahrscheinlichkeitsmaß $P[A]$ bildet $A \in E$ auf reelle Zahlen ab, so dass

- $0 \leq P[A] \leq 1$, $P[S] = 1$ und $P[\emptyset] = 0$
- falls $A \cap B = \emptyset \Rightarrow P[A \cup B] = P[A] + P[B]$

Typische Beispiele für Zufallsexperimente sind das Werfen einer Münze und eines Würfels. Um den Zusammenhang zwischen verschiedenen Ereignissen herzustellen, kann man bedingte Wahrscheinlichkeiten betrachten. Man schreibt üblicherweise $A|B$ mit der Interpretation A unter der Bedingung B. Es gilt

$$P[A|B] = \frac{P[A \cap B]}{P[B]} \tag{2.1}$$

Sei $S = A_1 \cup A_2 \cup \ldots \cup A_K$ und alle A_k seinen disjunkt, dann gilt (Satz von der Totalen Wahrscheinlichkeit)

$$P(B) = \sum_{k=1}^{K} P[B|A_k] \cdot P[A_k]$$

Von großer Bedeutung ist auch der Satz von Bayes, der eine Beziehung zwischen $A|B$ und $B|A$ herstellt.

$$P[A|B] = P[B|A] \cdot \frac{P[A]}{P[B]}$$

Eine *Zufallsvariable* (ZV) ist eine reellwertige Variable, deren Wert durch den Ausgang eines Zufallsexperiment bestimmt ist. Eine Zufallsvariable bildet also die Ausgänge eines Zufallsexperiments auf reelle Zahlen ab. Zufallsvariablen werden im Folgenden mit Großbuchstaben $X, Y, Z, \ldots$bezeichnet, während Kleinbuchstaben $x, y, z, \ldots$ den konkreten Wert einer Zufallsvariablen bezeichnen. Man unterscheidet zwischen *diskreten* und *kontinuierlichen Zufallsvariablen*. Eine

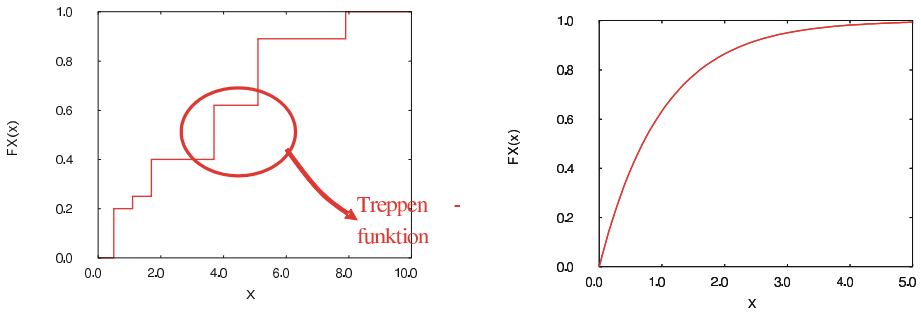

Abbildung 2.32: Verteilungsfunktion einer diskreten und einer kontinuierlichen Zufallsvariablen.

diskrete Zufallsvariable nimmt Werte aus einer endlichen oder abzählbaren Menge an. Typische Beispiele sind der Münzwurf (Werte $0, 1$), das Würfel (Werte $1, \ldots, 6$) oder die Anzahl eingehender Telefonanrufe an einer Vermittlungsstelle innerhalb einer Stunde (da keine obere Schranke a priori bekannt ist, werden theoretisch Werte aus $\mathbb{N}$ betrachtet). Kontinuierliche Zufallsvariablen können Werte aus einer überabzählbaren Menge annehmen. Beispiele sind die Zwischenankunftszeiten von Kunden an einem Schalter, die Bedienzeiten der Kunden oder die Regenmenge, die an einem Tag fällt. Wenn man davon ausgeht, dass diese Werte beliebig genau messbar wären, so können in diesen Fällen beliebige nicht negative Werte angenommen werden.

Zufallsvariablen können durch verschiedene Maßzahlen charakterisiert werden. Von zentraler Bedeutung ist die *Verteilungsfunktion* (Vfkt) $F(x) = P[X \leq x]$ für $-\infty \leq x \leq \infty$. Die Verteilungsfunktion charakterisiert die Zufallsvariable vollständig und es gilt

- $0 \leq F(x) \leq 1$
- $x_1 < x_2 \Rightarrow F(x_1) \leq F(x_2)$
- $\lim_{x \to -\infty} F(x) = 0$ und $\lim_{x \to \infty} F(x) = 1$

Oft kann die Zufallsvariable auch nur Werte aus einem Intervall annehmen. In diesem Fall ist $F(x) = 0$, falls x kleiner als der kleinste Wert ist, der angenommen werden kann und $F(x) = 1$, falls x größer als der größte Wert ist, der angenommen werden kann. Abbildung 2.32 zeigt zwei Beispiele für den Verlauf der Verteilungsfunktion. Für diskrete Zufallsvariablen ist $F(x)$ immer eine Treppenfunktion. Für kontinuierliche Zufallsvariablen kann die Verteilungsfunktion ebenfalls Sprungstellen aufweisen.

Diskrete ZV X mit Wertebereich W_X

Sei $p(x) = P[X = x]$ die Wahrscheinlichkeit, dass X den Wert x annimmt. Es gilt

- $p(x) = 0$ für $x \notin W_X$
- $0 \leq p(x) \leq 1$ für $x \in W_X$
- $\sum_{x \in W_X} p(x) = 1.0$
- $\sum_{x \in W_X \wedge x \leq y} p(x) = F(y)$

Neben der Verteilungsfunktion dienen die *Momente* zur Charakterisierung von Zufallsvariablen. Das $i$te Moment einer diskreten Zufallsvariablen ist definiert als

$$E(X^i) = \sum_{x \in W_X} p(x) \cdot x^i$$

Von besonderer Bedeutung ist das erste Moment $E(X) = E(X^1)$, der *Erwartungswert*. Der Erwartungswert gibt die Summe der mit ihren Wahrscheinlichkeiten gewichteten Werte an. Beim einem

fairen Würfel wäre der Erwartungswert 3.5. Das zweite Moment beschreibt die mögliche Variation der Werte der Zufallsvariablen. Als Maßzahl ist die *Varianz* $\sigma^2(X) = E(X^2) - E(X)^2 \geq 0$ und die *Standardabweichung* $\sigma(X)$ von Bedeutung. Eine größere Varianz/Standardabweichung deutet auf eine größere Variabilität der Werte einer Zufallsvariablen hin. Wird die Varianz um den Erwartungswert bereinigt, so erhält man den *Variationskoeffizienten* $VK(X) = \sigma(X)/E(X)$. Eine Maßzahl für die Abhängigkeit von zwei Zufallsvariablen X und Y ist die Kovarianz

$$C(X,Y) = E((X - E(X)) \cdot (Y - E(Y))) = E(X \cdot Y) - E(X) \cdot E(Y) \tag{2.2}$$

die auch kleiner 0 sein kann. Die letzte Umformung lässt sich unter Anwendung der folgenden Rechenregeln für Erwartungswerte herleiten.

- $E(c \cdot X) = c \cdot E(X)$ für eine Konstante c,
- $E(\sum_{i=1}^{n} c_i \cdot X_i) = \sum_{i=1}^{n} c_i \cdot E(X_i)$ für Konstanten $c_i$und Zufallsvariablen X_i

Die letzte Gleichung gilt sogar, wenn die Zufallsvariablen X_i abhängig sind. Dagegen gilt $E(X \cdot Y) = E(X) \cdot E(Y)$ nur falls $C(X,Y) = 0$, also z.B. bei unabhängigen Zufallsvariablen.

Es werden nun einige typische diskrete Verteilungen vorgestellt. Die Bernoulli-Verteilung mit Parameter $p \in [0,1]$ beschreibt binäre Entscheidungen. Mit Wahrscheinlichkeit p tritt ein Ereignis ein, mit Wahrscheinlichkeit $(1-p)$ tritt es nicht ein. Damit gilt

$$p(x) = \begin{cases} p & \text{falls x=1} \\ 1-p & \text{falls x=0} \\ 0 & \text{sonst} \end{cases} \qquad F(x) = \begin{cases} 0 & \text{falls } x < 0 \\ 1-p & \text{falls } 0 \leq x < 1 \\ 1 & \text{sonst} \end{cases} \qquad \begin{array}{l} E(X) = p \\ \sigma^2(X) = p \cdot (1-p) \end{array}$$

Die geometrische-Verteilung hat ebenfalls einen Parameter $p \in (0,1]$ und beschreibt die Anzahl der erfolglosen Versuche bis zu einem Erfolg, wenn der Erfolg in jedem Versuch mit Wahrscheinlichkeit p eintritt. Es gilt

$$p(x) = \begin{cases} p \cdot (1-p)^x & \text{falls } x \in \{0,1,2,\ldots\} \\ 0 & \text{sonst} \end{cases} \qquad F(x) = \begin{cases} 1 - (1-p)^{\lfloor x \rfloor + 1} & \text{falls } x \geq 0 \\ 0 & \text{sonst} \end{cases}$$

sowie $E(X) = (1-p)/p$ und $\sigma^2(X) = (1-p)/p^2$. Die geometrische Verteilung hat einige interessante Eigenschaften. Insbesondere besitzt die geometrische Verteilung als einzige diskrete Verteilung die so genannte *Gedächtnislosigkeitseigenschaft*. Dies bedeutet in diesem Fall, dass unabhängig vom Wert von x, die Wahrscheinlichkeit im nächsten Versuch einen Erfolg zu haben immer gleich bleibt. Die Gedächtnislosigkeiteigenschft ist insbesondere wichtig zur Herleitung von analytischen Lösungsansätzen.

Der *Poisson-Prozess* wird zur Modellierung realer Abläufe breit eingesetzt. Er beschreibt die Anzahl auftretender Ereignisse, wenn ein einzelnes Ereignis mit konstanter Rate λ (> 0) auftritt. Für ein Intervall der Länge 1 ist dann $p(x)$ die Wahrscheinlichkeit, dass genau x Ereignisse im Intervall auftreten. Es gilt

$$p(x) = \begin{cases} \frac{e^{-\lambda} \cdot \lambda^x}{x!} & \text{falls } x \in \{0,1,2,\ldots\} \\ 0 & \text{sonst} \end{cases} \qquad F(x) = \begin{cases} e^{-\lambda} \cdot \sum_{i=0}^{\lfloor x \rfloor} \frac{\lambda^i}{i!} & \text{falls } x \geq 0 \\ 0 & \text{sonst} \end{cases} \qquad \begin{array}{l} E(X) = \lambda \\ \sigma^2(X) = \lambda \end{array}$$

Werden allgemeine Intervalle der Länge t betrachtet, so muss in den obigen Formeln λ durch λt ersetzt werden. Ein Poisson Prozess modelliert reales Verhalten immer dann gut, wenn in der Realität Situationen betrachtet werden, bei denen Ereignisse aus vielen unabhängigen Quellen stammen können, die jede für sich mit geringer Rate Ereignisse generieren. Ein typisches Beispiel ist eine Vermittlungsstelle für Telefonanrufe. Viele potenzielle Anrufer können unabhängig voneinander Anrufe generieren und der einzelne Anrufer generiert selten einen Anruf. Es zeigt sich allerdings, dass das Modell des Poisson Prozesses zwar die Realität mit menschlichen Anrufern gut wiedergibt, durch zusätzliche Dienste, die inzwischen über das Telefon abgewickelt werden, treten aber andere Ankunftsmuster auf, die sich mit Poisson-Prozessen nicht mehr genau genug modellieren lassen.

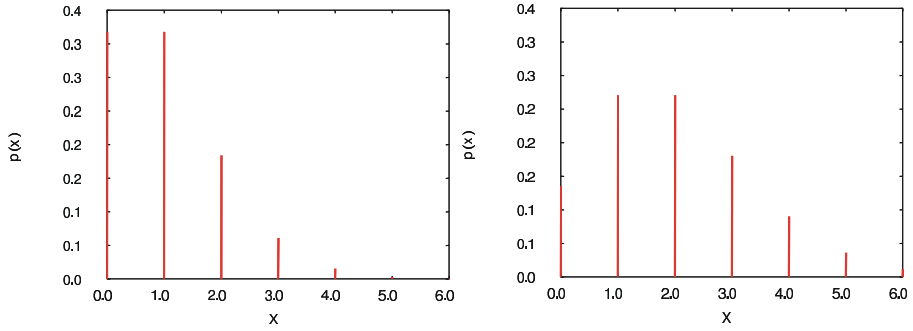

Abbildung 2.33: Poisson Prozess mit Parameter $\lambda = 1$ (linke Seite) und $\lambda = 2$ (rechte Seite).

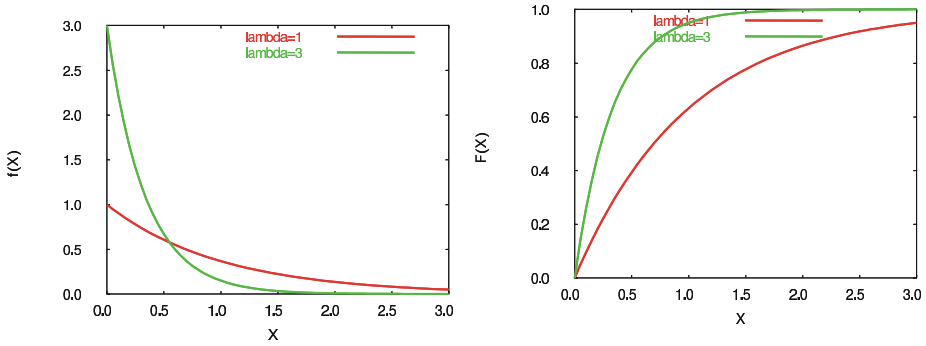

Abbildung 2.34: Dichtefunktion und Verteilungsfunktion der Exponentialverteilung.

Kontinuierliche ZV X mit Wertebereich W_X

Im Gegensatz zum diskreten Fall kann die Zufallsvariable nun überabzählbar viele Werte annehmen. Dies bedeutet, dass für die Wahrscheinlichkeit $p(x) = 0$ für alle $x \in W_X$ gelten kann und trotzdem eine Verteilung vorliegt. Deshalb wird statt der Wahrscheinlichkeit die Dichtefunktion (Dfkt) $f(x)$ für kontinuierliche Zufallsvariablen betrachtet. Für eine Dichtefunktion muss gelten

- $f(x) = 0$ für $x \notin W_X$
- $0 < f(x)$ für $x \in W_X$
- $\int_{x \in W_X} f(x)dx = 1.0$
- $\int_{x \in W_X \wedge x \leq y} f(x)dx = F(y)$
- $\int_y^z f(x)dx = F(z) - F(y) = P[x \in [y, z]]$ falls $y \leq z$
- $E(X^i) = \int_{x \in W_X} x^i \cdot f(x)dx$

Es sollen nun einige wichtige kontinuierliche Verteilungen vorgestellt werden. Die Exponentialverteilung mit Parameter $\lambda > 0$ ist durch die folgenden Funktionen/Maßzahlen charakterisiert.

$$f(x) = \begin{cases} \lambda \cdot e^{-\lambda x} & \text{falls } x \geq 0 \\ 0 & \text{sonst} \end{cases} \qquad F(x) = \begin{cases} 1 - e^{-\lambda x} & \text{falls } x \geq 0 \\ 0 & \text{sonst} \end{cases} \qquad \begin{array}{l} E(X) = 1/\lambda \\ \sigma^2 = 1/\lambda^2 \\ VK(X) = 1 \end{array}$$

Die Dichte- und Verteilungsfunktion der Exponentialverteilung wird in Abbildung 2.34 für $\lambda = 1$ und $\lambda = 3$ dargestellt. Exponentialverteilungen treten immer dann auf, wenn viele unabhängige Quellen für ein Ereignis existieren und jede Quelle selten ein Ereignis generiert. Ein typisches

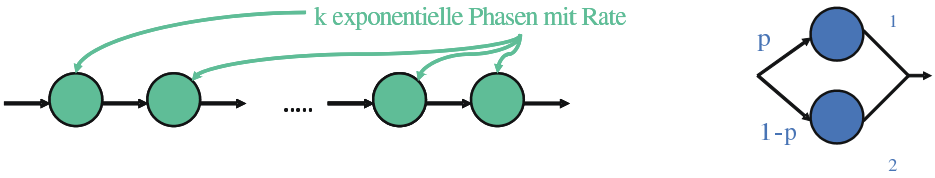

Abbildung 2.35: Struktur der Erlang- und Hyperexponential-Verteilung.

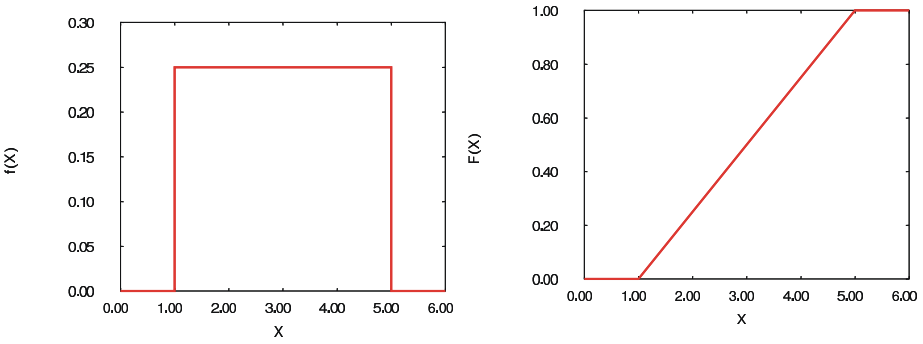

Abbildung 2.36: Dichtefunktion und Verteilungsfunktion der Gleichverteilung mit Parametern 1 und 5.

Beispiel sind die Zwischenankunftszeiten von Telefonanrufen an einer Vermittlungsstelle. Die Zeit zwischen dem Auftreten von Ereignissen in einem Poisson-Prozess ist exponentiell verteilt. Darüber hinaus ist die Exponentialverteilung die einzige kontinuierliche Verteilung, die gedächtnislos ist. Eine Zufallsvariable X ist gedächtnislos, wenn

$$P[X > t + s | X > t] = P[X > s]$$

für alle $s, t \geq 0$ gilt. Für die Exponentialverteilung ergibt sich

$$\begin{aligned} P[X > s + t | X > t] &= e^{-\lambda \cdot (s+t)} / e^{-\lambda t} \\ &= e^{-\lambda s} \qquad\qquad = P[X > s] \end{aligned}$$

Die Gedächtnislosigkeit erweist sich als wichtig für analytische Berechnungen, da die Zeit seit dem letzten Ereignis nicht in die Zustandsbeschreibung mit einbezogen werden muss und trotzdem die Verteilung der Eintrittszeit des nächsten Ereignisses ermittelt werden kann.

Durch das Verbinden von mehreren exponentiellen Phase lassen sich komplexere Verteilungen erzeugen. Zwei typische Beispiele werden in Abbildung 2.35 gezeigt und sollen kurz beschrieben werden. Bei der Erlang-k-Verteilung werden k exponentielle Phasen mit identischem Parameter λ durchlaufen. Die Gesamtzeit ergibt sich also aus der Summe der einzelnen Zeiten. Es gilt für Erlang-k-verteilte Zufallsvariablen $E(X) = k/\lambda$, $\sigma^2(X) = k/\lambda^2$ und $VK(X) 1/\sqrt{k}$. Damit ist die Erlang-Verteilung weniger variabel als die Exponentialverteilung. Durch die Hinzunahme weiterer Phasen sinkt die Variabilität. Bei der Hyperexponentialverteilung wird alternativ eine von zwei Phasen ausgewählt. Die Verteilung hat 3 Parameter, nämlich p, die Wahrscheinlichkeit Phase 1 zu wählen, sowie λ_1 und λ_2, die Raten der exponentiellen Phasen. Es gilt $E(X) = p/\lambda_1 + (1-p)/\lambda_2$ und $\sigma^2(X) = p(2-p)/\lambda_1^2 + (1-p)^2/\lambda_2^2 - 2p(1-p)/(\lambda_1\lambda_2)$. Man kann nun einfach zeigen, dass für die Hyperexponentialverteilung $VK(X) \geq 1$ immer gilt. Die Hyperexponentialverteilung ist also mindestens so variabel wie die Exponentialverteilung. Man kann sogar zeigen, dass jeder beliebige Variationskoeffizient größer gleich 1 mit einer Hyperexponentialverteilung bei entsprechender Parametrisierung erreichbar ist. Hyperexponentialverteilungen können auch für mehr als zwei Phasen definiert werden.

Die Gleichverteilung (siehe Abbildung 2.36) hat zwei Parameter a, b ($a < b$) und ist durch einen festen Wert der Dichtefunktion auf dem endlichen Intervall $[a, b]$ gekennzeichnet. Sie ist

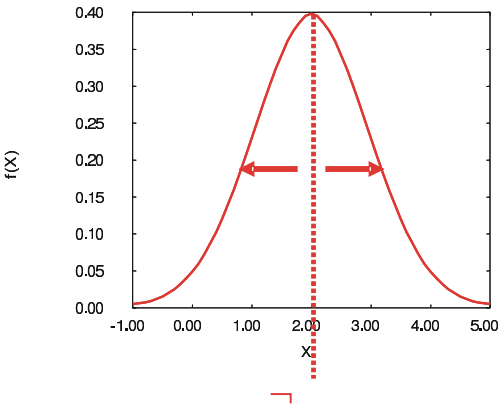

Abbildung 2.37: Dichtefunktion der Normalverteilung.

charakterisiert durch

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{falls } a \leq x \leq b \\ 0 & \text{sonst} \end{cases} \qquad F(x) = \begin{cases} 0 & \text{falls } x < a \\ \frac{x-a}{b-a} & \text{falls } a \leq x \leq b \\ 1 & \text{sonst} \end{cases} \qquad \begin{array}{l} E(X) = (a+b)/2 \\ \sigma^2(X) = (b-a)^2/12 \\ VK(X) \approx 1.73 \end{array}$$

Gleichverteilungen sind von zentraler Bedeutung in der Simulation, da die $[0,1]$-Gleichverteilung als Grundlage der Generierung von Zufallszahlen benutzt wird, wie im Laufe dieses Abschnitts gezeigt wird.

Die Normalverteilung hat zwei Parameter μ und σ, die ihren Mittelwert und ihre Standardabweichung beschreiben. Die Dichtefunktion der Normalverteilung lautet

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

und wird in Abbildung 2.37 gezeigt. Der Wert von μ kann benutzt werden, um die Dichtefunktion zu verschieben, während σ für die Form (bzw. Breite) der Dichtefunktion verantwortlich ist. Für die Verteilungsfunktion der Normalverteilung existiert keine geschlossene Form. Für normalverteilte Zufallsvariablen X mit Parametern μ und σ schreibt man oft $X \sim N(\mu, \sigma)$. Die Dichtefunktion der Normalverteilung ist symmetrisch um den Punkt μ und erreicht ihr Maximum in μ. Es gilt $f(x) > 0$ für alle x und $\lim_{x\to\infty} f(x) = \lim_{x\to-\infty} f(x) = 0$. Falls $X \sim N(0,1)$, dann ist $Y = \mu + \sigma \cdot X \sim N(\mu, \sigma)$ und es gilt

$$P[Y < y] = P[\mu + \sigma \cdot X < y] = P[X < (y-\mu)/\sigma]$$

Diese Wahrscheinlichkeit kann aus den Werten der $N(0,1)$-Verteilung ermittelt werden. Die kritischen Werte der $N(0,1)-$Verteilung, d.h. die Werte x mit $P[X < x] = \alpha$ für $X \sim N(0,1)$ und $\alpha = 0.9, 0.95, 0.99$, sind in Wahrscheinlichkeitstafeln in den meisten Statistikbüchern zu finden und können auch numerisch berechnet werden. Die $N(0,1)$-Verteilung bezeichnet man auch als Standardnormalverteilung. Der Normalverteilung kommt in der Statistik und damit auch in der Simulation eine zentrale Bedeutung zu. Dies liegt insbesondere im zentralen Grenzwertsatz begründet, der im Prinzip aussagt, dass die Summe unabhängiger identisch verteilter Zufallszahlen unter relativ allgemeinen Bedingungen gegen eine Normalverteilung konvergiert. Damit kann man im Prinzip durch mehrfaches Beobachten erreichen, dass der beobachtete Wert (fast) normalverteilt ist. Auf dieser Basis lassen sich statistische Aussagen machen, wie wir später sehen werden.

Schätzer für Verteilungsparameter

In der Statistik werden unbekannte Größen auf Grund von Beobachtungen geschätzt. Gleiches gilt für die Simulation. Wenn Parameter in einem Simulationsmodell gesetzt werden sollen, so

werden sie auf Grund von Schätzungen festgelegt. Wenn in einem Simulator Zufallsvariablen zur Beschreibung von Parametern benutzt werden, so sind auch die Ausgaben Zufallsvariablen und Aussagen über das Verhalten des simulierten Modells können nur auf Basis von Schätzungen erfolgen.

Definition 3 *Sei X eine Zufallsvariable mit unbekannter Verteilung. Eine Menge von Beobachtungen $x_1, \ldots, x_n$ nennt man eine Stichprobe.*

Wir gehen in der Regel davon aus, dass die einzelnen Beobachtungen einer Stichprobe unabhängig sind, wenn dies nicht anders gesagt wird. Oftmals sollen Aussagen über einen Parameter Θ der Verteilung von X auf Basis einer Stichprobe $x_1, \ldots, x_n$ gewonnen werden. Typische Beispiele für Θ wären der Erwartungswert, die Varianz oder $P[X < x]$.

Definition 4 *Ein Schätzer $\tilde{\Theta}$ für einen Parameter Θ der Verteilung einer Zufallsvariablen X auf Basis einer Stichprobe $x_1, x_2, \ldots, x_n$ ist eine Funktion $g(x_1, \ldots, x_n) \rightarrow \tilde{\Theta}$. g nennt man die Schätzfunktion.*

Für Schätzer gibt es eine Reihe von Eigenschaften, von denen einige vorgestellt werden. $\tilde{\Theta}$ heißt

- erwartungstreu, wenn $E(\tilde{\Theta}) = \Theta$,
- asymptotisch erwartungstreu, wenn $\lim_{n\rightarrow\infty} E(\tilde{\Theta}) = \Theta$,
- konsistent, wenn $\lim_{n\rightarrow\infty} P[|\tilde{\Theta} - \Theta| > \epsilon] = 0$ für jedes $\epsilon > 0$.

Man ist natürlich bestrebt erwartungstreue und konsistente Schätzer zu verwenden. Die Schätzung des Parametern wird uns im Laufe dieses und der folgenden Abschnitte noch mehrfach beschäftigen, insbesondere werden wir unterschiedliche Schätzer kennen lernen.

Grundlagen der Generierung von Zufallszahlen

Nach der kurzen Einführung in die benötigten Grundlagen der Wahrscheinlichkeitsrechnung und Statistik wird nun die Generierung von Zufallszahlen behandelt. Dies ist ein zentraler Aspekt jeder stochastischen Simulation. Zufallszahlen dienen dazu, stochastisches Verhalten in den eigentlich deterministischen Programmablauf zu bringen. Die Qualität der Simulationsergebnisse hängt ganz entscheidend von der Qualität der verwendeten Zufallszahlen ab. Die Beantwortung der Frage *"Was sind gute Zufallszahlen?"* ist damit essentiell, leider ist die Antwort aber nicht einfach und es wird sich im Lauf dieses Abschnitts zeigen, dass wir keine Methoden haben, die für erzeugte Zufallszahlen zweifelsfrei die Güte festlegen. Die Erzeugung von Zufallszahlen wird in den meisten Simulationsbüchern ausführlich beschrieben, z.B. [11, Kap. 7,8] oder [5, Kap. 7,8].

Ziel ist es, Realisierungen einer Zufallsvariablen X zu generieren. Es wird also eine Methode $zz(X) \rightarrow x$ gesucht, so dass für so erzeugte $x_1, x_2, \ldots$ gilt

- $P[X_j < y] = P[\mathrm{X}<\mathrm{y}]$ für alle y (damit gilt auch $E(X_j^i) = E(X^i)$ für alle $i, j > 0$) und
- X und X_j für beliebige i, j $(i \neq j)$ seien unabhängig.

wobei X_i die Zufallsvariable ist, aus der x_i resultiert. Die Generierung von Realisierungen einer Zufallsvariablen X nennt man das *Ziehen von Zufallszahlen (ZZ)*. Das Ziehen von Zufallszahlen erfolgt in drei Schritten:

1. Erzeugung von gleichverteilten ganzzahligen Zufallszahlen im Intervall $[0, m)$.
2. Transformation in (approximativ) $[0, 1)$-verteilten Zufallszahlen.
3. Transformation der Zufallszahlen in die gewünschte Verteilung.

Zwei grundsätzlich unterschiedliche Ansätze zur Generierung von Zufallszahlen existieren, die Generierung von echten Zufallszahlen und die Nutzung von Pseudozufallszahlen. Echte Zufallszahlen resultieren aus der Beobachtung zufälliger Prozesse. So kann das Werfen einer Münze oder das Würfel dazu benutzt werden, Zufallszahlen zu erzeugen. Wenn man davon ausgeht, dass die Münze oder der Würfel fair sind, d.h. unabhängig und identisch verteilte Resultate liefern, so kann man auf diese Weise Zufallszahlen erzeugen. Mit n Münzwürfen kann man zum Beispiel n Bits bestimmen und damit ganzzahlige Zufallszahlen aus dem Intervall $[0, 2^n]$ erzeugen. Da ein Simulationsprogramm keine Münzen oder Würfel werfen kann, böte sich in der Simulation eher die Beobachtung von zufälligen physikalischen Prozessen an. Beispiele für solche Prozesse sind der radioaktive Zerfall oder das weiße Rauschen. Der Nachteil echter Zufallszahlen ist ihre fehlende Reproduzierbarkeit. Es liegt natürlich in der Natur des Zufalls, dass wir das Verhalten eben nicht reproduzieren können. Für den Ablauf von Simulationsprogrammen hat dies aber einige entscheidende Nachteile. So läuft ein Programm, das echte Zufallszahlen nutzt, bei jedem Start unterschiedlich ab. Dies bedeutet, dass Debugging praktisch unmöglich ist. Die Erfahrungen aus dem Softwareentwurf zeigen, dass größere Programme ohne Debugging nicht realisierbar sind. Ein weiterer Nachteil fehlender Reproduzierbarkeit ist, dass Modelle nicht unter identischen Bedingungen (d.h. identischen Realisierungen des Zufalls) verglichen werden können.

Die Forderung nach Reproduzierbarkeit führt zu so genannten *Pseudozufallszahlen*. Wie der Name schon andeutet, handelt es sich dabei nicht um wirkliche Zufallszahlen, sondern um Zahlen, die nach einem Algorithmus erzeugt wurden. Für jemanden, der den Algorithmus kennt, sind die erzeugten Zahlen damit in keiner Weise zufällig, sondern rein deterministisch. Entscheidend ist, dass die erzeugten Zahlen ohne Kenntnis des Generierungsalgorithmus nicht von wahren Zufallszahlen zu unterscheiden sind. Die einfachste Form der Erzeugung von Pseudozufallszahlen ist das auslesen aus einer Tabelle. Eine solche Tabelle könnte man zum Beispiel dadurch füllen, dass zufällig Prozesse beobachtet werden. Die Tabellenlösung wurde in der Vergangenheit oft angewendet, so erstellte ein Herr Tippett im Jahr 1927 eine Tabelle mit 41600 gleichverteilten Zahlen aus den Daten der Finanzverwaltung. Heute werden Tabellen nicht mehr benutzt, da sehr viele Zufallszahlen für Simulationen benötigt werden und die notwendigen Tabellen einen immensen Speicherplatz erfordern würden und auch das Auslesen der Zahlen zu viel Zeit verbrauchen würde.

Algorithmen zur Erzeugung von Zufallszahlen

Die Alternative zur Tabellenlösung ist ein Generierungsalgorithmus der Form $x_i = g(s_i)$ und $s_{i+1} = f(s_i)$. Der Algorithmus hat einen internen Zustand s_i aus dem durch eine Funktion g eine Zufallszahl erzeugt wird und durch eine Funktion f wird der nächste Zustand generiert. Heutige Generatoren setzen $s_i = x_i$ und benötigen damit nur noch die Funktion f. Durch Setzen eines festen Startwertes x_0 wird immer wieder die selbe Sequenz von Zufallszahlen generiert. Damit ist die Reproduzierbarkeit offensichtlich gegeben. Durch die Wahl unterschiedlicher Startwerte können unterschiedliche Sequenzen von Zufallszahlen erzeugt werden. Offensichtlich erzeugt jeder solche Generierungsalgorithmus nur eine endliche Sequenz von Zufallszahlen, das $x_i = x_j$ dazu führen muss, dass $x_{i+k} = x_{j+k}$ für alle $k > 0$. Bei endlicher Zahlenlänge muss damit zwangsläufig irgendwann eine identische Zahl erreicht werden. Damit ein Generierungsalgorithmus "gute" Zufallszahlen erzeugt und in Simulationsmodellen einsetzbar ist, werden einige Anforderungen an ihn gestellt.

- Die generierten Zufallszahlen müssen gleichverteilt sein.
- Die generierten Zufallszahlen müssen unabhängig sein. Dies bedeutet, dass die Kenntnis der ersten n Zufallszahlen keinerlei Information über die $n + 1$te Zufallszahl liefert. Dies gilt natürlich nur, wenn der Generierungsalgorithmus unbekannt ist.
- Die Sequenzlänge bis zur Wiederholung einer Zahl muss lang sein.
- Die Erzeugung muss effizient sein.
- Der Algorithmus muss portabel sein.

Es gibt zahlreiche unterschiedliche Generierungsalgorithmen. Wir betrachten hier kurz den ersten Algorithmus der zur Erzeugung von Pseudozufallszahlen publiziert wurde und stellen dann anschließend die am weitesten verbreitete Klasse von Generatoren vor, die in fast allen Simulationswerkzeugen zu finden sind.

Der erste Algorithmus ist die bekannte "Midsquare Method" von von Neumann und Metropolis 1940 publiziert wurde. Ausgehend von einer achtstelligen Dezimalzahl Z_0 werden die mittleren vier Ziffern genommen. Vor diese vier Ziffern wird eine Dezimalpunkt gesetzt und es entsteht eine Zahl aus dem Intervall $[0, 1)$. Z_1 wird dadurch erzeugt, dass die mittleren vier Ziffern von Z_0 quadriert werden, wodurch wieder eine Zahl mit bis zu acht Ziffern entsteht. Bei weniger als acht Ziffern wird von links mit Nullen aufgefüllt. Anschließend wird mit Z_1 wie mit Z_0 verfahren. Ein Beispiel für eine Sequenz von Zufallszahlen wäre

$$\begin{aligned} Z_0 &= 43718244 \quad \rightarrow 0.7182 \\ Z_1 &= 51581124 \quad \rightarrow 0.5811 \\ Z_2 &= 33767721 \quad \rightarrow 0.7677 \\ Z_3 &= 58936329 \quad \rightarrow 0.9363 \\ Z_4 &= 87665769 \quad \rightarrow 0.6657 \\ Z_5 &= 44315649 \quad \rightarrow 0.3156 \end{aligned}$$

Auf den ersten Blick sehen die erzeugten die Zahlen recht zufällig aus. Trotzdem ist die Methode nicht gut, d.h. sie erzeugt keine wirkliche Sequenz von Pseudozufallszahlen, die die oben genannten Bedingungen erfüllen (überlegen warum!).

Die heute meistens verwendeten Generierungsalgorithmen basieren auf einer Arbeit von Lehmer (1951) und werden als *lineare Kongruenzgeneratoren* (LCGs) bezeichnet. Lineare Kongruenzgeneratoren sind durch die Funktion

$$x_i = \left(\sum_{j=1}^{r} a_j \cdot x_{i-j} + c \right) (\mod m)$$

mit $x_0, x_1, \ldots, x_{r-1}, a_1, \ldots, a_r, c \in \mathbb{N}$ gekennzeichnet. Man verwendet heute meistens die reduzierte Version

$$x_i = (a \cdot x_{i-1} + c)(\mod m)$$

Falls $c = 0$, so spricht man von einem multiplikativen Generator ansonsten von einem gemischten Generator. Den initialen Wert x_0 bezeichnet man auch als die Saat des Generators. Jeder LCG weist die folgenden Eigenschaften auf

- Falls $c > 0$ so gilt $x_i \in [0, m)$ und für $c = 0$ gilt $x_i \in (0, m)$
- $x_i = x_j \Rightarrow x_{i+k} = x_{j+k}$ für alle $k \geq 0$

Mit $\kappa = \min_{|i-j|}(x_i = x_j)$ bezeichnet man die Periode des Generators. Das folgende Beispiel wurde aus [15] entnommen. Die Generierungsvorschrift lautet

$$x_{i+1} = (5 \cdot x_i)(\mod 17)$$

Die folgende Tabelle zeigt die Sequenz der generierten Zufallszahlen ausgehend von $x_0 = 5$.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$5x_i$	25	40	30	65	70	10	50	80	60	45	55	20	15	75	35	5	25
x_{i+1}	8	6	13	14	2	10	16	12	9	11	4	3	15	7	1	5	8

Wie man sieht. erreicht der Generator die maximal mögliche Periode, alle Zahlen von 1 bis 16 werden erzeugt. In Simulationssystemen verwendete Generatoren haben natürlich eine deutlich größere Periode. Die folgenden Generatoren werden bzw. wurden in unterschiedlichen Systemen verwendet.

