KERNFORSCHUNGSZENTRUM

KARLSRUHE

Mai 1973 KFK 1845

Methodik der rechnergestützten Simulation

Fachgespräche vom 10. – 11. Mai 1973

veranstaltet vom
Institut für Datenverarbeitung in der Technik
Gesellschaft für Kernforschung, Karlsruhe

und dem
Institut für Informatik
Universität Stuttgart

GESELLSCHAFT
FÜR
KERNFORSCHUNG M.B.H.

KARLSRUHE

KERNFORSCHUNGSZENTRUM KARLSRUHE

KFK 1845

Institut für Datenverarbeitung in der Technik

Gesellschaft für Kernforschung Karlsruhe

Institut für Informatik

Universität Stuttgart

Methodik der rechnergestützten Simulation

Fachgespräche vom 10. - 11. Mai 1973 in Karlsruhe

Organisationskomitee:

O. Drobnik
E. Holler
F. Schumacher

H. Beilner
P. Christ
H. Ress
H. Rzehak
G. Stübel

Gesellschaft für Kernforschung m.b.H., Karlsruhe

Zusammenfassung: Methodik der rechnergestützten Simulation

Vom 1o.-11. Mai 1973 fand im Kernforschungszentrum Karlsruhe ein Fachgespräch über Methodik der rechnergestützten Simulation statt, zu dem das Institut für Datenverarbeitung in der Technik der Gesellschaft für Kernforschung, Karlsruhe und das Institut für Informatik der Universität Stuttgart eingeladen hatten.

Zweck der Fachgespräche war ein intensiver Gedankenaustausch zwischen Fachleuten auf dem Gebiet der Simulation über grundsätzliche Probleme, die bei der Planung und Verwirklichung von Simulationsprojekten auftreten.

Der hier vorgelegte Bericht umfaßt die während des Fachgesprächs gehaltenen Vorträge sowie einige ergänzende Angaben, die einen Überblick über die Zusammensetzung des Teilnehmerkreises und die Arbeitsgebiete der Vortragenden geben.

Abstract: Computer Simulation Methodology

A Workshop on "Computer Simulation Methodology" organized by the Institute of Data Processing in Technology of the Karlsruhe Nuclear Research Center and the Computer Science Department of the University of Stuttgart was held from May 1o - 11, 1973 at Karlsruhe.

The workshop's aim was to provide for an intensive exchange of thoughts among simulation specialists on fundamental problems in computer simulation design and experiments.

This report includes all papers presented at the workshop and gives a survey on the referees and their fields of activity.

Vorwort

Die rechnergestützte Simulation als Hilfsmittel zur Lösung komplexer, analytisch kaum angreifbarer Probleme findet in den unterschiedlichsten Wissensgebieten und Anwendungsbereichen immer mehr Verwendung. Die einzelnen Anwender haben meist unabhängig voneinander verschiedene und verständlicherweise auf ihre Probleme zugeschnittene Methoden der Simulation entwickelt. Die wachsende Bedeutung der Simulation als Problemlösungsmethode scheint es zu rechtfertigen, sie als eigenständiges Forschungsgebiet zu betrachten. Nur so ist es möglich, auf breiter Basis konsequent und koordiniert ihre allgemeingültigen Grundlagen zu erforschen. Dieser Prozeß ist nur durch spezielle Simulationsfachtagungen in Gang zu setzen - wie dies u.a. in den USA schon seit Jahren gehandhabt wird.

Kontakte zwischen dem Institut für Informatik der Universität Stuttgart und dem Institut für Datenverarbeitung in der Technik der Gesellschaft für Kernforschung Karlsruhe, die sich auf Grund des gemeinsamen Interesses an Problemen der rechnergestützten Simulation ergaben, führten zu der Idee, eine einschlägige Arbeitstagung zur Analyse des State of the Art im deutschsprachigen Raum zu veranstalten.

Das Organisationskomitee war sich von vornherein einig, die Tagung der Diskussion von Fachleuten vorzubehalten. Ein möglichst heterogener Teilnehmerkreis sollte die verschiedensten Interessengebiete und Anwendungsbereiche integrieren, was natürlich zu einer Erschwerung der Formulierung übergeordneter Gesichtspunkte der anläßlich der Tagung zu behandelnden Themenkreise führte.

Die Veranstalter entschlossen sich, die Methodik der rechnergestützten Simulation als Leitgedanken der Tagung hervorzuheben und die folgenden Themenkomplexe zur Diskussion zu stellen:

- Möglichkeiten und Grenzen der Simulation
  Begriffsbestimmung und Abgrenzung gegenüber anderen Methoden,
  Modellbildungsarten,
  Einsatzgebiete (Leistungssteigerung, Neuplanung, ...)
- Entwurf und Implementierung von Simulationsmodellen
  Simulationsziele (Formulierung des zu beantwortenden Fragenkomplexes)
  Systemanalyse,
  Modellsynthese (Strukturierung, Modellierungstiefe, ...),
  Implementierungshilfen (Sprachen, Programmsysteme, ...),
  Einsatz- und Benutzungsfreundlichkeit;
- Gültigkeitsbestätigung und Modelleinsatz
  Verifikation, Calibrierung, Validation,
  Experimententwurf, Ergebnisanalyse, Optimierung.

Im Rahmen dieser Thematik wurde von jedem Teilnehmer ein Gesprächsbeitrag in einer der beiden folgenden Formen erwartet:

a) Vorträge von ca. 2o-3o Minuten Dauer mit der Zielsetzung
   -Neuentwicklung und Tendenzen auf dem Gebiet der Simulationsmethodologie aufzuzeigen,
   -über Aspekte beim Einsatz von bekannten Methoden der Simulation in konkreten Projekten sowie über Motive bei der Wahl bestimmter Methoden und Erfahrungen bei ihrer Anwendung zu berichten.
b) Diskussionsbeiträge von ca. 1o Minuten Dauer über Probleme bei derzeit in der Planungs- und Entwicklungsphase stehenden Simulationsprojekten.

Schon um den Charakter eines speziellen Fachgesprächs zu verdeutlichen, schien es uns vor allem wichtig, daß in den Beiträgen die methodische Vorgehensweise ausführlicher dargelegt wird als die Bedeutung der Problemlösung für das jeweilige Fachgebiet. Daß einige Beiträge nicht ganz diesem Wunsch entsprachen, dürfte durch die kurzfristige Anberaumung der Tagung begründet sein.

Die Allgemeinheit der Thematik und die Zulassung von Berichten über abgeschlossene sowie über noch laufende oder erst geplante Projekte sollten neben dem gesicherten Wissensstand aktuelle Aktivitäten publik machen. Insbesondere sollten die Vorträge die Auswirkungen der Anwendung der Simulation auf unterschiedlichen Gebieten aufzeigen, um das Erkennen von Problemen als spezifische Simulationsprobleme zu ermöglichen. Um die stets notwendige Diskussion zu begünstigen, wurde auf einen kleinen Teilnehmerkreis Wert gelegt, obwohl dadurch der Anspruch auf Repräsentativität gemindert wurde. In diesem Zusammenhang wird auf das dem Tagungsband beigefügte Ergebnis der Teilnehmerumfrage zum Simulationsworkshop hingewiesen, in dem neben einer detaillierten Aufschlüsselung der Tagungsteilnehmer nach Zugehörigkeit zu Industrie, Universität usw. auch deren Erfahrungen mit der Simulation aufzuzeigen versucht wird.

Die Durchführung der Tagung validierte im wesentlichen die Überlegungen des Organisationskomitees. Es zeigte sich aber, daß die aktive Teilnahme in einer der von uns gewählten Formen des Vortrags bei 4o Teilnehmern innerhalb von 2 Tagen zu wenig Spielraum für die Diskussion der angesprochenen Probleme ließ. Dennoch konnte ein intensiver Gedankenaustausch angeregt werden, der sich insbesondere auf die zukünftige Kommunikation der Tagungsteilnehmer positiv auswirken wird.

Das Organisationskomitee dankt der Gesellschaft für Kernforschung und der Universität Stuttgart für die organisatorische Unterstützung bei der Vorbereitung und Durchführung der Tagung sowie bei der Herausgabe dieses Tagungsbandes.

Organisationskomitee:

Karlsruhe	O. Drobnik	Stuttgart:	H. Beilner
	E. Holler		P. Christ
	F. Schumacher		H. Ress
			H. Rzehak
			G. Stübel

Inhaltsverzeichnis

Einsatz und Aufbau von Systemen zur Simulation zeitkontinuierlicher Systeme in Echtzeit

1. Einleitende Bemerkungen zur Modellbildung und zur Modellierungstiefe

Die häufigste Aufgabenstellung für die Simulation zeitkontinuierlicher Systeme - im folgenden kurz dynamische Systeme genannt - ist die Nachbildung technischer Systeme. Die Zusammenhänge des realen Systems sind dann durch physikalische Gesetzmässigkeiten gegeben. Das Studium eines realen Systems erfordert stets gewisse Idealisierungen - sei es, um eine quantitative Analyse mit bekannten Methoden vornehmen zu können, oder um den Aufwand zu begrenzen. Diese Idealisierungen werden durch physikalische Überlegungen gerechtfertigt, d.h. man bestimmt durch eine qualitative Analyse, ob diese einen wesentlichen Einfluss auf die gewünschten Aussagen über das reale System haben. Man erhalt ein Modell des realen Systems, das stellvertretend für dieses untersucht wird. Durch Idealisierungen des realen Systems ergibt sich die gewünschte Modellierungstiefe. Im Verlauf einer Untersuchung wird man im allgemeinen von einer geringen zu einer immer grösseren Modellierungstiefe übergehen, d.h. man wird von einem stark idealisierten System zu immer realistischeren Systemen übergehen.

Zunächst wird man das dynamische System in Teilsysteme zerlegen. Bei geringer Modellierungstiefe wird dies durch den Aufbau des Systems aus einzelnen Baugruppen gegeben sein. Dies beinhaltet jedoch bereits die Idealisierung, dass das System durch konzentrierte Parameter beschreibbar ist. So ist z.B. ein elektrisches Netzwerk nur durch die Wirkung diskreter Bauelemente beschreibbar, solange die raumlichen Abmessungen der realen Bauelemente so klein sind, dass man

sich deren Wirkung an einem Punkt konzentriert denken kann. Die gegenseitige Beeinflussung der Teilsysteme gekennzeichnet durch die Vermaschung der Teilsysteme wird im Modell auch nur die Hauptwirkungen enthalten. So wird z.B. die gegenseitige Beeinflussung von Antriebseinheiten über die gemeinsame Stromversorgung unberücksichtigt bleiben.

Die Teilsysteme haben einen oder mehrere Eingänge und in der Regel einen Ausgang. Sie sind vollständig gekennzeichnet durch den Zusammenhang zwischen Ausgangsgrösse und den Eingangsgrössen. Dieser ist häufig durch Differentialgleichungen beschreibbar. In vielen Fällen können diese durch physikalische Überlegungen ermittelt werden. Bei einer experimentellen Ermittlung muss beachtet werden, dass der Zusammenhang nicht durch Wertetabellen beschreibbar ist, da der Differential- bzw. Integraloperator eine Abbildung in Funktionenräumen vermittelt. Durch Parameter-Schätzverfahren kann man zwar experimentell Aussagen zur Beschreibung eines Teilsystems gewinnen, doch setzen diese Verfahren Annahmen über den Aufbau der Gleichungen voraus, die zur Beschreibung herangezogen werden. Sind diese Annahmen nicht gültig, so erhält man auch keine korrekte Beschreibung. Das so gewonnene Modell kann nun auf zwei prinzipiell verschiedene Arten untersucht werden. Man kann die Differentialgleichungen des Gesamtsystems aufstellen und diese lösen. Bereits das Aufstellen der Gleichungen des Gesamtsystems bereitet mitunter beträchtliche Schwierigkeiten - z.B. benötigt man analytische Darstellungen empirischer Funktionen -, bedeutet aber in jedem Fall zusätzliche Arbeit. Die andere Methode ist, das Gesamtsystem zu simulieren. Dabei werden die Teilsysteme durch dynamische Vorgänge dargestellt (Simulationsmodell), die mit gewissen Idealisierungen das zur Beschreibung verwendete Modell realisieren. Dies kann ein numerischer Prozess oder ein analoges Modell sein. Man vermeidet das Aufstellen der Gleichungen für das Gesamtsystem. Es besteht eine direkte Beziehung zwischen den Grössen des realen Systems und den Grössen des Simulationsmodelles.

Ein Ändern des Modells für einzelne Teilsysteme ist ohne Änderung des übrigen Modells möglich. Der Zusammenhang zwischen diesen Modellbegriffen ist in Bild 1.1 dargestellt.

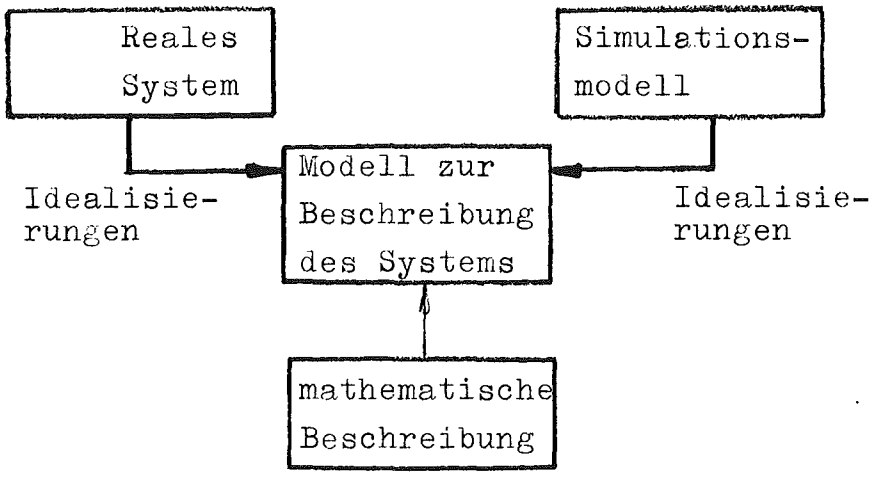


Bild 1.1 Zusammenhang zwischen realem System und Simulationsmodell

2. Darstellung der Teilsysteme

Die rechnergestützte Simulation kann die volle Flexibilität erst entfalten, wenn das Simulationsmodell leicht an veränderte Aufgabenstellungen angepasst werden kann. Es ist die Verwendung eines Simulationssystems zweckmässig, das Hilfsmittel zur bequemen Darstellung der Teilsysteme und deren Vermaschung bereitstellt. Bei hohen Ansprüchen an den Komfort wird es sich dabei um ein ganzes Programmsystem handeln.

Zur Darstellung der Teilsysteme gibt es verschiedene Möglichkeiten. Man wird einen numerischen Prozess einrichten, der näherungsweise der Modellbeschreibung entspricht. Sind die Teilsysteme durch Differentialgleichungen beschrieben, so handelt es sich bei dem numerischen Prozess um die schritt-

weise Lösung der Differentialgleichungen. Bekanntlich handelt es sich dabei nur um Näherungsverfahren. Zur Durchführung der Simulation können diese direkt programmiert werden. Grösseren Komfort erreicht man, wenn man sich einer blockorientierten Programmierung bedient, wie sie die CSSL-Empfehlungen vorsehen, z.B. durch Verwendung von CSMP. Der besondere Vorteil ist, dass der Aufbau des Simulationsmodelles aus Teilsystemen (Blöcke) direkt in die Programmierung übernommen wird. Da die Blöcke im Digitalrechner nur nacheinander für einen Lösungsschritt bearbeitet werden können, ist der Zeitaufwand beträchtlich. Er könnte reduziert werden, wenn man mehrere Blöcke (Teilsysteme) zusammenfasst und die numerische Lösung des so gewonnenen Blockes direkt programmiert. Dies erhöht jedoch die Zahl der insgesamt im Simulationssystem benötigten Blöcke.

Anstelle dass ein Teilsystem (Block) durch einen numerischen Prozess dargestellt wird, kann auch ein analoges Modell durch entsprechende Programmierung eines Analogrechners verwendet werden. Auf problemorientierter Ebene kann die Programmierung ebenfalls in einer Sprache in Anlehnung an die CSSL-Empfehlungen erfolgen. Derartige Sprachen (z.B. CSMP) sind grundsätzlich in eine Rechenschaltung für Analogrechner maschinell übersetzbar. Da es sich bei der Verwendung eines analogen Modelles nicht um einen numerischen Prozess handelt, sind die Abweichungen des Simulationsmodelles gegenüber der idealisierten Beschreibung prinzipiell anderer Art. Sie resultieren daraus, dass die Rechenelemente von ihrem idealen Verhalten abweichen. Wesentliche Unterschiede sind ferner die zeitgerechte parallele Simulation der Teilsysteme und die Tatsache, dass der Analogrechner ein Festpunktrechner ist, d.h. die Grössen müssen in den gewünschten Bereich, gewöhnlich das Intervall (-1, 1), transformiert werden.

Diese Darstellungen der Teilsysteme setzen voraus, dass eine gültige Beschreibung der Teilsysteme für die gewünschte

Modellierungstiefe bekannt ist. Ist dies nicht der Fall, und soll auch wegen der hohen Kosten auf eine entsprechende Untersuchung verzichtet werden, so kann das entsprechende Echtteil im Simulationsmodell verwendet werden. Dies macht entsprechende Vorkehrungen bei der Konzeption eines Simulationssystems nötig, die im wesentlichen auch bei der gemischten Verwendung numerischer Prozesse und analoger Modelle (hybride Rechnersysteme) erforderlich sind. Es ist anschaulich verständlich, dass ein analoges Modell vergleichsweise leicht durch ein Echtteil ersetzt werden kann. Wesentliche Punkte dabei sind:

Die Eingangsfunktionen für analoge Teilmodelle und Echtteile müssen zeitgerecht übergeben werden. Dabei beeinflussen nicht nur die Werte zu diskreten Zeitpunkten das Ergebnis, sondern es ist der gesamte Funktionsverlauf von Einfluss. Der serielle Ablauf im Digitalrechner muss mit der Modellzeit des Simulationsablaufes synchronisiert werden. Zur Datenwandelung ist geeignete Hardware nötig, die durch das Programmiersystem unterstützt sein sollte.

Ein wichtiger Sonderfall der Verwendung von Echtteilen ist der Mensch im Simulationsablauf. Hier trifft es zu, dass zu wenig bzw. zu unvollständige Beschreibungsmodelle für menschliches Verhalten beim Umgang mit technischen Systemen existieren. Auch sollte es das Ziel fortschrittlicher Technik sein, dass technische Systeme dem Menschen angepasst werden und nicht, wie in der Vergangenheit vielfach üblich, der Mensch sich den ohne Berücksichtigung menschlichen Verhaltens entwickelten Systemen anpassen muss. Dies erfordert die realitätstreue Nachbildung aller Kommunikationsmittel, mit denen Informationsaustausch zwischen Mensch und technischem System erfolgt.

3. Besondere Probleme der gemischten Verwendung digitaler und analoger Modelle bzw. Echtteile

Die Teilsysteme eines dynamischen Systemes vermitteln eine Abbildung der Eingangsfunktionen in die Ausgangsfunktion im Raum der reellen stetig (in Sonderfällen stückweise stetigen) beschränkten Funktionen. Da numerische Prozesse Operationen mit Werten von Variablen bedeuten, muss die unabhangige Problemvariable (Zeitachse) diskretisiert werden. Die kontinuierlichen Funktionen müssen durch diskrete Funktionen ersetzt werden. Für ein endliches Zeitintervall kann man diese durch eine endliche Anzahl von Variablen repräsentieren. Im Gegensatz dazu werden sowohl bei Echtteilen, wie bei analogen Modellen die Elemente des Raumes der reellen stetig beschränkten Funktionen direkt dargestellt; die Abbildungen, welche die Teilsysteme vermitteln, werden also im ursprünglichen Raum dargestellt. Für die Zusammenarbeit dieser verschiedenen Darstellungen der Teilsysteme ist also nicht nur die Wandelung einzelner Werte von der digitalen Darstellung in die analoge und umgekehrt nötig sondern die Abbildung diskreter Funktionen in kontinuierliche und umgekehrt. Digitale Simulationsmodelle einerseits und analoge Modelle bzw. Echtteile andererseits stellen demnach Operatoren in verschiedenen Räumen dar. Zur Zusammenarbeit benötigt man Operatoren, welche diese Räume ineinander abbilden. Damit erscheinen im Simulationsmodell des Gesamtsystems Teilsysteme (nämlich die Funktionentransformation), denen im realen System keine Teilsysteme entsprechen. Die Frage, wie weit dadurch Abweichungen vom Verhalten des realen Systems entstehen, muss im Zusammenhang mit der Wahl der digitalen Modelle gesehen werden, wie im folgenden dargelegt werden wird.

Die Abbildung zeitkontinuierlicher Funktionen in diskrete Funktionen wird man in natürlicher Weise so vornehmen, dass der Funktionswert im Diskretisierungspunkt übernommen wird. Da die Simulation in Echtzeit abläuft - die ablaufende Maschinenzeit stellt die unabhangige Problemvariable dar - werden dazu die zu wandelnden analogen Funktionen **in äquidistan-**

ten Abständen (Zykluszeit) abgetastet und der Analogwert digitalisiert. Die Zykluszeit muss so gewählt werden, dass nach deren Ablauf der Digitalrechner mit Sicherheit zur Verarbeitung der neuen Werte bereit ist. Sie hangt also von dem Umfang der in einem Zyklusintervall zu bearbeitenden numerischen Operationen ab und damit auch von den verwendeten digitalen Modellen, insbesondere den Integrationsverfahren. Die Zyklusintervalle werden durch einen entsprechend genauen Taktgeber bestimmt. Die Operationen am Digitalrechner müssen damit synchronisiert werden. Hybride Rechnersysteme verfügen über die entsprechende Hard- und Software. Informationstheoretisch genügt es, die Zykluszeit entsprechend dem Abtasttheorem zu wählen; wegen der weiter unten hergeleiteten Konsistenzbedingung muss man aber wesentlich kürzere Zykluszeiten wählen. Ist die Diskretisierung der kontinuierlichen Funktionen verhaltnismässig unproblematisch, so gilt das nicht für die Konstruktion kontinuierlicher Funktionen aus diskreten. Der lokale Verlauf zwischen zwei Diskretisierungspunkten muss aus dem globalen Verlauf sinnvoll konstruiert werden. Dies ist eine Interpolationsaufgabe. Für lineare Interpolation erhalt man z.B. eine stetige Funktion, die allerdings in den Stützstellen nicht differenzierbar ist. Stellt man höhere Anforderungen an die Glattheit der Funktion, so kann man aufwendigere Interpolationsverfahren verwenden.

Interpolationsverfahren erfordern, dass für die Bestimmung des Verlaufes zwischen t_i und t_{i+1} zumindest der Wert von t_{i+1} (z.B. bei linearer Interpolation) bekannt sein muss. Berücksichtigt man, dass analoge und digitale Modellteile im allgemeinen eine geschlossene Schleife bilden (vgl. Bild 3.1), so ist ersichtlich, dass dies nur erfüllbar ist, wenn man eine Zeitverzögerung der gewandelten Funktionen in Kauf nimmt.

Interpolation:

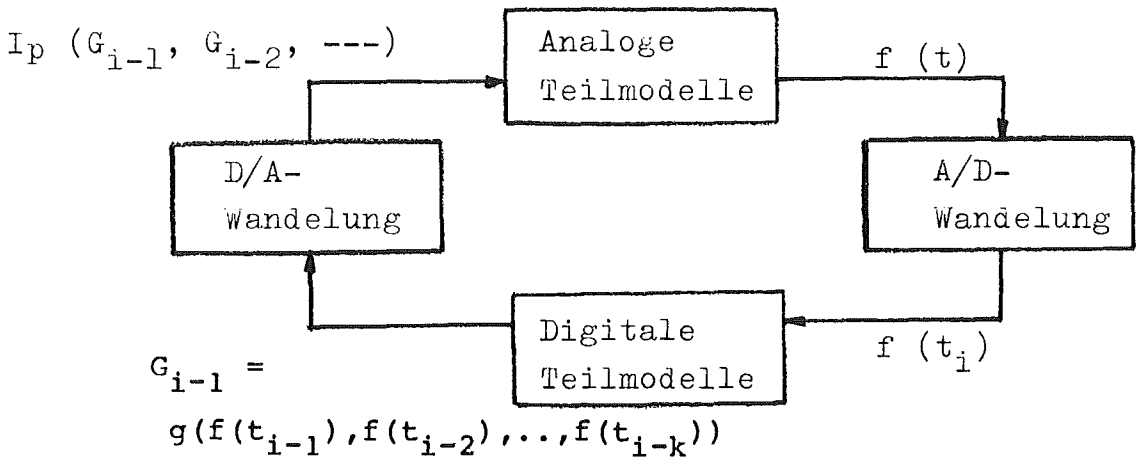
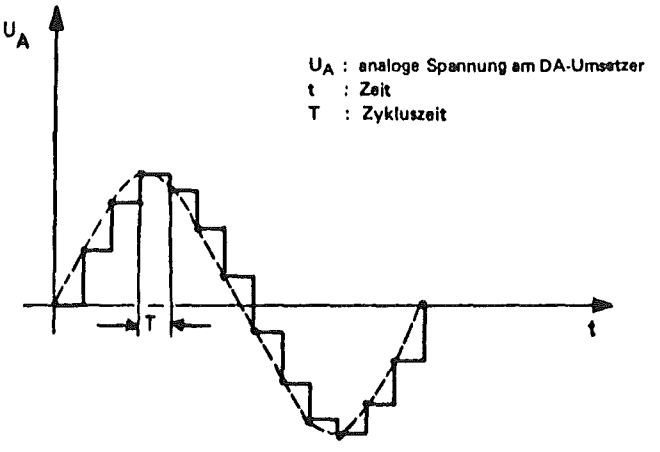


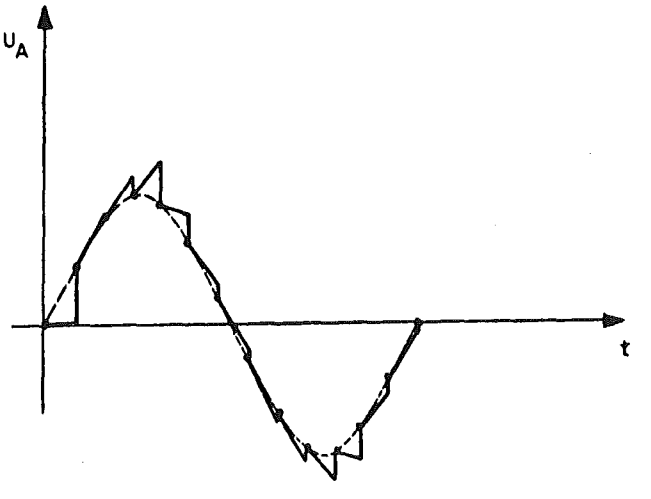
Bild 3.1 Geschlossene Schleife einer hybriden Simulation zur Zeit $t_i \leq t < t_{i+1}$.

In Bild 3.1 ist auch bereits eingetragen, dass durch den Zeitbedarf des Digitalrechners ohnedies eine Zeitverzögerung von einer Zykluszeit entsteht.

An Stelle der Interpolation werden häufig daher Extrapolationsverfahren angewendet. Es sind zwei Verfahren gebräuchlich: die Verwendung von Treppen- oder Rampenfunktionen. Bei der Anwendung von Rampenfunktionen wird im digitalen Teilmodell die Rampensteilheit zusätzlich als Näherungswert der Steigung der idealen Ausgangsfunktion des Teilmodelles berechnet. Bild 3.2 stellt beide Verfahren für eine Sinusfunktion als ideale Ausgangsfunktion gegenüber.



Extrapolation durch Treppenfunktion



Extrapolation durch Rampenfunktion

Bild 3.2 Extrapolation mittels Treppen- und Rampenfunktion

Bei beiden Verfahren erhält man Unstetigkeiten in den Stützstellen. Welches Verfahren besser ist, hängt vom Steigungs- und Krümmungsverhalten der idealen Funktion ab. Die Unstetigkeit wird in vielen Fällen weiter nicht stören z.B. wenn über diese Funktion integriert wird. Besonders stören jedoch diese Unstetigkeiten, wenn dadurch Eigenfrequenzen von Teilsystemen angeregt werden, die im Verhalten des realen Systemes keine Bedeutung haben. Dies ist bei der Verwendung von Echtteilen häufig der Fall, während man bei der Verwendung von

analogen Modellen die realen Teilsysteme meist so weit idealisiert hat, dass das Beschreibungsmodell diese Eigenfrequenzen nicht mehr aufweist. Zur Beseitigung der Unstetigkeiten verwendet man Tiefpässe, die jedoch wegen deren Einschwingverhalten Verzögerungszeiten bedeuten, so dass man sich den Verhältnissen bei der Verwendung von Interpolationsverfahren nähert.

Die Zeitverzögerungen durch den Zeitbedarf des Digitalrechners und die Verfahren zur Funktionsrekonstruktion spielen eine besonders unangenehme Rolle. Aus der numerischen Analysis ist eine Konsistenzbedingung bekannt, die eine Aussage darstellt, unter welchen Bedingungen die numerische Lösung gegen die exakte Lösung konvergiert, wenn die Schrittweite gegen Null strebt. Analog dazu wird man hier fordern, dass sich das Verhalten des Simulationsmodelles beliebig dem zur Beschreibung des realen Systems verwendeten Modell annähern lässt, wenn man nur die Zykluszeit beliebig klein wählt. Eine derartige Konsistenzbedingung für die hybride Lösung von Differentialgleichungen lässt sich formulieren.

Das dynamische System sei durch das System von Differentialgleichungen

$$y'(t) = f(y(t), t) \; ; \quad y(t_0) = y_0 \qquad (1)$$

beschrieben. Die Lösung von (1) wird dann dargestellt durch die Vektorfunktion

$$y(t) = y(t_0) + \int_{t_0}^{t} f(y(\tau), \tau)\, d\tau \qquad (2)$$

Da die Näherungslösung $\tilde{y}(t)$ im allgemeinen Fall als kontinuierliche Funktion zwischen den Stützstellen t_i ermittelt wird und die Lösung schrittweise fortgesetzt wird, gilt für

die Näherungslösung die Gleichung

$$\tilde{y}(t) = \tilde{y}(t_i) + \int_{t_i}^{t} \tilde{f}(\tilde{y}(\tau), \tau)\, d\tau, \qquad (3)$$

$$t_i \le t < t_{i+1}; \quad i = 0, 1, \ldots, n$$
$$\tilde{y}(t_0) = y(t_0)$$

Damit lässt sich die Konsistenzbedingung formulieren. Es gilt:

Es sei f und $\tilde{f}$ stetig bezüglich y bzw. $\tilde{y}$ und t und es gelte die Lipschitz-Bedingung. Man betrachtet die Folge $\tilde{y}^{(\Delta t)}(t)$ der Näherungslösungen gemäss (3) für $t_{i+1} - t_i = \Delta t \to 0$. Dann ist notwendig und hinreichend für die Konvergenz der Folge $\tilde{y}^{(\Delta t)}(t)$ gegen die exakte Lösung y (t), dass

$$\tilde{f}(y_i, t_i) = f(y_i, t_i).$$

Dies besagt, dass die exakte Funktion f derart durch eine Näherungsfunktion $\tilde{f}$ ersetzt werden muss, dass die Funktionswerte in den Diskretisierungspunkten übereinstimmen.

Beweis:

Der Fehler e_{i+1} im Punkte t_{i+1} ergibt sich gemäss

$$e_{i+1} = e_i + \int_{t_i}^{t_{i+1}} \tilde{f}(\tilde{y}(\tau), \tau)\, d\tau - \int_{t_i}^{t_{i+1}} f(y(\tau), \tau)\, d\tau$$

$$= e_i + \int_{t_i}^{t_{i+1}} \tilde{f}(\tilde{y}(\tau), \tau)\, d\tau - \int_{t_i}^{t_{i+1}} \tilde{f}(y(\tau), \tau)\, d\tau \qquad (4)$$

$$+ \int_{t_i}^{t_{i+1}} \tilde{f}(y(\tau), \tau)\, d\tau - \int_{t_i}^{t_{i+1}} f(y(\tau), \tau)\, d\tau$$

Dies kann man wie folgt abschätzen:

$$|e_{i+1}| \leq |e_i| + \left|\int_{t_i}^{t_{i+1}} \left(\tilde{f}(\tilde{y}(\tau),\tau) - \tilde{f}(y(\tau),\tau)\right) d\tau\right| \qquad (5)$$

$$+ \left|\int_{t_i}^{t_{i+1}} \left(\tilde{f}(y(\tau),\tau) - f(y(\tau),\tau)\right) d\tau\right|$$

Es sei

$$\varphi(\Delta t) = \max_{t \in (t_i, t_{i+1})} |\tilde{f}(y(t),t) - f(y(t),t)| \qquad (6)$$

Wegen $\tilde{f}(y(t_i), t_i) = f(y(t_i), t_i)$ und der Stetigkeit von $\tilde{f}$ und f gilt

$$\varphi(\Delta t) \to 0 \text{ für } \Delta t \to 0$$

Damit ergibt sich mit der Lipschitz- Konstanten L

$$|e_{i+1}| \leq |e_i| + \Delta t\,(L|e_i| + \varphi(\Delta t))$$

$$= |e_i|(1 + \Delta t \cdot L) + \Delta t \cdot \varphi(\Delta t)$$

Mit $e_0 = 0$ ist wegen $1 + \delta < \exp(\delta)$ für $\delta > 0$

$$|e_i| \leq \frac{\exp(i \cdot \Delta t \cdot L) - 1}{\Delta t \cdot L} \cdot \Delta t \cdot \varphi(\Delta t)$$

$$= \frac{\exp((t_i - t_0)L) - 1}{L} \varphi(\Delta t) \qquad (7)$$

Wegen $\varphi(\Delta t) \to 0$ für $\Delta t \to 0$ ist die hinreichende Bedingung bewiesen. Die Notwendigkeit ergibt sich aus Existenz und Eindeutigkeit der Lösung. Der Beweis enthält auch die konstruktive Aussage, dass gemäss (7) der Fehler mit zunehmender Integrationszeit kummuliert.

Durch die Zeitverzögerungen kann die Konsistenzbedingung von Sonderfällen abgesehen nicht erfüllt werden. Dieser Sachverhalt ist unbefriedigend, umreisst aber klar die Bedeutung dieses systembedingten Fehlers. Durch Benutzen der Taylor-Entwicklung kann man bei Übergabe der Funktionen an die digitalen Teilmodelle eine angenäherte Parallelverschiebung zur Kompensation der Verzögerungszeiten vornehmen. Dies ist jedoch nur für kurze Verzögerungen befriedigend, da im allgemeinen nur das lineare Glied der Taylor-Reihe zur Verfügung steht. Man ist daher gezwungen, den Fehler durch entsprechend kurze Zykluszeiten zu begrenzen. Dies erfordert eine effektive Programmierung des Digitalrechners. Die verwendeten numerischen Verfahren sollten möglichst einfach sein. Der Gewinn an Genauigkeit für aufwendigere numerische Verfahren wiegt die zusatzlichen Fehler durch Vergrösserung der Zykluszeit vielfach nicht auf.

4. Schlussbemerkungen

Simulationssysteme zur Simulation dynamischer Systeme in Echtzeit werden benötigt, um Echtteile in das Simulationssystem einbeziehen zu können. Als Echtteil in diesem Sinne ist auch der Mensch anzusehen, wenn dieser unmittelbar am Simulationsablauf beteiligt ist. Bedingt durch die Operationszeit des Digitalrechners und die Notwendigkeit der Funktionstransformation diskreter Funktionen in kontinuierliche entstehen Zeitfehler, die nur für entsprechend kurze Zykluszeiten befriedigend korrigiert werden können. Dies führt dazu, dass die Zykluszeit wesentlich kürzer gewählt werden muss, als sich aus dem Abtasttheorem ergibt. Bei umfangreichen Systemen mit schnell ablaufenden Vorgängen stösst man auch heute noch an die Leistungsfähigkeit digitaler Rechenanalgen. Auch ist es fraglich, ob die extrem hohen Kosten für die Verwendung superschneller Digitalrechner ökonomisch zu rechtfertigen sind.

Die Verwendung hybrider Rechensysteme bietet eine Reihe von Vorteilen. Zunächst enthalten diese standardmässig alle Hardware- und Software-Einrichtungen zur Synchronisation und zum Datenaustausch. Durch den programmierbaren Analogrechner kann die Funktionstransformation den Bedürfnissen des Einzelfalles angepasst werden. Die Zeitbillanz des Digitalrechners kann durch zwei Massnahmen entscheidend verbessert werden. Teilsysteme mit rasch ablaufenden Vorgängen können als analoge Modelle realisiert werden. Dadurch kann man ohne Verlust an Genauigkeit mit grösseren Zykluszeiten arbeiten. Durch die parallele Arbeitsweise des Analogrechners kann auch der Zeitbedarf des Digitalrechners pro Zyklus verringert werden, so dass man Zykluszeiten erreicht, die für die geforderte Genauigkeit erforderlich sind. Schliesslich kann ein analoges Teilmodell ohne Andern der Programmierung durch ein Echtteil ersetzt werden. Digitale Teilmodelle wird man dann in erster Linie dort verwenden, wo analoge Teilmodelle nicht befriedigend dargestellt werden können. Die Effektivität anderer Möglichkeiten - z.B. linearisierte analoge Teilmodelle zu verwenden und die Koeffizienten der Linearisierung von Zyklusintervall zu Zyklusintervall digital neu zu berechnen - ist noch nicht genügend untersucht worden.

Nachteilig für die Verwendung ist vor allem die Programmierung der analogen Systemkomponente. Wünschenswert wäre eine Programmierung auf problemorientierte Ebene durch Verwendung einer CSSL-Sprache. Allerdings sind bekannte Programmiersprachen zur Simulation kontinuierlicher Systeme bisher nur in digitalen Simulationssystemen ohne Berücksichtigung der Echtzeit-Forderungen implementiert worden.

Literatur:

Bekey, G.A., and J.W. Karplus:
Hybrid Computation. Wiley, 1968.

Henrici, P.: Discrete Variable Methods in Ordinary Differential Equations. Wiley, 1968.

Rzehak, H. Ein Beitrag zur Analyse des Einflusses systembedingter Fehler auf die Lösung von gewöhnlichen Differenzialgleichungen mit hybriden Rechensystemen.
Dissertation Universität Stuttgart, 1972.

Prinzipieller Aufbau einer Hybridsimulation

von

Ing.(grad) R. Vogel

Dornier GmbH Friedrichshafen

Inhalt:

1. Zielsetzung

Die Simulation ist im Flugzeugbau ein wichtiges Hilfsmittel für die Entwicklung und Konstruktion eines Fluggerätes sowie für die Entwicklung und Auslegung eines Flugreglers.

Mit Hilfe der Simulation kann schon im Vorstadium einer Neuentwicklung entschieden werden, ob diese Konzeption technisch machbar ist oder nicht.

Um eine Simulation durchführen zu können, muß zunächst ein einfaches mathematisches Modell des Fluggerätes erstellt werden, welches Flugmechanik und Aerodynamik enthält. Ferner muß in groben Zügen ein Reglermodell erarbeitet sein.

2. Mathematisches Modell

Ein Fluggerät wird zumeist durch folgende Differentialgleichungen beschrieben:

a. Triebwerksmodell, welches die Triebwerksdynamik enthält und durch Eingabe verschiedener Parameter (Gashebelposition, Fluggeschwindigkeit, Flughöhe) den entsprechenden Schub als Ausgangsparameter liefert.

b. 3 Kräftegleichungen (Flugmechanik und Aerodynamik), welche die Beschleunigungen in den 3 Flugkörperachsen liefern. Hieraus werden durch Integration Geschwindigkeiten und Wege (flugkörperfest) ermittelt.

c. 3 Momentengleichungen, die uns die Drehbeschleunigungen liefern, woraus wiederum durch Integration Drehgeschwindigkeit und Fluglage (gegenüber erdfestem System) ermittelt wird.

d. Transformation der flugkörperfesten Geschwindigkeiten und Wege in das erdfeste System.

3. Warum Hybrid ?

Bis zu dieser Stelle des Vortrags wurde das Wort Hybrid überhaupt noch nicht benutzt !
Man könnte genausogut auf die Idee kommen, Flugsimulationen rein digital bzw. rein analog abzuwickeln.
Warum also Hybrid ?

Die rein analoge Simulation scheidet aus wesentlichen Gründen aus, anhand von 2 Beispielen erläutert:

a. Beispiel Höhenauflösung:
   Der Einsatzbereich eines Flugzeuges liegt zwischen 0 - 10 km (10 km $\hat{=}$ 10^4m $\hat{=}$ 10^6cm). Nun muß aber bei Start und Landevorgängen die Höhenauflösung kleiner als 1 cm sein um z.B. Vorgänge am Fahrwerk untersuchen zu können (Federweg, Fahrwerk z.B. 30 cm).
   Eine Genauigkeit von 6 Zehnerpotenzen erreicht kein Analogrechner.

b. Beispiel Funktionen mehrerer Veränderlicher:
   Räumliche Kurvenscharen, wie sie in der Aerodynamik häufig vorkommen, lassen sich analog praktisch nicht realisieren.

Die rein digitale Simulation hat ebenfalls wesentliche Nachteile.

Bei der Simulation von Fluggeräten besteht die Notwendigkeit, einen Flugregler simulieren zu können. Würde man diese Reglersimulation digital durchführen, so müßte bei jeder Änderung eines Reglerparameters eine Lochkarte eingelesen werden.

Führt man die Reglersimulation dagegen analog durch, so kann man durch einfaches Verändern von Potentiometern die Reglerparameter verändern.

Aus obigen Gründen kommt man fast zwangsläufig zum Hybridrechner, in dessen Digitalteil man das mathematische Modell ablaufen läßt und in dessen Analogteil man den Regler (mit Stellmotoren) simuliert.

4. Rechnerkonfiguration

Im Hause Dornier steht eine komplette Hybridrechenanlage zur Verfügung, bestehend aus:

Digitalrechner	SDS 9300 mit entsprechender Peripherie
Analogrechner	BECKMANN EASE 2133
Interface	BECKMANN mit AD/DA-Wandlern, Interruptleitungen usw.
Fluglenkstand	DORNIER
Symbolgenerator mit Display	DORNIER

ferner Dreiachsentisch, verschiedene Schreiber usw.

5. Programmiersprache

Für obige Rechnerkonfiguration existiert ein Hauptprogramm "HYBRIDMASTER", welches alle notwendigen Steuerfunktionen des Hybridrechners erledigt (z.B. Taktsteuerung, Modesteuerung des Analogrechners usw.) und es erlaubt, das eigentliche Problem (mathem. Modell) in Form von FORTRAN-Unterprogrammen zu schreiben. Es gelten dabei natürlich die bekannten Vorteile der Fortranprogrammierung (schnelle Programmierbarkeit, leichte Lesbarkeit, leichtes Testen usw.)

6. Modesteuerung durch HYBRIDMASTER

Durch Tastendruck können folgende Zustände des Hybridrechners aufgerufen werden:

PARASET : Analogrechner im Mode "Potset", Setzen der Potentiometer durch den Digitalrechner oder von Hand, Einlesen von Parametern

IC : (INITIAL CONDITION) Analogrechner im Mode "Anfangsbedingungen setzen"; Durchlaufen eines Programmteiles, wodurch alle Anfangsbedingungen des mathem. Modells und des Analogrechners eingestellt werden.

COMPUTE : Durchlaufen des mathem. Modells (eigentliche Simulation) und AD/DA-Wandlung in einer vorgegebenen Taktzeit (Analogrechner im Mode "Dauerrechnen").

HOLD : "Einfrieren" des momentanen Simulationszustands (Analogrechner im Mode "Halt")

DIAGN : Ausgabe einer Diagnostik des "eingefrorenen" Simulationszustandes (Compute Diagnostik) bzw. des IC-Zustandes (IC-Diagnostik).

7. Zeitlicher Ablauf innerhalb eines Taktschrittes, Echtzeit erzielbare Taktzeiten

Hybridsimulationen werden sinnvollerweise nur dann angewandt, wenn das Problem (Simulation eines Fluggerätes) in Echtzeit ablaufen soll.

Da das mathematische Modell, in FORTRAN geschrieben, keine konstante Laufzeit aufweist (variable Laufzeit bei DO-Loops, Interpolationen usw.), muß ein externer Taktgeber dafür sorgen, daß die Zeit zum einmaligen Durchlaufen des mathematischen Modells immer konstant bleibt (s. Schaubild "Zeitablauf innerhalb einer Taktzeit").

Die wichtigsten, bei Dornier durchgeführten Hybridsimulationen weisen Programmlängen zwischen 8 - 20 kWorten Kernspeicherbedarf auf.

Es sind dies die Simulationen:

DO 31, KIEBITZ, AERODYNE, HOEHENFORSCHUNGSRAKETE usw.

Die Taktzeiten obiger Simulationen liegen zwischen 10 und 40 ms (100 - 25 Hz), wobei meist 40 ms erreicht wurden (10 ms nur in speziellen Fällen). Diese Taktzeiten sind klein genug, um vernünftige Integrationen der Beschleunigungen zu erhalten und um ein Display mit einer vernünftigen Bildfrequenz anzusteuern.

8. Intergration von Echtteilen in den Simulationsablauf

Wenn das Simulationsmodell durch Windkanalmessungen usw. soweit verfeinert ist, daß man von einer Endphase der Simulation sprechen kann, geht man daran, Echtteile in den Simulationsprozeß einzubeziehen:

a. Integration des echten Flugreglers (der Analogteil des Hybridrechners übernimmt nur noch die Anpassung). Hiermit können etwaige Fehler des echten Reglers beseitigt werden (Wegfall der Simulation des Reglers).

b. Integration von Rudermaschinen, Rudergestängen (Lose im Gestänge !) und Ruder, wobei Ruderwinkel mittels Potentiometer direkt am Ruder abgegriffen werden. (Wegfall der Simulation der Rudermaschinen, Zeitkonstante!)

c. Integration der Meßpakete Lotkreisel und Wendekreisel:

Hierzu müssen die Fluglagen des Fluggerätes, die durch das mathematische Modell geliefert werden, auf einen Dreiachsentisch gegeben werden, auf dem Lot- und Wendekreisel montiert sind.
Lot- und Wendekreisel liefern nun die echten Fluglagen bzw. Drehgeschwindigkeiten, die vom Flugregler benötigt werden. (Wegfall der Simulation der Hysterese des Lot-

kreisels und der Meßungenauigkeiten der Wende-Kreisel).

In diesem Stadium der Simulation bleibt vom Hybridrechner nichts mehr übrig. Es werden nur noch Flugmechanik und Aerodynamik mittels des mathematischen Modells im Digitalrechner simuliert.
Der Hybridrechner ist übergegangen in einen Prozeßrechner mit entsprechender Peripherie.

9. Grenzen der Hybridsimulation, Ausblick

Die wesentlichste Grenze der Hybridsimulation liegt wohl darin, daß wegen Mindestanforderungen an das mathematische Modell eine bestimmte Programmlänge und damit eine Mindesttaktzeit nicht unterschritten werden kann.

Das Shannon'sche Abtasttheorem sagt aus, daß die Taktfrequenz mindestens um den Faktor 2 höher sein muß, als die höchste Eigenfrequenz des mathematischen Modells.

Die Praxis hat gezeigt, daß dieser Faktor nicht bei 2 sondern zwischen 6 und 7 liegt. Bei Taktfrequenzen in der Größenordnung 25 Hz ergibt sich damit die maximale Eigenfrequenz des Fluggerätes zu 3 - 5 Hz.
Fluggeräte mit höheren Eigenfrequenzen (z.B. Raketen) können also nur in speziellen Fällen hybrid simuliert werden.

Die zweite Grenze der Hybridsimulation liegt in der ungenügenden Erfaßbarkeit vieler Einflußgrößen auf das mathematische Modell (z.B. ungenaue Windkanalmessungen).

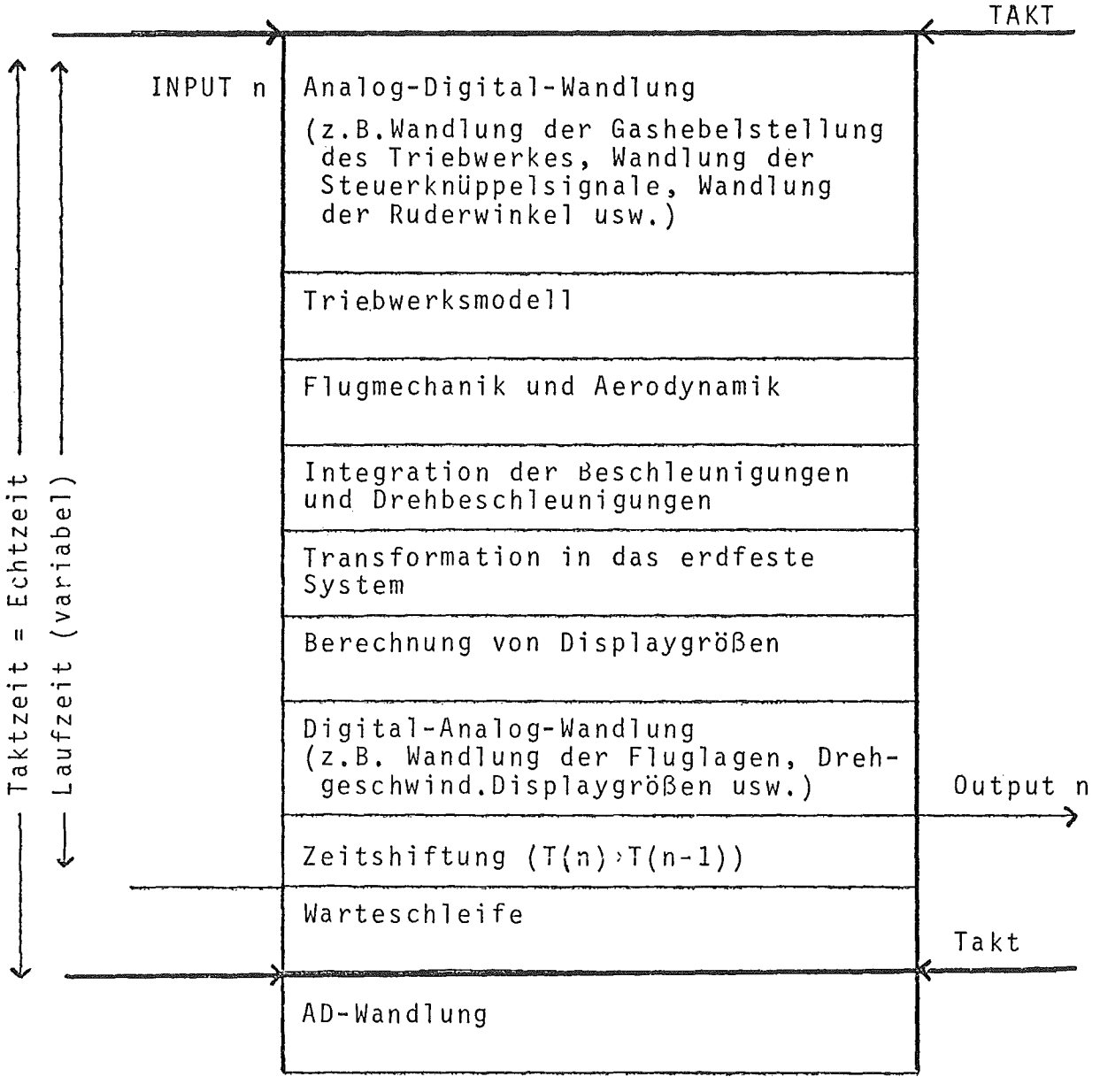
Diese wenigen Beispiele sollen aufzeigen, daß die Hybridsimulation eben doch "nur" ein Hilfsmittel bei der Ent-

wicklung und Auslegung neuer Fluggeräte ist.

Abschließend sei noch bemerkt, daß auch die beste Hybridsimulation echte Flugverusche nicht ersetzen kann.

Schaubild

"Zeitablauf innerhalb einer Taktzeit"



LOGISCHE UND FUNKTIONELLE SIMULATION EINER KOMPLEXEN HARDWARE-STEUERUNG ZUR DATENVORVERARBEITUNG

von H. Weisschuh
Institut für Nachrichtenvermittlung und Datenverarbeitung
Universität Stuttgart

---

1. Einführung

Zur Steuerung von technischen Prozessen werden heute oft digitale Hardware-Steuersysteme eingesetzt. Diese Systeme sind von ihrer Struktur her teilweise sehr komplex und es ist ein beachtlicher Aufwand an Hardware für deren Realisierung erforderlich. Bei der Entwicklung und beim Aufbau derartiger Steuersysteme treten folgende Probleme auf:

- die Überprüfung der logischen Funktion auf prinzipielle Fehler ist erst nach Aufbau eines Steuersystems möglich
- die Auswirkung einer fehlerbehafteten Umgebung ist schlecht nachprüfbar
- die Fehlersuche ist durch mangelnden Bedienungskomfort erschwert
- der Vergleich mehrerer Entwurfsvarianten ist nur schwer möglich
- die Änderungen oder Erweiterungen sind oft nur unter großem Zeit- und Kostenaufwand möglich.
- die Untersuchung des Zusammenwirkens mehrerer Steuersysteme ist nicht möglich.

Ein Hilfsmittel zur Lösung dieser Probleme ist die Simulation des Steuersystems mit Hilfe eines Digitalrechners.

Fehler in der logischen Funktion werden sofort erkannt. Die Untersuchung der Auswirkung einer fehlerbehafteten Umgebung ist durch Eingabe von falschen Eingabeinformationen möglich. Die Fehlersuche wird erleichtert, da Informationen über interne Zustände des Steuersystems ausgedruckt werden können. Der Vergleich mehrerer Entwurfsvarianten kann mit verschiedenen Programmvarianten durchgeführt werden und Änderungen am System lassen sich mittels einfacher Änderung des Simulationsprogrammes durchführen. Die Protokolle bei

der Simulation eignen sich besonders gut zur Änderungskontrolle und zur Dokumentation. Das Zusammenwirken mehrerer Steuereinheiten kann untersucht werden durch das Zusammenwirken mehrerer Simulationsprogramme, die jeweils ein Steuersystem nachbilden.

Im folgenden sollen am Beispiel einer Datenvorverarbeitungseinheit aus einem projektierten Fernsprechvermittlungssystem die Möglichkeiten, die die Simulation bietet, aufgezeigt werden.

2. Struktur und Simulation der Steuerung

2.1 Struktur und Funktionsweise

Die prinzipielle Struktur der Steuerung des hier zugrundegelegten Fernsprechvermittlungssystems /1/ ist im Bild 1 dargestellt. Die Baugruppen für die Übertragung der Sprache sind der Übersichtlichkeit wegen weggelassen, da sie für das Verständnis der Simulation nicht benötigt werden.

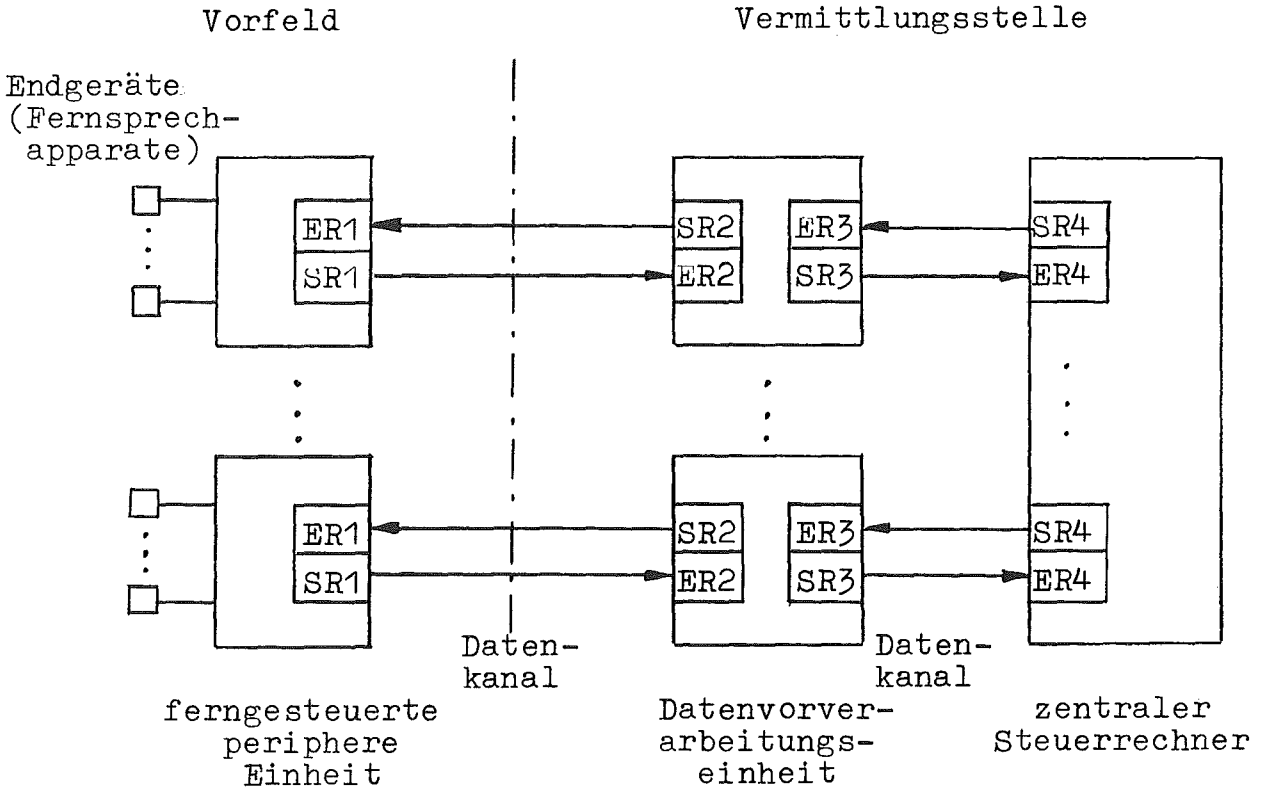


Bild 1 : Struktur der Steuerung der betrachteten Fernsprechvermittlungsstelle.

Die Fernsprechteilnehmer sind über ferngesteuerte periphere Einheiten, sog. Konzentratoren, an das Vermittlungssystem angeschlossen. Diese ferngesteuerten peripheren Einheiten haben von der Steuerung her gesehen die Aufgabe, einerseits Informationen über den Zustand der Teilnehmergeräte (Fernsprechapparate) und andererseits vom Teilnehmer eingegebene Steuerinformationen (z.B. Wählziffern) im Multiplexbetrieb zur Vermittlungsstelle zu übertragen. Weiterhin müssen sie auf Meldungen von der Vermittlungsstelle her bestimmte Funktionen ausführen.

Jede periphere Einheit wird durch eine Datenvorverarbeitungseinheit in der Vermittlungsstelle ferngesteuert. Diese Datenvorverarbeitungseinheit hat die Aufgabe, die von der Peripherie kommenden Informationen aufzubereiten, bestimmte Routineaufgaben durchzuführen und die daraus folgenden Ergebnisse in Form von Meldungen an den zentralen Steuerrechner weiterzuleiten. Weiter muß sie auf Meldungen des zentralen Rechners hin die peripheren Einheiten zu bestimmten Funktionen veranlassen. Zur Steuerung hat die Datenvorverarbeitungseinheit eine festverdrahtete Mikroprogrammsteuerung und verschiedene Datenspeicher zur Speicherung von Informationen, welche die periphere Einheit betreffen.

Der zentrale Steuerrechner steuert und überwacht das gesamte Vermittlungssystem. Durch die Datenvorverarbeitungseinheit wird der Datenfluß zum zentralen Rechner reduziert und weiterhin der zentrale Rechner von Routineaufgaben befreit und dadurch entlastet.

Die Datenvorverarbeitungseinheit ist jeweils über einen Datenkanal mit dem zentralen Steuerrechner und mit einer peripheren Einheit verbunden. Der Datenaustausch erfolgt über Sende (SR)- und Empfangsregister (ER), die jeweils eine Meldung aufnehmen können. Diese Register stellen die Schnittstelle nach außen dar. Trifft in einem der Empfangsregister (ER2 bzw. ER3) eine Meldung ein, so wird ein zu dieser Meldung spezifisches Mikroprogramm gestartet. Während dessen Ablauf werden in den Datenspeichern der Steuerung Änderungen vorgenommen. Als Ergebnis des Mikroprogrammes werden Anweisungen in Form von Meldungen entweder zur peripheren Einheit oder zum Rechner mittels der Senderegister (SR2 bzw. SR3) übertragen. Entsprechend den unterschiedlichen eintreffenden Meldungen sind unterschiedliche Mikroprogramme in der Steuerung vorhanden.

2.2 Logische Simulation der Datenvorverarbeitungseinheit

Zum Austesten des logischen Ablaufes der verschiedenen Mikroprogramme und ihrer gegenseitigen Beeinflussung durch die Verwendung von Informationen aus den gleichen Datenspeichern wurde ein Simulationsprogramm erstellt /2/. Das Flußdiagramm ist im Bild 2 angegeben. (Siehe Seite 5).

Das Programm gliedert sich in einen Teil für die Schnittstellenversorgung und in einen Teil, der den eigentlichen Simulator darstellt. Die Schnittstellenversorgung dient dazu, Meldungen, die im realen System von der peripheren Einheit oder vom zentralen Steuerrechner kommen, in die Empfangsregister ER2 bzw. ER3 einzutragen und umgekehrt die als Reaktion auf die Eingabe in den Senderegistern SR2 bzw. SR3 erzeugten Meldungen auszugeben. Die Meldungen in den Registern stellen Bitmuster dar und enthalten in der Regel eine Kennung und verschiedene Parameter.

Um die Benutzung des Programmes zu erleichtern, können die Meldungen mit mnemotechnischen Abkürzungen und die verschiedenen Parameter, die neben der Kennung, in der Meldung enthalten sind, als Zahlen ein- und ausgegeben werden. Dadurch erhält man eine leicht überschaubare Dokumentation. Weiter ermöglicht die Schnittstellenversorgung auf Wunsch eine Ausgabe der Inhalte der Datenspeicher der simulierten Datenvorverarbeitungseinheit.

Nach dem Eintragen der eingegebenen Meldung in das entsprechende Empfangsregister wird der eigentliche Simulator angesteuert. Dieser ist als Makroprogrammsimulator realisiert /3/. Der Decodierungsteil entnimmt die in einem der Empfangsregister stehende Meldung und steuert ein zu dieser Meldung spezifisches Unterprogramm an. Dieses Unterprogramm führt dieselben Aufgaben wie das zu simulierende Mikroprogramm aus. Das Ergebnis des Unterprogrammlaufes ergibt dann wieder eine Meldung, welche in das entsprechende Senderegister eingetragen wird. Damit ist die Aufgabe des Simulators beendet und der Schnittstellenversorgungsteil gibt die im simulierten Senderegister stehende Meldung aus.

Zur Untersuchung von Fehlern können falsche Meldungen, wie sie durch Übertragungsstörungen zwischen den Sende- und Empfangsregistern auftreten, eingegeben werden. Weiter kann die Folge der Meldungen, die von der peripheren Einheit kommen und im realen System eine bestimmte Reihenfolge haben, verändert werden oder es können sogar Meldungen unterdrückt werden.

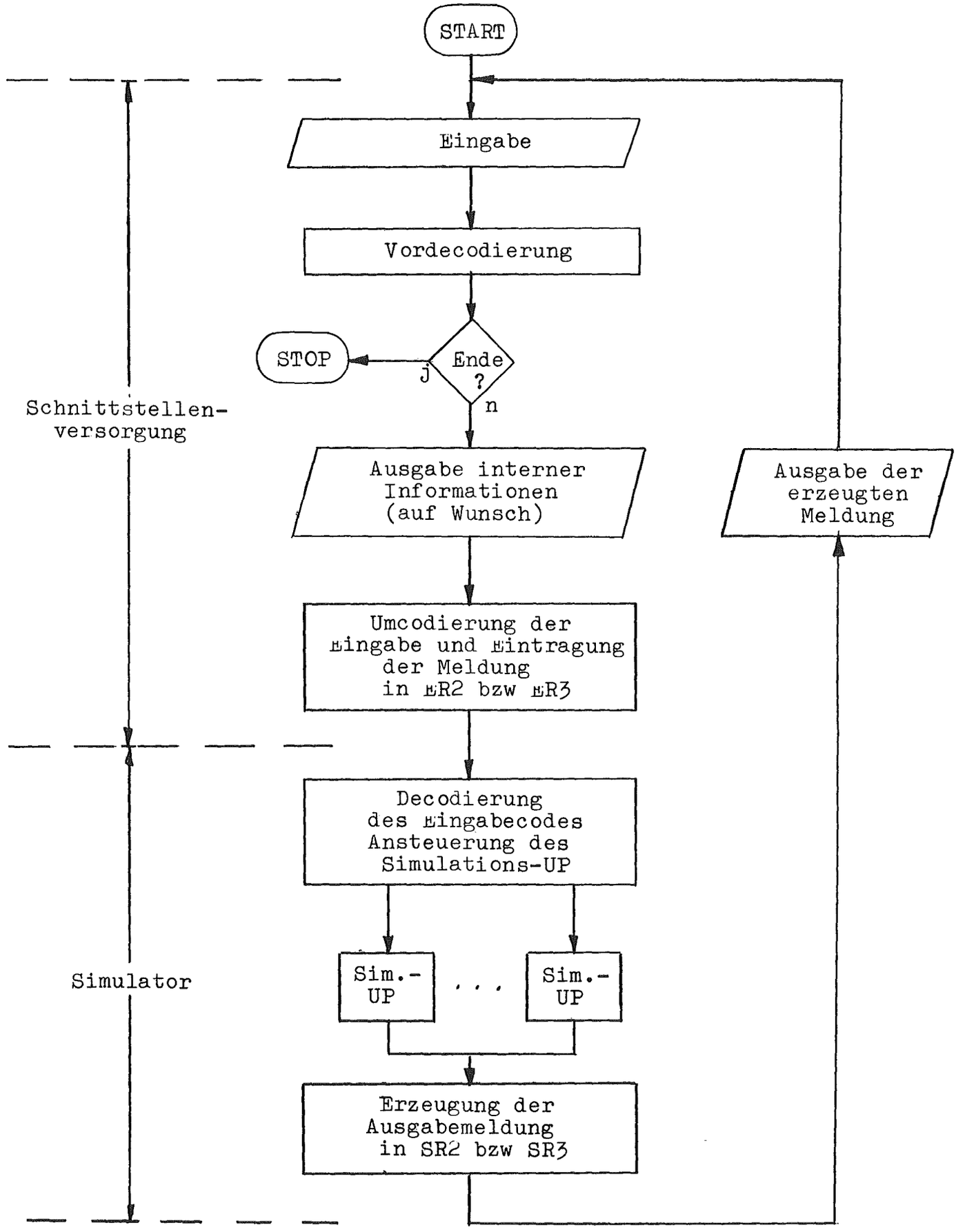


Bild 2 : Flußdiagramm für die logische Simulation.

2.3 Funktionelle Simulation der Datenvorverarbeitungseinheit

Die im Bild 1 dargestellte Steuerung des rechnergesteuerten Fernsprechvermittlungssystems enthält mehrere Datenvorverarbeitungseinheiten. Um das funktionelle Zusammenwirken dieser verschiedenen Einheiten mit dem Rechner auszutesten, ist es nun nicht erforderlich, alle diese Einheiten in Hardware zu realisieren und an den zentralen Steuerrechner anzuschließen. Es ist möglich, die Datenvorverarbeitungseinheit durch einen Simulator nach Kapitel 2.2 im zentralen Steuerrechner zu ersetzen und die periphere Einheit direkt an den zentralen Steuerrechner anzuschließen. Die sich damit ergebende Struktur des Gesamtsystems ist im Bild 3 angegeben.

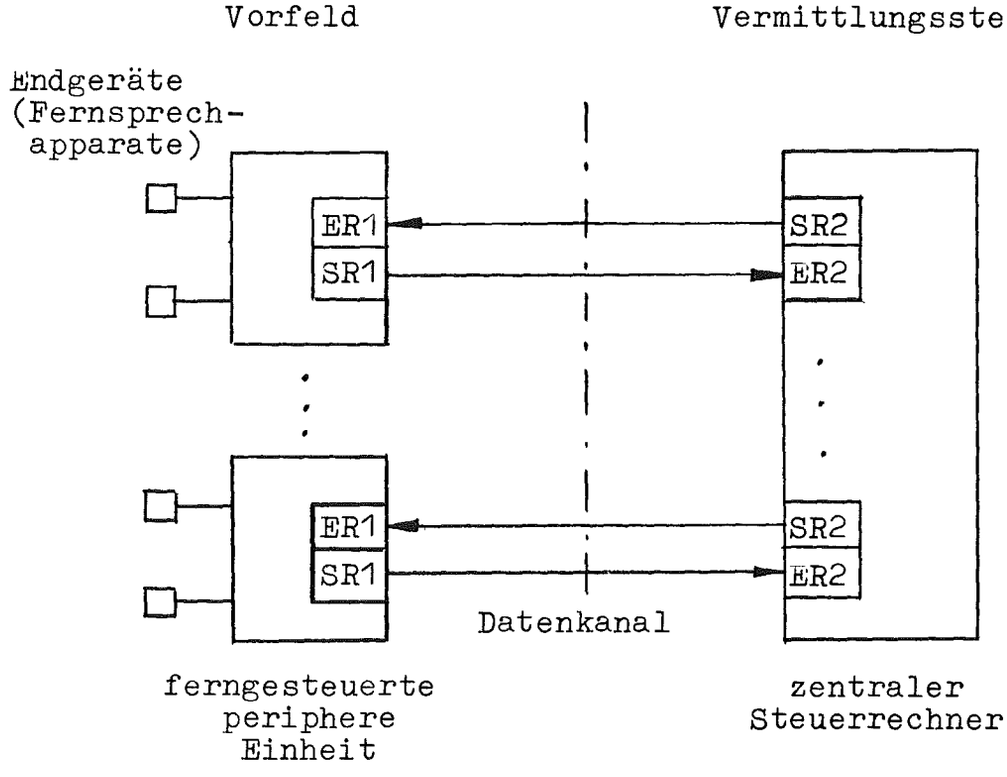


Bild 3 : Struktur der Steuerung der betrachteten Fernsprechvermittlungsstelle mit simulierten Datenvorverarbeitungseinheiten.

Die Ein-Ausgabeschnittstelle des Rechners stellt nun für die periphere Steuereinheit die Schnittstelle der Datenvorverarbeitungseinheit dar. (Sollten die Schnittstelle periphere Einheit/Datenvorverarbeitungseinheit und die Schnittstelle Datenvorverarbeitungseinheit/zentraler Steuerrechner unterschiedlich sein, so ist im zentralen Steuerrechner eine Schnittstellenumsetzung erforderlich, deren Realisierung jedoch oft per Programm möglich ist.

Die einzelnen Simulatoren werden durch ein übergeordnetes Steuerprogramm gesteuert. Dieses übergeordnete Steuerprogramm gibt die Kontrolle an die einzelnen Simulatoren, wenn im realen Empfangsregister des Steuerrechners (Meldung von der peripheren Einheit an die Datenvorverarbeitungseinheit) oder im simulierten Empfangsregister des Simulators eine Meldung steht.

Nach Bild 3 sind alle Datenvorverarbeitungseinheiten durch Simulatoren im zentralen Steuerrechner ersetzt. Es ist natürlich auch ein gemischter Betrieb möglich, d.h. man realisiert eine Datenvorverarbeitungseinheit in Hardware und ersetzt die anderen durch Simulatoren. Dadurch ist es möglich, die Datenvorverarbeitungseinheit im echten Betrieb zu erproben und andererseits ohne allzu großen Aufwand ein Zusammenwirken mehrerer solcher Steuereinheiten auszutesten.

3. Schlußbetrachtung

Die in Punkt 2.2 beschriebene logische Simulation erlaubt die Überprüfung der logischen Abläufe einer Steuereinheit. Eine Untersuchung auf der Schaltungsebene (Aufbau mit verschiedenen digitalen Bausteinen und deren Verschaltung) ist dadurch noch nicht möglich. Man könnte diese Untersuchung jedoch auch mittels Simulation durchführen. Hierzu sind in der Literatur einige Progammsysteme bekannt /4,5/.

Die in Punkt 2.3 beschriebene funktionelle Simulation erlaubt die Überprüfung des Zusammenwirkens mehrerer Steuereinheiten. Dabei ist allerdings zu beachten, daß das reale zeitliche Verhalten des Steuersystems nicht überprüft werden kann, da einerseits durch die Programmlaufzeiten der Simulatoren der zentrale Rechner für seine eigenen Steuerprogramme verzögert wird, und andererseits die Simulatoren sequentiell und nicht parallel arbeiten.

Das Simulationsprogramm wurde in der Assemblersprache des zentralen Steuerrechners geschrieben, um einerseits mit möglichst wenig Speicherplatz auszukommen, und andererseits die Laufzeiten des Simulators klein zu halten.

4. Literatur

/1/ Katzschner, L., Lörcher, W., Weisschuh, H. : On an Experimental Local PCM Switching Network.
Congressbook of the International Zürich Seminar on Integrated Systems for Speech, Video and Data Communications. Zürich, 15.-17.3.72; B6.

/2/ Gärtner, E. : Programm zur Simulation der Konzentratoranschlußschaltung einer PCM-Vermittlungsstelle
Studienarbeit Nr. 393 am Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart.

/3/ Risak, V. : Simulation von Digitalrechnern
Computer Monographien Bd. 6, Carl Hanser Verlag München 1971.

/4/ Pfeuffer, K. : Die Grundlagen des Programms ATEDIS
Lecture Notes in Economics and Mathematical Systems Bd. 78 zur 2. Jahrestagung der Gesellschaft für Informatik, 2.-4. Oktober 1972.

/5/ Clausert, H., Kahlert, I. : Untersuchung digitaler Schaltungen mit dem Programmsystem DICAP, 1. Teil: Simulation
Elektronische Datenverarbeitung 11 (1969), S. 443-451.

DYNAMISCHE SIMULATION ALS ENTSCHEIDUNGSHILFE IN DER BILDUNGSPLANUNG

Hartmut Bossel
Institut für Systemtechnik und Innovationsforschung
75 Karlsruhe-Waldstadt, Breslauerstr. 48

Zusammenfassung

Im Gegensatz zu rein technologischen und ökonomischen Prozessen lassen sich die Konsequenzen gesellschaftspolitischer Entscheidungen bisher nur schwer und mit geringer Verläßlichkeit vorhersagen. Das liegt einmal an der dominierenden Rolle menschlichen Verhaltens in solchen Systemen, zum anderen an der hochgradigen Komplexität ihrer Struktur mit einer großen Anzahl von meist nicht genau meßbaren Parametern und Veränderlichen.
Folglich bleibt die Dynamik des Systems unbekannt und wird vom Entscheidungsträger nur intuitiv abgeschätzt. Gelegentliche Fehlentscheidungen sind unvermeidbar.

In der vorliegenden Arbeit wurde die Eignung systemdynamischer Methoden als Entscheidungshilfe zur Bildungsplanung im Hochschulbereich untersucht. Konkret waren Einflüsse der Gebühren-, Stipendien- und Curriculumpolitik auf Ausfallquoten und Leistung von Hochschulstudenten unterschiedlicher Fähigkeiten und Bedürftigkeit zu untersuchen. Neben finanziellen Größen (Studienkosten, Lebenshaltungskosten, Einkommen verschiedener Art) arbeitet die Simulation mit einer größeren Zahl nicht oder nur schlecht meßbarer Größen (Motivation, Frustration, Leistungsdruck, Bildung, Bildungsdiskrepanz, u. a. m.). Die Ergebnisse zeigen, daß die vorsichtige und sorgfältige dynamische Simulation "weicher" gesellschaftspolitischer Systeme "vernünftige" Resultate liefern und als Entscheidungshilfe dienen kann.

Einführung

Amerikanische Colleges und Universitäten stehen vor dem Problem einer sehr hohen Ausfallquote. Im nationalen Durchschnitt verlassen 53.3 % der Studienanfänger im Laufe der vierjährigen Studienzeit die Hochschulen (1); nur die Hälfte der Abgänger bewirbt sich überhaupt wieder an anderen Hochschulen. Mit diesen Abgängen sind persönliche Härten, soziale Kosten und finanzielle Verluste verbunden für den Studenten, seine Eltern, die Hochschule und den Staat.

Manhat versucht, durch Faktorenanalyse ein Profil des typischen Studienabbrechers zu erstellen (1). Aus diesen Untersuchungen ist bekannt, daß die Durchhaltechancen umso größer sind, je höher die Oberschulnoten des Studenten waren, und je besser er bei Begabungsprüfungen abgeschnitten hat. Weitere signifikant positive Faktoren sind weitgehend gesicherte finanzielle Unterstützung durch die Eltern, Stipendien oder Ersparnisse und die Tatsache, daß der Student ein Mann und außerdem Nichtraucher ist und auf einen höheren akademischen Grad (Magister oder Doktor, nicht nur Diplom) hinstrebt.

Mit diesen Ergebnissen können der College-Administrator und die zuständigen Ausschüsse wenig anfangen, es sei denn, sie ließen nur noch die begabten, männlichen, nicht-rauchenden Kinder reicher Eltern zum Studium zu, um die Ausfallquote gering zu halten. (Diese Praktik ist an den privaten Ivy League Colleges jahrhundertalt.) Für die normale Hochschulverwaltung, die sich ihre Studentenzusammensetzung nicht aussuchen kann, liegt das Problem allerdings auf einer anderen Ebene: es ist zu untersuchen, durch welche bildungspolitischen Entscheidungen sich die Ausfallquote so weit wie möglich herunterschrauben läßt. Hier steht man aber sofort vor einem doppelten Dilemma, das eine intelligente Entscheidung außerordentlich erschwert: weder lassen sich aus zeitlichen, finanziellen und ethischen Gründen entsprechende Experimente durchführen, noch aber lassen sich einfache Ursache-Wirkung-Beziehungen entdecken, die den Studienabbruch bestimmen. Eine einfache Betrachtung zeigt schnell, daß eine Abbruchentscheidung vom Individuum aus einer ganzen Reihe von Gründen, die einzeln oder in Kombination auftreten können, getroffen werden kann: finanzielle Verhältnisse, schlechte Leistungen, mangelnde Motivation, Frustration, um nur die wichtigsten zu nennen. Diese Größen müssen im Entscheidungsprozeß des Bildungsplaners eingebaut sein bevor überhaupt mit relevanten Aussagen gerechnet werden kann. Die Einflüsse dieser (und anderer) Faktoren auf die Abbruchsentscheidung des Studenten, oder einer Menge von Studenten, aber intuitiv richtig abzuschätzen, wie bisher üblich, übersteigt die Fähigkeiten des menschlichen Gehirns. (Es soll trotzdem Leute geben, die sich so etwas zutrauen.) Falls jedoch eine explizite Theorie bestünde (sprich (Gedanken-)Modell), so ließen sich Konsequenzen von Entscheidungen relativ schmerzlos auf Rechenanlagen durchspielen. Diese Überlegungen veranlaßten die Universität

von Kalifornien (Campus Santa Barbara) im Frühjahr 1972 eine Studie zu vergeben, in der mit Hilfe der dynamischen Simulation das Verhalten und der Entscheidungsprozeß des einzelnen Studenten nachempfunden werden sollte. Die Studie wurde mit einem Zeitaufwand von einem Mann-Monat bei einer Gesamtrechenzeit (IBM 360/75) von einer Stunde (einschl. Fehlersuche und Empfindlichkeitsanalyse) durchgeführt. Die Studie gliedert sich in zwei Abschnitte, die hier kurz beschrieben werden. Zunächst mußte ein Modell für das Verhalten des Hochschulstudenten entwickelt werden. Mit diesem Modell wurden dann eine große Zahl von Simulationen durchgeführt, um die Einflüsse verschiedener Faktoren auf das Verhalten des simulierten Studenten festzustellen. Das Modell wurde schließlich dazu benutzt, um Verbesserungsvorschläge für den Vorlesungsplan und die Gebührenordnung zu machen.

Es gibt noch verhältnismäßig wenig Versuche, menschliches Verhalten dynamisch auf Rechenanlagen zu simulieren (andere s. (2) - (9)). Die Methode und die Ergebnisse sollten daher sehr kritisch untersucht werden, bevor sie weite Anwendung finden. Bei aller Vorsicht bin ich allerdings der Ansicht, daß sich mit der Simulation menschlichen Verhaltens ein fruchtbares neues Forschungsgebiet öffnet, das besonders das Werkzeug des Entscheidungsträgers sehr erweitern und zu besseren und menschlicheren Entscheidungen führen wird.

Warum rechnergestützte Simulation menschlichen Mikroverhaltens in einer gegebenen Situation? Es gibt dafür mehrere gute Gründe. Alle scheinen Anstrengungen auf dem Gebiet der Simulation zu rechtfertigen, um ihre Zuverlässigkeit und Brauchbarkeit als Werkzeug für die Verhaltensforschung und zur Entscheidungshilfe zu untersuchen:

(1) Verhaltensexperimente, die über Jahre laufen, sind normalerweise weder möglich, noch ethisch zulässig.

