Simulation fehlertoleranter Systeme mittels Importance Sampling

Diplomarbeit
von
Alexander Hartmann

angefertigt am
Lehrgebiet Technische Informatik I
der FernUniversität Hagen
(April 1993)

1. Gutachter: Prof. Dr. W. G. Schneeweiss
2. Gutachter: Prof. Dr. D. Tavangarian
Betreuer: F.-P. Holtmann, Dr. H. Bähring

Danksagung

Herrn Prof. Dr. Schneeweiss gilt mein besonderer Dank dafür, daß er mir ermöglicht hat, die vorliegende Diplomarbeit am Lehrgebiet Technische Informatik I anzufertigen. Ebenso danke ich ihm und Herrn Prof. Dr. Tavangarian für die Begutachtung der Arbeit.
Meinen Betreuern Herrn Holtmann und Herrn Dr. Bähring möchte ich für die Unterstützung, die Verbesserungsvorschläge und das intensive Korrekturlesen der Arbeit herzlich danken.

Abstract

This text describes the implementation of a simulation program called 'simsam' for fault–tolerant systems. The program runs on Sun UNIX–workstations using the SunView window–system.
Some variance reduction techniques are examined with respect to simulation of fault–tolerant systems. The techniques of the anithetic variate and the control variate are not efficient. Importance sampling shows a significant reduction of variance. So it was realized in the program. The implementation of the technique allows to use different types of probability distributions for modelling fault–tolerant systems.
Different variantes of importance sampling are presented, especially the dynamic importance sampling, which can be used in the program for some types of models.
The experimental part of the text discusses the behaviour of importance sampling in dependence of different parameters. Static and dynamic importance sampling are compared. The results of a simulation of a larger system with 32 components are presented.

Kurzfassung

Die vorliegende Diplomarbeit beschreibt die Implementierung des Programms `simsam` zur Simulation fehlertoleranter Systeme auf Sun Workstations. Es findet die graphische Benutzeroberfläche SunView unter UNIX Verwendung.
Weiterhin werden varianzreduzierende Maßnahmen auf ihre Verwendbarkeit für die Simulation fehlertoleranter Systeme untersucht. Die Methoden Antithetische Variable und Kontroll–Variable erweisen sich als nicht geeignet. Das Verfahren Importance Sampling ergibt dagegen signifikante Varianzreduktionen und wurde innnerhalb des Simulationsprogramms realisiert. Es arbeitet mit beliebigen Wahrscheinlichkeitsverteilungen, die für die Modellierung der Systeme eingesetzt werden können.
In der Arbeit werden verschiedene Importance–Sampling–Varianten vorgstellt, insbesondere das dynamische Importance Sampling, welches eingeschränkt auch in dem Programm eingesetzt werden kann.
In einem experimentellen Teil wird Importance Sampling hinsichtlich seines Verhaltens bezüglich verschiedener Parameter untersucht. Insbesondere werden die statische und die dynamische Form verglichen. Die Ergebnisse der Simulation eines größeren Systems mit 32 Komponenten werden vorgestellt.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die Ziele dieser Diplomarbeit sind:

- Untersuchung von varianzreduzierenden Maßnahmen auf ihre Eignung für die Simulation fehlertoleranter Systeme.
- Entwicklung eines Programms zur Simulation fehlertoleranter Systeme auf Sun Workstations unter einer graphischen Benutzeroberfläche. Die Funktionalität des Programms sollte sich an einer bestehenden PC–Lösung [lück88] orientieren.
- Implementierung eines geeigneten varianzreduziernden Verfahrens innerhalb des Programms. Dabei wurde Importance Sampling verwendet.

In [lück88] und [fuhrma88] werden die Implementierung eines Simulationsprogramm für Zuverlässigkeitsnetzwerke auf IBM–AT beschrieben. Mit der fortschreitenden Entwicklung der Rechnertechnologie wuchs die Notwendigkeit, ein solches Programm auch auf modernen Workstations verfügbar zu machen. Da trotz der gesteigerten Rechenleistung der Aufwand für sehr große Modellsysteme unpraktikabel groß war, wurde beschlossen, nach Möglichkeit die Rechenzeiten durch varianzreduzierende Maßnahmen zu verkürzen.
Das Ergebnis ist ein dreiteiliges Programmpaket, das im Rahmen des Projektes TIGER (Tool for Interactive Graphical Evaluation of Reliability) auf Sun Workstations unter UNIX realisiert wurde. Die Programme arbeiten mit der graphischen Benutzeroberfläche SunView. Der erste Teil ist der Blockschaltbildeditor `rbdedit` (siehe [löhr90a]), der es gestattet, Systeme mittels graphischer Eingabe von Zuverlässigkeitsblockschaltbildern zu definieren. Die technischen Daten eines Systems werden dann mittels des Modelleditors `mdledit` spezifiziert, welcher in [ruenge92] beschrieben wird. Den dritte Teil des Paketes bildet das in dieser Arbeit vorgestellte Programm `simsam` (Simulation fehlertoleranter Systeme mittels Importance Sampling).
Das Programm hat folgende Eigenschaften: Für die Systeme sind im Prinzip beliebige Verteilungen für Ausfall–, Fehler– und Reparaturdauern möglich. Bis

jetzt implementiert: Exponential–, Normal–, Weibull–, Gamma– und Gleichverteilung sowie die Angabe deterministischer Werte. [1] Das Ausfallverhalten kann durch eine Kombination mehrerer Verteilungen beschrieben werden, wodurch es z.B. möglich ist, eine Erhöhung der Ausfallrate auf Grund einer Belastung durch den Ausfall anderer Komponenten nachzubilden. Es können permanente, transiente und reparierbare Fehler auftreten. Die Reparaturstationen können sowohl endliche als auch unendliche Arbeitskapazitäten haben. Die Verwaltung ihrer Warteschlangen kann nach Kriterien wie z.B. Ausfallzeitpunkt, Reparaturdauern oder festen Prioritäten erfolgen. Reparaturen können durch andere Aufträge höherer Wichtigkeit unterbrochen werden. Weiter ist es möglich, daß eine Reparaturinstanz mehrere Aufträge gleichzeitig ausführt (Stichwort: Processor Sharing).
Das Programm kann stationäre und transiente Zuverlässigkeitskenngrößen wie z.B. Unverfügbarkeit, Lebensdauer und Überlebenswahrscheinlichkeit ermitteln. Stationäre Größen und solche, die mittels Importance Sampling nicht zu ermitteln sind, werden mit Hilfe der *Regenerativen Simulation* ermittelt. Im Rahmen des Importance Sampling lassen sich alle Wahrscheinlichkeitsparameter des Systems für die gesamte Simulationsdauer durch andere Parameter ersetzen. Außerdem ist es möglich, durch eine bestimmte Darstellung des Systems das sog. *Dynamic Importance Sampling* (DIS) zu realisieren. Das Prinzip von DIS ist es, Importance Sampling nur in bestimmten Situationen zu aktivieren.
Die Kapitel 2 und 3 beschäftigen sich mit den benötigten Grundlagen der Zuverlässigkeitstechnik und der Wahrscheinlichkeitstheorie sowie verschiedenen Varianzreduktionsmethoden. Im Kapitel 4 folgt eine Literaturstudie über den Einsatz von varianzreduzierenden Maßnahmen bei der Simulation fehlertoleranter Systeme. Im wesentlichen werden Importance–Sampling–Verfahren behandelt, darunter auch DIS. Außerdem wird das Prinzip der Regenerativen Simulation erläutert.
Im zweiten Teil der Arbeit werden die Fähigkeiten von `simsam` dargestellt: Im Kapitel 5 wird gezeigt, welche Typen von Systemen das Programm simulieren kann und welche Parameter es dazu gibt. Im Kapitel 6 werden verschiedene Darstellungsarten der Simulationsergebnisse vorgestellt und die dazu benötigten Formeln abgeleitet. Es folgt im Kapitel 7 eine Erläuterung der Benutzeroberfläche des Programms.
Der dritte Teil beschreibt die Realisierung von `simsam`: Im Kapitel 8 werden die Algorithmen, die zur Implementierung der verschiedenen Verteilungsfunktionen verwendet wurden, erklärt. Danach wird der eigentliche Simulationsalgorithmus dargestellt (Kapitel 9).
Im experimentellen Teil (Kapitel 10) werden verschiedene Varianzreduktionsmethoden getestet. Importance Sampling wird daraufhin untersucht, wie es sich

[1] Weitere Verteilungen können leicht hinzugefügt werden, wie in Kapitel 8.5 beschrieben wird.

unter der Veränderung einiger Systemparameter verhält, und die Ergebnisse der Simulation eines größeren Systems mit 32 Komponenten werden vorgestellt.
Den Abschluß der Arbeit bilden Zusammenfassung und Ausblick auf die Erweiterungsmöglichkeiten des Programms. Es folgt das Literaturverzeichnis. Im Anhang werden wichtige Grundlagen, Beweise und im Programm verwendete Algorithmen aufgeführt. Außerdem werden Hinweise zur Installation des Programms gegeben, seine Modulstruktur vorgestellt und anhand eines Beispiels gezeigt, wie man Modellsysteme eingibt, simuliert und die Ergebnisse darstellt.

Kapitel 2

Grundbegriffe der Zuverlässigkeitstechnik

Mit der fortlaufenden Technisierung steigt die Abhängigkeit des Menschen von der Technik. Insbesondere das Vordringen der Computer in alle Lebensbereiche führt dazu, daß ein Ausfall solcher Systeme schwerwiegende Folgen nach sich ziehen kann. Somit wird es immer wichtiger, Kenntnisse über die Zuverlässigkeit von technischen Systemen zu gewinnen. Das führt dazu, daß man genauere Informationen über Häufigkeit und Verteilung von Ausfällen erhält und somit bessere Vorhersagen machen kann. Insbesondere ist es dann möglich, die Systeme so zu gestalten, daß ihre Zuverlässigkeit erhöht wird.
Im folgenden Kapitel schildern wir zuerst, wie man die Zuverlässigkeitsstruktur von Systemen darstellt. Dann werden die wichtigsten Kenngrößen, die in diesem Zusammenhang von Bedeutung sind, genannt. Schließlich werden verschiedene Methoden vorgestellt, mit denen diese Größen ermittelt werden können. Ein einführendes Werk in das Gebiet ist [schnee91], vertiefend wird es in [schnee92] dargestellt.

2.1 Darstellung der Zuverlässigkeitsstruktur technischer Systeme

Ein technisches System kann man in verschiedene Teilsysteme zerlegen. Diese Teilsysteme können wiederum aus einzelnen Systemen aufgebaut sein und so weiter. Diesen Aufbau kann man in Form einer *Systemhierarchie* darstellen. Beispielsweise besteht ein Kugelschreiber aus einem Gehäuse, einer Mine und einer kleinen Sprungfeder. Die Mine besteht aus Hülle, Kugel und Tinte. Ebenso könnte man das Gehäuse weiter untergliedern (hier nicht dargestellt). Die Darstellung als Systemhierarchie sieht dann so wie in Abbildung 2.1 aus.
Man kann die Darstellung als Hierarchie nutzen, um bei der Ermittlung von Zuverlässigkeitskenngrößen (siehe Kapitel 2.2) die Komplexität zu reduzieren:

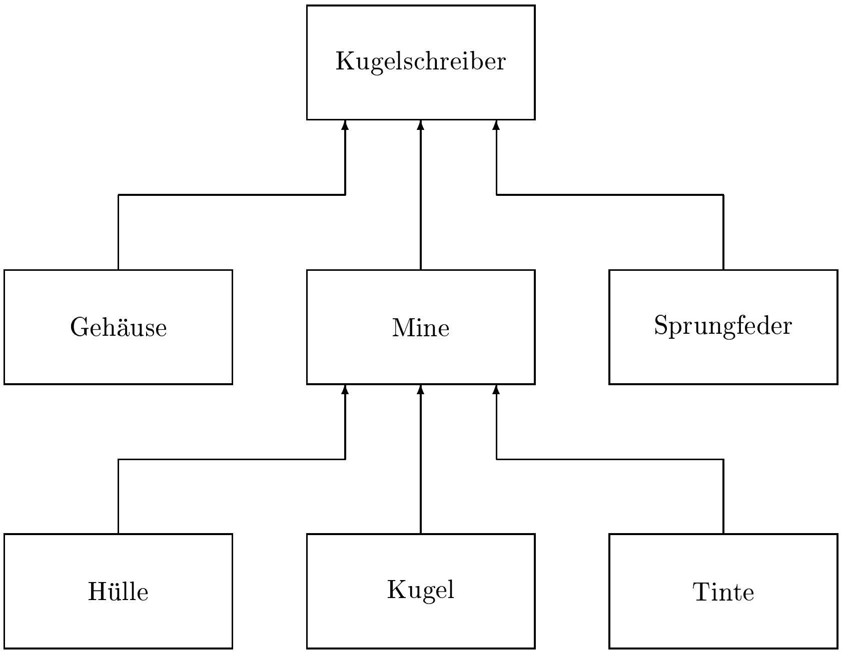

Abbildung 2.1: Systemhierarchie eines Kugelschreibers

Man betrachtet erst Teilsysteme, die auf niedrigen Hierarchieebenen liegen. Diese Teilsysteme werden dann bei der Gesamtbetrachtung des Systems als eine Einheit betrachtet. Die zuerst ermittelten Ergebnisse werden dabei als Parameter verwendet, welche die Teilsysteme beschreiben. Zum Beispiel kann man bei dem Kugelschreiber erst das Verhalten der Mine studieren, das sich aus den Parametern für Hülle, Kugel und Tinte ergibt. Bei der Behandlung des gesamten Kugelschreibers wird dann die Mine als atomar angenommen. Man hat somit zwei Modellsysteme mit je 3 Komponenten anstatt eines Systems mit 5 (atomaren) Komponenten betrachtet.
Aus einer Systemhierarchie kann man aber nicht ablesen, wie die Zuverlässigkeitsstruktur eines Systems aussieht, d.h. wann das Gesamtsystem in Abhängigkeit von den Zuständen seiner Teilsysteme defekt ist und wann nicht. Da es im folgenden nicht auf den hierarchischen Aufbau der Systeme ankommt, werden alle Teilsysteme, die gerade als atomar angesehen werden, als *Komponenten* $K_1, \ldots, K_n$ bezeichnet, egal ob sie sich selbst weiter untergliedern lassen oder nicht.
Jede Komponente kann einen der beiden Zustände INTAKT oder DEFEKT annehmen. Die Zustände der Komponenten zur Zeit t werden mit $Z_1(t), \ldots, Z_n(t)$ bezeichnet. Von Interesse ist nun, wie sich der Zustand $Z(t)$ des Gesamtsystems aus den Komponentenzuständen ergibt. Dieser Zusammenhang läßt sich mittels einer Funktion darstellen, welche in der Literatur *Strukturfunktion* oder

mißverständlich *Systemfunktion* [1] genannt wird:

$$\phi : \{\mathrm{INTAKT}, \mathrm{DEFEKT}\}^n \rightarrow \{\mathrm{INTAKT}, \mathrm{DEFEKT}\}$$

Es gilt dann $Z(t) = \phi(Z_1(t), \ldots, Z_n(t))$.
Wenn man den Zustand INTAKT mit dem Wahrheitswert FALSCH (=0) und den Zustand DEFEKT mit WAHR (=1) identifiziert, läßt sich die Strukturfunktion in Form einer Booleschen Funktion darstellen. Für große Systeme wird diese Darstellungsform aber zu unübersichtlich, so daß man graphische Darstellungsarten vorzieht. Die wichtigste Darstellungsform ist die des *Zuverlässigkeitsblockschaltbildes*, kurz *Blockschaltbild*. Sie wird im Anhang A.1 beschrieben.
Von Interesse ist nun, unter welchen Bedingungen ein System ausfällt, d.h. welche Komponenten dazu DEFEKT sein müssen. Eine kleinste Menge C von Komponenten, deren Ausfall zum Systemausfall führt, nennt man *minimalen Schnitt*. Dieser Name ist so zu erklären: Wenn man das Blockdiagramm an den Stellen, wo die Komponenten aus C sitzen, unterbricht, wird jeder Pfad vom Anfang bis zum Ende des Graphen unterbrochen, d.h. das Blockdiagramm wird zerschnitten.

2.2 Zuverlässigkeitskenngrößen

Diese Kenngrößen dienen dazu, ein System hinsichtlich seiner *Zuverlässigkeit* zu beschreiben, d.h. seiner Fähigkeit in einem bestimmten Zeitraum oder Zeitpunkt seinen Anforderungen zu genügen. Man unterscheidet zwischen *transienten* und *stationären* Kenngrößen. Transiente Größen beschreiben das System zu einem Zeitpunkt bzw. innerhalb einer Zeitdauer oder beziehen sich auf bestimmte Zustände des Systems. Stationäre Kenngrößen beschreiben das System im Grenzfall der unendlich langen Betriebszeit. Das ist natürlich nur möglich, falls das System reparierbar ist. Es werden im folgenden einige wichtige transiente und stationäre Kenngrößen genannt. Ihre genaue Definition findet sich in Anhang A.2.
Transiente Kenngrößen sind die *(Un–)verfügbarkeit*, die *Ausfall–* bzw. *Überlebenswahrscheinlichkeit* und die *mittlere Lebensdauer*. Stationäre Kenngrößen sind der *mittlere Ausfallabstand*, die *mittlere Ausfalldauer*, die *mittlere Brauchbarkeitsdauer* und die *stationäre Verfügbarkeit*.
Die Begriffe *Fehlerrate* und *Reparaturrate* werden im Anhang B eingeführt. Sie geben an, wie viele Komponenten pro Zeiteinheit ausfallen bzw. repariert werden.

[1] Unter einer Systemfunktion stellt man sich eher eine Funktion vor, die die Funktionsweise eines Systems beschreibt.

2.3 Methoden zur Ermittlung von Kenngrößen

Für die Ermittlung des Wertes einer Kenngröße gibt es drei Möglichkeiten: Experimente, analytische Berechnungen und Simulationen.
Experimente an realen Systemen sind die genaueste Möglichkeit, Kenngrößen zu bestimmen. Es sind keinerlei modellhafte Abstraktionen nötig, und man braucht keine genauen Kenntnisse über den internen Aufbau der Systeme. Der entscheidende Nachteil ist, daß Messungen an Systemen hoher Zuverlässigkeit einfach zu lange dauern. Man benötigt ja nicht nur einen Meßwert, sondern viele Werte, um das stochastische Verhalten der Systeme mit genügender Genauigkeit zu beschreiben. Außerdem sollen die Werte meistens schon in die Planung der betrachteten Systeme einfließen, damit man sie möglichst zuverlässig konstruieren kann. Der Bau vieler Testsysteme, besonders bei großen Projekten, wäre zu teuer. Schließlich gibt es Systeme, bei denen Experimente schlichtweg unmöglich sind, da das zu untersuchende Ereignis schwerwiegende Folgen hätte, z.B. ein Flugzeugabsturz.
Somit werden Experimente für Gesamtsysteme selten eingesetzt; man greift auf Berechnungen und Simulationen zurück. Für die Teilsysteme, die in der Hierarchie auf der niedrigsten Ebene stehen, sind Experimente allerdings unverzichtbar, weil Informationen benötigt werden, die als Grundlage für die Rechnungen und Simulationen dienen.
Analytische Rechnungen haben den Vorteil, daß sie, abgesehen von der Unsicherheit der Eingangsdaten, exakte Ergebnisse liefern. Ihr Nachteil ist, daß man sie nur für sehr kleine Systeme durchführen kann. Sie werden somit eingesetzt, um allgemeines Verständnis für die Materie zu entwickeln, Modelle zu erstellen und um die Ergebnisse von Simulationen an kleinen Beispielen überprüfen zu können.
Mittels numerischer Methoden können analytischen Rechnungen auch für größere Systeme (z.B. 100 Komponenten) durchgeführt werden. Beispiele sind die Programme SAVE [goyal86] und HARP [dugan86]. Der Nachteil der numerischen Programme ist, daß nur Modellsysteme mit bestimmten Eigenschaften berechnet werden können. Z.B. behandelt SAVE nur exponentielle Ausfall– und Reparaturverteilungen.
Mit der fortschreitenden Entwicklung der Computertechnik ist die Simulation ein wichtiges Verfahren geworden, um Kenngrößen zu ermitteln, insbesondere für große Systeme. Doch auch hier sind extreme Rechenzeiten (noch) an der Tagesordung, so daß man danach strebt, die Simulationen bei gleichbleibender Genauigkeit mit sog. *Varianzreduktionsverfahren* zu beschleunigen. Diese Diplomarbeit beschäftigt sich im wesentlichen mit der Implementierung eines solchen Verfahrens, dem *Importance Sampling*, welches im nächsten Kapitel erläutert wird.

Kapitel 3

Varianzreduzierende Maßnahmen

Die Genauigkeit einer Messung läßt sich erhöhen, indem man die Anzahl der Meßwerte vergrößert oder die Streuung der Werte verringert. Bei Verwendung von Simulationen läßt sich letzeres durch Einsatz varianzreduziernder Maßnahmen erreichen. Im Rahmen dieser Arbeit wird gezeigt, daß für die Simulation fehlertoleranter Systeme *Importance Sampling* am geeignetsten ist.
In diesem Kapitel nennen wir zuerst die für das Verständnis der Arbeit benötigten Grundlagen der Wahrscheinlichkeitsteorie und Statistik. Dann wird der Zweck und die Funktionsweise varianzreduzierender Maßnahmen erläutert.

3.1 Grundlagen der Wahrscheinlichkeitstheorie und Statistik

Die Literaturquellen dieses Abschnitts sind [moesch83] und [morgan84]. Im folgenden werden die Begriffe aus der Wahrscheinlichkeitstheorie genannt, die innerhalb dieser Arbeit verwendet werden. Sie werden im Anhang B definiert.
Eine *Zufallsgröße* (kurz ZG) ist eine Größe K, deren Verhalten zufällig ist, z.B. die Augenzahl bei einem Würfelwurf. Man unterscheidet zwischen *diskreten* und *kontinuierlichen* Zufallsgrößen, je nachdem ob die Menge der möglichen Werte höchstens abzählbar viele Elemente enthält oder nicht. Wenn man kontinuierliche Zufallsgrößen auf Computern simulieren möchte, muß man sie aber aufgrund der endlichen Zahlendarstellung durch diskrete Größen ausdrücken.
Führt man n Experimente mit einer Zufallsgröße aus, so sind *Mittelwert* $\overline{K}$ und *Stichprobenvarianz* S^2 wichtige Beschreibungsgrößen. Die Zahl n wird Stichprobenumfang genannt.
Mathematisch läßt sich eine Zufallsgröße durch ihre *Dichtefunktion* f, *Verteilungsfunktion* F oder die *Ereignisrate* λ (bzw. *Fehlerrate* bei Ausfällen) beschreiben. Alle drei Beschreibungsweisen sind äquivalent. Eine für Simulationen

wichtige Verteilung ist die *Gleichverteilung*.
Wichtige charakteristische Werte von Zufallsgrößen sind *Erwartungswert* $\mu = \mathrm{E}(K)$ und *Varianz* σ^2. Von Interesse ist besonders die Varianz zusammengesetzter Kenngrößen, was zum Begriff der *Kovarianz* Cov führt.
Weiterhin wird das Konzept der *bedingten Wahrscheinlichkeit* benötigt.

3.2 Varianzreduktion

Die Ergebnisse, die das Simulationsprogramm liefert, sind zuallererst Mittelwerte von Größen, die das simulierte System beschreiben. Nun taucht die Frage auf, wie genau das Ergebnis ist, d.h. in welchem Verhältnis die Mittelwerte zu den Erwartungswerten der Größen stehen.
Im Kapitel 6.1 wird gezeigt, daß man Mittelwerte in Form sogenannter Vertrauensintervalle $\mathrm{I} = [\overline{K} - \Delta, \overline{K} + \Delta[$ darstellen kann. Das Intervall gibt den Bereich an, in dem der Erwartungswert μ der Kenngröße mit einer bestimmten Vertrauenswahrscheinlichkeit β liegt. Diese Wahrscheinlichkeit geht streng monoton in die Intervallbreite 2Δ ein. Es gilt $\Delta = x\sigma/\sqrt{n}$ mit $x = \mathrm{N}_{0,1}^{-1}\left(\frac{\beta+1}{2}\right)$ (siehe Gleichung 6.2 in Kapitel 6.1).
Man erkennt aus der Darstellung des Vertrauensintervalls: Je größer die Varianz einer Kenngröße ist, desto größer ist das Intervall, d.h. desto größer ist die Ungenauigkeit des Ergebnisses. Das kann man auch aus der graphischen Auftragung der Normalverteilungsdichte $f_{\mu,\sigma}$ (siehe Kapitel 8.5) sehen. In Abbildung 3.1 sind die Dichten $f_{0,1}, f_{0,2}$ zweier Normalverteilungen mit den Varianzen $\sigma_1^2 = 1$ und $\sigma_2^2 = 4$ dargestellt. Die Dichtefunktion mit der höheren Varianz $f_{0,2}$ streut weiter um den Mittelwert als $f_{0,1}$.
Umgekehrt kann man eine Messung durch Erhöhung der Anzahl der Experimente beliebig genau machen. Das ist zwar einfach, kostet aber viel Zeit, da der Stichprobenumfang n nur mit dem Exponenten 0.5 eingeht. Man hat deshalb nach Methoden gesucht, mit denen sich die Varianz einer Meßgröße verringern läßt. Diese Methoden lassen sich bei Computerexperimenten am leichtesten verwirklichen. Im folgenden werden einige Verfahren geschildert.

3.2.1 Die Methode der Antithetischen Variablen

Bei Simulationen mit Zufallszahlen müssen diese Zahlen gemäß vorgegebener Verteilungen ausgelost werden. Üblicherweise werden die Zahlen so ausgelost, daß sie der $(0,1)$–Gleichverteilung gehorchen, und dann transformiert, so daß sie die gewünschte Verteilung realisieren (siehe Kapitel 8.3). Für die Verwirklichung der Gleichverteilung werden im allgemeinen Pseudozufallszahlengeneratoren verwendet, welche man so zu konstruiern versucht, daß die Zufallszahlen möglichst unabhängig voneinander sind.

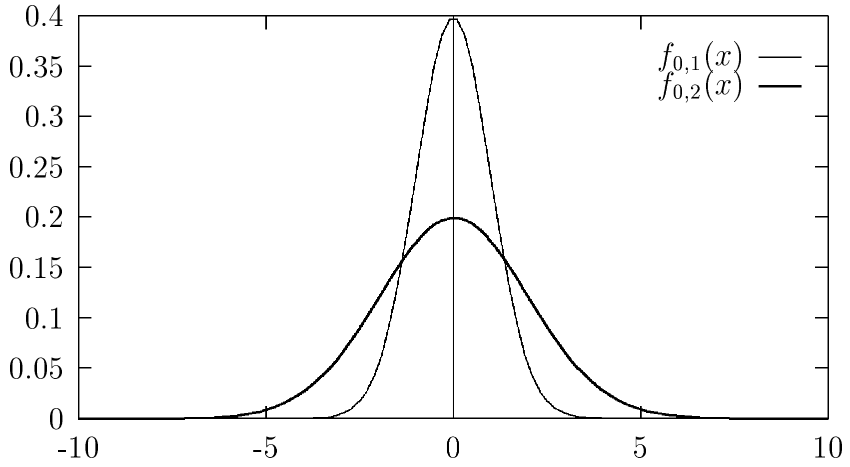

Abbildung 3.1: Darstellung zweier Normalverteilungen

Die Methode der *Antithetischen Variablen* [morgan84] beruht nun darauf, daß zwischen verschiedenen Zufallszahlen eine geeignete deterministische (Teil–) Beziehung, d.h. eine Korrelation hergestellt wird: Jede Zufallszahl Z wird doppelt verwendet. Bei jeder zweiten Anforderung einer Zufallszahl wird aber nicht Z sondern $1 - Z$, die sog. antithetische Variable, zurückgeliefert.
Das Verfahren kann deshalb funktionieren, weil mit Z auch $1 - Z$ $(0, 1)$–gleichverteilt ist. Sein Vorteil ist, daß es leicht zu realisieren ist, weil nur der Zufallszahlengenerator geändert werden muß.
Nachteilig ist, daß die Varianzreduktion von einer geeigneten „Weiterverwendung" der Zufallszahl Z abhängt, d.h. von den Transformationen. Je nachdem, welche Verteilungen simuliert werden sollen, kann die eingeführte Kovarianz positiv oder negativ sein. Außerdem hängt der Varianzgewinn bzw. Verlust von dem verwendeten Pseudozufallszahlengenerator ab, da dieser ja im Prinzip auch deterministisch arbeitet, den Determinismus aber „statistisch versteckt".

3.2.2 Die Methode der Kontroll–Variablen

Auch das Verfahren der *Kontroll–Variablen* beruht auf der Einführung von Korrelationen zwischen Zufallsgrößen (siehe [morgan84]).
Ausgangspunkt: Gesucht sei der Erwartungswert $\mathrm{E}(K)$ einer Zufallsgröße K. Um die Varianzreduktion zu erzielen, verwendet man eine weitere Zufallsgröße L, deren Erwartungswert $\mathrm{E}(L)$ bekannt ist. Statt nun einen Mittelwert $\overline{K}$ zu ermitteln, wird die Zufallsgröße $X := K - (L - \mathrm{E}(L))$ betrachtet. Ihr Mittelwert

kann genauso als Schätzwert für den Erwartungswert von K verwendet werden, denn es gilt $\mathrm{E}(X) = \mathrm{E}(K)$. Für die Varianz von X ergibt sich:

$$\mathrm{Var}(X) = \mathrm{Var}(K) + \mathrm{Var}(L) - 2\mathrm{Cov}(K, L) \tag{3.1}$$

Somit ergibt sich eine Varianzreduktion, falls die Bedingung $2\mathrm{Cov}(K, L) > \mathrm{Var}(L)$ erfüllt ist.
Das Verfahren hat den entscheidenden Nachteil, daß die Wahl der geeigneten Zufallsgröße L von K abhängt. Bei `simsam` ist es aber so, daß nicht nur K fast beliebig sein kann, sondern auch vor einem Simulationslauf überhaupt nicht bekannt ist, da sie sich aus einem komplizierten Zusammenspiel vieler Verteilungen ergibt. Somit kann diese Methode hier nicht verwendet werden.

3.2.3 Importance Sampling

Vorgegeben sei wiederum eine ZG K mit der Dichtefunktion f. Man möchte den Erwartungswert einer Funktion W der ZG bestimmen, d.h.:

$$\mathrm{E}(W(K)) = \int_{-\infty}^{+\infty} W(k) f(k)\, dk \tag{3.2}$$

In der normalen Simulation werden n Zufallswerte $k_1, \ldots, k_n$ gemäß der Dichtefunktion f ausgelost und folgender Mittelwert gebildet:

$$\overline{W(K)} := \frac{1}{n} \sum_{i=1}^{n} W(k_i) \tag{3.3}$$

Importance Sampling (IS) beruht auf der Verwendung einer zweiten Zufallsgröße L mit der Dichtefunktion g [vdi6 85], welche $g(k) \neq 0 \quad \forall\, k$ erfüllt. Denn es gilt:

$$\begin{aligned}
\mathrm{E}(W(K)) &= \int_{-\infty}^{+\infty} W(k) f(k)\, dk \\
&= \int_{-\infty}^{+\infty} W(k) \frac{f(k)}{g(k)} g(k)\, dk \\
&= \mathrm{E}(Wf/g(L)) \qquad (3.4)
\end{aligned}$$

Man kann also genauso gut n Zufallszahlen $l_1, \ldots, l_n$ gemäß der Dichtefunktion g auslosen und den Mittelwert $\overline{Wf/g(L)}$ ausrechnen, um einen erwartungstreuen Schätzwert für $\mathrm{E}(W(K))$ zu erhalten. Im folgenden werden wir den Wert $f(t)/g(t)$ auch *IS–Faktor* nennen, da alle Meßwerte $W(l)$ mit ihm multipliziert werden.
Interessant ist nun ein Vergleich der Varianzen beider Methoden, verwendet wird die Darstellung in Gleichung B.10 aus Anhang B:

$$\begin{aligned}
\mathrm{Var}(Wf/g(L)) &= \mathrm{E}(Wf/g(L)^2) - \mathrm{E}(Wf/g(L))^2 \\
&= \mathrm{E}(Wf/g(L)^2) - \mathrm{E}(W(K))^2 \\
\text{i.allg.} &\neq \mathrm{E}(W(K)^2) - \mathrm{E}(W(K))^2 \\
&= \mathrm{Var}(W(K))
\end{aligned} \tag{3.5}$$

Durch die Verwendung von IS kann sich somit die Varianz ändern. Es hängt nun von der Wahl der Dichtefunktion g ab, ob sich die Varianz verringert. Daß eine Varianzreduktion überhaupt möglich ist, zeigt folgendes Beispiel: Man wähle $g(k) := W(k)f(k)/\mathrm{E}(W(K))$. Für $Wf/g(k)$ ergibt sich durch Einsetzen: $Wf/g(k) = \mathrm{E}(W(K))$, d.h. alle Summanden des zu berechnenden Mittelwerts sind konstant, was eine Varianz von 0 ergibt. Dieses Beispiel hat natürlich nur akademischen Wert, da für diese Definition von g der gesuchte Erwartungswert bereits bekannt sein muß, womit die Durchführung einer Simulation nicht erforderlich ist.
In dem Kapitel 10 werden wir versuchen, einige grobe Regeln über die Wahl geeigneter IS-Dichtefunktionen g aufzustellen. Grundsätzlich ergab sich aus den Experimenten anderer Autoren, und dies wurde auch durch eigene Experimente bestätigt, daß IS für die Ermittlung kleiner Werte geeignet ist, d.h. Werte, die mit seltenen Ereignissen korrespondieren. Die IS–Parameter sollten so gewählt werden, daß diese Ereignisse häufiger, aber nicht zu häufig werden.
Das läßt sich für einen Spezialfall leicht einsehen. M sei eine reelle Menge, für die $k \in M$ unter K ein seltenes Ereignis ist, d.h. $\mathrm{P}(k \in M) \ll 1$. Die betrachtete Kenngröße habe die Gestalt $W = W' 1_{k \in M}$. Dabei ist W' eine Funktion und $1_{k \in M}$ ist die charakterische Funktion von M, d. h. sie liefert 1, wenn ihr Argument in M liegt und 0 sonst. Für die Kenngröße W sind somit nur Werte aus der Menge M wichtig. Wenn man nun die IS–Dichte g so wählt, daß $f(k)/g(k) \ll 1$ für alle $k \in M$ gilt, so ergibt sich:

$$\begin{aligned}
\mathrm{E}(Wf/g(L)^2) &= \mathrm{E}(W'^2 1^2_{k \in M} f^2/g^2(L)) \\
&\ll \mathrm{E}(W'^2 1^2_{k \in M} f/g(L)) \\
&= \mathrm{E}(W(K)^2)
\end{aligned} \tag{3.6}$$

Es ergibt sich also eine Varianzreduktion, indem das Ereignis $k \in M$ häufiger gemacht wurde. Leider sind in der Praxis aufgrund der komplexen Natur der Modellsysteme weder M noch f überschaubar. Sie ergeben sich durch Wechselwirkungen vieler Systemkomponenten. Auch die Wahl der IS–Dichte g erfolgt nicht direkt, sondern indem man viele einzelne IS–Verteilungen wählt, aus denen g hervorgeht. Vielfach ist es so, daß eine Wahl der IS–Verteilungen einige Bereiche aus M wahrscheinlicher macht und andere dagegen unwahrscheinlicher. Meistens muß man sich mit heuristischen Regeln für die Wahl der IS–Parameter begnügen. Einige werden in Kapitel 4 vorgestellt.

3.2.4 Verwendung deterministischer Werte

Das Grundverfahren bei Simulationen ist es, Werte auszulosen und aus diesen die gesuchten Größen zu berechnen. Die Varianz der berechneten Größe ist eine Folge des Ausloseprozesses. Wenn man nun einige der auszulosenden Werte durch ihre Mittelwerte ersetzt, so kann sich ggf. die Streuung der gemessenen Größe verringern.
Diese Methode ist kein Verfahren im eigentlichen Sinne, da es immer von dem jeweiligen Untersuchungsgegenstand abhängt, ob und wie es eingesetzt werden kann. Insbesondere ist im allgemeinen nicht klar, ob die ermittelten Werte erwartungstreu sind. Der Vorteil des Verfahrens ist aber, daß es sich ohne großen zusätzlichen Aufwand implementieren läßt. Man verwendet für einige Verteilungen einfach Stufenfunktionen, also deterministische Werte.
Beispiele für diese Methode finden sich auch im folgenden Kapitel, das sich im wesentlichen mit Importance Sampling beschäftigt.

Kapitel 4

Importance Sampling in der Simulation

Wie die Überlegungen im Kapitel 3 und unsere eigenen Experimente (siehe Kapitel 10) zeigen, ist Importance Sampling die einzige geeignete der vorgestellten Methoden zur Varianzreduktion. Entsprechend findet sich in der Literatur zur Simulation fehlertoleranter Systeme im wesentlichen nur dieses Verfahren. In diesem Kapitel wird geschildert, wie sich die Anwendung des Importance Sampling auf diesem Gebiet mit der Zeit entwickelt hat. Zunächst wurden Systeme durch Zustandsdiagramme dargestellt [conway87]. Auf diese Weise war es möglich, Systeme mit exponentialverteilten Lebens– und Reparaturdauern (siehe Abschnitt 8.5) zu simulieren. Die Ausfall- und Instandsetzungsraten waren für alle Komponenten konstant. Mit zustandsabhängigen Raten konnten in [carras90] auch propagierende Fehler simuliert werden, das sind Fehler, deren Auftreten in einer Komponente zu erhöhten Ausfallwahrscheinlichkeiten in anderen Systemteilen führt. Eine begrenzte Ausweitung auf nicht–exponentielle Verteilungen erfolgte in [geist89] und [goyal92]. Eine Ausweitung auf beliebige Ausfall–, Reparatur– und Wartungsprozesse ist in [nicola90] dargestellt.
Im folgenden werden die einzelnen Konzepte genauer erläutert und auch die dabei verwendeten Importance–Sampling–Varianten vorgestellt. Insbesondere wird dargestellt, wie die Autoren die IS–Dichtfunktion g, allgemein gesprochen die IS–Parameter, gewählt haben. Die Faustregel lautet dabei: Wähle die Parameter so, daß ein Systemausfall wahrscheinlicher wird.