Name	m	a	c
Unix	2^{32}	1103515245	12345
RANDU	2^{31}	65539	0
Simula/Univac	2^{31}	5^{13}	0
SIMPL-I (IBM)	$2^{31}-1$	48271	0
SIMSCRIPT II.5	$2^{31}-1$	630360016	0
L'Ecuyer	$2^{63}-25$	4645906587823291368	0

Die ersten drei Generatoren wurden zwar breit eingesetzt, haben sich inzwischen allerdings als schlechte Generatoren herausgestellt. Insbesondere der RANDU wurde in der Vergangenheit in vielen Simulationswerkzeugen verwendet. Es fällt auf, dass die "schlechten" Generatoren alle als Modul eine Zweierpotenz haben. Damit stellen sich natürlich zwei Fragen. Warum wählte man eine Zweierpotenz als Modul? Warum sind dann die Generatoren schlecht? Bevor die Fragen beantwortet werden, muss erst einmal festgelegt werden, was unter einem guten Generator zu verstehen ist. Ein guter Generator sollte die folgenden drei Bedingungen erfüllen.

1. Er sollte eine große Periode haben.
2. Er sollte eine effiziente Berechnung der Zufallszahlen erlauben.
3. Er sollte statistische Test bestehen.

Während die ersten beiden Bedingungen den Bedingungen für gute Zufallszahlen entsprechen, unterscheidet sich die dritte Bedingung von den auf Seite 68 geforderten Eigenschaften für gute Zufallszahlen. Dies liegt darin begründet, dass man für einen Generator nicht formal zeigen kann, dass zwei generierte Zufallszahlen x_i, x_j unabhängig sind und jeweils aus einer Gleichverteilung gezogen wurden. Man kann nur anhand von statistischen Test überprüfen, ob diese Annahmen plausibel sind. Entsprechende Testverfahren werden später vorgestellt.

Die erste Bedingung für einen guten Generator fordert eine große Periode, damit viele unterschiedliche Zufallszahlen erzeugt werden können. Dies bedeutet, dass m möglichst groß sein sollte, also im Bereich der maximal darstellbaren ganzen Zahl. Weiterhin sollten in der Periode möglichst m bzw. $m-1$ unterschiedliche Werte erzeugt werden. Man kann Bedingungen an die Parameter a, c und m stellen, damit die volle Periodenlänge erreicht wird. Für einen gemischten Generator müssen die Folgenden drei Bedingungen gelten [11, S. 408].

1. $ggT(m,c) = 1$ man sagt auch, dass m und c relativ prim sind
2. $a \mod p_i = 1$ für alle Primfaktoren p_i von m
3. $a \mod 4 = 1$ falls 4 ein Faktor von m ist

Die Bedingungen sind leicht zu prüfen, so dass ein gemischter Generator mit voller Periodenlänge konstruiert werden kann. Offensichtlich sind die Bedingungen nicht für multiplikative Generatoren anwendbar, da erste Bedingung für $c = 0$ nicht erfüllbar ist. Für multiplikative Generatoren gelten die beiden folgenden Resultate [5, S. 260].

- Falls $m = 2^b$, dann ist der maximale Wert für $\kappa = m/4 = 2^{b-2}$. Dieser Wert wird erreicht für ungerade Saaten x_0 und $a = 3 + 8k$ oder $a = 5 + 8k$ für $k = 0, 1, \ldots$
- Falls m eine Primzahl ist, dann $\kappa = m - 1$ erreicht, falls die kleinste ganze Zahl k, für die $a^k - 1$ durch m ganzzahlig dividiert werden kann, gleich $m - 1$ ist (a ist Primitivwurzel von m).

Die zweite Klasse von Generatoren bezeichnet man auch als *prime modulus multiplicative LCGs* (PMMLCGs). Sie sind natürlich besonders interessant, da sie im Gegensatz zu gemischten Generatoren keine Additionsoperation benötigen. Allerdings ist die Bedingung deutlich komplexer als die einfachen Bedingungen für gemischte Generatoren. Es ist tatsächlich so, dass PMMLCGs oft durch Ausprobieren und Prüfen erzeugt werden.

Die vorgestellten Generatoren hatten zumeist eine Periode im Bereich von $2^{32} \approx 4.2 \cdot 10^9$ der Wortlänge der üblichen Prozessoren zum Zeitpunkt ihrer Entwicklung. Heute ist die übliche Wortlänge zwar 64, aber es gibt erst sehr wenige Generatoren, die diese Wortlänge nutzen. Es bleibt natürlich die Frage, ob 2^{32} nicht ausreicht. Über lange Jahre hätte man diese Frage sicherlich mit ja beantwortet. Wenn man sich aber die Geschwindigkeit heutiger Prozessoren vor Augen führt, so benötigt ein PC mit 3 GHz Prozessor ca. 10 bis 15 Minuten um 2^{32} Zufallszahlen zu erzeugen. Simulatoren realistischer Systeme, wie etwa die Simulationen großer Rechnernetze oder Fertigungssysteme, laufen aber mehrere Stunden oder Tage. Auch wenn in der Simulation neben der Zufallszahlenerzeugung noch viele andere Aufgaben Zeit verbrauchen, ist es leicht nachvollziehbar, dass in einer langen Simulation der Zyklus der Zufallszahlen mehrfach durchlaufen wird. Dies widerspricht natürlich unserem Gefühl von Zufall. Deshalb besteht ein großes Interesse daran gute Generatoren mit größerer Periode zu entwickeln. Dies kann einmal dadurch geschehen, dass ganz neue Generatoren mit $m \approx 2^{64}$ oder noch größer entwickelt werden. Es können aber auch bekannt Generatoren kombiniert werden [13]. Wir betrachten k Generatoren mit Periode m_j für den $j$ten Generator. Sei $x_{i,j}$ der $i$te Wert des $j$ten Generators, dann ist

$$x_i = \left(\sum_{j=1}^{k} (-1)^{j-1} x_{i,j} \right) (\mod m_1 - 1)$$

$[0, m_1 - 2]$-gleichverteilt und die maximal erreichbare Periode lautet

$$\frac{\left(\prod_{j=1}^{k} (m_j - 1 \right)}{2^{k-1}}$$

Für die Wahl der eingesetzten Generatoren sei auf die Originalliteratur verwiesen. Durch die Kombination von Generatoren wurden Generatoren mit Periode 2^{191} und größer erzeugt. Diese Periodenlänge kann mit heutigen Rechnern auch bei parallelen Simulationsläufen nicht ausgeschöpft werden.

Die zweite Anforderung an Zufallszahlengeneratoren ist eine effiziente Generierung. Im Gegensatz zur Addition und Multiplikation ist Division sehr aufwändig. Deshalb sollte ein effizienter Generator möglichst keine oder wenige Divisionen benötigen. Falls $m = 2^b$ ist, so kann auf Divisionen verzichtet werden, da die modulo Operation durch das Weglassen von Stellen, d.h. shiften auf Bitebene, realisiert werden kann. So entspricht $10111011 \mod 2^6 = 00111011$ und in Dezimaldarstellung $187 \mod 64 = 59$. Leider zeigt sich aber, dass durch die modulo-Bildung mit einer Zweierpotenz Zyklen in den niederwertigen Bits auftreten. Dies kann am Beispiel von $x_{i+1} = (5x_i - 1) \mod 16$ erläutert werden. Ausgehend von $x_0 = 1$ ergibt sich die folgende Sequenz von Zahlen, die als Dezimal und als Binärzahlen dargestellt werden.

i	0	1	2	3	4	5	6	7
x_i	1	6	15	12	13	2	11	8
b_0	1	0	1	0	1	0	1	0
$b_1 b_0$	01	10	11	00	01	10	11	00
$b_2 b_1 b_0$	001	110	111	100	101	010	011	000

Während die Dezimaldarstellung recht zufällig aussieht, erkennt man in der Binärdarstellung eine sehr regelmäßige Struktur, die offensichtlich alles andere als zufällig zu sein scheint. Aus diesem Grund weisen Zufallszahlen, die mit einem Generator mit $m = 2^e$ erzeugt wurden, auch schlechte statistische Eigenschaften auf.

Als Alternative bietet es sich an $m = 2^e - 1$ zu wählen. Es ist noch zu zeigen, dass $z = (a \cdot x)(\mod 2^e - 1)$ effizient, d.h. ohne Division berechnet werden kann. Sei dazu $y = (a \cdot x)(\mod 2^e)$. Dies ist offensichtlich durch einfaches Abschneiden der überzähligen Bits berechenbar. Nun muss noch z auf Basis von y berechnet werden. Dies erfordert die folgende einfache Berechnung.

$$z = \begin{cases} y + k & \text{falls } y + k < 2^e - 1 \\ y + k - (2^e - 1) & \text{sonst} \end{cases}$$

mit $k = \lfloor a \cdot x / 2^e \rfloor$. Man kann die Formel leicht herleiten, wenn man sich überlegt, dass bei Berechnung $\mod 2^e$ statt $\mod 2^e - 1$ bei jedem Überlauf über 2^e genau eins zu wenig berechnet wird. Die Summation ergibt genau k. Wenn durch die Addition von k ein weiterer Überlauf erzeugt wird, so muss $2^e - 1$ abgezogen werden. Man benötigt also zusätzliche Additionen, kann aber die Division vermeiden. Das Verfahren wird auch als simulierte Division bezeichnet und lässt sich auch für Modulen der Form $2^e - q$ verallgemeinern.

Bevor der dritte Aspekt, nämlich das Bestehen von statistischen Tests betrachtet wird, werden die erzeugten Zufallszahlen auf das Intervall $[0, 1)$ bzw. $(0, 1)$ normiert. Dies geschieht dadurch, dass durch m dividiert wird. Wenn die Zufallszahlen vorher gleichverteilte ganze Zahlen im Intervall $[0, m)$ bzw. $(0, m)$ waren, so kann man die resultierenden Werte als approximativ $[0, 1)$ bzw. $(0, 1)$ gleichverteilt ansehen. Wir betrachten im Folgenden den Fall $[0, 1)$, wenn die 0 ausgeschlossen ist, kann vollkommen analog vorgegangen werden. Streng genommen liegt natürlich keine Gleichverteilung vor, da nur endliche viele Werte erzeugt werden. Falls m aber die selbe Länge wie die Mantissendarstellung hat, so liegen die erzeugten Werte optimal dicht und es werden all darstellbaren Zahlen erreicht. Damit kann die erzeugten Zufallszahlen als gleichverteilt ansehen, ähnlich wie double-Werte eine Approximation der reellen Zahlen sind.

Statistische Testverfahren für Zufallszahlen

Auf Basis der gleichverteilten Zufallszahlen können Testverfahren angewendet werden, um schlechte von guten Generatoren zu unterschieden. Gute Zufallszahlen müssen zwei Bedingungen erfüllen.

- Sie müssen gleichverteilt im Intervall $[0, 1)$ sein.
- Sie müssen unabhängig sein.

Da Zufallszahlen betrachtet werden, müssen zufällige Schwankungen in die Beobachtungen mit einbezogen werden und es ist nicht möglich Beweise zu führen. Stattdessen werden Testverfahren benutzt werden. Bevor konkrete Testverfahren vorgestellt werden, soll kurz allgemein auf Testverfahren in der Statistik eingegangen werden. Bei einem Test wird eine Hypothese H_0 aufgestellt. Im hier vorliegenden Fall könnte dies eine der beiden folgenden Alternativen sein.

Die mit Generator G erzeugten Zufallszahlen sind unabhängig, identisch $[0, 1)$-gleichverteilt oder

die mit Generator G erzeugten Zufallszahlen sind nicht unabhängig, identisch $[0, 1)$-gleichverteilt.

H_0 heißt *Nullhypothese* und entsprechend heißt $H_1 = \neg H_0$ *Alternativhypothese*. Testverfahren dienen dazu herauszufinden, ob H_0 gilt. Dies geschieht in der Regel auf Basis von Stichproben. Auf Grund der statistischen Schwankungen von Stichproben, können falsche Folgerungen gezogen werden. Tests sind keine Beweise! Man unterschiedet folgende Fehler.

- (Statistische) Fehler der 1. Art (α Fehler): H_1 wird angenommen, obwohl H_0 gilt. D.h. die Nullhypothese wird fälschlicherweise verworfen.
- (Statistische) Fehler der 2. Art (β Fehler): H_0 wird angenommen, obwohl H_1 gilt. D.h. die Nullhypothese wird fälschlicherweise angenommen.

Impliziert ein bestimmter Test mit Wahrscheinlichkeit $\leq \alpha$ Fehler der 1. Art, so heißt er "Test zum (Signifikanz-)Niveau α", unabhängig vom Fehler 2. Art. Natürlich sind aber Fehler 1. und 2. Art nicht unabhängig. Der triviale Test, der H_0 immer annimmt macht keinen Fehler der ersten Art, während der Test, der H_0 immer ablehnt, keinen Fehler der 2. Art macht. Beide Ansätze sind offensichtlich nicht sehr hilfreich. Die Bewertung von Testergebnissen beruht im Wesentlichen auf Tests zum Niveau α.

Beim Testen auf Basis einer Stichprobe wird die Stichprobe mit einer Teststatistik $S(Y_1, \ldots, Y_n)$ bewertet, wobei Y_i die Zufallsvariable ist, die den $i$ten Wert in der Stichprobe beschreibt. Je größer der Wert von $S(y_1, \ldots, y_n)$ für eine konkrete Stichprobe $y_1, \ldots, y_n$ ist, desto unwahrscheinlicher ist H_0 und damit steigt implizit die Wahrscheinlichkeit von H_1. Zur Anwendung sind die folgende Schritte notwendig:

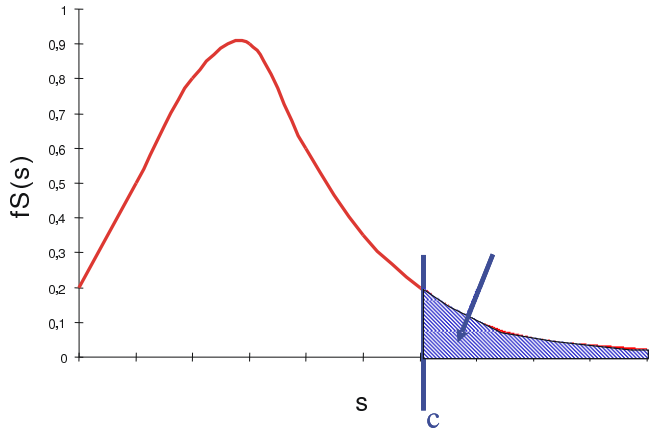

Abbildung 2.38: Prinzip des Vorgehens beim Testen.

- Bestimme die Verteilung $S(.)$ unter der Voraussetzung, dass H_0 gilt.
- Ermittle die kritischen Werte c_α (oder $c_{1-\alpha}$) ab denen H_0 zum Niveau α verworfen wird.

Eine Skizze des Prinzips ist in Abbildung 2.38 zu sehen. Der Wert von c_α wird so bestimmt, dass $P[S(Y_1, \ldots, Y_n) > c_\alpha | H_0] = \alpha$. Bei bekannter Verteilung von $S(.)$ ist die Bestimmung kein Problem. Übliche Werte für α sind 0.05 (signifikant) und 0.01 (hochsignifikant). Grundsätzlich kann ein Testverfahren einige schlechte Generatoren identifizieren, indem sie den Test nicht passieren. Es ist aber nicht möglich, die generelle Güte eines Generators nachzuweisen.

Der bisher beschriebene Testansatz basiert auf Stichproben und wird als empirischer Test bezeichnet. Alternativ gibt es einige theoretische Testverfahren, die auf Basis der Struktur und Parameter eines Generators Aussagen machen. Theoretische Testverfahren sind oft komplex und mathematisch anspruchsvoll, machen aber in der Regel Aussagen über alle generierten Zufallszahlen, während empirische Testverfahren nur die verwendet Stichprobe nutzen und damit für unterschiedliche Stichproben auch zu unterschiedlichen Resultaten kommen können.

Wir stellen im Folgenden erst einige empirische Testverfahren und anschließend theoretische Testverfahren vor. Für den Test, dass eine Gleichverteilung vorliegt können der Chi-Quadrat-Test und der Kolmogorov-Smirnov-Test verwendet werden, die in allgemeinerer Form als Anpassungstests in Abschnitt 2.4 vorgestellt werden.

Ein erstes Testverfahren zum Test der Runs-Test, der in verschiedenen Varianten existiert. Wir betrachten hier die einfachste Variante, die jeweils Paare von Zufallszahlen in Klassen einteilt (siehe auch [5, S. 270]). Ein Paar x_i, x_{i+1} gehört zur Klasse $+$, wenn $x_i < x_{i+1}$, ansonsten gehört es zur Klasse $-$. Für eine Stichprobe entsteht eine Sequenz von $+$ und $-$ Zeichen. Ein Run ist eine maximale Subsequenz identischer Zeichen. Sei n die Größe der Stichprobe und R_n die Zufallsvariable, die die Anzahl der Runs beschreibt. Für große n (laut Literatur reicht i.d.R. schon $n > 20$) ist R_n approximativ normalverteilt mit

$$E(R_n) = \frac{2n-1}{3} \quad \text{und} \quad \sigma^2(R_n) = \frac{16n-29}{90}.$$

Damit ist $Z = (R_n - (2n-1)/3)/\sqrt{(16n-29)/90} \sim N(0,1)$. Auf Basis der kritischen Werte der Normalverteilung (siehe Abbildung 2.39) kann dann die Hypothese der Unabhängigkeit angenommen oder verworfen werden. Sei r die Anzahl der Runs bei in einer Stichprobe der Größe n und $z = (r-(2n-1)/3)/\sqrt{(16n-29)/90}$, dann wird die Hypothese angenommen, wenn der Wert von z im mittleren, nicht schraffierten Bereich liegt. Werte im schraffierten Bereich führen zur Ablehnung. Die kritischen Wert der Normalverteilung werden auch mit $-z_{\alpha/2}$ und $z_{\alpha/2}$ bezeichnet. Also die Hypothese wird angenommen falls $-z_{\alpha/2} \leq z \leq z_{\alpha/2}$ und wird ansonsten abgelehnt.

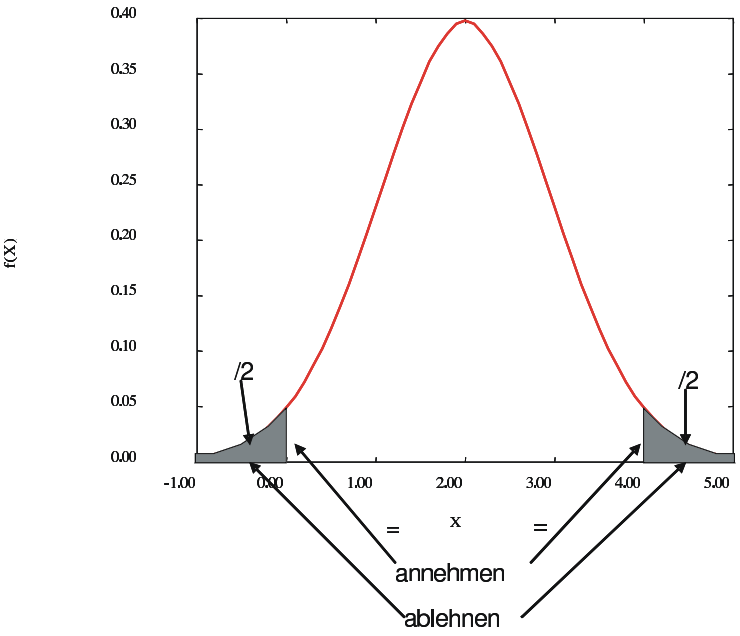

Abbildung 2.39: Kritische Werte der Normalverteilung und ihr Einsatz in Testverfahren.

Der schon bekannte Generator $x_{i+1} = (5 \cdot x_i)(\mod 17)$ mit $x_0 = 5$ erzeugt die folgende Stichprobe der Länge 16:

8	6	13	14	2	10	16	12	9	11	4	3	15	7	1	5
	+	+		+	+			+			+			+	
-			-			-	-		-	-		-	-		

Dies ergibt 10 Runs. Laut Theorie gilt $R(R_{16}) = (2 \cdot 16 - 1)/3 = 10.333$. Damit liegt eine gute Übereinstimmung vor. Für $\alpha = 0.05$ würde die Hypothese H_0 (= Zufallszahlen sind unabhängig) für $r \in [8, 13]$ akzeptiert. In der vorgestellten Form macht der Runs-Test keine Aussagen darüber, ob die Werte $[0, 1)$-gleichverteilt sind, sondern testet nur die Unabhängigkeit.

Ein weiteres Verfahren zum Test auf Unabhängigkeit ist der Test auf Autokorrelation. Seien $X_1, \ldots, X_n$ identisch verteilte Zufallsvariablen mit Erwartungswert $E(X)$ und Varianz $\sigma^2(X)$. Der Autokorrelationskoeffizient der Ordnung s ist definiert als

$$\rho(s) = \frac{\sum_{i=1}^{n-s} E((X_{i+s} - E(X))(X_i - E(X)))}{\sum_{i=1}^{n-s} E((X_i - E(X))^2)} = \frac{C(X_i, X_{i+s})}{\sigma^2(X)} = \frac{C_s}{C_0}.$$

Es gilt $-1 \leq \rho(s) \leq 1$ und für unabhängige X_i und X_{i+s} ist $\rho(s) = 0$. Die Umkehrung der letzten Aussage muss allerdings nicht gelten. Beim Test auf Autokorrelation wird überprüft inwieweit $\rho(s) \approx 0$ angenommen werden kann. Wenn statt der Zufallsvariablen eine konkrete Stichprobe eingesetzt, so benötigt man eine Schätzfunktion für den Wert von $\rho(s)$. Für $[0, 1)$-verteilte Zufallsvariablen X_i gilt $E(X) = 1/2$ und $\sigma^2(X) = C_0 = 1/12$. Da $C_s = E(X_i X_{i+s}) - E(X_i)E(X_{i+s})$ (siehe (2.2)), gilt auch

$$\rho(s) = \frac{C_s}{C_0} = \frac{E(X_i X_{i+s}) - E(X_i)E(X_s)}{\sigma^2(X)} = \frac{E(X_i X_{i+s}) - 1/4}{1/12} = 12E(X_i X_{i+s}) - 3$$

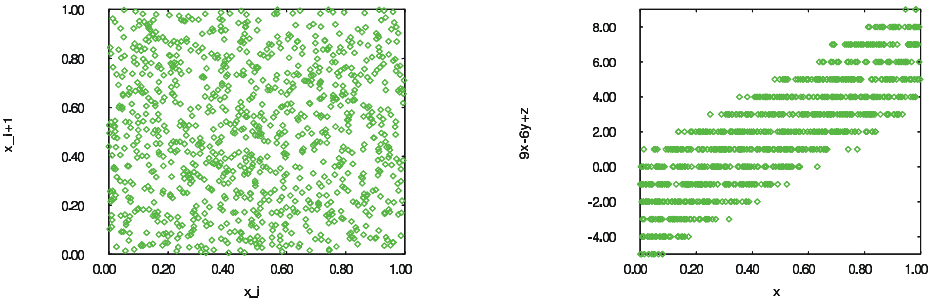

Abbildung 2.40: Darstellung von ZZs Tupeln und Tripeln für den RANDU-Generator.

Ein Schätzer für $\rho(s)$ lautet dann

$$\tilde{\rho}(s) = \frac{12}{h+1}\sum_{k=0}^{h} X_{1+ks} \cdot X_{1+(k+1)s} - 3$$

mit $h = \lfloor (n-1)/s \rfloor - 1$. $\tilde{\rho}(s)$ ist im Gegensatz zu $\rho(s)$ eine Zufallsvariable, deren konkrete Realisierung von der Stichprobe abhängt. Wie in [5, S. 279] beschrieben ist $\tilde{\rho}(s)$ für unabhängig identisch $[0,1)$-verteilte X_i normalverteilt mit Erwartungswert 0 und Standardabweichung $(\sqrt{13h+7})/(h+1)$. Damit kann für eine konkrete Stichprobe $x_1, \ldots, x_n$ der Wert

$$\hat{\rho}(s) = \frac{12}{h+1}\sum_{k=0}^{h} x_{1+ks} \cdot x_{1+(k+1)s} - 3$$

berechnet werden und $z = \hat{\rho}(s) \cdot (h+1)/(\sqrt{13h+7})$ gegen die kritischen Werte einer $N(0,1)$-Verteilung getestet werden.

Theoretische Testverfahren machen Aussagen über alle erzeugten Zufallszahlen. Wenn der Generator alle Zahlen aus dem Intervall $[0, m)$ erzeugt, also volle Periode hat, dann ist der Mittelwert der erzeugten Zufallszahlen $1/2 - 1/(2m)$ und die Varianz $1/12 - 1/(12m^2)$. Beide Werte liegen für große m sehr nahe an den "wahren" Werten.

Der wichtigste theoretische Tests leiten die Struktur der generierten Zufallszahlen im höher dimensionalen Raum her. Sei $x_1, x_2, \ldots$ die Sequenz der erzeugten Zufallszahlen. Überlappende d-Tupel $(x_1, \ldots, x_d), (x_2, \ldots, x_{d+1}), \ldots$ definieren jeweils Punkte im d-dimensionalen Hyperraum. Für LCGs fallen die Punkte auf relativ wenige Ebenen im Hyperraum der Dimension $d-1$. Im zweidimensionalen liegen die Punkte alle auf einem Gitter. Außerhalb dieses Gitters sind keine Punkte erreichbar. Die Qualität eines Generators hängt nun von der Anzahl der Hyperebenen ab, auf denen Punkte liegen können. Tests wie der Spektraltest oder Gittertest dienen zur Bewertung der Struktur in verschiedenen Dimensionen. Für niedrige Dimensionen kann man eine graphische Darstellung wählen. Höhere Dimensionen erfordern Transformationen.

Abbildung 2.40 zeigt für den Generator RANDU $(x_{i+1} = 65539 \cdot x_i)(\mod 2^{31}))$ die generierten Tupel (linke Seite der Abbildung) und die durch die Transformation $9x_i - 6x_{i+1} + x_{i+2}$ ins zweidimensional transformierte dreidimensionale Darstellung (rechte Seite der Abbildung). Während die zweidimensionale Darstellung recht zufällig aussieht, kann man an der transformierten dreidimensionalen Darstellung erkennen, dass die Werte auf sehr wenigen Hyperebenen im Dreidimensionalen liegen und der Generator schlechte statistische Eigenschaften hat.

Zusammenfassend kann man zum Testen von Zufallszahlengeneratoren die folgenden Aussagen machen:

- Es existiert eine Vielzahl unterschiedlicher Test und es ist unklar, welche Tests die besten Ergebnisse liefern.
- Tests können keine beweisbar guten Zufallszahlengeneratoren liefern, sondern nur schlechte aussondern.

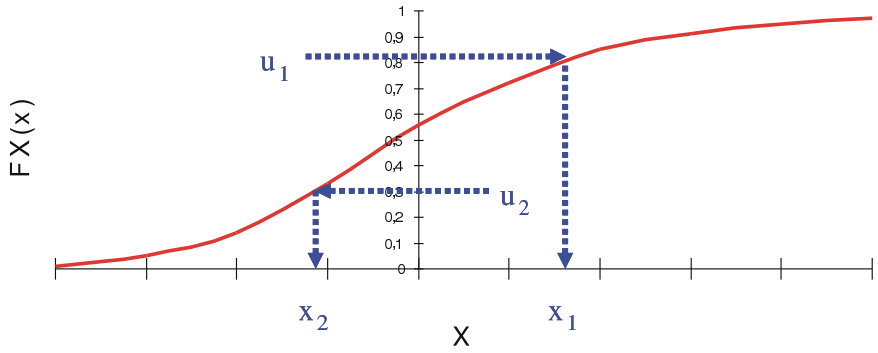

Abbildung 2.41: Skizze der inversen Transformation.

- Einige Generatoren wurden aufwändig getestet und haben sich bei allen Tests als "gut" bzgl. des Tests herausgestellt. Generatoren dieser Art sollten möglichst verwendet werden.
- Durch die Nutzung unterschiedlicher Generatoren in Simulationsläufen können Verzerrungen durch einzelne Generatoren entdeckt werden. Zur Umsetzung des Ansatzes muss das verwendete Simulationswerkzeug mehrere Generatoren beinhalten oder neue Generatoren müssen leicht integrierbar sein.

Transformation von $[0,1)$-verteilten Zufallszahlen

In den bisherigen Schritten wurden $[0,1)$-gleichverteilte Zufallszahlen erzeugt und bzgl. ihrer Güte bewertet. Für eine Simulation werden aber Zufallszahlen aus Verteilungsfunktion $FX(x)$ benötigt. Durch eine Transformation der Zufallszahlen u_i aus der $[0,1)$-Gleichverteilung in Zufallszahlen x_i aus der gewünschten Verteilung $FX(x)$ werden die gewünschten Zufallszahlen erzeugt. Unterschiedliche Transformationsmethoden werden zum Abschluss dieses Abschnitts vorgestellt. In der Regel liefern die Transformationen exakte Resultate, so dass gute $[0,1)$-gleichverteilte Zufallszahlen für gute Zufallszahlen aus einer beliebigen Verteilung sorgen.

Zur Erinnerung sei noch einmal erwähnt, dass für unsere $[0,1)$-gleichverteilten Zufallszahlen u gilt

$$FU(u) = \begin{cases} 0 & \text{falls } u < 0 \\ u & \text{falls } 0 \leq u < 1 \\ 1 & \text{sonst} \end{cases} \qquad fU(u) = \begin{cases} 1 & \text{falls } 0 \leq u < 1 \\ 0 & \text{sonst} \end{cases}$$

Für die erste Transformationsmethode nehmen wir an, dass die Verteilungsfunktion $FX(x)$ kontinuierlich ist, sowie für $x_1 < x_2$ mit $0 < F(x_1) \leq F(x_2) < 1$ auch $F(x_1) < F(x_2)$ gilt. Ist dies der Fall, so existiert die Umkehrfunktion F^{-1} von F. Das folgende, als *Inverse Transformation* bezeichnete Vorgehen ist naheliegend.

1. Generiere u aus eine $[0,1)$-Gleichverteilung
2. Transformiere $x = F^{-1}(u)$

Das Vorgehen wird noch einmal in Abbildung verdeutlicht. Formal gilt $P[X \leq x] = P[F^{-1}(u) \leq x] = P[u \leq F(x)] = F(x)$.

Die inverse Transformation soll am Beispiel der Exponentialverteilung kurz vorgestellt werden. Für die Exponentialverteilung $F(x) = 0$ falls $x \leq 0$ und $F(x) = 1 - e^{\lambda x}$ für $x > 0$. Sei u aus einer $(0,1)$-Gleichverteilung gezogen, dann werden die folgenden Schritte durchgeführt.

$$\begin{aligned} 1 - e^{-\lambda x} = u \Rightarrow \quad & e^{-\lambda x} = 1 - u \Rightarrow \\ -\lambda x = \ln(1-u) \Rightarrow \quad & x = \tfrac{\ln(1-u)}{-\lambda} \end{aligned}$$

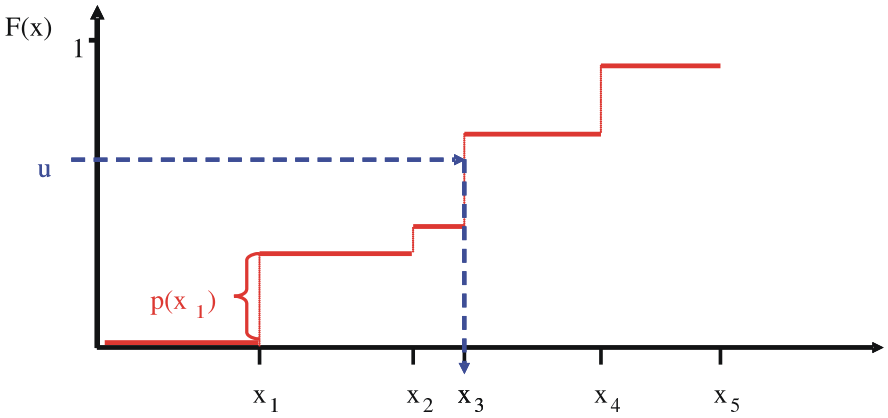

Abbildung 2.42: Inverse Transformation für eine diskrete Verteilung.

Da $1-u$ genauso wie u $(0,1)$-gleichverteilt ist, kann $1-u$ durch u ersetzt werden. Damit ergibt sich die folgende Transformation zur Erzeugung exponentiell verteilter Zufallszahlen.

1. Generiere u aus einer $(0,1)$-Gleichverteilung
2. Transformiere $x = \ln(u)/(-\lambda)$

Es stellt sich natürlich die Frage, ob das Vorgehen auf alle kontinuierlichen Verteilungen anwendbar ist. Dies gilt leider nicht, da $F^{-1}(x)$ in geschlossener Form verfügbar sein muss, um die Transformation durchzuführen. Dies ist nicht bei allen Verteilungen der Fall. Es gibt allerdings für zahlreiche kontinuierliche Verteilungen inverse Transformationen (siehe [11, Kap. 8.3]). Bevor wir zu anderen Ansätzen zur Generierung Zufallszahlen aus kontinuierlichen Verteilungen kommen, soll erst die inverse Transformation für diskrete Verteilungen kurz vorgestellt werden.

Offensichtlich funktioniert das Vorgehen für kontinuierliche Verteilungen im diskreten Fall nicht, da $F(x)$ eine Treppenfunktion ist, für die keine Umkehrfunktion gebildet werden kann. Man kann allerdings im diskreten Fall eine sehr ähnliche Vorgehensweise anwenden, wie in Abbildung 2.42 gezeigt wird. Für diskrete Verteilungen gilt offensichtlich $F(x) = P[X \leq x] = \sum_{x_i \leq x} p(x_i)$. Damit ergibt sich folgendes Vorgehen der inversen Transformation bei diskreten Zufallsvariablen.

1. Generiere u aus eine $[0,1)$-Gleichverteilung
2. Finde eine ganze Zahl I, so dass $\sum_{i<I} p(x_i) < u \leq \sum_{i \leq I} p(x_i)$ und liefern x_I als Wert zurück.

Für diskrete Verteilungen mit vielen Werten kann das Suchen nach dem geeigneten I aufwändig sein, insbesondere wenn linear gesucht wird und am Anfang viele Punkte mit kleinen Wahrscheinlichkeiten liegen. Deshalb sollte in diesen Fällen die Intervalle absteigen nach den Wahrscheinlichkeiten $p(x_i)$ geordnet werden. Das Vorgehen ist auch für diskrete Verteilungen mit unendlich vielen Werten, wie dem Poisson Prozess, geeignet.

Die inverse Transformation sollte immer dann eingesetzt werden, wenn die Umkehrfunktion in geschlossener Form vorliegt und leicht ausgewertet werden kann. Falls $F^{-1}(x)$ nicht in geschlossener Form vorliegt, so kann prinzipiell der Wert numerisch berechnet werden. Dies ist aber für manche Verteilungsfunktionen recht aufwändig und unter Umständen auch numerisch instabil. Aus diesem Grund sollen noch einige andere Möglichkeiten der Transformation vorgestellt werden.

Falls sich eine Zufallsvariable X als Summe von Zufallsvariablen Y_i $(i = 1, \ldots, I)$ mit Verteilungsfunktionen $F_i(y)$ darstellen lässt und für die einzelnen $F_i(y)$ Generierungsmethoden vorhanden sind, so können mit dem folgenden Algorithmus Realisierung von X generiert werden.

$x = 0$;
for $i = 1$ to I do

ziehe ZZ y aus $F_i(y)$;
$x = x + y$;
end for
return(x) ;

Das Verfahren bezeichnet man als *Konvolutionsmethode*. Ein Beispiel für eine Verteilung, die sich mit der Konvolutionsmethode behandeln lässt, ist die Erlang-k-Verteilung (siehe Abbildung 2.35). In diesem Fall sind alle $F_i(y)$ identisch, es muss aber natürlich in jedem Schleifendurchlauf eine neue Zufallszahl gezogen werden.

Falls sich eine Zufallsvariable X als konvexe Linearkombination von Zufallsvariablen Y_i ($i = 1, \ldots, I$) mit Gewichten p_i und Verteilungsfunktionen $F_i(y)$ darstellen lässt und für die einzelnen $F_i(y)$ Generierungsmethoden vorhanden sind, so können mit dem folgenden Algorithmus Realisierung von $X = \sum_{i=1}^{I} p_i \cdot Y_i$ generiert werden.

generiere i gemäß Verteilung p_i ;
ziehe ZZ x aus $F_i(x)$;
return(x) ;

Das Verfahren bezeichnet man als *Kompositionsverfahren*. Es kann zum Beispiel zur Generierung hyperexponentiell-verteilter Zufallsvariablen (siehe Abbildung 2.35) eingesetzt werden.