(2) Äußere Bedingungen und die Versuchsobjekte selber ändern sich mit der Zeit in einem Maße, das ein Wiederholen des Experiments mit einem abgeänderten Parameter, aber unter sonst gleichen Bedingungen, nicht zuläßt.

(3) Entscheidungsträger haben selten die Zeit, um auf den Ausgang von Realzeitexperimenten der Verhaltensforschung zu warten.

(4) In Simulationen können unterschiedliche Vorgehensweisen untersucht werden, selbst wenn sie als außergewöhnlich oder extrem betrachtet werden müssen.

(5) Rechnergestützte Simulation benötigt ein präzise formuliertes quantitatives Modell, d.h. die Formulierung einer sehr expliziten Theorie. Im Gegensatz zu einem verbalen oder intuitiven Modell ist das Rechnermodell eindeutig und offen zur Diskussion und Verbesserung.

(6) Die Entwicklung eines Simulationsmodells fördert klares Denken über den Gegenstand der Untersuchung. Im Laufe des Entwicklungsprozesses ergeben sich Wissenslücken, die oft durch einfache Experimente in der Wirklichkeit gefüllt werden können.

(7) Das menschliche Gehirn kann selbst komplizierte Systemstrukturen vom Detail her ausgezeichnet erkennen und beschreiben. Es kann Parameter und Veränderliche und die Abhängigkeiten zwischen Veränderlichen feststellen. Diese Aufgaben können Rechenanlagen heute noch nicht übernehmen. Die menschlichen Fähigkeiten sind allerdings äußerst beschränkt und unzuverlässig, sobald es um die genaue Vorhersage des dynamischen Verhaltens komplexer Systeme geht, d.h. im Grunde um die Lösung simultaner nicht-linearer Differentialgleichungen. Hier muß der Rechner einspringen. Es ist zu erwarten, daß menschliches Erkennungsvermögen, um die untrügliche Rechenkapazität des Computers erweitert, zu besseren Ergebnissen führen muß.

Allgemeiner Ansatz

Betrachten wir das Problem vorzeitigen Studienabbruchs, so ist es eingebettet in eine große Menge von Individuen unterschiedlicher Begabungen, Interessen, Motivationen und finanzieller Situationen, die in einer Wechselwirkung mit einer Hochschule stehen, die durch gewisse Verwaltungsmerkmale, Vorlesungspläne, Lehrkörper und Fakultäten, Unkosten und akademisches Klima gekennzeichnet ist. Diese Wechselwirkung erscheint als ein dynamischer Prozeß, der sich (bei den amerikanischen Hochschulen, die wir hier betrachten) über vier Jahre erstreckt und - vielleicht - mit der Verleihung des Baccalaurus-Diploms endet. Dieser komplexe dynamische Prozeß umfaßt nicht nur quantifizierbare Größen (finanzielle Größen), sondern auch andere, die schwer quantifizierbar sind (Notendruck, Frustration, usw.). Der Prozeß hängt außerdem völlig von individuellen Entscheidungen und Faktoren ab. Es ist offensichtlich, daß eine Simulation des Einzelverhaltens notwendig ist, um überhaupt einen gewissen Grad von Realitätsbezogenheit zu erreichen. Das aggregierte Verhalten einer Studentenpopulation könnte dann durch statistische Untersuchungen an einer großen Zahl probabilistisch simulierter Studenten studiert werden. Aus Gründen der Wirtschaftlichkeit wurde dieser Weg hier allerdings nicht begangen.

Damit stellt sich hier die Aufgabe der zeitabhängigen (dynamischen) Simulation eines komplexen deterministischen Systems meist kontinuierlicher Veränderlicher, wobei aber nur ein Teil der Veränderlichen und Parameter ohne weiteres und eindeutig quantifiziert werden können. Außerdem ist die Struktur des Systems hochgradig verschwommen. Selbst wenn sich eine Systemstruktur herausdestillieren ließe, so ist Eindeutigkeit damit noch nicht bewiesen.

Damit ergibt sich die Frage, ob in diesem Fall eine Simulation überhaupt sinnvolle Ergebnisse liefern kann. Es scheint, daß diese Frage aus zwei Gründen bejaht werden kann: Erstens stellt sich bei Erfassung verschwommener Systemstrukturen immer wieder heraus, daß selbst unabhängig voneinander (für den gleichen Zweck) entwickelte Systemstrukturen sich meist in wesentlichen Details nicht voneinander unterscheiden. Unterschiede können meist durch Diskussionen und weitere Untersuchungen beseitigt werden. Mit anderen Worten, selbst aus verschwommenen Systemen läßt sich iterativ im allgemeinen eine Systemstruktur herauslösen, die die wesentlichen Merkmale des Systems wiedergibt. Zweitens erfüllen die nicht oder schlecht quantifizierbaren Größen (Motivation, Frustration usw.) in einer Verhaltenssimulation keine Erhaltungssätze wie physikalische Größen. Sie lassen sich also auf relativen Skalen angeben und verwenden, wobei sich gewisse Punkte (Minima, Maxima, Nulldurchgänge usw.) oft recht leicht bestimmen lassen. Aus diesen beiden Punkten ergibt sich eine weitere wichtige Folgerung: Das dynamische Verhalten des Systems wird qualitativ richtig beschrieben, falls die funktionalen Abhängigkeiten des Prozesses kontinuierlich sind, falls diese Abhängigkeiten und die Parameter ungefähr richtig beschrieben worden sind, und falls weiter die Empfindlichkeitsuntersuchung des Modells über das mögliche Parameterfeld kein abnormales Verhalten zeigt.

Diese Beobachtung ist wichtig für die Simulation verschwommener Systeme: Exakte Lösungen können nicht erwartet werden, aber eine qualitativ richtige Beschreibung dynamischen Verhaltens ist wahrscheinlich. Bei der Entscheidungsfindung aber interessiert das dynamische Verhalten besonders.

Entwicklung des Verhaltensmodells

Die aufeinanderfolgenden Schritte bei der Entwicklung eines Simulationsmodells unterscheiden sich kaum von Fallzu Fall. Sie seien hier nur der Vollständigkeit halber erwähnt in der Reihenfolge ihrer Bearbeitung:

Sammlung von verbaler Information
Beschaffung statistischer und anderer Daten
Tiefergehende Information durch Interviews
Anlegen einer Datenbasis
Ermittlung der Zusammenhänge (Netzplan)
Aufstellung der Systemstruktur mit Identifizierung der Bestände, Ströme, Hilfveränderlichen usw.
Identifizierung der Parameter
Feststellung der Abhängigkeiten zwischen Veränderlichen
Programmierung
Probeläufe und Fehlersuche
Grobeinstellung
Qualitative Validierung und Experimente über den gesamten Parameterbereich und mögliche Anfangsbedingungen
Quantitative Validierung durch statistische, historische oder gleichzeitig ermittelte Daten
Simulation der Wirklichkeit
Untersuchung verschiedener Vorgehensweisen
Auffinden optimaler Vorgehensweisen

Im allgemeinen werden die späteren Schritte kleinere oder größere Abänderungen früherer Schritte und die Beschaffung zusätzlicher Informationen notwendig machen.

Veränderliche, Parameter, Abhängigkeiten

Es wurde von Anfang an eine möglichst vollständige Erfassung aller Einflußgrößen angestrebt. Zu diesem Zweck wurden zunächst (fiktive) Fallprotokolle aufgestellt, die in den letzten Jahren der Oberschule begannen und mit der Graduierung endeten. In diesen Protokollen wurde der Versuch gemacht, alle Faktoren und Prozesse zu erfassen, die überhaupt den leisesten Einfluß auf studentisches Verhalten während

dieser Jahre haben konnten. Diese Protokolle wurden von anderen durchgesehen und ergänzt um sicherzustellen, daß keine Einflüsse übersehen wurden.

Mit Hilfe der Protokolle wurden dann ausführliche Listen aller Faktoren, Einflüsse, Veränderlichen und Parameter aufgestellt. Jede Eintragung wurde dann einzeln gründlich betrachtet, um offensichtlich nicht vorhandene Einflüsse gleich auszuschalten, synonyme Eintragungen (Motivation, Antrieb, Streben usw.) und Gegensätzlichkeiten (Enthusiasmus/Frustration) in jeweils einer Veränderlichen zu kombinieren, oder andere aufzuspalten (Frustration durch finanziellen Druck, Notendruck oder Bildungslücke). Aus der bereinigten Liste wurden dann weiter die Veränderlichen, die konstanten Parameter und die funktionalen Abhängigkeiten zwischen Veränderlichen herausgesucht. Die hier eingeführten Bezeichnungen decken sich mit denen des Rechenprogramms und der Diagramme. Eine ausführliche Beschreibung des Gesamtprojekts findet sich in (10).

Es ist zu beachten, daß die Simulation für amerikanische Hochschulverhältnisse entwickelt wurde, die z.T. von den deutschen erheblich differieren.

Es wurden folgende Hauptveränderlichen identifiziert und in die Simulation aufgenommen:

Notendiskrepanz PD (zwischen tatsächlichem und erwünschtem Notendurchschnitt)
Notendruck P
Lernmotivation AM
Lernleistung ohne Ablenkung UP
Tatsächliche Lernleistung AP
Erreichte Lernbildung AA
Notendurchschnitt GPA
Relevante Bildung RA
Erwünschte Bildungsaktivität außerhalb des Lehrplans DE
Tatsächliche Bildungsaktivität außerhalb des Lehrplans AE
Erreichte Bildung außerhalb des Lehrplans PA
Erreichte Gesamtbildung TA
Lernzielfunktion EG
Bildungsdiskrepanz ED
Frustration wegen Bildungsdiskrepanz FED
Frustration wegen Notendruck FP
Frustration wegen finanziellem Druck FINFRUS
Gesamtfrustration F
Verfügbare Geldmittel BALANCE
Finanzieller Druck FINPRS
Einkommen durch Teilzeitarbeit und Sommerarbeit PART JOB, SUM JOB, EARNING
Unterstützung durch Eltern oder andere SUPPORT, AID
Stipendien SKOLR, SCHOL

Anleihen LOAN, DEBT
Ausgaben (Lebenshaltung und Gebühren) LIVING, FEES, COST

Die Eigenschaften des einzelnen Studenten werden durch eine größere Zahl von Parametern beschrieben:
Notenziel PG
Bildungsziel EDFAC
Selbstmotivierung CAMS
Begabung SATVM
Ablenkung CAPX
Wochenstunden CUC
Frustrationsschwelle FRUSTHR
Schwelle der Bildungsdiskrepanz EDTHR

Die folgenden Parameter beschreiben die finanzielle Situation des Studenten:
Ersparnisse SAVINGS
Wechsel / Quartal CHECK
Nettoeinkommen durch Sommerarbeit SUMMER
Einkommen durch Teilzeitarbeit / Woche EARN
Einkommen aus Anleihen / Woche LOANC
Einkommen aus Stipendien / Woche SKOLSHP
Lebenshaltungskosten / Woche LIVING
Gebühren / Quartal FEEC

Die Hochschule selbst wird durch folgende Parameter beschrieben:
Motivierung durch die Hochschule TAMI
Leistungseffekt der Hochschule TUPI
Relevanzzahl des Lehrplans TFRAC
Bildungsklima außerhalb des Lehrplans ENVIR

Vielleicht wichtigste Komponenten der Simulation sind die funktionalen Abhängigkeiten zwischen Veränderlichen. Fast alle diese Abhängigkeiten sind Funktionen der persönlichen Eigenschaften des Studenten und sollten individuell durch psychologische Tests ermittelt werden. (Solche Tests scheinen bisher nicht zu existieren, lassen sich aber wahrscheinlich in den meisten Fällen aufstellen.) Kurze Überlegungen zeigen, daß diese Abhängigkeiten in fast allen Fällen nichtlinear sein müssen. Da entsprechende psychologische Daten bisher nicht vorliegen, wurden die verschiedenen Abhängigkeiten geschätzt und für alle Studententypen unverändert verwendet. Die hauptsächlichen Abhängigkeiten waren:
Druck wegen Notendiskrepanz TP
Lernmotivierung durch Notendruck TAMP
Motivationsverlust durch Frustration CAMP
Unabgelenkte Lernleistung durch Motivation TUPM
Unabgelenkte Lernleistung durch Begabung UPA
Lernleistungsverlust durch Bildungsaktivität außerhalb des Lehrplans CAPE
Lernleistungsverlust durch Teilzeitarbeit APDROP

Erwünschte Bildungsaktivität außerhalb des Lehrplans wegen Bildungsdiskrepanz TDE
Abfall der Bildungsaktivität außerhalb des Lehrplans wegen Notendruck TPRES
Abfall der Bildungsaktivität außerhalb des Lehrplans wegen Frustration TFRUS
Bildungsfrustration wegen Bildungsdiskrepanz TFED
Bereitschaft zum Studienabbruch TREADY

Als Beispiel für die angenommenen funktionalen Zusammenhänge zeigt Bild 1 die Abhängigkeit der Lernmotivation AMP vom Notendruck AP; die anderen Abhängigkeiten wurden durch ähnliche Gedankengänge ermittelt und sind aus dem beigefügten Programm entnehmbar. Hierbei entsprechen 100 % Notendruck einer Notendiskrepanz von 1 Punkt. Im hier betrachteten Notensystem entsprechen 4 Punkte der Note "A" (sehr gut), 3 Punkte: "B" (gut), 2 Punkte: "C" (befriedigend), 1 Punkt: "D" (mangelhaft), 0 Punkte: "F" (ungenügend).

Der Verlauf der Funktion AMP über P stellte sich als besonders kritisch heraus; er mußte während des Abstimmungsprozesses auf die gegenwärtige Form gebracht werden, um realistische Ergebnisse zu erzielen. Wenn der Notendruck auf 50 % ansteigt, steigt gleichzeitig die Motivation steil auf 100 %. Bei weiterem Anstieg des Notendrucks bleibt die Motivation unverändert auf ihrem Maximalwert, um wieder steil auf Null abzusinken, sobald der Notendruck zu stark wird und der Student aufgibt. Wenn die Notenleistung besser ist als erwartet (negativer Notendruck), wird die Motivation nur geringfügig negativ.

Systemstruktur

Die Entwicklung der Systemstruktur besteht im wesentlichen darin, die Veränderlichen sinnvoll zu verbinden, wobei Hilfsveränderliche sowie Strom- und Bestandsvariable zu unterscheiden sind. Selbst für ein relativ einfaches System wie das gegenwärtige wird die Struktur sehr komplex. Es soll daher hier nur ein stark vereinfachtes Schaltbild besprochen werden. (In Bild 2 finden sich die Sektoren, die das Verhalten beschreiben. Der finanzielle Teil erscheint in Bild 3.) Kreise bedeuten Hilfsveränderliche, Kästen eine Integration (d.h. eine Kombination von Strom- und Bestandsveränderlichen). Kleinere Kästen mit Balken bedeuten Verzögerungsglieder.

Der Verhaltensteil besteht aus vier vernetzten Sektoren: dem Lernleistungssektor P-AM-AP-AA-GPA-PD-P; dem Bildungssektor RA-TA-ED-AE-PA-TA; dem Frustrationssektor FP-ED-FINFRUS-F; und dem Entscheidungssektor FRUSDEC-EDDEC-GPADEC-FINDEC-DEC. Der finanzielle Sektor besteht aus den Hauptveränderlichen LIVING FEES, PART JOB, SUM JOB, LOAN, SKOLR, SUPPORT, und BALANCE.

Der Lernleistungssektor P-AM-AP-AA-GPA-PD-P stellt die wichtigste Rückkopplungsschleife des Systems dar. Der Student nimmt eine Notendiskrepanz PD zwischen seinem Notendurchschnitt GPA und seinem Notenziel PD wahr. Daraus entsteht ein entsprechender Notendruck P und eine entsprechende Lernmotivation AM. Diese Motivation wird u. U. durch Frustration F verringert oder durch Enthusiasmus (-F) erhöht. Als Folge der Motivation AM entsteht eine gewisse Lernleistung AP, die weiter von persönlichen Parametern, von Teilzeitarbeit und anderen Tätigkeiten außerhalb des Lehrplans abhängt, Integration der Lernleistung AP über die Zeit führt zu einer Lernbildung AA, gemessen in den Einheiten (Notenpunkte x Zeit). Der jeweilige Notendurchschnitt GPA folgt nach Division durch die Zeit. Die Zeit zählt in Wochen, mit 12 Wochen je Quartal, drei Quartalen je Term, und vier Termen in vier Studienjahren. Die Simulation läuft deshalb über 144 Wochen.

Der Bildungssektor RA-TA-ED-AE-PA-TA würde nur eine geringe Rolle spielen, falls die Bildung über den Lehrplan (im Lernleistungssektor) mit den Bildungszielen des Studenten übereinstimmen würde. Gewöhnlich ist dies nicht der Fall, und nur ein gewisser Bruchteil FRAC des Lehrplanmaterials wird als relevant empfunden. Daraus ergibt sich (in der Auffassung des Studenten) eine relevante Lernbildung RA. Diese relevante Lernbildung RA kombiniert sich mit der persönlichen Bildung PA (die aus Bildungstätigkeiten außerhalb des Lehrplans AE entsteht) zur Gesamtbildung TA. Der Student verfolgt fortwährend die Entwicklung seiner Gesamtbildung TA und vergleicht sie mit seinem Bildungsziel EDFAC. Im allgemeinen wird er eine Bildungsdiskrepanz ED feststellen. Nach einer gewissen Verzögerung wird er daraufhin versuchen, sich außerhalb des Lehrplans durch zusätzliche Bildungsaktivität AE zusätzliche Bildung zu verschaffen. Die tatsächliche Höhe der Bildungsleistung außerhalb des Lehrplans wird allerdings durch den Notendruck P und durch das Frustrationsniveau F bestimmt.

Der Frustrationssektor FP-ED-FINFRUS-F kombiniert die aus fortwährendem Notendruck entstehende Frustration FP mit der finanziellen Frustration FINFRUS und der Frustration FED, die sich aus fortwährender (geglätteter) Bildungsdiskrepanz ED ergibt.

Im Entscheidungssektor FRUSDEC-EDDEC-GPADEC-FINDEC-DEC werden die jeweiligen Größen verschiedener Schlüsselveränderlicher, die den Studienabbruch bestimmen (Frustration, F, Bildungsdiskrepanz ED, Notendurchschnitt GPA, Guthaben BALANCE) mit vorgeschriebenen Schwellwerten (verschieden für verschiedene Studenten) verglichen. Der Student bricht das Studium ab, wenn diese Schwellwerte überschritten werden. Frustrationsschwelle und Bildungsdiskrepanzschwelle sind persönliche Parameter und von Student zu Student verschieden. Eine zeitabhängige Funktion READY beschreibt die Abbruchbereitschaft. Sie hängt ab vom Term, in dem sich der Student befindet. Während sie im ersten und letzten Jahr am niedrigsten

ist, ist sie im zweiten Jahr am höchsten (zu diesem Zeitpunkt ist der finanzielle Verlust für den Studenten noch relativ gering, sollte er einsehen, daß er einen Fehler gemacht hat).

Im finanziellen Sektor (Bild 3) werden die Einnahmen (aus Teilzeitarbeit PARTJOB, Sommerarbeit SUMJOB, Anleihen LOAN, Stipendien SKOLR und elterlicher Unterstützung SUPPORT) und die Ausgaben (Lebenshaltungskosten LIVING, Gebühren FEES) zusammengetragen. Daraus werden die jeweilig verfügbaren Geldmittel BALANCE und Gesamteinkommen und Gesamtunkosten berechnet. (Gesamtkosten COST, Gesamteinnahmen EARNINGS, Gesamtschuld DEBT, gesamte Stipendieneinnahmen SCHOL und gesamte (elterliche) Unterstützung AID.) Außerdem wird in diesem Sektor bestimmt, ob aufgrund des Notendurchschnitts Stipendien oder Anleihen erhalten werden können. Falls der Student eine Teilzeitarbeit übernimmt (sobald sein Guthaben BALANCE unter ein Minimum sinkt), folgt ein entsprechender Abfall in seiner Lernleistung (APDROP).

Die programmierte Systemstruktur ist komplexer als aus dem einfachen Schema ersichtlich. Es ist nicht sinnvoll, sie hier im einzelnen durchzusprechen. Ein Teilbeispiel möge genügen. Bild 4 zeigt den Lernleistungssektor.

Wir beschränken uns auf den Subsektor AMS, TAMI, AM, AMF in der Mitte des Bildes, der wie folgt zu lesen ist:
Die Hilfsveränderliche "Lernmotivation AM" wird berechnet aus der Motivation durch Notendruck (AMP), der Selbstmotivierung des Studenten (AMS), der Motivierung durch die Hochschule TAMI (die als Tafelfunktion vorliegt, um unterschiedliche Motivation in verschiedenen Studienjahren zu berücksichtigen), und einem Motivationsabfall durch Frustration AMF. Die Lernmotivation AM stellt dann eine Eingabe für die Berechnung der nächsten Hilfsveränderlichen UPM (unabgelenkte Lernleistung) dar. Die genauen Beziehungen folgen aus dem Rechenprogramm, das in DYNAMO programmiert wurde (s. Anhang).

Mit dem Vorliegen der Systemstruktur und ihrer Übersetzung in ein Rechenprogramm sind die Voraussetzungen für die Simulation gegeben. Es bleibt noch die Beschreibung des einzelnen Studenten und der Hochschule durch entsprechende Parameter.

Simulationen

Die Abstimmung, qualitative Überprüfung (Validierung) und spätere Simulationen des Modells wurden alle mit den selben 5 Studenten- und drei Hochschulprofilen vorgenommen. Diese Profile überstrichen das ganze mögliche Typenfeld (s. Bild 5)

Die Begabungen reichten von "hochbegabt" bis "schlechtbegabt"; das Bildungsziel konnte darin bestehen, einen gewissen Notendurchschnitt

zu halten, um diplomiert werden zu können, oder so viel Bildung wie möglich zu erlangen, ohne auf den Notendurchschnitt zu achten. Manche Studenten waren finanziell sorgenfrei, andere hatten erhebliche Schwierigkeiten, die erforderlichen Geldmittel zu beschaffen. Die Abbruchsschwellwerte für Frustration und Bildungsdiskrepanz unterschieden sich je nach dem Bildungsziel.

Drei Hochschultypen wurden untersucht: Eine "ausgezeichnete", eine "durchschnittliche" und eine "schlechte" Hochschule. Erhebliche Unterschiede bestanden hier in der Motivierung der Studenten durch die Hochschule, in der Anlei tung zu besonderer Leistung, in der Relevanzzahl des Lehrplans, und im Bildungsklima außerhalb des Lehrplans.

Aus der Zusammenstellung von Studenten- und Hochschultypen ergaben sich jeweils 15 Simulationsläufe (zusammen 1 min auf IBM 360/75). Die Ergebnisse dieser Läufe wurden mit der Wirklichkeit intuitiv verglichen. Aus diesen Abstimmungsläufen ergaben sich einige, meist geringfügige Änderungen von Parametern und funktionalen Abhängigkeiten.

Es ist noch zu bemerken, daß alle notwendigen Anfangswerte für die Simulation mit hoher Sicherheit bestimmt werden können; einzige Ausnahme ist die anfängliche Notenerwartung IGPA des Studenten. Dieser Punkt wurde deshalb gesondert untersucht (wobei IGPA = 0 bis 4). Es ergaben sich größere Abweichungen nur während des ersten Studienjahres. Danach stellten sich fast identische Ergebnisse ein. Diese Simulationen schienen weitgehend die Anpassungsprobleme des Studenten im ersten Studienjahr zu duplizieren.

Bild 6 und Bild 7 zeigen Ergebnisse eines Simulationslaufs (ein "durchschnittlicher Student D in einer "durchschnittlichen" Hochschule). Die oberen Kurven zeigen Bildungsdiskrepanz ED, Notendurchschnitt GPA und Lernleistung AP. Die Bildungsdiskrepanz ändert sich kaum und bleibt um rund 40 %. Die Lernleistung sinkt langsam ab; starke Abfälle treten jeweils am Ende jeden Terms auf, wo dem Studenten das Geld ausgegangen ist, und er eine Teilzeitarbeit übernehmen muß, die seine Lernleistung einschränkt. Trotz dieser Schwingungen kann der Student einen fast konstanten Notendurchschnitt von etwa 2.4 halten (4 = A = beste Leistung, 0 = F = schlechteste Leistung). Der Treppencharakter der GPA-Kurve ergibt sich aus der Feststellung dieses Durchschnitts nach jedem Quartal.

Die mittleren Kurven zeigen die verschiedenen Bildungskomponenten. Bei gleichförmigem Bildungszuwachs wären die Kurven linear. Falls der Student alle seine Bildung durch den vorgeschriebenen Vorlesungsplan bezieht, und falls alle diese Bildung von ihm als relevant betrachtet wird, so würde sich am Ende des Studiums eine Gesamtbildung von 100 % ergeben. Im vorliegenden Falle jedoch besteht eine erhebliche Bildungsdiskrepanz ED, und der Student versucht, sie zumindest teilweise durch Bildung außerhalb des Lehrplans abzudecken.

Die unteren Kurven zeigen die Frustrationskomponenten aus der finanziellen Lage, der Bildungsdiskrepanz (FED), und aus dem fortwährenden Notendruck (FP); sowie den Notendruck P. Während Frustration wegen der Bildungsdiskrepanz etwa konstant bleibt, verstärkt sich die finanzielle Frustration wegen der schlechten finanziellen Situation ständig. Damit steigt auch die Gesamtfrustration, und vorzeitiger Studienabbruch aus diesem Grunde wäre möglich, wenn die Frustrationsschwelle entsprechend niedrig läge.

Die finanzielle Situation ergibt sich aus den Kurven in Bild 7. Der Student beginnt sein Studium mit anfänglichen Ersparnissen und Unterstützung der Eltern. Dieses Geld wird langsam durch Lebenshaltungskosten und Studiengebühren verbraucht. Vor Ende jeden Terms muß er eine Teilzeitarbeit übernehmen, um Schulden zu vermeiden; außerdem arbeitet er während des Sommers. Die Kurven zeigen das jeweilige Guthaben BALANCE, die Gesamtkosten COST, die Gesamteinnahmen durch (elterliche) Unterstützung und die Gesamteinnahmen aus Arbeit.

Aus der Vielzahl der Veränderlichen, die das Rechenprogramm ausdruckt, erscheinen die Lernleistung AP, der Notendurchschnitt GPA, die Gesamtbildung TA und die Gesamtfrustration F als die wichtigsten. Ihnen wurde in den weiteren Untersuchungen besondere Aufmerksamkeit geschenkt.

Das Schema in Bild 8 zeigt eine Übersicht über die 15 zur Abstimmung des Modells benutzten Simulationsfälle. Die Ergebnisse entsprechen den intuitiven Erwartungen. Sie können hier im einzelnen nicht besprochen werden.

Weitere Untersuchungen

Nachdem sich aus der Modellabstimmung Simulationsläufe ergeben hatten, die zumindestens intuitiv als richtig erschienen, wurde das Modell für systematischere Untersuchung einzelner Faktoren eingesetzt. So wurde der Wechsel des durchschnittlichen Studenten so weit aufgebessert, daß Geldsorgen nicht mehr auftraten. Daraufhin wurden einzeln folgende Parameter geändert, um deren Einflüsse auf den Simulationslauf zu untersuchen:

Begabung des Studenten SATVM
Notenziel PG
Bildungsziel EDFAC
Relevanzzahl des Lehrplans FRAC
Bildungsklima außerhalb des Lehrplans ENVIR

Es würde ebenfalls zu weit führen, die Ergebnisse hier im einzelnen zu besprechen; sie werdem am Schluß summarisch zusammengefaßt werden. Es ergaben sich wieder Resultate, die mit faktischen

Beobachtungen oder intuitiven Einsichten zusammenfallen, und die dadurch das Vertrauen in das Modell bestärken. Mit dieser Rückendeckung wurden dann in zwei weiteren Schritten die Auswirkungen unterschiedlicher Lehrplangestaltung und anderer Gebührenordnungen untersucht.

Einfluß der Lehrplangestaltung: Der Student wird am besten und freudigsten lernen, wenn ihm seine Vorlesungen und Übungen für seine Bildungsziele relevant erscheinen. Es ist allerdings klar, daß dieses Lehrideal nur bis zu einem gewissen Grade verwirklicht werden kann, da gewisse gesellschaftliche Ziele oder Erfordernisse (studium generale, breite Ausbildung, organisatorische Hindernisse, Mangel an Lehrkräften oder Lehrmitteln) einer hundertprozentigen Erfüllung entgegenstehen. Unter diesen Umständen ist zu fragen, wie die Relevanzverteilung des Lehrplans über die Studienzeit optimal aussehen sollte, unter der Annahme, daß eine gegebene Menge für den Studenten relevanten und nicht relevanten Lehrstoffs zu vermitteln ist.

Als Beispiel sei angenommen, daß eine gegebene Lehrstoffmenge, über die vier Studienjahre verteilt, vom Studenten im ganzen als zu zwei Dritteln relevant empfunden wird. Es sei weiter angenommen, daß derselbe Lehrstoff bei gleichbleibender Gesamtrelevanzzahl von FRAC = 0,67 auf drei verschiedene Arten dargeboten werden kann: (1) gleichbleibende Relevanzzahl 0.67; (2) ansteigende Relevanzzahl (linearer Anstieg von 0.33 auf 1.0); (3) abfallende Relevanzzahl (linearer Abfall von 1.0 auf 0.33). Praktisch ausgedrückt würde das etwa bedeuten (1) gleichbleibende Mischung von allgemeiner und spezialisierter (für den Studenten relevanter) Ausbildung; (2) allgemeine Ausbildung zuerst, Spezialisierung später; (3) Spezialisierung zuerst, allgemeine Ausbildung später. Die Frage: welche Vorgehensweise wird bessere Resultate zeigen? Diese Frage ist intuitiv sehr schwer und nur mit geringer Sicherheit zu beantworten.

Bild 9 zeigt die Ergebnisse einer entsprechenden Simulation (Student C mit finanziellen Schwierigkeiten am Ende des Terms). Im Falle der ansteigenden Relevanz unterscheidet sich die Lernleistung nicht wesentlich von der für konstante Relevanz. Die Lernleistung ist allerdings wesentlich höher für den Fall der fallenden Relevanz über den Zeitraum der ersten drei Studienjahre. Der endgültige Notendurchschnitt ist entsprechend höher. Bei der Frustrationsverteilung ergeben sich noch größere Unterschiede; der Lehrplan mit der ansteigenden Relevanz zeigt von Anfang an hohe Frustration, während beim Lehrplan mit fallender Relevanz immerhin über die ersten zweieinhalb Jahre Enthusiasmus herrscht.

Aus dieser Simulation ist zu schließen, daß die Abbruchquote wahrscheinlich wesentlich verringert werden könnte durch Einführung relevanten Lehrstoffs in schon früh spezialisierten Lehrplänen

(problemorientiertes, projektorientiertes oder laufbahnorientiertes Studium). Später müßte der Lehrstoff dann weiter und breiter werden. Aus pädagogischen Beobachtungen ist zu schließen, daß sich eine Mehrzahl von Studenten die Verbreiterung selber suchen wird, womit die Relevanzzahl über die angenommene Zahl steigen würde, und die Ergebnisse noch verbessert würden.

Einfluß der Gebührenordnung: Der Begriff "Finanzielle Schwierigkeiten" rangiert als Grund für Studienabbruch an amerikanischen Hochschulen an erster Stelle. Studiengebühren werden an allen (staatlichen und privaten) Hochschulen der USA erhoben (außer Community Colleges). Sie können sich von mehreren hundert auf mehrere tausend Dollar im Jahr belaufen.

An der Universität von Kalifornien wird von kalifornischen Bürgern eine Gebühr von 225 Dollar pro Quartal verlangt. Im Vergleich zu den Lebenshaltungskosten (etwa Dollar 2000/Jahr) und den Zuschüssen des Staates zum Universitätsbetrieb (ebenfalls etwa Dollar 2000/Jahr) ist die Summe von Dollar 675/Jahr vergleichsweise gering, aber trotzdem die Ursache bitterer Beschwerden.

Bild 10 zeigt, wie sich Lernleistung und Frustration des durchschnittlichen Studenten C ändern würden, wenn die Studiengebühr wegfiele. Der endgültige Notendurchschnitt GPA würde erheblich steigen, und die zumeist durch finanzielle Sorgen verursachte starke Frustration in Enthusiasmus umschlagen. Es ist zu erwarten, daß durch diese Maßnahme die Abbruchquote gleichfalls erheblich verringert werden könnte. Auch in diesem Fall ist durch intuitives Urteil nur eine sehr unsichere Aussage zu gewinnen.

Übersicht über die Simulationsergebnisse

Das Schema in Bild 11 faßt die Ergebnisse der verschiedenen Simulationen sehr grob zusammen. Positive Effektessind durch ein Pluszeichen, negative durch ein Minuszeichen, unwesentliche Effekte durch eine Null angedeutet. Da ein hohes Maß an Frustration bzw. geringe Leistung oder hohe Bildungsdiskrepanz zu vorzeitigem Studienabbruch führen können, so ist anzunehmen, daß sich die Abbruchsquoten durch folgende Faktoren verringern lassen:

hohe Qualität der Hochschule
hohe Begabung des Studenten
hohe Relevanzzahl des Lehrplans
projekt-, problem- oder laufbahnorientierter Studienaufbau (zeitlich abfallende Relevanzzahl des Lehrplans)
gute finanzielle Unterstützung
Wegfall der Studiengebühren

Dagegen erhöhen
ein hohes Notenziel und
ein hohes Bildungsziel
des Studenten wegen höherer Frustration seine Abbruchchancen. Diese Ergebnisse mögen trivial erscheinen, doch scheinen sie immerhin anzudeuten, daß das Modell "vernünftige" Ergebnisse liefert und wahrscheinlich für die Bearbeitung einer Reihe von Fragen sinnvoll eingesetzt werden kann (Es ist zu berücksichtigen, daß das Modell in seiner jetzigen Form auf amerikanische Verhältnisse zugeschnitten ist.).

Abschluß

Die detaillierte verbale Beschreibung des Verhaltens von Studenten in amerikanischen Vier-Jahres-Hochschulen hat zu einem expliziten Strukturmodell studentischen Verhaltens, zur Quantifizierung von Parametern und funktionalen Abhängigkeiten und zur numerischen Simulation des Zeitverhaltens einer Zahl "typischer" Studenten in "typischen" Hochschulen geführt. Außerdem wurden die Einflüsse bestimmter Parameter in Einzelheiten untersucht, und die Konsequenzen bestimmten Lehrplanaufbaus und gewisser Gebührenordnungen verfolgt.

Die Ergebnisse einer Empfindlichkeitsuntersuchung über das ganze Feld der Parameter für Student und Hochschule scheinen die qualitative Gültigkeit des Modells zu bestätigen: Alle Ergebnisse waren sinnvoll und aus dem Rahmen fallende Ergebnisse wurden nicht beobachtet. Es ist deshalb anzunehmen, daß das Modell zumindest qualitativ die Wirklichkeit richtig simuliert, und daß es zur Entscheidungshilfe benutzt werden kann. Das Modell ist nur als erster, möglicherweise naiver, Schritt in weitgehend unerforschtem Neuland zu werten; Verbesserungen werden folgen müssen. Immerhin ist aus den Ergebnissen wohl die Folgerung zu ziehen, daß rechnergestützte Simulation auf dem Gebiet menschlichen Verhaltens wertvolle und ernstzunehmende Ergebnisse liefern kann, obwohl sich Eingabedaten oft nur durch Schätzungen quantifizieren lassen.

Literaturverzeichnis

(1) A. W. Astin, College Dropouts: A National Profile. American Council on Education, ACE Research Reports, Vol. 7, No. 1, February 1972 (Washington, D. C.).

(2) J. M. Dutton, W. H. Starbuck, Computer Simulation of Human Behavior John Wiley, New York 1971

(3) H. Guetzkow, P. Kotler, R. L. Schultz, Simulation in Social and Administrative Science --- Overviews and Case-Examples. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1972

(4) M. J. Shapiro, The House and the Federal Role: A Computer Simulation of Roll-Call Voting. American Political Science Review, 62, June 1968, 495-517

(5) A. E. Amstutz, Development, Validation, and Implementation of Computerized Microanalytic Simulations of Market Behavior. In D. B. Hertz and J. Malese (eds,), Proceedings of the Fourth International Conference on Operational Research. Wiley-Interscience, New York, N. Y., 1966

(6) E. B. Roberts, D. I. Abrams, H. B. Weil, A Systems Study of Policy Formulation in a Vertically-Integrated Firm. Management Science, 14, August 1968, B-674 - B-694

(7) A. I. Siegel, J. J. Wolf, Man-Machine Simulation Models: Psychosocial and Performance Interaction. Wiley-Interscience, New York, N. Y. 1969

(8) W. R. Fey, J. E. Knight, The Dynamics of Educational Institutions. In Proceedings of the 1971 Summer Computer Simulation Conference. Board of Simulation Conferences (BSC), 5975 Broadway, Denver, Colorado, 80216.

(9) W. T. Newell, Simulation of Natural Resource Management and Sociological Systems: An Application of Industrial Dynamics. In Proceedings of the 1971 Summer Computer Simulation Conference. Board of Simulation Conferences (BSC), 5975 Broadway, Denver, Colorado 80216

(10) H. Bossel, Dynamic Simulation of the College Student and the Dropout Problem. University of California, Santa Barbara, UCSB-ME-72-3, 1972

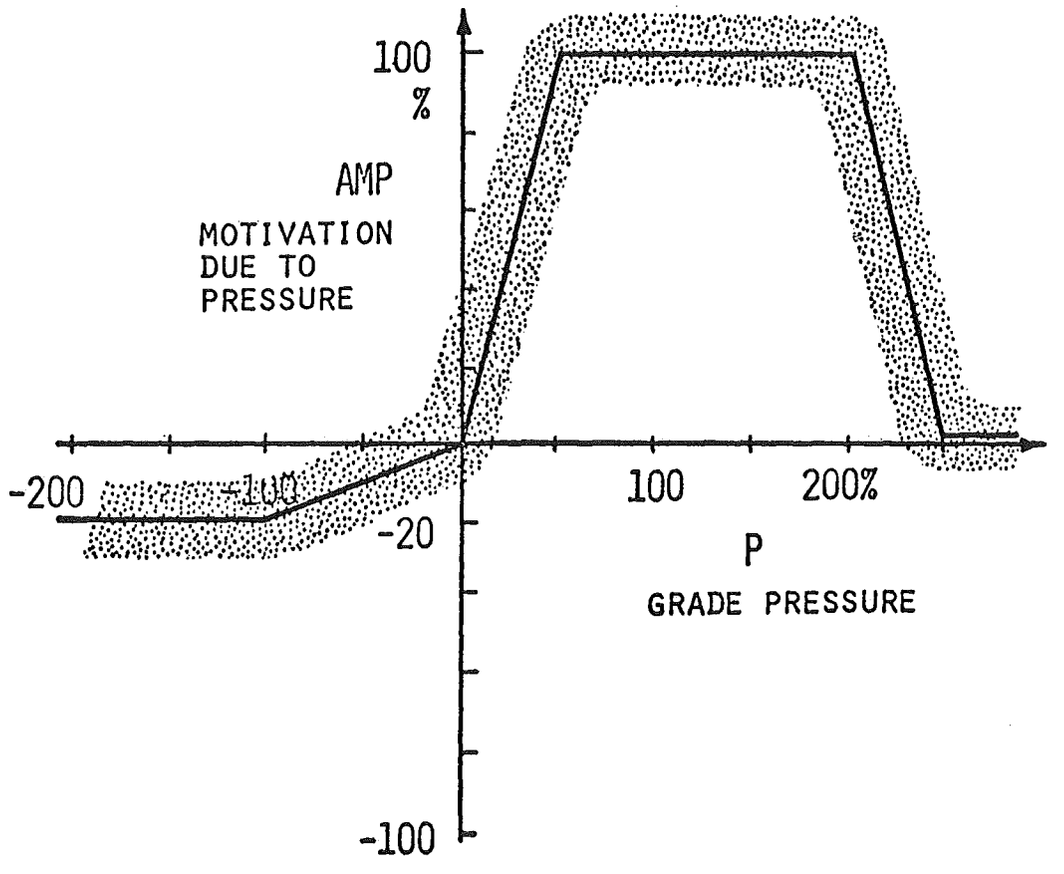
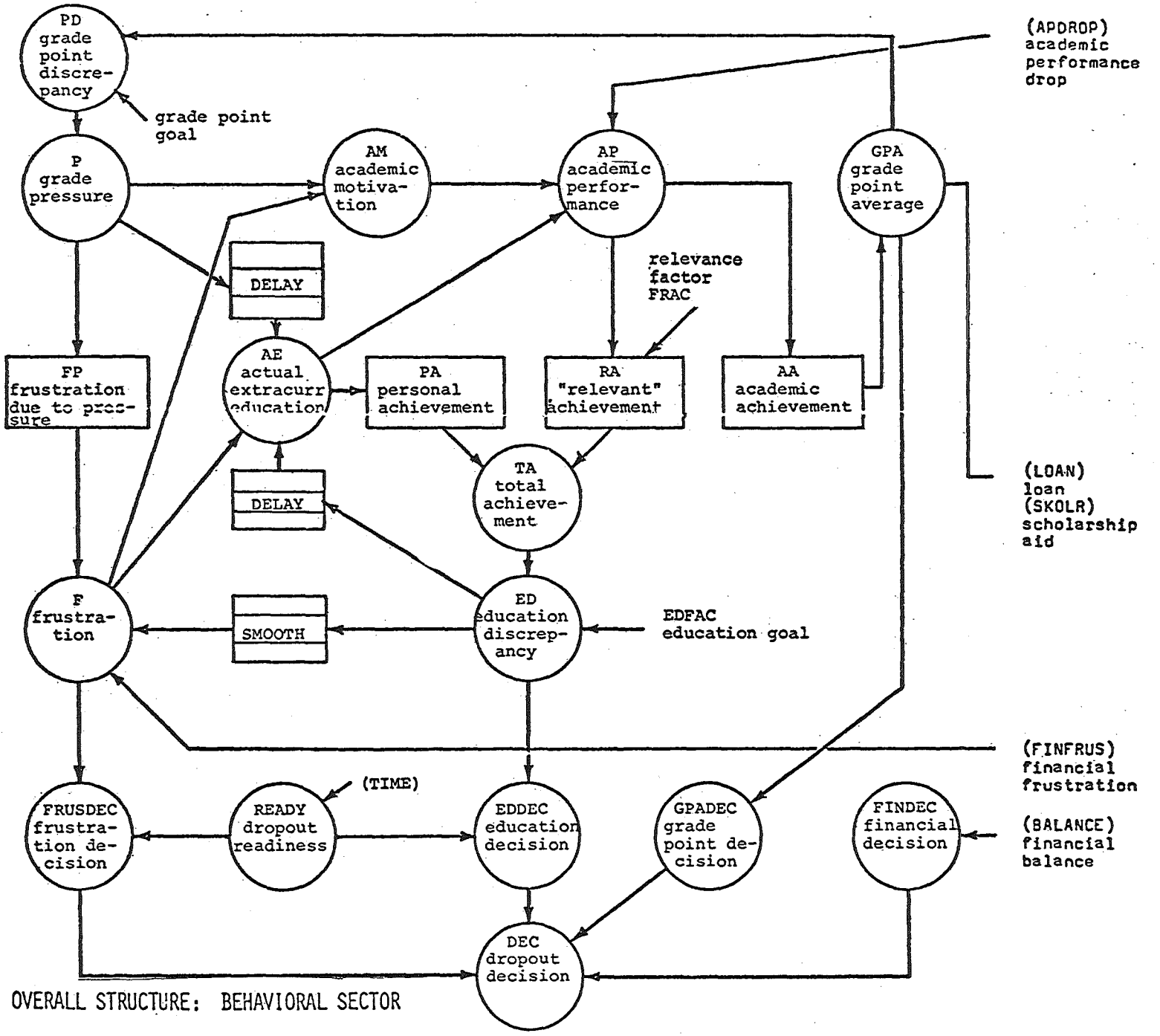

100
%
AMP
MOTIVATION
DUE TO
PRESSURE
-200
-100
-20
100
200%
P
GRADE PRESSURE
-100

BILD 1



OVERALL STRUCTURE: BEHAVIORAL SECTOR

BILD 2

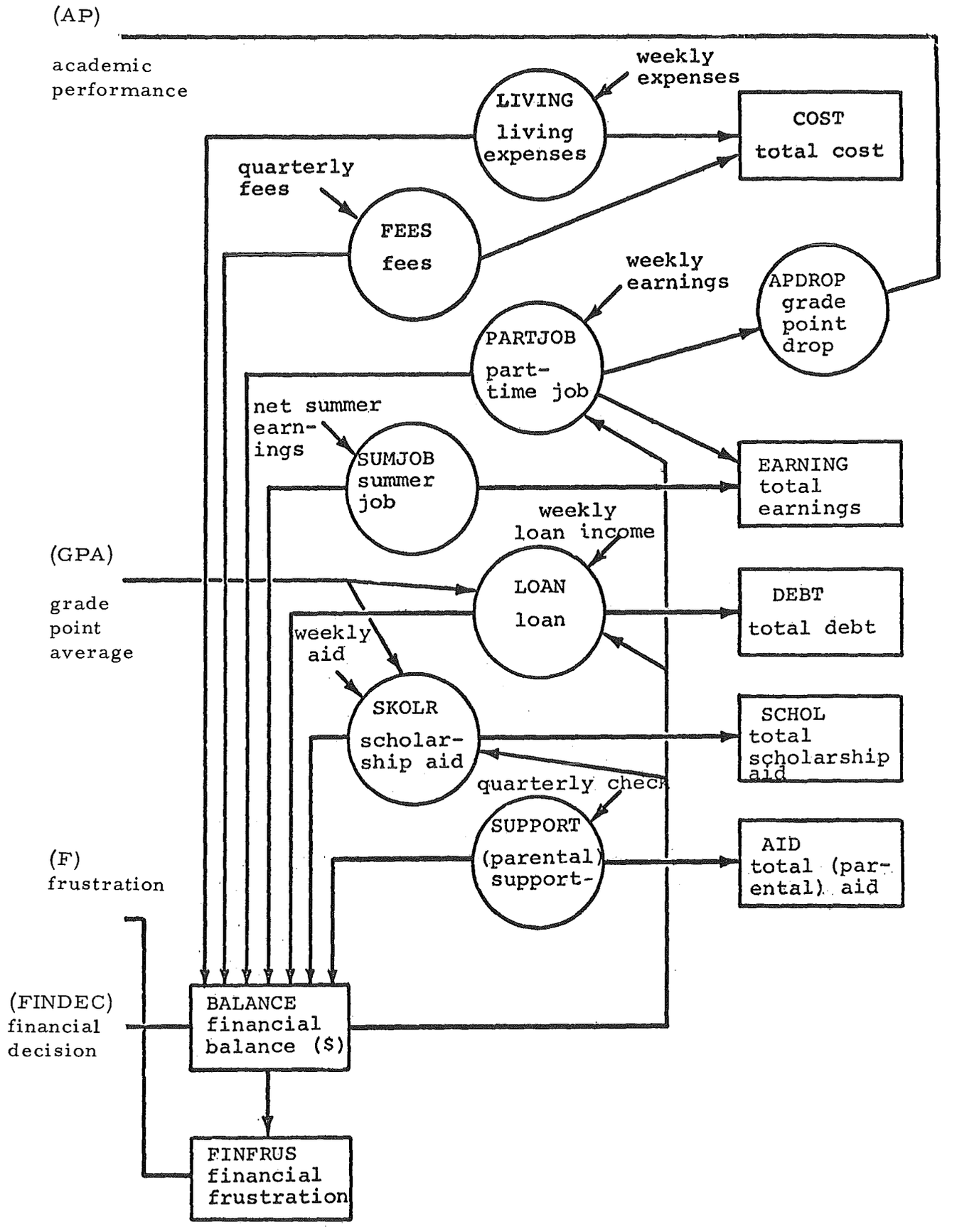
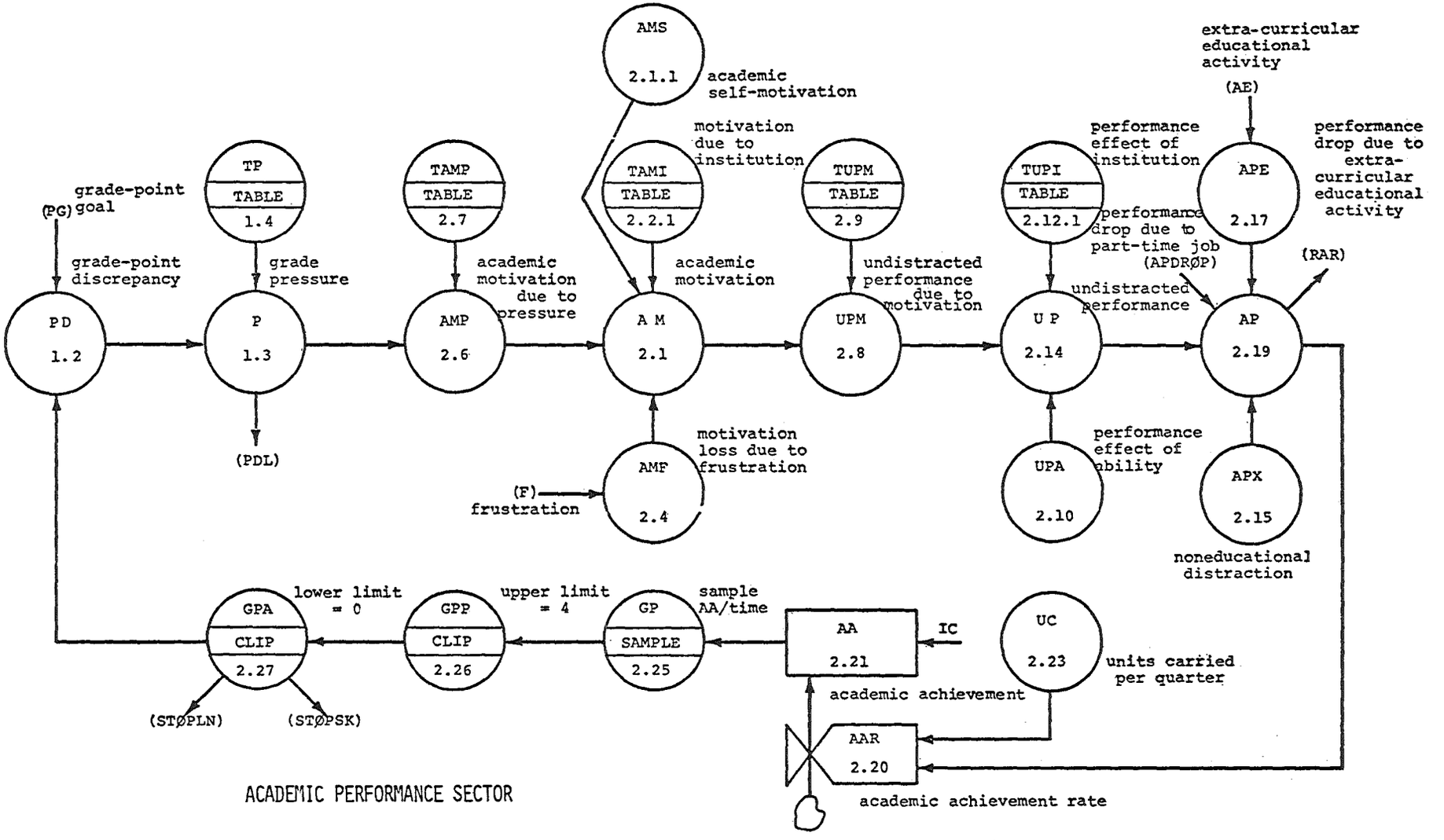


BILD 3 OVERALL STRUCTURE: FINANCIAL SECTOR



ACADEMIC PERFORMANCE SECTOR

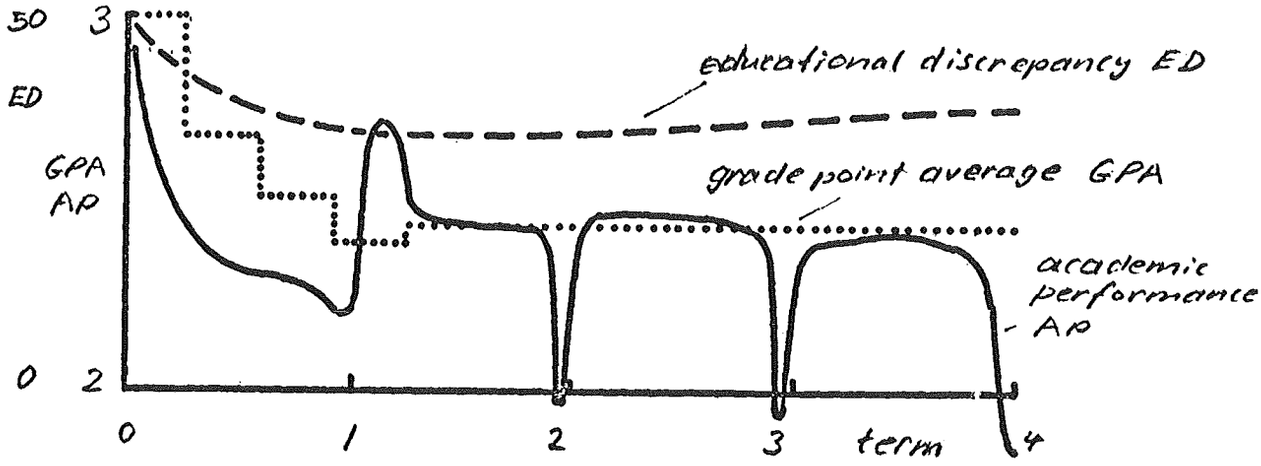
BILD 4

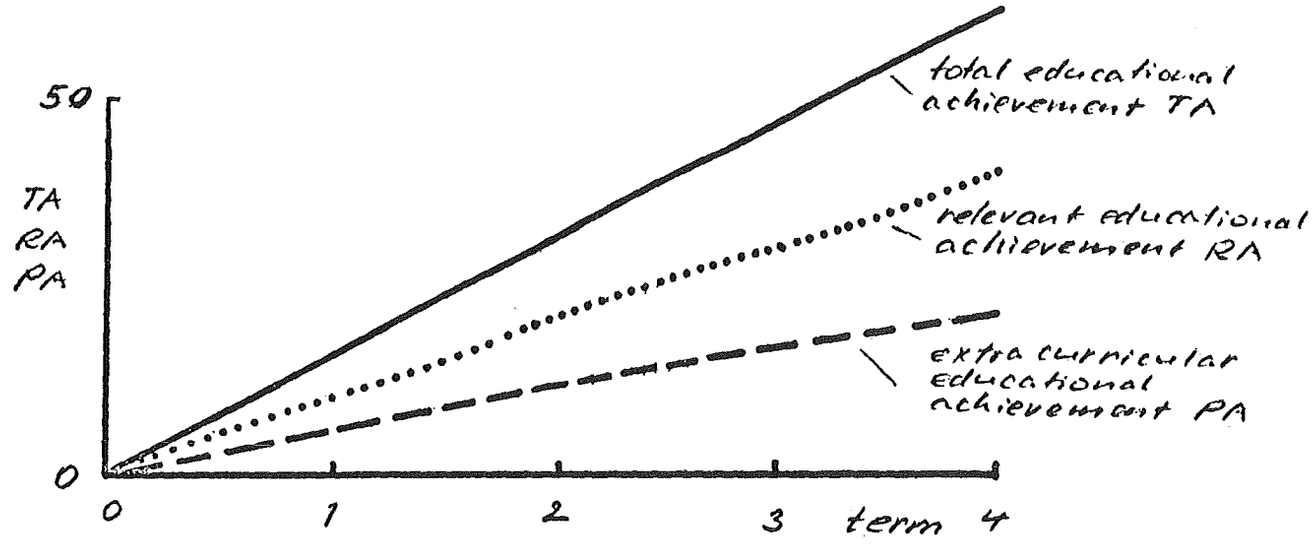
STUDENT TYPES

Type Description	A "privileged degree student"	B "privileged education student"	C "average degree student"	D "average education student"	modified D similar to "D", no financial problems	E "under-privileged (degree) student"
grade-point goal PG	3.75	2.5	2.5	2.5	variable	2.5 gp
education goal EDFAC	100	100	62.5	100	variable	62.5 %
self-motivation CAMS	20	20	0	20	0	20 %
achievement test SATVM	1350	1350	1150	1150	variable	950 score
extracurr. distraction CAPX	0	0	.25	0	0	0 gp
initial grade-point avg. IGPA	3.5	3	3	3	variable	2 gp
units/quarter CUC	15	15	15	15	15	15
initial savings SAVINGS	500	500	500	500	500	250 $
support/quarter CHECK	850	850	600	600	800	300 $/12 wk
net summer income SUMMER	0	0	500	500	500	500 $
part-time job income/ week EARN	-	-	50	50	50	50 $/wk
loan income/week LOANC	-	-	50	50	50	50 $/wk
scholarship income/ week SKOLSHP	-	-	50	50	50	50 $/wk
living expenses/week LIVING	50	50	50	50	50	50 $/wk
fees/quarter FEES	225	225	225	225	225	225 $/wk
part-time job? JOB	0	0	1	1	1	1
loans? LOANS	0	0	0	0	0	0
scholarship aid? SKOLAR	0	0	1	1	1	1
frustration threshold FRUSTHR	100	200	150	200	200	300 %
educational discrepancy threshold EDTHR	67	33	100	50	50	100 %

INSTITUTION TYPES

Description	"excellent institution"	"average institution"	"poor institution"
motivation due to institution TAMI	20	0	-20 %
performance effect of institution TUPI	.5	0	-.25 gp
relevance fraction of curriculum TFRAC	1	.67	.33
extracurr. educational environment ENVIR	1	.5	.25


50
3
ED
GPA
AP
0
2
educational discrepancy ED
grade point average GPA
academic performance AP
0
1
2
3
term
4


50
TA
RA
PA
0
total educational achievement TA
relevant educational achievement RA
extra curricular educational achievement PA
0
1
2
3
term
4

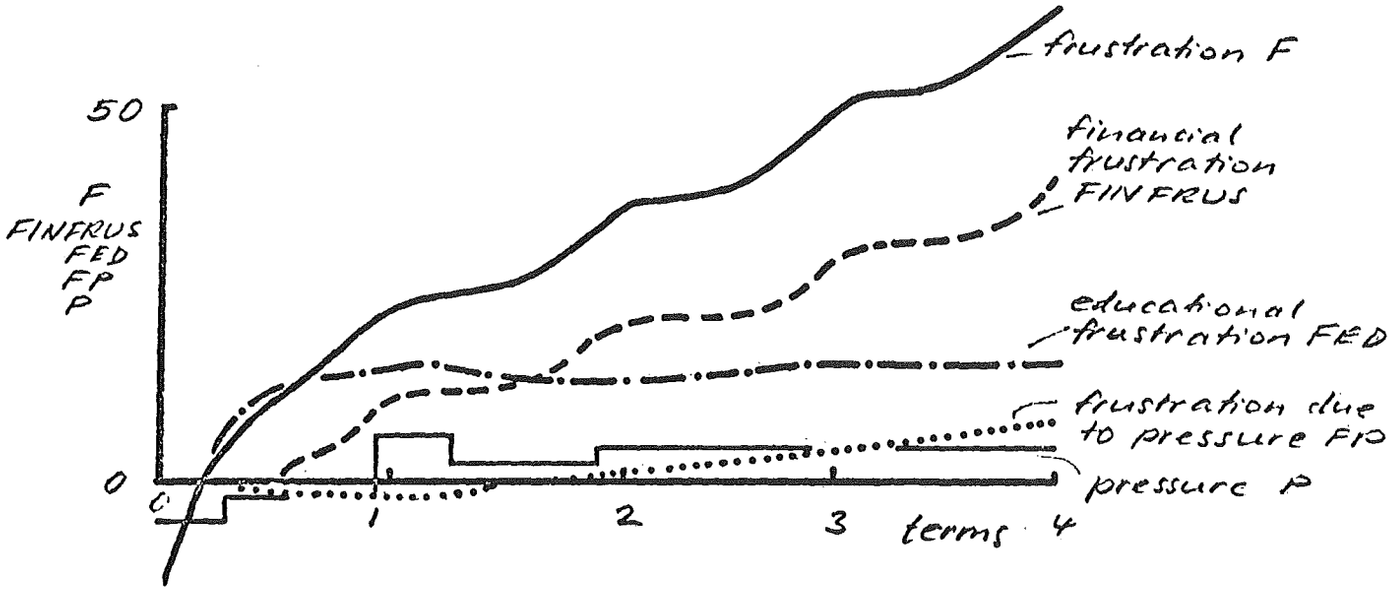

50
F
FINFRUS
FED
FP
P
0
frustration F
financial frustration FINFRUS
educational frustration FED
frustration due to pressure FP
pressure P
0
1
2
3
terms
4

BILD 6

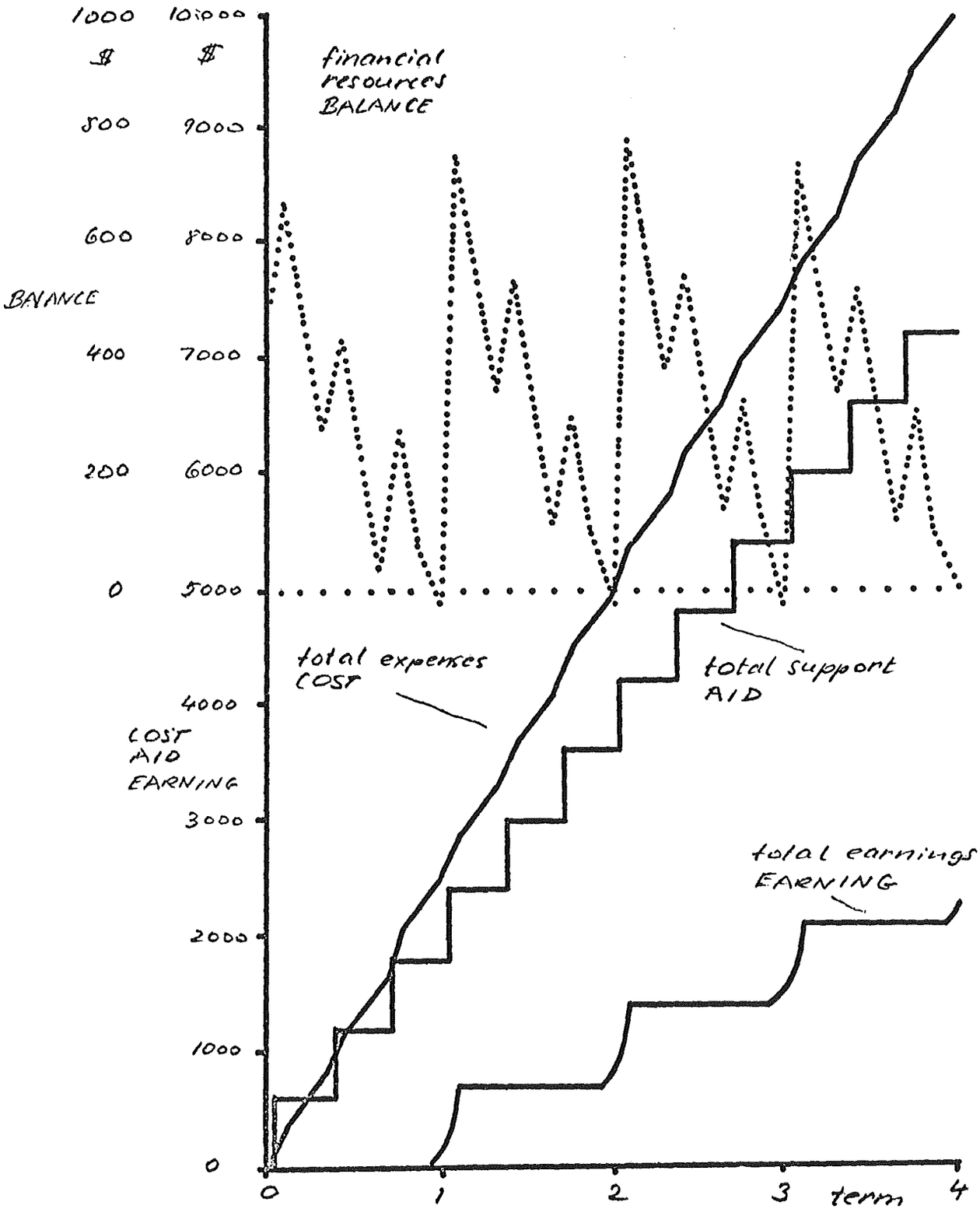

1000 $
10000 $
800
9000
600
8000
BALANCE
400
7000
200
6000
0
5000
financial resources BALANCE
total expenses COST
total support AID
4000
COST AID EARNING
3000
total earnings EARNING
2000
1000
0
0
1
2
3
term
4

BILD 7

SUMMARY OF FINAL RESULTS FOR FIVE DIFFERENT STUDENTS IN THREE DIFFERENT INSTITUTIONS

Student	Institution	Final grade-point average GPA (gp)	Total educational achievement TA (%)	Total frustration F (%)
"A" "privileged degree student"	excellent average poor	4.0 3.72 3.48	132 80 40	-93 8 156
"B" "privileged education student"	excellent average poor	4.0 3.73 2.69	133 82 43	-180 -98 37
"C" "average degree student"	excellent average poor	4.0 2.45 2.19	104 53 28	-158 76 227
"D" "average education student"	excellent average poor	3.95 2.41 2.18	101 62 31	-89 129 252
"E" "under-privileged (degree) student"	excellent average poor (36wks)	2.61 2.06 (1.68)	65 43 (5)	34 226 (190)

BILD 8

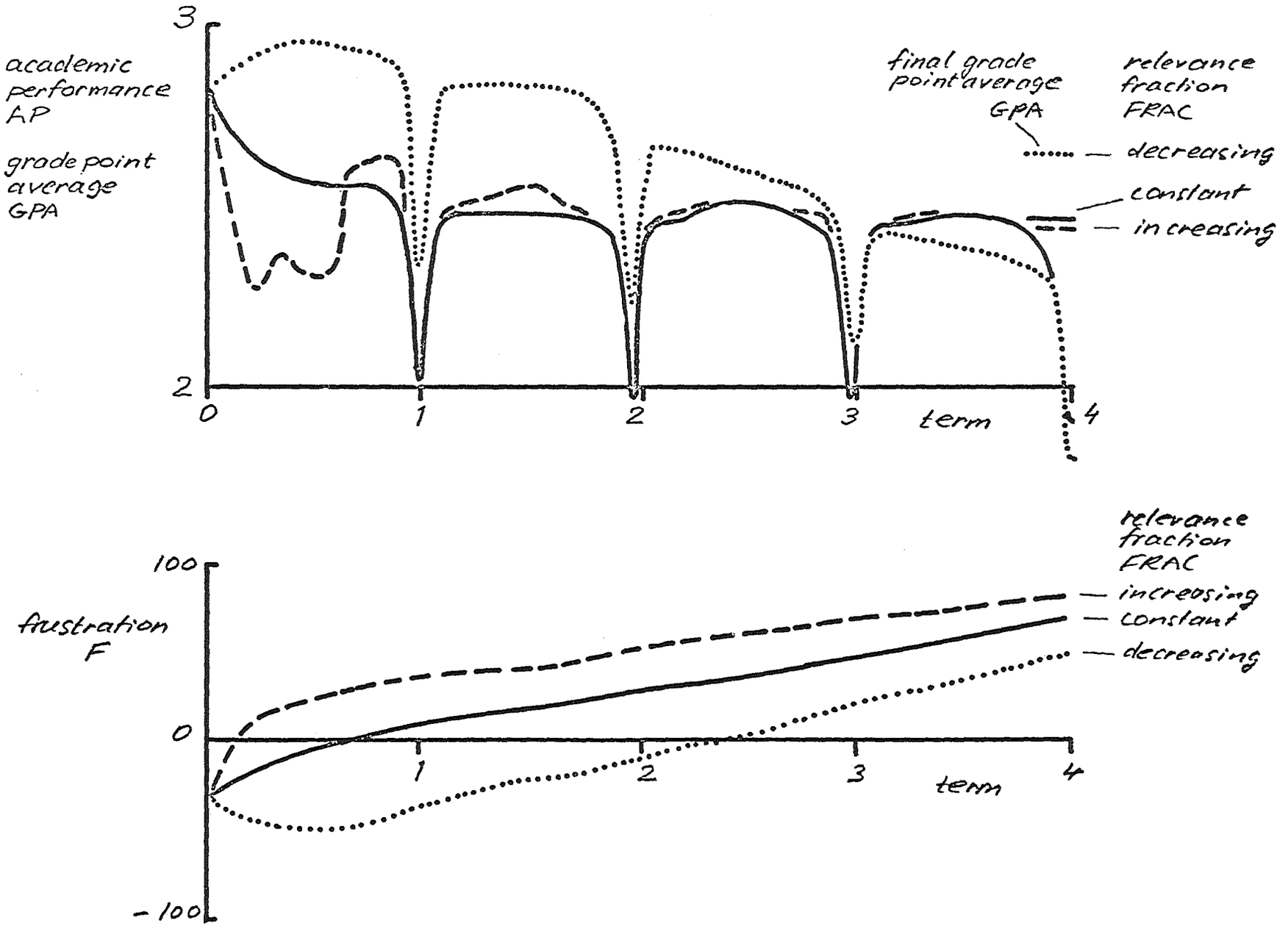
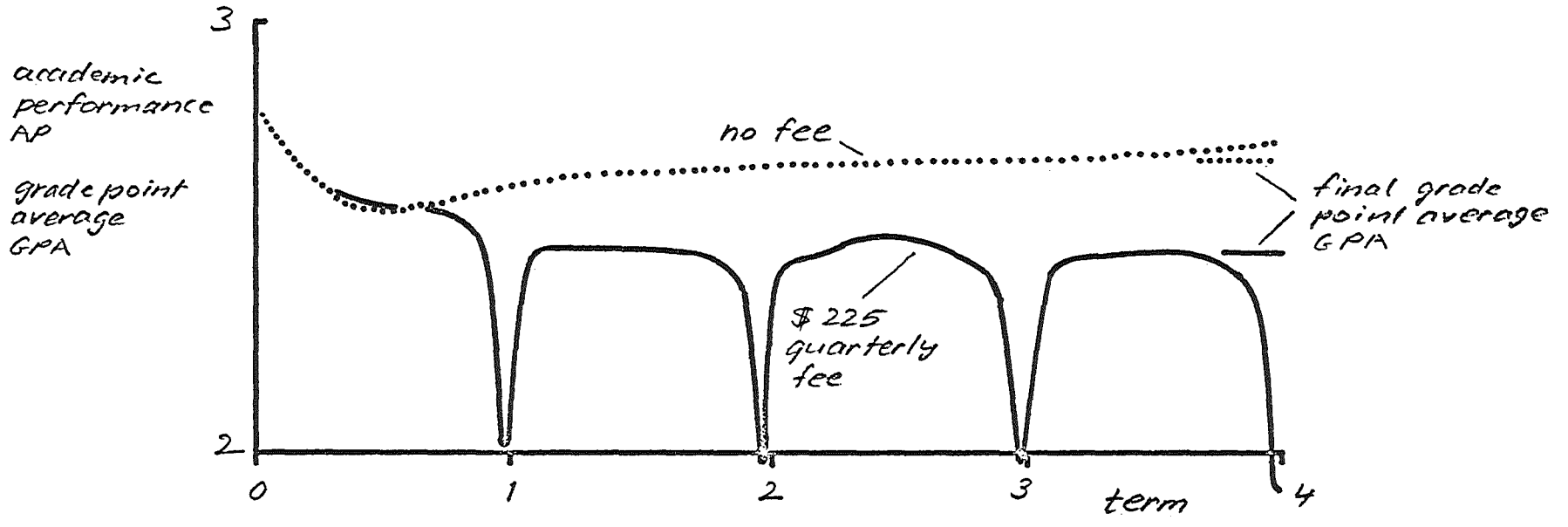

3
2
academic performance AP
grade point average GPA
final grade point average GPA
relevance fraction FRAC
decreasing
constant
increasing
0
1
2
3
term
4
100
-100
frustration F
relevance fraction FRAC
increasing
constant
decreasing
0
1
2
3
term
4

BILD 9


3
academic performance AP
grade point average GPA
no fee
final grade point average GPA
$225 quarterly fee
2
0
1
2
3
term
4

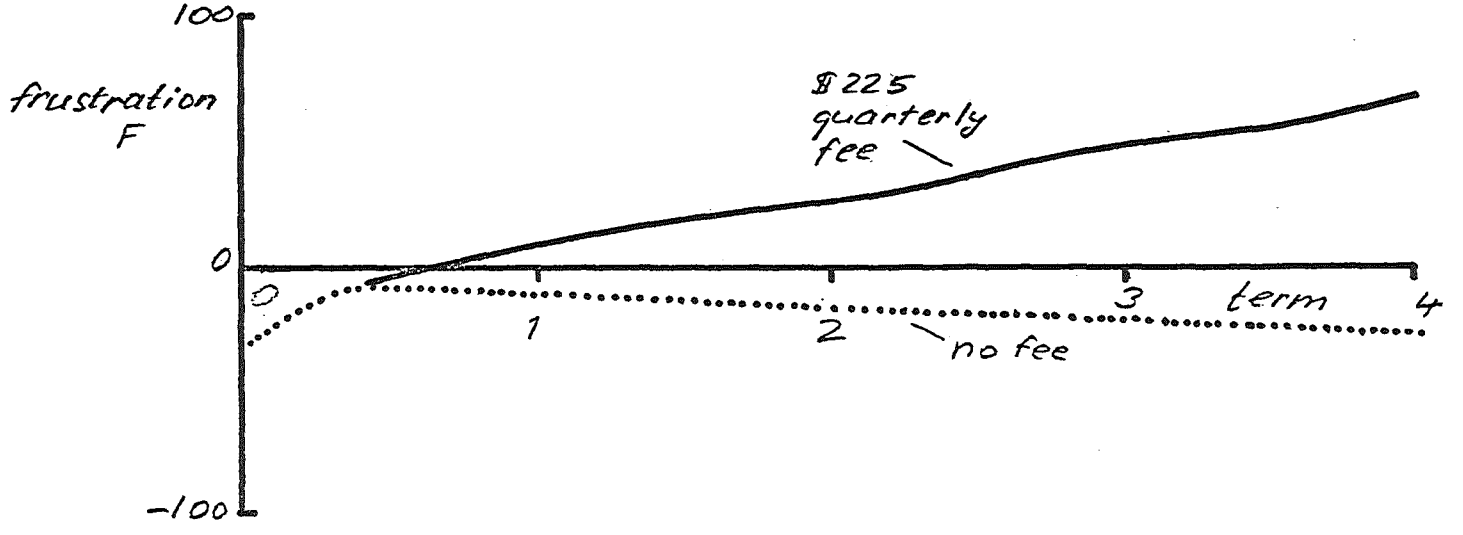

100
frustration F
$225 quarterly fee
0
0
1
2
3
term
4
no fee
-100

BILD 10

MAJOR CONCLUSIONS

Factor	Major effects on: grade-point average GPA	total educational achievement TA	total frustration F
excellence of institution	+	+	+
high student ability SATVM	+	+	+
high grade-point goal PG	+	0	-
high educational goal EDFAC	0	+	-
good extra-curricular educational possibilities ENVIR	0	+	0
high relevance fraction of curriculum FRAC	0	+	+
decreasing (vs. increasing) relevance fraction FRAC	+	0	+
better financial support CHECK	+	0	+
no fee ($225/quarter) FEEC	+	0	+