4.1 Simulation durch Zustandsdiagramme

Ein System mit n Komponenten, deren Lebens– und Reparaturdauerverteilungen alle exponentiell sind, kann man mittels eines Zustandsdiagramms darstellen: Jeder mögliche Wert des Zustandsvektors $(Z_1, \ldots, Z_n)$ repräsentiert einen Systemzustand X, wobei jedes Z_i die Werte INTAKT und DEFEKT annehmen kann. Jeder Ausfall und jede Reparatur einer Komponente entspricht einem Zustandsübergang. Die Wechsel der Zustände erfolgen mit konstanten Raten, die gerade die Ausfall– bzw. Reparaturraten sind. Da so ein Zustandsgraph mit der Größe des Systems n exponentiell wächst, wird er schnell sehr groß. Man faßt deshalb äquivalente Zustände zusammen und addiert die Raten der Übergänge, die in diese Zustände führen.

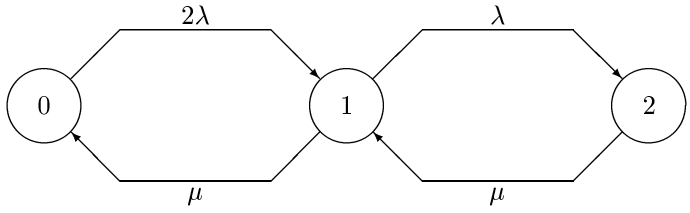

Abbildung 4.1: Zustandsdiagramm eines 1–von–2–Systems

In Abbildung 4.1 ist als Beispiel das Zustandsdiagramm eines 1–von–2–Systems gezeigt [conway87]. Beide Komponenten fallen mit der Rate λ aus und werden mit der Rate μ (nur eine Reparatur auf einmal) repariert. Im Zustand 0 sind alle Komponenten INTAKT, d.h. $0 \equiv (\mathrm{INTAKT}, \mathrm{INTAKT})$. Im Zustand 1 ist genau eine Komponente ausgefallen, d.h. hier wurden die Zustände (INTAKT,DEFEKT) und (DEFEKT,INTAKT) zusammengefaßt. Im Zustand 3 sind schließlich beide Komponenten DEFEKT, d.h. das System ist ausgefallen. In [conway87] wird dieses System wie folgt simuliert: Man startet im Zustand 0. Es wird nun angenommen, daß das System in jedem Zustand X solange verweilt, wie es der Erwartungswert der Verteilung aller aus X herausführenden Übergänge angibt. Diese Verteilung ist exponentiell, wobei die Rate gleich der Summe aller Raten dieser Übergänge ist. Man nennt diese Verweilzeiten auch *(Zustands–) Aufenthaltsdauern.* Während der Simulation wird natürlich nicht solange gewartet, wie die Aufenthaltsdauer angibt, sondern nur eine Variable, welche die Zeit in der Simulation angibt, um diese Dauer erhöht. Da der Erwartungswert einer Exponentialverteilung mit der Rate α gleich $1/\alpha$ ist, ergeben sich für das obige System Zustandsaufenhaltsdauern von $1/2\lambda$, $1/(\lambda+\mu)$ und $1/\mu$. Der nächste Zustand wird dann unter allen möglichen Folgezuständen X' ausgelost, wobei deren Wahrscheinlichkeiten $p(X, X')$ Quotienten aus der jeweiligen Übergangsrate $\lambda(X, X')$ und der Summe aller Übergangsraten aus dem aktuellen Zustand sind ($N(X)$ sei die Menge aller von X aus erreichbaren Zustände):

$$p(X, X') = \lambda(X, X') / \sum_{Y \in N(X)} \lambda(X, Y) \tag{4.1}$$

Man nennt das Ergebnis einer so durchgeführten Umsetzung eines Zustandsdiagramms in ein Modell mit Aufenthaltszeiten und Übergangswahrscheinlichkeiten auch *Diskrete–Zeit–Zustandsdiagramm*, da Zustandsübergänge nur noch an bestimmten Zeitpunkten erfolgen. Das äquivalente Diskrete–Zeit–Zustandsdiagramm für das obige Beispiel ist in Abbildung 4.2 dargestellt [conway87].

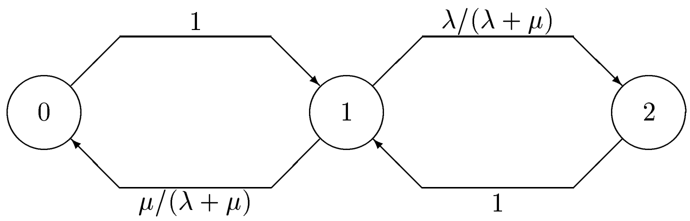

Abbildung 4.2: Diskrete–Zeit–Diagramm eines 1–von–2–Systems

Bei dieser Darstellungsweise wird Importance Sampling realisiert, indem für die Übergangswahrscheinlichkeiten andere Werte $p'(X, X')$ gewählt werden. Eine Dichtefunktion wie in der allgemeinen Form in Kapitel 3.2 tritt wegen der diskreten Zeitstruktur nicht mehr auf.
Bei der Berechnung der Mittelwerte wird in der allgemeinen Form des Importance Sampling der Faktor $f(k)/g(k)$ verwendet. Im Fall der Simulation mittels Zustandsdiagramm tritt am Ende eines Meßdurchgangs kein Wert k aus einem kontinuierlichen Bereich auf, aus dem der Meßwert berechnet wird, sondern der Wert ergibt sich aus der Folge der Zustände $X_1, \ldots, X_k$, die durchlaufen wurden. Der IS–Faktor, der hierbei auftritt, ist das Verhältnis der Wahrscheinlichkeiten dieses Pfades im ursprünglichen und im IS–System. Diese Wahrscheinlichkeiten ergeben sich als Produkte der einzelnen Übergangswahrscheinlichkeiten:

$$\frac{\mathrm{P}(X_1, \ldots, X_k)}{\mathrm{P}'(X_1, \ldots, X_k)} = \prod_{i=2}^{k} \frac{p(X_{i-1}, X_i)}{p'(X_{i-1}, X_i)} \tag{4.2}$$

Üblicherweise sind die Ausfallraten sehr klein gegen die Reparaturrate, z.B. $\lambda = 10^{-4}$ und $\mu = 1$, so daß sehr viele Simulationsschritte benötigt werden, um in den interessanten Zustand 2 zu gelangen. Die Methode, die in [conway87] verwendet wird, um durch IS diesen Zustand häufiger zu erreichen, nennt sich *Failure Biasing*: In jedem Zustand, aus dem sowohl Ausfall– als auch Reparaturübergänge hinausführen (in dem Beispliel also nur der Zustand 1), werden die Ausfallraten mit einem Faktor multipliziert, so daß die Summe aller Ausfallwahrscheinlichkeiten den Wert BIAS $\in [0, 1]$ annimmt. Entsprechend ist die Summe aller Reparaturwahrscheinlichkeiten 1-BIAS.

Conway und Goyal empfehlen Werte von BIAS in der Nähe von 0.5. Sie erreichten damit bei der Berechnung der Unverfügbarkeit kleinerer Systeme Genauigkeitsgewinne um den Faktor 15 bis Faktor 40 (bei der gleichen Anzahl der simulierten Ereignisse =Übergänge), das heißt, die Breite der Vertrauensintervalle wurde um diesen Faktor verkleinert. Für ein großes System mit 70 Komponenten wurde der Faktor 288 erreicht. Dabei wurden die Ergebnisse mittels eines numerischen Werkzeuges SAVE [goyal86] verifiziert. Leider wird in [conway87] nicht gesagt, um welchen Faktor der Rechenaufwand durch die Verwendung von Importance Sampling ansteigt, so daß sich der Nutzen der Methode nicht abschätzen läßt.

4.2 Zustandsabhängige Raten

Das Simulationsverfahren aus [carras90] verwendet die oben vorgestellten Zustandsdiagramme direkt, d.h. Zustandsübergänge können jederzeit erfolgen. Als Erweiterung werden zustandsabhängige Raten zugelassen. Auf diese Weise können sich fortpflanzende Fehler dargestellt werden: Eine Komponente K hat normalerweise die Ausfallrate λ_1. Falls nun der Ausfall einer anderen Komponente $\hat{K}$ zu einer höheren Belastung von K führt, erhalten die Zustandsübergänge für einen Ausfall von K, die aus Zuständen herausführen, in denen $\hat{K}$ DEFEKT ist, die Rate $\lambda_2 > \lambda_1$.
Das Failure Biasing nach [conway87] hat den Nachteil, daß alle Ausfallraten eines Zustandes mit dem gleichen Faktor multipliziert werden. So sind die Ausfallwahrscheinlichkeiten in einem Zustand unabhängig davon, wie stark der jeweilige Ausfall zum gesamten Systemausfall beiträgt. Carrasco verwendet deshalb ein erweitertes Schema, das *Failure Distance Biasing*. Für jeden Zustand wird die sog. *Fehlerdistanz* berechnet, das ist die Anzahl der Komponenten, die noch minimal ausfallen müssen, damit das Gesamtsystem DEFEKT ist. Bei der Wahl der IS–Parameter erhalten dann jene Übergänge eine höhere Wahrscheinlichkeit, durch die sich die Fehlerdistanz verringert. Somit wird das System noch stärker in Richtung des Ausfalls „getrieben“, welches ja gerade der Zweck des Failure Biasing sein soll.
Weiterhin wird ein Verfahren, genannt *Estimator Decomposition*, vorgestellt. Damit ist es möglich, Kenngrößen zu ermitteln, bei denen man Importance Sampling nicht direkt anwenden kann. Man verwendet dabei die sog. *Regenerative Simulation*. Siehe dazu das Kapitel 4.5.
Die günstigsten IS–Paramter werden in [carras90] mittels eines iterierten Try–and–Error–Verfahrens ermittelt: Ausgehend von festgelegten Startparametern läßt man die Simulation mehrmals laufen und ändert die Parameter immer so, daß sich die Genauigkeit erhöht. Das Verfahren wurde auf ein System, bestehend aus 38 Komponenten angewendet. Es zeigte sich, daß man mittels Failure Biasing typischerweise 100 mal so viel Ereignisse simulieren mußte wie bei Ver-

wendung von Failure Distance Biasing, um die gleiche Genauigkeit zu erreichen.

4.3 Variable Aufenthaltsdauern

Die zuerst vorgestellte Variante arbeitet mit konstanten Zustandsaufenthaltsdauern. Auf diese Weise erhält man schon eine gewisse Varianzreduktion, da die Aufenthaltszeiten eben nicht variieren. Bei beiden vorherigen Varianten kann man nur Modelle simulieren, die rein exponentiell verteilte Zeiten haben.
In [geist89] wird deshalb die Dauer von Aufenthalten in jedem Zustand X gemäß einer Dichtefunktion $f(t|t', X)$ ausgelost, die auch von der aktuellen simulierten Zeit t' des Systems abhängen kann. Wenn derselbe Zustand zu verschiedenen Zeiten eingenommen wird, können somit die Aufenthaltsdauern nach unterschiedlichen Verteilungen ausgelost werden. Welcher Übergang dann nach Ablauf des Aufenthalts erfolgt, wird wieder durch Übergangswahrscheinlichkeiten bestimmt. Durch dieses Verfahren ist es somit möglich, etwas allgemeinere Systeme zu simulieren. Die Einschränkung liegt darin, daß in einem Zustand allen Prozessen, die zum Verlassen des Zustandes führen, die gleiche Aufenthaltsdauerverteilung zugewiesen wird: Es ist somit egal, ob das nächste Ereignis eine (relativ schnelle) Reparatur ist, oder ob ein Ausfall vorliegt.
Es werden von den Autoren nur Systeme bis zu einem bestimmten Zeitpunkt T („Mission Time") betrachtet. Die Verteilungen, welche die Zustandsaufenthaltsdauern angeben, werden für IS so normiert, daß der nächste Zustandsübergang mit Sicherheit vor Erreichen des Zeitpunktes T erfolgt („*Forced Transitions*").
Bei der Wahl der IS–Übergangswahrscheinlichkeiten wird eine Form des Failure Biasing verwendet: Im Zustand X werden alle niedrigen Wahrscheinlichkeiten (z.B. für Ausfälle) mit einem Faktor multipliziert, so daß die Summe der entsprechenden Wahrscheinlichkeiten den Wert $w(X)$ hat. Analog werden hohe Übergangswahrscheinlichkeiten auf $1-w(X)$ normiert. Es erhalten die Zustände, aus denen Ausfallereignisse herausführen, welche einen großen Beitrag zum Systemausfall liefern, einen hohen Biaswert $w(X)$. Innerhalb eines Zustandes hängen die IS–Übergangswahrscheinlichkeiten somit nur vom Verhältnis der ursprünglichen Wahrscheinlichkeiten ab.
Diese Art der Parameterwahl wurde mit dem ursprünglichen Failure Biasing verglichen, bei dem für alle Zustände $w(X) = 0.5$ gewählt wurde. Es wurde die Überlebenswahrscheinlichkeit für $t = 10$ bei einem System aus 3 Komponenten ermittelt. Es wurden allerdings nur zeitunabhängige exponentielle Verteilungen für die Aufenthaltsdauern verwendet. Das modifizierte Verfahren erzielte ein um den Faktor 50 kleineres Vetrauensintervall [1] (siehe Kapitel 6.1) bei gleicher Anzahl der simulierten Ereignisse als das „$w = 0.5$"–Verfahren. Außerdem benötigte das ursprüngliche Failure Biasing eine um 63% höhere Rechenzeit.

[1] Zur gleichen Vertrauenszahl β.

Auch bei [goyal92] wären im Prinzip beliebige Aufenthaltsdauerverteilungen möglich, aber nur exponentielle werden angewendet. Zusätzlich werden noch zwei Importance–Sampling–Verfahren angegeben, welche die Aufenthaltsdauerverteilungen berücksichtigen.

- *Forced Transitions* wird nur im Zustand 0, wo alle Komponenten INTAKT sind, angewendet.
- *Conditioning*: Die Zeitspannen, die ein System im Zustand 0 verbleibt, werden deterministisch festgelegt.

Die beiden Verfahren wurden verglichen. Es ergab sich, daß sie für die Berechnung der Unzuverlässigkeit gleichwertig sind, aber bei der Ermittlung der Verfügbarkeit zu einem bestimmten Zeitpunkt ist Conditioning um bis zu Faktor 3 genauer, d.h. das Vertrauensintervall ist um diesen Faktor kleiner.

4.4 Beliebige Prozesse

In [nicola90] wird geschildert, wie man Systeme, bei denen die auftretenden Ereignisse gemäß beliebiger Verteilungen ausgelost werden, mittels Importance Sampling simulieren kann. Dieses Verfahren bildet die Grundlage unseres Simulationsprogramms `simsam`; es wird deshalb nun etwas ausführlicher in der Form vorgestellt, in der es dort realisiert wurde.
Die Simulation erfolgt nicht mehr mittels eines Zustandsgraphen, sondern über eine Menge E von Ereignissen, welche in Form einer Liste verwaltet wird. Zu Beginn der Simulation, bei $t = 0$, wird die Ereignismenge initialisiert, üblicherweise mit Ausfallereignissen für alle Komponenten. Der Anfangszustand werde mit X_0 bezeichnet. Der Zeitpunkt jedes Ereignisses $e \in E$ wird gemäß einer Verteilung mit der Dichtefunktion $f(t; e)$ und der Verteilungsfunktion $F(t; e)$ ausgelost.
Es wird nun immer das Ereignis der Liste bearbeitet, das als nächstes stattfindet. Mit e_i $(i > 0)$ wird ein Ereignis bezeichnet, welches insgesamt als i-tes bearbeitet wird, mit t_i der Zeitpunkt, an dem es stattfindet, und mit $t^*_{e_i}$ die Zeit, zu der es erzeugt wurde. Eine Ereignismenge, in der e_i die minimale Ereigniszeit hat, wird mit E_i bezeichnet. Den Zustand, der sich nach dem i–ten Ereignis einstellt, nennen Nicola et al. X_i.[2]
Durch jedes Ereignis können neue Ereignisse erzeugt werden, z.B. kann durch den Ausfall einer Komponente ein Reparaturereignis entstehen. Welche Ereignisse genau erzeugt werden, kann über Wahrscheinlichkeiten bestimmt werden. Es sei $p_i(E, E')$ die Wahrscheinlichkeit, daß beim i–ten Zustandsübergang aus der Ereignismenge E die Ereignismenge E' ensteht. Diese Wahrscheinlichkeit

[2] In [nicola90] wird der $(i + 1)$–te Übergang dagegen von e_i ausgelößt. Diese Bezeichnungsweise halten wir für etwas verwirrend.

kann natürlich von allerlei Faktoren wie der Zeit oder vorangehenden Zuständen abhängen, was durch den Index i am p symbolisiert wird.
Das Importance–Sampling–Verfahren erlaubt es nun, alle Verteilungen $F(t;e)$ durch Verteilungen $F'(t;e)$ und alle Wahrscheinlichkeiten p_i durch Wahrscheinlichkeiten p'_i zu ersetzen. Die Wahrscheinlichkeitsdichte für einen Pfad, der durch eine Zustandsfolge $X_1, \ldots, X_k$ mit den Übergangszeiten $t_1, \ldots, t_k$ und den Ereignismengen $E_1, \ldots, E_{k+1}$ charakterisiert wird, wird mit

$$\mathrm{P}(X_1, \ldots, X_k; t_1, \ldots, t_k; E_1, \ldots, E_{k+1})$$

bezeichnet. Entsprechend wird die Wahrscheinlichkeitsdichte dieses Pfades unter IS mit $\mathrm{P}'(\ldots)$ bezeichnet. Weiterhin sei $p(X, X'; t, t'; E, E')$ die Wahrscheinlichkeitsdichte, daß aus dem Zustand X, der zur Zeit t erreicht wurde und die Ereignismenge E zur Folge hatte, zur Zeit t' in den Zustand K' übergegangen wird und sich die Ereignismenge E' ergibt. Für IS sei $p'(\ldots)$ entsprechend definiert.
Der IS–Faktor ist der Quotient der Pfadwahrscheinlichkeitsdichten, welche sich als Produkte von Übergangsdichten darstellen lassen[3]:

$$\begin{aligned} &\frac{\mathrm{P}(X_1, \ldots, X_k; t_1, \ldots, t_k; E_1, \ldots, E_{k+1})}{\mathrm{P}'(X_1, \ldots, X_k; t_1, \ldots, t_k; E_1, \ldots, E_{k+1})} = \\ &\prod_{i=2}^{k} \frac{p(X_{i-1}, X_i; t_{i-1}, t_i; E_i, E_{i+1})}{p'(X_{i-1}, X_i; t_{i-1}, t_i; E_i, E_{i+1})} \end{aligned} \tag{4.3}$$

Es wird nun gezeigt, wie die Übergangsdichten $p(X_{i-1}, X_i; t_{i-1}, t_i; E_i, E_{i+1})$ berechnet werden. Die Wahrscheinlichkeitsdichte, daß das Ereignis e_i genau zur Zeit t_i stattfindet ist:

$$f(t_i - t^*_{e_i}; e_i), \tag{4.4}$$

denn zur Zeit t_i hat e_i gerade das Alter $t_i - t^*_{e_i}$.
Man weiß aber auch noch, daß e_i alle die vorhergehenden Übergänge, die nach seiner Erzeugung erfolgten, nicht ausgelöst hat, d.h. bis zur Zeit t_{i-1} nicht stattfand. Dieses Nichtauslösen hat die Wahrscheinlichkeit

$$1 - F(t_{i-1} - t^*_{e_i}; e_i) \tag{4.5}$$

Daraus ergibt sich: Die Wahrscheinlichkeitsdichte dafür, daß e_i genau den i–ten Übergang zur Zeit t_i auslöst, unter der Voraussetzung, daß die vorherigen Zustandsübergänge nicht von e_i ausgelöst wurden, ist gerade:

[3] In unserem Programm wird die Ereignismenge E_1 deterministisch festgelegt: Für jede Komponente wird ein Ausfallereignis erzeugt. Damit ergibt sich hieraus kein Beitrag zum IS–Faktor.

$$f(t_i - t^*_{e_i}; e_i)/(1 - F(t_{i-1} - t^*_{e_i}; e_i)) \tag{4.6}$$

Weiter muß man berücksichtigen, daß die anderen Ereignisse, die sich vor dem i–ten Übergang in der Ereignismenge befinden, also die Ereignisse $e \in E^*_i := E_i \setminus \{e_i\}$, diesen Zustandsübergang nicht auslösen, unter der Vorausetzung, daß sie die vorhergehenden Übergänge auch nicht ausgelöst haben. Die bedingte Wahrscheinlichkeit dafür ist gerade (für jedes dieser e):

$$(1 - F(t_i - t^*_e; e))/(1 - F(t_{i-1} - t^*_e; e)) \tag{4.7}$$

Die Übergangswahrscheinlichkeitsdichte von X_{i-1} nach X_i ist das Produkt der Wahrscheinlichkeit (4.6), der Auswahlwahrscheinlichkeit $p_i(E_i, E_{i+1})$ und aller Wahrscheinlichkeiten (4.7):

$$\begin{aligned} &p(X_{i-1}, X_i; t_{i-1}, t_i; E_i, E_{i+1}) = \\ &\qquad \frac{f(t_i - t^*_{e_i}; e_i)}{1 - F(t_{i-1} - t^*_{e_i}; e_i)} p_i(E_i, E_{i+1}) \prod_{e \in E^*_i} \frac{1 - F(t_i - t^*_e; e)}{1 - F(t_{i-1} - t^*_e; e)} \end{aligned} \tag{4.8}$$

Die entsprechende Dichtefunktion $p'(X_{i-1}, X_i; t_{i-1}; t_i, E_i, E_{i+1})$ erhält man, indem man alle f durch f', alle F durch F' und alle p_i durch p'_i ersetzt. Durch Einsetzen in Gleichung 4.3 erhält man den gesamten IS–Faktor. Man könnte nun argumentieren, daß sich viele der Werte $1 - F(t - t^*_e; e)$ in (4.3) gegeneinander wegkürzen. Das ist aber nur der Fall, wenn sich die Verteilungen im Laufe der Simulation nicht ändern. Wenn sich aber z.B. die Ausfallrate einer Komponente ändert, weil sie durch einen Defekt belastet wird, muß der Ausfallzeitpunkt neu berechnet, d.h. „*rescheduled*" werden und danach die geänderte Verteilung benutzt werden. Wie Rescheduling arbeitet, wird im Abschnitt 8.4 beschrieben.
Für die Wahl der IS–Verteilungen wird in [nicola90] ein besonderes Schema verwendet, das *Dynamic Importance Sampling* (DIS): Zuerst werden für alle Komponenten ihre normalen Verteilungen genommen. Bei einem Ausfall einer Komponente K werden solche Komponenten, deren Ausfälle das gesamte System wesentlich näher an den Zustand DEFEKT bringen, gemäß ihrer IS–Verteilungen rescheduled[4]. Falls K repariert wurde, ohne daß es zu einem Systemausfall kam, werden die Komponenten wieder mit den ursprünglichen Verteilungen rescheduled.
Das Verfahren wurde von den Autoren auf ein zweikomponentiges System angewendet. Alle Ausfall– und Reparaturverteilungen waren 2–stufig exponentiell.[5]

[4]Diese Komponenten könnten gerade jene sein, die in den minimalen Schnitten enthalten sind, welche auch K enthalten.

[5]Die Verteilung ist exponentiell. Mit einer Wahrscheinlichkeit q wird die Rate λ_1 und mit $1 - q$ die Rate λ_2 verwendet.

Für die Ermittlung der stationären zeitunabhängigen Unverfügbarkeit konnte durch den Einsatz von DIS die Varianz um den Faktor 11 verringert werden.
Für ein größeres System mit 32 Komponenten (nur exponentielle Verteilungen) wurde bei der Berechnung der Unverfügbarkeit eine Varianzverringerung um den Faktor 12.8 erzielen. Für das gleiche System mit noch extremeren Parametern konnte überhaupt nur mittels Importance Sampling ein brauchbares Ergebnis erreicht werden.
In unserem Programm wird ein statisches Importance Sampling verwendet: Für jede Verteilung und jede Wahrscheinlichkeit des Modellsystems gibt es eine entsprechende IS-Größe, die für die Auslosung während der gesamten Simulation verwendet wird. Trotzdem ist es möglich, für viele Systeme DIS zu modellieren, indem der vom IS unabhängige Reschedulingmechanismus zusammen mit einer Abänderung des Modellsystems, welche die Erwartungswert nicht beeinflußt, verwendet wird. Diese Vorgehensweise wird im Abschnitt 5.4 erläutert.
Die Autoren geben für DIS eine Regel zur Wahl der IS–Parameter an: Falls ein Modell nur exponentiell verteilte Ausfälle und Reparaturen mit den Raten λ und μ enthält und aus n Komponenten besteht, so sollten alle Ausfallraten für IS mit einem Faktor r multipliziert werden, für den gilt:

$$\mu/10 \leq r(n-1)\lambda \leq \mu \tag{4.9}$$

Alle Ausfallraten zusammen kommen so in die Größenordnung der Reparaturrate. Auf diese Weise sollen die ermittelten Werte korrekt bleiben und die Varianzreduktion maximal werden.
Für Systeme mit verschiedenen Raten kann man entweder für jede Komponente einzelne Faktoren r bestimmen oder eine Art Mittelwert der Raten verwenden.

4.5 Regenerative Simulation

Zunächst erklären wir anschaulich, warum es notwendig ist, die *Regenerative Simulation* anzuwenden. Daraus ergibt sich dann die Funktionsweise dieses Verfahrens. Zum Schluß wird dargestellt, wie zwei wichtige Zuverlässigkeitskenngrößen damit ermittelt werden.
Ziel des Simulationsprogramms ist es, Kenngrößen durch Einsatz von Importance Sampling mit hoher Genauigkeit zu berechnen. Wie bereits erwähnt, kann IS direkt nur für Größen eingesetzt werden, die „klein“ sind, d.h. bei Wahrscheinlichkeiten klein gegen 1 und bei Zeiten klein gegen die simulierte Systemzeit. Verwendet man IS für Größen, die diese Bedingungen nicht erfüllen, so kann sich die Varianz der Größen erhöhen oder der berechnete Mittelwert wird sogar falsch (siehe: Kapitel 10 und [nicola90] sowie [goyal92]).
Weiter kann man wie folgt sehen, daß sich stationäre Kenngrößen nicht mittels der einfachen Form des Importance Sampling ermitteln lassen: Man betrachte als Beispiel die zeitunabhängige stationäre Verfügbarkeit U, d.h. den Anteil der

Zeit, in dem ein System INTAKT ist. Mittels normaler Simulation kann diese Größe wie folgt ermittelt werden: Zu irgendeinem Zeitpunkt der Simulation startet man eine Meßphase der Länge Δt und ermittelt in ihr die Unverfügbarkeit. Das macht man für viele Meßphasen und bildet dann den Mittelwert. Dieser Wert dürfte sich nicht ändern, wenn man z.B. die Länge der Meßphasen verdoppelt, da die stationäre Unverfügbarkeit U eine Verhältnisgröße ist.
Wenn man nun Importance Sampling verwendet, so muß für die Berechnung des i–ten Wertes der Unverfügbarkeit ein IS–Faktor f_i berechnet werden. Man nehme zunächst an, er habe zu Beginn der i-ten Meßphase den Wert $f_i = 1$. Im Laufe der Simulation wird nun f_i durch fortwährende Multiplikation mit Verhältnissen von Übergangswahrscheinlichkeiten (oder Dichten) berechnet. Am Ende der Meßphase wird die dort ermittelte i–te Unverfügbarkeit U_i mit f_i multipliziert und zur Mittelwertbildung herangezogen, das ist der Wert $U_i f_i$. Wenn man nun die Länge der Meßphase verdoppelt ($\Delta t' = 2\Delta t$), sollte sich auch U_i im Mittel nicht ändern ($U_i' \approx U_i$). Für die Berechnung von f_i' wurden aber nun ca. doppelt soviele Werte aufmultipliziert, der IS–Faktor sollte somit in der Größenordnung von f_i^2 liegen: $f_i' \approx f_i^2$. Da nun aber f_i sicherlich wesentlich von 1 verschieden ist (denn sonst hätte das IS ja keine Wirkung), ist somit $U_i f_i$ stark verschieden von $U_i' f_i'$, obwohl sie ja eigentlich ungefähr gleich sein sollten, weil aus ihnen derselbe Mittelwert berechnet werden soll.
Man könnte nun einwenden, daß die Festsetzung $f_i := 1$ zu Beginn der Meßphase falsch ist, da ja ihr Ausgangszustand unbekannt ist, dieser aber von der Wahl der Verteilungen abhängt und somit auch davon, ob IS verwendet wird oder nicht. Das ist sicher richtig. Nun stellt sich die Frage, welchen Ausgangswert man für f_i von welchem Zustand an nehmen soll. Als erste Möglichkeit bietet sich an, zu Beginn eines Simulationsdurchgangs, also zur Zeit $t = 0$, $f = 1$ zu setzen und diesen Faktor dann immer neu zu berechnen, bis der Durchgang beendet ist. Da aber die Unverfügbarkeit eine stationäre Größe ist, sollte man sie innerhalb eines Durchgangs zu verschiedenen Zeitpunkten ermitteln können. Der IS–Faktor wird aber wegen der dauernden Multiplikation entweder immer größer oder immer kleiner, womit eigentlich gleichberechtigte Werte von U_i mit verschiedenen Faktoren multipliziert würden, was auch nicht richtig sein kann. Man benötigt somit einen genau definierten Zeitpunkt, an dem die Berechnung einer Größe beginnt, der möglichst unabhängig von der Wahl der Wahrscheinlichkeitsverteilungen ist. Außerdem benötigt man einen ebenso genau definierten Endpunkt der Messung, wie oben bereits gezeigt wurde. Es bietet sich somit der Zustand, in dem alle Komponenten INTAKT sind, der sog. *regenerative Zustand*, als Anfangs- *und* Endpunkt an. Dieser Zustand ist im allgemeinen ohnehin der Startzustand bei $t = 0$. Man verwendet den Namen *Regenerative Simulation* für die Simulationen, bei denen genau ein Zyklus zwischen zwei regenerativen Zustandes betrachtet wird. Dieser Name rührt daher, daß das System nach Durchlaufen eines solchen Zyklus' wieder so gut wie vorher ist, d.h. sie sich

regeneriert hat.[6] Näheres zu dem Verfahren findet man in [smith55], [crane75] und [haas87].
Der Erwartungswert $\mathrm{E}(U)$ der Unverfügbarkeit läßt sich als Quotient aus den Erwartungswerten von Dauer des Zyklus $\mathrm{E}(T_C)$ und Dauer des Systemausfalls $\mathrm{E}(T_D)$ innerhalb des Zyklus' darstellen [nicola90]:

$$\mathrm{E}(U) = \frac{\mathrm{E}(T_D)}{\mathrm{E}(T_C)} \tag{4.10}$$

Beide Größen lassen sich nun einfach mittels Regenerativer Simulation ermitteln. Die Aufteilung auf zwei Größen ist deshalb günstig, weil sich $\mathrm{E}(T_C)$ schon mit geringem Aufwand mittels normaler Simulation ermitteln läßt. Eine Verwendung von Importance Sampling würde sogar eventuell die Genauigkeit verschlechtern. Der größte Teil der Rechenzeit kann für die Berechnung von $\mathrm{E}(T_D)$ mittels IS verwendet werden, was wegen $T_D \ll T_C$ hohe Varianzreduktionen bringt. In [goyal92] wird empfohlen, 10% der Rechenzeit für $\mathrm{E}(T_C)$ und den Rest für $\mathrm{E}(T_D)$ zu verwenden.
Auch die mittlere Zeit bis zum ersten Ausfall (MTTFF) läßt sich auf diese Weise berechnen. Man simuliert einen regenerativen Zyklus und schaut, ob das System ausfällt. Wenn ja, nimmt man den Ausfallzeitpunkt T_F als Meßwert. Wenn nein, fällt das System irgendwann später aus, man nimmt als Meßwert die Zykluszeit T_C. Um den so gemachten Fehler herauszurechnen, skaliert man den Wert mit der Wahrscheinlichkeit, daß der zweite Fall eintritt und erhält [nicola90]:

$$\mathrm{E}(\mathrm{MTTFF}) = \frac{\mathrm{E}(\min(T_F, T_C))}{\mathrm{P}(T_F < T_C)} \tag{4.11}$$

Auch hier ist es so, daß der „große“ Wert, d.h. $\min(T_F, T_C)$, mittels normaler Simulation und die Wahrscheinlichkeit mittels IS ermittelt werden sollten. Das ist auch der Grund, warum MTTFF als Quotient dargestellt und nicht durch direkte Simulation ermittelt wird: Importance Sampling wäre nicht anwendbar, da diese Größe nicht „klein“ ist.[7]
Der Nachteil dieses Verfahrens ist, daß es nur für Systeme einsetzbar ist, die eine regenerative Struktur zeigen. Insbesondere müssen also die Reparaturen so gut wie neu (AGAN) sein, damit der Zustand, in dem alle Komponenten wieder INTAKT sind, dem Startzustand entspricht. Anderfalls müssen erweiterte Verfahren verwendet werden, bei denen nicht nur der erste Zyklus „alle INTAKT $\rightarrow$ alle INTAKT“, sondern auch spätere Zyklen betrachtet werden.

[6] Man sagt, daß ein solches System eine *regenerative Struktur* zeigt.

[7] Im Fall einer regenerativen Struktur ist die mittlere Zeit bis zum ersten Ausfall gleich der mittleren Brauchbarkeitsdauer (MTTF), welche eine stationäre Größe ist.

Kapitel 5

Das Modell

Im folgenden beschreiben wir, wie Systeme aussehen, die das Programm `simsam` simulieren kann, d.h. welche Eigenschaften ein System haben kann, und wie sie sich auf die Simulation auswirken. Außerdem werden die Parameter vorgestellt, die für Importance Sampling änderbar sind, und es wird gezeigt, wie man die bereits vorgestellte Variante, das dynamische Importance Sampling, realisiert. Im Anhang G wird an einem Beispiel die Eingabe und Simulation eines größeren Systems vorgeführt.
Ein System besteht grundsätzlich aus drei verschiedenen Arten von Objekten: Aus Komponenten, aus Fehlertypen und aus Reparaturstationen. Beispielweise besteht das System „Kuli" aus den Komponenten „Mine", „Feder" und „Gehäuse" (vereinfachtes Modell). Als Fehler kann man sich z.B. „Mine leer", „Feder verklemmt" und „Gehäuse zerbrochen" vorstellen. Die ersten beiden Fehler können von den Reparaturstationen „Benutzer" bzw. „Kuli–Fachmann" behoben werden, der dritte Fehler wird als dauerhaft angesehen.

5.1 Komponenten

Bei einer Zuverlässigkeitssimulation, muß für jede Komponente zu jedem Zeitpunkt bekannt sein, ob sie INTAKT oder DEFEKT ist, d.h. sie hat einen *Zustand*. Wichtig ist nun, wie sich die Zustände der Komponenten auf den Zustand des Systems auswirken. Das wird durch ein sog. Blockdiagramm beschrieben, das der Benutzer der Simulation mittels des Programms `rbdedit` aus [löhr90a] eingeben kann. Der Aufbau von Blockdiagrammen wurde bereits im Kapitel 2 beschrieben.
Als Beispiel wird nun ein „Drei–Minen–Kuli" betrachtet. Er wird als funktionstüchtig angesehen, wenn mindestens eine der „Minen" sowie die „Hülle" und die „Feder" funktionieren (die Minen seien gleichfarbig). Es ergibt sich somit das Blockdiagramm in Abbildung 5.1
Als nächstes wird beschrieben, welche Daten der Benutzer für Komponenten

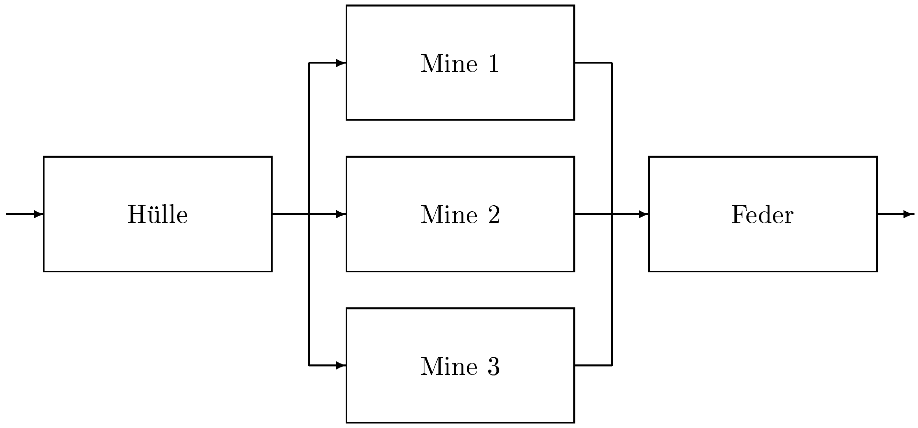

Abbildung 5.1: Ein Blockdiagramm

vorgeben kann. Dafür dienen die Programme `rbdedit` aus [löhr90a] (für die Komponenten selbst) und `mdledit` aus [ruenge92] (für die Simulationsdaten). Die Namen von Variablen werden im folgenden *kursiv* geschrieben, ggf. in Kleinschreibung.