Die bisher vorgestellten Methoden generieren Zufallszahlen direkt. Dies bedeutet, dass zur Generierung einer Zufallszahl der gewünschten Verteilung eine vorgegebene Anzahl von $[0, 1)$-verteilten Zufallszahlen generiert werden muss. Bei der nun vorgestellten *Verwerfungsmethode* ist dies nicht der Fall. Es sollen Realisierungen einer Zufallsvariablen X mit bekannter Dichtefunktion $f_X(x)$ gezogen werden. Weiterhin existiere eine Zufallsvariable Y mit Dichtefunktion $f_Y(x)$ für die ein Generierungsverfahren bekannt ist und ein Parameter α, so dass $\alpha \cdot f_Y(x) \geq f_X(x)$ für alle x ist. Da $f_X(x)$ und $f_Y(x)$ Dichtefunktionen sind, gilt $\int_{-\infty}^{\infty} f_X(x)dx = \int_{-\infty}^{\infty} f_Y(x)dx = 1$, so dass $\alpha > 1$ sein muss, wenn $f_X(x)$ und $f_Y(x)$ nicht identisch sind. Wenn die Voraussetzungen erfüllt, sind kann folgender Generierungsalgorithmus eingesetzt werden.

repeat
ziehe ZZ y gemäß $f_Y(x)$;
ziehe ZZ x aus einer $[0, \alpha \cdot f_Y(x))$-Gleichverteilung
until $x \leq f_X(y)$;
return(y) ;

Das Ziehen einer Zufallszahl aus einer $[a, b)$-Gleichverteilung ($a < b$) wird einfach dadurch realisiert, dass eine $[0, 1)$-verteilten Zufallszahl u mittels $a + u \cdot (b - a)$ transformiert wird.

Wir wollen uns kurz den Beweis anschauen, warum mit dem Algorithmus wirklich Zufallszahlen der gesuchten Verteilung generiert werden. Dazu muss gezeigt werden, dass $P[X \leq x] = \int_{-\infty}^{x} f_X(y)dy$ für jedes x gilt. Da die beiden Zufallszahlen y und x unabhängig voneinander sind, gilt $P[X \leq x] = P[Y \leq x|A]$ wobei A die Bedingung ausdrückt, dass y akzeptiert wird. Die bedingte Wahrscheinlichkeit kann wie folgt dargestellt werden (siehe Seite 61)

$$P[Y \leq x|A] = \frac{P[A \cap Y \leq x]}{P[A]} \tag{2.3}$$

Für die Wahrscheinlichkeit $P[A]$ gilt

$$P[A|Y = y] = P\left[u \leq \frac{f_X(y)}{\alpha \cdot f_Y(y)}\right] = \frac{f_X(y)}{\alpha \cdot f_Y(y)}$$

wobei u eine $[0, 1)$-gleichverteilte Zufallszahl ist und damit gilt

$$P[A] = \int_{-\infty}^{\infty} \frac{f_X(y)}{\alpha \cdot f_Y(y)} \cdot f_Y(y)dy = \frac{1}{\alpha}$$

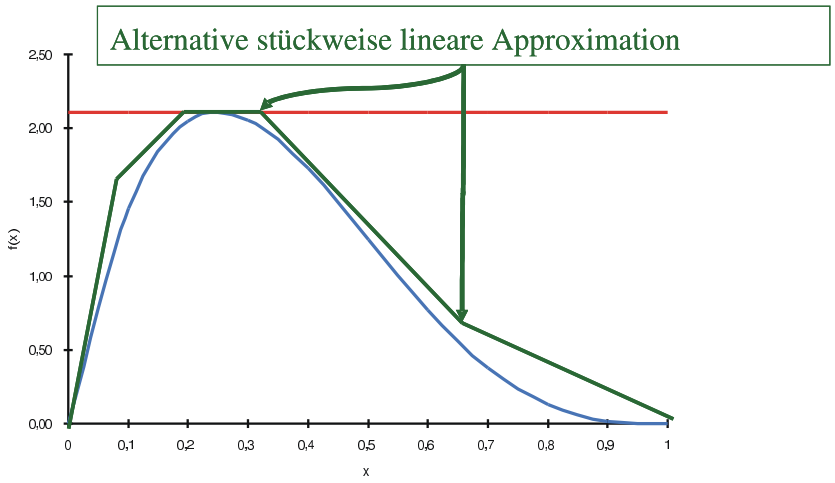

Abbildung 2.43: Anwendung der Verwerfungsmethode am Beispiel einer $\beta(2,4)$-Verteilung.

Für den Term im Zähler von (2.3) gilt

$$\begin{aligned} P[A \cap Y \leq x] &= \int_{-\infty}^{x} P[A \cap Y \leq x | Y = y] \cdot f_Y(y) dy \\ &= \int_{-\infty}^{x} P[A|Y = y] \cdot f_Y(y) dy \\ &= \frac{1}{\alpha} \int_{-\infty}^{x} f_X(y) dy \end{aligned}$$

Damit gilt auch

$$P[X \leq x] = P[Y \leq x | A] = \frac{P[A \cap Y \leq x]}{P[A]} = \frac{\alpha^{-1} \cdot \int_{-\infty}^{x} f_X(y) dy}{\alpha^{-1}} = \int_{-\infty}^{x} f_X(y) dy$$

Die Anwendung der Verwerfungsmethode soll nun kurz am Beispiel einer $\beta(2,4)$-Verteilung mit $f(x) = 20 \cdot x \cdot (1-x)^3$ falls $0 \leq x \leq 1$ und 0 sonst vorgeführt werden. Die Dichtefunktion ist durch ein Rechteck der Höhe 2.11 beschränkt, wie in Abbildung 2.43 gezeigt. Damit ergibt sich folgender Generierungsalgorithmus

repeat
    ziehe y aus einer $[0,1)$-Gleichverteilung ;
    ziehe x aus einer $[0,2.11)$-Gleichverteilung ;
until $x \leq 20 \cdot y \cdot (1-y)^3$;

Die Effizienz der Verwerfungsmethode hängt von den zwei folgenden Punkten ab.

1. Der Effizienz der Generierung von Zufallszahlen mit Dichtefunktion $f_Y(x)$
2. Der Wahrscheinlichkeit, dass eine Zufallszahl akzeptiert wird

Die Wahrscheinlichkeit in Punkt 2 lautet $1 - \int_{-\infty}^{\infty} (\alpha \cdot f_Y(x) - f_X(x)) dx / \alpha$ und entspricht der Differenz der Funktionen $f_X(x)$ und $\alpha \cdot f_Y(x)$. Durch eine bessere Anpassung der dominierenden Funktion $\alpha \cdot f_Y(x)$ an $f_X(x)$ kann die Wahrscheinlichkeit der Verwerfung reduziert werden. Wird dazu eine komplexere Funktion $f_Y(x)$ verwendet, so steigt dabei in der Regel der Aufwand in Schritt 1.

Für das Beispiel der $\beta(2,4)$-Verteilung könnte statt eines Rechtecks auch ein stückweise lineare Approximation der Dichtefunktion verwendet werden, wie auch in Abbildung 2.43 angedeutet. Dadurch lässt sich die Verwerfungswahrscheinlichkeit deutlich reduzieren, die Generierung von y erfordert aber mehr Aufwand.

Die Generierung von normalverteilten-Zufallszahlen ist von großer praktischer Bedeutung, kann mit den bisher vorgestellten Methoden aber nicht effizient durchgeführt werden. Die Umkehrfunktion $F^{-1}(x)$ der Verteilungsfunktion existiert nicht in geschlossener Form, da für die Verteilungsfunktion nicht einmal eine geschlossene Darstellung existiert. Auch die Verwerfungsmethode ist nicht effizient anwendbar, da keine Dichtefunktion bekannt ist, aus der effizient Zufallszahlen generiert werden können und die gleichzeitig genügend nahe an der Dichtefunktion der Normalverteilung liegt. Es gibt aber mehrere Methoden um normalverteilte Zufallszahlen zu generieren. Zwei dieser Methoden sollen kurz vorgestellt werden.

Es sei noch einmal angemerkt, dass es ausreicht $N(0,1)$-verteilte Zufallszahlen zu generieren, da für $X \sim N(0,1)$ die Zufallsvariable $Y = \mu + \sigma \cdot X$ aus einer $N(\mu,\sigma)$-Verteilung stammt. Die erste Generierungsmethode basiert auf dem zentralen Grenzwertsatz. Nach dem zentralen Grenzwertsatz konvergiert die Summe unabhängig identisch verteilter Zufallszahlen gegen eine Normalverteilung. Diese Konvergenz ist besonders schnell für gleichverteilte Zufallszahlen. Ein einfacher Ansatz besteht nun darin, einfach n $[0,1)$-verteilte Zufallszahlen zu addieren, davon $n/2$ (den Erwartungswert) abzuziehen und durch $\sqrt{n/12}$ (die Standardabweichung) zu dividieren. Das Ergebnis konvergiert für wachsendes n gegen eine $N(0,1)$-Verteilung. Also eine $N(0,1)$-verteilte Zufallszahl x wird nach der folgenden Formel erzeugt.

$$x = \frac{\left(\sum_{i=1}^{n} u_i\right) - \frac{n}{2}}{\sqrt{n/12}}$$

wobei u_i aus eine $[0,1)$-Verteilung gezogen werden. Oft wird $n = 12$ gewählt, was ausreichend ist, um eine relativ gute Approximation der Normalverteilung zu erreichen. Offensicht können in diesem Fall keine Werte größer 6 oder kleiner -6 erzeugt werden. Die Wahrscheinlichkeit, dass eine $N(0,1)$-Verteilung einen Wert außerhalb des Intervalls $[-6,6]$ annimmt liegt aber bei ungefähr $5.7 \cdot 10^{-7}$ und ist damit für die meisten Anwendungen zu vernachlässigen. Der Nachteil dieser Art der Generierung von normalverteilten Zufallszahlen ist, dass für eine Zufallszahl aus $N(0,1)$ 12 Zufallszahlen aus der $[0,1)$-Gleichverteilung erzeugt werden müssen. Dies ist natürlich aufwändig.

Eine andere Methode zur Generierung von $N(0,1)$-verteilten Zufallszahlen geht auf Box und Muller (1958) zurück (siehe auch [11, S. 465]). Seien u_1 und u_2 zwei $(0,1)$-gleichverteilte Zufallszahlen, dann sind

$$x_1 = \cos(2 \cdot \pi \cdot u_1) \cdot \sqrt{-2 \cdot \ln(u_2)} \quad \text{und} \quad x_2 = \sin(2 \cdot \pi \cdot u_1) \cdot \sqrt{-2 \cdot \ln(u_2)}$$

zwei unabhängig identisch $N(0,1)$-verteilte Zufallszahlen. Man benötigt damit zwei gleichverteilte Zufallsvariablen, um zwei normalverteilte Zufallszahlen zu erzeugen. Man kann allerdings zeigen, dass die Unabhängigkeit von x_1 und x_2 nicht gegeben ist, wenn u_1 und u_2 zwei konsekutive Werte aus einem LCG sind. Es sollten also entweder unabhängige Ströme eines LCGs zu Generierung von u_1 und u_2 verwendet werden oder eine andere Methode zur Generierung. Unabhängige Ströme entstehen dadurch, dass mit unterschiedlichen Saaten $x_0^{(1)}$ und $x_0^{(2)}$ begonnen wird und u_1, u_2 aus unterschiedlichen Sequenzen gezogen werden. Die Saaten sollten dabei so gewählt werden, dass die resultierenden Sequenzen sich möglichst spät überlappen. In den meisten Generatoren sind Mechanismen vorgesehen, um adäquate Saaten zu generieren, wie wir später noch sehen werden.

Bisher haben wir die Generierung von unabhängigen Zufallszahlen betrachtet und in den meisten Simulationsmodellen werden auch unabhängige Zufallszahlen eingesetzt. Viele zufällige Prozesse in der Realität sind aber nicht unabhängig. So weist die Größe und das Gewicht eines Menschen sicherlich eine Korrelation auf, genauso wie die Temperatur und die Regenmenge oder die Zahl Ein-/Ausgabeoperationen und der CPU-Zeitbedarf eines Jobs auf einem Rechner. Die Annahme von unabhängigen Zufallsvariablen in der Simulation, für Prozesse, die in der Realität korreliert sind, kann zu starken Verfälschungen führen. So sind zum Beispiel Zwischenankunftszeiten von Nachrichten an einem Router in einem Rechnernetz stark positiv korreliert. Wird im Simulationsmodell des Routers diese Korrelation nicht beachtete und die Pufferbelegung analysiert, so liefert das Modell eine deutlich geringer Wahrscheinlichkeit eines Pufferüberlaufs als in der Realität beobachtet werden kann und führt damit zu einer Unterdimensionierung des Routers.

Wir wollen die Generierung abhängiger Zufallsvariablen nur sehr kurz betrachten. Es soll allerdings darauf hingewiesen werden, dass vor der Generierung zwei andere Probleme auftreten. Zum einen muss die Abhängigkeit geschätzt werden, was nicht einfach ist und gerade bei starken Abhängigkeiten sehr viele Beobachtungen erfordert. Zum anderen muss die resultierende Verteilung kompakt dargestellt werden, was in vielen Fällen ebenfalls problematisch ist.

Formal liegt ein Zufallsvektor $\mathbf{X} = (X_1, \ldots, X_d)$ mit Verteilungsfunktion $F_{X_1,\ldots,X_d}(x_1, \ldots, x_d)$ vor. Wenn sich die Verteilungsfunktion als Menge von d bedingten Verteilung der Form $F_i(x_i|x_1, \ldots, x_{i-1})$ darstellen lässt, so ist ist die Generierung relativ einfach. Es muss für jedes $F_i(x_i|x_1, \ldots, x_{i-1})$ ein geeigneter Generator gefunden werden und anschließend kann x_1 gemäß $F_1(x)$ generiert werden, anschließend x_2 gemäß $F_2(x|x_1)$ usw. Bis schließlich alle x_i vorhanden ist und der resultierende Vektor zurück gegeben wird.

Als einfaches Beispiel soll die Generierung von Zufallszahlen aus einer bivariaten Normalverteilung betrachtet werden. X_1 und X_2 seien zwei korrelierte normalverteilte Zufallsvariablen mit Erwartungswerten μ_i, Standardabweichungen σ_i und Korrelationskoeffizient $\rho = C(X_1, X_2)/(\sigma_1 \cdot \sigma_2)$. Dann können mit dem folgenden Algorithmus Realisierungen generiert werden.

1. Erzeuge z_1 und z_2 als unabhängige $N(0, 1)$-verteilte Zufallszahlen.
2. $x_1 = \mu_1 + \sigma_1 \cdot z_1$
3. $x_2 = \mu_2 + \sigma_2 \cdot (\rho \cdot z_1 + \sqrt{(1 - \rho^2)} \cdot z_2)$

x_1 und x_2 sind dann Realisierungen aus einer bivariaten Normalverteilung. Der beschriebene Ansatz kann auf multivariate Normalverteilungen erweitert werden.

In der Praxis werden oft stochastische Prozesse zur Modellierung korrelierter Vorgänge eingesetzt. Die Behandlung dieser Thematik geht allerdings über den Stoff der Vorlesungen hinaus.

2.4 Modellierung von Eingabedaten

Nachdem wir die Methoden zur Beschreibung und Realisierung von Zufallszahlen in Simulationsmodellen kennen gelernt haben, stellt sich natürlich die Frage, wie man an die Verteilungen zur Beschreibung der Parameter eines Simulationsmodells kommt. Dieser Frage wird in diesem Abschnitt nachgegangen. Es hat sich herausgestellt, dass viele reale Vorgänge durch Zufallsvariablen oder stochastische Prozesse in einem Modell adäquat abgebildet werden können. Gleichzeitig sollte klar geworden sein, dass eine genügend genaue Modellierung realer Abläufe notwendig ist, um eine ausreichende Abbildungsgüte des Modells zu erreichen.

Es gibt zwei wesentliche Quellen zur Datengewinnung. Die erste Quelle ist das a priori Wissen, welches vorhanden ist. So können unter Umständen Resultate aus vorherigen oder ähnlichen Modellierungen wiederverwendet werden oder es kann auf die vorhandene Erfahrung oder die Theorie zurückgegriffen werden. So ist aus der Theorie bekannt, dass gewisse Prozesse sich mit bestimmten Verteilungen gut modellieren lassen. Beispiele sind Weibull-Verteilungen für das Ausfallverhalten von Komponenten, Normalverteilungen für viele Größen aus einer großen Grundgesamtheit. Man muss allerdings darauf achten, ob die Bedingungen, die bisher galten, auch für die neue Modellierung gelten. Ein Beispiel wäre der Ankunftsprozess von Verbindungswünschen an einer Vermittlungsstelle. In der Vergangenheit konnte dieser Prozess sehr gut durch einen Poisson-Prozess beschrieben werden. Durch die Einführung der automatischen Wahlwiederholung und die Verwendung der Telefonleitungen für vielschichtige Dienste, änderte sich der Ankunftsprozess, so dass die Modellierung als Poisson-Prozess nicht länger adäquat ist.

Die zweite Quelle bei der Modellierung sind natürlich Daten aus Messungen, also Stichproben der zu modellierenden Größen. Im Idealfall werden die Messungen genau an dem System vorgenommen, welches durch das Modell abgebildet wird. Oftmals sollen aber gerade Systeme modelliert werden, die so nicht in der Realität vorhanden sind. Deshalb müssen Daten an ähnlichen Systemen oder mit Hilfe vorhandener und als gut befundener Modelle gewonnen werden. Wenn feststeht wo und wie Daten erhoben werden sollen, müssen die folgenden Schritte durchgeführt werden, um zu einer Repräsentation im Modell zu gelangen:

1. Die Daten müssen erhoben und zur Auswertung aufbereitet werden.
2. Es ist eine Entscheidung zu fällen, wie die gemessenen Daten im Modell dargestellt werden. Dazu existieren die folgenden Möglichkeiten:
   (a) Es wird eine Konstante zur Darstellung im Modell verwendet, der zugehörige Parameter ist damit deterministisch.
   (b) Man benutzt eine diskrete empirische Verteilung zur Repräsentation der Daten im Modell.
   (c) Man benutzt eine kontinuierliche empirische Verteilung zur Repräsentation der Daten im Modell.
   (d) Man benutzt eine stochastische Verteilung zur Repräsentation der Daten im Modell. Falls 2.d gewählt wurde, müssen noch folgende Schritte ausgeführt werden.
3. Auswahl des zu verwendenden Verteilungstyps.
4. Schätzung der Verteilungsparameter aus der Stichprobe.
5. Überprüfung der Passgüte der gewählten Verteilung durch einen Anpassungstest.

Die einzelnen Schritte sollen nun etwas näher beschrieben werden.

Methoden zur Datenerhebung

Die *Sammlung und Messung von Daten* ist ein aufwändiges und oft frustrierendes Unterfangen, dessen Zeitbedarf bei den meisten Modellierungsprojekten deutlich unterschätzt wird. Da die erhobenen Daten aber die Grundlage für die weiteren Schritte sind, können Nachlässigkeiten an dieser Stelle schnell dazu führen, dass die Realität nur sehr ungenau im Modell abgebildet wird. Insgesamt gilt das GIGO-Prinzip (Garbage in Garbage out), aus schlechten Daten können keine vernünftigen Modelle entstehen.

Die zentrale Frage lautet, was sind gute Daten oder was ist eine gute Stichprobe. Dies lässt sich nicht allgemein beantworten und hängt sehr stark von der Anwendung ab. Insgesamt ist das Ziel, auf Basis einer endlichen Stichprobe das übliche Verhalten zu beschreiben. Die Stichprobe muss also repräsentativ sein. Da viele Abläufe stark schwanken, ist vorab zu definieren, für welche Situationen das Verhalten repräsentativ sein soll. Wird zum Beispiel das Verkehrsaufkommen auf einer Straße untersucht, so kann man starke Unterschiede beim Verkehrsaufkommen in Abhängigkeit von der Uhrzeit und dem Wochentag beobachten. Je nach Ziel der Modellierung müssen Daten immer an bestimmten Tagen zu festgelegten Uhrzeiten erhoben werden, wenn zum Beispiel die Phase hohen Verkehrsaufkommens analysiert werden soll, oder es muss kontinuierlich gemessen werden, wenn der gesamte Ablauf nachgebildet wird. Auch an dieser Stelle ist also die Festlegung des Ziels der Modellierung von zentraler Bedeutung.

Bei der Datenerhebung können eine Reihe von Problemen auftreten. Dies gilt insbesondere, wenn auf Basis vorhandener Daten modelliert werden soll. Gerade bei Simulationsprojekten, die von externen Dienstleistern durchgeführt werden, was in der Praxis oft der Fall ist, muss dieser Weg gegangen werden, da eine unbeschränkte Datenerhebung nicht möglich oder nicht gewünscht ist. So wird und darf ein Telekommunikationsunternehmen keine detaillierte Erhebung der Verbindungsdaten erlauben. Dem stehen Datenschutz- und Wettbewerbsgründe entgegen.

Bei vorhandenen Daten kann es vorkommen, dass zu wenige Daten vorliegen. Der Stichprobenumfang ist zu gering oder die Daten sind nur als summarische Statistiken vorhanden. Im Extremfall liefern die Daten nur qualitative Informationen, aus denen sich keine quantitativen Größen ableiten lassen. Es gibt durchaus auch die Situation, dass zu viele Daten vorhandenen sind. Dies ist gerade dann der Fall, wenn automatische Messungen ablaufen oder Traces geschrieben werden. Beispiele findet man in der Informatik im Rechnernetzbereich, wo die Messung des Verkehrsaufkommens zu immensen Datenmengen führt, aber auch in den Naturwissenschaften insbesondere der Physik existieren zahlreiche Experimente, bei denen das Datenaufkommen selbst mit heutigen Rechnern

kaum analysierbar und speicherbar ist. Das Problem bei zu großen Datenmengen liegt darin, wesentliche Informationen zu extrahieren, was unter Umständen einen sehr hohen Aufwand erfordert oder mit verfügbaren Techniken nicht in vertretbarer Zeit möglich ist. Ein weiteres Problem ist die Verfügbarkeit falscher Daten. So müssen Daten zu der zu modellierenden Situation passen. Gerade bei Systemen, die sich in der Planungsphase befinden, muss auf Daten zurückgegriffen werden, die aus anderen Systemen stammen und deshalb oft nicht zur eigentlichen Zielstellung passen.

Die folgenden Hinweise dienen dazu die Datenerhebung zu unterstützen, können aber auch verwendet werden, um die Qualität vorhandener Daten zu bewerten.

1. Vor der eigentlichen Datenerhebung sollten einige Vorläufe stattfinden, um das Erhebungsintervall, die Auflösung der Messung und den Stichprobenumfang festzulegen. So ist bei stark schwankenden oder korrelierten Daten ein größerer Stichprobenumfang notwendig als bei unabhängigen und wenig variierenden Daten.

2. Falls mögliche sollten die Daten während der Erhebung analysiert werden, um so relevante von irrelevanten Informationen zu unterschieden.

3. Wenn mehrere Stichproben kombiniert werden, so ist sicherzustellen, dass die Stichproben homogen sind, d.h. aus identischen Verteilungen stammen. Dies kann mit entsprechenden Testverfahren überprüft werden.

4. Es ist darauf zu achten, inwieweit Daten zensiert (censored data) sind. Man spricht dann von zensierten Daten, wenn auf Grund der Datenerhebung/Messumgebung gewisse Effekte nicht beobachtbar sind. Ein Beispiel ist die Beobachtung der Ausfallzeit von Komponenten in einem Intervall $[0, T)$. In diesem Fall können offensichtlich keine Ausfallzeiten größer T beobachtet werden. Eine andere Art der Zensierung tritt auf, wenn nur zu diskreten Zeitpunkten gemessen wird und dadurch Werte aufgerundet werden.

5. Die erhobenen Daten sollten auf Korreliertheit bzw. Unkorreliertheit untersucht werden. Die dazu vorhandenen Methoden werden später vorgestellt.

6. Bei der Messung sollte zwischen Eingabe- und Ausgabedaten unterschieden werden. Zur Modellparametrisierung werden Eingabedaten benötigt, während Ausgabedaten zur Validierung eingesetzt werden. In einem Warteschlangennetz ist zum Beispiel die Zwischenankunftszeit eine Eingabegröße, während die Verzögerungszeit eine Ausgabegröße ist.

Repräsentation und Bewertung von Daten

Nachdem Daten vorhanden sind, ist zu entscheiden, wie diese im Modell repräsentiert werden. Im wesentlichen geht es darum, die Struktur und die Abhängigkeiten der Daten zu erkennen. Dazu existieren zwei Ansätze, nämlich Maßzahlen und graphische Darstellungen. Maßzahlen sind im Wesentlichen Schätzung von Momenten, Quantilen oder Korrelationskoeffizienten. Die zugehörigen Schätzer werden später beschrieben. An dieser Stelle sollen graphische Darstellungen vorgestellt werden. Sie dazu $(x_1, \ldots, x_n)$ eine Stichprobe und $(y_1, \ldots, y_n)$ sei die nach Größe geordnete Stichprobe (d.h. $y_i \leq y_{i+1}$). Eine graphische Darstellung gibt einen visuellen Eindruck der Stichprobe und muss vom Modellierer bewertet werden. Damit hat die Bewertung graphischer Darstellungen auch immer einen subjektiven Aspekt.

Die naheliegenste Form der Visualisierung sind Histogramme als Approximation der Dichtefunktion. Ein Histogramm wird dadurch erstellt, dass der Datenbereich, unter Umständen nach Elimination von Ausreißern, in gleich große Abschnitte/Zellen unterteilt wird. Die Daten werden dann den Zellen zugeordnet. Anschließend wird das Histogramm graphisch dargestellt, wobei die Höhe einer Zelle proportional zur Anzahl der Werte in der Zelle ist. Als freier Parameter der Histogrammdarstellung bleibt die Intervallbreite. Leider hängt der visuelle Eindruck manchmal von der Intervallbreite ab. Deshalb sollten immer mehrere Intervallbreiten zur Bewertung de Daten ausprobiert werden. Ist die Intervallbreite zu groß, so werden viele Effekte heraus gemittelt,

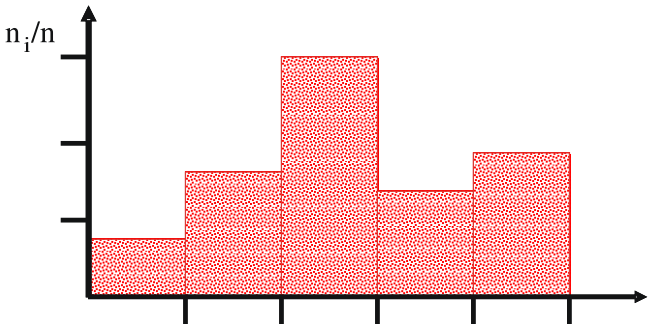

Abbildung 2.44: Beispiel eines Histogramms.

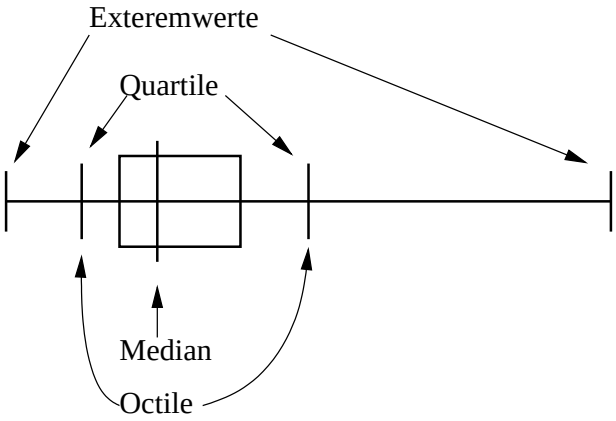

Abbildung 2.45: Beispiel für einen Box-Plot.

während eine zu kleine Intervallbreite, mit nur wenigen Werten pro Zelle, zu starken statistischen Schwankungen führt.

Eine Alternative zum Histogramm ist die graphische Darstellung von Maßzahlen. Insbesondere Schätzer für die Quantile der Verteilung eignen sich zur Visualisierung. Die Maßzahlen werden mit Hilfe der geordneten Stichprobe $(y_1, \ldots, y_n)$ definiert.Die folgenden Quantialschätzer sind von besonderer Bedeutung:

- Median y_i mit $i = (n+1)/2$
- Quartile y_j und y_{n-j+1} mit $j = (\lfloor i \rfloor + 1)/2$
- Octile y_k und y_{n-k+1} mit $k = (\lfloor j \rfloor + 1)/2$
- Extremwerte y_1 und y_n

Dabei wird $y_{m+0.5}$ für $m \in \mathbb{N}$ als $0.5 \cdot (y_m + y_{m+1})$ definiert.

Die genannten Maßzahlen können graphisch in einem sogenannten Box-Plot dargestellt werden, wie in Abbildung 2.45 gezeigt. Mit Hilfe von Box-Plots können wichtige Eigenschaften von Stichproben und Verteilungen visualisiert werden. So kann das Verhältnis des Median zu den Extremwerten oder dir Symmetrie der Verteilung abgelesen werden. Insbesondere sind Box-Plots unabhängig von frei zu wählenden Parametern.

Zur Darstellung der Abhängigkeiten zwischen Daten eignen sich Plots von Datentupeln, wie in Abbildung 2.46 gezeigt. Dabei werden aufeinander folgende Werte jeweils als ein Punkt im zweidimensionalen Raum dargestellt. Während in der linken Abbildung keine Abhängigkeiten erkennbar

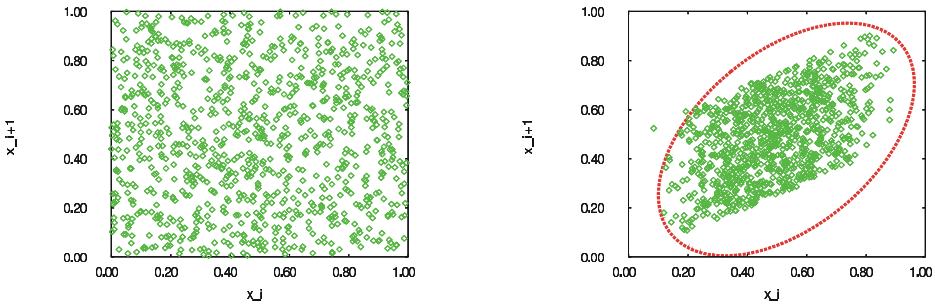

Abbildung 2.46: Beispiele für Tupel konsekutiver Werte aus einer Stichprobe.

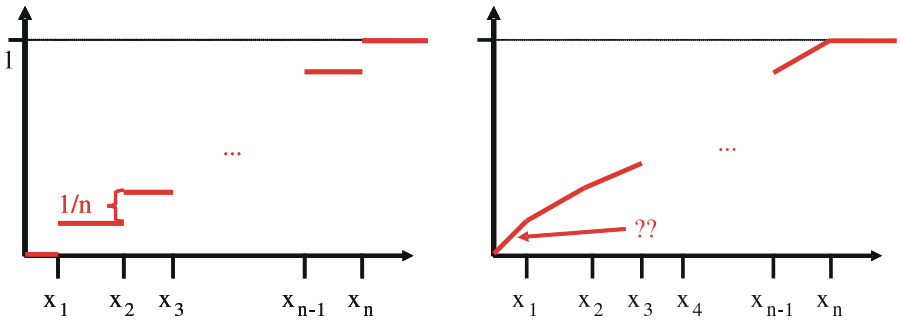

Abbildung 2.47: Diskrete und kontinuierliche empirische Verteilung.

sind, zeigt die rechte Abbildung eine deutlich positive Korrelation. Man kann die Darstellung auch benutzen, um Daten über eine Distanz k (≥ 1) in Beziehung zu setzen.

Nachdem die vorliegenden Daten analysiert wurden, sind sie im Simulationsmodell geeignet zu repräsentieren. Falls keine oder nur sehr geringe Schwankungen in den Daten beobachtet werden, so können sie durch eine Konstante im Modell beschrieben werden. Ansonsten gibt es drei unterschiedliche Möglichkeiten:

- Repräsentation durch eine theoretische Verteilung aus der Menge der verfügbaren Verteilungen. Die Verfügbarkeit von Verteilungen richtet sich u.a. nach der Menge der Verteilungen, die in der verwendeten Simulationssoftware verfügbar sind.
- Repräsentation der Daten durch eine diskrete empirische Verteilung.
- Repräsentation der Daten durch eine kontinuierliche empirische Verteilung.

Bevor wir die Auswahl-Anpassung theoretischer Verteilungen einführen und die Vor- und Nachteile theoretischer/empirischer Verteilungen diskutieren, soll kurz auf empirische Verteilungen zur Datenrepräsentation eingegangen werden.

Empirische Verteilungen zur Datenrepräsentation

Die Idee der empirischen Verteilung besteht darin, die Daten möglichst direkt als Repräsentanten im Modell zu verwenden und damit kein mathematisches Modell zur Datenbeschreibung zu generieren. Auf extreme Weise wird dies bei der Trace-getriebenen Simulation durchgeführt, bei der genau die Abläufe aus der Realität kopiert werden. Dies bedeutet, dass die verfügbare Datenmenge die maximale Länge des Simulationslaufs bestimmt und stochastische Einflüsse nur bzgl. der nicht durch die Trace-Daten beschriebenen Teile integriert werden können. Empirische Verteilungen benutzen dagegen die Daten zur Generierung von Zufallszahlen. Damit können beliebig lange Simulationsläufe durchgeführt werden und durch unterschiedliche Saaten des Zufallszahlengenerators werden auch unterschiedliche Sequenzen von Zufallszahlen aus den Daten erzeugt. Die beiden

grundsätzlichen Varianten empirischer Verteilungen, nämlich die diskrete und die kontinuierliche Version, werden in Abbildung 2.47 dargestellt.

Bei der diskreten empirischen Verteilung werden die Daten als direkte Repräsentanten gewählt. Wenn n Werte in der Stichprobe vorliegen, so wird mit Wahrscheinlichkeit $1/n$ einer der Werte ausgewählt und als Zufallszahl verwendet. Die zugehörige Verteilungsfunktion ist durch eine homogene Treppenfunktion mit Sprungstellen der Höhe $1/n$ gekennzeichnet. Das Ziehen einer neuen Zufallszahl erfolgt jeweils gedächtnislos. Alle Charakteristika der verwendeten Verteilung (z.B. Momente, Quantile etc.) entsprechen damit genau den Charakteristika der Stichprobe.

Die kontinuierliche empirische Verteilung benutzt die ermittelten Daten als Stützstellen zur Konstruktion einer Verteilungsfunktion. Die einfachste Form ist eine lineare Interpolation. Man unterscheidet zwischen einer Interpolation zwischen den gemessenen Daten und einer zusätzlichen Extrapolation am Ende und/oder am Anfang. Im ersten Fall wird zwischen den Werten y_{i-1} und y_i wird eine Gerade mit den Endpunkten $(y_{i-1}, (i-1)/(n-1))$ und $(y_i, i/(n-1))$ gezogen. Im zweiten Fall wird $n-1$ im Nenner durch n oder $n+1$ ersetzt. Ausgehend vom kleinsten und/oder größten Wert kann dann extrapoliert werden, wobei der Endpunkt der Extrapolation zusätzlich festzulegen ist. Wenn man zum Beispiel Zeiten betrachtet, so ist bekannt, dass die generierten Zufallszahlen nur nicht negativ sein können. Damit bietet es sich an, zwischen 0 und y_1 eine weitere Gerade zu ziehen (siehe rechte Seite von Abbildung 2.47). Kontinuierliche empirische Verteilungen weisen in der Regel andere Charakteristika als die Stichprobe auf. Zufallszahlen aus kontinuierlichen empirischen Verteilungen werden in zwei Schritten gezogen. Im ersten Schritt wird eine diskrete Zufallszahl $[1, n-1]$ (bzw. $[1, n]$ oder $[1, n+1]$, falls extrapoliert wurde) gezogen. Diese bestimmt das Intervall. Anschließend wird der Wert durch Ziehen einer gleichverteilten Zufallszahl aus $[y_{i-1}, y_i)$ erzeugt.

Vergleich empirischer und theoretischer Verteilungen

Ob empirische oder theoretische Verteilungen in der Simulation verwendet werden sollen wird in der Literatur kontrovers diskutiert und hängt letztendlich primär von der Problemstellung und Datenlage ab. Die folgenden Argumente sprechen für die Verwendung empirischer Verteilungen bzw. gegen die Verwendung theoretischer Verteilungen:

- Die Anpassung theoretischer Verteilungen ist immer mit Informationsverlust verbunden. Die Stichprobe beinhaltet die gesamte gemessene Information.
- Die Wahl einer Verteilungsfamilie zur Repräsentation der Daten mit einer theoretischen Verteilung ist unklar.
- Die Schätzung der Verteilungsparameter ist in vielen Fällen nicht robust, so dass kleinere Schwankungen in den Daten zu deutlichen Unterschieden bei den Parametern führen.
- Es gibt theoretische Verteilungen, für die keine effizienten Methode zur Generierung von Zufallszahlen existiert.

