Skript zur Vorlesung

Simulation

(Die Vorlesung wurde im Sommersemester 1997
von Prof. Dr.-Ing. Claudio Moraga gehalten)

Notizen von Peter Fasching
(überarbeitet von C. Moraga)

In dieser Vorlesung geht es um die drei Begriffe:
System
Modell
Simulation

Gliederung:

0. Vorwort

0.1 Grundsätzliches zur Vorlesung

Die Vorlesung heißt offiziell „Simulation“. Der vollständige Titel ist: „**Modellierung und Simulation**“. Die Vorlesung umfaßt vier Vorlesungsstunden und zwei Übungsstunden.
Die Vorlesung ist eine

- Wahlpflichtveranstaltung für den Studiengang *Angewandte Informatik*
- Spezialvorlesung für den Studiengang *Informatik*

Die Vorlesung findet im Sommersemester statt und wird abwechselnd von Prof. Moraga und Prof. Beilner gehalten.

0.2 Grundsätzliches zum Skript

Beide Professoren (Moraga und Beilner) setzen recht unterschiedliche Schwerpunkte in der Vorlesung „Simulation“. Da dieses Skript sich an der Vorlesung von Prof. Moraga orientiert, ist es auch nur als Begleitmaterial für dessen Vorlesung geeignet. Für die Vorlesung von Prof. Beilner gibt es eine Folienkopie.
Das Skript solle eine von der Vorlesung unabhängige Vorbereitung auf die Prüfung ermöglichen. Dies Skript erhebt den Anspruch, zu etwa 80% den prüfungsrelevanten Stoff zu enthalten. Bei den restlichen 20% verweisen wir in den entsprechenden Kapiteln auf die Literatur. Das Ziel dieses Skriptes ist also:
Das Skript soll den Studenten eine effiziente Vorbereitung auf die Prüfung ermöglichen.
Zu diesem Zeitpunkt fehlt leider noch ein wichtiger Abschnitt in diesem Skript. Dieser Abschnitt enthält ein **Beispiel für die Modellierung eines Systems**. Allerdings sollte gerade die Modellierung eines gegebenen Beispiel-Systems ausführlich geübt werden! Die in der Bereichsbibliothek einzusehende **Folienkopie** zu Prof. Moragas Simulations-Vorlesung enthält jedoch ein Beispiel hierzu. (Ein Bauer muß sich entscheiden, wieviel Weizen und wieviel Raps er anbauen soll.) Die dazu nötige Theorie kann dem vorliegenden Skript entnommen werden.
In wie fern sich das Skript für die Bearbeitung der **Übungsaufgaben** eignet, hängt von den Übungsaufgaben ab und ist momentan noch nicht absehbar.
Mit

Rahmen und Schattierung

hervorgehobene Textstellen sollten auswendig gelernt werden. Mit einer Ausnahme: Im Kapitel „Erzeugung von Zufallszahlen“ wurden die Verfahren zur „Generierung von Zufallszahlen“ weder eingerahmt noch schattiert, obwohl man diese Verfahren ebenfalls herunterbeten können sollte.
Mein Tip: Kopiert alle Seiten, auf denen eingerahmte und schattierte Texte stehen. Anschließend könnt ihr die eingerahmten Texte ausschneiden und auf DIN A5- oder A6-Karteikarten kleben. Die Breite des Textes sollte sich gut dazu eignen.

0.3 Literatur

Unbedingt notwendig ist eigentlich nur ein Buch:
„**Computer simulation and modelling**“ (Kapitel 1 bis 7), Francis **Neelamkavil**, 1987,
Bereichsbibliothek Informatik, Signatur: 3583/Neel, oder in der Hauptbibliothek, Signatur: Sn 17019
Das Buch wird jedoch leider nicht mehr verkauft, was bei einem Preis von gut 70$ (inklusive aller Nebenkosten macht das etwa 130 bis 140 DM) eher unrelevant ist. Eine Kopie des Inhaltsverzeichnisses, der Kapitel 1 bis 7 sowie des Registers (insgesamt etwa 170 Seiten) ist allerdings sehr empfehlenswert. Die übrigen Kapitel einschließlich dem Kapitel 11 über die „kontinuierliche System-Simulation“ sind nicht prüfungsrelevant.
Außerdem empfehlenswert erscheinen mir:

- **Diskrete Simulation** (Kapitel 1, 2, 4.1, 4.4.1, 5),
  Bereichsbibliothek Informatik, Signatur: 3583/Jent
- **Digitale Simulation kontinuierlicher Systeme**,
  Bereichsbibliothek Informatik, Signatur: 3583/Page
- (Prof. Moraga hält sich im Kapitel über die „diskrete Simulation kontinuierlicher Systeme“ zwar eng an das vorige Buch, aber wer die Grundlagen auch wirklich verstehen will, empfehle ich:)
  Numerische Methoden (Kapitel 8),
  Hauptbibliothek, L Ma 43 +24
- Euer lieblings-Statistik-Buch, möglichst ein umfangreiches Nachschlagewerk.

0.4 Hauptthemen

In dieser Vorlesung geht es um die drei Begriffe:

- System
- Modell
- Simulation

Der Aufbau der Vorlesung orientiert sich an diesen drei zentralen Begriffen:

1) Einleitung
2) **System**e
   (Systemanalyse)
3) **Modell**ierung
   (Klassifikation von Modellen, Modellierungsmethode)
   (Verifikation, Validierung und Zertifikation (Bescheinigung))
4) **Simulation** 1.Teil: Grundlagen und Grundsätzliches
5) **Simulation** 2.Teil: Erzeugung von Zufallszahlen
   (inklusive dem Testen von Hypothesen/Zufallszahlen)
6) **Simulation** 3.Teil: Diskrete Simulation diskreter Systeme
7) **Simulation** 4.Teil: Diskrete Simulation kontinuierlicher Systeme
8) Anhang

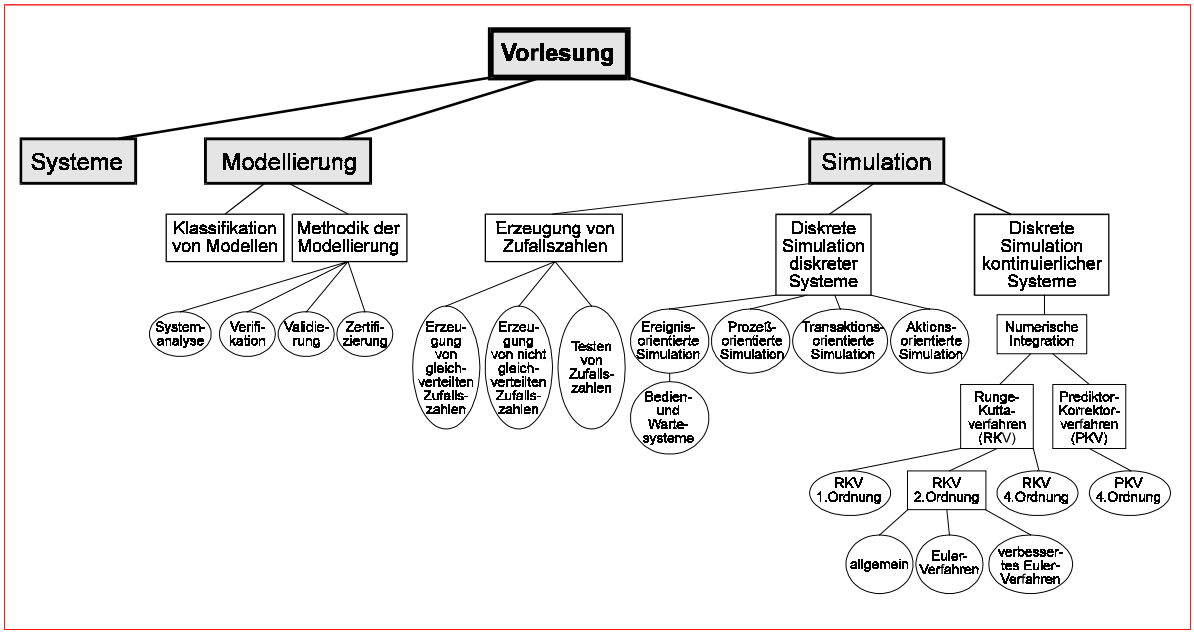


Abb.: Graphische Strukturierung der Inhalte dieser Vorlesung.

1 Einführung

1.1 Grundsätzliches zur Vorlesung

Diese Vorlesung soll "Modellierung und Simulation" heißen.
Prof. Moraga wird sie aus dieser Sicht halten.
Arbeiten in Bereich „Modellierung und Simulation“ haben eine lange Geschichte, verteilt über mehrere Wissenschaften. Das Studium von „Modellierung und Simulation“ hat sich langsam in der Statistik (etwa ab 1960) und in der Informatik etabliert.
(Bemerkung: An der Uni Dortmund werden auch simulationsbezogene Vorlesungen am FB Statistik angeboten).

1.2 Voraussetzungen

Um im Bereich „Modellierung und Simulation“ arbeiten zu können, sind einige Voraussetzungen notwendig. Die wichtigste Voraussetzungen sind:

- Gesunder Menschenverstand
- Fachliches Wissen
- Mathematisches Wissen (insbesondere Statistik)
- Programmiererfahrung
- Abstraktionsvermögen und systematisches Denken

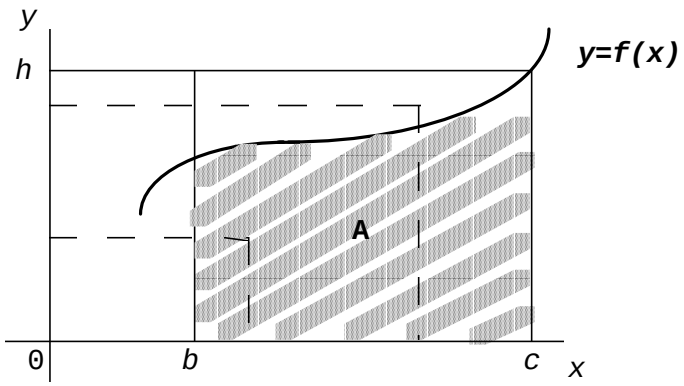
1.3 Verschiedene Arten von Lösungen

Ist ein ein Problem mit einer mathematischen Formulierung gegeben, kann es sein, daß das Problem

- eine (oder mehrere) optimale Lösung(en)
- eine (oder mehrere) gute (aber nicht optimale) Lösung(en)
- eine (oder mehrere) verträgliche Lösung(en)
- unverträgliche Lösungen
- überhaupt keine Lösung

hat.
Modellierung und Simulation kann zunächst als ein Fach verstanden werden, das sich mit der Methoden zum Finden von guten Lösungen zu praktischen Problemen beschäftigt.
Wie es leider zu erwarten war, hat die Modellierung und Simulation neue Entwicklungsimpulse während des 2. Weltkrieges erhalten. John von Neumann und Stanislaw Ulam benutzten mit Erfolg die sogenannte **Monte Carlo - Methode** bei der Lösung von Problemen in der Teilchenphysik.

1.4 Einführendes Beispiel: Integralberechnung



Aufgabenstellung
Zu berechnen sei die Fläche A.
Analytische Lösung
Das genaue, analytische Ergebnis ließe sich durch das Integral berechnen:

$$A = \int_b^c f(x)\,dx,$$

mit

$$0 \le f(x) \le h,$$

$$b \le x \le c$$

Lösung durch Simulation
Es sei **R** das Viereck mit der Basis (b,c) und der Höhe h. Ferner sei es Z(x,y) ein Zufallspunkt in **R**. Die Wahrscheinlichkeit *p*, das der Punkt Z in **A** liegt, ist:

$$p = \frac{A}{R}$$

$$= \frac{A}{(c-b)\cdot h}$$

Angenommen, daß N Zufallspunkte erzeugt worden sind, und K davon in **A** liegen. Dann gilt bei großem N:

$$p \approx \frac{K}{N}$$

und

$$\lim_{N\to\infty} p = \frac{A}{(c-b)\cdot h}$$

Daraus folgt:

$$A = (c-b)\cdot h \cdot \lim_{N\to\infty} p$$

$$\approx (c-b)\cdot h \cdot \frac{K}{N}$$

(Bei sehr großem N erhält man damit eine Ergebnis für A, das mit allen relevanten Nachkommaziffern mit dem exakten Ergebnis von A übereinstimmt.)
Rechnergestützte Modellierung und Simulation als eine Methode zur Problemlösung ist dann angebracht, wenn eine analytische, numerische, mathematische bzw. eine experimentelle physikalische Lösung

- zu schwer
- zu langsam (zu aufwendig)
- zu teuer
- zu gefährlich
- oder einfach unmöglich ist.

Offensichtlich können Simulationsergebnisse ungenauer als die analytischen sein.

1.5 Kontinuierliche und diskrete Systeme

Kontinuierliches System:
Z.B. Temperaturänderungen in einem Kraftwerk
Diskretes System:
Z.B. Bewegung von Kunden in einer Bank.

Definitionsbereich	Diskret		Kontinuierlich	
Wertebereich	Diskr. 1.	Kont. 2.	Diskr. 3.	Kont. 4.

1. Echte diskrete Systeme
2. Abtastsysteme (ggf. Annäherung zu K.-Systeme.)
3. Quantizierte Systeme (vgl. SPICE)
4. Echte kontinuierliche Systeme

1.6 Kontinuierliche und diskrete Variablen

1.6.1 Beispiel: Verkehrsampeln

Die "Programmierung" von Verkehrsampeln in einer komplexen Straßenkreuzung ist kein triviales Problem. (Man denke an die Indupark-Kreuzung zum Wertkauf/Metro oder an Zufahrtskreuzungen zum

Fußballstadium). Es gibt keine einfache analytische Methode, um herauszufinden, welche die beste Ampelprogrammierung ist, die Staus vermeidet und allen Verkehrsteilnehmer "gerecht" wird.

- Ampelsequenz: diskretes Ereignis
- Fahrzeugbewegung: kontinuierliches Ereignis
- Fahrzeugenankunft: stochastisch
- Schaltgangsequenz: (halb)deterministisch
- Stromversorgung: statisch, kontinuierlich
- Ampelsteuerung: dynamisch

1.7 Überblick über die Kapitel

1.7.1 Die diskreten Simulationsmodelle

Grundsätzlich:
diskrete Annäherung zur Realität
kontinuierliche Änderungen werden durch eine Reihe diskreter Ereignisse repräsentiert
Also:
Eine diskrete Simulation besteht aus der Beobachtung und Analyse der Ergebnisse, die durch das Erzeugen von (ggf. zufallsverteilten) Ereignissen mit Hilfe eines Simulationsmodells erreicht werden können.

1.7.2 Der Modellierungs-und Simulationsprozeß

Das "mathematische Modell" kann hier außer einer eigentlichen mathematischen Beschreibung, die folgenden Informationen umfassen:
logische Aussagen
falls <Bedingung> dann <1. Aktion> sonst <2. Aktion>

- statistische Daten aus (ggf. experimentellen) Messungen
- Daten aus Wahrscheinlichkeitsverteilungen

"Erstellung eines verfeinerten mathematischen Modells"
hier ist erforderlich bzw. kann folgendes erforderlich sein zu

- approximieren
- diskretisieren
- partitionieren (im Sinne des "Teilen zu Herrschen")
- sequentiallisieren bzw. partitionieren (im Sinne einer Betriebsmittelzuordnung. Z.B. bei Mehrprozessorsystemen)

um eine **effiziente** algorithmische Beschreibung des Modellverhaltens zu erreichen.
(Zu "Validierung und Verifikation", siehe Folie 12. Verweis zum späteren Kapitel.)
Simulationsmodelle können optimale Lösungen nicht identifizieren. Sie unterstützen nur den Vergleich verschiedener, alternativer Lösungen.
Eine „Übervereinfachung“ des Simulationsmodells kann zu einem (unverträglichen) Verlust an Genauigkeit und Allgemeingültigkeit führen.
Eine „Überverkomplizierung“ des Simulationsmodells beeinträchtigt die Simulationseffizienz (und steigert die Simulationskosten).
Eine Simulation ist teurer als eine analytische Lösung und bedarf viel detailliertere Eingaben.
Ein Simulationsverfahren kann manche strukturelle Änderungen (z.B. von linearen zu einem nicht-linearen Modell) viel besser vertragen als reine analytische Verfahren.
Simulation ist auch ein Experimentierverfahren. Der Entwurf von Experimenten ist **kein** triviales Problem. Kenntnisse u.a. aus der Psychologie, Verhaltensforschung und Wahrscheinlichkeitsrechnung sind erforderlich!

1.7.3 Die kontinuierlichen Simulationsmodelle

Diskrete Simulationsmodelle werden in der Regel beschrieben durch **algebraischer Gleichungen**.

Kontinuierliche Simulationsmodelle werden in der Regel beschrieben durch **Differentialgleichungen (Differentialgleichungssystem)**.
Ursprünglich -(gelegentlich auch heute)- wurden (werden) **Analogrechner** für die kontinuierliche Simulation eingesetzt.
Mit der Entwicklung von

- Digitalrechnern,
- Programmiertechniken und,
- speziellen Simulationssprachen (vgl. PACTOLUS, 1964 , IBM 1620)

reduziert sich die kontinuierliche Simulation auf die numerischen Lösung des Differentialgleichungssystems.

2 Systeme

2.1 Definitionen & Grundlagen

2.1.1 Def.: System

Ein System ist eine abgegrenzte Anordnung von Objekten, die sich gegenseitig beeinflussen.

2.1.2 Def.: Systemgrenzen

Die Systemgrenzen umfassen alle Objekte eines Systems sowie ihre Beziehungen untereinander.

2.1.3 Def.: Systemumgebung

Die Systemumgebung umfaßt die Elemente des Universums außerhalb der Systemgrenzen. Diese Elemente können das Verhalten des Systems zwar beeinflussen, sie sind aber vom System selbst nicht kontrollierbar.

2.1.4 Def.: Offenes System

Ein offenes System ist ein System, dessen Systemumgebung nicht leer ist.

2.1.5 Def.: Geschlossenes System

Ein geschlossenes System ist ein System, dessen Systemumgebung leer ist.

2.1.6 Def.: Zustand eines Systems

Der Zustand eines Systems wird bestimmt von:

- Objekte (Komponenten),
- Attributen (charakteristische Eigenschaften (Substantive und Adjektive)),
- Aktionen

zu einer gegebenen Zeit.

Sie sind erforderlich, um das zukünftige Verhalten des Systems vorhersagen zu können.

Der Zustand eines Systems ist die geordnete Menge der Werte der Zustandsvariablen, die bestimmt werden von: Siehe oben.

2.1.7 Def.: Variablen

Variablen repräsentieren die Beziehungen zwischen den Komponenten und dem System.

2.1.8 Def.: Deterministisches System

[23] Die Übergangswahrscheinlichkeit des Systems von einem bestimmten in einen anderen bestimmten Zustand ist gleich 100% oder 0%.

2.1.9 Def.: Stochastisches System

[23] Die Übergangswahrscheinlichkeit des Systems von einem bestimmten in einen anderen bestimmten Zustand ist mindestens für ein Paar von Zuständen kleiner als 100% und größer als 0%.

2.1.10 Zwei Arten von Systemen

[7] Es gibt zwei Arten von Systemen:
1) Kontinuierliche Systeme:
   Variablen unterliegen fließenden Änderungen.
2) Diskrete Systeme:
   Variablen verändern sich unverzüglich in diskreten Schritten.

2.1.11 Aufteilung in Subsysteme

[21] Ein System besteht aus mehreren Subsystemen, die wiederum aus mehreren Subsystemen bestehen können.
[18] Ein System kann häufig in seine Subsystemen aufgeteilt werden, aber manchmal gehen dadurch wichtige Eigenschaften verloren oder werden verfälscht.

2.1.12 Uneinheitlichkeit von Nachbildungen

[19-20] Die Nachbildung eines Systems muß nicht einheitlich sein. Eine Unterscheidung der verschiedenen Nachbildungen des Systems ist nötig. Eine mögliche Unterscheidung kann nach dem Zweck der Nachbildung erfolgen.

2.2 Elemente von Systemen

[21] Mit einem System werden assoziiert:

- Objekte [entities]
- Attribute von Objekten (Eigenschaften; Parameter und Variablen)
- Aktionen (Prozesse, welche die Attribute von Objekten verändern)
- Beziehungen/Abhängigkeiten [interrelationships] (zwischen Objekten und zwischen Objekten und dem System) (Abhängigkeiten werden durch Parameter und Variablen ausgedrückt)

2.2.1 Kategorisierung von Variablen und Systemen

[22] Einteilung, Kategorisierung von:

- Variablen
  - abhängig - unabhängig (frei zu simulieren)
  - kontrollierbar - unkontrollierbar
  - kontinuierlich - diskret
- System
  - natürlich - künstlich
  - stabil - instabil
  - dynamisch - statisch
  - deterministisch - stochastisch
  - anpassungsfähig [adaptive] - anpassungsunfähig
  - linear - nicht linear
  - sich (teilweise) wiederholend - periodisch - sich nicht wiederholend [unique]

2.2.2 Nicht beobachtbare Variablen

[22] Einige Variablen können weder beobachtet noch bewertet werden. Diese Variablen müssen abgeschätzt werden.

2.2.3 (Statische und dynamische) Zustände

Welche Eigenschaften haben Systeme mit statischen Zuständen?
[23] Das System nimmt letztendlich einen bestimmten Zustand für die Zukunft ein.
Welche Eigenschaften haben Systeme mit dynamischen Zuständen?
[23] Das System nimmt letztendlich keinen bestimmten Zustand für die Zukunft ein.
Wie werden statische Zustände mathematisch charakterisiert?
[24] Algebraische Gleichungen charakterisieren statische Zustände.
Wie werden dynamische Zustände mathematisch charakterisiert?
[24] Differentialgleichungen oder Differenzengleichungen charakterisieren (normalerweise) die dynamischen Zustände.

2.3 Untersuchungsmethoden von Systemen (Methoden der Systemanalyse und -Bewertung)

2.3.1 Ziele bei der Untersuchung eines Systems [22]:

- Lernen
- Entwerfen (Gestalten) [design]
- Verändern oder Beibehalten
- Kontrollieren (wenn möglich)

des Verhaltes des Systems.

2.3.2 Am Anfang der Untersuchung eines Systems

[25] Am Anfang der Untersuchung eines Systems muß zunächst folgendes festgelegt werden:

- Definition des Systems
- Systemgrenzen
- Systemumgebung

2.3.3 Def.: Prototyp

Ein Prototyp ist:

- ein dynamisches physikalisches Modell und
- eine funktionale Nachbildung des originalen Systems.

2.3.4 Def.: Imitationsmodell

Ein Imitationsmodell ist die Nachbildung der Natur/Landschaft.

2.3.5 Def.: Analoges Modell (aus Prüfungsprotokoll)

Nachbildung eines anderen, ähnlichen Systems, das durch das gleiche Gleichungssystem. beschrieben wird.

2.3.6 Vier Untersuchungsmethoden

[1] Systeme können untersucht werden durch:

- Direktes Experimentieren am System
- Konstruktion eines Prototyps
- Konstruktion von analytischen (mathematischen/logischen) Modellen
- Simulation

2.3.7 Bewertung der vier Untersuchungsmethoden

Direktes Experimentieren und Prototypen

[21] Die konventionellen Untersuchungsmethoden durch das direkte Experimentieren oder durch die Konstruktion von Prototypen gehören immer mehr zur Vergangenheit.