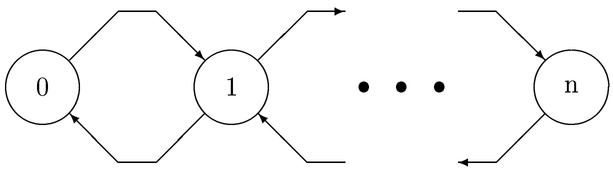

Abbildung 5.2: Modifikationsstufen einer Komponente

Jede Komponente hat, wie oben ersichtlich, einen *Namen* und, für die interne Verwaltung, eine *Nummer*. Das Ausfallverhalten der Komponenten wird durch Wahrscheinlichkeitsverteilungen bestimmt, welche als eine Reihe von sog. *Modifikationsstufen* (oder Graden) angeordnet sind, die — von 0 an beginnend —- durchnumeriert werden (siehe Abbildung 5.2). Jede Modifikation enthält eine *Verteilung* mit ihren *Parametern*. Zu Beginn einer Simulation befindet sich jede Komponente im Grad 0[1]. Falls nun der Ausfall einer anderen Komponente die

[1]Man sagt auch Modifikationsstufe 0, was ausdrückt, daß das Ausfallverhalten der Komponente eben nicht modifiziert wurde.

Komponente beeinflußt oder sie als Ersatzkomponente dient, kann sich die *Modifikationsstufe* erhöhen und bei Reparatur der ausgefallenen Komponente wieder erniedrigen. Da im allgemeinen eine Beeinflussung eine Komponente stärker belastet, ist es sinnvoll, wenn der Benutzer für höhere Grade Verteilungen mit größeren Ausfallraten wählt. Für den Wechsel der Modifikation während der Simulation kann man noch wählen, ob bei der Bestimmung der neuen Ausfallzeit die *vergangene Lebensdauer* berücksichtigt werden soll oder nicht.
Für die Ausfallverteilungen und alle anderen in der Simulation auftretenden Verteilungen können verwendet werden: Exponential–, Normal–, Weibull–, Gamma– und Gleichverteilung sowie die Angabe eines determinierten Zeitpunkts (entspricht einer Stufenfunktion). [2]
Falls nun bei einer Komponente ein Ausfall auftritt, so ist für diesen Ausfall einer von mehreren möglichen Fehlern verantworlich. Somit werden für jede Komponente *Fehlertypen* mit der *Wahrscheinlichkeit* ihres Auftritts angegeben.[3]. Die Summe dieser *Wahrscheinlichkeiten* muß für jede Komponente den Wert 1.0 ergeben. Weiterhin kann man für jeden *Fehlertyp* eine Reihe von Komponenten angeben, die durch den Auftritt des Fehlers sekundär *zerstört* werden, sowie Komponenten, die dadurch *beeinflußt* werden. Die sekundär *zerstörten* Komponenten fallen gleichzeitig mit der primär vom Ausfall betroffenen Komponente aus. Für die *beeinflußten* Komponenten wird die *Modifikationsstufe* um eins erhöht.
Weiterhin kann für jede Komponente definiert werden, ob sie als Ersatzkomponente eingesetzt werden kann. Parameter sind die *Reserveart* (KALT, WARM oder HEISS) und die Menge von *ersetzbaren Komponenten*, für die die Ersatzomponente eingesetzt werden kann. Immer wenn eine *ersetzbare Komponente* ausfällt und die *Reserveart* KALT oder WARM ist, wird die *Modifikationsstufe* der Ersatzkomponente um eins erhöht.
Falls eine Komponente in eine höhere Modifikationstufe gebracht werden soll, welche nicht existiert, so verbleibt sie in ihrem höchsten Grad.
Die Komponentendaten sind graphisch in Abbildung 5.3 dargestellt.

[2]Weitere Verteilungen können leicht implementiert werden, wie in Ahang F beschrieben wird.

[3]In der Realität lößt ein Fehler einen Ausfall aus. In dem Programm wird zuerst ein Ausfallzeitpunkt mit einer für alle Fehlertypen gleichen Verteilung ausgelost und dann der Fehlertyp.

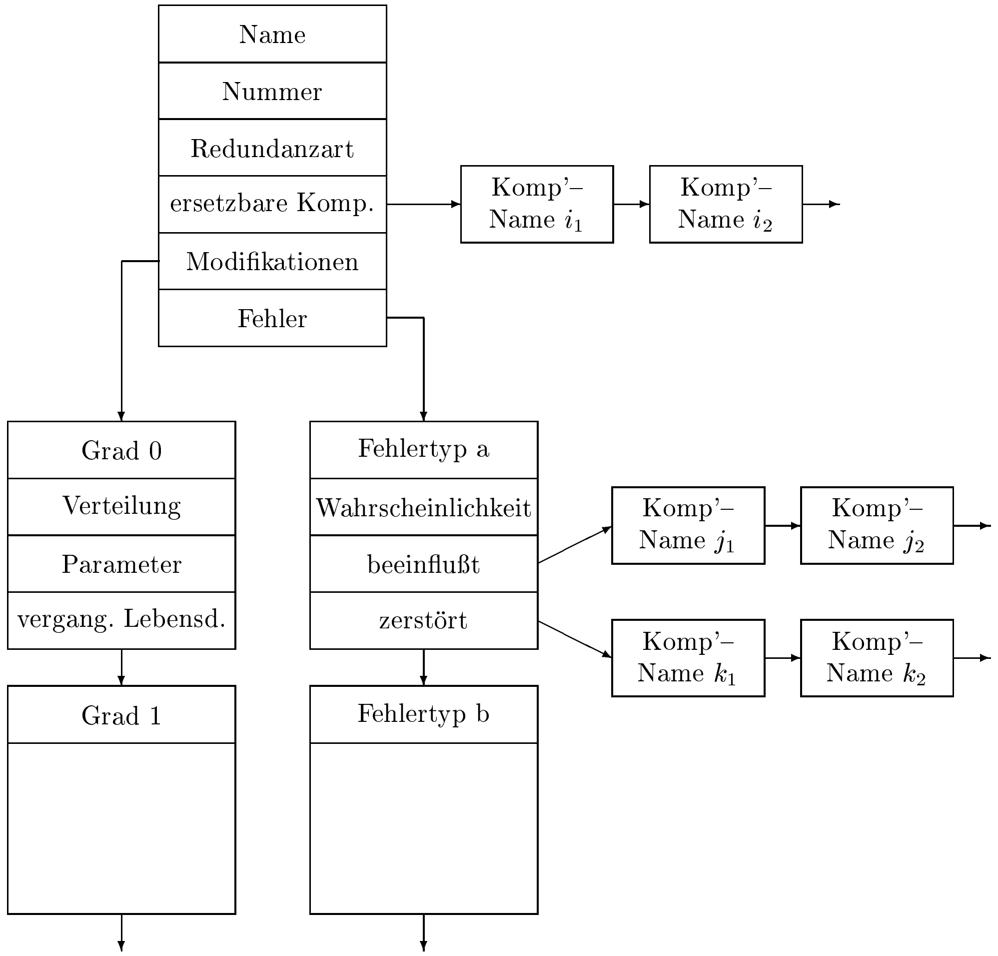

Abbildung 5.3: Komponentendaten

5.2 Fehlertypen

Jeder Fehlertyp (siehe Abbildung 5.4) hat einen *Namen*.
Ein aufgetretener Fehler kann nach einiger Zeit von alleine verschwinden, es gilt dann *Fehlerart* = TRANSIENT, anderfalls *Fehlerart* = PERMANENT.
Für TRANSIENTE Fehler gibt eine Wahrscheinlichkeits*verteilung* mit ihren *Parametern* die Dauer ihres Auftretens an.
Bei PERMANENTEN Fehlern wird unterschieden, ob sie *reparierbar* sind oder nicht. Bei jedem reparierbaren Fehler muß noch der Name einer *Reparaturstation* angegeben werden, die den Fehler beheben kann.

<table>
<tr><td colspan="3" align="center">Name</td></tr>
<tr><td colspan="3" align="center">Fehlerart</td></tr>
<tr><td colspan="2" align="center">PERMANENT</td><td align="center">TRANSIENT</td></tr>
<tr><td colspan="2" align="center">reparierbar</td><td rowspan="2" align="center">Verteilung</td></tr>
<tr><td align="center">nein</td><td align="center">ja</td></tr>
<tr><td></td><td align="center">Reparaturstat.</td><td align="center">Parameter</td></tr>
</table>

Abbildung 5.4: Daten eines Fehlertyps

5.3 Reparaturstationen

Sollen PERMANTE Fehler behoben werden, so ist dies Aufgabe einer Reparaturstation (siehe Abbildung 5.5). Wie alle Objekte eines Systems haben auch sie *Namen*.
Jede Station hat eine *Anzahl Instanzen*, das ist die maximale Anzahl von Reparaturen, die die Station gleichzeitig ausführen kann. Falls die Anzahl der durchzuführenden Aufträge größer oder gleich ist als die Variable *Anzahl Instanzen* angibt, gelangen neue Aufträge in eine Warteliste. Wie die Reihenfolge in dieser Liste organisiert ist, bestimmt der Parameter *Reparaturstrategie*. Es gibt folgende Strategien:

- FIFO: Die Ausfälle werden in der Reihenfolge ihres Auftretens bearbeitet.
- LIFO: Die Ausfälle werden in der umgekehrten Reihenfolge ihres Auftretens bearbeitet. Je neuer der Reparaturauftrag ist, desto eher wird er ausgeführt.
- RANDOM: Es wird eine zufällige Reihenfolge gewählt.

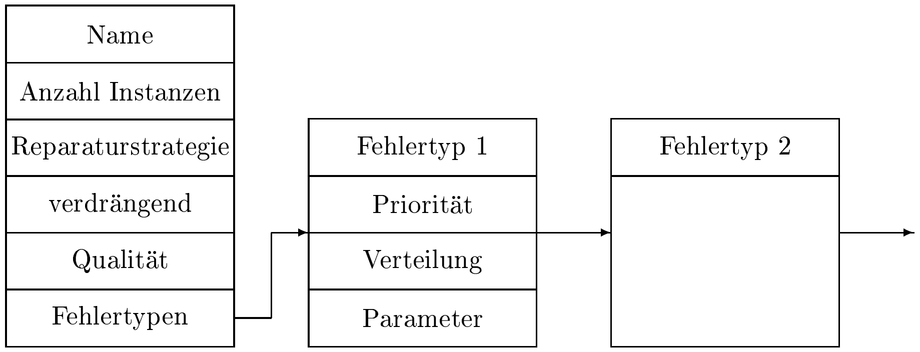

Abbildung 5.5: Daten einer Reparaturstation

- INFINITE SERVER: Es können beliebig viele Reparaturen gleichzeitig (ohne Zeitverlust) ausgeführt werden. Der Parameter *Anzahl Instanzen* und die Verwaltung einer Warteliste entfallen hier.

- PRIORITY: Jeder von der Station reparierbare Fehlertyp erhält eine *Priorität* von 1 (hoch) bis 4 (niedrig). Die Ausfälle werden in der Reihenfolge der *Prioritäten* der dabei aufgetretenen Fehler beseitigt. Fehlertypen mit gleichen *Prioritäten* werden nach FIFO bearbeitet.

- PROCESSOR SHARING: Auch hier repariert eine Station alle Fehler gleichzeitig, jeder Teil der Reparaturzeit wird aber gleichmäßig auf alle Aufträge verteilt. Beispiel: Es seien zwei Reparaturen auszuführen, wobei die eine 2 Stunden, die andere 3 Stunden benötige. Somit ist der erste Auftrag nach 4 Stunden erledigt; zu diesem Zeitpunkt ist jede Komponente 2 Stunden repariert worden. Die zweite Reparatur ist dann nach 5 Stunden beendet. Komplizierter ist es, wenn zwischenzeitlich ein weiterer Reparaturauftrag hinzukommt, z.B. nach 2 Stunden eine einstündige Reparatur. Zu diesem Zeitpunkt sind die beiden Komponenten eine Stunde bearbeitet worden. Nun sind die erste und die dritte Komponente nach weiteren drei Stunden wieder INTAKT, die zweite erst eine Stunde später. Bei jedem neuen Auftrag müssen somit die Reparaturzeiten neu berechnet werden.

- REPAIR TIME PRIORITY (RTP): Es werden die Komponenten zuerst bearbeitet, deren Fehler sich am schnellsten beheben lassen.

- INVERS RTP: Die langwierigsten Reparaturen werden zuerst durchgeführt.

Falls der Parameter *verdrängend* den Wert NEIN hat, werden alle begonnenen Reparaturen ohne Unterbrechung beendet, auch wenn ein Reparaturauftrag ent-

steht, der in der Warteliste vor einem bereits begonnenden Auftrag eingeordet würde. Anders ist es, wenn *verdrängend* = JA gilt. Es ist dann möglich, daß bereits begonnene Reparaturen gestoppt und (wieder) in die Warteliste eingeordnet werden. Z.B. bei der *Reparaturstrategie* LIFO wäre das bei jedem neuen Auftrag der Fall.
Die *Qualität* der Station gibt an, ob die Komponenten nach einer Reparatur so gut wie neu (AGAN = as good as new) sind oder nicht (NAGAN = not AGAN). Bei einer Reparaturqualität von AGAN werden die Modifikationsstufen der Komponenten nach einer Wiederinstandsetzung auf 0 gesetzt, sonst auf 1.
Für jede Station werden die Fehler, die sie beseitigen kann, in einer Liste aufgeführt. Für jeden *Fehlertyp* gibt eine *Verteilung* mit dazugehörigen *Parametern* die Reparaturdauer an. Außerdem ist noch die oben erwähnte *Priorität* enthalten, die für die Strategie PRIORITY benötigt wird.

5.4 Importance–Sampling–Parameter

Für Importance Sampling lassen sich alle Systemparameter ändern, welche Verteilungen oder Wahrscheinlichkeiten sind. Das sind die Verteilungen für Lebens–, Fehlerauftritts– sowie Reparaturdauern und die Auftrittswahrscheinlichkeiten der Fehler bei den Komponenten. Die gewählten IS–Werte werden während der gesamten Simulation verwendet. Wir haben somit ein statisches Verfahren realisiert.
Es wird nun gezeigt, wie man Dynamic Importance Sampling (DIS) nach [nicola90] trotz der statischen Struktur realisieren kann. Zuerst wird angenommen, daß sich die Komponenten nicht gegenseitig beeinflußen, wohl aber ggf. gegenseitig zerstören, und daß es keine Ersatzkomponenten gibt. Somit benötigt jede Komponente nur eine Modifikation (Grad 0).
DIS arbeitet ja so: Bei Ausfall einer Komponente K wird für diejenigen Komponenten, deren Defekt stark zu einem Systemausfall beitragen würde, eine neue Ausfallzeit bestimmt. Dazu wird eine IS–Verteilung mit erhöhter Ausfallrate verwendet. Falls es bis zur Reparatur der Komponente K keinen Ausfall gab, werden wieder die ursprünglichen Verteilungen verwendet.
In dem Rahmen des Programms wird das so realisiert: Im Modelleditor `mdledit` wird für jede Komponente eine Modifikationsstufe 1 eingegeben, die eine Kopie ihrer Modifikation 0 ist. Beide Modifikationen sollen die Alterung bei Rescheduling berücksichtigen. Außerdem wird für jede Komponente festgelegt, daß sie diejenigen Komponenten beeinflußt, mit denen sie gemeinsam in einem minimalen Schnitt enthalten ist. Auf diese Weise ändert sich der zu ermittelnde Erwartungswert nicht, da ja für die beiden Modifikationen dieselben Verteilungen verwendet werden. Die IS–Parameter werden dann in `simsam` wie folgt gesetzt: Alle Systemparameter werden übernommen, außer den Parametern der hinzugefügten Modifikationsstufen. Für sie werden die Ausfallraten erhöht, im

einfachsten Fall alle um einen konstanten Faktor, der z.B. nach der in [nicola90] gegebenen Regel gewählt werden kann.
Nun seien bereits im ursprünglichen System Beeinflussungen vorhanden. Hier darf man im Modelleditor für die beeinflußte Komponenten keine Modifikationsstufen und keine neuen Beeinflussungen hinzufügen. Es würde sich sonst das System ändern, denn die so bereits vorhandenen höheren Modifikationen haben sicherlich eine andere Ausfallcharakteristik als die niedrigeren. Aber das ist meistens kein Problem, da Komponenten, deren Ausfall andere beeinflußt, im allgemeinen im selben minimalen Schnitt wie die beeinflußten Komponenten enthalten sind. In diesem Fall werden im Simulationsprogramm nur die IS–Parameter der höheren Modifikationen gegenüber den normalen Systemparametern geändert.
Genauso wird mit Komponenten, die als Ersatzkomponente verwendet werden, verfahren, falls deren Reserveart KALT oder WARM ist.

Kapitel 6

Realisierung der Auswertung

Im Anhang C.1 beschreiben wir, wie aus den gesammelten Ergebnissen der Simulationsdurchgänge der statistische Mittelwert und die Varianz berechnet wird. Im folgenden wird erläutert, wie die Ergebnisse in Form von Vertrauensintervallen, Hypothesentests und Gütefunktionen dargestellt werden. Ausserdem wird gezeigt, wie man die dazu benötigten Werte auch für Quotienten von Kenngrößen berechnen kann.
Bei allen drei in dem Programm realisierten Darstellungsmethoden muß der Benutzer eine *Vertrauenszahl* β vorgeben. Sie gibt an, mit welcher Sicherheit die aus den Werten gewonnenen Aussagen als gültig angenommen werden können. Typische Werte sind 0.9, 0.95, und 0.99. Den Wert $1 - \beta$ nennt man *Irrtumswahrscheinlichkeit*.
Im folgenden wird angenommen, daß n Experimente durchgeführt wurden, um das Verhalten einer Kenngröße K zu ermitteln, welche gemäß einer (unbekannten) Verteilung mit der Dichtefunktion $f(k)$ verteilt ist. Als Ergebnis der Simulation erhält man Schätzwerte $\overline{K}$ des Erwartungswertes $\mu := \mathrm{E}(K)$ sowie S^2 der Varianz $\sigma^2 := \mathrm{Var}(K)$.
Im folgenden wird der *zentrale Grenzwertsatz* (z.B. [moesch83]) verwendet: Für $n \to \infty$ strebt die Verteilungsfunktion der Größe $\frac{\overline{K}-\mu}{\sigma/\sqrt{n}}$ gegen die Standardnormalverteilung $\mathrm{N}_{0,1}$ mit der Dichtefunktion

$$f_{0,1}(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-x^2/2\right). \tag{6.1}$$

Für praktische Anwendungen kann man diese Näherung ab ca. $n = 30$ verwenden. Wir gehen davon aus, daß bei allen Simulationen $n >> 30$ erfüllt ist. Deshalb wird die Näherung bei allen drei Darstellungsarten verwendet.

6.1 Vertrauensintervalle

Das *Vertrauensintervall* I:= $[\overline{K} - \Delta_m, \overline{K} + \Delta_m[$ wird so bestimmt, daß der tatsächliche Erwartungswert μ mit der Wahrscheinlichkeit β in I liegt. Die Grenzen des Vertrauensintervalls werden durch Invertierung der Standardnormalverteilung gewonnen (siehe Anhang C.2):

$$\Delta_m = \frac{\sigma}{\sqrt{n}} x_r := \frac{\sigma}{\sqrt{n}} \mathrm{N}_{0,1}^{-1}\left(\frac{\beta+1}{2}\right) \tag{6.2}$$

Die Berechnung von Δ_m gemäß dieser Formel setzt voraus, daß die Varianz σ bekannt ist. Das ist im allgemeinen aber nicht der Fall. Unter den oben genannten Bedingungen für die Verwendung der Normalverteilung, kann man aber statt σ auch den aus der Simulation ermittelten Wert S verwenden.
Die Darstellung eines Vertrauensintervalls im Simulationsprogramm ist in Abb. 6.1 gezeigt. Die im Fenster auftretenden Objekte werden im Abschnitt 7.5 erläutert.

Abbildung 6.1: Darstellung eines Vertrauensintervalls

6.2 Hypothesentests

Oft stellt man *Hypothesen* über den Mittelwert μ einer Kenngröße K auf. Nach einer Simulation möchte der Benutzer dann wissen, ob aufgrund der ermittelten Werte die Hypothesen abgelehnt werden sollen, oder nicht. Wenn man sie (noch) nicht ablehnt, so akzeptiert man sie (vorerst).
Man unterscheidet drei Arten von Hypothesen (H ist ein vom Benutzer vorgegebener Wert):

- *Zweiseitige Hypothesen* haben die Form $\mu = H$. Bei vorgegebenem Vetrauen β ist eine solche Hypothese zu akzeptieren, falls H in das oben berechnete Vertrauensintervall I fällt. Das bedeutet, daß mit Steigerung des Vertrauens β, die Wahrscheinlichkeit für die Ablehnung einer Hypothese

Abbildung 6.2: Beurteilung einer Hypothese

fällt. Daraus folgt, daß man dann lieber eine falsche Hypothese akzeptiert, ehe man eine richtige ablehnt.

- Eine *rechtsseitige Hypothese* ist $\mu \geq H$. Sie wird akzeptiert, falls die Wahrscheinlichkeit für das Zutreffen der Hypothese mindestens β ist, d.h. wenn $\mathrm{P}\left(\mu \geq H\right) \geq \beta$ gilt.
- Analog wird eine *linksseitige Hypothese* $(\mu < H)$ nicht abgelehnt, wenn $\mathrm{P}\left(\mu < H\right) \geq \beta$ gilt.

Wie man eine zweiseitige Hypothese behandelt, geht aus Abschnitt 6.1 hervor. Unter Verwendung der Abkürzungen $x_r = \mathrm{N}_{0,1}^{-1}(\beta)$ und $\Delta_r = x_r\sigma/\sqrt{n}$ wird eine Hypothese $\mu \geq H$ akzeptiert, falls gilt:

$$\overline{K} - H \geq \Delta_r \tag{6.3}$$

Der Beweis dazu findet sich in Anhang C.3
Analog dazu wird eine linkseitige Hypothese akzeptiert, wenn gilt (wobei $x_l = \mathrm{N}_{0,1}^{-1}(1-\beta)$ und $\Delta_l = x_l\sigma/\sqrt{n}$):

$$\overline{K} - H \leq \Delta_l \tag{6.4}$$

In der Praxis wird, wie oben bereits erwähnt, die Stichprobenvarianz S^2 anstatt der unbekannten Varianz σ^2 verwendet.
In Abb. 6.2 wird die Beurteilung von Hypothesen in `simsam` dargestellt.

6.3 Gütefunktionen

Im vorhergehenden Abschnitt wurde gezeigt, unter welchen Bedingungen man eine Hypothese akzeptiert, wobei die Wahrscheinlichkeit β für das Zutreffen

Abbildung 6.3: Darstellung einer Gütefunktion

der Hypothese vorgegeben war. Weiterhin kann man nun fragen, wie groß die Wahrscheinlichkeit $G(\mu)$ ist, daß, falls μ der tatsächliche Erwartungswert der Kenngröße K ist, eine Hypothese abgelehnt wird. Die Funktione $G(\mu)$ nennt man *Gütefunktion.*
Die Gütefunktionen für die rechtsseitige, linksseitige bzw. zweiseitige Hypothesen sind gegeben durch (x_r, x_l, x_m wie in den vorhergehenden Abschnitten 6.1 und 6.2):

$$
\begin{aligned}
G_r(\mu) &= N_{0,1}\left(x_r + \frac{H-\mu}{\sigma/\sqrt{n}}\right) && (6.5)\\
G_l(\mu) &= 1 - N_{0,1}\left(x_l + \frac{H-\mu}{\sigma/\sqrt{n}}\right) && (6.6)\\
G_m(\mu) &= 1 - N_{0,1}\left(x_m + \frac{H-\mu}{\sigma/\sqrt{n}}\right) + N_{0,1}\left(-x_m + \frac{H-\mu}{\sigma/\sqrt{n}}\right) && (6.7)
\end{aligned}
$$

Als Beispiel wird in Anhang C.4 die rechtsseitige Gütefunktion berechnet. In der Praxis wird die Größe σ durch S angenähert.
Wie das Programm eine zweiseitige Gütefunktion darstellt, ist in Abb. 6.3 zu sehen.

6.4 Auswertung von zusammengesetzten Kenngrößen

Bei der Verwendung der Regenerativen Simulation werden Kenngrößen durch Quotienten anderer Kenngrößen dargestellt. Es stellt sich nun die Frage, wie

man die oben dargestellten Formeln zu modifizieren hat. Formal: Gegeben seinen die Kenngrößen K_1 und K_2. Gesucht wird nun eine Darstellung von $\sigma/\sqrt{n}$ für die Kenngröße K_1/K_2 als Funktion der empirischen Standardabweichungen σ_1, σ_2 und der Stichprobenumfänge n_1, n_2.
Das Problem läßt sich nicht mit wahrscheinlichkeitstheoretischen Mitteln lösen. Zwar könnte man die Varianz von K_1/K_2 mittels der Kovarianz dieser Größen darstellen, diese steht aber im allgemeinen nicht zur Verfügung, da die beiden Größen zumeist in unterschiedlichen Simulationsläufen ermittelt werden. Damit sind üblicherweise auch die Umfänge der Stichproben n_1 und n_2 verschieden, so daß man keinen gemeinsamen Stichprobenumfang n (arithmetisches, geometrisches Mittel ?) festlegen kann.
Es bietet sich ein Rückgriff auf die Fehlerrechnung an : Hat man eine nicht direkt meßbare Größe G, die eine Funktion $G(m_1, \ldots, m_k)$ mehrerer Meßgrößen $m_1, \ldots, m_k$ ist, so kann man den maximalen Fehler ΔG mittels der Fehler $\Delta m_1, \ldots, \Delta m_k$ bestimmen:

$$\Delta G = \sum_{i=1}^{k} \left| \frac{\partial G}{\partial m_k} \right| \Delta m_k \tag{6.8}$$

Wendet man diese Formel auf den gesuchten Ausdruck $\sigma/\sqrt{n}$ für die kombinierte Kenngröße an, so ergibt sich:

$$\frac{\sigma}{\sqrt{n}} = \left| \frac{1}{\overline{K_1}} \right| \frac{\sigma_1}{\sqrt{n_1}} + \left| \frac{\overline{K_1}}{\overline{K_2}^2} \right| \frac{\sigma_2}{\sqrt{n_2}} \tag{6.9}$$

Kapitel 7

Benutzeroberfläche

In diesem Kapitel wird der Aufbau der Benutzeroberfläche geschildert. Es wird vorausgesetzt, daß der Leser mit fensterorientierten Systemen allgemein vertraut ist.
Zuerst wird der Aufbau des Programmfensters und die Funktion der Teilfenster erläutert. Dann wird im einzelnen beschrieben, wie man Modellsysteme einläd, ihre Importance–Sampling–Parameter wählt, Simulationsläufe spezifiziert und die Ausgabe der Ergebnisse steuert.
Die Systemvoraussetzungen für den Betrieb und die Installation des Programms sind in Anhang D aufgeführt.

7.1 Aufbau des Programmfensters und Benutzerführung

Nach Start des Simulationsprogramms erscheint das in Abbildung 7.1 auf Seite 45 dargestellt Programmfenster. In Abbildung 7.2 ist diese Benutzeroberfläche schematisch dargestellt.
Die Benutzung des Programms läuft in folgenden Schritten ab:

- Zuerst muß ein System geladen werden. Das erfolgt im obersten Teilfenster.

- Nun kann der Benutzer, wenn er möchte, in den unter dem obersten Fenster liegenden fünf Fenstern die Importance–Sampling–Parameter für sein System einstellen. Das erfolgt durch Eingabe der Parameter oder durch Einladen eines Datensatzes.

- Als nächstes kann man die Zuverlässigkeitskenngrößen, welche in der Simulation ermittelt werden sollen, im dritten Fenster von unten festlegen.

- Im darüberliegenden Teilfenster kann man die Parameter eines Simulationslaufs wie Anzahl der Durchgänge und die Importance Sampling Option wählen und die Simulation starten. Außerdem ist es möglich, sich Ergebnisse auch während der Simulation anzeigen zu lassen. Dabei muß man dann angeben, wie viele Simulationsdurchgänge zwischen zwei Erneuerungen der Anzeige liegen sollen.

- Nach Ende bzw. während einer Simulation werden die Ergebnisse der einzelnen Kenngrößen in dem Fenster angezeigt, wo sie auch spezifiziert wurden.

- Im zweiten Fenster von unten kann der Benutzer aussuchen, welche Kenngrößen auf welche Art graphisch dargestellt werden sollen. Die Ausgabe erfolgt im untersten Fenster, entweder auf Anforderung oder während der Simulation.

- Das Programm wird beendet, indem man den `Quit`–Button im obersten Fenster mit der linken Maustaste selektiert (s.u.).

In den folgenden Kapiteln werden die Funktionen der Teilfenster und ihre Bedienung im einzelnen geschildert.

Abbildung 7.1: Benutzeroberfläche des Simulationsprogramms

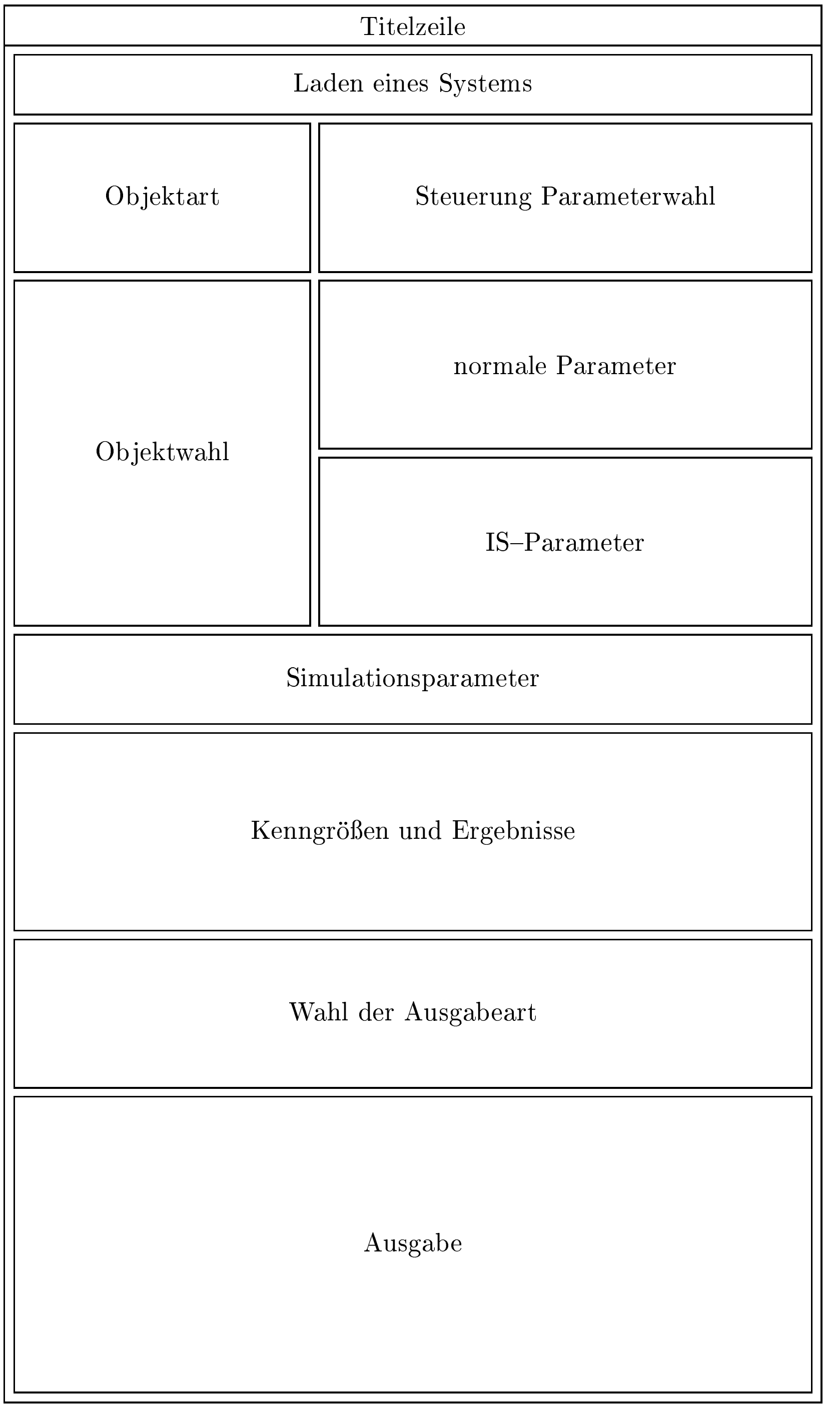

Abbildung 7.2: Schema der Benutzeroberfläche

7.2 Laden eines Systems und Beenden des Programms

Um ein Modellsystem simulieren zu können muß es erst mittels der Programme `rbdedit` und `mdledit` eingegeben und dort abgespeichert werden. Dann kann man es in das Simulationsprogramm einladen. Dazu dient das oberste Fenster des Programms (siehe Abb. 7.3).

Abbildung 7.3: Fenster zu Laden von Sysytemen

Im Texteingabefeld links, das mit `System Name:` markiert ist, wird der Name des gewünschten Systems — mit der Endung `.rbd` — eingegeben. Dazu muß sich der Mauszeiger in dem Fenster befinden. Wenn man nun mit dem Mauspfeil auf den `Load`–Button zeigt und die linke Maustaste drückt, d.h. wenn man den Button *selektiert* oder *anklickt*, erscheint zuerst ein kleines Fenster, indem man gefragt wird, ob man wirklich ein System einladen will. Diese Sicherheitsabfrage erfolgt, weil das Einladen alle Daten löscht, die sich momentan im Programm befinden. Wenn man den `no`–Button anklickt, kehrt das Programm in seinen ursprünglichen Zustand zurück. Bei Selektion des `yes`–Buttons wird das System geladen, sein Name wird in der Titelzeile angezeigt und alle Importance–Sampling–Parameter werden auf den normalen Systemwert gesetzt. Die Verwendung von Importance Sampling hätte dann keine Wirkung.
Falls beim Laden des Systems ein Fehler auftrat, wird er angezeigt, und das Programm kehrt in den Zustand zurück, den es ganz am Anfang hatte: Es sind keine Daten geladen, in der Titelzeile wird `no system` angezeigt.
Möchte man das Programm beenden, so muß man den `Quit` –Button anklicken. Es erfolgt dann eine Sicherheitsabfrage. Falls man sie mit „`confirm`“ beantwortet, wird das Programm abgebrochen.

7.3 Wahl der Importance–Sampling–Parameter

Für die Eingabe der IS Parameter dienen die fünf Fenster, die sich unterhalb des Fensters „Laden eines Systems“ befinden (Abb. 7.4). Man kann diese Parameter von Hand setzen oder einen früher gespeicherten Parametersatz einladen.
Für das IS kann man alle die Parameter ändern, die das zufällige Verhalten des Modellsystems beschreiben. Das sind:

- Für jede Modifikationsstufe einer Komponente ist eine Wahrscheinlichkeitsverteilung angegeben, die das Ausfallverhalten der Komponente be-

Abbildung 7.4: Fenster zum Eingeben der IS–Parameter

schreibt. Für IS können nun jeweils andere Verteilungen oder auch nur andere Verteilungsparameter gewählt werden.

- Für jede Komponente werden die Namen einiger Fehlerarten (oder Fehlertypen) angegeben, die bei einem Ausfall der Komponente auftreten können. Jede Fehlerart hat eine konstante Auftrittswahrscheinlichkeit. Diese Wahrscheinlichkeiten kann man für IS ändern.

- Für jeden reparierbaren Fehler, ist bei seiner zuständigen Reparaturstation eine Verteilung angegeben, die die Dauer der Reparatur beschreibt. Diese Verteilung kann im Rahmen von IS ebenfalls geändert werden.

- Gleiches gilt für die Verteilungen, die die Auftrittsdauer von transienten Fehlern angeben.