BILD 11

```
FULL DRCPCUT SIMULATICN 12 MAY 72    5/24/72

*  FULL DRCPCUT SIMULATICN 12 MAY 72
NCTE  UPDATE 14 MAY 72
NCTE
NCTE  *************************************************************************
NCTE  FOLLOWING EQUATICNS ARE FCR FRUSTRATICN SECTOR (F):
A  PC.K=PG-GPA.K                    PERFCRMANCE DISCREPANCY (GP)          1.2
A  P.K=TABHL(TP,PC.K,-4,4,1)        PRESSURE (PCT)                        1.3
T  TP=-100,-75,-5C,-25,0,1CC,2CC,300,400                                  1.4
R  FPR.KL=P.K                       FRUSTRATICN RATE (PCT)                1.5
A  FCLIP1.K=CLIP(100,FP.K,FP.K,100) UPPER CLIP OF FP                      1.6
A  FCLIP2.K=CLIP(FCLIP1.K,-10C,FCLIP1.K,-100)  LOWER CLIP OF FP           1.7
A  FUN.K=CLIP(FCLIP1.K,FCLIP2.K,FP.K,0) CLIPPED FP                        1.8
L  FP.K=FUN.J+CT*FPR.JK*CFP         FP LEVEL                              1.9
N  FUN=FP                           INITIAL FRUSTR. D/T PRESSURE          1.10
C  CFP=.0278 =1/36                  NORMALIZING CCNST (1/WEEKS)           1.12
A  F.K=FP.K+FEC.K+FINFRUS.K         FRUSTRATICN (PCT)                     1.13
NCTE
NCTE  *************************************************************************
NCTE  FOLLOWING EQUATICNS ARE FOR ACADEMIC ACHIEVEMENT SECTOR (AA):
A  AM.K=AMI.K+AMP.K+AMF.K+AMS.K     ACADEMIC MOTIVATION (PCT)             2.1
A  AMS.K=CAMS                       ACADEMIC SELF-MOTIVATICN (PCT)        2.1.1
A  AMI.K=TABHL(TAMI,TIME.K,0,144,36) MOTIVATION D/T INSTITUTICN (PC)       2.2
A  AMF.K=CAMF*F.K                   AM D/T FRUSTRATICN/ENTHUS. (PCT)      2.4
C  CAMF=-0.25                      SCALING FACTCR FCR FRUSTRATION        2.5
A  AMP.K=TABHL(TAMP,P.K,-200,300,25) AM D/T PRESSURE (PCT)                 2.6
T  TAMP=-20,-2C,-20,-20,-20,-15,-10,-5,0,50,100,100,100,100,100,100,100,  2.7
X  50,0,0,0
A  UPM.K=TABHL(TUPM,AM.K,-140,180,40)  UNDISTR. PERFOR. D/T MOTIV. (GP)   2.8
T  TUPM=-2.8,-2.0,-1.2,-0.4,0.4,1.2,2.0,2.0,2.0   TABLE FOR UPM (GP)      2.9
A  UPA.K=(SATVM-1150)/200           200 SATVM = 1 GP, MEAN 1150           2.10
A  UPI.K=TABHL(TUPI,TIME.K,0,144,36) PERFORMANCE EFF. OF INST. (GP)       2.12
A  UP.K=UPM.K+UPA.K+UPI.K+2.5       UNDISTRACTED PERFORMANCE              2.14
A  APX.K=CAPX                       EXTERNAL DISTRACTION (GP)             2.15
A  APE.K=CAPE*AE.K                  EFFECT OF EXTRACURR. ED.ACT (GP)      2.17
C  CAPE=-0.03                      DROP OF 3 GP FOR 100 PCT AE           2.18
A  AP.K=LP.K+APX.K+APE.K+APC3CP.K   ACTUAL ACADEMIC PERFORMANCE (GP)      2.19
R  AAR.KL=AP.K                      ACADEMIC ACHIEVEMENT RATE (GP)        2.20
L  AA.K=AA.J+(((DT*UC.J*AAR.JK)/7.2)/12.)   ACADEMIC ACHIEVEMENT (PCT)    2.21
A  UC.K=CLC                         UNITS CARRIED PER QUARTER             2.23
A  GP.K=SAMPLE((((AA.K*7.2*12)/15)/TIME.K),12,IGPA) GP SAMPLE, 12 WKS     2.25
A  GPP.K=CLIP(4,GP.K,GP.K,4)        CLIP GPA TO 4 IF GT. 4                2.26
A  GPA.K=CLIP(GPP.K,C,GPP.K,0)      CLIP GPA TO 0 IF LT. 0                2.27
NCTE
NCTE  *************************************************************************
NCTE  FOLLOWING EQUATICNS ARE FOR PERS'L ACAC. ACHIEVEMENT SECTOR (PA):
A  EDDL.K=DELAY1(EC.K,12)           DELAYED EDUC. DISCREPANCY (PCT)       3.1
A  DE.K=TABHL(TDE,EDDL.K,-1C0,1CC,20)  DESIRED EXTRACURR. EDUC.ACTIVITY   3.2
T  TDE=0,0,0,0,0,0,20,40,6C,8C,100  TABLE FOR DE (PCT)                    3.3
A  PDL.K=DELAY1(P.K,1)              DELAYED PRESSURE (PCT)                3.4
A  PRSFAC.K=TABLE(TPRES,PDL.K,-2CC,300,50)  PRESSURE FACTCR (0...1)       3.5
T  TPRES=.5,.5,.5,.5,.5,.25,C,0,C,0,0  TABLE FOR PRSFAC                   3.6
A  FRFAC.K=TABHL(TFRUS,F.K,-2CC,300,50) FRUSTRATICN FACTCR (0...1)        3.7
T  TFRUS=.5,.5,.5,.5,.5,.25,C,0,C,C,0  TABLE FRFAC                        3.8
A  AE.K=DE.K*ENVIR*(PRSFAC.K+FRFAC.K)  ACTUAL EXTRACURR. ED. ACT. (PCT)   3.9
R  PAR.KL=AE.K                      PERS'L ED. ACHIEVEMENT RATE           3.11
L  PA.K=PA.J+CT*PAR.JK*PAFAC        PERS'L ED. ACHIEVEMENT (PCT)          3.12
C  PAFAC=0.0139 =0.5/36             NORMALIZATICN TO 100 PCT (1/WKS)      3.14
NCTE
```

```
FULL CROPCUT SIMULATION 12 MAY 72    5/24/72

NOTE  ********************************************************************
NOTE  FOLLOWING EQUATIONS ARE FOR EDUCATIONAL DISCREPANCY SECTOR (ED):
A  FRAC.K=TABHL(TFRAC,TIME.K,0,144,36)   PERCEIVED RELEVANCE OF CURR.      4.1
R  RAR.KL=AP.K*FRAC.K                    RELEV. ACAD. ACH. RATE            4.3.1
L  RA.K=RA.J+((((CT*UC.J*RAR.JK)/7.2)/12.)  RELEV. ACAD. ACHIEVEMT. (PC)   4.3.2
A  TA.K=PA.K+RA.K                        TOTAL PERS'L ED. ACHIEVMT. (PCT)  4.4
A  EG.K=EDFAC*TIME.K/144                 PERS'L EDUC. GOAL FUNCTION (PCT)  4.5
A  EC.K=SWITCH(0,(EG.K-TA.K)/EG.K,TIME.K)*100  EDUC. DISCREPANCY (PCT)     4.7
A  EDSMTH.K=SMOOTH(EC.K,12)              ED DELAY, 12 WKS                  4.8
A  FED.K=TABHL(TFED,EDSMTH.K,-100,100,25)    FRUSTRATION D/T ED (PCT)     4.9
T  TFED=-100,-100,-100,-70,-30,15,50,100,100 TABLE FOR FED (PCT)          4.10
NOTE
NOTE  ********************************************************************
NOTE  FOLLOWING EQUATIONS ARE FOR FINANCIAL SECTOR:
R  NETFLO.KL=SUPPORT.K+LOAN.K+SKOLR.K+PARTJOB.K+SUMJOB.K-LIVING.K-FEES.K  5.1
NOTE                                     NET INCOME RATE ($/WK)
L  BALANCE.K=BALANCE.J+DT*NETFLO.JK      BALANCE ($)                       5.2
N  BALANCE=SAVINGS+EARNING+AID+SCHOL-COST  INITIAL BALANCE ($)             5.3
A  SUPPORT.K=PULSE(CHECK,0,12)           PARENTAL SUPPORT ($/QUARTER)      5.5
A  SUMJOB.K=PULSE(SUMMER,36,36)          NET SUMMER INCOME ($)             5.7
A  PARTJOB.K=EARN*STEP(1,STARTJB.K)*(1-STEP(1,STOPJB.K))*JOB               5.9
NOTE  PARTTIME JOB IF REQIRED BY BALANCE AND PERMITTED BY GPA
A  STARTJB.K=CLIP(200,TIME.K,BALDLY.K,100)  PARTJOB STARTS IF BAL.LT.100  5.11
A  STOPJB.K=CLIP(TIME.K,200,BALDLY.K,500)  PARTJOB STOPS IF BAL.GT.500    5.12
A  APDROP.K=STEP(1,STARTJB.K)*(1-STEP(1,STOPJB.K))*(-0.5)*JOB             5.13
NOTE  ACAD. PERFORMANCE DROP OF 0.5 GP IF PARTTIME JOB
A  BALDLY.K=DELAY1(BALANCE.K,2)          2 WK DELAY BEFORE JOB FOUND       5.14
A  LOAN.K=LOANC*STEP(1,STARTLN.K)*(1-STEP(1,STOPLN.K))*LOANS              5.15
NOTE  LOAN IF REQUIRED BY BALANCE AND PERMITTED BY GPA
A  STARTLN.K=CLIP(200,TIME.K,BALDLY.K,0)  LOAN STARTS IF BAL. LT. 0       5.17
A  STOPLN.K=CLIP(200,TIME.K,GPA.K,1.5)   NO LOAN IF GPA LT. 1.5           5.18
A  SKOLR.K=SKOLSHP*STEP(1,STARTSK.K)*(1-STEP(1,STOPSK.K))*SKOLAR          5.19
NOTE  SCHOLARSHIP IF REQUIRED BY BALANCE AND PERMITTED BY GPA
A  STARTSK.K=CLIP(200,TIME.K,BALDLY.K,200)  SKOLSHP IF BAL.LT.$200        5.21
A  STOPSK.K=CLIP(200,TIME.K,GPA.K,3)     NO SKOLSHIP IF GPA LT. 3         5.22
A  FEES.K=PULSE(FEEC,0,12)               QUARTERLY FEES                   5.24
A  FINPRS.K=TABHL(TFINPRS,BALANCE.K,-1000,1000,100)  FINANCIAL PRESSURE   5.26
T  TFINPRS=100,100,100,100,100,100,100,100,100,80,60,40,20,0,0,
X  0,0,0,0,0
NOTE  TABLE FOR FIN.PRESS. AS FN. OF BALANCE
R  FINFR.KL=FINPRS.K                     FRUSTRATION RATE D/T FINPRS       5.28
A  FFCLIPU.K=CLIP(100,FINFRUS.K,FINFRUS.K,100)  UPPER CLIP OF FINFRUS      5.29
A  FFCLIPL.K=CLIP(FFCLIPU.K,-100,FFCLIPU.K,-100)  LOWER CLIP OF FINFRUS    5.30
A  FFRUS.K=CLIP(FFCLIPU.K,FFCLIPL.K,FINFRUS.K,0)  CLIPPED FINFRUS          5.31
L  FINFRUS.K=FFRUS.J+DT*FINFR.JK*CFF     FINANCIAL FRUSTRATION (PCT)      5.32
N  FFRUS=FINFRUS                         INITIAL FINANCIAL FRUSTRATION    5.34
C  CFF=0.0278 =1/36                      NORMALIZING CONST                5.35
R  LOANR.KL=LOAN.K                       LOAN RATE ($/WK)                 5.36
L  DEBT.K=DEBT.J+DT*LOANR.JK             DEBT ($)                         5.37
R  EARNR.KL=SUMJOB.K+PARTJOB.K           EARNING RATE D/T JOBS ($/WK)     5.39
L  EARNING.K=EARNING.J+DT*EARNR.JK       ACCUMULATED EARNINGS ($)         5.40
R  EXPR.KL=LIVING.K+FEES.K               EXPENSE RATE ($/WK)              5.42
L  COST.K=COST.J+DT*EXPR.JK              ACCUMULATED COST ($)             5.43
R  AIDR.KL=SUPPORT.K                     SUPPORT RATE ($/WK)              5.46
L  AID.K=AID.J+DT*AIDR.JK                ACCUMULATED SUPPORT ($)          5.47
R  SCHOLR.KL=SKOLR.K                     SCHOLARSHIP RATE ($/WK)          5.49
L  SCHOL.K=SCHOL.J+DT*SCHOLR.JK          ACCUMULATED SCHOLARSHIPS ($)     5.50
NOTE
```

```
FULL DROPOUT SIMULATION 12 MAY 72    5/24/72

NOTE  *****************************************************************
NOTE  FOLLOWING EQUATIONS ARE FOR DECISION SECTOR:
A  DEC.K=GPADEC.K*FINDEC.K*FRUSDEC.K*EDUCDEC.K COMBINED DECISION FUNCT.   6.1
A  LENGTH.K=SWITCH(TIME.K,144,DEC.K)                                      6.2
A  GPS.K=SAMPLE(GPA.K,36,2)              PROVIDES NEW SAMPLE              6.3.1
R  GPASMPL.KL=SAMPLE(GPA.K,36,2)         SAMPLE GPA EVERY 36 WKS          6.3
A  LASTSPL.K=GPASMPL.JK                  SAVE OLD GPA SAMPLE              6.4
A  GPADEC.K=CLIP(1,0,((GPS.K+LASTSPL.K)/2),1.8) DROPOUT IF AVGGPA LT.1.8  6.5
A  FINDEC.K=CLIP(1,0,BALANCE.K,-200)     DROPOUT IF $ 200 IN THE RED      6.6
A  READY.K=TABHL(TREADY,TIME.K,0,144,36) DROPOUT READINESS                6.7
T  TREADY=0,1,.75,.5,0                                                    6.8
A  FRUSMOD.K=F.K*READY.K                 MODIFIED FRUSTRATION             6.9
A  FRUSSPL.K=SAMPLE(FRUSMOD.K,36,0)      SAMPLE FRUSMOD EVERY 36 WKS      6.10
A  FRUSDEC.K=CLIP(0,1,FRUSSPL.K,FRUSTHR) DROPOUT DEC. D/T FRUSTRATION     6.11
A  EDMOD.K=ED.K*READY.K                  MODIFIED EDUC. DISCREPANCY       6.13
A  EDSPL.K=SAMPLE(EDMOD.K,36,0)          SAMPLE EDMOD EVERY 36 WKS        6.14
A  EDUCDEC.K=CLIP(0,1,EDSPL.K,EDTHR)     DROPOUT DEC. D/T EDUC. DISCREP.  6.15
NOTE
NOTE  *****************************************************************
NOTE  PERSONAL PARAMETERS
C  PG=3.0                                PERFORMANCE GOAL (GPA)           1.1
N  FP=0                                  INITIAL FRUSTR. D/T PRESSURE     1.11
C  CAMS=20                               ACADEMIC SELF-MOTIVATION (PCT)   2.1.2
C  SATVM=1200                            SAT V+M SCORE OF STUDENT         2.11
C  CAPX=0  (GP)                          EXTERNAL DISTRACTION             2.16
N  AA=0    (PCT)                         INITIAL ACADEMIC ACHIEVEMENT     2.22
C  CUC=15                                CONST UNITS CARRIED PER QUARTER  2.24
C  IGPA=2.5                              INITIAL GPA                      2.28
N  PA=0     (PCT)                        INITIAL PERSL EDUCL ACHIEVEMENT  3.13
N  RA=0     (PCT)                        INITIAL REL. ACAD. ACHVMT (PC)   4.3.3
C  ECFAC=100     .GE. 25*PG              PERS'L GOAL FACTOR (PCT)         4.6
NOTE  *****************************************************************
C  SAVINGS=500  $                        INITIAL SAVINGS                  5.4
C  CHECK=750  $                          QUARTERLY CHECK                  5.6
C  SUMMER=500  $                         NET SUMMER INCOME                5.8
C  EARN=50  $/WK                         PARTTIME JOB INCOME              5.10
C  LOANC=50 $/WK                         LOAN INCOME                      5.16
C  SKOLSHP=50  $/WK                      SCHOLARSHIP INCOME               5.20
A  LIVING.K=50 $/WK                      LIVING EXPENSES                  5.23
C  FEEC=225 $                            FEES PER QUARTER                 5.25
N  FINFRUS=0    (PCT)                    INITIAL FINANCIAL FRUSTRATION    5.33
N  DEBT=0                                INITIAL DEBT ($)                 5.38
N  EARNING=0     $                       INITIAL EARNINGS                 5.41
N  COST=0     $                          INITIAL COST ($)                 5.45
N  AID=0     $                           INITIAL AID FUNDS ($)            5.48
N  SCHOL=0     $                         INITIAL SCHOLARSHIP ($)          5.51
C  JOB=1,LOANS=0,SKOLAR=1                FINANCIAL ACTIONS                5.52
C  FRUSTHR=150  (PCT)                    FRUSTRATION THRESHOLD            6.12
C  EDTHR=50  (PCT)                       EDUC. DISCREPANCY THRESHOLD      6.16
NOTE
NOTE  *****************************************************************
NOTE  INSTITUTIONAL PARAMETERS
T  TAMI=0,0,0,0,0                        MOTIVATION D/T INSTITUTION (PC)  2.2.1
T  TUPI=0,0,0,0,0                        PERFORMANCE EFFECT OF INST (GP)  2.12.1
C  ENVIR=0.50                            ENVIRONMENT FOR X.ED,ACT,(0...1  3.10
T  TFRAC=1,1,1,1,1                       PERCEIVED RELEVANCE OF CURR      4.1.1
NOTE
NOTE  *****************************************************************
```

```
FULL DROPOUT SIMULATION 12 MAY 72    5/24/72

    PRINT GPA,AP,RA,TA,DE,P,FEC,F,DEBT,AID,EARNING,SUPPORT,LIVING,FEES,
    X     GP,AA,PA,ED,AE,FP,FINFRLS,BALANCE,LOAN,SCHCL,COST
    PLOT  GPA=G(1,5)/AP=P(1,5)/F=F(-200,600)
    PLOT  GPA=G(0,4)/AA=A(0,200)/RA=R(0,200)/PA=X(0,200)/TA=E(0,200)
    X     /EC=C(-100,100)/AP=P(0,4)
    PLOT  P=Q(-200,200)/FE=1(-200,200)/FEC=2(-200,200)/FINFRUS=3(-200,200)
    X     /F=F(-200,200)
    PLOT  BALANCE=$(-1000,1000)/DEBT=B(0,10000)/AID=A(0,10000)/EARNING=
    X     W(0,10000)/SCHCL=S(0,10000)/COST=C(0,10000)
    RUN   FULL DROPOUT SIMULATION 14 MAY 72 - BASIC MODEL


    SPEC  DT=1/PRTPER=4/PLTPER=4
    CP PG=2.5,CAMS=20,SATVM=1150,CAPX=0,IGPA=3,EDFAC=100
    CP SAVINGS=500,CHECK=600,SUMMER=500,EARN=50,LOANC=50,SKOLSHP=50,
    X  FEEC=225,JOB=1,LOANS=0,SCLAR=1,FRLSTHR=300,EDTHR=100
    T  TAMI=20,20,20,20,20
    T  TUPI=0.5,0.5,0.5,0.5,0.5
    T  TFRAC=1,1,1,1,1
    C  ENVIR=1
    RUN   AVG EDUC. STUDENT, EXCELLENT INSTITUTION
```

Möglichkeiten der kombinatorischen Analyse von Simulationsmodellen

von Georg Petrich
Institut für Datenverarbeitung in der Technik
Kernforschungszentrum Karlsruhe

Kurzfassung

Durch kombinatorische Manipulation von Häufigkeitsverteilungen kann die Behandlung von Parallelabläufen und von Abbruchfehlern in Simulationsexperimenten erleichtert werden. Möglichkeiten und Einschränkungen werden kurz diskutiert.

Einleitung

Zu den nicht-trivialen Problemen, die der Anwender der Monte-Carlo-Methode beim Experimentieren mit logischen und mathematischen Modellen stets neu zu berücksichtigen hat, gehören insbesondere

a) Probleme, die sich bei der Abbildung von Parallelabläufen im Modell auf das sequentiell ablaufende Monte-Carlo-Verfahren ergeben

b) das Problem, wann eine Monte-Carlo-Versuchsreihe unter Einhaltung vorgegebener Fehlergrenzen abgebrochen werden kann, insbesondere wenn sehr seltene Ereignisse beobachtet werden sollen.

Das sich aus a) ergebende Fehlerrisiko kann durch trickreiche Programmiertechniken verhältnismäßig klein gehalten werden. Die Frage der Fehlerbestimmung bei Monte-Carlo-Versuchen läßt sich dagegen nicht allgemein gültig beantworten. Hilfsmittel für die Fehlerrechnung sind daher auch nur in beschränktem Maß integrierter Bestandteil einer Simulationssprache.

In vielen Fällen liegen für die Eingangsgrößen von zu simulierenden diskreten Systemen Messungen über die Häufigkeitsverteilungen der Größen vor. Die Messungen werden in der Regel zur Anpassung der Parameter analytischer Wahrscheinlichkeitsverteilungen verwendet, aus denen für die Simulation des Modells zufällige Anfangswerte gezogen werden.

Die Auswahl der meist kontinuierlichen Wahrscheinlichkeitsverteilungen und ihre Anpassung an diskrete Messungen der Häufigkeiten aus einem beschränkten Bereich bringt zusätzliche Fehlerrisiken mit sich, besonders wenn die Verteilungen asymmetrisch oder mehrgipflig sind.

Der Versuch liegt daher nahe, Teile der Ereignisfolgesimulation eines Modells durch direkte Manipulation der gemessenen Häufigkeitsverteilungen zu ersetzen. In der Literatur finden sich für dieses Verfahren nur sehr wenige Ansätze[1)]. Ein vielfaches Durchlaufen des Modells mit jeweils zufällig aus den Anfangsverteilungen gezogenen Werten geht dann über in ein einmaliges Durchschleusen von Häufigkeitsverteilungen durch das Modell. Parallelabläufe bieten dabei keine grundsätzlichen Schwierigkeiten. Insbesondere können aber sehr seltene Ereignisse bis auf Rundungsfehler der Maschine exakt erfaßt werden, da der durch eine endliche Zahl von Versuchen erzeugte Abbruchfehler wegfällt.

Methodik:

Im folgenden seien kurz die wichtigsten kombinatorischen Funktionen zur Manipulation von Häufigkeitsverteilungen beschrieben. Unter einer Häufigkeitsverteilung $q(p)$ sei eine endliche arithmetische Wertefolge $p_1, p_2, \ldots, p_k, p_{k+1}, \ldots, p_n$ ($p_{k+1} - p_k$ = const) mit den Häufigkeiten $q(p_1), q(p_2), \ldots, q(p_k), \ldots, q(p_n)$ verstanden, wobei

$$0 \leqslant q(p_k) \leqslant 1 \text{ und } \sum_{k=1}^{n} q(p_k) = 1.$$

Addition ADD

Zu einer Warteschlange x(k) kommt eine zweite Warteschlange y(m). Die resultierende Warteschlange z(n) ergibt sich analog zur Faltung zweier kontinuierlicher Wahrscheinlichkeiten zu

$$z(n) = \sum_{k=k_{min}}^{k_{max}} x(k) \cdot y(n-k)$$

$$\text{mit } n = (k_{min} + m_{min}),\ (k_{min} + m_{min}+1),\ \ldots\ ,\ (k_{max} + m_{max})$$

Subtraktion SUB

Von einer Warteschlange x(k) werden mit der Häufigkeit y(m) gerade m Elemente entfernt. Die Restwarteschlange z(n) ist

$$z(n) = \sum_{k=k_{min}}^{k_{max}} x(k) \cdot y(k-n)$$

$$\text{mit } n = (k_{min} - m_{max}),\ (k_{min} - m_{max}+1),\ \ldots\ ,\ (k_{max} - m_{min})$$

Minimum MIN

Eine Warteschlange x(k) wartet auf eine Bedienstation, die mit der Häufigkeit y(m) gerade m Elemente gleichzeitig bearbeiten kann. Die Häufigkeit z(n) gibt die Wahrscheinlichkeit, daß gerade n Elemente bedient werden

$$z(n) = \sum_{\substack{k=n \\ m \geqslant n}} x(k) \cdot y(m) + \sum_{\substack{m=n \\ k>n}} x(k) \cdot y(m)$$

$$\text{mit } n = \min(k_{min}, m_{min}),\ \ldots\ ,\ \min(k_{max}, m_{max})$$

Willkürliche Bearbeitung (random serve) RAN

Eine Bedienstation bearbeitet eine Warteschlange mit k Elementen gleicher Priorität und Häufigkeit x(k). Jedes Element erfordere die gleiche Bearbeitungszeit, und die Bedienstation fasse nur jeweils ein Element. Die Wahrscheinlichkeit, daß die akkumulierte Warte- und Bearbeitungszeit für die Abarbeitung eines bestimmten Elementes gerade n Takte in Einheiten der Bedienungszeit beträgt, ist z(n):

$$z(0) = x(0)$$

$$z(n) = \sum_{k=n}^{k_{max}} \frac{1}{k} \cdot x(k) \quad \text{mit } n = 1,2,\ldots,k_{max}$$

Ein einfaches Beispiel soll den Gebrauch der beschriebenen kombinatorischen Primitivfunktionen verdeutlichen.

Abb.1 zeigt die Histogramm-Darstellungen von 3 Warteschlangen W_1, W_2, W_3. Die "Addition" (Faltung)

$$W = W_1 \text{ ADD } W_2 \text{ ADD } W_3$$

ergibt die resultierende Gesamtwarteschlange W.

Der Warteschlange W stehen beispielsweise 10 Bedienstationen voll und 10 Bedienstationen zu je 60% zur Verfügung. Jede Bedienstation fasse 1 Element. Das Angebot für W ist dann die Häufigkeitsverteilung

$$B = B_1 \text{ ADD } B_2 \text{ ADD } B_3 \ldots\ldots \text{ ADD } B_{20}$$

an freien Bedienstationen. Die Häufigkeitsverteilung

$$M = W \text{ MIN } B$$

findet Platz in den Bedienstationen, während

R = W SUB B

die Restwarteschlange ist.

Bei den bisherigen Überlegungen ist die Frage, wie das Modell durch die Zeit bewegt wird, unberücksichtigt geblieben. Bei der Monte-Carlo-Methode ist ein bestimmtes Ereignis mit Sicherheit für einen bestimmten Zeitpunkt eingetragen. Wenn nicht eine sehr dichte Belegung der Zeitachse vorliegt, wird eine Ereignissteuerung der Zeittaktsteuerung überlegen sein.

Bei der hier beschriebenen Methode ist im allgemeinen ein Ereignis mit Wahrscheinlichkeiten kleiner als 1 auf der Zeitachse eingetragen. Die Folge ist eine dichte Belegung der Zeitachse.

Im allgemeinen Fall geht hier daher die Ereignissteuerung der Zeitverwaltung in eine Taktsteuerung über: die Modellverwaltung muß zu jedem Zeittakt aufgerufen werden. Der Mehraufwand an Verwaltungsarbeit ist dennoch zu rechtfertigen, da das Modell im Gegensatz zur Monte-Carlo-Methode nur ein einziges Mal die Zeitachse durchläuft.

Zusammenfassung

Der Einsatz der kombinatorischen Manipulation von Häufigkeitsverteilungen ist grundsätzlich zur Simulation diskreter Systeme geeignet, die diskontinuierliche Änderungen erfahren. Die durch das System laufenden Individuen sollten entweder ununterscheidbar sein, oder ihre Unterscheidbarkeit sollte keine wesentliche Aussagekraft für die Beurteilung des Modells besitzen. Ist diese Forderung nicht erfüllt, so müssen für die wesentlichen Individuen eigene Häufigkeitsverteilungen angelegt werden, was im Extremfall zu nicht realisierbaren Speicherplatzanforderungen führen kann. Die Methode hat gegenüber der Monte-Carlo-Simulation den Vorteil, eine echte Parallelbearbeitung von Modellelementen zu ermöglichen und unabhängig von der Varianz der Verteilungen wegen des nur einmal erfol-

genden Modelldurchlaufs keine Abbruchfehler zu erzeugen. Die Speicherplatzanforderung ist am kleinsten und die Manipulierbarkeit der stochastischen Vektoren am größten, wenn die im Modell auftretenden Zufallsvariablen voneinander stochastisch unabhängig sind. Bei abhängigen Größen müssen unter Umständen mehrdimensionale Häufigkeitsmatrizen mitgeführt werden.

Die Generalisierbarkeit der Methode ist begrenzt durch den tatsächlichen Speicherbedarf. Um die Methode weiter auszubauen, müßten daher Überlegungen zur Klasseneinteilung von Häufigkeiten und zum Speichern dünn belegter Häufigkeitsverteilungen angestellt werden. Das kann jedoch zu den bereits von der Monte-Carlo-Simulation her bekannten Schwierigkeiten bei der Erfassung von Extremfällen führen.

Die besonders durch den Speicher gesetzten Grenzen zum Einsatz der beschriebenen Methode lassen keine sinnvolle Verarbeitung zu einer allgemeinen Simulationssprache zu. Es könnte dagegen in vielen Fällen von Vorteil sein, einzelne Zwischenergebnisse im Laufe einer Monte-Carlo-Simulation zum Zweck der Fehlerreduktion und Zeitersparnis durch kombinatorische Manipulation von Häufigkeitsverteilungen weiter zu verarbeiten.

Literatur

1) Digitale Simulation
Lecture Notes in Operations Research and Mathematical Systems, Vol.51, K. Bauknecht, W. Nef (Ed.), Springer, Berlin-Heidelberg-New York 1971

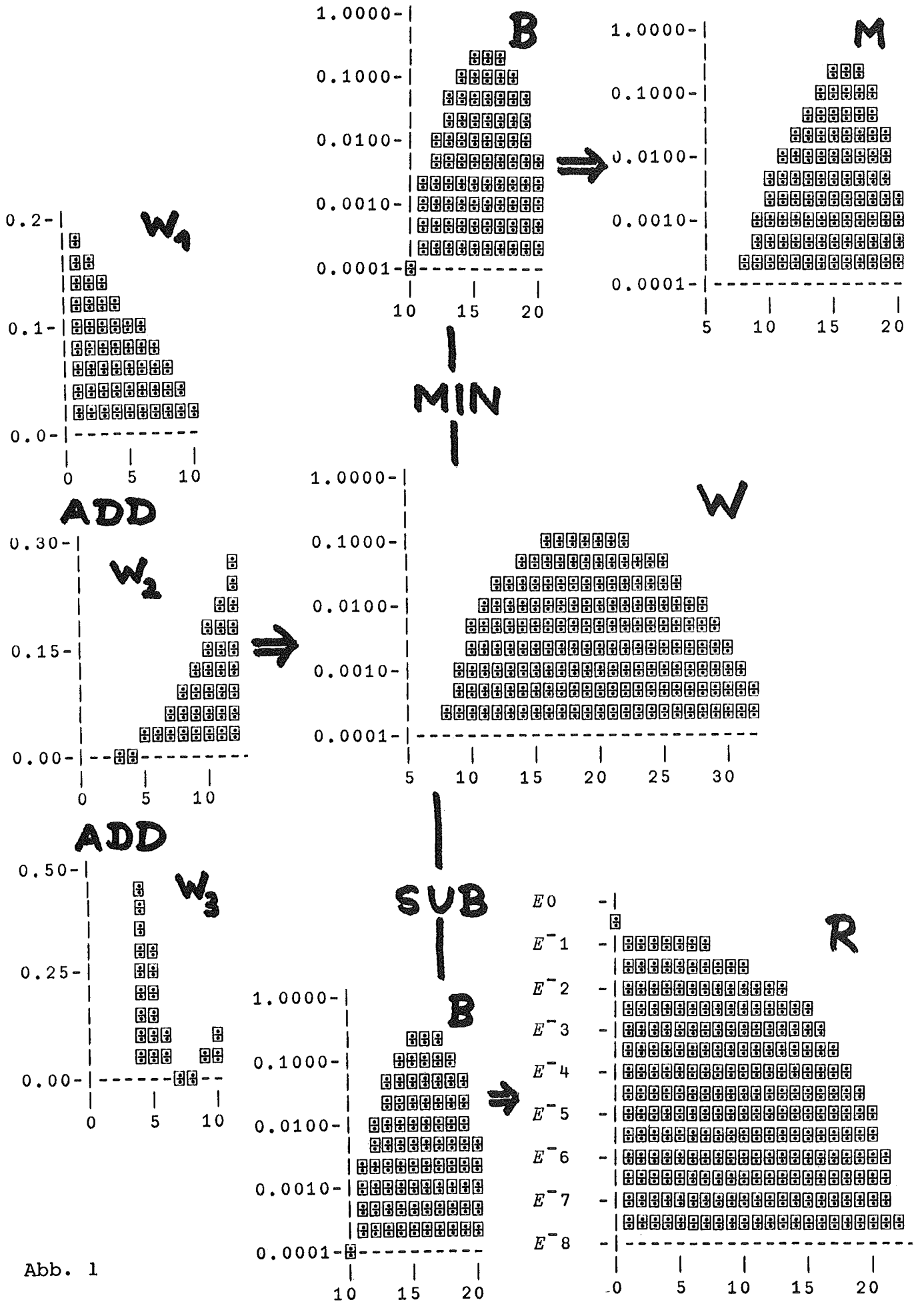


Abb. 1

Ein Programmsystem zur Simulation von Straßenverkehrssystemen

Harald Reß
Institut für Informatik der Uni Stuttgart

Kurzvortrag beim Workshop über "Methodik der rechnergestützten Simulation" am 10.5. - 11.5.1973 in Karlsruhe.

Bei der Untersuchung von komplexen Verkehrssystemen wird die Simulation häufig angewendet. Dies gilt insbesondere für die Beurteilung der Leistungsfähigkeit von Straßenverkehrssystemen. Ein Beispiel hierfür ist die Untersuchung von Nahverkehrssystemen, welche in zunehmendem Maße das tägliche Verkehrschaos in den Großstädten lösen sollen. Um die Auswirkungen ausgebauter oder neuer Nahverkehrsmittel auf den Verkehrsablauf untersuchen zu können, ist es zunächst notwendig, den Verkehrsablauf in beliebigen Straßennetzen zu erfassen. Dies soll im folgenden mit Hilfe der Simulation versucht werden.

Erste Simulationen beschäftigten sich mit dem Verkehrsablauf an einzelnen Kreuzungen mit **Lichtsignalanlagen.** Später wurden einzelne Straßenzüge und Straßennetze untersucht. Die ersten Arbeiten entstanden in den USA. Ziel der ersten Untersuchungen war es,den Verkehrsablauf darzustellen. Spätere Arbeiten beschäftigten sich zum Beispiel mit der Optimierung von Lichtsignalanlagen in bestimmten Stadtgebieten. Auch in Deutschland wenden Verkehrsingenieure die Simulation als Hilfsmittel in zunehmendem Maß an. Dabei liegen konkrete **Fragestel**lungen vor, wie z.B. die Auswirkung eines gestaffelten Arbeitsbeginns auf den Verkehrsablauf oder die Leistungssteigerung eines Autobahnnetzes durch den Einsatz von

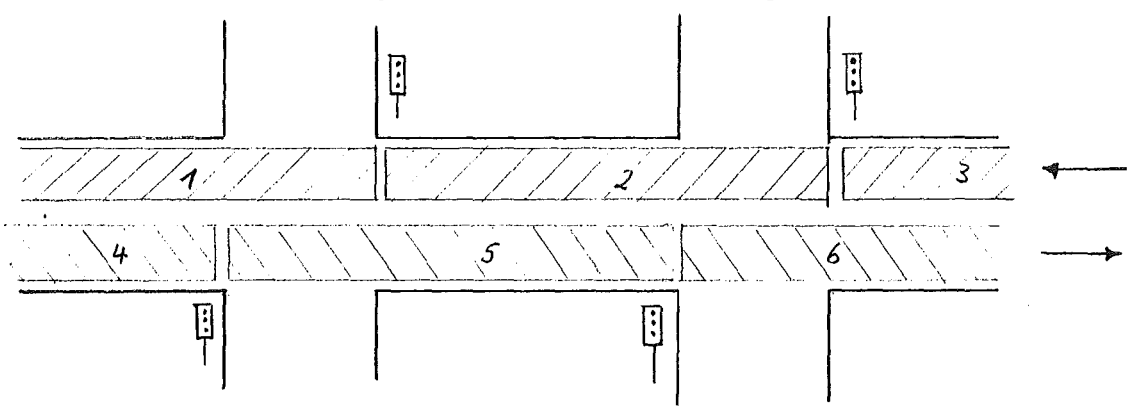
Wechselwegweisern. Die vielen Fragestellungen bringen es mit sich, daß viele Simulationsmodelle entstehen, wobei sicher oftmals ähnliche Probleme mehrfach bearbeitet werden. Die meisten dieser Untersuchungen haben eines gemeinsam: sie beobachten den Fluß von Verkehrsfahrzeugen. Wesentlich in diesen Systemen ist das Auseinander- oder Zusammenfließen von Verkehrsströmen und die Behinderung von Verkehrsströmen entweder durch eine Lichtsignalanlage, einen Polizisten oder durch einen anderen Verkehrsstrom. Aus diesen Gründen wird versucht, ein Verkehrsmodell zu erstellen, mit welchem die meisten Straßenverkehrsprobleme untersucht werden können.

Ein Straßenverkehrsnetz besteht immer aus mehreren gleichartigen Teilsystemen. Beispiele hierfür sind eine Kreuzung mit Lichtsignalanlage, eine Kreuzung mit Verkehrszeichen oder Kreuzungen verbindende Straßenabschnitte. Es ist darum möglich,ein Verkehrsnetz durch geeignete Teilsysteme zu beschreiben. Ebenso kann der Verkehrsfluß in den einzelnen Teilsystemen beschrieben werden. Diese Überlegungen führen zu einem Simulationsmodell, das sich baukastenartig aus Teilmodellen zusammensetzt.
Das Modell soll so aufgebaut sein, daß leicht einzelne Bausteine (Teilmodelle) ausgetauscht werden können. Dadurch ist es möglich, auf spezielle Wünsche eines Verkehrsingenieurs einzugehen.

Im Gegensatz zu vielen Simulationen im Verkehrswesen wird hier eine ereignisorientierte Simulation verwendet, um die Rechenzeit möglichst klein zu halten. Die Untersuchung umfangreicher Verkehrsnetze mit Hilfe einer periodenorientierten Simulation, wie sie meist verwendet wird, erscheint zeitlich recht umfangreich.

Jeder einzelne Baustein des Modells stellt einen Straßenabschnitt dar und wird durch ein Warteschlangensystem mit einem Schalter je Fahrspur realisiert. Die Länge eines Straßenabschnittes ergibt den Stauraum eines derartigen Systems; jedes Teilmodell kann also nur endlich viele Kraftfahrzeuge aufnehmen. Bei starkem Verkehr führt dies unter Umständen zu Blockeffekten im Gesamtsystem, welche leicht im Modell berücksichtigt werden können. Die Bedienzeiten an einem Schalter hängen von der Grünzeit einer Ampel, von der Lücke aufeinanderfolgender Fahrzeuge im bevorrechtigten Verkehrsstrom oder von einem Polizisten ab.

Für ein Verkehrsnetz mit Lichtsignalanlagen soll die Einteilung in Teilsysteme aufgezeigt werden. Die einzelnen Teilsysteme sind hier Straßenabschnitte in einer Fahrtrichtung zwischen zwei Kreuzungen. Am Anfang eines jeden Abschnitts befindet sich eine Ampel. In der Abbildung sind



für eine Straße die Teilsysteme eingezeichnet. Jedes Teilsystem kann mehrere Spuren umfassen, Spurwechsel ist möglich.

Die Bewegung der Fahrzeuge in den Teilmodellen wird einheitlich dargestellt. Tritt ein Fahrzeug in das System ein, so wird es von Teilsystem zu Teilsystem bewegt. Das Vorrücken der Fahrzeuge in den einzelnen Straßenabschnitten geschieht entgegen der Fahrtrichtung. Entsprechend werden die einzelnen Bausteine aktiviert. Im Beispiel ist also die Reihenfolge der Abschnitte 1, 2, 3 bzw. 6, 5, 4.

Die Aktivierung der einzelnen Teilmodelle richtet sich hier nach den jeweiligen Ampelphasen. Jedes einzelne Teilmodell wird dann aktiviert, wenn die ihm zugeordnete Ampel auf "grün" umschaltet. Die Bewegung der Fahrzeuge wird spätestens dann unterbrochen, wenn das Ende der nächsten Rotphase erreicht ist. Die nächste Aktivierung des Teilmodells findet also spätestens nach einer Umlaufphase statt. Diese "Vorrückzeit" eines Teilmodells wird jedoch noch vom vorangehenden Teilmodell mitbestimmt. Im Normalfall können die Fahrzeuge zeitlich nicht länger vorrücken als die Fahrzeuge im vorhergehenden Straßenabschnitt.

Weitere einfache Teilsysteme sind Straßenabschnitte

mit einer Verengung bzw. mit einer Erweiterung. Im ersten Fall fließen zwei Fahrzeugströme z.B. im Reißverschlußverfahren zusammen, im anderen Fall teilt sich ein Fahrzeugstrom in zwei neue Fahrzeugströme.

Auch diese Warteschlangen besitzen einen Schalter; z.B. eine Ampel.

Das Zusammenfließen zweier Verkehrsströme liegt auch bei der Zufahrt zu einer Schnellstraße oder Autobahn vor. Hier richtet sich die Bedienzeit für ein Fahrzeug am Schalter des wartepflichtigen Stromes nach den Lücken im Hauptstrom. Die Fahrzeuge des wartepflichtigen Stromes werden gemäß diesen Lücken an der Einfädelstelle in den Hauptstrom eingefügt.
Mit diesem Modell soll sowohl eine mikroskopische als auch eine makroskopische Untersuchung möglich sein. Bei der mikroskopischen Simulation werden einzelne Fahrzeuge betrachtet und von Teilmodell zu Teilmodell bewegt. Über jedes Fahrzeug kann somit genau Buch geführt und sein Weg durch das System verfolgt werden. Bei der makroskopischen Simulation werden nur noch Verkehrsströme als Ganzes betrachtet, eine Verfolgung eines einzelnen Fahrzeugs ist dann nicht mehr möglich. Hier werden nur noch Aussagen z.B. über die Geschwindigkeit des Verkehrsstromes und über die Länge dieses Verkehrsstromes gemacht. An einer Kreuzung mit Lichtanlage muß dann u.a. geprüft werden, welcher Anteil eines Stromes bei "grün" die Kreuzung passieren kann. Es ist klar, daß eine makroskopische Untersuchung wesentlich zur Verkürzung der Rechenzeit beiträgt.

W. Jentsch

Einfluß der Modellstruktur auf den Lösungsaufwand bei dynamischen elektrischen Netzwerken

1. Einleitung

Ein mathematisches Modell eines zu simulierenden Objektsystems ist i.a. auf verschiedene Weisen zu bilden, auch wenn es sich nicht um verschieden genaue Beschreibungen des Objektsystems handelt. Solche gleich genauen, d.h. mathematisch gleichwertigen Beschreibungen ergeben dieselbe "wahre" Lösung.

Bei der numerischen Lösung können sich solche mathematisch gleichwertigen Modelle recht unterschiedlich verhalten hinsichtlich der zulässigen Integrationsschrittweite, des Rechenaufwandes zur Auswertung der Differentialgleichung und der Berechenbarkeit aller interessierender Systemgrößen. Diese Unterschiede führen zu einem verschiedenen Lösungsaufwand.

Dieser Einfluß auf den Lösungsaufwand soll hier am Beispiel dynamischer elektrischer Netzwerke betrachtet werden. Er ist am ausgeprägtesten beim induktiven Teilnetzwerk mit nichtlinearen Induktivitäten.

Im folgenden werden daher ohmisch-induktive Netzwerke behandelt.

2. Mathematische Modelle eines ohmisch-induktiven Netzwerkes nach dem Umlaufstrom- und dem Umlaufffluß-Verfahren

2.1 Struktur der Modelle

Wir betrachten hier mathematische Modelle eines ohmisch-induktiven Netzwerkes, die auf dem "Umlauf-Schnittmengen-Verfahren" beruhen, und benutzen die in [1] eingeführte Darstellweise.

Bild 1 zeigt die "normale Schaltung" eines ohmisch-induktiven Netzwerkes in allgemeiner, kompakter Form. Das Netzwerk soll keine Stromquellen enthalten; diese sind für die Allgemeinheit der hier interessierenden Modelleigenschaften bedeutungslos.

Wir gelangen in 3 Schritten zu den beiden zu diskutierenden Modellen eines ohmisch-induktiven Netzwerkes.

In Bild 2 ist das Strukturbild des mathematischen Modells des Netzwerkes - in 3 Formen - dargestellt, wobei die induktive Teilschaltung I kompakt beschrieben ist, vgl. Bild 7 in [1]. Das Strukturbild in Bild 2.1 beschreibt die Widerstands-Teilschaltung ausführlich. Durch kompakte Beschreibung der Rückkopplungsschleife in der Widerstands-Teilschaltung mittels der durch

$$\underline{R}^{Q} = (\underline{G}_G - \underline{H}_{GR}\underline{G}_R\underline{F}_{RG})^{-1} \qquad \underline{G}_G = (\underline{R}_G)^{-1} \tag{1.1}$$

gegebenen 'Schnittmengen-Widerstandsmatrix' erhält man das Strukturbild von Bild 2.2. Durch Einführung der 'Induktivitätsumlauf-Widerstandsmatrix'

$$\underline{R}_L^{B} = -\underline{F}_{LG}\underline{R}^{Q}\underline{H}_{GL} \tag{1.2}$$

und der 'Induktivitätsumlauf-Ersatzurspannungen'

$$\underline{v}_I^{B} = -(\underline{F}_{LV} + \underline{F}_{LG}\underline{R}^{Q}\underline{H}_{GR}\underline{G}_R\underline{F}_{RV})\cdot\underline{v} \tag{1.3}$$

ergibt sich schließlich das in Bild 2.3 dargestellte kompakte Strukturbild.

In Bild 3 sind die Strukturbilder der mathematischen Modelle der induktiven Teilschaltung nach dem Umlaufstrom- und dem Umlaufffluß-Verfahren dargestellt. Bei dem Umlaufstrom-Verfahren sind die induktiven Umlaufströme $\underline{i}_L$ Zustandsgrößen; der Ableitungsvektor $\dot{\underline{i}}_L$ ist die Lösung eines linearen Gleichungssystems, in das im allgemeinen Fall nichtlinearer Kennlinien die differentiellen Induktivitäten $\underline{l}_L = (\underline{\gamma}_L)^{-1}$ und $\underline{l}_r$ eingehen:

$$\boxed{\underline{l}^{B}\cdot\dot{\underline{i}}_L = \underline{u}_I^{B}} \quad , \quad \underline{l}^{B} = \underline{l}_L - \underline{F}_{Lr}\underline{l}_r\underline{H}_{rL} \; . \tag{2.1}$$

$\underline{l}^{B}$ ist die differentielle "Umlauf-Induktivitätsmatrix". Bei dem Umlaufffluß-Verfahren sind die Umlaufflüsse $\underline{\psi}^{B}$ Zustandsgrößen; die induktiven Umlaufströme $\underline{i}_L$ werden hieraus im allgemeinen Fall durch Lösen des nichtlinearen Gleichungssystems

$$\boxed{\underline{f}_L(\underline{\psi}^{B} - \underline{F}_{Lr}\cdot\underline{f}_r(-\underline{H}_{rL}\cdot\underline{i}_L)) - \underline{i}_L = \underline{0}} \tag{3}$$

gewonnen.

Bild 4 zeigt schließlich die Strukturbilder der beiden zu diskutierenden mathematischen Modelle. Bei dem Umlaufstrom-Verfahren - s. Bild 4.1 - ist die Lösung des linearen Gleichungssystems in der Form

$$\dot{\underline{i}}_L = \underline{\gamma}^B \cdot \underline{u}_I^B \quad , \quad \underline{\gamma}^B = (\underline{l}^B)^{-1} \tag{2.2}$$

dargestellt. Die Abhängigkeit der Matrix $\underline{\gamma}^B$ von den differentiellen Induktivitäten $\underline{l}$, die ihrerseits letztlich von $\underline{i}_L$ abhängen, wird durch einen zweiten Rückkopplungsweg beschrieben. Bei dem Umlauffluß-Verfahren - s. Bild 4.2 - liegt nur der eine - bei dem Umlaufstrom-Verfahren in derselben Form vorhandene - Rückkopplungsweg über $\underline{R}_L^B$ vor.

2.2 Eigenschaften der Modelle

Wir betrachten nun einige Eigenschaften der beiden Modelle nach Bild 4, die auf den Lösungsaufwand wesentlichen Einfluß haben.

2.21 Eigenwerte des Differentialgleichungs-Systems

Die Eigenwerte der Funktionalmatrix des Dgl.-Systems haben auf die zulässige Integrationsschrittweite einen bestimmenden Einfluß.

Bei dem Umlaufstrom-Verfahren - s. Bild 4.1 - ist die Normalform des Dgl.-Systems durch

$$\dot{\underline{i}}_L = \underline{f}(t, \underline{i}_L) = \underline{\gamma}^B(\underline{i}_L) \cdot (\underline{v}_I^B - \underline{R}_L^B \cdot \underline{i}_L) \tag{4.1}$$

gegeben. Die zugehörige Funktionalmatrix ist

$$\boxed{\underline{A} = \left[\frac{\partial f^i}{\partial i_{Lk}}\right]} \quad \boxed{= -\underline{\gamma}^B \cdot \underline{R}_L^B + \underline{\Gamma}_*^B \cdot (\underline{v}_I^B - \underline{R}_L^B \cdot \underline{i}_L)} \tag{4.2}$$

$$\text{mit} \quad \underline{\Gamma}_*^B = \left[\frac{\partial}{\partial i_{L1}} \underline{\gamma}^B \quad \frac{\partial}{\partial i_{L2}} \underline{\gamma}^B \; ..\right] .$$

Bei dem Umlaufluß-Verfahren - s. Bild 4.2 - ist die Normalform des Dgl.-Systems durch

$$\dot{\underline{\psi}}^B = \underline{f}(t, \underline{\psi}^B) = \underline{v}_I^B - \underline{R}_L^B \cdot \underline{i}_L(\underline{\psi}^B) \tag{5.1}$$

gegeben. Die zugehörige Funktionalmatrix ist

$$\boxed{\underline{A} = \left[\frac{\partial f^i}{\partial \psi_k^B}\right]} \quad \boxed{= -\underline{R}_L^B \cdot \underline{\gamma}^B} \tag{5.2}$$

Der Term $-\underline{R}_L^B \underline{\gamma}^B$ entspricht dem Term $-\underline{\gamma}^B \underline{R}_L^B$ in (4.2).

Der wesentliche Unterschied der beiden Funktionalmatrizen nach (4.2) und (5.2) ist der durch den zweiten Rückkopplungsweg - s. Bild 4.2 - bedingte zusätzliche Term in (4.2). Dieser führt zu einer wesentlichen Vergrößerung von Eigenwerten (dem Betrage nach), wenn Arbeitspunkte auf den magnetischen Kennlinien der Induktivitäten in Bereichen starker Krümmung liegen.

Der zusätzliche Term in (4.2) tritt bei linearen magnetischen Kennlinien (d.h. bei konstanten Induktivitäten) nicht auf. Man kann ihn auch bei nichtlinearen magnetischen Kennlinien dadurch vermeiden, daß ersatzweise Polygonzug-Kennlinien verwendet werden; allerdings müssen dann die Knickpunkte der Kennlinien "eingegabelt" werden, d.h. es müssen Zwischenschritte so eingeschoben werden, daß das Überschreiten der Knickpunkte praktisch in einem Schrittzeitpunkt erfolgt.

2.22 Art der zu lösenden Gleichungssysteme

Bei beiden Modellen sind Gleichungssysteme zu lösen.

Bei dem Umlaufstrom-Verfahren - s. Bild 3.1 - tritt immer nur ein lineares Gleichungssystem (2.1) auf. Bei glatten nichtlinearen Kennlinien, d.h. bei sich laufend ändernden differentiellen Induktivitäten, ist es vorteilhaft, das Gleichungssystem in der Form (2.1) zu lösen. Bei linearen Kennlinien ist es immer, bei bereichsweise linearen Kennlinien ist es oft vorteilhaft, die inverse Matrix $\underline{\gamma}^B = (\underline{l}^B)^{-1}$ zu bilden und damit $\underline{i}_L$ nach (2.2) zu bestimmen.

Bei dem Umlaufffluß-Verfahren - s. Bild 3.2 - ist bei nichtlinearen magnetischen Kennlinien das nichtlineare Gleichungssystem (3) - z.B. nach dem Newton'schen Verfahren - zu lösen. Bei bereichsweise linearen Kennlinien kann sich eine wesentliche Verringerung des Rechenaufwandes z.B. infolge geringerer Anzahl von Iterationen bei dem Newton'schen Verfahren oder durch Anwendung eines speziellen, auf diesen Fall zugeschnittenen Verfahrens ergeben. Im Falle linearer Kennlinien wird aus (3) ein lineares Gleichungssystem.

2.23 Möglichkeiten der Bestimmung der induktiven Spannungen

Wenn nur Restzweig-Induktivitäten auftreten, ist die Bestimmung der Spannungen an diesen sehr einfach; die induktiven Spannungen $\underline{u}_L$ sind gleich den induktiven Umlaufspannungen $\underline{u}_I^B$, s. Bild 3.1. Wenn auch Baumzweig-Induktivitäten vorkommen, sind die Spannungen an den einzelnen Induktivitäten besonders zu ermitteln.

Beim Umlaufstrom-Verfahren ist dies relativ einfach, s. Bild 3.1. Aus den Ableitungen der induktiven Restzweigströme $\dot{\underline{i}}_L$ folgen über $\dot{\underline{i}}_\Gamma$ die Spannungen an den Baumzweig-Induktivitäten $\underline{u}_\Gamma$. Die Spannungen an den Restzweig-Induktivitäten $\underline{u}_L$ ergeben sich schließlich durch topologiegerechte ($\hat{=}$ Baummatrix $\underline{F}_{L\Gamma}$) Subtraktion der Spannungen $\underline{u}_\Gamma$ von den induktiven Umlaufspannungen $\underline{u}_I^B$.

Beim Umlauffluß-Verfahren ist es schwieriger, s. Bild 3.2. Eine Möglichkeit besteht hier darin, die Flüsse Ψ_L und Ψ_r zu differenzieren; dies hat zwei schwerwiegende Nachteile: die numerische Differentiation liefert nur ungenaue oder grob falsche Ergebnisse (bei Sprüngen der induktiven Spannungen) und die Ergebnisse sind jeweils erst zu einem späteren Zeitpunkt erhältlich als die übrigen Größen. Diese Nachteile kann man nur durch den zusätzlichen Aufwand vermeiden, die Ableitungen $\dot{i}_L$ nach dem Umlaufstrom-Verfahren zu bestimmen und dann die induktiven Spannungen wie bei diesem zu ermitteln.

3. Beispiele

Wir betrachten als Beispiele zwei einfache ohmisch-induktive Netzwerke, deren Induktivitäten nichtlinear sind. Simulationstests wurden mit dem RK4-Verfahren mit konstanter Schrittweite - abgesehen vom "Eingabeln" - durchgeführt.

3.0 Magnetische Kennlinien

Bild 5 zeigt zwei magnetische Kennlinien, die für die zu diskutierenden Simulationen benutzt wurden. Die Kennlinie K1 ist eine glatte Kennlinie, bei der ψ, l, dl/di durch relativ einfache Ausdrücke in Abhängigkeit von i anzugeben sind. Die Kennlinie K2 ist eine einfache Polygonzug-Kennlinie, welche die glatte Kennlinie K1 recht gut annähert.

3.1 Beispiel 1: 1-phasige nichtlineare Drossel

3.11 Beschreibung des Netzwerkes

Bild 6.1 zeigt das einfachste ohmisch-induktive Netzwerk mit einer Spannungsquelle in Form der "normalen Schaltung".

In Bild 6.2 sind die zugehörigen Strukturbilder entsprechend dem Umlaufstrom-Verfahren und dem Umlaufffluß-Verfahren dargestellt. Im ersten Fall -1 lautet die Dgl.

$$i = \frac{1}{l}\,(v - R \cdot i) \quad = \quad f\,(t,\, i) \tag{6.1}$$

und die zugehörige "Funktionalmatrix"

$$\boxed{f_i = -\frac{R}{l} - \left(\frac{1}{l^2} \cdot \frac{dl}{di}\right) \cdot (v - Ri)}\ . \tag{6.2}$$

Im zweiten Fall -2 gilt die Dgl.

$$\dot{\psi} = v - R \cdot i(\psi) \quad = \quad f\,(t, \psi) \tag{7.1}$$

mit

$$\boxed{f_\psi = -\frac{R}{l}}\ . \tag{7.2}$$

3.12 Vergleich des Umlaufstrom- und Umlaufffluß-Verfahrens

3.121 Glatte magnetische Kennlinie

Bei dem Umlaufstrom-Verfahren macht sich der Einfluß des zweiten Terms in (6.2) auf den Eigenwert f_i bereichsweise stark bemerkbar; man beachte, daß in den kritischen Term nicht nur die Krümmung der magnetischen Kennlinie, sondern auch die induktive Spannung $u_L = v - Ri$ - als Faktor - eingeht. Der Verlauf von f_i ist in

dem Bild 6.3-1 eingetragen. Die hohen Spitzen von f_i führen dazu, daß die Schrittweite h = 0,5 ms noch zulässig ist und die Schrittweite h = 1 ms schon zu Instabilität führt.

Bei dem Umlaufluß-Verfahren bleibt der Betrag des Eigenwertes f_ψ unter einer wesentlich kleineren Schranke. Mit der Schrittweite h = 2 ms, die wohl wegen der Darstellgenauigkeit nicht wesentlich überschritten werden kann, bleibt man weit unter der Stabilitätsgrenze.

3.122 Polygonzug-Kennlinie

Der Ersatz der glatten Kennlinie K1 durch die Polygonzug-Kennlinie ergibt bei einem nur sehr geringen Genauigkeitsverlust bei beiden Verfahren Vorteile hinsichtlich des Rechenaufwandes.

Beim Umlaufstrom-Verfahren kann jetzt die Schrittweite h = 2 ms wie bei dem Umlaufluß-Verfahren benutzt werden, da f_i hier den "natürlichen Wert" hat. Allerdings müssen die Knickpunkte der Polygonzug-Kennlinie "eingegabelt" werden, da innerhalb eines Integrationsschrittes keine Sprünge der Ableitung $\dot{i}$ liegen dürfen. Für eine Halbperiode sind bei einer festen Grundschrittweite von h = 2 ms insgesamt 11 Schritte[1)] nötig; dies sind wesentlich weniger Schritte als bei der glatten Kennlinie, die 20 Schritte erfordert.

Beim Umlaufluß-Verfahren kann selbst bei der Schrittweite h = 2 ms auf die "Eingabelung" verzichtet werden, da durch die Knicke in der Kennlinie hier auch nur Knicke in der Ableitung $\dot{\psi}$ auftreten. Der Vorteil hinsichtlich des Rechenaufwandes liegt hier in der einfacheren linearen Interpolation zur Gewinnung von i aus ψ. Hier sind bei der Schrittweite von 2 ms nur 5 Schritte für eine Halbperiode erforderlich. Das Umlaufluß-Verfahren ist also in diesem Beispielsfall bei relativ großen Schrittweiten deutlich überlegen.

3.2 Beispiel 2: 3-phasige nichtlineare Drossel

3.21 Beschreibung des Netzwerkes

Bild 7.1 zeigt ein einfaches 3-phasiges ohmisch-induktives Netzwerk mit drei ein symmetrisches Drehspannungssystem bildenden Spannungsquellen in Form der "normalen Schaltung".

---

1) 5 Schritte für die Grundschrittzeitpunkte;
je Knickpunkt 3 zusätzliche Schritte bei "Eingabelung"
durch Interpolation (2 Schritte zur Gewinnung des Interpolationspolynoms, 1 Schritt für den interpolierten einzuschiebenden Schritt).

In Bild 7.2 sind die zugehörigen Strukturbilder entsprechend dem Umlaufstrom- und dem Umlaufffluß-Verfahren in der Art des allgemeinen Bildes 4 dargestellt. Der Eingangsvektor $\underline{v}_I^B$ ist durch

$$\underline{v}_I^B = \begin{bmatrix} v_1 - v_3 \\ v_2 - v_3 \end{bmatrix} \tag{8}$$

gegeben. Das Diagramm in Bild 7.2-2 soll das nichtlineare Gleichungssystem veranschaulichen, das beim Umlaufffluß-Verfahren im Fall der glatten Kennlinie K1 (für alle 3 Induktivitäten) zu lösen ist.

3.22 Vergleich des Umlaufstrom- und des Umlaufffluß-Verfahrens

3.221 Glatte magnetische Kennlinie

Bei dem Umlaufstrom-Verfahren macht sich auch hier der zweite Term in (4.2) in gleicher Weise wie beim Beispiel 1 bemerkbar. Die Schrittweite h = 0,5 ms ist noch zulässig; die Schrittweite h = 1 ms ergibt Instabilität. Bild 7.3 zeigt den genauen Verlauf der drei Ströme, der auch mit h = 0,5 noch sehr gut angenähert wird.

Bei dem Umlaufffluß-Verfahren bleibt die Schrittweite h = 2 ms wie beim Beispiel 1 weit unter der Stabilitätsgrenze. Die größere zulässige Schrittweite wird aber durch einen erheblichen Aufwand zur Lösung des nichtlinearen Gleichungssystems (3) erkauft.

3.222 Polygonzug-Kennlinie

Bei dem Umlaufstrom-Verfahren sind in der ersten Halbperiode 8"Eingabelungen" erforderlich; bei einer Grundschrittweite von h = 2 ms sind dann insgesamt 34 Schritte für eine Halbperiode nötig. Dafür tritt hier der Aufwand zur Lösung eines nichtlinearen Gleichungssystems nicht auf.

Bei dem Umlaufffluß-Verfahren ist wie im Beispiel 1 die Schrittweite h = 2 ms weit unter der Stabilitätsgrenze und ein "Eingabeln" der Knickpunkte nicht erforderlich. Die Verwendung der Polygonzug-Kennlinie führt zwar zu einer merklichen Verminderung des Aufwandes zur Lösung des nichtlinearen Gleichungssystems (geringere Zahl von Iterationsschritten), doch ist der verbleibende Aufwand immer noch erheblich.

Literatur

[1] W. Jentsch : Zur Aufstellung des mathematischen Modells eines dynamischen elektrischen Netzwerkes über den "normalen Baum". ETZ-A, erscheint im Herbst 1973.

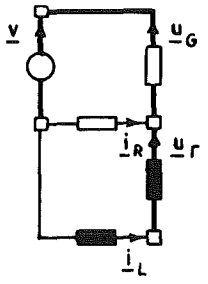
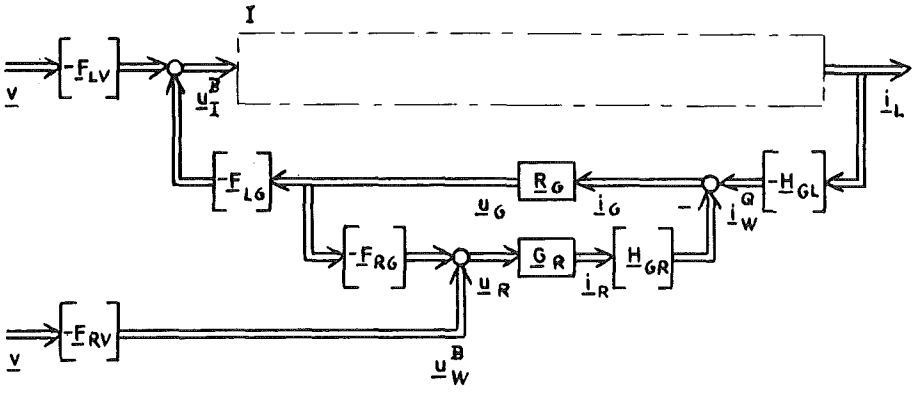
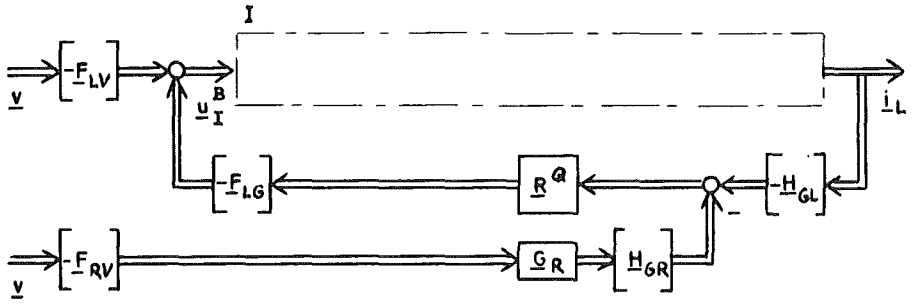


Bild 1 Normale Schaltung
eines ohmisch-induktiven Netzwerks
in allgemeiner, kompakter Form

2.1



2.2



2.3

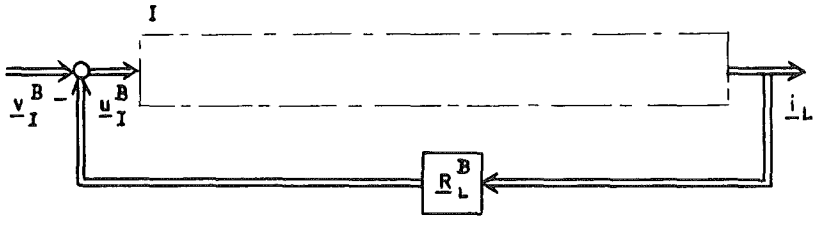
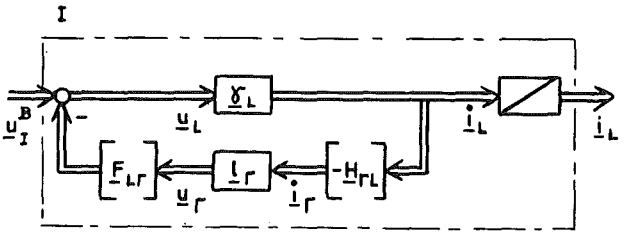


Bild 2 Strukturbild
des mathematischen Modells eines ohmisch-induktiven Netzwerks
mit kompakter Beschreibung der induktiven Teilschaltung I

3.1



3.2

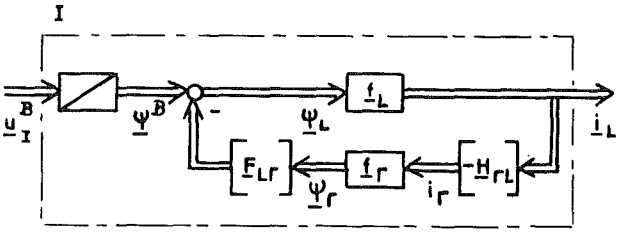
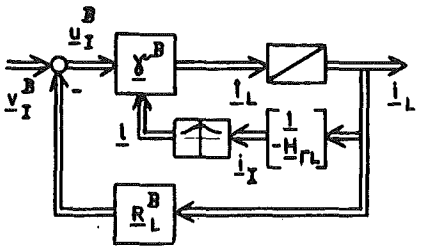


Bild 3 Strukturbilder
der mathematischen Modelle der induktiven Teilschaltung
nach dem

3.1 Umlaufstrom-Verfahren

3.2 Umlauffluß-Verfahren

4.1



4.2

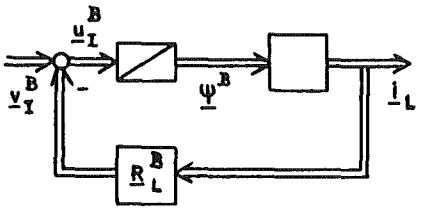


Bild 4 Strukturbilder
der mathematischen Modelle eines ohmisch-induktiven Netzwerks
in kompakter Form
nach dem

4.1 Umlaufstrom-Verfahren

4.2 Umlauffluß-Verfahren

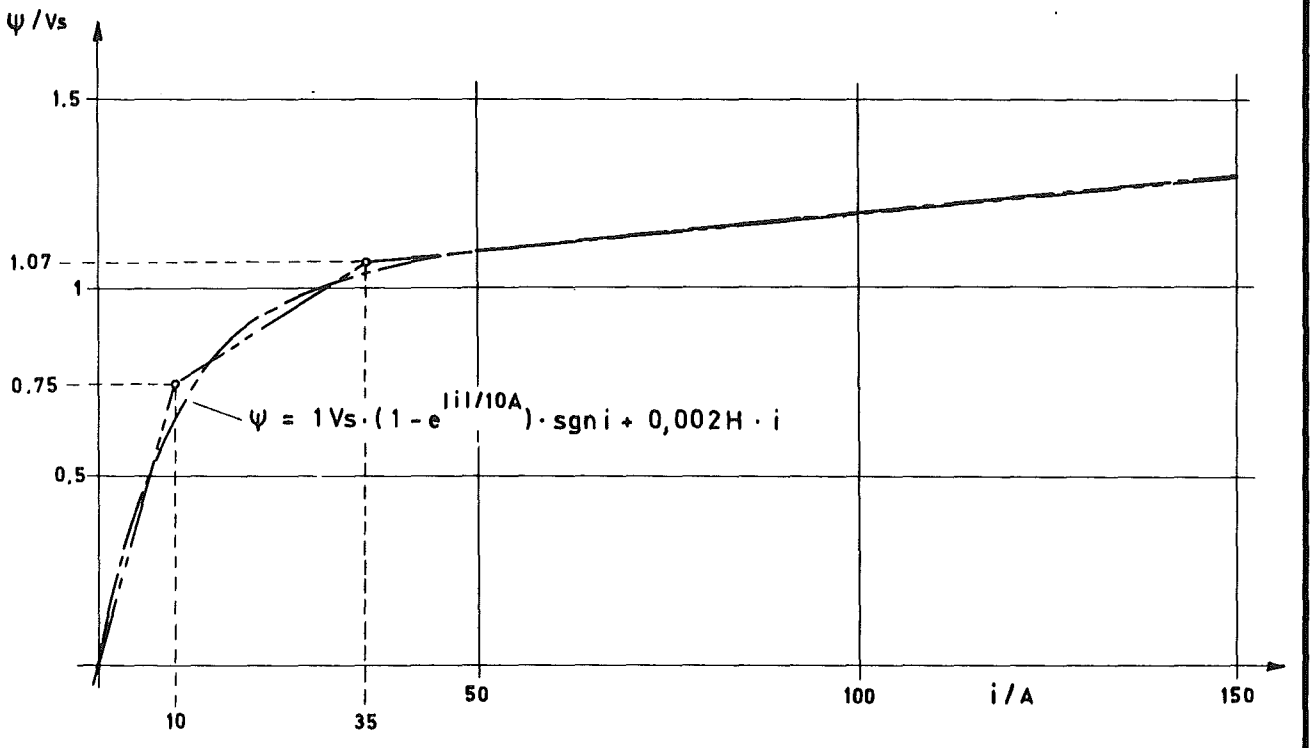
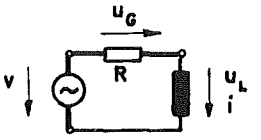


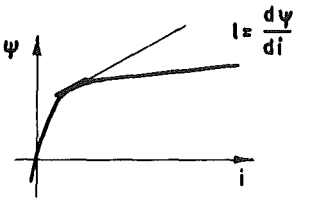
Bild 5 Magnetische Kennlinien

——·—— K1 : glatte Kennlinie

——··—— K2 : Polygonzug-Kennlinie

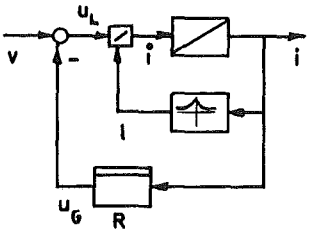
6.1

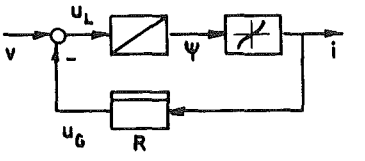




-1 -2

6.2





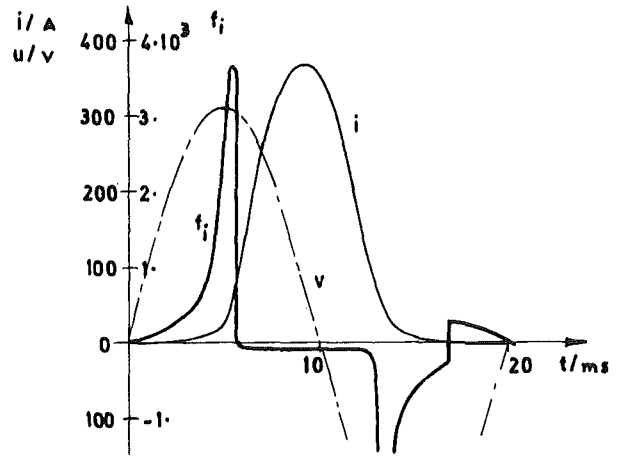
$$\dot{i} = \frac{1}{l}(v - Ri) = f(t, i)$$

$$f_i = -\frac{R}{l} - \left(\frac{1}{l^2} \cdot \frac{dl}{di}\right) \cdot (v - Ri)$$

$$\dot{\psi} = v - R \cdot i(\psi) = f(t, \psi)$$

$$f_\psi = -\frac{R}{l}$$

6.3



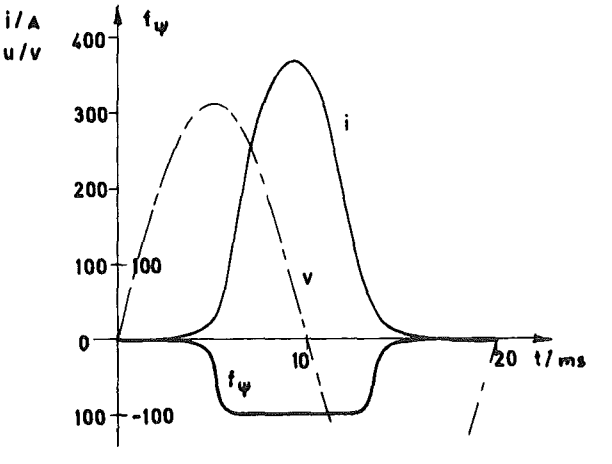


Bild 6 Beispiel : 1-phasige nichtlineare Drossel

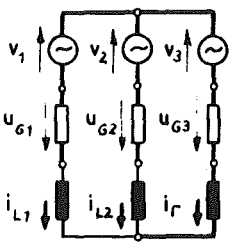
6.1 "Normale Schaltung"

6.2 Strukturbild

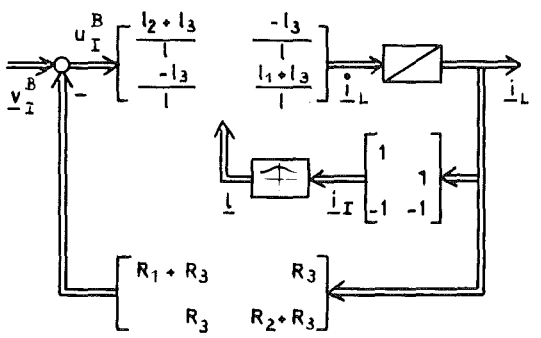
6.3 Zeitdiagramme
für: $\hat{v} = 311V$ $R = 0{,}2\Omega$
$f = 50Hz$ magnetische Kennlinie K1

-1 Umlaufstrom-Verfahren
-2 Umlaufffluß-Verfahren

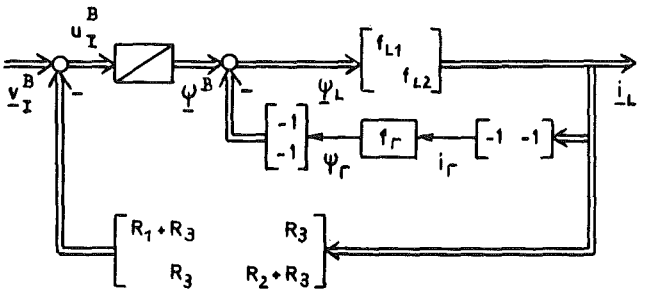
7.1



7.2 -1



-2



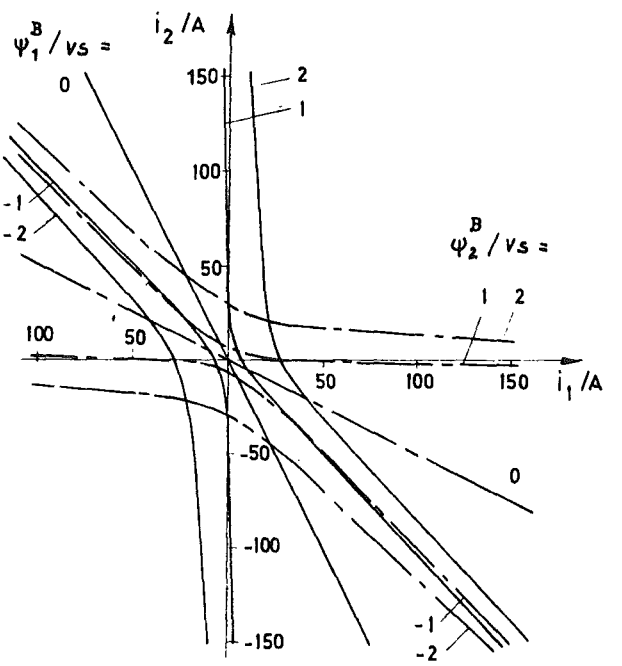


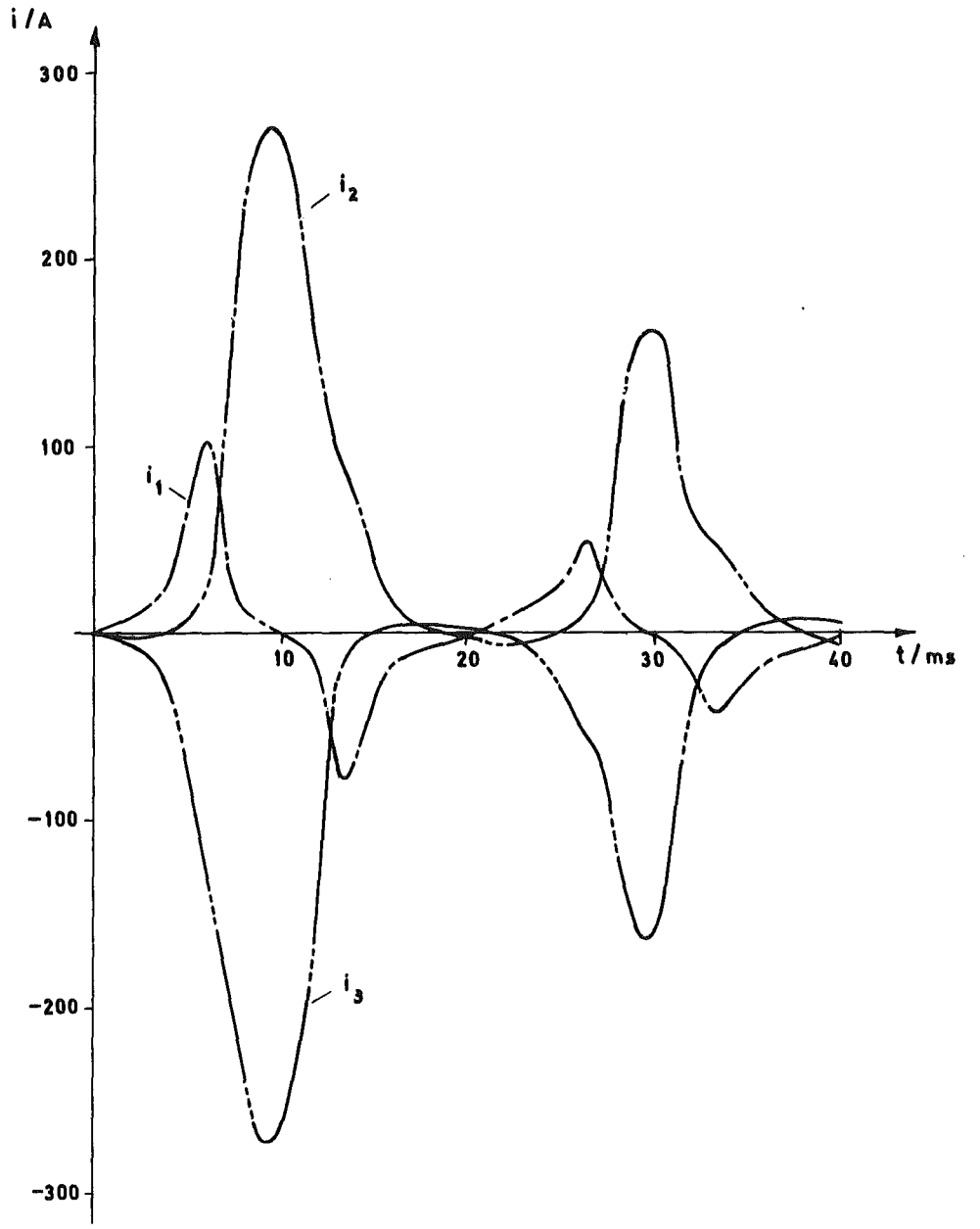
Bild 7 Beispiel : 3-phasige nichtlineare Drossel

7.1 "Normale Schaltung"

7.2 Strukturbild

7.3 Zeitdiagramme
für: $\hat{v}$ = 311V R_v = 0,2Ω
f = 50Hz magnetische Kennlinie K1

7.3



Dipl.-Ing. D. Büge
Dr.-Ing. H.G. Siepmann
VW Wolfsburg

Aufgaben und technische Lösung der rechnergestützten Simulation bei VW

Fahrsimulatoren können einem Fahrer -entsprechend dem technischen Aufwand- ein genaues und bis in den Fahrgrenzbereich richtiges Fahrgefühl vermitteln. Grundlagen dafür sind einerseits das genaue und den Grenzbereich erfassende theoretische Modell des Fahrzeugs und zum anderen die exakte Vermittlung all der berechenbaren Eindrücke -Sicht, Geräusch, Beschleunigungskräfte-, die auch im fahrenden Fahrzeug auf den Fahrer wirken.

Die im Rahmen der Automobilforschung und -entwicklung nutzbaren Vorteile solcher Simulationseinrichtungen liegen

1. in der Darstellungsmöglichkeit beliebiger, parametrisch bestimmter Fahrzeuge (Wiedergabe eines beliebigen Fahrzeugverhaltens) und

2. in der genauen Meßbarkeit von Größen, die das Fahrverhalten kennzeichnen.

Die damit lösbaren Hauptaufgaben sind entsprechend

1. die technische Weiterentwicklung eines bestehenden Fahrzeugkonzeptes unter Einschränkung des teuren Prototypenbaues.

2. die Entwicklung eines optimalen Fahrzeugkonzeptes anhand von Reihenfahrversuchen und statistischer Auswertung der subjektiven und objektiven Angaben und Meßdaten über das Fahrverhalten der Versuchspersonen auf verschiedenen simulierten Fahrzeugmodellen.

(Im April 1973 wurde eine solche Reihenuntersuchung mit anderer Zielsetzung, nämlich der 'Untersuchung des Fahrerverhaltens unter Alkoholeinfluß', durchgeführt)

Das technische Hilfsmittel zur Vermittlung der Fahreindrücke ist der Fahrsimulator.
In diesem Fall handelt es sich dabei um eine Kabine, die hydraulisch um die drei Drehachsen bewegt werden kann.

Der Fahrer wird in eine solche Lage gedreht, daß die ja immer wirkende Schwerkraft richtungsmäßig die Gesamtbeschleunigungskraft abhängig vom errechneten Fahrzustand wiedergibt.

Ein Schwarzweißmonitor stellt den für jeden Beobachtungspunkt -z.B. auch solche abseits der Straße- richtigen Sichteindruck her, ein quadrophones System die Geräuscheindrücke.

Zur Lenkmomentsimulation ist ein Torquemotor eingesetzt.
Die Bedienelemente und die Anzeigeinstrumente entsprechen denen eines VW 412.

Das technische Hilfsmittel zur Darstellung des theoretischen Fahrzeugmodells (Antrieb, Fahrzeugdynamik) unter der Echtzeitbedingung, das die Signale zur Ansteuerung des Fahrsimulators liefern muß, ist zur Zeit eine Analogrechenanlage der Firma HSI (drei Konsolen SS 1oo, 3oo Operationsverstärker).

Weitere Merkmale der Analogrechenanlage sind:

kartenprogrammierte Funktionsgeber, Digitalpotentiometer, Resolver.

Trotz besonderer Rechentechniken (z.B. analoges Multiplexing bei der Funktionsdarstellung) zwingen die vielen nichtlinearen Zusammenhänge zu Modellvereinfachungen.

Eine bessere Unterstützung (-Verbesserung des Modells und Unterstützung der Versuchsabwicklung-) der Simulation gelingt beim Einsatz zusätzlicher Digitalkapazität im Verbund mit der Analogrechenanlage, wie es ab Juli 1973 geplant ist.

Dann übernimmt eine analogrechnernah aufgestellte IBM/7-Anlage die Organisation der Datenübertragung zwischen den beiden Hauptkomponenten der Hybriddatenverarbeitung, nämlich der schon erwähnten Analogrechenanlage HSI-SS 1oo und dem ca. 3 km entfernt stehenden Mehrzweckrechner IBM/36o - 195.

Die 4o.8 KBaud TP-Leitung dient der Übertragung der digitalisierten Problemdaten zwischen Analogrechenraum und IBM - Großrechner, die 6oo Baud TP-Leitung der Betriebssteuerung und Programmierung der IBM/7 durch das Großsystem IBM/36o-195.

Die Kopplung des Analogrechners HSI - SS 1oo mit dem Digitalsystem (Daten und Steuerung) erfolgt mit Hilfe der IBM/7 Prozeßperipherie im skizzierten Umfang.

Die zur Unterstützung des Hybridbetriebs einzusetzende Software muß der speziellen Hardware-Situation angepaßt werden.

Im Großsystem IBM/36o garantiert ein sich dem IBM - Betriebssystem MVT überlagernder Real-Time-Monitor die höchstpriorisierte Interruptfähigkeit vom Analogsystem her.

Ein spezielles Betriebssystem in der IBM/7 übernimmt gesteuert vom IBM/36o System die Datenübertragunsorganisation zwischen Analogrechner und Digitalgroßrechner.