Analytische Methoden

[19] Die konventionellen analytischen Methoden sind nicht leistungsfähig genug, um die heutigen multidisziplinären Probleme zu handhaben.

Simulation

Die Simulation wird immer häufiger eingesetzt.
(Die Simulation ist übrigens Thema des folgenden Kapitels.)

3 Modellierung

3.1 Definitionen & Grundlagen

Modellieren ist die Begabung, die kleinste (Teil-) Menge von Variablen (eines Systems) auszuwählen, um ein Modell zu bilden, das die gewünschten Eigenschaften des realen Systems in angemessener Art und Weise darstellen.

3.1.1 Def.: Modell

Ein Modell ist eine vereinfachte Nachbildung (Repräsentation) eines Systems.

[135] Ein Modell eines Systems besteht aus Komponenten/Objekten und ihren gegenseitigen Beziehungen. Die Komponenten werden beschrieben durch Attribute.

Ein Modell ist ein bestimmter Ausdruck einer Hypothese über das Verhalten eines Systems.

3.1.2 Def.: Simulationsmodell

Ein Simulationsmodell ist ein Modell, das in ein Programm eingebettet ist.

3.1.3 Ziel (der Modellierung und Simulation)

[31] Modelle ermöglichen/unterstützen:

- das Lernen (Verständnis) des Systems
- das Entwerfen (Gestalten) des Systems
- das Kontrollieren des Verhaltens des Systems (wenn möglich)
- die Vorhersage von Antworten des Systems auf eine Aktion, ohne diese Aktion wirklich auszuführen
- das Durchführen kontrollierter Experimente am Modell
- das schnelle, billige und harmlose Bewerten von alternativen Eingaben/Systemen

[31-32] Außerdem:

- Modelle können beim **Training** von Personal eingesetzt werden.
- Modelle sind hilfreich beim klaren **Ausdrücken von Ideen**.
- Experimente mit den Modellen führen oft zu **neuen Hypothesen und Formalisierungen** von Wissen über das System.
- Mit Hilfe mathematischer Modelle können **optimale Lösungen** gefunden werden.
- [2] Ohne die Hilfe mathematischer Modelle können zwar nicht unbedingt optimalen aber dafür mindestens **befriedigenden Lösungen** gefunden werden.
- [31] Modelle werden zur strategischen und taktischen **Planung** eingesetzt.

3.1.4 Nachteile von Modellen

3.1.4.1 Unperfekter Charakter von Modellen [10]:

- Das reale System ist niemals vollständig und
- das Modell ist niemals eine exakte Nachbildung des realen Systems.

3.1.4.2 Abhängigkeit vom Modellierer

[25] Die Definition der einzelnen Systemelemente und ihre Attribute variieren stark von einer analysierenden Person zur nächsten, weil die Menschen voreingenommen sind.

3.1.4.3 Simplifikation bei der Modellierung

[8] Für die Simulation muß bei der Modellierung ein möglichst geeigneter Kompromiß zwischen Übersimplifikation und Detailtreue angestrebt werden.

Daraus folgt die Konsequenz:

Minimierung der Anzahl der Zustandsvariablen:

[23] Die Anzahl der möglichen Permutationen und Kombinationen von Zustandsvariablen erhöht sich stark mit der Erhöhung der Anzahl und des Wertebereiches dieser Variablen.
<=> Bei Erhöhung der Anzahl der Komponenten oder Zustände erhöht sich die Komplexität des Systems stark.

3.1.5 Computer und Modellierung/Simulation

3.1.5.1 Entwicklung der Computer (Heutzutage: Hochleistungscomputer)

[11] Das Aufkommen der Simulation war an die Entwicklung der Computer gekoppelt.
Parallel dazu wird die Weiterentwicklung dieses Gebietes durch die Hochleistungrechner der jüngsten Generation sowie die neuen Software-entwicklungsumgebungen stark beeinflußt.
[29-30] Heutzutage werden immer öfter Modelle von (existierenden oder noch nicht existierenden) Systemen konstruiert und anschließend mit Hilfe von Hochleistungscomputer (High-Speed-Computern) untersucht.
Die Hochleistungscomputer sind ein mächtiges Werkzeug und ein mächtiger Partner geworden im gesamten Modellierungs-Prozeß.

3.1.5.2 Die Rolle des Computers bei der Modellierung

[29-30] Die Computer sind die Gastgeber [host] von Modellen und sie stellen eine Umgebung zur Verfügung, in welche Modelle erschaffen, getestet und manipuliert werden.

3.2 Klassifikation von Modellen

Es gibt eine Reihe von Typen von Modellen. Siehe dazu Neelamkavil, S.32ff.
[30] Der Ausdruck „Modell“ hat verschiedene Bedeutungen, je nach der Art des Modells.
[30] Das Modell eines Systems kann

- mental,
- physikalisch oder
- symbolisch

sein.
Eine mögliche Klassifikation von Modellen:

a) kontinuierlich: Die Variablen, die ein kontinuierliches Modell beschreiben, werden kleinen Änderungen unterzogen.
diskret: Die Variablen, die ein diskretes Modell beschreiben, nehmen Werte nur in diskreten Schritten ein.
hybrid: Es sind sowohl Eigenschaften vom kontinuierlichem als auch vom diskreten Modell vorhanden.

b) linear: Bei linearen Modellen besteht eine lineare Beziehung zwischen Input und Output.
nicht-linear: Bei nicht-linearen Modellen besteht eine nicht-lineare Beziehung zwischen Input und Output.

c) deterministisch: Deterministische Modelle haben vorhersagbare Beziehung zwischen Input und Output.
stochastisch: Stochastische Modelle haben keine zuverlässig vorhersagbare Beziehung zwischen Input und Output.

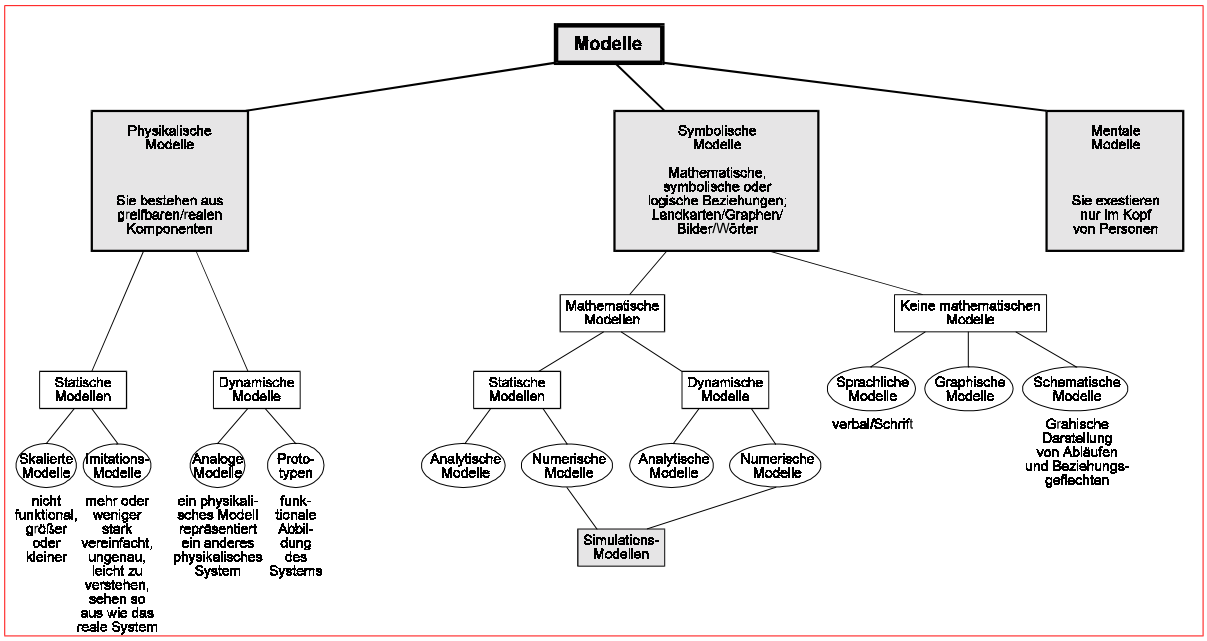


Abb.: Kategorisierung der Modelltypen.

3.3 Methodik des Modellierens

3.3.1 Wieso benötigt man beim Modellieren eine Methode?

[38] Die Modellierung ist ein interaktiver Prozeß und benötigt deshalb eine Methodik.

3.3.2 Nötige Kenntnisse von Modellierern

[39] Der Modellierer muß mit den Konzepten und Techniken vertraut sein, die zur Untersuchung solcher Probleme nötig sind, die untersucht werden sollen.

3.3.3 Dokumentation

Die Dokumentation besteht aus:

- einer vollständigen Beschreibung des **Modells** mit den **Voraussetzungen** (Annahmen)
- einer Beschreibung der mathematischen und computertechnischen **Methoden**
- einer Beschreibung der durchgeführten **Versuche**
- einer Beschreibung der **Validität** des Modells
- einer Beschreibung der **Kosten**
- **Empfehlungen** fürs weitere Arbeiten am/mit dem Modell/System

3.3.4 Die Methodik der Modellierung in 6 Schritten [39-45]:

1)	**Problem-Beschreibung:**	Das Problem muß klar und korrekt neu formuliert werden.
2)	**Systemanalyse:**	1. Beschreibung der **charakteristischen Elemente/Eigenschaften** des Systems:: • Der Objekte (Komponenten) • Der Attribute (Variablen, Parameter) der einzelnen Objekte [S.41 oben] • Der (offensichtlichen) Beziehungen zwischen den einzelnen Objekten (siehe auch 3.) • Eingaben (Input des Systems) und Ausgaben (Output des Systems) • Der Systemgrenze • Der Systemumgebung 2. Zusammentragen und Analyse von relevanten **Daten**. 3. Bestimmung von **Mustern, Beziehungen und statistischen Eigenschaften** des Systems mit Hilfe von Graphen, Histogrammen, und Regressionsanalyse.
3)	**Formulierung des Modells:**	1. **Visualisierung** (z.B. durch ein Flußdiagramm). 2. **Simplifizierung** (Vereinfachung): Nicht mehr Variablen wie gerade nötig. 3. Suchen einer optimalen/suboptimalen **Lösung**, z.B. durch Lineare Programmierung.
4)	**a) Verifikation („Syntax“):**	Die **Verifikation** ist eine Überprüfung, ob das Modell eine glaubwürdige Nachbildung von dem ist, was erwartet wurde (was in den Spezifikationen festgelegt wurde).
	b) Validierung („Semantik“):	Die **Validierung** ist eine Überprüfung der Übereinstimmung mit der Realität. Man stellt sich dabei die Frage: Ist das Modell eine Nachbildung des Systems mit angemessener Genauigkeit?
	c) Zertifikation (Bescheinigung):	Die **Zertifzierung** eines Modells ist a) die offizielle Bestätigung der Korrektheit des Modells durch unabhängige (neutrale) nationale und internationale Gremien oder b) eine Validierung durch unabhängige (neutrale) Gutachter.
5)	**Implementation**	
6)	**Dokumentation**	

3.4 Überprüfung von Modellen (Verifikation, Validierung, Zertifikation)

3.4.1 Vertifikation

3.4.1.1 Def.: Verifikation

Die Vertifikation ist eine Überprüfung der (laut Skript):

- Strukturellen Konsistenz
  (Suchen nach Instabilitäten und Ungenauigkeiten)
- Algorithmischen Korrektheit
- Programmtechnischen Korrektheit
  (Suchen nach Schreib- und Programmierfehlern)

3.4.1.2 Vorgehensweise bei der Verifikation

[76-77] Zur Vertifikation gehört das:

- Korrekte Überprüfen des Programms
- Anhören anderer Experten
- Finden von falschen Voraussetzungen
- Überprüfen von Konvergenz, Genauigkeit und Robustheit (auch bei fehlerhaften Daten)
- Finden von Schreibfehlern

3.4.2 Validierung

3.4.2.1 Def.: Validierung

[Skript] Die Validierung ist ein Verfahren zur Überprüfung, ob das Modell eine adäquate Nachbildung des originalen Systems ist und ob es dessen Verhalten mit Hinblick auf die angestrebten Anwendungen ausreichend genau imitiert.

[In meinen Worten:]

Die Validierung ist eine Überprüfung der Übereinstimmung mit der Realität.

Man stellt sich dabei die Frage:
Ist das Modell eine Nachbildung des Systems mit angemessener Genauigkeit?

3.4.2.2 Vorgehensweise bei der Validierung

Allgemein:

Bei der Validierung wird die Antwort des Modells mit der Antwort des beobachteten Systems oder mit den theoretischen Voraussetzungen verglichen.

Speziell:

Die Validierung besteht aus der Überprüfen der

- Validität der Konzepten
- Validität der Methodik
- Validität der Daten [73]
- Validität der Ergebnisse
- Validität der Schlußfolgerungen
- Validität der Inferenz

3.4.2.3 Validität der ...

3.4.2.3.1 Validität der Konzepte

[71] Die Validität der Konzepte/Theoreme kann nicht anhand des Modells sondern nur in der Realität getestet werden.
[71] Es gibt bei der Untersuchung der Validität von Konzepten zwei Ansätze:
1) Rationaler Ansatz:
   - Die grundlegenden Annahmen werden (eventuell einzeln) getestet/überprüft.
   - Das Modell selbst wird als wahr angesehen, wenn die Annahmen auch wahr sind.
2) Empirischer Ansatz:
   - Die Validität der grundlegenden Annahmen ist egal.
   - Die auf die grundlegenden Annahmen aufbauenden Annahmen werden getestet.
   - Die grundlegenden Annahmen werden somit indirekt gleich mit-getestet.

3.4.2.3.2 Validität der Methodik

[72] Gemeint ist die Methodik, die bei der Formulierung des Modells und der nachträglichen Lösung des Problems angewandt wird.
[72] Mögliche Fehlerquellen:
Die Überprüfung der Validität der Methodik besteht aus einer Überprüfung der:
- Approximation (Zu große Fehler durch die Approximation)
- Auswahl der mathematischen Lösungsverfahren

3.4.2.3.3 Validität der Daten [73]

Die Daten können aus einer Reihe von Gründen fehlerhaft sein. ...
Die Überprüfung der Validität der Daten besteht aus einer Überprüfung der:
- Datenerfassung (unter anderem der Vollständigkeit)
- Datenverarbeitung
- Repräsentativität

Die Fehlerhaftigkeit der Daten kann z.B. mit der Chi-Quadrat-Methode untersucht werden.

3.4.2.3.4 Validität der Ergebnisse

[74] Es geht hierbei darum, wie sehr der Modell-Output mit den theoretischen und experimentellen Ergebnissen übereinstimmt. Also: Nur die Ergebnisse zählen.
[74] Die Ergebnisse können z.B. mit den folgenden Methoden verglichen und interpretiert werden:
- Analyse der Varianz
- Analyse der Regression
- Faktor-Analyse (?)
- Chi-Quadrat-Test ...

3.4.2.3.5 Validität der Schlußfolgerungen

[75] Wenn nach ausführlichen Untersuchungen alle vernünftigen Menschen die gleiche Schlußfolgerung ziehen, dann ist die Schlußfolgerung valide.

3.4.2.3.6 Validität der Inferenz

Überprüfung des Modells auf Meinungskonformität.

3.4.2.4 Zwei Punkte sind zu beachten

[76] Bei der Validierung sind zwei Punkte vom Erschaffer wie auch vom Benutzer des Modells zu beachten:
1) Die **Genauigkeit** der vom Modell ermittelten Ergebnisse darf die vorher festgelegten Schranken nicht unterschreiten.

2) Das Modell soll abhängig von der vereinbarten **Anwendung** erschaffen und benutzt werden. Somit muß die Validität nicht perfekt sein, sondern den an die Anwendung des Modells gestellten Anforderungen entsprechen.

3.4.2.5 Lockern der Einschränkung

[75] Um ein Modell noch gründlicher auf Validität zu überprüfen, kann man seine Einschränkungen immer mehr lockern.

3.4.2.6 Nur angenäherte Validität ist möglich

[10] Validität kann nur angenähert, aber nie erreicht werden. [70] Eine wahre Validierung ist also unmöglich. Die beiden einzig möglichen Ergebnisse einer Validierung lauten deshalb:
a) Entweder es ist nicht-Validität gezeigt worden oder
b) Es ist bisher noch keine nicht-Validität gezeigt worden.
[71] Die Frage bei der Validierung lautet:
Wie nahe kommt das zu überprüfende Modell der Validität, und zwar in Beziehung auf die jeweiligen Anforderungen.
Diese Frage kann durch die Untersuchung der Validität des Konzepts, der Methodik, der Daten, der Ergebnisse der Schlußfolgerungen und der Inferenz beantwortet werden.

3.4.2.7 Zwei Arten von Validierungsfehlern

3.4.2.7.1 Modellersteller-Risiko

Ein gültiges Modell wird doch nicht validiert. (Das Verwerfen eines schon erstellten Modells ist teuer.)

3.4.2.7.2 Modellanwender-Risiko

Ein falsches Modell wird validiert. (Eine Anwendung eines fehlerhaften Modells kann ein Desaster anrichten.)

3.4.3 Bescheinigung/Zertifikation

3.4.3.1 Def.: Modell-Bescheinigung/-Zertifikation

[Zum Teil von mir:] Die Zertifizierung eines Modells ist a) die offizielle Bestätigung der Korrektheit des Modells durch unabhängige (neutrale) nationale und internationale Gremien oder b) eine Validierung durch unabhängige (neutrale) Gutachter.

3.4.3.2 Wieso ist die Zertifikation sinnvoll?

• Weil jeder Modellierer voreingenommen ist. Deshalb variiert die Definition der einzelnen Systemelemente und ihre Attribute stark von einer analysierenden Person zur nächsten. Eine Validierung durch eine weitere Person bringt mehr Sicherheit. • Weil somit die Verantwortung auf mehrere Schultern verteilt wird.

3.4.3.3 Modell-Bescheinigung/Zertifikation in der Praxis

In der Realität ist die Zertifikation eines Modells jedoch kaum möglich, da
- unabhängige Fachleute selten sind und
- eine Zertifikation teuer ist.

3.4.4 Vergleich: Vertifikation und Validierung

3.4.4.1 Möglichkeiten

[77] Die Vertifikation ermöglicht das Untersuchen der Richtigkeit aller einzelnen Schritte bei der Modellierung.
[77] Die Validierung läßt nur einen Vergleich zwischen dem gesamten Modell und dem realen System (und dessen Output) zu.

3.4.4.2 Überprüfung von ...

[79] Die **Vertifikation** überprüft die
- Korrektheit
- Konsistenz und
- Vollständigkeit

der Implementation des Modells im Hinblick auf die Entwurfs-Spezifikationen.

[79] Die **Validierung** überprüft die
- Korrektheit und
- Konsistenz

eines Modells im Hinblick auf das reale System.

3.4.5 Notwendigkeit von Vertifikation und Validierung

[79] Vertifikation und Validierung müssen ein Teil jedes Modellierungs-Projektes sein, und zwar von Beginn an.

[ca.69] Validierung ist das Herz aller wissenschaftlicher Forschung.

3.4.6 Reihenfolge bei der Modellierung

[10] Das vereinfachte spezifische Modell sowie auch das Simulationsmodell werden zunächst einem Verifikationstest, dann einem Validitätstest unterzogen.

3.5 Beispiel (für die Analyse von diskreten Systemen): Poststelle

3.5.1 Beschreibung des Beispiels

[138] Eine kleine Poststelle bedient Kunden, einen nach dem anderen. Die Poststelle arbeitet auf einer first-come-first-served -Basis (FIFO). Außerdem handelt es sich dabei um ein single-server-single-queue -System.

3.5.2 Die interessierenden Fakten

3.5.2.1 Die interessanten Punkte sind [138]:

- Rate der ankommenden Kunden
- Rate der Bedienung
- Auslastung der Bedienung
- durchschnittliche Anzahl von Kunden in der Schlange (Länge der Schlange)
- durchschnittliche Anzahl von Kunden im System
- durchschnittlich in der Schlange verbrachtet Wartezeit
- durchschnittlich im System verbrachte Zeit

3.5.2.2 Die Zustandsvariablen des Systems sind [138]:

- Ankunftszeit jedes Kunden in der Schlange
- Anzahl von Kunden in der Schlange
- Zustand der Bedienung (beschäftigt oder nicht)

3.5.2.3 Ereignisse [138]:

- Ankunft eines Kunden
- Abreise eines Kunden

3.5.3 Die Analyse [selfmade]

3.5.3.1 Analyse in zwei Schritten

Die Analyse erfolgt in zwei Schritten:
Schritt 1: Daten sammeln durch Beobachtung
Schritt 2: Anwenden einer Methode des Problem-Lösens auf die in Schritt 1 gewonnenen Daten. Dabei kann man einen von drei möglichen Ansätzen verfolgen:

3.5.3.2 Approximierungs-Methode

[140] Da nur wenige Daten über einen kurzen Zeitraum vorliegen, muß approximiert werden. Mit einfachen Formeln werden die „interessanten Punkte“ (siehe oben) berechnet. Dabei kommen Formeln aus der Stochastik zum Einsatz.

3.5.3.3 Theoretische Methode

[141] Es wird nach analytischen (mathematischen) Zusammenhängen gesucht, welche das System beschreiben (z.B.: Exponentielle Verteilung der Ankunftszeiten von Kunden). Diese Zusammenhänge werden mit statistische Verfahren (Chi-Quadrat) mit den beobachteten Daten verglichen. Wenn die angenommenen Zusammenhänge zu den Daten passen, wird das System durch die (relativ einfache) mathematische Analyse der mathematischen Zusammenhänge untersucht. Diese Analyse beschränkt sich bei dem Poststellen-Beispiel nur auf ein Einsetzen von wenigen Werten in einfache Gleichungen. Die Komplexität der Analyse ist in erster Linie nicht mehr abhängig von dem Umfang der Meßdaten.
Die theoretische Methode besteht also aus drei Schritten:
1) Suchen nach analytischen Zusammenhängen
2) Vergleich der Zusammenhängen mit den Daten
3) Mathematische Analyse der mathematischen Zusammenhänge

3.5.4 Simulation

[143] Die drei wichtigsten Teile eines Simulations-Modells:
- Die Nachbildung von Ankünften neuer Komponenten [entities] in das Simulations-Modell
- Die Nachbildung von dem, was mit diesen Komponenten in dem Modell passiert
- Die Mechanismen für die Beendigung der Simulation

3.5.5 Drei Arten der Simulation

[151] Die Tabellierung [scheduling] des nächsten Ereignisses und das Updaten des Systemzustandes durch die „Ereignis-orientierte Simulation“ (Methode 2) kann auf verschiedene Arten implementiert werden. Die wichtigsten Ansätze zur Auflistung von Ereignissen sind:
- Ereignisorientierter Ansatz [event scheduling]
- Prozeßorientierter Ansatz [process interaction]
- Aktivitätsorientierter Ansatz [Activity scanning]

Bei allen drei Ansätzen wird die Uhrzeit nur dann vorgerückt, wenn zur aktuellen Uhrzeit keine Zustandsänderung mehr auftreten kann.