Für die Verwendung von theoretischen Verteilungen und damit gegen die Verwendung von empirischen Verteilungen spricht:

- Empirische Verteilungen hängen stark von der Stichprobe ab. Unterschiedliche Stichproben eines Systems können zu unterschiedlichen Verteilungen und damit auch zu deutlichen Unterschieden im Modellverhalten führen. Insbesondere Ausreißer in den Messungen können das Verhalten stark beeinflussen.
- Empirische Verteilungen erlauben die Realisierung von Zufallszahlen in der gemessenen Bandbreite, die Extrapolation von Werten ist schwierig und kann zu Verfälschungen führen.

- Die Darstellung theoretischer Verteilungen ist in den meisten Fällen deutlich kompakter. Für empirische Verteilungen müssen vollständige Stichproben gespeichert werden. Kleine Stichprobenumfänge stellen das Verhalten nicht genau genug dar, während große Stichprobenumfänge den Speicherbedarf erhöhen.
- Die Generierung aus theoretischen Verteilungen ist dann effizienter, wenn die empirische Verteilung sehr viele Stützstellen aufweist.
- Es gibt oftmals theoretisch und empirisch fundierte Gründe, für bestimmte Abläufe einen ganz bestimmten Verteilungstyp zu wählen.
- Die Verwendung theoretischer Verteilungen erlaubt unter Umständen die Anwendung alternativer Analyseansätze, während empirische Verteilungen nur in Kombination mit Simulation einsetzbar sind.

In der Praxis werden in großen Modellen meistens theoretische Verteilungen eingesetzt, da die Verwendung empirischer Verteilungen zu aufwändig bzgl. Datenerhebung und Darstellung der Verteilungen im Simulationsprogramm ist.

Anpassung theoretischer Verteilungen

Die Anpassung theoretischer Verteilungen erfolgt in den folgenden drei Schritten, die nachfolgend näher erläutert werden.

1. Bestimmung des Verteilungstyps.
2. Schätzung der Parameter.
3. Bestimmung der Anpassungsgüte.

Zur Wahl des Verteilungstyps kann man folgende Szenarien unterscheiden.

a) Der Verteilungstyp ist aus der Theorie bekannt, die Parameter werden aus einer Stichprobe geschätzt.

b) Der Verteilungstyp und die Parameter werden aus einer Stichprobe geschätzt.

c) Es gibt keine theoretischen Erkenntnisse und auch keine Stichprobe.

In den Fällen a) und b) werden jeweils die Schritte 2. und 3. ausgeführt. Fall b) erfordert zusätzlich Schritt 1. Fall c) ist eher unangenehm, da kaum verwertbare Information vorhanden ist. Dieser Fall soll kurz am Ende der Beschreibung der Anpassung von theoretischen Verteilungen behandelt werden.

Begonnen wird mit der Bestimmung des Verteilungstyps. Im Fall a) lässt sich dieser aus der Theorie ableiten. Die Theorie beruht natürlich auf empirischen Erkenntnissen, die man im Laufe der Zeit gewonnen hat. So können gewisse Abläufe in der Realität mit bestimmten Verteilungen in Verbindung gebracht werden. Einige Beispiele dazu:

- Falls eine Zufallsvariable aus einer größeren Anzahl unabhängiger zufälliger Ereignisse resultiert, so könnte sie normalverteilt sein (siehe zentraler Grenzwertsatz).
- Eine Zufallsvariable, die das Minimum einer größeren Anzahl zufälliger Ereignisse ist, könnte Weibull-verteilt sein.
- Eine Zufallsvariable, die zeitlich aufeinander folgende Ereignisse beschreibt, die einzeln mit konstanter Rate auftreten, könnte exponentiell-verteilt sein.
- Eine Zufallsvariable, die das Produkt einer größeren Anzahl zufälliger Einflüsse ist, könnte log-normal-verteilt sein.

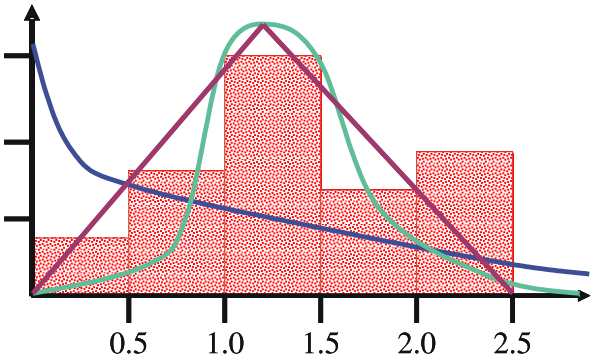

Abbildung 2.48: Histogramm und Dichtefunktionen zur Bewertung der Anpassung.

- Die Ausfallzeiten technischer Systeme können oft durch Mischungen aus Weibull-Verteilungen repräsentiert werden.
- Die Zugriffsdauern auf Web-Server können oftmals durch Pareto-Verteilungen repräsentiert werden.
- ...

Theoretische Erkenntnisse erlauben die Identifikation von Verteilungstypen, sollten aber in jedem konkreten Fall hinterfragt und getestet werden.

Im Fall b) muss der Verteilungstyp aus der Stichprobe geschätzt werden. Auch hier ist Vorabinformation über die Struktur und Herkunft der Daten oft sinnvoll, um die Klasse der möglichen Verteilungen einzugrenzen. Weitere Eingrenzungen können dadurch erfolgen, dass Verteilungscharakteristika aus der Stichprobe geschätzt werden (mehr zu Schätzungen später). So lassen sich Mittelwert und Varianz das Verhältnis von Median zu Erwartungswert oder höhere Momente aus der Stichprobe schätzen und für theoretischer Verteilungen berechnen. Durch Vergleich der Werte aus der Stichprobe mit den für eine bestimmte Verteilung erreichbaren Werte, lassen sich einzelne Verteilungen ausschließen. So ist zum Beispiel der Variationskoeffizient $VK(Y) = \sigma(Y)/E(Y)$ einer exponential-verteilten Zufallsvariablen Y immer 1. Falls die Schätzung aus der Stichprobe einen von 1 verschiedenen Wert ergibt, so kann die Exponentialverteilung zur Repräsentation der Daten ausgeschlossen werden. Auf diese Weise lässt sich die Menge der möglichen Verteilungen in der Regel reduzieren, es ist aber meistens keine Festlegung auf einen Verteilungstyp möglich.

Dichtefunktionen lassen sich am besten durch visuellen Vergleich der Histogrammdarstellung der Stichprobe mit dem Verlauf der Dichtefunktion vergleichen. In einem ersten Schritt wird auf Grund von Erfahrung und Wissen dem Histogramm ein Verteilungstyp zugeordnet. Der konkrete Vergleich zwischen Histogramm und Dichtefunktion ist allerdings erst nach der Parameterschätzung möglich. Ausgehend von der Wahl eines Verteilungstyps werden die Parameter geschätzt und dadurch der "beste" Repräsentant aus der Familie von Verteilungen ermittelt. Die zugehörige Dichtefunktion kann anschließend in Kombination mit dem Histogramm dargestellt werden (siehe Abbildung 2.48). Heute wird der Anpassungsprozess durch Software unterstützt. Die Software führt natürlich keinen visuellen Vergleich durch, sondern schätzt die Parameter für alle Verteilungen aus einer Liste von Verteilungen und berechnet anschließend Maßzahlen, die die Qualität der Anpassung widerspiegeln (siehe auch Bewertung der Anpassung in Schritt 3.). Die Verteilung mit der besten Anpassung wird anschließend ausgewählt.

Auch wenn das Vorgehen automatisierbar ist, so ist das Ergebnis keineswegs eindeutig, sondern von der Histogrammdarstellung abhängig. Diese beinhaltet aber eine Reihe von Freiheitsgraden, durch die die Histogrammdarstellung und damit auch die Verteilungsanpassung beeinflusst wird. Die beiden folgenden Entscheidungen müssen getroffen werden.

- Die Anzahl und Breite der Zellen ist festzulegen.

  - I.d.R. sollten Zellen gleicher Breite gewählt werden, ansonsten muss die Höhe angepasst werden.
  - Die Zahl der Zellen ist so zu wählen, dass
    * die Form der Dichte erkennbar wird und
    * jede Zelle genügend Werte (≥ 10) enthält. Bei weniger Werten ändert sich das Aussehen des Histogramms zu schnell, wenn die Stichprobe nur leicht variiert.

- Der überdeckte Bereich ist zu definieren. Dies bedeutet, dass
  - der Start der ersten Zelle und das Ende der letzten Zelle festgelegt werden müssen und
  - entschieden werden muss, welche Werte als Ausreißer vernachlässigt werden.

Es ist anzuraten, die Anpassung für mehrere Histogrammparameter durchzuführen, um bewerten zu können, inwieweit die erzeugte Verteilung von den Histogrammparametern abhängt.

Schätzung der Verteilungsparameter

Der nun folgende Schritt ist die Parameterschätzung. Jede Verteilungsfamilie weist Parameter auf. Bei der Exponentialverteilung ist dies die Rate λ, bei der Normalverteilung der Mittelwert μ und die Standardabweichung σ, bei der Dreiecksverteilung, die linke Grenze a, die rechte Grenze b und der Modalwert c. Ziel der Parameterschätzung ist es, die Parameter so zu wählen, dass die Stichprobe durch die Verteilung möglichst gut repräsentiert wird. Zur formalen Darstellung sei $\Theta = (\Theta_1, \ldots, \Theta_p)$ der Parametervektor und $\mathbf{x} = (\mathbf{x_1}, \ldots, \mathbf{x_n})$ die Stichprobe. Unter Verwendung dieser beiden Vektoren sei $f(\mathbf{x}, \Theta)$ eine allgemeine Darstellung der Dichtefunktion mit Parametervektor Θ für Stichprobe $\mathbf{x}$. Θ ist so zu wählen, dass Verteilung und Stichprobe möglichst gut korrespondieren. Es gibt mehrere Methoden zur *Parameterschätzung* aus der Stichprobe. Wir betrachten hier die *Momentenmethode* und die *Maximum-Likelihood-Methode*.

Momentenmethode:

Bei der Momentenmethode werden die Parameter der Verteilung als Funktion der Momente dargestellt. Dies ist für die meisten Verteilungen möglich. Sei $(y_1, \ldots, y_n)$ dir konkrete Stichprobe, jeder Wert y_i ist die Realisierung einer Zufallsvariablen Y_i, alle Y_i seien unabhängig und identisch verteilt. Wir definieren $\tilde{Y}^i$ als einen Schätzer für $E(Y^i)$ und $\hat{Y}^i$ als den konkreten Schätzwert. Der folgende Schätzer ist erwartungstreu

$$\tilde{y}^i = \frac{1}{n}\sum_{j=1}^{n} (Y_j)^i \quad \text{und} \quad \hat{Y}^i = \frac{1}{n}\sum_{j=1}^{n} (y_j)^i$$

ist der resultierende Schätzwert. $\tilde{S}^2$ und $\hat{S}^2$ bezeichnen den Schätzer und Schätzwert für die Varianz. Ein erwartungstreuer Schätzer für die Varianz lautet

$$\tilde{S}^2 = \frac{1}{n-1}\sum_{j=1}^{n} \left(Y_j - \tilde{Y}^1\right)^2$$

Es soll nun eine Darstellung $\Theta_j = \phi_j(E(X^1), \ldots, E(X^p))$ gefunden werden, so dass Parameter Θ_j als Funktion $\phi_j(.)$ der ersten p Momente dargestellt wird. Ein Momentschätzer $\tilde{\Theta}_j$ entsteht dadurch, dass die Momente durch die Schätzer für die Momente ersetzt werden, also $\tilde{\Theta}_j = \phi_j(\tilde{Y}^1, \ldots, \tilde{Y}^p)$. Wir betrachten als einfache Beispiele die Exponential- und die Normalverteilung. Für den Parameter λ der Exponentialverteilung gilt $\lambda = (E(Y))^{-1}$. Damit ergibt sich der Schätzer $\tilde{\lambda} = 1/\tilde{Y}^1 = n/\left(\sum_{j=1}^{n} Y_j\right)$. Für die Normalverteilung gilt $\mu = E(Y^1)$ und $\sigma = \sqrt{(E(Y^1))^2 - E(Y^2)}$, mit dem resultierenden Schätzer $\mu = \tilde{Y}^1$ und $\sigma = \tilde{S}$.

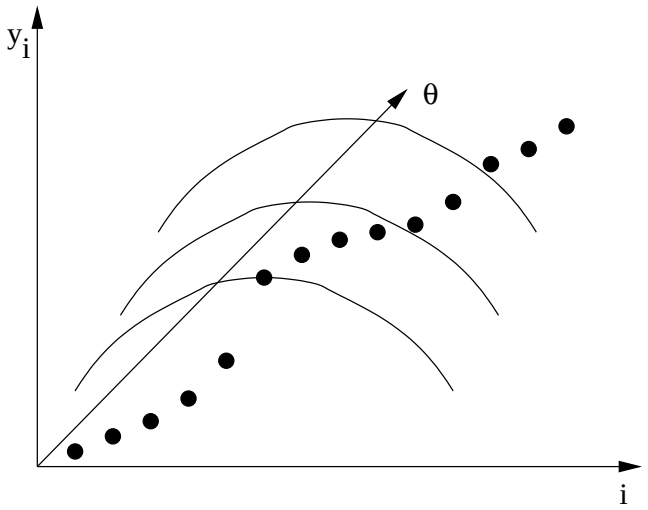

Abbildung 2.49: `Prinzip der ML-Methode.`

Die aus der Momentenmethode resultierenden Schätzer sind meistens asymptotisch erwartungstreu und konsistent. Trotzdem sind es oft keine guten Schätzer, da nur die Momente, nicht aber die Form der Verteilungs-/Dichtefunktion berücksichtigt werden. Deshalb sind, wenn vorhanden, die im Folgenden vorgestellten maximum-likelihood-Schätzer in der Regel vorzuziehen.

Maximum-likelihood-Schätzer

Die Basis der maximum-likelihood-Schätzung ist die Suche nach der plausibelsten Lösung. Dies soll zuerst an einem einfachen Beispiel erläutert werden. Wir betrachten dazu eine diskrete Verteilung mit Parameter θ, so dass $p_\theta(x)$ der Wahrscheinlichkeit von x bei Parameter θ entspricht. Wir nehmen nun an, dass die Stichprobe $(y_1, \ldots, y_n)$ festliegt und der Parameter θ variabel ist. Die Likelihoodfunktion ist dann definiert als

$$L(\theta) = p_\theta(y_1) \cdot \ldots \cdot p(y_n)$$

Ziele der maximum-likelihood (ML)-Methode ist es, Parameter θ so zu wählen, dass $L(\theta)$ maximal wird. Also $\max_\theta(L(\theta))$. Der Punkt der maximalen Beobachtungswahrscheinlichkeit wird gesucht. Das Vorgehen kann auch für den Fall von Parametervektoren Θ benutzt werden. In diesem Fall ist ein mehrdimensionales Optimierungsproblem zu lösen.

Das Prinzip der ML-Methode für den kontinuierlichen Fall wird in Abbildung 2.49 dargestellt. Durch Variation von θ verschiebt sich die Dichtefunktion, so dass der Wert der Likelihoodfunktion sich ändert. Im kontinuierlichen Fall ist der Ansatz allerdings weniger intuitiv, da die Wahrscheinlichkeit in jedem Punkt 0 sein kann. Deshalb wird der Wert der Dichtefunktion $f_\theta(x)$ benutzt. Die Likelihoodfunktion lautet dann

$$L(\theta) = \prod_{j=1}^{n} f_\theta(y_j)$$

Gesucht wird $\theta_{\max}$ mit $L(\theta_{\max)} \geq L(\theta)$ für alle θ.

Das Vorgehen soll kurz am Beispiel der Exponentialverteilung erläutert werden. Es gilt

$$L(\lambda) = \prod_{j=1}^{n} \lambda \cdot e^{-\lambda \cdot y_j} = \lambda^n \cdot e^{-\lambda \cdot \sum_{j=1}^{n} y_j}$$

Das Maximum dieses Produktes ist aufwändig zu ermitteln, deshalb macht man sich zu Nutze, dass eine Funktion und deren Logarithmus, sofern er definiert ist, die Extremwerte an den selben

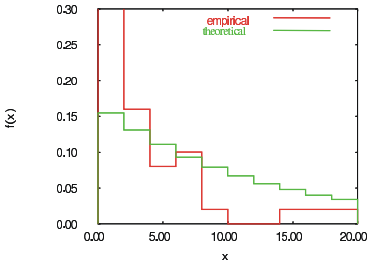

Abbildung 2.50: Histogramm der empirischen Verteilung und der theoretischen Verteilung.

Stellen haben. Da die Likelihoodfunktion nicht negativ ist, ist ihre Logarithmus definiert, wenn der Wert größer 0 ist. Im speziellen Fall der Exponentialverteilung ist der Wert der Likelihoodfunktion größer 0 für $0 < \lambda < \infty$. Da dies der Bereich der relevanten Parameterwerte ist, gilt

$$l(\lambda) = \ln(L(\lambda)) = n \cdot \ln \lambda - \lambda \cdot \sum_{j=1}^{n} y_j$$

Die Ableitung von $l(\lambda)$ lautet $l'(\lambda) = n/\lambda - \sum_{j=1}^{n} y_j$. Die Nullstelle der Ableitung ist gerade der ML-Schätzer für λ. Also $\hat{\lambda} = n/\left(\sum_{j=1}^{n} y_j\right)$. In diesem Fall stimmen Momenten- und ML-Schätzer überein. Dies ist natürlich nicht immer so. Für die Normalverteilung liefert die ML-Methode die Schätzer $\tilde{\mu} = \tilde{Y}^1$ und $\tilde{\sigma} = \sqrt{(n-1)/n \cdot \tilde{S}^2}$ im Gegensatz zu $\tilde{\sigma} = \tilde{S}$ als Momentenschätzer.

Die ML-Methode kann auch für Parametervektoren Θ angewendet werden. Allgemein wird dann

$$l(\Theta) = \ln(L(\Theta)) = \sum_{j=1}^{n} \ln\left(f_\Theta(y_j)\right)$$

bzgl. des Parametervektors Θ maximiert. Oft hat das resultierende Optimierungsproblem keine geschlossene Lösung und muss deshalb mit einem Verfahren zur nichtlinearen Optimierung (siehe auch Kapitel 7) gelöst werden. Für viele bekannte Verteilungen sind ML-Schätzer für die Parameter bekannt (siehe [11]).

ML-Schätzer haben in der Regel die folgenden Eigenschaften:

- Sie sind asymptotisch erwartungstreu.
- Sie sind konsistent.
- Sie sind asymptotisch normalverteilt. Dies bedeutet, dass mit den Methoden aus Abschnitt 2.5 Konfidenzintervalle für die Parameter bestimmbar sind und damit auch quantifiziert werden kann, wie robust die Schätzung war.
- Sie sind in der Regel nicht schlechter bzw. besser als die Momentenschätzer.

Aus den genannten Gründen werden in den meisten Fällen ML-Schätzer verwendet, falls solche bekannt sind oder ermittelt werden können.

Bestimmung der Anpassungsgüte

Nachdem eine theoretische Verteilung angepasst wurde, ist die Anpassung zu bewerten. Dieser Schritt kann auch verwendet werden, um unterschiedliche Verteilungsanpassungen zu vergleichen. Man unterscheidet zwischen heuristischen Prozeduren und statistischen Testverfahren.

Heuristische Prozeduren vergleichen die Struktur der theoretischen und der empirischen Verteilungs- oder Dichtefunktion. Der Vergleich der Dichtefunktionen erfolgt wieder auf Basis von Histogrammen. Sei dazu k die Anzahl der Intervalle und Δ_i das $i$te Intervall. Wenn n_i die Anzahl der Werte

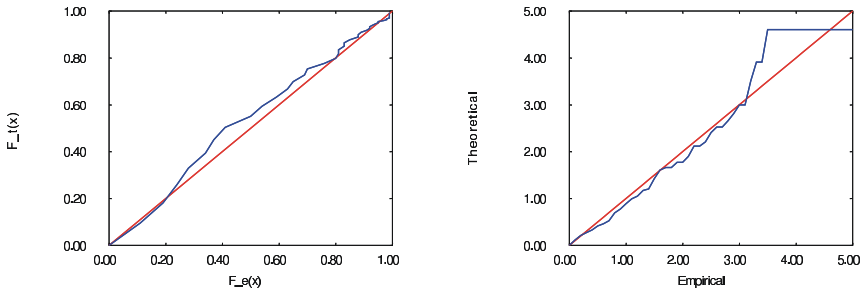

Abbildung 2.51: P-P-Plot und Q-Q-Plot.

der Stichprobe im $i$ten Intervall ist, so ist $h_i = n_i/n$ die relative Häufigkeit der Stichprobenwerte im Intervall Δ_i. Die Wahrscheinlichkeit einen Wert der theoretischen Verteilung im Intervall Δ_i zu finden lautet $p_i = \int_{x \in \Delta_i} f_t(x)dx$, wobei $f_t(x)$ die Dichtefunktion der theoretischen Verteilung ist. Die Abstandmaße $D = \sum_{i=1}^{k} |h_i - p_i|$ und $D' = \sum_{i=1}^{k} (h_i - p_i)^2$ beschreiben die Passgüte zwischen empirischer und theoretischer Verteilung. Der Wert der Abstandsmaße hängt von der Lage und Größe der Histogrammzellen ab. Es ist auch nicht a priori klar, welcher Wert von D bzw. D' eine gute oder zumindest ausreichende Anpassung beschreibt. Auf Grund des Vergleichs der Werte von D kann aber die Anpassungsgüte unterschiedlicher Verteilungen verglichen werden. Über die Qualität der Anpassung sagt oftmals ein visueller Vergleich der Histogramme, wie in Abbildung 2.50 gezeigt, mehr aus als die Maßzahl.

Neben den Dichtefunktionen können auch die Verteilungsfunktionen verglichen werden. Auf Grund der Struktur von Verteilungsfunktionen ist der rein visuelle Vergleich von empirischer und theoretischer Verteilungsfunktion nicht sehr hilfreich. Stattdessen haben sich $Q-Q$-Plots und $P-P$-Plots als alternative Darstellungsformen eingebürgert. Sei $F_t(x)$ die theoretische Verteilungsfunktion und $F_e(x)$ die empirische Verteilungsfunktion. Für die Werte der empirischen Verteilungsfunktion gilt $F_e(x) = \max_{y_i \leq x}(i/n)$. Beim $Q-Q$-Plot werden die Quantile der beiden Verteilungsfunktionen zueinander in Beziehung gesetzt. Sei q_j $(0 < q_j < 1)$ das $j$te dargestellte Quantil, dann wird auf der x-Achse $\max_{y_i \leq q_j}(i/n)$ und auf der y-Achse $F_t^{-1}(q_j)$ abgetragen (siehe die rechte Graphik in Abbildung). Wenn F_t und F_e vollständig korrespondieren und die Kurve für alle Werte von q dargestellt wird, so entsteht eine Strecke durch die Punkte $(0,0)$ und $(1,1)$. Abweichungen zeigen die Unterschiede zwischen beiden Verteilungsfunktionen. Beim $P-P$-Plot werden die Wahrscheinlichkeiten miteinander in Beziehung gesetzt. Für x wird $F_e(x)$ auf der x-Achse abgetragen und $F_t(x)$ auf der y-Achse. Bei vollständiger Korrespondenz entsteht wieder eine Strecke mit Steigung 1, Abweichungen davon zeigen Abweichungen der Verteilungsfunktionen an. Während $Q-Q$-Plots Unterschiede im hinteren Teil der Verteilungsfunktion (im tail) betonen, zeigen $P-P$-Plots mehr die Unterschiede in der Mitte.

Anpassungstests

Die bisher vorgestellten Vergleichsverfahren geben kein Maß an, um zu entscheiden, wann die theoretische Verteilung die empirischen Daten genau genug modelliert. Auf Grund stochastischer Schwankungen ist zu erwarten, dass zwangsläufig Unterschiede zwischen Stichprobe und theoretischer Verteilung existieren müssen. Eine Möglichkeit Hinweise zu bekommen, wann eine Modellierung ausreichend genau ist, sind Anpassungstests. Wie bei allgemeinen Testverfahren (siehe auch Seite 73ff) wird eine Hypothese H_0 aufgestellt:
H_0: Die Stichprobe $(y_1, \ldots, y_n)$ wurde aus der Verteilung $F_t(x)$ gezogen

Ein Anpassungstest liefert die Antwort, ob H_0 verworfen oder angenommen werden soll. Wie bei allen statistischen Testverfahren ist das Ergebnis mit Fehlern behaftet. Wie oft üblich, wird der Fehler der ersten Art vorgegeben. D.h. die Hypothese wird mit Wahrscheinlichkeit α abgelehnt, obwohl die Stichprobe aus der Verteilung gezogen wurde. Über den Fehler der zweiten Art werden in der Regel keine Angaben gemacht.

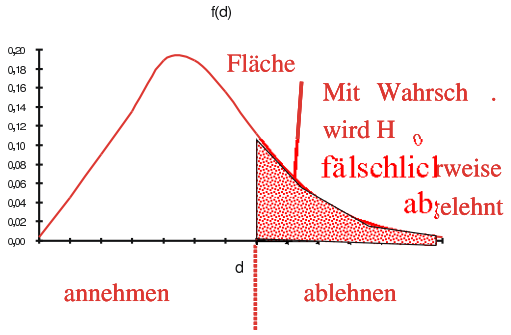

Abbildung 2.52: Skizze des Vorgehens beim Chi-Quadrat-Test.

Neben diesen grundsätzlichen Grenzen der Testverfahren beobachtet man bei Anpassungstests noch ein zweites Phänomen. So wird für kleine Stichprobenumfänge H_0 meistens angenommen, da zu wenig Information für eine Ablehnung vorhanden ist. Für große Stichprobenumfänge wird dagegen H_0 in fast allen Fällen abgelehnt, da zwangsläufig Unterschiede zwischen empirischen Daten und theoretischer Verteilung existieren. Man sollte sich diesen Schwächen von Testverfahren bewusst sein. Trotzdem bieten Testverfahren in Kombination mit den anderen vorgestellten Verfahren zum Vergleich von empirischer und theoretischer Verteilung ein mächtiges Instrumentarium zur Modellierung von Daten mittels theoretischer Verteilungen. Es sollen im Folgenden kurz die zwei bekanntesten Anpassungtests vorgestellt werden.

Chi-Quadrat-Test

Der Chi-Quadrat-Test ist ein sehr altes Testverfahren, welches schon um 1900 entwickelt wurde. Er basiert auf dem formalen Vergleich der Histogramme der empirischen und theoretischen Verteilung. Seien $b_0 < b_1 < \ldots < b_k$ die Intervallgrenzen, so dass $\Delta_i = [b_{i-1}, b_i)$. Wie bereits weiter oben definiert ist $n_i = |\{y_j | y_j \in \Delta_i\}|$ die Anzahl der Element im Intervall Δ_i und $p_i = \int_{y \in \Delta_i} f_t(x)dx$ die Wahrscheinlichkeit der theoretischen Verteilung einen Wert im Intervall Δ_i zu generieren. Als Abstandsmaß zwischen empirischer und theoretischer Verteilung wird $\sum_{j=1}^{k} (n_j - p_j \cdot n)^2 / (n \cdot p_j)$ verwendet. Wie bereits erläutert, steigt die Wahrscheinlichkeit, dass die Stichprobe aus der theoretischen Verteilung generiert wurde, wenn der Wert der Summe kleiner wird. Damit ein Testverfahren definiert werden kann, muss allerdings eine Teststatistik vorhanden sein, d.h. es muss bekannt sein, welche Verteilung der zu testende Wert hat, wenn H_0 gilt.

Bevor eine solche Teststatistik definiert wird, wird kurz die χ^2-Verteilung eingeführt. Seien dazu $Y_1, \ldots, Y_k$ unabhängige $N(0,1)$ verteilte Zufallsvariablen, dann ist $Y = \sum_{i=1}^{k} Y_i$ χ^2-verteilt mit k Freiheitsgraden. Die χ^2-Verteilung ist eigentlich eine Familie von Verteilungen (jeder Freiheitsgrad definiert eine Verteilung), die keine explizite funktionale Form hat. Die kritischen Werte liegen aber in vertafelt Form vor und sind in den meisten Statistikbüchern zu finden. $\chi^2_{k,1-\alpha}$ bezeichnet den Wert, der mit Wahrscheinlichkeit α von χ^2-Verteilung mit k Freiheitsgrade überschritten wird (siehe auch Abbildung 2.52). Als Teststatistik des Chi-Quadrat-Tests wird

$$d = \sum_{j=1}^{k} \frac{(n_j - n \cdot p_j)^2}{n \cdot p_j}$$

verwendet. d ist die Realisierung einer Zufallsvariablen D. Falls H_0 gilt, dann

- ist D asymptotisch χ^2-verteilt (d.h. für große n ist D approximativ χ^2-verteilt), wobei
  - falls keine Verteilungsparameter aus der Stichprobe geschätzt wurden, die χ^2-Verteilung k Freiheitsgrade hat,
  - während bei der Schätzung von p Freiheitsgraden aus der Stichprobe, die Zahl der Freiheitsgrade sich auf $k - p - 1$ reduziert.

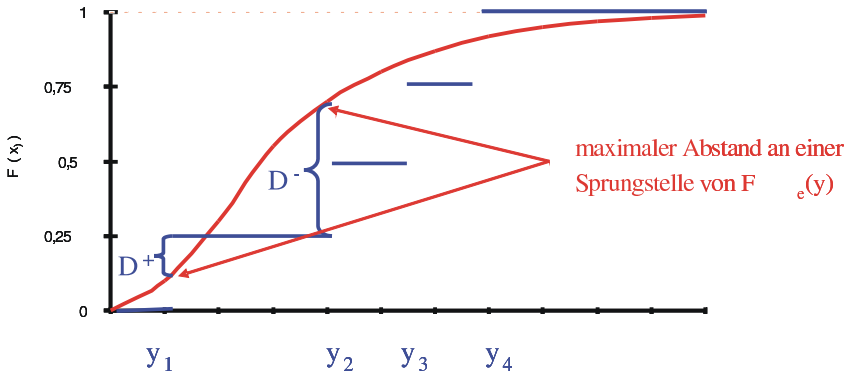

Abbildung 2.53: Skizze zum Vorgehen beim Kolmogorov-Smirnov-Test.

Damit ergibt sich folgendes Vorgehen:

- Festlegung der Intervalle,
- Berechnung von d,
- Vergleich des Wertes von d mit dem kritischen Wert zum gewählten Signifikanzniveau,
  - falls d kleiner als der kritische Wert ist, Hypothese annehmen,
  - ansonsten Hypothese ablehnen.

Der Test beinhaltet als subjektive Komponente die Lage und Breite der Intervalle. Bei kontinuierlichen Verteilungen wird empfohlen, die Intervalle so zu wählen, dass p_j für alle Intervalle identisch ist. Dies bedeutet, dass die Intervallbreiten variieren. Weiterhin sollte p_j so gewählt werden, dass $n \cdot p_j \geq 5$. Bei diskreten Verteilungen sollten sich die Intervalle nach der Lage der diskreten Werte richten. Für große Datenmengen wird als Intervallzahl ein Wert zwischen $\sqrt{n}$ und $n/5$ vorgeschlagen.

Kolmogorov-Smirnov-Test

Der Kolmogorov-Smirnov-Test vergleicht die theoretischer Verteilungfunktion $F_t(x)$ mit der empirischen Verteilungsfunktion $F_e(x)$. $F_e(x) = \max_{y_i \leq x}(i/n)$ ist eine Treppenfunktion, als Teststatistik wird der maximale Abstand zwischen $F_t(x)$ und $F_e(x)$ verwendet. Also

$$D_n = \max_x(|F_e(x) - F_t(x)|)$$

Falls nötig, kann das Maximum durch das Supremum ersetzt werden. Da $F_t(x)$ eine monoton nicht fallende Funktion ist und $F_e(x)$ eine Treppenfunktion ist, kann man sich leicht überlegen, dass der maximale Abstand zwischen beiden Funktionen an einer Sprungstelle von $F_e(x)$ angenommen werden muss (siehe auch Abbildung 2.53). Der maximale Abstand kann deshalb wie folgt definiert und ermittelt werden

$$D_n^+ = \max_{1 \leq i \leq n}(i/n - F_t(y_i)) \quad D_n^- = \max_{1 \leq i \leq n}(F_t(y_i) - (i-1)/n\} \quad D_n = \max(D_n^-, D_n^+)$$

Ein großer Wert von D_n deutet auf eine schlechte Anpassung hin. Zur Anwendung des Tests müssen zwei Fälle unterschieden werden:

- Falls keine Verteilungsparameter aus der Stichprobe geschätzt wurden, so ist H_0 zu verwerfen, falls $\left(\sqrt{n} + 0.12 + \frac{0.11}{\sqrt{n}}\right) \cdot D_n \geq c_{1-\alpha}$. Die Werte von $c_{1-\alpha}$ sind vertafelt und in den meisten Statistikbüchern zu finden.

- Falls die Verteilungsparameter aus der Stichprobe geschätzt wurden, so ist die Teststatistik verfälscht und die vertafelten Werte sind zu optimistisch (d.h. H_0 würde zu oft angenommen). In diesem Fall existieren Teststatistiken nur für spezielle Verteilungen, wie die Normalverteilung, Exponentialverteilung oder Weibull-Verteilung.

Chi-Quadrat- und Kolmogorov-Smirnov-Test können beide auch für das Testen von Zufallszahlen verwendet werden.

Datenmodellierung ohne Stichprobe und theoretische Information

Dies ist der Fall c, der bisher ausgespart wurde. In der Praxis tritt dieser Fall aber leider häufiger bei der Planung von Systemen auf. Eine genaue Modellierung ist in einem solchen Fall natürlich nicht möglich, trotzdem können in der Regel zumindest eingeschränkte Informationen über die Daten gewonnen werden. So kann oftmal zwischen diskreten und kontinuierlichen Werten unterschieden werden und es können minimale und maximale Werte festgelegt werden. Manchmal kann auch ein Mittelwert geschätzt werden. In diesen Fällen, können die Daten dann durch eine Gleicherverteilung, Dreiecksverteilung oder Beta-Verteilung im Modell repräsentiert werden.

Korrelierte Daten und stochastische Prozesse

Bisher wurde immer von unabhängig, identisch verteilten Daten ausgegangen. Die meisten Methoden der Statistik und auch die vorhandene Software zur Datenmodellierung (z.B. Arena Input Analyzer, ExpertFit, etc.) ist auch auf diesen Fall beschränkt. In der Realität zeigt sich aber oftmals, dass die Annahmen nicht gelten. So sind Daten

- korreliert bzgl. der Zeitintervalle
  - Ankunftsprozesse in einem Rechnernetz, die Korrelation über mehrere Zeitskalen aufweisen,
  - Fehler in einem technischen System, die durch gegenseitige Beeinflussung von Komponenten entstehen,
- korreliert bzgl. verschiedener Messgrößen
  - Größe und Gewicht von Menschen,
  - CPU-Zeitbedarf und Speicherplatzbedarf von Jobs in einem Rechner,
- über einen längeren Zeitraum nicht identisch verteilt sind
  - Ankunftsprozess in einem Restaurant,
  - Zugriffe auf einen Web-Server.

Während der dritte Fall, nämlich zeitlich variierende Prozesse, oft dadurch abgebildet werden kann, dass für unterschiedliche Zeiträume unterschiedliche Verteilungen angepasst werden, muss die Korrelation von Daten direkt bei der Modellierung berücksichtigt werden und hat damit Einfluss auf die stochastischen Modelle. Es reicht nicht mehr aus, Verteilungen anzupassen, sondern es müssen komplexere Modelle, wie Zufallsvektoren, Markovsche Ankunftsprozesse, nichtstationäre Poisson-Prozesse, autoregressive Modelle oder multivariate Normalverteilungen verwendet werden. Während die Generierung von Zufallszahlen aus diesen Verteilungen in der Regel mehr oder weniger problemlos realisiert werden kann, ist die Anpassung der Modelle auf Basis von Stichproben ein komplexes und erst in Ansätzen gelöstes Problem.

An dieser Stelle soll als ein einfaches Beispiel die Schätzung der Parameter einer bivariaten Normalverteilung vorgestellt werden (siehe Seite 81). Seien $(X_{11}, \ldots, X_{1n})$ und $(X_{21}, \ldots, X_{2n})$

die Stichproben der beiden Zufallsvariablen X_1 und X_2. Wir nehmen an, dass aus den beiden Stichproben ableitbar ist, dass die beiden Zufallsvariablen normalverteilt sind. Ferner seinen

$$\tilde{X}_j = \frac{1}{n} \cdot \sum_{i=1}^{n} X_{ji} \quad \text{and} \quad \tilde{S}_j^2 = \frac{1}{n-1} \cdot \sum_{j=1}^{n} \left(X_{ji} - \tilde{X}_j \right)^2$$

die Schätzer für den Erwartungswert und die Varianz ($j = 1, 2$). Der Schätzer für die Kovarianz lautet

$$\widetilde{C}(X_1, X_2) = \frac{1}{n-1} \cdot \sum_{i=1}^{n} \left(\left(X_{1i} - \tilde{X}_1 \right) \cdot \left(X_{2i} - \tilde{X}_2 \right) \right) = \frac{1}{n-1} \cdot \left(\sum_{i=1}^{n} X_{1i} \cdot X_{2i} - n \cdot \tilde{X}_1 \cdot \tilde{X}_2 \right)$$

und

$$\tilde{\rho} = \frac{\widetilde{C}(X_1, X_2)}{\tilde{S}_1 \cdot \tilde{S}_2}$$

ist der Schätzer für den Korrelationskoeffizienten. Mit diesen Parametern ist die bivariate Normalverteilung vollständig spezifiziert und es können Realisierungen mit den schon vorgestellten Ansätzen zur Generierung erzeugt werden.