Im ersten Stadium des Hybridbetriebes wird auch in Anpassung an die begrenzte Datenübertragunskapazität zu einer Zeit nur die Abwicklung eines Hybridjobs möglich sein.

Engpässe dieser technischen Lösung sind die Problemdaten-Übertragungskapazität (ca. o.5 ms/Wert) und die Zahl und Genauigkeit der DA-Wandler.

Für einen Hybridbetrieb unter voller Ausnutzung der AD- und DA - Kapazität ergibt sich damit bei Vernachlässigung der eigentlichen Rechenzeit eine minimale FRAME- Zeit von T_T = 2o ms. (bei Problemsignalfrequenzen von fs = 5 Hz $\triangleq$ T_s = 2oo ms)

Durch Prediktion müssen stabilitätsgefährdende Phasenverschiebungen kompensiert werden.

Die Möglichkeit dazu besteht z.B. durch die prediktive Aufbereitung der Eingabedaten für das Digitalprogramm in der IBM/7; denn die Analogdateneingabe in die IBM/7 erfolgt wesentlich schneller (o,o5 ms/Wert) als die anschließende Übertragung zum Großsystem (o,5 ms/Wert).

Das vorstehend beschriebene System stellt eine Lösungsmöglichkeit bzw. im Hinblick auf die Entwicklung eine Gruppe solcher Möglichkeiten dar. Die Lösung ist dabei eindeutig bestimmt durch die technischen Gegebenheiten und die Forderung nach Begrenzung des zusätzlichen Aufwandes und nur unter diesem Gesichtspunkt optimal.

Simulationsprobleme im Bereich des Maschinenbaus und der Reaktortechnik

Dr. E. G. Schlechtendahl
Institut für Reaktorentwicklung
Gesellschaft für Kernforschung mbH.,
75 Karlsruhe
Postfach 3640

1. Im Bereich des Maschinenbaus und der Reaktortechnik sind häufig Prozesse mit meist kontinuierlich veränderlichen Zustandsgrößen zu simulieren. Die typische Darstellungsform derartiger Prozesse ist ein System gekoppelter gewöhnlicher Differentialgleichungen.

2. In vielen Anwendungsfällen werden hierzu Programme erstellt, in denen

   die Beschreibung des Prozesses und
   die Algorithmen zur Simulation

   eng miteinander verkoppelt sind. Lediglich die Festlegung der Prozessparameterwerte ist normalerweise davon getrennt. Von Simulationsprogrammen im engeren Sinne wird jedoch dort gesprochen, wo auch die ersten beiden Bestandteile der Simulationsaufgabe getrennt sind. Dabei haben sich besonders solche Programme bewährt, die nicht die für einen Analogrechner übliche Darstellung eines Blockschaltbildes (wie z.B. in DAS2[1]), sondern eine gleichungsorientierte, FORTRAN-ähnliche Prozessbeschreibung erwarten (DYSYS [2]). Das System CSMP [3] enthält Bestandteile beider Darstellungsweisen.

3. Einige der Probleme, die bei Simulationsprogrammen im engeren Sinne noch vor wenigen Jahren häufig auftauchten, sind heute in den meisten Systemen gelöst: so z.B. Simulation von Transportvorgängen [1,2,3], Stabilitäts- und Genauigkeitsprobleme (durch Schrittweitenautomatik [1,2,3]) numerische Ungenauigkeiten in der Bestimmung des Ausgangszustandes (durch Konsistenziteration [2]) oder Anpassung der Kernspeichererfordernisse an die jeweiligen Problemgröße (durch Precompiler [3] oder Erstellung eines besonderen Hauptprogrammes

gemäß Abb.1). Es bleiben jedoch eine Reihe von Problemen übrig, die auch heute noch in vielen Anwendungsfällen es nötig machen, Sonderprogramme zur Simulation bestimmter Prozesse zu erstellen. Diese Probleme hängen meist damit zusammen, daß die Simulationsalgorithmen oft nicht für alle Teilbereiche des Gesamtprozesses (sowohl strukturell wie auch zeitlich gesehen) geeignet sind. Ein typisches Beispiel ist der auf den Boden fallende und zurückprallende Ball [4]. Hier ist zum Zeitpunkt des Stoßes die sonst geltende Differentialgleichung zu ersetzen durch eine Stoßgleichung, die eine diskrete Zustandsänderung beschreibt. An einem anderen Beispiel, das schon eher praktische Bedeutung hat, soll dieses Problem näher erläutert werden. Es handelt sich dabei um einen Kranbalken, an dessen Ende eine Winde eine schwere Last anhebt, bis das Kranseil eine minimale freie Länge erreicht hat. In diesem Beispiel treten zwei Stoßvorgänge auf: 1. in dem Augenblick, in dem das zuvor schlaffe Seil sich strafft und die Masse vom Boden abhebt, 2. zum Zeitpunkt des Anhaltens der Winde. Eine Dämpfung ist nicht berücksichtigt.Als Zustandsgrößen zur Beschreibung des Problems wählt man zweckmäßigerweise (neben anderen) den Impuls IMPULS des Gesamtsystems und den Impuls IMPB des Balkens allein. Es gilt nun für schwebende Last die Differentialgleichung

$$\frac{d\ \text{IMPULS}}{dt} = \text{Summe aller Kräfte/Gesamtmasse}$$

und für aufliegende Last die algebraische Gleichung

$$\text{IMPULS} = \text{IMPB}$$

Das Problem besteht darin, daß bei jedem numerischen Integrationsverfahren der Systemzustand nur zu diskreten Zeiten definiert ist und daß mit großer Wahrscheinlichkeit die obige Änderung der Gleichungsstruktur zwischen zwei derartigen Stützstellen eintritt. Damit entfällt die jedem Integrationsverfahren zugrundeliegende Hypothese der Stetigkeit des Funktionsverlaufes zwischen den Stützstellen. In DYSYS wird folgendermassen vorgegangen.
Immer wenn zu einem Zeitpunkt t_i, der eine Stützstelle darstellt, der Wert y^+ einer Variablen y nach Aufruf der Problemroutine DYNAMO von dem Wert y^- dieser Variable vor diesem Aufruf abweicht, so gilt eine algebraische Gleichung anstelle einer vermuteten Differentialgleichung. Dies führt dazu, daß zum Zeitpunkt t_{i-1} anstelle der dort herrschenden Ableitung $\left(\frac{dy}{dt}\right)_{t_{i-1}}$ eine Ersatzableitung der Größe

$$\left(\frac{dy}{dt}\right)_{t_{i-1},\text{Ersatz}} = \frac{y_i^+ - y_{i-1}}{t_i - t_{i-1}}$$

angenommen wird. Diese Vorgehensweise bewirkt sowohl beim Übergang von Differential-

gleichung zu algebraischer Gleichung wie auch in umgekehrter Richtung daß die übliche Schrittweitenautomatik auch an diesen Übergangsstellen die Genauigkeit der Lösung kontrollieren kann. Ein Nachteil dieses Verfahrens ist, daß alle Zustandsvariablen bei jedem Integrationsschritt auf derartige Änderungen überwacht werden.
Abb.2 zeigt die Darstellung des hier gebildeten Simulationsproblems als Fortransubroutine für DYSYS [2]. Die Abbildungen 3 bis 5 zeigen deutlich die Auswirkungen der Stöße auf das Systemverhalten für ein Beispiel.

5. Ein Problem, das von Simulationssystemen bislang nicht abgedeckt wird, bieten die sogenannten "stiff equations", obwohl Methoden hierfür vorhanden sind [5]. Es kommt vor, daß innerhalb eines Systems von Differentialgleichungen eine Gruppe von Zustandsgrößen durch in sich stabile Gleichungen mit derart kurzen Zeitkonstanten beschrieben wird, daß sie stets nahezu im Gleichgewicht mit den restlichen Zustandsgrößen sind (quasistationär). Wenn sie dennoch vom gleichen Simulationsalgorithmus behandelt werden, erzeugen sie einen unnötig hohen Rechenaufwand dadurch, daß sie eine Schrittweite in der Größenordnung der kleinsten Zeitkonstanten erzwingen. In Spezialprogrammen wird dieses Problem dadurch gelöst, daß Gruppen von Differentialgleichungen vergleichbarer Zeitkonstanten zusammengefaßt und unabhängig von den restlichen integriert werden [6]. Dabei ist stets von den"schnellen" zu den "langsamen" vorzugehen und eine besondere Synchronisierung an gemeinsamen Stützstellen nötig. Bislang bieten allgemein verwendbare Simulationsprogramme hierfür keine Unterstützung.

6. Ein weiteres Problem liegt darin, daß für partielle Differentialgleichungen bestimmte Typen (Wärmeleitungsgleichung)spezielle günstige Algorithmen [6] bekannt sind, deren Anwendbarkeit verloren geht, wenn man durch räumliche Diskretisierung daraus einen Satz gewöhnlicher Differentialgleichungen bildet. Das System PDEL [7]ist speziell zur Simulation partieller Differentialgleichungen entwickelt worden, bietet aber nicht die Möglichkeit, darin gleichzeitig auch gewöhnliche Differentialgleichungen zu behandeln.

7. Die heute existierenden Simulationsprogramme sind im praktischen Einsatz Spezialprogrammen vor allen Dingen dadurch unterlegen, daß die von Ihnen

verfügbar gemachten Algorithmen nur Modelle sehr einfacher Struktur zulassen. Neuentwicklungen sollten besonders die Möglichkeiten bieten, "hybride Modelle" zu erstellen, d.h. Modelle, die mit unterschiedlichen Algorithmen arbeiten, zu synchronisieren.

Literatur

[1] Koepp, C.
DAS 2 - Ein dynamischer Simulator mit Totzeitgliedern für Digitalrechner
KFK 1142, EUR 3695,(1970)

[2] Schlechtendahl, E.G.
DYSYS - A Dynamic System Simulator for Continuous and Discrete Changes of State
KFK 1209, (1970)

[3] CSMP - Continuous System Modelling Program
IBM

[4] Schlechtendahl, E.G.
DYSYS 71, ein System für die integrierte Simulation kontinuierlicher Prozesse, diskrete Ereignisse und von Transportvorgängen
Angewandte Informatik 3/1972, p.115-120

[5] Gear, C.W.
The Automatic Integration of Ordinary Differential Equations
Comm.ACM 14, 3(1971),p.176-190

[6] Kessler, G.
Zur numerischen Lösung der ortsabhängigen dynamischen Gleichungen schneller Brutreaktoren mit Hilfe eines Variationsprinzipes
KFK 781/I,EUR 3957 d(1968)

[7] Cardenas, A.F.; Karplus, W.F.
Design and organisation of a translator for a partial differential equation language
AFIPS Conf.Proc.(1970),p.513-523

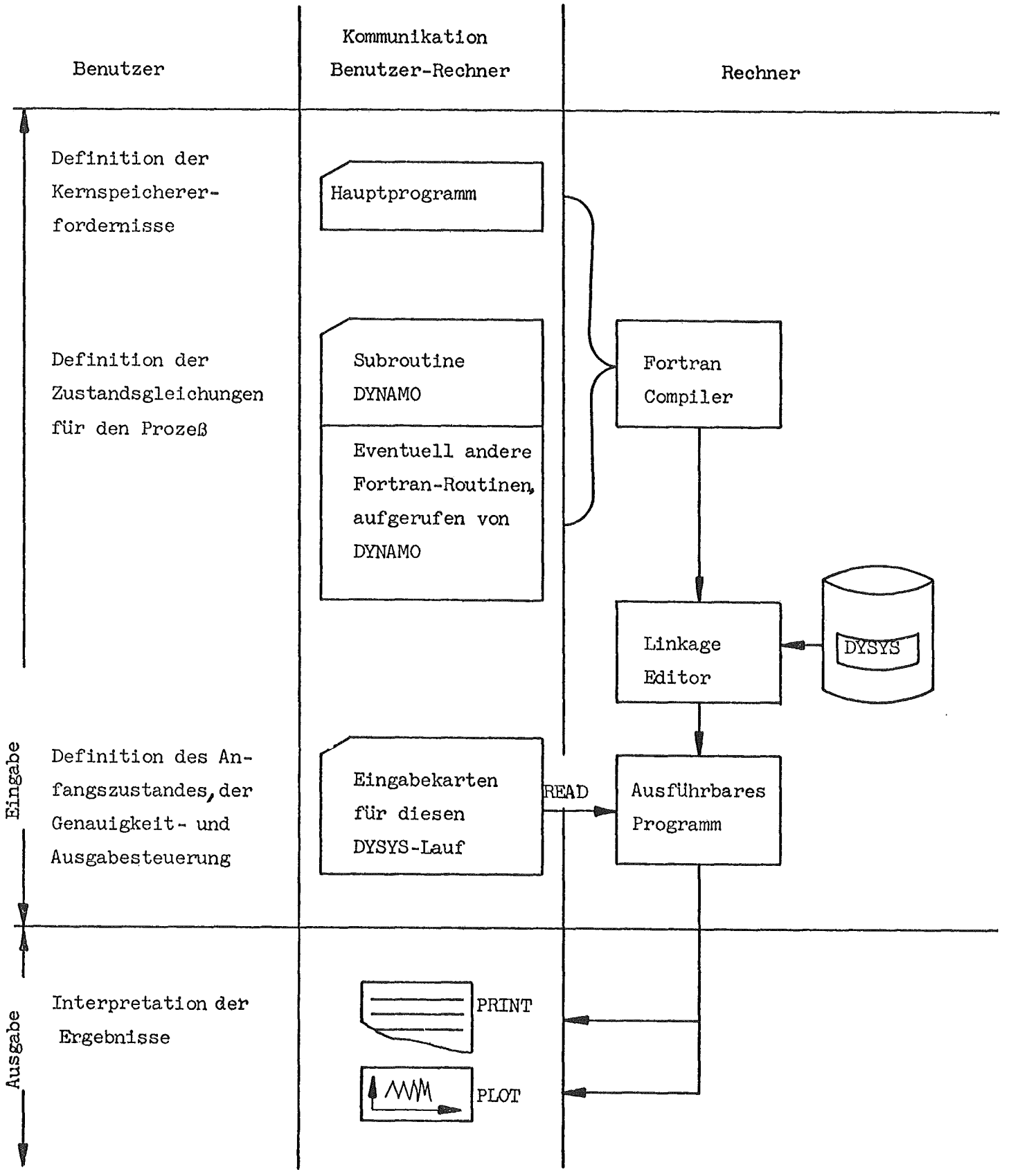


Abb.1: Ablauf einer Simulation mit DYSYS

```
0002            SUBROUTINE DYNAMO

                SEIL   SEILLAENGE
                VSEIL  AUFZUGGESCHWINDIGKEIT DES SEILES
                V      GESCHWINDIGKEIT DES BALKENENDES
                Y      AUSBIEGUNG DES BALKENS
                YL     HOEHE DER LAST
                IMP    IMPULS DES BALKENS OHNE LAST
                IMPULS IMPULS DES GESAMTSYSTEMS
0003            COMMON /INTVAR/ ZEIT,IMPULS,IMPB,Y,YL,SEIL
               1,V,YH
0004            COMMON /DERIV/ DIMPUL,DIMPB,DY,DYL,DSEIL
0005            COMMON /DATA/ HOEHE,LAST,SMIN,MRED,C,VREL
0006            REAL KRAFT,MASSE,MRED,LAST,IMPULS,IMPB

0007            IF (ZEIT.GT.0) GO TO 1
0009            G = -9.81
0010            Y = G * MRED / C
0011          1 CONTINUE
0012            IF (SEIL.GT.SMIN) GO TO 5
                   DIE FOLGENDEN GLEICHUNGEN GELTEN BEI STEHENDER WINDE
0014            VSEIL = 0.
0015            SEIL = SMIN
0016            GO TO 2
                   DIE FOLGENDEN GLEICHUNGEN GELTEN BEI LAUFENDER WINDE
0017          5 VSEIL = VREL
0018          2 CONTINUE
                   DIE FOLGENDEN GLEICHUNGEN GELTEN STETS
0019            DSEIL = - VSEIL
0020            YL = HOEHE + Y - SEIL
0021            IF (YL.LE.0) GO TO 3
                   DIE FOLGENDEN GLEICHUNGEN GELTEN BEI SCHWEBENDER MASSE
0023            V = (IMPULS - LAST * VSEIL) / (MRED + LAST)
0024            DIMPUL = - C * Y + G * (MRED + LAST)
0025            IMPB = V * MRED
0026            DYL = V + VSEIL
0027            DIMPB = - C * Y + G * MRED
0028            GO TO 4
0029          3 CONTINUE
                   DIE FOLGENDEN GLEICHUNGEN GELTEN BEI AUFLIEGENDER MASSE
0030            YL = 0.
0031            IMPULS = IMPB
0032            V = IMPULS / MRED
0033            DIMPB = - C * Y + G * MRED
0034            DIMPUL = DIMPB
0035            DYL = AMAX1 (0.,V + VSEIL)
0036          4 DY = V
0037            YH = HOEHE + Y
0038            RETURN
0039            END
```

Abb.2: Unterprogramm zur Simulation des Balkenproblems

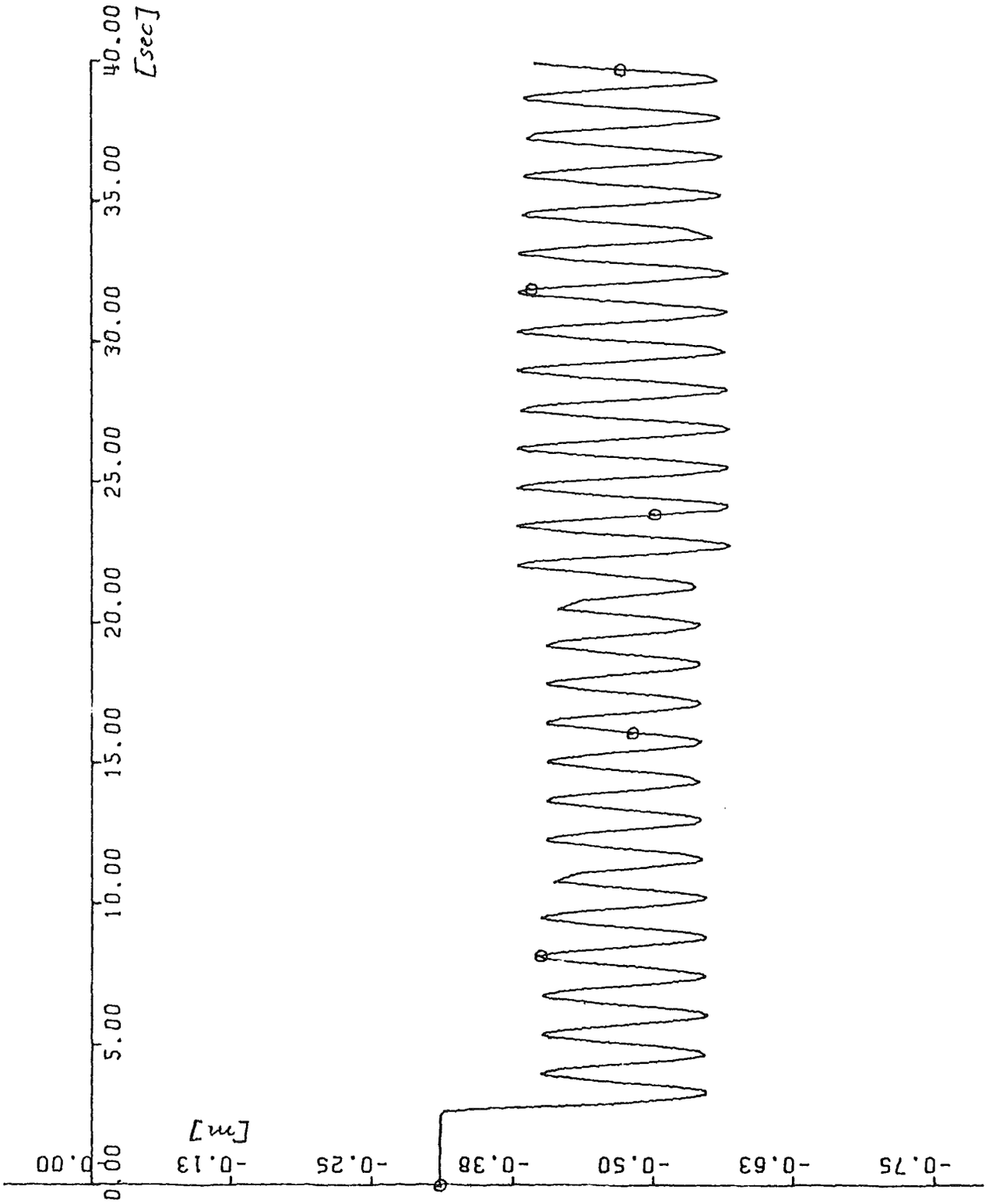


Abb.3: Auslenkung des Balkenendes

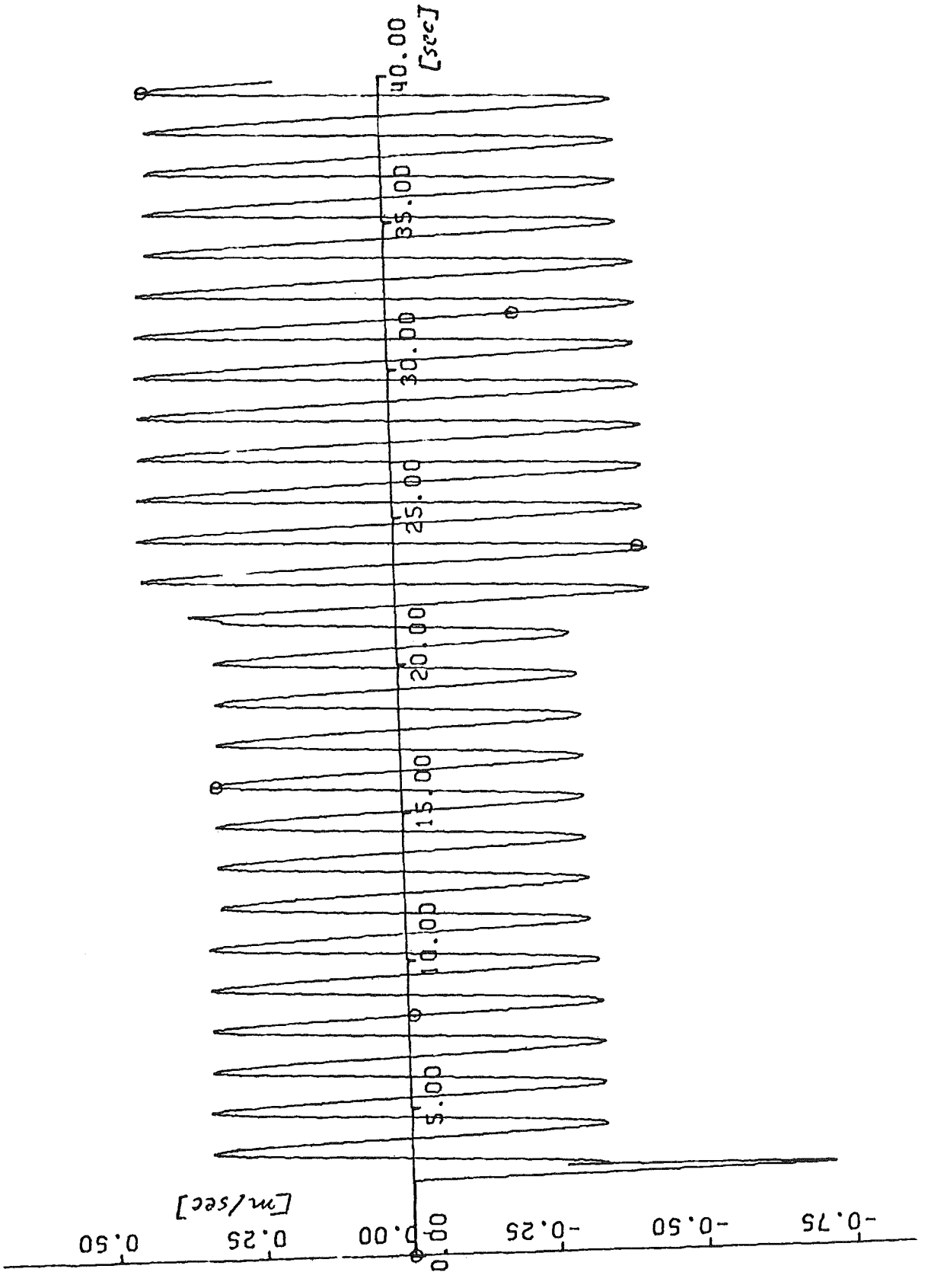


Abb.4: Geschwindigkeit des Balkenendes

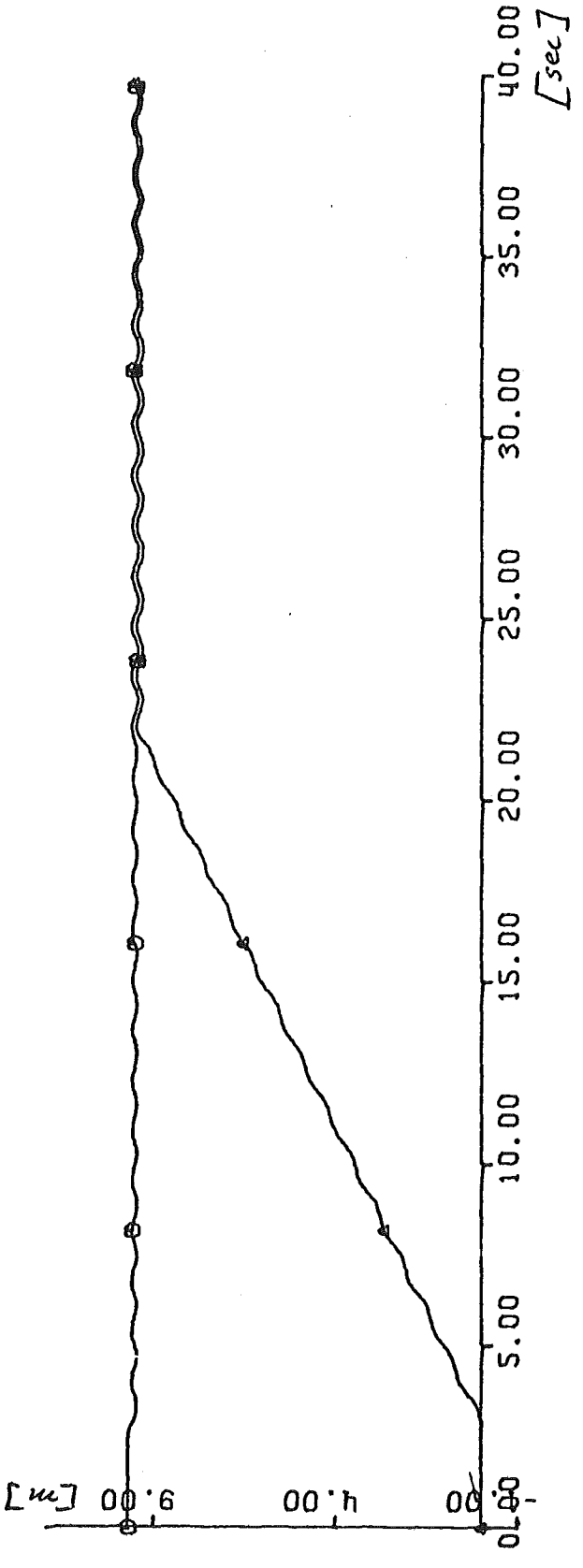


Abb.5: Verlauf von Last und Balkenende

Dipl. Ing. Volker Hecht

Themenkomplex: Entwurf und Implementierung
von Simulationsmodellen

Thema: Einheitliche Programmierung eines Hybrid-System auf problemorientierter Ebene für die Simulation von Echtzeitproblemen

1. Einleitung

Obwohl die Simulation vieler Prozesse auch auf entsprechenden Digitalrechnern in Echtzeit durchgeführt werden kann, ist häufig der Einsatz eines Hybrid-Systems - bestehend aus einem Digital- und einem Analogrechner - vor allem aus Kostengründen sinnvoll. Nur ausreichend schnelle und daher meist grosse Digitalrechner können zur Lösung von Echtzeitproblemen verwendet werden, ob dann ein solcher Grossrechner auch sonst immer ausgenützt werden kann, erscheint bei vielen Anwendern fraglich. Eine echte Alternative kann das Hybrid-System bieten, das mit einem wesentlich kleineren Digitalrechner auskommt und dazu noch die bekannten Vorteile des Analogrechners wie

parallele Arbeitsweise
echte Integration
Austauschbarkeit mit Echtteilen
1 : 1 - Zuordnung zwischen analogem Modell und realem System und enge Mensch-Maschine-Beziehung

aufweist.

Durch die Ankopplung eines Analogrechners werden aber nicht nur zusätzliche Vorteile gewonnen, sondern auch zusätzliche Nachteile müssen in Kauf genommen werden. Spezifische Analogrechnerprobleme wie

Erstellen der Rechenschaltung und
Skalierung

sowie eine geeignete Problemaufteilung auf die beiden Rechner, die für den sinnvollen Einsatz eines Hybrid-Systems Voraussetzung ist, tragen dazu bei, dass Hybrid-Systeme fast ausschliesslich Hilfsmittel für Benutzer mit Spezialkenntnissen sind. Nur bei einer gemeinsamen Programmierung des Hybrid-Systems können neue Benutzer hinzugewonnen werden, die sich dann lediglich in besonderen Fällen, z.B. in Fehler - oder extra gewünschten Situationen mit dem angekoppelten Analogrechner zu befassen haben. Voraussetzung ist dann aber, dass das Programmiersystem in der Lage ist, dem Benutzer die Lösung dieser speziellen, von der Kopplung und den Eigenschaften des Analogrechners herrührenden Aufgaben weitgehend abzunehmen.

2. Die Simulationssprache

Unter den bekannten Sprachentwicklungen für die Simulation zeitkontinuierlicher Systeme erscheinen die CSSL-Sprachen (Continuns System Simulation Languages [1]) für die Programmierung des gesamten Systems als hybride Quellensprache (Hybrid Source Language) besonders günstig. Mit dieser Sprache lassen sich sowohl die Struktur eines Modelles als auch das Ablaufschema der eigentlichen Simulation leicht beschreiben. Ein wichtiges Kennzeichen der CSSL-Sprachen ist das Makro-Konzept moderner Assemblersprachen, denn in Form von Makros lassen sich die analogen Operatoren bequem nachbilden. Da keine Unterprogrammsprünge notwendig sind, wird eine schnelle Simulationsausführung erreicht. Bei den üblichen Makro-Konzepten wird bei einem Makroaufruf eine komplette, durch den Makro-Namen bezeichnete Befehlsfolge an der entsprechenden Stelle eingefügt. Als Nachteil ist dann aber bei einer schnelleren Simulationsausführung ein höherer Speicherplatzbedarf zu verzeichnen. Durch Hinzufügen von benutzereigenen Makros kann die Sprache für individuelle Probleme beliebig erweitert werden.

Ein einfaches Beispiel ist die Darstellung eines Potentiometers (Multiplikation mit einem konstanten Faktor):

```
MACRO POT  (OV = A, IV)
OV = A * IV
ENDMAC
```

In einem Makro werden im Gegensatz zu Unterprogrammen Zusammenhänge zwischen Variablen, anstatt zwischen Zahlenwerten formuliert. Dabei werden die Ausgangsvariablen durch eine Verknüpfungsvorschrift aus den Eingangsvariablen gewonnen, die in dem Makro definiert wird. Deshalb stellen die Makros ein geeignetes Element zur Strukturbeschreibung von Simulationsmodellen dar. Wie oft ein Makro zu durchlaufen ist, kann aus diesem nicht entnommen werden. Auf dem Analogrechner werden Makros parallel bearbeitet, während auf dem Digitalrechner die Bearbeitung in einer quasiparallelen Folge durchgeführt wird.

Der Benutzer hat weiter die Möglichkeit, sich eine eigene Makro-Bibliothek zu schaffen. Damit steht eine Programmierhilfe ähnlich den Unterprogrammen bei höheren Sprachen zur Verfügung.

Ein wesentlicher Gesichtspunkt bei der Auswahl einer geeigneten Quellensprache ist die Forderung, dass die ausgezeichneten Möglichkeiten eines Dialogs, wie sie vom Analogrechner her bekannt sind, weiterhin in entsprechenden Fällen benützbar sein sollen.

Diesen Überlegungen kommen die CSSL-Standards entgegen, denn sie enthalten Sprachelemente, die es dem Benutzer gestatten, wahrend eines Simulationslaufs direkte Eingriffe vorzunehmen. Sollen diese Eingriffsmöglichkeiten zu einem echten Dialog benützt werden, dann wird eine Anzeige benötigt, die unmittelbar Auskunft über das aktuelle Verhalten von entsprechenden Systemgrössen gibt. Diese Zusammenhänge können in Form von Kurven z.B. auf Datensichtgeräten anschaulich dargestellt werden. Da die Sichtgeräte für Simulationen in Echtzeit bei schnellen Vorgängen den extremen Zeitbedingungen nicht mehr genügen, bietet sich stattdessen die Verwendung des Oszillographen des Analogrechners an.

Die Dialogfähigkeit - gekoppelt mit entsprechenden Abfrage-Statements - kann ausgenützt werden, um verschiedene Parameter des analogen Teilmodelles durch Potentiometereinstellungen direkt zu verändern, ohne dass dafür spezielle digitale Routinen benötigt werden.

Eine besondere Verbesserung der Ausdrucksfähigkeit wird durch die Möglichkeit von Sprachanschlüssen an prozedurale Sprachen wie FORTRAN oder PL/1 erreicht. Dies bedeutet, dass die Eigenschaften einer prozeduralen Sprache z.B. FORTRAN, durch Hinzufügen von problemorientierten Operatoren (z.B. für die Integration), einer problemorientierten Syntax (z.B. vom Benutzer definierte Makros) und von Organisationsmerkmalen (z.B. Sortierung) wesentlich erweitert und auf den speziellen Problemkreis zugeschnitten werden. Neben komplizierteren Sachverhalten, wie z.B. logischenBedingungen, können damit auch vorteilhaft Input/output Vorgänge in der prozeduralen Sprache beschrieben werden.

Bei der Wahl einer CSSL-kompatiblen Sprache sind ausserdem die zur Verfügung stehenden Übersetzer ein wichtiger Gesichtspunkt (siehe 4.). Die Zielsprache der bekannten Übersetzer ist i.a. FORTRAN, d.h. aus dem in der Quellensprache abgefassten Programm wird ein gültiges FORTRAN-Programm erzeugt. Ist auch der Übersetzer wiederum in FORTRAN abgefasst, dann ist man nahezu maschinenunabhängig.

3. Das Programmiersystem

In [2] wurden vier verschiedene Sprachebenen vorgestellt, die den Strukturen eines Hybrid-Systems angepasst sind. (Bild 3.1) Mit diesen Begriffen stellt die erwähnte Problemaufteilung eine Übersetzung aus der hybriden Quellensprache in eine unterteilte Quellensprache (Allocated Source Language) dar.

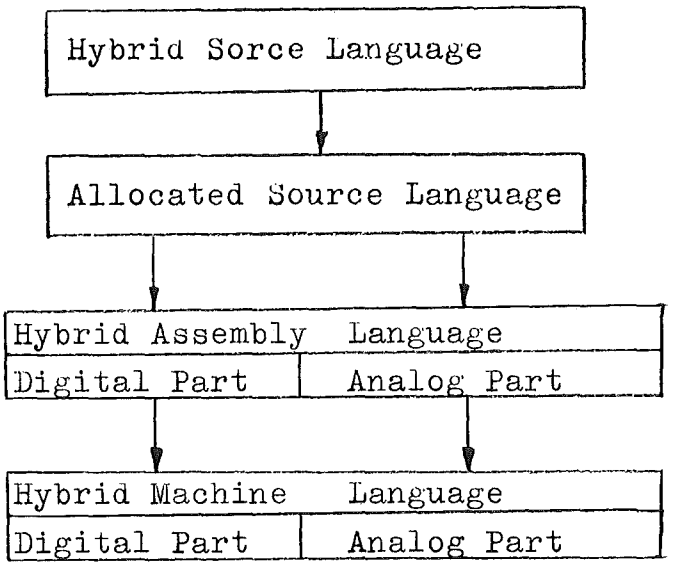


Bild 3.1: Sprachebenen des Programmiersystems

Die Schwierigkeiten dieser Übersetzung können umgangen werden, indem man dem Benutzer die Entscheidung überlässt, wie er sein Problem aufteilen will. In diesem Fall ist es dann unumgänglich, dass der Benutzer zumindest über Grundkenntnisse der Arbeitsweise eines Analogrechners verfügen muss. Bei einem komfortableren Programmiersystem sollte diese Systemaufteilung automatisch erfolgen, wobei aber nach wie vor spezielle Benutzerwünsche zugelassen sind. Bei der Strategie der Systemaufteilung sind verschiedene Gesichtspunkte zu berücksichtigen. Der endgültige analoge Systemanteil darf bezüglich Art und Anzahl nur jene Rechenelemente enthalten, die auf dem Analogrechner verschaltbar sind. Ausserdem liegt der benötigten Systemaufteilung die Überlegung zugrunde, möglichst grosse Teile des zu simulierenden Gesamtmodelles auf dem Analogrechner zu bearbeiten.

Diese Strategie steht im Gegensatz zu anderen Ansätzen, bei denen nur wenige Verknüpfungen auf dem Analogrechner realisiert werden sollen, damit auch ein einfacheres Patching- System ausreichend ist. Überlegt man sich aber, dass der Transfer der Daten zwischen Analog- und Digitalrechner und umgekehrt auf jeden Fall zusätzlich Zeit kostet, dann muss jene Zeitersparnis, die man durch analoge Ausführung der Operationen gewinnt, diesen Zeitverlust mindestens ausgleichen.

Zieht man in einer Zeitbilanz das einfache numerische Euler-Integrationsverfahren, für das zwei Operationen benötigt werden, nämlich Multiplikation mit der Schrittweite und Addition zum

alten Wert, zum Vergleich heran, so ergibt sich bei analoger Bearbeitung einer Differentialgleichung 2. Ordnung mit 2 Koeffizienten ungleich null

$$y'' + a_1 y' + a_0 y = 0 \quad (a_1 \neq 0,\ a_0 \neq 0) \qquad (3.1)$$

folgende Zeitbilanz: insgesamt benötigt man 2 Multiplikationen mit einem konstanten Faktor und 2 Additionen, hinzu kommen 2 Multiplikationen und 2 Additionen zur Durchführung der Integrationen.

Geht man davon aus, dass der Integrand auf dem Digitalrechner schon zur Verfügung steht, dann braucht man z.B. auf dem TR 86 von Telefunken im günstigsten Falle bei Festkommazahlen für die digitale Integration pro Schritt rund 42 μs.

Führt man diese Integration auf dem Analogrechner durch, dann muss der Wert des Integranden vom Digitalrechner an den Analogrechner übertragen werden. Dafür erspart man sich die Zeit für die digitale Berechnung eines Integrationsschrittes, im Beispiel 2 Multiplikationen und 2 Additionen. Damit kann gerade die zusätzlich benötigte Zeit für einen einzigen Datentransfer kompensiert werden.

Neben diesen Randbedingungen, die durch die Hardware des Systems gegeben sind, sollten die aus der Systemaufteilung resultierenden digitalen und analogen Systemteile in ihrer speziellen Problemstellung auf die Eigenschaften der einzelnen Rechner zugeschnitten sein.

Teilsysteme mit hohen Eigenfrequenzen sind vorzugsweise analog zu bearbeiten. Die Grösse der Zeitkonstanten bei der Integration und eine schon vorher bekannte numerische Instabilität bei digitalen Integrationsverfahren sollten hier zur Auswirkung kommen.

Die CSSL-Sprachen sehen mehrere Möglichkeiten vor, das zu simulierende Modell zu beschreiben. So ist eine analogrechner- oder differentialgleichungsorientierte Strukturbeschreibung möglich. Eine automatische Systemaufteilung muss von der Art der Strukturbeschreibung unabhänging sein, deshalb sind die in der hybriden Quellensprache enthaltenen Strukturaussagen durch eine Zwischenübersetzung in eine einheitliche syntaktische Form zu überführen.

Als Elemente der Zwischensprache eignen sich Grundverknüpfungen, wie sie von den einzelnen analogen Rechenelementen ausgeführt werden können, denn nur dann lassen sich die verschiedenen Kriterien der Systemaufteilung bequem ansetzen, die auf der analogen Rechenschaltung basieren. Da wegen der von den Operationsverstärkern hervorgerufenen Vorzeichenumkehr keine eindeutige Abbildung von der Differentialgleichung in eine Rechenschaltung erfolgen kann, ist es ökonomischer, sich erst nach erfolgter Systemaufteilung mit dem Vorzeichenproblem zu befassen.

Zur Beschreibung der Zwischensprache eignet sich eine Darstellung als gerichteter endlicher, zusammenhängender Graph $\vec{G}$ (P,K), mit P der Menge aller Knoten und K der Menge aller gerichteten Kanten. Bild 3.2 zeigt ein Beispiel für eine Differentialgleichung 2. Ordnung

$$\frac{dx^2}{dt^2} + a\,\frac{dx}{dt} + bx = 0 \qquad (3.2)$$

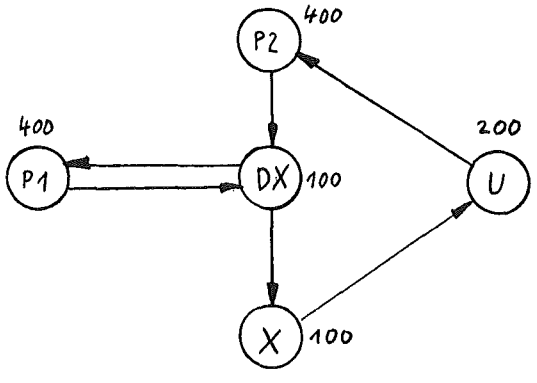


Bild 3.2: Graph der Zwischensprache

Dabei entsprechen den Knoten Variable und die Kanten verbinden jeweils zwei an einer Verknüpfung beteiligten Variable. Die Richtung der Kante gibt an, welche Variable durch Anwendung der Verknüpfungsvorschrift aus dem Vorgängerelement entsteht. Gibt es für einen Knoten Pj mehrere Kanten Kij, die verschiedene Vorgängerelemente Pi mit Pj verbinden, dann bedeutet dies, dass mehrere Vorgängerelemente gemeinsam an dieser Verknüpfung beteiligt sind.

Die Art der Verknüpfung wird durch die Bewertung eines Knoten ausgedrückt. Die Bewertung des Nachfolgerelementes gibt an, durch welche Verknüpfung er aus den Vorgängerelementen entsteht. Die Bewertungsziffer ist eine reelle Zahl $d_i = D(P_i)$ mit $di > 0$.

Die Übersetzung in die unterteilte Quellensprache erfolgt nun in einem zweiten Schritt, indem auf die Zwischensprache des 1. Schrittes die verschiedenen Kriterien der Systemaufteilung in Form von graphentheoretischen Zusammenhängen angewendet werden.

Liegt nun für die weitere Bearbeitung der Umfang des analogen und digitalen Teilsystems fest, dann muss für den analogen Systemteil die äquivalente Rechenschaltung erstellt werden. Dabei ist auch die Vorzeichenumkehr der Operationsverstärker zu berücksichtigen. Die Anzahl der erforderlichen Umkehrer kann zumindest durch Abzählen der innerhalb einer geschlossenen Schleife aneinandergereihten Operationsverstärker reduziert werden. Steht ein leistungsfähiges Automatic-Patching-System zur Verfügung, dann wird durch dieses das Erstellen der Rechenschaltung einschliesslich der Verschaltung der Datenübertragungskanäle zwischen Analog- und Digitalrechner übernommen. Da von einem Patching-System lediglich eine Untermenge aller am Analogrechner herstellbaren Verbindungen verschaltet werden kann, ist zwischen Patching-System und Teilungsalgorithmus eine Kommunikation vorzusehen. Bei einem einfacheren Programmiersystem ohne Patching-Algorithmus muss der Benutzer anhand einer vom System gelieferten Liste selbst die Rechenschaltung erstellen. Eine automatische Überprüfung, ob die gesteckte Rechenschaltung auch der ausgegebenen Liste entspricht, kann wesentlich zur Vereinfachung beitragen.

Als weitere Dienstleistung hat das Programmiersystem noch die Skalierung des analogen Teilmodelles oder zumindest eine Hilfestellung dabei zu erbringen. Da nur bei speziellen Problemen automatische Skalierverfahren bekannt sind, bieten sich hier zwei Verfahren [3] an.

Da das gesamte Problem bei einer entsprechenden Anweisung auf dem Digitalrechner bearbeitet wird, können die für die Amplitudennormierung benötigten Maximalwerte in einem Testlauf gewonnen werden. Aus ökonomischen Gründen erscheint jedoch ein anderes Verfahren sinnvoller, bei dem durch Übersteuerungsanzeigen nur auf jene Grössen hingewiesen wird, die reskaliert werden müssen. Diese nachträgliche Korrektur- die Reskalierung - baut auf einer Grobskalierung auf. Dadurch wird erreicht, dass nicht mehrere Vorbereitungsläufe nötig sind und die Korrekturschritte nur so gross werden, dass die Verstärker immer gut ausgesteuert sind. Tritt dabei auf der digitalen Seite ein Wert X auf, der nicht in dem auf dem Analogrechner darstellbaren Wertebereich $-1 < X < +1$ liegt, dann erscheint keine Fehlermeldung. Allerdings muss sichergestellt sein, dass die an den Analogrechner zu übermittelnden Werte wieder in diesem Bereich liegen.

4. Realisierung des Systems

An zwei ursprünglich eigenständigen Entwicklungen, ACTRAN [4] und HIFIPS [3] wird gegenwärtig an der gemeinsamen Implementierung gearbeitet. Bei diesem neuen System unter dem Namen HCS-1 (Hybrid Compiler System 1) sind verschiedene Versionen vorgesehen. Bild 4.1.

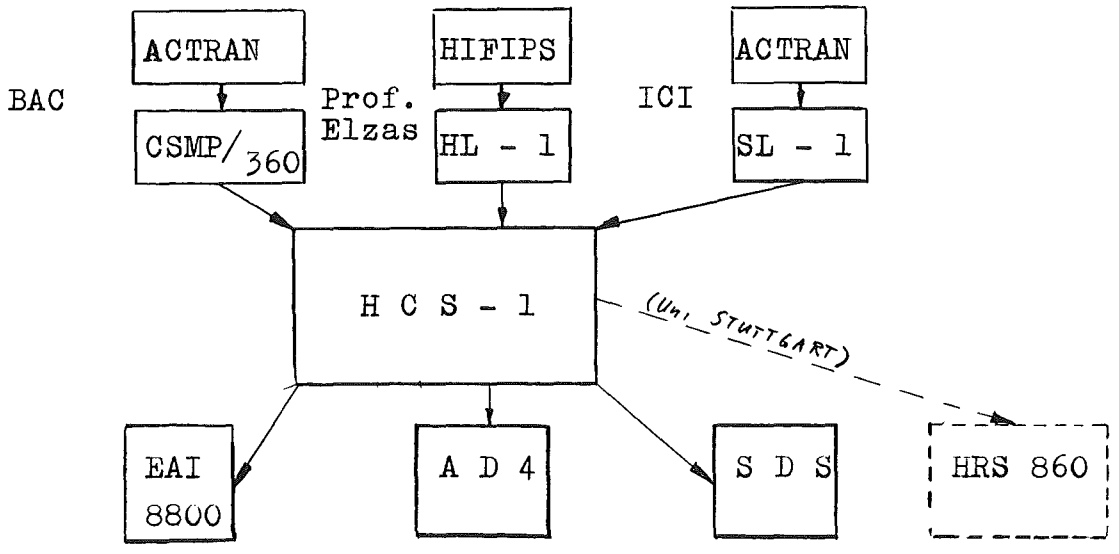


Bild 4.1: Versionen von HCS-1

Es existieren 3 verschiedene Quellensprachen: CSMP/360 und SL-1 vom System ACTRAN her und HL-1 von HIFIPS. Je nach gewünschter Quellensprache unterscheidet sich das Compiler-System jedoch u.U. ganz wesentlich. Auch der verwendete Analogrechner ist wegen dem entsprechenden analogen Maschinencode für das Compiler-System von Bedeutung.

Alle drei wählbaren Quellensprachen haben gemeinsam, dass sie die Problemaufteilung gänzlich dem Benutzer überlassen, der dann über entsprechende Kenntnisse verfügen sollte. Streng genommen erfüllen diese Quellensprachen damit nicht die Anforderungen einer hybriden Quellensprache, sondern können als unterteilte Quellensprachen angesehen werden. Automatic-Patching-Systeme stehen noch nicht zur Verfügung, jedoch sind die entsprechenden Schnittstellen für deren Einsatz vorgesehen. Testhilfen für die Skalierung müssen über einen rein digitalen Simulationslauf gewonnen werden, da die Maximalwerte der in dem analogen Teil realisierten Variablen anzugeben sind. Die Entwicklung automatischer Skalierverfahren ist noch nicht abgeschlossen. Von den ursprünglichen Konzeptionen von ACTRAN und HIFIPS sind viele Eigenarten noch erhalten. So wendet sich ACTRAN vorzugsweise an den alten Benutzerkreis von Hybrid-Systemen und sieht neben einer

Vielfalt von Systemoperatoren noch die Möglichkeit vor, direkt in die Quellensprache noch Teile von analogem Maschinen-Code mit einzufügen (EDITOR). In der digitalen Sektion können alle in FORTRAN aufrufbaren Funktionen verwendet werden, während man sich in der analogen Sektion auf die vorhandenen Systemoperatoren zu beschränken hat. Nur bei Verwendung des relativ komplizierten Maschinen-Codesist es möglich, sich innerhalb der analogen Sektion von dem starren Konzept der Systemoperatoren zu lösen. Bei HL-1 besteht innerhalb der analogen Sektion die Möglichkeit, Makros mit Hilfe von analogen Maschinen-Code-Symbolen selbst zu definieren. Mit diesen Symbolen werden dabei die Funktionen dargestellt, die auf dem Analogrechner AD 4 realisiert sind. Beide Sprachen sind für ein komfortables Programmiersystem nicht in allen Details geeignet. Jedoch erscheint die streng logisch gegliederte Sprache HL-1 und der entsprechende Teil von HCS-1 besser als Basis für ein Programmiersystem geeignet, das die in Punkt 1 erläuterten Vorstellungen verwirklicht.

Die Entwicklung des in [5] beschriebenen Patching-Systems PATOMAT ist soweit gediehen, dass es bei vielen Problemen die Verschaltung der Rechenschaltung selbständig übernehmen kann. Dabei ist daran gedacht, das beschriebene Compiler-System HCS-1 bzw. Teile davon, so umzubauen und zu erweitern, dass es auf dem Hybrid-System HRS 860 von Telefunken implementiert werden kann. Mit Applied Dynamics Europe, das die kommerziellen Rechte für HCS-1 vertritt, werden Verhandlungen über einen entsprechenden Vertrag geführt.

In einer nächsten Ausbaustufe soll dieses System durch eine automatische Systemaufteilung ergänzt werden.

Literaturhinweise:

[1] The SCi Continons System Simulation Language (CSSL)
Simulation, Dec. 1967

[2] Franklin, Strauß: A Hybrid Computer Programming System
FJCC 1969

[3] Elzas: Present State of HIFIPS
AICA/IFIP-Konferenz, München 1970

[4] Rook: ACTRAN: An Analogue/Hybrid Compiler
AICA/IFIP-Konferenz, München 1970

[5] Bannas, Hecht, Portugall, Reuss, Rzehak:
Erleichterung des Einsatzes hybrider Rechnersysteme durch Programmierung auf problemorientierter Ebene.
Technical Report des "Instituts für Informatik", Universität Stuttgart (in Vorbereitung)

S C A L E - eine digitale Implementierungshilfe für analoge und hybride Simulationen.

D. Heppner

1. Einführung:

Den Vorteilen der analogen Simulation dynamischer Systeme und Lösung gewöhnlicher oder partieller Differentialgleichungen, wie

direkter Zugriff zu allen Problemgrößen,
vom Problemumfang nahezu unabhängige Rechenzeit,
hohe Rechengeschwindigkeit,
funktionale Darstellung der Ergebnisse,
keine numerischen Schwierigkeiten bei der Integration,

steht der zeitlich große Aufwand der Problemaufbereitung und Programmierung gegenüber. Die Kopplung von Analog- und Digitalrechner zu einem Hybridsystem erschließt zwar die rechentechnischen Möglichkeiten beider Rechnertypen und erhöht den Bedienungskomfort wesentlich, aber die zusätzlich auftretenden Schwierigkeiten können oft nur von speziell ausgebildeten Benutzern überwunden werden. Die Ursachen sind zum Teil in der unzureichenden und für den Benutzer oft unüberschaubaren hardware der Analogrechner und der Koppelwerke zum Digitalrechner zu suchen. Eine von der Kenntnis der hardware unabhängige software für Analogrechner und Hybridsysteme ist kaum zu finden. Hierin scheint der Hauptgrund für die geringe Verbreitung des Analogrechners zu liegen. Es ist interessant festzustellen, daß durch eine relativ einfache Aenderung der hardware (z.B. in ihrer Funktion festverschaltete Rechenwerke, digital gesteuerte Maschinenkoeffizienten statt Potentiometer und Eingangsfaktoren) und ein darauf basierendes software-Paket die Benutzung von Analogrechnern erheblich vereinfacht werden kann.

Die für den Analogrechner notwendige Skalierung der Problemvariablen und Berechnung der Einstellwerte ist eine mühsame Arbeit und Ursache vieler Programmierfehler. Diese Aufgabe kann der Digitalrechner wesentlich schneller und sicherer lösen. Weitere Fehlerquellen stellen Steckfehler durch den Programmierer und hardware-Fehler durch defekte Rechenelemente dar. Ein digitales Programm zur Aufbereitung eines Problems für Analogrechner oder Hybridsysteme sollte folgende Eigenschaften besitzen:

1. Die Eingabe der Problembeschreibung soll so einfach und bequem wie möglich sein und nicht die Kenntnis einer bestimmten Programmiersprache voraussetzen.
2. Eine digitale Simulation des Problems zur Berechnung der Skalierung.
3. Ausgabe einer Steckliste für den analogen und logischen Teil des Programms.
4. Ausgabe der skalierten und unskalierten Problemmaxima und -koeffizienten.
5. Ausgabe eines statischen Tests für alle analogen und logischen Rechenelemente zur Ueberprüfung auf Steck- oder hardware-Fehler.
6. Ausgabe von Ergebnissen als Diagramme oder Tabellen.
7. Einfache Einstellanweisungen für Funktionsgeber.

Für Hybridsysteme gelten weitere Punkte:

8. Uebergabe der unter 3. bis 5. genannten Listen an das Betriebssystem (hybrider Interpreter) zur Einstellung der Maschinenkoeffizienten und Durchführung des statischen Tests.
9. Falls ein automatisches Programmierbrett vorhanden, Bereitstellung der Schaltmatrix und der zugehörigen Korrespondenzlisten zwischen Problem- und Maschinenvariablen.

In neuerer Zeit wurden Programme wie APSE [1], ACTRAN [2] und HIFIPS [3] entwickelt, die weitgehend alle oben genannten Punkte beinhalten. APSE (Automatic Programming and Skaling of Equations) gestattet eine analytische Formulierung des Problems und zerlegt die Problemgleichungen in einzelne Rechenoperationen, die der analogen Programmierung entsprechen. ACTRAN (Analog Compiler and TRANslator) und HIFIPS sind Erweiterungen der Simulationssprache CSSL und setzen die Kenntnis dieser Sprache voraus. Sie bieten aber den Vorteil, daß der Benutzer ohne Programmänderung wahlweise digitale oder hybride Simulation durchführen kann. In [4] ist eine Uebersicht hybrider software zu finden.

Die folgenden Gedanken führten zur Entwicklung des Programms SCALE:

a) Es fällt den Benutzern von Analogrechnern oder Hybridsystemen relativ leicht, die gegebenen Problemgleichungen in eine Strukturskizze bzw. Blockstruktur zu überführen. Aus der Vielfalt möglicher Strukturen kann sich der Benutzer die für ihn optimale auswählen. Die Uebersetzung der analytischen Formulierung durch den Digitalrechner erzeugt stets eine fest vorgegebene Struktur.

b) Eine blockorientierte Problemdarstellung gestattet die Definition nahezu beliebiger analoger und logischer Rechenelemente und es muß dem Benutzer in einfacher Weise möglich sein, das Programm mit von ihm beschriebenen Rechenelementen zu erweitern. Häufig vorkommende Strukturen wie Uebertragungsfunktionen 1. und 2. Ordnung sollen als Unterprogramme aufrufbar sein. So kann z.B. ein Regelungstechniker sein Problem in der ihm gewohnten Form der Blockstruktur beschreiben und SCALE übersetzt sein Problem in eine analoge Strukturskizze mit

optimalen Einstellwerten.

c) Durch entsprechenden Aufbau und sinnvolle Beschränkung kann der Aktivspeicherbedarf des Programms so klein gehalten werden, daß die Rechnung auch auf dem zumeist nur aus einem Prozeßrechner bestehenden Digitalteil des Hybridsystems durchgeführt werden kann. Aus diesem Grund enthält SCALE keine Sortierroutine, die die Liste der Strukturblöcke in eine für den Digitalrechner optimale Reihenfolge bringt und algebraische Schleifen auflöst. Es ist beabsichtigt, nach der Fertigstellung von SCALE ein spezielles Sortierprogramm zu entwickeln. Da Einstellanweisungen für Funktionsgeber unabhängig von der eigentlichen Problemaufbereitung sind, können sie in einem separaten Programm erstellt werden.

2. Allgemeine Uebersicht:

Das Programm SCALE besteht aus folgenden Teilen:

a) Ueberprüfung der eingegebenen Programmstruktur auf Syntaxfehler
b) Integration der Problemgleichungen
c) Berechnung der Skalierung und Maschinenkoeffizienten
d) Ausgabe der Steckliste mit symbolischen Adressen
e) Erstellen des statischen Tests
f) Ausgabe von Lösungsfunktionen.

In den folgenden Abschnitten sollen Eingabe, Ablauf der Rechnung und Ausgabe an Hand eines Beispiels beschrieben werden.

Das Beispiel wurde aus dem EAI-Report Nr. 472

Automatic Programming and Scaling of Equations

(APSE) entnommen, um einen direkten Vergleich zwischen APSE und SCALE zu ermöglichen. Es beschreibt die Radaufhängung eines Kraftfahrzeuges bei Kurvenfahrt (Abb. 1).

2.1 Blockstruktur:

Die analoge Strukturskizze (Abb. 2) enthält alle unskalierten Problemvariablen und -koeffizienten und berücksichtigt die Vorzeichenumkehr der analogen Rechenelemente. Die Blockstruktur für SCALE (Abb. 3) unterscheidet sich nur unwesentlich von der analogen Strukturskizze. Zunächst ist zwischen analogen und logischen Rechenelementen zu unterscheiden. Die Einteilung erfolgt nach der Art der Ausgangsvariablen. Die analogen Variablen werden fortlaufend, mit 1 bei den Integrierern beginnend, indiziert. Es müssen zunächst alle Variablen aus den Integrierern aufgezählt werden. Dann folgen die restlichen analogen Variablen, wobei sowohl den Konstanten +1 und -1 als auch den Schaltern Indizes zugeordnet werden. Auf gleiche Weise werden die logischen Variablen numeriert. Hier ist zunächst bei Variablen aus speichernden Elementen wie Flip-Flops zu beginnen. Die logischen Elemente von SCALE enthalten nicht, wie beim Analogrechner üblich, die komplementären Ausgänge und es muß bei Bedarf ein Negierer (NEGA) eingefügt werden. Die Indizierung der Problemkoeffizienten erfolgt in beliebiger Folge fortlaufend ab 1. Da die symbolische Adresse eines Rechenelementes identisch mit dem Index der Ausgangsvariablen ist, erübrigt sich in den meisten Fällen das zusätzliche Zeichnen der Blockstruktur für SCALE und es genügt, die symbolischen Adressen der Elemente und Koeffizienten in die analoge Strukturskizze einzutragen.

2.2 Eingabe der Programmstruktur:

Die Eingabe des analogen und logischen Programms erfolgt durch die Beschreibung der Blockstruktur (Abb. 4a). Jeder Block entspricht einem logischen oder analogen Rechenelement (= Rechenoperation) bzw. einem beschreibenden Unterprogramm. Es ist dem Benutzer von SCALE möglich, seinem Problem angepaßte Strukturblöcke selbst zu definieren.

Die Strukturblöcke werden grundsätzlich nach integrierenden und algebraischen bzw. logischen Funktionen unterschieden. Zu den integrierenden Blöcken zählen Integrierer und Uebertragungsfunktionen, die im Zeitbereich durch Differentialgleichungen beschrieben werden. Sie bilden in beliebiger Reihenfolge die Prozedur INTG (INTegrierer Gleichungen). In Sonderfällen müssen spezielle Funktionsgeber eingefügt werden.

Die algebraischen und logischen Strukturblöcke sind in der Prozedur ALOG (Analoge und LOgische Gleichungen) so aufzuführen, daß ihre Eingangsvariablen durch vorhergehende Strukturblöcke definiert sind. Die Ausgangsvariablen von Integrierenden Strukturblöcken gelten als definiert, da ihnen Anfangswerte zugewiesen werden.

Die auf dem Analogrechner programmierbare implizite Rechnung wie algebraische Schleifen, Umkehrfunktionen, etc. ist mit SCALE auf Grund der Arbeitsweise des Digitalrechners nicht auf gleiche Weise darstellbar. Durch Umformung der Programmstruktur und Einfügen spezieller Programmteile lassen sich vielfach algebraische Schleifen auflösen. Mit Hilfe der Prozedur NOCHECK (Abb. 4b) werden die zu algebraischen Schleifen gehörenden Variablen bei der Ueberprüfung auf Syntaxfehler übersprungen. Für Umkehrfunkti-

onen sind spezielle Rechenwerke zum Teil definiert oder müssen zusätzlich eingefügt werden.

2.3 Eingabe der Daten:

Die für SCALE einzugebenden Daten werden nach Betriebs- und Problemdaten unterschieden. Die Betriebsdaten geben Auskunft über Umfang und Ablauf der Rechnung. Zu den Problemdaten gehören z.B. Anfangswerte der integrierenden Blöcke, Problemkoeffizienten und Funktionsgebertabellen. In der Prozedur KOEF (Abb. 4b) werden die Problemparameter eingelesen und die Koeffizienten der Blockstruktur errechnet.

Da die Eingabe der Problemstruktur und der Daten getrennt erfolgt, können mehrere Läufe ohne erneute Kompilation durchgeführt werden. Dabei speichert SCALE die absoluten Maxima der Problemvariablen aus allen Rechnungen, so daß die endgültigen Einstellwerte für den Analogrechner die Variation aller Problemdaten berücksichtigen.

2.4 Integration der Problemgleichung:

Zur Bestimmung der absoluten Maxima der Problemvariablen und Ausgabe einer beschränkten Anzahl von Lösungskurven werden die Problemgleichungen unskaliert integriert. SCALE überführt dabei die eingegebene Problemstruktur in ein System von Differentialgleichungen 1. Ordnung, um dann eine quasi-stetige Integration durchzuführen. Die Integration erfolgt unter automatischer Schrittweitenkontrolle, deren relativer Abbruchfehler durch den Benutzer bestimmt wird. Die Abtastung der Maximalwerte und Abfrage der Zustände der logischen Variablen erfolgen nach jedem Integrationsschritt. Sind die Maximalwerte aller Problemvariablen bekannt, so kann die Integration durch entsprechende Betriebs-

daten übersprungen werden.

Vielfach soll die Skalierung der Problemvariablen auf vom Benutzer bestimmte Funktionsmaßstäbe oder Maximalwerte erfolgen. Diese Funktionsmaßstäbe können mit den Daten eingegeben werden. SCALE vergleicht nach der Integration die errechneten mit den eingelesenen Funktionsmaßstäben, um zu überprüfen, ob die eingegebenen Funktionsmaßstäbe die Bedingung

$$\sigma_i \cdot |y_i|_{max} \leq 1$$

erfüllen. Falls nicht, wird die Skalierung mit den errechneten Werten durchgeführt (vergl. Abschnitt 2.5.2).

2.5 Ausgaberoutinen:

Die folgenden Abschnitte beschreiben die Ausgaberoutinen von SCALE.

2.5.1 Eingegebene Daten:

Zur Kontrolle werden eingegebene Betriebs- und Problemdaten wieder ausgedruckt.

2.5.2 Kenngrößen der Problemvariablen:

In diesem Teil (Abb. 5) werden tabellarisch folgende Werte ausgegeben:

Symbolische Variablen, Art der Rechenelemente, absolute Problemmaxima, minimale Funktionsmaßstäbe*), errechnete Funktionsmaßstäbe, analoge Maxima**).

*)In dieses Feld wurden zunächst, falls vorhanden, die vorgegebenen Funktionsmaßstäbe eingelesen und nach Abschnitt 2.4 überschrieben.

**)Die Skalierung bezieht sich auf die minimalen Funktionsmaßstäbe*).

Wird ein Programm mit mehreren Datensätzen durchlaufen, erfolgt der Ausdruck der Kenngrößen nach jeder Rechnung.

2.5.3 Steckliste:

Die Steckliste, getrennt nach analogen und logischen Variablen, enthält alle notwendigen Steckverbindungen für den Analogrechner (Abb. 6). Die Rechenelemente erhalten symbolische Adressen, die gleich dem Index der Ausgangsvariablen sind. Unterprogramme werden in einzelne Rechenelemente aufgelöst. Außerdem erhalten die Rechenelemente symbolische Namen, wie

INTG	Integrierer
SUMM	Summierer
MULT	Multiplizierer
+REF	pos. Referenz (+1)
ANDG	UND-Gatter.

Die Bezeichnung der Koeffizientenpotentiometer ergibt sich aus der Verwendung:

ICPOT n	Anfangswertpotentiometer des Integrierers n. (n ist die symbolische Adresse des Integrierers)
INPOT n	Eingangspotentiometer des Integrierers n ohne Problemkoeffizienten. (n ist die symbolische Adresse des Integrierers)
KOPOT i	Koeffizientenpotentiometer des eingegebenen Problemkoeffizienten i.

Unterprogramme werden durch die Ueberschrift:

SUBR n name n symbolische Adresse
name Bezeichnung des Strukturblockes

angezeigt. Dann folgen die Steckverbindungen.

2.5.4 Einstellwerte:

Die Listen (Abb. 7) für die Einstellwerte der Koeffizientenpotentiometer sind nach Integrierpotentiometern (INPOT, ICPOT) und Problemkoeffizienten (KOPOT) aufgeteilt. Die Tabelle für die Integrierpotentiometer enthält folgende Spalten:

Symbolische Adresse, Wert des INPOT, Spalte zur Eintragung der aktuellen Adresse des INPOT, Eingangsfaktor des Integrierers, Wert des ICPOT, Spalte zur Eintragung der aktuellen Adresse des ICPOT.

Für die Problemkoeffizienten gelten die Spalten:

Symbolische Adresse, Wert des KOPOT, Spalte zur Eintragung der aktuellen Adresse des KOPOT, Eingangsfaktor, Wert des Problemkoeffizienten, Skalierungsfaktor.

Zur Variation der Problemkoeffizienten benutzt man die Gleichung:

(Wert des KOPOT)×(Eingangsfaktor) =
(Wert des Problemkoeffizienten)×(Skalierungsf.)

Die Berechnung der Einstellwerte gilt nur für die zuvor erstellte Steckliste.

2.5.5 Statischer Test:

Mit Hilfe des statischen Tests wird das auf dem Analogrechner programmierte Problem auf Steckfehler und defekte Rechenelemente überprüft. Integrierern, die keine oder zu kleine Anfangswerte haben, werden für den statischen Test "günstige" Werte zugewiesen. Aus diesem Grund sind in einer Liste (Abb. 8) die Werte für die Anfangswertpotentiometer (ICPOT) erneut mit Vorzeichen aufgeführt.

Die von SCALE zugewiesenen Anfangswerte sind durch * gekennzeichnet.

Die folgende Tabelle (Abb. 8) enthält vorzeichenrichtig die Ausgangswerte aller Rechenelemente und für die Integrierer die Summe der Eingangsvariablen ohne Vorzeichenumkehr. Analoge und logische Variable sind getrennt aufgeführt.

2.5.6 Ausgabe von Lösungsfunktionen:

Zum Vergleich der digitalen und analogen Simulation (dynamischer Test) können vom Benutzer beliebige Variable ausgewählt werden, die tabellarisch oder, falls ein Zeilendrucker mit 90 Spalten oder mehr vorhanden, in je einem Diagramm (Abb. 9) ausgegeben werden. Für jede Funktion sind 50 äquidistante Wertepaare festgelegt. Das Diagramm hat annähernd das Format von 20 cm × 20 cm, entsprechend den Sichtgeräten und X-Y-Schreibern der Analogrechner.

3. Schlußbemerkungen:

Der modulare Aufbau von SCALE gestattet nahezu beliebige Erweiterungen. Zur Vorbereitung hybrider Simulationen können der Datentransfer über das Koppelwerk und die Berechnungen des Digitalrechners eingefügt werden.

SCALE enthält keine Sortierroutine zur Auflösung algebraischer Schleifen, damit der Speicherbedarf so gering wie möglich gehalten wird. Es ist beabsichtigt, ein separates Sortierprogramm zu entwickeln. Ein Formelübersetzer, ähnlich APSE, erscheint problematisch, da die Auswahl einer für den Benutzer optimalen Blockstruktur schwer zu beschreiben ist.

Das hier beschriebene Beispiel benötigt ca. 15 K Worte Kernspeicher. Durch Einschränkungen in den Ausgaberoutinen kann der Kernspeicherbedarf erheblich reduziert werden. Die Rechenzeit betrug ca. 150 sec auf einer ICL 1907.

Literatur:

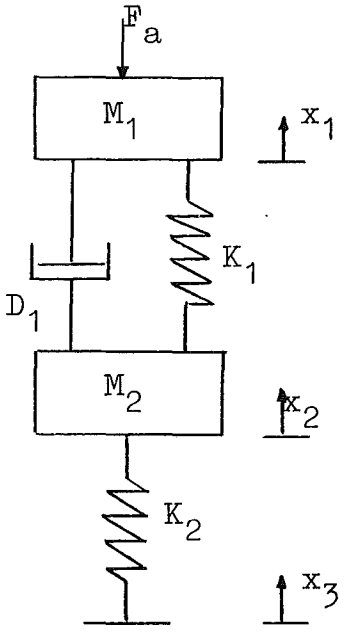
[1] EAI : Automatic Programming and Scaling of Equations. Report Nr. 472

[2] Rook,P.E. : Automated Compilation and Set-up of a Simulation on a Hybrid Computer using ACTRAN. Sammelband des Informatik-Seminars HYBRIDE SYSTEME, Universität Stuttgart, Februar 1972

[3] Elzas,M.S. : Present state of HIFIPS, a hybrid interactive formula interpreting programming system. Proc. of AICA-Conference, München 1970

[4] Franklin,M.A., Strauss,J.C. : Automated programming of analog hybrid computers - a review. Simulation, Januar 1972

Ersatzsystem:



x_1 : Auslenkung der Karosserie
x_2 : Auslenkung des Rades
x_3 : Straßenprofil
F_a : Aequivalente Kraft entsprechend der zentrifugalen Beschleunigung
M_1 : 1/4 Masse des Fahrzeugs ohne Achsen und Räder
M2 : Masse eines Rades und 1/2 Achse
K_1 : Federkonst. der Federung
K_2 : Federkonst. des Reifens
D_1 : Stoßdämpferkonstante

Problemgleichungen:

1.Gleichung:

$$M_1 \cdot \ddot{x}_1 + D_1 \cdot (\dot{x}_1 - \dot{x}_2) + K_1 \cdot (x_1 - x_2) = - F_a$$

$$\dot{x}_1(0) = 0 \; ; \; x_1(0) = 0$$

2.Gleichung:

$$M_2 \cdot \ddot{x}_2 + D_1 \cdot (\dot{x}_2 - \dot{x}_1) + K_1 \cdot (x_2 - x_1) + K_2 \cdot x_2 = K_2 \cdot x_3$$

$$\dot{x}_2(0) = 0 \; ; \; x_2(0) = 0$$

3.Gleichung:

$$x_3 = \begin{cases} x_{3H} & \text{für } 0 < t \leq 0.05 \\ 0 & \text{für } t > 0.05 \end{cases}$$

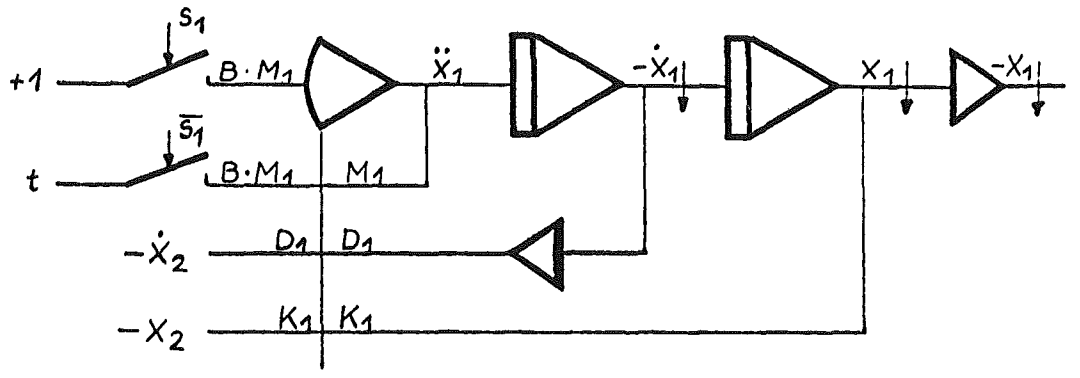
$$F_a = \begin{cases} B \cdot M_1 \cdot t & \text{für } 0 < t \leq 1.0 \\ B \cdot M_1 & \text{für } t > 1.0 \end{cases}$$

4.Gleichung:

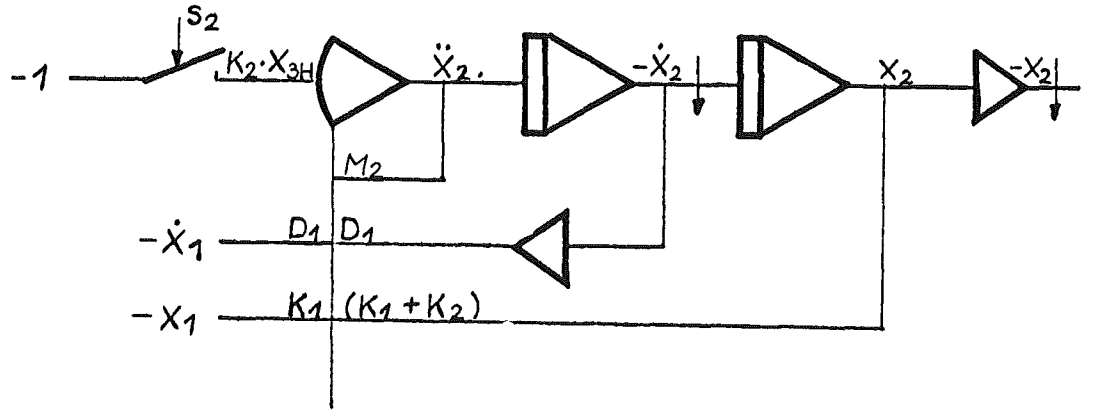
$$I_R = \int_0^t x_1^2 \cdot t \cdot dt \stackrel{!}{=} \text{Min} \quad \text{(Beurteilungsfunktion)}$$

Abb. 1: Problembeispiel

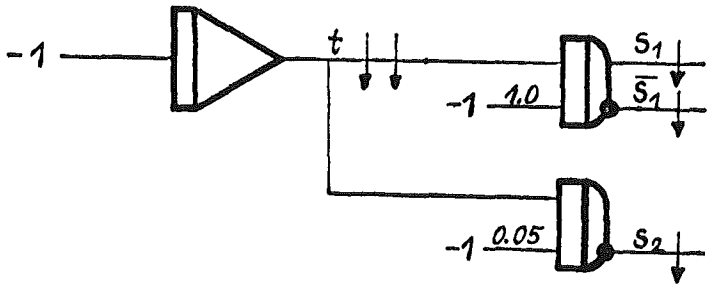
1. Gleichung:



2. Gleichung:



3.Gleichung:



4. Gleichung:

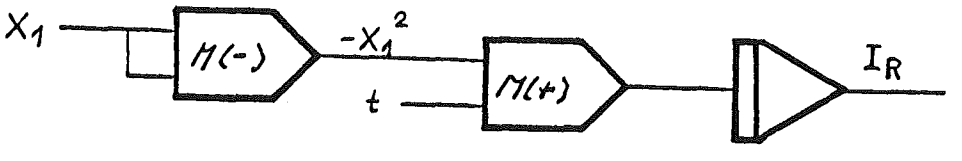
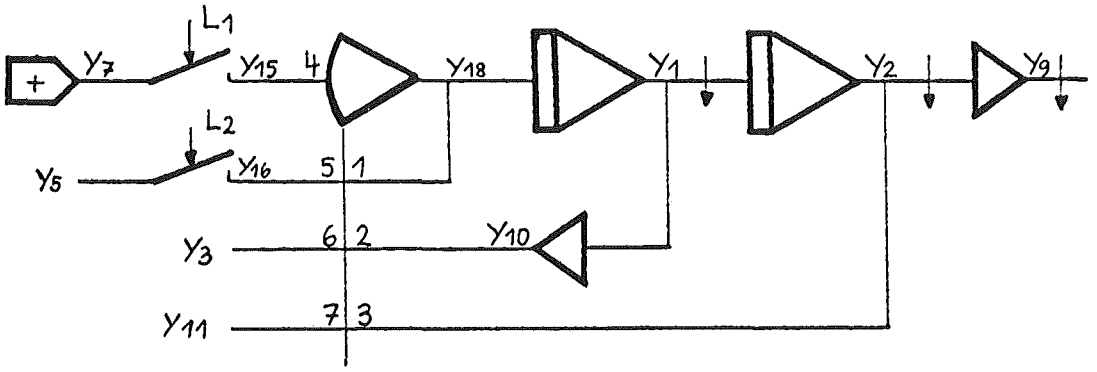
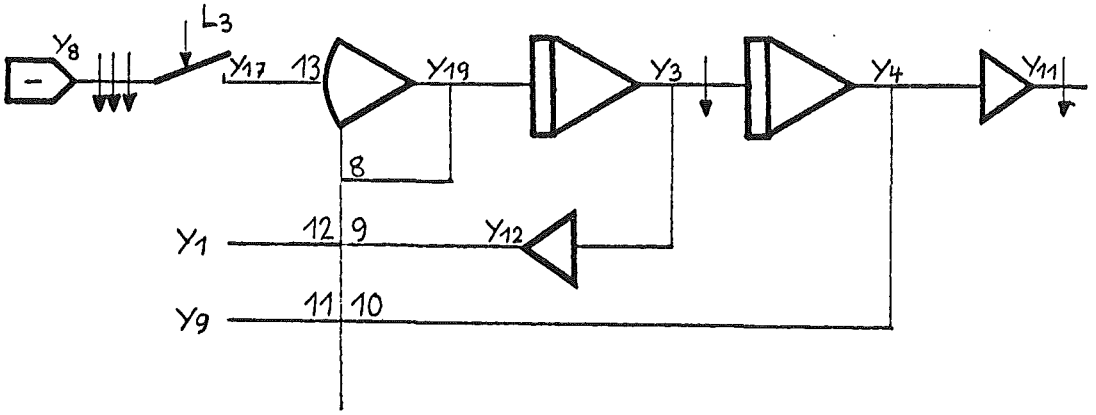


Abb. 2: Analoge Strukturskizze

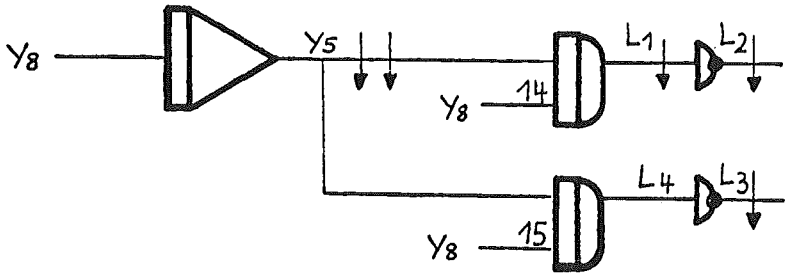
1. Gleichung:



2. Gleichung:



3.Gleichung:



4. Gleichung:

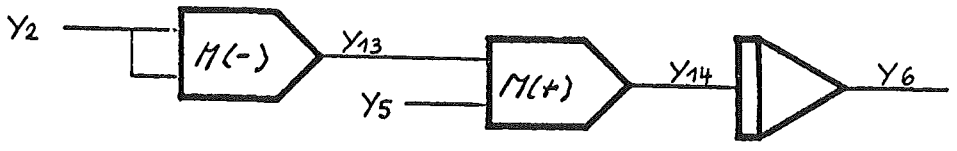


Abb. 3: Blockstruktur für SCALE

```
'comment' BEISPIEL: RADAUFHAENGUNG NACH APSE;
'comment' TRIVIALE PROGRAMMIERUNG;

'procedure' INTG(Y,DY);
'array'Y,DY;
   'begin'
   'comment' INTEGRIERER BLOECKE;
   INT (Y,DY) VAR:(18,1);
   INT (Y,DY) VAR:(1,2);
   INT (Y,DY) VAR:(19,3);
   INT (Y,DY) VAR:(3,4);
   INT (Y,DY) VAR:(8,5);
   INT (Y,DY) VAR:(14,6);
   'end';

'procedure' ALOG(Y);
'array'Y;
   'begin'
   'comment' ANALOGE U. LOGISCHE BLOECKE;
   CON (Y) VAR:(7)        VZ:(+1.0);
   CON (Y) VAR:(8)        VZ:(-1.0);
   KOMP(Y) VAR:(5,8)   KOEF:(14)   LOG:(1);
   KOMP(Y) VAR:(5,8)   KOEF:(15)   LOG:(4);
   NEGA(Y) VAR:(1,2);
   NEGA(Y) VAR:(4,3);
   SWIT(Y) VAR:(7,15)  KOEF:(4)    LOG:(1);
   SWIT(Y) VAR:(5,16)  KOEF:(5)    LOG:(2);
   SWIT(Y) VAR:(8,17)  KOEF:(13)   LOG:(3);
   INV (Y) VAR:(2,9);
   INV (Y) VAR:(1,10);
   INV (Y) VAR:(4,11);
   INV (Y) VAR:(3,12);
   SUM7(Y) VAR:(15,16,3,11,10,2,18) KOEF:(4,5,6,7,2,3,1);
   SUM6(Y) VAR:(17,1,9,12,4,19)     KOEF:(13,12,11,9,10,8);
   MULT(Y) VAR:(2,2,13)   VZ:(-1.0);
   MULT(Y) VAR:(13,5,14)  VZ:(+1.0);
   'end';
```

Abb. 4a: Elementenliste

```
'procedure' KOEF;
   'begin'
   'real' M1, M2, D1, K1, K2, X3H, B;
   'comment' EINLESEN DER PARAMETER;
   M1 := readb;
   M2 := readb;
   D1 := readb;
   K1 := readb;
   K2 := readb;
   X3H:= readb;
   B  := readb;
   'comment' BERECHNUNG DER KOEFFIZIENTEN;
   KO[1]  := M1;
   KO[2]  := D1;
   KO[3]  := K1;
   KO[4]  := B×M1;
   KO[5]  := B×M1;
   KO[6]  := D1;
   KO[7]  := K1;
   KO[8]  := M2;
   KO[9]  := D1;
   KO[10] := K1+K2;
   KO[11] := K1;
   KO[12] := D1;
   KO[13] := K2×X3H;
   KO[14] := 1.0;
   KO[15] := 0.05;
   'end';

'procedure' NOCHECK;
'begin'       'end';
```

Abb. 4b: Koeffizientenliste

KENNGROESSEN DER PROBLEMVARIABLEN

VAR	ELEM	ADR	PROBL MAX	SIGMA MIN	SIGMA INT	DYN MAX
1	INTG		-.3708& 00	.2000& 01	.2000& 01	-.7415
2	INTG		.2228&-01	.2000& 02	.2000& 02	.4456
3	INTG		.2127& 01	.2000& 00	.2000& 00	.4253
4	INTG		.8053&-01	.1000& 02	.1000& 02	.8053
5	INTG		.2000& 01	.5000& 00	.5000& 00	1.0000
6	INTG		.1310&-03	.5000& 04	.5000& 04	.6549
7	+REF		.1000& 01	.1000& 01	.1000& 01	1.0000
8	-REF		-.1000& 01	.1000& 01	.1000& 01	-1.0000
9	INVT		-.2228&-01	.2000& 02	.2000& 02	-.4456
10	INVT		.3708& 00	.2000& 01	.2000& 01	.7415
11	INVT		-.8053&-01	.1000& 02	.1000& 02	-.8053
12	INVT		-.2127& 01	.2000& 00	.2000& 00	-.4253
13	MULT		-.4964&-03	.4000& 03	.4000& 03	-.1985
14	MULT		-.1635&-03	.2000& 03	.2000& 03	-.0327
15	SWIT		.1000& 01	.1000& 01	.1000& 01	1.0000
16	SWIT		.1000& 01	.5000& 00	.5000& 00	.5000
17	SWIT		-.1000& 01	.1000& 01	.1000& 01	-1.0000
18	SUMM		.8939& 01	.1000& 00	.1000& 00	.8939
19	SUMM		-.2403& 03	.2000&-02	.2000&-02	-.4806

ENDE DER KENNGROESSEN

Abb. 5: Kenngrößen

STECKLISTE FUER DEN ANALOGRECHNER

1. ELEMENTE MIT ANALOGEN EINGANGSGROESSEN

ELEMENT		EINGANGSVARIABLE VON					
INTG	1	+REF		ICPOT	1	IC / CI	.1
		SUMM	18	INPOT	1	K. 10	
INTG	2	-REF		ICPOT	2	IC / CI	1.0
		INTG	1	INPOT	2	K. 10	
INTG	3	-REF		ICPOT	3	IC / CI	.1
		SUMM	19	INPOT	3	K. 10	
INTG	4	-REF		ICPOT	4	IC / CI	.1
		INTG	3	INPOT	4	K. 10	
INTG	5	-REF		ICPOT	5	IC / CI	1.0
		-REF		INPOT	5	K. 1	
INTG	6	-REF		ICPOT	6	IC / CI	.1
		MULT	14	INPOT	6	K. 10	
KOMP	1	INTG	5	EING.	1		
		-REF	8	KOPOT	14	EING. 2	
KOMP	4	INTG	5	EING.	1		
		-REF	8	KOPOT	15	EING. 2	
SWIT	15	+REF	7	KOPOT	4	K. 1	
SWIT	16	INTG	5	KOPOT	5	K. 1	
SWIT	17	-REF	8	KOPOT	13	K. 10	
INVT	9	INTG	2	EING.			
INVT	10	INTG	1	EING.			
INVT	11	INTG	4	EING.			
INVT	12	INTG	3	EING.			
SUMM	18	SUMM	18	KOPOT	1	K. 10	
		SWIT	15	SJ			
		SWIT	16	SJ			
		INTG	3	KOPOT	6	K. 10	
		INVT	11	KOPOT	7	K. 1	
		INVT	10	KOPOT	2	K. 1	
		INTG	2	KOPOT	3	K. 1	
SUMM	19	SUMM	19	KOPOT	8	K. 10	
		SWIT	17	SJ			
		INTG	1	KOPOT	12	K. 1	
		INVT	9	KOPOT	11	K. 1	
		INVT	12	KOPOT	9	K. 10	
		INTG	4	KOPOT	10	K. 10	

Abb. 6: Steckliste

MULT	13	.	INTG	2	+X	INVT	-X
			INTG	2	+Y	INVT	-Y
MULT	14	.	MULT	13	-X	INVT	+X
			INTG	5	+Y	INVT	-Y

2. ELEMENTE MIT LOGISCHEN EINGANGSGROESSEN

ELEMENT			EINGANGSVARIABLE VON		
NEGA	2	.	KOMP	1	EING.
NEGA	3	.	KOMP	4	EING.
SWIT	15	.	KOMP	1	SB
SWIT	16	.	NEGA	2	SB
SWIT	17	.	NEGA	3	SB

ENDE DER STECKLISTE

Abb. 6: Steckliste (Forts.)