3.5.5.1 Ereignis-Tabellierung [event scheduling]

[145] Das Eintreten von Ereignissen in einem System wird durch die Generierung des nächsten Zeitpunktes eines Ereignisses simuliert. Dieser Zeitpunkt wird durch eine Verteilung generiert, welche die Zeit zwischen den Ankünften oder die Dauer der Bedienung ausdrückt.
Siehe meine handschriftliche Abbildung: Flußdiagramm für die Simulation eines Poststellen-Systems.
Eine geordnete List von Ereignissen 1, 2, ... k und der Zeitpunkte ihres Eintretens t_1, t_2, ... t_k werden „gemerkt“. Die passende Ereignis-Routine R_i

wird ausgeführt im simulierten Zeitpunkt t_i wobei $t_i = \min(t_1, t_2, ... t_k)$.

3.5.5.2 Prozeß-Interaktion

[152] Siehe Neelamkavil.

3.5.5.3 Activity scanning

[154] Siehe Neelamkavil.

4 Simulation Teil 1: Grundlagen

4.1 Def.: Simulation

Simulation ist das **Nachbilden** ausgewählter relevanter Aspekte des Verhaltens eines Systems.
Weiteres zum Begriff Simulation:

- Die Simulation erfolgt in **Echtzeit** bzw. in „skalierter“ Zeit.
- Simuliert wird, indem **Experimente** mit dem Modell des Systems durchgeführt werden.
- Der Begriff *Simulation* stammt vom lateinischen *simulare* und meint somit „*Ähnlich-machen*“, „*Nachbilden*“.

4.2 Wann wird simuliert?

Für die Simulation müssen drei Bedingungen erfüllt sein:

- Das **mathematische Modell** hat keine effiziente (analytische bzw. numerische) Lösung.
- Die **Validierung** vom Modell und von Ergebnissen ist möglich.
- Die erwartete **Genauigkeit** der Simulations- Ergebnisse ist konsistent mit den Anforderungen des Problems.

Falls alle Bedingungen erfüllt sind, dann sollte simuliert werden, wenn mindestens eine der beiden folgenden Punkte erfüllt sind:

- Es wäre
  - **teuer** oder
  - **aufwendig** oder
  - **gefährlich** oder gar
  - **unmöglich**,

mit dem realen System oder einem physikalischen Modell zu experimentieren.

- Man möchte das vorherige, gegenwärtige bzw. zukünftige Systemverhalten bzgl. einer künstlichen **Zeitskala** studieren.

4.3 Nachteil der Simulation

- [3] Simulations-Ergebnisse sind wesentlich ungenauer als analytische Ergebnisse.
- [8] Simulationsmodelle können nicht optimale Lösungen bestimmen, sie können nur mehrere alternative Lösungen vergleichen.

4.4 Was unterscheidet die Simulation von der Monte-Carlo-Simulation?

[4] Bei der **Simulation** wird ein
stochastischer Prozeß durch ein
stochastisches Modell nachgebildet.

[4] Bei der **Monte-Carlo-Simulation** wird ein
deterministischer Prozeß durch ein (approximiertes)
stochastisches Modell nachgebildet.

4.5 Zwei Ansätze in der Simulation

[7] Es gibt zwei Ansätze in der Simulation:

1) Kontinuierlicher Ansatz:
   Ein kontinuierliches Modell für ein kontinuierliches System oder
   eine kontinuierliche Approximation eines diskreten Systems.
2) Diskreter Ansatz:
   Ein diskretes Modell für ein diskretes System oder
   eine diskrete Approximation eines kontinuierlichen Systems.

Beispiele:

Kontinuierliches Modell für ein kontinuierliches System	Differentialgleichung
Kontinuierliche Approximation eines diskreten Systems	Ein exponentielles Modell für das Bevölkerungswachstum
Diskretes Modell für ein diskretes System	Ganzzahlige Kodierung der Stelle der Bücher in einer Bibliothek
Diskrete Approximation eines kontinuierlichen Systems	Ein digitalisiertes Bild

4.5.1 Kontinuierlicher Ansatz [7]:

Die kontinuierlichen Simulationsmodelle werden beschrieben durch:

- ein deterministisches Differential oder
- algebraische Gleichungen

Kontinuierliche Systeme werden mit analogen oder digitalen Computern simuliert.

4.5.2 Diskreter Ansatz [7]:

Zufalls-Ereignisse werden vom Computer an verschiedenen Zeitpunkten generiert. Die Ergebnisse/Antworten des Systems auf die Zufalls-Ereignisse werden beobachtet und analysiert.

5 Simulation Teil 2: Erzeugung von Zufallszahlen

5.1 Grundlagen aus der Wahrscheinlichkeitsrechnung

Die Kenntnis der Grundlagen aus der Wahrscheinlichkeitsrechnung werden in dieser Vorlesung vorausgesetzt. Es folgt deshalb nur ein kurzer Überblick über die Begriffe und grundlegenden Zusammenhänge.

5.1.1 Verteilungsfunktion und Dichtefunktion

Die Dichtefunktion wird als die erste Ableitung der Verteilungsfunktion definiert.

$$f_X(u) = F_X'(u)$$

Oder:

$$F_X(u) = \int_{-\infty}^{u} f(\tau)\, d\tau$$

Die Dichtefunktion hat die folgenden Eigenschaften:

$f_X(u)\, du = F_X(u)$
$f_X(u)\, du = 1$
$f_X(u) \geq 0$ für alle u
$p\{ a \leq x(w) \leq b \}$
$= f_X(u)\, du$
$= F_X(b) - F_X(a)$

5.1.2 Zufallsvariablen und Zufallszahlen

Zahlreiche Phänomene beruhen (wenigstens modell-mäßig) auf bestimmten physikalischen Mikrovorgängen oder Elementarergebnissen, wie beispielsweise

- Die Wärmebewegung eines Gasmoleküls
- Das Ergebnis eines Wurfs mit einem Würfel

Sie reichen jedoch nicht aus zur Beschreibung der Gesamtphänomene wie der Temperatur und des Druckes eines Gases bzw. des gesamten Spiels mit einem Würfel.

Derartige Elementarergebnisse sind im einzelnen *nicht vorhersagbar*:

- Die Quantenphysik kennt Elementarergebnisse, die (nach heutiger Auffassung) prinzipiell unvorhersagbar sind.
- Der einzige Wurf mit einem Würfel ist sehr wohl ein determinierter mechanischer Vorgang, der deshalb als "zufällig" betrachtet wird, weil sein Bewegungsablauf viel zu verwickelt ist, als man ihn mit den bekannten Gesetzen der Mechanik beschreiben könnte, wobei die Schwierigkeit bereits bei der Unmöglichkeit beliebig genauer Angaben und damit die Reproduzierbarkeit der Anfangsbedingungen beginnt.

5.1.3 Ereignisse

Die zunächst noch ungeordnete Gesamtheit der obigen Ergebnisse bildet die Grundlage für bestimmte Zusammenfassungen und Auswahlvorschriften, die man Ereignisse nennt. Sie treten unter den jeweiligen Bedingungen nur mit einer gewissen Wahrscheinlichkeit auf.
Beim Spiel mit einem Würfel kommen die möglichen Ergebnisse durch determinierte Einzelwürfe zustande, aber die Vorhersagbarkeit der einzelnen Wurfergebnisse geht verloren. Die Durchführung einer hinreichend großen Anzahl n von Würfelvorgängen zeigt, daß die Ergebnisse alle mit der gleichen **relativen Häufigkeit**

$$H_n(W) = \frac{k(W)}{n} \qquad W \in E = \{ 1, 2, 3, 4, 5, 6 \}$$

auftreten; k gibt an, wie oft sich das Ergebnis w einstellt. Als erste Annäherung kann man sagen, daß

$$\lim_{n \to \infty} H_n(W) = p(W)$$

gilt.

5.1.4 Zufallsvariable

Um eine "versuchsunabhängige" Notation zu ermöglichen, definiert man

$$x: \; E \to R$$

Dadurch entsteht eine auf der Ergebnismenge E definierte Funktion $x(w)$, die Zufallsvariable genannt wird, wenn die Ungleichung

$$a < x(w) \le b; \qquad a,b \in R$$

eine Teilmenge von E definiert, der als Ereignis eine Wahrscheinlichkeit zugeordnet werden kann. Man nennt $x(w_i) = x_i$ eine "Zufallsgröße" oder eine "Realisierung der Zufallsvariable". (Beim Würfelspiel ist die Zufallsvariable eine Funktion, die beim jeden Wurf einen der Werte 1 bis 6 annimmt.)
Allgemein wird gefordert, daß eine Zufallsvariable mit einer von Null verschiedenen Wahrscheinlichkeit nur endliche Werte annehmen kann. Ein Ereignis wird durch die jeweiligen Ergebnisse w aus E definiert, die der angegebenen Ungleichung genügen.

5.1.5 Verteilungsfunktion

Die **Verteilungsfunktion** über die Wahrscheinlichkeit für das Eintreten des Ereignisses

$$A = \{x(w) \leq b\}$$

wird dadurch eingefügt, daß man die Schranke b über den gesamten Wehrbereich der Zufallsvariable $x(w)$ variiert.

$$F_X(b) := p(x(w) \leq b)$$

Die Verteilungsfunktion gibt die Wahrscheinlichkeit dafür an, daß die Ungleichung $x(w) \leq b$ erfüllt ist und ordnet somit dem Ereignis A eine Zahl zu: p(A).

Eigenschaften der Verteilungsfunktion:

- $0 \leq F_X(b) \leq 1$
- F wächst monoton von 0 nach 1
- Aus $b_1 > b_2$ folgt $F_X(b_1) > F_X(b_2)$
- Die Wahrscheinlichkeit für das Eintreten des Ereignisses
  $$a \leq x(w) \leq b$$
  ist gegeben durch
  $$p(a \leq x(w) \leq b) = F_X(b) - F_X(a)$$
- Die Wahrscheinlichkeit, daß die Zufallsvariable genau einen Wert c annimmt, ist gegeben durch
  $$p(x(w) = c) = F_X(c + 0) - F_X(c - 0);$$
  d.h. bei unstetigen Verteilungen nimmt
  $$p(x(w) = c)$$
  den Wert der Sprunghöhe bei c an, bei stetigen Verteilungen ist die "Punktwahrscheinlichkeit"
  $$p(x(w) = c) = 0$$

5.1.6 Beispiel: Parkplatzsimulation

Ankunftszeit (Minuten) X	Anzahl angekommender Autos	relative Häufigkeit (bzgl. 50)	kumulative Verteilung F(X)
0 - 9	14	0,28	0,28
10 - 19	10	0,20	0,48
20 - 29	12	0,24	0,72
30 - 39	14	0,28	1,00
	n=50	Σ=1,00	

Modell:
Die relative Häufigkeit wurde aus der tatsächlichen Anzahl von angekommenen Wagen in einem bestimmten Zeitintervall berechnet.
Die Verteilung deutet dann an, mit welcher Wahrscheinlichkeit PKWs *bis zu dem Zeitpunkt* angekommen sind.
Aus r = 0,52 mit 0,48 < 0,52 < 0,72 folgt, das der LKW im Zeitraum von 20 bis 30 Minuten einzuordnen ist.

$$t_i = \frac{30-20}{0{,}72-0{,}48} = \frac{10}{0{,}24} = 41{,}66,$$

$$X_j = 20 + (0{,}52 - 0{,}48) \cdot 41{,}66 = 21{,}66$$

Ergebnis:

X_j = 21,66 min.

5.1.7 Beispiel: Tankstellen-Simulation

Fall	Säule	Kraftstofftyp	Durchschnitt. Kundenzahl	Kumulative Distribution
X_0	1	Bleifreies Benzin	1/3	1/3
X_1	2	Benzin Super	1/6	1/2
X_2	3	Superplus Bleifrei	1/3	5/6
X_3	4	Diesel	1/6	1
			[p(X)]	[F(X)]

5.2 Generierung von Zufallszahlen

Bemerkung: Hier werden nur die Algorithmen zur Generierung von Pseudo-Zufallszahlen angegeben. Zu den wichtigsten Algorithmen sind im Buch von Neelamkavil auch **Beispiele** zu finden.

Zufallszahlen können generiert werden durch:

- Tabellen
- physikalische Methoden (Würfel, Münze, Glücksrad)
  Nachteil: Nicht reproduzierbar, umständlich zu erzeugen
- algebraische Methoden
  Vorteil: Reproduzierbar, einfach und schnell zu erzeugen

5.3 Pseudo-Zufallszahlen

Pseudo-Zufallszahlen (oder Quasi-Zufallszahlen) erhält man durch die Implementierung der algebraischen Methoden auf einem Rechner.

Nachteil: Die Länge der Folge der generierbaren Zufallszahlen ist endlich

5.4 Anforderungen an Pseudo-Zufallszahlen (Generatoren)

Die Kriterien von guten Pseudo-Zufallszahlen-Generatoren:

- Der Generator muß **verschiedene Reihen** von Zahlen erzeugen können, die den folgenden Anforderungen genügen:
- **Gleichmäßige** Verteilung zwischen 0 und 1
- Statistische **Unabhängigkeit**
- **Reproduzierbar**
- Keine **Wiederholung** einer Folge von Zufallszahlen für jede gewünschte Länge der Reihe
- **Schnelle** Generierung,
  geringer Computer-Speicherplatzverbrauch

5.5 Generierung von Pseudo-Zufallszahlen

5.5.1 Die Generatoren im Überblick

- Lineare kongruentiale Generatoren
- Multiplikative kongruentiale Generatoren
- Kongruentiale Generatoren für Mikrocomputer
- Generatoren nach der von Neumann`s Mittelquadrat-Methode

5.5.2 Lineare kongruentiale Generatoren

Diese Generatoren werden häufig verwendet.

Beschreibung

X_0 = Wurzel [seed]
P = Periode (Anzahl Zahlen bis Wiederholung)

1) Wähle C, C_0, M und X_0,
   i := 0.
2) Anwendung der Formel:
   $X_{i+1} = (CX_i + C_0) \bmod M$
   (Das Ergebnis wird abgerundet.)
3) Normierung des Ergebnisses
   (z.B. auf den Bereich [0,1]),
   Ausgabe des Ergebnisses r_i.
4) i := i+1,
   gehe zu 2).

Maximale Periode

$P = M = 2^b - 1$

Wahl der Werte

1) C_0 ist ungerade
2) C_0 ist teilerfremd zu M ($C_0 \cdot k \neq M$ für nat. k)
3) ist ein Vielfaches jeder Primzahl, die M teilt
4) ist ein Vielfaches von 4, wenn M ein Vielfaches von 4 ist.
5) M ist möglichst groß
6) Die Wortlänge der Maschine (ohne Vorzeicheninformation) beträgt b Bit $\Rightarrow$ $M = 2^b-1$
7) $C_0 = 0{,}21 \cdot M$
8) $C = 8 \cdot K+5$, K = 0, 1, 2 ...

Bedingungen für die maximale Periode

Die Punkte 1) bis 4) sind die Kriterien/Bedingungen für die maximale Periode.

Bemerkung

1) *Vorteile:*
   - Normierung durchs schnelle Shiften des Dezimalpunktes möglich.
   - Große Periode, nahezu M.
2) Neben der möglichst großen Periode ist auch die unkritische Wahl von X_0 ein Ziel bei der Wahl der Werte.

5.5.3 Multiplikative kongruentiale Generatoren

Beschreibung

X_0 = Wurzel [seed]
P = Periode (Anzahl Zahlen bis Wiederholung)

1) Wähle C, M und X_0,
   i := 0.
2) Anwendung der Formel:
   $X_{i+1} = (CX_i) \bmod M$
   (Das Ergebnis wird abgerundet.)
   Gehe dazu wie folgt vor:

   a) Bilde das Produkt CX_i.
      Das Produkt ist (höchstens) 2b Bits lang.

   b) Schneide die b höchsten Bits ab und
      setze X_{i+1} gleich den übriggebliebenen b
      niedrigsten Bits.
3) Normierung des Ergebnisses
   (z.B. auf den Bereich [0,1]).

   Gehe dazu wie folgt vor:
   Bilde die Pseudo-Zufallszahl
   $r_{i+1} = \frac{X_{i+1}}{2^b} = X_{i+1} \cdot 2^{-b}$ durch b-maliges nach links Shiften des Dezimalpunktes des binären Wertes von X_{i+1}.

   Ausgabe des Ergebnisses r_i.
4) i := i+1,
   gehe zu 2).

Es gibt zwei Arten von Generatoren:
1.Generator

Maximale Periode

$$P_{max} = \frac{M}{4} = 2^{b-2}$$

Wahl der Werte

1) $M = 2^b$,
2) $b \geq 4$,
3) $C = 8 \cdot K+5$, $K = 0, 1, 2 ...$,
4) X_0 ist ungerade.

Bemerkung

Vorteil: Der Zufallswert kann durch einfaches und schnelles Abschneiden des Overflows erreicht werden, wenn die obigen vier Bedingungen erfüllt sind.

Nachteil: Kleine Periode, nur 1/4 von M.

2.Generator
(nach Hutchinson, für 32-bit (31+Vorzeichen-Bit) -Rechner)

Maximale Periode

$P = M - 1$

Wahl der Werte

1) Es gilt: b = 31.
2) M ist die größte Primzahl, welche kleiner als 2^b ist.
   Z.B: $M = 2^{31}-1 = 2.147.483.647$
3) :Z.B:: C = 630.360.016

Bemerkung

1) *Vorteile:*
   - Normierung durchs schnelle Shiften des Dezimalpunktes möglich.
   - Große Periode, nahezu M.
2) Die statistischen Eigenschaften sind besser, wenn die drei obigen Bedingungen erfüllt sind.
3) Für die Periodenlänge gilt:
   $P = M-1 = (2^{31}-1)-1 = 2^{31}-2$.
4) Da für die zu ermittelnde Zufallszahl gelten soll:
   $0 \le r < 1$,
   ist eine Normierung/Bereichsanpassung notwendig. Dafür müßte X_{i+1} durch $P+1 = M$ dividiert werden. Da aber bei geschickter Wahl von M gilt:
   $M \cong 2^b$,
   wird statt $M = 2^{31}-1$ durch die folgende Größe dividiert:
   $2^b = 2^{-31} = 0{,}46566613$ E-09

Vorteile / Nachteile

+ einfach
+ leicht zu implementieren
+ schneller als die linearen kongruentialen Generatoren
- keine volle Periode möglich

5.5.4 Generatoren für Mikrocomputer

für 8- und 16-bit- Maschinen:

$$X_{i+1} = \left(5^{13} X_i\right) \bmod 2^{35} \qquad (\text{Das Ergebnis wird abgerundet})$$

5.5.5 Von Neumann`s Mittelquadrat-Methode

Beschreibung

Siehe Neelamkavil S.108

Vorteile / Nachteile

+ einfach
+ leicht zu implementieren
- schlechte Zahlenfolgen (also. nur historisch interessant!)
- generiert schnell zu Null

5.6 Testen von Pseudo-Zufallszahlen

Bemerkung: Hier werden nur die Algorithmen zum Testen von Pseudo-Zufallszahlen angegeben. Zu den wichtigsten Algorithmen sind im Buch von Neelamkavil auch **Beispiele** zu finden.
Der Output von Pseudo-Zufallszahlen-Generatoren muß auf **Gleichverteilung** und **Unabhängigkeit** getestet werden.
Dafür stehen einige **statistische Tests** zur Verfügung. Aber keiner dieser Tests kann **perfekte Zufälligkeit** garantieren. Deshalb sollte jeder Generator im Zusammenhang seiner Anwendung getestet werden.

5.6.1 Die Test-Methoden im Überblick

- Chi-Quadrat-Methode
- Kolmogorov-Smirnov-Test
- Moment-Test
- Serieller Test
- Runs Test
- Gap Test

Außer den ersten beiden Tests können die Tests nur auf gleichverteilte Zufallszahlen angewendet werden.
Die ersten beiden Tests sind sehr praxisrelevant.

5.6.2 Chi-Quadrat-Methode (Häufigkeits-Test)

Ziel

Der Chi-Quadrat-Häufigkeits-Test wird angewendet, um die **Uniformität** einer Menge von Zufallszahlen zu überprüfen.
Mit dem Chi-Quadrat-Test läßt sich die Abweichung zwischen beobachteten und theoretischen Werten messen (Test fürs „goodness-of-fit“). Ein kleiner Wert des Chi-Quadrat (χ_1^2) bedeutet eine gute Übereinstimmung zwischen Experiment und Vorhersage.

Gegeben

n	=	Anzahl der Pseudo-Zufallszahlen
$r_1, r_2, \ldots r_n$	=	Folge von n Pseudo-Zufallszahlen, die über dem Intervall (0;1) verteilt sind.

Vorgehensweise

1) Bestimmung von χ_1^2

Der Wertebereich der Zufallszahlen sei das Intervall $[\alpha;\beta[$ (z.B. [0;1[).
Das Intervall (0;1) wird in s Unterbereiche (Klassen) $[\alpha_i ; \beta_i[$ unterteilt.
Dann sei f_i für i = 1, 2, ... s die beobachtete Häufigkeit von Pseudo-Zufallszahlen aus dem i-ten Intervall, d.h.:
f_i = „Wieviele Zahlen, die größer/gleich α_i und gleichzeitig kleiner als β_i sind, sind in der Folge von Zufallszahlen enthalten?“
Allgemein:

$$\chi_1^2 = \sum_{i=1}^{s} \frac{(\text{beobachtetes } f_i - \text{erwartetes } f_i)^2}{\text{erwartetes } f_i}$$

Speziell, wenn eine Gleichverteilung mit der Stichprobe (beobachtete Werte) verglichen werden soll:

$$\chi_1^2 = \frac{\sum_{i=1}^{s} \left(f_i - \frac{n}{s} \right)^2}{\frac{n}{s}}$$

Wahl der Werte n und s:

$n \geq 100$,

$(n/s) \geq 5$

(mindestens 5 Pseudo-Zufallszahlen pro Unterbereich (bei gleich großen Unterbereichen)).

$s \approx \sqrt{n}$ (Nach Bosch, S.376)

Für den Fall, daß n nicht gegeben ist, läßt sich n wie folgt berechnen:

$$n = \sum_{i=1}^{s} f_i$$

2) Bestimmung von χ^2

Der Wert von χ^2 ist nur von d und α abhängig und kann direkt aus Chi-Quadrat-Tabellen entnommen werden.

d = s-1 = Freiheitsgrad, häufig in % angegeben

α = Signifikanz-Niveau. Es gibt an, mit welcher Wahrscheinlichkeit die Hypothese von der Gleichverteilung nicht verworfen kann. (Neelamkavil, S.94f)

3) Testentscheidung: Vergleich von χ_1^2 mit χ^2

Wenn gilt

$$\chi_1^2 < \chi^2,$$

dann akzeptieren wir die Hypothese, daß die Menge von Pseudo-Zufallszahlen $r_1, r_2, \ldots r_n$ gleichmäßig verteilt sind.

Genauer betrachtet, damit ist nur gesagt, daß die Hypothese *mit der Stichprobe* konsistent ist.

5.6.3 Kolmogorov-Smirnov-Test

5.6.3.1 Vergleich von zwei Folgen von Stichproben

Ziel

Mit dem Kolmogorov-Smirnov-Test läßt sich die Abweichung zwischen beobachteten und theoretischen Werten messen (Test fürs „goodness-of-fit“). Ein kleiner Wert des Wertes K_n bedeutet eine gute Übereinstimmung zwischen Experiment und Vorhersage.