Das Fenster „Objektart“ dient nun dazu zu wählen, welche Art von IS–Parametern man als nächstes eingeben möchte. Das erfolgt durch Anklicken eines der vier folgenden Texte mit der linken Maustaste, ihre Reihenfolge entspricht der Reihenfolge im letzten Absatz:

- `Component Modifications`

- `Component Fault Types`

- `Repair Stations`

- `Fault Types`

Daraufhin erscheint im darunterliegenden Fenster eine Namensliste mit den Objekten der gewählten Objektart, welche im System enthalten sind. Ein Objekt

wird durch Anklicken seines Namens mit der linken Maustaste ausgewählt. Der Name wird dann in der Schriftart **Fettdruck** dargestellt. Zu Beginn ist immer das erste Objekt der Liste ausgewählt.
Falls es mehr Objekte gibt, als Namen in das Fenster passen, so kann man mit dem am linken Rand angebrachten *Rollbalken* (engl. *scrollbar*) den dargestellten Ausschnitt verändern: Klickt man einen der kleinen Pfeile am oberen oder unteren Ende des Balkes mit der linken bzw. rechten Maustaste an, so wandert der Ausschnitt eine Zeile nach unten bzw. oben. Analog kann man den Ausschnitt abschnittsweise verändern, indem man das Feld zwischen den beiden Pfeilpaaren anklickt. Falls man in diesem Bereich die mittlere Maustaste betätigt, so wird ein Ausschnitt angezeigt, dessen Lage in der gesamten Liste der Position entspricht, die man in dem Feld angeklickt hat.
Die Daten des ausgewählten Objektes werden in den beiden Fenstern rechts vom Objektwahlfenster angezeigt. Im oberen Fenster sind die Systemdaten, also eine Wahrscheinlichkeitsverteilung mit Namen und Parametern oder ein Name eines Fehlers mit dessen Auftrittswahrscheinlichkeit, angegeben. Diese Daten lassen sich nicht verändern. Im unteren Fenster sind die entsprechenden Daten für IS dargestellt. Diese Parameter sind änderbar. Am Anfang haben sie aber die gleichen Werte wie die Systemparameter, welche auch *normale* Parameter genannt werden.
Oft ist es so, daß zu einem Objekt mehrere Parametersätze gehören: Eine Komponente kann mehrere Modifikationen und Fehlertypen haben und eine Reparaturstation kann mehrere Fehlertypen reparieren. In diesem Fall sind die Parametersätze in Form von Listen angeordnet. Man kann dann mittels zweier Buttons, welche sich im Fenster zur Steuerung der Parameterwahl befinden, die Liste durchlaufen. Mit `prev`, geht man ein Element zurück und mit `next` ein Element vor. Wenn man an einem Listenanfang steht, ist der `prev`–Button nicht sichtbar, gleiches gilt am Ende einer Liste für den `next`-Button.
Nun wird erklärt, wie man die IS–Parameter ändern kann (im Fenster gleichen Namens).
Falls eine Verteilung dargestellt wird, so wird in der obersten Zeile die Nummer der Modifikation, der Name eines reaparierbaren Fehlertyps oder nichts (bei transienten Fehlern) angezeigt. Darunter steht der Name der augenblicklich gewählten Verteilung rechts vom Wort `Distribution`. Dazwischen erscheinen zwei kleine gebogene Pfeile. Diese Anordung heißt *Choice*. Unterhalb der Verteilung werden ihre Parameter mit Namen und augenblicklichen Werten angezeigt. Wenn man das Choice mit der linken Maustaste anklickt, wird eine andere Verteilung mit den zugehörigen Parametern, welche dann alle auf 0 gesetzt sind, sichtbar. Durch mehrfaches Anklicken des Choice kann man so die gewünschte Verteilung anwählen. Um Zeit zu sparen kann, man auch das Choice mit der rechten Taste anklicken. Es erscheint dann ein Menü, aus dem man die gewünschte Verteilung aussuchen kann. Es stehen bis jetzt folgende Verteilungen zur Auswahl: (Ihre mathematischen Definitionen sind im Kapitel 8.5

angegeben)

- `Exponential`: Die Exponentialverteilung mit dem Parameter Ausfallrate (`Rate`).
- `Weibull`: Die Weibullverteilung hat die Parameter Ausfallrate `Rate` und einen exponentiellen Formparameter `Exponent`.
- `Normal`: Die Parameter der Normalverteilung sind Mittelwert `Mean` und Standardabweichung `Stand. Dev.`.
- `Gamma`: hat die Parameter Ausfallrate `Rate` und einen Formparameter `Shape`.
- `Uniform`: Bei der Gleichverteilung werden der linke Rand `Left Border` und der rechte Rand `Right Border` des Intervalls angegeben, in dem die Auftrittswahrscheinlichkeit größer als 0 ist.
- `Deterministic`: Bei der deterministischen Verteilung wird der Zeitpunkt `Event Time` angegeben, an dem das Ereignis stattfinden soll.

Um die Verteilungsarameter zu ändern, muß man sie zuerst mit der linken Maustaste anklicken. Es erscheint dann ein kleines schwarzes Dreick unter dem Wert, welches die aktuelle Cursourpostion anzeigt[1]. Mittels der Tastatur kann man nun einen Wert eingeben, falls sich der Mauspfeil weiterhin in dem Fenster "IS Parameter" befindet. Die `Return`–Taste braucht nach der Eingabe nicht betätigt zu werden.
Analog erfolgt die Eingabe der Auftrittswahrscheinlichkeiten für die Fehler, die bei einer Komponente auftreten können. Der Wert der Wahrscheinlichkeit wird unterhalb des Fehlernamens hinter dem Wort `Probability` dargestellt (siehe Abb. 7.5).

Abbildung 7.5: Eingabe einer Fehlerauftrittswahrscheinlichkeit

Achtung: Als Parameterwerte für die Verteilungen sind, außer beim Mittelwert (`Mean`) der Normalverteilung, nur Werte größer als Null erlaubt. Außerdem muß

[1] Wenn der Mauszeiger sich nicht in dem Fenster befindet, erscheint eine kleine Raute

die Summe der Auftrittswahrscheinlichkeiten der Fehler einer Komponente den Wert 1.0 ergeben. Falls man bei diesen Eingaben etwas falsch gemacht hat, so erscheint eine Fehlermeldung, wenn man eine Simulation starten will.
Die eingegebenen Parameter kann man abspeichern, indem man im Fenster „Steuerung Parameterwahl“ hinter dem Wort `IS Parameter Set Name` einen Namen für den Datensatz eingibt und den `Save`–Button selektiert. Diese Daten werden dann in dem Verzeichnis gespeichert, in dem auch die restlichen Systemdaten stehen. Analog kann man durch Anklicken des `Load`–Buttons einen Datensatz einladen. Achtung: Wenn beim Einladen eines Datensatzes ein Fehler auftritt, werden alle IS–Parameter auf den Wert der normalen Systemparameter gesetzt.

7.4 Spezifikation eines Simulationslaufs

Die Spezifikation eine Simulationslaufs erfolgt in den zwei Fenstern „Simulationsparameter“ und „Kenngrößen und Ergebnisse“, welche in Abbildung 7.6 dargestellt sind.

Abbildung 7.6: Fenster zum Spezifizieren eines Simulationslaufs

Im unteren Fenster kann man die Zuverlässigkeitskenngrößen angeben, die während einer Simulation ermittelt werden. Am linken Rand des Fensters, neben dem Rollbalken, ist für jede Kenngröße ein Kästchen. Wenn man dieses Kästchen mit der linken Maustaste anklickt, kann man ein Häkchen ein– bzw. ausschalten. Falls das Häkchen vorhanden ist, bedeutet das, daß die rechts angegebene Kenngröße beim nächsten Simulationslauf berechnet wird. Man kann also in einem Lauf mehrere Größen gleichzeitig ermitteln.
Rechts neben dem Kästchen ist ein Choice, mit dem man die Kenngröße einstellen kann. Je nachdem, welche Größe gewählt wird, erscheinen noch weiter rechts bis zu zwei Parameter, die die Kenngröße genauer bestimmen. Es gibt bis jetzt folgende Größen:[2]

[2] Wie man weitere Kenngrößen implementieren kann wird in Abschnitt 9.3 beschrieben.

- `--` bedeutet: keine Kenngröße.
- Die Systemunzuverlässigkeit `Unreliability` zur Zeit `Time`, welches der Parameter dieser Größe ist.
- Die (Mean) `Down Time` ist die mittlere Zeit, die sich das Gesamtsystem während eines regenerativen Zyklus' im Zustand DEFEKT befand (ohne Parameter).
- Die mittlere Dauer eines regenerativen Zyklus' ist die (Mean) `Cycle Time`.
- `min(T_F,T_C)` ist das Minimum aus Zykluszeit und Zeit für den ersten Ausfall des Systems.
- `P(T_F<T_C)` ist die Wahrscheinlichkeit, daß ein Systemausfall vor dem Vollenden eines regenerativen Zyklus' stattfindet.
- Die Punktunverfügbarkeit `Point Unav.` ist die Wahrscheinlichkeit, daß das System zur Zeit `Time` (1.Parameter) DEFEKT ist.
- Alle oben angeführten Systemgrößen stehen auch für einzelne Komponenten zur Verfügung. Dem Namen der Größe ist dabei `(K)` angehängt. Als Parameter kommt dann zu jeder Kenngröße der Name der betrachteten Komponente `K` hinzu. [3]

In der zweiten Zeile jeder Kenngröße stehen die bereits ermittelten Ergebnisse. Links erscheint die Anzahl der Meßwerte `n`, daneben der daraus ermittelte Mittelwert `Mean` und rechts die Stichprobenvarianz `Variance`. Achtung: Immer wenn man eine Kenngröße oder einen ihrer Parameter ändert, werden die gespeicherten Ergebnisse gelöscht, da sie zu der nicht abgeänderten Kenngröße gehörten.

Im Fenster oberhalb der Kenngrößen kann ein Simulationslauf eingestellt und gestartet werden. In der ersten Zeile kann man eingeben, wieviel Durchgänge die Simulation laufen soll, d.h. wie viele Meßwerte für jede Kenngröße ermittelt werden sollen. Daneben kann man mit einem Choice aussuchen, ob die Simulation mit oder ohne Importance Sampling durchgeführt werden soll: `Importance Sampling: on/off`.

Weiterhin ist es möglich, daß während eines Simulationslaufs die Ergebnisse aktualisiert und graphisch dargestellt werden. Dazu muß man bei `Graphic Update: yes` wählen. Rechts daneben erscheint dann ein Texteingabelement `Cycle Length`, in dem man eingeben kann, in welchem Abstand, gemessen in Anzahl Durchgängen, die Ausgabe erfolgen soll. Wie die graphische Ausgabe spezifiziert wird, erläutern wir im folgenden Abschnitt.

[3] Diese Größen stehen für Subsysteme zur Zeit noch nicht zur Verfügung, da vom Blockschaltbildeditor noch keine Strukturfunktionen von Subsystemen geliefert werden.

Wenn man den `Run`–Button mit der linken Maustaste selektiert, so startet die Simulation. Für alle mit einem Häkchen versehenen Kenngrößen werden soviele *zusätzliche* Werte berechnet, wie die Anzahl der gewählten Durchgänge angibt. D.h. die Ergebnisse aus vorherigen Simulationsläufen bleiben erhalten. Man kann so sukzessiv immer neue Durchgänge starten, bis man z.B. mit der Genauigkeit einer Zuverlässigkeitskenngröße zufrieden ist.

7.5 Ausgabe

Im zweiten Fenster von unten spezifiziert man, welche Ergebnisse auf welche Art im untersten Fenster dargestellt werden sollen (siehe Abb. 7.7 und Kapitel 6).

Abbildung 7.7: Fenster zur Ausgabe der Ergebnisse

In der obersten Zeile links kann man wählen, ob eine einzelne Kenngröße oder der Quotient zweier Kenngrößen dargestellt werden soll. Es sei daran erinnert, daß die Darstellung stationärer Größen mittels regenerativer Simulation in der Form von Quotienten erfolgt und zwar: MTTFF = $\min(T_F, T_C)/\mathrm{P}(T_F < T_C)$ sowie U = down time/cycle time. Je nach der gewählten Darstellungsart gibt man dann rechts die Nummer(n) einer oder zweier Größe(n) aus dem Kenngrößenfenster ein.
In der zweiten Zeile kann man mittels eines Choice die Darstellungsart wählen. Es gibt folgende Möglichkeiten, für die man jeweils einen Parameter `Confidence` =: β einzugeben hat:

- `Confidence Interval`: Es wird ein Intervall angezeigt, in dem der Mittelwert der jeweiligen Kenngröße mit der Wahrscheinlichkeit β liegt. Weiterhin hat man die Größe des dargestellten Bereichs, in den das Vertrauensintervall eingezeichnet wird, mittels `min.Value` und `max.Value` anzugeben.

- `Test of Hypothesis`: Es wird angezeigt, ob eine Hypothese unter dem gegebenen Vertrauenswert β verworfen wird oder nicht. Man muß mittels eines Choices die Art der Hypothese `Test Side:left/right/two` und den Wert der Hypothese `Hypothesis` eingeben.

- `Inv. Oper. Charact.`: Es wird die Gütefunktion einer Hypothese (Eingabe `Hypothesis`) über einem Intervall (Eingabe `min.Value` und `max.Value`) angezeigt.

Wenn man den Button `Show` mit der linken Maustaste anklickt, wird die so spezifizierte Darstellung im untersten Fenster angezeigt, soweit für die angegebenen Kenngrößen schon Ergebnisse vorliegen. Ein andere Möglichkeit ist, die Anzeige während der Simulation aktualisieren zu lassen.

Kapitel 8

Realisierung der Verteilungen

In diesem Kapitel werden die mathematische Methoden beschrieben, die im Zusammenhang mit der Darstellung von Wahrscheinlichkeitsverteilungen für die Realisierung des Programms verwendet wurden. Es wird erklärt, wie man aus der Dichtefunktion die Verteilungsfunktion erhält und die Verteilungsfunktion invertiert. Dann wird gezeigt, wie man Pseudozufallszahlen erzeugen kann, und wie man erreicht, daß diese Zahlen einer bestimmten Verteilung gehorchen. Es wird das Verfahren vorgestellt, mit dem Ereignisse rescheduled werden. Zum Abschluß wird erläutert, wie die Verfahren für die im Simulationsprogramm auftretenden Verteilungen eingesetzt wurden.

8.1 Verteilungsfunktionen

Voraussetzung ist, daß eine Dichtefunktion f durch eine Gleichung gegeben sei. Wie im Anhang B gezeigt wird, muß man die Dichtefunktion integrieren, um die Verteilungsfunktion zu erhalten. Am günstigen ist es, wenn man das auftretende Integral für jedes Argument der Verteilungsfunktion analytisch berechnen kann. Man erhält damit auch für die Verteilungsfunktion eine Formel.
Falls die Dichtefunktion nicht integrabel ist, bietet es sich an, das Integral für jedes während der Simulation auftretende Argument numerisch zu berechnen. Bei der einfachsten Methode wird das Intervall, über dem integriert werden muß, in kleine Teilintervalle konstanter Größe eingeteilt und das Integral durch eine Summe ersetzt. Die genaue Realisierung des Verfahrens ist in Anhang E.1 geschildert.
Der Vorteil dieser Methode ist, daß sie für jede Dichtefunktion anwendbar ist, auch wenn sie über viele Parameter verfügt. Ihr Nachteil ist der hohe Rechenzeitbedarf, welcher mit der Genauigkeit ansteigt. Für die Simulationen komplexer Systeme ist dieses Verfahren nicht akzeptabel.
Als Ausweg bietet sich folgendes Verfahren [jordan78] an, welches aber nicht völlig universell einsetzbar ist: Die Idee ist, eine nicht geschlossen darstellba-

re Funktion h mittels einer Linearkombination von Funktionen $\phi_1, \ldots, \phi_n$ zu approximieren:

$$h(x) \approx \sum_{k=1}^{n} c_k \phi_k(x) =: \Phi(x) \tag{8.1}$$

Man geht davon aus, daß das Funktionensystem der ϕ_k gegeben ist und sucht nun eine Bedingung für die Wahl der Linearfaktoren $c_1, \ldots, c_n$. Dazu betrachtet man die Funktion h an $N \geq n$ Stellen $x_1, \ldots, x_N$ und fordert nun, daß die Summe der Fehlerquadrate von Φ zu h an diesen Stellen minimal wird. Wie man die Linearfaktoren berechnet wird im Anhang E.2 dargestellt.
Das Verfahren, das den Namen *diskrete Gaußsche Fehlerquadratmethode* trägt, hat den Vorteil, daß es sehr schnell ist. Außerdem kann man mit ihm auch Funktionen approximieren, von denen nur einige Werte, z.B. Meßwerte, bekannt sind. Seine Nachteile: Die Genauigkeit hängt stark von der Wahl der Stützstellen und des Funktionensystems ab. Günstigerweise wählt man solche Funktionen ϕ_k, die schon ähnlich wie die Zielfunktion h geformt sind. Besonder schwer wiegt: Funktionen, die stark von Parametern abhängen, d.h. die sich nicht durch eine Transformation auf eine parameterlose Form bringen lassen, können mit diesem Verfahren nicht dargestellt werden. Ein Beispiel einer solchen Funktion ist die Gammaverteilung.

8.2 Invertierung von Verteilungsfunktionen

Auch für die Umkehrung einer Verteilungsfunktion kann ggf. die diskrete Fehlerquadratmethode verwendet werden.
Falls das nicht möglich ist, es aber eine schnelle Realisierung der Verteilungsfunktion selbst gibt, bietet sich folgendes *Verschachtelungsverfahren* an: Man berechnet die Verteilungsfunktion an irgendeiner Stelle x_K. Je nachdem ob $F(x_K)$ größer oder kleiner als der gesuchte Wert p ist, wird x_K um einen kleinen Betrag verkleinert oder vergrößert. Dieses Vorgehen wird so lange wiederholt, bis man genügend nahe an den tatsächlichen Wert $x = F^{-1}(p)$ herangekommen ist. Die Suchschrittweite wird dabei immer dann verringert, wenn man den Wert x passiert, d.h. wenn sich das Vorzeichen von $F(x_K) - p$ ändert. Das Verfahren wird ausführlich im Anhang E.3 beschrieben.
Falls bereits die Verteilungsfunktion nur über numerische Integration ermittelt werden kann, würde die Verwendung des Verschachtelungsverfahrens für die Berechnung der Umkehrfunktion den Zeitbedarf noch um eine Potenz erhöhen. Es wird deshalb die „umgekehrten Integration“ angewendet. Dabei werden von einem Startwert $t = a$ an (hier $a = 0$, da nur Zeitverteilungen gebraucht werden) mit der Schrittweite $\Delta > 0$ die Werte $f(t) * \Delta$ so lange aufsummiert bis der Wert p erreicht wird (Siehe Anhang E.4).

Auch dieses Verfahren läßt sich durch die Wahl einer kleinen Schrittweite Δ sehr genau machen, ist aber viel zu langsam und wird deshalb nur im Notfall (hier: für eine Form der Gammaverteilung) verwendet.

8.3 Auslosung von Zufallszahlen

Bei der Simulation von fehlertoleranten Systemen ist es nötig, für die einzelnen Komponenten Ausfall– und Reparaturzeiten auszulosen. Ganz allgemein geht es darum, Zahlen zufällig zu ermitteln, so daß sie einer bestimmten Verteilung mit der Dichtefunktion f bzw. mit der Verteilungsfunktion F gehorchen. Dieses Auslosen der Zahlen erfolgt im allgemeinen zweistufig: Erst werden mit Hilfe von Zufallszahlengeneratoren gleichverteilte ganze Zahlen aus einem Bereich $[0, m-1]$ ausgelost, die dann so transformiert werden, daß die gewünschte Verteilung realisiert wird.
Einige Möglichkeiten für die Realisierung eines Zufallszahlengenerators sind in Anhang E.5 dargestellt.
In `simsam` wird ein Zufallszahlengenerator verwendet, der bei dem C–Compiler der Firma Sun mitgelieferten wurde. Er erzeugt Zufallszahlen aus dem Bereich $[0, 2^{31}-1]$ ($m = 2^{31}$) und hat die maximale Periode m.
Der Ausgangspunkt der Realisierung aller Verteilungen ist die $(0,1)$–Gleichverteilung. Die n-te gleichverteilte Zufallszahl u_n wird dadurch erzeugt, daß die n-te Zufallszahl durch die Größe des Zufallszahlenbereichs m geteilt wird:

$$u_n := x_n/m \tag{8.2}$$

Um nun aus einer gleichverteilten Zufallszahl u eine Zahl zu gewinnen, die einer bestimmten Verteilung gehorcht, gibt es einige allgemeine Methoden, die für alle Verteilungsfunktionen funktionieren, und viele spezielle. Im Anhang schildern wir die allgemeinen Verfahren *Inversionsmethode* (Anhang E.6) und *Verwerfungsmethode* (Anhang E.7) . Falls das Inverse einer Verteilungsfunktion in geschlossener Form zur Verfügung steht, ist die Inversionsmethode sehr schnell. Für andere Verteilungen, für die man nur die Dichtefunktion f hat, muß die Verwerfungsmethode verwendet werden.
Spezielle Verfahren werden im Kapitel 8.5 für die Normalverteilung und für die Gammaverteilung aufgeführt.

8.4 Rescheduling

In jedem technischen System kommt es vor, daß sich die Belastung einer Komponente ändert und damit auch ihr Ausfallverhalten. Beispiel: Das Reserverad

eines Autos altert auch, während es im Kofferraum liegt. Sobald es eingesetzt wird, steigt seine Belastung, d.h. seine Ausfallrate wird größer.
Nun beschreiben wir, wie solche Änderungen in dem Programm simuliert werden (das Verfahren arbeitet so wie die in [bähr91] geschilderte Methode): Es sei für eine Komponente zur Zeit t_0 ein Ereignis mit der Auftrittszeit $t_1 := t_0 + t_L$ ermittelt worden. Im folgenden wird angenommen, daß dies der Ausfall einer Komponente sei. Die Zeit t_L ist dann die Lebensdauer der Komponente. Der Wert t_L wurde gemäß einer Verteilung mit der Dichte f und der Verteilungsfunktion F ausgelost. Zur Zeit $t_0 \leq t' \leq t_1$ ändere sich das Ausfallverhalten der Komponente. Dieses Verhalten werde nun durch die Dichte g mit Verteilungsfunktion G beschrieben. Das Auslosen einer neuen Ausfallzeit (allg. Ereigniszeit) t_1' wird *Rescheduling* genannt.
Bei `simsam` gibt es nun zwei Möglichkeiten: Im ersten Fall wird die bereits eingetretene Alterung der Komponente nicht berücksichtigt. Es wird einfach eine Lebensdauer t_L' gemäß der Dichte g ausgelost und als neue Ausfallzeit $t' + t_L'$ genommen. Die tatsächliche Lebensdauer der Komponente ist dann $t' + t_L' - t_0$. Beim Einsatz von Importance Sampling werden Dichte- und Verteilungsfunktion benötigt. Bei Nichtberücksichtigung der Alterung hat die Verteilungsfunktion H für die Lebensdauer t_L der Komponente vor und nach t' folgende Gestalt: (Dabei ist $F_{t'} := F(t' - t_0)$ die Wahrscheinlichkeit, daß die Komponente bis zum Zeitpunkt t' ausfällt.)

$$H(t_L) = \begin{cases} F(t_L) & f\ddot{u}r \quad t_L \leq t' - t_0 \\ F_{t'} + [1 - F_{t'}]G(t_L - (t' - t_0)) & f\ddot{u}r \quad t_L > t' - t_0 \end{cases} \tag{8.3}$$

Durch Differenzieren erhält man die Dichtefunktion h:

$$h(t_L) = \begin{cases} f(t_L) & f\ddot{u}r \quad t_L \leq t' - t_0 \\ [1 - F_{t'})]\, g(t_L - (t' - t_0)) & f\ddot{u}r \quad t_L > t' - t_0 \end{cases} \tag{8.4}$$

Dieses Verfahren ist einfach, aber etwas unrealistisch. Bei der Berücksichtigung der Alterung wird wie folgt vorgegangen:
Der bereits erfolgte Verschleiß der Komponente entspricht dem Wert $F_{t'}$. Falls t' nahe bei t_0 liegt , ist die Komponente nicht so stark gealtert wie bei größeren t', was mit einem kleineren Wert von $F(t' - t_0)$ übereinstimmt. Für $t \geq t'$ soll nun statt F die Verteilung G verwendet werden, z.B. bei der Berechnung des IS–Faktors. Falls man aber bei $t = t'$ auf G zugreift, erhält man den Wert $G(t' - t_0)$ für den Alterungswert, was sicherlich inkonsistent ist. Es wird deshalb eine Korrekturzeit t_G eingeführt, die so gewählt wird, daß $G(t' + t_G - t_0) = F(t' - t_0)$ gilt. Das entspricht einer Verschiebung der Verteilungsfunktion G um den Betrag $-t_G$ nach rechts auf der Zeitachse (siehe Abbildung 8.1). Diese Korrekturzeit wird nun bei allen Verteilungen verwendet: Vor der Neuauslosung hatte sie den Wert t_F für die Verteilungsfunktion F.[1] Falls die Auslosung für

[1]Entsprechend gilt $F_{t'} := F(t' + t_F - t_0)$.

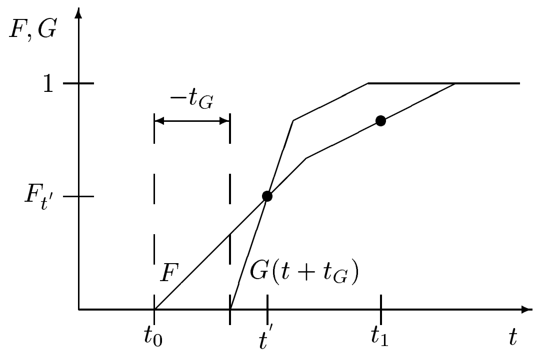

Abbildung 8.1: Situation zur Reschedulzeit

das Ereignis mittels F noch nicht rescheduled war, gilt $t_F = 0$, was im folgenden angenommen wird. Somit lautet die Bedingung:

$$\begin{aligned} G(t' + t_G - t_0) &= F_{t'} \\ \Leftrightarrow \qquad t_G &= G^{-1}(F_{t'}) + t_0 - t' \end{aligned} \tag{8.5}$$

Man beachte daß $t_G < 0$ gilt, wenn F schneller den Wert $F_{t'}$ erreicht als G, d.h. im Mittel eine höhere Ausfallrate hat.

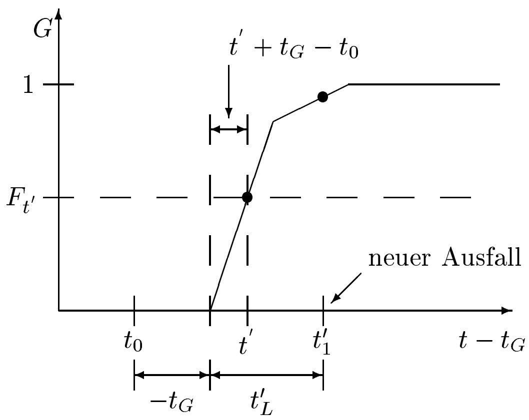

Abbildung 8.2: Auslosung des neuen Zeitpunkts

Für den neuen Ausfall kommen nur Werte von G zwischen $F_{t'}$ und 1 in Frage (siehe Abbildung 8.2[2]). Die neue Ausfallzeit t_1' wird nun durch Inversion von G

[2]In der Abbildung wird als x–Achse nicht t sondern $t - t_G$ verwendet, da es für die Erklärung zunächst nicht darauf ankommt, zu welcher Zeit was passiert, sondern an welcher Stelle der Verteilungsfunktion man sich befindet.

mittels eines in $[F_t, 1]$ gleichverteilten Wertes ermittelt: Falls u ein mittels einer $(0, 1)$–Gleichverteilung ausgeloster Wert ist, ist t'_L die neue Zeitdauer für G:

$$t'_L = G^{-1}(u \cdot (1 - F_{t'}) + F_{t'}) \tag{8.6}$$

Die neue Ausfallzeit t'_1 ergibt sich von der aktuellen Zeit t' an durch Addition der Differenz von t'_L und der Entfernung des aktuellen Zeitpunkts vom Beginn der Verteilung (auf der t–Achse gemessen). Die Entfernung ist nach der obigen Konstruktion gerade $t' + t_G - t_0$.[3] Das kann man sich so erklären: Der neue Ausfall hat vom Beginn der Verteilung G den Abstand t'_L. Der aktuelle Zeitpunkt ist aber t'. Die Lebensdauer vom Beginn der Verteilung G bis t' ist somit quasi schon „verbraucht“, man muß also nur die Differenzzeit berücksichtigen. Da die Verteilung G um $-t_G$ verschoben worden, also bei $t_0 - t_G$ beginnt, ist die bereits „verbrauchte“ Lebenszeit gerade $t' - (t_0 - t_G)$. Somit ergibt sich:

$$\begin{aligned} t'_1 &= t' + t'_L - (t' + t_G - t_0) \\ &= t_0 + t'_L - t_G \end{aligned} \tag{8.7}$$

Das Ergebnis von t'_1 entspricht der Erwartung: Bei t_0 war die Komponente völlig INTAKT, sie hat jetzt eine um t_G korrigierte Lebensdauer t'_L.
Beim Zugriff auf Verteilungs– und Dichtefunktion braucht man nur die Korrekturzeit zu berücksichtigen. Somit ergibt sich für gesamte Lebensdauer der Komponente vor und nach dem Reschedulen (hier wird t_F nicht mehr als 0 angenommen):

$$H(t_L) = \begin{cases} F(t_L + t_F) & f\ddot{u}r \quad t_L \le t' - t_0 \\ G(t_L + t_G) & f\ddot{u}r \quad t_L > t' - t_0 \end{cases} \tag{8.8}$$

$$h(t_L) = \begin{cases} f(t_L + t_F) & f\ddot{u}r \quad t_L \le t' - t_0 \\ g(t_L + t_G) & f\ddot{u}r \quad t_L > t' - t_0 \end{cases} \tag{8.9}$$

8.5 Verteilungen im Programm

Im folgenden werden die in `simsam` verwendeten Verteilungen aufgeführt und jeweils erläutert, wie die Verteilungsfunktion, ihr Inverses und das Auslosen von Zufallszahlen gemäß der Verteilung realisiert sind.

8.5.1 Exponentialverteilung

Die Exponentialverteilung ist die einfachste Verteilung, die einigermaßen realistisch das Ausfallverhalten eines technischen Bauteils beschreibt. Folglich ist sie sehr stark verbreitet. Ihre Dichtefunktion lautet:

[3] Da für die Auslosung von t'_L nur der Bereich zur Verfügung steht, in dem $G(t) > F_{t'}$ gilt, ist t'_L immer größer als $|t' + t_G - t_0|$, woraus $t'_1 > t'$ folgt.

$$f(t) = \lambda \exp(-\lambda t) \quad \text{für } t \geq 0 \tag{8.10}$$

Der Parameter $\lambda > 0$ ist die *Ausfallrate*. Je größer λ ist, desto schneller fällt im Mittel eine Komponente aus: Erwartungswert und Varianz der Exponentialverteilung sind $1/\lambda$ und $1/\lambda^2$.
Die Verteilungsfunktion läßt sich geschlossen errechnen:

$$F(t) = 1 - \exp(-\lambda t) \quad \text{für } t \geq 0 \tag{8.11}$$

Das Inverse der Verteilungsfunktion ist:

$$F^{-1}(p) = -\log(1-p)/\lambda \quad \text{für } 0 \leq p \leq 1 \tag{8.12}$$

Das Auslosen exponentialverteilter Zahlen wird durch Inversion realisiert. Dafür kann auch $-\log(p)/\lambda$ verwendet werden, da mit einer gleichverteilten Zahl p auch $1-p$ gleichverteilt ist.

8.5.2 Weibullverteilung

Die Weibullverteilung ist eine Verallgemeinerung der Exponentialverteilung. Ihre Dichtefunktion lautet:

$$f(t) = \alpha\beta t^{\beta-1} \exp(-\alpha t^{\beta}) \quad \text{für } t \geq 0 \tag{8.13}$$

Die Parameter sind *Ausfallrate* $\alpha > 0$ und *Exponent* $\beta > 0$. Mittelwert μ und Varianz σ^2 sind nach [bratle83]

$$\mu = \Gamma(1 + 1/\beta)/\alpha \tag{8.14}$$

und

$$\sigma^2 = [\Gamma(1 + 2/\beta) - (\Gamma(1 + 1/\beta))^2]/\alpha^2 \tag{8.15}$$

Dabei ist $\Gamma(x)$ die *Gammafunktion*, deren Definion in Anhang E.8 zu finden ist.
Die Weibull–Verteilungsfunktion lautet:

$$F(t) = 1 - \exp(-\alpha t^{\beta}) \quad \text{für } t \geq 0 \tag{8.16}$$

Auch diese Verteilung läßt sich algebraisch invertieren, weshalb auch hier das Inversionsverfahren zur Erzeugung weibullverteilter Zufallszahlen verwendet wird: (Wieder kann bei der Auslosung statt $1-p$ auch p verwendet werden.)

$$F^{-1}(p) = [-\log(1-p)/\alpha]^{1/\beta} \quad \text{für } 0 \leq p \leq 1 \tag{8.17}$$

8.5.3 Normalverteilung

Schon allein aus dem zentralen Grenzwertsatz ergibt sich die Bedeutung der Normalverteilung. Ihre Dichtefunktion lautet:

$$n_{\mu,\sigma}(t) = \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{(t-\mu)^2}{2\sigma^2}) \tag{8.18}$$

Der Erwartungswert ist μ und die Varianz σ^2.
Die Verteilungsfunktion läßt sich nicht in geschlossener Form ausrechnen. Wir verwenden für die Standardnormalverteilung $N_{0,1}$ eine Approximation nach [bratle83], welche für $0 \leq t \leq 6$ folgende Form hat:

$$\tilde{N}(t) := 1 - f_1 P(1/(1+q_1 t)) \exp(-t^2/2) \tag{8.19}$$

Dabei ist P ein Polynom fünften Grades, welches zusammen mit den Konstanten f_1 und q_1 in Anhang E.9 angegeben ist.
Für $t \geq 6$ wird $\tilde{N}(t) = 1$ verwendet. Für $t < 0$ wird die Symmetrie der Normalverteilung ausgenutzt: $N_{0,1}(-t) = 1 - N_{0,1}(t)$. Die allgemeine Verteilung mit den Parametern μ und σ hängt wie folgt mit der Standardnormalverteilung zusammen:

$$N_{\mu,\sigma}(t) = N_{0,1}((t-\mu)/\sigma) \tag{8.20}$$

Die Genauigkeit dieser Näherung beträgt 5 Stellen.
Auch für das Inverse der Normalverteilung wird eine Approximation verwendet. Quelle ist [odeh74]. Dort wird die Verkettung einer rationalen Funktion mit Quadratwurzel und Logarithmus für das Inverse der Standardnormalverteilung verwendet:

$$\hat{N}^{-1}(p) := Q(\sqrt{-\log([1-p]^2)}) \quad \text{für } 10^{-18} \leq p \leq 0.5 \tag{8.21}$$

Für $0 \leq p \leq 10^{-18}$ wird $\hat{N}^{-1} = -8$ genommen. Für $0.5 < p \leq 1$ wird wiederum die Symmetrie der Normalverteilung ausgenutzt: $\hat{N}^{-1}(p) := -\hat{N}^{-1}(1-p)$. Die rationale Funktion Q ist in Anhang E.10 angegeben.
Insgesamt ergibt sich folgende Näherung, deren Genauigkeit wie oben 5 Stellen beträgt:

$$N_{\mu,\sigma}^{-1}(p) \approx \mu + \sigma \hat{N}^{-1}(p) \tag{8.22}$$

Für die Realisierung der Auslosung könnte man das Inversionsverfahren mit der Näherung (8.22) verwenden. Es gibt aber ein schnelleres Verfahren, welches exakt ist, die Box–Muller Methode [box58]. Sie erzeugt standardnormalverteilte Zahlen k, die dann mittels der Transformation $w = \mu + \sigma k$ gemäß $N_{\mu,\sigma}$–verteilt sind. Da die ausgelosten Zahlen Zeiträume angeben sollen, werden negative Zahlen auf 0 aufgerundet.

Das Verfahren lost zwei standardnormalverteilte Zahlen aus. Dazu werden zwei $(0,1)$–gleichverteilte Zahlen u_1, u_2 ausgelost und wie folgt transformiert:

$$\begin{aligned} k_1 &= \sqrt{-2\log(u_2)}\cos(2\pi u_1) \\ k_2 &= \sqrt{-2\log(u_2)}\sin(2\pi u_1) \end{aligned} \tag{8.23}$$

Der Beweis für die Korrektheit der Methode findet sich in Anhang E.11.

8.5.4 Gammaverteilung

Von den Verteilungen des Programms, benötigt die Gammaverteilung in ungünstigen Fällen die längste Rechenzeit. Ihre Dichtefunktion lautet:

$$f_{\alpha,\beta}(t) = \alpha(\alpha t)^{\beta-1}\exp(-\alpha t)/\Gamma(\beta) \tag{8.24}$$

Die Parameter der Verteilung sind *Ausfallrate* $\alpha > 0$ und *Exponent* oder *Formparameter* $\beta > 0$. Als Erwartungswert und Varianz ergeben sich nach [bratle83] die Werte β/α und β/α^2.
Zuerst wird der Fall behandelt, daß β eine natürliche Zahl ist. Dann gilt $\Gamma(\beta) = (\beta-1)!$ und die Verteilungsfunktion läßt sich geschlossen ausrechnen:

$$F_{\alpha,\beta}(t) = 1 - \exp(-\alpha t)\sum_{j=0}^{\beta-1}(\alpha t)^j/j! \tag{8.25}$$

Für das Inverse der Verteilungsfunktion verwenden wir den oben vorgestellten Verschachtelungsalgorithmus. Für die Auslosung gammaverteilter Zufallszahlen wird allerdings im Programm nicht die Inversionsmethode verwendet, da folgendes Verfahren schneller ist:
Man erhält aus β Zufallszahlen $u_1, \ldots, u_\beta$, die $(0,1)$–gleichverteilt sind, mittels folgender Umformung eine gammaverteilte Zahl w (Beweis siehe Anhang E.12):

$$w = -\sum_{i=1}^{\beta}\log(1-u_i)/\alpha \tag{8.26}$$

Da mit u_i auch $1-u_i$ gleichverteilt sind, kann in Formel 8.26 auch $\log(u_i)$ verwendet werden.
Im allgemeinen Fall, daß β irgendeinen positiven reellen Wert annimmt, läßt sich die Verteilungsfunktion nicht mehr geschlossen ausrechnen. In `simsam` wird deshalb numerische Integration verwendet. Für die Berechnung der Umkehrfunkion wird die umgekehrte Integration (Anhang E.4) angewendet.
Für die Auslosung gammaverteilter Zahlen läßt sich leider nicht die Verwerfungsmethode verwenden, da die Dichtefunktion mit den Parametern zu stark

die Form ändert, so daß man nicht auf einfache Weise eine geeignete einhüllende Funktion finden kann. Glücklicherweise ist man aber auch nicht auf die Inversionsmethode angewiesen. Bei uns findet ein Algorithmus aus [berman70] Anwendung, der auf Untersuchungen in [jöhnk64] basiert (siehe Anhang E.12).

8.5.5 Gleichverteilung

Die Dichtefunktion einer Gleichverteilung über dem Intervall $[a, b]$ lautet:

$$f(t) = \begin{cases} 1/(b-a) & \text{für } t \in [a, b] \\ 0 & \text{sonst} \end{cases} \tag{8.27}$$

Somit ergibt sich für die Verteilungsfunktion:

$$F(t) = \begin{cases} 0 & \text{für } t < a \\ \frac{t-a}{b-a} & \text{für } a \leq t \leq b \\ 1 & \text{für } t > b \end{cases} \tag{8.28}$$

Die Berechnung der Umkehrfunktion ist einfach. Folglich werden — wie schon oben angedeutet — gleichverteilte Zufallszahlen nach der Inversionsmethode erzeugt.