EINSTELLWERTE DER KOEFFIZIENTENPOTENTIOMETER

1. INTEGRIERER

SYMB ADR	INPOT	ADR EING	ICPOT	ADR
1	.2000	10	.0000	
2	1.0000	10	.0000	
3	1.0000	10	.0000	
4	.5000	10	.0000	
5	.5000	1	.0000	
6	.2500	10	.0000	

2. PROBLEMKOEFFIZIENTEN

SYMB ADR	KOPOT	ADR EING	PROBL KOEF	FAKTOR
1	.2500	10	.2500& 02	.1000& 00
2	.5000	1	.1000& 03	.5000&-02
3	.5000	1	.1000& 04	.5000&-03
4	.0500	1	.5000& 01	.1000&-01
5	.1000	1	.5000& 01	.2000&-01
6	.5000	10	.1000& 03	.5000&-01
7	1.0000	1	.1000& 04	.1000&-02
8	.5000	10	.2000& 01	.2500& 01
9	.2500	10	.1000& 03	.2500&-01
10	2750	10	.5500& 04	.5000&-03
11	.2500	1	.1000& 04	.2500&-03
12	.2500	1	.1000& 03	.2500&-02
13	.1875	10	.3750& 03	.5000&-02
14	.5000	1	.1000& 01	.5000& 00
15	.0250	1	.5000&-01	.5000& 00

Abb. 7: Einstellwerte

WERTE FUER DEN STATISCHEN TEST

1. ANALOGE WERTE

1.1 INTEGRIERERANFANGSWERTE

SYMB ADR	ICPOT		ADR
1	.1167	*	
2	-.1333	*	
3	-.1500	*	
4	-.1667	*	
5	-.1833	*	
6	-.2000	*	

1.2 STATISCHE WERTE UND ABLEITUNGEN

SYMB ADR	ELEM	ADR	WERT	ABLEITUNG/10	ADR
1	INTG		-.1167	.0581	
2	INTG		.1333	.1167	
3	INTG		.1500	.0042	
4	INTG		.1667	-.0750	
5	INTG		.1833	.0500	
6	INTG		.2000	.0008	
7	+REF		1.0000		
8	-REF		-1.0000		
15	SWIT		.0000		
16	SWIT		.1833		
17	SWIT		.0000		
9	INVT		-.1333		
10	INVT		.1167		
11	INVT		-.1667		

Abb. 8: Statischer Test

12	INVT	=.1500
18	SUMM	=.2907
19	SUMM	=.0042
13	MULT	=.0178
14	MULT	=.0033

2. LOGISCHE WERTE

SYMB ADR	ELEM	ADR	WERT
1	KOMP		0
4	KOMP		L
2	NEGA		L
3	NEGA		0

ENDE DER WERTE FUER DEN STATISCHEN TEST

Abb. 8: Statischer Test (Forts.)

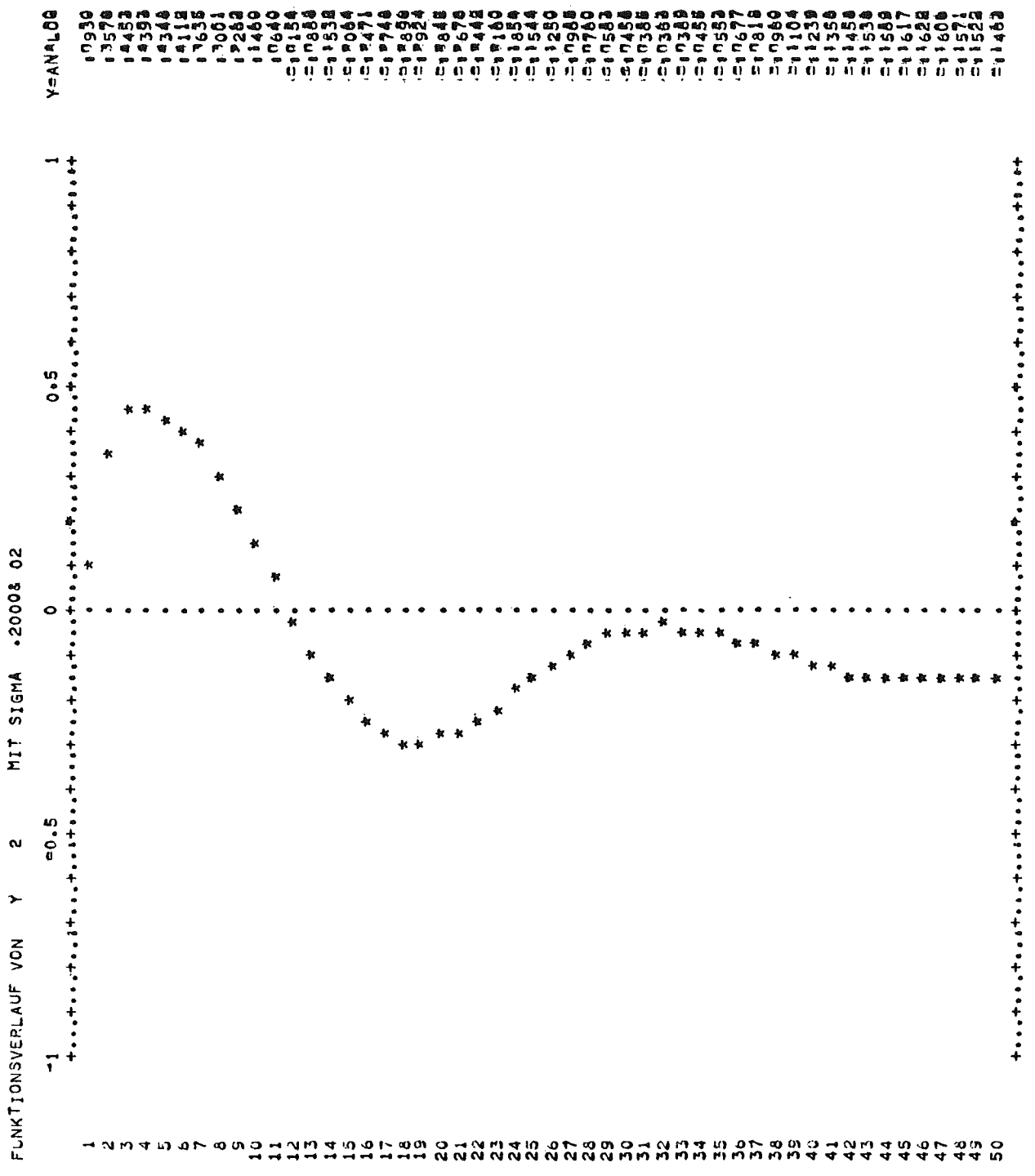


Abb. 9: Ausgabe von Funktionen

Das Simulationsprogrammiersystem MUNICH

M. FEILMEIER

1. Einleitung.

Dem Simulationsprogrammiersystem MUNICH liegt die Absicht zugrunde, ein einheitliches Programmiersystem für die hybride und (oder) digitale Simulation zeitkontinuierlicher Systeme zu schaffen.Hardwaremäßig sind hierzu eine vollautomatisierte Hybridrechenanlage und ein wenigstens mittelgroßer Digitalrechner nötig; die programmiertechnische Grundlage bildet CSSL.

2. Das autopatch.

Ein kritischer Punkt ist die Notwendigkeit eines autopatch, also einer automatischen Beschaltungsvorrichtung für den Analogteil des Hybridsystems. Wir haben deshalb als erstes mit dem Entwurf und der Konstruktion eines autopatch begonnen. Es wird in spätestens zwei Monaten verfügbar sein und für einen 80 Rechenkomponenten enthaltenden Analogrechner ca. 1200 Schalter enthalten.

3. Das Programmiersystem MUNICH.

Als höhere Simulationssprache dient eine CSSL - Version, deren genaue Festlegung nach Abstimmung mit möglichst vielen anderen, an Simulationsproblemen interessierten, erst noch erfolgen wird. (Momentan arbeiten wir mit einer Teilmenge CSSL-M von CSSL; der Übersetzer hierfür wurde manuell erstellt und umfaßt ca. 3000 Fortran-Anweisungen). Die Fortranteile eines CSSL - Programms und die DERIVATIVE SECTIONS, die digital realisiert werden sollen, werden in ein Fortran-Programm übersetzt; die analog zu realisierenden DERIVATIVE SECTIONS werden in eine einfache, blockorientierte Simulationssprache, nämlich DARE II-M, übersetzt.(Der CSSL - Übersetzer wird im übrigen eine automatische Schleifenbehandlung nach Stein-Mundstock durchführen.) Ein Benutzer kann aber auch selbst DARE II-M Programmteile erstellen. Die vom Übersetzer erzeugten DARE II-M Teile werden automatisch oder nach Benutzervorgabe mit den am Digitalteil ablaufenden Programmteilen verbunden.

Das nun in DARE II-M Form vorliegende "abstrakte" Analogschaltbild wird durch digitale Simulation (der Übersetzer hierfür - im wesentlichen ein spezieller Makroprozessor - ist vorhanden) skaliert. Das "skalierte" DARE II-M Programm dient als Eingabe für die autopatch - software.

Der autopatch - software obliegt es, das dem skalierten DARE II-M Programm entsprechende Analogschaltbild zu erstellen. Hierzu dient ein allocation - Algorithmus, der den DARE II-M-Blöcken analoge Rechenkomponenten zuordnet, und ein routing-Algorithmus, der nichtblockierende Wege durch die Mehrstufenschaltmatrizen des autopatch findet. Beide Algorithmen, d.h. also die autopatch - software, stehen unmittelbar vor ihrer Fertigstellung.

Die Parallele Logik des Analogteils tritt bei unseren Überlegungen zunächst etwas in den Hintergrund, da hier verschiedene Richtungen der technischen Weiterentwicklung denkbar sind.

4. Zusammenfassung.

Die geplanten und teilweise schon durchgeführten Arbeiten bezwecken die Erstellung eines Simulationsprogrammsystems, bei dem der Benutzer in einfachster Weise entscheiden kann, welche Programmteile analog/hybrid oder digital realisiert werden sollen. Ein Programmiersystem dieser Art wird nur möglich, wenn gleichzeitig die technischen Voraussetzungen (autopatch) geschaffen werden. Wir bauen deshalb das Gesamtprojekt von "unten" her auf.

Burkhardt
Gießler
Mathe

GMD/I 7

CSMP-I - Ein dialogfähiges Simulationsprogrammsystem zur Simulation zeitkontinuierlicher Systeme

CSMP-I - ist ein Simulationsprogrammsystem für den TR 440 und stellt eine Weiterentwicklung von CSMP/360 dar.

CSMP/360 ist ein Programmsystem für Rechenanlagen der IBM/360-Serie mit einer Mindestkernspeichergröße von 128 K Bytes. CSMP/360 gehört zur Klasse der Simulatoren für kontinuierliche Systeme; die CSMP-Simulationssprache baut auf der Verfahrenssprache FORTRAN auf und verbindet alle Vorteile einer blockorientierten Sprache mit allen Möglichkeiten von FORTRAN, insbesondere zur Darstellung algebraischer und logischer Zusammenhänge. Es können sehr komplexe Systeme (mit bis zu 300 Integratoren) untersucht werden, die auch zeitabhängige Koeffizienten und beliebige Nichtlinearitäten enthalten dürfen. Dabei kann von einem Blockschaltbild oder einem System gewöhnlicher Differentialgleichungen ausgegangen werden.

Das Simulationsprogrammsystem CSMP/360 besteht im wesentlichen aus dem Organisationsprogramm, dem Übersetzer, dem Simulator und einer Bibliothek mit vordefinierten Funktionsblöcken. Der Übersetzer liest die in CSMP abgefaßte Problembeschreibung (für jeweils ein Modell) ein, prüft sie auf Vollständigkeit und formale Richtigkeit, sortiert die Strukturaussagen - unter Auswertung spezieller Bearbeitungsaussagen zur Steuerung der Übersetzung - in eine zulässige Rechenfolge und übersetzt sie in ein FORTRAN-Unterprogramm UPDATE; prozedurale Bereiche, die ja schon in FORTRAN kodiert sind, werden dabei unverändert übernommen. Parameter und Bearbeitungsaussagen zur Steuerung der Simulationsausführung und der Ergebnisausgabe werden auf einem externen Speicher zwischengespeichert und dann vom Simulator ausgewertet.

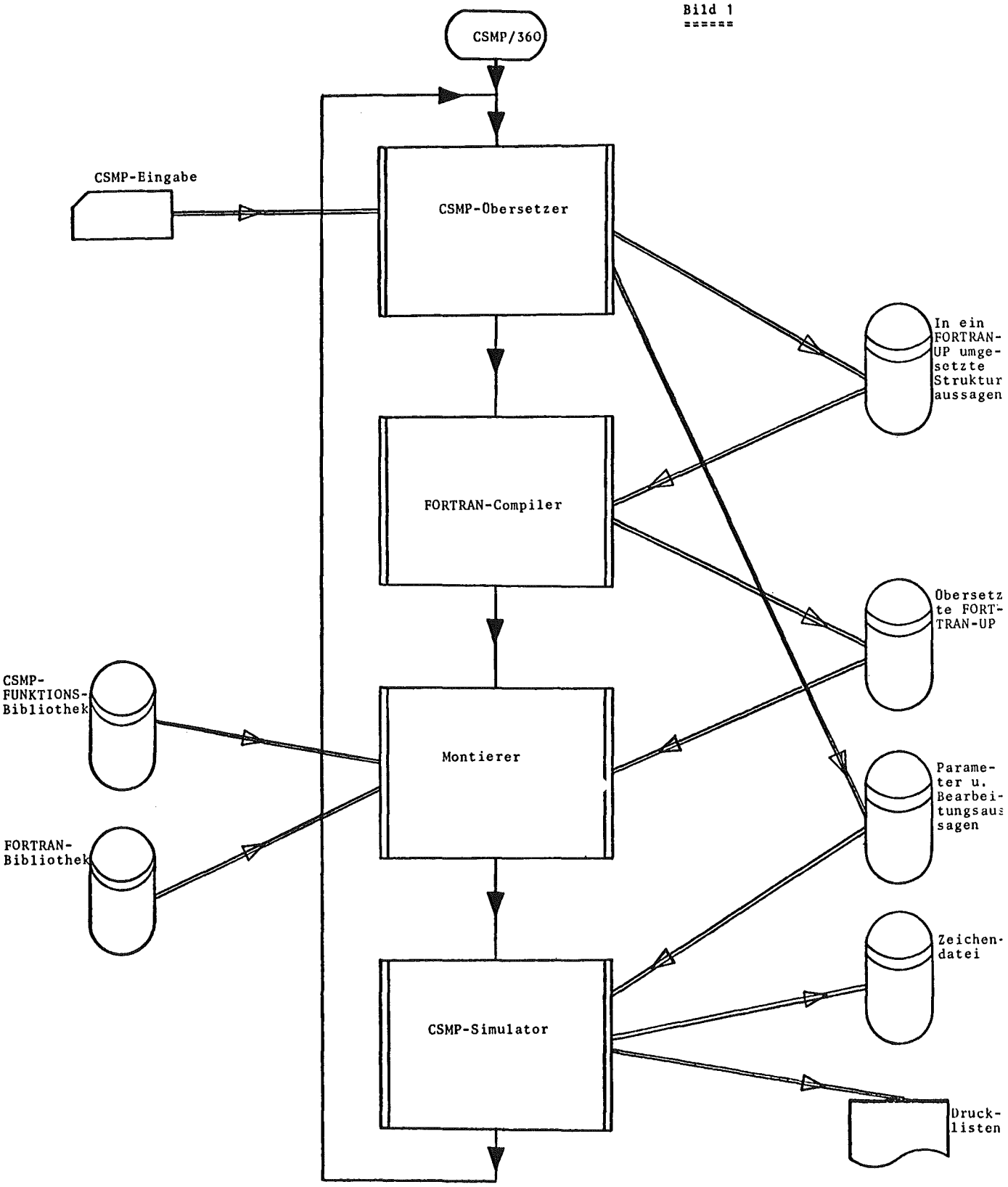
Das vom Übersetzer erzeugte FORTRAN-Unterprogramm UPDATE und evtl. weitere explizit vom Anwender hinzugefügte FORTRAN-Unterprogramme werden dann vom FORTRAN-Compiler übersetzt und in einem Verknüpfungslauf mit den benötigten CSMP-Funktionsblöcken und den Programmen mit den eigentlichen Simulationsalgorithmen, die schon in übersetzter Form vorliegen, zum Simulator verbunden.
Der Simulator übernimmt die Durchführung der eigentlichen Simulation, d. h. im wesentlichen die Zeitrasterung, die Integrationen und die Ausgabe der Ergebnisse.
Das Organisationsprogramm schließlich sorgt für den geschilderten und in Bild 1 nochmals graphisch dargestellten Ablauf der einzelnen sich einander im Kernspeicher jweils überlagernden Phasen. Es besteht aus einem kernspeicherresidenten Teil und den Hauptprogrammen des CSMP-Übersetzers und -Simulators.

Obwohl CSMP/360 ein hochentwickeltes Simulationsprogrammsystem ist und eine große Flexibilität im Hinblick auf individuelle Erweiterungen und Abwandlungen durch den Anwender besitzt, läßt es im wesentlichen drei Möglichkeiten vermissen:

1. Dialogfähigkeit
   CSMP/360 ist für reinen Batch-Betrieb konzipiert und bietet von daher keinerlei Interaktionsmöglichkeiten. Simulation aber ist eine iterative Technik, die in der Phase der Modellerprobung wie auch in der Phase der Simulationsdurchführung (etwa bei Optimierungsproblemen) ein hohes Maß an Interaktion erfordert. Die Interaktionsmöglichkeiten zeichnen ja gerade den Analogrechner vor dem Digitalrechner aus. Neben dem Batch-Betrieb, der für gewisse Fälle seine Berechtigung hat, sollte daher Dialogbetrieb von Bildschirmkonsolen (und mit Einschränkungen auch von Fernschreibern) aus möglich sein.

2. Fähigkeit zu Aufbau und Handhabung benutzereigener Makrobibliotheken und zur Handhabung benutzereigener Funktionsbibliotheken. CSMP/360 gibt zwar jedem Benutzer die Möglichkeit, eigene zusätzliche Funktionen in Form von FORTRAN-Unterprogrammen und eigene Makros zu definieren und sich dadurch einen seinem

Bild 1


CSMP/360
CSMP-Eingabe
CSMP-Übersetzer
In ein FORTRAN-UP umgesetzte Struktur aussagen
FORTRAN-Compiler
Übersetzte FORTRAN-UP
CSMP-FUNKTIONS-Bibliothek
Montierer
Parameter u. Bearbeitungsaussagen
FORTRAN-Bibliothek
Zeichendatei
CSMP-Simulator
Drucklisten

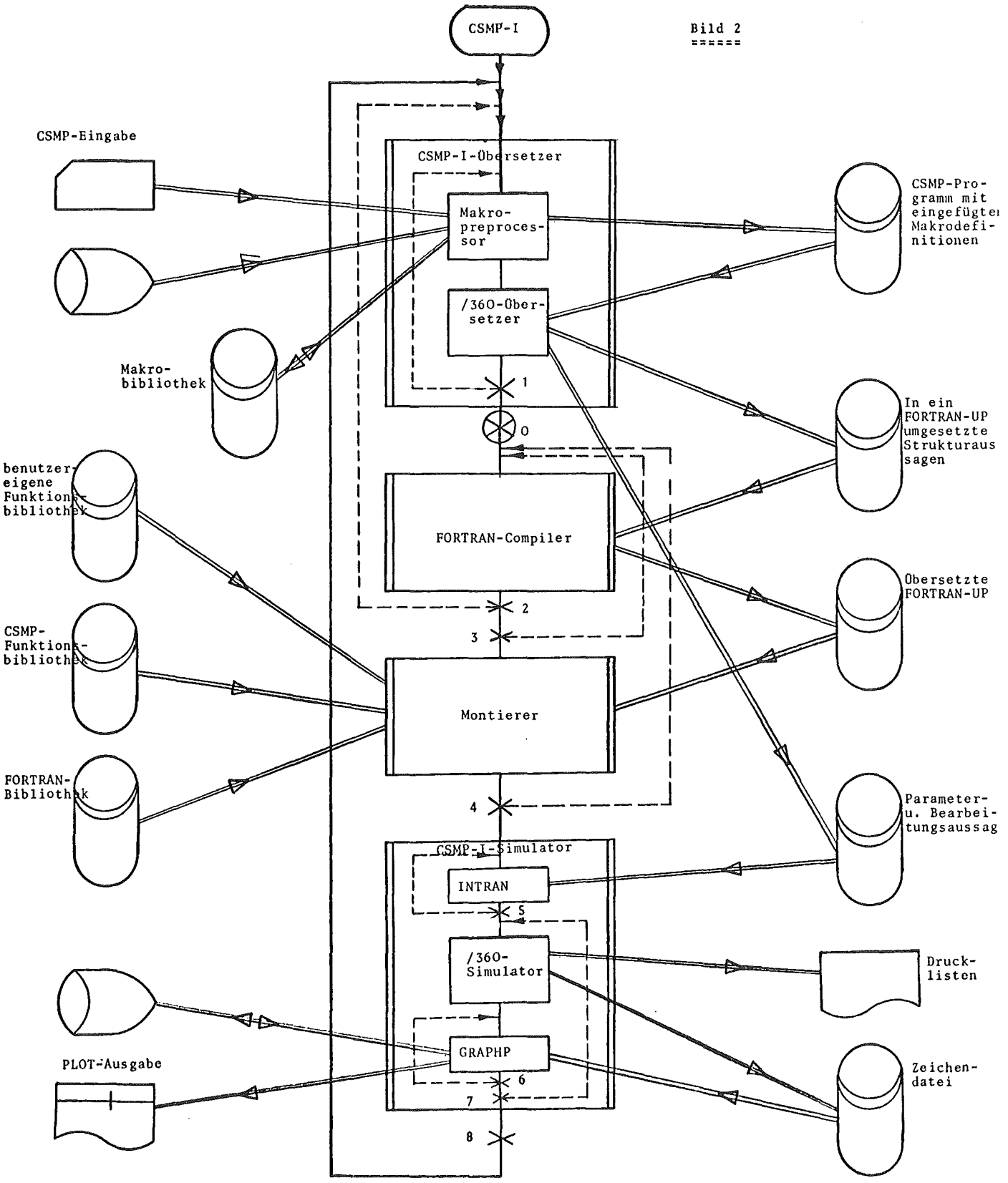
Problemkreis angepaßten Sprachvorrat zu schaffen, setzt aber voraus, daß die Definitionen immer explizit Bestandteil der Eingabe zu dem Modell sind, in dem die zugehörigen Funktions- und Makroaufrufe stehen. Wesentlich bequemer ist es, ausgetestete Funktionen und Makros in Bibliotheken abzulegen, aus denen sie dann bei Bedarf automatisch zugeladen werden können.

3. Möglichkeiten zur graphischen Ausgabe auf Plotter und Bildschirmkonsolen
   Komfortable Möglichkeiten zur graphischen Ausgabe sind die notwendige Ergänzung zur Dialogfähigkeit. Simulationsergebnisse in Kurvenform lasseen sich leichter auswerten als Ergebnislisten, insbesondere lassen sich die Ergebnisse eines Simulationslaufes oder auch mehrerer Simulationsläufe - in einem Kurvendiagramm aufgetragen - unmittelbar vergleichen.

In CSMP-I wurden die skizzierten Möglichkeiten realisiert.
Um die Dialogfähigkeit zu erreichen, wurde das Organisationsprogramm neu konzipiert (siehe Bild 2); Übersetzer, Simulator und die Bibliothek von vordefinierten Funktionsblöcken wurden von CSMP/360 bis auf geringe Modifikationen unverändert übernommen.

Im Dialogbetrieb sind an den in Bild 2 mit * gekennzeichneten Stellen Eingriffsmöglichkeiten gegeben. Dialogbetrieb wird vom Organisationsprogramm immer dann ermöglicht, wenn CSMP-I im sog. Gesprächsmodus von einer Konsole aus gestartet wurde.
*1 erlaubt es, Fehler im CSMP Programm, die vom Übersetzer festgestellt wurden, zu korrigieren und das CSMP Programm erneut übersetzen zu lassen.
*2 gibt die gleiche Möglichkeit, wenn der FORTRAN-Compiler noch Fehler im FORTRAN-Unterprogramm UPDATE entdeckt hat.
In *3 lassen sich vom Benutzer geschriebene FORTRAN-Unterprogramme korrigieren und erneut vom FORTRAN-Compiler übersetzen.
In *4 kann man auf einen vom Montierer festgestellten Fehler (fehlendes Unterprogramm) reagieren, das Unterprogramm hinzufügen, vom FORTRAN-Compiler übersetzen und erneut montieren lassen.

Bild 2


CSMP-I
CSMP-Eingabe
CSMP-I-Übersetzer
Makro-preprocessor
/360-Übersetzer
Makro-bibliothek
1
0
benutzereigene Funktionsbibliothek
FORTRAN-Compiler
2
3
CSMP-Funktionsbibliothek
Montierer
FORTRAN-Bibliothek
4
CSMP-I-Simulator
INTRAN
5
/360-Simulator
GRAPHP
6
7
8
PLOT-Ausgabe
CSMP-Programm mit eingefügten Makrodefinitionen
In ein FORTRAN-UP umgesetzte Strukturaussagen
Übersetzte FORTRAN-UP
Parameter- u. Bearbeitungsaussag
Drucklisten
Zeichendatei

In *5 kann man noch in Parameter und Bearbeitungsaussagen erkannte Fehler beheben.
In *6 kann man sich nach Belieben graphische Ausgabe erstellen lassen oder auch bei der Programmeingabe spezifizierte Ausgaben unterdrücken.
In *7 kann man spezifizierte Parameterstudien unterdrücken und andere spezifizieren.
In *8 kann man die Simulation eines weiteren schon spezifizierten Modells unterdrücken und ein anderes spezifizieren.
* O stellt einen Eingangspunkt dar und ermöglicht es, die Übersetzungsphase zu umgehen und mit einem schon in einer vorhergehenden Übersetzung erzeugten FORTRAN-Unterprogramm UPDATE zu arbeiten.

Makrodefinitionen werden auf Anforderung durch einen eigens geschriebenen Makro-Preprocessor vor der Übersetzung aus dem CSMP-Programm in die benutzereigene Makrobibliothek kopiert bzw. aus ihr entnommen. Eine Makrodefinition in einem CSMP-Programm, die in die Makrobibliothek einzutragen ist, wird dazu in Spalte 6 durch ein '+' gekennzeichnet. Eine Makrodefinition wird in ein CSMP-Programm eingefügt, wenn ein formaler Makroaufruf mit einem '+' in Spalte 1 erkannt wird.

Die Makrobibliothek ist eine RAN-Datei (wahlfreier Zugriff über Satznummer), die im allgemeinen in der LFD (langfristige Datenhaltung) oder auf Wechselplatten liegt. Jede Makrodefinition nimmt genau einen Satz ein. Zu den einzelnen Makrodefinitionen wird über eine Liste zugegriffen, in der alle Makronamen enthalten sind; der Position eines Makronamens in der Liste entspricht die Satzposition der zugehörigen Makrodefinition.
Der Makro-Preprozessor prüft nicht, ob die aktuellen Parameter im Makroaufruf in Typ, Anzahl und Stellung den Formalparametern in der Makrodefinition entsprechen; evtl. Unstimmigkeiten werden jedoch vom Übersetzer festgestellt. Mit Hilfe zweier spezieller geschriebener TR 440 Kommandos kann sich der Anwender über den Inhalt seiner Makrobibliothek informieren und Teile löschen.

Benutzereigene Funktionsbibliotheken sind unter ausschließlicher Verwendung von vorhandenen TR 440 Betriebssystemkomponenten verwirklicht, da es im TR 440 Betriebssystem möglich ist, eine Hierarchie von Bibliotheken zu definieren, die Montageobjekte enthalten und die der Montierer der definierten Rangfolge entsprechend nach Montageobjekten durchsucht.

Um graphische Ausgabe auf Plotter (BENSON) und Bildschirm (SIG 100) zu ermöglichen, wurde ein Programmpaket GRAPHP entwickelt; der CSMP-Aussagenvorrat wurde um die Bearbeitungsaussagen GRAPH, GRAPHN, GRAPHF und LABELG erweitert. Mit den Bearbeitungsaussagen GRAPH und GRAPHN kann spezifiziert werden, in welcher Weise eine Zeichendatei am Ende eines jeden Simulationslaufes ausgewertet werden soll, in die während des Simulationslaufes die Werte der in einer PREPARE-Aussage angegebenen Variablen eingespeichert wurden. GRAPHF kennzeichnet Folgekarten und mit LABELG lassen sich Texte über ein Diagramm schreiben.
In ein Diagramm können bis zu zehn abhängige Variable aus demselben Simulationslauf oder auch aus verschiedenen vorangegangenen Simulationsläufen gezeichnet werden.
Die Skalierung kann

- für alle gemeinsam nach dem tatsächlich aufgetretenen größten Maximum und kleinsten Minimum oder auch nach einem vorgegebenen Maximum und Mimimum
  oder
- für alle oder auch nur einige einzeln nach ihren tatsächlich aufgetretenen Maxima und Minima oder auch nach einzeln vorgegebenen Maxima und Minima

erfolgen.
Der Anwender kann die Zeichendatei in einen eigenen LFD-Bereich kopieren und auch erst später über eigens geschriebene TR 440 Kommandos auswerten; dies ist eine Möglichkeit, die insbesondere für das Zusammenstellen von Diagrammen, Dokumentationen und Veröffentlichungen wünschenswert war.

CSMP-I wird von uns in der beschriebenen Form seit Ende 1972 betrieben. Unsere bisherigen Erfahrungen haben unsere Erwartungen auf ein effektiveres Arbeiten mit einem Simulationsprogrammsystem, die uns zu dieser Entwicklung veranlaßt haben, voll bestätigt.

Vorschläge zur Erweiterung von DYNAMO

O. K o l b e
Institut A für Mechanik
Universität Stuttgart

1 Einleitung

Die Sprache DYNAMO (dynamic models) ist am MIT zur Beschreibung und numerischen Simulation von dynamischen Systemen entwickelt worden. Da bei der Definition von DYNAMO die Systemphilosophie von Forrester /1/ Pate stand, können soziale und ökonomische Systeme in DYNAMO besonders problemnah formuliert werden, ohne daß deshalb die Beschreibung von dynamischen Systemen mit anderem Hintergrund weniger benutzerfreundlich wäre.

Die Klassifizierung der Systemgrößen nach Forrester in Zustands-, Fluß- und Hilfsvariablen, sowie in Konstanten und Tabellen zwingt den Benutzer bei der Programmierung implizit zu der stets notwendigen Struktur- und Funktionsanalyse des untersuchten Systems.

Ein DYNAMO-Programm enthält außer einigen wenigen Spezifikationen zur Steuerung von Rechnung und Ausgabe nur die Bestandteile des Modells. Die Verknüpfungen der Modellgrößen untereinander werden in der Form von Wertzuweisungen entweder direkt angegeben oder über eine DYNAMO-interne Makrotechnik produziert. Bei der Übersetzung werden die Wertzuweisungen eines Modells sortiert und bei der Simulation nach einem festen Algorithmus durchlaufen. Das Initialisieren eines Modells wird von DYNAMO unterstützt, indem die fehlenden Anweisungen soweit möglich aus dem Modell erzeugt werden.

Wiederholungsläufe mit geänderten Konstanten- und Tabellenwerten oder mit veränderter Ausgabe werden vom Benutzer durch entsprechende DYNAMO-Anweisungen vorbereitet, die vom Übersetzer interpretiert werden. Nach einer solchen Rekompilation wird die Simulationsrechnung erneut gestartet. Der Benutzer stellt also seine

Wiederholungsläufe von Hand bereit, und zwar im Batchbetrieb schon vor dem ersten Simulationslauf.

Für detaillierte Beschreibungen der vorhandenen DYNAMO-Realisierungen sei auf die Referenzen /2/ und /3/ verwiesen.

2 Erweiterungen des Sprachumfangs

In diesem Abschnitt sollen einige Erweiterungen des Sprachumfangs vorgeschlagen werden, die dem Autor für eine Vergrößerung der Anwendungsgebiete von DYNAMO sinnvoll erscheinen.

2.1 Anschluß externer Funktionen

Da DYNAMO keine prozedurale Sprache ist, müssen Verzweigungen (bedingte Ausdrücke) mit Hilfe von eingebauten Funktionen realisiert werden. Da diese Funktionen schlechterdings nicht alle denkbaren Fälle direkt erfassen können, besteht das Bedürfnis, entweder wie in CSMP prozedurale Einschübe zuzulassen, oder den Anschluß benutzergeschriebener FORTRAN-Funktionen zu ermöglichen. Die erste Lösung geht der Konzeption von DYNAMO als deskriptiver Sprache vollständig "gegen den Strich" und soll hier nicht weiter erörtert werden. Die zweite Möglichkeit ist seit kurzem (Anfang 1973) in der neuesten Version von DYNAMO II enthalten /2/.

2.2 Anschluß externer Routinen

In DYNAMO I war die Manipulation variabler Felder möglich (tabellarische Erfassung periodischer Signale oder Beschreibung von Totzeitintervallen). Obwohl Operationen wie das Laden oder Schiften eines Feldes an sich typische Aufgaben für Routinen sind, wurde in DYNAMO I für diese Zwecke eine etwas unglückliche Schreibweise in der Form symbolischer Wertzuweisungen verwendet. In DYNAMO II wurde aus diesem Grund auf Feldmanipulationen verzichtet. (Die Sprachversion DYNAMO-TR4 des Autors enthält noch diese Möglichkeiten.)

Da Feldverarbeitungen einerseits von einigem Interesse sind und andererseits die DYNAMO I-Notation ziemlich obsolet ist, kann hier nur eine Erweiterung des Sprachvorrats um den Aufruf externer Routinen Abhilfe schaffen. Dabei muß die Möglichkeit vorgesehen wer-

den, daß Unterprogrammaufrufe von numerischen Ereignissen abhängig sein können (getaktetes Schiften von Feldern). Durch neu einzuführende Anweisungstypen ist zu spezifizieren, wann der (bedingte) Aufruf einer FORTRAN- oder Bibliotheksprozedur fällig sein soll, nämlich vor oder nach der Simulationsrechnung, am Anfang oder Ende jedes Zeitschrittes.

Für einen Übersetzer bedeutet die Einführung von Routinen nicht unbedingt eine wesentliche Komplikation, da die Steuerung des Simulationsablaufs bereits implizit den Aufruf von Routinen nach diesem Konzept erfordert. Dafür können auf diese Weise DYNAMO-Modelle auf externe Dateien zurückgreifen (Vergleich mit realen Daten) oder post mortem Operationen aktivieren (statistische Auswertung der Simulation, Start eines benutzergeschriebenen Ausgabeprogramms).

Die Einführung von Routinen in den Sprachumfang von DYNAMO muß durch die Deklaration der extern berechneten Variablen komplettiert werden, und zwar aus Rücksicht auf eine vollständige Fehlersuche und auf den Sortieralgorithmus.

Da in /2/ beim Aufruf von FORTRAN-Funktionen die Möglichkeit von Seiteneffekten nicht ausgeschlossen ist, ist in DYNAMO II der (unbedingte) Aufruf von Routinen in jedem Simulationsschritt und vor der Simulation schon versteckt realisiert, wenn auch auf eine wenig elegante Weise.

2.3 Felder variabler Länge

Erweitert man den Typenvorrat von DYNAMO um Felder variabler Länge, so lassen sich mit Hilfe einer geeigneten Bibliothek von Routinen in DYNAMO auch Warteschlangen und Stapel verarbeiten (Verkehrsprobleme). In geeigneten Deklarationsanweisungen ist die maximale Länge solcher Felder festzulegen.

2.4 Implizite Schleifen

Durch die Verwendung freier Indizes können Rechenanweisungen für ganze Felder gleichartiger Größen auf DYNAMO-Niveau vereinfacht notiert werden. (Die Einführung impliziter Schleifen ist von Pugh für eine geplante MIT-Version von DYNAMO III vorgesehen.)

2.5 Flexible Abbruchbedingung

Zur Vermeidung unnötiger Rechenzeiten sollte der Abbruch einer Simulation auch durch dynamische Ereignisse herbeigeführt werden können. Dazu würde es ausreichen, einige Modellgrößen als Kriteriumsvariablen zu deklarieren, wobei der Vorzeichenwechsel einer dieser Größen den vorzeitigen Simulationsabbruch herbeiführt. Das bisher verwendete Abbruchkriterium einer vorgegebenen Simulationsdauer kann in das allgemeinere Prinzip durch eine intrinsische Kriteriumsgröße eingebettet werden.

3 Erweiterung der Modellkontrolle

Wie in der Einleitung schon angedeutet wurde, arbeitet DYNAMO bis jetzt im "load-and-go"-Rythmus einen Quellfile ab, der das Modell und eventuelle Wiederholungsläufe enthält. Nun erfordert gerade die Kalibrierung der Parameter eines größeren Modells eine Vielzahl von Wiederholungsläufen mit Parameterwerten, die sich erst aus vorhergegangenen Läufen ergeben. Darüber hinaus folgen die Parametervariationen häufig aus Algorithmen wie dem "maximum likelyhood"-Verfahren (Vergleich des Modellverhaltens mit realen Daten aus der Vergangenheit). Es besteht also ein natürlicher Bedarf der Benutzer dafür, ein in DYNAMO formuliertes Modell von einem Programm in einer algorithmischen Sprache wie FORTRAN aus aktivieren zu können. (Derselbe Bedarf entsteht, wenn in DYNAMO formulierte Betriebsmodelle zur Entscheidungshilfe in ein Management Informationssystem integriert werden sollen.)

Zu diesem Zweck muß im Rechner eine DYNAMO-Realisierung anders als bisher konstruiert werden. Eine Möglichkeit besteht darin, den eigentlichen Compiler aus dem Modell ein Unterprogramm nach FORTRAN-Konventionen und eine Referenzdatei produzieren zu lassen, wobei das Unterprogramm den Namen des Modells trägt und eine Simulationsrechnung durchführen kann, während die Datei Vereinbarungslisten und Ausgabetabellen enthält.

Die bisherige Abwicklung von Simulationsläufen kann durch ein Kontrollprogramm realisiert werden, das vom Compiler als Nachfolger gestartet wird und abwechselnd das Unterprogramm und den

Recompiler aktiviert. Die aktuellen Werte der Modellkonstanten für die Rechnung und die aktuellen Ausgabetabellen kann das Unterprogramm dabei jeweils der zugeordneten Datei entnehmen.

Die Aktivierung eines Modells durch ein FORTRAN-Modell kann einfach durch den Aufruf des entsprechenden, gleichbenannten Unterprogramms erfolgen, das als Parameter zwei Felder hat, wobei über das erste Feld das Modell mit den flexiblen Kenndaten versorgt wird und im zweiten Feld die relevanten Simulationsergebnisse abzulegen sind.

Auf der Seite von DYNAMO werden für den Anschluß an FORTRAN einige neue Sprachelemente benötigt, um das Modell zu benennen und um die Ein- und Ausgangsgrößen des Modells auszuzeichnen.

Es kann auch daran gedacht werden, durch geeignete Steueranweisungen für häufig auftretende Anwendungsfälle Standardrahmenprogramme zu erzeugen. Von besonderem Interesse wären hier ein statisches Modelloptimierungsverfahren zur Kalibrierung, sowie ein Kontrollprogramm, das in geschachtelten Schleifen einige Modellkonstanten variiert.

4 Beispiel

Zur Illustration einiger der oben vorgeschlagenen DYNAMO-Erweiterungen sei die Kalibrierung der Koeffizienten α und β des Anfangswertproblems

$$\ddot{x} + \alpha(x^2-\beta)\dot{x} + x = 0 \quad ; \quad x(0) = 1 \quad , \quad \dot{x}(0) = 0$$

programmiert. Von der tatsächlichen Trajektorie x(t) seien 201 äquidistante Messungen im Intervall $0 \leq t \leq 2$ bekannt.

DYNAMO III-Programm

```
*      VAN DER POL-GLEICHUNG MIT UNBEKANNTEN KOEFFIZIENTEN
NAME   VDPOL
PUT    ERROR                                  { Kennzeichnung der
GET    PLTPER,ALFA,BETA,XM                    { Übergabegrößen
T      XM(201)
PLOT   X=X,XSOLL=*/V=V
```

```
L     X.K=X.J+DT*V.J
L     V.K=V.J-DT*(X.J+ALFA*V.J*(X.J*X.J-BETA))
N     X=1
N     V=0
L     ERROR.K=ERROR.J+DT*DIFF.J*DIFF.J*TRIGGER(TAKT)
N     ERROR=0
A     DIFF.K=X.K-XSOLL.K
A     XSOLL.K=XM(TIME.K/TAKT+1)
C     DT=0.001/LENGTH=2/TAKT=0.01
RUN   ERGEBNISSE
```

FORTRAN-Funktion

```
      FUNCTION FEHLER(AB)
      DIMENSION AB(2),UEBGB(205)
      COMMON PLTPER,ALFA,BETA,XMLONG,XM(201),RES(1)
      EQUIVALENCE (PLTPER,UEBGB(1))
      ALFA=AB(1)
      BETA=AB(2)
      CALL VDPOL(UEBGB,RES)
      FEHLER=RES(1)
      RETURN
      END
```

FORTRAN-Hauptprogramm

```
      DIMENSION AB(2)
      COMMON PLTPER,ALFA,BETA,XMLONG,XM(201)
      EXTERNAL FEHLER
...   Einlesen von XM und von Rohwerten für α und β.
      XMLONG=201.0
      PLTPER=0.0          (Einfrieren der Ausgabe)
...   Aufruf eines Gradientenverfahrens für FEHLER
      PLTPER=0.01         (Aktivieren der Ausgabe)
      DUMMY=FEHLER(AB)
      STOP
      END
```

Anmerkungen:

- TRIGGER(TAKT) bedeutet den Aufruf eines geplanten, eingebauten DYNAMO-Makros, der in Abständen TAKT den Wert 1 und sonst den Wert 0 liefert.
- Wie aus der Anweisung für XSOLL ersichtlich ist, wird die DYNAMO-Notation eines Feldelements an FORTRAN angelehnt.
- Bei der Übergabe eines Feldes zwischen FORTRAN und DYNAMO wird zusätzlich die aktuelle Länge des Feldes mitgeliefert (im Beispiel XMLONG=201 vor dem Feld XM). Diese Übereinkunft ist wegen der vorgesehenen Felder mit variabler Länge notwendig.
- Die Entwicklung eines Precompilers auf FORTRAN-Basis für die skizzierte Sprachversion DYNAMO III ist an der Universität Stuttgart geplant.

5 Referenzen

/1/ Forrester, J.W.: Principles of Systems, Wright-Allen Press, 1968.

/2/ Pugh, A.L. III: DYNAMO II User's Manual, Fourth Edition, MIT Press, 1973.

/3/ Fuchs, G. und Kolbe, O.: Simulation dynamischer Systeme mit DYNAMO, Vorlesungsskript, Institut für Informatik der Universität Stuttgart, WS 1972/73.

Forschungsgruppe : "Systeme zur Informationsverwaltung"
am Institut für Informatik
Universität Stuttgart

Vortragender : Stübel, G.

Beteiligte Personen : Beck, R.; Christ, P.; Falkenberg, E.;
Schneider, H.-J.; Sedlacek, K.-D.;
Stübel, G.

Thema : **Heuristische Verfahren bei der Simulation von Managementplanungsmodellen**

Gesprächsbeitrag für das Fachgespräch über
" Methodik der Rechnergestützten Simulation"
Karlsruhe 10.5. - 11.5.1973

1. Einführung in das Gesamtprojekt

Montgomery und Urban [7] beginnen ihr Buch "Management/Science in Marketing" mit einem Dialog von 1988. Kern dieses Dialogs ist die Entscheidung über die Einführung eines neuen Produktes. Die Zukunftsvision liegt darin, daß sämtliche Entscheidungsalternativen in diesem Dialog mit Hilfe eines Simulationsmodells während des Gesprächs getestet werden. Erst als die Teilnehmer einen Überblick über die Auswirkungen der Fragen "Was geschieht mit einem Unternehmensziel z.B. Return on Investment, wenn eine bestimmte Maßnahme, z.B. Einführungswerbung ergriffen wird" gewonnen haben, somit also das Risiko abschätzen können, werden sie aktuelle Maßnahmen ergreifen.

Diese Motivation liegt der Konzeption des Unternehmensgesamtmodells, das im folgenden kurz umrissen werden soll, zugrunde.

Die Untersuchungen stützen sich auf die Problemstellung aus einem Unternehmen, das mit 1500 Beschäftigten und einem Umsatz von 84 Mill. zu den Betrieben mittlerer Größenordnung gehört.

Das Unternehmen produziert sanitäre Einrichtungen für Badezimmer und verkauft diese im wesentlichen an Großhändler der Sanitär- und Elektrobranche.

Aus dieser Firma liegen empirische Daten für die Ausgangswerte des Modells vor. Wichtiger als die Daten ist jedoch die Kenntnis über die geistigen Modelle (mental models) der Entscheidungsträger, sowie über die Größen,an denen sie sich orientieren. Dazu schreibt Amstutz:

"Subjective evaluation is present whether or not an explicit model is established. However as a model is made explicit, the level of abstraction - the extent to which the model builder has reduced the situation to a limited and manageable number of factors - becomes more obvious"

Das Simulationsmodell stellt also die dynamischen Beziehungen zwischen den vom Planenden als wesentlich angesehenen Größen dar und kann so Aussagen über die Folgen von Entscheidungen liefern, ohne bereits optimale Alternativen auszuwählen.

2. Entwurf des Gesamtmodells

Da auf den verschiedenen Stufen der Unternehmenshierarchie die unterschiedlichen Modellvorstellungen zum Tragen kommen, ist es sinnvoll, daß diese Hierarchie auch im Aufbau des Modells zum Tragen kommt.

Dazu wird das Modell in Elementarebenen eingeteilt. Diese bilden gemäß den Einteilungskriterien

- Phaseneinteilung (Planung, Realisierung, Kontrolle)
- Funktionseinteilung (Vertrieb, Fertigung ....)
- Hierarchie (Top-, Middle-, Operierende Ebene)

ein dreidimensionales Gebäude.

Besonderer Wert wird dabei auf die Schnittstellen zwischen den einzelnen Teilmodellen gelegt. Diese sind so aufgebaut, daß nur wenige Größen, die in einem Mittel-Zweck Verhältnis stehen, d.h. die zugleich Ziel- und Steuerungsgrößen darstellen, an andere Teilsysteme übergeben werden (management by objectives).

3. Teilsysteme

Die relative Selbständigkeit einzelnen Teilsysteme wird dadurch erreicht, daß die Kommunikation mit der Umwelt sowie mit abhängigen Teilsystemen zunächst aufgrund von funktionalen Annahmen über deren

Verhalten erfolgt (1). Diese Responsefunktionen werden dann durch weitere Teilmodelle ersetzt. Ausgehend vom Topmodell werden so die Strukturen immer weiter verfeinert (top down approach).

4. Heuristiken zur Simulation des Unternehmensgeschehens

Bei der Untersuchung der vorliegenden Entscheidungsaufgaben kann festgestellt werden, daß es sich hauptsächlich um unvollständig formulierte Entscheidungsaufgaben handelt. Diese liegen nach Klein [6] dann vor, wenn

(1) wesentliche Elemente der Aufgabenstellung unbekannt sind, oder sich einer genauen Erfassung (ins sondere in Form numerischer Ausdrücke) entziehen, weil die Strukturen zu komplex sind.

(2) das Lösungskriterium nicht eindeutig formuliert ist. Dadurch wird es möglich, daß subjektive, nicht nachprüfbare Wertungen darüber entscheiden, ob eine Lösung vorliegt.

Zur Lösung solcher Probleme beschäftigten sich vor allem Newell, Simon [8] mit heuristischen Entscheidungsmodellen. Wir verstehen hier unter heuristischem Prinzip "eine Regel, die dazu beiträgt, daß im Durchschnitt der Zeitaufwand zur Lösung von Entscheidungsaufgaben verringert wird" [6].
Diese empirisch-kognitive Orientierung geht von der Forschungsstrategie aus, daß bei der Entwicklung von Computermodellen zur maschinellen Problemlösung die Abbildung des menschlichen Problemlösungsverhalten erfolgversprechend ist [2]. Gesucht ist ein Verfahren, das aus strategischen Oberzielen, wie Wachstum, Erfolg, Risiko Werte für Subziele wie Auftragseingang, Deckungsbeitrag ermittelt.

Obwohl eine Reihe von Untersuchungen über Unternehmensziele und Zielsysteme vorliegen (besonders zu erwähnen wäre hier E. Heinen: "Grundlagen betriebswirtschaftlicher Entscheidung - Das Zielsystem der Unternehmung" [5], fehlt eine geeignete mathematische Definition des Begriffs "Zielsystem". Das ist aber unabdingbare Voraussetzung für die Formulierung einer allgemeinen Methode zur Ermittlung derjenigen Maßnahme, die alle gesetzten Ziele optimal erreicht.

Das Problem der optimalen Zielerreichung liegt darin, daß einzelne Ziele eines Systems nicht unabhängig voneinander suboptimiert werden können, weil infolge konkurrierender Ziele sich bei einem Ziel ein unerwünschtes Extremum einstellt, wenn ein anderes optimiert wird. Beispiel: Die Minimierung des Zieles "Risiko" kann eine unerwünschte Minimierung von "Wachstum" bedeuten.
Eine Lösung des Problems ist möglich, wenn man jedem Ziel ein bestimmtes subjektives Anspruchsniveau und darüberhinaus bestimmten Zielen eine Rangstufe der Wichtigkeit oder Bevorzugung zuordnet.

Hier möchte ich nun eine mathematische Definition eines Zielsystems vorschlagen, die sich als sinnvoll für die Ordnung von Zielen zu einem System und für ein allgemeines Lösungsverfahren erwiesen hat.
Definition: Es sei eine indizierte, disjunkte Mangenfamilie $\mathfrak{Z}$ (Indexmenge I) und eine nichtleere Menge von Abbildungen $f: \mathfrak{A} \to X_j$, $X_j \in Z$ $(j \in I)$ gegeben. Für alle Teilmengenfamilien $\mathfrak{T} \subset \mathfrak{Z}$ sei genau eine Mengenfamilie erklärt durch
$\mathfrak{A} = \{ A \mid A$ enthält von jeder Menge $X_i \in \mathfrak{T}$ $(i \in I)$ genau ein Element $x \in X_i \}$

Eine Menge $X_i \in Z$ heißt dann Zielmenge O-ter Ordnung, wenn es eine Abbildung $f: \mathfrak{A} \to X_j$, $X_j \in \mathfrak{Z}$, gibt, für die $X_i \neq X_j$ gilt. Eine Menge $X_j \in \mathfrak{Z}$ heißt Zielmenge n-ter Ordnung, wenn es eine Abbildung $f: \mathfrak{A} \to X_j$ gibt, für die gilt: Es gibt mindestens eine Menge $X_i \in \mathfrak{T}$ ($\mathfrak{T}$ ist die zu $\mathfrak{A}$ gehörige Teilmengenfamilie), die Zielmenge (n-1)-ter Ordnung ist, und alle anderen $X_i \in \mathfrak{T}$ sind ausschließlich Zielmengen kleiner als (n-1)-ter Ordnung.
Wenn nun für $\mathfrak{Z}$ eine Bedingung der folgenden Art erfüllt ist, dann heißt $\mathfrak{Z}$ ein Zielsystem:
Zu jedem $X_j \in \mathfrak{Z}$ existiert höchstens eine Abbildung $f: \mathfrak{A} \to X_j$.
Aufgrund dieser Definition läßt sich folgender Satz beweisen:

Satz: Es existiert eine Zerlegung eines Zielsystems $\mathfrak{Z}$ in Klassen mit Zielmengen gleicher Ordnung.

Man kann außerdem eine Reihe weiterer Definitionen prägen für Begriffe wie Ziel, Basismenge, Maßnahme, optimierbares Zielsystem, zulässiges Bild, zulässige Lösung, optimale Lösung usw. Teilweise sind

die Definitionen schon vom Begriff her klar, und so soll jetzt nur noch ein Algorithmus zum Auffinden mindestens einer zulässigen Lösung skizziert werden.
Der Algorithmus beruht darauf, dass nacheinander in jeder Klasse von Zielmengen des Systems der Durchschnitt aller Bereiche von zulässigen Bildern der einzelnen Zielmengen ermittelt wird. Zulässige Bilder kann man mithilfe der Monte-Carlo-Methode finden. Auf diese Weise wird auch ein Bereich der Basisvariablen für zulässige Lösungen mit einer Sicherheit 1 ermittelt. Die Vorgabe, wieviel Prozent aller zulässigen Lösungen dieser Bereich umfassen soll, bestimmt die Anzahl der Rechenschnitte, die ausserdem noch proportional zur Anzahl der Zielmengen ist. Es kommt jetzt nur noch darauf an, eine geeignete Kombination von Werten der Basisvariablen zu finden, die eine zulässige Lösung ausmachen.
Mindestens eine zulässige Lösung findet man durch folgendes Vorgehen:
Nach der Bestimmung der Grenzen für zwei Basisvariablen wird der Koordinatenursprung in den Anfang der Bereiche zulässiger Werte für Basisvariablen gelegt. Dann muss unter der Voraussetzung, dass die Menge aller zulässigen Lösungen ein zusammenhängendes Gebiet bilden, eine Gerade durch den Ursprung mindestens eine zulässige Lösung treffen (Bild 5).

So kann das Erfolgsziel mittels der Kennzahl Return on Investment = Gewinn gemessen werden.
Gesamtkapital

Mögliche Zielwerte richten sich nach der banküblichen Verzinsung des Kapitals, also ca. 8%. Je nach Unternehmensgeschichte liegt also das Anspruchsniveau für den Return on Investment entsprechend der Unternehmensstrategie und der subjektiven Einschätzung über diesem Wert. Dieser Zweck kann mithilfe der Mittel Umsatzrentabilität = Gewinn/Umsatz oder des Mittels der Umschlagshäufigkeit = Umsatz/Gesamtkapitalrentabilität erreicht werden.
Das heuristische Prinzip der Verselbständigung von Zweck-Mittel Komponenten beinhaltet nun neue Aussagen über die Selektion des

relevanten Feldbereiches d.h. es werden nun neue Regeln und selbständige Aussagen über das Anspruchsniveau der Subziele gemacht.

Die Abbildung der Subziele ist durch die Multiplikation der beiden Grössen gegeben.

$$\frac{\text{Return on Investment}}{\text{(ROI)}} = \frac{\text{Gewinn}}{\text{Umsatz}} \cdot \frac{\text{Umsatz}}{\text{Gesamtkapital}}$$

ROI = Umsatzrentabilität x Gesamtkapitalumschlagshäufigkeit

Mit Hilfe des oben beschriebenen Programms, das das Abstimmungsproblem zwischen den einzelnen Unternehmensressorts simuliert, werden alle Planwerte der Elementarebene Planung im Top-Management bestimmt.
Im dynamischen Modell, das nach der Methode von System dynamics aufgebaut ist, werden nun je nach Abweichungen von gegebenen Plankenngrössen Massnahmen getroffen, die das entsprechende Subziel zu vebessern sucht. Eine entsprechende Entscheidungsregel lautet in Dynamo

$$\text{Aufwendungen für Vertrieb. KL} = \frac{\text{geplante Aufwendungen für den Vertrieb. JK}}{(1+q.\ \text{Umsatzabweichungen. JK})}$$

q ist ein subjektiver Parameter.
Gemäss einer gegebenen Responsefunktion [1] antwortet der Markt entsprechend verzögert (Delay 3. Ordnung) mit einem veränderten Auftragseingang.

Die Umgebung des Simulationsmodells

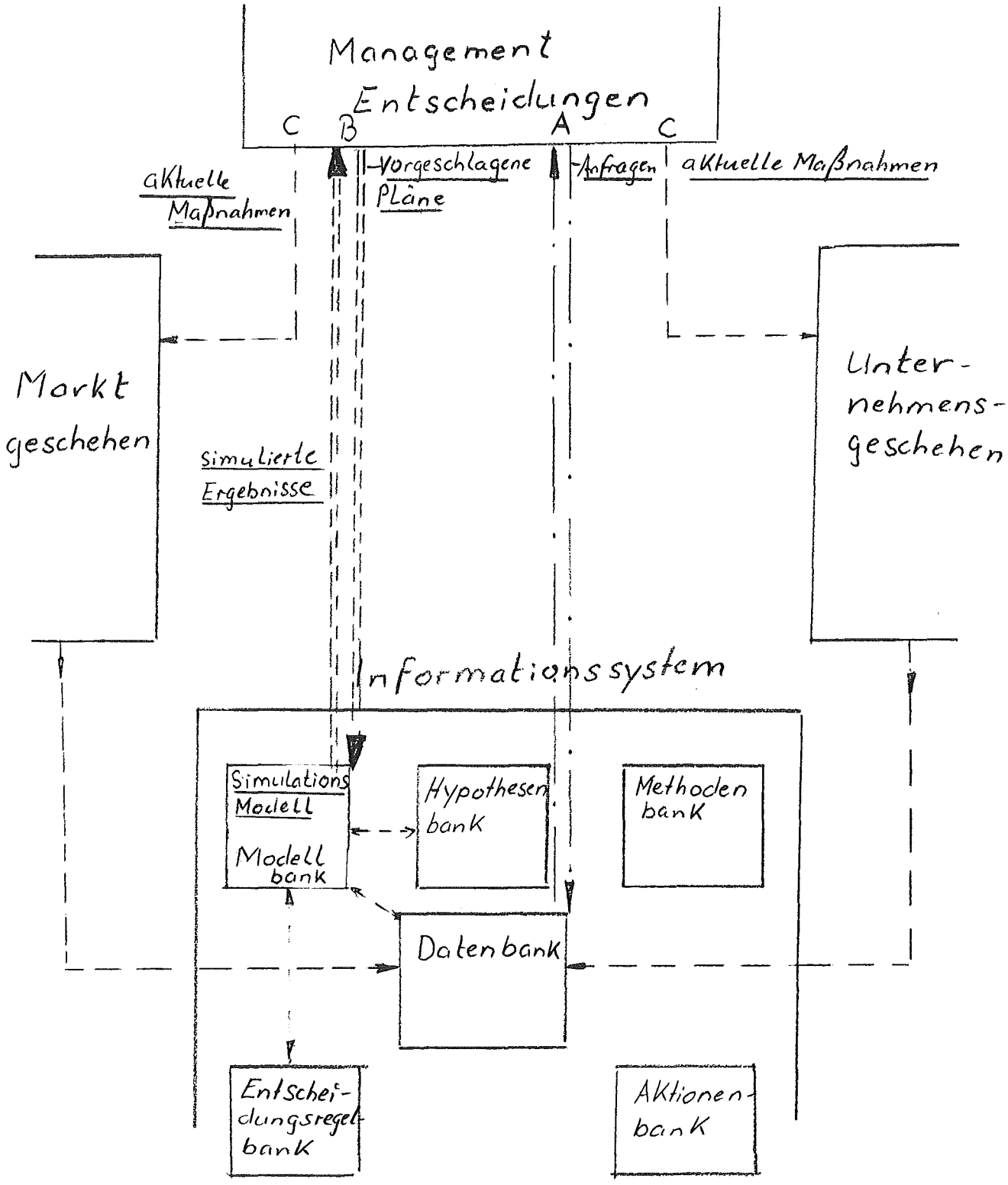


Bild 1

ELEMENTAREBENEN EINES
UNTERNEHMENSMODELLS

GLIEDERUNG NACH

MANAGEMENTEBENEN

FUNKTIONEN

PHASEN

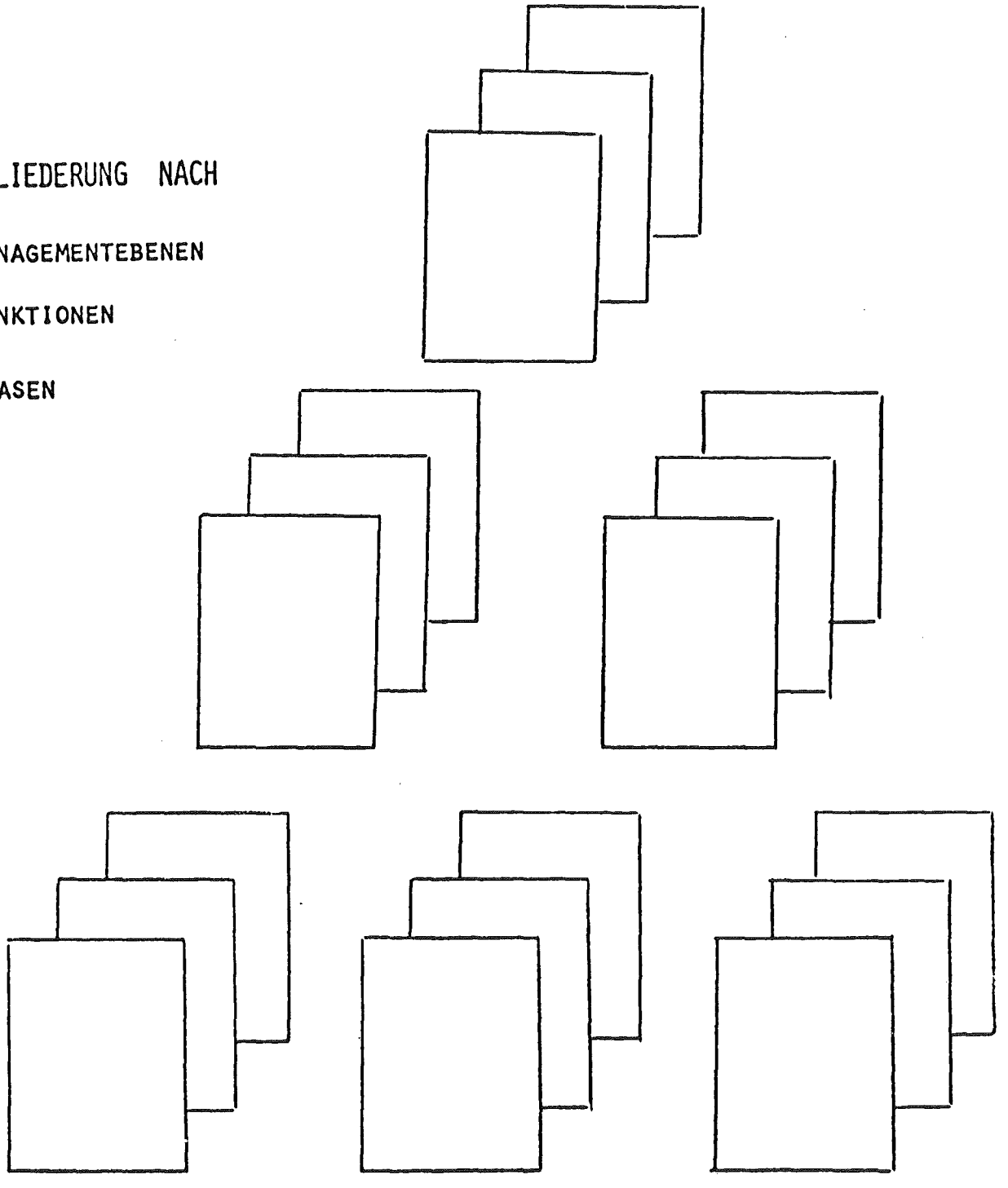
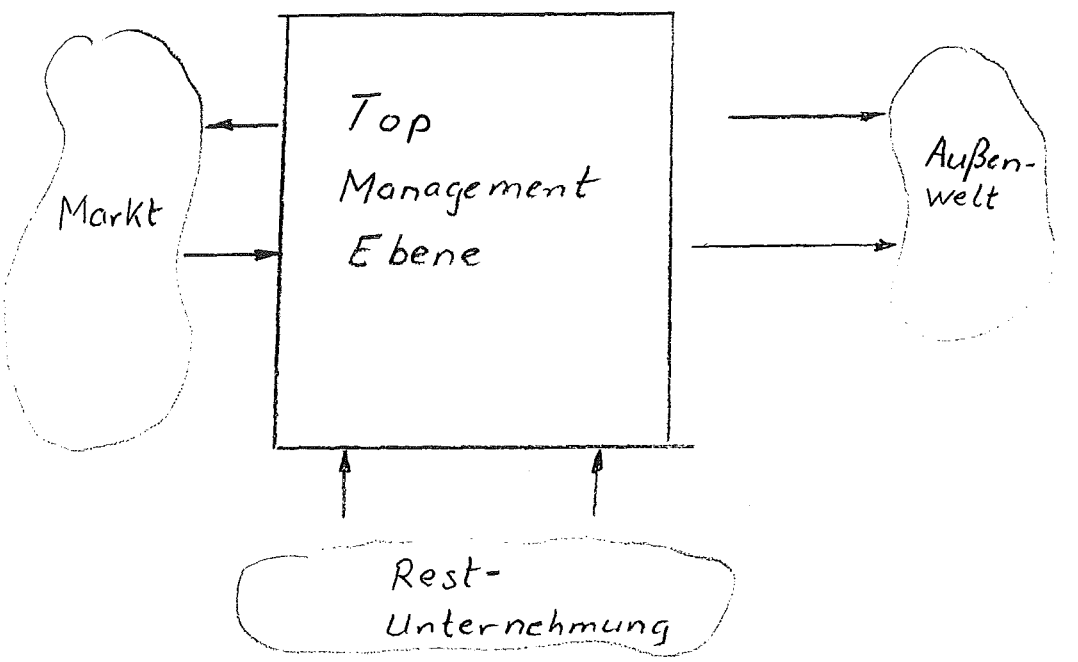


Bild 2

Stufen der Modellrealisierung



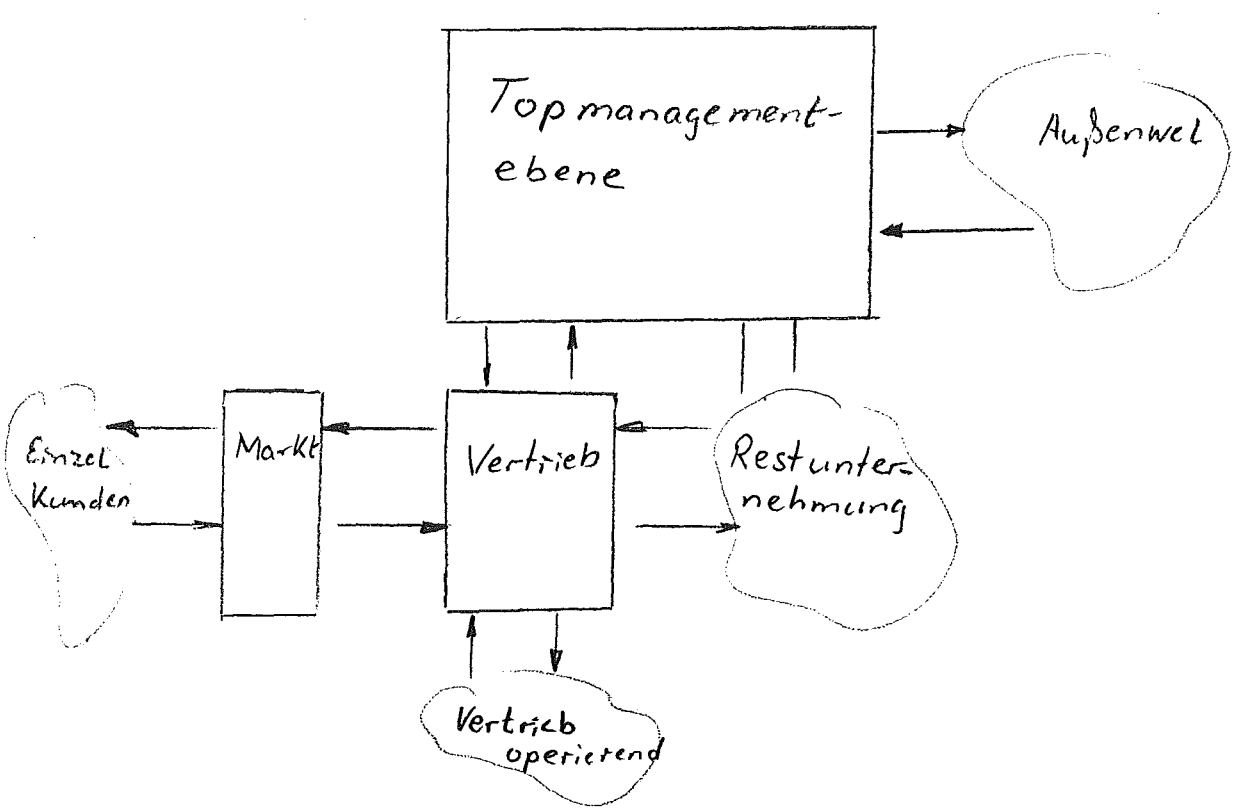


Bild 3

Abbildung 4: Optimierbares Zielsystem (eine Zielmenge n-ter Ordnung)

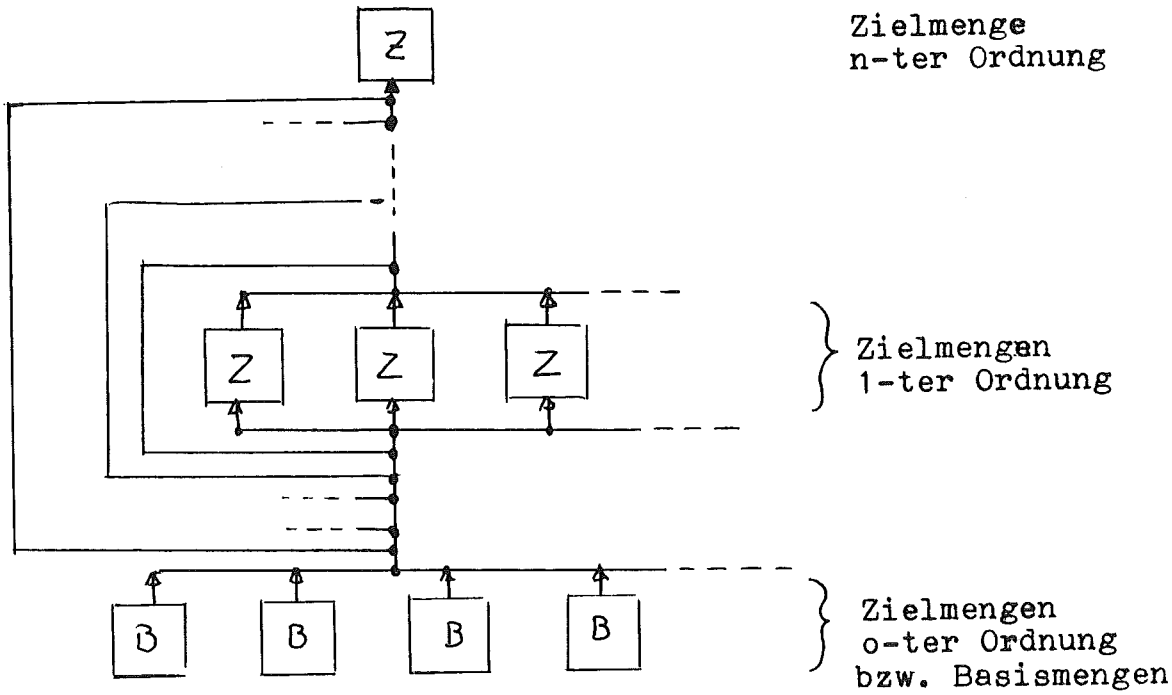
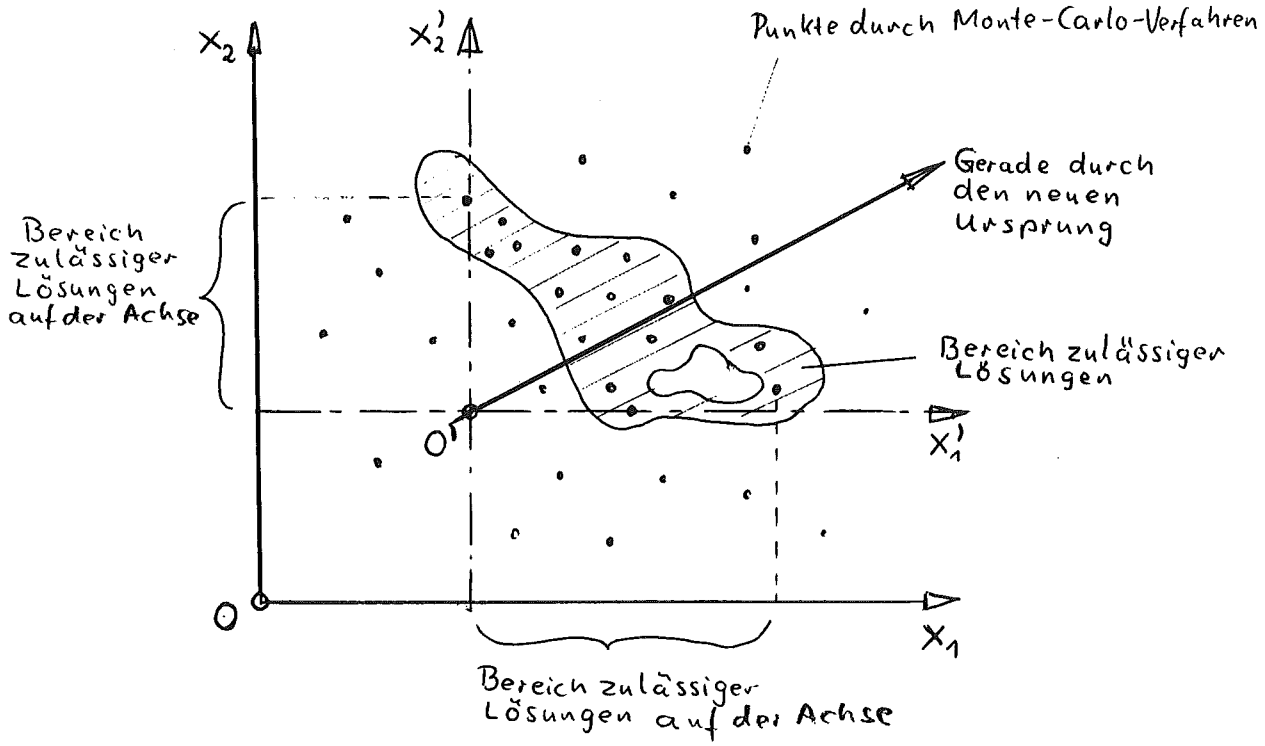


Abbildung 5: Algorithmus zum Auffinden einer zulässigen Lösung bei 2 Basisvariablen.