Gegeben

n	=	Anzahl der Pseudo-Zufallszahlen
$r_1, r_2, \ldots r_n$	=	Folge von n Pseudo-Zufallszahlen, die über dem Intervall (0,1) verteilt sind.

Der Test wird jedoch nicht unbedingt immer auf eine Folge von Pseudo-Zufallszahlen angewendet. Deshalb ist der allgemeinere Ausdruck Stichprobe hier besser geeignet. Statt von

$r_1, r_2, \ldots r_n$

sprechen wir besser von der Stichprobe, die außerdem noch sortiert wurde:

$X_1 \le X_2 \le X_3 \le \ldots \le X_n$

Konzept

Aus den einzelnen Werte von X_i wird eine diskrete Verteilungsfunktion $F_n(X)$ gebildet. Wir testen, ob die Verteilungsfunktion $F_n(X)$ zu der bestimmten Verteilungsfunktion $F(X)$ paßt. Die Abweichung beider Verteilungsfunktionen wird mit dem Wert von K_n ausgedrückt.

Vorgehensweise (Bosch, S.382)

1) Bestimmung der empirischen Verteilungsfunktion

Aus einer Stichprobe vom Umfang n bestimme man die empirische Verteilungsfunktion:

$$F_n(X) = \frac{\text{Anzahl der Stichprobenwerte, die kleiner oder gleich X sind}}{n}$$

$$= \frac{\text{Anzahl der } X_i \le X}{n}$$

$$= \begin{cases} 0 & \text{falls} \quad X < X_1 \\ \frac{i}{n} & \text{falls} \quad X_i \le X \le X_{i+1} \\ 1 & \text{falls} \quad X_n \le X \end{cases}$$

Die Sprungstellen von F_n seien $X_1^*, X_2^*, X_3^*, \ldots$.

2) Bestimmung der Testgröße

Die Testgröße K_n ist der Betrag der größten Abweichung zwischen der Empirischen Verteilungsfunktion und der kontinuierlichen Verteilungsfunktion:

$$K_n = \text{maximum}\left(\left|F_n(X) - F(X)\right|\right)$$

$$= \text{maximum}\left(K_n^+, K_n^-\right), \qquad K_n^+, K_n^- \text{ siehe unten.}$$

Die Testgröße K_n wird in drei Schritten berechnet:

$$K_n^+ = \max\left\{\max_{i=1,2,\ldots}\left(F_n\left(X_i^*\right) - F\left(X_i^*\right)\right) \;;\; 0\right\}$$

$$K_n^- = \max\left\{\max_{i=1,2,\ldots}\left(F\left(X_i^*\right) - F_n\left(X_{i-1}^*\right)\right) \;;\; 0\right\} \quad \text{mit} \quad F_n\left(X_0^*\right) = 0$$

$$K_n = \max\left(K_n^+, K_n^-\right)$$

3) Bestimmung der Quantile Q_n

Man gebe ein Testniveau α vor.

Für $n < 40$ benutzt man eine Tabelle.
für $n \geq 40$ benutzt man die Näherung

$$Q_n \approx \sqrt{-\frac{1}{2n}\ln\frac{\alpha}{2}}$$

(Quantil: siehe Bosch, S.141)
4) Testentscheidung: Vergleich von K_n und Q_n
Wenn gilt

$K_n < Q_n$,

dann akzeptieren wir die Hypothese, daß die Menge von Pseudo-Zufallszahlen $r_1, r_2, \ldots r_n$ zur Verteilungsfunktion $F(X)$ passen.

5.6.3.2 KS-Test auf Gleichverteilung

Größe bzw. Repräsentativität einer Stichprobe
Beispiel:

b = 13,
$M = 2^{13} - 1 = 8191$,
C = 127,
X1 = 13.
n = 10.

Die folgende Tabelle enthält nur die theoretischen Werte:

i	X_i	r_i	F(x)	$F_n(i)$	$K_n^+(i)$	$K_n^-(i)$
1	0013	0,00158	0,00158			
2	1651	0,20156	0,00473			
3	4902	0,59846	0,03064			
4	0038	0,00473	0,20156			
5	4826	0,58918	0,58918			
6	6768	0,82627	0,59846			
7	7672	0.93663	0,82627			
8	7806	0,95299	0.93663			
9	0251	0,03064	0,95299			
10	8000	0,97790	0,97790			

In der folgenden Tabelle sind neben den theoretischen auch die gemessenen Werte eingetragen:

i	X_i	r_i	F(x)	$F_n(i)$	$K_n^+(i)$	$K_n^-(i)$
1	0013	0,00158	0,00158	0,1	0,1	0.001
2	1651	0,20156	0,00473	0,2	0,2	0
3	4902	0,59846	0,03064	0,3	**0,27**	0
4	0038	0,00473	0,20156	0,4	0,2	0
5	4826	0,58918	0,58918	0,5	0	0,18
6	6768	0,82627	0,59846	0,6	0,01	0,09
7	7672	0.93663	0,82627	0,7	0	0,22
8	7806	0,95299	0.93663	0,8	0	0,23
9	0251	0,03064	0,95299	0,9	0	0,15
10	8000	0,97790	0,97790	1,0	0,02	0,02

$Q_{10\,;\,0,95} = 0,489$

5.6.4 Moment-Test

Bedingung für perfekte Erfüllung des Tests:
Das erste Moment muß 1/2, das zweite 1/3 und das dritte 1/4 ergeben, wenn n sehr groß ist:

$$\lim_{n\to\infty}\left(\frac{1}{n}\sum_{i=1}^{n} r_i\right) = 1/2,$$

$$\lim_{n\to\infty}\left(\frac{1}{n}\sum_{i=1}^{n} r_i^{\,2}\right) = 1/3,$$

$$\lim_{n\to\infty}\left(\frac{1}{n}\sum_{i=1}^{n} r_i^{\,3}\right) = 1/4.$$

Je genauer die Wert 1/2, 1/3 und 1/4 bei endlichem n angenommen werden, desto sicherer liegt (im Hinblick auf diesen Test) eine zufällige Zahlenfolge vor.

5.6.5 Serieller Test

Ziel

Der serielle Test wird angewendet, um die Zufälligkeit oder Unabhängigkeit einer Menge von aufeinanderfolgenden Zufallszahlen zu überprüfen.

Gegeben

Eine Folge von n Pseudo-Zufallszahlen, die über dem Intervall (0,1) verteilt sind.

Vorgehensweise

1) Aus der Folge der n Pseudo-Zufallszahlen werden $N = n/2$ Gruppen von jeweils zwei aufeinanderfolgenden Zufallszahlen gebildet.
2) Anschließend wird gezählt, wie oft die einzelnen Gruppen in der Folge von Zufallszahlen vorkommen.
3) Haben die Zufallszahlen d dezimale Stellen hinter dem Komma, dann gibt es d^2 mögliche verschieden Zweier-Gruppen.

Theoretisch müßte jede Gruppe N/d^2 in der Folge von Zufallszahlen vorkommen.

Die Abweichungen von diesem theoretischen Wert werden mit dem Chi-Quadrat-Test durchgeführt.

4) Die Untersuchung wird mit Gruppen von 3, 4, ... aufeinanderfolgenden Zahlen wiederholt.

5.6.6 „Gap test“

Ziel

Gegeben

$r_1, r_2, ... r_n$. = Eine Folge von Zufallszahlen.
n = Anzahl der Zufallszahlen.

Definition: Gap

Definiert man nun ein Intervall $[\alpha,\beta]$, dann liegen einige der Zufallszahlen in diesem Intervall, die restlichen außerhalb. Eine Folge von aufeinanderfolgenden Zufallszahlen, die alle außerhalb des Intervalls $[\alpha,\beta]$ liegen, nennt man Gap.
K = Länge des Gap.

Vorgehensweise

1) Die Anzahl der Gaps der Länge 0, 1, 2, ... wird gezählt.
2) Die theoretischen Werten werden nach der folgenden Formel berechnet:

Die Wahrscheinlichkeit, daß ein Gap der Länge K in der Folge enthalten ist, ist:

$$p(K) = (\beta - \alpha) \cdot (1 - (\beta - \alpha))^K$$

3) Die theoretischen Werten werden mit $n \cdot p(K)$ mit Hilfe des Chi-Quadrat-Tests verglichen.

5.6.7 „Runs test“

Ziel

Der serielle Test wird angewendet, um die Unabhängigkeit einer Menge von aufeinanderfolgenden Zufallszahlen zu überprüfen.

Definition: Run

Ein Run ist eine Folge von identischen Werten w (in einer Folge von Werten), wenn der Wert unmittelbar vor und nach dieser Folge $\neq$w ist.

Gegeben

$r_1, r_2, \ldots r_n$ = eine (beliebige) Folge von Zufallszahlen.
n = Anzahl der Zufallszahlen in der gesamten Folge.

Vorgehensweise

1) Die Folge von Zufallszahlen wird in eine Folge von binären Werten transformiert. Dazu wähle man eine der folgenden zwei Methoden.
   1.Methode zur Transformation
   $r_i < 0.5 \Rightarrow$ i-ter Wert in der binären Folge ist eine 0
   $r_i \geq 0.5 \Rightarrow$ i-ter Wert in der binären Folge ist eine 1

   2.Methode zur Transformation
   $r_i < r_{i+1} \Rightarrow$ i-ter Wert in der binären Folge ist eine 0
   $r_i \geq r_{i+1} \Rightarrow$ i-ter Wert in der binären Folge ist eine 1
2) Die Anzahl der Runs der verschiedenen Längen in der Folge wird gezählt.
3) Berechnung der theoretischen Werte zur jeweiligen Methode mit Hilfe der folgenden Formeln.
   Theoretische Werte zur 1.Methode:
   Die erwartete gesamte Anzahl der Runs $= \frac{n+1}{2}$
   Die erwartete Anzahl der Runs mit der Länge K $= \frac{n-K+3}{2^{K+1}}$
   Theoretische Werte zur 2.Methode:
   Die erwartete gesamte Anzahl der Runs $= \frac{2n-1}{3}$
   Die erwartete Anzahl der Runs mit der Länge K, für K < n-1 $= 2 \cdot \frac{\left(K^2+3K+1\right) \cdot n - \left(K^3+K^2-K-4\right)}{(K+3)!}$
   Die erwartete Anzahl der Runs mit der Länge K, für K = n-1 $= 2 \cdot \frac{1}{n!}$
4) Die Abweichung der einzelnen gezählten von den einzelnen theoretischen Werte wird mit der Chi-Quadrat-Methode berechnet:
$$\chi_1^2 = \sum_{i=1}^{s} \frac{(\text{beobachteter Wert} - \text{erwarteter Wert})^2}{\text{erwarteter Wert}}$$
5) Die mit der Chi-Quadrat-Methode berechneten Werte werden mit den Werten der Chi-Quadrat-Tabelle verglichen.

Übrigens deutet eine sehr kleine Anzahl von Runs auf „nicht-Zufälligkeit“ hin.

5.6.8 Andere Tests

Die anderen in Neelamkavil kurz beschriebenen Tests werden hier nur der Vollständigkeit wegen genannt:

5.6.8.1 Poker Test

5.6.8.2 Maximum Test

5.6.8.3 „Autocorrelation test“

5.7 Generierung von Zufallsvariablen

Wenn eine Pseudo-Zufallszahl Y dazu benutzt wird, um eine andere Variabel X zu generieren, dann wird X eine Zufallsvariable und X kann gleichmäßig oder ungleichmäßig verteilt sein.
Die Zufallsvariable einer gegebenen Wahrscheinlichkeitsverteilung, die den Input eines Simulationsmodells nachbilden, werden generiert, indem einer Menge von gleichmäßig verteilten Zufallszahlen die Charakteristika der gewünschten Verteilung aufgeprägt wird.

5.7.1 Auflistung einiger Zufallsverteilungen

5.7.1.1 Stetige Zufallsverteilungen

- Gleichmäßige Verteilung
- Exponentielle Verteilung
- Normalverteilung
- Empirische Verteilung

5.7.1.2 Diskrete Zufallsverteilungen

- Poisson-Verteilung
- Binomial/Bernoulli-Verteilung
- Empirische Verteilung

5.7.2 Transformation in eine beliebige Verteilungsfunktion

5.7.2.1 Inverse Transformations-Methode

Gegeben

- Eine Verteilungsfunktion oder Dichtefunktion der gewünschten Wahrscheinlichkeitsverteilung
- Eine Folge von gleichmäßig verteilten Zufallszahlen $r_1, r_2, \ldots$ aus dem Bereich (0,1).

Vorgehensweise

Falls nur die Dichtefunktion und nicht die Verteilungsfunktion gegeben ist, muß noch die Verteilungsfunktion bestimmt werden:

$$F(X) = \int_{-\infty}^{X} f(X')\,dx$$

Dann wird

$$F(X) = r$$

gesetzt. Daraus ist die inverse Transformation zu bestimmen, entweder analytisch oder notfalls numerisch oder empirisch.

$$F(X) = r$$

$$\Rightarrow \quad X = F^{-1}(r)$$

Wir erhalten somit zu jedem Zufallswert r_i der gleichmäßigen Verteilung einen Zufallswert X_i der gewünschten Verteilung:

$$\boxed{X_i = F^{-1}(r_i)}$$

5.7.2.2 Ablehnungsmethode (Rejection method)

Gegeben

$r_1, r_2, ... r_n$. = Eine Folge von Zufallszahlen.

Konzept

Wir nehmen an, die Dichtefunktion f(X) sei restlos von einem Rechteck umgeben. In der Horizontalen wird das Rechteck bei a und b begrenzt, in der Vertikalen bei 0 und h, und zwar so, daß gilt:

$a \le X \le b,$

$0 \le f(X) \le h$

$\Rightarrow 0 \le f(x)/h \le 1$

Ein mit Hilfe von zwei Zufallszahlen berechneter Punkt wird in das Rechteck gesetzt. Liegt der Punkt in der vertikalen Richtung unter oder auf der Kurve, wird die horizontale Koordinate des Punktes als Ergebnis ausgegeben.

Vorgehensweise

1) Generiere die beiden Zufallszahlen r_i und r_{i+1}.
2) Berechne die Koordinaten des Punktes:
   $X_i = a + (b-a)\cdot r_i$
3) $Y_i = h\cdot r_{i+1}$
4) Wenn gilt $Y_i \le f(X_i)$, dann gebe X_i als Ergebnis heraus,
   sonst gehe zu 2)

5.7.3 Transformation in eine stetige Verteilungsfunktion (Generierung von stetigen Zufallszahlen)

5.7.3.1 Gleichmäßige Verteilung

Für die Dichtefunktion und die Verteilungsfunktion der Gleichverteilung gilt:

$$f(X) = \begin{cases} \frac{1}{b-a} & \text{für: } a < X \le b \\ 0 & \text{sonst} \end{cases}$$

$$\Rightarrow F(X) = \frac{X-a}{b-a} = r; \quad 0 \le r < 1$$

Vorgehensweise

Damit erhält man (mit Hilfe der inversen Transformationsmethode) zu jeder Zufallszahl r_i aus dem Bereich $0 \le r < 1$ eine Zufallszahl X_i aus dem Bereich:

$a \le X_i < b$:

$$\boxed{X_i = a + r_i \cdot (b-a)}$$

1.Variation

Wenn X_i nur ganzzahlige Werte annehmen soll, dann gilt:

$$\boxed{X_i = a + \text{Trunc}(r_i \cdot (b-a+1))}$$

Trunc schneidet die Nachkommastellen ab.

2.Variation

Wenn der Zufallszahlengenerator Zahlen aus dem Intervall [0, 999] generiert:

$$\boxed{X_i = a + \text{Trunc}\left(\frac{r_i}{1000} \cdot (b-a+1)\right)}$$

Es gilt dabei: $0 \le r < 1$, $a \le X_i < b$.

5.7.3.2 Exponentielle Verteilung

Grundlagen

Die exponentielle Verteilung beschreibt einen Poisson-Prozeß.

$$f(X) = \begin{cases} \lambda \cdot e^{-\lambda \cdot X} & \text{für: } \lambda > 0 \text{ und } X \geq 0 \\ 0 & \text{sonst.} \end{cases}$$

Vorgehensweise

Benutzt wird die inverse Transformations-Methode:

$$F(X) = \int_{-\infty}^{X} f(z)\,dz$$

$$= 1 - e^{-\lambda \cdot X}.$$

$$F(X) \overset{!}{=} r$$

$$\Rightarrow X = -\frac{1}{\lambda}\log(1-r)$$

$$\Rightarrow \boxed{X = -\frac{1}{\lambda}\log r}$$

5.7.3.3 Normalverteilung

(Siehe auch: Neelamkavil, S.123-126)

Grundlagen

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

Die Dichte f ist symmetrisch zur Achse $x = \mu$. μ ist gleichzeitig der Erwartungswert, der Median und der Modalwert / das Maximum. f hat an den Stellen $\mu \pm \sigma$ Wendepunkte. Je größer die Varianz σ^2 ist, desto flacher wird die Kurve der Dichte f.

Vorgehensweise

1.Methode

1) Generiere die Zufallszahl r_i.
2) Berechne:

$$\boxed{X_i = \sqrt{\frac{\pi}{8}} \cdot \log\left(\frac{1+r_i}{1-r_i}\right)}$$

2.Methode

1) Generiere die zwei gleichmäßig verteilten Zufallszahlen r_i und r_{i+1}
2) Berechne:

$$\boxed{\begin{aligned} X_i &= \sqrt{-2\log r_i} \cdot \cos(2\pi \cdot r_{i+1}) \\ X_{i+1} &= \sqrt{-2\log r_i} \cdot \sin(2\pi \cdot r_{i+1}) \end{aligned}}$$

3.Methode

1) Substitution durch die Variable Z:

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

$$\Rightarrow f(Z) = \frac{e^{-\frac{1}{2}Z^2}}{\sigma\sqrt{2\pi}}, \quad Z = \frac{(X-\mu)}{\sigma}$$

2) Konstruiere eine Tabelle mit Werten von Z und F(Z) (von Tabellen in Stochastik-Büchern übernehmen oder approximieren)
3) Generiere die Zufallszahl r.
4) Es sei: r = F(Z).
   Lese zum Wert von r d.h. vom Wert von F(Z) den zugehörigen Wert von Z von der Tabelle ab. Eventuell ist eine Interpolation notwendig.
5) Wegen $Z = \frac{(X-\mu)}{\sigma}$ läßt sich Z wie folgt berechnen:

$$\boxed{X = Z\sigma + \mu}$$

5.7.3.4 Empirische Verteilung

Gegeben

$X_1, X_2, ... X_n$ = Eine Anzahl an Zufallsvariablen
Eine stetige Verteilungsfunktion F(X). Damit erhält man:
$Y_1 = F(X_1)$, = Eine Anzahl an zu den
$Y_2 = F(X_2)$, ... Zufallsvariablen passenden
$Y_n = F(X_n)$ „Verteilungswerten“
Falls die inverse Transformationsmethode nicht angewendet werden kann, muß wie folgt vorgegangen werden:

Vorgehensweise

1) Generiere gleichverteilte Zufallszahlen r_i, wobei gilt:
   $0 \le r_i < 1$
2) Wenn $r_i = Y_p$, dann ist das Ergebnis X_p.

Wenn $r_i \neq Y_p$ (kein „Verteilungswert“ paßt genau), dann wird das Ergebnis X_p linear interpoliert:
Für $Y_k < r_i < Y_{k+1}$ gilt

$$X_p = X_k + \frac{X_{k+1} - X_k}{Y_{k+1} - Y_k}(r_i - Y_k)$$

5.7.4 Transformation in eine diskrete Verteilungsfunktion (Generierung von diskreten Zufallszahlen)

5.7.4.1 Poisson-Verteilung

Grundlagen

$$p(X) = \frac{\lambda^X e^{-\lambda}}{X!}, \qquad X = 0, 1, 2, \ldots$$

Vorgehensweise

1) Generiere die Zufallszahl r_1.
2) Berechne X_1 mit der Formel:

$$X_1 = -\frac{1}{\lambda} \log r_1$$

3) Setze: i = 2.
4) Generiere die Zufallszahl r_i.
5) Berechne X_i mit der Formel:

$$X_i = -\frac{1}{\lambda} \log r_i$$

6) Wenn gilt

$$\sum_{j=1}^{i-1} X_j \le 1 \le \sum_{j=1}^{i} X_j ,$$

dann ist i-1 die gewünschte Poissonvariable.
Sonst: Setze i := i+1, gehe zu Schritt 4.

5.7.4.2 Binomial/Bernoulli-Verteilung

Grundlagen

p sei die Wahrscheinlichkeit (beim Münzenwerfen), erfolgreich zu sein (und q = 1-p sei die Wahrscheinlichkeit, zu verlieren).
Es werden N Experimente (Münzwürfe) durchgeführt.
Die Wahrscheinlichkeit, bei N Experimenten genau X-mal erfolgreich zu sein, ist:

$$p(X) = \binom{N}{X} \cdot p^X \cdot q^{N-X}, \qquad X = 0, 1, 2, \ldots n$$

Vorgehensweise

1.Methode

1) Generiere N gleichmäßig verteilte Zufallszahlen im Bereich (0,1).
2) Zähle die Anzahl n der Zufallszahlen, die größer sind als p.
3) Der Wert der binomialen Zufallsvariable ist dann:
   $\boxed{X = N - n}$

2.Methode

5.7.4.3 Empirische Verteilung

Gegeben

$X_1, X_2, \ldots$ = Folge von Werten einer Zufallsvariable
$p(X_1), p(X_2), \ldots$ = Wahrscheinlichkeiten, daß das Ereignis $X_1, X_2, \ldots$ eintrifft

Vorgehensweise

1) Generiere die Zufallszahl r, $0 \le r < 1$.
2) Setze: i = 2.
3) Wenn gilt
   $$\sum_{j=1}^{X_{i-1}} p(X_j) \le r \le \sum_{j=1}^{X_i} p(X_j),$$
   dann ist X_{i-1} die gewünschte Zufallsvariable.
   Sonst: Setze i := i+1, gehe zu Schritt 2.

6 Simulation Teil 3: Diskrete Simulation diskreter Systeme

Bei der „diskreten Simulation diskreter Systeme“ wird ein zeitdiskretes Simulationsmodell erstellt.

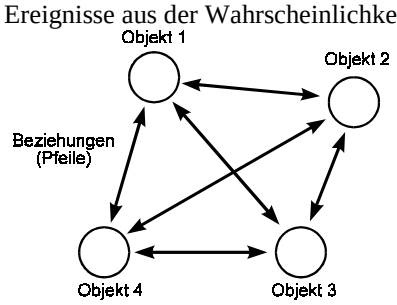
6.1 Definitionen & Grundlagen

6.1.1 Überblick

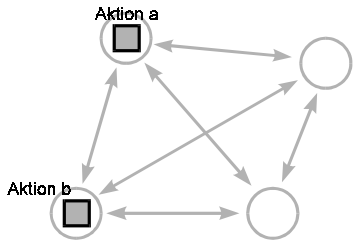
Wiederholung:
Beschreibung der statischen Struktur eines Systems
Ein System ist eine abgegrenzte Anordnung von Objekten, die sich gegenseitig beeinflussen.
Beschreibung der dynamische Struktur eines Systems
Der Zustand eines Systems wird bestimmt von:

- Objekte (Komponenten),
- Attributen (charakteristische Eigenschaften (Substantive und Adjektive)),
- Aktionen

zu einer gegebenen Zeit.
Veränderungen des Zustands eines Systems nennt man sinnvollerweise **Zustandsänderungen**.
Die Frage lautet nun: Wie läßt sich die statische und dynamische Struktur sowie die Zustandsänderungen programmiertechnisch darstellen?
Zur Darstellung der statischen und dynamischen Struktur muß sich der Programmierer passende Datenstrukturen aussuchen. Die Zustandsänderungen werden allgemein als Ereignisse dargestellt. Beachte: Hier sind nicht die Ereignisse aus der Wahrscheinlichkeitsrechnung gemeint.