Im Prinzip können ähnliche Verfahren auch zur Schätzung von Korrelationen höherer Ordnung eingesetzt werden. Allerdings sind die dabei auftretenden praktischen Probleme nicht zu vernachlässigen, da die Schätzer oft wenig robust und selbst wieder korreliert sind.

2.5 Auswertung von Simulationsläufen

An dieser Stelle sei noch einmal an das Ziel einer Simulationsstudie erinnert. Es sollen Aussagen über ein System S gewonnen werden, indem ein Modell analysiert wird. Wie herausgearbeitet wurde, soll der Einfluss der kontrollierbaren Eingabegrößen C und teilweise auch der unkontrollierbaren Eingabegrößen U auf die Ausgabegrößen P ermittelt werden. Wir haben gesehen, dass wir Zufallseinflüsse durch deterministische Algorithmen in das Modell integrieren können und die Realisierung des Zufalls damit nur von der Saat des Zufallszahlengenerators abhängt. Wenn wir die Saat des Zufallszahlengenerators als Teil von U betrachten, so soll der Einfluss der Größen C über alle möglichen Saaten des Zufallszahlengenerators ermittelt werden.

Zuerst einmal soll das Vorgehen etwas eingeschränkt werden, indem wir annehmen, dass der Wert eines Leistungsmaßes Y ermittelt werden soll. Ohne dass an dieser Stelle schon festgelegt wird, was Y genau ist bzw. wie es beobachtet wird, können wir davon ausgehen, dass in stochastischen Modellen Y schwankende Verläufe zeigt, d.h. mehrfaches Beobachten wird zu unterschiedlichen Resultaten führen. In der Simulation bedeutet dies, dass durch unterschiedliche Saaten des Zufallszahlengenerators variierende Resultate entstehen. Auch in der Realität können viele Beispiele für schwankende Verläufe von dynamischen Abläufen beobachtet werden. So ist das Verkehrsaufkommen an einer Straßenkreuzung nicht deterministisch, die Ausfallzeiten von technischen Komponenten variieren, das Wachstum von Pflanzen ist auch bei identischen Umgebungsbedingungen für eine Menge von Pflanzen nicht gleich. Um also Aussagen über Y zu treffen, müssen im Prinzip Aussagen über alle möglichen Abläufe gewichtet mit der Wahrscheinlichkeit eines Ablaufs getroffen werden. Aus praktischer Sicht ist diese Anforderung in zweifacher Hinsicht problematisch. Die Zahl der möglichen Abläufe ist oft unendlich und die Wahrscheinlichkeit des Auftretens eines bestimmten Ablauf ist in der Regel nicht bestimmbar. Die nahe liegende Möglichkeit, alle möglichen Abläufe zu beobachten und die resultierenden Beobachtungen von Y gewichtet aufzusummieren entfällt damit und es müssen Aussagen über alle Abläufe auf der Basis einer endlichen Beobachtung getroffen werden. Solche Aussagen können nur mit gewisser, noch näher zu spezifizierender Wahrscheinlichkeit korrekt sein.

Zentral für die Auswertung von Simulationsläufen ist der Begriff des stochastischen Prozesses.

Definition 5 *Ein stochastischer Prozess $Y(t)$ ist eine Funktion über dem Parameterraum T, deren Resultate Zufallsvariablen sind.*

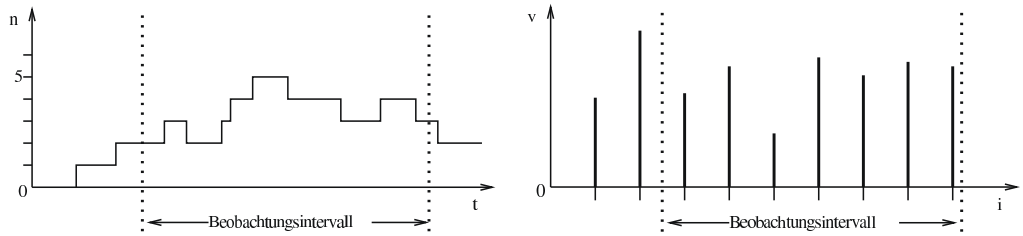

Abbildung 2.54: Typische Beobachtungsszenarien von Simulationsläufen.

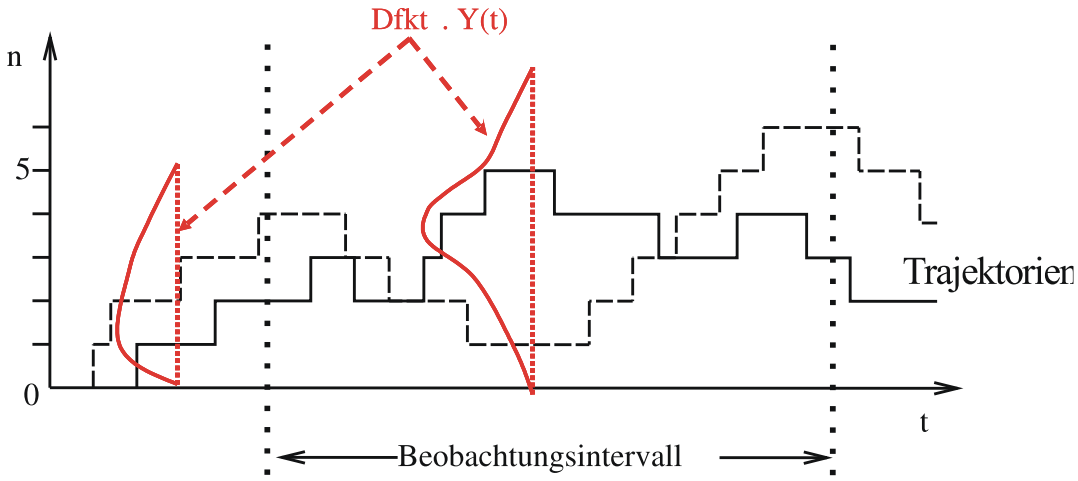

Abbildung 2.55: Beobachtung unterschiedlicher Trajektorien.

Der Parameterraum T wird in der Regel als Zeit interpretiert, wobei oft zwischen zeitdiskreten Prozessen ($T = \mathbb{N}$) und zeitkontinuierlichen Prozessen ($T = \mathbb{R}_{\geq 0}$) unterschieden wird. Für einen stochastischen Prozess kann man für jedes $t \in T$ die Momente, Dichtefunktion und Verteilungsfunktion der Zufallsvariablen $Y(t)$ definieren. Eine Realisierung von $Y(t)$ über T bezeichnet man als eine Trajektorie. Einige Beispiele für stochastische Prozesse wären:

- Das Würfeln mit zwei unterscheidbaren Würfeln, so dass $Y(t)$= Y(t-1) + Wert des 1. Würfels - Wert des 2. Würfels, mit $Y(0) = 0$. Der Parameterraum ist diskret und $Y(t)$ kann diskrete Werte im Intervall $[-5t, 5t]$ annehmen.
- Die Kundenzahl an einem Bankschalter, so dass sich der Wert von $Y(t)$ zu den Ankunfts- und Abgangszeiten von Kunden ändert. In diesem Fall ist der Parameterraum kontinuierlich. Wenn man davon ausgeht, dass in jedem Intervall potenziell eine beliebige Kundenzahl ankommen kann und der Schalterraum unendlich viele Kunden aufnehmen kann, so kann $Y(t)$ Werte aus $\mathbb{N}$ annehmen.

Typische Beobachtungsszenarien der Simulation haben wir bereits vorher kennen gelernt. Abbildung 2.2 zeigt die zwei generellen Beobachtungsszenarien, nämlich die Beobachtung eines Zustands über die Zeit, bei der der Zustand sich zu Ereigniszeitpunkten sprunghaft ändert und die Beobachtung des Schicksals diskreter, temporärer Einheiten der Simulation, bei der jede Einheit einen Wert annimmt.

Grundlegendes Beobachtungsszenarium

Die Darstellung in Abbildung 2.54 gibt jeweils einen Ablauf wieder, bei stochastischen Simulationen verläuft aber jede Trajektorie etwas anders, so dass eher eine Beobachtungsszenarium wie in Abbildung 2.55 gezeigt vorliegt. Bei wiederholten Beobachtungen werden unterschiedliche Abläufe

beobachtet. Zu jedem Zeitpunkt $t \in T$ nimmt $Y(t)$ einen Wert gemäß der Dichtefunktion der zugehörigen Zufallsvariablen ein. Der Verlauf der Dichtefunktionen ist in der Abbildung quer zum Ablauf der Trajektorien dargestellt. Wie wir später sehen werden, reicht selbst die Kenntnis der Dichtefunktionen allein nicht aus, um Trajektorien zu charakterisieren, da in den meisten Fällen die Werte einer Trajektorie korreliert sind. Z.B. die Kundenzahl an einem Schalter zum Zeitpunkt t hängt stark von der Kundenzahl zum Zeitpunkt $t - \Delta$ ab, wenn Δ relativ klein ist. Die daraus resultierende Komplexität der Beobachtungslage führt dazu, dass nicht der gesamte stochastische Prozess, sondern nur einzelne Charakteristika per Simulation analysiert werden. Ein Beispiel wäre die Ermittlung von $E(Y(t))$ für ein $t \in T$.

Als Hilfsmittel für die Analyse können Realisierungen von Y während der Simulation beobachtet werden. Wir nehmen an, dass insgesamt m Beobachtungen während eines Simulationslaufs durchgeführt werden und Y_j die Zufallsvariable ist, die Beobachtung j beschreibt. Der Parameter j kann unterschiedliche Bedeutungen haben.

- Es können Kunden gezählt werden und $Y_j$beschreibt die Verweilzeit des $j$ten Kunden im System.
- Man kann den Durchsatz eines Modells einer Fertigungsstraße nach j Stunden als $j$te Beobachtung interpretieren.
- Die Zeit des Ausfalls der $j$ten Komponente kann ebenfalls als $j$te Beobachtung dienen.

Der Wert von j ergibt sich damit aus der Simulationszeit oder dem Auftreten eines bestimmten Zustands oder Ereignisses. Durch Variation der Saat des Zufallszahlengenerators können unterschiedliche Trajektorien erzeugt werden. Wir nehmen im Folgenden an, dass n unterschiedliche Trajektorien beobachtet werden und bezeichnen mit y_{ij} die Realisierung von Y_j in der $i$ten Beobachtung (Trajektorie). Die Beobachtungen können in einer Matrix angeordnet werden, wobei Zeilen Beobachtungen einer Trajektorie und Spalten Beobachtungen einer Zufallsvariablen beinhalten.

$$\begin{array}{ccccc} y_{11}, & \cdots & y_{1j}, & \cdots & y_{1m} \\ \vdots & & \vdots & & \vdots \\ y_{i1}, & \cdots & y_{ij}, & \cdots & y_{im} \\ \vdots & & \vdots & & \vdots \\ y_{n1}, & \cdots & y_{nj}, & \cdots & y_{nm} \end{array}$$

Spalte j beschreibt Beobachtungen Y_j, die gemäß $F_{Y_j}(y)$ verteilt sind. Betrachten wir nun zwei Beobachtungen y_{ij}und y_{kl} und deren Relation. Falls $j = l$ und $i \neq k$, so beschreiben beide Werte von Y_j. Wenn die Sequenzen von Zufallszahlen, die zur Beobachtung von y_{ij} und y_{kj}führen sich nicht überlappen, so sind die generierten Zufallszahlen als unabhängig anzusehen und damit sind auch y_{ij}und y_{kj} unabhängige Beobachtungen von Y_j. Falls $j \neq l$ und $i \neq k$, so beschreiben y_{ij} und y_{kl} Beobachtungen $Y_j$und Y_l. Wie der Zusammenhang zwischen diesen Zufallsvariablen ist, muss für ein spezielles Problem unterschieden werden, wie später gezeigt wird.

Als Beispiel soll ein so genanntes $M/M/1$-System dienen. Die Schreibweise bezeichnet eine Station an der Kunden mit unabhängigen exponentiell verteilten Zwischenankunftszeiten eintreffen, in einem Warteraum potenziell unendlicher Kapazität nach FCFS bis zur Bedienung warten und anschließend von einem einzelnen Bediener eine exponentiell verteilte Zeit bedient werden (siehe auch 3.1). Sei λ die Ankunftsrate (= mittlere Anzahl Aufträge, die pro Zeiteinheit eintreffen) und μ die Bedienrate (= mittlere Anzahl Aufträge, die pro Zeiteinheit bedient werden, wenn der Bediener permanent arbeitet). Seien $\lambda = 0.5$ und $\mu = 1$ die konkreten Werte. Ferner sei Y_j die Kundenzahl im System nach j Zeiteinheiten und n_0 sei die Anzahl Kunden im System zum Zeitpunkt 0. Abbildung 2.56 zeigt den Verlauf von $E(Y(t))$ für unterschiedliche Werte von n_0. Es ist zu beachten, dass der Verlauf des Erwartungswerts dargestellt wird, also Aussagen über alle Trajektorien gemacht werden und nicht eine konkrete Trajektorie. Eine einzelne Trajektorie würde eine Treppenfunktion bilden. Es fällt auf, dass der Verlauf von $Y(t)$ von n_0 beeinflusst

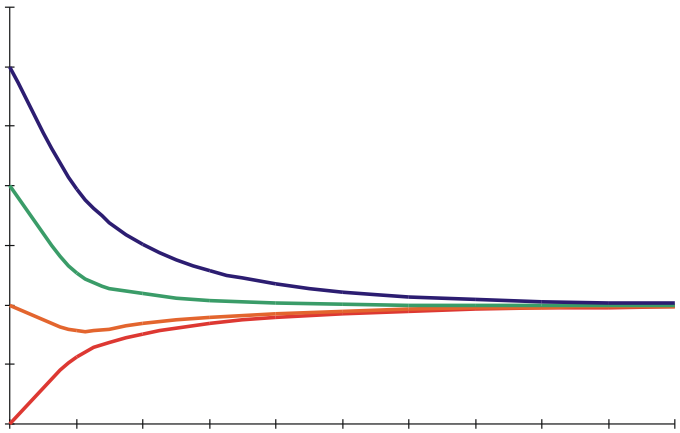

Abbildung 2.56: Verlauf von $E(Y(t))$ für unterschiedliche Startwerte n_0.

wird. Offensichtlich konvergiert in diesem Beispiel aber $Y(t)$ für $t \to \infty$ gegen einen festen Wert, unabhängig von n_0.

Das kleine Beispiel macht zwei Problem deutlich, nämlich die Abhängigkeit der beobachteten Größe vom initialen Zustand des Modells und die Abhängigkeit von aufeinander folgenden Beobachtungen.

Schätzung von $E(Y)$

In der Simulation soll Y mit Hilfe von Beobachtungen charakterisiert werden. In der Regel soll $E(Y)$ ermittelt werden, wobei durch entsprechende Interpretation des Erwartungswerts zahlreiche Größen ermittelt werden können, wie später noch gezeigt wird. $E(Y)$ soll geschätzt werden, deshalb benutzen wir wieder als Schreibweise $\tilde{Y}$ für den Schätzer und $\hat{Y}$ für den konkreten Schätzwert. Um Aussagen über die Qualität des Schätzer zu machen, werden Informationen über $\sigma^2(Y)$, die Varianz der beobachteten Größe benötigt, die natürlich ebenfalls nur geschätzt werden kann.

Wenn ausgehend von der bereits beschriebenen Beobachtungslage $E(Y_j)$ aus n Beobachtungen ermittelt werden soll, so ist

$$\hat{Y}_j = \frac{1}{n} \cdot \sum_{i=1}^{n} y_{ij}$$

die nahe liegende Methode zur Bestimmung des Schätzwertes. Man nennt $\hat{Y}_j$ einen *Punktschätzer* für $E(Y_j)$. Offensichtlich variiert der Wert von $\hat{Y}_j$ für unterschiedliche Saaten des Zufallszahlengenerators für die einzelnen Beobachtungsläufe. Um Aussagen über $E(Y_j)$ machen zu können, müssen wir Aussagen über mögliche Werte von $\hat{Y}_j$ machen und letztendlich beantworten, wie weit $\hat{Y}_j$ von $E(Y_j)$ entfernt ist. Da $\hat{Y}_j$ eine Realisierung von $\tilde{Y}_j$ ist und $\tilde{Y}_j$eine Zufallsvariable ist, können Aussagen nur auf Grund der Verteilungscharakteristika gemacht werden.

Wir haben bereits gesehen, dass $\tilde{Y}_j$ein erwartungstreuer Schätzer von $E(Y_j)$ ist womit $E(\tilde{Y}_j) = E(Y_j)$ gilt. Dies gilt sogar, wenn die einzelnen Beobachtungen nicht unabhängig sind. Weiterhin ist der Schätzer konsistent und damit gilt $\lim_{n\to\infty} P[|\tilde{Y}_j - E(Y_j)| > \epsilon] = 0$ für alle $\epsilon > 0$. Die Erwartungstreue und Konsistenz eines Schätzer macht keinerlei Aussagen über die Qualität des konkreten Schätzwertes nach n Beobachtungen. Ein erster Schritt, um zu solchen Aussagen zu gelangen ist die Ermittlung der Varianz des Schätzers $\tilde{Y}_j$. Die Varianz ist ein Maß für die mögliche Schwankung der Realisierungen $\hat{Y}_j$ und gibt damit Hinweise über die Genauigkeit der Schätzung.

Die folgende Formel zeigt die Herleitung der Varianz des Schätzer bei n Beobachtungen.

$$\begin{array}{lll} \sigma^2(\tilde{Y}_j) & = E\left(\left(Y - E(\tilde{Y}_j)\right)^2\right) & = \\ E\left(\left(\frac{1}{n}\cdot\sum_{i=1}^{n} Y_{ij} - E(Y_j)\right)^2\right) & = E\left(\left(\frac{1}{n}\cdot\sum_{i=1}^{n}(Y_{ij} - E(Y_j))\right)^2\right) & = \\ \frac{1}{n^2}\cdot\left(\sum_{i=1}^{n} E\left((Y_{ij} - E(Y_j))^2\right)\right. & \left. + \sum_{i=1}^{n}\sum_{k=1,k\neq i}^{n} E\left((Y_{ij} - E(Y_j))\cdot(Y_{kj} - E(Y_j))\right)\right) & \end{array} \tag{2.4}$$

Man sieht, dass in die Varianz die Abhängigkeiten zwischen den Beobachtungen durch die Doppelsummen einfließen. Im Gegensatz zum Erwartungswert des Schätzers wird die Varianz durch korrelierte Beobachtungen beeinflusst. Wenn angenommen wird, dass die beobachteten Werte y_{ij} $(i = 1, \ldots n)$ unabhängig und damit unkorreliert sind, so ist der Wert der Doppelsumme 0 und die Berechnung der Varianz reduziert sich auf die folgende einfache Form.

$$\sigma^2(\tilde{Y}_j) = \frac{1}{n^2}\sum_{i=1}^{n} E\left((Y_{ij} - E(Y_j))^2\right) = \frac{1}{n^2}\cdot n\cdot\sigma^2(Y_j) = \frac{\sigma^2(Y_j)}{n}$$

Aus dieser Darstellung lässt sich ablesen, dass die Schwankungsbreite des Schätzer

- mit der Schwankungsbreite der Beobachtungsgröße steigt und
- mit der Anzahl Beobachtungen fällt.

Damit kann für eine vorgegebene Beobachtungsgröße die Schwankung der Schätzung dadurch reduziert werden, dass die Zahl der Beobachtungen erhöht wird. Dies ist ein nachvollziehbares Verhalten. Zur konkreten Bestimmung der Varianz des Schätzer wird die unbekannte Varianz der Beobachtungsgröße benötigt, die damit ebenfalls geschätzt werden muss. Ein erwartungstreuer Schätzer für $\sigma^2(Y_j)$ lautet

$$\tilde{S}_j^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(Y_{ij} - \tilde{Y}_j\right)^2 \tag{2.5}$$

Damit ist $\tilde{S}_j^2/n$ ein erwartungstreuer Schätzer für $\sigma^2(\tilde{Y}_j)$ und $\hat{S}_j^2/n$ ist der konkrete Schätzwert. Die Varianz des Schätzers allein macht noch keine konkrete Aussagen über die potenzielle Abweichung eines Schätzwertes vom wahren Wert. Dazu sind weitere Verfahren und Annahmen notwendig. Bevor die zugehörigen Methoden vorgestellt werden, soll auf die Berechnung der Schätzwerte für Erwartungswert und Varianz eingegangen werden.

Praktische Berechnung der Schätzer

Beginnen wir mit der Berechnung von $\hat{Y}_j$. Es ist unpraktisch erste alle Werte y_{ij} zu speichern und abschließend $\hat{Y}_j$zu berechnen, da damit bei vielen Beobachtungen ein hoher Speicherplatzbedarf einhergeht und Ergebnisse auch erst zum Ende der Simulation vorliegen. Besser ist die Verwendung der Rekursion

$$\hat{Y}_j(k) = \frac{k-1}{k}\hat{Y_j(k-1)} + \frac{y_{jk}}{k}$$

mit $\hat{Y}_j(0) = 0$ und $k = 1, \ldots, n$. Die Berechnung ist für relativ große Werte von n numerisch stabil. Falls n sehr groß wird, so kann mehrstufig vorgegangen werden, indem nicht einzelne Werte y_{jk}, sondern Mittelwerte aus mehreren Beobachtungen aufsummiert werden und damit die Zahl der zu summierenden Werte reduziert wird.

Problematischer ist die Berechnung der Stichprobenvarianz. Die direkte Auswertung der Formel für den Schätzer der Varianz

$$\hat{S}_j^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(y_{ij} - \hat{Y}_j\right)^2$$

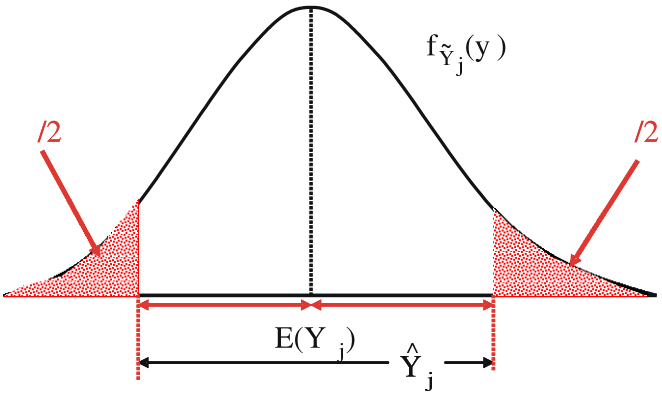

Abbildung 2.57: Skizze zur Verteilung des Schätzers $\tilde{Y}_j$.

führt dazu, dass die Varianz erst berechnet werden kann, wenn $\hat{Y}_j$ vorliegt. Da dies erst der Fall ist, wenn alle Werte ermittelt wurden, müssen alle y_{ij} gespeichert werden. Alternativ können die Summen der y_{ij} und y_{ij}^2 gespeichert werden und

$$\hat{S}_j^2 = \frac{1}{n-1}\left(\sum_{i=1}^{n}(y_{ij})^2 - n\cdot\left(\hat{Y}_j\right)^2\right)$$

berechnet werden. Dieses Vorgehen erfordert zwar wenig Speicherplatz, ist aber für große n numerisch instabil. Trotzdem wird in den meisten Implementierungen mit dieser Variante gearbeitet. Die Größe von n wird dadurch beschränkt, dass nicht einzelne Werte sondern Mittelwerte aus mehreren Beobachtungswerten in die Formel einfließen, wie später noch gezeigt wird.

Interpretation und Bestimmung von Konfidenzintervallen

Auch nach Schätzung der Varianz bleibt noch die Frage, wie weit der ermittelte Schätzer $\hat{Y}_j$ vom wahren Wert $E(Y_j)$ entfernt ist. Da wir es mit Zufallsvariablen zu tun haben, kann der wahre Abstand nur ermittelt werden, wenn $E(Y_j)$ bekannt ist, dann ist aber eine Bestimmung per Simulation unnötig. Die korrekte Frage müsste eigentlich lauten:

Wie groß ist die Wahrscheinlichkeit, dass $\tilde{Y}_j$ um mehr als ϵ von $E(Y_j)$ entfernt ist?

Also die Wahrscheinlichkeit α ist zu ermitteln, so dass $P[|\tilde{Y}_j - E(Y_j)| \geq \epsilon] = \alpha$. Angenommen die Dichtefunktion $f_{\tilde{Y}_j}(y)$ sei bekannt, dann liegt eine wie in Abbildung 2.57 gezeigte Situation vor. Um den wahren Wert $E(Y_j)$ wird ein Intervall der Breite 2ϵ gezogen, so dass die Wahrscheinlichkeit, dass der Wert von $\tilde{Y}_j$ innerhalb des Intervalls liegt $1-\alpha$ entspricht (d.h. $\int_{E(Y_j)-\epsilon}^{E(Y_j)+\epsilon} f_{\tilde{Y}_j}(x)dx = 1-\alpha$). Mit Wahrscheinlichkeit αliegt der Wert von $\tilde{Y}_j$ außerhalb des definierten Intervalls. Bei unbekannter Verteilung liefert die Kenntnis der Varianz des Schätzers oder eines Schätzwertes $\hat{S}_j^2$ nur ein Indiz für die Breite des Intervalls, aber kein konkretes Intervall.

Diese Beobachtungslage legt zwei Interpretationen nahe:

1. Wie groß ist die Wahrscheinlichkeit, dass $\hat{Y}_j$aus dem Intervall $[E(Y_j)-\epsilon, E(Y_j)+\epsilon]$ herausfällt?

2. Wie groß ist die Wahrscheinlichkeit, dass das Intervall $[E(Y_j)-\epsilon, E(Y_j)+\epsilon]$ den wahren Wert $E(Y_j)$ nicht enthält?

Die erste Fragestellung ist weniger relevant, da $E(Y_j)$ nicht bekannt ist. Hilfreicher ist der zweite Fall, da dort Aussagen über $E(Y_j)$ ausgehend vom bekannten $\hat{Y}_j$ getroffen werden. Damit kann

man prinzipiell, d.h. falls die Verteilung $\tilde{Y}_j$ bekannt wäre, Aussagen über ein Intervall treffen, in dem $E(Y_j)$mit vorgegebener Wahrscheinlichkeit liegt.

Offensichtlich existiert eine Abhängigkeit zwischen α und ϵ. Ausgehend von Abbildung 2.57 wird α kleiner, wenn ϵ größer und umgekehrt. Bisher wurde dabei so argumentiert, dass zu einem ϵ ein passendes α gesucht wurde. Es wurde also die Breite des Intervalls vorgegeben und dann die Wahrscheinlichkeit ermittelt, dass der wahre Wert im Intervall liegt. In der Praxis ist allerdings eine andere Fragestellung üblicher. Es wird die Wahrscheinlichkeit α vorgegeben und dazu die passende Intervallbreite ϵ ermittelt. Die übliche Fragestellung lautet:

Wenn man mit Wahrscheinlichkeit $1-\alpha$*sicher sein will, dass* $E(Y_j)$ *nicht mehr als* ϵ *von* $\hat{Y}_j$ *abweicht, welches* ϵ*ist dann zu wählen?*

Man nennt $\hat{Y}_j \pm \epsilon$ **ein** $(1-\alpha)\cdot 100\%$ **Konfidenzintervall für** $E(Y_j)$**.** $[\hat{Y}_j - \epsilon, \hat{Y}_j + \epsilon]$ **ist ein Intervallschätzer.**

Offensichtlich sind Intervallschätzer ungleich wertvoller als Punktschätzer, da sie Wahrscheinlichkeitsaussagen machen, während Punktschätzer keinerlei Angaben darüber enthalten, wie groß die Variabilität des Schätzer ist. Bisher haben wir nur Schätzer für den Erwartungswert und die Varianz von $\tilde{Y}_j$ kenne gelernt, ohne Kenntnis der Verteilung des Schätzers kann daraus kein Konfidenzintervall gewonnen werden. Wir werden im Folgenden zwei Methoden zur Bestimmung von Konfidenzintervallen vorstellen. Im ersten Fall werden Schranken berechnet, die für alle möglichen Verteilungen gelten und im zweiten Fall wird eine bestimmte Verteilung vorausgesetzt, die unter sehr allgemeinen Annahmen beobachtet werden kann.

Konfidenzintervalle nach Tschebyscheff

Die bekannte Ungleichung von Tschebyscheff lautet: Für eine Zufallsvariable X mit Erwartungswert $E(X)$ und Varianz $\sigma^2(X)$ gilt für beliebiges $c > 0$:

$$P[|X - E(X)| \geq c] \leq \sigma^2(X)/c^2$$

Angewendet auf die Berechnung von Konfidenzintervallen wie beschrieben, wird c durch ϵ und X durch $\tilde{Y}_j$ersetzt. Die Varianz von $\tilde{Y}_j$ lautet $\sigma^2(Y_j)/n$, so dass

$$P[|\tilde{Y}_j - E(Y_j)| \geq \epsilon] \leq \frac{\sigma^2(Y_j)}{n \cdot \epsilon^2}$$

Wir können dies an einem einfachen Beispiel, nämlich der Berechnung des 90% Konfidenzintervalls für $\hat{Y}_j$ vorführen. Dann ist $\alpha = \sigma^2(Y_j)/(n \cdot \epsilon^2) = 0.1$. Bei einem Stichprobenumfang von 10 gilt dann

$$\frac{\sigma^2(Y_j)}{10 \cdot \epsilon^2} = 0.1 \;\Leftrightarrow\; \epsilon^2 = \sigma^2(Y_j) \;\Leftrightarrow\; \epsilon = \sigma(Y_j)$$

Bei einem errechneten Punktschätzer $\hat{Y}_j$ ist $[\hat{Y}_j - \epsilon, \hat{Y}_j + \epsilon]$ in diesem Fall das gesuchte Konfidenzintervall.

Bei einer festen Varianz des beobachteten Wertes Y_j, was vorausgesetzt werden kann, erfordert eine Halbierung der Breite des Konfidenzintervalls eine Vervierfachung der Zahl der notwendigen Beobachtungen, da

$$\frac{\sigma^2(Y)}{n \cdot \epsilon^2} = \frac{\sigma^2(Y)}{m \cdot (\epsilon/2)^2} \;\Leftrightarrow\; m = 4 \cdot n$$

Dies impliziert auch, dass der Aufwand zur Ermittlung sehr genauer Simulationsresultate (d.h. schmaler Konfidenzintervalle) sehr groß werden kann.

Die Konfidenzintervallbestimmung nach Tschebyscheff hat zwei wesentliche Nachteile:

1. Die Berechnung des Konfidenzintervalls verwendet die Varianz $\sigma^2(Y_j)$ der beobachteten Größe. Dieser Wert ist aber in praktische allen relevanten Fällen unbekannt und muss durch den geschätzten Wert $\hat{S}_j^2$ ersetzt werden. Damit sind die Voraussetzungen von Tschebyscheff verletzt und die ermittelten Schranken gelten nicht mehr.

2. Tschebyscheff liefert sehr breite Konfidenzintervalle, da der Satz allgemein für beliebige Verteilungen gilt und damit Extremfälle einbezieht, die für die gegebene Beobachtungslage selten oder nie auftreten.

Üblicherweise wird der "pessimistische Charakter" der Konfidenzintervalle nach Tschebyscheff genutzt, um die Verwendung $\hat{S}_j^2$ statt $\sigma^2(Y_j)$ als Intervallschätzer zu rechtfertigen.

Wir betrachten als einfaches Beispiel das Werfen einer fairen Münze und interpretieren Zahl=1, Kopf=0. Dann gilt $E(Y) = 0.5$ und $\sigma^2(Y) = 0.25$. Es werden zwei Experimentserien mit jeweils 10 Würfen durchgeführt und einzeln und als konkatenierte Serie mit 20 Experimenten ausgewertet. Im Einzelnen liegen folgende Ergebnisse vor.

1. Experimentserie 1: 0000101001 liefert $\hat{Y} = 0.3$ und $\hat{S}^2 = 0.23333$. Daraus ergibt sich für $\alpha = 0.1$ das Konfidenzintervall $[-0.183, 0.783]$. Aus der Kenntnis des Modells folgt, dass der Wert nicht kleiner als 0 sein kann, so dass die untere Grenze durch 0 ersetzt werden kann.

2. Experimentserie 2: 0110111001 liefert $\hat{Y} = 0.6$ und $\hat{S}^2 = 0.26667$. Für $\alpha = 0.1$ erhält man das Konfidenzintervall $[0.084, 1.116]$. Auch hier hilft die Kenntnis des Modells, so dass die obere Grenze durch 1.0 ersetzt werden kann.

3. Durch Konkatenation der beiden Serien entsteht eine Serie der Länge 20 mit $\hat{Y} = 0.45$, $\hat{S}^2 = 0.2605$ und einem Konfidenzintervall $[0.089, 0.811]$ für $\alpha = 0.1$.

Es sollen noch einmal die Aussagen eines Intervallschätzer vor Augen geführt werden. Im obigen Fall liegt der wahre Wert in 90% der Fälle im ermittelten Intervall und liegt mit 10% Wahrscheinlichkeit außerhalb des Intervalls. Also bei 10 Experimenten der obigen Art, sollte im Mittel ein Experiment ein Intervall liefern, das 0.5 nicht enthält. Da Tschebyscheff aber sehr pessimistische Schranken liefert, kann man in der Regel sagen, dass der wahre Wert mit mindestens Wahrscheinlichkeit $1 - \alpha$ im ermittelten Intervall liegt.

Konfidenzintervalle nach dem zentralen Grenzwertsatz

Um bessere, also schmalere Konfidenzintervalle zu gewinnen, muss Information über die Verteilung des Schätzers $\tilde{Y}_j$ in die Konfidenzintervallbestimmung einbezogen werden. Offensichtlich hängt die Verteilung $\tilde{Y}_j$ aber von der in der Regel unbekannten Verteilung von Y_j ab. Die Frage bleibt damit, ob Aussagen über die Verteilung von $\tilde{Y}_j$ ohne Einbeziehung der Verteilung von Y_j gemacht werden können. Zur Hilfe kommt uns dabei der zentrale Grenzwertsatz.

Satz 1 *Seien $Y_{:1}, \ldots Y_n$ unabhängig identisch verteilte Zufallsvariablen mit endlichem Erwartungswert $E(Y) = \mu$ und endlicher Varianz σ^2 und sei $\tilde{\mu} = 1/n \cdot \sum_{i=1}^{n} Y_i$. Sei ferner eine Zufallsvariable Z definiert als*

$$Z_n = \frac{\tilde{\mu} - \mu}{\sigma/\sqrt{n}}$$

Dann approximiert die Verteilung von Z_n für große n die $N(0,1)$-Verteilung beliebig genau (d.h. $\lim_{n\to\infty} F_{Z_n}(z) = \Phi(z)$ wobei $\Phi(z)$ die Verteilungsfunktion von $N(0,1)$ ist).

Der zentrale Grenzwertsatz gilt auch, wenn σ^2 durch $\tilde{S}^2$ ersetzt wird. Damit können, falls n groß genug ist, Konfidenzintervalle aus den "kritischen" Werten der Normalverteilung gewonnen werden. Ausgehend von den vertafelten Werten von $N(0,1)$ kann damit wie folgt ein Konfidenzintervall für $\tilde{Y}$ gewonnen werden.

$$P[|Z| \geq \epsilon] = P\left[\left|\frac{\tilde{Y} - E(Y)}{\tilde{S}/\sqrt{n}}\right| \geq \epsilon\right] = \alpha \;\Rightarrow\; P\left[\left|\tilde{Y} - E(Y)\right| \geq \epsilon \cdot \tilde{S}/\sqrt{n}\right] = \alpha$$

Damit lautet das Konfidenzintervall

$$\tilde{Y} \pm z_\alpha \cdot \tilde{S}/\sqrt{n}$$

Verteilung	n=5	n=10	n=20	n=40
Normal	0.910	0.902	0.898	0.900
Exponential	0.854	0.878	0.870	0.890
Lognormal	0.758	0.768	0.842	0.852
Hyperexp.	0.584	0.586	0.682	0.774

Tabelle 2.2: Überdeckung der mit der t-Verteilung ermittelten Konfidenzintervalle (aus [11, S. 257]).

wobei z_α der Wert des $\alpha/2$ Quantils von $N(0,1)$ ist (siehe Abbildung 2.57).

Übertragen auf unser Würfelbeispiel liefert die Normalverteilung ein Konfidenzintervall von $[0.262, 0.638]$ für die konkatenierte Sequenz. Dies ist eine deutliche Reduktion des Intervalls $[0.089, 0.811]$, das mittels Tschebyscheff berechnet wurde. Allgemein liefert die Normalverteilung deutlich schmalere Konfidenzintervalle als Tschebyscheff.