8.5.6 Deterministische Verteilungen

Außerdem ist es möglich Zeitpunkte von Ereignissen festzulegen. Die passenden Dichtefunktionen werden mittels *Deltafunktionen* dargestellt. Soll das Ereignis zur Zeit t_0 stattfinden, so lautet seine Dichtefunktion:

$$f(t) = \delta(t - t_0) \tag{8.29}$$

Die Deltafunktion hat folgende definierende Eigenschaften: Sie verschwindet fast überall

$$\delta(x) = \begin{cases} \infty & \text{für } x = 0 \\ 0 & \text{sonst} \end{cases} , \tag{8.30}$$

und es gilt für jede stetige Funktion f:

$$f(x_0) = \int_{-\infty}^{+\infty} f(x)\delta(x - x_0)\, dx \tag{8.31}$$

Damit ergibt sich die Verteilungsfunktion der deterministischen Verteilung zu:

$$F(t) = \begin{cases} 0 & \text{für } t < t_0 \\ 1 & \text{für } t \geq t_0 \end{cases} \tag{8.32}$$

Das Inverse der Verteilungsfunktion wird für Rescheduling benötigt. Da man die Inversion nicht ausführen kann, verwenden wir die konstante Nullfunktion, welche der Idee des Rescheduling entspricht: Vor dem Ausfall zeigte die mit der deterministischen Verteilung versehene Komponente keinen Verschleiß.
Bei Berechnungen im Zusammenhang mit IS werden Dichtefunktion und Verteilungsfunktion benötigt. Da man bei IS deterministische Parameter nicht ändern darf, ergibt sich aus diesen Funktionen kein Beitrag zum IS–Faktor, so daß man sie dafür auf konstant 1 setzten kann. Auf diese Weise hat man auch keine Probleme mit Divisionen durch 0 oder der Frage, was eine Division durch die Deltafunktion bedeutet.

Kapitel 9

Ablauf der Simulation

In diesem Kapitel wird geschildert, wie der Simulationsalgorithmus abläuft. Er basiert auf der Verarbeitung einer Ereignisliste. Wie der Algorithmus in einzelne Programmodule aufgeteilt ist, kann man in Anhang F nachlesen.

9.1 Verwaltung der Ereignisliste

Der Ablauf der Simulation besteht aus einer Abfolge von Ereignissen. Ein Beispiel eines Ereignisses: Zur Zeit 5.7s fällt die Komponente mit dem Namen „Generator" aus. Jedes Ereignis besteht somit aus bestimmten Informationen: *Zeitpunkt* und *Art* (Ausfall, Reparatur) des Ereignisses sowie Nummer der betroffenen *Komponente* (ermöglicht einen schnelleren Zugriff als der Name). Die Ereignisse werden in Form einer doppelt verketteten Liste verwaltet, die nach den Zeitpunkten geordnet ist (siehe Abbildung 9.1). Neben dieser Liste der aktuellen Ereignisse existiert für jede Reparaturstation eine genauso aufgebaute Warteliste, in die Reparaturen eingeordnet werden, die noch nicht ausgeführt werden. Die Reihenfolgen in diesen Listen hängen von den Bedienstrategien der einzelnen Stationen ab: Die Reparatur, die als nächstes ausgeführt wird, steht vorne. Immer wenn ein Ereignis aus einer Warteliste in die Ereignisliste übertragen wird, muß der *Zeitpunkt* neu berechnet werden.
Weitere Ereignisdaten dienen internen Berechnungen. Im wesentlichen werden sie benötigt, um den IS–Faktor zu ermitteln.
Die *Erzeugzeit* gibt an, wann das Ereignis erzeugt wurde. Diese Zeit wird von Reparaturstationen verwendet, die nach LIFO– oder FIFO–Strategien ihre Aufträge bearbeiten.
Die *Startzeit* gibt an, von welchem Zeitpunkt an sich das Ereignis in der Liste befindet. Sie wird verwendet, um die Dauer des Aufenthalts in der Liste zu berechnen und daraus den IS–Faktor. Die Startzeit muß nicht unbedingt gleich der Erzeugzeit sein, da Reparaturereignisse sich auch zeitweise in Wartelisten (quasi inaktiv) aufhalten können.

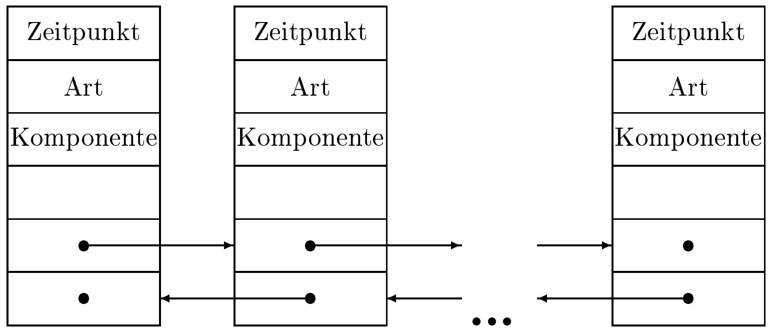

Abbildung 9.1: Eine Ereignisliste

Bei Ereignissen, die rescheduled wurden, wird auf die Verteilungsfunktionen nicht mit der aktuellen Zeit zugegriffen, sondern mit einer Zeit, die von der aktuellen Zeit um eine konstante Differenz entfernt liegt (siehe Kapitel 8.4). Diese Differenz wird in *Korrekturzeit* gespeichert.
Bei einem Ereignis, das sich nicht während seiner gesamten Existenz in der Ereignisliste aufhielt, interessiert für die Berechnung des IS–Faktors nur die Dauer des Listenaufenthalts. Dazu wird der Wert *Alter* verwendet, der angibt, wie lange sich das Ereignis vor dem letztmaligen Ausordnen in der Ereignisliste aufhielt. Hauptsächlich wird dieser Parameter für Reparaturen benötigt, die in eine Warteliste verdrängt werden, oder für Rescheduling.
Für Reparaturereignisse, die sich in der Warteliste einer Reparaturstation befinden, muß die noch verbleibende Reparaturzeit bekannt sein. Sie wird in *Restdauer* gespeichert.
Ein *Zufallswert*, der zwischen 0 und 1 liegt, dient dazu, die Reihenfolge der Reparatur bei Stationen mit der Bedienstrategie RANDOM festzulegen.
Bei Reparaturstationen, die nach der PROCESSOR–SHARING Strategie durchgeführt werden, wird eine Komponente immer während eines Bruchteils der Zeit repariert, die sie sich in der Liste aufhält. Das muß bei der Berechnung des IS–Faktors berücksichtigt werden. Dieser Bruchteil, der gleich der Zahl der gleichzeitig reparierten Komponenten ist, wird in *Divisionsfaktor* gespeichert.
Die im folgenden aufgeführten Ereignisdaten dienen der schnelleren Verarbeitung. Man könnte sie auch auf Umwegen über die Daten des simulierten Systems erhalten:
Die *Qualität* gibt an, ob das Ergebnis einer Reparatur so gut wie neu („as good as new“) ist oder nicht.
Für die Ermittlung von Reparaturreihenfolgen wird die *Priorität* einer Reparatur gespeichert.
Für jedes Reparaturereignis wird die Nummer der *Reparaturstation* gespeichert.

Sie wird benötigt, wenn nach Beendigung einer Reparatur ein Ereignis aus der zugehörigen Warteliste aktiviert werden soll.
Nach Beendigung einer Reparatur muß die Belastung der von der defekten Komponente beeinflußten Komponenten vermindert werden (durch Erniedrigen der Modifikationsstufe). Ein Zeiger auf diese Komponenten wird in *Beeinflußt* festgehalten.
Bei der Berechnung des IS–Faktors muß auf die ursprüngliche und auf die IS–Verteilung zugegriffen werden. Dafür werden die Zeiger *Normale Verteilung* und *IS-Verteilung* verwendet. Außerdem werden bei Rescheduling ohne Berücksichtigung der Alterung zwei *Verteilungsfaktoren* [1] benötigt, um die normale und die IS–Verteilung zu skalieren.
In Tabelle 9.1 werden alle Elemente eines Ereignisses noch einmal zusammenfassend aufgeführt. Die Zeigerwerte *nächstes Element* und *vorheriges Element* dienen der Doppelverkettung der Liste.

Zeiger	ganzahlige Werte	Fließkommawerte
Beeinflußt	Art	Zeitpunkt
Normale Verteilung	Komponente	Erzeugzeit
IS Verteilung	Divisionsfaktor	Startzeit
nächstes Ereignis	Qualität	Korrekturzeit
vorheriges Ereignis	Priorität	Alter
	Reparaturstation	Restdauer
	Verteilungsfaktor	Zufallswert

Tabelle 9.1: Werte eines Ereignisses

9.2 Der Simulationsalgorithmus

Ein Simulationsdurchgang beginnt immer bei der *aktuellen Zeit* 0.0 in dem Zustand, in dem alle Komponenten INTAKT sind, d.h. im regenerativen Zustand. Der IS–Faktor hat den Wert 1.0. Als erstes werden für alle Komponenten Ausfallereignisse ausgelost und daraus die Ereignisliste aufgebaut. Die eigentliche Simulation besteht nun aus der wiederholten Verarbeitung des jeweils ersten Ereignisses der Liste, bis ein Simulationsdurchgangs beendet ist. Wie das Ende eines Durchgangs bestimmt wird und wie die Zuverlässigkeitskenngrößen ermittelt werden, wird in Kapitel 9.3 erklärt. Es folgt eine Beschreibung der Verarbeitung des jeweils ersten Ereignisses der Ereignisliste (siehe Abbildung 9.2).

[1] Das ist der Wert $F_{t'}$ aus Kapitel 8.4.

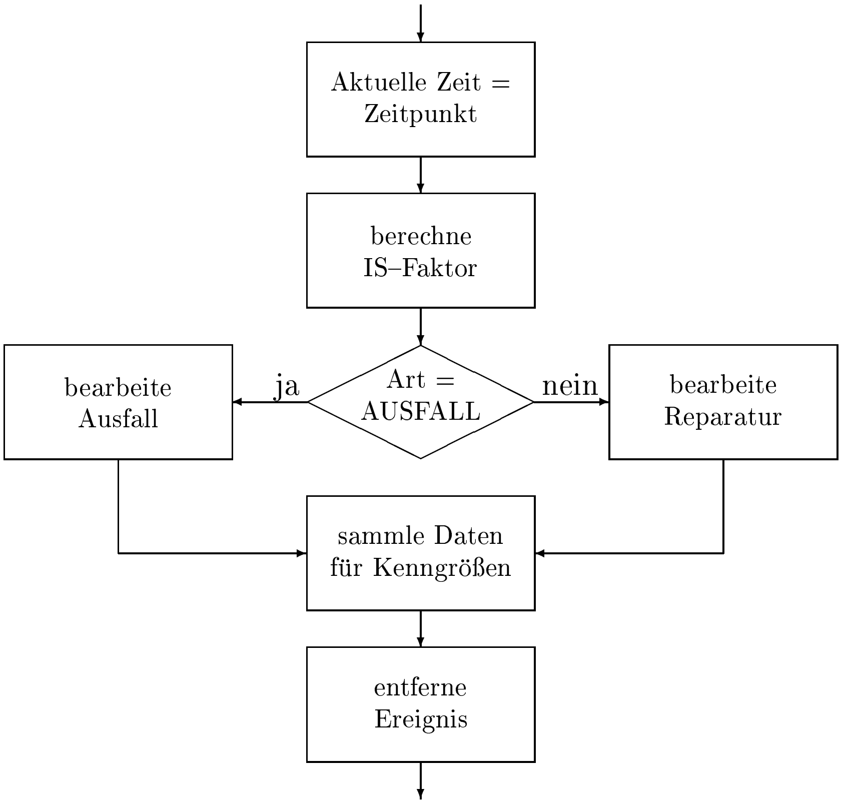

Abbildung 9.2: Der Simulationsalgorithmus

Die *aktuelle Zeit* ist der *Zeitpunkt* an dem das erste Ereignis der Liste stattfindet. Dann wird aus der Ereignisliste der IS–Faktor neu berechnet. Je nachdem, ob das erste Ereignis einen Ausfall oder eine Reparatur (genauer: das Ende einer Reparatur) beschreibt, wird es von verschiedenen Teilalgorithmen bearbeitet. Zum Schluß wird es aus der Ereignisliste entfernt, nachdem die Daten für die Berechnung der Kenngrößen aktualisiert wurden.

9.2.1 Bearbeitung eines Ausfallereignisses

Die Verarbeitung eines Ausfalls ist in Abbildung 9.3 dargestellt.

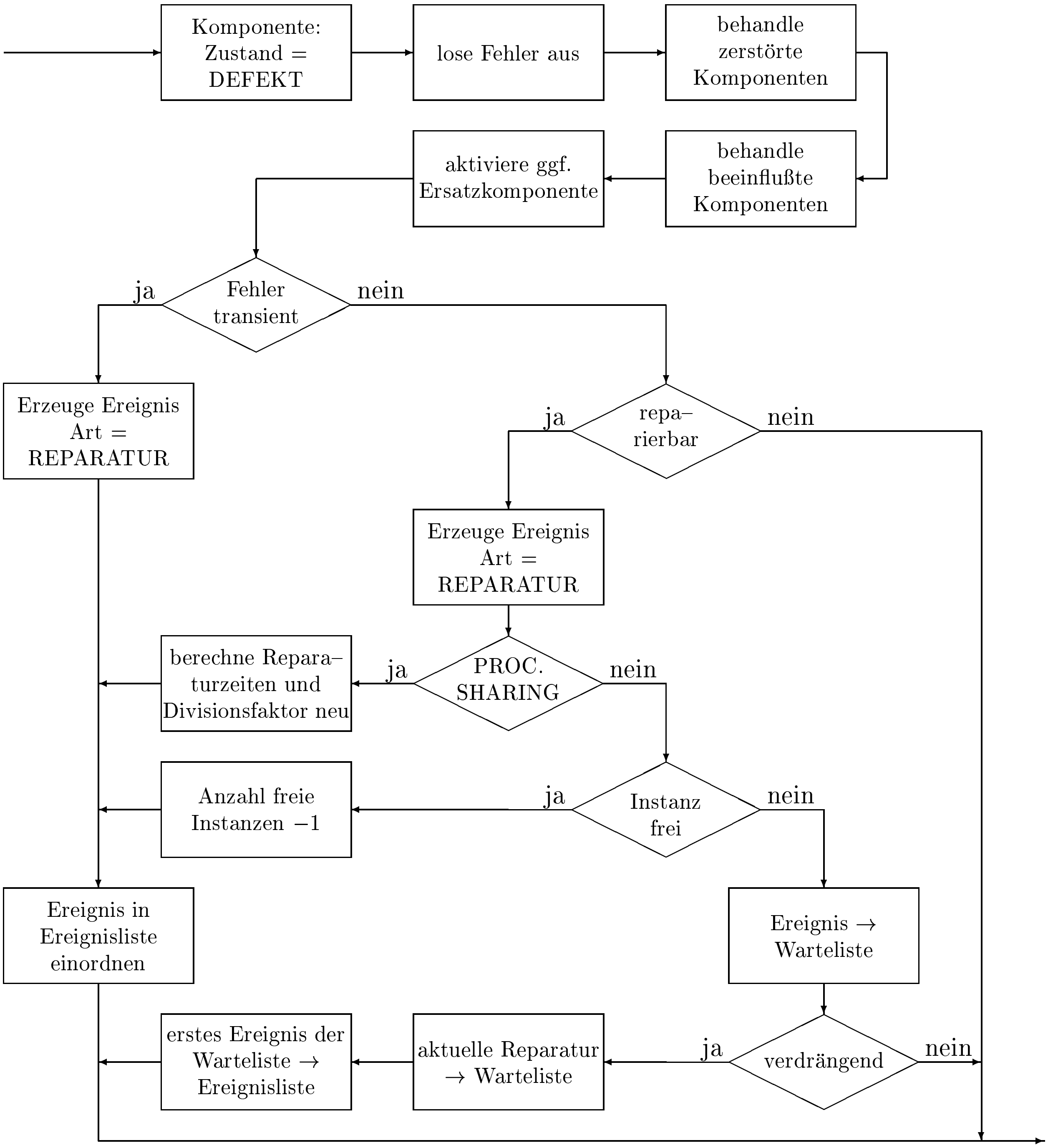

Abbildung 9.3: Behandlung eines Ausfalls

Als erstes wird der Zustand der vom Ausfall betroffenen Komponente auf DEFEKT gesetzt. Dann wird aus der Liste der Fehler, die bei dem Ausfall der betroffenen Komponente möglich sind, einer ausgelost und der IS–Faktor entsprechend neu berechnet. Durch die Fehlerart ist eine Menge von Komponenten festgelegt, die durch den Ausfall zerstört bzw. beeinflußt werden.
Für INTAKTE Komponenten, die zerstört werden, wird folgendes durchgeführt: Das zu dieser Komponente vorhandene Ereignis wird aus der Ereignisliste entfernt. Dann wird ein Ausfallereignis mit den *Zeitpunkt* = *aktuelle Zeit* erzeugt und hinter dem gerade bearbeiteten Ereignis in die Ereignisliste eingefügt.
Für jede beeinflußte Komponente K_b wird *Anzahl beeinflußt* um eins erhöht. Falls K_b INTAKT ist, wird ihre *Modifikationsstufe* um eins erhöht und gemäß der neuen Wahrscheinlichkeitsverteilung der *Zeitpunkt* des nächsten Ausfalls rescheduled. Falls eine Erhöhung der Modifikation nicht möglich ist, verbleibt die Komponente in ihrer höchsten Stufe. Bei der Rückgängigmachung der Beeinflussung darf im letzten Fall die *Modifikationsstufe* nicht wieder verringert werden. Beispiel: Eine Komponente hat die Stufen 0 bis 2. Sie wird von 4 Komponenten beeinflußt, somit nimmt sie die *Modifikationsstufe* 2 an. Bei Zurücknahme der ersten beiden Beeinflussungen ändert sich die Stufe der Modifikation nicht, erst bei Reparatur der dritten beeinflussenden Komponente sinkt sie auf 1.
Als nächstes wird die Ersatzkomponente aktiviert, indem ihre *Anzahl Ersetzungen* um eins erhöht wird, falls die *Reserveart* KALT oder WARM ist. Falls möglich und falls die Komponente INTAKT ist, wird dann die *Modifikationsstufe* verkleinert und der nächste Ausfall rescheduled.
Falls der oben ausgeloste Fehler TRANSIENT ist, wird gemäß den Fehlerdaten eine Ausfalldauer ausgelost, damit ein Reparaturereignis erzeugt und dieses in die Ereignisliste eingeordnet.
Im anderen Fall, wenn der Fehler permanent ist, wird zuerst überprüft, ob der Fehler reparierbar ist. Falls nicht, geschieht nichts weiter, d.h. die Komponente ist dauerhaft DEFEKT (bis zum nächsten Simulationsdurchlauf). Andernfalls wird aus den Daten der zuständigen Station die Reparaturdauer ausgelost und ein Reparaturereignis erzeugt.
Die aufwendige *Reparaturstrategie* PROCESSOR SHARING wird gesondert behandelt: Für alle bereits in der Ereignisliste befindlichen Reparaturen, die von der Reparaturstation bearbeitet werden, welche für die aktuelle Fehlerart zuständig ist, muß der *Divisionsfaktor* um eins erhöht werden. Weiterhin müssen die *Zeitpunkte* dieser Ereignisse neu berechnet werden, da sich die Dauer der Bearbeitung der Komponenten durch die Annahme eines weiteren Auftrags vergrößert. Dazu wird für alle Reparaturereignisse, die zu der zuständigen Reparaturstation gehören, folgendes durchgeführt:

- sie werden aus der Ereignisliste entfernt;
- ihr *Alter* wird um die Zeit erhöht, während der sie seit dem letztmaligen Einordnen in die Ereignisliste repariert wurden;
- es wird ihre restliche Reparaturdauer, d.h. die *Restdauer* berechnet;
- sie werden in die Warteliste der Reparaturstation, nach steigenden *Restdauern* sortiert, eingefügt.

Als nächstes wird das gerade erzeugte Reparaturereignis mit *Restdauer* = (*Zeitpunkt* – *aktuelle Zeit*) eingeordnet.
Um nun die neuen *Zeitpunkte* zu berechnen, wird das jeweils erste Ereignis der Warteliste

- aus der Warteliste entfernt,
- sein *Zeitpunkt* aus der *Restdauer*, der *aktuellen Zeit*, den *Restdauern* der Vorgänger in der Warteliste sowie der Anzahl der noch in der Warteliste befindlichen Ereignissen berechnet,
- es in die Ereignisliste eingefügt.

Für den Fall, daß die *Reparaturstrategie* nicht PROCESSOR SHARING war, läuft der Algorithmus wie folgt weiter: Zuerst wird überprüft, ob die zuständige Reparaturstation noch eine weitere Instanz frei hat, was insbesondere bei der Strategie INFINITE SERVER immer gilt. Ist das der Fall, wird die *Anzahl freie Instanzen* um eins vermindert, und das Ereignis kommt in die Ereignisliste.
Falls keine freien Instanzen zur Verfügung stehen, wird das Ereignis in die Warteliste der Reparaturstation eingeordnet.
Nur falls '*verdrängend*' = JA gilt, kann die Reparatur der gerade betrachteten Komponente eventuell doch beginnen: Aus der Menge der aktiven Reparaturereignisse der Station wird das Ereignis in die Warteliste verdrängt, dessen Bearbeitung gemäß der *Reparaturstrategie* zuletzt beginnen würde. Das Ereignis, das dann in der Warteliste an erster Stelle steht, wird in die Ereignisliste übertragen.

9.2.2 Bearbeitung eines Reparaturereignisses

Die Verarbeitung einer Reparatur ist in Abbildung 9.4 dargestellt.

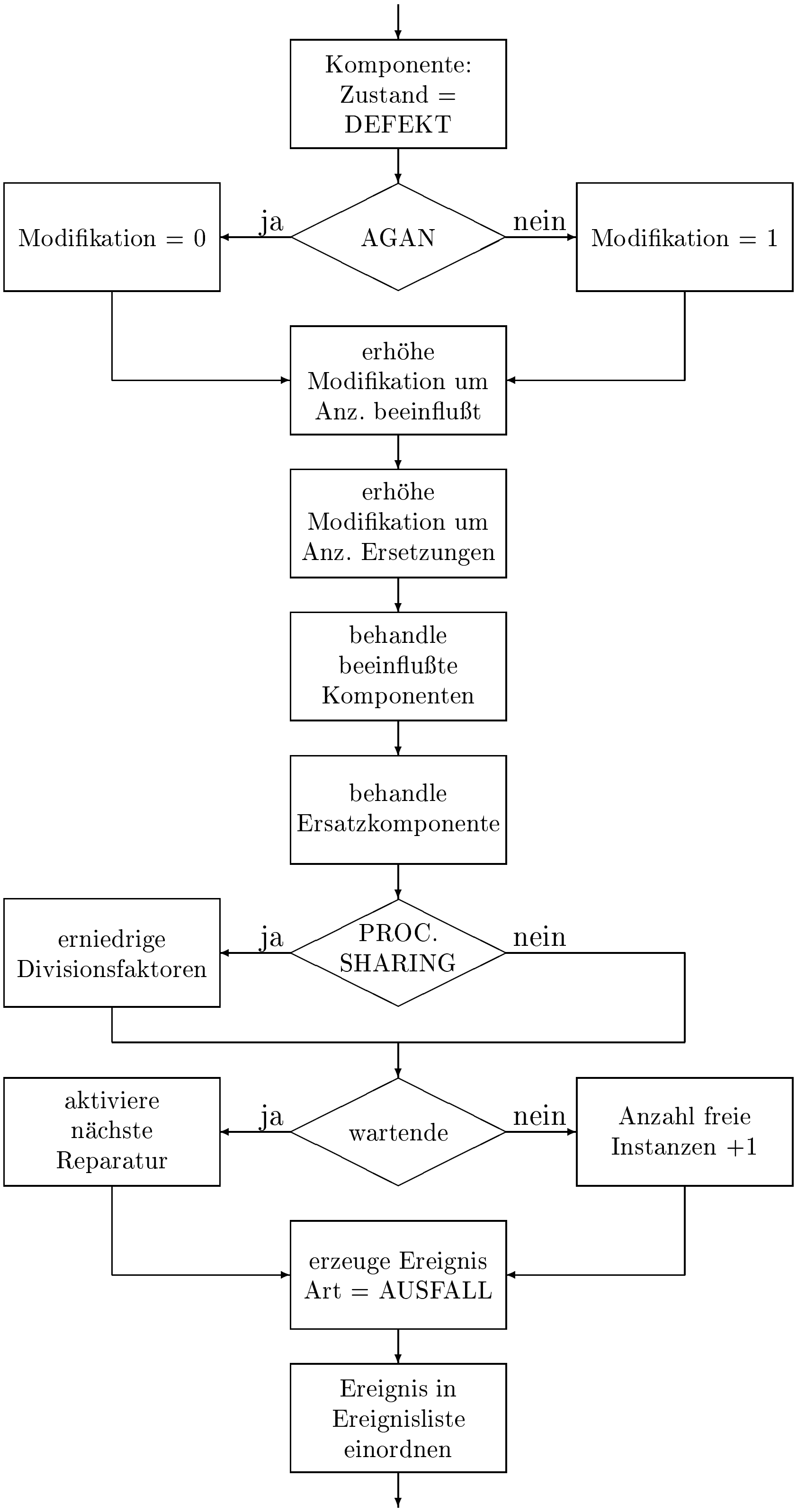

Abbildung 9.4: Behandlung einer Reparatur

Als erstes wird der *Zustand* der reparierten Komponente K_r auf INTAKT gesetzt.
Dann wird die *Modifikationsstufe* von K_r gewählt. Falls die Komponente so gut wie neu ist (AGAN = as good as new) wird die niedrigste Stufe 0 genommen, sonst (NAGAN = not AGAN) die Modifikationsstufe 1.[2] Bei später ausgeführten Reduzierungen der aktuellen Stufe muß darauf geachtet werden, daß der Wert nicht unter den jetzt angenommenen sinken darf. Beispiel: Die Komponente hat 3 Modifikationsstufen (0 bis 2), sie wird NAGAN repariert, daraus folgt *Modifikationsstufe* = 1. Nun wird sie durch 2 Komponentenausfälle beeinflußt, woraus sich die Stufe 2 ergibt. Bei Reparatur der ersten dieser Komponenten ändert sich die Stufe nicht, erst wenn die zweite Komponente repariert ist, wird wieder die Stufe 1 angenommen.
Falls die K_r von anderen gerade ausgefallenen Komponenten beeinflußt wird, dann wird ihre *Modifikationsstufe* um so viel erhöht, wie es beeinflussende Komponenten gibt.
Für jede Komponente, für die K_r gerade als warme oder kalte Reserve tätig ist, wird die *Modifikationsstufe* um eins erhöht.[3]
Als nächstes werden die von der reparierten Komponente beeinflußten Komponenten bearbeitet: Deren Variablen *Anzahl beeinflußt* werden um eins vermindert. Für jede ehemals beeinflußte Komponente, die INTAKT ist, wird die *Modifikationsstufe* — wenn möglich — um eins reduziert und die Zeit des nächsten Ausfalls neu ausgelost (rescheduled).
Danach wird eine evtl. vorhandene Ersatzkomponente deaktiviert:[4] Falls sie INTAKT ist und die *Reserveart* KALT oder WARM ist, wird die *Anzahl Ersetzungen* jeder Ersatzkomponente um Eins reduziert. Falls möglich, wird ihre *Modifikationsstufe* vermindert und der nächste Ausfall rescheduled.
Falls die Komponente mit der *Reparaturstrategie* PROCESSOR SHARING repariert wurde, wird für alle Reparaturereignisse, die zu der selben Reparaturstation gehören, der *Divisionsfaktor* um eins vermindert.
Falls die Warteschlange der Reparaturstation, die die Komponente repariert hat, nicht leer ist, wird ihr erstes Ereignis entnommen und in die Ereignisliste eingeordnet, d.h. diese Reparatur wird gestartet bzw. fortgesetzt. Für dieses Ereignis wird *Zeitpunkt = aktuelle Zeit + Restdauer* gesetzt. Im Fall der leeren Warteschlange wird dagegen die *Anzahl freie Instanzen* um eins erhöht.

[2] Die Bezeichnung des nicht–modifizierten Zustandes einer Komponente mit Modifikationsstufe 0 wurde aus Kompatibilitätsgründen zu dem Programm `mdledit` aus [ruenge92] verwendet.

[3] Da das Programm `mdledit` nicht die Eingabe von Verteilungsfunktionen für Reservezustände unterstützt, ist diese Hilfskonstruktion nötig. Eine Unterscheidung zwischen warmer und kalter Reserve erschien uns deshalb auch nicht sinnvoll.

[4] Da das Programm `mdledit` keine echten Reservezustände unterstützt, wird immer Verdrängung der Ersatzkomponente nach der Reapartur angenommen. Andernfalls müßte bei Nicht–Verdrängung eine AGAN–reparierte Komponente — im Rahmen des oben angegebenen Hilfsschemas — die Modifikationsstufe -1 erhalten.

Zuletzt wird gemäß der oben ermittelten *Modifikationsstufe* der nächste Ausfall von K_r ausgelost und dieses Ausfallereignis in die Ereignisliste eingeordnet.

9.3 Ermittlung der Kenngrößen

Wie oben bereits erläutert, kann der Algorithmus auf direktem Weg nur transiente Kenngrößen ermitteln. Wie das geschieht, wird nun geschildert.
Jede Kenngröße wird durch drei Werte dargestellt: Einen Zeiger auf eine Statusfunktion, einen Zeiger auf eine Wertefunktion und einige Parameter, die die Kenngröße genauer bestimmen (siehe Abbildung 9.5).

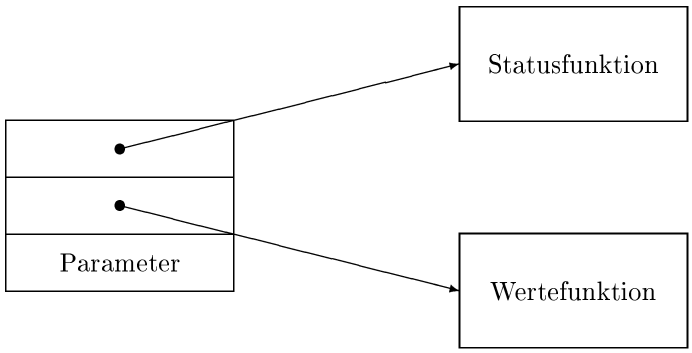

Abbildung 9.5: Darstellung einer Kenngröße

Die Statusfunktion gibt an, ob ein Simulationsdurchgang für die Ermittlung der jeweiligen Kenngröße noch fortgesetzt werden muß (sie liefert dann WEITER) oder nicht (GESTOPPT). Wenn ein Durchgang beendet ist, wird der Wert der Kenngröße, der sich aus dem Durchgang ergibt, mit der Wertefunktion bestimmt und dann mit dem IS–Faktor multipliziert abgespeichert. Beide Funktionen benötigen für die Berechnung ihrer Werte gewisse globale Variablen. Dazu gehören z.B. die *aktuelle Zeit*, der *Systemzustand* und die *Defektzeit*, das ist die Gesamtzeit, die das System im laufenden Durchgang DEFEKT war. Die Parameter, die die Kenngrößen genauer bestimmen, werden den beiden Funktionen bei jedem Aufruf übergeben.
Als Beispiel wird nun die Ausfallwahrscheinlichkeit des Gesamtsystems bis zur Zeit t betrachtet (Der Zeitpunkt t ist der Parameter dieser Kenngröße). Die Statusfunktion gibt GESTOPPT zurück, wenn die *aktuelle Zeit* die Zeit t überschritten hat oder wenn das Gesamtsystem voher DEFEKT ist. Als Wertefunktion wird eine Indikatorfunktion verwendet, die 1.0 zurückliefert, wenn die *aktuelle Zeit* kleiner als die Parameterzeit t ist, denn dann muß das System ausgefallen sein, da die Wertefunktion nur aufgerufen wird, wenn die Statusfunktion GESTOPPT liefert. Im anderen Fall liefert die Wertefunktion eine

0.0, da dann das System nicht vor der Zeit t ausfiel.
Aufgrund dieser Realisierung ist es leicht möglich neue Kenngrößen zu dem Programm hinzuzufügen. Man muß dazu nur geeignete Status– und Wertefunktionen definieren und Zeiger auf diese Funktionen in das Array eintragen, in dem die Kenngrößen gespeichert sind.[5]
Während einer Simulation können mehrere Kenngrößen gleichzeitig bestimmt werden. Dazu werden sie zusammen mit ihrem aktuellen Status in einem Array abgespeichert. Zu Beginn eines Simulationsdurchgangs werden alle Statuswerte auf WEITER gesetzt. Immer wenn ein Ereignis bearbeitet wurde, werden für die Kenngrößen, die noch WEITERlaufen, die Statuswerte mittels der zugehörigen Statusfunktion neu berechnet. Falls das Ergebnis nun GESTOPPT ist, wird die Wertefunktion aufgerufen. Ein Durchgang endet, wenn alle Kenngrößen GESTOPPT sind.

[5] ggf. benötigt man noch weitere globale Variablen, in denen Zwischenergebnisse gespeichert werden.

Kapitel 10

Experimente

In diesem Kapitel werden die Ergebnisse unserer Experimente vorgestellt. Im ersten Teil sind die Versuche mit der Antithetischen Variablen beschrieben. Der zweite Teil enthält Experimente zu Importance Sampling. Schließlich werden Beispiele mit dynamischen IS sowie Simulationen eines großen Systems vorgestellt, welche mit dem Programm `simsam` ausgeführt wurden.
Alle Zeit– und Ratenwerte sind auf eine beliebige Zeiteinheit bezogen, z.B. eine Stunde. Sie wird im folgenden nicht weiter angegeben. Immer wenn von Varianzwert gesprochen wird, ist damit die Stichprobenvarianz gemeint, falls es nicht anders gesagt wird.

10.1 Versuche zur Antithetischen Variablen

Das Verfahren der Antithetischen Variablen beruht darauf, nur einen Teil der benötigten Zufallszahlen zu erzeugen und den anderen Teil aus den erzeugten Zahlen zu berechnen.
Als erster Test wurde einfach der Mittelwert der Exponentialverteilung, anders gesagt, die mittlere Lebensdauer einer Komponente mit exponentialverteilter Lebensdauer, experimentell bestimmt. Die Ausfallrate betrug $\lambda = 0.01$. Zunächst wurde die normale Simulation angewendet, dann wurde das Verfahren so modifiziert, daß jede $(0, 1)$–verteilte Zufallszahl u noch ein zweites mal als $1 - u$ benutzt wurde. Die in der Tabelle 10.1 dargestellten Ergebnisse wurden aus je 10^4 Werten gemittelt. In der dritten Zeile sind die analytisch berechneten Werte dargestellt.
Wie man erkennt, bringt die Anwendung der antithetische Variablen in diesem Fall keinerlei Vorteil: Der Mittelwert ist nur zufälligerweise genauer, denn die Varianz ist hier ein wenig größer. Wir haben deshalb ein etwas komplizierteres semideterministisches Verfahren benutzt: u, z seien zwei ausgeloste $U(0, 1)$–verteilte Zufallszahlen. Als erste Zahl für die Simulation wird u verwendet und als nächstes die Zahl $u := u + b * (z - 0.5)$, wobei b ein konstanter Faktor ist.

Art der Ermittlung	Mittelwert	Varianz
normale Simulation	98.5	9550
antiteth. Variable	100.0	9976
Rechnung	100.0	10000

Tabelle 10.1: einfache antithetische Variable

Falls die so ermittelte Zahl u größer als 1 ist, wird sie um 1 vermindert, falls sie kleiner 0 ist, um 1 vergrößert. In Tabelle 10.2 ist wiederum die Simulation einer Komponente mit exponentialverteilter Lebensdauer ($\lambda = 0.01$) dargestellt. Die Mittelwerte wurden aus 10000 Meßwerten gebildet.

Faktor b	Mittelwert	Varianz
1.0	99.6	10053
0.9	99.4	9809
0.8	101.0	10094
0.7	100.8	9638
0.6	99.0	10141
0.5	100.3	9669
0.4	102.8	10214
0.3	97.8	9587
0.2	104.3	10401
0.1	90.5	8709

Tabelle 10.2: semideterminierte Variable

Wie man sieht, brachte auch diese Variante nicht das gewünschte Ergebnis. Dieser Mißerfolg bewog uns dazu, eine in [morgan84] dokumentierte Simulation einer M/M/1–Warteschlange durchzuführen. An einer solchen Warteschlange stellen sich Kunden, deren Ankunftsabstände mit der Rate λ exponentialverteilt sind, vor einem Schalter an. Die Bedienzeit der Kunden am Schalter ist ebenfalls exponentialverteilt, aber mit dem Parameter μ.
Im Experiment wurde die mittlere Wartezeit w der ersten 200 Kunden für $\lambda = 0.6$ und $\mu = 1$ berechnet, deren theoretischer Wert bei 2.5 liegt. Aus 100 solcher Werte wurde ein Mittelwert $\overline{w}$ gebildet. In [morgan84] wurde das Experiment je dreimal ohne bzw. mit Einsatz der Antithetischen Variablen (Verwendung von u und $1 - u$) durchgeführt. In Tabelle 10.3 sind die Ergebnisse dargestellt.
Wie man sieht, ergibt sich eine Varianzreduktion um den Faktor von ca. 2.5. Die gleichen Experimente wurden auf der Sun 3/50 durchgeführt, wobei ein

	normale Simulation		antithet. Var.	
Lauf	Mittelwert	Varianz	Mittelwert	Varianz
1	2.513	0.3102	2.464	0.1338
2	2.467	0.4010	2.408	0.1851
3	2.533	0.5233	2.543	0.1676
Mittelwert	2.504	0.4115	2.472	0.1622

Tabelle 10.3: Ergebnisse der M/M/1 nach Morgan

Zufallszahlengenerator, der bei dem Sun C–Compiler enthalten ist[1], verwendet wurde. Die Ergebnisse von 10 Durchgängen sind in Tabelle 10.4 aufgeführt.

	normale Simulation		antithet. Var.	
Lauf	Mittelwert	Varianz	Mittelwert	Varianz
1	2.515	0.2666	2.374	0.1409
2	2.544	0.2897	2.374	0.1546
3	2.397	0.1483	2.413	0.1441
4	2.460	0.2017	2.502	0.1785
5	2.479	0.2358	2.455	0.2877
6	2.383	0.2136	2.446	0.1913
7	2.372	0.3680	2.490	0.2657
8	2.432	0.3173	2.485	0.2770
9	2.519	0.2987	2.538	0.2273
10	2.469	0.2073	2.430	0.1615
Mittelwert	2.457	0.2547	2.451	0.2029

Tabelle 10.4: unsere Ergebnisse der M/M/1

Die Varianz der normalen Meßwerte ist kleiner als bei Morgan. Umgekehrtes gilt für die Meßwerte, die sich bei Verwendung der Antithetischen Variablen ergaben. Dementsprechend fällt der Varianzgewinn mit dem Faktor 1.26 sehr gering aus. Unserer Ansicht nach liegt das am verwendeten Zufallsgenerator. Es wurde deshalb eine anderer Zufallszahlengenerator verwendet, der aus [bähr83] stammt:

$$u_{n+1} := (209u_n + 13489) \bmod 2^{16} \tag{10.1}$$

Die Ergnisse von 10 Experimenten unter den gleichen Bedingungen wie oben finden sich in Tabelle 10.5.