Statische Struktur des Modells



Dynamische Struktur des Modells

6.1.2 Beziehung zwischen Zustand und Zeit

Bei der Modellierung eines Systems muß nicht nur ein bestimmter Zustand abgebildet werden. Es ist viel mehr darzustellen, wie sich der Systemzustand in Abhängigkeit der Zeit verändert.
Es geht also um die prinzipielle Möglichkeit, den Zusammenhang zwischen der
statischen Struktur
und dem
dynamischen Verhalten
eines Systems in einem diskreten Modell abzubilden.

6.1.3 Objekte, Attribute

Die statische Struktur eines Systems (vgl. Definition) kann man sich so vorstellen, daß sie aus einer Vielzahl von *Objekten* (Komponenten) besteht, die durch ein Netz von Beziehungen miteinander verbunden sind. Die Objekte werden von Attributen charakterisiert, die für jedes individuelle Objekt bestimmte (quantitative oder qualitative) Werte annehmen.
Die Attribute können sowohl

deskriptiv
(Beschreibung einer Eigenschaft eines isolierten Objektes)
als auch
relational
(Beschreibung der Beziehung zu anderen Objekten)
werden.
Um das dynamische Verhalten eines Systems abzubilden, müssen Attributwerte im Modell mit einem Zeitindex versehen werden. (Der Index kann übrigens als *Indexattribut* aufgefaßt werden, das allen Objekten gemeinsam ist.)
Neben der Zeit sind andere Indexattribute denkbar, beispielsweise, wenn die *räumliche* Dynamik bzw. die *Kosten*- Dynamik eines Systems durch Simulation untersucht werden soll. (Schadstoffausbreitung bzgl. Umweltschutz bzw. Wechselkursabhängigkeiten.)
Das Indexattribut Zeit wird als **Zeitbasis** eines Simulationsmodells bezeichnet.
Die diskrete, abzählbare Zeitbasis, die in zeitdiskreten Simulationsmodellen verwendet wird, bedeutet eine Einschränkung: Vorgänge, deren exakte Beschreibung eine **kontinuierliche** Zeitbasis erfordern würden, werden von der Modellierung ausgeschlossen.
Es besteht jedoch die Möglichkeit einer **Approximation** kontinuierlicher Vorgänge durch zeitdiskrete Modelle.
(Bei der Verwendung eines Digitalrechners ist dies allerdings sogar prinzipiell unumgänglich.)

6.1.4 Zustand

Der Zustand eines Objektes ist das n-Tupel aller Attributwerte dieses Objektes.
Der Zustand des gesamten Systems ist die geordnete Menge aller Objektzustände.

6.1.5 Diskrete Ereignis-Simulation

Die Beziehungen zwischen Zustand und Zeit kann in einem diskreten Simulationsmodell auf dreierlei Weise hergestellt werden:

- durch ein **Ereignis**, das die Veränderung eines Objektzustandes zu einem bestimmten Zeitpunkt, dem *Ereigniszeitpunkt*, bewirkt. (Zur Erinnerung: Ereignisse können sowohl endogener als auch exogener Art sein.)
- durch eine **Aktivität**, die aus einer Menge von Operationen besteht, die während eines Zeitintervalls ausgeführt werden. Die Wirkung einer Aktivität (also die Zustandsänderung des Objekts) wird dem Endzeitpunkt des Intervalls zugeordnet.
- durch einen **Prozeß**, der aus einer Folge von Aktivitäten besteht, die auf ein bestimmtes Objekt bezogen sind und während einer Zeitspanne ablaufen.

Bei der **diskreten Ereignis-Simulation** (bzw. ereignisorientierten Simulation) wird eine modellinterne Simulationsuhr von der Ablaufkontrolle jeweils bis zum nächts-folgenden Ereigniszeitpunkt vorgerückt.
Dies impliziert, daß *die Zeit zwischen zwei benachbarten Ereignissen im allgemeinen variable ist.* Der Simulationszeitpunkt eines Ereignisses kann beliebig gewählt werden, wodurch eine wirklichkeitsgetreue Abbildung von Zustandsänderungen möglich ist.
Dies wäre nicht der Fall, wenn man die Simulationsuhr in konstanten Zeitintervallen weiterschalten würde. Dann müßten alle Ereigniszeitpunkte auf das Ende eines solchen Intervalls verschoben werden, was bei ungünstiger Wahl der Schrittlänge erhebliche Verfälschungen des Modellverhaltens bewirken kann.
Es ist nicht auszuschließen, daß Ereignisse simultan auftreten, also in einem Ereigniszeitpunkt zusammenfallen. Ein sequentiell arbeitender Rechner ist jedoch nur in der Lage, die Ereignisse (Zustandsänderungen) nacheinander abzuarbeiten: Das erzwingt eine Sequentialisierung paralleler Ereignisse. (Prioritätenfestlegung).

6.2 Sichtweisen der diskreten Simulation

6.2.1 Vergleich der Sichtweisen/Simulationsansätze bei der diskreten Simulation

	Ereignisorientierte S.	Prozeßorientierte S.	Transaktionsorient. S.	Aktionsorientierte Simulation
	Vorgehensweise			
1)	Die Ablaufkontrolle holt das Ereignis mit dem frühesten Ereigniszeitpunkt aus der Liste (Ereignisliste). (Haben mehrere Ereignisse den gleichen, frühesten Ereigniszeitpunkt, muß sequenzialisiert werden.)	Die Ablaufkontrolle holt den Prozeß mit dem frühesten „Aktivierungs-zeitpunkt“ aus der Liste. (Haben mehrere Prozesse den gleichen, frühesten Aktivierungszeitpunkt, muß sequenzialisiert werden.)	Die Ablaufkontrolle holt die Transaktion mit dem frühesten „Aktivierungs-zeitpunkt“ aus der Liste. (Haben mehrere Transaktionen den gleichen, frühesten Aktivierungszeitpunkt, muß sequenzialisiert werden.)	Es gibt keine Liste, dafür aber eine feste Anzahl von Aktivitäten. Die Ablaufkontrolle prüft die Zustände aller Aktivitäten. Diejenigen Aktivitäten, welche die Bedingungen erfüllen, werden alle unverzüglich ausgeführt.
2)	Die Simulationsuhr wird auf den Zeitpunkt des Ereignisses vorgerückt.	Die Simulationsuhr wird auf den Zeitpunkt des Prozesses oder auf dessen reaktivierten Teil vorgerückt.	Die Simulationsuhr wird auf den Zeitpunkt der Transaktion vorgerückt.	Die Simulationsuhr wird auf den Zeitpunkt der Aktion vorgerückt.
3)	Die Ereignisroutine führt die von dem Ereignis dargestellten Tätigkeiten aus (Zustandsänderungen).	Das aktuelle Teilstück des Prozesses wird abgearbeitet. Eventuell wird der Prozeß unterbrochen, bevor er beendet ist.	Die Transaktion wird zum passenden Block geleitet. Im Block können dann die Parameter der Transformation verändert werden.	Die Operationen der Aktivitäten werden ausgeführt.
4)	Gehe zu Schritt 1).	Gehe zu Schritt 1).	Gehe zu Schritt 1).	Gehe zu Schritt 1).
	Bemerkungen			
	Zustandsänderungen werden nachgebildet durch: Ereignisse.	Zustandsänderungen werden nachgebildet durch: Prozesse. (Prozeß = Die auf ein bestimmtes Objekt bezogenen Attribute und Aktivitäten.) Prozesse müssen nicht vollständig abgearbeitet werden. Sie können unterbrochen (deaktiviert) und wieder reaktiviert werden. Das aktuelle Prozeßteilstück ≈ Ereignis	Zustandsänderungen werden ausgelöst durch: Transaktionen in Blöcken. Wichtig ist die Aufteilung in statische System-Anteile (Blöcke) und dynamische System-Anteile (Transaktionen). Das Aufeinandertreffen einer Transaktion auf einen Block ≈ Ereignis	Zustandsänderungen werden nachgebildet durch: Aktivitäten. Aktivität = Der Zustand einer Komponente über einem Intervall. Eine Aktivität ist begrenzt durch zwei aufeinanderfolgende Ereignisse, die nicht unbedingt zur gleichen Komponenten gehören müssen. Eine Aktivität besteht aus einer Menge von Operationen. Mit jeder Aktivität ist eine Bedingung verbunden. Um zu bestimmen, ob die Bedingung erfüllt ist, wird der System-Zustand und die Systemzeit betrachtet. Mit dem Abarbeiten der Aktivitäten werden die Ereignisse gleich mit-abgearbeitet.

6.2.2 Entscheidung für eine Sichtweise

Beziehung zwischen Systemzustand und Simulationszeit

Man muß sich für eine von folgenden Sichtweisen entscheiden:

- Ereignisorientiert
- Prozeßorientiert
- Transaktionsorientiert
- Aktivitätsorientiert

6.2.3 Ereignisorientierte Sicht (Teil 1)

(ereignisorientierten Simulation)

6.2.3.1 Prinzip

Dieser Ansatz beruht auf einer detaillierten Beschreibung von Ereignissen in Form von **Ereignisroutinen**.
In einem ereignisorientierten Modell werden die **Zustandsänderungen** nachgebildet, die zu den Ereigniszeitpunkten stattfinden, nicht jedoch die Tätigkeiten, die in den dazwischenliegenden Intervallen ablaufen.
Zeitlich ausgedehnte Vorgänge werden somit auf eine Folge von Ereignissen reduziert, von denen jedes einzelne einem Punkt auf der Zeitachse (dem Ereigniszeitpunkt) zugeordnet ist. Unabhängig davon wieviel Rechenzeit zur Ausführung einer Ereignisroutine auf einem Rechner real verbraucht wird, geschieht sie konzeptuell **zeit-verzugslos**.
Die interne Simulationsuhr wird vor der Ausführung der Ereignisroutine zum Ereignis-Zeitpunkt vorgestellt, **steht** während der Ausführung **still** und springt nach vollzogener Zustandsänderung auf den Zeitpunkt des nächsten Ereignisses vor.
Die ereignisorientierte Sicht erlaubt eine klare Trennung zwischen der Struktur und Verhalten des zu simulierenden Systems.

6.2.3.2 Aktive / passive Phase

In der aktiven Phase

Die aktiven Phasen liegen in den Ereigniszeitpunkten, d.h. während der Ausführung einer Ereignisroutine.
Die **Zustandsänderungen** werden nachgebildet, die zu den Ereigniszeitpunkten stattfinden.
Die interne Simulationsuhr steht während der Ausführung der Ereignisroutine still.

In der passiven Phase

Die passiven Phasen liegen jeweils z**wischen zwei** Ereigniszeitpunkten, d.h. nach der Ausführung einer Ereignisroutine.
Die modellinterne **Simulationsuhr** wird von der **Ablaufkontrolle** jeweils bis zum nächst folgenden Ereigniszeitpunkt vorgerückt.
Zeitlich ausgedehnte Vorgänge werden somit auf einer **Folge von Ereignissen** reduziert.

6.2.3.3 Wahl des Simulationszeitpunktes eines Ereignisses

Die Zeit zwischen zwei benachbarten Ereignissen ist im allgemeinen variabel.
Der Simulationszeitpunkt eines Ereignisses kann beliebig gewählt werden.
Bei simultanen Ereignissen muß eine **Sequentialisierung** dieser parallelen Ereignisse erfolgen (Prioritätenfestlegung).

6.2.3.4 Funktion der Ereignisroutinen

Die Ereignisroutinen bestimmen in ihrem Zusammenspiel das Verhalten des Modells, indem sie

- Attribut-Werte von Objekten ändern
- temporäre Objekte generieren oder löschen

- einer Ereignisliste neue, geplante Ereignisse hinzufügen oder aus der Liste streichen.

6.2.3.5 Statische und dynamische Komponente

Die ereignisorientierte Sicht erlaubt eine klare Trennung zwischen der Struktur und dem Verhalten des zu simulierenden Systems.
Im Modell werden unterschieden:

- Statische Komponenten (Datenstrukturen repräsentieren die Objekte mit Attribute)
- Dynamische Komponente (Ereignisroutinen repräsentieren die Ereignisse, sowie durch irgend etwas (?) repräsentierte temporäre Objekte)

6.2.3.6 Ablaufkontrolle

Die Ablaufkontrolle erfolgt durch einen Programmteil, der die in einer Ereignisliste nach ihren Ereigniszeitpunkten geordneten Ereignisse sequentiell abarbeitet (nächstes Ereignis - Strategie).
Bei diesem Ansatz wird die sogenannte **tote Zeit**, also die Zeit zwischen den Ereignispunkten, übersprungen.

6.2.3.7 Vorteile

Bei der ereignisorientierte Simulation von Vorgängen, die im wesentlichen aus dem Belegen und Freigeben von Bedienstationen bestehen, erhält man folgende Vorteile:

- Die Ablaufsteuerung ist wesentlich einfacher zu realisieren und
- die Programme laufen meist schneller

als bei prozeß- oder transaktionsorientierten Modellen.

6.2.3.8 Anwendung

Die ereignisorientierte Formulierung eines Modells ist besonders sinnvoll, wenn die zu simulierenden Vorgänge im wesentlichen aus dem **Belegen** und **Freigeben** von Bedienstationen bestehen, ohne daß die Modellkomponenten komplex interagieren.

6.2.4 Prozeßorientierte Sicht

6.2.4.1 Prozeß

Im prozeßorientierten Weltbild werden die auf ein Objekt bezogenen Aktivitäten mit den Objektattributen in ihrer Gesamtheit zu einem **Prozeß** zusammengefaßt.

6.2.4.2 Aktive / passive Phase

In der aktiven Phase

In den aktiven Prozeßphasen werden die Ereignisse nachgebildet.
Während seiner aktiven Phasen führt ein Prozeß Zustandsänderungen durch.
Diese Zustandsänderungen laufen zeitverzuglos ab, d.h. die Simulationsuhr steht während der aktiven Phasen still (vgl. ereignisorientierte Sicht).
Das dabei ausgeführte Teilstück der Prozeßroutine ist in dieser Hinsicht mit einer Ereignisroutine vergleichbar.
Während der aktiven Phasen steht die Simulationsuhr still.

In der passiven Phase

In den passiven Prozeßphasen wird alles nachgebildet, was nicht Ereignis ist.
Damit sind die (zeitkonsumierenden) Aktivitäten einer Systemkomponente und ihre passiven Phasen gemeint.
Ein Prozeß hat nach Durchführung der Zustandsänderungen - im Gegensatz zu einer Ereignisroutine, die dann stets terminiert - jedoch die Möglichkeit, in einen inaktiven Zustand einzutreten. Die Ausführung des Prozesses kann dann zu einem späteren Zeitpunkt fortgesetzt, d.h. der Prozeß kann **reaktiviert** werden.
Nach der Reaktivierung wird das nächste Teilstück der Prozeßroutine abgearbeitet.
Weil in aktiven Prozeßphasen keine Simulationszeit verbraucht wird, dienen sie zur Darstellung von Ereignissen.
Die (zeitkonsumierenden) Aktivitäten einer Systemkomponente und ihre passiven Phasen werden dagegen durch *inaktive* Prozeßzustände abgebildet.
Ein Prozeß muß den Zeitpunkt seiner Reaktivierung nicht notwendigerweise selbst bestimmen.
Der häufigere Fall besteht vielmehr darin, daß ein Prozeß sich selbst passiv macht und später durch andere Prozesse zur Reaktivierung vorgemerkt wird.

6.2.4.3 Reaktivierung eines Prozesses

Ein Prozeß hat nach Durchführung der Zustandsänderung die Möglichkeit, in einen inaktiven Zustand einzutreten.
Die Ausführung des Prozesses kann dann zu einem späteren Zeitpunkt fortgesetzt , d.h. der Prozeß kann reaktiviert werden. Nach der Reaktivierung wird das nächste Teilstück der Prozeßroutine abgearbeitet.
Ein Prozeß muß den Zeitpunkt seiner Reaktivierung nicht notwendigerweise selbst bestimmen.
Der häufigere Fall besteht vielmehr darin, daß ein Prozeß sich selbst passiviert und später durch andere Prozesse zur Reaktivierung vorgemerkt wird.

6.2.4.4 Beispiel

Um beispielsweise die Bearbeitung eines Werkstücks durch eine Maschine während eines Zeitintervalls Δt darzustellen, wird man einen Maschinenprozeß definieren, der seine Reaktivierung nach Ablauf von Δt Zeiteinheiten vormerkt und sich gleichzeitig deaktiviert. Diese Operation entspricht dem **Ereignis** "*Beginn der Bearbeitung*".
Die Ablaufsteuerung sorgt dann für die Reaktivierung des Prozesses zum vorgemerkten Zeitpunkt (sofern kein anderer Prozeß diesen verschoben oder gelöscht hat). Dies entspricht dem **Ereignis** "*Ende der Bearbeitung*".
Eine Aktivität fängt mit einem Ereignis an und hört mit einem Ereignis auf. In der Phase zwischen den Ereignissen „Beginn der Bearbeitung“ und „Ende der Bearbeitung“ ist der Prozeß (programmtechnisch) nicht aktiv. Dieser Phase entspricht die **Aktivität** "*Bearbeiten*".

6.2.4.5 Funktion der Prozesse

Ein Prozeß kann während seiner aktiven Phase:

- Objektattribute modifizieren
- neue Prozesse generieren
- Prozesse beenden
- sich selbst deaktivieren, wobei die Kontrolle an einen anderen Prozeß übergeht
- die Aktivierung anderer Prozesse zu bestimmten Zeitpunkten planen oder geplanten Aktivierungen verschieben oder löschen

6.2.4.6 Ablaufkontrolle

Die Ablaufkontrolle sorgt dafür, daß Prozesse in der richtigen Reihenfolge und in den vorgesehenen Zeitabständen ablaufen.
Die zu aktivierenden Prozesse werden in einer nach Zeitpunkten geordneten Liste notiert. Eine Hauptroutine übernimmt die Abarbeitung dieser Liste. Dabei wählt sie den jeweils nächsten Prozeß aus. Im Gegensatz zu einer Ereignisroutine wird der Prozeß jedoch nicht notwendigerweise von seinem Anfang abgearbeitet. Falls er bereits einmal aktiv war, wird seine Abarbeitung an der Stelle fortgesetzt, an der er sich selbst deaktiviert hat. Es ist also notwendig, den Status eines Prozesses zum Zeitpunkt seiner Deaktivierung aufzubewahren. Dazu gehören sowohl die Werte seiner Attribute als auch die Information, an welcher Stelle er fortzusetzen ist.
Die zu aktivierenden Prozesse werden in einer nach Zeitpunkten geordneten **Ereignisliste** notiert.

6.2.4.7 Vergleich: Prozeßorientierte und ereignisorientierte Sicht

Faßt man die (Re-)Aktivierung eines Prozesses als Ereignis auf, so läßt sich die Ablauf-Steuerung ähnlich wie im ereignisorientierten Weltbild beschreiben:

6.2.4.8 Vorteil

- Sowohl die Aktiven als auch die passiven Phasen einer Systemkomponente können relativ wirklichkeitsgetreu abgebildet werden.
- Es entfällt die Auflösung logisch zusammengehöriger Abläufe auf verschiedene Ereignisroutinen.

6.2.4.9 Nachteil

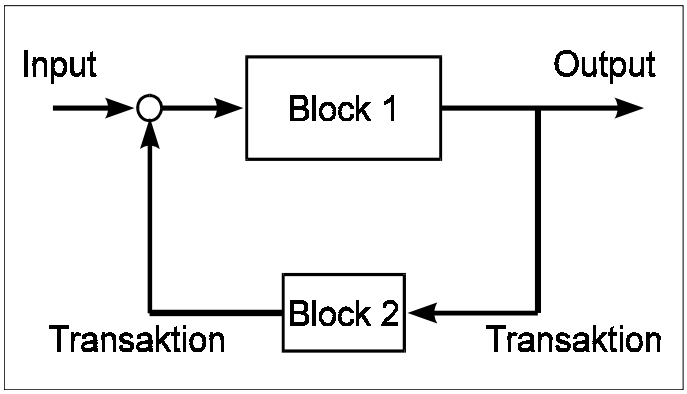
Eine gewisse Unübersichtlichkeit der Systemstruktur.

6.2.4.10 Anwendung

Der prozeßorientierte Ansatz ist sinnvoll bei der Modellierung nebenläufiger Vorgänge mit komplexen Interaktionen.

6.2.5 Transaktionsorientierte Sicht

[activity scanning]
Grundlage dieses Konzepts ist die (aus der Systemanalyse und Reglungstechnik bekannte) **Blockdiagrammtechnik** zur Beschreibung des Systemverhaltens.
Die wesentlichen Elemente sind neben den **Blöcken** (mit fest vorgegebenen Funktionen) die **Transaktionen**, die auf ihrem Weg durch die einzelnen Blöcke verändert werden.



6.2.5.1 Statische / dynamische Systemkomponente

- Die statischen Systemkomponenten sind die **Blöcke**, da sie permanent vorhanden sind. Dazu gehören z.B.:
  - Bedienstationen unterschiedlichster Art
  - Speicher (die Lager mit vorgegebener Kapazität darstellen)
  - Leitstellen (logische Schaltstellen, die einzelne Transaktionen gemäß bestimmter Abhängigkeiten steuern)
- Die dynamischen Systemkomponenten (temporären Elemente) sind die **Transaktionen**. Die Transaktionen werden durch eine Anzahl von Parametern (indikative Attribute) charakterisiert, die auf dem Weg durch das modellierte System in den einzelnen Blöcken verändert werden können. Somit wird eine Zustandsänderung durch eine Transaktion in einem Block ausgelöst (vergleiche mit der objektorientierten Programmierung).
  Sie kann darin bestehen, daß Bearbeitungsstationen belegt oder freigegeben werden, sich Warteschlangen vergrößern oder Speicherinhalte verändern.

6.2.5.2 Ablaufkontrolle

Die Ablaufkontrolle läßt sich wiederum am besten durch Rückgriff auf den ereignisorientierten Ansatz verdeutlichen:
Das Aufeinandertreffen von Transaktion und Block kann als Ereignis betrachtet werden. Die Transaktionen sind in der Ereignisliste nach dem Zeitpunkt ihres Eintritts in den nächsten Block und nach Prioritäten geordnet.