Es bleibt aber die Frage, wie groß n sein muss, damit die Annahme der Normalverteilung für den Schätzer gerechtfertigt ist und damit die nach dem zentralen Grenzwertsatz ermittelten Konfidenzintervalle den wahren Wert wirklich mit der vorgegebenen Wahrscheinlichkeit enthalten.

Die notwendige Größe von n hängt offensichtlich von der unbekannten Verteilung von Y ab. Es ist deshalb wichtig, sich vor Augen zu führen, welche Auswirkungen eine irrtümliche Normalverteilungsannahme hat. Auch dies hängt natürlich von der wahren Verteilung von Y ab. Im ungünstigen Fall, kann der wahre Wert $E(Y)$ mit einer Wahrscheinlichkeit, die deutlich größer als α ist, außerhalb des berechneten Konfidenzintervalls liegen. Man spricht dann davon, dass die Abdeckung des Konfidenzintervalls kleiner als α ist. Dies bedeutet, dass die Simulationsresultate eine Präzision vortäuschen, die nicht gerechtfertigt ist und auf dieser Basis unter Umständen falsche Entscheidungen getroffen werden. Eine solche Situation sollte möglichst vermieden werden.

Leider sind nur sehr wenige Resultate über die Verteilung von $\tilde{Y}$ bekannt, selbst unter der unrealistischen Annahme, dass die Verteilung von Y bekannt ist. Eines der wenigen bekannten Resultate betrachtet den Fall, dass Y selbst schon normalverteilt ist. In diesem Fall ist

$$\frac{\tilde{Y} - E(Y)}{\tilde{S}/\sqrt{n}}$$

t-verteilt mit $n-1$ Freiheitsgraden. Die kritischen Werte der t-Verteilung sind, ähnlich wie die Werte der Normalverteilung, vertafelt (siehe [5, Tabelle A.5]). Damit lautet das Konfidenzintervall

$$\hat{Y} \pm t_{n-1,1-\alpha/2} \cdot \frac{\hat{S}}{\sqrt{n}} \tag{2.6}$$

wobei $t_{n-1},1-\alpha/2$ das $1-\alpha/2$ Quantil einer t-Verteilung mit $n-1$ Freiheitsgraden ist. Für $n \to \infty$ konvergiert die t-Verteilung gegen eine Normalverteilung. Für kleine n sind die Quantile der t-Verteilung allerdings größer als die zugehörigen Quantile der Normalverteilung, so dass die Konfidenzintervalle pessimistischer sind. Auf unser Beispiel angewendet liefert die t-Verteilung ein Konfidenzintervall von $[0.252, 0.647]$ im Gegensatz zu $[0.262, 0.638]$ mittels der Normalverteilung. Auch wenn in der Praxis Y nicht unbedingt normalverteilt ist, werden oftmals Konfidenzintervalle mit Hilfe der t-Verteilung und nicht mit der Normalverteilung ermittelt, da die resultierenden Konfidenzintervalle etwas pessimistischer sind, ohne gleich so breit wie bei Verwendung von Tschebyscheff zu sein.

Mangels theoretischer Resultate, kann eine Untersuchung der Abdeckung von Konfidenzintervallen nur experimentell und damit empirisch erfolgen. Dazu werden Zufallszahlen aus einer bekannten Verteilung gezogen und es wird ein Konfidenzintervall für den Erwartungswert ermittelt. Dieses Experiment wird mehrfach wiederholt und der Anteil der Konfindenzintervalle bestimmt, die den wahren Wert enthalten. Bei k Wiederholungen und Wahrscheinlichkeit α sollten $\approx (1-\alpha)\cdot k$ Experimente Konfidenzintervalle liefern, die den wahren Wert beinhalten. Tabelle 2.2 gibt die Resultate einiger Experimente wieder. Es wurden jeweils $k = 500$ Wiederholungen der Experimente

durchgeführt und 90% Konfidenzintervalle für den Erwartungswert aus $n = 5, 10, 20, 40$ Zufallszahlen einer gegebenen Verteilung bestimmt. In der Tabelle wird der Anteil der Experimente angegeben, bei denen der wahre Wert im Konfidenzintervall lag. Da es sich um Experimente handelt, ist nicht zu erwarten, dass genau 90% der Experimente Werte im Konfidenzintervall liefern und 10% Werte außerhalb liefern. Trotzdem sollte bei 500 Wiederholungen der ermittelte Wert in diesem Bereich liegen. Wie die Experimente zeigen, ist die Überdeckung für die Normalverteilung sehr gut. Dies war zu erwarten, da genau eine t-Verteilung vorliegt. In den anderen Fällen scheinen die Konfidenzintervalle zu optimistisch zu sein. So wird die Überdeckung mit steigendem n zwar besser, für die Verteilungen die sich deutlich von einer Normalverteilung unterscheiden, wie die Lognormalverteilung und insbesondere die Hyperexponentialverteilung, ist die Überdeckung aber auch für $n = 40$ noch nicht zufriedenstellend und die Simulationsergebnisse würden eine nicht gegebene Sicherheit vortäuschen. In diesen Fällen sind also deutlich höhere Beobachtungszahlen notwendig.

Bestimmung weiterer Größen neben $E(Y)$

Bisher wurde die Auswertung der Simulation auf die Schätzung von Erwartungswerten eingeschränkt. Auch wenn Erwartungswerte von Größen wie Verweilzeiten oder Pufferfüllungen erste Aussagen über das Systemverhalten erlauben, sind sie oft nicht ausreichend für die Systemanalyse. So gibt eine mittlere Pufferfüllung von 10 keine Hinweise darauf, wie oft ein Puffer der Größe 20 überlaufen wird. Eine mittlere Bearbeitungszeit von 0.1 Sekunden in einem System mit Realzeitanforderungen sagt nicht darüber aus, wie viele Bearbeitungen länger als die Zeitschranke von 0.5 Sekunden dauern. Für solche und ähnliche Fragestellungen werden üblicherweise Schätzungen der folgenden Größen benötigt.

- Die Schätzung höherer Momente $E(Y^k)$ für $k = 2, 3, 4, \ldots$
- Die Schätzung von der Werten der Verteilungsfunktion an einer Stelle$y$, also $P[Y \leq y] = F(y)$.
- Die Schätzung von p-Quantilen ($p \in (0, 1)$), also des Wertes y_p, so dass $P[Y \leq y_p] = F(y_p) = p$.

Die ersten beiden Fälle lassen sich sehr einfach auf das bisherige Vorgehen abbilden. Dazu wird eine neue Resultatvariable X definiert, deren Erwartungswert gerade die gesuchte Größe ist. Für die Schätzung des $k$ten Moments von Y_i wäre $X = (Y_i)^k$, $\tilde{X} = (1/n) \cdot \sum_{j=1}^{n} (Y_{ij})^k$ wäre der Schätzer und $\hat{X} = (1/n) \cdot \sum_{j=1}^{n} (y_{ij})^k$ der konkrete Schätzwert. Für die Schätzung eines Wertes der Verteilungsfunktion definieren wir eine binäre Zufallsvariable X mit

$$X = \begin{cases} 1 & \text{falls } y \leq Y \\ 0 & \text{sonst} \end{cases}$$

Dann ist $E(X) = P[Y \leq y]$ und $\tilde{X}$ kann wie üblich ermittelt werden.

Auch wenn die einzelnen Größen sich als Schätzer von Erwartungswerten darstellen lassen, so ist ihre praktische Ermittlung oftmals problematisch. Gerade bei höheren Momenten und großen/kleinen Werten der Verteilungsfunktion, die von wenigen Werten über-/unterschritten werden, sind oftmals sehr große Stichproben notwendig, um akzeptable Konfidenzintervalle zu erreichen.

Komplexer ist die Ermittlung von Quantilen. Da der Wert von y_p nicht vorab bekannt ist, werden Quantile oftmals erst nach Ermittlung aller Werte der Stichprobe aus der geordneten Stichprobe ermittelt. Dies hat natürlich den Nachteil, dass alle Werte gespeichert werden müssen und Schätzer erst nach Ablauf der Simulation zur Verfügung stehen. In [16] wird eine Methode vorgestellt, mit deren Hilfe Quantile auch während der Simulation geschätzt werden können. Der zugehörige Algorithmus ist allerdings relativ komplex und würde den Rahmen der Vorlesung sprengen.

Klassifizierung von Simulationen bzgl. der Auswertung

Bisher haben wurde die Auswertung einer Resultatgröße relativ abstrakt untersucht. Es soll nun konkret unterschieden werden, welche Arten von Resultaten in einer Simulation ermittelt werden können. Dazu betrachten wir zuerst einige Beispiele.

- In einer Fertigungsstraße, die im Dreischichtbetrieb 7 Tage in der Woche läuft, soll der Durchsatz ermittelt werden. Es kann davon ausgegangen werden, dass die Belastung des Systems unabhängig von der Zeit ist und auch die einzelnen Arbeitsschritte zeitunabhängig ausgeführt werden. Weiterhin kann man davon ausgehen, dass die Fertigungsstraße über einen längeren Zeitraum störungsfrei arbeitet.

- Die Kapazität eines Kommunikationsnetzes soll geplant werden. Dazu wird die zu erwartende Ankunftsrate von Nachrichten und die Belegungszeit in Hochlastphasen geschätzt und es wird eine Leitungskapazität gesucht, so dass eine vorgegebene Dienstqualität eingehalten wird. Zur Ermittlung dieser Kapazität wird das System für die zur Verfügung stehenden Kapazitäten analysiert.

- Eine optimale Belieferungsstrategie eines Einzelhändlers mit Speiseeis soll ermittelt werden. Die Belieferungsmenge muss natürlich abhängig vom Verbrauch sein. Der Verbrauch richtet sich nach dem Wetter und der Temperatur, die wiederum von der Jahreszeit beeinflusst werden.

- Die Zeit bis zum Ausfall eines Computersystems soll bestimmt werden. Dazu sind die Ausfallzeiten der einzelnen Komponenten und deren Interaktion bekannt.

- Die Kassenauslastung eines Supermarktes zwischen 9^{00} und 12^{00} Uhr ist zu ermitteln. Der zu erwartende Kundenstrom richtet sich natürlich nach der Uhrzeit. Weiterhin ist davon auszugehen, dass der Supermarkt um 9^{00} Uhr öffnet.

Die verschiedenen Aufgabenstellungen stellen unterschiedliche Anforderungen an die Auswertung der Simulationsmodelle. In den ersten beiden Fällen wird ein System betrachtet, bei dem von einer sehr langen Laufzeit ausgegangen wird, während der sich das System gleich verhält. Es ist zu beachten, dass gleiches Verhalten nicht bedeutet, dass das Verhalten deterministisch ist. Es kann sehr wohl stochastische Schwankungen geben, die beteiligten Zufallsvariablen und stochastischen Prozesse in der Modellbeschreibung sind aber unabhängig von der Modellzeit. Das dritte Beispiel beschreibt ebenfalls eine Situation, in der das System über einen langen Zeitraum untersucht wird (mehrere Jahre). Im Gegensatz zu den ersten beiden Beispielen ändern sich aber die Zufallseinflüsse im Modell mit der Modellzeit (d.h. Jahreszeit). Das vierte Beispiel beschreibt einen Ablauf, der mit dem Ausfall des Systems endet. Die Simulation terminiert also, wenn ein bestimmter Systemzustand erreicht wird. Auch im fünften Beispiel haben wir es mit einer terminierenden Simulation zu tun. Im Gegensatz zum vorherigen Beispiel ist das Ende der Simulation aber nun durch die Modellzeit 12^{00} vorgegeben. Im Gegensatz zu den ersten drei Beispielen, läuft das System nicht längere Zeit, da der Supermarkt abends schließt und am nächsten Morgen praktisch neu begonnen wird.

Die Beispiele liefern eine Basis, auf der man Simulationsexperimente bzgl. ihrer Auswertung unterscheiden kann. Zum einen betrachtet man *terminierende Simulation*. Die Simulation startet zum Zeitpunkt $t = 0$ und endet zu einem Zeitpunkt T_E. T_E kann entweder zu Beginn der Simulation vorgegeben werden oder ergibt sich im Laufe der Simulation durch das Eintreten eines Zustandes oder Ereignisses. Ziel ist die Ermittlung von $Y(t)$ zu einem Zeitpunkt $t = T_E$ oder in einem Intervall $[T_0, T_E]$[1] mit $0 \leq T_0 < T_E$. Zum anderen werden *nicht-terminierende Simulationen* betrachtet. Dieser Fall wird gleich etwas weiter unterteilt, da bei nicht-terminierenden Simulation unterschiedliche Verhalten auftreten können.

[1] Es können natürlich auch offenen oder halboffene Intervalle betrachtet werden.

M/M/1-System			Sys. mit Ausfällen		
n	Abdeckung	Breite rel. zu $\hat{Y}(n)$	n	Abdeckung	Breite rel. zu $\hat{Y}(n)$
5	0.880 ± 0.024	0.67	5	0.708 ± 0.033	1.16
10	0.864 ± 0.025	0.44	10	0.750 ± 0.032	0.82
20	0.886 ± 0.023	0.30	20	0.800 ± 0.029	0.60
40	0.914 ± 0.021	0.21	40	0.840 ± 0.027	0.44

Tabelle 2.3: Beispiele für die Abdeckung der Konfidenzintervalle nach (2.7) für ein M/M/1-System und ein System mit ausfallenden Komponenten [11, S. 508-509]

Terminierende Simulation

Wie bereits dargestellt wird bei der terminierenden Simulation das gesuchte Resultat entweder zum Zeitpunkt T_E oder im Intervall $[T_0, T_E]$ analysiert. Ziel ist die Schätzung von $E(Y)$, mit $Y = Y(T_E)$ oder $Y = \int_{T_0}^{T_E} Y(t)dt$. Zur Bestimmung des Schätzers $\tilde{Y}$ werden Beobachtungen der Zufallsvariable Y benötigt. Im Gegensatz zur auf Seite 98 dargestellten Situation kann aus einem Simulationslauf genau eine Beobachtung ermittelt werden. Die Beobachtungen aus n Simulationsläufen werden deshalb als $Y_1, \ldots, Y_n$ nummeriert. Jedes Y_i ist eine Zufallsvariable mit einer Verteilung, die der von Y entspricht. Der Simulationslauf wird n mal wiederholt, man spricht in diesem Fall von n *Replikationen*. Wir erweitern die Notation etwas und benutzen $\tilde{Y}(n)$, $\hat{Y}(n)$, $\tilde{S}^2(n)$ und $\hat{S}^2(n)$ für die entsprechenden Größen nach n Replikationen. Ferner ist y_i der Wert der $i$ten Replikation. Das Konfidenzintervall wird dann wie üblich berechnet.

$$\hat{Y}(n) \pm t_{n-1,1-\alpha/2} \cdot \sqrt{\frac{\hat{S}^2(n)}{n}} \tag{2.7}$$

Die Abdeckung dieses Konfidenzintervalls hängt ab, von der Nähe der Verteilung von Y zu einer Normalverteilung, der Größe von n und der Unabhängigkeit der Y_i.

Die Unabhängigkeit der Y_i hängt von der Unabhängigkeit der verwendeten Zufallszahlen ab. Ausgehend von einer Saat s_i wird Y_i erzeugt. Da üblicherweise LCGs (siehe Abschnitt 2.3) zur Generierung verwendet werden, entspricht s_i einem Wert in der Sequenz der generierten Zufallszahlen. Die ausgehend von s_i und s_j generierten Zufallszahlen können als unabhängig gelten, wenn sich die Sequenzen nicht überlappen. Dies bedeutet, dass die mit s_j startende Sequenz nicht s_i enthalten darf und umgekehrt. Um dies praktisch zu erreichen gibt es verschiedene Möglichkeiten. Viele Generatoren erlauben die Definition von Zufallszahlenströmen, die an weit auseinander liegenden Punkten im Zyklus beginnen. Es kann für jede Replikation ein neuer Strom definiert werden, so dass sich die Zufallszahlen nicht überlappen, solange nicht mehr Zahlen erzeugt werden, als der Abstand zwischen den Strömen beträgt. Wenn man davon ausgeht, dass die Replikationen nacheinander ausgeführt werden, was wir tun werden, so kann die Saat am Ende einer Replikation (= der aktuelle Wert von $x_i$im Generator) ausgelesen werden und als neue Saat für die nächste Replikation verwendet werden. Eine Funktion zum Auslesen der Saat ist bei den meisten Zufallszahlengeneratoren vorhanden. Mit diesem Vorgehen erreicht man Folgen unabhängiger Zufallszahlen in den Replikationen, solange der Generator an sich eine Folge unabhängiger Zufallszahlen erzeugt und insgesamt weniger als κ Zufallszahlen erzeugt werden.

Auch wenn die verwendeten Zufallszahlen unabhängig sind, bedeutet dies nicht zwangsläufig, dass die Abdeckung der nach (2.7) bestimmten Konfidenzintervalle wirklich α ist, da die Berechnung von normalverteilten Beobachtungen ausgeht und sonst nur asymptotisch für $n \to \infty$ gilt. Auch hier kann die Abdeckung nur empirisch an Hand von Beispielen untersucht werden, bei denen der exakte Wert bekannt ist. In diesem Fall werden dann n Experimente durchgeführt. Wenn s dieser Experimente den wahren Wert enthalten, so ist $\hat{p} = s/n$ ein Schätzer für die Abdeckung des Konfidenzintervalls. Auch für diesen Schätzer kann man ein Konfidenzintervall zum Signifikanzniveau α' bestimmen, welches dann

$$\hat{p} \pm z_{\alpha'} \cdot \sqrt{\frac{\hat{p} \cdot (1 - \hat{p})}{n}}$$

lautet, wobei $z_{\alpha'}$ das α' Quantil von $N(0,1)$ ist.

Die Ergebnisse von zwei Beispielen sind in Tabelle 2.3 zu finden. Das erste Beispiel ist ein $M/M/1$-System mit Bedienrate $\mu = 1$ und Ankunftsrate $\lambda = 0.9$. Ausgehend von einem leeren System wird die mittlere Verweilzeit der ersten 25 Aufträge ermittelt. Der Mittelwert und das zugehörige Konfidenzintervall wird für $n = 5, 10, 20, 40$ Replikationen bestimmt und das Experiment 500mal wiederholt, um die Abdeckung des 90% Konfidenzintervall zu schätzen. Die Abdeckung wird mit dem zugehörigen $\alpha' = 95\%$ Konfidenzintervall geschätzt. Wie in der Tabelle deutlich wird, ist die Abdeckung des Konfidenzintervalls in allen Fällen, also auch für kleine n, gut und die Breite des Konfidenzintervalls nimmt mit steigendem n ab, wie es zu erwarten war.

In einem zweiten Beispiel wird ein System betrachtet, das aus drei Komponenten besteht. Jede Komponente hat eine Weibull-verteilte Ausfallzeit. Das System arbeitet, wenn die erste und die zweite oder die dritte Komponente arbeiten. Da die Weibull-Verteilung stark von einer Normalverteilung abweicht, ist zu erwarten, dass auch die Ausfallzeit insgesamt stark von der Normalverteilung abweicht. Darüber hinaus war im ersten Beispiel der Resultatwert aus den 25 Einzelresultaten zusammengesetzt, während nun der erste oder zweite Komponentenausfall zum Systemausfall führt. Wie man aus der Tabelle entnehmen kann, ist die Abdeckung der Konfidenzintervalle relativ schlecht, in keinem Fall ist der wahre Wert 0.9 im Konfidenzintervall enthalten. In diesem Beispiel ist also n deutlich größer zu wählen, um eine vernünftige Abdeckung zu bekommen.

Bisher wurde die Anzahl Replikationen und das Signifikanzniveau α vorgegeben und es entstand ein Konfidenzintervall unbekannter Breite. Eigentlich soll aber ein Resultat mit vorgegebener Genauigkeit ϵ^* und vorgegebenem Signifikanzniveau α ermittelt werden. Es soll also gelten

$$1 - \alpha \approx P[\tilde{Y} - \epsilon \leq E(Y) \leq \tilde{Y} + \epsilon] = P[|\tilde{Y} - E(Y)| \leq \epsilon] \leq P[|\hat{Y} - E(Y)| \leq \epsilon^*]$$

Sei $n(\epsilon^*)$ die Anzahl Replikationen, die notwendig ist, damit die halbe Breite des Konfidenzintervalls kleiner gleich ϵ^* ist.

$$n(\epsilon^*) = \min \left\{ n : t_{n-1,1-\alpha/2} \sqrt{\frac{\hat{S}^2(n)}{n}} \leq \epsilon^* \right\}$$

Der Wert von $n(\epsilon^*)$ lässt sich nicht vorab bestimmen. Deshalb wird das Konfidenzintervall während der Simulation bestimmt und mit dem gewünschten Wert verglichen. Falls das Konfidenzintervall die gewünschte Breite hat, wird die Simulation abgebrochen.

In der Regel ist eine Genauigkeit γ relativ zum zu ermittelnden Wert $E(Y)$ zu bestimmen. Also $t_{n-1,1-\alpha/2} \cdot \sqrt{\hat{S}^2(n)/n}/E(Y) \leq \gamma$ sollte gelten, falls $E(Y) \neq 0$. Für $E(Y)$ gleich 0 oder nahe bei 0 sollte mit absoluten Werten gearbeitet werden. Da $E(Y)$ unbekannt ist, wird es durch $\hat{Y}(n)$ ersetzt und wir erhalten $\epsilon^* = \gamma \cdot \hat{Y}(n)$. Während der Simulation kann der Wert der notwendigen Beobachtungen $n(\gamma)$ aus den Ergebnissen nach n ($< n(\gamma)$) Beobachtungen geschätzt werden. Es gilt dann

$$t_{n-1,1-\alpha/2} \cdot \sqrt{\frac{\hat{S}^2(n)}{n(\gamma)}} \leq |\hat{Y}(n)| \cdot \gamma \;\Rightarrow\; n(\gamma) \approx \frac{(t_{n-1,1-\alpha/2})^2 \cdot \hat{S}^2(n)}{(\hat{Y}(n) \cdot \gamma)^2}$$

Ein weiterer Aspekt, der bisher nicht explizit behandelt wurde, ist die Bestimmung des initialen Zustands zu Beginn der Simulation. Bei der terminierenden Simulation ist der initiale Zustand oft vorgegeben. Übliche Beispiele sind, dass alle Komponenten lauffähig sind, das System leer ist, etc. Wenn der initiale Zustand nicht vorgegeben ist, so soll oft in einem typischen Zustand gestartet werden. Ein solcher kann aus Messungen resultierende oder in einer Vorabsimulation vor dem Start der eigentlichen Simulation erzeugt werden. Einige weitere Details werden bei der nicht-terminierenden Simulation betrachtet.

Nicht-terminierende Simulation

Im Gegensatz zur terminierenden Simulation mit einem endlichen Beobachtungszeitraum geht es bei der nicht-terminierenden Simulation darum, ein System über einen potenziell unendlichen

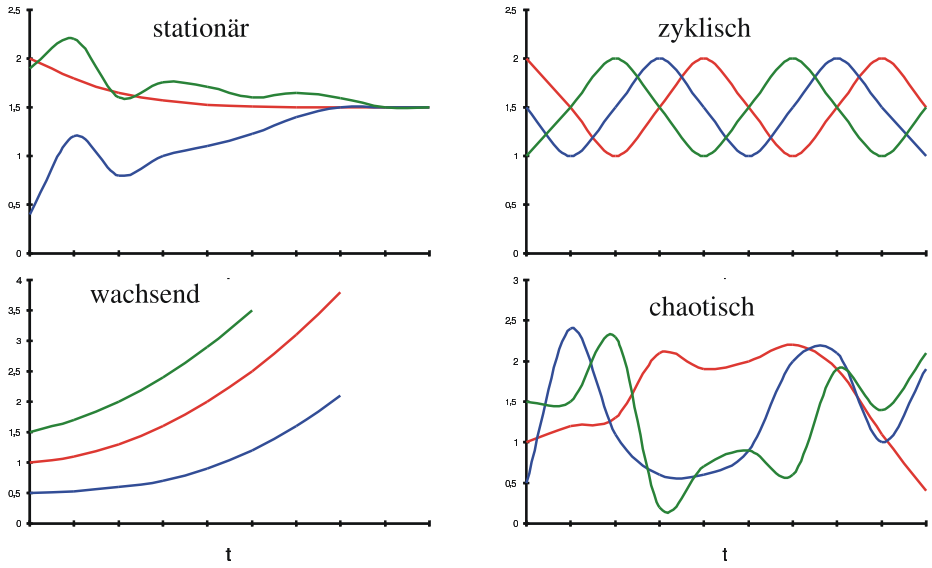

Abbildung 2.58: Mögliche Verläufe von $Y(t)$ für nicht-terminierende Simulationen.

Zeitraum zu beobachten und zu analysieren. In der Realität läuft natürlich keine System und auch kein Modell unendlich lange. Bei nicht terminierenden Simulationen wird aber davon ausgegangen, dass das System lange genug läuft, um typisches Verhalten zu zeigen. Bei unendlicher/sehr langer Laufzeit kann ein System/Modell unterschiedliche Verhaltensweisen zeigen. Abbildung 2.58 stellt den Verlauf von $E(Y(t))$ für vier verschiedene Systeme und jeweils drei unterschiedliche initiale Zustände s_0 dar. Der initiale Zustand wird auch als Startzustand bezeichnet. Es ist zu beachten, dass hier tatsächlich der Verlauf von $E(Y(t))$ dargestellt wird, der so nicht beobachtet werden kann. Eine einzelne Simulation würde stochastische Schwankungen aufweisen und ließe sich deshalb nicht so eindeutig einordnen. Es lassen sich auf Basis der Beobachtung von $E(Y(t))$ vier Ablaufmuster unterscheiden.

Im ersten Fall, dem stationären Verhalten, konvergiert $E(Y(t))$ gegen einen Wert für $t \to \infty$ unabhängig vom initialen Zustand. Im Allgemeinen gilt dann sogar $\lim_{t\to\infty} Y(t) = Y$ für eine Zufallsvariable Y, d.h. nicht nur der Erwartungswert, sondern die gesamte Verteilung konvergiert unabhängig vom Startzustand. Bei zyklischem Verhalten ändert sich $E(Y(t))$, es existiert aber ein $\Delta > 0$, so dass $E(Y(t)) = E(Y(t + \Delta))$. Je nach Startzustand verschiebt sich der Verlauf. Bei wachsendem Verhalten wächst $Y(t)$ mit t über alle Grenzen, d.h. $\lim_{t\to\infty} Y(t) = \infty$. Der letzte Fall ist das chaotische Verhalten, bei dem $E(Y(t))$ in Abhängigkeit von s_0 ganz unterschiedliche Verhaltensmuster aufweist. Kleine Änderungen im Startzustand können zu größeren Änderungen beim beobachteten Verhalten führen.

Wir werden uns nur mit den ersten beiden Abläufen beschäftigen, da die beiden letzten Abläufe vom Standpunkt der Systemanalyse nur durch terminierende Simulationen untersucht werden können. Primär wird uns die stationäre Simulation interessieren, da diese sehr breit eingesetzt wird, wenn Systeme lange laufen und einen *eingeschwungenen Zustand* erreichen, d.h. $E(Y(t))$ ist konstant über einen großen Teil des Beobachtungsintervalls $[0, T]$.

Stationäre Simulation

Sei $E(Y(t)|s_0)$ der Erwartungswert von Y zum Zeitpunkt t unter der Bedingung, dass das Modell zum Zeitpunkt 0 im Zustand s_0 war. Bei den im Folgenden behandelten Systemen wird davon ausgegangen, dass das Modell den stationären Zustand unabhängig vom Startzustand erreicht. D.h. $\lim_{t\to\infty} E(Y(t)|s_0) = E(Y)$ für alle s_0. Ziel der Simulation ist die Bestimmung von $E(Y)$. Da die Anforderung als Grenzwert definiert ist, werden keine Aussagen über endliche Beobachtungs-

intervalle gefordert. In den meisten Fällen wird deshalb gelten

$$E(Y(t)|s_0) \neq E(Y(t)|s_0') \quad \text{und} \quad E(Y(t')|s_0) \neq E(Y(t)|s_0)$$

für $t \neq t'$ und $s_0 \neq s_0'$. Um sinnvolle Ergebnisse per Simulation ermitteln zu können, wird deshalb weiterhin vorausgesetzt, dass ein Zeitpunkt t_s existiert, so dass für alle $t \geq t_s$

$$E(Y(t)|s_0) \approx E(Y(t)|s_0') \approx E(Y)$$

für alle s_0, s_0'. Auch diese Anforderung ist natürlich formal nicht nachweisbar und muss vorausgesetzt und mittels Beobachtungen der Simulation validiert werden. In der Regel wird der Zeitpunkt t_s ab dem $E(Y(t)|s_0) \approx E(Y)$ gilt von s_0 abhängig. Die Wahl von s_0 wird später untersucht und es wird davon ausgegangen, dass s_0 fest vorgegeben ist. Zur Vereinfachung der Schreibweise benutzen wir in diesem Fall $E(Y(t)$ statt $E(Y(t)|s_0)$.

Der erste Schritt der Auswertung ist die Bestimmung von t_s, so dass $E(Y(t)) \approx E(Y)$. Da vor dem Zeitpunkt t_s offensichtlich $E(Y(t)) \neq E(Y)$ gilt, sollte wie folgt bei der Simulation vorgegangen werden:

- Starte den Simulator im Zustand s_0 zum Zeitpunkt $t = 0$.
- Simuliere bis zum Zeitpunkt t_s, ohne Daten zu erheben.
- Starte die Beobachtung der Simulation zum Zeitpunkt t_s und simuliere bis zum Simulationsende.

Analog zum allgemeinen Fall, können einer oder mehrere Simulationsläufe durchgeführt werden. Bei einer Beobachtungslage wie auf Seite 98 wurden n Simulationsläufe durchgeführt und in jedem Lauf m Beobachtungen getätigt. Alle Beobachtungen wurden nach dem Zeitpunkt t_s erhoben. Zur Wahl von $t_s$gibt es zwei widersprüchliche Argumente:

- Das Sicherheitsargument impliziert, dass t_s möglichst groß gewählt wird, damit man sicher ist, dass $E(Y(t)) \approx E(Y)$ für $t \geq t_s$ wirklich gilt.
- Das Aufwandsargument impliziert, dass t_s möglichst klein gewählt wird, damit möglichst viele Beobachtungen ermittelt werden können.

Nahe liegender Weise ist t_s natürlich vom Modell abhängig und sollte so klein wie möglich und so groß wie nötig gewählt werden. Das Intervall $[0, t_s)$ bezeichnet man auch als *transiente Phase* der Simulation. Es gibt zahlreiche Heuristiken und statistische Tests zur Ermittlung von t_s. Kein Ansatz kann als wirklich befriedigend angesehen werden, wie auch in der Literatur bestätigt wird. Die meisten Simulationswerkzeuge beinhalten deshalb auch keine automatischen Methoden zur Ermittlung von t_s, es wir vielmehr vorausgesetzt, dass t_s vom Modellierer gesetzt wird. Dies ist natürlich kritisch und kann zu Verfälschungen der Simulationsresultate führen. Deshalb sollen einige Methoden zur Ermittlung von t_s aus der Beobachtung der Simulation vorgestellt werden.

t_s muss auf Grund der Beobachtung der zu messenden Leistungsgröße ermittelt werden. Sei $(y_1, \ldots, y_n)$ die zugehörige Stichprobe. Gesucht wird ein Index i, so dass die Werte $y_1, \ldots, y_{i-1}$ bei der Auswertung unberücksichtigt bleiben, da sie vor t_s erhoben wurden. Die folgenden beiden Regeln zur Bestimmung von i wurden von Conway 1963 und 1978 publiziert.

- Sei $y_k^+ = \max(y_k, \ldots, y_n)$ und $y_k^- = \min(y_k, \ldots, y_n)$, wähle $i = \min_k(y_k^- < y_k < y_k^+)$.
- Sei $y_k^+ = \max(y_1, \ldots, y_k)$ und $y_k^- = \min(y_1, \ldots, y_k)$, wähle $i = \min_k(y_k^- < y_k < y_k^+)$.

Der Vorteil der zweiten Variante ist, dass zur Laufzeit entschieden werden kann, ab wann Daten erhoben werden, während in der ersten Variante erst die gesamte Stichprobe vorliegen muss. Abbildung 2.59 zeigt ein Beispiel für die Anwendung der Methode. Beide Regeln funktionieren allerdings nicht, wenn sie direkt auf einzelne Messdaten angewendet werden. Die einzelnen Werte der

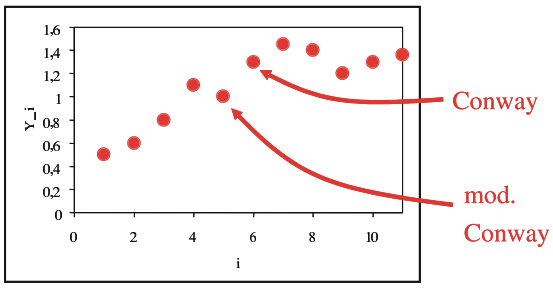

Abbildung 2.59: Beispiel für die Anwendung der ersten und modifizierten Methode von Conway.

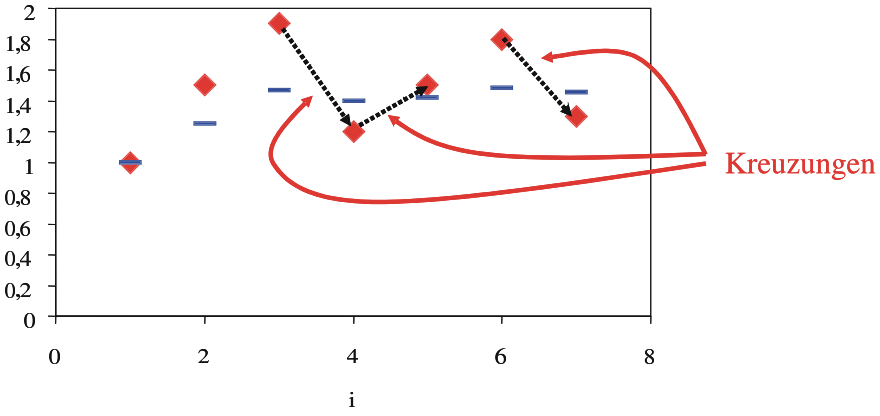

Abbildung 2.60: Beispiel für die Anwendung der "crossing the mean" Regel.

Stichprobe müssen vielmehr Mittelwerte mehrerer Messungen sein, ansonsten wird die Länge der transienten Phase gravierend unterschätzt. Die Zahl der Einzelwerte, aus denen Mittelwerte gebildet werden, hängt vom Verlauf der Beobachtungen ab, weshalb der Ansatz kaum automatisierbar ist.

Etwas einfacher einzusetzen ist die so genannte "crossing the mean" Regel. Der Mittelwert der ersten k Werte der Stichprobe ist gegeben durch

$$\bar{Y}_k = \frac{1}{k} \cdot \sum_{j=1}^{k} y_j$$

Der Wert von i wird so gewählt, dass $\bar{Y}_k$ $(k = 1, \ldots, i)$ von den Werten y_{k-1} und y_k genau K mal gekreuzt wird. Eine Kreuzung liegt dann vor, wenn $y_{k-1} < \bar{Y}_k < y_k$ oder $y_{k-1} > \bar{Y}_k > y_k$. K wird in der Regel zwischen 3 und 5 gewählt. Abbildung 2.60 zeigt ein Beispiel für die Anwendung der Regel.

Der letzte Ansatz benutzt die so genannte "batch means" Methode, die auch zur Ermittlung der Konfidenzintervalle für Resultate stationärer Simulationen eine große Rolle spielt. Die Stichprobe wird dazu in Gruppen (batches) der Größe i $(i = 5, 10, \ldots, n)$ unterteilt. Der Gruppenmittelwert lautet dann

$$\bar{Y}_j(i) = \frac{1}{i} \cdot \sum_{k=(j-1)\cdot i+1}^{i \cdot j} y_k$$

Wenn m Gruppen entstehen, so ergibt sich der Gesamtmittelwert aus

$$\bar{Y}(i) = \frac{1}{m} \cdot \sum_{k=1}^{m} \bar{Y}_k(i)$$

Mit wachsendem i (steigender batch-Größe), schwindet der Einfluss der ersten Beobachtungen. Falls die Sequenz $\bar{Y}_2(i), \ldots, \bar{Y}_m(i)$ keinen Trend mehr erkennen lässt, werden die ersten i Beobachtungen gelöscht und das Resultat aus den verbleibenden Werten berechnet. Auf Seite 115 wird ein Verfahren zur Auswertung stationärer Simulation vorgestellt, dass diese Art der Bestimmung von i und damit t_s einbezieht.

Weitere Ansätze und theoretische Resultate zur Ermittlung von t_s kann man in [4] finden.

Da t_s von s_0 abhängt, sollte s_0 natürlich möglichst so gewählt werden, dass $t_s$möglichst klein ist. So wird es bei einem hoch ausgelasteten System länger dauern bis der stationäre Zustand erreicht wird, wenn mit leeren Puffern gestartet wird, als wenn die Puffer direkt gefüllt werden. Bei einem System mit niedriger Auslastung ist es gerade umgekehrt. In der Literatur gibt es viele Heuristiken zur Wahl des Startzustandes. Üblicherweise reicht es aus, einen einfach implementierbaren und erreichbaren Zustand zu wählen. Dies wird oft ein System mit leeren Puffern und funktionsfähigen Maschinen sein.