Literatur

(1) Amstutz, Arnold E.
Computer Simulation of Competitive Market Response
MIT Press, Cambridge 1967

(2) Frese, Erich
Heuristische Entscheidungsstrategien der Unternehmensführung.
Zeitschrift für betriebswirtschaftliche Forschung.
23 Sg. 1971

(3) Cyert, Richard M., March, James G.
A Behavioral Theory of the Firm,
Englewood Cliffs, N.J. 1963

(4) Ansoff, GOR H.
Management Strategie,
München 1966

(5) Heinen, E., Grundlagen betriebswirtschaftlicher Entscheidungen -
Das Zielsystem der Unternehmung. Wiesbaden 19

(6) Klein, K.-H. Heuristische Entscheidungsmodelle,
Wiesbaden 1971

(7) Montgomery, David B., Urban Glen
Management Science in Marketing
Englewood Cliffs 1969

(8) Simon, H.A., Newell, A.,
Heuristic Problem Solving by Computer in:
Computer Augmentation of Human Reasoning.
Washington 1965

Adolf Reinhardt
Technische Universität Berlin

Rechnergestützte Simulation zur Entwicklung von Fertigungssystemstrukturen+

1. Entwicklung von Fertigungssystemen

Die Automatisierung in der Einzel- und Kleinserienfertigung hat im Gegensatz zur Massenfertigung bisher vereinzelt zur Entwicklung komplexer Fertigungssysteme geführt, in denen die Werkstückherstellung vom Rohteil bis zum Fertigteil automatisch abläuft. Untersuchungen über die zeitlichen Verhältnisse, die den Durchlauf von Werkstücken im Bereich der konventionellen Fertigung bestimmen, zeigen ein Verhältnis von ca. 5% Auftragszeit zu 95% Transport-, Liege - und Wartezeit bezogen auf die Gesamtdurchlaufzeit / 1 /.
Der erhebliche Anteil der Transport- und Liegezeiten ist ein wichtiges Kriterium für die Entwicklung solcher Systeme wie sie beispielsweise das Bild 1 zeigt. Für einzelne realisierte Systeme werden die erreichten Effekte durch folgende Aussagen beschrieben / 2 /:

Senkung des Arbeitskräftebedarfs um 70%;
Minderung der Produktionsfläche um 50%;
Steigerung der Arbeitsproduktivität um 300%.

Die Erfahrung macht deutlich, daß sowohl die Planung als auch der Einsatz solcher Systeme ein Investitionsrisiko darstellen. Die Entwicklung ist durch das zeitraubende Experiment mit Hardware und Software an Prototypen gekennzeichnet, was die Untersuchung alternativer Konzepte ausschließt. Über die rechnergestützte Simulation als Planungsalternative liegt bisher in diesem Bereich so gut wie keine Erfahrung vor, sieht man von den zahlreichen Job-Shop-Simulationen ab, die das Problem kaum treffen.

+ Unter Mitarbeit von A.-W.Kamp, Chr.Hampel, E.Nullmeier, P.Wagner

2. Modellexperimente

Der Übergang vom Experiment am real aufgebauten System zu Untersuchungen am quasirealen System, dem Modell, muß zwei grundlegende Bedingungen erfüllen:

- Das Modell muß die Nachteile des experimentellen Entwurfs aufheben: es soll also billig, gefahrlos, zeitraffend und praktisch unbegrenzt veränderbar sein.
- Das Modellexperiment soll die Vorteile in Bezug auf den Erkenntnis- und Entwicklungsprozeß im realen Experiment annähern; es soll Interaktionsmöglichkeiten enthalten, das Systemmodell in seinem Zustand sichtbar machen und die Entwicklung und Erprobung von Entscheidungs- und Steueralgorithmen ermöglichen.

An das Modellsubstrat "Digitalrechner" ist damit zunächst die Forderung der Dialogfähigkeit zu stellen, wie sie heute bereits in Teilnehmersystemen vorhanden ist.
Für die Erfassung des räumlich-zeitlichen Verhaltens im Modell bieten sich interaktive Bildsichtgeräte an.
Für den schrittweisen Übergang vom Modellexperiment zur Echtzeitsimulation, in der Komponenten eines realen Systems angeschlossen werden, sind Prozeßrechnereigenschaften erforderlich.
Entsprechend werden zur Modellierung Simulationssprachen benötigt, die vom Konzept her bereits als Dialogsprachen aufgebaut sind und um Elemente von graphischen Sprachen und Prozeßsprachen erweitert werden können.

3. Simulationssprachen

Entsprechend dem Entwicklungsstand von DVA und Programmiersprachen wurden in den 50er Jahren für spezielle Systemplanungen sogenannte Simulatoren in Assemblersprachen und später in Compilersprachen erarbeitet, deren Änderungen und Anpassungen auf differenzierte Problemstellungen umfangreiche Programmierarbeiten zur Folge hatten. Die Entwicklung zu allgemein benutzbaren Simulationssprachen wurde sowohl aus dem Problembereich der Analyse und Planung als auch aus dem der DVA-Programmierung vorwärts getrieben.

Für die in der Systemplanung notwendigen Gedankenmodelle und -experimente wurden Fachsprachen mit eindeutiger Begriffs- und Symboldefinition zur verbalen Beschreibung bzw. visuellen Darstellung dieser Modelle entwickelt. Die programmierte Form dieser Gedankenmodelle sollte als weiteren Schritt das Testen der Modellogik ermöglichen und im Modellexperiment das "Gedankenspiel" fortführen / 3 /.

Modellierungsprozeß

Im Vordergrund stand zunächst der Modellierungsprozeß. Es galt ein System so zu strukturieren, daß es in seinem möglichen Zustand durch permanente und temporäre Einheiten (Entities) mit ihren Eigenschaften (Attributes) zu jedem Zeitpunkt statisch eindeutig beschrieben war. Außerdem sollte die Dynamik des Systems, der Systemprozeß, als Folge von Zustandsänderungen dargestellt werden. Aktivitäten bewirken dabei Zustandsänderungen die zu Ereignispunkten realisiert werden. Hinsichtlich der Modellierung lassen sich zwei Sprachkonzepte unterscheiden. In einer Simulationssprache wie GPSS werden Modelle aus Flußdiagrammsymbolen zusammengesetzt, die den funktionalen Systemaufbau gut veranschaulichen. Sprachen wie GASP, SIMSCRIPT, GSP, CSL usw. sind anweisungsorientierte Sprachen und zielen auf eine mehr verbale Systembeschreibung. Bezüglich der Darstellung der Systemdynamik können ereignisorientierte (z.B. GASP,SIMSCRIPT) und aktivitätenorientierte (z.B. CSL) Sprachen unterschieden werden. Neuere Entwicklungen wie SOL und SIMULA führen den Begriff "Prozeß" ein, der Serien von Aktivitäten zeit- oder statusbedingt zu dynamischen Einheiten zusammenfaßt / 4 /.

Die Möglichkeit der Wahl zwischen den einzelnen Sprachkonzepten und deren Einschätzung für den konkreten Anwendungsbereich sind trotz des breiten Spektrums durch die jeweils greifbare DVA bestimmt.

Modellprozeß

Der Modellprozeß wird im Rahmen der üblichen Simulationssprachen

in drei Phasen zerlegt (Bild 2) / 5 /:

- Initiierung des Modellanfangszustandes analog einem realen Systemzustand mit wenigstens einem Anfangsereignis.
- In einem Simulationszyklus wird das Modell entsprechend dem Prozeßalgorithmus nach einem internen Zeitmechanismus von einem Modellzustand zum nächsten fortgeschaltet. Das Erzeugen von charakteristischen Zahlenreihen, die Organisation von Daten und die Registrierung des veränderten Modellzustandes erfolgen automatisch.
- Zu einem vorgewählten Zeitpunkt wird der Simulationslauf mit der Verarbeitung der registrierten Daten zu Statistiken beendet.

Dieser Ablauf des Modellprozesses hat den Nachteil, daß der Modellbauer von der Initiierung des Modells bis zur Ergebnisausgabe im "Dunkeln" sitzt. Mittels der Ergebnisse über das Modellverhalten in Form von Statistiken einzelner Modellvariablen mit Mittelwert, Standardabweichungen usw., eventuellen Zwischenausdrucken zu bestimmten vorgewählten Zeitpunkten, muß der Ablauf des Modellprozesses rekonstruiert und die Validität des Modelles ermessen werden.

4. Fertigungssystemsimulation

4.1 Problemstellung

Für die Analyse und Planung von Fertigungssystemen bestehend aus numerisch gesteuerten Werkzeugmaschinen, die über den Material- und Informationsfluß verkettet sind, wird die Methodik der Simulation an konkreten Beispielen untersucht. Die uns derzeit im Stapelbetrieb zur Verfügung stehenden Simulationssprachen GPSS und SIMSCRIPT II auf den DV-Anlagen IBM 360/67 und SIMULA auf der Anlage CDC 6500 werden auf die Einsatzfähigkeit für diesen Bereich an konkreten Fertigungssystem-Modellen untersucht. Ausgangspunkt der Modellentwicklung ist dabei die Simulationssprache GASP, die die Programmiersprache FORTRAN zur Grundlage hat. Sie besteht aus ca. 30 Unterprogrammen und entspricht im Konzept SIMSCRIPT. GASP kann auf allen verfügbaren DVA eingesetzt

werden und ist somit als Grundlage für vergleichende Betrachtungen sowohl der verschiedenen Simulationssprachen als auch der verschiedenen DVA benutzbar. Weiterhin kann GASP durch FORTRAN-Routinen

- für die Darstellung räumlicher und zeitlicher Verhaltensänderung in Systemmodellen über Bildsichtgeräte und
- für die Echt-Zeit-Simulation mit realen Systemkomponenten für die Online-Prozeßkopplung erweitert werden.

Speziell für diese Möglichkeiten steht das Teilnehmersystem PDP 10-PDP 15 mit interaktivem Bildsichtgerät und Prozeßkoppeleinrichtung zur Verfügung. Neben dem Vergleich von Simulationssprachen und DV-Anlagen werden Konzepte für Programmsysteme bzw. Simulationssprachen erarbeitet, die speziell am Beispiel der Fertigungssystemsimulation überprüft werden.

4.2 Top-Down-Modellierung

Die Vorgehensweise wird dabei durch ein Top-Down-Verfahren bestimmt. In diesem Verfahren wird ein System als Menge von Subsystemen verstanden, die wieder je nach Ziel und Tiefe der Betrachtung in Subsysteme gegliedert werden können usw. In der untersten Ebene bestehen die Subsysteme aus Einheiten, die in den Begriffen der benutzten Sprache beschrieben werden können bzw. in einer Fortentwicklung des Modells durch detailliertere Einheiten ersetzt werden können.
Ziel des ersten und hier vorgetragenen Schrittes ist der Werkstückfluß in einem verketteten Fertigungssystem zur Ermittlung von Systemstrukturen. Diese sollen eine flexible Werkstücksteuerung nach Kriterien wie hohe Auslastung von Bearbeitungseinheiten, kurze Transport- und Wartezeiten von Werkstücken erfüllen.
Zu diesem Zweck wurde zu einem in Bild 1 dargestellten System ein dialogfähiges Modell entwickelt.

4.3 Systembeschreibung / 6 /

Das Fertigungssystem besteht aus mehreren numerisch gesteuerten Bearbeitungsstationen mit automatischer Werkzeug- und Werkstück-

wechseleinrichtung, die über Transporteinrichtungen miteinander verkettet sind. Das Transportsystem umfaßt Drehtische und Rollbahnen, die als Vorlagerbzw. Zwischenlager vor und hinter den Bearbeitungsstationen liegen oder als Förderstrecken zwischen jeweils zwei Drehtischen eingebaut sind. Ein Drehtisch kann eine im System umlaufende Werkstückpalette entweder zu einer Bearbeitungsstation oder zu einem anderen Drehtisch lenken. Jede Palette nimmt jeweils ein Werkstück auf.
Die Rollbahnen (Vorlager, Zwischenlager, Förderstrecken) dienen sowohl zum Palettentransport als auch zur Lagerung von Halbfabrikaten. Dabei entsprechen die Rollbahnlängen den Lagerkapazitäten. Durch die endliche Kapazität des Transportsystems können bei hoher Auslastung Teilbereiche des Systems und schließlich das Gesamtsystem blockiert werden.

4.4 Modellaufbau /6 /

a) Subsysteme und permanente Einheiten

Das in Bild 3 schematisch dargestellte System wurde in folgende Subsysteme gegliedert:

A Bearbeitungssystem
B Fördersystem
C Ausgabesystem
D Eingabesystem.

Diese Subsysteme werden in einer weiteren Gliederung durch folgende permanente Einheiten beschrieben:

A1 Vorlagerstrecke
A2 Bearbeitungsstation
A3 Auslagerstrecke
B1 Förderstrecke
B2 Drehtisch
C1 Lagerstrecke
C2 Entspannvorrichtung
C3 Fertigteillager
D1 Rohteillager
D2 Aufspannvorrichtung.

Diese Einheiten werden als relativ autonom entsprechend der Zielrichtung der Modellstufe angenommen. Autonom heißt, daß

in einer Einheit interne Material- und Informationsflüsse existieren können, die jedoch für die Anschlußstellen an andere Einheiten bedeutungslos sind. So interessieren an der Einheit "Bearbeitungsstation", die die Arbeitsmaschine mit Werkstück- und Werkzeug-Wechseleinrichtung sowie eine Steuereinrichtung enthält, für die Wechselwirkung im System allein die Zustände "belegt", "frei", "blockiert" oder "blockierend" und die Zeitpunkte, zu denen die Einheit diese Zustände annimmt. Gleiches gilt für die Einheiten B2, C2, D2. Der Werkzeug- und Informationsfluß speziell in dieser Einheit wird somit als gegeben angenommen, bzw. bleibt einer vertiefenden Strukturierung der Einheit als Subsystem offen. Die Einheit "Drehtisch" wird zusätzlich in ihrer Stellung erfaßt. Die Förder-, Auslager- und Vorlagerstrecken werden in ihrem jeweils aktuellen Zustand sowohl durch die Anzahl der ruhenden als auch der sich bewegenden Paletten beschrieben.

b) Temporäre Einheiten

Die durch das FS zu leitenden Werkstücke werden als temporäre Einheiten über GASP-Attribute-Variablen beschrieben. Die zur Betrachtung der Werkstücke ausgewählten Eigenschaften kennzeichnen:

- den zeit- und ortsbezogenen Zustand eines Werkstückes
- die Kennzeichnung eines Werkstückes als Element eines Losauftrages mit einem spezifischen Arbeitsplan
- die Information über die an einem Werkstück bereits ausgeführten bzw. noch auszuführenden Fertigungsabschnitte.

Sie enthalten somit die Daten, die für den Transport eines Werkstückes von einer Einheit zur nächsten notwendig sind. Die umfassenden Werkstückdaten mit Arbeitsplänen können in Stammdateien rechnerintern oder auf externen Massenspeichern geführt werden. Sie dienen außerdem zur Aufnahme der für die Steuerung der Einheiten notwendigen umfangreichen Information, bei einer Erweiterung des Modells zur Informationsflußbetrachtung.

c) Aktivitäten und Ereignisse

Das Einleiten von Aktivitäten im Systemmodell wird über sogenannte Aktivitätsroutinen bewerkstelligt, beispielsweise für das Belegen einer permanenten Einheit bzw. für den Palettentransport. Ereignisroutinen enthalten die Entscheidungsfolge für die Ereignispunkte, zu denen die Aktivitäten "Aufspannen","Rollen", "Drehen", "Bear-

beiten", "Entspannen" abgeschlossen werden. Zur Beschickung des Systemmodells werden in einem Hilfsprozeß charakteristische Aufträge erzeugt, die zu systemgerechten Arbeitsablaufplänen verarbeitet und als Stammdaten für den Modellprozeß zur Verfügung stehen. Extreme Zustände im Systemmodell wie sie durch die Blockierung einzelner Einheiten,durch Blockierungsketten- und -schleifen entstehen können, werden über Hilfsprogramme zur Entblockierung und Schleifenabfrage registriert und behoben.

4.5 Ergebnisse der Modellexperimente

Der Dialog mit dem Systemmodell erfolgt über Blattschreiber und ein interaktives Bildsichtgerät. Für die räumlich-geometrische Darstellung der Systemstruktur wird dabei eine symbolische Darstellung entsprechend Bild 3 benutzt. Für die Interaktionen mit dem Systemmodell sind sowohl Displayroutinen , die den Systemzustand zu gewünschten Modellzeitpunkten oder aufgrund festgelegter Zustandsbedingungen wiedergeben, als auch Aktionsroutinen zur Beeinflussung des Modellprozesses realisiert. Im Modell CINCIN-01 kann wahlweise eingesehen werden:der Zustand einzelner Subsysteme und Einheiten, der Stand und Inhalt einzelner Dateien oder die Dateienstatistik, der Auftragsstand und die Stammdatenliste. Über Aktionsroutinen kann zu beliebigen Zeitpunkten der Parametersatz der Einheiten, die Disziplin geordneter Einheiten oder die Entscheidungsstrategien für die Belegung und den Transport im Systemmodell geändert werden.
Diese Interaktionsmöglichkeiten erleichtern im ersten Schritt des Modellexperimentes die Überprüfung der Modellgültigkeit. Im Anlaufverhalten kann die allmähliche Entwicklung des Modellprozesses beobachtet werden. Fehler in der Modellogik werden frühzeitig erkannt. Dadurch läß sich die Zahl der Testläufe verringern und damit auch die Rechenzeit und die Menge von Papierausdrucken. Durch schrittweise Veränderung einzelner Modellparameter können Modellabläufe unter verschiedenen Bedingungen während eines Simulationslaufes erzeugt werden. Aus der Erkenntnis über die Auswirkungen der Blockierung in Teilbereichen des Systems, die zur Blockierung des Gesamtsystems führen, wurden beispielsweise Testroutinen

erstellt, die kritische Entwicklungen rechtzeitig erkennen.

Als wesentliches Element in der vorliegenden Betrachtung gilt die heuristische Entwicklung der Strategien zur Belegung der Bearbeitungseinheiten aus ihren Vorlagern und zur Steuerung des Werkstücktransports über die Drehtische an den Verzweigungsstellen im Transportsystem. Durch die Veränderung der Drehtischgrundstellungen in System 1 (Bild 4) entsprechend dem aktuellen Werkstückhauptstrom läßt sich die Blockierungshäufigkeit in Teilbereichen des Systems mindern und dadurch sowohl eine geringe Steigerung des Auslastungsgrades der Bearbeitungseinheiten als auch eine Verkürzung der Werkstückdurchlaufzeiten erreichen. Veränderungen der Rollbahnen und dabei vor allem der Vorlager A 1 (Bild 3) führen zur Systemstruktur System 2 (Bild 4) mit einer Auslastungssteigerung um 7% bei gleichbleibendem Werkstückspektrum. Die Flexibilität des Systems wird dadurch jedoch nicht erhöht.

Die Systemstruktur System 3 ermöglicht im Gegensatz zu System 1 und 2 alternative und kombinierte Belegungsstrategien aufgrund des wahlfreien Zugriffs zu Werkstücken auf drehbaren Vorlagern. Dieser Strukturänderung muß allerdings in einem modifizierten Modell CINCIN-02 durch eine neue Beschreibung der Einheit A1 (Vorlagerstrecke) Rechnung getragen werden.
Eine alternative Anwendung der Strategien

"first come, first served" (FCFS) und
"kürzeste Operationszeit zuerst" (KOZ)

in Abhängigkeit von der Vorlagerbelastung führt beispielsweise zu einer Steigerung der Auslastung um mehr als 30% und zu einer entsprechenden Steigerung des Werkstückdurchsatzes im Vergleich zu System 1.
Die Veränderung des Transportsystems bewirkt Systemstrukturen wie sie in Bild 5, System 4-6,dargestellt sind.Eine Erhöhung der Systemflexibilität erschwert die Systemlenkung und die Nutzung dieser Flexibilität.
Die Vorteile dialogfähiger Modelle liegen - wie die Erfahrung im Rahmen dieser Arbeit gezeigt hat - darin, daß über die Interaktionsmöglichkeit Erkenntnisse über den Modellprozeß gewonnen werden können, als eine Grundlage zur Entwicklung und Verbesserung von Entscheidungsstrategien und Steueralgorithmen

zur Lenkung realer Systeme.

5. Zusammenfassung

Die hier behandelte Problemstellung geht aus von der Reduzierung des Objektbereiches auf den technischen Aspekt. Der Objektbereich ist durch die reale Umgebung, in der ein Fertigungssystem eingesetzt werden soll, bestimmt und damit wesentlich durch ökonomische, soziale und psychische Wirkungszusammenhänge. Die Betrachtung eines Fertigungssystems als technisches Gebilde schränkt die Aussagekraft der Simulationsergebnisse in Bezug auf diesen Objektbereich ein, was als solches bewußt bleiben muß.

Der Modellierungsprozeß wird in jedem Simulationsansatz mehrfach durchlaufen, sowohl in der Phase der Modellvalidierung als auch in der Fortentwicklung des Modells auf Grund einer modifizierten Zielsetzung oder Vertiefung der Betrachtung. Die Entwicklung von Simulationssprachen mit höherer Begriffsqualität und vor allem deren Einsatzbereitschaft auf der jeweils benutzten DVA können diesen Prozeß, der durch zeitraubenden Programmierungsaufwand und -fehlersuche gekennzeichnet ist, vereinfachen und mehr Raum für die Problematik des Objektbereiches schaffen.

Im Rahmen der betrachteten Fertigungssystemsimulation hat der Modellprozeß neben der Erzeugung hochaggregierter Verhaltensaussagen in der Form von Statistiken vor allem den Zweck, durch die Interaktionsmöglichkeit mit dem Modellprozeß das Verhalten komplexer Systeme in einzelnen Verläufen zu erkennen, um damit Gedankenmodelle in ihrer Logik zu überprüfen und zu reproduzieren.
Dazu muß der Modellprozeß vor den Augen des Systemanalytikers ablaufen. Eine Mensch-Maschine-Kommunikation über Bildsichtgeräte ist dabei eine Voraussetzung, die durch die Entwicklung entsprechender Simulationssprachen bzw. deren Erweiterung erfüllt werden muß.

L i t e r a t u r:

1. Optiz, H.: Wirtschaftlicher Einsatz numerisch gesteuerter Werkzeugmaschinen unter Berücksichtigung technologischer Aspekte,
Fertigung und Betrieb 21 (1971) H.9, S.531-539

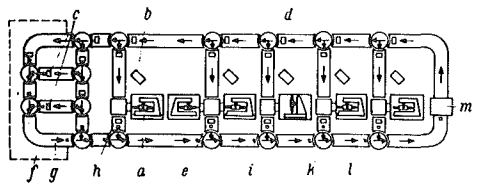
2. --: ROTA-F 125 NC System Niles,
WMW-Export-Import Prospekt Nr. 5917/1971

3. Clementson, A.T.: Control and Simulation Language,
Online-Seminars, The Hague, Jan. 1972

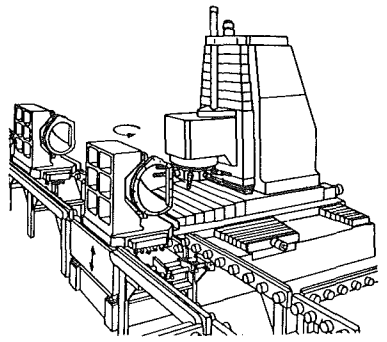
4. Kiviat, P.J.: Development of Discrete Digital Simulation Languages,
Simulation, Febr. 1972

5. Reinhardt, A.: Simulation eines Fertigungsprozesses mit GASP,
TZ f. prakt. Metallbearbeitung Jg. 66 (1972), H.2

6. Kamp, A.-W./Reinhardt, A.: Simulation von Systemen mit Werkzeugmaschinen in Schöne, A./Hrsg./:
Simulation technischer Systeme,
Carl Hanser Verlag (erscheint Sept. 1973)



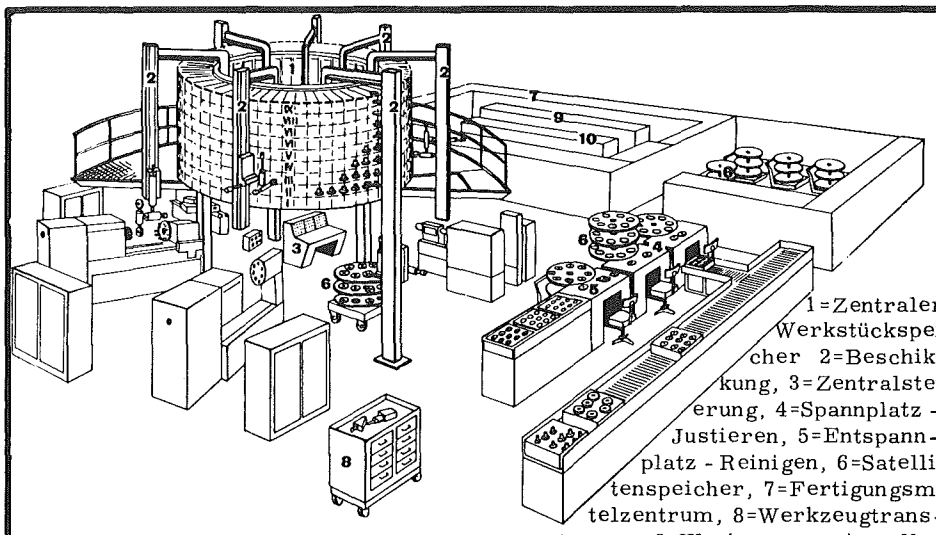
Plan eines automatischen Fertigungssystems für Getriebegehäuse



Bearbeitungsstation mit Werkstücktisch, Vorrichtung und Werkstück

a horizontale oder vertikale Bearbeitungsstation, b numerische Steuerung, c Adressensetzer d Adressenleser, e Adressenumwandler, f Ladestation, g Hauptförderer, h Querförderer, i Drehtisch-Signalgeber, k Drehtisch, l Werkstücktisch, m in die Fertigung einbezogene Waschstation

Liniensystem (CINCINNATI)



1=Zentraler Werkstückspeicher 2=Beschikkung, 3=Zentralsteuerung, 4=Spannplatz - Justieren, 5=Entspannplatz - Reinigen, 6=Satellitenspeicher, 7=Fertigungsmittelzentrum, 8=Werkzeugtransportwagen, 9=Werkzeugvoreinstellen, 10=WZ-Regale I. . . IX=Speicherringe

Sternsystem (ROTA-F 125 NC)

TU BERLIN FB 20, Automatisierung SYSTEMSIMULATION u. PROZESSLENKUNG	Fertigungssysteme aus verketteten, numerisch gesteuerten Werkzeugmaschinen	SIMULATION ZEITDISKRETER SYSTEME A. Reinhardt

Bild 1

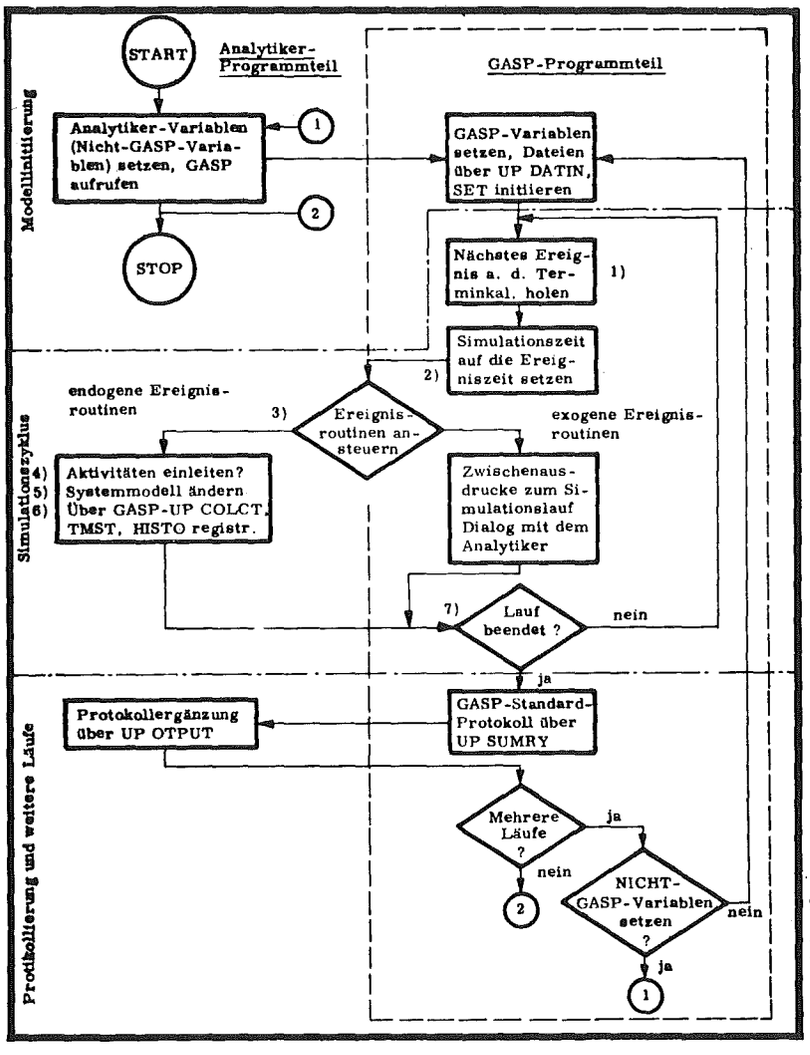
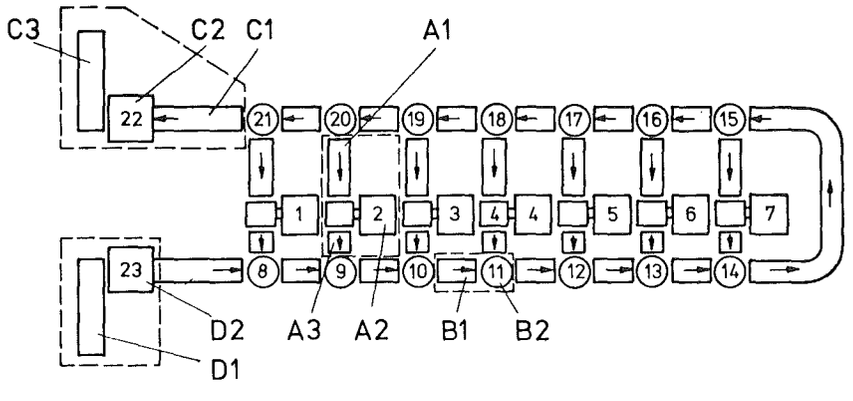
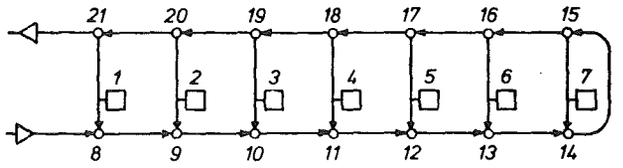


Bild 2:
Flußdiagramm zum Modellprozeß

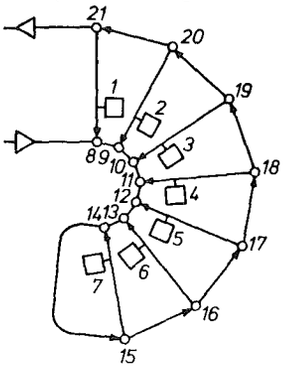


A1) Vorlagerstrecke
A2) Bearbeitungsstation
A3) Auslagerstrecke
B1) Förderstrecke
B2) Drehtisch
C1) Lagerstrecke
C2) Entspannvorrichtung
C3) Fertigteillager
D1) Rohteillager
D2) Aufspannvorrichtung

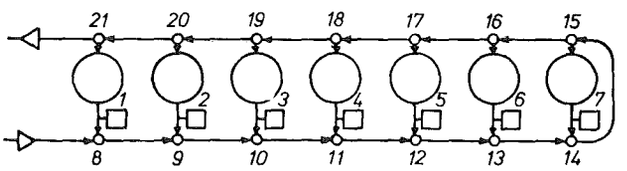
Bild 3: Systemmodell für NCMX = 7 Bearbeitungseinheiten, gegliedert in Teilsysteme und permanente Einheiten



System 1: Einbahnverkehr, Zugriff nach FCFS, gleiche Förderstreckenlängen in beiden Hauptförderrichtungen und in den Vorlagern



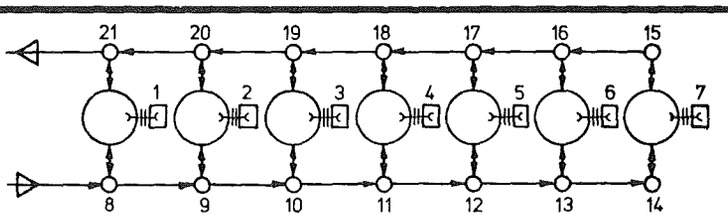
System 2: Einbahnverkehr, Werkstückzugriff nach FCFS belastungsabhängige Förderstrecken- und Vorlagerlängen



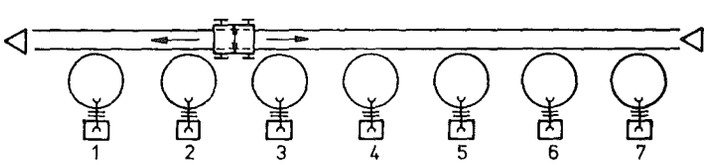
System 3: Einbahnverkehr, wahlfreier Werkstückzugriff, Vorlagerauslastung über Belegungsstrategien steuerbar

Bild 4:

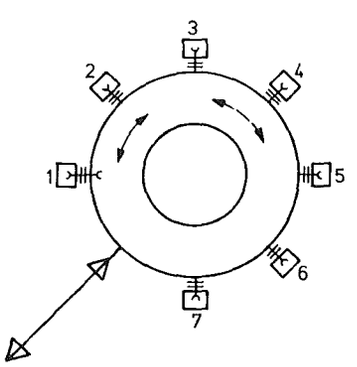
Strukturen verketteter Fertigungssysteme 1-3



System 4: Einbahnverkehr in Hauptförderrichtung, Transportwegverkürzungen als Folge richtungsabhängiger Ein- und Auslagerung in Vorlager, Zugriff wie im System 3



System 5: Liniensystem mit einer Hauptförderstrecke und Zweirichtungsverkehr durch Transportfahrzeug, Zugriff wie im System 3



System 6: Sternsystem, gestapelte Lagerringe mit entflochtenen Transportwegen, maschinenzugeordnete Transportsysteme, Werkstückzugriff wie im System 3

Bild 5:

Strukturen verketteter Fertigungssysteme 4-6

Dipl.-Math. W. Kleemann
ATH Duisburg

Das Simulationsprojekt Erzumschlag und Möllervorbereitung Schwelgern

Auf Grund von langfristigen Planungen kam man bei der ATH zu dem Entschluß, auf dem Gelände des Werkhafens Schwelgern einen Großraumhochofen zu errichten. Dort befanden sich schon verschiedene Anlagen zur Möllervorbereitung, z.B. eine Sinteranlage und eine Erzbrech- und -siebanlage (kurz EBUS-Anlage).

Um die Versorung des Hochofens mit geeignetem Möller sicherzustellen, sollten diese Anlagen erweitert werden. Anstelle einiger Erzlager sollten Erzmischbetten eingerichtet werden, damit die Schwankungen in der Erzanfuhr ausgeglichen werden könnten. Auf Stückerzmischbetten werden die eintreffenden Erze verschiedener chemischer Zusammensetzung mit geeigneten Zuschlagstoffen so vermischt, wie es für den Einsatz in den Hochofen erforderlich ist. Feinerze, Konzentrate und Zuschläge werden auf den Feinerzmischbetten zum Einsatz in die Sinteranlage gemischt. Die Planung nach konventioneller Art war schon so weit gediehen, daß auf Grund dessen eine neue EBUS-Anlage und eine neue Sinteranlage gebaut wurden. Zur Bewältigung der Entladung der größeren Erzmengen sollten zwei weitere Schiffsentlader angeschafft werden und schließlich müßte das vorhandene Transportbandsystem den veränderten Umständen angepaßt werden. Um zu prüfen, ob die geplante Auslegung den zu erwartenden Anforderungen entsprach, wurde beschlossen, eine Simulation auf einer Rechenanlage durchzuführen.

Hierfür mußten zunächst die erforderlichen Daten gesammelt werden. Die Leistungsdaten der bestehenden und geplanten Anlagen und Geräte wie Schiffsentlader, Transportbänder, EBUS-Anlage usw. waren relativ leicht zu beschaffen. Es stellte sich jedoch heraus, daß die vorhandenen Planungs- und Mittelwerte mit Sicherheitszuschlägen im allgemeinen nicht den Anforderungen genügten, die bei der Simulation auf dem Rechner zu stellen waren. So wurde z.B. vermutet, daß die Leistung der Schiffsentlader von verschiedenen Einflüssen wie Beschaffenheit der zu entladenen Erze, Anzahl der Kammern des zu entladenen Schiffs usw. abhängig seien. Es wurde deshalb notwendig, Zeit-Mengendiagramme der Schiffsentladung aufzunehmen. Gespräche mit Praktikern über die möglichen Einflußgrößen und math.-statistische Auswertungen zeigten, daß nur bei wenigen Parametern der Einfluß auf die Schiffsentladung so weit konkretisiert werden konnte, daß sie über Zufallsgeneratoren in der Simulation berücksichtigt werden konnten.

Die Beschaffung der Daten über Art, Menge und chemischer Zusammensetzung der Rohmaterialien wie Erze, Zuschlag- und Kreislaufstoffe machte mehr Schwierigkeiten. Die chemische Zusammensetzung der vollen Mischbetten muß nämlich mit geringen Toleranzen einer vorgegebenen Sollanalyse entsprechen. Die zur Verfügung stehenden Daten über Mengen und Arten der geplanten Erzlieferungen waren jedoch mehr nach langfristigen Einkaufs- und Seetransportplanungsgesichtspunkten ausgerichtet. Über Menge und Art der Zuschläge waren nur grobe Angaben vorhanden. Die für die Simulation geforderte Genauigkeit war in diesen Ausgangsdaten nicht enthalten.

Um nun die zur Verfügung stehenden Daten über die Rohstoffplanung mit den Forderungen des Hochofenbetriebs nach einer geeigneten chemischen Zusammensetzung des Möllers in Einklang zu bringen, wurde es notwendig, ein spezielles FORTRAN-Programm zu entwickeln, um die eigentliche Simulation nicht mit diesen Rechnungen zu belasten.

Eingabedaten für dieses Programm sind die Mengen und chemischen Zusammensetzungen der im Möller verwendeten Stoffe. Als Ergebnis erhält man die Durchschnittsanalyse an verschiedenen Punkten des Materialflusses. Durch wiederholte Korrekturen der Stoffmengen konnte die Jahresdurchschnittsanalyse mit Hilfe dieses Programms auf die Genauigkeit gebracht werden, die die metallurgische Praxis erfordert.
Das Programm leistet aber noch einiges darüber hinaus. So wird durch das Programm auch ermittelt, welche Erzsorten am besten miteinander vermischt werden sollten, damit die Mischung möglichst nahe bei der gewünschten Sollanalyse liegt. Danach könnte man die Anlieferung dann so planen, daß zusammenpassende Erze möglichst gleichzeitig eintreffen.

Anschließend konnte die Entwicklung des Simulationsmodells und die Programmierung desselben beginnen. Zunächst wurde entschieden, das Modell in GPSS/360 zu programmieren. Der Grund dafür war, daß eine IBM 360/50 (später eine IBM 370/155) zur Verfügung stand und daß schon vorher ein kleineres Problem mit GPSS bearbeitet worden war.

Nach einigen Simulationen von Teilbereichen, die dazu dienten, den Entladerhythmus der Schiffe und das Verhalten der Istanalyse der Mischbetten bei den gegebenen Schwankungen der Rohstoffanalysen zu überprüfen, wurde das Gesamtmodell entwickelt.

Eine Abschätzung des zur Verfügung stehenden Kernspeicherplatzes führte dazu, daß am Umfang des Gesamtmodells Abstriche gemacht werden mußten. Danach wurde in der Simulation nur der Bereich von der Schiffs-, Waggon- und LKW-Entladung über das Bandsystem und die EBUS-Anlagen bis zu den Feinerzmischbetten berücksichtigt. In diesem Bereich mußten 54 Geräte und Anlagen (Facilities) sowie 21 Bunker und Lager (Storages) simuliert werden. Ferner mußten die Daten über die anzuliefernden und gelieferten Mengen sowie über die chemische Zusammensetzung und über andere Materialeigenschaften von 34 Erzsorten und 21 anderen Stoffen in

einer Matrix entsprechenden Umfangs für die Simulation bereitgehalten werden. Es wurde ein Zeitraum von 17 Wochen in Zeiteinheiten von einer Minute simuliert.

Das Modell sieht in groben Zügen folgenden Ablauf vor:

Die Erze treffen per Schiff, die sonstigen Stoffe per Eisenbahn oder LKW ein. Die für die Ermittlung der Fahrpläne bzw. Zwischenankunftszeiten notwendigen Daten mußten aus den Angaben der Jahreserzplanung entnommen werden, die, wie schon erwähnt, mehr nach kaufmännischen Gesichtspunkten orientiert war.
Die Prüfung ob entladen werden darf, geschieht in zwei Teilen. Zunächst wird geprüft, ob das eingetroffene Material seiner chemischen Zusammensetzung nach auf ein im Aufbau befindliches Mischbett paßt. Danach wird geprüft, ob es einen freien Weg von der Entladung bis zu dem ausgewählten Mischbett gibt. Wenn nicht beide Bedingungen erfüllt sind, muß das Material warten.
Während der Wartezeit werden die beiden oben genannten Prüfungen ständig wiederholt, bis eine gewisse Zeit verstrichen ist, danach wird das Material auf Vorratslager entladen.

Sind jedoch beide Bedingungen erfüllt, so kann auf das Mischbett entladen werden. Die Stärke des Materialflusses von der Entladestelle bis zu den Feinerzmischbetten richtet sich nach dem schwächsten Glied in der Kette der auf diesem Weg liegenden Einrichtungen. Sobald eine eingetroffene Einheit, d.h. ein Schiff oder ein Eisenbahnzug usw. vollständig entladen ist, wird das betreffende Mischbett mengen-und analysenmäßig fortgeschrieben. Ist ein Mischbett voll geworden, so wird es bei Bedarf abgebaut, indem in regelmäßigen Zeitabständen (hier in 2-Stundenabschnitten) die von der Sinteranlage benötigte Menge vom Inhalt subtrahiert wird, bis es leer ist. Danach kann es wieder beschickt werden. Wartungszeiten für Gerät und Pausen sind im Modell mit berücksichtigt.

Als Ergebnis erhält man u.a. Angaben über die Auslastung der einzelnen Geräte und Anlagen und über die Ausnutzung der Kapazitäten der Pufferbunker und Lager aus denen man dann sehen kann, ob das System den Anforderungen gerecht wird. Außerdem erhält man detaillierte Angaben über den zeitlichen Ablauf des Auf- und Abbaus der Mischbetten und die dabei jeweils erzielten Analysen. Bei verschiedenen Tests zeigte es sich, daß mit Hilfe des Simulationsprogramms Feinheiten des Betriebsablaufs überprüft werden konnten.

Zum Austesten des Programms wurde eine Erzplanung benutzt, bei der die Richtigkeit der Zahlen nicht besonders geprüft wurde. Hierbei bildeten sich zunächst sehr große Warteschlangen von Schiffen, die unklassiertes Grubenerz transportierten. Da diese Erze über die EBUS-Anlagen laufen müssen, wo sie zu Fein- und Stückerz verarbeitet werden, wurde der Auslastungsgrad dieser Anlagen überprüft. Es stellte sich heraus, daß dieser Auslastungsgrad knapp unter 9o % lag, was viel zu hoch ist. Eine Nachrechnung ergab, daß unter den gemachten Voraussetzungen die Kapazitäten der beiden EBUS-Anlagen nicht ausreichten, um alles eintreffende unklassierte Grubenerz zu verarbeiten. Nachdem in der Simulation die Leistungszahlen der EBUS-Anlage auf das Doppelte erhöht worden war, ergaben sich realistischere Resultate.

In GPSS-Begriffen hat das Modell folgenden Umfang:

3oo aktive bewegliche Einheiten (Transactions)
17oo Befehle (Blocks)
6o Einrichtungen (Facilities)
4o Lager (Storages)
1o Warteschlangen (Queues)
2o Logische Schalter (Logic switches)
9o Tabellen (Tables)

25 Funktionen (Functions)
1oo Variable (Variables)
14 Matrizen (Matrix Savevalues)
14o Zwischenspeicher (Savevalues)

Der Bedarf an Kernspeicherplatz betrug 262 K und die Maschinenlaufzeit für einen Simulationslauf von 17 Wochen betrug 3 Stunden.

An zusätzlichen Möglichkeiten des GPSS wurden benutzt:

1) Die Möglichkeit der Neudefinition der Anzahl der zu benutzenden Programmiereinheiten wie Blöcke, Transaktionen usw. in GPSS-Ausdrucksweise "Reallocate".

2) Der Report-Generator des GPSS zur Aufbereitung der Ausgabedaten in eine auch für Außenstehende lesbare Form.

3) Die Speicherung des Simulationsprogramms mit Daten auf Magnetband zur Ermöglichung eines vereinfachten Änderungsdienstes, in GPSS-Ausdrucksweise "update".

4) Die Möglichkeit, zu gewissen Zeitpunkten des Maschinenlaufs den Gesamtzustand des Modells auf Platte zwischenzuspeichern, um bei Programmabbruch mit diesem Zustand neubeginnen zu können. Dies ist eine Art Checkpointmethode, in GPSS läuft sie unter dem Namen "Read-Save".

Diese Simulation war für alle Beteiligten außerhalb der Datenverarbeitungsabteilung die erste Maschinensimulation, mit der sie konfrontiert wurden. Aus diesem Grund war es nicht besonders vorteilhaft, daß dieses Simulationsprojekt einen so großen Umfang annahm. Es wäre günstiger gewesen, wenn zunächst einmal mehrere kleinere Projekte mit der Rechenanlage durchgespielt worden wären, damit alle Beteiligten einige Erfahrungen mit der computerunterstützten Simulation hätten sammeln können.

Die Einsatzmöglichkeit von Diffusionsmodellen für den Umweltschutz am konkreten Beispiel der Stadt New York

Dipl.-Volksw. Ch.A. Raehmel
IBM Deutschland GmbH
Anwendungsunterstützung in W u. V
53oo Bonn-Bad-Godesberg
Stephan-Lochner-Str. 2

Gliederung

1. Einleitung

Das PALO ALTO Scientific Center der IBM hat ein Diffusionsmodell zur Vorhersage von Schwefeldioxyd-Konzentration für ein Stadtgebiet entwickelt.
Das Modell besteht aus einer Anzahl mathematischer Gleichungen, die mit Hilfe eines Digitalrechners für gegebene Emissionsraten von Luftverschmutzungen unter gegebenen meteorologischen Bedingungen (hauptsächlich Windvektoren und atmosphärische Stabilitätsklassen) gelöst werden. Die Ergebnisse sind numerische Werte, die die SO_2-Konzentration an räumlichen Punkten im zu untersuchenden Gebiet angeben. Die Häufigkeitsverteilung der Schadstoffemission wird graphisch als ein System von Isolinien in eine Karte des Gebietes eingetragen.

Das Vermischen und Abnehmen der Schadstoffkonzentration wird im Diffusionsmodell der zufälligen Bewegung von Luftwirbeln und Turbulenzzonen in der Atmosphäre zugeschrieben, die ebenfalls die Schadstoffe verbreiten. Die Ausbreitungs- und Transportrate wird durch Windgeschwindigkeit, Temperatur und topographische Bedingungen bestimmt.
Die Benutzung eines solchen Modells erforderte Wetter- und Schadstoffdaten eines Echtzeit-Warnnetzes und einen vollständigen Emissionskataster innerhalb des betrachteten Gebietes.

Nach der Implementierung mußte das Modell eine gewisse Zeitspanne laufen zur Validierung, d.h. Vergleich der Emissionsverhersagen mit den nachfolgenden tatsächlichen Beobachtungen.

2. Das physikalische System des Modells

Drei Basiseingaben waren notwendig, um das mathematische Modell zur Simulation der Luftverschmutzungskonzentration aufzubauen:

die Luftverschmutzungsquellen
die Boden- oder Oberflächenstruktur
die Atmosphäre

Für das zu untersuchende Stadtgebiet wurden nur Punktquellen, wie Kraftwerke, Ölraffinerien, Verbrennungsanlagen und Flächenquellen, wie Wohngegenden, Industrieansiedlung im Diffusionsmodell berücksichtigt. Linienquellen, wie Autostraßen wurden nicht in das Modell aufgenommen.

Die Bodenstruktur ist für das Modell insofern wichtig, da drei Parameter sehr entscheidend sind:

die Höhe der Emissionsquellen
die Oberflächengestaltung
die topographischen Gegebenheiten

Die atmosphärischen Parameter im Modell schließen ein:

die Windgeschwindigkeit und Windrichtung
die vertikale Temperaturstrukturierung
die Turbulenzgegebenheiten der Partikel

3. Das Gaußsche Rauchfahnen-Modell

Da es im Augenblick unmöglich ist, exakt die atmosphärischen Prozesse detailliert zu simulieren, die eine Schadstoffausbreitung bewirken, hat man zur Voraussage der Luftverschmutzung mehrere Näherungslösungen entworfen. Eines der erfolgreichsten ist das Gaußsche Rauchfahnen-Modell. Das Modell basiert auf der Annahme, daß Vermischung und Verdünnung von Schadstoffpartikeln in der Luft vom atmosphärischen Zufallsprozeß der Turbulenzausbreitung abhängig ist, d.h. statistisch gesehen folgt die Ausbreitung von Partikel einer emittierenden Quelle einer mathematischen Ausbreitungsfunktion. Die Ausbreitungsrate, mit der die Teilchen transportiert werden, sind prinzipiell von der durchschnittlichen Windgeschwindigkeit, dem vertikalen Temperaturgradienten und den örtlichen topographischen Gegebenheiten abhängig.

Im Aufbau des Gaußschen Rauchfahnen-Diffusionsmodells wurde angenommen, daß die durchschnittliche Schadstoff-Konzentration von einer Punktquelle ausgehend in Querwindrichtung einer Gauß-Verteilung entspricht. Der Transport der Schadstoffe in Windrichtung führt zu einer Abnahme der Konzentration in dieser Richtung.
Abbildung 1 zeigt die durchschnittliche Ausbreitungsverteilung einer gasförmigen Verschmutzung einer erhöhten Punktquelle.

Die Basisgleichung lautet:

$$\chi = \frac{Q}{\pi . \bar{U} . \sigma_y . \sigma_z} \exp. -\frac{1}{2} \left(\frac{Y^2}{\sigma_y^2} + \frac{H^2}{\sigma_z^2} \right)$$

wobei

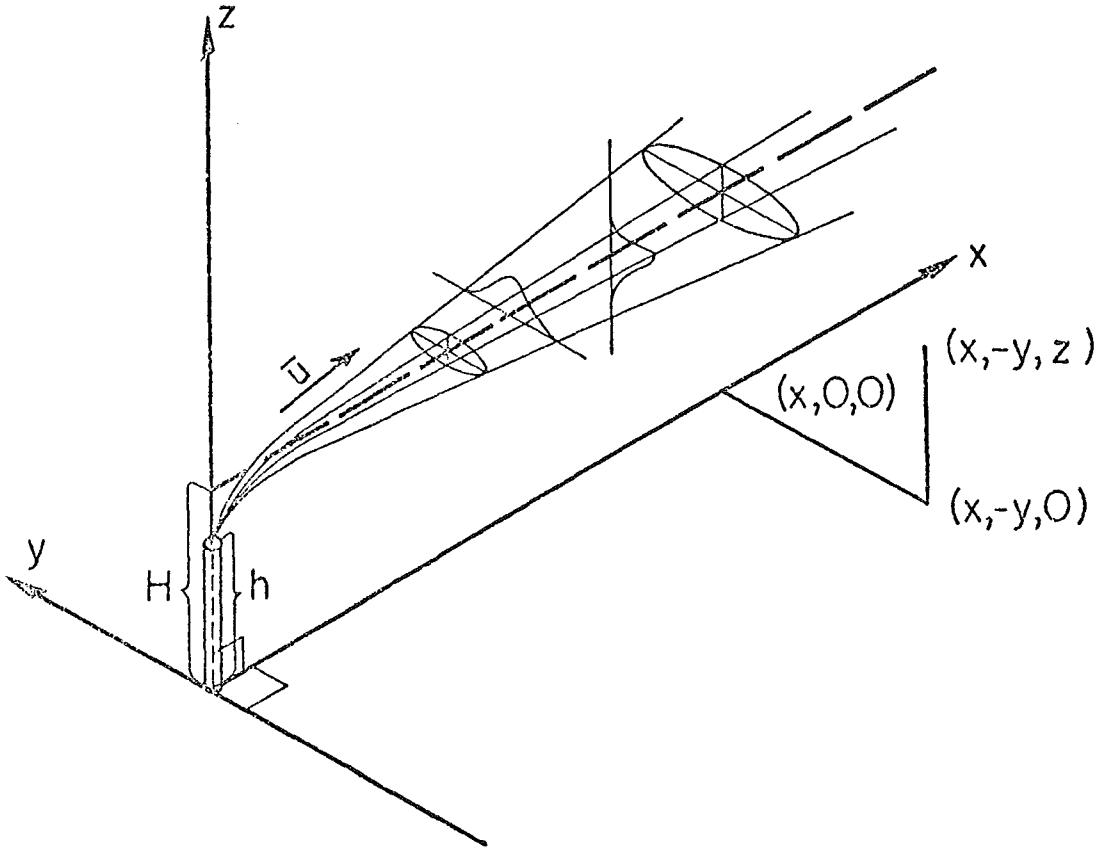
χ (x,y,o)	= Oberflächenkonzentration
Q	= Emissionsrate der Punktquelle
H	= Schornsteinhöhe
U	= durchschnittliche Windgeschwindigkeit
x	= Achse des Windes in Windrichtung
y	= Achse des Querwindes
$\sigma_y(x)$	= horizontale Ausbreitungskoeffizienten
$\sigma_z(x)$	= vertikale Ausbreitungskoeffizienten

Mathematisch gesehen repräsentieren die σ_y - und σ_z-Parameter in der Gleichung die Standardabweichung der Gauß-Kurve. Physikalisch bestimmen diese Parameter die Rate, mit der sich die Rauchfahne als Funktion der Bewegung in Windrichtung ausbreitet.

Ferner wurde angenommen, daß die Oberläche vollständig die Rauchfahne aufnimmt. Diese Gleichung gilt nur für stabile Verschmutzungsteile wie SO_2 und CO.

GAUSS-VERTEILTES RAUCHFAHNEN-DIFFUSIONSMODELL

GAUSS-DARSTELLUNG EINER KONTINUIERLICHEN PUNKTQUELLE



$$\frac{\chi \bar{u}}{Q} = \frac{1}{\pi \sigma_y \sigma_z} \exp\left\{-\frac{1}{2}\left[\left(\frac{y}{\sigma_y}\right)^2 + \left(\frac{H}{\sigma_z}\right)^2\right]\right\}$$

Abbildung 1

Die Parameter σ_y und σ_z der Gleichung sind experimentellen Beobachtungen entnommen worden. Eine praktische Begrenzung der Gleichung war: keine räumliche Veränderung von durchschnittlicher Windgeschwindigkeit und Windrichtung. Entscheidend ist die Gegebenheit eines repräsentativen durchschnittlichen Windfeldes für das zu untersuchende Stadtgebiet.

Es erwies sich für nicht sehr praktikabel, innerhalb des New Yorker Stadtgebietes für jede Quelle eine partielle Betrachtung durchzuführen. Man schloß alle diese Punktquellen zu einer Flächenquelle zusammen mit einer gleichen Emissionsrate. Im Modell wurde die integrierte Rauchfahnen-Formel angenommen.

Die Gleichung in Abbildung 2 sagt aus, daß die Gesamtkonzentration von Flächenquellen das Ergebnis der Summe des Schadstoffbeitrages aller in einem Gebiet zusammengefaßten Punktquellen darstellt.

$$\chi_{TOTAL}\ (X,Y) = \sum_n \chi_n\ (\Delta_n A, X, Y)$$

wobei

H = durchschnittliche Höhe der Flächenquellen

χ_n = Beitrag zur Schadstoffkonzentration jeder Punktquelle

$\Delta_n A$ = Fläche der Punktquellen

4. Die Grundzüge des IBM PALO ALTO Diffusionsmodells

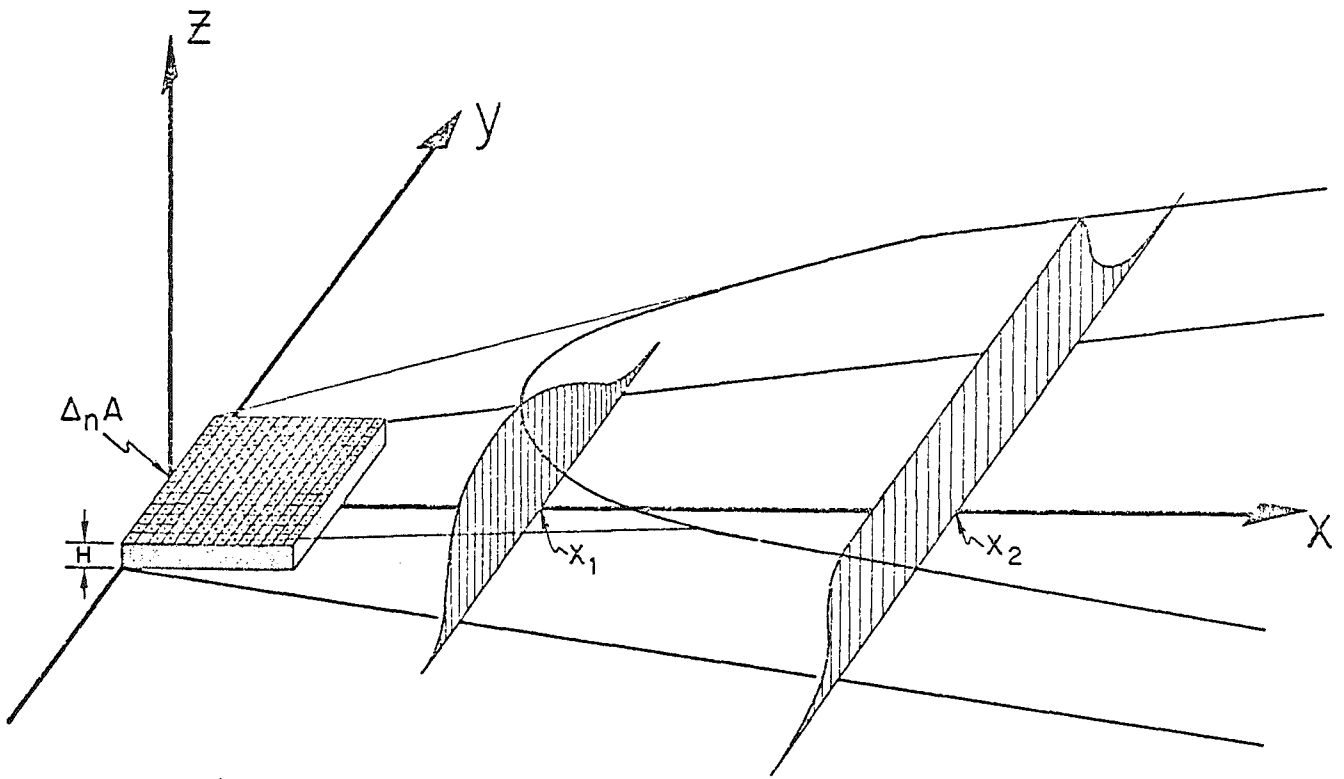
Das Computer-Programm berechnet die Beiträge der Luftverschmutzung von Punkt- und Flächenquellen unabhängig voneinander. Die Voraussage der SO_2-Verschmutzung umfaßte ein Gebiet von 1600 Quadratmeilen im Stadtgebiet von New York.
Inversionswetterlagen, wie sie jederzeit auftreten können, gehen in das Programm als Variable ein. Die Diffusionskoeffizienten σ_y und σ_z wurden als Funktionen der atmosphärischen Stabilität, der Windrichtungsentfernung und des Terrains angesehen. Hier hatten Meteorologen wichtige Vorarbeit geleistet.
Dem Programm wurden die von Pasquill [1)] definierten und durch neuere Untersuchungen verbesserten sechs Stabilitätsklassen zugrunde gelegt (Abbildung 3).

---

1) Pasquill, F.; The estimation of the dispersion of windborne material,
Meteorological Magazine, 90, 1961, Seiten 33-49.

Abbildung 2

DIFFUSION EINER RAUCHFAHNE AUSGEHEND VON EINER FLAECHENQUELLE



$$\chi_{TOTAL}(x,y) = \sum_n \chi_n(\Delta_n A, x, y)$$

ATMOSPHAERISCHE STABILITAETSKLASSEN

SCHLUESSEL ZU DEN STABILITAETSKLASSEN

OBERFLAECHEN-WINDGESCHWIND. M SEC^{-1}	AM TAGE SONNENEINSTRAHLUNG STARK	AM TAGE SONNENEINSTRAHLUNG GEMAESSIGT	AM TAGE SONNENEINSTRAHLUNG LEICHT	BEI NACHT LEICHT BEWOELKT	BEI NACHT BEWOELKT
< 2	A	A-B	B		
2-3	A-B	B	C	E	F
3-5	B	B-C	C	D	E
5-6	C	C-D	D	D	D
> 6	C	D	D	D	D

Die neutrale Klasse D sollte für Bewölkungen während des Tages und bei Nacht angenommen werden

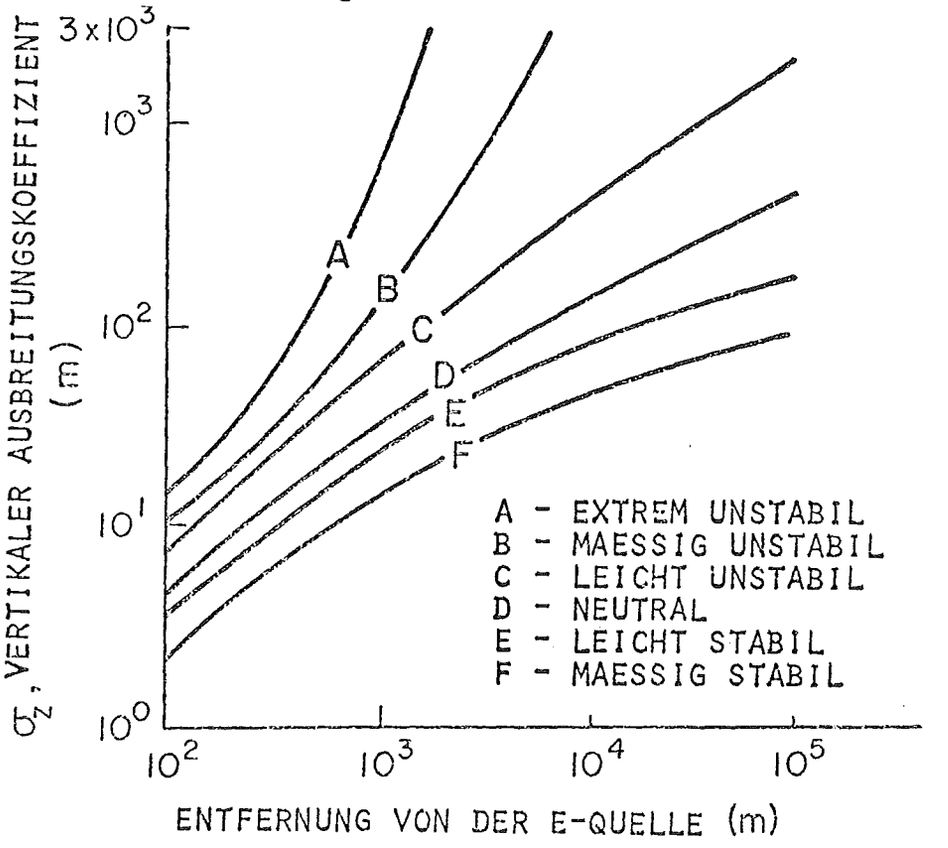


Abbildung 3

5. Die Computer-Procedure

Das Computer-Programm ist in FORTRAN IV (H) geschrieben und erfordert 2oo K. Das Hauptprogramm besteht aus vier Teilen:

1. Analyse der Daten aus dem Emissionskataster
2. Analyse der meteorologischen Daten
3. Berechnung der Oberflächenverschmutzungs-Konzzentration
4. Graphische Ausgabe der berechneten Werte

Die Analyse der Daten aus dem Emissionskataster schließt neben den Flächen- und Punktquellendaten auch deren geographische Position ein, die in einem Koordinatensystem festgehalten werden. Dieses System wird im Modell als ein rechteckiges Koordinatensystem definiert. Die x-Achse repräsentiert die Ost-West-Richtung und die y-Achse die Nord-Süd-Richtung.
Die tägliche Emissionsrate von Flächenquellen wurde nach folgender Formel bestimmt:

$$Q_d = \frac{\gamma_1 \cdot Q_a}{365} + \frac{\gamma_2 \cdot DD \cdot Q_a}{TDD}$$

Q_d = tägliche Emissionsrate von SO_2
Q_a = jährliche Emissionsrate von SO_2
DD = Differenz von 65^o F und der täglichen Durchschnittstemperatur, wenn letztere 65^o oder niedriger ist; sonst ist DD=O
TDD = klimatologischer Durchschnittswert eines Jahrestages
γ_1, γ_2 = Proportionalitätsfaktoren für Wassererhitzung und Zentralheizung. $\gamma_1 + \gamma_2 = 1$

Das Computer-Programm berechnet Zwei-Stunden-Durchschnittsemissionen für Punkt- und Flächenquellen. Die Berechnung wurde für die Zeitspanne von o bis 24 Uhr durchgeführt.

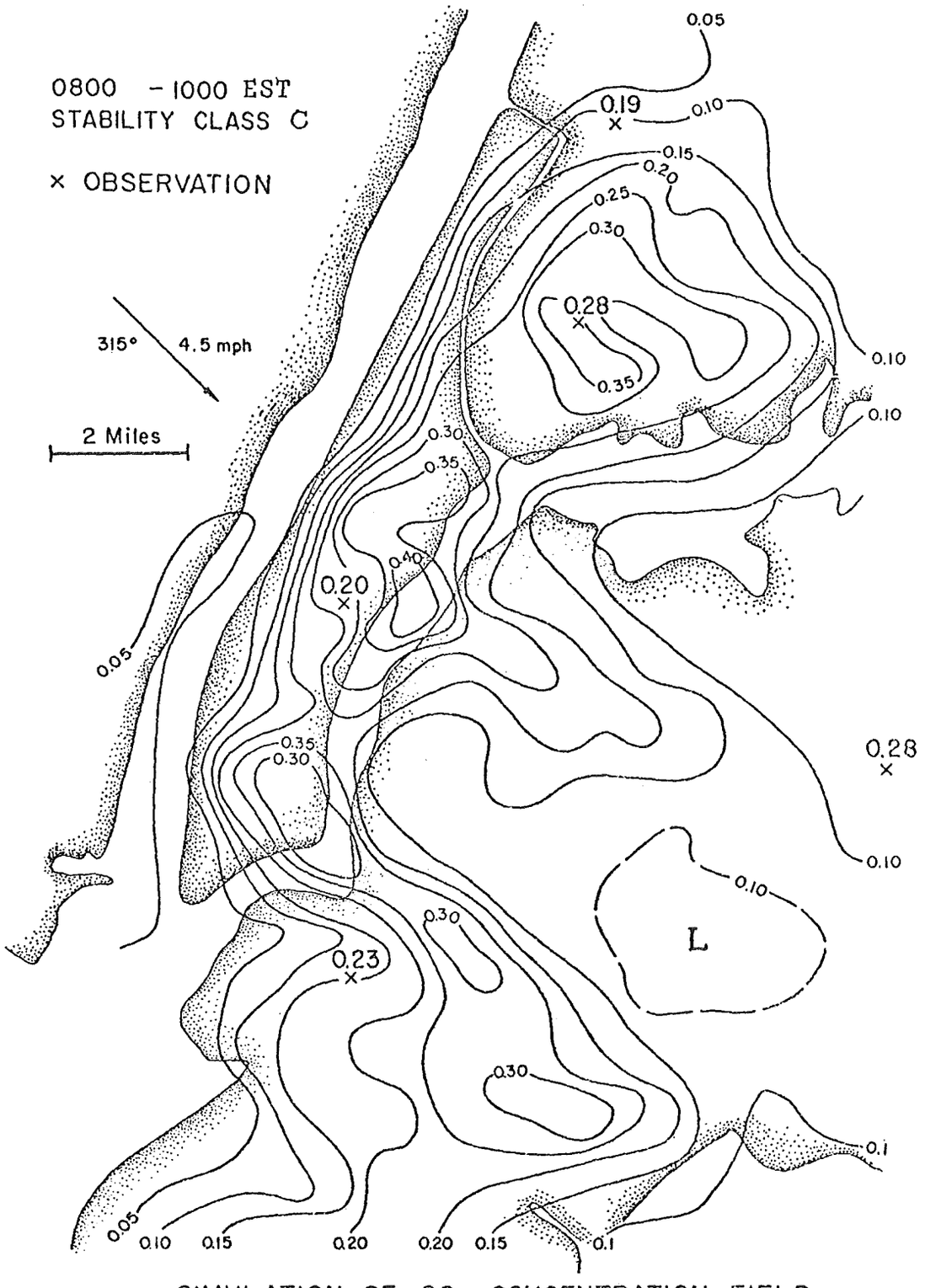
Die Ausführungszeit betrug für eine Zwei-Stunden-Simulation mit 5oo Flächenquellen auf einem System /36o Modell 91 15 Sekunden und auf einem System /37o Modell 145 5 Minuten.

Die ermittelten Werte wurden über einen IBM 225o-Bildschirm herausgegeben.

6. Die Ergebnisse, dargestellt im IBM Diffusionsmodellfilm (16 mm)

Die gewonnenen Ergebnisse wurden als ein System von Isolinien dargestellt, um die Häufigkeitsverteilung der Schadstoffkonzentration graphisch zu verdeutlichen. Die Simulationsergebnisse wurden anschließend mit tatsächlichen Schadstoffmessungen verglichen. Es stellte sich heraus, daß die Abweichung der vorausgesagten Ergebnisse von den aktuell gemessenen nur bis 15 % betrug.

Ein hervorragendes Ergebnis!



SIMULATION OF SO_2 CONCENTRATION FIELD
JAN. 11, 1971

Abbildung 4

7. Literaturhinweis

Shieh, L.J.; Halpern, P.K.; Clemens, B.A.; Wang, H.H.; Abraham, F.
Air Quality Diffusion Model: Application to New York City,
in IBM Journal of Research and Development,
Vol. 16, No. 2, März 1972.

Abraham, F.F.; Halpern, P.K.; Shieh, L.J.; Wang, H.H.
An Introduction to Air Quality Diffusion Modelling,
in IBM Data Processing Division, Palo Alto Scientific Center,
No. 32o-3298, Juni 1972.

Test- und Meßeinrichtungen für ein Simulationssystem zur Untersuchung des dynamischen Verhaltens von Betriebssystemsoftware

von Jürgen Nehmer

Kurzfassung

Messungen des dynamischen Verhaltens von Betriebssystemsoftware werden im IDT an einer virtuellen PL/1-Maschine (PLIM) durchgeführt, deren Instruktionsvorrat grob in zwei Klassen zerfällt: die Maschineninstruktionen und die Pseudoinstruktionen. Der Bericht beschreibt nach einer übersichtsmäßigen Einführung in das Simulationssystem die Wirkungsweise der Pseudoinstruktionen, die einerseits das Austesten (Debugging) der modellierten Software unterstützen und andererseits die Simulation und Messung von Programmlaufzeiten ermöglichen.

1. Einführung

Bei der Entwicklung von Betriebssoftware sind Systemingenieure häufig vor die Aufgabe gestellt, aus einer Reihe sich anbietender Alternativen die geeignete Strategie, den optimalen Algorithmus oder die passende Datenstruktur auszuwählen. Die Entscheidungen sind dabei in der Regel nicht trivial, da sie oftmals Kenntnisse über das dynamische Verhalten des betrachteten Systementwurfs voraussetzen. Die Erfahrungen zeigen, daß selbst simple Systeme in ihrem dynamischen Ablauf nur schwer überschaubar sind und intuitiv abgeleitete Vorhersagen in hohem Maße spekulativ sind.

Für die quantitative Beurteilung der dynamischen Eigenschaften von Betriebssystemsoftware wurde daher im IDT ein Simulationsinstrument (PLIM = virtuelle PL/1-Maschine) entwikkelt, das sich als nützliches Hilfsmittel für die Modellierung und Analyse von Betriebssoftware erwies. Im ersten Teil

des Berichtes werden die grundlegenden Konzepte von PLIM vorgestellt.

Der zweite Teil konzentriert sich ausschließlich auf die Simulations-, Test- und Meßeinrichtungen von PLIM, die ein universelles Hilfsmittel für

- das Austesten (Debugging) des Modells,
- die Messung der Ausführungszeiten beliebiger Instruktionssequenzen,
- die Messung von Verzögerungszeiten,
- die Anfertigung von "Schnappschüssen",
- die Modellierung der Subsystem-Umgebung

darstellen.

2. Der organisatorische Aufbau des Simulationssystems

Der in PLIM realisierten Lösung liegt die Idee zugrunde, PL/1-Statements als Maschineninstruktionen einer virtuellen PL/1-Maschine aufzufassen. Sie habe die Fähigkeit, PL/1-Programme auf der Basis von einzelnen Statements zugeordneten Zeitbewertungen statementweise, zeitkontrolliert auszuführen. Eine derart modifizierte PL/1-Programmfolge hat z.B. folgendes Aussehen:

```
          .
          .
          .
$3        X = A+B;
$4        IF X < 10 THEN GOTO WEITER;
          Z = X (Y+25);
          PUT SKIP LIST (Z);
$2        COUNT = COUNT+X;
          .
          .
          .
```

Die mit einem führenden $-Zeichen angeführten "Zeitstempel" stellen dabei in einem abstrakten Sinne Synchronisationspunkte dar:

Die in ganzen Vielfachen einer Grundtaktzeit angegebenen Ausführungszeiten werden zunächst mit den übrigen Aktivitäten des Systems von einer zentralen Eventsteuerung koordiniert. Die eigentliche Ausführung eines zeitgewichteten PL/1-Statements erfolgt jeweils nach Ablauf der angegebenen Zeitbewertung.

Die Statementfolge zwischen zwei Synchronisationspunkten ist - äquivalent den Maschineninstruktionen realer Maschinen - unteilbar und wird in einem Zuge ausgeführt.

Abb.1 zeigt den organisatorischen Aufbau des gesamten Simulationssystems. In der oberen Hälfte ist die Kontrollstruktur dargestellt, die in drei Ebenen untergliedert ist.
Wir unterscheiden

- die Eventsteuerung,
- die Ablaufsteuerung der PL/1-Maschine,
- das Modell.

Event- und Maschinensteuerung zusammen bilden die virtuelle PL/1-Maschine (kurz: PLIM). Das Modell besteht aus dem zu simulierenden Programmsystem und ist aus einer Kollektion von PL/1-Prozeduren (Modellprogramme) zusammengesetzt, an die hinsichtlich des Aufbaus hier nicht näher spezifizierte Anforderungen gestellt werden [1,2].

Im unteren Teil der Abbildung ist der Erstellungsprozeß für Modellprogramme dargestellt.

Der eigentliche Quellcode wird zunächst durch einen speziellen Makroprozessor modifiziert und dann durch den standardmäßigen PL/1-Optimizing Compiler übersetzt.

Eine Besonderheit des hier realisierten Verfahrens ist, daß Modellprogramme entgegen der sonst üblichen Technik nicht interpretiert, sondern statementweise von der PL/1-Maschine durchlaufen werden. Der dadurch notwendige Kontrollflußwechsel zwischen PL/1-Maschine und Modellprogrammen, der

nach jedem zeitbewerteten PL/1-Statement stattfindet, erfordert eine entsprechende Aufbereitung des Quellcodes der Modellprogramme, die durch den Makroprozessor vorgenommen wird.

Alle Modellprozeduren sind auf PL/1-Ebene reentrant. Als Folge einer Prozeduraktivierung wird ihnen ein Initialisierungsbereich zugeordnet, der den augenblicklichen Befehlszähler (Label des nächst auszuführenden PL/1-Statements), die Rückwärtsverkettung zur aufrufenden Prozedur, Pointer auf die Arbeitsbereiche (BASED-Strukturen) sowie die aufgelaufenen Ausführungszeiten aufnimmt. Die begleitenden Speicherallokationen und -freigaben, Normierungen und schritthaltenden Updatungen der Initialisierungsbereiche werden automatisch durchgeführt und sind damit Bestandteil der PLIM-Hardware. Auf die Verwaltung der Ausführungszeiten in den Initialisierungsbereichen wird später noch einmal ausführlicher eingegangen, da sie den Schlüssel für die Messung von Programmlaufzeiten bilden.

Der Instruktionsvorrat für die PL/1-Maschine setzt sich neben den durch die PL/1-Konventionen festgelegten Standardoperationen aus Systeminstruktionen und Pseudoinstruktionen zusammen.

Systeminstruktionen bilden das notwendige Instrument zur Zustandskontrolle der virtuellen Maschine. Siebzehn derartige Instruktionen sind gegenwärtig implementiert, die über vereinbarte Prozeduraufrufe von allen Modellprogrammen benutzbar sind:

$CALL(Programmname)
$RETURN
$JUMP(Programmname)
$RESUME(Programmstatus)
$SAVE(Programmstatus)
$LOAD_PARMADDR(Parameteradresse)
$STORE_PARMADDR(Parameteradresse)

$INT_STATUS(Ablageadresse)
$ENABLE
$DISABLE
$MASK
$UNMASK
$LOCK(Lockadresse)
$UNLOCK(Lockadresse)
$SIGNAL(Prozessor, Priorität)
$IDLE
$SVC(SVC-Nummer)

Desweiteren stehen 7 Pseudoinstruktionen zur Verfügung, die ausschließlich Simulations-, Test- und Meßzwecken dienen:

$CLOCK
$CPU_TIME
$RUN(Zeit)
$INTERRUPT(Zeit, Interrupt-Nr., Priorität)
$TEST(Modus)
$STOP
$PR_DUMP

Ihre Wirkungsweise wird im nachfolgenden Kapitel beschrieben und der Gebrauch an einigen Beispielen demonstriert.

3. Wirkungsweise und Anwendung der Pseudoinstruktionen

In Abb.2 ist der prinzipielle Aufbau eines Modells für die PLIM dargestellt, das in drei Teile zerfällt:

- das Meßobjekt,
- die Meßobjektumgebung,
- den Monitor.

Dem Monitorprogramm wird in der Regel ein Prozessor fest zugeordnet, der nicht Teil der Meßkonfiguration ist und dadurch eine asynchrone, rückwirkungsfreie Beobachtung des

eigentlichen Meßobjektes erlaubt.

Für die Zeitmessung stehen zwei Pseudoinstruktionen zur Verfügung:

- $CLOCK

  Der Aufruf dieser Pseudoinstruktion liefert die absolute Simulationszeit in ganzen Vielfachen eines Grundtaktes, der frei wählbar ist.

  In dem nachfolgenden Beispiel soll, abhängig von der absoluten Zeit, eine vorgegebene Aktion ausgelöst werden:

```
      .
      .
      .
      IF $CLOCK >= ZEITSCHRANKE THEN DO;
                                        .
                                        .
                                        .
                                      END;
                                 ELSE;
      .
      .
      .
```

- $CPU_TIME

  Der Aufruf dieser Pseudoinstruktion liefert die Ausführungszeit einer Prozedur in ganzen Vielfachen der Grundtaktzeit. Dabei gelten folgende Regeln:

  - die Ausführungszeit, die im Initialisierungsbereich einer Prozedur abgelegt wird, erhält beim Anlegen des Initialisierungsbereiches (erste Aktivierung) den Wert 0,
  - bei Sprüngen ($JUMP) und Unterprogrammsprüngen ($CALL) wird die aufgelaufene Ausführungszeit an das aufgerufene Programm übertragen und im Initialisierungsbereich abgelegt.

- bei Rücksprüngen ($RETURN) wird die aufgelaufene Zeit an die aufrufende Prozedur zurückgereicht und ersetzt dort den bisherigen Wert.

Da das Updaten der Ausführungszeit schritthaltend nach jedem Statement erfolgt, haben Unterbrechungen der Prozeduren, die eine Rückstellung des zugehörigen Prozesses verursachen, keinen Einfluß auf den Meßvorgang.