6.2.6 Ereignisorientierte Sicht (Teil 2)

6.2.6.1 Komponenten ereignisorientierter Simulationsmodelle

(Der ereignisorientierte Ansatz kann als Grundlage für das Verständnis anderer Ansätze dienen)
Die Grundkomponenten ereignisorientierter Simulationsmodelle sind:

Zustands-variablen:	Die Menge aller Variablen zur Beschreibung des Systemzustandes zu einem bestimmten Zeitpunkt.
Simulationsuhr:	Eine Variable, die den aktuellen Stand der Simulationszeit angibt.
Ereignisliste:	Eine Liste mit Zeitpunkt und Typ der geplanten Ereignisse, nach Ereigniszeitpunkten aufsteigend geordnet.
Statistische Zähler:	Variablen zur Sammlung der statistischen Ergebnisse des Simulationsablaufs.
Initialisierungs-routine:	Eine Prozedur zur Initialisierung des Simulationsmodells zum Zeitpunkt 0.
Zeitführungs-routine:	Eine Prozedur zur Selektion des nächsten Ergebnisse aus der Ereignisliste und Vorstellen der Simulationsuhr auf den nächsten Ereigniszeitpunkt.
Ereignis-routinen:	Prozeduren zur Aktualisierung des Modellzustandes in Abhängigkeit vom jeweiligen Ereignistyp. (Für jeden Ereignistyp existiert eine Ereignisroutine).
Ergebnis-routinen:	Eine Prozedur zur Berechnung der statistischen Schätzwerte der Ergebnisvariablen (anhand der statistischen Zähler) und zur Ausgabe des Ergebnisprotokolls am Ende des Simulationsablaufs.
Steuer-programm:	Ein Programmteil, der wiederholt die Zeitführungsroutine für die Bestimmung des nächsten Ereignistyps aufruft und die zugehörige Ereignisroutine aktiviert, bis der Simulationslauf beendet ist.

6.2.6.2 Ausführung einer Ereignisroutine

In einer Ereignisroutine finden folgende Aktionen statt:

- Aktualisierung des Systemzustandes gemäß den Erfordernissen des entsprechenden Ereignistyps
- Sammeln von Informationen (Aktualisierung der statistischen Zähler)
- Erzeugen neuer Ereignisse, Festsetzung ihres Ereigniszeitpunktes und Eintragen in die Ereignisliste, gegebenenfalls Löschen von Ereignissen aus der Liste.

Das Eintragen eines neuen Ereignisses in die Ereignisliste wird auch als „Vormerken“ oder „Ansetzen“ des Ereignisses bezeichnet. Auf Programmebene wird hierfür eine **Ereignisnotiz** erzeugt, die den Typ und Zeitpunkt des Ereignisses verzeichnet.

Die Ereignisnotiz wird in die **Ereignisliste** eingetragen. Sie stellt ein geplantes Ereignis dar, das erst dann aktuell stattfindet, wenn die Ereignisnotiz an die vorderste Stelle der Liste gelangt ist und die zugehörige Ereignisroutine aktiviert wird.
Wenn eine Ereignisroutine ein neues Ereignis erzeugt und vormerkt, so sind zwei Fälle zu unterscheiden:
Erstens kann es sich um das nächste Ereignis des gleichen Typs handeln, ohne daß die beiden Ereignisse auf der konzeptuellen Ebene in einer Kausalbeziehung stehen. Dies gilt typischerweise für eine Folge von *exogenen* Ankunftsereignissen (z.B. die Ankunft von Kunden oder Aufträgen im betrachteten System). Obwohl diese Ereignisse konzeptuell voneinander unabhängig sind, ist es praktisch, ihre Abfolge im Programm so darzustellen, daß jedes Ereignis dieses Typs seinen Nachfolger erzeugt und vormerkt. Häufig sind nämlich statistische Kennwerte der Zwischenankunftszeiten solcher Ereignisse bekannt.
Zweitens kann es sich um ein Ereignis handeln, das auf der konzeptuellen Ebene vom aktuellen Ereignis kausal abhängig ist. Beispielsweise kann der Beginn der Bedienung eines Kunden als kausale Folge seines Eintretens in einen Laden betrachtet werden.
Das Vormerken eines neuen Ereignisses als Operation auf Programmebene besitzt also zwei unterschiedliche Interpretationen auf der konzeptuellen Ebene.

6.2.7 Weitere Sichtweisen

6.2.7.1 Materialorientierte Sicht

Bei der materialorientierten Sicht werden bei der Modellierung primär die **Materialflüsse** durch Bedienstationen und die dabei **zurückgelegten Wege** betrachtet.
Beispiel: Ein materialorientiertes Modell des Kundenflusses durch ein Einkaufszentrum würde beispielsweise das *Aufsuchen* bestimmter Verkaufsstände als Bedienstationen aus der Perspektive der Kunde darstellen.

6.2.7.2 Maschinenorientierte Sicht

Bei der maschinenorientierten Sicht steht der **Bearbeitungsvorgang** selbst im Vordergrund.
Beispiel: Bezüglich des Einkaufszentrums würde primär das *Geschehen* an den einzelnen Verkaufsständen modelliert

6.3 Diskrete Simulation von Bedienungs-/Wartesystemen

6.3.1 Typische Modellklassen

Es gibt drei Modellklassen (?), die zu jeweils einer Problemstellung, das heißt zu einem Systemtyp, passen:

Problemstellung	Modellklasse
Bedienungs- / Wartesysteme	Untersuchung des Warteverhaltens von Aufträgen oder Kunden in einem System von Bearbeitungs- oder Bedienstationen.
Lagerhaltungs-systeme	Abschätzung von Lagerhaltungskosten bei unterschiedlichen Bestellstrategien.
Ausfallanfällige Systeme	Analyse der Einsatzbereitschaft eines Systems, das wegen auftretender Fehler oder zu Wartungszwecken außer Betrieb genommen wird.

Die Bedienungs- / Wartemodelle stellen eine der wichtigsten Modellklassen in der diskreten Simulation dar. Sie beschreiben Systeme, in denen Kunden oder Aufträge, die bestimmte Bedienungsanforderungen stellen, von Bedienstationen mit einer Anzahl von Bedieneinheiten abgefertigt werden.

6.3.2 Definition: Bedienungs-/Wartesystemen

Sie beschreiben Systeme, in denen Agenten (Kunden oder Aufträge) von Bedienstationen mit einer Anzahl von Bedieneinheiten abgefertigt werden.
Die Agenten stellen bestimmte Bedienungsanforderungen.
Mit Modellen von Bedienungs-/Wartesystemen werden primär das Warteverhalten von Aufträgen und die Auslastung von Bedienstationen untersucht.

6.3.3 Bedienstationen, Bedieneinheiten, Agenten

Es folgt die beispielhafte Auflistung einiger Bedienstationen mit passenden Bedieneinheiten und Agenten (das heißt Aufträge und Kunden).

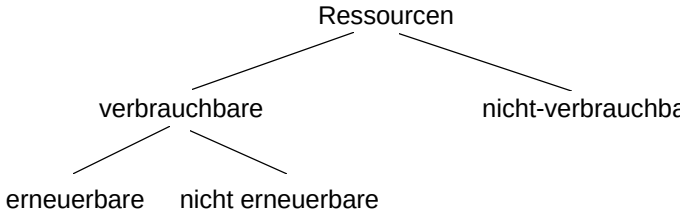
Bedienstation	Bedieneinheit(en)	Agent (Auftrag, Kunde)
Verkaufsstand	Verkäuferin	Käufer
Rechnersystem	Drucker	Druckauftrag
Fertigungssystem	Maschinengruppe	Produktionsobjekt
Krankenhaus	Röntgenlabor	Röntgenoskopieauftrag
SB-Tankstelle	Dieselsäule	Pkw-Fahrer
Wüste	Oase	Kamele

6.3.4 Ressourcen

Sowohl Bedienstationen als auch Bedieneinheiten werden gelegentlich mit **Ressourcen** (Hilfsmitteln) bezeichnet.

Übrigens:

Resource ist ein englisches Wort.
Ressource ist ein Duden-adoptiertes französisches Wort (allerdings mit englischer Aussprache).



6.3.5 Zielkriterien

Mit Bedienungs- / Wartemodellen werden primär das Warteverhalten von Aufträgen und die Auslastung von Bedienstationen untersucht.
Gesucht ist die günstigste Bedienstrategie unter den zum Teil konkurrierenden Zielkriterien:

- minimale Anzahl von Bedienstationen
- minimale Wartezeiten
- maximale Auslastung der Bedienstationen

6.3.6 Zwei Ansätze zur Untersuchung von Bedienungs-/Wartesystemen

6.3.6.1 Analytische Lösungen

Für bestimmte Standardmodelle von Bedienungssystemen existieren analytische Lösungen.

6.3.6.2 Zeitdiskrete Simulation

Ist keine analytische Lösung vorhanden, dann wird eine angenäherte Lösung durch stochastische, zeitdiskrete Simulation von Bedienstationen ermittelt.

6.3.7 Warteschlangen

Sollte eine Anforderung zum gegeben Zeitpunkt nicht erfüllbar sein, werden die Kunden bzw. Aufträge in **Warteschlangen** vor den Bedienstationen eingereiht.

6.3.8 Ressourcen

Sowohl Bedienstationen als auch Bedieneinheiten werden gelegentlich mit Ressourcen (Hilfsmittel) bezeichnet.

6.3.9 Systemstruktur

Ein System läßt sich beschreiben durch seine

- Objekte
- Attribute (Eigenschaften der Objekte)
- Beziehungen (zwischen den Objekten) (Relationen)
- Aktivitäten

6.3.9.1 Objekte

Bedienungs-/Wartemodelle bestehen aus Objekten. Diese Objekte sind nachfragende **Agenten** (Kunden, Aufträge) und nicht verbrauchbare **Ressourcen** (Verkaufsstände, Praxisräume, Betriebsmittel, Maschinengruppen und so weiter, also allgemein *Bedienstationen*).

6.3.9.2 Attribute (Merkmale, Eigenschaften)

Objekte sind jeweils durch eine Menge von Attributen charakterisiert.

6.3.9.2.1 Merkmale/Attribute der Agenten

- Ankunftsverteilung
- Bedienungsbedarf
- Präferenzen für eine bestimmte Bedieneinheit
- Ressourcenbedarf
- Wartebereitschaft
- Priorität
- Bedienungsfolge

6.3.9.2.1.1 Ankunftsverteilung

Die Ankunftsverteilung eines Kunden- Bzw. Auftragstyps legt die Zeitpunkte fest, zu denen Objekte dieses Typs in das betrachtet System eintreten.
Bei einer *deterministischen* Ankunftsverteilung wird im einfachsten Fall eine Zwischenankunftszeit fest vorgegeben.

Bei einer *stochastischen* Ankunftsverteilung sind die Zwischenankunftszeiten als eine Folge stochastischer Zufallsvariablen realisiert. (Häufig: Exponential- oder Erlangverteilung)
Die Ankunftsverteilung eines Kunden- bzw. Auftragstyps legt die Zeitpunkte fest, zu denen Objekte diese Typs in das betrachtete System eintreten.
Bei einer deterministischen Ankunftsverteilung wird im einfachsten Fall eine Zwischen-Ankunftszeit d.h. ein Zeitintervall zwischen den Ankunftszeitpunkten zweier aufeinanderfolgender Kunden bzw. Aufträge, fest vorgegeben.
Im Falle einer stochastischen Ankunftsverteilung sind die Zwischenankunftszeiten als eine Folge stochastischer Zufallsvariablen, die dem gleichen statistischen Verteilungs-Gesetz folgen, realisiert. Häufig wird hierfür die Exponential- oder die Erlang-verteilung gewählt.
Weitere Eigenschaften der Ankunftsverteilung sind Anfangs- und Endzeitpunkt oder die maximale Zahl von Ankünften.
Eine Alternative zur Modellierung der Ankunftsverteilung als Kundenattribut besteht darin, sie als Attribut eines eigenständigen Objekts Quelle anzusehen, das gemäß der angegebenen Verteilung Kunden bzw. Aufträge eines bestimmten Typs generiert.

6.3.9.2.1.2 Bedienungsbedarf

Der Bedienungsbedarf ist die Art der Anforderungen von Aufträgen an die Bedienstationen.
Die Aufträge können sich durch ihren Bedienungsbedarf unterscheiden.
Der Bedienungsbedarf kann dargestellt werden durch: 1) Attribute und 2) die Definition verschiedener Typen von Objekten für Agenten mit unterschiedlichen Anforderungen.
Die Aufträge können sich durch ihren Bedienungsbedarf (die Art ihrer Anforderungen an die Bedienstationen) unterscheiden. (Die Aufgaben in einem Rechnersystem haben beispielsweise unterschiedliche Betriebsmittel-anforderungen.) Der Bedienungsbedarf kann durch Attribute dargestellt werden. Eine andere Möglichkeit besteht darin, für Kunden oder Aufträge mit unterschiedlichen Anforderungen verschiedene Typen von Objekten zu definieren.

6.3.9.2.1.3 Präferenzen für eine bestimmte Bedieneinheit

Ein Kunde hat Präferenzen für bestimmte Bedieneinheiten einer Bedienstation, wenn seine Anforderungen nur von diesen Bedieneinheiten erfüllt werden können.
Im Modell müssen Bedieneinheiten mit individuellen Attributen und (positiven sowie negativen) Präferenzen darstellbar sein.
Es ist möglich, daß die Anforderungen von Kunden nur von bestimmten Bedieneinheiten einer Bedienstation (z.B. Mechaniker einer Werkstatt) erfüllt werden können - sei es, daß nur diese eine bestimmte Leistung erbringen können (z.B. einen Wagentyp besonders gut kennen), sei es, daß die Kunden von sich aus bestimmte Präferenzen haben. Die Präferenzen können sich auf mehrere Bedieneinheiten beziehen (Herr Hagel *oder* Herr Lück). Es können auch negative Präferenzen bestehen ("auf keinen Fall ein Lehrling!"). Also im Modell müssen Bedieneinheiten mit individuellen Attributen und positive sowie negative Präferenzen darstellbar sein.

6.3.9.2.1.4 Ressourcenbedarf

Auf den Ressourcenbedarf ist zu achten, wenn mehrere Kunden für ihre Bedienung bestimmte Kombinationen von Bedienstationen gleichzeitig belegen müssen. Dies kann zu Verklemmungs-Problemen führen, was die Modellierung erschwert.
Ein weitere Fall, der die Modellierung von Anforderungen erschwert, ist gegeben, wenn (mehrere) Kunden für ihre (nebenläufige) Bedienung bestimmte Kombinationen von Bedienstationen gleichzeitig belegen müssen. Es können Verklemmungsprobleme auftreten. Sowohl im Realsystem als auch im Modell muß hierfür eine Verklemmungs-freie Lösungsstrategie vorgesehen werden.

6.3.9.2.1.5 Wartebereitschaft

Es kann vorkommen, daß Kunden, die auf eine belegte Bedienstation stoßen, nicht bereit sind, länger als eine bestimmte Frist auf ihre Bedienung zu warten. Derartige Verluste von Agenden sind zu berücksichtigen.
Es kann vorkommen, daß Kunden, die auf eine belegte Bedienstation stoßen, nicht bereit sind, länger als einer bestimmten Frist auf ihre Bedienung zu warten. Sie verlassen also das System (bzw. die Warteschlange zur jeweiligen Bedienstation) ggf. nach einer bestimmten Wartezeit. Derartige Verluste von Kunden oder Aufträgen müssen in der Modellstruktur und der Beurteilung des Modellverhaltens berücksichtigt werden.

6.3.9.2.1.6 Priorität

Aufträge können verschieden hohe Prioritäten haben.
Bestimmte Aufträge können in einem System mit höherer Priorität behandelt werden als andere (z.B. ein eiliger Auftrag in einer Werkstatt).
Hier ist zu unterscheiden, ob die Prioritäten ausschließlich bei der Verwaltung der Warteschlangen berücksichtigt werden (die Wartenden werden nach Prioritäten sortiert), oder ob höher priorisierte Aufträge die Bearbeitung anderer Aufträge sogar unterbrechen und sie damit aus der Bedieneinheit verdrängen können.

6.3.9.2.1.7 Bedienungsfolge

Auf die Bedienungsfolge ist zu achten, wenn ein Auftrag mehrere Bedienstationen in Folge benötigt.
Von der linearen Bedienungsfolge gibt es drei mögliche Abweichungen:
1) Die beachteten Aufträge haben alle den gleichen Auftragstyp. Die Aufträge absolvieren jeweils einen von mehreren alternativen Abschnitten der Bedienungsfolge.
2) Die beachteten Aufträge haben verschiedene Auftragstypen. Die Aufträge absolvieren einen gemeinsamen Abschnitt der Bedienungsfolge. Anschließend trennen sich die Aufträge wieder.
3) Aufträge werden in Teilaufträge zerlegt, deren Ergebnisse dann an anderer Stelle wieder zusammengeführt werden müssen.

Häufig benötigt ein Auftrag nicht nur eine Bedienstation, sondern mehrere in Folge, die den Auftragsweg von der Ankunft im System bis zu seinem Austritt beschreibt.

Die Bedienungsfolge besagt erstens, welche Bedienstationen des Systems der Auftrag durchläuft, und zweitens, wie diese zeitlich geordnet sind.

Von der linearen Bedienungsfolge gibt es drei mögliche Abweichungen.
Im ersten Fall sind für einen Auftragstyp mehrere Alternative Abschnitte von Bedienungsfolgen vorgesehen. Die Entscheidung darüber, welche der möglichen Bedienungsfolgen für einen individuellen Auftrag gewählt wird, kann entweder zufällig, aufgrund bestimmter Auftragsattribute oder abhängig von Systemzustand (z.B. Länge der Warteschlangen vor den Bedienstationen) erfolgen.
Die zweite Möglichkeit liegt in der Verbindung von Bedienungsfolgen von Aufträgen verschieden Typs. Die Aufträge absolvieren einen gemeinsamen Abschnitt ihrer Bedienungsfolge und trennen sich dann wieder. Von jedem der beteiligten Auftragstypen kann eine bestimmte Anzahl von Aufträgen erforderlich sein, damit alle Aufträge gemeinsam den Weg fortsetzen können. In diesem Fall besteht zwischen den Aufträgen eine Synchronisationsbeziehung.
Ein dritter Fall der Abweichung von linearen Bedienungsfolgen ist gegeben, wenn Aufträge in Teilaufträge zerlegt werden, deren Ergebnisse dann an anderer Stelle wieder zusammengeführt werden müssen. Auch hier bestehen Synchronisationsbeziehungen.

6.3.9.2.2 Merkmale der Bedienstationen

- Anzahl und Leistungsfähigkeit der (verfügbaren) Bedieneinheiten
- Bedienzeitverhalten
- Warteraum

- Bedienstrategie (Wartenschlangen-Abarbeitungsstrategie) und Bedienungsunterbrechung

Anzahl und Leistungsfähigkeit

Dem Bedienungsbedarf, den Präferenzen für bestimmten Bedieneinheiten und dem Ressourcenbedarf auf Auftragsseite stehen Anzahl und Leistungsfähigkeit der Bedieneinheiten gegenüber, die eine Bedienstation aufweist.

Bedienzeitverhalten

Wie die Zwischenankunftszeiten können auch die Bedienzeiten deterministisch oder stochastisch verteilt sein.
Es ist zu berücksichtigen, daß bei Bedienungsunterbrechungen evtl. ein Verwaltungs-Mehraufwand hinzugerechnet werden muß.

Warteraum

Es ist zu unterscheiden zwischen Systemen:
– mit Warteraum/Warteschlange -
ohne Warteraum/Warteschlange (Verlustsystem)
– mit Warteräumen mit endlicher Kapazität -
mit Warteräumen mit unendlicher Kapazität

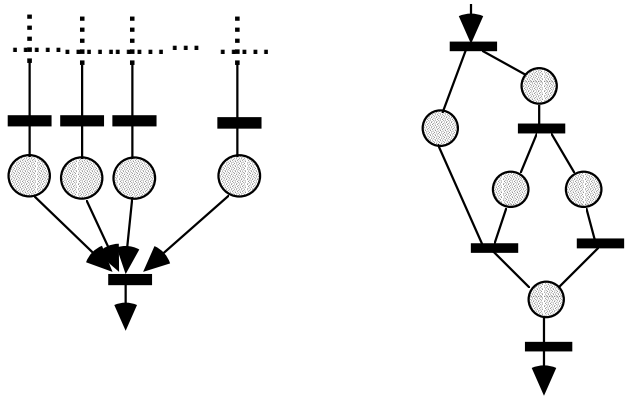
Bedienstrategie und Bedienungsunterbrechung

Die Bedienstrategie entscheidet über die Reihenfolge der Bedienung der Aufträge, die sich im Warteraum der Bedieneinheit befinden, unter Berücksichtigung ihrer Priorität. Dabei kann die Bedienung unterbrochen werden, beispielsweise um eintreffende höher priorisierte Aufträge sofort zu bedienen. Die aus der Bedienstation verdrängten Aufträge werden dann wieder in die Warteschlange eingereiht. Die häufigsten Bedienstrategien für Aufträge mit gleicher Priorität sind:
Die häufigsten Bedienstrategien für Aufträge mit gleicher Priorität sind:

- Vorwärtsstrategie (FIFO)
- Rückwärtsstrategie (LIFO)
- Abarbeiten des kürzesten Auftrages
- Zeitscheibenverteilung
- Zufällige Auswahl

6.3.9.3 Relationen

Eine wichtige Relation zwischen Objekten in Bedienungs-/Wartemodellen ist die Synchronisation von Aufträgen
Die **Synchronisation von Aufträgen** ist eine (neue) wichtige Relation zwischen Objekten in Bedienungs-/Wartemodellen. Eine typische Synchronisationsbeziehung liegt vor, wenn sich an einer bestimmten Stelle eines B/W-Modells eine bestimmte Anzahl von Aufträgen ansammeln muß, bevor diese ihren Weg fortsetzen können (siehe folgende Abbildung, links).
Erheblich kompliziert wird das Problem, wenn Aufträge in teilweise nebenläufige Teilaufträge zerlegt werden. Hier besteht den Bedarf, die zusammengehörigen Teilaufträge zu synchronisieren (siehe folgende Abbildung, rechts).



Eine Synchronisation von Ressourcen ist ebenfalls denkbar

(z.B. Krankenwagen + Fahrer + Sanitäter beim Einsatz).

6.3.9.4 Aktivitäten

In Bedienungs-/Wartemodellen ist nur eine Aktivität von Bedeutung: Die **Bedienung**.
Die Dauer der Aktivität „Bedienung“ muß aus der Kombination von Ressourcen- und Kundenattributen ermittelt werden.

6.3.9.5 Untersuchte Leistungsgrößen

Bei Bedienungs-/Wartemodellen werden im allgemeinen der Mittelwert und Varianz der folgenden Größen ermittelt und untersucht:

- Auslastung der Bedienstationen (in Prozent)
- Systemfüllung (Anzahl der Aufträge im System)
- Aufenthaltsdauer (Wartezeit plus Bedienzeit)
- Befriedigung der Anforderungen:
  - Erfüllung von Präferenzen
  - Warteverhalten der Kunden (Wartezeit bzw. Warteschlangenlänge)
  - Verlustwahrscheinlichkeit für jeden Kundentyp
    (Als Verlustwahrscheilichkeit wird die Wahrscheinlichkeit bezeichnet, mit der ein Auftrag das System ohne Bedienung verläßt)

6.3.10 Beispiel: Telefonische Kartenbestellung

Es folgt die Untersuchung eines Bediensystem mit mehrerer Bedienstationen und begrenztem Warteraum.