Die Simulation wird durch das beschriebene Vorgehen in zwei Phasen unterteilt. In Phase 1, die das Intervall $[0, t_s)$ umfasst, werden keine Daten erhoben. In der zweiten Phase, die zum Zeitpunkt t_s beginnt und mit dem Ende der Simulation endet, werden Daten erhoben. Auf Grund der Annahmen kann davon ausgegangen werden, dass alle in der zweiten Phase erhobenen Werte aus der gleichen Verteilung stammen. Dies bedeutet natürlich nicht, dass die Daten unabhängig sind. Zur Erinnerung sei noch einmal erwähnt, dass zwei Zufallsvariablen X und Y unabhängig sind, wenn $P[X \leq x, Y \leq Y] = P[X \leq x] \cdot P[Y \leq y]$ gilt. In diesem Fall ist die Kovarianz $C(X, Y)$ gleich 0. Wie in (2.4) gezeigt ergibt sich die Varianz des Schätzer $\tilde{Y}$ zu

$$\sigma^2(\tilde{Y}) = \frac{1}{n^2} \cdot \sum_{i=1}^{n} \sum_{j=1}^{n} C(Y_i, Y_j) \quad \text{mit} \quad C(Y_i, Y_i) = \sigma^2(Y_i) \tag{2.8}$$

Die Varianz des Schätzer geht direkt in die Breite der Konfidenzintervalle ein (siehe (2.6)). Die bisherige Schätzung $\hat{S}^2$ (siehe (2.5)) berücksichtigt nur die Terme in der obigen Formel bei denen $i = j$ und führt daher zu erheblichen Verfälschungen, falls die restlichen Terme nicht fast 0 sind. Wenn die Kovarianz positiv ist, wäre die tatsächliche Varianz größer als der ermittelte Wert und die so berechneten Konfidenzintervalle wären zu optimistisch, d.h. die Abdeckung wäre u.U. deutlich kleiner als $1-\alpha$. Dies bedeutet, dass die Simulation eine nicht gerechtfertigte Sicherheit vortäuscht.

Da nicht terminierendes Verhalten untersucht wird, gibt es zwei Möglichkeiten der Datenerhebung

1. n Simulationsläufe mit je m Beobachtungen und
2. ein Simulationslauf mit $n \cdot m$ Beobachtungen

In beiden Varianten werden gleich viele Beobachtungen erzeugt. In der ersten Version treten allerdings n transiente Phasen auf, so dass n-mal Werte vor t_s verworfen werden. Bei der zweiten Variante werden nur einmal Werte verworfen. Damit liefert die zweite Variante mehr verwertbare Beobachtungen bei gleichem Aufwand. Betrachten wir nun die Unabhängigkeit der Beobachtungen.

Im ersten Fall sind Y_{ij} und Y_{kj} unabhängig, wenn $i \neq k$ und die Saaten des Zufallszahlengenerators entsprechend gewählt werden. Damit ergibt sich ein nahe liegendes Vorgehen:

- Führe n Replikationen durch, wobei die Länge m so gewählt wird, dass y_{im} nach t_s erhoben wird.
- Verwende y_{im} $(i = 1, \ldots, n)$ zur Berechnung von $\hat{Y}(n)$.

Der Wert von t_s kann mit einem der vorgestellten Ansätze oder mit einer Kombination der Ansätze ermittelt werden. Falls die batch means Methode verwendet wird, so kann y_{im} durch den Mittelwert der batches $2, \ldots, m$ ersetzt werden. Das Verfahren lässt sich sehr gut parallelisieren, da die einzelnen Simulationsläufe unabhängig voneinander ausgeführt werden können.

Der Nachteil des Vorgehens, das auch als unabhängige Replikationen bezeichnet wird, ist die mehrfache Simulation der transienten Phase. Dieser Punkt spricht für die Anwendung der zweiten Variante. Seien Y_j und Y_k die Zufallsvariablen, die die $j$te und $k$te Beobachtung beschreiben $(1 \leq j, k, \leq n \cdot m)$. Falls beide Beobachtungen nach t_s erhoben wurden, so können sie als identisch verteilt angesehen werden. Die Unabhängigkeit ist zumindest in manchen Fällen fragwürdig, wie man an den folgenden Beispielen sehen kann. Sei dazu $j < k$.

- Wenn Y_j die Wartezeit des $j$ten Kunden und Y_k die Wartezeit des $k$ten Kunden ist, so haben beide Kunden u.U. gleichzeitig gewartet, wenn die Differenz zwischen k und j nicht zu groß ist oder die Warteschlange sehr lang war. Offensichtlich sind beide Größen dann positiv korreliert.
- Wenn Y_j die Pufferbelegung zum Zeitpunkt $j \cdot \Delta$ ist und k nicht viel größer als j ist, so umfasst die Pufferbelegung Y_k zum Zeitpunkt $k \cdot \Delta$ zum Teil die selben Kunden. Auch in diesem Beispiel sind die Zufallsvariablen positiv korreliert.

Die Beispiele zeigen, dass oft positive Korrelationen auftreten werden und, wie oben erläutert, die Konfidenzintervalle zu schmal sein werden bzw. nicht die gewünschte Abdeckung haben. Eine Berücksichtigung der Kovarianz oder besser Autokorrelation, da die Werte Realisierungen eines Prozesses sind, erfordert zusätzliche Annahmen, die in der Praxis aber meistens erfüllt sind. So geht man davon aus, dass der Prozess schwach stationär ist. Dies bedeutet, dass

$$\sigma^2(Y_i) = \sigma^2(Y) = \sigma^2 \text{ und } \; C(Y_i, Y_j) = C_F(Y, |i.j|)$$

Die Varianz ist konstant und identisch für alle Werte und die Kovarianz hängt vom Abstand der Werte ab. Wir haben bis nicht definiert, was Abstand bedeutet. Den unterschiedlichen Interpretationen der Resultate entsprechend unterscheidet man zwischen individuenbezogenen Werten, wie der Verweilzeit eines Kunden im System, und zeitbezogenen Werten, wie der Pufferfüllung zu den Zeitpunkten t_i und t_j. Bei zeitbezogenen muss man in der Regel voraussetzen, dass der Abstand $|t_i - t_j| = |t_k - t_l|$ ist, falls $|i - j| = |k - l|$ ist. Auf dieser Basis kann man den Autokorrelationskoeffizienten der Ordnung s definieren. Es gilt

$$\rho(s) = \rho(|i - j|) = \frac{C_F(Y, |i - j|)}{\sigma^2(Y)} \tag{2.9}$$

wobei $\rho(0) = 1$ ist. Bei bekannten Autokorrelationskoeffizienten kann man die folgende Darstellung der Varianz des Schätzers $\tilde{Y}$ aus (2.8) und (2.9) herleiten.

$$\begin{aligned}
\sigma^2(\tilde{Y}) &= \tfrac{1}{n^2} \cdot \sum_{i=1}^{n} \sum_{j=1}^{n} C(Y_i, Y_j) \\
&= \tfrac{1}{n^2} \cdot \sum_{i=1}^{n} \sum_{j=1}^{n} \rho(|i - j|) \cdot \sigma^2(Y) \\
&= \tfrac{\sigma^2(Y)}{n} \cdot \left(1 + \tfrac{1}{n} \cdot \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} \rho(|i - j|) \right) \\
&= \tfrac{\sigma^2(Y)}{n} \cdot \left(1 + \tfrac{2}{n} \cdot \sum_{i=1}^{n} \sum_{j=1, j > i}^{n} \rho(|i - j|) \right) \\
&= \tfrac{\sigma^2(Y)}{n} \cdot \left(1 + \tfrac{2}{n} \cdot \sum_{s=1}^{n-1} (n - s) \cdot \rho(s) \right)
\end{aligned}$$

Wie bereits ausgeführt, kann man in den meisten Fällen davon ausgehen, dass $\rho(s) > 0$ gilt und damit die Summe positiv ist und die Varianz des Schätzers vergrößert wird. Es ist natürlich nicht realistisch, $\rho(s)$ und $\sigma^2(Y)$ als bekannt vorauszusetzen. Deshalb müssten die Werte von $\rho(s)$ auch

aus der Stichprobe geschätzt werden. Dies ist zwar im Prinzip möglich, führt aber in der Praxis zu Problemen, da die bekannten Schätzer wenig robust sind und insbesondere gegenseitig abhängig sind. Dies bedeutet, dass die Schätzungen für mehrere $\rho(s)$ voneinander abhängigen, wenn also der Wert von $\rho(1)$ unterschätzt wird, so wir mit großer Wahrscheinlichkeit auch der Wert von $\rho(2)$ unterschätzt, wenn die selben Daten verwendet werden. Man müsste also entweder für jedes s neue Daten verwenden, was den Aufwand zu stark erhöhen würde oder es müssten weitere Annahmen über die Struktur von $\rho(s)$ gemacht werden. Letzteres ist die Basis der Spektralmethode und auch der autoregressiven Modelle. Da beide Ansätze einige, zum Teil nicht einfach zu prüfenden, Voraussetzungen benötigen und nicht blind anwendbar sind, werden sie hier nicht weiterverfolgt.

Wir wollen hier einen Ansatz betrachten, der mit relativ wenigen Voraussetzungen auskommt und weitgehend automatisierbar ist. Wir nehmen dazu an, dass $|\rho(s)|$eine monoton fallende Funktion mit $\lim_{s\to\infty} \rho(s) = 0$. Also bei steigendem Abstand gibt es eine fallende Abhängigkeit von Beobachtungen. Dies Annahme ist plausibel, wie man an den folgenden Beispielen ablesen kann.

- Die Wartezeit des $k$ten Kunden wird nicht von der Wartezeit des $j$ten Kunden abhängen, wenn j lange vor k im System war.
- Die Population zum Zeitpunkt $k \cdot \Delta$ ist unabhängig von der Population zum Zeitpunkt $j \cdot \Delta$, wenn der Abstand $|j \cdot \Delta - k \cdot \Delta|$ genügend groß ist.

Wenn die Annahme stimmt, so gibt es einen Wert s_c, so dass $\rho(s) \approx 0$ für $s \geq s_c$. Ein nahe liegender Ansatz ist die batch means-Methode mit einer batch-Größe, die groß genug ist, um anzunehmen, dass die batch-Mittelwerte unkorreliert sind. Aus den batch-Mittelwerten kann dann ein Konfidenzintervall für den Mittelwert berechnet werden, wobei die Ansätze für unabhängige Daten angewendet werden.

Es wird nun ein Verfahren vorgestellt, um Konfidenzintervalle für $E(Y)$ aus den Beobachtungen eines einzelnen Simulationslaufs zu schätzen. Seien $(y_1, \ldots, y_m)$ die beobachteten Werte. Die ersten i_0Werte werden verworfen, da sie zur transienten Phase gehören. Der Wert von i_0 kann mit einer der beschriebenen Methoden geschätzt werden. Die restlichen Daten werden in batches der Größe i_1 unterteilt. Die Schätzung von i_1 ist ebenfalls schwierig, da die Größe so gewählt werden muss, dass die Mittelwerte der batches unkorreliert sind. Für gegebene m, i_0 und i_1 entstehen $(m-i_0)/i_1$ batches[2]. Batch k umfasst die Werte $i_0 + (k-1) \cdot i_i + 1$ bis $i_0 + k \cdot i_1$ und es gilt

$$\bar{Y}_k(i_1) = \frac{1}{i_1} \cdot \sum_{j=i_0+(k-1)\cdot i_1+1}^{k\cdot i_1} y_j$$

Ausgehend von einer gegebenen Beobachtungszahl m kann man die Anzahl der batches K wählen. Ein größeres K sorgt dafür, dass mehr batch-Mittelwerte zur endgültigen Auswertung vorliegen. Bei einem kleineren K werden mehr Werte in einem batch zusammengefasst, dadurch sinkt die Varianz der batch-Mittelwerte und die Verteilung der batch-Mittelwerte wird ähnlicher zu einer Normalverteilung. Letztere Argumente überwiegen in der Regel, deshalb wird K meist zwischen 10 und 30 gewählt.

Es bleibt damit noch die Frage, ob die batch Mittelwerte unkorelliert sind. Dies kann natürlich nur auf Basis der Stichprobenwerte untersucht werden. Üblicherweise wird dazu $\rho(1)$ geschätzt und falls dabei $\rho(1) \approx 0$ gilt, wird Unabhängigkeit angenommen. Der Schätzwert für den Autokorrelationskoeffizienten der Ordnung 1 wird wie folgte berechnet

$$\hat{\rho}(1) = \frac{\sum_{k=1}^{K-1} \left(\left(\bar{Y}_k - \hat{Y} \right) \cdot \left(\bar{Y}_{k+1} - \hat{Y} \right) \right)}{\sum_{k=1}^{K-1} \left(\bar{Y}_k - \hat{Y} \right)^2}$$

[2] Wir nehmen zur Vereinfachung an, dass die Werte so gewählt werden, dass der Wert ganzzahlig ist.

Damit kann die folgende heuristische batch means-Methode zur Bestimmung der Konfidenzintervalle für stationäre Größen aus einem Simulationslauf.

1. Ermittle während der Simulation den Wert von i_0.
2. Wähle ein initiales $m \geq 10 \cdot i_0$
3. Unterteile die Daten in K $(100 \leq K \leq 400)$ batches und bestimme während des Simulationslaufs deren Mittelwerte $\bar{Y}_k$ $(k = 1, \ldots, K)$
4. Bestimme den Schätzwert $\hat{\rho}(1)$ für den Autokorrelationskoeffizienten der Ordnung 1.
5. Test Autokorrelation
   (a) falls $\hat{\rho}(1) \leq 0.2$ reduziere die Anzahl batches auf L $(20 \leq L \leq 30)$ durch Zusammenfassen der bisherigen batch Mittelwerte und bestimme Konfidenzintervalle aus einer t-Verteilung mit $L - 1$ Freiheitsgraden.
   (b) ansonsten verdopple m und fahre bei 3. fort.

Der Ansatz lässt sich relativ einfach implementieren. Insbesondere wird nur ein Array der Größe K zur Datenspeicherung benötigt, wenn die Mittelwerte jeweils zur Laufzeit berechnet werden.

Schätzung mehrerer Leistungsmaße aus einem Simulationslauf

In den meisten Simulationsläufen wird mehr als ein Leistungsmaß ermittelt. Beispiel wären die Ermittlung der Verweilzeit an mehreren Stationen oder die Bestimmung von Durchsätzen und mittleren Pufferbelegungen. Wenn für alle Leistungsmaße Konfidenzintervalle ermittelt wurden, stellt sich die Frage, welche Aussage man über die Qualität aller Resultate machen kann.

Wir nehmen an, dass k Leistungsmaße ermittelt werden sollen und α_i die Signifikanzwahrscheinlichkeit des $i$ten Leistungsmaßes ist. Wenn die Schätzer für die einzelnen Leistungsmaße unabhängig sind, dann liegen mit Wahrscheinlichkeit

$$1 - \alpha = \prod_{i=1}^{k} (1 - \alpha_i)$$

alle Leistungsmaße innerhalb der ermittelten Konfidenzintervalle. Wenn alle α_i identisch sind, so kann man über die Wahrscheinlichkeiten einer Binomialverteilung Wahrscheinlichkeiten dafür ermitteln, dass l von k Resultatwerte innerhalb ihrer Konfidenzintervalle liegen.

Die Annahme unabhängiger Schätzer für mehrere Leistungsmaße ist aber in den meisten Fällen unrealistisch. So sind z.B. Verweilzeit und Warteschlangenlänge an einer Station stark miteinander korreliert, auch die Verweilzeit an aufeinander folgenden Warteschlangen ist oftmals korreliert. Wenn korreliert Schätzer vorliegen, so gibt es sehr wenige Resultate. Das allgemeinste Ergebnis ist die so genannte Bonferroni-Ungleichung, die eine untere Schranke für die Wahrscheinlichkeit liefert, dass alle Konfidenzintervalle den wahren Wert beinhalten.

$$1 - \alpha \geq 1 - \sum_{i=1}^{k} \alpha_i$$

Man sieht, dass für größere k schnell der triviale Fall $1-\alpha \geq 0$ erreicht wird. Für kleinere Werte von k, vorgegeben Breiten der Konfidenzintervalle ϵ_i und vorgegebene Signifikanzwahrscheinlichkeit α kann das folgende heuristische Vorgehen angewendet werden.

- Bestimme während der Simulation α_i, so dass alle Konfidenzintervallbreiten $\leq \epsilon_i$ sind.
- Falls $(1 - \sum_{i=1}^{k} \alpha_i) \leq 1 - \alpha$ beende die Simulation, ansonsten fahre mit der Simulation fort.

Die während der Simulation ermittelten α_i hängen von den vorgegebenen ϵ_i ab und unterscheiden sich für die verschiedenen Leistungsmaße. Der Ansatz funktioniert praktisch nur für kleine Werte von k, da der Aufwand auf Grund der konservativen Schätzung von $1 - \alpha$ sonst sehr groß wird.

Schätzung stationärer zyklischer Resultate

Das bisherige Vorgehen eignete sich für stationäre Prozesse, bei denen der Erwartungswert gegen einen festen Wert konvergiert. Wie in Abbildung 2.58 existiert auch stationär zyklisches Verhalten, welches mittels nicht-terminierender Simulation analysierbar ist. In diesem Fall gibt es eine Zykluszeit Δ_C, so dass $F_{Y_i}(y) = F_{Y_{i+\Delta_C}}(y)$ bzw. $F_{Y_t}(y) = F_{Y_{t+\Delta_C}}(y)$ ($i \in \mathbb{N}$, $t \in \mathbb{R}_{\geq 0}$). In manchen Fällen ist Δ_C aus der Modellbeschreibung bekannt. So ist in allen Modellen in denen ein Zeitablauf über Tage, Wochen etc. beschrieben wird, der Zeitablauf aus der Simulation bekannt. In anderen Fällen muss Δ_C erst geschätzt werden. Dies kann dadurch geschehen, dass batches unterschiedlicher Größe gebildet werden und die batch Mittelwerte auf Gleichheit getestet werden. Falls die batch Größe Δ_C entspricht, so sind die batch Mittelwerte identisch.

Nachdem Δ_C bekannt ist, können Werte $E(Y_i)$ bzw. $E(Y_t)$ mit $i, t \in [0, \Delta_C)$ geschätzt werden. Dazu werden die Beobachtungen $y_{i+k\cdot\Delta_C}$ bzw. $y_{t+k\cdot\Delta_C}$ ($k = 0, 1, 2 \ldots$) verwendet und die Methoden der stationären Analyse benutzt. Falls mehrere Werte aus einer Simulation zu ermitteln sind, ergeben sich die schon angesprochenen Probleme der Ermittlung simultaner Konfidenzintervalle.

2.6 Simulationssoftware

Der Abschnitt über Simulationssoftware wird in dieser Version der Notizen nicht weiter ausgeführt, da es sich im Wesentlichen um eine informelle Vorstellung unterschiedlicher Ansätze handelt, die am Besten in der Originalliteratur über die entsprechenden Softwaresysteme nachgelesen werden kann. Für eine Übersicht über Simulationssoftware sei auf die Folienkopien zur Vorlesung verwiesen. Weitere Ausführungen findet man auch in [5, Kap. 4] und [11, Kap. 3]

2.7 Möglichkeiten und Grenzen der Simulation

Wir haben Simulation als eine Methode kennen gelernt, um Experimente an realen Systemen durch Experimente an einem Modell zu ersetzen. Es wurde deutlich, dass die Statistik eine zentrale Rolle bei der Modellbildung und Modellauswertung spielt und dass dadurch viele Aspekte nicht beweisbar sondern nur testbar sind. Damit ist natürlich die Möglichkeit von Modellierungsfehlern und Fehlinterpretationen gegeben. Man sollte sich deshalb der Möglichkeiten und Grenzen der Simulation bewusst sein, wenn man diese einsetzt. In diesem Abschnitt werden einige generelle Aspekte untersucht, während der folgende Abschnitt sich Methoden zur Bewertung der Realitätsnähe von Simulationsmodellen widmet.

Die Vorteile der Simulation als Instrument der Systemanalyse liegen klar auf der Hand.

- Viele reale Systeme können mit analytischen Modellen nicht genau genug beschrieben werden, während in einem Simulationsmodell die realen Abläufe sehr gut nachgebildet werden können.
- Simulation bietet die Möglichkeit Systeme in ganz unterschiedlichen Umgebungen und unter unterschiedlichen Bedingungen zu analysieren. Diese Bedingungen können aus der realen Umwelt stammen, können aber auch vollkommen irreal sein.
- Modifikationen am System sind oft durch einfache Änderungen am Simulator modellierbar.
- Experimente können prinzipiell relativ einfach und kostengünstig durchgeführt werden. Da in der Simulation praktisch alle Faktoren beeinflussbar sind, lassen sich Experimente unter quasi beliebigen Bedingungen durchführen.
- Simulation erlaubt es Systeme auch in sehr kurzen oder sehr langen Zeitintervallen zu beobachten, wenn der Simulator die Zeit entsprechend streckt oder staucht.
- Mit heutigen Rechnerkapazitäten lassen sich viele und aufwändige Simulationsexperimente mit den vorhandenen Ressourcen durchführen.

Diese Vorteile der Simulation verdecken oft den Blick auf die leider auch vorhandenen Nachteile des Vorgehens.

- Die meisten Simulationsmodelle sind stochastisch, so dass Resultate nur geschätzt werden können. Die dazu verwendeten Methoden basieren fast immer auf zwar plausiblen aber nicht beweisbaren Annahmen, so dass für einzelne Modelle falsche Resultate ermittelt werden.
- Simulationsmodelle haben einen hohen Datenbedarf, der bei vielen realen Experimenten nicht befriedigt wird. So kann ein Modell nicht besser als der schwächste Punkt sein. In der Simulation bedeutet dies oft, dass zwar detaillierte Modelle erstellt werden, die benötigten Parameter aber nur auf Grund von sehr wenigen Beobachtungen geschätzt werden und in manchen Fällen nur auf Grund von nicht validierten Annahmen gesetzt werden.
- Simulationsmodelle sind aufwändig und teuer in der Entwicklung,
  - neben den üblichen Problemen bei der Erstellung großer Programme
  - treten simulationsspezifische Probleme, wie die Datenmodellierung auf.
- Simulationsmodelle liefern in manchen Fällen extreme Datenmengen, die oft nicht detailliert genug analysiert werden, so dass Resultate oft überinterpretiert oder falsch interpretiert werden.

Grundsätzlich lassen sich Fehlinterpretationen von Simulationsresultaten nicht ausschließen. Es gibt allerdings zahlreiche Fehler, die sich bei einem systematischen Vorgehen der Modellerstellung und Modellauswertung vermieden lasen. Einige der üblichen Fehler sind im Folgenden aufgezählt.

- Die Modellerstellung wird ohne konkrete Zielsetzung durchgeführt, oft verbunden mit einer späteren Nutzung des Modells für unterschiedliche und heterogene Ziele.
- Ein falscher Detaillierungsgrad des Modells wird gewählt. Oft werden unnötige Details abgebildet und relevante Aspekte vergessen.
- eine Unterschätzung des Aufwandes für Datenerhebung und Validierung in Modellierungsprojekten. Oft werden diese Schritte erst am Ende er Simulationsstudie angegangen und dann aus Zeitmangel sehr kurz abgehandelt. So werden dann Mittelwerte statt Verteilungen werden für Parameter eingesetzt und die Resultate werden nicht validiert. Zum Teil wird nicht einmal eine Plausibilitätsprüfung durchgeführt.
- Die Animation, die Teil vieler moderner Simulatoren ist, wird oft überbewertet. So zeigt eine Animation zwar die Abläufe, sie ist aber keine Garantie für eine korrekte Modellierung.
- Statistische Methoden werden oftmals nicht oder nur in sehr geringen Umfang eingesetzt. Dadurch werden Eingabedaten falsch modelliert und Simulationsresultate nicht ausgewertet. In vielen Fällen werden Simulationsresultate nur bzgl. der ermittelten Mittelwerte interpretiert, ohne dass die Qualität der Mittelwertschätzer untersucht wird.
- Es wird keine Sensitivitätsanalyse durchgeführt. D.h. es wird nicht untersucht, wie sich die Ergebnisse bei Änderung einzelner Parameter ändern. Dieser Schritt ist aber immens wichtig, um das Systemverhalten zu verstehen und um die Plausibiltät der ermittelten Resultate nachzuweisen.
- Der Gültigkeitsbereich der Modelle wird oft nicht abgeschätzt. So mag ein Modell für den Istzustand korrekte Ergebnisse liefern, es ist aber nicht klar, dass bei Änderungen der Systemparameter (z.B. der Ankunftsraten, Puffergrößen etc.) das Modell das System immer noch genau genug abbildet.

Man sieht, dass Simulationen und Simulationsergebnisse mit einer gewissen Vorsicht zu interpretieren sind. Simulation ist nicht das Allheilmittel für alle Probleme der Systemanalyse, kann aber, wenn sie richtig eingesetzt wird, wichtige Ergebnisse liefern. Eine zentrale Aussage ist, dass Simulation dann eingesetzt werden sollte, wenn keine analytische Analysemethode verfügbar ist. Dies bedeutet, dass immer zuerst untersucht werden sollte, ob das Problem mit Hilfe analytischer Ansätze lösbar ist.

2.8 Validierung von Modellen

Zur Erinnerung sei noch einmal erwähnt, dass Modelle erstellt werden, um Experimente an realen System durch Experimente am Modell zu ersetzen. Auf Basis der Resultate des Modells sollen Entscheidungen getroffen werden, die das System betreffen. Dies bedeutet, dass die Folgerungen aus der Modellanalyse, im hier betrachten Fall aus dem Simulationsexperimenten, weitestgehend den Folgerungen entsprechen sollen, die aus entsprechenden Objekt-Analysen des Systems gewonnen würden. In der Praxis sind die dazu notwendigen Objekt-Experimente unter Umständen gar nicht möglich, da das zu untersuchende System nicht existiert. Dies bedeutet, dass in irgend einer Weise plausibel gemacht werden muss, dass Modell- und System-Experiment zu gleichen oder zumindest ähnlichen Folgerungen führen. Dieser Abschnitt beschäftigt sich damit, welche Anforderungen an Modellexperimente gestellt werden müssen, wie Modelle so verändert werden können, dass sie die benötigten Resultate liefern und welche mathematischen Methoden existieren, um Verhaltensunterschiede zwischen Modell und Realsystem zu bewerten.

Experimente werden unter bestimmten Bedingungen durchgeführt, d.h. die kontrollierbaren Größen C werden auf bestimmte Werte eingestellt und die unkontrollierbaren Größen U nehmen bestimmte Werte an. Wir sprechen in diesem Zusammenhang von einer Experimentsituation oder einer Situation. Wenn das Modell und das reale System in einer Situation beobachtet werden so führen sie zu gleiche Folgerungen, wenn die beobachteten Resultate identisch sind. Ist eine solchen Identität der Resultate zu erwarten oder überhaupt erreichbar? Es gibt zwei Gründe, die dagegen sprechen. Zum einen sind das reale System und das Modell nicht identisch, das Modell beschreibt nur eine Abbildung des Systems unter einem vorgegebenen Analyseziel. Zum anderen zeigen Modell und System in den meisten Fällen stochastisches Verhalten, so dass auch eine mehrmalige Beobachtung des Systems oder des Modells unter gleichen Bedingungen, beim Modell aber mit variierender Saat des Zufallszahlengenerators, unterschiedliche Ergebnisse beobachtet werden. Damit ist die zentrale Frage, welche Verhaltensunterschiede tolerierbar sind.

Grundlagen des Vergleichs von System und Modell

Um System und Modell zu vergleichen ist eine vernünftige Definition von *Realitätstreue* oder *Gültigkeit* (engl. validity) notwendig. Sei dazu V_R ein Verhaltensmaß für das Realsystem und V_S das Verhalten für das simulierte Modell. $D(V_R, V_S)$ ist das benötigte Maß für die auftretenden Verhaltensunterschiede. Realitätstreue soll bejaht werden, falls $D(V_R, V_S)$ unter einem problemabhängig und zielabhängig festzulegendem Grenzwert liegt. Dieser Grenzwert ist idealerweise so zu wählen, dass unvermeidlich existierende Verhaltensunterschiede, die die zu treffende Entscheidung nicht beeinflussen, toleriert werden.

Bevor wir ein Maß für Verhaltensunterschiede festlegen, sollen die Ursachen für Verhaltensunterschiede betrachtet werden. Zwischen Modell und System gibt es strukturelle Unterschiede, da es sich bei dem Einen um ein Computerprogramm und beim Anderen am einen Ausschnitt aus der realen Umwelt handelt. Der Prozess der Modellierung führt zu beabsichtigten Abstraktionen und Aggregierungen der Realität. Dazu kommen unbeabsichtigte Ungenauigkeiten und sogar Fehler. Eine weitere Quelle für Abweichungen sind die Parameterwerte, die oftmals durch Zufallsvariablen angeglichen, nicht aber exakt wiedergegeben werden. Schließlich sind noch die stochastischen Schwankungen des Verhaltens zu beachten, die schon bei einem System/Modell zu variierenden Beobachtungen führen.

Das Vorgehen bei der Modellbildung soll noch einmal rekapituliert werden. Wie in Abbildung

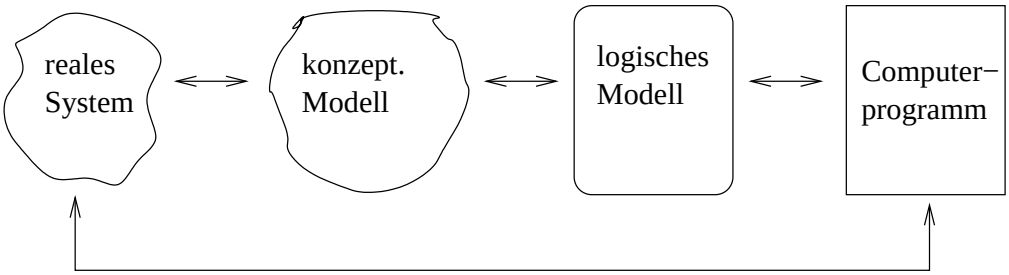

Abbildung 2.61: Transformationsschritte im Rahmen der Modellbildung.

2.61 gezeigt, erfolgt die Modellbildung in mehreren Schritten. Ausgehend von einem schlecht strukturiertem Realsystem und einer in den meisten Fällen vage definierten Problemstellung soll wohl definiertes Computerprogramm entstehen. Dieser Übergang erfolgt in mehreren Schritten, bei jedem dieser Schritte kann es zu Fehlern, Irrtümern oder Verzerrungen kommen, die dazu führen, dass Realsystem und Modell bzgl. der zu messenden Resultatgrößen abweichen. Wenn man sich die in Abbildung 2.61 gezeigten Schritte genauer anschaut, so werden beim Übergang vom realen System zum konzeptuellen Modell die Systemgrenzen und die relevanten Systemelemente festgelegt. Bleiben dabei wichtige Aspekte unberücksichtigt, so wird das Simulationsmodell keine Aussagen über das System erlauben. Beim Übergang vom konzeptuellen Modell zum logischen Modell werden die relevanten Zusammenhänge zwischen Systembestandteilen festgelegt und die Umwelteinflüsse auf das System definiert. Auch in diesem Schritt existiert eine Vielzahl von Möglichkeiten, relevante Zusammenhänge ungenau oder falsch darzustellen oder wegzulassen. Der nachfolgende Schritt des Übergangs vom logischen Modell zu Computerprogramm wird heute durch eine Vielzahl von Methoden und Werkzeugen unterstützt, birgt aber trotzdem eine Reihe von Fehlermöglichkeiten. So ist selbst bei einer vollständigen formalen Spezifikation der Nachweis, dass das resultierende Programm die Spezifikation implementiert alles andere als trivial und durchaus Gegenstand aktueller Forschungsarbeiten. Aus den Resultaten des Simulationsprogramms müssen anschließend Rückschlüsse auf das Systemverhalten gezogen werden und es müssen mögliche Änderungen am System hergeleitet werden. Auch an dieser Stelle können vielfältige Fehler auftreten, indem Resultate falsch interpretiert werden und zu falschen Schlussfolgerungen genutzt werden.

Die Terminologie im Kontext des Vergleichs und der Anpassung von System- und Modellverhalten ist oft uneinheitlich. Wir wollen die folgende Begriffsbildung verwenden.

- **Verifikation**: Unter Verifikation soll die Bestätigung aller Modell-Eingabegrößen und Modell-Annahmen inklusive struktureller Annahmen, der Programmverifikation und der Bestätigung der Eingabegrößen und Parameter verstanden werden. Teilweise, wie bei der Programmverifikation, können dazu formale Methoden eingesetzt werden, die zu beweisbaren Resultaten führen. Bei der Bestätigung von Eingabegrößen und Parametern handelt es sich formal um ein Problem der Statistik, so dass keine Beweise sondern nur Wahrscheinlichkeitsaussagen möglich sind. Auch für die Nutzung stochastischer Methoden zum Nachweis der Gültigkeit von Annahmen soll der Begriff Verifikation verwendet werden, auch wenn dies in der Literatur nicht unumstritten ist.

- **Validierung**: Der Begriff Validierung soll für die Bestätigung der Modellresultate verwendet werden. Es ist nachzuweisen oder besser plausibel zu machen, dass die Resultate des Modells nicht zu stark von den Resultaten des Systems abweichen.

- **Kalibrierung**: Ziel der Kalibrierung ist die Reduktion der Verhaltensunterschiede $D(V_R, V_S)$. Falls diese Unterschiede als zu groß bewertet werden, umfasst die Kalibrierung Ansätze zur Anpassung des Modells an das System. Dies geschieht durch gezielte Änderungen am Modell.

Auch wenn davon ausgegangen wird, dass große Mühe auf eine gute Wahl der Eingabegrößen gelegt wurde, steigt zwar die Hoffnung auf ein gültiges Modell, es gibt keine Garantie für ein

gültiges Modell. Ein erster Schritt ist nahe liegender Weise der Vergleich von System und Modell für eine Situation, falls dies möglich ist. Stimmen die ermittelten Ergebnisse überein, so könnte bei einer positivistischen Sichtweise davon ausgegangen werden, dass das realisierte Modell das System genügend genau abbildet. Dies bedeutet andererseits, dass man durch gezielte Änderungen das Modell so anpassen muss, dass Modellresultate und verfügbare Resultate aus dem System übereinstimmen. Ein solches Vorgehen nutzt die Kalibrierung, ohne Verifikation und Validierung einzusetzen und ist potenziell sehr gefährlich. Man muss sich vor Augen führen, dass Modelle genutzt werden, um Aussagen über Systeme in solchen Situationen zu machen, in denen das System nicht analysiert werden kann oder soll. Die Situationen zu denen das Systemverhalten bekamt ist, sind eigentlich uninteressant. Identische oder ähnliche Resultate über bekannte Situationen sind natürlich kein Beweis für ähnliche Resultate in unbekannten Situationen.

Der Ansatz der Kalibrierung ohne Validierung wird leider oft in der induktiven Modellierung verwendet. So wird ein mathematisches Modell an Daten aus der Vergangenheit angepasst und damit die Zukunft vorhergesagt. Dies führt oft zu abstrusen Aussagen, insbesondere wenn über einen kurzen Zeitraum in der Vergangenheit exponentielles Wachstum beobachtet wurde. Beispiele für solche Modelle sind insbesondere aber nicht nur in der Ökonomie zu finden. Man erinnere sich dazu an manche Prognosen über die Entwicklung des Aktienmarktes zu Zeiten des Internet-Booms.

Es soll in den nachfolgenden Abschnitten ein fundiertes Vorgehen beschrieben werden, welches Kalibrierung mit Validierung und Verifikation vereint und so zu belastbaren Aussagen kommt. Auf Grund der Problemstellung wird sich aber zeigen, dass die Qualität eines Modells nur sehr eingeschränkt beweisbar ist und deshalb in den meisten Fällen nur plausibel gemacht werden kann. Grundsätzlich gibt es einen Kosten-Nutzen Effekt. Man kann durch zusätzlichen Aufwand die Plausibilität eines Modell und das Vertrauen in ein Modell steigern, muss dazu aber zusätzlichen Aufwand in Kauf nehmen. Da dieser Prozess grundsätzlich nicht endet, muss ein Kompromiss zwischen Aufwand und Qualität der Abbildung gefunden werden. Wo dieser Kompromiss liegt hängt vom vertretbaren Aufwand und natürlich von der Zielstellung der Modellierung ab. Es stellt sich dabei primär die Frage, welche Folgen falsche Entscheidungen auf Grund der Modellresultate in der Praxis haben.

Verifikation in der Modellbildung von Simulationsmodellen

Für die Verifikation werden oft formale und automatisierbare Techniken eingesetzt. Man unterschiedet zwischen simulationsspezifischen und allgemeinen Schritten. Die allgemeinen Schritte umfassen primär Ansätze der Programmverifikation, wie sie bei jedem größeren Software-Projekt notwendig sind. Ein wichtiger Aspekt ist die Verifikation des Simulationsprogramms am logischen oder konzeptuellen Modell. Dies ist eine Frage der Programmverifikation, die wir hier nicht behandeln wollen, da sie Inhalt anderer Vorlesungen ist. Weitere Aspekte der Verifikation des Simulationsprogramms sind

- eine strukturierte Programmierung mit dem Test/Debugging von Modulen und Subprogrammen,
- eine Code-review durch andere Mitarbeiter,
- die Verwendung von (semi-)automatischen Spezifikationstechniken und
- ein inkrementeller Entwurf und Test.