[1] Er liefert gleichverteilte natürliche Zahlen aus $[0, 2^{31}-1]$ mit einem kongruenten Verfahren der Zykluslänge 2^{31} [sunos87].

	normale Simulation		antithet. Var.	
Lauf	Mittelwert	Varianz	Mittelwert	Varianz
1	2.445	0.2234	2.578	0.2468
2	2.525	0.2622	2.445	0.1051
3	2.514	0.3558	2.426	0.1606
4	2.449	0.1791	2.487	0.3401
5	2.406	0.1465	2.464	0.1829
6	2.474	0.2216	2.453	0.2635
7	2.527	0.2308	2.463	0.2648
8	2.454	0.2585	2.511	0.2857
9	2.433	0.2468	2.462	0.2995
10	2.484	0.2208	2.468	0.2126
Mittelwert	2.471	0.2346	2.466	0.2362

Tabelle 10.5: Ergebnisse der M/M/1 mit Zufallszahlengenerator von Bähring

Hier sind nun überhaupt keine Unterschiede mehr zwischen dem Verfahren mit und dem ohne Antithetische Variable zu erkennen.
Die Ergebnisse dieses Abschnitts haben uns dazu bewogen, das Verfahren der Antithetischen Variablen nicht in `simsam` zu verwenden.

10.2 Erste Experimente mit Importance Sampling

Um das IS–Verfahren zu testen und einen Überblick über sein Verhalten zu gewinnen, wurde zuerst eine einzelne Komponente simuliert. Danach wurden Untersuchungen an Systemen angestellt, die aus mehreren Komponenten bestanden.
Alle diese Experimente erfolgten mit dafür speziell entwickelten Programmen und nicht mit `simsam`.

10.2.1 Eine Komponente

Als erster Importance–Sampling–Test wurde eine Komponente simuliert, dessen Lebensdauer exponentialverteilt mit Parameter $\lambda = 0.01$ ist (ohne Reparatur). Für IS wurde dieser Parameter mit einem Faktor F multipliziert. Es wurden für verschiedene Werte von F (von 1 an in 4er Schritten erhöht) jeweils die Ausfallwahrscheinlichkeit im Zeitintervall $[0, t = 10]$ als Mittelwert aus 10000 Werten berechnet. Die Mittelwerte für $F \in [0, 69]$ lagen alle im Intervall $[0.092, 0.099]$ um den theoretischen Wert 0.095. Im Abbildung 10.1 sind die Werte der Stich-

probenvarianz in Abhängigkeit von F dargestellt. Der theoretische Wert der Varianz von 0.086 für $F = 1$ ist als waagerechte Linie dargestellt.
Man sieht: Importance Sampling liefert einen Varianzgewinn für $1 < F < 61$. Die minimale Varianz liegt bei $F = 17$, was gemäß $F_{\text{min}} = 1.7/(\lambda * t)$ einer Berechnung in [vdi6 85] entspricht. Sie ist um den Faktor $0.0848/0.0046 = 18.4$ kleiner als die Varianz bei $F = 1$. Grundsätzlich kann man aus dem Diagramm ablesen:
Die Varianz kann sich verringern, wenn seltene Ereignisse häufiger gemacht werden; man darf sie aber nicht zu häufig machen.
Diese Aussage wurde auch durch alle späteren Experimente bestätigt.

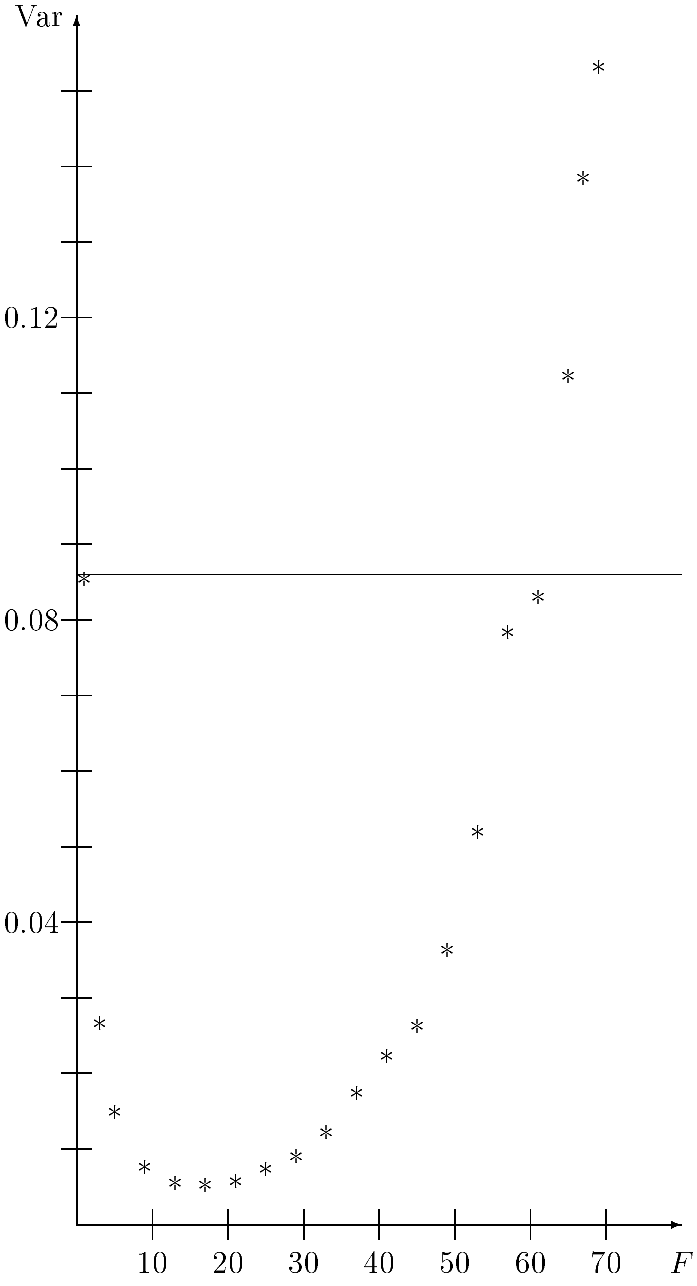

Abbildung 10.1: Varianz der Ausfallwahrscheinlichkeit einer Komponente

10.2.2 Mehrere Komponenten

Um einen besseren Eindruck vom Verhalten des statischen Importance Sampling zu erhalten, haben wir für ein 1–von–2– sowie ein 1–von–3–System die Wahrscheinlichkeit $\mathrm{P}(T_F < t)$ eines Ausfalls bis zur Zeit t ermittelt. Alle Komponenten hatten exponentialverteilte Ausfall– ($\lambda = 0.01$) und Fehlerdauerzeiten ($\mu = 0.1$). Der Parameter t wurde so gewählt, daß die Ausfallwahrscheinlichkeiten von der Größenordnung des oben dargestellten einkomponentigen Systems waren: $t = 50$ für das 1–von–2– und $t = 200$ für das 1–von–3– System. Es wurden jeweils 5000 Meßwerte für verschiedene IS–Ausfallraten $\lambda' = F * \lambda$ ermittelt. In der Tabelle 10.6 sind Mittelwert, Varianz und Vertrauensintervall für beide Systeme, einmal mit und einmal ohne IS, angegeben. Das IS wurde mit dem angegebenen günstigsten Wert für F durchgeführt, welcher wie bei dem einkomponentigen System mittels einer Schleife über verschiedene Faktoren gewonnen wurde (mit Schrittweite 0.1).

	F	Mittelwert P	Varianz P	90% Vertrauensint.
1v2 o. IS	–	0.066	0.0615	± 0.0058
1v2 m. IS	3.5	0.061	0.0112	± 0.0025
1v3 o. IS	–	0.040	0.0386	± 0.0046
1v3 m. IS	1.5	0.037	0.0173	± 0.0031

Tabelle 10.6: $\mathrm{P}(T_F < t)$, 1–von–2–System / 1–von–3–System

Die maximalen Varianzgewinne sind durch den Faktor 5.5 bzw. den Faktor 2.2 gegeben. Wenn man noch das Ergebnis des einkomponentigen Systems hinzunimmt, erkennt man, daß der Varianzgewinn bei steigender Redundanz aber gleichbleibender Größenordung des Meßwertes fällt.

10.2.3 Serien–Parallelsysteme

Bis jetzt wurden die Ausfallraten aller Komponenten für Importance Sampling mit dem gleichen Faktor multipliziert. Im folgenden werden die beiden dreikomponentigen Systeme A und B betrachtet (siehe Abbildung 10.2 und 10.3).
Die Komponenten beider Systeme haben exponentialverteilte Lebens- und Reparataurdauern mit den Parametern Ausfallrate $\lambda = 0.01$ und Reparaturrate $\mu = 0.1$. Wir haben für beide Systeme die Ausfallwahrscheinlichkeit P bei $t = 20$ ermittelt. Es interessierte uns die minimale Varianz dieser Größe bei Verwendung von IS. Dazu wurden einmal für alle Komponenten die gleichen IS–Faktoren $F_1 = F_2 \in \{1.0, 1.2, \ldots, 5.0\}$ und das andere mal F_1 und F_2 unabhängig voneinander aus $\{1.0, 1.2, \ldots, 9.0\}$ verwendet.

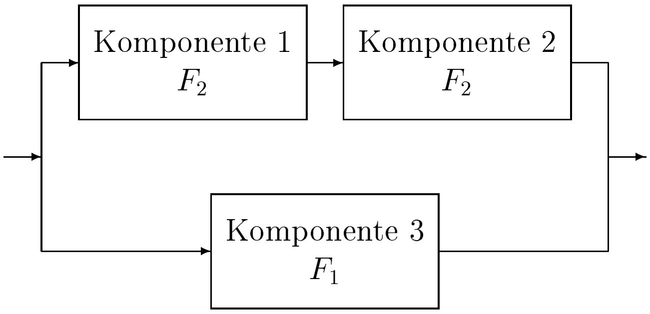

Abbildung 10.2: System A

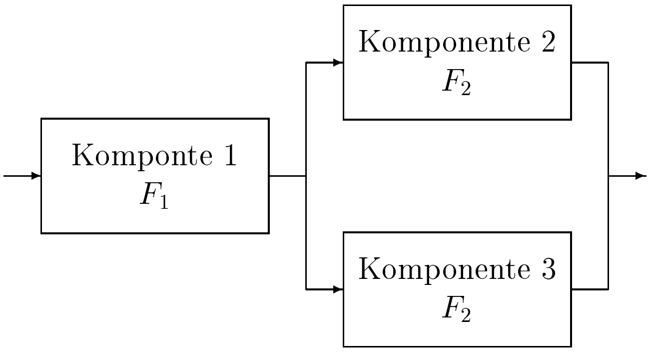

Abbildung 10.3: System B

In Tabelle 10.7 sind die Ergebnisse der Simulation für System A dargestellt: Für die normale Simulation und die beiden jeweils günstigsten IS–Faktoren sind Mittelwert und Varianz von P angegeben.
Wie man sieht, läßt sich durch Verwendung von verschiedenen IS–Faktoren eine kleine Verminderung der Varianz erreichen. Wenn man die Struktur von System A betrachtet, so erkennt man, daß die Komponenten K_1 und K_2 aufgrund ihrer Serienschaltung grob gesagt doppelt so häufig zu Systemausfällen beitragen wie K_3. Insofern erscheint es vernünftig, daß für eine hohe Varianzreduktion F_1 etwa doppelt so groß wie F_2 sein muß, denn auf diese Weise fallen beide Pfade des Blockdiagramms etwa gleichhäufig aus.
Die entsprechenden Ergebnisse für System B sind in Tabelle 10.8 dargestellt.
In diesem Fall ist die Varianz durch Verwendung zweier IS–Faktoren noch mal um den Faktor $0.61/0.25 = 2.44$ geringer als bei Verwendung eines einzigen Faktors. Allerdings verhalten sich die günstigen Faktoren gerade entgegengesetzt zu denen von System A: Die Komponente, die ohnehin mehr zu einem Systemausfall beiträgt, erhält den größeren IS–Faktor.
Bei der Wahl der IS–Parameter kommt es also wesentlich darauf an, die Struktur

	Mittelwert P	Varianz P
ohne IS	$3.46 * 10^{-2}$	$3.37 * 10^{-2}$
$F_1 = F_2 = 3.2$	$3.43 * 10^{-2}$	$0.34 * 10^{-2}$
$F_1 = 8.0, F_2 = 3.6$	$3.55 * 10^{-2}$	$0.29 * 10^{-2}$

Tabelle 10.7: P($T_F < 20$) für System A

	Mittelwert P	Varianz P
ohne IS	$2.04 * 10^{-1}$	$1.62 * 10^{-1}$
$F_1 = F_2 = 3.2$	$1.94 * 10^{-1}$	$0.61 * 10^{-1}$
$F_1 = 6.8, F_2 = 1.8$	$1.98 * 10^{-1}$	$0.25 * 10^{-1}$

Tabelle 10.8: P($T_F < 20$) für System B

des simulierten Systems zu beachten. Im Blockdiagramm parallele Pfade sollten insgesamt gleich häufig ausfallen, da für einen Systemausfall beide Pfade defekt sein müssen. Bei Serienschaltungen reicht es, wenn eine Komponente der Serie ausfällt. Hier ist nicht ganz klar, wie man dann die Parameter wählen soll, aber offenbar ist eine unsymmetrische Struktur günstig.

10.3 Beispielsimulationen mit dem Programm

Zuerst wurden Experimente zu Regenerativer Simulation und dynamischen Importance Sampling durchgeführt. Dann wurde eine größeres Modellsystem simuliert. Weiterhin wurden Experimente mit einer nicht–exponentiellen Verteilung durchgeführt.

10.3.1 Dynamic Importance Sampling

Als erstes wurde ein 1–von–3–System simuliert, dessen Komponenten eine mit $\lambda = 0.01$ exponentialverteilte Lebensdauer hatten. Alle Defekte wurden sofort repariert, die Reparaturdauer war mit $\mu = 0.1$ ebenfalls exponentialverteilt. Im folgenden wurde mittels Regenerativer Simulation die zeitunabhängige stationäre Unverfügbarkeit ermittelt, deren Erwartungswert nach [schnee92] den Wert

$$U = \left(\frac{\lambda}{\lambda + \mu}\right)^3 = 7.51 * 10^{-4}$$

hat. Dazu wurden zuerst die Werte für Zykluszeit T_C und Ausfallzeit T_D durch normale Simulation ermittelt. Dann wurde die Grundstruktur für dynamisches

Importance Sampling (DIS) festgelegt, indem für jede Komponente eine zweite Modifikation eingerichtet wurde, die auch mit $\lambda = 0.01$ exponentialverteilt war. Weiterhin wurde festgelegt, daß der Ausfall jeder Komponente die anderen beeinflußt. Diese Struktur sollte eigentlich nichts an den Ergebnissen ändern. Für DIS wurden dann die Ausfallraten der zweiten Modifikationen mit dem Faktor F multipliziert. Wir haben die Berechnung von T_D erst mit ausgeschaltetem IS, d.h. $F = 1$, und dann mit $F = 3$, $F = 4$ und $F = 5$ durchgeführt. Jeder Wert wurde aus $2 * 10^5$ Meßwerten gemittelt. In Tabelle 10.9 sind für die verschiedenen Experimente die Rechenzeit T_R für je 1000 Meßwerte, die Ausfallzeit, deren Varianz und die 90%–Vertrauensintervalle für U angegeben. Für alle Werte der Verfügbarkeit wurde das anfängliche Meßergebnis der Zykluszeit mit $T_C = 44.36 \pm 0.16$ (95% Vertrauensintervall) verwendet, wobei die 90%–Vertrauensintervalle für U mittels des in Kapitel 6.4 vorgestellten Fehlerfortpflanzungsverfahrens erfolgte.

	T_R/s	Mittelwert T_D	Varianz T_D	Unverfügbarkeit U
ohne IS	5	$3.328 * 10^{-2}$	$2.244 * 10^{-1}$	$(7.50 \pm 0.41) * 10^{-4}$
DIS $F = 1$	10	$3.262 * 10^{-2}$	$2.287 * 10^{-1}$	$(7.35 \pm 0.42) * 10^{-4}$
DIS $F = 3$	28	$3.359 * 10^{-2}$	$5.476 * 10^{-2}$	$(7.57 \pm 0.22) * 10^{-4}$
DIS $F = 4$	28	$3.258 * 10^{-2}$	$4.585 * 10^{-2}$	$(7.34 \pm 0.20) * 10^{-4}$
DIS $F = 5$	28	$3.364 * 10^{-2}$	$6.316 * 10^{-2}$	$(7.58 \pm 0.23) * 10^{-4}$

Tabelle 10.9: Simulation eins 1–von–3–Systems mit DIS

Der theoretische Wert ist in allen 90%–Vertrauensintervallen enthalten. Damit können Rescheduling, bei dem die verschiedenen Modifikationsstufen gleiche Verteilunsfunktionen haben, und DIS den Mittelwert erwartungstreu ermitteln, wie schon vorher gefordert. Hier ist eine deutliche Varianzreduktion durch den Einsatz von DIS zu erkennen (bis zu Faktor 4.9), die allerdings nicht groß genug ist, um den zusätzlichen Rechenzeitbedarf (Faktor 5.6) zu kompensieren. Für dieses kleine System lohnt also der Einsatz von DIS nicht. Allerding ist die Wahl des IS–Parameter, die gemäß der in Kapitel 4.4 angegebenen Regel aus [nicola90] erfolgte, nicht sehr kritisch.

Um einen Vergleich von DIS mit IS durchzuführen, wurde auch für dieses System die Ausfallwahrscheinlichkeit bis $t = 200$, d.h $P(T_F < 200)$, mittels DIS ermittelt. Es ergaben sich die in Tabelle 10.10 aufgeführten Werte. Die Mittelwerte wurden aus je $20 * 10^3$ Werten ermittelt. Für das statische IS wurde der Faktor genommen, der im Kapitel 10.2.2 die größte Varianzreduktion ergab.

In diesem Fall liefert somit die dynamische Form des Importance Sampling eine geringere Varianz als die statische Variante. Der Vergleich der Rechenzeiten für 1000 Werte (ohne IS:15s, IS:75s, DIS:95s) ergibt, daß sich DIS gegenüber dem normalen IS auszahlt. Insgesamt lohnt sich der Einsatz dieser Varianzreduktion

	Mittelwert P	Varianz P	90% Vertrauensintervall
ohne IS	$3.960 * 10^{-2}$	$3.803 * 10^{-2}$	$\pm 0.22 * 10^{-2}$
stat. IS $F = 1.5$	$3.922 * 10^{-2}$	$1.969 * 10^{-2}$	$\pm 0.16 * 10^{-2}$
DIS $F = 3$	$3.924 * 10^{-2}$	$1.045 * 10^{-2}$	$\pm 0.12 * 10^{-2}$
DIS $F = 4$	$3.818 * 10^{-2}$	$1.223 * 10^{-2}$	$\pm 0.13 * 10^{-2}$
DIS $F = 5$	$4.362 * 10^{-2}$	$2.016 * 10^{-1}$	$\pm 0.52 * 10^{-2}$

Tabelle 10.10: P($T_F < 200$), 1–von–3 System, Vergleich IS/DIS

aber wegen der geringen Systemgröße bzw. wegen der Größe des zu ermittelnden Wertes nicht, denn der Faktor der zusätzlich benötigten Rechenzeit ist größer als der Faktor der Varianzreduktion. Außerdem sieht man aus dem Wert für DIS mit $F = 5$ in Tabelle 10.10, daß zu hohe IS–Parameter zu noch schlechteren Ergebnissen führen als die klassische Simulation.

10.3.2 Ein komplexeres System

Um die Leistungsfähigkeit des Programmes einschätzen zu können, wurde folgendes System mit 32 Komponenten, welches [nicola90] entnommen wurde, simuliert. Es ist in Abbildung 10.4 dargestellt.

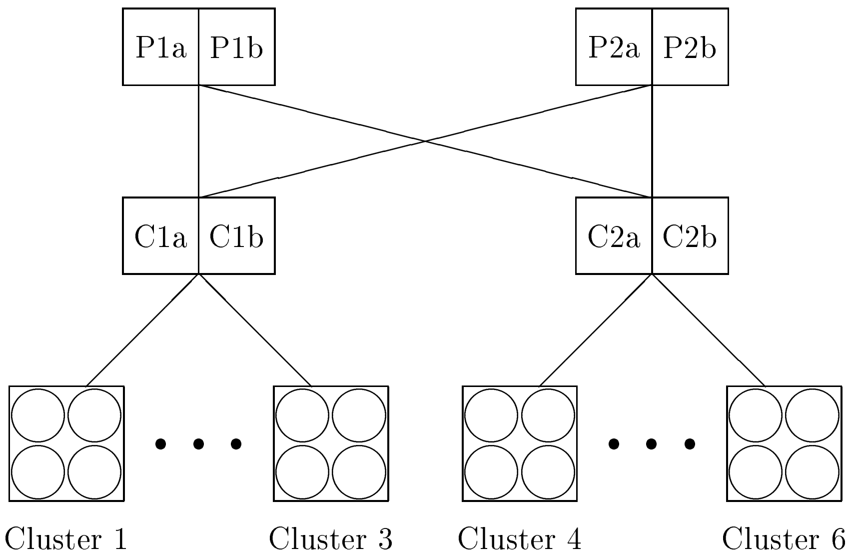

Abbildung 10.4: Computer Modellsystem aus 32 Komponenten

Es besteht aus zwei Prozessorsätzen zu je zwei Prozessoren, die mittels zwei-

er Controllersätze mit je zwei Controllern Zugriff auf 6 Festplattencluster haben. Jeder Festplattencluster besteht aus 4 Platten. Jeweils ein Drittel der „eigentlichen“ Daten jeder Platte befinden sich auf einer anderen Festplatte in ihrem Cluster. Auf diese Weise kann pro Cluster eine Platte ausfallen, ohne daß es zu Datenverlust kommt. Alle Ausfall– und Reparaturverteilungen sind exponentiell. Die Ausfallraten von Prozessoren, Controllern und Platten sind 1/2000, 1/2000 und 1/6000. Fehler werden von einer Reparaturstation in FIFO–Reihenfolge repariert. Das System ist INTAKT, solange ein Prozessor und ein Controller aus jedem Satz und drei Platten aus jedem Cluster INTAKT sind. Man erkennt, daß der Autor die Ausfallraten gerade so gewählt hat, daß bei Ausfall irgendeiner Komponente die Summe der Ausfallraten der Komponenten, von denen noch genau eine für einen Systemdefekt ausfallen muß, immer 1/2000 ist. Auf diese Weise tragen alle drei Arten von Baugruppen gleichmäßig zu Defekten des gesamten Systems bei, und man kann für IS alle Ausfallraten mit dem gleichen Faktor multiplizieren.[2]
Nun wurde die stationäre Unverfügbarkeit U mittels Regenerativer Simulation ermittelt. Zuerst wurden die Zykluszeit T_C und die Ausfallzeit T_D ohne Varianzreduktion als Mittelwert aus je 10^5 Werten ermittelt: Zweimal auf normalem Wege, dazu benötigte das Programm 36 Sekunden für 1000 Werte, und einmal mit der Grundstruktur für DIS (alle Modifikationen verdoppelt + Beeinflussungen), wofür 43 Sekunden für 1000 Werte nötig waren. In der Tabelle 10.11 sind die Mittelwerte, die Varianzen und die sich ergebenden 90%–Vertrauensintervalle für die Unverfügbarkeit angegeben.

	Mittelwert	Varianz	90% Vertrauensint. U
ohne IS 1	$T_D = 4.914 * 10^{-4}$	$8.422 * 10^{-4}$	
	$T_C = 1.257 * 10^2$	$1.564 * 10^4$	$(3.91 \pm 1.22) * 10^{-6}$
ohne IS 2	$T_D = 6.100 * 10^{-4}$	$1.250 * 10^{-3}$	
	$T_C = 1.257 * 10^2$	$1.551 * 10^4$	$(4.85 \pm 1.49) * 10^{-6}$
doppelte Mod.	$T_D = 5.999 * 10^{-4}$	$1.106 * 10^{-3}$	
	$T_C = 1.262 * 10^2$	$1.582 * 10^4$	$(4.75 \pm 1.40) * 10^{-6}$

Tabelle 10.11: Ergebnisse für ein größeres System I

Man erkennt: Alle Vertrauensintervalle enthalten den von Nicola et al. mit einem numerischen Tool (SAVE) ausgerechneten Wert von $U = 4.055 * 10^{-6}$, aber trotz der großen Stichprobe sind die Werte noch sehr ungenau. Man muß aber differenzieren: z.B. gilt für die Werte der zweiten Meßreihe: $T_D = 6.1*10^{-4} \pm 30\%$ und $T_C = 125.7 \pm 0.52\%$ (90%–Vertrauensintervalle). Es wird somit bestätigt:

[2] Nach den Ergebnissen aus Kapitel 10.2.3 könnte es sich lohnen, zu untersuchen, wie sich verschiedene IS–Faktoren für Prozessoren/Controller und Festplatten auswirken.

„Große“ Werte lassen sich wesentlich genauer ermitteln als kleine. Man benötigt deshalb dafür kein Importance Sampling.
und
Bei Regenarativer Simulation sollte ein Großteil der Rechenzeit für die Berechnung der “kleinen“ Werte verwendet werden.
Aus diesem Grund wenden wir im folgenden nur für die Ausfallzeit T_D das dynamische Importance Sampling an. Alle Ausfallraten der zweiten Modifikationen werden mit dem Faktor F multipliziert. Es wurden F–Werte gemäß der im Kapitel 4.4 dargestellten Regel [nicola90] verwendet. Die Mittelwerte wurden wiederum aus 10^5 Werten berechnet. Die für 1000 Durchgänge benötigte Zeit war 130 Sekunden, also das 3.6–fache der normalen Simulationszeit. In Tabelle 10.12 sind Mittelwert und Varianz für T_D und das sich ergebende 90%–Vertrauensintervall für U angegeben. Für letzteren Wert wurde die Zykluszeit aus „ohne IS 2“ verwendet.

F	Mittelwert T_D	Varianz	90% Vertrauensint. U
10	$5.626 * 10^{-4}$	$1.756 * 10^{-4}$	$(4.48 \pm 0.57) * 10^{-6}$
20	$5.404 * 10^{-4}$	$8.557 * 10^{-5}$	$(4.30 \pm 0.40) * 10^{-6}$
40	$5.558 * 10^{-4}$	$1.285 * 10^{-4}$	$(4.42 \pm 0.49) * 10^{-6}$
100	$6.009 * 10^{-4}$	$2.735 * 10^{-4}$	$(4.78 \pm 0.71) * 10^{-6}$

Tabelle 10.12: Ergebnisse für ein großes System II

In allen Vertrauensintervallen bis auf das von $F = 100$ ist der numerische Wert $4.055 * 10^{-6}$ enthalten. Der Vergleich der Varianzen von T_D mit den oben ermittelten Werten ergibt eine deutliche Verbesserung (z.B. bei $F = 20$ mit “ohne IS I“ um den Faktor 9.8). Bei einer Erhöhung der Laufzeit um den Faktor 3.6 lohnt sich somit der Einsatz von DIS. Man benötigt das 2.7–fache der Rechenzeit, um ohne Einsatz von Varianzreduktion die gleiche Genauigkeit zu erhalten. Bei [nicola90] wurde Varianzgewinn durch DIS um den Faktor 12.8 verkleinert, was etwa in der gleichen Größenordnung wie unser Wert liegt. Allerdings wurde deren Wert bei $F = 100$ erreicht, welcher bei uns schon zu schlechten Ergebnissen führte. Eine Erklärung könnte sein: Nicola et al. berücksichtigen möglicherweise beim Rescheduling nicht die bereits zurückliegende Lebensdauer einer Komponente, sie benötigen somit höhere IS–Ausfallraten, um die Ausfallprozesse um den gleichen Betrag zu beschleunigen wie in `simsam`.

10.3.3 Deterministische Werte

Als weiteres wurde versucht für das System mit 32 Komponenten eine Reduktion der Varianz durch Einsatz deterministischer Werte zu erhalten. Wir haben statt der exponentiellen Reparaturverteilung mit Rate 1 die konstante Reparaturzeit

1 verwendet. Es fanden sich bei normaler Simulation aus 10^5 Durchgängen die Mittelwerte $T_D = 2.338 * 10^{-4} \pm 70\%$ und $T_C = 9.664 * 10^1 \pm 0.3\%$ (90% Vertrauensintervalle). Die Zeiten wurden somit nicht richtig ermittelt. Erklärung: Große Reparaturdauern sind zwar unwahrscheinlicher als kleine, tragen aber viel mehr zur Ausfallzeit bei, da wärend einer längeren Zeitspanne der Ausfall einer anderen Komponente wahrscheinlicher ist. Wenn man nur eine Zeit verwendet, wird das nicht berücksichtigt, es gibt weniger Ausfälle. In dieser „brutalen“ Form ist das Verfahren somit wohl nicht geeignet.

10.3.4 Nicht–exponentielle Verteilung

Im folgenden werden die Ergebnisse einer Simulation eines 1–von–3–Systems geschildert, dessen Ausfall– und Reparaturzeiten gammaverteilt sind. Der Formparameter der Gammaverteilung war $\beta = 2.0$. Die Ausfallrate betrug 0.01, die Reparaturrate 0.1. Ermittelt wurde die Ausfallwahrscheinlichkeit $P(T_F < 200.0)$. Es wurden folgende Varianten verwendet:

- Eine normale Simulation ohne jede Varianzreduktion mit $20 * 10^3$ Durchgängen.
- Statisches Importance Sampling, bei dem alle Ausfallraten mit dem Faktor $F = 1.5$ multipliziert wurden ($20 * 10^3$ Durchgänge).
- Dynamisches Importance Sampling, mit den Faktoren $F = 1, 1.5, 2, 3$ (je $5 * 10^3$ Durchgänge).
- Dynamisches Importance Sampling, bei dem für die zweiten Modifikationsstufen die Exponentialverteilung mit den Ausfallraten $\lambda = 0.003$, 0.002, 0.005, 0.01, 0.02 verwendet wurde (je $5 * 10^3$ Durchgänge).

In Tabelle 10.13 sind die Zeiten T_R, die benötigt wurden um 1000 Werte zu berechnen, sowie die Mittelwerte und Varianzen für P dargestellt.
Aus der Tabelle ist ersichtlich, daß für diesen Fall das statische Importance Sampling besser geeignet ist als das dynamische, da letzteres entweder falsche Ergebnisse liefert oder eine kleinere Varianzreduktion zeigt. Ausserdem benötigt DIS für die Gammaverteilung viel Rechenzeit, da dafür die inverse Verteilungsfunktion verwendet wird, welche nur mittels des zeitintensiven Verschachtelungsverfahrens gewonnen werden kann.
Der untere Teil der Tabelle zeigt, daß es ungünstig ist, eine IS–Verteilung zu verwenden, die eine andere Form als die ursprüngliche Verteilung hat. Nur für sehr kleine Parameter stimmen die Ergebnisse einigermassen, was aber mit einer sehr großen Varianz erkauft wird.

		T_R	Mittelwert P	Varianz P
ohne IS		11	$(1.22 \pm 0.13) * 10^{-2}$	$1.21 * 10^{-2}$
stat. IS	$F = 1.5$	70	$(1.27 \pm 0.09) * 10^{-2}$	$0.56 * 10^{-2}$
DIS	$F = 1.0$	320	$(1.32 \pm 0.27) * 10^{-2}$	$1.30 * 10^{-2}$
	$F = 1.5$		$(0.93 \pm 0.13) * 10^{-2}$	$0.31 * 10^{-2}$
	$F = 2.0$		$(0.81 \pm 0.09) * 10^{-2}$	$0.14 * 10^{-2}$
	$F = 3.0$		$(0.59 \pm 0.05) * 10^{-2}$	$0.05 * 10^{-2}$
DIS (exp)	$\mu = 0.002$	170	$(0.98 \pm 0.63) * 10^{-2}$	$7.41 * 10^{-2}$
	$\mu = 0.003$		$(1.07 \pm 0.40) * 10^{-2}$	$2.93 * 10^{-2}$
	$\mu = 0.005$		$(0.81 \pm 0.29) * 10^{-2}$	$0.68 * 10^{-2}$
	$\mu = 0.01$		$(0.59 \pm 0.08) * 10^{-2}$	$0.11 * 10^{-2}$
	$\mu = 0.02$		$(0.38 \pm 0.05) * 10^{-2}$	$0.04 * 10^{-2}$

Tabelle 10.13: $\mathrm{P}(T_F < 20)$ für 1 von 3 System mit Gammaverteilung

Kapitel 11

Zusammenfassung und Ausblick

Die Untersuchung der varianzreduzierenden Verfahren ergab:

- Die Methode der Kontroll–Variablen ist zu sehr von den zu simulierenden Systemen abhängig, also für ein universelles Simulationsprogramm nicht geeignet.
- Antithetische Variablen zeigen nicht die erhofften Varianzreduktionen. Wir vermuten eine zu starke Abhängigkeit von den verwendeten Zufallszahlengeneratoren.
- Der Einsatz deterministischer Werte für einzelne Verteilungen führt zu falschen Ergebnissen.
- Das Verfahren des Importance Sampling eignet sich gut für den Einsatz in einem universellen Simulationsprogramm fehlertoleranter Systeme.

Importance Sampling wurde innerhalb des Programms `simsam` realisiert. Es werden im Prinzip beliebige Ausfall– und Reparaturmechanismen unterstützt. Außerdem ist es möglich, das Ausfallverhalten von Komponenten in Abhängigkeit der Zustände anderer Komponenten zu ändern (Rescheduling). Dadurch ist es möglich die dynamische Variante des IS anzuwenden.
Bei den Untersuchungen des Importance Sampling ergab sich:

- Man muß die IS–Parameter so wählen, daß Systemausfälle wahrscheinlicher, aber nicht zu wahrscheinlich werden. Wählt man die Werte zu hoch, werden die Ergebnisse sogar falsch.
- Die Wahl geeigneter IS–Parameter ist schwierig. Sie hängt von den verwendeten Verteilungen und von der Struktur des Systems ab.
- Für manche Systeme ist dynamisches Importance Sampling besser geeeignet, für andere die statische Variante.

- Man sollte für die IS–Verteilungen nur solche verwenden, die eine ähnliche Form wie die ursprünglichen Systemverteilungen haben.
- Für größere Systeme lohnt sich der Einsatz von IS. Die Rechenzeit konnte bei einem Beispielsystem um den Faktor 2.7 reduziert werden.

Diese Varianzgewinne sind aber noch nicht ganz befriedigend, wie der Vergleich mit der Literatur zeigt. Jedoch muß man bedenken, daß diese Ergebnisse mit Verfahren erzielt wurden, die auf spezielle Systemarten zugeschnitten waren.
Zur Verbesserung der Varianzgewinne ist es nötig, noch genauere Untersuchungen anzustellen, wie man die IS–Parameter günstig zu wählen hat. Man benötigt Vorschriften, die sowohl die Struktur als auch die Parameter der Systeme berücksichtigen. Wie man anhand der Experimente gesehen hat, sind solche Regeln nicht einfach zu gewinnen. Das gilt vor allem deswegen, weil für verschiedene Systeme unterschiedliche Varianten von Importance Sampling am besten geeignet sind.
Eine andere Möglichkeite, zu guten Parametern zu gelangen, ist die Verwendung eines Verfahrens, das mit Startwerten beginnt, die mittels numerischer Tools bestimmt wurden. Es werden damit Simulationen durchgeführt und dabei wiederholt die Parameter in Richtung größerer Varianzgewinne geändert. Außerdem könnte man sich vorstellen, für verschiedene Systeme günstige Parameter mittels großer Testreihen zu ermitteln und diese Parameter–Muster in neuronalen Netzen zu speichern. Für andere Systeme wählen dann die neuronalen Netze günstige Parameter mittels der erlernten Regeln.
Das Programm selbst kann in vielerlei Hinsicht erweitert werden:

- Verwendung weiterer Verteilungsfunktionen (empirische Verteilungen über Tabellen werden zur Zeit realisiert),
- Implementierung anderer Reparaturstrategien,
- Ermittlung weiterer Zuverlässigkeitskenngrößen.