6.3.10.1 Systemspezifikation

Es soll die telefonische Kartenbestellung für die WM-Finale als ein spezielles Bedienungssystem mit begrenztem Warteraum untersucht werden. Das zu Modellierende System ist folgendermaßen spezifiziert:

1) Es handelt sich ausschließlich um eine telefonische Kartenbestellung.
2) Es stehen insgesamt 500 Bedienungskräfte zur Verfügung.
3) Es sind 1800 Telefonleitungen geschaltet (d.h. die Kapazität des "Warteraums" gleich 1300).
4) Jeder Kunde bucht nur eine einfache oder eine Luxus-Karte.
5) Wenn alle Bedienungskräfte belegt sind, so ertönt die Ansage "Bitte warten!".
6) Wenn eine Bedienungskraft frei wird, soll derjenige Kunde bedient werden, der schon am längsten wartet.
7) Die Kunden warten nur eine begrenzte Zeit. Die mittlere Wartebereitschaft beträgt 4 Minuten (normalverteilt). Nach Ablauf seiner Werte Wartebereitschaft verzichte der Kunde endgültig auf die telefonische Kartenbestellung und sucht sich einen Schwarzhändler.

6.3.10.2 Systemleistung

Die Systemleistung ist hinsichtlich der folgenden Kriterien zu analysieren:

- mittlere Wartezeit der Anrufer
- Auslastung der Bedienungskräfte
- mittlere Warteschlangenlänge
- Auslastung der Leitungen
- Anteil der sofort bedienten Anrufe
- Anteil der bedienten Anrufe mit Wartezeit
- Anteil der Anrufe, die keine freie Leitung erhalten
- Anteil der Anrufe, die innerhalb der Wartebereitschaft nicht bedient werden

6.3.10.3 Kundenzufriedenheit

Die Kundenzufriedenheit könnte nach der folgenden Kriterien bewertet werden:

- Anteil der Kunden, die eine einfache Karte wollten und sie bekommen haben
- Anteil der Kunden, die eine einfache Karte wollten aber bereit waren eine Luxus-Karte zu kaufen als keine einfache mehr vorhanden war

- Anteil der Kunden, die eine einfache Karte wollten aber keine Karte bekommen haben
- Anteil der Kunden, die eine Luxus-Karte wollten und sie bekommen haben
- Anteil der Kunden, die eine Luxus-Karte wollten aber bereit waren eine einfache Karte zu bestellen als keine Luxus-Karte mehr vorhanden war
- Anteil der Kunden, die eine Luxus-Karte wollten und keine bekommen haben

6.3.10.4 Modellentwurf

6.3.10.4.1 Zustandsvariablen

Die Zustandsvariablen des Modells sind
- der Bedienungsstatus einer jeden Bedienungskraft (*frei* oder *beschäftigt*)
- die Anzahl der wartenden Anrufer in der Warteschlange
- die Anzahl der freien Leitungen

Ein Zustandsdiagramm kann erzeugt werden. Eine Verfeinerung des Zustandsdiagramms führt zur hierarchischen Präzisierung der Ereignisse.

6.3.10.4.2 Ereignisse

Die folgenden Ereignisse könnten beispielsweise zuerst extrahiert werden:
1) Kundenanruf:
   Generierung des Anrufs und Zuteilung einer Bedienungskraft (falls vorhanden) oder Einreihen in die Warteschlange, wenn eine freie Leitung vorhanden.
2) Bedienungs-Ende:
   Freisetzen der Bedienungskraft oder Zuweisung eines neuen Anrufers (falls vorhanden)
3) Warten-Ende:
   Löschen eines Kunden mit abgelaufener Wartebereitschaft aus der Warteschlange. Löschen des ersten Kunden (wegen Bedienungszuteilung) aus der Warteschlange.
4) Simulations-Ende:
   Ende der Simulation, Berechnung der Simulationsergebnisse, Erstellung des Simulationsprotokolls.

Diese groben Ereignisbeschreibungen (die als Ereignistypen betrachtet werden können) müssen weiter verfeinert werden, bis sie als Ereignisroutinen präzisiert werden.

6.3.10.4.3 Ereignisroutinen

Bei Aufruf der Ereignisroutine *Kundenanruf* wird unmittelbar das nächste Ereignis des gleichen Typs generiert und in die Ereignisliste eingetragen. Hier wird die bereits erwähnte Technik verwendet, eine Folge gleichartige Ereignisse dadurch zu generieren, daß jedes Ereignis des entsprechenden Typs seinen Nachfolger erzeugt und vormerkt.
Findet eine Kunde keine freie Leitung vor (ist die Kapazität des Warteraumes also erschöpft), so endet die Ereignisliste ohne weitere Aktionen.
Ist eine Bedienungskraft unbeschäftigt, erfolgt der Aufruf einer Prozedur *Bedienung*.
In der Ereignisroutine Kundenanruf wird aus Gründen der Vereinfachung nur dann ein Kundenprotokoll generiert, wenn dieser nicht sofort bedient, sondern in eine Warteschlange eingetragen wird. Denn nur in diesem Fall muß einem Kunden ein Attribut mit der nach der vorgegebenen Normalverteilung generierten *Wartebereitschaft* zugewiesen werden.
Für den Fall der sofortigen Bedienung werden allein die entsprechenden statistischen Zähler aktualisiert (Dekrementierung der Anzahl der freien Bediener).
Das Ende der Bedienung führt dazu, daß eine Leitung freigegeben wird und, falls noch Kunden auf anderen Leitungen warten, deren erster bedient wird.
Anschließend wird das Kundenprotokoll gelöscht. (Dies kann bei Simulation einer größeren Anzahl von Kunden aus Speicherplatzgründen notwendig sein.)

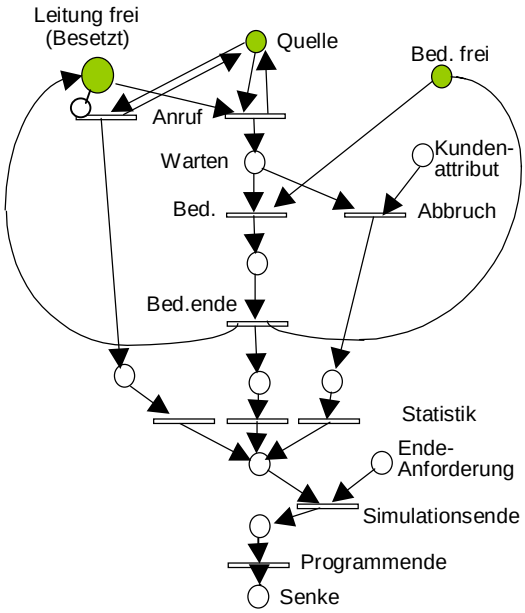
6.3.10.5 Simulationsergebnisse

Typische Ergebnisse einer Simulation von 180.000 Kundenanrufen könnten wie folgt aussehen:

- 158.000 Anrufe konnten bedient werden
- 1.610 Anrufer hatten keine freie Leitung vorgefunden
- 20.390 Anrufer gaben nach abgelaufener Wartebereitschaft vorzeitig auf
- Die durchschnittliche Wartezeit betrug ca. 70 Sekunden (wobei auch Anrufer mit sofortiger Bedienung berücksichtigt wurden)
- Ohne die sofort bedienten Kunden ergab sich für die wartenden Anrufer eine durchschnittliche Wartezeit von über 90 Sekunden
- Mittlere Auslastung eines Bedieners = (Summe aller Bedienzeiten / 500) · Simulationszeit
- Mittlere Auslastung der Leitungen = (Summe aller Bedien- plus Wartezeiten / 1.800) · Simulationszeit

6.3.10.6 Graphische Darstellung der Ereignisse

Die Ereignisse mit ihren Verflechtungen werden mit Hilfe eines Petri-Netz - Modells visualisiert.



7 Simulation Teil 4: Diskrete Simulation kontinuierlicher Systeme

7.1 Grundlagen

7.1.1 Beispiel für ein kontinuierliches System

System „fliegender Ball“.

7.1.2 Die Beschreibung eines kontinuierlichen Systems

Ein kontinuierliches System wird mit **kontinuierlichen Systemvariablen** beschrieben.
Im allgemeinen wird dies mit **partiellen Differentialgleichungen** gemacht.
Damit liegt ein strukturkontinuierliches mathematisches Modell vor.

7.1.3 Diskrete Simulation durch Diskretisierung

Aufgabenstellung: Ein kontinuierliches System soll simuliert werden.

1) Erstellung eines Modells des Systems;
   das Verhalten des Systems wird durch ein Differentialgleichungssystem beschrieben.
2) Numerische Lösung des Differentialgleichungssystems (Numerische Integration, siehe nächsten Abschnitt). Man erhält ein System aus Differenzengleichungen.
3) Diskrete Simulation durch die Anwendung der Differenzengleichungen.

Bei der diskreten Simulation kontinuierlichen Systeme wird das kontinuierliche System durch ein (geeignetes) diskretes System angenähert (approximiert).
Durch eine **Diskretisierung** werden die partiellen Differentialgleichungen in partielle Differenzengleichungen umgewandelt.
So wird aus z.B. aus dem Differentialgleichungssystem

$$y_1' = f_1(t, y_1)$$
$$y_2' = f_2(t, y_2)$$

ein System aus Differenzengleichungen:

$$y_1^* = f_1(t, y_1)$$
$$y_2^* = f_2(t, y_2)$$

Die Diskretisierung partieller Differentialgleichungen ist keine leichte Aufgabe. Hierzu gehören u.a. die Probleme, welche Art von Differenzen ($\leftarrow, \leftrightarrow, \rightarrow$), welche Ordnung der Differenzen (erste oder höhere) und welche Schrittweite der Diskretisierung.

7.2 Numerische Integration

Im allgemeinen Fall, daß das System integrierende Glieder enthält, handelt es sich um die numerische Integration eines Systems gewöhnlicher Differentialgleichungen.

Die Vielfalt der bekannten Verfahren zur numerischen Integration läßt sich in zwei Gruppen einteilen:

Einschritt- und Mehrschrittverfahren (ESV und MSV).

7.2.1 Einschrittverfahren (ESV)

Die ESV benutzen Informationen über die Lösungskurven $\mathbf{y(t)}$ nur aus dem neuen Schritt, d.h. nur an Punkten innerhalb des Intervalls $[t_{n-1}, t_n]$.
Diese Verfahren greifen also nur auf Werte bis zum Zeitpunkt t_{n-1}, nicht aber auf frühere zurück.
Die Kenntnis des Wertesatzes $\mathbf{y_0}$ des Anfangspunktes genügt also zum Start des Verfahrens. Verfahren dieser Art werden "selbststartend" genannt.

7.2.2 Mehrschrittverfahren (MSV)

Die MSV benutzen Informationen über die Lösungskurven aus mehreren Schritten.
Diese Verfahren greifen also auf Werte zurück, die zu vor t_{n-1} liegenden Zeitpunkten gehören.
Die Kenntnis mehrerer Wertesätze von Anfangspunkten ist also zum Start des Verfahrens notwendig. Verfahren dieser Art sind also nicht selbststartend, sondern erfordern einen Startprozeß, der die notwendigen vorhergehenden Wertesätze liefert.

7.2.3 Problemstellung

Gegeben sei ein Differentialgleichungssystem.

Die Ordnung des Differentialgleichungssystem ist in der Praxis oft größer als eins, das heißt es kommen nicht nur erste sondern auch zweite, dritte oder noch höhere Ableitungen im Differentialgleichungssystem vor.
Das zu simulierende physikalische (kontinuierliche) System wird durch ein Modell abgebildet. Zur Durchführung einer digitalen (also diskreten) Simulation ist die numerische Lösung des durch das Rechenmodell gegebenen Gleichungssystems erforderlich.
Das zu integrierende Differentialgleichungssystem wird als Satz von Differentialgleichungen erster Ordnung in der Form eine Vektorgleichung geschrieben:

$$\mathbf{y}' = \mathbf{f}(t, \mathbf{y}) \quad ; \quad \mathbf{y(0)} = \mathbf{y_0}$$

wobei

$$y = \begin{pmatrix} y_1(t) \\ \vdots \\ y_m(t) \end{pmatrix}$$

$\mathbf{y} = [y_1(t), y_2(t),...,y_m(t)]^t$,
$\mathbf{y}' = [y'_1(t), y'_2(t),...,y'_m(t)]^t$,

$\mathbf{f}(t, \mathbf{y}) = [f_1(t,\mathbf{y}), f_2(t,\mathbf{y}),..., f_m(t,\mathbf{y})]^t$,
$\mathbf{y_0} = [y_{1,0}, y_{2,0}, ..., y_{m,0}]^t$,
$y_{i,0} = y_i(0)$

Über die Art des DGSs

$$\mathbf{y}' = \mathbf{f}(t, \mathbf{y})$$

wird nur die Voraussetzung gemacht, daß die Ableitungsfunktionen $f_i(t, \mathbf{y})$ so beschaffen sind, daß die Ableitungen $y_i'(t)$ bereichsweise stetige Funktionen sind.
In diesem Rahmen darf das System beliebig nichtlinear sein. Die Fälle linearer DGS (mit ggf. konstanten Koeffizienten) sind in obiger Voraussetzungen als Sonderfälle enthalten. Diese Sonderfälle haben jedoch hinsichtlich der Ausführung der numerischen Integration keine besondere Bedeutung.
Die genaue Lösung des DGSs

$$\mathbf{y}' = \mathbf{f}(t, \mathbf{y})$$

sind stetige Funktionen

$$y_1(t), y_2(t), ...$$

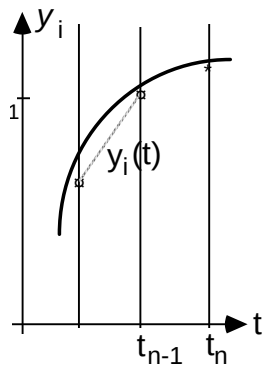
(durchgehende Verläufe). Die Verfahren der numerischen Integration gehen notwendigerweise schrittweise vor und ergeben daher diskrete Werte

$$y_{1,i}, y_{2,i},...$$

in den Schrittzeitpunkten t_i (durch ¤ bzw. * gekennzeichnet). Diese diskrete Werte stützen kontinuierliche Lösungsfunktionen (gestrichelte Verläufe)

$$y_1(t), y_2(t),...,$$

die z.B. durch Interpolation gewonnen werden können.
Im allgemeinen sind (schon) die diskreten Werte $\mathbf{y_i}$ nur Näherungswerte (erst recht sind $\mathbf{y}(t)$ nur Näherungsfunktionen!).



Das Verfahren der numerischen Integration geht für den Übergang von einem bereits erreichten Wertesatz $\mathbf{y_{n-1}}$ zu dem neuen Wertesatz $\mathbf{y_n}$ mit einer Integrationsschritt der Schrittweite $\Delta t = h$ im allgemeinen so vor, daß eine abwechselnde Folge von **f**- und **y**-Werten berechnet wird.
Hierzu wird zur Bestimmung eines Steigungswertes **f** die DG

$$\mathbf{y'} = \mathbf{f}(t, \mathbf{y})$$

innerhalb des Intervalls $[t_{n-1}, t_n]$ an einem Punkt $P = (t,\mathbf{y})$ ausgewertet. Zur Ermittlung eines **y**-Wertes werden **y**- und **f**-Werte linear kombiniert. Die Schrittweite h ist im allgemeinen (zumindest bereichsweise) konstant.

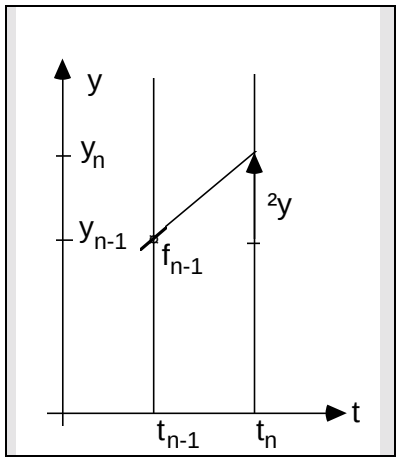
7.3 Einschritt-Verfahren: Die Runge-Kutta-Verfahren

7.3.1 Das Runge-Kutta-Verfahren erster Ordnung

Die Einfachste Version eines RK-Verfahrens ist die Methode von Euler, die nichts anderes ist als die Bestimmung des Zuwachses Δy der Näherungslösung mittels der Steigung im Punkt

$P_{n-1} = (t_{n-1}, y_{n-1})$:

$\Delta y = h \bullet f(t_{n-1}, y_{n-1})$



oder anders ausgedrückt:

$$y_n = y_{n-1} + h \bullet f(t_{n-1}, y_{n-1})$$
$$= y_{n-1} + h \bullet f_{n-1}$$

mit

$h = t_n - t_{n-1}$.

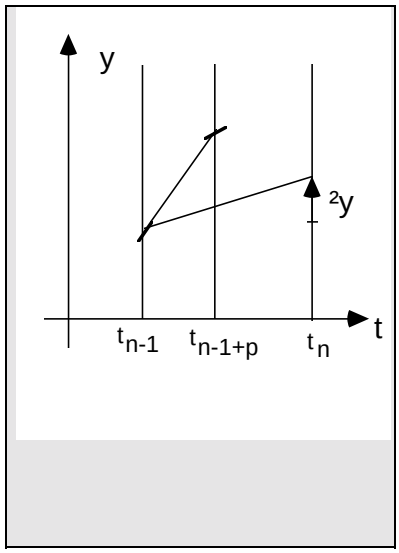
Diese Methode ist sehr grob (sie ist nur von 1. Ordnung). Sie liefert nur dann ein genaues Ergebnis, wenn das Problem trivialerweise eine lineare Lösungskurve hat.

7.3.2 Das Runge-Kutta-Verfahren 2. Ordnung

Eine Näherungslösung 2. Ordnung erhält man durch Hinzunahme des Steigungswertes an einem weiteren Punkt

P_{n-1+p}, $(p < 1)$:

$$\Delta y = h \bullet [\, c_1 f(P_{n-1}) + c_2 f(P_{n-1+p})]$$
$$= c_1 h \bullet f(P_{n-1}) + c_2 h \bullet f(P_{n-1+p})$$
$$= c_1 h \bullet f(\, t_{n-1}, y_{n-1}\,) + c_2 h \bullet f[\, t_{n-1+p} \bullet h \;,\; y_{n-1+p} \bullet h \bullet f(P_{n-1})\,]$$



Für

$$c_1 = 1 - c_2$$

und

$$p = 1/(2c_2)$$

kann eine genaue Lösung erreicht werden, wenn diese Lösung eine ganze rationale Funktion höchstens 2. Grades in t ist, wobei beim 2. Grade die Einschränkung gilt, daß

$\mathbf{y}' = \mathbf{f}(t)$

nur von t aber nicht von **y** abhängen darf.

7.3.2.1 Das modifizierte Eulersche Verfahren

Die Wahl $c_2 = 1$ ergibt das sogenannte modifizierte Eulersche Verfahren.
Es gilt:

$$c_1 = 1 - c_2 = 1 - 1 = 0;$$
$$p = 1/(2c_2) = 0{,}5;$$
$$P_{n-1+0,5} = P_{n-0,5}$$

$$\Delta y = h \bullet f(\, P_{n-1+p}\,)$$
$$= h \bullet f(\, t_{n-1} + 0{,}5 \bullet h \;,\; y_{n-1} + 0{,}5 \bullet h \bullet f(P_{n-1})\,)$$
$$= h \bullet f(\, t_{n-1} + 0{,}5 \bullet h \;,\; y_{n-1} + 0{,}5 \bullet h \bullet f(t_{n-1}, y_{n-1})\,)$$
$$y_n = y_{n-1} + \Delta y$$

Bei diesem Verfahren liegt der Punkt P_{n-1+p} auf der Tangente an Punkt P_{n-1} bei
$t = t_{n-1} + 0{,}5h$.
Der Steigungswert, den die Differentialgleichung für diesen Punkt liefert, bestimmt den Zuwachs Δy über die Schrittweite h.

Beispiel

$$y'(t,y) = u(t) - y(t)$$

mit

$$u(t) = \begin{cases} 1 & \text{für: } t \geq 0 \\ 0 & \text{sonst} \end{cases}$$

und

$$y(0) = 0.$$

Es sei:

$$h = 0{,}5.$$

Für

$$n > 0$$

gilt:

$$y'(t_{n-1}, y(t_{n-1})) = 1 - y(t_{n-1})$$
$$= 1 - y_{n-1}$$
$$= y'_{n-1}$$

$$y_{n-1} + 0{,}5h(1 - y_{n-1})$$
$$= y_{n-1} + 0{,}25(1 - y_{n-1})$$
$$= 0{,}25(1 + 3y_{n-1})$$
$$\Delta y = 0,\, 5y'(t_{n-1} + 0{,}25 \;,\; y_{n-1} + 0{,}25(1 - y_{n-1}))$$
$$= 0{,}5(1 - 0{,}25(1 + 3y_{n-1}))$$
$$= 0{,}375(1 - y_{n-1})$$

$y_n = y_{n-1} + \Delta y$
$= y_n\text{-}1 + 0{,}375(1 - y_{n-1})$
$= 0{,}125(3 + 5y_{n-1})$

n	y'_{n-1}	Δy	y_n
0	------	------	0
1	1	3/8 = 0,375	3/8 = 0,375
2	5/8 = 0,625	15/64 = 0,234	39/64 = 0,609
3	25/64 = 0,391	75/512 = 0,146	387/512 = 0,756
4	125/512 = 0,244	375/4096 = 0,092	3471/4096 = 0,847

7.3.2.2 Das verbesserte Eulersche Verfahren / Heunsches Verfahren

Die Wahl

$c_1 = c_2 = 0,5$

ergibt das sogenannte verbesserte Eulersche Verfahren. (Das Verfahren wird manchmal als Heunsches Verfahren bezeichnet.)

Es gilt:

$p = 1$

und

$\Delta y = 0,5(k_1 + k_2)$

mit

$k_1 = h \cdot f(t_{n-1}, y_{n-1}) = h \cdot f_{n-1}$

$k_2 = h \cdot f(t_{n-1} + h \ , \ y_{n-1} + k_1) = h \cdot f^p_n$

$y_n = y_{n-1} + 0,5 \cdot h \cdot (f_{n-1} + f^p_n)$

Bemerkung

Hier wird der zweite f-Wert nicht an einem Zwischenpunkt, sondern erst an der nächste Schrittstelle. Dieses Verfahren stellt einen Grenzfall zu den Prediktor-Korrektor-Verfahren dar.