In dem nachfolgenden Beispiel soll unter Zuhilfenahme dieser Zeitbasis die mittlere Ausführungszeit in einer Schleife gemessen werden. Das entsprechende PLIM-Programm könnte dann etwa folgendes Aussehen haben:

```
          .
          .
          .

          SUMME = SUMME+1;
$7        GET(N);
          STARTZEIT=$CPU_TIME;
          LABEL:;
$3        IF N  =0 THEN DO;
                    .
                    .
                    .
$3                  N=N-1;
$2                  GOTO LABEL;
                         END;
          ENDZEIT = $CPU_TIME;
          ZEITSUMME = ZEITSUMME+(ENDZEIT-STARTZEIT);

          .
          .
          .
```

- $RUN(Zeit)

  Diese Pseudoinstruktion versetzt den ausführenden Prozessor für die angegebene Ausführungszeit in einen PSEUDO-RUNNING-Zustand. Er wird dort ähnlich behandelt als führe er Instruktionen aus. Wesentlicher Unterschied zu einer normalen Maschineninstruktion ist, daß der Prozessor innerhalb einer PSEUDORUNNING-Phase potentiell unterbrechbar ist.

Im Falle einer Unterbrechung (Voraussetzung: enabled) wird die aufgelaufene Ausführungszeit aufakkumuliert und die Restzeit des angebrochenen PSEUDORUNNING-Intervalls im Initialisierungsbereich der Prozedur für einen späteren Restart abgespeichert.

Die $RUN-Instruktion bietet damit die Möglichkeit einer Zwei-Stadien-Simulation:

- Der Modellkern wird in der Regel mikroskopisch dargestellt, alle Operationen sind algorithmisch aufgelöst.
- Die Umgebung des eigentlichen Meßobjekts wird überwiegend makroskopisch nachgebildet, in dem die typischen Ausführungszeiten von Programmen durch - z.B. statistisch verteilte - Zeitstrecken mit Hilfe der $RUN-Instruktion angenähert werden.

Die Simulation von Benutzerprozessen im Rahmen einer Betriebsorganisation kann z.B. durch den Wechsel von Aktiv- und Wartephasen nach folgendem Muster geschehen:

```
        .
        .
        .
        ANF:;
        AKTIVPERIODE = CALL EXPON_Z; /+ Zufallszahl +/
        $RUN(AKTIVPERIODE);
        $SVC(23); /+ E/A-Wunsch +/
$O      GOTO ANF;
        .
        .
        .
```

Nach Annahme des SVC's kann durch eine unabhängige Prozedur die Wartephase simuliert werden, nach deren Ablauf die Kontrolle an den Benutzerprozeß zurückgeht.

- $INTERRUPT(Zeit, Interrupt-Nr., Priorität)

Diese Pseudoinstruktion dient der Simulation aller Zeitvorgänge, die außerhalb der Prozessoren ablaufen und von ihnen lediglich initialisiert werden wie alle Ein/Ausgabe-Vorgänge.

Der SVC des vorherigen Beispiels könnte z.B. ein Ein/Ausgabe-Treiberprogramm anwerfen, das in seinem Kern folgende Instruktionen ausführt:

```
        .
        .
        .
$4      IF GERAET = 'FREI' THEN DO;
$5                         $INTERRUPT(EXPON_Z,7,2);
$0                               END;
                           ELSE DO;
$100                       CALL ENQUEUE(REQUEST);
                                 END;
        .
        .
        .
```

- $TEST(Modus)

Die $TEST-Pseudoinstruktion dient der Einstellung zweier Simulationsmodi: dem Produktionsmodus ('P'-Modus) oder Test-Modus ('T'-Modus). Im Testmodus werden alle Instruktionen unter Angabe des absoluten Ausführungszeitpunktes, des ausführenden Prozessors und des Namens der Prozedur ausgedruckt. Er bildet ein unentbehrliches Hilfsmittel beim Austesten eines Modells.

Das nachfolgende Beispiel, das als Teil des Monitors aufgefaßt werden kann (Abb.2), zeigt wie einmalige und periodisch wiederkehrende Testphasen in einen Produktionslauf eingeblendet werden können:

```
        .
        .
        .
PUT SKIP LIST('ENDZEIT?  :');
GET LIST(ENDZEIT);
$INT(ENDZEIT,4,1);
PUT SKIP LIST('TEST ?:');
GET LIST(MODUS);
```

```
        IF MODUS = 'T' THEN DO;
           PUT SKIP LIST('PERIODIC?:');
           GET LIST(ART);
           IF ART = 'PERIODIC' THEN DO;
              PUT SKIP LIST('GIB:PERIODE,TESTPHASE:');
              GET LIST(PERIODE, TESTPHASE);
              $INT(0,5,1,);
                                        END;
                                     ELSE DO;
              PUT SKIP LIST('GIB:STARTZEIT, TESTPHASE:');
              GET LIST(STARTZEIT, TESTPHASE);
              $INT(STARTZEIT,5,1);
                                           END;
                                  END;
        $IDLE;
$O;     .
        .
        .
        LAB(4):; /+ Simulationsende +/
$O      $STOP;
$O      LAB(5):; /+ Beginn einer Testphase +/
        $INT(TESTPHASE,6,1);
        IF ART = 'PERIODIC' THEN DO;
           $INT(PERIODE,5,1);
                                  END;
        $TEST('T');
        $IDLE;
$O;
        LAB(6):; /+ Ende der Testphase +/
        $TEST('P');
        $IDLE;

        .
        .
        .
```

In dem obigen Beispiel wurde auch die Anwendung einer weiteren Pseudoinstruktion demonstriert:

- $STOP

  Sie stellt ein Hilfsmittel zum kontrollierten Abbruch eines Simulationslaufs dar, in dem sie alle Events in der Eventsteuerung löscht.

- $PR_DUMP

  Diese letzte der aufgezählten Pseudoinstruktionen ermöglicht die Ausgabe der Statusinhalte aller Prozessoren des

Systems zu jedem beliebigen Zeitpunkt und stellt ebenfalls eine wertvolle Hilfe beim Austesten der Modelle dar.

Zusammenfassung

An einigen Beispielen aus dem Bereich der Simulation von Betriebssoftware wurde gezeigt, daß mit einem beschränkten Satz von Simulations-Primitives ein weites Spektrum von Anwendungen überdeckt werden kann. Der Aufwand für die Implementierung dieser Primitives ist vergleichsweise niedrig und deshalb oft einer integrierten Lösung überlegen (in Simulationssprachen wie z.B. GPSS), die in der Regel einen hohen Aufwand erfordert und die Simulationseffizienz mindert.

Literaturhinweise

[1] Fleck, G., Nehmer, J.
Simulation von Betriebssoftware auf einer virtuellen PL/1-Maschine
Lecture Notes in Economics and Mathematical Systems 78
Springer Verlag, Berlin-Heidelberg-New York, 1972
S. 253-262

[2] Rupp, M., Nehmer, J., Fleck, G., Hilse, D., Friehmelt, R.
PLIM - Eine virtuelle PL/1-Maschine
KFK 1712, Kernforschungszentrum Karlsruhe, Mai 1973

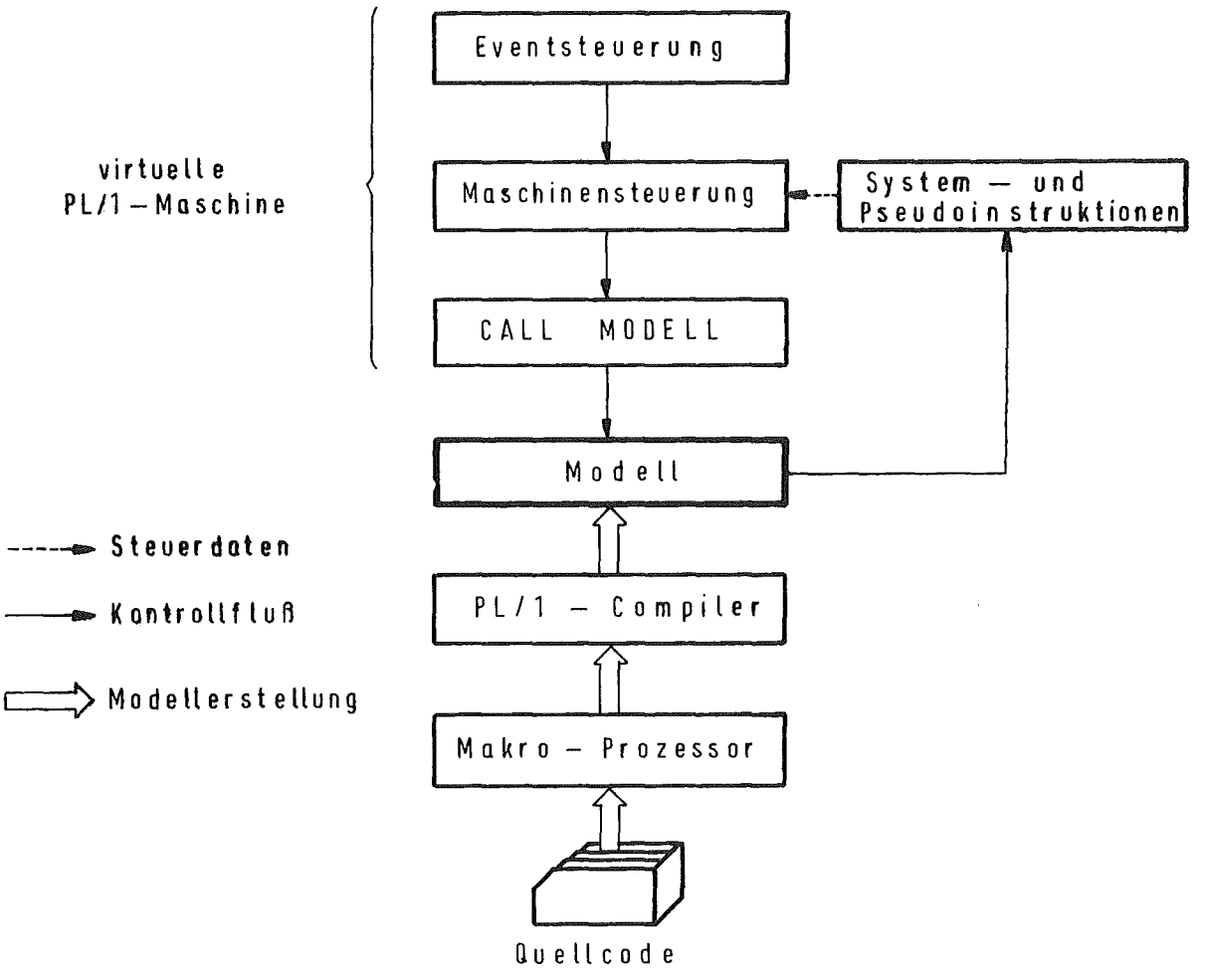


Abb.1 Organisation des Simulationssystems

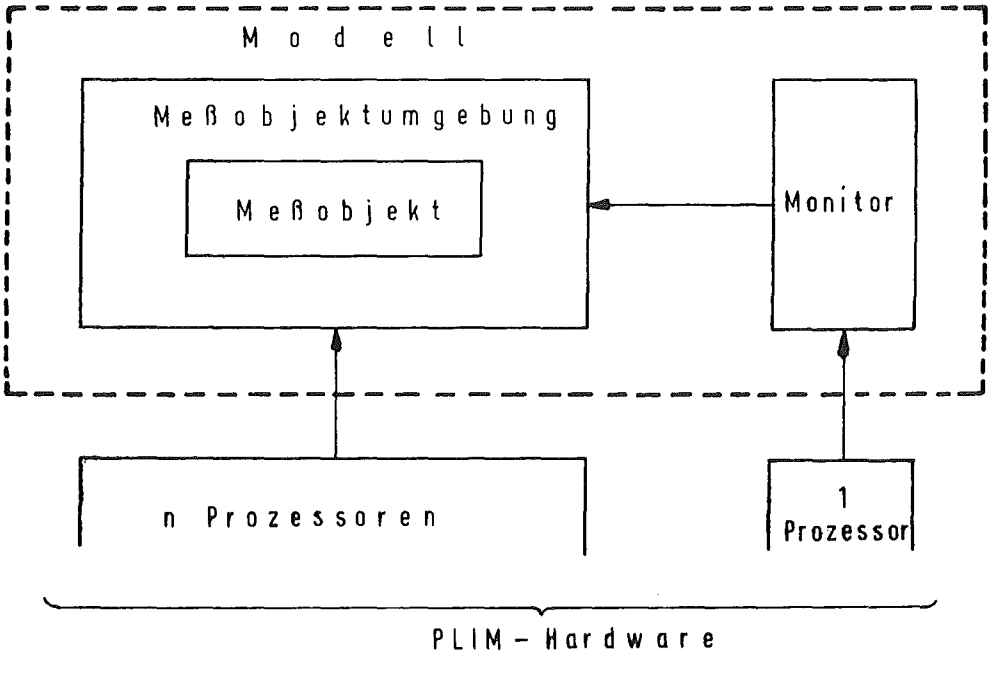


Abb.2 Prinzipielle Organisation eines Modells unter PLIM

Modellbildung im Rahmen der Simulation von Teilnehmersystemen

R. Isernhagen
Philips Forschungslaboratorium Hamburg GmbH, 2 Hamburg 54

1. Einführung

Das Ziel der Arbeiten über die ich hier berichte, besteht darin, verschiedene Konzepte und Konfigurationsalternativen von Terminalsystemen an Hand von vereinfachten Modellen zu untersuchen.

Im Bild 1 sehen Sie ein Beispiel für ein solches Terminalsystem, ein sog. Datensammelsystem. Die gezeigte Konfiguration setzt sich aus verschiedenen Typen von Modulen zusammen: Geräte, Adapter und Rechnerkerne (CPU's).

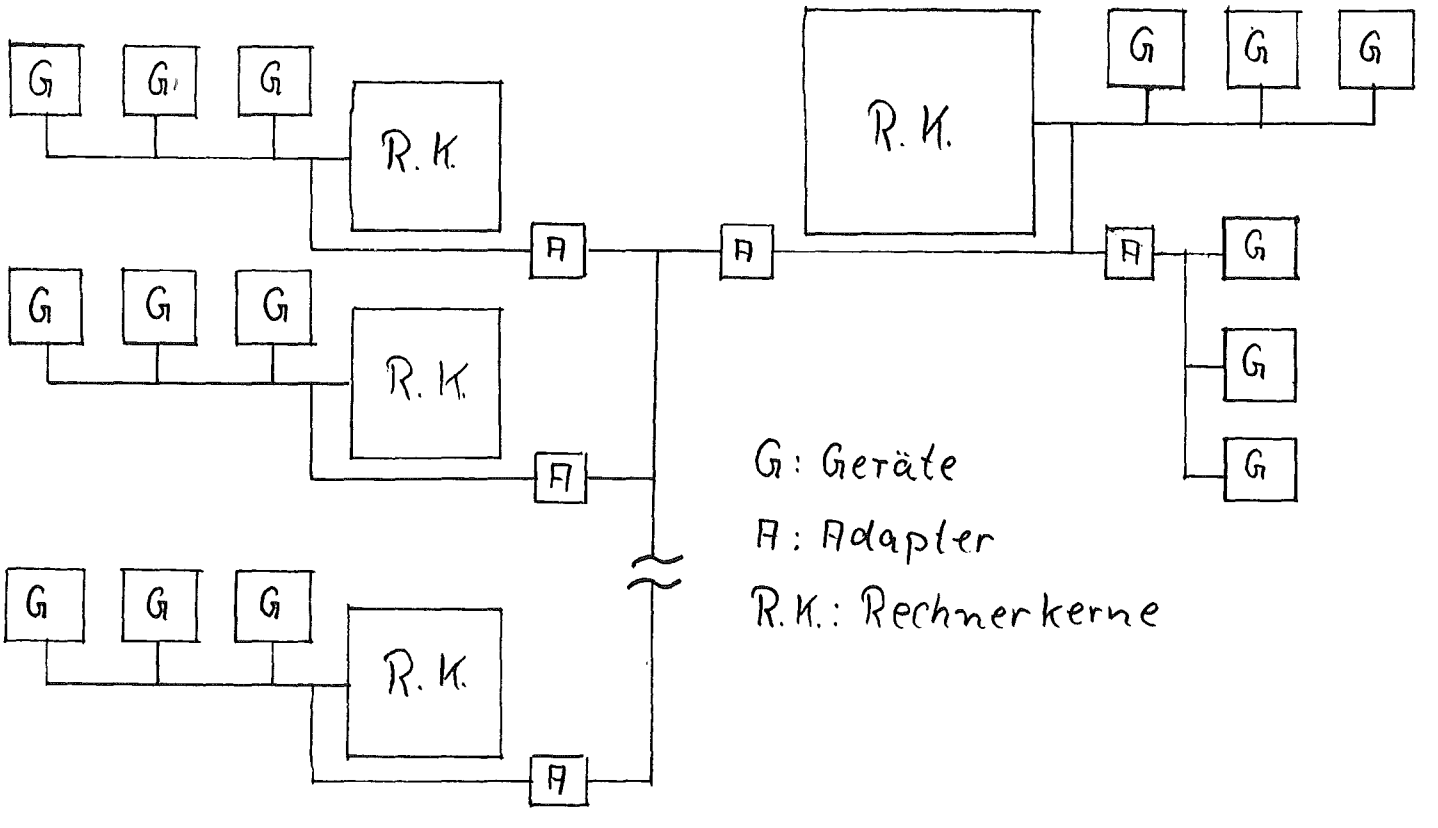


Bild 1. Konfigurationsbeispiel für ein Terminalsystem
(Datensammelsystem)

Als Einsatzgebiet für ein solches System kann z.B. an ein Großhandelsunternehmen gedacht werden. Für jeden Bereich wie z.B. Lagerwesen, Buchhaltung oder Verkauf wird ein intelligentes Terminal - bestehend aus einem kleinen Rechner und einigen Ein-Ausgabegeräten - vorgesehen, mit dem die dort jeweils anfallenden Datenverarbeitungsaufgaben wie z.B. Inventur, Debitorenbuchhaltung oder Fakturierung erledigt werden. Den Gesamtanwendungskomplex für ein Terminalsystem nenne ich im weiteren Applikation.

Sinn der Modellbildung und der an den Modellen durchzuführenden Untersuchungen soll es sein, qualitative und quantitative Aussagen über das Verhalten der einzelnen Module, über das Zusammenwirken verschiedener Module und über den Informationsfluß zwischen den Modulen zu machen.

2. Die Beziehung zwischen Modell und Objekt

Das Objekt - das ist das reale System - ist in vielen hierarchisch abgestuften Ebenen strukturiert, die Schnittstellen zwischen diesen Ebenen sind jeweils "Sprachen" oder "Befehlslisten" (Bild 2).

In unseren Modellen dagegen wird die gesamte hierarchische Struktur ignoriert, die Konfiguration wird vielmehr so beschrieben, wie sie sich dem Anwender darstellt, das Ergebnis ist das Konfigurations-Modell.

Für die Implementierung des Konfigurations-Modells - man könnte auch sagen für seine Realisierung - wird das für die Behandlung solcher Probleme bei uns entwickelte Simulationspaket

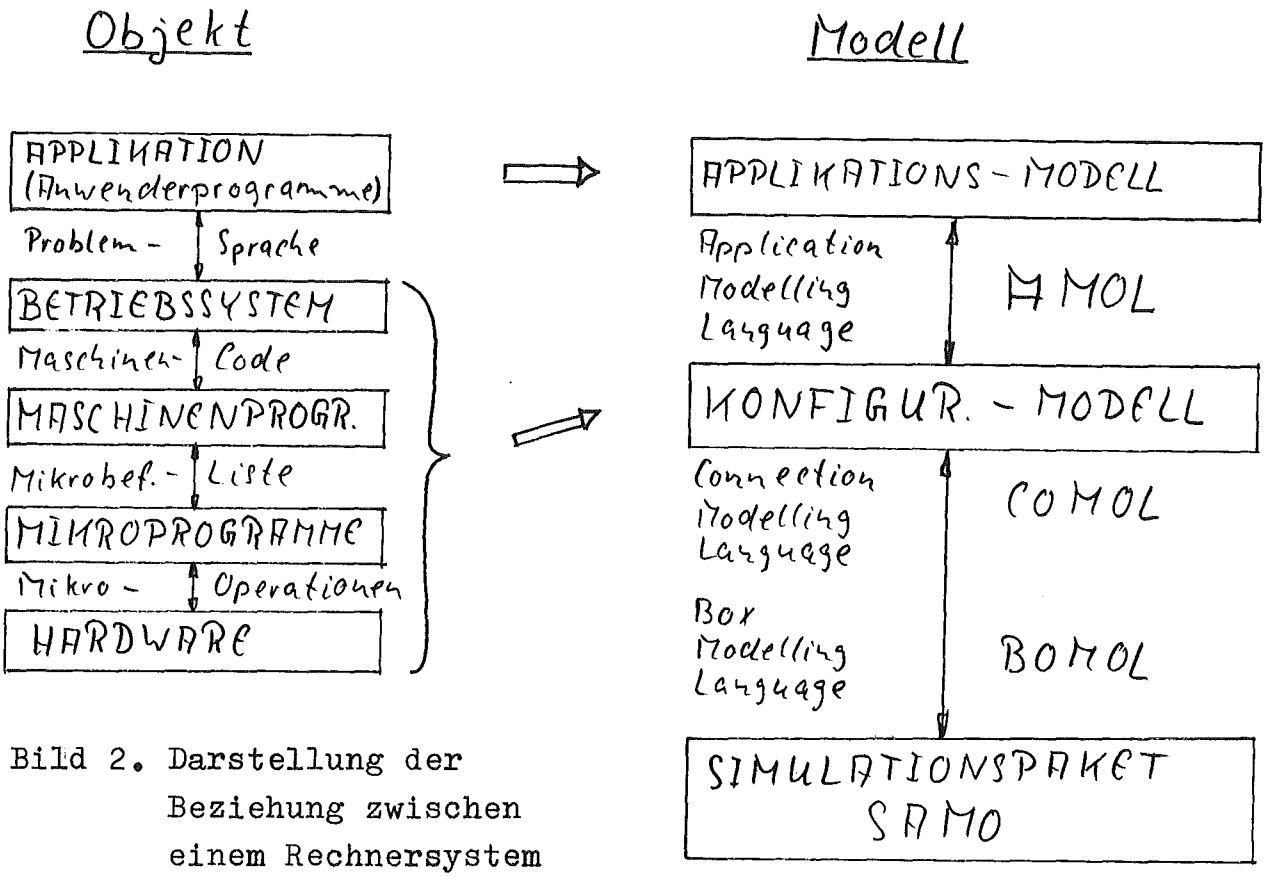


Bild 2. Darstellung der Beziehung zwischen einem Rechnersystem und seinem Modell

SAMO verwendet. Bei der Anwendung dieses Paketes werden die einzelnen Module der Konfiguration jeweils mit Hilfe einer sog. Box Modelling Language (BOMOL) erfaßt, während die Connection Modelling Language (COMOL) das Verbindungsnetz zwischen den Modulen nachbildet, also die Konfiguration festlegt.

Völlig getrennt vom Konfigurations-Modell existiert ein Applikations-Modell, das aus den Anwenderprogrammen herzuleiten ist. Das Applikations-Modell wird vom Konfigurations-Modell verarbeitet, analog wie im Objekt die Applikation von der Konfiguration. Im Objekt ist die Schnittstelle zwischen den beiden Teilen im allgemeinen eine Problemsprache wie z.B. COBOL, im Modell heißt die entsprechende Sprache AMOL von Application

Modelling Language. Über das Applikations-Modell will ich hier aus Zeitgründen nicht sprechen, sondern mich jetzt dem Konfigurations-Modell zuwenden.

3. Das Konfigurations-Modell

Das Konfigurations-Modell besteht aus den Teil-Modellen für die einzelnen Module wie Rechnerkerne, Geräte und Adapter und aus einem Modell des Verbindungsnetzes. Die Modellierung der verschiedenen Module erfolgt unabhängig voneinander. Im folgenden werde ich die Vorgehensweise der Modellbildung und das Konzept, nach dem die Teil-Modelle aufgebaut sind, am Beispiel eines Rechnerkerns skizzieren.

Wir können drei Phasen bei der Modellbildung unterscheiden:

In der ersten Phase wird das Pflichtenheft des Teil-Modells definiert. Es besteht aus der Auflistung aller sog. Elementaraktionen, die es ausführen kann. Im Falle unseres Rechnerkerns erhalten wir z.B. etwa die in Bild 3a gezeigte Liste.

ADRESSE SENDEN
STEUERBEFEHL SENDEN
DATEN SENDEN
STATUSMELDUNG EMPFANGEN
DATEN EMPFANGEN
DATEN VERKNÜPFEN
INTERRUPT VERARBEITEN

Bild 3a. Beispiele für Elementaraktionen eines Rechnerkerns

In der zweiten Phase wird jede Elementaraktion näher definiert. Hier geht es z.B. um Fragen der Codierung von Informationen und um das Festlegen und Beschreiben von Parametern für die einzelnen Elementaraktionen.

In der dritten Phase wird die Reihenfolge definiert, in der diese Elementaraktionen ausgelöst werden sollen. Diese Reihenfolge ist natürlich nicht starr, sondern hängt im wesentlichen von drei Einflußgrößen ab, und zwar von

- der jeweils in Ausführung befindlichen Elementaraktion,
- der von außen in das Teil-Modell kommenden Information,
- dem gerade zu bearbeitenden AMOL-Statement des Applikationsmodells.

Das Ergebnis dieser Phase läßt sich sehr anschaulich durch ein Zustandsdiagramm darstellen, wie es in Bild 3b angedeutet ist.

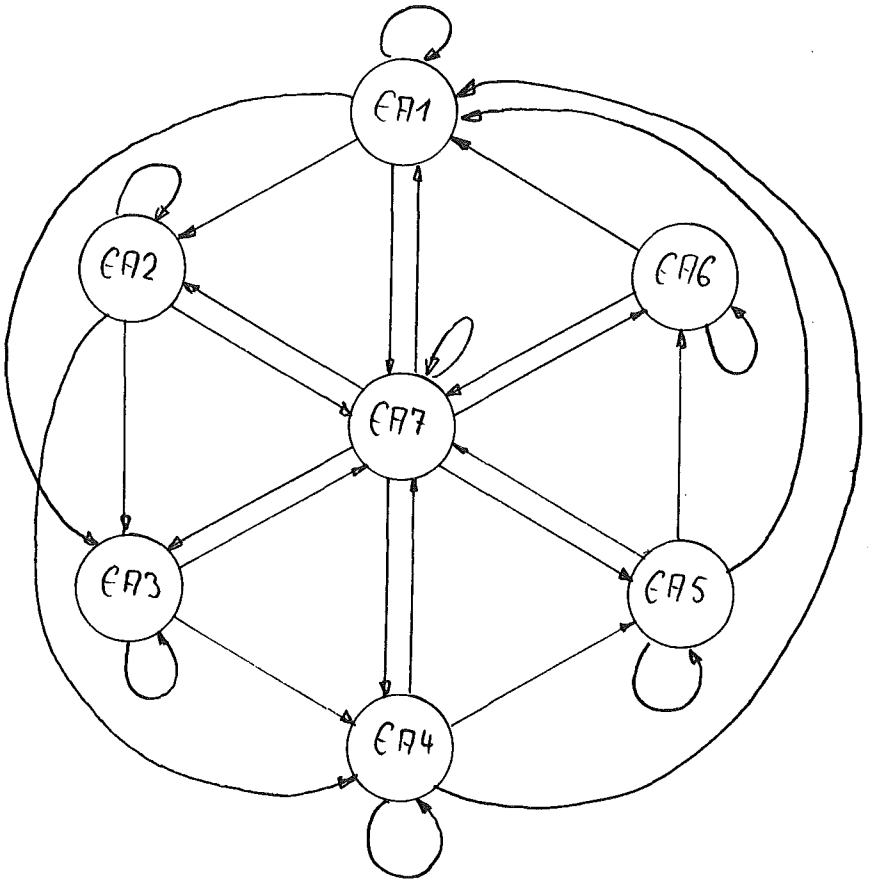


Bild 3b. Darstellung des Ablaufs von Elementaraktionen durch ein Status-Diagramm

Die als Kreise dargestellten Zustände sind den Elementaraktionen zugeordnet (EA1 bis EA7). Man sieht, daß es hinsichtlich der Reihenfolge, diesen Graphen zu durchlaufen, von jeder Elementaraktion aus gesehen mehrere Möglichkeiten gibt. Das Teil-Modell wird also gesteuert vom Applikations-Modell, von der Umwelt und vom internen Zustand. Es stellt demnach - automatentheoretisch gesehen - eine Erweiterung des Moore-Automaten dar.

Bild 4 zeigt das strukturelle Konzept des Teil-Modells für einen Rechnerkern. Es enthält als wesentliche Bestandteile

- einen Speicher in den der betreffende Teil des Applikations-Modells eingebracht wird
- eine Liste der Elementaraktionen mit allen in der Phase 2 der Modellbildung festgelegten Spezifikationen
- eine Aktionssteuerung, die für die Ausführung der jeweiligen Elementaraktion sorgt
- ein Aktionsprogramm, in dem die Reihenfolge der Elementaraktionen, wie sie im Zustandsdiagramm festgelegt wurde, steht.

Besonderes Gewicht wird bei dem Konzept der Modellbildung, wie es an diesem Beispiel skizziert wurde, auf Modularität und Flexibilität gelegt: Alle wesentlichen Teile des Modells können leicht abgeändert werden.

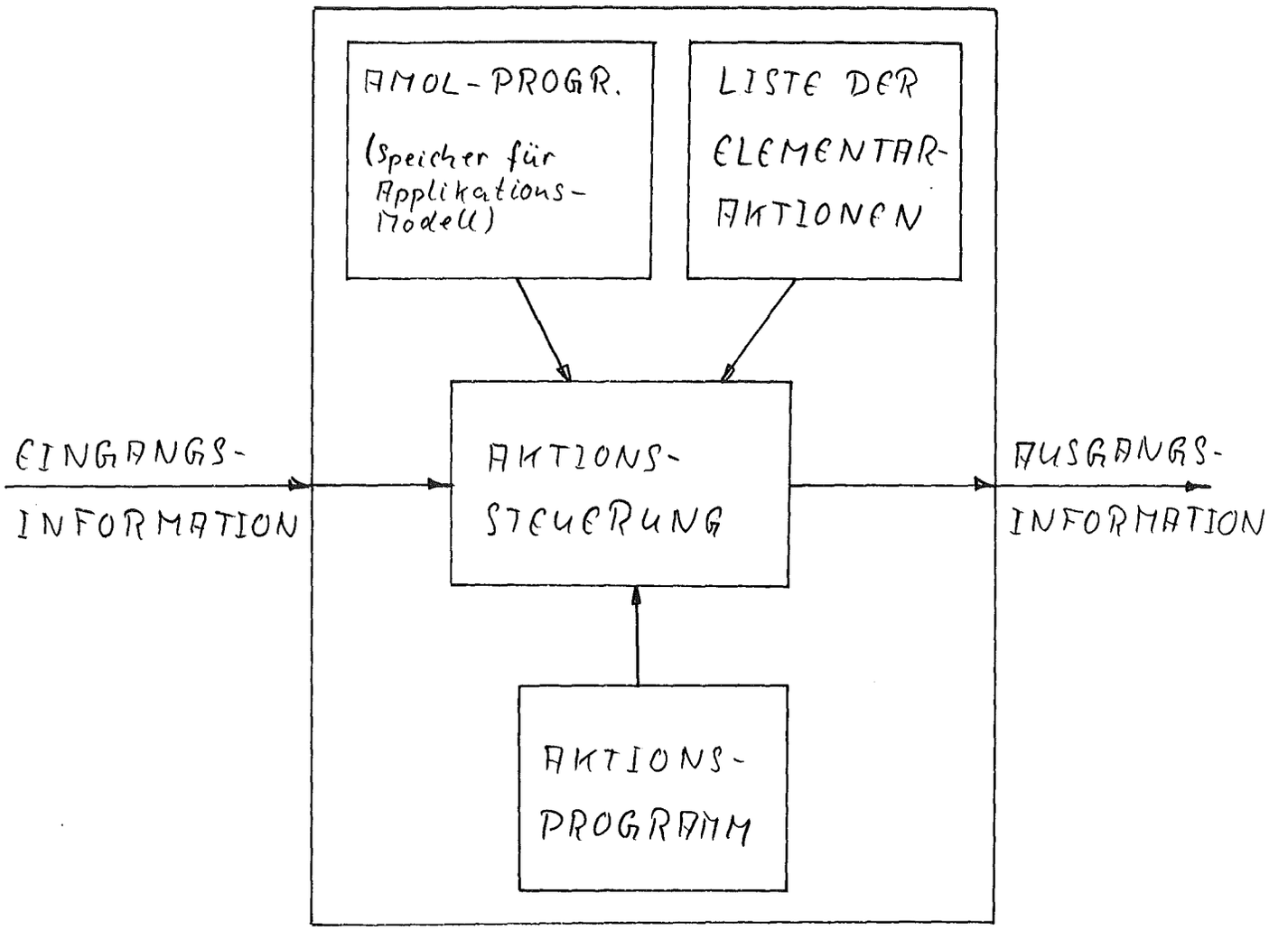


Bild 4. Konzept für die Struktur des Teil-Modells Rechnerkern

4. Stand der Arbeiten

Die ersten Voruntersuchungen, bei denen kleinere Konfigurationen - jeweils ein Rechnerkern und einige Geräte - modelliert und simuliert wurden, sind abgeschlossen.

Zur Zeit wird das Modell für das anfangs skizzierte Datensammelsystem bearbeitet.

5. Schlußbemerkung

Wenn ich versuchen soll, die bei der Modellbildung auftretenden Probleme und Schwierigkeiten zu benennen, so würde ich drei Kategorien anführen, und zwar

1. Definition der Schnittstelle zwischen Konfigurations- und Applikations-Modell, d.h. Festlegung der Applikation Modelling Language (AMOL), um feste Randbedingungen für die Entwicklung von Konfigurations-Modellen zu erhalten.
2. Das Zusammentragen und Ordnen aller Informationen des zu simulierenden Objektes, die für die Modellbildung wesentlich sind.
3. Der vom Objekt zum Modell führende Abstraktionsprozeß, der unter anderem eine starke Informationsreduzierung beinhaltet.

30.4.1973/RI/br

ANWENDERFREUNDLICHKEIT IN DER SIMULATIONSTECHNIK

Von Klaus Lagemann

Mitteilung aus dem Philips Forschungslaboratorium Hamburg

1. Einleitung

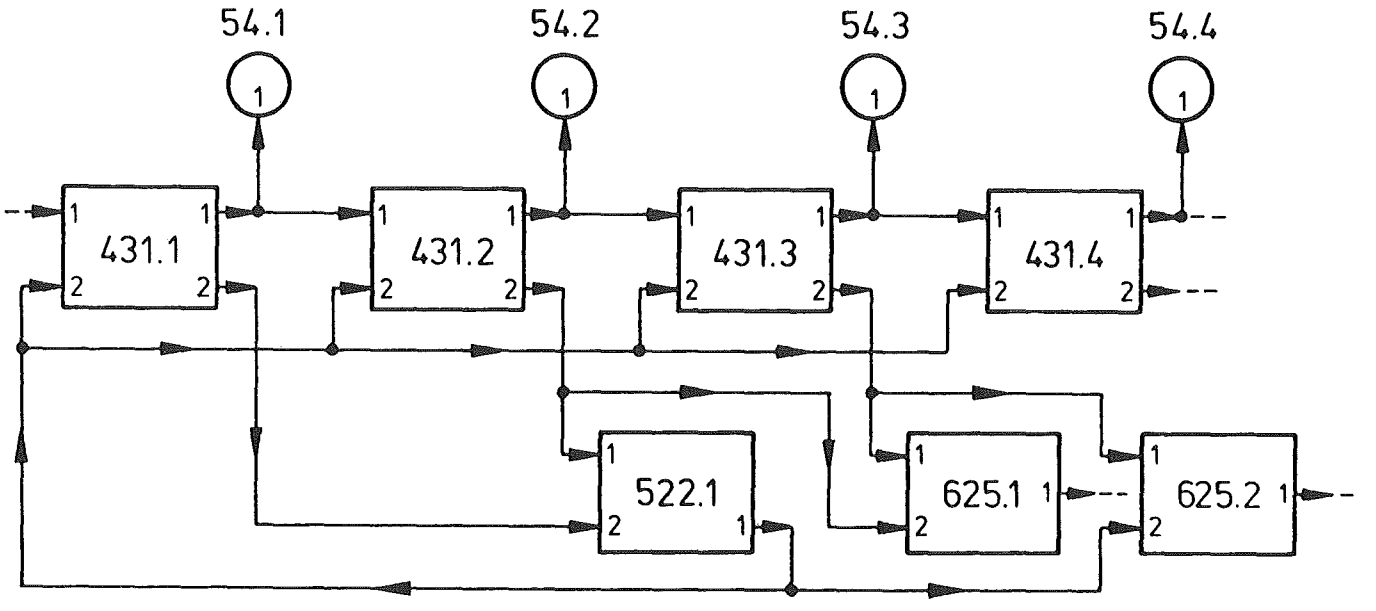
Die mit jeder neuen Rechnerfamilie immer komplizierter werdenden Systemüberlegungen veranlaßten unsere Entwicklungsleitung, sich beim Entwurf von neuen Rechnern der mittleren Datentechnik mit der Simulationstechnik auseinanderzusetzen. Die Entwicklungsleitung ging dabei von der praxisgerechten Vorstellung aus, daß die einzelnen Entwickler sich natürlich im wesentlichen um die Probleme des Entwurfsobjekts, also des neuen Rechners zu kümmern hätten und daß die Simulationstechnik dabei lediglich die Funktion eines Untersuchungswerkzeuges einnimmt, dessen Handhabung im Prinzip genauso nebenbei erlernt wird, wie man dies z.B. beim Untersuchungswerkzeug 'Oszillograf' gewohnt ist. In Wirklichkeit erweist sich die Simulationstechnik aber als Spezialisten-Handwerk; aus der Sicht des Entwicklungslabors erscheint sie sogar als praxisferne Spezialisten-Wissenschaft.

Für unser Forschungslabor ergab sich daher die Frage, wie die Simulationstechnik als Entwicklungswerkzeug anwendungsfreundlicher gestaltet werden könnte. Es entstand dabei das Simulationsprogrammpaket SAMO, das zunächst auf der Rechenanlage Electrologica X8 und danach auch auf der Control Data CD 6600 installiert wurde. Die folgenden Abschnitte zeigen, wie das Paket SAMO von der vordringlichen Forderung nach Anwenderfreundlichkeit geprägt ist und welche Erfahrungen bisher damit erzielt wurden.

2. Aufbau und Beschreibung eines Modells in SAMO

Über das Programmpaket SAMO wurde bereits am 12.4.72 auf der NTG-Fachtagung "Betriebssysteme" in Darmstadt berichtet, so daß hier eine Zusammenfassung genügen soll:
Der vorgesehene Anwendungsbereich, also die Simulation von Rechnern, legt es nahe, das Modell des zu simulierenden Objekts als System von Black-Boxes und gegenseitigen Verbindungen dar-

a) Der Modellplan



Es existieren in diesem Plan 4 Black-Box-Typen (Typennummern: 431, 522, 625, 54). Die Exemplare des gleichen Typs werden fortlaufend durchnumeriert.

b) Die Modellbeschreibung der Verbindungen in COMOL

Syntax für die Grundform einer Verbindung:

cr <Black-Box-Typ-Nr.> . <Exemplar-Nr.> . <Ausgangs-Nr.> = <Black-Box-Typ-Nr.> . <Exemplar-Nr.> . <Eingangs-Nr.>

Das vollständige Verbindungsnetz enthält neben der Grundform auch noch zusammengesetzte Formen. Die dabei verwendeten Ausklammerungsregeln verstehen sich beim Vergleich mit dem Modellplan von selbst:

cr 431.1.2 = 522.1.2	Grundform
cr 431.2.2 = 522.1.1 + 625.1.2	Verzweigung, ausgedrückt durch das Plus-Zeichen
cr 431.3.2 = 625(1+2)1	Verzweigung, mit Ausklammerung gemeinsamer Nummern
cr 522.1.1 = 431(1++4)2 +625.2.2	Vielfach-Verzweigung; Doppel-Plus drückt "von-bis" aus
cr 431(1--3)1 = 431(2--4)1	Wiederholung ähnlicher Verbindungen, ausgedrückt durch
cr 431(1--4)1 = 54(1--4)1	das Minus- bzw. Doppel-Minus-Zeichen

Bild 1: Modellbildung für die Systemstruktur eines Objekts

zustellem (Bild 1). Dabei ist es belanglos, ob eine Black-Box im einfachsten Fall einem Verknüpfungsglied, z.B. einem NAND-Gatter, entspricht, oder ob sie z.B. das Verhalten eines kompletten Rechners in einem Rechnerverbundsystem widerspiegelt. Wichtig ist dagegen, daß eine Black-Box tatsächlich stellvertretend für einen abgegrenzten Teil der Wirklichkeit steht. Die praktische Konsequenz läßt sich an einem einfachen Beispiel aufzeigen:

Gegeben sei der Hardware-Versuchsaufbau einer Zentraleinheit mit den typischen Baugruppen "Rechenwerk", "Leitwerk", und "Speicher". Jede Baugruppe ist in sich so abgeschlossen, daß sie sich gegen die übrigen Baugruppen deutlich abgrenzen und auch für sich alleine testen läßt. Um das Zusammenspiel der Baugruppen zu prüfen, sind passende Verbindungen anzulegen, die zwischen den Baugruppen liegen und diesen selbst nicht angehören. Diese vom Hardware-Aufbau her gewohnte Denk- und Untersuchungsweise sollte auch in die Simulationstechnik übernommen werden. Eine Baugruppe, also eine Black-Box ist in sich abgeschlossen und unabhängig von anderen Black-Boxes darzustellen. Daraus folgt z.B. , daß Zustandsvariable nicht in der einen Black-Box gesetzt und in einer anderen Black-Box abgefragt werden dürfen. Die Korrespondenz der Black-Boxes untereinander erfolgt, genauso wie im Hardware-Aufbau, lediglich über Zwischen-Verbindungen.

Für unser Simulationsprogrammpaket ergab sich daher eine natürliche Zweiteilung der Modellbeschreibungssprache.

a) Die Sprache BOMOL (Box-Modeling-Language) dient der Beschreibung von Black-Boxes. Bei der Vielfalt denkbarer Black-Box-Typen mit wechselnder Komplexität und Modellierungstiefe wird eine möglichst flexible Sprache gebraucht. BOMOL umfaßt daher voll die Sprache ALGOL und sieht im wesentlichen zusätzlich noch die Parallelprogrammierung vor. Selbstverständlich braucht der Anwender bei der Formulierung von Black-Boxes nur die tatsächlich dem Objekt entstammenden Fragen zu bedenken. Alle simulationsspezifischen Probleme, wie z.B. Generierung der Modellzeit, Überwachung und Serialisierung gleichzeitiger Vorgänge, Stabilitätskontrollen übernehmen die im nächsten Abschnitt erwähnten Hilfsprogramme.

b) Die Sprache COMOL (Connection Modeling Language) ist dagegen sehr einfach. Im Prinzip genügt es, die Verbindungen einfach aufzulisten, indem paarweise der Black-Box-Ausgang und der

zugehörige Black-Box-Eingang genannt werden. Schon das Bild 1 läßt erkennen, daß sich durch Wiederholung ähnlicher Verbindungen oder durch Verzweigungen oft kürzere Schreibweisen ergeben. Die verschiedenen Formen der Verbindungen gehen durch Formelumwandlung ineinander über. Da die üblichen problemorientierten Sprachen aber gerade für die Formelbehandlung kaum Hilfen anbieten, wurde die Sprache COMOL als neuer, allerdings sehr einfacherRegelsatz konzipiert.

3. Der Aufbau des Programmpakets

Das Simulationsprogrammpaket SAMO läßt sich - bildlich gesprochen - als ein Gestell mit zunächst leeren Fächern auffassen. Wesentlich ist die deutliche Trennung von Anwenderteil und Herstellerteil. Der Herstellerteil nimmt die bereitgestellten Hilfsmittel auf, während die Fächer des Anwenderteis für die einzelnen Black-Box-Typen und Verbindungsnetze vorgesehen sind. Es sind vier Auftragsarten BOX, NET, TIE, RUN möglich, die der Anwender über ein spezielles Fach einzeln oder kombiniert auslösen kann:

Der Auftrag BOX: Dieser Auftrag zielt darauf ab, einen einzelnen Black-Box-Typ für sich allein zu behandeln, z.B. ihn neu einzubringen, zu korrigieren oder zu löschen. Dabei veranlaßt der BOMOL—ALGOL-Übersetzer eine Syntaxprüfung und erzeugt dann eine ALGOL-Darstellung des betreffenden Black-Box-Typs. Die Fächer 1 bis 99 sind als Bibliothek zur Aufnahme häufig gebrauchter Black-Box-Typen reserviert.

Der Auftrag NET: Ein in COMOL formuliertes Verbindungsnetz gibt lediglich wieder, wie die Verbindungen zwischen den Black-Boxes künftig liegen werden. Eine COMOL-Beschreibung läßt sich also ohne Rücksicht darauf anfertigen, ob die darin genannten Black-Box-Typen schon als BOMOL-Programme vorliegen. Der Ablauf des Auftrags NET mit Einbringen, Korrigieren oder Löschen eines Verbindungsnetzes, Syntaxprüfung und Übersetzung in eine ALGOL-Darstellung gleicht äußerlich weitgehend dem Auftrag BOX.

Der Auftrag TIE: Durch diesen Auftrag werden jeweils die ALGOL-Darstellungen einiger Black-Box-Typen und eines Verbindungsnetzes zu einer Gesamtbeschreibung des Modells zusammengefaßt. Erst durch diese "statische ALGOL-Modell-Beschreibung" wird auch eine Konsistenz-Prüfung möglich; z.B. kann erst jetzt

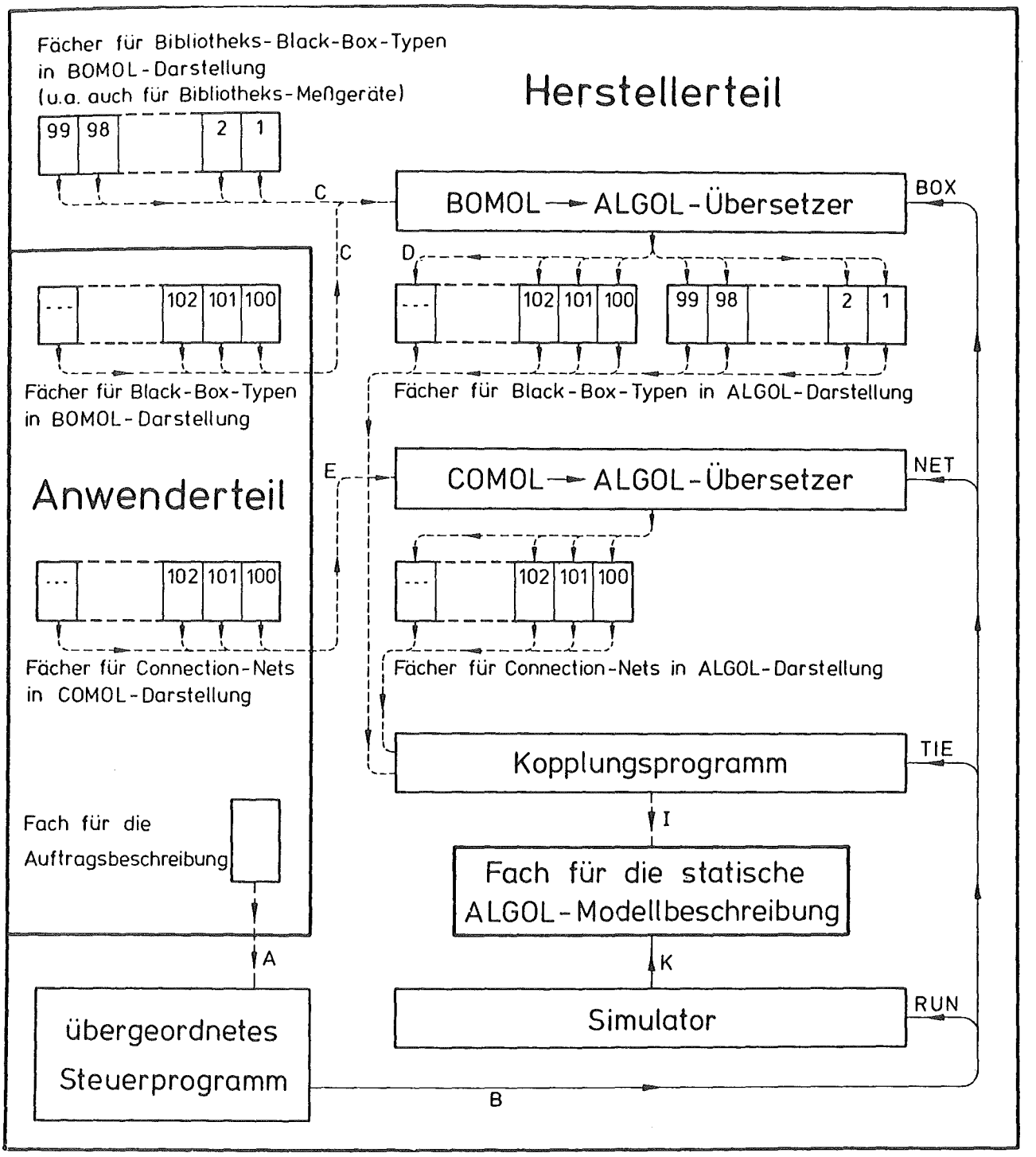


Bild 2: Zusammenarbeit zwischen Anwender und Hersteller in der Phase "Modell betreiben und untersuchen"

Durchgezogene Pfeile: Steuerungen
Gestrichelte Pfeile : Transport von Programmen oder Daten
Pfeil A : Das Steuerprogramm holt sich die Auftragsbeschreibung
Pfeil B : Das Steuerprogramm aktiviert die Aufträge BOX, NET, TIE und/oder RUN
Pfeile C und D : Der BOMOL → ALGOL-Übersetzer holt sich die Black-Box-Typen in BOMOL-Darstellung und liefert sie in ALGOL-Darstellung zurück
Pfeile E und F : Der COMOL → ALGOL-Übersetzer holt sich ein Verbindungsnetz in COMOL-Darstellung und liefert es in ALGOL-Darstellung zurück
Pfeile G,H, I : Das Kopplungsprogramm holt sich die ALGOL-Darstellungen der Black-Box-Typen und eines Verbindungsnetzes und formt daraus eine statische ALGOL-Modellbeschreibung
Pfeil K : Unter Kontrolle des Simulators wird das Modell dynamisch betrieben

festgestellt werden, ob ein in einer Verbindung genannter Black-Box-Eingang in der BOMOL-Darstellung des betreffenden Black-Box-Typs versehentlich gar nicht existiert.

Der Auftrag RUN: Erst mit diesem Auftrag beginnt der eigentliche Simulationslauf. Dabei werden die in der statischen ALGOL-Modellbeschreibung genannten möglichen Vorgänge dynamisch wirklich durchgespielt. Dazu bedarf es eines weiteren Hilfsprogramms, nämlich des Simulators. Er muß die Modellzeit erzeugen und in ihr die im Objekt ablaufenden Zeitvorgänge richtig widerspiegeln. Deshalb hat er über alle in den Black-Boxes gleichzeitig laufenden Vorgänge Buch zu führen und zu gegebener Modellzeit den Informationsaustausch über die Verbindungen zu veranlassen. Dieser Simulator arbeitet "vorausschauend" und überspringt ereignislose Spannen in der Modellzeit. Für die Programmlaufzeit ist daher nicht die Schrittweite der Modellzeit sondern die Anzahl der auftretenden Ereignisse maßgebend.
Der Simulator nimmt nicht zur Kenntnis, was die von ihm kontrollierten Black-Boxes und Verbindungen inhaltlich bedeuten. Er ließ sich somit als eigenständiger Modul in den Gesamtrahmen einfügen und erfüllt seine Aufgabe unabhängig vom Anwendungsbereich.

Der modulare Aufbau von SAMO erlaubt eine weitgehend unabhängige Verbesserung der einzelnen Module. Die im Herstellerteil bereitgestellten Hilfsmittel umfassen z.Z. etwa 7000 ALGOL-Zeilen.

4. Erfahrungen mit SAMO aus der Sicht der Anwenderfreundlichkeit

4.1 Erfahrungen zum Erlernen von SAMO

Die Regeln von COMOL zur Beschreibung der Verbindungen lassen sich in einer halben Stunde erlernen. Wer die Sprache ALGOL 60 kennt, kann sich mit BOMOL in etwa einem Tag vertraut machen. Es hat sich gezeigt, daß auch bei komplizierteren Black-Box-Typen die für ALGOL charakteristischen Ausdrucksmittel, wie die dynamischen Feldgrenzen, die Blockstruktur oder die spezielle Prozedur-Vereinbarungstechnik kaum benötigt werden. Man könnte BOMOL statt an ALGOL also ebensogut an FORTRAN, COBOL oder sogar BASIC anlehnen.

Bemerkenswert erscheint, daß die Anwender von BOMOL bei der Formulierung von Zeitabhängigkeiten häufig Fallen vermuten und sich nicht auf den durch die Hilfsprogramme gelieferten Zeit-

kontrollmechanismus verlassen mögen, ohne ihn selbst genau verstanden zu haben.

Insgesamt erscheinen die Schwierigkeiten beim Erlernen der Sprachen BOMOL und COMOL vernachlässigbar klein gegenüber den Problemen bei der Modellbildung (s. Punkt 4.6).

4.2. Erfahrungen zur Wiedergabe der Ergebnisse

Die Wünsche der Anwender an die Art der Ergebnisanzeige sind erfahrungsgemäß recht vielschichtig und unterscheiden sich vor allem nach Probephase und Nutzphase der Simulation.

a) In der Probephase der Simulation soll das Modell logisch einwandfrei gestaltet werden. Erfahrungsgemäß gerät ein fehlerhaftes Modell zunächst schon nach wenigen Simulationsschritten (Modellzeitschritten) in einen erkennbar unsinnigen Gesamtzustand. Es ist dann für die Fehlersuche von großem Vorteil, wenn ein möglichst vollständiges Protokoll über alle Zustandswechsel automatisch angefertigt wurde. Mit wachsender Fehlerfreiheit dagegen nimmt der Ausstoß an Protokolldaten enorm zu. Das automatische Protokoll muß daher über besondere Steuerdaten selektiv nach Modell-Ort und Modell-Zeit auf die vermutete Fehlerquelle hin eingegrenzt werden können.

b) In der Nutzphase fallen die gesuchten Ergebnisse an. Hierbei wird die vom Hardware-Versuchsaufbau gewohnte Untersuchungstechnik nachgeahmt, wonach die Anzeigegeräte je nach Bedarf an passend ausgewählte Stellen der Versuchsanordnug angeschlossen werden. Die Black-Box-Bibliothek enthält daher die Modelle von oft gebrauchten Anzeigegeräten, so daß die Ergebnisse wahlweise in Form von Tabellen, Impulsdiagrammen, Balkendiagrammen oder statistischen Angaben (Mittelwert, Streuung) ausgegeben werden können.(s. Bild 3 u. 4). Der Anwender kann sich mit BOMOL auch eigene Anzeige-Boxen formulieren.

Sowohl in der Probephase als auch in der Nutzphase eines Simulationslaufs sollte in regelmäßigen Zeitabständen der Gesamtzustand des Modells "gerettet" werden. Dadurch wird es möglich, einen wegen irgendwelcher Fehler abgebrochenen Simulationslauf später bei dem zuletzt gerettetem Gesamtzustand wieder aufzugreifen, so daß der fehlerfreie Zeitabschnitt nicht nochmals

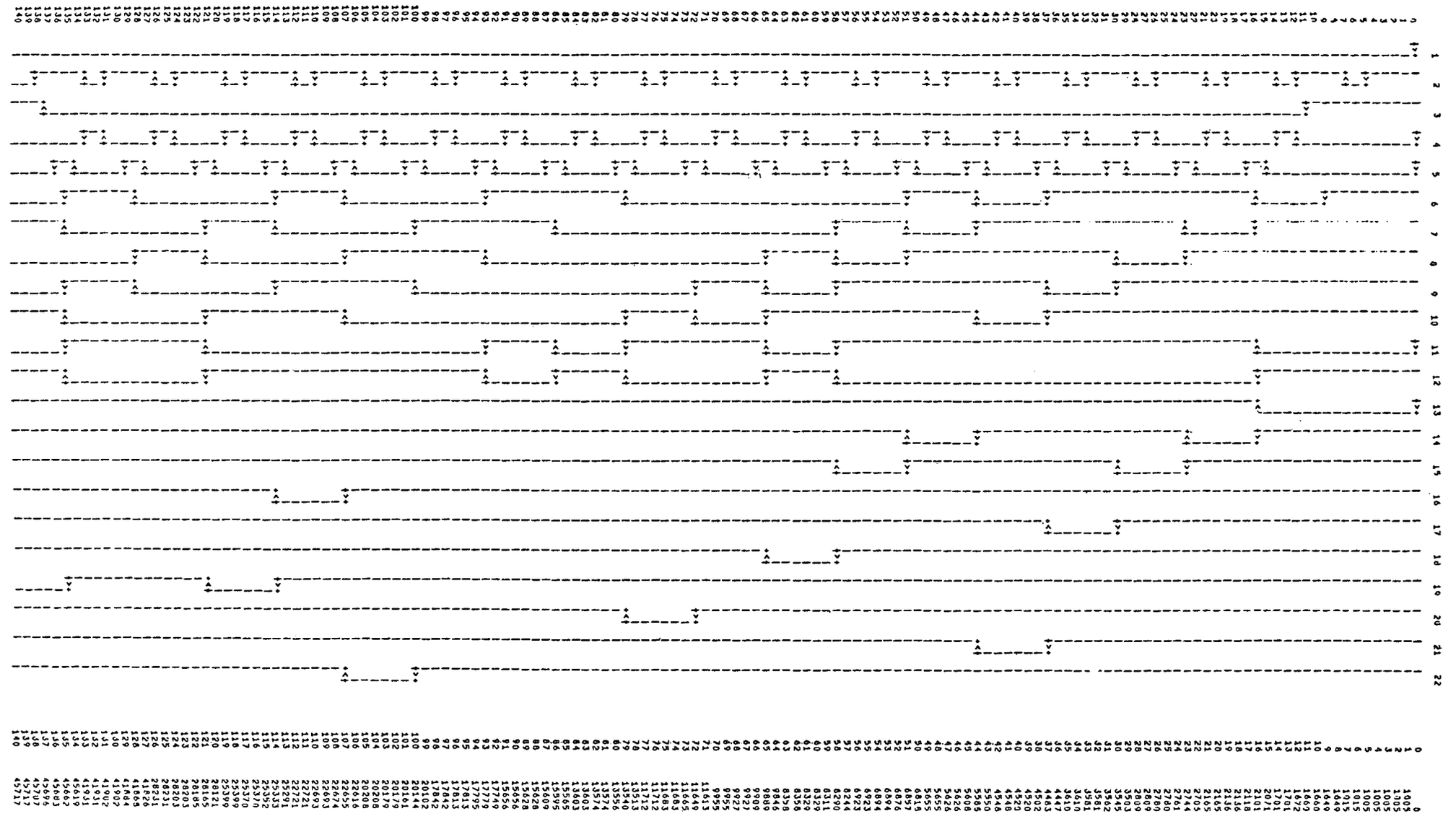


Bild 3: Vollständiger Diagrammabdruck zur Simulation eines Schaltwerks

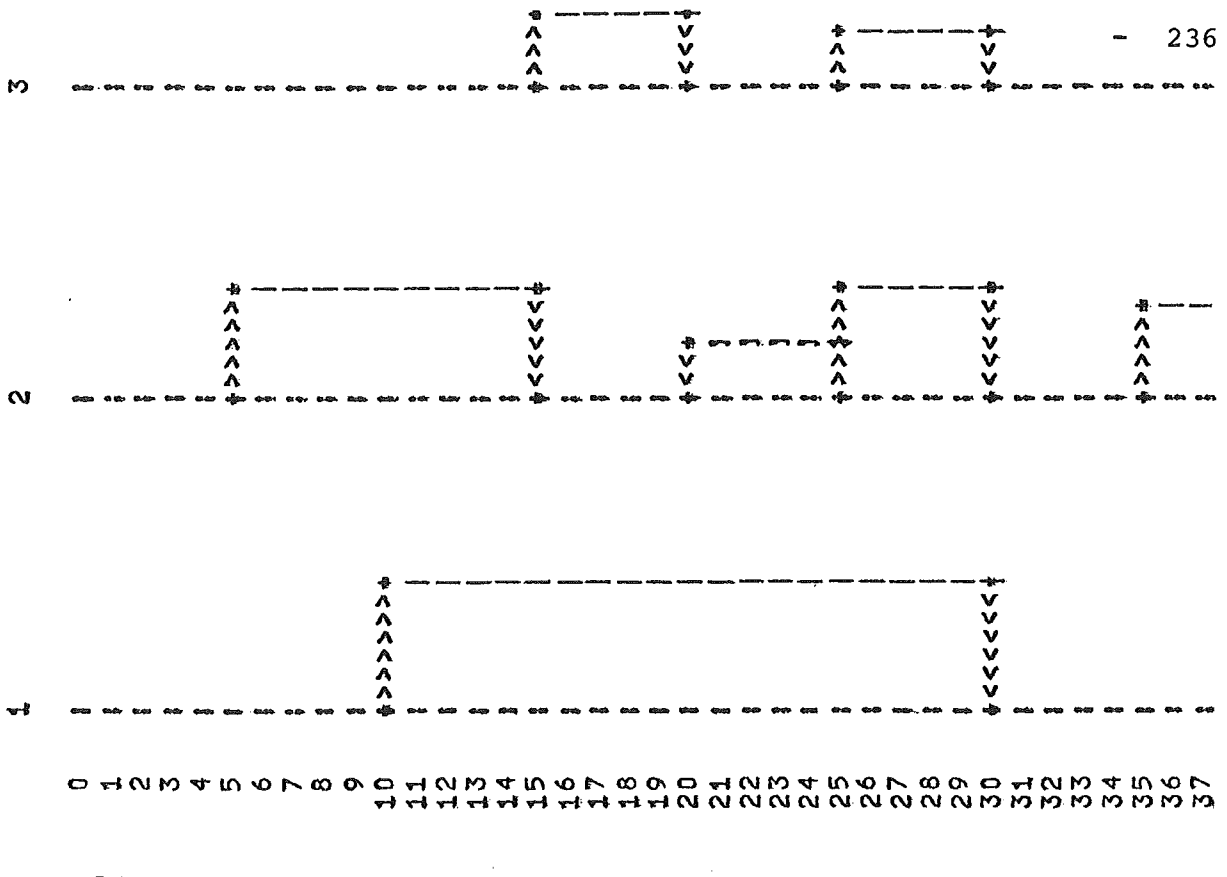


Bild 4: Impulsdiagramm mit wechselnder Amplitude (Ausschnitt)

BUS1 UNBENUTZT (LEERZEIT)

ATMOD : 17276
ANZAHL DER MESSUNGEN : 330
MINIMUM : 1
MAXIMUM : 1019
MITTELWERT : +,1770606060608$_{10}$+002
STANDARDABWEICHUNG : +,8600548663957$_{10}$+002

MESSWERT		ANZAHL
+1	**********	220
+3	******	10
+8	*********	16
+11	**	3
+14	*********	16
+16	****	8
+19	*******	13
+22	***	6
+24	*	1
+27	*****	9
+30	**	3
+32	*	2
+35	***	6
+40	**	3
+43	*	1
+46	**	3
+54	*	1
+75	**	3
+203	*	1
+355	*	1
+547	*	1
+555	*	1

Bild 5: Messung von Leerzeiten auf einem Rechner-Bus

Ergebnisanzeige durch Balkendiagramm, Mittelwert, Standardabweichung, Minimum und Maximum

zu simulieren ist. Außerdem läßt sich ein längerer Simulationslauf in kürzere Teilläufe zerschneiden, wodurch man sich oft den Prioritätenregelungen des simulierenden Rechners besser anpassen kann. Das "Retten" der Gesamtzustände erfordert allerdings einen hohen Aufwand bei den Hilfsprogrammen und an Speicherplatz.

4.3. Erfahrungen mit den Hilfsprogrammen

Das Bestreben, dem Anwender bei der Beschreibung seines Modells möglichst viele Unannehmlichkeiten zu ersparen, führt schnell zu umfangreichen Hilfsprogrammen. Damit sind zwei Gefahren verbunden:

a) Bei Maschinenfehlern der Rechenanlage bleibt das Programmpaket in irgendeinem dem Anwender unverständlichen Stadium stehen. Es ist insbesondere bei der X8 nicht leicht möglich, dem Anwender auch bei Fehlerabbruch ein klares Bild über den erreichten Stand der Auftragsabwicklung zu vermitteln.

b) Entgegen den ursprünglichen Erwartungen sind weitaus die meisten Simulationsaufträge von der Art "BOX" und "NET" und dienen somit dem Aufbau des Modells. Es handelt sich - aus der Sicht des Anwenders - meistens um kleine Aufträge, von denen er sich eine schnelle Erledigung erhofft. Aus der Sicht der simulierenden Rechenanlage jedoch entsteht - wegen der vielen Hilfsprogramme - stets ein größerer Job, so daß leicht durch Prioritäten bedingte Wartezeiten auftreten.

4.4. Erfahrungen mit den Laufzeiten

Nahezu jeder Anwender hat zunächst Schwierigkeiten mit den Laufzeiten der Simulationsprogramme. Das änderte sich auch nicht nach einer Steigerung der Laufgeschwindigkeit um den Faktor 100. Offensichtlich geht der Anwender von einer ihm sinnvollen maximalen Laufzeit aus - z.B. 30 Minuten - und er befreit sein Modell solange von überflüssigen Details, bis er in dieser Laufzeit gerade genügend viele Ergebnisse bekommt. Demnach erweist sich die Laufzeit-Frage nicht so sehr als Problem des Simulators als vielmehr als Problem einer geeigneten Modellbildung.

4.5. Erfahrungen beim Tranfer von der X8 auf die CD 6600

Das für die X8 fertiggestellte Programmpaket SAMO sollte so schnell wie möglich auf die CD 6600 in Frankfurt umgestellt werden, damit es sowohl von Hamburg als auch von Eiserfeld aus über Terminals benutzbar wurde. Um eine optimale Anpassung an die CD 6600 zu erreichen, hätte es einer genauen Kenntnis dieser Anlage bedurft. Eine solche Kenntnis ist aber allein durch Befragen der Rechenzentrumsverantwortlichen oder durch das Studium von Manuals nicht zu zu erreichen; vielmehr bedarf es wohl eigener Erfahrung mit der Installation von Programmpaketen, die in unserem Fall aber nicht vorlagen. Daher setzen wir 'einfach' das ganze Paket mit Hilfe von speziellen Umcodierungsprogrammen - sozusagen im Verhältnis 1:1 - auf die CDC-Anlage um. Die X8-Eingabe-/Ausgabe-Anweisungen wurden dabei als CD 6600-ALGOL-Prozeduren vereinbart, deren Rumpf aus den entsprechenden CDC-Eingabe-/Ausgabe-Anweisungen komponiert ist. Zwei besondere Probleme waren zu lösen:

a) Die Übersetzer von BOMOL nach ALGOL bzw. von COMOL nach ALGOL verarbeiten nur Text in X8-Code. Für das auf der CDC-Anlage installierte Paket wurden daher zusätzliche Code-Wandlungsprogramme nötig, die bei jedem Simulationsauftrag aktiviert werden.

b) Die Plattenspeicher-Organisationen der beiden Anlagen sind sehr unterschiedlich. Die Anpassung gelang zunächst nur dadurch, daß während der Simulationsläufe ständig ein größerer Datenvorrat in den Arbeitsspeicher verlagert wurde. Der entsprechend große Kernspeicherbedarf hatte äußerst ungünstige Folgen auf die Einordnung in die Warteschlange des Rechners und damit auf die Wartezeiten.

Letztlich ist also die X8 auf der CDC-Anlage emuliert worden, und zwar recht provisorisch. Jedenfalls wirkt sich dieser Umstand auch recht negativ auf die Rechenzeiten bei den vorbereitenden Simulationsaufträgen "BOX" und "NET" aus. Bei den Nutzläufen im Auftrag "RUN" dagegen läßt sich die hohe Rechengeschwindigkeit der CD 6600 schon recht gut ausnutzen.

Nach der halbjährigen Umstellungszeit und nach nunmehr ausreichender Kenntnis der CDC-Anlage wird das Paket zur Zeit Zug um Zug optimiert. Dabei werden einige Hilfsprogramme auch in FORTRAN ausgedrückt.

4.6. Bemerkungen zur Modellbildung

Auch ein anwenderfreundliches Simulationsprogrammpaket löst nicht das Hauptproblem der Simulationstechnik: die Modellbildung. Geeignete Modellbildung kann die Simulationslaufzeiten z.B. um den Faktor 100 verkürzen, ohne daß irgendwelche Ergebnisse vergröbert oder verfälscht werden. Das Thema Modellbildung ist bei uns zu einem eigenständigen Forschungsprogramm geworden. Zu den Zielen diese Programms gehört das Auffinden allgemeingültiger Modellstrukturen und deren automatische Umsetzung in die Anwender-Begriffswelt am Datensichtgerät. Zum andern brauchen wir Methoden zur Modellierung von Software. Außerdem hoffen wir, die Kenntnisse über geeignete Modellierungstiefe, Confidence-Level, richtige Randbedingungen oder zweckmäßigen Experimentaufbau über das Stadium einer Erfahrungswissenschaft hinaus systematisieren und damit der Entwicklungspraxis leichter zugänglich machen zu können.

Literatur:

/1/ K. Lagemann: Ein modular aufgebautes Programmpaket zur Simulation von Rechnersystemen, Nachrichtentechnische Fachberichte, Band 44 (1972): Rechner- und Betriebssysteme; Analyse, Simulation und Entwurf.

/2/ K. Lagemann: Optimierung von Teilnehmersystemen mit intelligenten und nicht intelligenten Datenendstationen: Teil 2, Modellbildung und Simulation für den Systementwurf,
Forschungsantrag an das Bundeswissenschaftsministerium vom 14.2.72, Philips Forschungslaboratorium Hamburg

SIMULATION IN DER NACHRICHTENVERKEHRSTHEORIE: PROBLEMSTELLUNGEN UND PROGRAMMIERSPRACHEN

von G. Kampe, P. Kühn und M. Langenbach-Belz

Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart

1. EINLEITUNG

In der Nachrichtenverkehrstheorie werden Probleme untersucht, die in Vermittlungssystemen für Fernsprech- oder Datenverkehr und in Rechnersystemen bei der Behandlung des Nachrichtenverkehrsflusses auftreten. Dabei wird neben der Analyse bestehender Systeme auch die Synthese neuer Systeme (Optimierung) vorgenommen. Ein System ist im wesentlichen durch seine Struktur, die Betriebsart und die statistischen Eigenschaften des Nachrichtenverkehrs gekennzeichnet. Als Untersuchungsmethoden sind Verkehrsmessungen, exakte oder näherungsweise Berechnungen sowie Simulationen auf Digitalrechnern üblich. Da die betrachteten Systeme oft sehr komplex sind, kann in vielen Fällen auch mit Großrechnern keine exakte Berechnung durchgeführt werden. Daher bildet die Simulation in der Nachrichtenverkehrstheorie überall dort ein unentbehrliches Hilfsmittel, wo entweder noch kein Rechenverfahren bekannt ist oder wo der Gültigkeitsbereich von Verfahren zur näherungsweisen Berechnung untersucht wird.

Der rechnerischen Analyse bzw. der Analyse durch Simulation geht die Modellbildung voraus. Die tatsächlichen Betriebsmittel und Vorgänge der Wirklichkeit werden dabei durch ein möglichst wirklichkeitsgetreues Modell beschrieben. Im zweiten Abschnitt der Arbeit werden zunächst die wesentlichsten Merkmale (Strukturelemente, Betriebsstrategien, Prozesse) von Modellen der Nachrichtenverkehrstheorie (kurz: verkehrstheoretische Modelle) vorgestellt, und es wird anhand dreier typischer Beispiele aus Vermittlungssystemen, Rechnersystemen sowie Netzen gezeigt, wie aus dem tatsächlichen System ein Modell entsteht. Es wird ferner kurz auf die Fragestellungen eingegangen, welche durch die Simulation beantwortet werden sollen.

Im dritten Abschnitt der Arbeit werden kurz die prinzipiellen Verfahren zur Simulation solcher Modelle anhand eines Beispiels aufgezeigt. Aufbauend auf den Erkenntnissen der vorangegangenen beiden Abschnitte wird im vierten Abschnitt der Arbeit auf Forderungen an Sprachen für Simulationsprogramme in der Nachrichtenverkehrstheorie eingegangen. Anhand eines einfachen Simulationsmodells (einstufiges Warte-Verlustsystem) wird schließlich im fünften Abschnitt eine Gegenüberstellung der folgenden vier Programmiersprachen vorgenommen:

FORTRAN IV
GPSS/360
SIMSCRIPT I.5
SIMULA 67.

Das betrachtete System wurde in jeder dieser vier Sprachen durch ein Simulationsprogramm nachgebildet. Die dabei über diese Sprachen gewonnenen Erfahrungen (z.B. Programmieraufwand, Speicherplatzbedarf, Programmlaufzeit) werden diskutiert.

2. PROBLEMSTELLUNGEN DER NACHRICHTENVERKEHRSTHEORIE

2.1 Allgemeines

In Nachrichtenvermittlungssystemen werden von Teilnehmern Verbindungswünsche (Anforderungen) gestellt, wodurch diese Teilnehmer aufgrund der gewählten Ziffern mit jeweils einem Zielteilnehmer automatisch verbunden werden sollen. Die äußeren Anreize lösen innerhalb des Vermittlungssystems eine Reihe von Vorgängen aus wie Verbinden des Teilnehmers mit einem freien Verbindungssatz, Anschalten an ein zentralisiertes Register, Aufnahme und Verarbeitung von Wählziffern, Einstellen der Koppelnetze zwischen rufendem und gerufenem Teilnehmer usw. Durch die Vielzahl der Verbindungswünsche und deren statistisch schwankende Eintreffzeitpunkte bzw. Belegungsdauern bedingt, können sich innerhalb des Vermittlungssystems Engpässe ergeben, welche sich durch Wartezeiten (Verzögerungen im Verbindungsaufbau) bzw. Besetztfälle äußern. Um einen hinreichend guten Service garantieren zu können, ist es erforderlich, die zentralen "Bedienungseinrichtungen" (Leitungen, Register, Markierer, Prozessoren, Koppelnetze usw.) ausreichend zu dimensionieren. Hierbei entstehen neben Fragen der notwendigen Anzahl von Einheiten vor allem Fragen hinsichtlich der optimalen Struktur und optimalen Betriebsstrategie, nach welcher die Bedienungseinrichtungen den Anforderungen zugeteilt werden.

Innerhalb von Rechnersystemen entstehen ganz ähnliche Probleme. Die einzelnen Anwenderprogramme stellen sehr unterschiedliche Anforderungen an die Betriebsmittel des Rechnersystems (Zentralprozessor, Ein/Ausgabeeinheiten, Hintergrundspeicher usw.). Größere Rechner werden heute weitgehendst nach der Multiprogramming-Betriebsweise organisiert. Die gleichzeitige Konkurrenz von mehreren Anwenderprogrammen bezüglich der Betriebsmittel kann ebenfalls auf interne Engpässe führen, welche durch passende Auslegung dieser Betriebsmittel bzw. intelligente Vergabe durch das Betriebssystem vermieden werden können.

Der Nachrichten-Weitverkehr für Ferngespräche, Fernschreiben bzw. Daten erfolgt über ausgedehnte Netze. Diese Netze bestehen aus Knoten mit Vermittlungs- und Steuereinrichtungen, welche über Nachrichtenkanäle untereinander verbunden sind. Die Struktur dieser Netze sowie die Betriebsstrategie haben einen entscheidenden Einfluß auf die Abwicklung des Nachrichtenverkehrs, insbesondere hinsichtlich momentaner Engpässe, welche durch Verkehrsspitzen bzw. Teilausfälle entstehen können.

2.2 Elemente verkehrstheoretischer Modelle

Im folgenden werden die wichtigsten Elemente zur vollständigen Beschreibung eines verkehrstheoretischen Modells zusammengestellt. Sie lassen sich in drei Kategorien einteilen: Strukturelemente, Betriebsstrategien und Prozesse.

2.2.1 Strukturelemente

Die Struktur eines Modells definiert die möglichen "Verkehrswege" von Anforderungen durch das System und gibt ferner die Art, Lage und Anzahl seiner Komponenten ("Strukturelemente") an. Die wichtigsten Strukturelemente sind in Tabelle 1 mit einer kurzen Definition zusammengefaßt. Während des Flusses von Anforderungen durch das System können Elemente wie Quellen, Bedienungseinheiten, Koppelpunkte usw. verschiedene diskrete Zustände einnehmen ("frei" oder "belegt" bzw. "blockiert", "offen" oder "geschlossen" usw.).

2.2.2 Betriebsstrategien

Die Betriebsstrategien eines Modells beschreiben die "Verkehrsregeln", nach welchen die Anforderungen durch das System laufen. Je nach Strukturelement lassen sich verschiedene Arten von Strategien unterscheiden, vergl. Tabelle 2.

2.2.3 Prozesse

Der Nachrichtenverkehr zeichnet sich i.a. durch statistische Eigenschaften aus, wie z.B. die statistisch schwankenden Zeitabstände zwischen den Anforderungen, welche von einer großen Teilnehmerzahl an ein Nachrichtenvermittlungssystem gestellt werden bzw. die Dauern, in welchen von diesen Teilnehmern zentrale Einrichtungen (Leitungen, Register usw.) belegt werden. Im verkehrstheoretischen Modell lassen sich die statistisch schwankenden Eigenschaften durch den "Ankunftsprozess" bzw. den "Belegungsprozess" beschreiben, welcher durch die Wahrscheinlichkeitsverteilungsfunktion (VF) der Ankunftsabstände von Anforderungen bzw. der Belegungsdauern definiert wird. Messungen in realen Systemen haben ergeben, daß die meisten auftretenden Prozesse hinreichend genau durch wenige Standard-Verteilungsfunktionen beschrieben werden können.

Bei Ankunftsprozessen wird unterschieden, ob die Anforderungen von einer endlichen Anzahl q bzw. einer unendlich großen Anzahl von Quellen ($q \to \infty$) erzeugt werden. Dementsprechend wird die VF A(t) der Ankunftsabstände T_A entweder je freie Quelle oder für die Gesamtheit der Quellen definiert:

$$A(t) = P\left\{ T_A \leqq t \right\} . \qquad (1)$$

Ein weiterer wichtiger Fall entsteht, wenn der Ankunftsprozeß von einer unendlich großen Zahl von Quellen erzeugt, bei Erreichen eines bestimmten Systemzustands jedoch "abgeschnitten" wird.

Entsprechend werden die Belegungsdauern T_H durch ihre VF H(t) beschrieben:

$$H(t) = P\left\{ T_H \leqq t \right\} . \qquad (2)$$

Die häufigsten Typen von VF für Prozesse der Nachrichtenverkehrstheorie sind in Tabelle 3 zusammengestellt.

2.3 Beispiele verkehrstheoretischer Modelle

Die Modellbildung werde anhand dreier charakteristischer Beispiele aus Vermittlungssystemen, Rechnersystemen bzw. eines Netzknotens demonstriert. Es wird in jedem Beispiel außerdem kurz auf die Fragestellungen eingegangen, welche durch die Analyse beantwortet werden sollen.

2.3.1 Mehrstufige Koppelnetzwerke in Nachrichtenvermittlungssystemen

Bei der Durchschaltung von Verbindungen zwischen den Teilnehmern bzw. zwischen inneren Verbindungssätzen und zentralisierten Registern werden häufig mehrstufige Koppelanordnungen (Linksysteme) verwendet. Dabei werden die Verbindungen über mehrere Koppelmatrizen (Koppelvielfache) konjugiert durchgeschaltet.