Alle diese Punkte gehören zu einem strukturiertem Vorgehen im Software Engineering und sind nicht Inhalt dieser Vorlesung.

Es gibt darüber hinaus ein Reihe simulationsspezifischer Ansätze zur Verifikation von Simulationsprogrammen. Der Simulator sollte unter unterschiedlichen aber realistischen Umgebungsparametern (Situationen) getestet werden. Mögliche Umgebungsparameter ergeben sich aus der Beobachtung der Realität und der Zielstellung der Simulationsexperimente. Ein wichtiger Aspekt ist das Erstellen und Analysieren von Traces. Die meisten Simulatoren bieten die Möglichkeit

detaillierte Traces zu erzeugen. Diese können auf unterschiedliche Weise analysiert und visualisiert werden. Eine Möglichkeit der Visualisierung ist die Animation im Modell. Wenn das Modell graphisch spezifiziert wurde, so können in der Graphik die Abläufe angezeigt und vom Benutzer bewertet werden. Auf diese Weise lässt sich manchmal Fehlverhalten erkennen. Gleichzeitig stellen Traces eine Verbindung zwischen Simulation und Realität dar. Im Prinzip können Traces auch in der Realität gemessen werden und dann mit Traces der Simulation verglichen werden oder sogar als Eingabe der Simulation verwendet werden (trace-getriebene Simulation). Mit Hilfe von Beobachtungen und auch formalen Testverfahren lassen sich so simulierte und reale Abläufe vergleichen. Die Bewertung des Simulationsverhaltens wird ebenfalls einfacher, wenn man bestimmte Abläufe vorgibt oder Modelle vereinfacht, indem z.B. die Zufallszahlen durch deterministische Größen ersetzt werden.

Neben diesen Untersuchungen am Modell, reduziert die Verwendung von etablierten und zuverlässigen Simulationswerkzeugen die Fehlermöglichkeiten. Für viele Simulationswerkzeuge existieren bereits vordefinierte Modelle von Standardsystemen, die als Teilmodelle komplexer Modell genutzt werden können. Da diese Modelle oft schon vielfach verwendet und getestet wurden, sind sie weniger fehlerbehaftet als neu geschriebene Modelle. Beispiele für solche Standardmodelle findet man im Bereich der Rechnernetzsimulation, wo für viele Simulatoren bereits Simulationsmodelle für die üblichen Protokolle existieren. Als Beispiel sei auf die frei verfügbaren Simulatoren ns-2 [1] und OMNet++ [2] verwiesen. Ähnliche Standardmodell gibt es auch in anderen Bereichen.

Als letzter Punkt im Kontext der Verifikation des Simulationsmodells soll kurz die Verifikation von Eingabegrößen mit statistischen Testverfahren erwähnt werden. Das dazu notwendige Vorgehen wurde in Abschnitt 2.4 detailliert vorgestellt.

Die meisten der vorgestellten Ansätze zur Verifikation führen nicht zu Beweisen. Deshalb muss der Begriff Verifikation, der manchmal auf den Beweis von Eigenschaften beschränkt wird, weiter gefasst werden, indem vom Beweis zum Nachweis übergegangen wird.

Allgemeine Überlegungen zur Validierung

Ein Simulationsmodell soll nützlich sein, d.h. mit sinnvoller Genauigkeit Aussagen über das modellierte System erlauben. Eine wichtige Überlegung ist, dass der Begriff *Validität* eines Modells nicht absolut definierbar ist. Validierung ist immer Modell-individuell. Je nach Anforderung und Zielsetzung der Modellierung ist Validität sehr unterschiedlich festzulegen. So sind an ein Modell, mit dem sicherheitskritische Aspekte eines technischen Systems untersucht werden deutlich höhere Ansprüche zu stellen als an ein Modell, das die Konfiguration eines Rechnernetzes unter Kapazitätsplanungsgesichtspunkten bewerten soll. Da das Modell für die Systemanalyse verwendet wird oder nicht verwendbar ist, muss eine binäre Entscheidung im Validierungsprozess getroffen werden. Trotzdem ist Validierung graduell, Modelle sind mehr oder weniger valide und es ist zu entscheiden, ab welcher Grenze das Modell akzeptiert wird. Da Validität nicht beweisbar ist, ist sie oft das Ergebnis eines Verhandlungsprozesses verschiedener Partner. So sind in umfangreicheren Simulationsprojekten neben den eigentlichen Modelliern auch die Entwerfer und Bertreiber des modellierten Systems und das Management involviert. Zwischen allen beteiligten Personen muss Einvernehmen hergestellt werden, dass das Modell hinreichend genaue Aussagen über das System erlaubt. Dies setzt voraus, dass Validierung als projektbegleitender Prozess betrieben wird.

Man unterscheidet die folgenden Ansätze der Validierung.

- Die *funktionsbezogene Validierung* untersucht die Plausibiltät des Systems. Es werden dabei oft qualitative Aussagen verglichen und untersucht, wie sich das Modell in Extremsituationen und für einfach nachvollziehbare Abläufe verhält.

- Die *theoriebezogene Validierung* vergleicht die Übereinstimmung des Simulationsmodells mit einem analytischen Modell. In der Regel wird das analytische Modell das System nur für eingeschränkte Situationen nachbilden. Es erlaubt aber für diese Situationen einen einfachen Resultatvergleich und kann auch eingesetzt werden, wenn das System nicht existiert und deshalb auch keine Daten über das Systemverhalten erhoben werden können.

- Die *ergebnisbezogene Validierung* vergleicht die Ergebnisse des Simulationsmodells mit denen des realen Systems und stellt fest, welches Abweichungen tolerierbar sind.

Im Folgenden wird primär die ergebnisbezogene Kalibrierung und Validierung untersucht. Die Kalibrierung und Validierung setzt dann voraus, dass Daten über das Realsystem vorhanden sind oder erhoben werden können, damit V_R bestimmt werden kann. Man unterscheidet den Fall, bei dem Daten im Realsystem erhoben werden können und den Fall, bei dem das Realsystem nicht existiert und Daten anderweitig beschafft werden müssen.

Falls das Realsystem verfügbar ist und Daten erhoben werden können oder auf erhobene Daten zurückgegriffen werden kann, so treten immer noch eine Reihe von Problemen auf. Ähnlich wie bei der Modellierung von Eingabedaten können die gemessenen Daten gestört sein, es können Daten in falscher Form vorliegen etc. Da die Probleme ähnlich zur Modellierung von Eingangsdaten sind, kann auf die in Abschnitt 2.4 vorgestellten Methoden zurückgegriffen werden. Falls das zu modellierende System nicht existiert, so müssen Daten aus ähnlichen Systemen, Schätzungen oder sogar aus anderen Modellen verwendet werden. Die dabei auftretenden Probleme unterscheiden sich nicht wesentlich von den Problemen bei der Datenerhebung im realen Systeme. Die verfügbaren Daten sind aber oftmals ungenauer und damit weniger repräsentativ für das zu modellierende System. Damit sind natürlich auch die Möglichkeiten der Validierung und Kalibrierung eingeschränkt.

Schritte zur Kalibrierung von Modellen

Ziel der Kalibrierung ist die Identifikation und Beseitigung von Verhaltensunterschieden zwischen realem System und Modell. Zur Beseitigung von Verhaltensunterschieden muss das Modell angepasst d.h. geändert werden. Es gibt zwei Klassen möglicher Änderungen.

- *Strukturänderungen* modifizieren die Struktur des Modells, indem der Programm-Code geändert wird und neue Aspekte hinzugenommen oder vorhandene Abläufe modifiziert werden.
- *Parameteränderungen* modifizieren Werte der einzelnen Modellparameter. Dazu sind keine Änderungen an der Struktur des Simulationsprogramms notwendig.

Parameteränderungen sind natürlich in den meisten Fällen viel einfacher auszuführen als Strukturänderungen. Bei der Modellierung komplexer Systeme erfordert die Kalibrierung aber in fast allen Fällen auch Strukturänderungen. Bevor Änderungen am Modell durchgeführt werden können, müssen erst die Ursachen für Verhaltensunterschiede abgeleitet werden. Strukturelle Ungenauigkeiten und Fehler sind primär aus dem Vergleich qualitativen Verhaltens etwa in Form von Traces oder Animationen ableitbar. Parameter Ungenauigkeiten können durch den Vergleich quantitativer Größen erkannt werden. In beiden Fällen ist es oft hilfreich den Bereich, in dem Probleme auftreten, möglichst frühzeitig einzugrenzen. Dazu sind detaillierte Informationen über das Verhalten notwendig. So gibt die Beobachtung der Abweichung der Antwortzeit eines komplexen Systems/Modells wenig Aufschluss über die Gründe der Abweichung. Wenn aber festgestellt wird, dass die Verweilzeiten an bestimmten Komponenten stark abweichen, so sind notwendige Änderungen viel einfacher zu identifizieren.

Bei stochastischen Modellen, wie wir sie fast ausschließlich betrachten, ergeben sich zwei spezifische Testprobleme. Bei Strukturänderungen entstehen mehrere Modelle $S_1, \ldots S_K$ und es ist auf Basis der zugehörigen Werte $D(V_R, V_{S_k})$ zu entscheiden, ob ein Modell besser als ein anderes ist. Da die zugehörigen D's Zufallsvariablen sind, ist festzustellen, ob die Unterschiede zwischen den Konfigurationen/Modellen das normale Schwankungsmaß überschreiten, d.h. ob die Unterschiede signifikant sind.

Bei Parameteränderungen steht das Tuning von Parametern im Mittelpunkt. Wenn $S(p)$ das Modell mit Parametervektor p ist, so wird $p_{opt} = \arg\min_p D(V_R, V_{S8(p)})$ gesucht. Dies beschreibt ein typisches stochastisches Optimierungsproblem.

Methodisch gesehen ist Kalibrieren eines stochastischen Simulators damit identisch zum Experimentieren mit einem stochastischen Simulator. Für ein gegebenes Realsystem R mit Verhalten V_R wird mit Hilfe von Experimenten ein Modell S mit Verhalten V_S gesucht, so dass $D(V_R, V_S)$

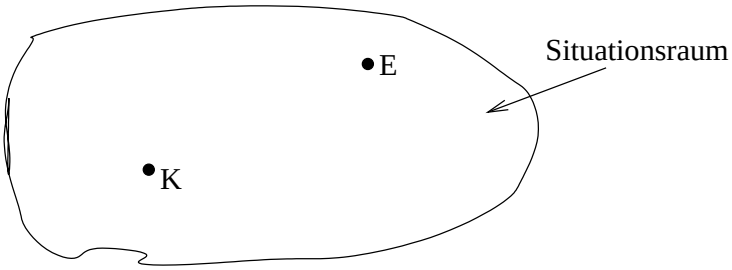

Abbildung 2.62: Situation nach der Kalibrierung.

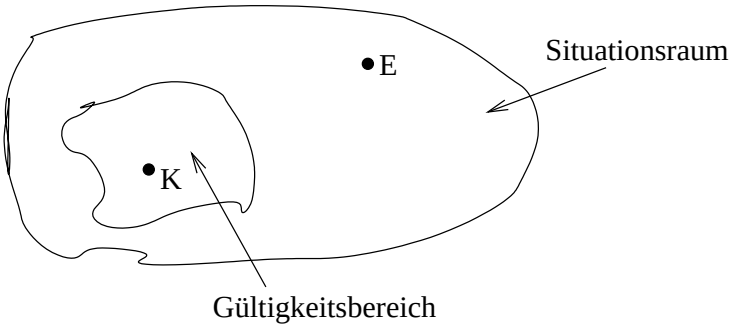

Abbildung 2.63: Kalibriersituation mit Gültigkeitsbereich.

kleiner als eine vorgegebene Schranke ist. Um zu S zu gelangen wird ein vorhandenes Modell durch Ausprobieren struktureller Ändeungen und das Tuning von Parametern verbessert. Methoden des Experimentierens, die wir hier nicht detailliert behandeln werden, sind hilfreich für das Finden einer guten Lösung. Für die detaillierte Methodik sei auf die Literatur verwiesen [11, Kap. 12][14].

Angenommen wir hätten unser primäres Ziel erreicht und ein Modell E gefunden, so dass $D(V_R, V_E) < D_{ertr}$ für die vordefinierte obere Schranke der akzeptablen Verhaltensunterschiede D_{ertr} gilt. Ist damit auch das Ziel erreicht, dass Folgerungen aus Objektexperimenten identisch sind zu Folgerungen aus Modell-Experimenten? Dies ist zumindest fraglich, wie aus folgenden Überlegungen deutlich wird. Wir haben über Änderungen am Modell Simulator-Verhalten und Modell-Verhalten angepasst, für einen Zustand der Umwelt und einen Zustand des Systems, also für eine Situation. Ziel der Simulation sind aber Aussagen über andere Situationen, für die das Systemverhalten unbekannt ist. Das prinzipielle Problem wird in Abbildung 2.62 verdeutlicht. Der Situationenraum wird in der Abbildung zweidimensional dargestellt und für einen Punkt K in diesem Situationenraum wurde die Nähe des Modellverhaltens nachgewiesen. Als Ersatz für das System soll das Modell aber an einem Punkt E eingesetzt werden. Ist der Unterschied zwischen Modell- und Systemverhalten an der Stelle E genauso oder zumindest ähnlich wie an der Stelle K? Dies ist zumindest fraglich. Anzunehmen ist eher, dass es einen Bereich G gibt in dem in dem $D(V_r, V_S) < D_{ertr}$ gilt und einen Bereich $\neg G$ in dem $D(V_R, V_S) \geq D_{ertr}$ gilt.

Die Frage reduziert sich damit zu der Frage, ob E innerhalb oder außerhalb von G liegt. Diese Frage ist prinzipiell nur beantwortbar, wenn vom Verhalten für gewisse Situationen auf Verhalten in anderen Situationen geschlossen werden kann. Dies würde gleichzeitig ermöglichen Aussagen über Systemverhalten ohne Experimente am Realsystem zu erlangen. Bei praktisch allen realen Systemen ist dies aus prinzipiellen Gründen unmöglich. Statt eines Beweises für die Gültigkeit eines Modells kann nur eine gewisse Zuversicht in die Gültigkeit erreicht werden. Dies ist natürlich ein deutlich schwächeres Resultat, aber letztendlich ein Problem jedes vorausschauenden Modellierens.

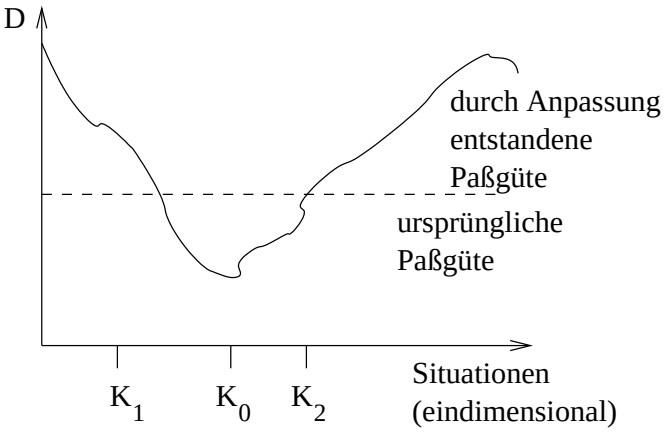

Abbildung 2.64: Das Problem der Überanpassung durch zusätzliche Kalibrierung.

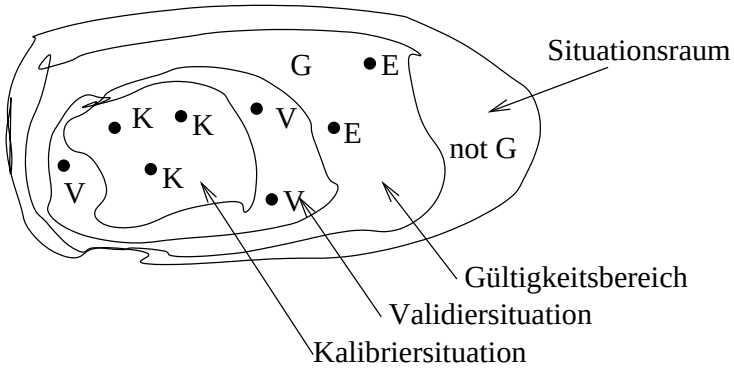

Abbildung 2.65: Kombinierte Kalibrierung und Validierung.

Trotzdem kann aus der Gültigkeit für eine Situation K durchaus etwas mehr an Information gewonnen werden. So ist anzunehmen, dass Situationen nahe bei K eher zu G gehören als Situationen, die sich deutlich unterschieden, da wir oft stetiges Verhalten in den Parametern beobachten. Dieses Phänomen ist in Abbildung 2.63 dargestellt. Es muss aber in der Praxis nicht so sein, dass der Gültigkeitsbereich sich als ein zusammenhängender Bereich darstellt.

Das bisherige Vorgehen ging davon aus, dass ein Modell S gefunden wird, welches Verhaltensabweichungen liefert, die unterhalb eines tolerierbaren Wertes liegen. Man könnte nun weiter den begonnen Weg voranschreiten und S durch zusätzliche Modifikationen so verbessern, dass $D(V_R, V_S)$ weiter reduziert wird, um so zu einem besseren Modell zu gelangen. In der Praxis zeigt sich allerdings oft, dass dieses Vorgehen nicht zu einer Vergrößerung von G führt. Im Gegenteil, man beobachtet bei diesem Vorgehen oft das Problem der Überanpassung. Verhaltensunterschiede werden für die Kalibriersituation reduziert, steigen aber gleichzeitig für andere Situationen. Gerade bei sehr detaillierten Modellen mit vielen Parametern lässt sich im Prinzip jedes Verhalten anpassen, dies sagt aber nichts über die Validität des Modells über einen größeren Bereich aus. Die Gefahr der Überanpassung kann dadurch reduziert werden, dass nicht nur an einer Stelle, sondern an mehreren Stellen kalibriert wird. Dies setzt natürlich voraus, dass entsprechende Daten vorhanden sind.

Dennoch bleibt zu zeigen, dass keine Überanpassung erfolgte. Das Vertrauen in das Modell kann nur erhöht werden, wenn die Gültigkeit für Situationen nachgewiesen wird, die nicht bereits zur Kalibrierung verwendet wurden. Diese Überprüfung bezeichnet man als Validierung. Es wird überprüft, ob die Verhaltensunterschiede für Validiersituationen so ähnlich ist, wie für Kalibriersituationen. Im Gegensatz zu einer Kalibriersituation wird das Ergebnis einer Validiersituation nur

bewertet, nicht aber zur Modellanpassung genutzt. Abbildung 2.65 skizziert das Vorgehen. Die verfügbaren Daten über das Systemverhalten werden unterteilt. Ein Teil wird zur Kalibrierung und der Rest zur Validierung verwendet.

Trotzdem bleibt ungewiss, ob eine unbekannte Experimentsituation E innerhalb oder außerhalb des Gültigkeitsbereichs G liegt. Dies kann nur retrospektivisch nachgewiesen werden, ist dann aber für Vorhersagen uninteressant. Trotzdem erhöht die Validierung, auch die retrospektivisch Validierung, das Vertrauen in ein Modell. Es bleibt aber immer fraglich, ob Experimentsituationen E, die weit weg von Kalibrier- und Valdiersituationen liegen durch das Modell genügend genau beschrieben werden. Gerade diese Situationen sind aber beim vorausschauenden Modellieren von besonderem Interesse.

Messung von Verhaltensunterschieden

Die Messung von Verhaltensunterschieden ist wesentlich bei der Kalibrierung zur Reduktion der Verhaltensunterschiede und bei der Validierung zur Bewertung der Verhaltensunterschiede. Kalibrierung und Validierung sollen das Vertrauen in das Modell erhöhen. Deshalb ist ein mathematisch fundierter Vergleich der Verhalten notwendig. Es stellen sich die folgenden Fragen:

1. Mit dem Verhalten welchen Systems soll das Verhalten des Simulators verglichen werden?
2. Welche Verhaltens-Aspekte sollen für den Vergleich gewählt werden?
3. Welche Vergleichs-Methoden und -Techniken sollen eingesetzt werden?

Frage 1 zielt darauf, das erwünschte Simulatorverhalten festzulegen. Ideal ist die Beobachtung des realen Objekt-Systems. Dies setzt voraus, dass das System in den gewünschten Situationen und bzgl. der gewünschten Größen beobachtbar ist oder Beobachtungen bereits existieren. Falls keine vorhandenen Aufzeichnungen genutzt werden können, muss das System in den gewünschten Situationen betreiben werden und das gewünschte Verhalten gemessen werden. In diesem Kontext ist die trace-getriebene Simulation zur Verifikation des Modells aber auch zur Validierung des Verhaltens hilfreich. Ein Trace wird im realen System gemessen und der Simulator mit den Trace-Daten betrieben. Das beobachtete Simulatorverhalten kann dann qualitativ bzgl. der auftretenden Abläufe und quantitativ bzgl. der ermittelten Resultate mit dem Realsystem verglichen werden. Die Einbringung von Trace-Daten in den Simulationsabläufen ist in vielen Simulatoren prinzipiell möglich, kann aber sehr aufwändig sein, da die verfügbaren Daten normalerweise erst interpretiert und gefiltert und anschließend an geeigneter Stelle in die Simulation eingebunden werden müssen. Dies erfordert meistens zusätzlichen Programmieraufwand im Simulator und passende Filter zur Filterung relevanter Information aus einem Trace.

Falls Daten über das Real-System nicht vorhanden sind und auch nicht erhoben werden können, so kann versucht werden auf Daten aus ähnlichen Systemen zurückzugreifen. Dieses Vorgehen erfordert eine gewisse Sorgfalt, da sicherzustellen ist, dass das ähnliche System sich auch ähnlich bzgl. relevanter Größen verhält.

Wenn keine Systemdaten erhältlich sind, so kann auf Daten aus anderen Modellen zurückgegriffen werden. Es wurde bereits erwähnt, dass für Marginalsituationen durchaus analytische Modelle zum Vergleich benutzt werden können. Ansonsten können Daten aus anderen Simulationsmodellen dieses oder ähnlicher Systeme verwendet werden. Es ist aber offensichtlich, dass die Gefahr von unzureichenden oder falschen Daten wächst und damit V_R nur sehr ungenau bestimmbar ist.

Alternative Methoden der Gewinnung von Daten über reale Systeme basieren auf der Nutzung von Expertenwissen. Es geht im Wesentlichen darum, ob ein Experte das Verhalten des Simulators vom Verhalten des (potenziellen) Realsystems unterschieden kann. Dieser Ansatz ist in der Praxis aber schwer umsetzbar und wird kaum eingesetzt.

Frage 2 beschäftigt sich mit der Festsetzung des zu vergleichenden Verhaltens. Da es das Ziel ist, das System auf Basis mehrerer ausgewählter Leistungskriterien zu bewerten, sollte diese auch zum Vergleich herangezogen werden. Bei mehreren Maßen treten dabei oft Zielkonflikte auf, d.h. die Verbesserung des Simulatorverhaltens bzgl. eines Maßes führt zur Verschlechterung bzgl. eines

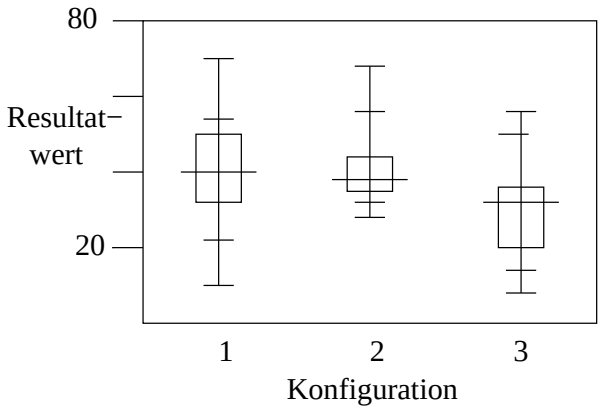

Abbildung 2.66: Vergleich von Box-Plots für drei Konfigurationen.

anderen Maßes. Das gewählte Vorgehen muss deshalb ähnlich zum Vorgehen bei einer multikriteriellen Optimierung sein. Entweder werden die Verhaltensunterschiede in den einzelnen Maßen auf einen skalaren Wert abgebildet oder das Ergebnis von $D(V_R, V_S)$ ist ein Vektor mit einem Element pro Leistungsmaß. Auch D_{ertr} wird dann als Vektor definiert und $D(V_R, V_S) < D_{ertr}$ muss komponentenweise gelten. Wir gehen im folgenden allerdings davon aus, dass Verhalten sich durch eine skalare Größe beschrieben lässt. Die verwendeten Verfahren lassen sich zwar prinzipiell auf mehrdimensionale Probleme erweitern, dies ist aber meist alles andere als trivial, da im mehrdimensionalen Fall die beobachteten Größen meistens stark korreliert sind.

Die Beantwortung der 3. Frage hängt davon ab, womit das Simulatorverhalten verglichen wird. Beim Vergleich mit einem analytischen Modell wird ein Stichprobe mit einer analytischen Verteilung verglichen und es ist zu entscheiden, ob die Stichprobe hinreichend ähnlich oder unähnlich ist. Dieses Problem trat aber bereits bei der Modellierung von Eingabedaten mit theoretischen Verteilungen in Abschnitt 2.4 auf. Wie dort beschrieben können der χ^2- oder K.-S.-Test zum Vergleich verwendet werden. Beim Vergleich des Simulatorverhalten mit dem Verhalten des Realsystems oder eines anderen Simulators sind zwei Stichproben zu vergleichen. Die dazu notwendigen Methoden werden nun vorgestellt.

Vergleich von Stichproben

Das Verhalten sei durch Zufallsvariablen V_R und V_S beschrieben. Die Zufallsvariable kann z.B. die Verweilzeit eines Kunden an einem Bankschalter oder die mittlere Pufferbelegung eines Routers in einem Rechnernetz beschreiben. Es liegen zwei Stichproben $v_R = (v_{R_1}, \ldots, , v_{R_n})$ und $v_S = (v_{S_1}, \ldots, v_{S_m})$ vor,die Realisierungen von V_R und V_S beschreiben. Zu entscheiden ist, ob beide Stichproben hinreichend ähnlich sind. Man unterschiedet Bewertungsverfahren in

- subjektive Verfahren, die auf dem Vergleich graphischer Repräsentationen der Stichproben basieren und durch einen Menschen bewertet werden
- objektive Methoden, die auf statistischen Test- oder Bewertungsverfahren basieren und ein Resultat aus den Stichproben ableiten.

Beim subjektiven Vorgehen, welches auch als Inspektionsansatz bezeichnet wird, werden die Werte der Stichproben visualisiert . Wie wir bereits kennen gelernt haben, gibt es unterschiedliche Visualisierungsmöglichkeiten. So kann die Dichtefunktion durch ein Histogramm dargestellt werden, die Verteilungsfunktion kann dargestellt werden, Q-Q- oder P-P-Plots können benutzt werden oder Box-Plots (siehe Abbildung 2.66) verglichen werden. Einige graphische Darstellungen haben frei Parameter, deren Festlegung zu unterschiedlichen Repräsentationen führen kann. Des weiteren muss entschieden werden, ob die ganze Stichprobe repräsentiert wird oder ob Ausreißer erst eliminiert werden. Die abschließende Entscheidung, ob die Stichproben ähnlich oder unähnlich sind hat

immer eine subjektive Komponente und sollte, wenn möglich, von mehreren Personen unabhängig getroffen werden.

Trotz der subjektiven Einflüsse ist die Inspektionsmethode ein wichtiges Entscheidungskriterium, da die gesamte Verteilung betrachtet werden kann. Sie sollte allerdings mit objektiven Verfahren ergänzt oder komplettiert werden.

Vergleich von Konfidenzintervallen

Es gibt zwei unterschiedliche objektive Ansätze, den Vergleich von Konfidenzintervallen und statistische Testverfahren. Der Vergleich von Konfidenzintervallen wird hier detailliert beschrieben, während auf Testverfahren nur kurz am Ende des Abschnitts eingegangen wird. Zum Vergleich der Konfidenzintervalle setzen wir voraus, dass die Beobachtungen in beiden Stichproben unabhängig identisch verteilt sind. Auf Basis von Konfidenzintervallen sollen die Mittelwerte beider Stichproben verglichen werden. Es wird ein Konfidenzintervall für die Differenz der Stichprobenmittelwerte bestimmt. Enthält dieses die 0 nicht, so können die Mittelwerte als unterschiedlich angesehen werden. Gleichzeitig gibt die Lage und Breite des Konfidenzintervalls Auskunft über die Unterschiede und es kann entschieden werden, ob diese tolerabel sind. Wir betrachten zwei Verfahren

- ungepaarte Stichproben (Welch-Verfahren [11, Kap. 10.2.2]),
- gepaarte Stichproben (paired t-Konfidenzintervalle [11, Kap. 10.2.1]).

Für das Welch-Verfahren müssen die Werte zwischen den beiden Stichproben unabhängig sein. Es werden aber keine Annahmen bzgl. der Varianz gemacht. Die Schätzer für Mittelwert und Varianz der beiden Stichproben werden wie üblich definiert.

$$\begin{aligned} \tilde{\mu}_R &= \tfrac{1}{n}\sum_{i=1}^{n} V_{R_i} \quad \tilde{S}_R^2 = \tfrac{1}{n-1}\left(\sum_{i=1}^{n}\left(v_{R_i}-\tilde{\mu}_R\right)^2\right) \\ \tilde{\mu}_S &= \tfrac{1}{m}\sum_{i=1}^{m} V_{S_i} \quad \tilde{S}_S^2 = \tfrac{1}{m-1}\left(\sum_{i=1}^{m}\left(v_{S_i}-\tilde{\mu}_S\right)^2\right) \end{aligned}$$

Uns interessiert die Differenz $\mu_{RS} = \mu_R - \mu_S$. Wenn wir ein Konfidenzintervalle für μ_{RS} ermittelt haben, so kann entschieden werden, ob die Stichproben als ähnlich oder unähnlich klassifiziert werden. Falls $m = n$ ist, so kann die Stichprobe der Differenz $v_{RS} = (v_{RS_1}, \ldots, v_{RS_n})$ mit $v_{RS_i} = v_{R_i} - v_{S_i}$ gebildet werden. Aus dieser Stichprobe kann der Mittelwert $\hat{\mu}_{RS}$ und die Varianz $\hat{S}_{RS}^2$ geschätzt werden., so dass

$$\hat{\mu}_{RS} \pm t_{n-1,1-\alpha/2} \cdot \sqrt{\frac{\hat{S}_{RS}^2}{n}}$$

ein Konfidenzintervall für den Mittelwert ist. Dieses Konfidenzintervall ist formal aber nur korrekt, wenn

- beide Stichproben gleiche Beobachtungszahlen beinhalten $n = m$ und
- die Varianzen von V_R und V_S identisch sind.

Gleiche Beobachtungszahlen sind zwar immer zu erreichen, indem Werte aus einer Stichprobe weggelassen werden. Oft ist es aber so, dass für das Realsystem Stichproben fester Länge vorliegen und die Erhebung zusätzlicher Daten aufwändig ist, während für der Simulation sehr einfach zusätzliche Daten generiert werden können. Die Beschränkung auf Stichproben gleicher Länge führt dann zu einem Informationsverlust. Die Forderung nach identischer Varianz ist natürlich nicht realistisch. Es zeigt sich allerdings, dass bei gleicher Stichprobenlänge und nicht zu stark differierenden Varianzen das berechnete Konfidenzintervall eine recht gute Abdeckung hat.

Für den allgemeinen Fall ungleicher Stichprobenumfänge und ungleicher Varianzen ist die Varianz des Schätzers für μ_{RS}nicht exakt bestimmbar die folgende Approximation von Welch [17] hat sich allerdings in vielen Experimenten als sehr gut erwiesen.

$$\hat{\mu}_{RS} = \hat{\mu}_R - \hat{\mu}_S \ , \qquad \hat{S}^2_{RS} = \frac{\hat{S}^2_R}{n} + \frac{\hat{S}^2_S}{m} \ , \qquad \hat{f} = \frac{\left(\hat{S}^2_{RS}\right)^2}{\frac{1}{n+1}\cdot\left(\frac{\hat{S}^2_R}{n}\right)^2 + \frac{1}{m-1}\cdot\left(\frac{\hat{S}^2_S}{m}\right)^2}$$

Damit wird das folgende Konfidenzintervall berechnet.

$$\hat{\mu}_{RS} \pm t_{\hat{f},1-\alpha/2} \cdot \hat{S}_{RS}$$

Da $\hat{f}$ in der Regel keine ganze Zahl sein wird, kann der Wert gerundet werden oder der Wert wird interpoliert als

$$t_{\hat{f},1-\alpha/2} = \left(\hat{f} - \left\lfloor \hat{f} \right\rfloor\right) \cdot t_{\lfloor \hat{f} \rfloor,1-\alpha/2} + \left(\left\lceil \hat{f} \right\rceil - \hat{f}\right) \cdot t_{\lceil \hat{f} \rceil,1-\alpha/2}$$

Das folgende Beispiel stammt aus [11, Kap. 10.2].

i	v_{R_i}	v_{S_i}	$v_{R_i} - v_{S_i}$
1	126.97	118.21	8.76
2	124.31	120.22	4.09
3	126.68	122.45	4.23
4	122.66	122.68	0.02
5	127.23	119.40	7.83

Die Werte für V_R sind in der Stichprobe größer als die von V_S. Die Frage ist, ob dieser Unterschied signifikant ist. Die folgenden Schätzwerte resultieren aus den Daten. $\hat{\mu}_R = 125.57$, $\hat{\mu}_S = 120.59$, $\hat{S}^2_R = 4.00$ und $\hat{S}^2_S = 3.76$, sowie $\hat{\mu}_{RS} = 4.98$, $\hat{S}^2_{RS} = 1.55$ und $\hat{f} = 7.99$. Dies liefert ein Konfidenzintervall von $[2.67, 7.29]$ zum Signifikanzniveau $\alpha = 0.1$ und von $[0.81, 9.15]$ zum Signifikanzniveau $\alpha = 0.01$. In beiden Fällen ist 0 nicht im Konfidenzintervall enthalten, so dass der Erwartungswert der beobachteten Größe im Realsystem als größer angenommen werden kann. Inwieweit der beobachtete Unterschied ausreicht, um das Modell als nicht ausreichend valide anzusehen hängt von den Anforderungen an das Ergebnis ab. Weiterhin muss man sich vor Augen führen, dass eine Schätzung auf Basis von 5 Werten eher unzuverlässig ist, da die Annahmen des zentralen Grenzwertsatzes bei so kleinen Beobachtungszahlen nicht gelten müssen.

Der zweite betrachtete Ansatz zur Konfidenzintervallberechnung basiert auf so genannten gepaarten Stichproben. Als zusätzlich Voraussetzung wird n=m benötigt. Es werden aber keine Annahmen bzgl. identischer Varianz von $V_R$und V_S oder Unabhängigkeit zwischen den Stichproben vorausgesetzt. Aus diesem Grund eignet sich der Ansatz auch für dir trace-getriebene Simulation, wenn v_R Beobachtungen aus einem Trace beschreibt und v_S Resultate einer Simulation beinhaltet, bei der einige Parameter des Traces mit verwendet wurden, so dass die Werte der Stichproben positiv korreliert sind. $\hat{\mu}_{RS}$ wird wie in der vorherigen Methoden berechnet und für die Varianz wird die folgende Formel benutzt.

$$\hat{S}^2_{RS} = \frac{\sum_{i=1}^{n} \left((v_{R_i} - v_{S_i}) - \hat{\mu}_{RS}\right)^2}{n \cdot (n-1)}$$

Damit kann das folgende Konfidenzintervall berechnet werden.

$$\hat{\mu}_{RS} \pm t_{n-1,1-\alpha/2} \cdot \hat{S}_{RS}$$

Aus dem vorherigen Beispiel resultiert $\hat{S}_{RS} = 1.56$ und Konfidenzintervalle $[1.66, 8.30]$ zum Signifkanzniveau $\alpha = 0.1$ und $[-2.20, 12.16]$ zum Signifikanzniveau $\alpha = 0.01$. Im letzteren Fall ist die 0 im Konfidenzintervall enthalten.

Wenn beide vorgestellten Ansätze anwendbar sind, so ist nicht klar, welche Methode schmalere Konfidenzintervalle liefert. Es sollte deshalb immer nach Datenlage über die Anwendung entschieden werden.

Neben dem Vergleich von Konfidenzintervallen für Mittelwertschätzern besteht noch die Möglichkeit Testverfahren anzuwenden. Im Gegensatz zu Konfidenzintervallen quantifizieren Test den Abstand der Stichproben aber nicht. Da Realsystem und Simulator fast nie vollständig identische Verhalten haben, neigen Testverfahren dazu bei größeren Stichprobenumfängen Gleichheit der Stichproben abzulehnen. Insofern ist der Vergleich der Konfidenzintervalle in den meisten Fällen vorzuziehen.