7.3.3 Das Runge-Kutta-Verfahren 4. Ordnung

Das in den digitalen Simulationssystemen am häufigsten benutzten Runge-Kutta-Verfahren ist das Runge-Kutta-Verfahren 4. Ordnung (RK4). Das RK4-Verfahren zeichnet sich durch eine (relativ) einfache, übersichtliche Formel, hohe Genauigkeit und gute Stabilität aus.
Beim RK4-Verfahren werden die Steigungswerte an vier Stellen innerhalb des Intervalls $[t_{n-1}, t_n]$ für die Bildung des Funktions-Zuwachses herangezogen gemäß dem allgemeinen Ansatz:

$$\begin{aligned}\Delta y &= h\,[c_1 f(P_1) + c_2 f(P_2) + c_3 f(P_3) + c_4 f(P_4)] \\ &= c_1 k_1 + c_2 k_2 + c_3 k_3 + c_4 k_4\end{aligned}$$

mit

$$\begin{aligned}k_1 &= h\, f(t_{n-1} ; y_{n-1}) &&= h\, f_{n-1} \\ k_2 &= h\, f(t_{n-1}+0{,}5h ; y_{n-1}+0{,}5\,k_1) &&= h\,(f_n - 0{,}5)^{P1} \\ k_3 &= h\, f(t_{n-1}+0{,}5h ; y_{n-1}+0{,}5\,k_2) &&= h\,(f_{n-0{,}5})^{P2} \\ k_4 &= h\, f(t_{n-1}+h ; y_{n-1}+k_3) &&= h\,(f_n)^{P3}\end{aligned}$$

und

$$\Delta y = (1/6)k_1 + (1/3)k_2 + (1/3)k_3 + (1/6)k_4$$

Die vier auftretenden Ableitungswerte lassen sich wie folgt berechnen:

$$\begin{aligned}f_{n-1} &= f(t_{n-1};\ y_{n-1}) \\ (f_{n-0{,}5})^{P1} &= f(t_{n-1}+0{,}5h ; y_{n-1} + h\,0{,}5\, f_{n-1}) \\ &= f(t_{n-1}+0{,}5h ; (y_{n-0{,}5})^{P1}) \\ (f_{n-0{,}5})^{P2} &= f(t_{n-1}+0{,}5h ; y_{n-1} + h\,0{,}5\,(f_{n-0{,}5})^{P1}) \\ &= f(t_{n-1}+0{,}5h ; (y_{n-0{,}5})^{P2}) \\ (f_n)^{P3} &= f(t_{n-1}+h ; y_{n-1} + h\,(f_{n-0{,}5})^{P2}) \\ &= f(t_n ; (y_n)^{P3})\end{aligned}$$

7.4 Mehrschritt-Verfahren: Die Prediktor-Korrektor-Verfahren

Es sei

$\mathbf{y}' = \mathbf{f}(t, \mathbf{y})$.

Bei den Differenzengleichungen wird aus zu verschiedenen diskreten Punkten gehörenden, schon bekannten y- und f-Werten, mittels einer Differenzengleichungsformel ein **Prediktorwert** y^p gebildet. Mit diesem wird die Differentialgleichung ausgewertet, d.h. die zugehörige Ableitungsfunktion f^p berechnet. Im allgemeinen wird dann unter Verwendung des f^p-Wertes in entsprechender Weise ein weiterer Prediktorwert oder ein für den betreffenden Integrationsschritt endgültiger **Korrektorwert** y^k bestimmt.

Bei einem Prediktor-Korrektor-Verfahren (PK-Verfahren) werden vorhergehende y- und f-Werte benutzt, die zu verschiedenen äquidistanten Zeitpunkten gehören. Zuerst wird daraus mittels einer Prediktor-Formel ein **Vorhersagewert** y^p_n für den neuen Zeitpunkt t_n gebildet. Der zu diesem Wert nach der DG zugehörige Ableitungswert

$f^p_n = f(t_n , y^p_n)$

geht dann in eine Korrektor-Formel ein, die den **verbesserten** y-Wert y^k_n ergibt. Einfache PK-Verfahren betrachten y^k_n als *endgültiger* y-Wert y_n und bestimmen damit den Ableitungswert

$f_n = f(t_n , y_n)$.

Die allgemeine Form einer P-und einer K-Formel lautet:

$$y^p_n = a^*_1 y_{n-1} + ... + a^*_k y_{n-k} + h\bullet(b^*_1 f_{n-1} + ... + b^*_k f_{n-k})$$
$$y^k_n = a_1 y_{n-1} + ... + a_k y_{n-k} + h\bullet(b_0 f^p_n + b_1 f_{n-1} + ... + b_k f_{n-k})$$

Die Koeffizienten

a_i, a^*_j, b^*_1 sowie b_m

für

$i, j, l \in \{ 1, 2, ..., k \}, \quad m \in \{ 0, 1,..., k \}$

werden so gewählt, daß bei Verwendung von möglichst wenigen und möglichst wenig zurückliegenden y- und f-Werten die *gewünschte Genauigkeitsordnung* sowie eine *ausreichende numerische Stabilität* erreicht werden.

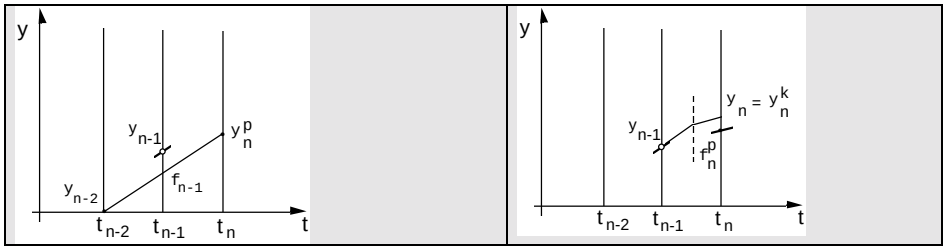
7.4.1 Prediktor-Korrektor-Verfahren 2. Ordnung

Als Beispiel werde das folgende PK-Verfahren 2. Ordnung angegeben:

$y^p_n = y_{n-2} + h\bullet(2f_{n-1})$
(d.h.: $a^*_1 = 0, \; a^*_2 = 1, \; a^*_{i>2} = 0,$
$b^*_1 = 2, \; b^*_{j>1} = 0$)

$y^k_n = y_{n-1} + h\bullet(0{,}5f^p_n + 0{,}5f_{n-1})$
(d.h.: $a_1 = 1, \; a_{i>1} = 0,$
$b_0 = b_1 = 0{,}5, \; b_{j>1} = 0$)

Der Prediktorwert y^p_n wird aus dem Wert y_{n-2} mit der Steigung f_{n-1} gebildet. Der endgültige, korrigierte Wert $y_n = y^k_n$ entsteht aus y_{n-1} unter Benutzung der *mittlere* Steigung aus f_{n-1} und f^p_n.



Beispiel

$y'(t, y(t)) = u(t) - y(t)$

mit

$y(0) = 0,$
$y'(0) = 1,$
$y(0,5) = 0,375,$
$y'(0,5) = 0,625,$
$h = 0,5$

$\forall n > 0$ gilt:

$$y'(t_{n-1}, y(t_{n-1})) = 1 - y(t_{n-1}) = 1 - y_{n-1} = y'_{n-1}$$

$$y^k_n = y_{n-1} + 0,5 \cdot h \cdot (1 - \underbrace{(y_{n-2} + 2 \cdot h \cdot f_{n-1})}_{y^p_n} + f_{n-1})$$

$$= y_{n-1} + 0,5 \cdot h \cdot (1 - y_{n-2} + f_{n-1}(1 - 2h))$$

$$\boxed{y^k_n = y_{n-1} + 0,25 \cdot (1 - y_{n-2})}$$

n = 2	$y^k_2 = y_1 + 0,25 \cdot (1 - y_0)$	0,375 + 0,25	= 0,625
n = 3	$y^k_3 = y_2 + 0,25 \cdot (1 - y_1)$	0,625 + 0,25(0,625)	= 0,78125
n = 4	$y^k_4 = y_3 + 0,25 \cdot (1 - y_2)$	0,78125 + 0,25(0,375)	= 0,87425

7.4.2 Das Runge-Kutta-Verfahren 4. Ordnung als Prediktor-Korrektor-Verfahren 4. Ordnung

Mit den allgemeinen Formeln für das Prediktor-Korrektor-Verfahren läßt sich auch das Runge-Kutta-Verfahren 4. Ordnung als Prediktor-Korrektor-Verfahren darstellen:

$(y_{n-0,5})^{P1} = y_{n-1} + h\, b^*_1\, f_{n-1}$ $\quad b^*_1 = 0,5$

$(y_{n-0,5})^{P2} = y_{n-1} + h\, b^*_2\, (f_{n-0,5})^{P1}$ $\quad b^*_2 = 0,5$

$b_0 = b_3 = 1/6$

$(y_n)^{P3} = y_{n-1} + h\, b^*_3\, (f_{n-0,5})^{P2}$ $\quad b^*_3 = 1$

$b_1 = b_2 = 1/3$

$y_n = y_{n-1} + h\,[\, b_0\, f_{n-1} + b_1\, (f_{n-0,5})^{P1} + b_2\, (f_{n-0,5})^{P2} + b_3\, (f_n)^{P3}\,]$

7.5 Vergleich: Das Runge-Kutta-Verfahren und das Prediktor-Korrektor-Verfahren

- theoretische Lösung
- numerische Integration (mit h = 0,5)

			RK-V	RK-V	PK-V	PK-V
n	$\mathbf{e^{-t}}$	$\mathbf{1 - e^{-t}}$	$\mathbf{f_n}$	$\mathbf{y_n}$	$\mathbf{f_n}$	$\mathbf{y_n}$
0	1,0	0,0	1,0	0,0		
1	0,60653	0,39346	0,625	0,375		
2	0,367879	0,63212	0,391	0,609	0,375	0,625
3	0,22313	0,77686	0,244	0,756	0,21875	0,78125
4	0,13533	0,86466		0,847	0,15275	0,84725

7.6 Beurteilung der Brauchbarkeit von Integrationsverfahren

Zur Beurteilung der Brauchbarkeit eines Integrationsverfahrens sind die folgenden Aspekte zu betrachten:

- Die Genauigkeit
- Die numerische Stabilität
- Der Rechenaufwand

7.6.1 Genauigkeit / Fehler, Rechenaufwand

Eine möglichst große **Genauigkeit** ist erreichbar, indem man sich mit vielen kleinen Schritten an das Ergebnis „heranschleicht". Kleine Schritte bedeuten eine kleine **Schrittweite**. Viele (kleine) Schritte bedeuten allerdings auch einen hohen **Rechenaufwand**.
Mit zunehmender Genauigkeit nimmt also auch der Rechenaufwand zu. Ein Abwägen zwischen Genauigkeit und Rechenaufwand ist also unerläßlich.
Eine große Genauigkeit bedeutet in der Numerik ein kleiner **Fehler**. So ist bei der numerischen Integration für die Genauigkeit im allgemeinen der **Diskretisierungs- oder Abbrechfehler** verantwortlich.
In der Praxis ist ein zulässiger Fehler vorgegeben. Da mit kleiner Schrittweite der Rechenaufwand erheblich steigt, wird die Schrittweite wird so groß wie möglich gewählt, wie es der zulässigen Fehler erlaubt.
Eventuell muß eine kleinere Schrittweite gewählt werden, wenn die Sicherstellung der numerischen Stabilität dies erfordert.

7.6.2 Numerische Stabilität

Damit ein Verfahren zur numerischen Integration **numerisch stabil** ist, muß die Schrittweite in einem definierten **Stabilitätsbereich**s liegen. Ansonsten zeigt das Verfahren eine numerische Instabilität.
Diese Instabilität ist vom gewählten numerischen Verfahren abhängig. Die Instabilität tritt auch dann auf, wenn die genaue Lösung des gegebenen Differentialgleichungssystems stabil ist.

Zur Instabilität kann es kommen, wenn die **Fehlerfortpflanzungsfunktion** nicht gedämpft ist. Dann können sich die mit jedem Schritt entstehenden Abbrechfehler in verheerender Folge aufbauen.

7.6.3 Integrationsfehler

- Rundungsfehler
- Abbrechfehler

7.6.3.1 Rundungsfehler

Rundungsfehler entstehen durch die näherungsweise Zahlendarstellung. Da diese Fehler in den meisten Anwendungsfällen sehr viel kleiner sind als der Abbrechfehler, werden hier nicht weiter behandelt.

7.6.3.2 Abbrechfehler

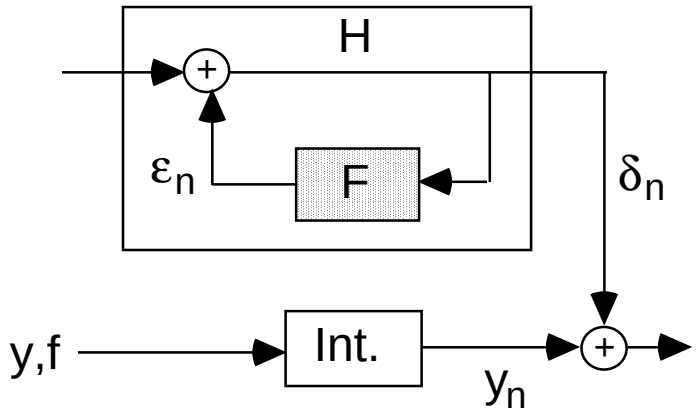
Der Abbrechfehler ist ein durch den Näherungscharakter der Integrationsformel bedingter, prinzipieller Fehler. Der Name weist darauf hin, daß dieser Fehler als durch das **Abbrechen** einer Reihenentwicklung entstanden gedeutet werden kann.
Unabhängig von der Art des Integrationsverfahrens (ESV oder MSV) ist es zwischen dem im n-ten Schritt **erzeugten Abbrechfehler** $\mathbf{R_n}$ und dem insgesamt zum Ende des n-ten Schrittes **aufgelaufenen Abbrechfehler** $\boldsymbol{\delta_n}$ zu unterscheiden.
R_n wird auch als **Schrittfehler**, δ_n als **Gesamtfehler** oder kurz als **Fehler** bezeichnet.

7.6.3.2.1 Schrittfehler

Der Schrittfehler R_n ist so definiert, daß er der Fehler des Ergebnisses y_n ist, das von der Integrationsformel geliefert wird, wenn in diese y- und f-Werte eingesetzt werden, die dem genauen Lösungsverlauf entnommen sind:

$$R_n = y\hat{}_n - y_n$$

Der Fehler δ_n entsteht über die jeweilige (durch die gegebene DG und die benutzte Integrationsformel bestimmte) **Fehlerhäufungsfunktion** H aus dem Schrittfehler Rn dadurch, daß zu R_n der über die jeweilige **Fehlerfortpflanzungsfunktion** F aus δ_n übertragene Fehler ε_n hinzugefügt wird.



7.6.3.2.2 Abbrechfehler beim Runge-Kutta-Verfahren erster Ordnung

Es wird der Fall einer einzigen gewöhnlichen DG 1. Ordnung betrachtet:

$$y' = f(t,y)$$

Die Taylor-Entwicklung der genauen Lösung y(t) um den Punkt n-1 ergibt:

$$y_n = y_{n-1} + hf + \frac{h^2}{2!}f' + \frac{h^3}{3!}f'' + \dots$$

Hierin sind f, f', f'',... genaue Werte von totalen Ableitungen nach t an der Stelle t_{n-1}, y_{n-1}.
Es sei g(t,y(t)) eine (zusätzliche) Hilfefunktion. Es gilt:

$$g' = \frac{d}{dt} g$$
$$= \frac{\partial g}{\partial t} + \frac{\partial g}{\partial y}\frac{dy}{dt}$$
$$= \frac{\partial g}{\partial t} + \frac{\partial g}{\partial y} y'$$
$$= \frac{\partial g}{\partial t} + f\frac{\partial g}{\partial y}$$
$$= \left(\frac{\partial}{\partial t} + f\frac{\partial}{\partial y}\right)\cdot g$$
$$= Dg$$
$$g_t + f \bullet g_y$$

Damit kann man f', f'',... wie folgt berechnen:

$f' = (y'') = Df = f_t + f \bullet f_y$

$f'' = (y''') = D(Df) = D(f_t + f \bullet f_y)$

$$= \left(\frac{\partial}{\partial t} + f\frac{\partial}{\partial y}\right)\cdot\left(f_t + f \cdot f_y\right)$$
$$= (f_t + f \bullet f_y)t + f \bullet (f_t + f \bullet f_y)y$$
$$= f_{tt} + f \bullet f_{yt} + f_t \bullet f_y + f \bullet (f_{ty} + f \bullet f_{yy} + f_y \bullet f_y)$$
$$= (f_{tt} + 2f \bullet f_{yt} + f^2 \bullet f_{yy}) + (f_t + f \bullet f_y) f_y$$

(Ferner gilt:

$$= (f_{tt} + 2f \bullet f_{yt} + f^2 \bullet f_{yy}) = f'' - f' \bullet f_y$$

)

7.6.3.2.3 Abbrechfehler beim Runge-Kutta-Verfahren 2. Ordnung

Beim RK-Verfahren 2. Ordnung gilt:

$\Delta y = h\bullet[\, c_1 f(P_{n-1}) + c_2 f(P_{n-1}+p)]$
$= c_1 h\bullet f(P_{n-1}) + c_2 h\bullet f(P_{n-1}+p)]$
$= c_1 h\bullet f(t_{n-1}, y_{n-1}) + c_2 h\bullet f[(t_{n-1} + p\bullet h) , (y_{n-1} + p\bullet h\bullet f(P_{n-1}))]$

mit

$c_1 + c_2 = 1;$
$p = 1/(2c_2);$

(ferner gilt:

$pc_2 = 0{,}5;$
$p^2 c_2 = 1/(4c_2)$

)

Der Term

$f[(t_{n-1}+p\bullet h) , (y_{n-1}+ p\bullet h\bullet f(P_{n-1}))]$

wird nach Taylor um den Punkt P_{n-1} entwickelt.

$f[(t_{n-1}+p\bullet h) , (y_{n-1}+ p\bullet h\bullet f)] =$
$= f + (ph)\, Df + ((ph)^2/2)\, D^2 f + ...$
$= f + (ph)(f_t + f\bullet f_y) + ((ph)^2/2)\,(f_{tt} + 2f\bullet f_{yt} + f^2\bullet f_{yy}) + ...$
$= f + (ph)f' + ((ph)^2/2)\,(f'' - f'\bullet f_y) + ...$

Damit erhält man für $y^\wedge_n = y_{n-1} + \Delta y$ den Ausdruck

$y^\wedge_n =$
$= y_{n-1} + c_1 h\bullet f + c_2 h\,[\, f + (ph)(f_t + f\bullet f_y) + ((ph)^2/2)\,(f_{tt} + 2f\bullet f_{yt} + f2\bullet f_{yy}) +...]$
$= y_{n-1} + c_1 h\bullet f + c_2 h\bullet f + c_2 h(ph)(f_t + f\bullet f_y) + c_2 h((ph)^2/2)\,(f_{tt} + 2f\bullet f_{yt} + f^2\bullet f_{yy}) +...$
$= y_{n-1} + (c_1 + c_2)h\bullet f + (h^2/2)\, f' + (h^3/4c_2)\,(f'' - f'\bullet f_y) + ...$
$= y_{n-1} + h\bullet f + (h^2/2)\, f' + (h^3/4c_2)\,(f'' - f'\bullet f_y) + ...$

Schließlich laßt sich folgendes feststellen:

$R_n = y_n - y^\wedge_n$
$= h^3 \cdot \left[\frac{1}{8c_2} f_y f' + \left(\frac{1}{6} - \frac{1}{8c_2} \right) f'' + \dots \right].$

Also dieser Abbrecher ist von 3. Ordnung in h.

Aus

$R_n = y_n - y^\wedge_n$
$= h^3 \cdot \left[\frac{1}{8c_2} f_y f' + \left(\frac{1}{6} - \frac{1}{8c_2} \right) f'' + \dots \right]$

folgt für das modifizierte Eulersche Verfahren ($c_2 = 1$):

$R_n = (h^3/24)\,(3f'\bullet f_y + f'') + ...$

und für das verbesserte Eulersche Verfahren ($c_2 = 1/2$):

$R_n = (h^3/12)\,(3f'\bullet f_y - f'') + ...$

Weiteres zum Runge-Kutta-Verfahren:

Siehe: http://www.learn-line.nrw.de/Themen/Modell/runge.htm

8 Anhang

Die Kapitel

- Eigenschaften einer guten Programmiersprache für die diskrete Simulation
- Grundsätze ergonomischer Dialoggestaltung

8.1 Kontinuierliche Simulation diskreter Systeme

Diskrete Systeme werden stets diskret simuliert.

8.2 Kontinuierliche Simulation kontinuierlicher Systeme

- Analogrechner
- Hybride Rechner
- Indirekte digitale Simulation
- "CSSL"

und was nun?

8.3 Eigenschaften einer guten Programmiersprache für die diskrete Simulation

[S.155] Zusätzlich zu den Anforderungen an eine normale gute Programmiersprache muß eine Programmiersprache für Simulationen außerdem noch effiziente Mechanismen anbieten für die/das:

1) Beschreibung des modellierten Systems
2) Vorrückung der Simulations-Zeit
3) Verfolgung von Ereignissen, Aktivitäten [activities] und Interaktionen zwischen Prozessen [process interactions]
4) Erschaffung und Löschung von Ereignissen
5) Beibehaltung/Zwischenspeicherung von assoziierten Datenstrukturen
6) Untersuchung [scanning] von Operationen und die Möglichkeit der Ausführung bestimmter Programm-Abschnitte [programm segments]
7) Datenbasiertes Management (für die Speicherung und die Wiedergewinnung von Daten, Ergebnissen und dem Modell)
8) Aufspüren und Korrigieren von Fehlern
9) Generierung von Zufallsvariablen durch passende Verteilungen
10) Interaktive graphische Simulation
11) Sammeln, Analysierung und Präsentation (flexible Berichte, Farb-Graphik, Animation) von Ergebnissen
12) Selbst-Dokumentation (der Quellcode muß auch von Analytikern verstanden werden können, die keine Programmierer sind)

8.4 Grundsätze ergonomischer Dialoggestaltung

DIN 66 234 (Bildschirmarbeitsplätze: Grundsätze ergonomischer Dialoggestaltung)

1) **Aufgabenagemessenheit**
   Ein Dialog ist aufgabenangemessen, wenn er die Erledigung der Arbeitsaufgaben des Benutzers unterstützt, ohne ihn durch Eigenschaften des Dialogsystems unnötig zu belasten.
2) **Selbstbeschreibungsfähigkeit**
   Ein Dialog ist selbstbeschreibungsfähig, wenn dem Benutzer auf Verlangen Einsatzzweck sowie Leistungsumfang des Dialogsystems erläutert werden können und wenn jeder Dialogschritt unmittelbar verständlich ist, oder der Benutzer auf Verlangen dem jeweiligen Dialogschritt entsprechende Erläuterungen erhalten kann.
3) **Steuerbarkeit**
   Ein Dialog ist steuerbar, wenn der Benutzer die Geschwindigkeit des Ablaufs und Reihenfolge von Arbeitsschritten oder Art und Umfang von Ein- und Ausgaben beeinflussen kann.
4) **Erwartungskonformität**
   Ein Dialog ist erwartungskonform, wenn er den Erwartungen der Benutzer entspricht, die sie aus Erfahrung mit bisherigen Arbeitsabläufen oder aus der Benutzerschulung mitbringen sowie im Umgang mit dem Benutzerhandbuch bilden.
5) **Fehlerrobustheit**
   Ein Dialog ist fehlerrobust, wenn trotz erkennbar fehlerhafter Eingaben das beabsichtigte Arbeitsergebnis mit minimale, oder ohne Korrekturaufwand erreicht wird. Dazu müssen dem Benutzer die Fehler zum Zwecke der Behebung verständlich gemacht werden.

8.5 Simulation in der Zukunft

- Hochleistungsprozessoren, "Klein"rechnerentwicklung, Numerische Prozessoren, Parallele Rechnerarchitekturen.
- Speichertechnologien
- Softwareentwicklungsumgebungen
- Graphische interaktive Benutzeroberflächen
- **?** Ist eine erneute "Wiederentdeckung" von analogen bzw. hybriden Rechner (oder sogar die Entwicklung von "Simulationsrechner") zu erwarten?

Viel Vergnügen!