Weiterhin wäre es wünschenswert folgende Punkte zu realisieren. Sie waren eigentlich schon für `simsam` vorgesehen, konnten aber nicht implementiert werden, da die Modellierungsprogramme `rbdedit` und `mdledit` die nötigen Parameter nicht zur Verfügung stellen:

- Ermittlung von Zuverlässigkeitskenngrößen auch für Subsysteme,
- Einführung eines Systemzustandes BLOCKIERT mit eigenen Modifikationsstufen,
- Realisierung eines Systemzustandes RESERVIERT mit einer eigenen Verteilungsfunktion,

- sowie Verwendung von Verteilungen, die durch Approximation von Meßwerten gewonnen wurden (die direkte Darstellung von Meßwerten über Tabellen wird wohl zu langsam sein).

Außerdem wäre es z.B. wünschenswert, das dynamische Importance Sampling direkt zu implementieren, damit die Benutzung vereinfacht und eine größere Zahl von Systementypen verarbeitet werden kann.

Literaturverzeichnis

[bähr83] H. Bähring, Mikroprozessorsystem erzeugt stochastische Impulsfolgen, Elektronik, 19 (1983), S.61ff

[bähr91] H. Bähring, J. Dunkel, The Impact of Load Changing on Lifetime Distributions, Reliability Engineering and System Safety, 31(1991), S.99–110

[berman70] M.B. Berman, Generating Random Variates from Gamma Distributions with Non–Integer Shape–Parameters, Report R–641–PR, The RAND Corporation, Santa Monica, Calif., 1970

[box58] G.E.P. Box, M.E. Muller, A Note on the Generation of Random Normal Deviates, Ann. Math. Stat., 29(1958), S.610–611

[bratle83] P. Bratley, B.L. Fox, L.E. Schrage, A Guide to Simulation, Springer–Verlag, New York, 1983

[carras90] J.A. Carrasco, Fast Simulation of large Markovian Reliability / Aviability Models for Repairable Fault–Tolerant Computer Systems, Proc. 20th Symp. on Fault–Tolerant Computing, 1990

[carras91] J.A. Carrasco, Failure Distance–based Simulation of Repairable Fault–Tolerant Systems, Proc. Modeling Techniques and Tools, 1991

[conway87] A.E. Conway, A. Goyal, Monte Carlo Simulation of Computer Models Availability/Reliability Models, Proc. 17th Symp. on Fault–Tolerant Computing, 1987

[crane75] M.A. Crane, D.L. Iglehart, Simulating Stable Stochastic Systems III: Regenerative Processes and Discrete Event Systems, Oper. Res., 23(1975), No.1, S.33–45

[dugan86] J.B. Dugan et al. The Hybrid Automated Reliability Predictor, A/AA J. Guidance, Control and Dynamics, 9(1986), No.3, S.319–331

[fuhrma88] K. Fuhrmann, Modellierung von Fehlern in fehlertoleranten Systemen unter Berücksichtigung stochastischer Abhängigkeiten, Diplomarbeit am Lehrgebiet Technische Informatik I der FernUniversität Hagen, 1988

[geist89] R.M. Geist, M.K. Smotherman, Ultrahigh Reliability Estimates through Simulation, Proc. annual Reliability and Maintainability Symp., 1989

[goyal86] A. Goyal, W.C. Carter, E. de Souza e Silva, S.S. Lauenburg, The System Availability Estimator, IEEE Transactions on Computers, 1986

[goyal92] A. Goyal, P. Shahabuddin, P. Heidelberger, V.F. Nicola, P.W. Glynn, A Unified Framework for Simulating Markovian Models of Highly Dependable Systems, IEEE Transactions on Computers, 1992

[haas87] P.J. Haas, G.S. Shedler, Regenerative Generalized Semi–Markov Processes, Commun. Statist.–Stochastic Models, 3(1987), S. 409–438

[hochst91] D. Hochstädter, Statistische Methodenlehre, Verlag Harri Deutsch, Frankfurt am Main, 1991

[hull62] T.E.Hull, A.R. Dobell, Random Number Generators, SIAM Rev., 4(1962), 230-254

[jöhnk64] M.D. Jöhnk, Erzeugung von betaverteilten und gammaverteilten Zufallszahlen, Metrika, 8(1964), S.5–15

[jordan78] G. Jordan–Engeln, Numerische Mathematik für Ingenieure, Bibliogr. Inst., Mannheim, 1978

[löhr90a] K. Löhring, Entwurf und Implementierung eines grafischen Editors zur Bearbeitung von Zuverlässigkeits–Blockschaltbildern, Diplomarbeit am Lehrgebiet Technische Informatik I der FernUniversität Hagen, 1990

[löhr90b] K. Löhring, Handbuch zum Blockschaltbildeditor, Lehrgebiet Technische Informatik I der FernUniversität Hagen, 1990

[lück88] C. Lücking, Interaktive Simulation von Zuverlässigkeitsnetzwerken, Diplomarbeit am Lehrgebiet Technische Informatik I der FernUniversität Hagen, 1988 [moesch83] O. Moeschlin, W.Eberl, Wahrscheinlichkeitstheorie I, Kurs 1262 der FernUniversität Hagen, 1983

[morgan84] B.J.T. Morgan, Elements of Simulation, Chapman and Hall, London, 1984

[nicola90] V.F. Nicola, M.K. Nakayama, P. Heidelberger, A. Goyal, Fast Simulation of Dependability Models with General Failure, Repair and Maintenance Processes, IEEE Transactions on Computers, 1990

[odeh74] R.E. Odeh, J.O. Evans, Algorithm AS70: Percantage Points of the Normal Distribution, Appl.Stat., 23(1974), S.96–97

[ruenge92] I. Rüngeler, Entwurf und Implementierung eines Editors zur Beschreibung des Zuverlässigkeitsverhaltens von hierarchisch strukturierten Systemen, Diplomarbeit am Lehrgebiet Technische Informatik I der FernUniversität Hagen, 1992

[schnee91] W.G. Schneeweiss, Grundbegriffe für praktische Zuverlässigkeitsanalysen, Datakontext–Verlag, Köln, 1991

[schnee92] W.G. Schneeweiss, Zuverlässigkeitstechnik – von den Komponenten zum System, Datakontext–Verlag, Köln, 1992

[smith55] W.L. Smith, Regenerative Stochastic Processes, Proc. Royal Society A, 232(1955), S.6–31

[sobol 74] I.M. Sobol, Die Monte–Carlo–Methode, Deutscher Verlag der Wissenschaften, Berlin, 1974

[sunman86] Sunview Programmers Guide, Sun Microsystems, 1986

[sunos87] SunOS Reference Manual 3, Sun Microsystems, 1987

[vdi1 86] Voraussetzungen und Anwendungsschwerpunkte von Zuverlässigkeitsanalysen, VDI Normen 4008 Blatt 1, 1986

[vdi6 85] Monte Carlo Simulation, VDI Normen 4008 Blatt 6, 1985

Anhang A

Ergänzungen zur Zuverlässigkeitstechnik

A.1 Zuverlässigkeitsblockdiagramme

Im folgenden wird der Aufbau von *Zuverlässigkeitsblockdiagrammen* und der Begriff des *n–von–m–Systems* anhand von Beispielen erläutert.
Ein n–von–m–System ist ein System mit m Komponenten, von denen mindestens n INTAKT sein müssen, damit das System funktioniert.
Ein Zuververlässigkeitsblockdiagramm ist ein Graph mit einem Anfangs– und einem Endknoten. Auf den Kanten liegen Komponenten. Eine Kante ist nicht passierbar, wenn die zugehörige Komponente DEFEKT ist. Das durch den Graphen dargestellt System ist DEFEKT, falls es keinen passierbaren Pfad durch den Graphen gibt.
Ein System, das z.B. aus drei Komponenten K_1, K_2, K_3 besteht, die alle drei INTAKT sein müssen, damit das System funktioniert, wird in Form dreier Kästchen auf einem Pfad dargestellt (Abbildung A.1). Um den Pfad in dem Blockschaltbild zu durchlaufen, muß man alle drei Kästchen passieren. Man spricht dabei von einer *Serienstruktur*, das System nennt man *3–von–3–System*. Die Systemfunktion ist in diesem Fall die Disjunktion der Komponentenzustände: $Z = Z_1 \vee Z_2 \vee Z_3$.

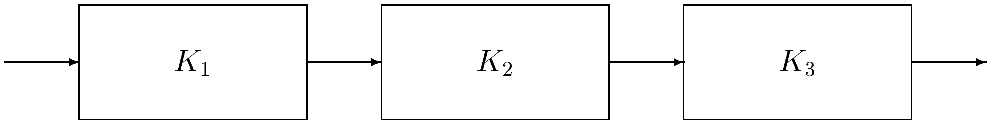

Abbildung A.1: Blockschaltbild eines 3–von–3–Systems

Dieses System hat den Nachteil, das es bereits bei dem Ausfall einer Komponente selbst ausfällt. Eine solche Systemstruktur ist für hohe Anforderungen an

die Zuverlässigkeit ungeeignet. Man führt deshalb Ersatzkomponenten ein. Man sagt dann, das System ist *redundant*.
Ein Beispiel eines redundanten Systems ist das sog. *1–von–3–System*. Es besteht aus 3 gleichwertigen Komponenten, von denen mindestens eine funktionieren muß, damit das System INTAKT ist. Die Systemfunktion ist die Konjunktion der Komponentenzustände: $Z = Z_1 \wedge Z_2 \wedge Z_3$. Als Blockschaltbild wird ein solches System, welches man auch *Parallelsystem* nennt, wie in Abbildung A.2 dargestellt: Damit man den Graphen von links nach rechts durchlaufen kann, muß mindestens einer der drei Pfade passierbar sein, d.h. mindestens eine Komponente muß INTAKT sein.

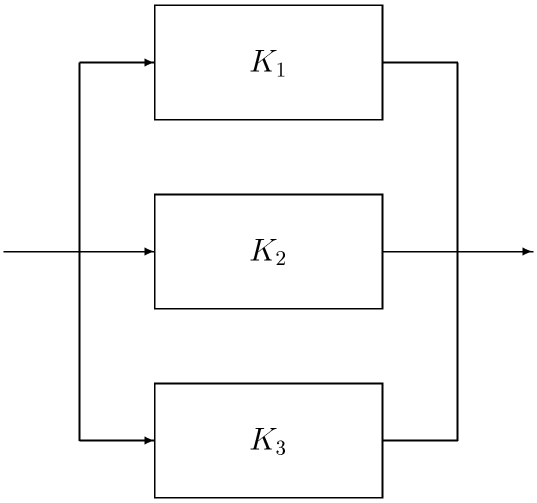

Abbildung A.2: Blockschaltbild eines 1–von–3–Systems

Blockschaltbilder beliebiger Modellsysteme kann man leicht als Kombination von Serien– und Parallelsystemen aufbauen. Manche Blockschaltbilder lassen sich kürzer darstellen, wenn man von dieser Form abweicht. Das führt zu sog. *Brückensystemen*, bei denen es Komponenten auf Kanten gibt, die in beiden Richtungen passierbar (also ungerichtet) sind.

A.2 Definitionen der Kenngrößen

Es folgen die Definitionen der wichtigsten Zuverlässigkeitskenngrößen nach DIN 40041.
Transiente Größen:

- Die *Verfügbarkeit* ist die Wahrscheinlichkeit, das System zu einem bestimmten Zeitpunkt im Zustand INTAKT anzutreffen. Das Komplement dieser Größe nennt man *Unverfügbarkeit*. Im Englischen werden diese Größen bei fehlender Reparatur *(un–) reliability* genannt.

- Die *Ausfallwahrscheinlichkeit* (engl. *probability of failure*) ist die Wahrscheinlichkeit, daß das System vom Anfangszustand an bis zu einem bestimmten Zeitpunkt ausfällt. Das Komplement dieser Größe heißt *Überlebenswahrscheinlichkeit*.

- Die *Lebensdauer* ist die Zeit von der Inbetriebnahme eines nicht reaparierbaren Systems bis zum Ausfall. Der mathematische Erwartungswert der Lebensdauer heißt *mittlere Lebensdauer* (engl. *mean life*).

Stationäre Kenngrößen:

- Der *Ausfallabstand* (engl. *time between failures* = TBF) ist die Zeit zwischen zwei Systemausfällen. Der Erwartungswert ist der *mittlerer Ausfallabstand* (engl. *mean time between failures* = MTBF).

- Die *Ausfalldauer* (engl. *down time* = DT oder time to repair = TTR) ist die Zeit vom Ausfall des Systems bis es wieder INTAKT ist. Entsprechend ist die *mittlere Ausfallzeit* (engl. *mean down time* = MDT oder *mean time to reapair* = MTTR) definiert.

- Die *Brauchbarkeitsdauer* (engl. *time to failure* = TTF) ist die Zeit, die ein System ununterbrochen INTAKT ist. Ihr Erwartungswert heißt mittlere Brauchbarkeitsdauer (MTTF).

- Die *Verfügbarkeit* kann man auch eine stationäre Größe sein. Im Englischen wird sie dann mit *availability* bezeichnet. Außerdem kann man die Verfügbarkeit auch völlig zeitunabhängig als Anteil der Betriebszeit, die das System INTAKT ist, ansehen. Sie wird dann im Englischen *steady state availability* genannt.

Anhang B

Grundlagen der Wahrscheinlichkeitstheorie und Statistik

Man betrachte eine Größe K, deren Verhalten zufällig ist, d.h. wenn man eine Messung dieser Größe vornimmt, ist das Ergebnis nicht vorhersehbar. Unter „Messung" verstehen wir im folgenden sowohl reale Messungen in der Umwelt als auch Ergebnisse von Simulationen auf Rechenanlagen mittels (Pseudo–) Zufallszahlengeneratoren. Ein Beispiel einer solchen sog. *Zufallsgröße* (kurz: ZG) ist die Augenzahl beim Wurf eines Würfels. Solche Zufallsgrößen, die nur endlich oder abzählbar viele Werte annehmen können, nennt man *diskret*, sonst *kontinuierlich*.
Ein Beispiel einer kontinuierlichen ZG erhält man mittels einer gleichmaßig mit Masse belegten runden Scheibe, die in ihrem Mittelpunkt drehbar gelagert ist und am Rand eine Marke besitzt. Wenn man nun die Scheibe mittels eines starken Stoßes in Rotation versetzt, so kommt sie irgendwann zur Ruhe. Die Größe K kann in diesem Fall der Winkel sein, der sich zwischen der Position der Marke und einer ausserhalb der Scheibe angebrachten Markierung — vom Scheibenmittelpunkt gemessen — ergibt. Diese ZG kann im Bogenmaß alle Werte aus dem Intervall $[0, 2\pi[$ annehmen.
Im Zusammenhang mit dem Simulationsprogramm `simsam` sind nur kontinuierlich Zufallsgrößen von Interesse. Aufgrund des endlichen Wertebereichs der Zahlendarstellung im Computer muß man die kontinuierlichen Zufallsgrößen durch diskrete mit einem sehr großem Wertebereich ausdrücken.
Da eine ZG ein nichtdeterminiertes Verhalten hat, ist das Ergebnis eines einzelnen Experimentes nicht von Interesse (außer bei den neuesten Lottozahlen). Man muß eine größere Zahl n (genannt *Stichprobenumfang*) von Experimenten durchführen, um eine das Verhalten einer ZG zu beschreiben. Das Ergebnis dieser Experimente sind n Meßwerte $k_1, \ldots, k_n$. Wichtige Kenngrößen der Meßreihe sind *Mittelwert* $\overline{K}$ und *Stichprobenvarianz* S^2:

$$\overline{K} = \frac{1}{n}\sum_{i=1}^{n} k_i \tag{B.1}$$

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(k_i - \overline{K}\right)^2 \tag{B.2}$$

Der Mittelwert gibt an, welches „durchschnittliche" Meßergebnis man erzielt hat. Die Stichprobenvarianz ist ein Maß dafür, wie stark die einzelnen Meßwerte um den Mittelwert streuen.
Das Verhalten einer ZG läß sich mittels der sogenannten *Dichtefunktion* f vollständig beschreiben. Der Wert $f(k)\,dk$ gibt an, mit welcher Wahrscheinlichkeit ein Experiment ein Ergebnis aus dem infinitesimalen Intervall der Größe dk um den Wert k liefert. Es muß somit $f(k) \geq 0 \quad \forall\, k$ gelten.
Die Wahrscheinlichkeit $\mathrm{P}\,(k_1 < k \leq k_2)$, daß das Ergebnis eines Zufallsexperiments im makroskopischen Intervall $]k_1, k_2]$ liegt, ist somit:

$$\mathrm{P}\,(k_1 < k \leq k_2) = \int_{k_1}^{k_2} f(k)\,dk \tag{B.3}$$

Da das Ergebnis jedes Experiment mit Sicherheit irgendeine Zahl liefert, muß gelten:

$$\int_{-\infty}^{+\infty} f(k)\,dk = 1 \tag{B.4}$$

Für das Beispiel der gedrehten Scheibe lautet die Dichtefunktion:

$$f_s(k) = \begin{cases} 1/2\pi & \text{falls} \quad k \in [0, 2\pi[\\ 0 & \text{sonst} \end{cases} \tag{B.5}$$

Für Dichtefunktionen dieser Art, bei denen die Wahrscheinlichkeitsdichte über einem Intervall konstant ist, hat man einen besonderen Namen: *Gleichverteilung*. Man bezeichnet sie mit $\mathrm{U}(k_1, k_2)$ wobei k_1 und k_2 die Intervallgrenzen sind. In obigen Fall liegt somit eine $\mathrm{U}(0, 2\pi)$–Verteilung vor. Wichtig für die Erzeugung von Zufallszahlen ist $\mathrm{U}(0, 1)$, da man durch Transformtion dieser Standardverteilung alle Verteilungen realisieren kann (siehe Kapitel 8).
Eine wichtige, von der Dichtefunktion abgeleitete Größe, ist die Verteilungsfunktion F, die wie folgt definiert ist:

$$F(k') := \mathrm{P}(k \leq k') = \int_{-\infty}^{k'} f(k)\,dk \tag{B.6}$$

Damit ist dann:

$$\mathrm{P}(k_1 < k \leq k_2) = F(k_2) - F(k_1) \tag{B.7}$$

Mittels der Dichtefunktion kann man den *erwarteten* Mittelwert, den sog. Erwartungswert $\mu := \mathrm{E}(K)$, und die erwartete Stichprobenvarianz, genannt Varianz $\sigma^2 := \mathrm{Var}(K) := \mathrm{E}([K - \mathrm{E}(K)]^2)$, berechnen. Das sind die Werte, die sich bei beliebig vielen Experimenten ergeben würden, d.h. im Grenzfall $n \to \infty$. Es gilt:

$$\mathrm{E}(K) = \int_{-\infty}^{+\infty} k f(k)\, dk \tag{B.8}$$

Wie man sieht, ist die Größe $\mathrm{E}(K)$ linear in K ,d.h. falls K_1, K_2 Zufallsgrößen und a_1, a_2 reelle Zahlen sind, so ist $\mathrm{E}(a_1 K_1 + a_2 K_2) = a_1 \mathrm{E}(K_1) + a_2 \mathrm{E}(K_2)$. Weiterhin gilt dann:

$$\sigma^2 = \int_{-\infty}^{+\infty} (k - \mathrm{E}(K))^2 f(k)\, dk \tag{B.9}$$

Eine häufig benutzte Darstellung der Varianz erhält man unter Ausnutzung der Linearität des Erwartungswerts:

$$\begin{aligned} \mathrm{Var}(K) &= \mathrm{E}([K - \mathrm{E}(K)]^2) \\ &= \mathrm{E}(K^2 - 2K\mathrm{E}(K) + \mathrm{E}(K)^2) \\ &= \mathrm{E}(K^2) - 2\mathrm{E}(K)\mathrm{E}(K) + \mathrm{E}(K)^2 \\ &= \mathrm{E}(K^2) - \mathrm{E}(K)^2 \end{aligned} \tag{B.10}$$

Oft ist man an der Varianz von zusammengesetzten Kenngrößen interessiert. Am wichtigsten ist die Summe zweier Kenngrößen $K + L$. Es gilt dann (unter Verwendung der Linearität der Erwartungswertbildung):

$$\begin{aligned} \mathrm{Var}(K + L) &= \mathrm{E}([K + L - \mathrm{E}(K) - \mathrm{E}(L)]^2) \\ &= \mathrm{E}([K - \mathrm{E}(K)]^2) + \mathrm{E}([L - \mathrm{E}(L)]^2) + \\ &\quad\ 2\mathrm{E}([K - \mathrm{E}(K)][L - \mathrm{E}(L)]) \end{aligned}$$

Mittels der Definition $\mathrm{Cov}(K, L) := \mathrm{E}([K - \mathrm{E}(K)][L - \mathrm{E}(L)])$ (Cov bedeutet *Kovarianz*) ergibt sich somit:

$$\mathrm{Var}(K + L) = \mathrm{Var}(K) + \mathrm{Var}(L) + 2\mathrm{Cov}(K, L) \tag{B.11}$$

Man nennt Zufallsgrößen, deren Kovarianz kleiner Null ist, *negativ korreliert* (bei größer Null, *positiv korreliert*). Korrelierte Zufallsgrößen sind nicht unabhängig voneinander. Zwischen ihnen besteht ein deterministischer (Teil–) Zusammenhang.

Zum Schluß wird noch das Konzept der *bedingten Wahrscheinlichkeit* erläutert. Gegeben seien N Zufallsexperimente, bei denen N_A–mal das Ergebnis A, N_B–mal das Ergebnis B und $N_{A\wedge B}$–mal sowohl A als auch B ermittelt wurden. Beispiel: Man steht an einer Ampel und betrachtet N Autos, die sie passieren. A sei das Ereignis „Auto fährt ohne Halt an der Ampel vorbei“ und B bedeute „Die Ampel ist gerade rot“. Somit ist $N_{A\wedge B}$ die Anzahl der Autos, die die Ampel bei Rot passieren.
Die relativen Häufigkeiten für die Ereignisse A und B bei den Messungen sind einfach:

$$\begin{aligned} \mathrm{h}_N(A) &= N_A/N && \text{(B.12)} \\ \mathrm{h}_N(B) &= N_B/N && \text{(B.13)} \\ \mathrm{h}_N(A \wedge B) &= N_{A\wedge B}/N && \text{(B.14)} \end{aligned}$$

Im Rahmen des *statistischen Wahrscheinlichkeitsbegriffs* erhält man durch Bildung des Grenzwertes (nicht im Sinne der Analysis) von $N \to \infty$ die Wahrscheinlichkeiten P:

$$\begin{aligned} \mathrm{P}(A) &= \lim_{N\to\infty} \mathrm{h}_N(A) && \text{(B.15)} \\ \mathrm{P}(B) &= \lim_{N\to\infty} \mathrm{h}_N(B) && \text{(B.16)} \\ \mathrm{P}(A \wedge B) &= \lim_{N\to\infty} \mathrm{h}_N(B) && \text{(B.17)} \\ & && \text{(B.18)} \end{aligned}$$

Besonders interessant ist die Wahrscheinlichkeit dafür, daß ein Auto an der Ampel vorbeifährt, unter der Bedingung, daß die Ampel rot ist. Man interessiert sich also für den Anteil der Verkehrsrowdies. Diese bedingte Wahrscheinlichkeit bezeichnet man mit $\mathrm{P}(A|B)$. Es gilt (mittels den Gleichungen B.12 und B.14) für die relative bedingte Häufigkeit $\mathrm{h}_N(A|B)$:

$$\mathrm{h}_N(A|B) = \frac{N_{A\wedge B}}{N_B} = \frac{N_{A\wedge B}}{N}\frac{N}{N_B} = \frac{\mathrm{h}_N(A\wedge B)}{\mathrm{h}_N(B)} \qquad \text{(B.19)}$$

Wiederum erhält man durch den Grenzübergang $N \to \infty$ die entsprechenden Wahrscheinlichkeiten:

$$\mathrm{P}(A|B) = \frac{\mathrm{P}(A\wedge B)}{\mathrm{P}(B)} \qquad \text{(B.20)}$$

Diese Darstellung der bedingten Wahrscheinlichkeit als Quotient kann man verwenden, um die Größe *Fehler*– oder *Ausfallrate* einzuführen. Wenn eine Zufallsgröße mit Dichtefunktion f und Verteilungsfunktion F die Lebensdauer von

Komponenten beschreibt, so ist Fehlerrate der Anteil der pro Zeiteinheit ausfallenden Komponenten von den Komponenten, die noch intakt sind. Die Fehlerrate $\lambda(t)$ ist somit die Wahrscheinlichkeitsdichte für einen Ausfall zur Zeit t unter der Bedingung, daß die Komponente bis zur Zeit t nicht ausgefallen ist. Damit ergibt sich:

$$\lambda(t) := \frac{f(t)}{1 - F(t)} \tag{B.21}$$

Analog sind *Reparaturrate* bzw. ganz allgemein die *Ereignisrate* definiert.

Anhang C

Ergänzungen zur Auswertung

C.1 Ermittlung von Mittelwert und Varianz

Das Ergebnis eines Simulationslaufs mit n Durchgängen für eine Kenngröße K sind die Werte $k_1, \ldots, k_n$.
Stichprobenmittel $\overline{K}$ und Stichprobenvarianz S^2 werden wie folgt berechnet: Während der Simulation werden laufend die Größen n, $s_k = \sum_{i=1}^{n} k_i$ und $s_{k^2} = \sum_{i=1}^{n} {k_i}^2$ durch Aufsummation ermittelt. Es gilt dann

$$\overline{K} = \frac{s_k}{n} \tag{C.1}$$

und

$$S^2 = \frac{s_{k^2} - {s_k}^2/n}{n-1} \tag{C.2}$$

Die Darstellung in Gleichung C.2 ergibt sich mit wenigen Umformungen (mit $\overline{K} = \sum_{i=1}^{n} k_i/n$):

$$\begin{aligned}
\frac{s_{k^2} - {s_k}^2/n}{n-1} &= \frac{\sum_{i=1}^{n} {k_i}^2 - \left(\sum_{i=1}^{n} k_i\right)^2/n}{n-1} \\
&= \frac{\sum_{i=1}^{n} {k_i}^2 - 2\overline{K}\sum_{i=1}^{n} k_i + \sum_{i=1}^{n} \overline{K}^2}{n-1} \\
&= \frac{\sum_{i=1}^{n} \left(k_i - \overline{K}\right)^2}{n-1} \\
&= S^2
\end{aligned}$$

Der Vorteil dieser Methode ist, daß pro Kenngröße nur drei Werte gespeichert werden müssen anstatt $n+1$ Werte, die bei direkter Speicherung von n und den Werten $k_1, \ldots, k_n$ nötig wären. Außerdem ist die Anzahl der Stichproben nicht durch die Speichergröße oder die Größe eines Arrays begrenzt.

C.2 Berechnung eines Vertrauensintervalls

Es sei I das Vertrauensintervall $[\overline{K}-\Delta_m, \overline{K}+\Delta_m[$. Es wird nun gezeigt, wie Δ_m gewählt werden muß, damit der Erwartungswert μ der Kenngröße K mit der Wahrscheinlichkeit β in I liegt.

$$
\begin{aligned}
& \mathrm{P}\left(\overline{K}-\Delta_m \leq \mu < \overline{K}+\Delta_m\right) && = \beta \\
\Leftrightarrow \quad & 1-\left[\mathrm{P}\left(\mu < \overline{K}-\Delta_m\right)+\mathrm{P}\left(\overline{K}+\Delta_m \leq \mu\right)\right] && = \beta \\
\Leftrightarrow \quad & 1-\left[\mathrm{P}\left(\overline{K}-\mu > \Delta_m\right)+\mathrm{P}\left(\overline{K}-\mu \leq -\Delta_m\right)\right] && = \beta
\end{aligned}
$$

Zweckmäßigerweise wählt man nun $\Delta_m = x_m\sigma/\sqrt{n}$ und wendet den zentralen Grenzwertsatz an:

$$
\begin{aligned}
\Leftrightarrow \quad & 1-\left[\mathrm{P}\left(\tfrac{\overline{K}-\mu}{\sigma/\sqrt{n}} > x_m\right)+\mathrm{P}\left(\tfrac{\overline{K}-\mu}{\sigma/\sqrt{n}} \leq -x_m\right)\right] && = \beta \\
\Leftrightarrow \quad & 1-\left[1-\mathrm{N}_{0,1}(x_m)+\mathrm{N}_{0,1}(-x_m)\right] && = \beta
\end{aligned}
$$

Aufgrund der Symmetrie von $f_{0,1}(x)$ kann man $\mathrm{N}_{0,1}(x)-\mathrm{N}_{0,1}(-x) = 2\mathrm{N}_{0,1}(x)-1$ verwenden und dann die Normalverteilung invertieren:

$$
\begin{aligned}
\Leftrightarrow \quad & 2\mathrm{N}_{0,1}(x_m)-1 && = \beta \\
\Leftrightarrow \quad & x_m && = \mathrm{N}_{0,1}^{-1}\left(\frac{\beta+1}{2}\right)
\end{aligned}
\tag{C.3}
$$

C.3 Akzeptierung einer Hypothese

Es wird gezeigt, wie man eine rechtsseitige Hypothese daraufhin prüft, ob man sie akzeptieren kann:

$$
\begin{aligned}
& \mathrm{P}\left(\mu \geq H\right) && \geq \beta \\
\Leftrightarrow \quad & \mathrm{P}\left(-\mu \leq -H\right) && \geq \beta \\
\Leftrightarrow \quad & \mathrm{P}\left(\tfrac{\overline{K}-\mu}{\sigma/\sqrt{n}} \leq \tfrac{\overline{K}-H}{\sigma/\sqrt{n}}\right) && \geq \beta
\end{aligned}
\tag{C.4}
$$

Wieder wird die Näherung des zentralen Grenzwertsatzes verwendet:

$$
\begin{aligned}
\Leftrightarrow \quad & \mathrm{N}_{0,1}\left(\tfrac{\overline{K}-H}{\sigma/\sqrt{n}}\right) && \geq \beta \\
\Leftrightarrow \quad & \overline{K}-H && \geq \mathrm{N}_{0,1}^{-1}(\beta)\sigma/\sqrt{n}
\end{aligned}
$$

Mit den Abkürzungen $x_r = \mathrm{N}_{0,1}^{-1}(\beta)$ und $\Delta_r = x_r \sigma/\sqrt{n}$ wird somit die Hypothese $\mu \geq H$ akzeptiert, falls gilt:

$$\overline{K} - H \geq \Delta_r \tag{C.5}$$

C.4 Berechnung der Gütefunktion

Die Gütefunktion für den rechtsseitigen Test kann man wir folgt berechnen (ähnlich wie in [hochst91]): Da eine rechtseitige Hypothese bei $\overline{K} - H < \Delta_r$ abgelehnt wird, gilt (Δ_r, x_r: siehe Kapitel 6):

$$\begin{aligned} \mathrm{G}_r(\mu) &= \mathrm{P}\left(\overline{K} - H < \Delta_r\right) \\ &= \mathrm{P}\left(\overline{K} - \mu < \Delta_r + H - \mu\right) \\ &= \mathrm{P}\left(\frac{\overline{K} - \mu}{\sigma/\sqrt{n}} < x_r + \frac{H - \mu}{\sigma/\sqrt{n}}\right) \end{aligned}$$

Wieder wird die Wahrscheinlichkeit P aufgrund des zentralen Grenzwertsatzes durch die Normalverteilung ausgedrückt:

$$\mathrm{G}_r(\mu) = \mathrm{N}_{0,1}\left(x_r + \frac{H - \mu}{\sigma/\sqrt{n}}\right) \tag{C.6}$$

Anhang D

Systemvoraussetzungen und Installation

Das Programm `simsam` läuft auf Workstations der Firma Sun, Baureihe 3/xx mit einem Mikroprozessor MC680x0, Coprozessor MC68881, mind. 4MB Speicher und dem Fenstersystem SunView unter SunOS 4.1.1.
Zum Programmbetrieb muß das File `simsam` in irgendein Verzeichnis kopiert werden. Das Programm dient dazu, Modellsysteme hinsichtlich ihrer Zuverlässigkeitseigenschaften zu simulieren. Dazu müssen auf dem Rechner die Programme `rbdedit` [löhr90a] und `mdledit` [ruenge92] installiert sein. Diese Programme dienen dazu, solche Modellsysteme einzugeben. Insbesondere müssen die Environmentvariablen `MDLEDIT_DIR` und `MDLDATA`, welche auch für `mdledit` benötigt werden, gesetzt sein. Sie geben die Pfade an, unter denen das Programm die Daten der Modellsysteme findet.
Das Programm wird durch Eingabe von „`simsam &`" im Hintergrund gestartet. Dazu muß man sich in dem Verzeichnis befinden, in dem das Programm steht, oder dieses Verzeichnis muß im Suchpfad angegeben sein. Es ist empfehlenswert, gleichzeitig mit dem Simulationsprogramm auch die beiden Programme zur Eingabe der Systeme laufen zu lassen, da man so den besten Überblick über die Simulation behält und auch Änderungen effektiv durchführen kann. Aber Achtung: Ändert man Parameter des Systems, so muß es in den Editorprogrammen abgespeichert und in `simsam` neu geladen werden. Davor sollte man die IS–Parameter sichern, da sie beim Laden zurückgesetzt werden.

Anhang E

Verfahren für Verteilungsfunktionen

E.1 Numerische Integration

Numerische Integrationen werden in dem Programm durchgeführt, indem die zu integrierende Funktion durch eine Stufenfunktion approximiert und das Integral durch eine Summe ersetzt wird (siehe Abbildung E.1).

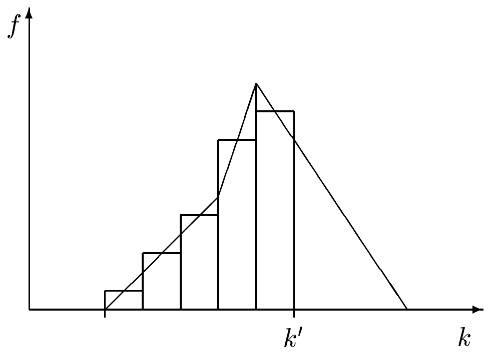

Abbildung E.1: numerische Integration einer Dichtefunktion

Bei der Integration einer Dichtefunktion müßte man bei $-\infty$ beginnen. Da das Integral über der Dichtefunktion über alle Werte aber 1 ergibt, muß die Dichtefunktion im negativen Unendlichen gegen 0 konvergieren. Man kann sich somit einen Startpunkt aussuchen, von dem an f einen signifikant großen Wert erreicht und ab dort integrieren. In dem Simulationsprogramm repräsentieren die Verteilungen Lebens– und Reparaturdauern, welche naturgemäß positiv sind. Wir wählen somit als Startpunkt den Wert $k = 0$.
Teilt man das Integrationsintervall $[0, k']$ in N Teilintervalle, so hat jedes eine Breite von $\Delta = k'/N$. Die Idee der numerische Intgration liegt nun darin, die

Dichtefunktion in einem Teilintervall als konstant anzusehen, z.B. kann man den Wert der Funktion in der Teilintervallmitte nehmen. Damit ergibt sich folgende Näherung der Verteilungsfunktion, die umso genauer ist, je größer N ist:

$$F(k') \approx \sum_{i=1}^{N} f\left((i-0.5)\Delta\right)\Delta \tag{E.1}$$

E.2 Die Gaußsche Fehlerquadratmethode

Es wird nun gezeigt, wie man eine nicht geschlossen darstellbare Funktion $h(x)$ mittels einer Linearkombination von Funktionen $\phi_1, \ldots, \phi_n$ approximieren kann:

$$h(x) \approx \sum_{k=1}^{n} c_k \phi_k(x) =: \Phi(x) \tag{E.2}$$

Man geht davon aus, daß das System der Funktionen ϕ_k gegeben ist, und sucht nun eine Bedingung für die Wahl der Linearfaktoren $c_1, \ldots, c_n$. Dazu betrachtet man die Funktion h an $N \geq n$ Stellen $x_1, \ldots, x_N$ und fordert nun, daß die Summe der Fehlerquadrate von Φ zu h an diesen Stellen minimal wird, d.h.

$$D^2 := \sum_{i=1}^{N} \left(h(x_i) - \Phi(x_i)\right)^2 \quad \rightarrow \quad \text{Minimum} \tag{E.3}$$

Das ergibt die Bedingungen:

$$\begin{aligned} & \frac{\partial D^2}{\partial c_j} && = 0 \quad j = 1, \ldots, n \\ \Leftrightarrow \quad & -2 \sum_{i=1}^{N} \left(h(x_i) - \Phi(x_i)\right) \frac{\partial \Phi(x_i)}{\partial c_j} && = 0 \quad j = 1, \ldots, n \end{aligned}$$

Mit Gleichung E.2 ergeben sich $\frac{\partial \Phi(x_i)}{\partial c_j} = \phi_j(x_i)$ und somit folgendes lineares Gleichungssystem für die Koeffizienten c_j:

$$\sum_{k=1}^{n} c_k \sum_{i=1}^{N} \phi_k(x_i)\phi_j(x_i) = \sum_{i=1}^{N} h(x_i)\phi_j(x_i) \quad j = 1, \ldots, n \tag{E.4}$$

Unter der Voraussetzung der linearen Unabhängigkeit des Funktionensystems $\phi_1, \ldots, \phi_n$ ist dieses Gleichungssystem eindeutig lösbar (für $N \geq n$). Für den Fall $N = n$ ist $\Phi(x)$ eine Interpolation der Stützpunkte $(x_i, h(x_i))$. Für $N < n$ ist die Lösung nicht eindeutig.

Anmerkung: In dieser Form des Verfahrens gehen alle Stützstellen mit der gleichen Wichtigkeit ein. Falls z.B. von einigen Werten bekannt ist, daß sie genauer sind als andere, kann man das mit Hilfe von Gewichtungsfaktoren berücksichtigen.

E.3 Das Verschachtelungsverfahren

Folgendes iterative Verfahren berechnet das Inverse einer gegebenen Verteilungsfunktion:
Gegeben sei der Wert $p \in [0,1]$, bei dem die Verteilung invertiert werden soll, d.h. gesucht ist $x = F^{-1}(p)$. Man startet mit einem „Kandidaten“ $x_K \geq 0$ und berechnet $F_K := F(x_K)$. Falls nun $F_K > p$ gilt, folgt wegen des monotonen Anstiegs von F, das x_K zu groß gewählt wurde. Man verkleinert also x_K um einen Wert Δx und berechnet F_K neu. (Falls umgekehrt $F_K < p$ galt, wird x_K vergrößert.) Man verändert nun x_K so oft um Δx, bis man „über das Ziel hinausschießt“, d.h. bis $F_K < p$ gilt. In diesem Fall dreht man die Suchrichtung um und verkleinert Δx, d.h. man setzt $\Delta x := -\Delta x / M$, wobei $M > 1$ ist. Da nur Verteilungen verwendet werden, die für positive Werte definiert sind, muß auch bei Unterschreiten der 0 die Suchrichtung umgekehrt, die Schrittweite verkleinert und $x_K := 0$ gesetzt werden. Das Verfahren wird so lange wiederholt, bis F_K genügend nahe an p liegt, d.h. bis $|F(x_K) - p| < G$, gilt, wobei G eine vorgegebene Genauigkeit ist.
Zusammenfassend sieht der verwendete „Verschachtelungsalgorithmus“ so aus:
(Wir verwenden $G = 10^{-5}$)

```
Gegeben: Wahrscheinlichkeit p ∈ [0,1]
Wähle  Kandidaten x_K ≥ 0
       Schrittweite Δx_K > 0
       Schrittfaktor M > 1
       Genauigkeit G ≪ 1
Falls F(x_K) > p
       Δx := −Δx
Solange |F(x_K) − p| > G tue
       x_K := x_K + Δx
       Falls Δx · (F(x_K) − p) > 0 oder x_K < 0
              Δx := −Δx/M
              Falls x_K < 0
                     Setze x_k := 0
Gebe x_K aus
```

E.4 Die Umgekehrte Integration

Die „umgekehrte Integration“ dient dazu eine Verteilungsfunktion zu invertieren, die nur über die Dichtefunktion f gegeben ist. Es wird von einem Startwert a (hier $a = 0$, da nur Zeitverteilungen gebraucht werden) mit der Schrittweite $\Delta > 0$ so lange aufsummiert, bis der Wert p erreicht wird:

```
Gegeben: Wahrscheinlichkeit p ∈ [0,1]
Wähle  Startwert a
       Schrittweite Δ > 0
Setze  Summe S := 0
       Zähler i := 0
Solange S < p tue
       S := S + Δ · f(a + (i + 0.5)Δ)
       i := i + 1
Gebe aus a + (i − 0.5)Δ
```

E.5 Zufallszahlengeneratoren

Verfahren zur Erzeugung von Zufallszahlen sind in [sobol 74] beschrieben.
Der Ausgangspunkt aller Methoden ist die Auslosung von gleichwahrscheinlichen ganzen Zahlen aus einem Intervall $[0, m-1]$. Dazu kann man große Tabellen von Zufallszahlen, die irgendeinmal „von Hand“ ausgelost wurden, verwenden. Einen „echten“ Zufallszahlengenerator kann man konstruieren, indem man das Rauschen von Elektronikröhren ausnutzt. Der Nachteil dieser Methode ist aber, daß man die sich zeitlich ändernde Charakteristik einer Röhre nicht im Griff hat und somit die Güte, d.h. ob alle Werte gleichwahrscheinlich sind und genügend streuen, nicht kontrollieren kann. Außerdem sind die so ermittelten Werte nicht reproduzierbar. Deshalb haben sich heutzutage die sog. *Pseudozufallszahlen* durchgesetzt, die zwar deterministisch gewonnen werden, deren statistisches Verhalten aber genügend stochastisch ist.
Weit verbreitet sind die sog. *kongruenten* Pseudozufallszahlengeneratoren Bei ihnen wird aus einer gegebenen (alten) Zufallszahl x_n die nächste Zahl x_{n+1} so ermittelt:

$$x_{n+1} = (ax_n + b) \bmod m \tag{E.5}$$

Den Zufallsgenerator kann man durch Wahl eines Startwertes x_0 einstellen. Bei unserem Programm wird x_0 aus der aktuellen Zeit gewonnen, so daß die Wiederholung eines Experiments im allgemeinen ein neues Ergebnis liefert. Der Nachteil von Pseudozufallszahlen ist, daß sie sich nach einer gewissen Periode wiederholen. Für den Fall, daß M eine Potenz von 2 ist, wurde in [hull62] gezeigt, daß die Zykluslänge genau dann maximal wird, wenn b ungerade und $a := 4c + 1$ für irgendeine natürliche Zahl c gewählt werden.
Zur Zeit wird noch ein kongruenter Zufallszahlengenerator verwendet, der die maximale Periode $m = 2^{31}$ hat und bei dem C–Compiler der Firma Sun mitgeliefert wird. [1]

[1] Die Parameter a und b sind uns nicht bekannt.

E.6 Die Inversionsmethode

Die Inversionsmethode basiert auf folgenden Überlegungen [morgan84]: Falls u $(0,1)$–gleichverteilt ist, so ist die Wahrscheinlichkeit, daß u kleiner ist als ein Wert $p \in [0,1]$ gerade p selbst (denn $\mathrm{P}(u \leq p) = \int_0^p 1\,dx$):

$$\mathrm{P}(u \leq p) = p \tag{E.6}$$

Das gilt insbesondere, falls man $p = F(t)$ für eine Verteilungsfunktion F (t beliebig) wählt. $F(t)$ ist die Wahrscheinlichkeit, daß eine gemäß F ausgeloste Zufallszahl k einen Wert kleiner oder gleich t annimmt: Damit ergibt sich:

$$\begin{aligned} & \mathrm{P}(u \leq F(t)) && = \mathrm{P}(k \leq t) \\ \Leftrightarrow \quad & \mathrm{P}(F^{-1}(u) \leq t) && = \mathrm{P}(k \leq t) \end{aligned} \tag{E.7}$$

Formel (E.7) bedeutet, daß man k wie folgt wählen muß:

$$k = F^{-1}(u) \tag{E.8}$$

E.7 Die Verwerfungsmethode

Es folgt eine Schilderung der Verwerfungsmethode [morgan84]. Sie dient der Erzeugung von Zufallszahlen, die einer Dichtefunktion f gehorchen.

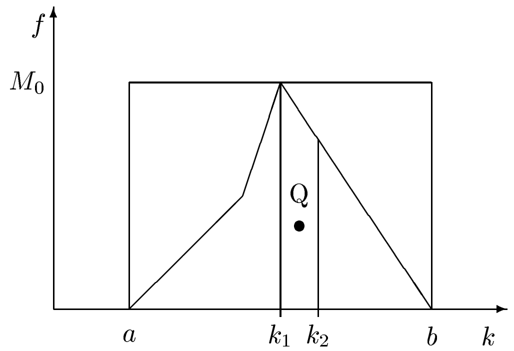

Abbildung E.2: Die Verwerfungsmethode

Es wird zuerst eine Verteilung betrachtet, deren Dichtefunktion nur in einem Intervall $[a,b]$ nicht verschwindet und eine obere Schranke M_0 hat. Es sei Q ein Punkt, der zufällig im Rechteck $[0, M_0] \times [a,b]$ plaziert wird, wobei jeder Punkt gleich wahrscheinlich sei (siehe Abbildung E.2).

Die Wahrscheinlichkeit, daß der Punkt Q sich unterhalb der Kurve der Dichtefunktion befindet, ist das Verhältnis der Fläche unter der Kurve zur Fläche des Quadrats. Das ergibt wegen der Normierung der Dichtefunktion:

$$\mathrm{P}(\text{Q unter Kurve}) = \int\limits_a^b f(k)\,dk/M_0(b-a) = 1/M_0(b-a) \tag{E.9}$$

Die Wahrscheinlichkeit, daß Q im Bereich zwischen k_1 und k_2 unter der Kurve liegt, ist dann:

$$\mathrm{P}(\text{Q im Bereich } [k_1, k_2] \text{ unter Kurve}) = \int\limits_{k_1}^{k_2} f(k)\,dk/M_0(b-a) \tag{E.10}$$

Die Wahrscheinlichkeit, daß Q, falls er unter der Kurve liegt, auch im Bereich zwischen k_1 und k_2 liegt, ist somit:

$$\begin{aligned}
\mathrm{P}(\text{Q in } [k_1, k_2] \quad | \text{ Q unter Kurve}) &= \frac{\mathrm{P}(\text{Q in } [k_1, k_2] \text{ unter Kurve})}{\mathrm{P}(\text{Q unter Kurve})} \\
&= \int\limits_{k_1}^{k_2} f(k)\,dk \\
&= \mathrm{P}(k_1 < k \leq k_2) \qquad \text{(E.11)}
\end{aligned}$$

Man kann deshalb so vorgehen: Man lost einen Punkt im Quadrat $[0, M_0] \times [a, b]$ aus, indem man eine (a, b)–gleichverteilte Größe k und eine $(0, M_0)$–gleichverteilte Zufallszahl y auslost. Dies erfolgt nach der Inversionsmethode, indem zwei $(0, 1)$–gleichverteilte Werte u_1 und u_2 ausgelost und $k := a + (b - a)u_1$ sowie $y := M_0 u_2$ gesetzt werden. Falls nun der Punkt Q$= (k, y)$ unter der Kurve der Dichtefunktion f liegt, d.h. falls $y \leq f(k)$ gilt, wird der Wert k als Ergebnis der Auslosung genommen. Im anderen Fall wird k verworfen und ein neuer Punkt Q ausgelost und das Verfahren so lange wiederholt, bis ein Wert k akzeptiert wird. Die so ausgelosten Zahlen k sind dann gemäß der Verteilung verteilt, welche die Dichtefunktion f hat.

Diese Variante des Verwerfungsverfahrens hat zwei Nachteile: Der Anteil der verworfenen Werte ist für „typische“ Dichtefunktionen recht hoch, und es können nur Dichtefunktionen verwendet werden, die nur über einem endlichen Intervall ungleich 0 sind. Diese Nachteile vermeidet folgende Variante:

Man verwendet als „einhüllende“ Funktion keine Rechteckfunktion, sondern eine um den Faktor $c > 1$ vergrößerte Dichtefunktion $h(k)$, für die ein schnelles Ausloseverfahren existiert. Die Konstante c muß so gewählt werden, daß $c \cdot h(k) \geq f(k) \;\; \forall k$ gilt. Man geht dann so vor: Es wird eine Zahl k gemäß der Dichtefunktion h ausgelost. Weiterhin wird eine $(0, 1)$–gleichverteilte Zahl u ermittelt. Der Kandidat k wird akzeptiert, falls $u \cdot c \cdot h(k) \leq f(k)$ erfüllt ist (siehe

Abb. E.3), anderfalls muß eine neue Zahl k ausgelost werden. Die akzeptierten Zufallszahlen k sind dann gemäß der Dichtefunktion f verteilt.

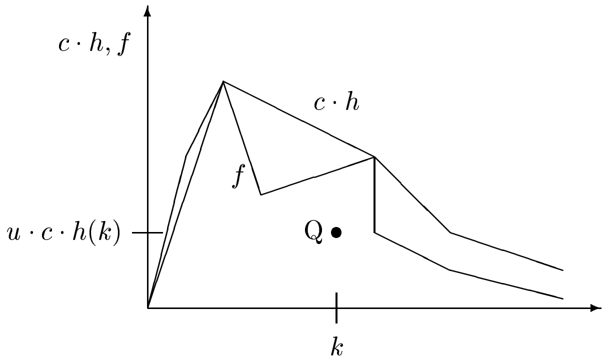

Abbildung E.3: allgemeines Verwerfungsverfahren

Bei diesem Verfahren ist der Anteil unbrauchbarer Werte k um so geringer, je besser sich h an f "anschmiegt", d.h. je kleiner der Faktor c gewählt werden kann.

E.8 Definition der Gammafunktion

Die Gammafunktion ist wie folgt definiert:

$$\Gamma(x) := \int_0^\infty t^{x-1} \exp(-t)\, dt \quad \text{für } x > 0 \tag{E.12}$$

Speziell gilt: $\Gamma(n+1) = n!$ für jede natürliche Zahl n einschließlich der 0.

E.9 Parameter der Normalverteilungsnäherung

Für die Approximation der Normalverteilung wird folgendes Polynom verwendet:

$$P(x) = p_1 t + p_2 t^2 + p_3 t^3 + p_4 t^4 + p_5 t^5 \tag{E.13}$$

Die einzelnen Parameter wurden so gewählt (f_1, q_1: siehe Kapitel 8.5):

$$f_1 = 0.3989422803$$

$$
\begin{aligned}
q_1 &= 0.2316419 \\
p_1 &= 0.31938153 \\
p_2 &= -0.356563782 \\
p_3 &= 1.781477937 \\
p_4 &= -1.821255978 \\
p_5 &= 1.330274429
\end{aligned}
$$

E.10 Parameter der rationalen Funktion

Die rationale Funktion, welche bei der Approximation der inversen Normalverteilung verwendet wird, hat folgende Form:

$$Q(y) = y - \frac{p_0 + p_1 y + p_2 y^2 + p_3 y^3 + p_4 y^4}{q_0 + q_1 y + q_2 y^2 + q_3 y^3 + q_4 y^4} \tag{E.14}$$

Die Parameter wurden in [odeh74] wie folgt angegeben:

$$
\begin{array}{ll}
p_0 = 0.322232431088 & q_0 = 0.099348462606 \\
p_1 = 1 & q_1 = 0.588581570495 \\
p_2 = 0.342242088547 & q_2 = 0.531103462366 \\
p_3 = 0.0204231210245 & q_3 = 0.10353775285 \\
p_4 = 0.0000453642210148 & q_4 = 0.0038560700634
\end{array}
$$

E.11 Die Box–Muller–Methode

Seien u_1, u_2 zwei $(0,1)$–gleichverteilte Zahlen und es seien k_1, k_2 wie folgt definiert:

$$
\begin{aligned}
k_1 &= \sqrt{-2\log(u_2)}\cos(2\pi u_1) \\
k_2 &= \sqrt{-2\log(u_2)}\sin(2\pi u_1)
\end{aligned} \tag{E.15}
$$

Folgender Beweis, der in [bratle83] skizziert ist, zeigt, daß k_1 und k_2 normalverteilt sind:
Die Dichtefunktion zweier unabhängiger Standardnormalverteilungen ist einfach das Produkt der einzelnen Dichtefunktionen:

$$f(k_1, k_2) = \frac{1}{2\pi}\exp(-[k_1^2 + k_2^2]/2) \tag{E.16}$$

Das Paar (k_1, k_2) kann mittels Polarkoordinaten (r, Θ) ausgedrückt werden:

$$
\begin{aligned}
k_1 &= r \cdot \cos\Theta \\
k_2 &= r \cdot \sin\Theta
\end{aligned} \tag{E.17}
$$

Daraus folgt:

$$k_1^2 + k_2^2 = r^2 \tag{E.18}$$

Diese Transformation wird nun bei der Berechnung der Dichtefunktion in Polarkoordinaten verwendet. Man muß aber berücksichtigen, daß die Dichtefunktion im Integral für die Verteilungsfunktion verwendet wird, d.h. bei der Transformation muß die zugehörige Funktionaldeterminante berechnet werden:

$$\begin{aligned}
\left|\frac{\partial(k_1,k_2)}{\partial(r,\Theta)}\right| &= \begin{vmatrix} \partial k_1/\partial r & \partial k_1/\partial\Theta \\ \partial k_2/\partial r & \partial k_2/\partial\Theta \end{vmatrix} \\
&= \begin{vmatrix} \cos\Theta & -r\cdot\sin\Theta \\ \sin\Theta & r\cdot\cos\Theta \end{vmatrix} \\
&= r\cdot\cos^2\Theta + r\cdot\sin^2\Theta \\
&= r
\end{aligned} \tag{E.19}$$

Damit ergibt sich die Dichtefunktion in Polarkoordinaten:

$$f_p(r,\Theta) = \frac{r}{2\pi}\exp(-r^2/2) \tag{E.20}$$

Da in der Dichtefunktion Θ nicht auftaucht, folgt, daß Θ über $[0, 2\pi]$ gleichverteilt ist. Man lost somit eine $(0,1)$–gleichverteilte Zahl u_1 aus und setzt:

$$\Theta = 2\pi u_1 \tag{E.21}$$

Die Verteilungsfunktion für den Parameter r ergibt sich durch Integration von f_p über den Kreis mit Radius r:

$$\begin{aligned}
F_p(r) &= \int_0^{2\pi}\int_0^r \frac{\tilde{r}}{2\pi}\exp(-\tilde{r}^2/2)\,d\tilde{r}\,d\tilde{\Theta} \\
&= \int_0^r \tilde{r}\exp(-\tilde{r}^2/2)\,d\tilde{r} \\
&= 1-\exp(-r^2/2)
\end{aligned} \tag{E.22}$$

Diese Gleichung läßt sich leicht invertieren. Man erhält somit durch folgende Transformation Werte, die gemäß F_p verteilt sind:

$$r = \sqrt{-2\log(1-u_2)} \tag{E.23}$$

Da mit u_2 auch $1-u_2$ über $[0,1]$ gleichverteilt ist, verwendet man:

$$r = \sqrt{-2\log u_2} \tag{E.24}$$

Wenn man nun die Gleichungen E.21 und E.24 in die Formeln E.17 einsetzt, erhält man die Transformationen in (E.15).

E.12 Gammaverteilte Zufallszahlen

Im folgenden wird gezeigt, wie man gammaverteilte Zufallszahlen auslosen kann. Zunächst wird der Fall betrachtet, daß β eine natürliche Zahl ist.
Dazu wird zuerst gezeigt[2], daß, wenn man die β–fache Summe von exponentialverteilten Größen mit der Rate α bildet, die so entstandene ZG die Dichtefunktion der Gammafunktion hat (= Aussage $(*)$), d.h. es wird.

$$z = \sum_{i=1}^{\beta} k_i, \tag{E.25}$$

betrachtet, wobei die k_i gemäß der Dichtefunktion $\hat{f}(t) = \alpha \exp(-\alpha t)$ verteilt sind. Der Beweis wird per vollständiger Induktion geführt :
$\beta = 1$: Die Dichtefunktion der Gammaverteilung lautet in diesem Fall:

$$f(t) = \alpha(\alpha t)^{1-1} \exp(\alpha t)/(1-1)! = \hat{f}(t) \tag{E.26}$$

$\beta \to \beta + 1$: Annahme sei, daß die Aussage $(*)$ für den Wert β erfüllt ist. Nun wird die Summe

$$z' = \sum_{i=1}^{\beta+1} k_i = \sum_{i=1}^{\beta} k_i + k_{\beta+1} \tag{E.27}$$

betrachtet. Gemäß Induktionsannahme hat der erste Summand die Dichtefunktion $f_{\alpha,\beta}$ und der zweite die Dichtefunktion $\hat{f}$. Für die Berechnung der kombinierten Dichtefunktion muß man berücksichtigen, daß sich ein Summenwert s auf verschiedene Weisen aus zwei Summanden darstellen läßt: $s = x + (s - x)$, wobei $x \in [0, s]$ gilt. Es ergibt sich die kombinierte Dichtefunktion f (mittels des sog. *Faltungsintegrals*) als:

$$\begin{aligned} f(s) &= \int_0^s f_{\alpha,\beta}(x)\hat{f}(x-s)\,dx \\ &= \int_0^s \alpha(\alpha x)^{\beta-1} \exp(-\alpha x)/(\beta-1)! \cdot \alpha \exp(-\alpha[x-s])\,dx \\ &= \alpha(\alpha)^{\beta} \exp(-\alpha s) \int_0^s \frac{x^{\beta-1}}{(\beta-1)!}\,dx \\ &= \alpha(\alpha s)^{\beta} \exp(-\alpha s)/\beta! \\ &= f_{\alpha,\beta+1}(s) \end{aligned} \tag{E.28}$$

[2]Man kann den Beweis auch kürzer mit Hilfe des sog. *Faltungssatzes* führen [schnee92], wofür aber der Begriff der *Laplace–Transformation* benötigt wird.

Damit ist die Aussage $(*)$ bewiesen.
Die exponentialverteilten Zahlen werden mittels Inversion ausgelost. Also erhält man aus β Zufallszahlen $u_1, \ldots, u_\beta$, die $(0,1)$–gleichverteilt sind, mittels folgender Umformung eine gammaverteilte Zahl w:

$$w = -\sum_{i=1}^{\beta} \log(1-u_i)/\alpha \tag{E.29}$$

Da mit u_i auch $1-u_i$ gleichverteilt sind, kann in Formel E.29 auch $\log(u_i)$ verwendet werden.
Der Algorithmus für Gammaverteilungen mit beliebigen reellen Parameter β, ist in [berman70] basierend auf [jöhnk64] angegeben:

> `Setze` $m := \lfloor \beta \rfloor$
> $\quad\quad q := \beta - m$
> `Lose` k `Zahle` u_i `gemäß` $U(0,1)$ `aus`
> `Setze` $z := -\sum_{i=1}^{k} \log(u_i)$
> `Wiederhole bis` $x + y \leq 1$ `gilt:`
> $\quad$ `Erzeuge` $(0,1)$`--gleichverteilte Zahlen` u, v
> $\quad$ `Setze` $x := u^{1/q}$
> $\quad\quad\quad y := v^{1/(1-q)}$
> `Setze` $w := x/(x+y)$
> `Erzeuge eine` $(0,1)$`--gleichverteilte Zahl` s
> `Setze` $t := -\log(s)$
> `Gebe` $(z + w \cdot t)/\alpha$ `aus`

Anhang F

Aufbau des Programms

Das Simulationsprogramm `simsam` wurde in der Programmiersprache C auf einer Workstation Sun 3/50 mit dem von Sun mitgeliefertem C Compiler realisiert. Um bei der Entwicklung Compilezeit zu sparen und die Übersichtlichkeit und damit die Wartbarkeit des Programms zu verbessern, habe wir es in einzelne Teile — genannt *Module* — aufgeteilt. Im folgenden wird geschildert, aus welchen Modulen das Programm besteht und welche Aufgaben sie erfüllen.
In Abbildung F.1 ist der strukturelle Zusammenhang der Module dargestellt. Sie werden durch Kästchen repräsentiert. Die Pfeile bedeuten, daß ein in der Hierarchie untenstehendes Modul Funktionen an ein höher stehendes exportiert. Der links unten stehende Programmteil `glob_var` exportiert globale Variablen an alle anderen Module.
Die in Abbildung F.1 dargestellten Programmteile entsprechen je einem File, das den gleichen Namen wie das jeweilige Modul und die Endung `.c` trägt. Im folgenden werden diese Programmteile beschrieben.

F.1 Modul glob_var

In `glob_var` sind alle globalen Variablen enthalten, die zur Durchführung der eigentlichen Simulation dienen, also keine Variablen zur Realisierung der Benutzeroberfläche. Das sind:

- Systemdaten für Komponenten, Reparaturstationen und Fehlertypen,
- Importance–Sampling–Parameter,
- Zeiger auf Ereignisliste und Wartelisten für Reparaturstationen,
- Daten, die die Zustände der Systembestandteile während der Simulation beschreiben,
- Definitionen der einzelnen Kenngrößen,

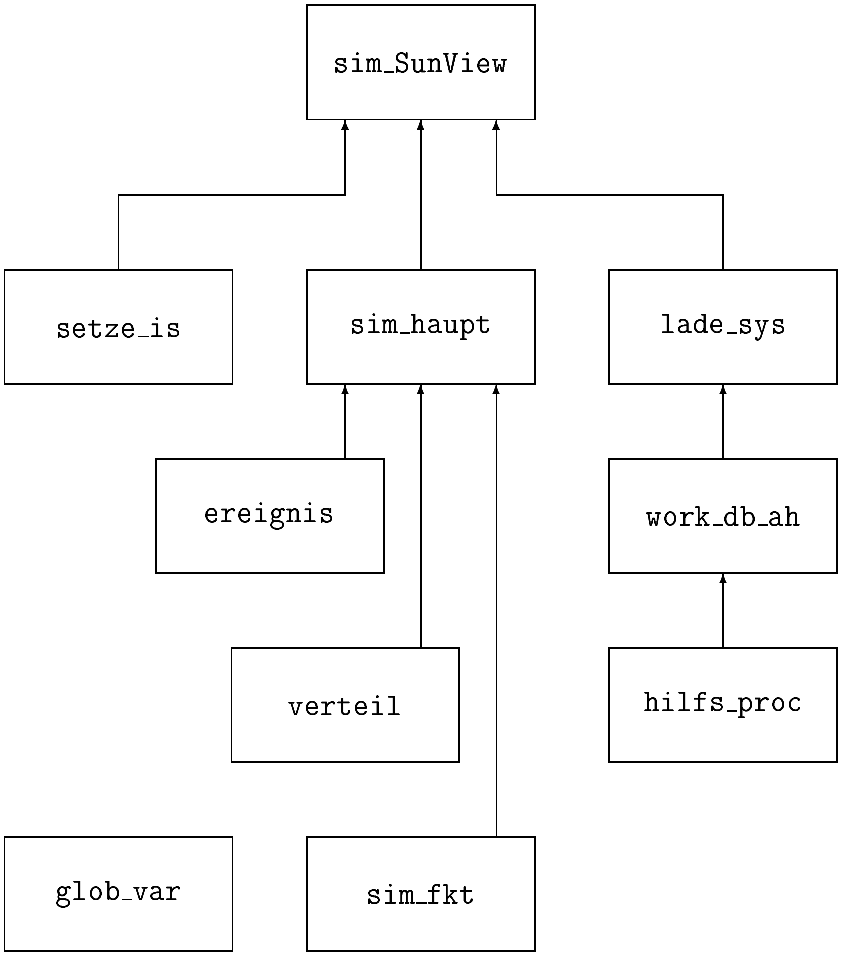

Abbildung F.1: Struktur des Programms

- die Ergebnisse von Simulationsläufen.

F.2 Modul sim_sunview

Dieser Programmteil realisiert die Benutzeroberfläche unter dem Fenstersystem *SunView*: Enthalten sind die dafür benötigten Funktionen und globalen Variablen. Falls das Programm an eine andere Oberfläche angepaßt werden soll, so muß nur dieses Modul geändert werden.
Es wird nun zuerst der allgemeine Aufbau von Benutzeroberflächen geschildert, die mit SunView realisiert werden (siehe auch [sunman86]).
Ein Programm unter SunView läuft in einem Fenster ab. Ein Fenster besteht aus einem *Rahmen*, in dem mehrere *Unterfenster* enthalten sind. Es gibt verschiedene Arten von Unterfenstern:

- *Textfenster* erlauben die Darstellung und Veränderung von Texten. Diese Fensterart findet in unserem Programm keine Verwendung.

- Ein sog. *Canvas* dient zur Ausgabe von Graphiken — ggf. mit Texten versehen — in einem Fenster. Im Programm wird ein Canvas zur Darstellung der Ergebnisse einer Simulation in Form von Vertrauensintervall, Hypothesenbeurteilung oder Gütefunktion verwendet.

- Die wichtigste Unterfensterart ist das *Panel*. In einem Panel können verschiedene Arten von Objekten dargestellt werden, die es dem Benutzer ermöglichen, mit dem Programm zu kommunizieren. Die wichtigsten Objekte sind:

  - *Messages* dienen zur Ausgabe von Strings.
  - *Buttons* sind beschriftete Felder. Wenn der Benutzer sie mit der Maus anklickt, werden bestimmte Aktionen ausgeführt. Wird z.B. in dem Programm der `Run`–Button angeklickt, so startet die Simulation.
  - In *Text*objekte kann der Benutzer über die Tastatur Eingaben machen, z.B. den Namen eines zu simulierenden Systems oder die Anzahl der Durchgänge eingeben.
  - Mittels sog. *Choices* kann man eine von mehreren Möglichkeiten auswählen, z.B. gemäß welcher Verteilungsfunktion ein Ausfall ausgelost werden soll.
  - *Toggles* ermöglichen es, verschiedene Optionen einzeln an und aus zu stellen. Beispiel: Auswahl, welche Kenngrößen im nächsten Simulationslauf berechnet werden sollen und welche nicht.

- Ein TTY–Unterfenster schließlich simuliert ein reines ASCII–Terminal. Es dient dazu, Programme laufen zu lassen, die nicht speziell für das Fenstersystem geschrieben sind; es findet bei uns keine Verwendung.

Alle Unterfenster können entweder vollständig auf dem Bildschirm dargestellt werden oder auch nur ausschnittsweise. Im zweiten Fall kann man an den Fensterrändern sog. *Scrollbars* anbringen, mit denen man den angezeigten Ausschnitt einstellen kann.
Bei Programmen, die unter SunView laufen, liegt die Kontrolle des Programms beim Betriebssystem: Zuerst erzeugt das Programm alle graphischen Objekte. Dann wird ein Unterprogramm des Betriebssystems aufgerufen, welches erst bei Abbruch des Programms beendet wird. Wenn nun der Benutzer eine Aktion mit einem Objekt des Fensters ausführt, wird diese Aktion vom sog. *Notifier* registriert, der wieder zeitweise die Kontrolle dem Benutzerprogramm übergibt: Es ruft eine mit dem Objekt assoziierte Funktion, eine *Notify–Prozedur* auf, die

dann die gewünschte Aktion ausführt und danach ins Betriebssystem zurückkehrt.
Das Modul `sim_sunview` besteht nun aus den Funktionen, die die graphischen Objekte der Benutzeroberfläche erzeugen, den zugehörigen globalen Variablen und den Notify–Prozeduren. Letztere rufen dann die Funktionen auf, die die eigentlichen Aktionen durchführen, wie Laden und Speichern von Daten sowie die Ausführung der Simulation. Außerdem sind noch die Funktionen zur Darstellung der Ergebnisse in diesem Modul enthalten.

F.3 Modul setze_is

Dieses Programmteil verwaltet die Systemdaten für das Importance Sampling (IS). Es gibt Funktionen, die die dafür benötigten dynamischen Datenstrukturen erzeugen bzw. löschen, die Systemdaten in die IS–Daten übertragen, sowie Funktionen, die diese Daten in Files abspeichern und wieder einladen können.

F.4 Modul lade_sys

Dieses Modul stellt Funktionen zur Verfügung, mit denen ein System, was zuvor mit dem Programmen `mdledit` und `rbdedit` eingegeben wurde, eingeladen werden kann; d.h. es werden die Daten der Komponenten, der Reparaturstationen, der Fehlertypen und die Systemfunktion eingeladen und zum Teil dynamische Datenstrukturen erzeugt, in denen sie abgelegt werden. Die eingeladenen Systeme werden auf Konsistenz und Vollständigkeit überprüft. Weiterhin werden Verweise, die in Form von Komponentennamen vorliegen, in Adressen umgewandelt. Damit ist ein schnellerer Ablauf der Simulation möglich. Zu guter Letzt gibt es eine Funktion, die die Datenstrukturen eines Systems wieder löscht, damit ein neues System eingeladen werden kann.
Um diese Aufgaben zu erledigen greift, `lade_sys` auf das Modul `work_db_ah` zu.

F.5 Untermodul work_db_ah

Dieses Modul ist die Anpassung eines Programmteils, welches aus [ruenge92] übernommen wurde. Es enthält Funktionen, die auf niedriger Ebene für das Laden der Daten eines zu simulierenden Systems zuständig sind. Dafür verwendet es die Funktionen eines vom Betriebssystem zur Verfügung gestellten Datenbanksystems mit dem Namen `ndbm`, das auf der Basis von Hashtabellen arbeitet. Es ermöglicht direkten Zugriff auf die Daten mittels eines Schlüssels.

F.6 Hilfsmodul hilfs_proc

In `hilfs_proc` werden dem Modul `work_db_ah` Funktionen zur Verfügung gestellt, mit denen dieses Modul, welches ja ursprünglich für `mdledit` entwickelt wurde, in `simsam` lauffähig ist.

F.7 Modul sim_haupt

Dieses Programmteil ist für die eigentliche Durchführung der Simulation zuständig. Neben dem eigentlichen Hauptalgorithmus existieren Funktionen zur Initialisierung der Daten, zur Berechnung des IS–Faktors, zur Durchführung des Rescheduling, zur Behandlung der Reparaturstationen sowie zur Sammlung der Daten während eines Simulationslaufs. Außerdem sind die Status– und Wertefunktionen enthalten, die zur Realisierung der verschiedenen Kenngrößen dienen.
Die im folgenden dargstellten Programmteile unterstützen das Modul `sim_haupt`.

F.8 Untermodul ereignis

Hier finden sich Funktionen zur Verwaltung der Ereignisliste und der Wartelisten der Reparaturstationen: Ereignisse können erzeugt, gelöscht, und nach bestimmten Sortierkriterien in Listen ein– bzw. wieder ausgekettet werden.

F.9 Untermodul verteil

Das Modul `verteil` ist für die mathematischen Aufgaben im Zusammenhang mit einem Simulationslauf verantwortlich. Hier stehen Funktionen zur Verfügung, mit denen Dichtefunktion, Verteilungsfunktion und ihr Inverses für die folgenden Wahrscheinlichkeitsverteilungen realisiert werden:

- Exponentialverteilung
- Weibullverteilung
- Normalverteilung
- Gammaverteilung
- Gleichverteilung
- deterministische Verteilungen

Will man weitere Verteilungen in dem Programm realisieren, so muß man für sie nur eine Konstante in `glob_const.h` definieren (s.u.) und in den Funktionen für die Dichte–, Verteilungs–, und inversen Verteilungsfunktionen sowie für die Zufallszahlenauslosung entsprechende Programmabschnitte einfügen.
Weiterhin können ganze gleichverteilte Zahlen aus einem Intervall $[0, m-1]$, gleichverteilte Zufallszahlen aus dem Bereich $[0, 1]$, sowie mittels Transformation Zufallszahlen gemäß den oben angegebenen Verteilungen erzeugt werden. Mittels einer Initialisierungsfunktion kann der Zufallszahlengenerator auf einen Anfangswert gestellt werden, der von der aktuellen Tageszeit in Sekunden abhängt. Die Funktionen dieses Moduls werden durch Mittelwert– und Varianzberechnungen komplettiert.

F.10 Untermodul sim_fkt

In `sim_fkt` kann der Gesamtzustand eines Systems aus dem aktuellen Zustandsvektor der Komponenten berechnet werden. Die Systemfunktion ist codiert in einem Array gespeichert.

F.11 Headerfiles

Weiterhin gibt es noch sogenannte *Headerfiles*, die exportierte Datentypen, Konstanten, Deklarationen globaler Variablen und die Funktionsköpfe von Funktionen, die nicht einen `int`–Wert zurückliefern, enthalten. Es gibt folgende Headerfiles:

- `glob_const.h` enthält die Definitionen (mit `#define`) aller globalen Konstanten. Achtung: Die ersten Konstanten dieses Files legen die Maximalgrößen der Modellsysteme fest.
- `glob_var.h` enthält die Deklarationen der in `glob_var.c` definierten globalen Variablen, sowie einige Datentypen.
- `kompdb_ah.h` enthält die Definitionen von Datentypen, die zur Speicherung eines zu simulierenden Systems dienen. Dieses File ist die Anpassung eines Files, das aus [ruenge92] stammt.
- `verteil.h` enthält Funktionsköpfe aus `verteil.c`.
- `ereignis.h` enthält Funktionsköpfe aus `ereignis.c`.
- `sim_haupt.h` enthält Funktionsköpfe aus `sim_haupt.c`.

Anhang G

Ein Beispielsystem

In diesem Teil wird beispielhaft geschildert, wie ein Modellsystem eingegeben und eine Simulation durchgeführt wird. Das wird an Hand des größeren Systems mit 32 Komponenten demonstriert, welches in Kapitel 10.3 vorgestellt wurde.
Zuerst muß das Zuverlässigkeitsblockdiagramm des Systems mit dem Programm `rbdedit` [löhr90a] eingegeben werden. Ein Auschnitt des Blockdiagramms ist in Abbildung G.1 zu sehen, welche die Benutzeroberfläche von `rbdedit` zeigt.
Die Zuverlässigkeitsparameter des Modellsystems werden mit dem Programm `mdledit` [ruenge92] eingegeben. Es folgen einige wichtige Fenster dieses Programms.
In Abbildung G.2 ist gezeigt, wie ein Modifizierungsgrad eingegeben wird. Es ist ein Grad der Stufe 1, welcher die gleichen Werte hat wie der 0. Grad der Komponente. Diese Verdoppelung wird für die Realisierung des dynamischen Importance Sampling vorgenommen (siehe Kapitel 5).
Die Definition des einzigen Fehlertyps, der in dem Modell auftritt, ist in Abb. G.3 dargestellt.

Abbildung G.1: Eingabe des Blockdiagramms

Abbildung G.2: Eingabe eines Modifizierungsgrades

Abbildung G.3: Eingabe des Fehlertyps

Die Reparaturstation wird in dem in Abb. G.4 gezeigten Fenster eingegeben.

Abbildung G.4: Eingabe der Reparaturstation

Die Simulation erfolgt mit dem Programm `simsam`, dessen Benutzeroberfäche in Abbildung G.5 dargestellt ist. Dort ist der Zustand des Programms nach Ablauf einer Simulation zu sehen. Es wurde die stationäre Unverfügbarkeit als Quotient mittels Regenerativer Simulation ermittelt. Importance Sampling war inaktiv. Im untersten Teilfenster ist das sich ergebende 90%–Vertrauensintervall dargestellt.
In Abbildung G.6 ist schließlich zu sehen, wie für dynamisches Importance Sampling die Ausfallrate eines Grades 1 einer Komponente um den Faktor 20 erhöht wurde. Die Ergebnisse der SImulation sind in Abschnitt 10.3 aufgeführt.

Abbildung G.5: Benutzeroberfläche des Simulationsprogramms

Abbildung G.6: Eingabe eines IS–Parameters

Erklärung

Ich versichere, daß ich diese Diplomarbeit selbständig verfaßt und keine anderen Quellen und Hilfsmittel als die angegebenen benutzt habe. Die Stellen, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen sind, wurden in jedem einzelnen Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht. Das gleiche gilt für beigefügte Skizzen und Darstellungen. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Hagen, den