Bild 1 zeigt ein Beispiel für ein Strukturmodell eines 4-stufigen Linksystems mit g_j Koppelmatrizen in Stufe j, j = 1,2,3,4. Die Koppelmatrizen aufeinanderfolgender Stufen sind über Zwischenleitungen nach

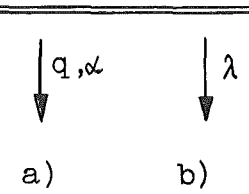
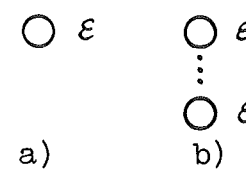
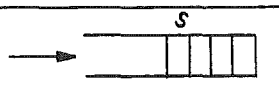
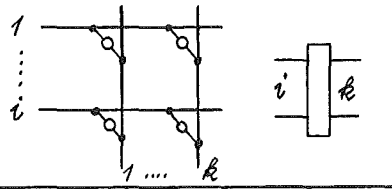
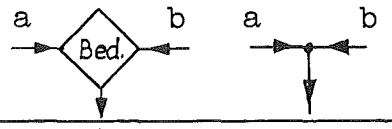
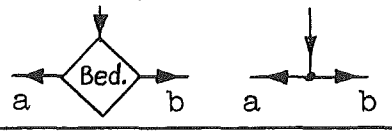
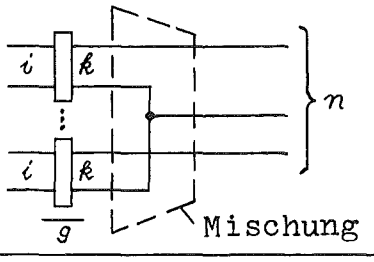
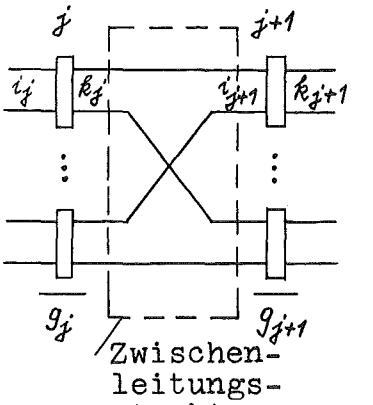
SYMBOL	ART	DEFINITION
	Verkehrs-quellen	Erzeugung von Anforderungen entsprechend eines "Ankunftsprozesses" a) $q < \infty$ Verkehrsquellen, Ankunftsrate α je freie Verkehrsquelle b) $q \to \infty$ Verkehrsquellen, gesamte Ankunftsrate λ
	Bedienungs-einheiten	Bedienung von jeweils einer Anforderung entsprechend eines "Belegungsprozesses" a) n=1 Bedienungseinheit, Enderate ε b) n>1 Bedienungseinheiten (Bündel, Gruppe), Enderaten $\varepsilon_1,\ldots,\varepsilon_n$
	Warte-speicher	Speicherung von max. s Anforderungen (Warteschlange)
	Koppelpunkt Schalter	Durchschalteelement für Anforderungen
	Koppel-matrix Koppel-vielfach	Element zur Durchschaltung zwischen i Eingängen und k Ausgängen. Jede Durchschaltung erfolgt über einen Koppelpunkt zwischen einem bestimmten Eingang und Ausgang
	Zusammen-führung	Element zur bedingten Zusammenführung des Verkehrs aus a bzw. b (Betriebsstrategie regelt Priorität)
	Verzweigung	Element zur bedingten Verzweigung des Verkehrs nach a bzw. b (Betriebsstrategie regelt Richtung)
	Mischung	Schema zur Zusammenführung (Konzentration) von Leitungen im Mischungsverhältnis (g.k):n > 1. Beschreibung durch Matrix MS[1:k, 1:g], wobei MS[x,y] = Nummer der Leitung an Ausgang x der Koppelmatrix y.
	Zwischen-leitungs-struktur	Schema zur Verbindung zwischen Ausgängen von Koppelmatrizen der Stufe j mit den Eingängen von Koppelmatrizen der Stufe j+1 ("Zwischenleitungen" oder "Linkleitungen") Verschiedenartige Verbindungsschemata dabei möglich: "geordnet aufgelegt", "zyklisch vertauscht aufgelegt", ferner auch Konzentration(Mischung) zwischen den Stufen möglich.

Tabelle 1. Strukturelemente verkehrstheoretischer Modelle

Tabelle 2. Betriebsstrategien verkehrstheoretischer Modelle

ART	Auswahlstrategien innerhalb einer Gruppe von Bedienungseinheiten	Auswahlstrategien innerhalb einer Warteschlange	Auswahlstrategien bei der Zusammenführung des Verkehrs	Auswahlstrategien bei der Verzweigung des Verkehrs
DEFINITION	Auswahl einer von mehreren freien Bedienungseinheiten aus einer Gruppe von Bedienungseinheiten	Auswahl einer von mehreren wartenden Anforderungen innerhalb einer Warteschlange	Auswahl einer von mehreren Herkunfts-Richtungen bzw. Herkunfts-Warteschlangen	Auswahl einer von mehreren Ziel-Richtungen bzw. Ziel-Warteschlangen
TYPISCHE BEISPIELE	- Sequentielles Absuchen mit/ohne Nullstellung - Zufällige Auswahl - Auswahl einer bestimmten Bedienungseinheit (Punktwahl) - Punktwahl mit mehrfachen Wiederholversuchen bei jeweils anderen Bedienungseinheiten	- Ankunftsreihenfolge (first-in, first-out FIFO) - Zufällige Auswahl (RANDOM) - Inverse Ankunftsreihenfolge (last-in, first-out LIFO) - Reihenfolge nach kürzester Bedienungszeit (shortest-job-first SJF) - Auswahl nach Prioritätsklassen (unterbrechende Priorität, nichtunterbrechende Priorität)	- Ankunftsreihenfolge - Zufällige Auswahl - Auswahl nach vorgegebenen Wahrscheinlichkeiten - Auswahl zyklisch fortschreitend - Auswahl nach momentanen Warteschlangenlängen - Auswahl nach Prioritätsklassen (unterbrechende Priorität, nichtunterbrechende Priorität, alternierende Priorität, Unterbrechungs-Distanz-Priorität, Unterbrechungs-Verzögerungs-Priorität)	- Auswahl einer bestimmten Richtung - Zufällige Auswahl einer Richtung - Sequentielles "Absuchen" von Richtungen nach dem "Überlaufprinzip" (Überlauf von Primär- auf Sekundärbündel, Primärspeicher auf Sekundärbündel, Primärspeicher auf Sekundärspeicher) - Auswahl nach Prioritätsklassen - Auswahl zyklisch fortschreitend

ART	DEFINITION
negativ-exponentielle VF	$P\{T \leq t\} = 1 - \exp(-\mu t)$, Mittelwert $E[T] = \frac{1}{\mu}$
konstante VF	$P\{T \leq t\} = \begin{cases} 0 & \text{für } 0 \leq t < 1/\mu \\ 1 & \text{für } t \geq 1/\mu \end{cases}$, $E[T] = \frac{1}{\mu}$
Erlang-k- VF	$P\{T \leq t\} = 1 - \exp(-k\mu t)\sum_{i=0}^{k-1} \frac{(k\mu t)^i}{i!}$, $E[T] = \frac{1}{\mu}$
Hyper-exponentielle VF k-ter Ordnung	$P\{T \leq t\} = 1 - \sum_{i=1}^{k} p_i \exp(-\mu_i t)$, $E[T] = \frac{1}{\mu} = \sum_{i=1}^{k} p_i \frac{1}{\mu_i}$ und $\sum_{i=1}^{k} p_i = 1$

Tabelle 3. Häufig auftretende Prozesse verkehrstheoretischer Modelle

einer vorgebbaren Zwischenleitungsstruktur ("Verdrahtungsschema") verbunden. Zwischen Stufe 1 und 2 ist zusätzlich noch eine Konzentration (Mischung) vorhanden. Die Koppelpunkte und Zwischenleitungen, welche an einer durchgeschalteten Verbindung beteiligt sind, bleiben während der gesamten Belegungsdauer (Gesprächsdauer bzw. Registerbelegungszeit) belegt. (Vergleiche durchgezogene Belegung in Bild 1).

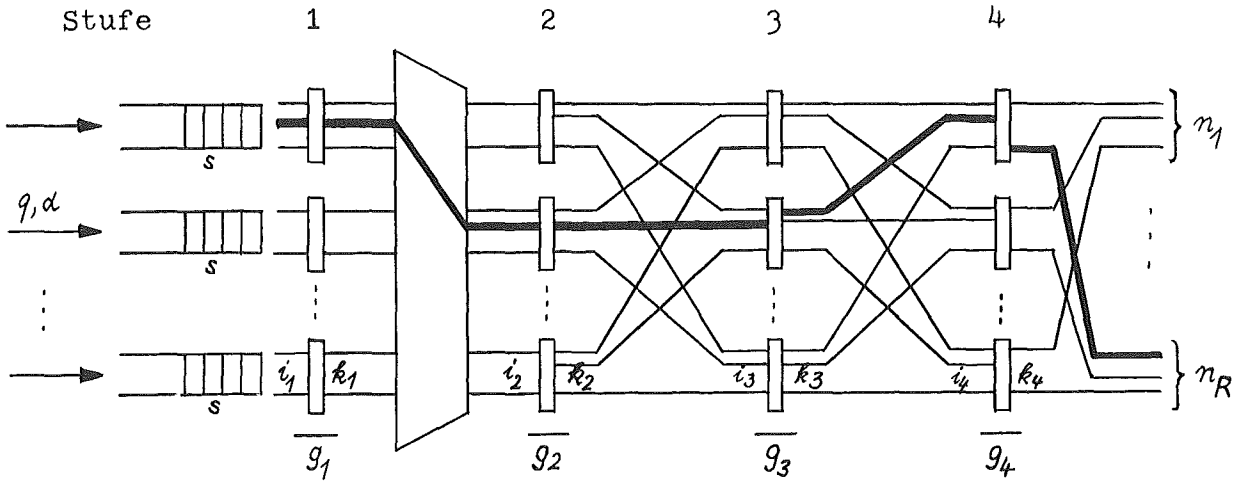


Bild 1. Mehrstufiges Koppelnetzwerk (Linksystem)

Am Ausgang des Linksystems erfolgt eine Verkehrsverzweigung in R Richtungen mit n_r Abnehmerleitungen (= Bedienungseinheiten), r=1,2,...,R. Die Anforderungen werden von g_1 Gruppen aus jeweils q Verkehrsquellen (Ankunftsrate α je freie Quelle) erzeugt. Je Gruppe ist ein Wartespeicher der Kapazität s vorgesehen.

Als Betriebsstrategien kommen verschiedene Auswahlstrategien für Bedienungseinheiten, Warteschlangen bzw. Anforderungen innerhalb von Warteschlangen in Frage, vergl. 2.2.2. Die Ankunftsabstände sind i.a. negativ-exponentiell, die Belegungsdauern entweder ebenfalls negativ-exponentiell (Gespräche) oder Erlang-k-verteilt (Aufnahme und Verarbeitung von Wählziffern im Register).

Als wichtigste Kenngrößen des Systems sind Warte- bzw. Verlustwahrscheinlichkeiten, mittlere Warteschlangenlängen, Mittelwerte und VF der Wartezeiten sowie die einzelnen Bündelbelastungen zu ermitteln.

2.3.2 Multiprogramming in Rechnersystemen

Eine wirkungsvolle Ausnutzung von Prozessoren (P) und E/A-Kanälen in Rechnersystemen ist durch Simultanarbeit und Konkurrenz von mehreren Programmen oder Programmteilen (Segmente, Seiten) im Hauptspeicher möglich. In Bild 2 ist ein Warteschlangenmodell angegeben, welches die verkehrstheoretisch wesentlichsten Teile eines Rechnersystems mit Multiprogramming und Paging wiedergibt.

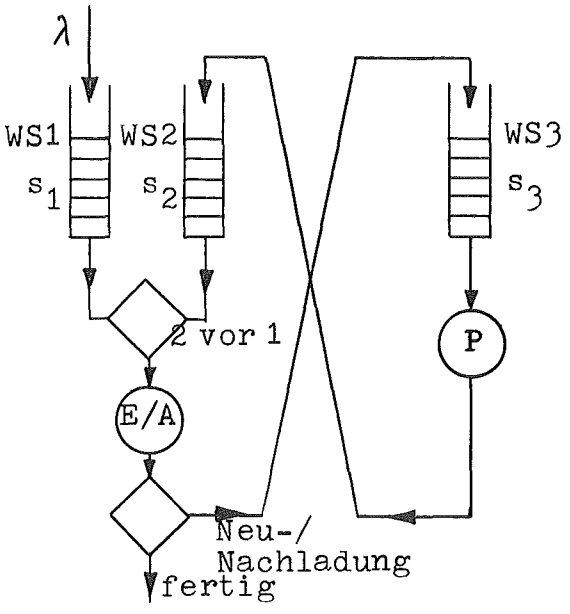


Bild 2. Multiprogramming-Rechnermodell

Neue Anforderungen (Programme) werden mit der Rate λ erzeugt und warten in einem Hintergrundspeicher (WS1), bis sie über den E/A-Kanal in den Hauptspeicher (WS3) mit einer Anfangsladung von Seiten transportiert werden. Der Prozessor P bearbeitet ein Programm bis zur Beendigung bzw. einer Unterbrechung infolge "page fault" und erzeugt - nach einer Verwaltungszeit - eine Anforderung an den E/A-Kanal, welche in WS2 warten kann bis zur Auspeicherung des fertigen Programms bzw. Nachladung einer neuen Seite.

Als Betriebsstrategie müssen zunächst die Prioritätsregel bei der Zusammenführung (WS1, WS2) und die Verzweigungs-Bedingungen (nach E/A-Kanal) angegeben werden (vergl. Bild 2). Als Auswahlstrategien innerhalb der Warteschlangen kommen Prioritäten bzw. die normale Ankunftsreihenfolge in Frage. Die Ankunftsabstände neuer Programme seien z.B. negativ-exponentiell verteilt mit Ankunftsrate λ. Ein in der Praxis jedoch häufig zutreffender Fall ist die Annahme einer "gesättigten" Warteschlange WS1, wofür der Ankunftsprozess ohne Bedeutung ist. Die Belegungsdauern des E/A-Kanals sind ganze Vielfache einer Seitentransferzeit, die Prozessor-Belegungszeiten für ein ganzes Programm bzw. ein Programmteil sind i.a. hyperexponentiell verteilt.

Als interessierende Kenngrößen des Systems werden Mittelwerte und VF der Warte- und Verweilzeiten, Belastungen des Prozessors P, des E/A-Kanals und der einzelnen Wartespeicher sowie die mittlere Zahl von Unterbrechungen je Programm gemessen.

2.3.3 Netzknoten

Nachrichtennetze für den Fernsprech- und Datenverkehr sind nach einer Mischform aus reinen Stern- und Maschennetzen aufgebaut. Dadurch ist ein gewünschtes Ziel auf mehreren Wegen erreichbar und das Netz bei momentanen Engpässen durch Verkehrsspitzen bzw. Teilausfällen noch funktionsfähig. Zur wirtschaftlichen Ausnutzung der Übertragungsleitungen werden jedoch die Direktwege hoch belastet, während die Zweit- und Drittwege den "Überlaufverkehr" aufnehmen und deshalb weniger stark belastet werden dürfen.

In Bild 3 ist ein Ausschnitt aus einem Netz gezeigt, welches u.a. die Knoten i, j und k enthält. Der Verkehr von Knoten i nach j wird zunächst

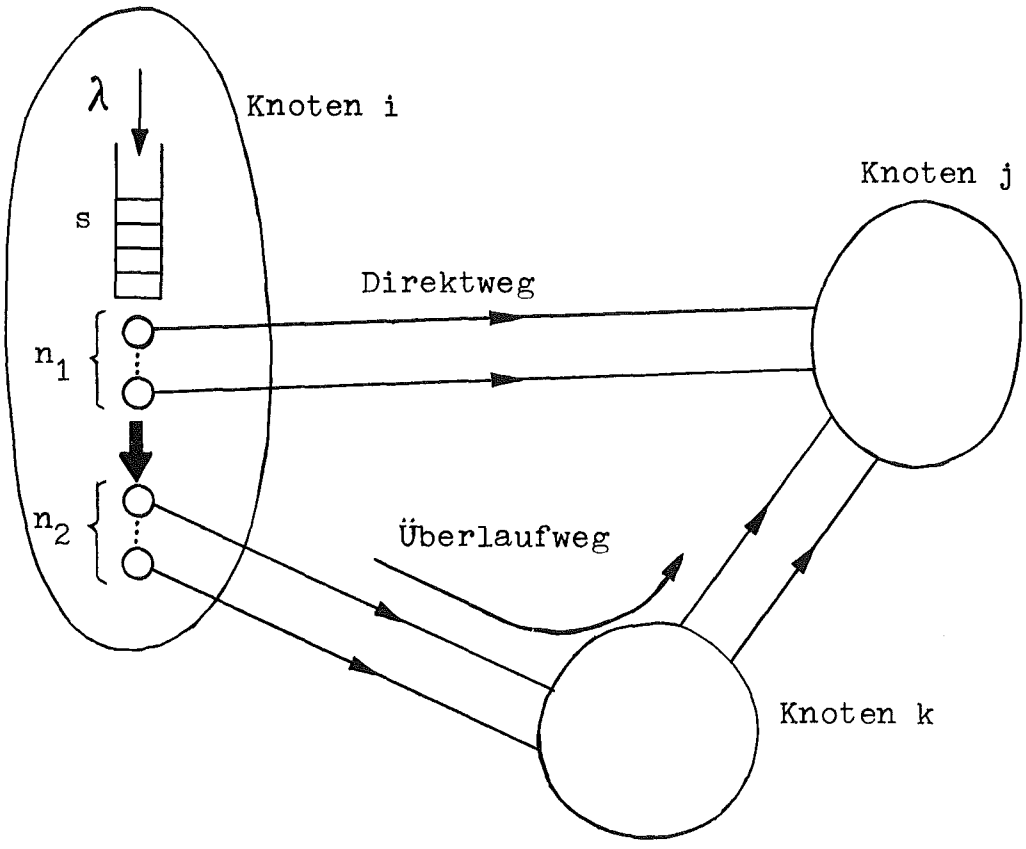


Bild 3. Netzknoten mit Direkt- und Überlaufweg

dem billigeren Direktweg mit n_1 Leitungen (Primärbündel) angeboten. Ist der Direktweg voll belegt, so läuft der Verkehr auf den teureren Überlaufweg mit n_2 Leitungen (Sekundärbündel) über und wird über Knoten k nach Knoten j geführt. Sind beide Wege belegt, so wird ein Platz im Wartespeicher (Kapazität s) belegt. Ist auch der Wartespeicher voll belegt, so wird eine ankommende Anforderung abgewiesen und geht verloren. Die Betriebsstrategien definieren den Überlauf vom Direktweg auf den Überlaufweg, die Auswahl von freien Leitungen innerhalb eines Bündels sowie die Auswahl wartender Anforderungen aus dem Wartespeicher. Die Ankunftsabstände seien z.B. negativ-exponentiell verteilt, die Belegungsdauern konstant (Datenverkehr).

Als Kenngrößen des Systems sind die Belastungen auf Direkt- und Überlaufweg, Warte- und Verlustwahrscheinlichkeit sowie Mittelwert und VF der Wartezeiten von Interesse.

3. SIMULATIONSVERFAHREN

3.1 Beschreibung des Beispiel-Modells

Mit den Elementen aus Abschnitt 2 lassen sich reale Systeme als Modelle nachbilden, wie dies durch die Beispiele in Abschnitt 2.3 angedeutet wurde. Im folgenden soll das Beispiel aus Abschnitt 2.3.3 näher betrachtet werden, um daran
- die Anwendung der Simulationstechnik zu beschreiben und
- Erfahrungen bei der Anwendung von Programmiersprachen zu zeigen (Abschnitt 5).

Bild 4 zeigt nochmals das Modell sowie eine Beschreibung, die sich auf die Struktur, die Betriebsstrategien und die Prozesse bezieht.

Ziel der Simulation ist es nun, dieses Modell so durch ein Simulationsprogramm auf einem Digitalrechner nachzubilden, daß die folgenden Meßgrößen als Ergebnisse gewonnen werden können:

- vor dem Wartespeicher:
  - In der Simulation realisierte Ankunftsrate der Anforderungen
  - Verlustwahrscheinlichkeit
- im Wartespeicher:
  - Wartewahrscheinlichkeit
  - Belastung
  - mittlere Wartezeit
  - Wartezeitverteilung
- in Primärbündel und Sekundärbündel jeweils:
  - Belastung
  - Zustandswahrscheinlichkeiten
  - Varianz der Anzahl gleichzeitig belegter Leitungen

3.2 Wahl des Simulationsverfahrens

Durch das Simulationsverfahren soll vor allem das d y n a m i s c h e Verhalten des Modells (der Ablauf) richtig abgebildet werden. Bild 5 zeigt dazu ein Ablaufdiagramm, welches erkennen läßt, daß es zwei Z e i t p u n k t e gibt, in denen eine Z u s t a n d s ä n d e r u n g stattfindet:

- Eine Anforderung kommt an
  - und belegt eine Leitung im Primärbündel
  - oder belegt eine Leitung im Sekundärbündel
  - oder belegt einen Warteplatz
  - (oder geht "verloren", was den Zustand des Modells nicht verändert)
- Eine Anforderung gibt eine Leitung frei
  - und eine wartende Anforderung rückt nach
  - oder die Leitung bleibt frei

Da der Ablauf des Geschehens im Modell durch diese Zustandsänderungen oder E r e i g n i s s e vollständig beschrieben werden kann, (wobei die Zeitspanne zwischen den einzelnen Zustandsänderungen durch Ankunfts- bzw. Belegungsprozess bestimmt wird,)genügt es, in der Simulation nur zu diesen d i s k r e t e n Zeitpunkten die Vorgänge im Modell nachzubilden (im Gegensatz zur kontinuierlichen Simulation z.B. auf einem Analogrechner). Wenn auch die Belegungsdauern negativ exponentiell verteilt wären, würde es zur Messung von Mittelwerten sogar genügen, nur die Wahrscheinlichkeiten für das Eintreten einer Zustandsänderung zu berücksichtigen (Monte-Carlo-Methode, Roulette-Methode,/1/). Im betrachteten Modell muß jedoch

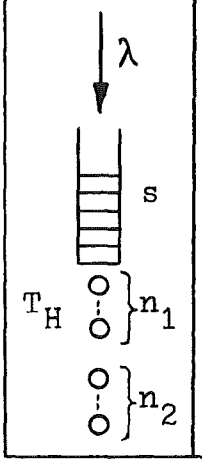
MODELL	MODELLBESCHREIBUNG		
	STRUKTUR	BETRIEBSSTRATEGIEN	PROZESSE
	Verkehrsquellen $q \to \infty$	Verlust, wenn alle Warteplätze belegt sind	Ankunftsrate λ Ankunftsabstände negativ-expon. verteilt
	Wartespeicher mit s Warteplätzen	Auswahl einer wartenden Anforderung nach FIFO	
	Primärbündel n_1 Leitungen	Absuchen sequentiell mit Überlauf von Primärbündel auf Sekundärbündel	Belegungsdauern im Primärbündel u.Sekundärbündel konstant $T_H = 1/\varepsilon$
	Sekundärbündel n_2 Leitungen		

Bild 4. Modellbeschreibung für das Simulationsbeispiel

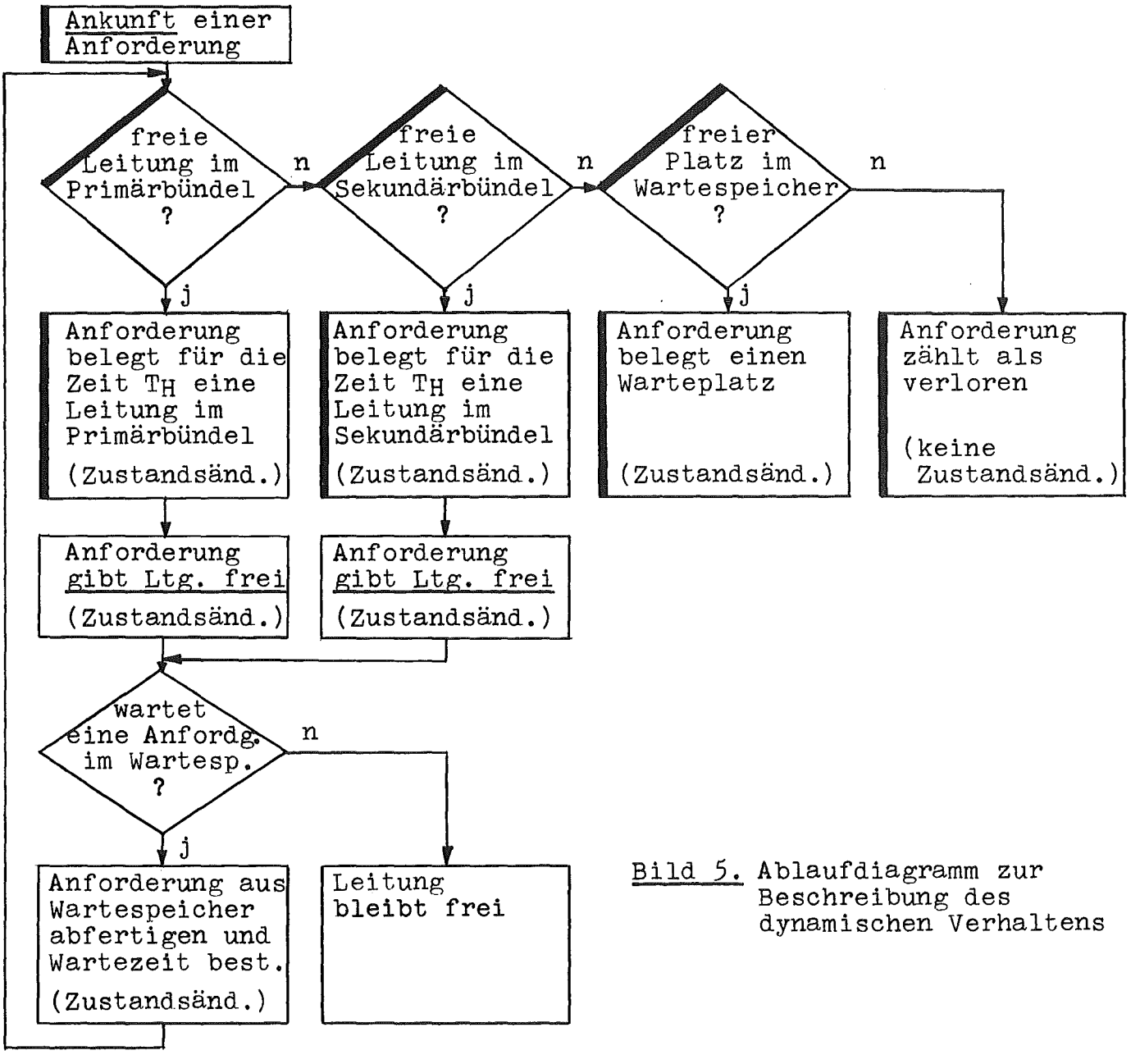


Bild 5. Ablaufdiagramm zur Beschreibung des dynamischen Verhaltens

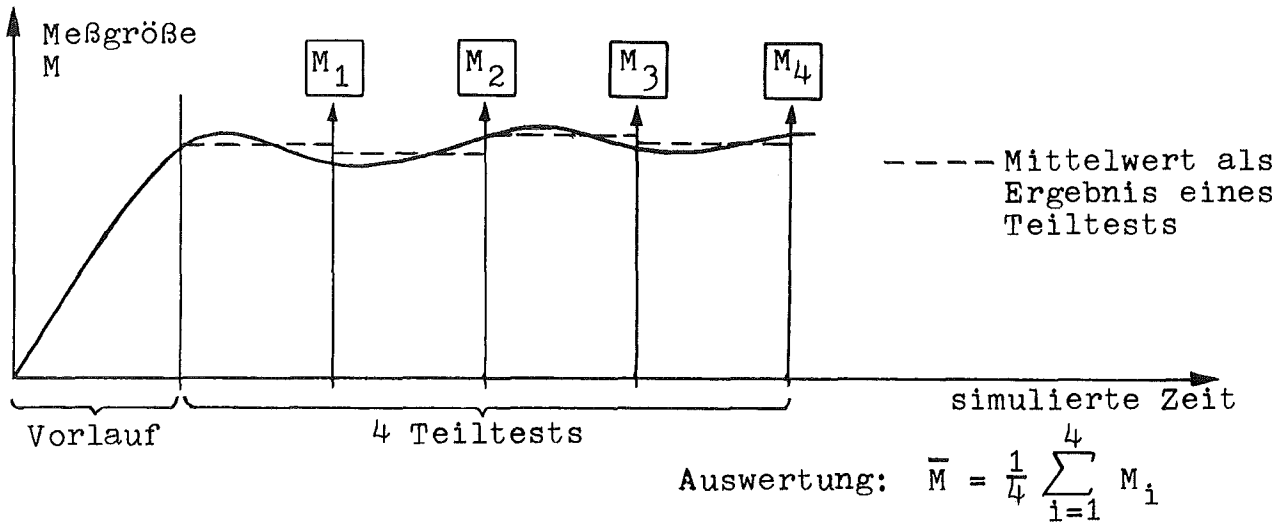


Bild 6. Auswertung einer Meßgröße

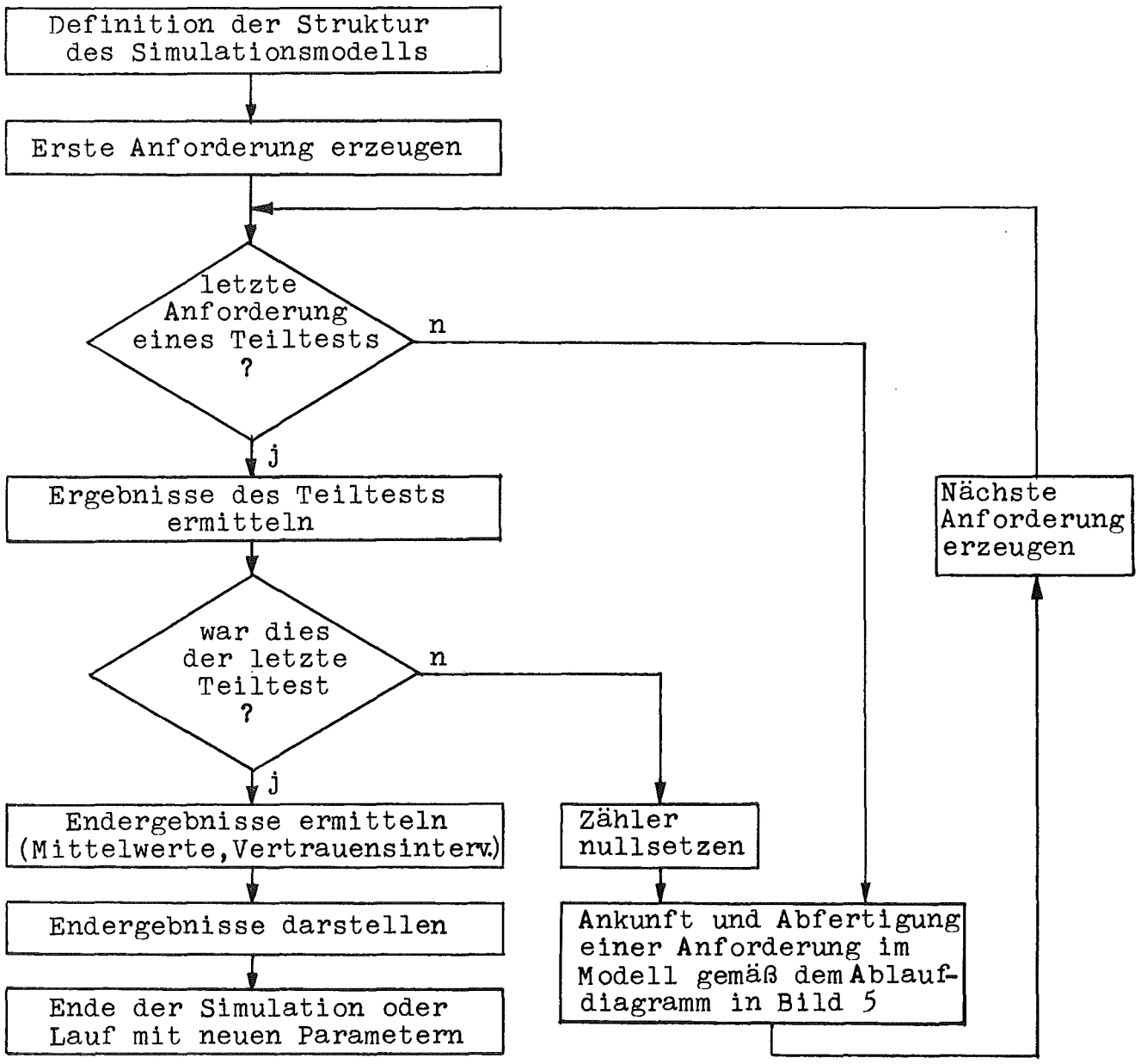


Bild 7. Ablaufdiagramm für das Rahmenprogramm

wegen der konstanten Belegungsdauern die z e i t t r e u e (event by event) Simulationsmethode /2/ angewandt werden, zumal außerdem noch individuelle Wartezeiten für die Berechnung der Wartezeitverteilung zu messen sind.

3.3 Durchführung und Auswertung der Simulation

Da die Zustandsänderungen im Modell durch die Ankunft und Abfertigung von Anforderungen vor sich gehen, wird sich erst nach einer gewissen Zeit (VORLAUF) ein stationärer Zustand im Modell einstellen, vgl. Bild 6 .Eine Aussage über die statistische Sicherheit, mit der das Endergebnis $\overline{M}$ (Mittelwert) einer Meßgröße ermittelt wird, läßt sich durch eine Unterteilung der Simulation in TEILTESTS machen. Aus den Teiltestergebnissen kann das Vertrauensintervall für die Meßgrößen errechnet werden (vgl./3/ bis /5/). Die Durchführung von VORLAUF, TEILTESTS und Auswertung erfolgt im sogenannten Rahmenprogramm (vgl. /6/,/7/), das in Bild 7 als Ablaufdiagramm dargestellt ist.

Im betrachteten Modell ist die Zahl der Quellen unendlich groß. Daher ist die Ankunftsrate der Anforderungen in jedem Augenblick gleich λ und der negativ-exponentiell verteilte Ankunftsabstand T_A ist mathematisch sehr einfach darstellbar als

$$T_A = - \frac{1}{\lambda} \ln(z)$$

mit der Zufallsvariablen z , die zwischen 0 und 1 gleichverteilt ist. Viele Beispiele in der Nachrichtenverkehrstheorie weisen jedoch eine endliche Zahl von Quellen auf. Die hierbei in der Simulation auftretenden Probleme werden ausführlich in /8/ diskutiert.

4. GRUNDSÄTZLICHE FORDERUNGEN AN SIMULATIONSSPRACHEN FÜR DIE NACHRICHTENVERKEHRSTHEORIE

In den Abschnitten 2 und 3 wurden die Problemstellungen und die Simulationsmethode erläutert, wie sie für die Nachrichtenverkehrstheorie typisch sind. Daraus ergeben sich die im folgenden aufgezählten Forderungen an eine "ideale" Simulationssprache (IS) für die Durchführung einer diskreten,zeittreuen Simulation gemäß Abschnitt 3.2.

4.1 Modellbeschreibung

Die Elemente der IS sollen den typischen Elementen (Abschnitt 2.2) der Modelle aus der Nachrichtenverkehrstheorie entsprechen und sich bequem miteinander verknüpfen lassen (z.B. bei Zwischenleitungsanordnungen). Variablennamen kann der Benutzer beliebig festlegen. M a k r o b e f e h l e sollen die Nachbildung der Betriebsstrategien sowie der Prozesse gestatten. Die simulierte Zeitachse ist durch ein selbständiges Zeitkontroll-Programm zu verwalten (Überwachung der Ereignisfolge), wobei Zeitgrößen als Gleitkommazahlen gelten.

4.2 Logische und arithmetische Operationen,Statistik

In der IS sind ganze Zahlen, Gleitkommazahlen und Boolesche Ausdrücke als Variable zuzulassen. In einer Funktions-Bibliothek sind die häufig benutzten Funktionen (z.B.ln,Wurzelfunktion) bereitzustellen.Für die gebräuchlichsten Verteilungsfunktionen sollen Prozeduren zur Verfügung stehen(vgl.Abschnitt 2.2.3).Die in der Simulation gewonnenen Meßdaten sind ohne großen Programmieraufwand statistisch auszuwerten (Mittelwerte,Vertrauensintervalle,Histogramme).

4.3 Eingabe, Ausgabe

Die Eingabe soll ohne ein vorgeschriebenes Format erfolgen. Die Anfangswerte und die Struktur des Modells seien einfach zu ändern. Für die Gestaltung der Drucker-Ausgabe (oft Tabellen) sollten bequeme Anweisungen zur Verfügung stehen und Ausgabebefehle für das Ablegen von Daten auf externen Speichern (Band, Platte) sollten nicht fehlen.

4.4 Testmöglichkeiten

Syntaktische und logische Fehler sowie eine falsche Reihenfolge im Ablauf der Ereignisse sind durch "Momentaufnahmen" (Ausdrucken der Werte aller Variablen) oder Prüfroutinen sichtbar zu machen. Der Compiler soll über eine detaillierte Fehlerdiagnose verfügen, und im Benutzerhandbuch ist die Abhilfe im Fehlerfall allgemeinverständlich zu erläutern.

4.5 Sonstiges

Die IS soll leicht erlernbar **sein**, sparsam mit Rechenzeit und Speicherplatz umgehen und eine übersichtliche Programmierung ermöglichen, welche gleichzeitig eine Modell-gerechte Formulierung des Simulationsproblems erlaubt.

5. ERFAHRUNGEN MIT 4 PROGRAMMIERSPRACHEN

Im folgenden Abschnitt wird gezeigt, welche Erfahrungen bei der Erstellung eines zeittreuen Simulationsprogrammes mit den Programmiersprachen FORTRAN IV , GPSS/360 , SIMSCRIPT I.5 und SIMULA 67 gemacht wurden. Als Simulationsmodell diente das Beispiel nach Bild 4 . Es sollen dabei weder die Simulationsprogramme (vgl. /9/ bis /12/) noch die Programmiersprachen (vgl. /13/ bis /21/) im einzelnen ausführlich erklärt werden. Für einen grundlegenden Vergleich von Programmiersprachen sei auf die Literaturstellen /22/ bis /32/ verwiesen.

5.1 Allgemeines über FORTRAN, GPSS, SIMSCRIPT, SIMULA

Während FORTRAN (wie ALGOL) eine universelle problemorientierte Programmiersprache darstellt, zählen die Simulationssprachen GPSS, SIMSCRIPT und SIMULA zu den speziellen problemorientierten Programmiersprachen und erfüllen in einigen Teilen die in Abschnitt 4 gestellten Forderungen. Insbesondere besitzen die drei letztgenannten Programmiersprachen einige Sprachelemente zur Beschreibung typischer Modellelemente: In Tabelle 4 ist eine Zusammenstellung einiger wichtiger Merkmale für die 3 Simulationssprachen angegeben. Bei der Beschreibung der Anwendung der einzelnen Sprachen werden diese Merkmale näher erläutert (Abschnitte 5.3 bis 5.5) .

Da nicht nur die Leistungsfähigkeit, sondern auch die Verfügbarkeit einer Programmiersprache über ihren Einsatz entscheidet, sind in Tabelle 5 einige Compiler-Hersteller für GPSS, SIMSCRIPT und SIMULA genannt. Ein FORTRAN-Compiler ist sicherlich für nahezu jeden Digitalrechner erhältlich. Für das Simulationsbeispiel wurden Compiler der Anlagen IBM 360/40 (GPSS/360) sowie CD 6600 (FORTRAN IV, SIMSCRIPT I.5 , SIMULA67) verwendet.

Tabelle 4. Einige Merkmale von GPSS, SIMSCRIPT, SIMULA

M E R K M A L	G P S S	S I M S C R I P T	S I M U L A
- bewegliche Einheit (z.B. Anforderung)	TRANSACTION	TEMPORARY ENTITY	PROCESS } Exemplar einer ACTIVITY
- Bedienungseinheit (z.B. Leitungsbündel)	FACILITY (1Platz) STORAGE (>1Platz)	PERMANENT ENTITY	PROCESS } Exemplar einer ACTIVITY
- Eigenschaft einer Einheit (z.B. Zeitpunkt, zu dem eine Anforderung in einen Wartespeicher gelangt)	PARAMETER	ATTRIBUTE	ATTRIBUTE
- Gruppe, in die Einheiten eingeordnet werden können (z.B. Wartespeicher)	CHAIN	SET	SET
- Ereignistyp	abhängig vom Block, den die TRANSACTION durchläuft	festgelegt im EVENT, einem abgeschlossenen Programmteil	festgelegt durch Operationsregeln im PROCESS, die in seiner aktiven Phase ausgeführt werden
- Zeitkontrolle	blockierte oder aktive TRANSACTION wird in internen CHAINS organisiert	EVENTS werden durch Anweisungen in den internen Kalender eingetragen	SEQUENCING SET zur Organisation der aktiven Phasen der PROCESSES

G P S S			S I M S C R I P T			S I M U L A	
Version/Herst./Serie			Version/Herst./Serie			Hersteller /	Serie
II	UNIVAC	1107 1108	?	PHILCO-FORD	210 211 212	BURROUGHS	B5500
III	CONTROL DATA CORP.	3600	I	IBM	7090 7094	CONTROL DATA CORP.	6400 6500 6600
III	IBM	7090 7094 7040 7044	I.5	CONTROL DATA CORP.	3600 3800 6400 6500 6600	IBM	360/ 370/
/360	IBM	360/ 370/	I.5	GENERAL ELECTRIC	625 635	UNIVAC	1107 1108
			I.5	IBM	360/		
			I.5	RCA Info-Systems	Spectra 70 ab Mod.45		
			I.5	UNIVAC	490 1107 1108		
			II	IBM	360/		

Tabelle 5. Compiler für GPSS, SIMSCRIPT, SIMULA (nach /32/)

5.2 Anwendung von FORTRAN IV auf das Simulationsbeispiel

Da in FORTRAN keine Sprachelemente vorhanden sind, welche typische Modellelemente (Wartespeicher, Bedienungseinheiten) beschreiben, muß das Modell mit Hilfe von indizierten Variablen nachgebildet werden, die z.B. den Belegungszustand und die Belegungsstatistik festhalten. Die negativ-exponentiell verteilten Ankunftsabstände werden durch gleichverteilte Pseudozufallszahlen (vgl. Abschnitt 3.3) nachgebildet, die bei FORTRAN durch eine Bibliotheksfunktion aufgerufen werden können. Der Zeitablauf wird über ein Feld gesteuert, in welches jeweils das als nächstes anstehende Ereignis eingetragen wird. Die typischen Vorgänge im Verlauf der Simulation (Suchen einer freien Leitung, Belegen eines Warteplatzes, Auswerten eines Teiltests) können durch SUBROUTINES und FUNCTIONS nachgebildet werden, wodurch das Programm eine gewisse Übersichtlichkeit erhält. Im wesentlichen entspricht das Ablaufdiagramm des FORTRAN-Programms den in Bild 5 und Bild 7 gezeigten Ablaufdiagrammen. Die Beschreibung der Vorgänge in der Simulation durch jeweils mehrere Anweisungen bildet allerdings einen Nachteil für die Lesbarkeit des Programms.

Die Eingabe erfolgt bei FORTRAN IV formatgebunden, die Gestaltung der Ausgabe (z.B. Tabellen) ist für die Zwecke der Simulation etwas unhandlich. Logische und syntaktische Fehler werden ausführlich analysiert. Die Kontrolle des zeitlichen Ablaufs ist durch Druckbefehle möglich. FORTRAN IV ist gut dokumentiert und leicht erlernbar. Programmlaufzeit und Speicherplatzbedarf : siehe Abschnitt 5.6 .

5.3 Anwendung von GPSS/360 auf das Simulationsbeispiel

5.3.1 Modellbeschreibung bei GPSS/360

Die Simulationssprache GPSS/360 bildet das Modell und die darin ablaufenden Ereignisse durch ein Ablaufdiagramm nach, dessen Blöcke jeweils mit einem Makrobefehl beschrieben werden. Jeder Block ist durch eine Reihe von Parametern in seiner Wirkung festgelegt. Diese Wirkung übt der Block auf die beweglichen Teile des Modells, die TRANSACTIONS aus, sobald sie ihn durchlaufen bzw. in ihm verweilen (z.B. eine Anforderung belegt eine Leitung). Jede TRANSACTION kann sich zu einem bestimmten Zeitpunkt nur an e i n e r Stelle im Ablaufdiagramm aufhalten. TRANSACTIONS können in Wartespeicher eingereiht werden und unter vorschreibbaren Bedingungen diesen wieder verlassen. Eine TRANSACTION ist durch maximal 100 PARAMETER beschreibbar.

Der Ablauf der Ereignisse im Modell hängt bei GPSS/360 vom Zustand aller im Modell befindlichen TRANSACTIONS ab, wobei stets eine bestimmte TRANSACTION soweit durch das Ablaufdiagramm transportiert wird, bis sie entweder einen ADVANCE-Block erreicht (Belegungsdauer) oder z.B. aufgrund eines bereits belegt angetroffenen FACILITY-Blocks (Bedienungseinheit) am Weiterkommen gehindert wird. Dann sucht die Programmsteuerung die nächste bewegbare TRANSACTION im Modell.

Einige Sprachelemente, die im Simulationsbeispiel Verwendung fanden:

GENERATE-Block	-erzeugt die Anforderungen (TRANSACTIONS) in zeitlichem Abstand entsprechend einer vorgebbaren FUNCTION (Wertepaare, welche die Stützstellen beschreiben.
STORAGE	-mehrere Bedienungseinheiten, 1 Leitungsbündel
QUEUE	-Wartespeicher, in dem die vor einem STORAGE wartenden Anforderungen Platz finden.
TEST-Block	-Abfrage.
TERMINATE	-Löschen einer Anforderung nach Beendigung ihres Durchlaufs.

5.3.2 Logische und arithmetische Operationen, Statistik bei GPSS/360

Bezüglich der arithmetischen Operationen entstehen für das Simulationsbeispiel dadurch Schwierigkeiten, daß bei GPSS/360 Variable eine Boolesche Größe darstellen oder nur ganzzahlige Werte annehmen können. Dies bedeutet, daß alle Ergebnisse, die sich auf Wahrscheinlichkeiten beziehen, bei der Berechnung z.B. mit dem Faktor 10^5 multipliziert werden müssen. Hierdurch darf allerdings (bei Zwischenrechnungen) nicht die o b e r e Zahlenbereichsgrenze von $2^{31}-1$ überschritten werden. Ein weiterer Nachteil liegt darin, daß wichtige Bibliotheksfunktionen fehlen: Für die Berechnung des Vertrauensintervalls wird die Wurzelfunktion benötigt, die bei GPSS/360 durch Stützstellen einzugeben ist (Polygonzug) . Sehr vorteilhaft ist die automatische Erstellung von statistischen Daten bei Bedienungseinheiten und Wartespeichern. Z.B. werden für einen Wartespeicher die folgenden 8 Daten ermittelt:

- Belastung
- Gesamtzahl der TRANSACTIONS, die gewartet haben
- Gesamtzahl der TRANSACTIONS, die nicht warten mußten
- Wahrscheinlichkeit, daß nicht gewartet werden muß
- mittl. Wartezeit aller durchgelaufenen TRANSACTIONS
- mittl. Wartezeit aller wartenden TRANSACTIONS
- momentante Anzahl von wartenden TRANSACTIONS
- maximale Anzahl von wartenden TRANSACTIONS

5.3.3 Eingabe, Ausgabe bei GPSS/360

Die wesentlichen Eingabedaten zur Beschreibung des Simulationsbeispiels sind im Simulationsprogramm in aufeinanderfogenden Karten in einem gesonderten Programmteil (MACRO) zusammengefaßt worden. Für die Ausgabe auf einem Drucker stehen bei GPSS/360 zwei Typen von Befehlen zur Verfügung:
- Auswahl, Betiteln und Kommentieren von Statistiken
- graphische Ausgabe (Diagramme)

5.3.4 Testmöglichkeiten bei GPSS/360

Die automatisch erstellten Statistiken erleichtern das Austesten wesentlich. Die meisten Programmierfehler werden außerdem durch den Compiler analysiert und durch eine Codenummer gekennzeichnet.

5.3.5 Sonstiges bei GPSS/360

Der kompakte übersichtliche Aufbau von GPSS/360 trägt zur schnellen Erlernbarkeit der Sprache bei. Der Übergang vom Ablaufdiagramm zum Programm ist direkt durchführbar. Die Lesbarkeit des Programms leidet allerdings ein wenig darunter, daß der Programmierer an die vorgeschriebenen Parameternamen gebunden ist und nicht eigene Parameterbezeichnungen Modell-bezogen wählen kann. Programmlaufzeit und Speicherplatzbedarf: siehe Abschnitt 5.6

5.4 Anwendung von SIMSCRIPT I.5 auf das Simulationsbeispiel

5.4.1 Modellbeschreibung bei SIMSCRIPT I.5

Die Struktur dieser Simulationssprache erfordert eine klare Aufteilung des Modells in einen statischen Teil (definiert durch PERMANENT ENTITIES und deren ATTRIBUTES) und einen dynamischen Teil, d.h. Programmabschnitte, welche die Durchführung von Zustandsänderungen gestatten (EVENTS). Für das Simulationsbeispiel werden gemäß Abschnitt 3.2 zwei EVENTS benötigt:
- Ankunft einer Anforderung mit der Abfrage, ob sie sofort abgefertigt werden kann oder wartet oder verloren geht;
- Freigabe einer Leitung mit der Abfrage, ob eine Anforderung wartet und diese Leitung aufs neue belegt.

Zur Organisation der simulierten Zeitachse steht ein "innerer Kalender" zur Verfügung, in den der Zeitpunkt für das Stattfinden eines Ereignisses entweder schon zu Beginn der Simulation (EXOGENOUS EVENT) oder erst im Verlauf der Simulation in Abhängigkeit von gewissen Bedingungen eingetragen wird (ENDOGENOUS EVENT). Das jeweilige Ereignis findet statt, indem die Anweisungen des betreffenden Programmteils ausgeführt werden. Als "bewegliche" Elemte des Modells werden TEMPORARY ENTITIES erzeugt, die durch TEMPORARY ATTRIBUTES beschrieben werden und sich in Gruppen (SETS) einordnen lassen (FIFO, LIFO oder abhängig vom Zahlenwert eines TEMPORARY ATTRIBUTE). Die Definition aller Größen (außer der lokalen Variablen) und eine Aufteilung in TEMPORARY VARIABLES, PERMANENT VARIABLES, SETS und FUNCTIONS erfolgt anhand eines DEFINITION-Formulars, in dem auch Angaben über den Speicherbedarf jeder Variablen (Ganzworte, Halbworte, Drittelworte) zu machen sind.

Einige Sprachelemente, die im Simulationsbeispiel Verwendung fanden:

CREATE ruf - erzeugt eine Anforderung (TEMPORARY ENTITY), die unter dem Namen "ruf" angesprochen werden kann.

CAUSE ankft AT 2.0 - veranlaßt, daß das Ereignis "Ankunft einer Anforderung" für den Zeitpunkt 2.0 in den inneren Kalender eingetragen wird.

FILE ruf IN wart - ordnet die TEMPORARY ENTITY "ruf" in den Wartespeicher "wart" ein.

DESTROY ruf - zerstört die TEMPORARY ENTITY "ruf" und deren TEMPORARY ATTRIBUTES. Hierdurch werden die betreffenden Speicherzellen frei für andere, neue TEMPORARY ENTITIES (dynamische Speicherplatzverwaltung).

5.4.2 Logische und arithm. Operationen, Statistik bei SIMSCRIPT I.5

Die Verwandtschaft von SIMSCRIPT und FORTRAN zeigt sich darin, daß die Möglichkeiten für arithmetische Operationen in beiden Sprachen gleich sind, allerdings mit der Ausnahme, daß SIMSCRIPT keine Boolesche Variablen kennt. Zur Herstellung von statistischen Daten gibt es bei SIMSCRIPT Anweisungen, welche die laufende Integration von Meßwerten über der Zeit sowie die Bildung von z.B. Mittelwert, Varianz oder Standardabweichung aus einer Werteserie erlauben. Als Funktionen stehen die in der FORTRAN-Unterprogramm-Bibliothek genannten zur Verfügung. Zusätzlich können eigene FUNCTIONS definiert oder Funktionsverläufe durch Wertepaare (Stützstellen) eingegeben werden.

5.4.3 Eingabe, Ausgabe bei SIMSCRIPT I.5

Die Werte aller PERMANENT VARIABLES zu Beginn der Simulation und damit auch der Anfangszustand des Modells werden in einem INITIALIZATION-Formular festgelegt. Das vorgeschriebene Format erfordert dabei einen erhöhten Aufwand beim Erstellen der Eingabedaten. Die Gestaltung der Ausgabe auf einem Drucker wird in "reports" festgelegt, die eine übersichtliche Formulierung der Druckbefehle gestatten. Allerdings benötigen die vom Compiler erzeugten zugehörigen Hilfsprogramme einen verhältnismäßig großen Speicherplatz. Schreibbefehle für den Transport von Daten auf externe Speicher sind vorhanden.

5.4.4 Testmöglichkeiten bei SIMSCRIPT I.5

Da das Simulationsprogramm aus selbständigen Teilen (EVENTS, SUBROUTINES) besteht, kann kein geschlossenes Ablaufdiagramm für den gesamten zeitlichen Simulationsablauf erstellt werden. Durch Druckanweisungen innerhalb eines EVENT läßt sich jedoch der richtige zeitliche Ablauf überprüfen. Der Compiler druckt beim Übersetzen ausführliche Fehlerangaben aus. Die Fehlerdiagnose während des Simulationslaufs war allerdings im vorliegenden Fall nicht immer zufriedenstellend.

5.4.5 Sonstiges bei SIMSCRIPT I.5

Für FORTRAN-Kenner ist SIMSCRIPT leicht erlernbar. Die freie Namensgebung macht das Programm gut lesbar. In der neueren Version SIMSCRIPT II /15/ sind einige Nachteile, vor allem bezüglich der streng formalen Dateneingabe, behoben. Programmlaufzeit und Speicherplatzbedarf : siehe Abschnitt 5.6 .

5.5 Anwendung von SIMULA 67 auf das Simulationsbeispiel

5.5.1 Modellbeschreibung bei SIMULA 67

Die Elemente im Simulationsmodell, welche während der Simulation Veränderungen erfahren o d e r Veränderungen auslösen, werden durch PROCESSES nachgebildet, die individuell durch ATTRIBUTES gekennzeichnet sind und deren Verhalten durch eine Folge von zugehörigen Anweisungen (Operationsregeln) festgelegt ist. Dabei bewirken die Anweisungen unter anderem, daß der PROCESS sich in einer aktiven Phase (entsprechend einem stattfindenden Ereignis,EVENT) oder in einer passiven Phase befindet. Eine Ablaufsteuerung (SEQUENCING SET) besorgt die zeitliche Abfolge der aktiven Phase bei allen im System befindlichen PROCESSES, wobei immer nur 1 PROCESS aktiv sein kann. PROCESSES mit gleicher Wirkungsweise (d.h. gleicher Folge von Operationsregeln, aber unterschiedlichen ATTRIBUTE-Werten) sind durch ein und denselben Programmabschnitt (ACTIVITY) beschrieben. Die Variable ELEMENT ermöglicht einen Bezug auf einen PROCESS. ELEMENTS können in Gruppen (SETS) eingeordnet werden. Die Vereinbarung der ACTIVITIES,ATTRIBUTES etc. erfolgt (wie bei ALGOL) in einem Vereinbarungsteil jeweils zu Beginn der ineinander verschachtelten Blöcke.

Einige Sprachelemente, die im Simulationsbeispiel Verwendung fanden:

'ACTIVATE''NEW' ruf 'AT' 2.0	- veranlaßt den Aufruf des PROCESS "ruf" (durch 'NEW' ruf) und seine Aktivierung,d.h. die Ausführung seiner Operationsregeln, zum Zeitpunkt 2.0
CURRENT.INTO(wart)	- das ELEMENT des gerade aktiven PROCESS (repräsentiert eine Anforderung) wird in den Wartespeicher gebracht.

Ein individueller PROCESS wird beendet, d.h. Speicherplatz wird wieder frei, sobald in der Ausführung der ihn beschreibenden Operationsregeln die Anweisung 'END' erreicht wird.

5.5.2 Logische und arithm. Operationen, Statistik bei SIMULA 67

Die Entwicklung von SIMULA aus ALGOL 60 ist unter anderem daran erkennbar, daß beide Sprachen dieselben logischen und arithmetischen Operationen aufweisen und die Syntax von ALGOL 60 fast vollständig in SIMULA übernommen wurde. Eine Vielzahl von Verteilungsfunktionen kann in SIMULA 67 direkt über Prozedurnamen aufgerufen werden, wodurch die Nachbildung von Prozessen sehr erleichtert wird. Zur Erstellung statistischer Daten stehen Prozeduren zur Verfügung, welche z.B. Werte über der Zeit integrieren oder Histogramme errechnen.

5.5.3 Eingabe,Ausgabe bei SIMULA 67

Die Eingabe erfolgt völlig formatfrei: einzulesende Daten müssen nur durch mindestens eine Leerstelle getrennt sein. Für die Gestaltung der Ausgabe steht eine große Zahl von Anweisungen zur Verfügung (bequeme Tabellenausgabe).

5.5.4 Testmöglichkeiten bei SIMULA 67

Sowohl während der Übersetzung als auch während des Simulationslaufs werden Fehler ausführlich analysiert und ausgedruckt. Der richtige zeitliche Ablauf kann mit Druckbefehlen an den entscheidenden Programmstellen kontrolliert werden.

5.5.5 Sonstiges bei SIMULA 67

Für ALGOL-Kenner ist SIMULA relativ bequem erlernbar, wobei das Verständnis des Grundkonzeptes dieser Sprache zunächst Schwierigkeiten bereiten kann. Natürlich muß berücksichtigt werden, daß bei SIMULA 67 neben einer Anwendung als Simulationssprache allgemein eine handliche Listenverarbeitung möglich ist, wodurch die Sprache im Ganzen umfangreich wird. Das erstellte Simulationsprogramm ist durch die freie Wahl der Variablennamen und die Blockstruktur der Sprache gut lesbar. Programmlaufzeit und Speicherplatzbedarf: siehe nächster Abschnitt .

5.6 Vergleichsresultate

Zusammenfassend kann bezüglich einer Eignung für Simulationsprobleme in der Nachrichtenverkehrstheorie festgestellt werden, daß die drei Simulationssprachen GPSS, SIMSCRIPT und SIMULA folgendermaßen zu beurteilen sind:

- Strukturelemente sowie der Ablauf der Vorgänge im Modell lassen sich sehr bequem in GPSS nachbilden (direkte Umsetzung des Ablaufdiagramms).
- Betriebsstrategien sind vor allem in SIMSCRIPT günstig zu verwirklichen.
- Prozesse können nur einfach nachgebildet werden, wenn die häufigsten Verteilungsfunktionen zur Verfügung stehen. Der Mangel an Bibliotheksfunktionen bei GPSS ist besonders ungünstig, während SIMULA vorbildlich eine ganze Reihe von Verteilungsfunktionen bereitstellt.
- Die Erstellung statistischer Daten erfolgt bei GPSS automatisch (erhöhter Rechenzeitbedarf), doch können Variable nur ganzzahlige Werte annehmen. SIMSCRIPT und SIMULA stellen ausreichende Hilfsmittel zur Statistikerstellung bereit.
- Für die Eingabe von Daten und Anfangsbedingungen ist die formatfreie Form bei SIMULA besonders hilfreich.
- Alle drei Simulationssprachen verfügen, soweit die im Simulationsbeispiel benutzten Compiler ein solches Urteil zulassen, über eine ausreichende Fehlerdiagnose.
- Die Ausgabe gestaltet sich bei SIMSCRIPT durch report-Formulare sehr einfach und übersichtlich.
- Der Lernaufwand nimmt von GPSS über SIMSCRIPT bis SIMULA zu, doch hängt dies auch vom jeweils zur Verfügung stehenden Handbuch ab.

Zur Feststellung von Programmlaufzeit und Speicherplatzbedarf wurden die in FORTRAN IV, GPSS/360, SIMSCRIPT und SIMULA erstellten Programme mit denselben Eingabeparametern versehen und damit jeweils ein

Simulationslauf durchgeführt. Bild 6 zeigt die Ergebnisse. Der Vergleich mit GPSS/360 ist nicht ganz zutreffend, da hierbei ein anderer Rechner (IBM 360/40) als bei den übrigen drei Programmen eingesetzt wurde. Neben der Wahl des Compilers beeinflussen selbstverständlich auch noch andere Gegebenheiten grundsätzlich das Vergleichsergebnis: die Geschicklichkeit und Erfahrung des Programmierers, die Komplexität des Modells, der Umfang der gewünschten statistischen Auswertung| und die Anforderungen an die Ausgabe der Ergebnisse.

PROGRAMMIER-SPRACHE	SPEICHER-PLATZ (Worte à 60 bit)	ÜBERSETZUNGS-ZEIT (sec)	ZENTRAL-RECHNER-ZEIT (sec)	E/A-ZEIT (sec)	LOCH-KARTEN
FORTRAN IV	14 000	2.0	9.7	13.1	450
GPSS/360	30 000	79.2	2 049.5		340
SIMSCRIPT	33 000	48.5	34.2	58.5	350
SIMULA	17 700	6.3	379.4	19.5	360

Bild 8. Vergleichswerte für das Simulationsbeispiel
$s = 5$, $n_1 = 3$, $n_2 = 2$, VORLAUF: 500 Anforderungen
10 TEILTESTS: je 5000 Anforderungen

Unter Berücksichtigung aller bisherigen Erfahrungen, die naturgemäß beschränkt sein müssen, ergibt sich der Eindruck, daß für viele Probleme der Nachrichtenverkehrstheorie zur Zeit SIMSCRIPT und SIMULA als geeignete Simulationssprachen betrachtet werden können.

LITERATURVERZEICHNIS

/1/ KOSTEN, L.: On the Measurement of Congestion Quantities by Means of Fictitious Traffic. HET PTT-Bedrijf (1948/49), 15-25

/2/ NEOVIUS, G.: Artificial Traffic Trials Using Digital Computers. Ericsson Technics 11(1955), 279-291

/3/ LOTZE, A.: Über die statistische Sicherheit von Verkehrsmessungen. NTZ 11(1958)1, 5-7

/4/ SCHMETTERER, L.: Einführung in die mathematische Statistik. Springer-Verlag, Wien, New York, 2.Aufl.1966

/5/ KÜMMERLE, K.: Ein Vorschlag zur Berechnung der Vertrauensintervalle bei Verkehrstests. A.E.Ü.23(1969)10, 507-511

/6/ HUBER, M., WAGNER, W.: Simulation von Nachrichtenvermittlungssystemen. Aus: Nicht-numerische Informationsverarbeitung, Herausgeber: R. Gunzenhäuser, Springer-Verlag, Wien, New York, 1968

/7/ WAGNER, H. DIETRICH, G.: Bestimmung der Verkehrsleistung von Wartesystemen durch künstlichen Fernsprechverkehr. NTZ 17(1964)6, 273-279

/8/ WEISSCHUH, H.: Theoretical Aspects of Finite Source Input Process Simulation. 7th International Teletraffic Congress, Stockholm, 1973

/9/ RAKOTOMANGA, D.: Simulationsprogramm in FORTRAN für ein Warte-Verlustsystem mit Überlauf. Monografie, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1970

/10/ THIEL, K.-H.: Simulationssprache GPSS. Monografie, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1970

/11/ PFEIFFER, H.: Nachbildung der Datenübertragung zwischen zwei Vermittlungsrechnern mit Hilfe der Simulationssprache SIMSCRIPT, Teil 1. Monografie, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1970

/12/ DITZEL, D.: Simulationssprache SIMULA. Monografie, Institut für Nachrichtenvermittlung und Datenverarbeitung, Universität Stuttgart, 1972

/13/ IBM: General Purpose Simulation System/ 360 User's Manual. IBM Nr. H20-0326-3, 1968

/14/ MARKOWITZ, H.; HAUSNER, B., KARR, H.: SIMSCRIPT: A Simulation Programming Language. The Rand Corporation, Santa Monica, Calif., RM-3310, Nov. 1962

/15/ KIVIAT, P.J., VILLANUEVA, R., MARKOWITZ, H.: The SIMSCRIPT II Programming Language. Prentice-Hall, Inc., New Jersey, 1968

/16/ KAMPE, G.: SIMSCRIPT. Vieweg-Verlag, Braunschweig, 1971

/17/ RAEHMEL, Ch.-A.: SIMSCRIPT-eine Simulationssprache. Computerpraxis 5(1972)11, 321-327

/18/ DAHL, O.J., NYGAARD, K.: SIMULA, a language for programming and description of discrete event systems. Introduction and User's Manual. Norwegian Computing Center, Oslo, 1966

/19/ CONTROL DATA CORPORATION: 6400/6500/6600 Computer Systems: SIMULA Reference Manual. Nr. 60234800 Rev.A, Palo Alto, 1969

/20/ NIEDEREICHHOLZ, J.: Einführung in die Simulationssprache SIMULA. El.Datenverarbeitung 12(1970)5, 204-207

/21/ BOESCH, R., BENN, J.: SIMULA 6000 - Beispiele. Rechenzentrum der FIDES-Treuhandvereinigung, Publ.36, 1. Ausgabe 1970

/22/ TOCHER, K.D.: Review of Simulation Languages. Operations Research Quarterly, 16(1965)2, 189-217

/23/ FENNEL, H.J.: Simulation und die Simulationssprachen SIMSCRIPT und GPSS II. El.Datenverarbeitung 7(1965)3, 130-140

/24/ TEICHROEW, D., LUBIN, J.F.: Computer Simulation - Discussion of Technique and Comparison of Languages. Comm.ACM 9(1966)10, 723-741

/25/ KRASNOW, H.S.: Dynamic Presentation in Discrete Interaction Simulation Languages. In "Digital Simulation in Operations Research", S.H.HOLLINGDALE (Ed.), The English Universities Press Ltd., London, 1967

/26/ WEINERT, A.E.: A SIMSCRIPT-FORTRAN case study. Comm.ACM 10(1967)12, 781-792

/27/ BUXTON, J.N.(Ed.): Simulation Programming Languages. North-Holland Publishing Company, Amsterdam, 1968

/28/ KIVIAT, P.J.: Digital Computer Simulation: Computer Programming Languages. The Rand Corporation, Santa Monica, Calif., RM-5883-PR, Jan.1969

/29/ NOSKA, T.: Ermittlung der für Planung und Betrieb von Hüttenwerken geeigneten Simulationssprachen. Dissertation an der Montanistischen Hochschule Loeben, 1970

/30/ BERGER, B., SEIBT, D., STRUNZ, H.: Bibliographie des Betriebswirtschaftlichen Institutes für Organisation und Automation an der Universität Köln zum Thema "Programmiersprachen". El.Datenverarbeitung 11(1969)5, 225-234; 7, 330-341; 8, 387-397

/31/ LÜTTGENS, H.-G., SEIBT, D.: Ergänzungen zu Literatur /28/. Angewandte Information 13(1971)3, 137-144

/32/ EMSHOFF, J.R., SISSON, R.L.: Simulation mit dem Computer. Verlag Moderne Industrie, München, 1972

Burkhardt
Hinsch

GMD / I 7

Simulation von Nachrichtenquellen in blockorientierten Simulationsmodellen

An ein DFV-Netz werden quantitative Anforderungen gestellt, denen es unter gewissen Randbedingungen genügen soll. Die Frage, ob und wieweit ein konzipiertes Netz diesen Anforderungen genügen wird, läßt sich mit exakten mathematischen Methoden kaum beantworten. Eine quasi experimentelle Technik quantitative Aussagen über ein Netz vor seiner Realisierung zu gewinnen, bildet die Simulation.

Auf dem Gebiet der Simulation arbeiten wir schwerpunktmäßig an der Entwicklung von Simulationsmodellen unterschiedlichen Detaillierungsgrades. Ziel ist die Entwicklung vorprogrammierbarer Module für die einzelnen Komponenten eines DFV-Netzes. Wir untersuchen dabei auch die Eignung von Simulationssprachen für diesen Zweck (GPSSV, SIMSCRIPT II, CSMP/360).
Damit soll einerseits erreicht werden, daß die im allgemeinen sehr langwierige Phase der Modellbildung verkürzt wird,und andererseits, daß - in gewisser Weise als Folge dessen - früher und gezielter die für die Simulation nötigen Daten zusammengestellt werden können.

In diesem Zusammenhang haben wir uns auch mit dem Problem auseinandergesetzt, Nachrichtenquellen als eine der Komponenten eines DFV-Netzes nachzubilden. Wir sind zu dem Ergebnis gekommen, daß wir mit den folgenden 4 Grundtypen von Modellen die wesentlichen in der Praxis vorkommenden Fälle erfassen:

1. Ungesteuerte Nachrichtenquelle, die nur Nachrichten einer Nachrichtenart erzeugt,

2. ungesteuerte Nachrichtenquelle , die Nachrichten mehrerer Nachrichtenarten nach einer bestimmten statistischen Verteilung erzeugt,

3. gesteuerte Nachrichtenquelle, die logisch untereinander verbundene Nachrichtenfolgen erzeugt,

4. zwei sich gegenseitig steuernde Nachrichtenquellen zur Nachbildung eines Dialogs.

Im folgenden werden jeweils die einzelnen Modelle und ihre Formulierung in einer blockorientierten Simulationssprache (GPSS) erläutert.

1. Simulation von ungesteuerten Nachrichtenquellen, die nur Nachrichten einer Nachrichtenart erzeugen.

   Modell:

   Die Nachrichtenquelle soll ungesteuert Nachrichten einer definierten Nachrichtenart und einer um einen definierten Wert normalverteilten Länge erzeugen. Die Nachricht soll in der Verarbeitungseinheit um eine Zeit verzögert werden, die ebenfalls um einen definierten Wert normalverteilt ist. Der zeitliche Abstand zwischen den einzelnen Nachrichten liegt gleichverteilt um einen Mittelwert in einem Intervall definierter Länge.

Programm :

Jede Nachricht wird durch eine Transaction repräsentiert mit jeweils drei Parametern:

- Par. 1 enthält die Nachrichtenart,
- Par. 2 enthält die Nachrichtenlänge,
- Par. 3 enthält die Verzögerungszeit.

Als Modifikator für Nachrichtenlänge und Verzögerungszeit dient jeweils die Funktion FN1, die eine Normalverteilung mit Mittelwert $\mu = 1$ und Standardabweichung $\sigma = 0{,}1$ approximiert.

Im Programm werden 3 Nachrichtenquellen nachgebildet, die jew. Nachrichten der Nachrichtenart 1, 2 und 3 mit einer einheitlichen mittleren Länge von 500 Zeichen und einer einheitlichen mittleren Verzögerungszeit von 100 ms erzeugen.

Der zeitliche Abstand zwischen den einzelnen Nachrichten liegt bei allen 3 Nachrichtenquellen gleichverteilt im Intervall 15 000 ms $\pm$ 5000 ms.

Als Verarbeitungseinheit wird ein einfacher Konzentrator mit Geschwindigkeitsumsetzung nachgebildet.

Die ankommenden Nachrichten werden in einen gemeinsamen Speicher übernommen und in eine Verarbeitungswarteschlange eingereiht, aus der sie auf der Basis FIFO verarbeitet werden. Die Verarbeitungszeit wird Par. 3 der Nachricht entnommen . Nach der Verarbeitung wird die Nachricht über die abgehende Leitung direkt aus dem Speicher gesendet. Verarbeitungseinheit und abgehende Leitung bleiben für die

Dauer des Sendens belegt; auch der durch die Nachricht belegte Speicher wird erst nach Abschluß des Sendens freigegeben.

2. Simulation einer ungesteuerten Nachrichtenquelle, die Nachrichten mehrerer Nachrichtenarten nach einer bestimmten statistischen Verteilung erzeugt.

Modell:
Die Nachrichtenquelle soll ungesteuert Nachrichten dreier verschiedener Nachrichtenarten erzeugen, die mit definierter Wahrscheinlichkeit auftreten. In Abhängigkeit von der Nachrichtenart werden jeder Nachricht wie unter 1 eine definierte Länge und Verzögerungszeit zugeordnet. Der zeitliche Abstand zwischen den einzelnen Nachrichten liegt gleichverteilt um einen Mittelwert in einem Intervall def. Länge.

Programm:
Jede Nachricht wird durch eine Transaction repräsentiert mit jeweils drei Parametern:

- Parameter 1 enthält die Nachrichtenart,
- Parameter 2 enthält die Nachrichtenlänge,
- Parameter 3 enthält die Verzögerungszeit.

Als Modifikator für Nachrichtenlänge und Verzögerungszeit dient jeweils die Funktion FN1 , die eine Normalverteilung auf Mittelwert $\mu = 1$ und Standardabweichung $\sigma = 0{,}1$ approximiert. Im Programm wird eine Nachrichtenquelle nachgebildet, die jew. Nachrichten der Nachrichtenart 1, 2 und 3 erzeugt mit einer mittleren Länge von 500, 400 und 300 Zeichen und einer mittleren Verzögerungszeit von 100, 80 und 50 ms. Die Wahrscheinlichkeit für das Auftreten der Nachrichtenarten 1, 2 und 3 sei 0,2, 0,5 und 0,3. Der zeitliche Abstand liegt gleichverteilt im Intervall 25000 $\pm$ 5000 ms.

3. Simulation einer gesteuertern Nachrichtenquelle, die logisch untereinander verbundene Nachrichtenfolgen erzeugt.

Modell:

Die Nachrichtenquelle soll logisch untereinander verknüpfte Nachrichtenfolgen erzeugen, wobei jeder Nachricht in Abhängigkeit von der Nachrichtenart der vorangegangenen Nachricht eine von 4 möglichen Nachrichtenarten A, B, C und Z zugeordnet wird. Die Nachrichtenarten bilden eine Markow'sche Kette. In der folgenden Matrix sind die Wahrscheinlichkeiten des Übergangs von den Nachrichtenarten A, B, C, Z in die Nachrichtenarten A, B, C, Z aufgeführt, wobei das Matrixelement (X, Y) (X = A, B, C, Z; Y = A, B, C, Z) die Übergangswahrscheinlichkeit von Nachrichtenart X in Nachrichtenart Y darstellt.

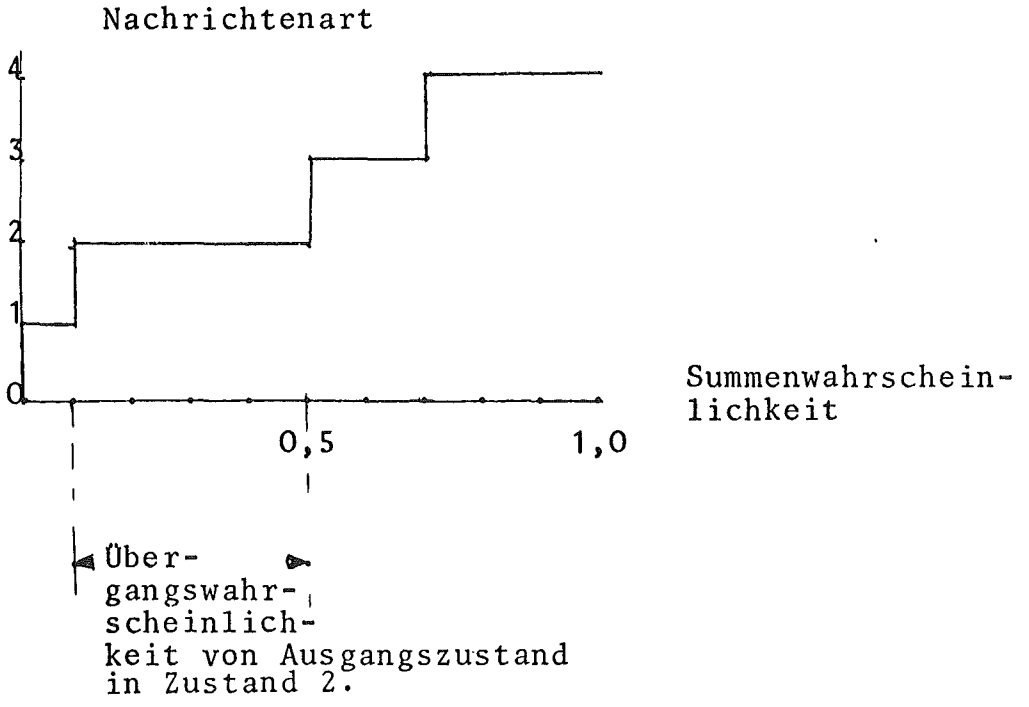
	A	B	C	Z
A	0,1	0,4	0,2	0,3
B	0,0	0,1	0,3	0,6
C	0,0	0,4	0,1	0,5
Z	0,2	0,0	0,0	0,8

Da das System immer nur von einer einzigen Nachricht durchlaufen werden soll, wird die Erzeugung einer neuen Nachricht durch die Verweilzeit im System der vorangegangenen Nachricht gesteuert. Diese Nachrichtenquelle ist ein primitives Modell für einen Dialog. Die Nachricht vom Typ A ist dann als Beginn eines Dialogs zu interpretieren. Die Typen B oder C sind mögliche weitere Anfragen und der Typ Z bedeutet die Beendigung des Dialogs und keine Aktivität an der DEE. In diesem Dialog werden also die Antworten der DVA nur pauschal in den an der DEE eingegebenen Anfragen mitberücksichtigt.

Programm:

Jede Nachricht des Modells wird in GPSS in eine Transaction abgebildet. Das Programm gliedert sich in folgende Abschnitte:

1. Nichtausführbarer Definitionsteil des Programms
Sämtliche Veränderliche des Programms wurden im Definitionsteil festgelegt, um leicht Änderungen vornehmen zu können. Die Matrix der Übergangswahrscheinlichkeiten wurde in die Funktionen 1 bis 4 abgebildet. Aus der 1. Zeile der Matrix ergibt sich z.B. folgende Funktion (Den Nachrichtenarten wurden die Kennziffern KZ = 1, 2, 3, 4 zugeordnet):



Die Bearbeitungszeit an der DEE ist abhängig von der vorangegangenen und der gegenwärtigen Nachrichtenart (Ausgabezeit auf dem Bildschirm, Eintippzeit). Das wird in der folgenden Matrix dargestellt (Einheit ms):

Erzeugte Nachrichtenart

Ausgangsnachrichtenart	1	2	3	4
1	2000	1500	1000	300
2	0	130	90	300
3	0	1200	800	300
4	1700	0	0	1

Die Zeilen dieser Matrix werden in die Funktionen 5 - 8 umgesetzt.

Die Zufallsabweichungen von z.B. Nachrichtenlänge oder Bearbeitungszeiten um einen Mittelwert werden mit Hilfe der Funktion 9 erzeugt. Über die Funktionen 10 und 11 wird auf Nachrichtenlänge und Eingabezeit zugegriffen.

Über den arithmetischen Ausdruck in Variable 1 wird die Funktionsnummer der benötigten Funktion zur Bestimmung der Bearbeitungszeit berechnet. Variable 2 dient zur Berechnung der Table-Nr. und Variable 3 zur Berechnung der Sendezeit.

Der Zustand zu Beginn der Simulation wird in den Savevaluelocations 1 und 2 festgelegt. Anfangsnachrichtenart ist Z (KZ=4), Ausgangsnachrichtenart war A (KZ=1).

2. Ausführbarer Teil

Der ausführbare Teil gliedert sich in folgende Abschnitte:

2.1 Erzeugen der Transactions

Die Transactions werden im Generate-Block erzeugt. Facility 3 steuert ihre zeitliche Aufeinanderfolge, da der Generate-Block von einer Transaction nur verlassen werden kann, wenn Facility 3 nicht besetzt ist. Die Transaction wird verzögert um die Zeit, die der Bediener an der DEE braucht, um die Antwort auszuwerten und eine neue Nachricht einzutippen. War die neue Nachrichtenart Z, dann wird die Transaction vernichtet und Facility 3 freigegeben. Die neue Nachrichtenart wird aus der vorangegangenen Nachrichtenart, die in X1 gespeichert war, bestimmt. Die alte Nachrichtenart wird in X3 und die neue in X1 gespeichert.

Den Parametern der Transaction werden die zugehörigen Werte zugeordnet: P1 die Nachrichtenart, P2 die Nachrichtenlänge.

2.2 Durchlaufen der Verarbeitungseinheiten
Die Transaction gelangt in den Puffer (entspricht Storage 1) der DEE. Die Nachricht wird gesendet. Facility 2 bildet den Engpass nach, der dadurch entsteht, daß während einer bestimmten Zeitspanne nur eine einzige Nachricht gesendet werden kann. Die Nachricht wird um die Sendezeit verzögert. Die Storages und Facilities werden freigegeben. Die Transaction wird vernichtet.

2.3 Ausgabe
Neben der normalen statistischen Ausgabe in GPSS wurde weitere Ausgabe durch den QUEUE -Block vor Storages und Facilities erzeugt. Mit Hilfe des Tabulate-Blocks und der Tables 1 - 16 wurde die Anzahl aller Transactions, die während der Simulation einem bestimmten Nachrichtentyp zugeordnet waren, ausgedruckt. Das ermöglichte eine Überprüfung der Matrix der Übergangswahrscheinlichkeiten.

4. Simulation von zwei sich gegenseitig steuernden Nachrichtenquellen zur Nachbildung eines Dialogs.

Modell:

Die zwei sich gegenseitig steuernden Nachrichtenquellen entsprechen im Dialog einer DEE und einer DVA. Die DEE als Nachrichtenquelle erzeugt Nachrichten vom Typ A, B, C, Z (entsprechend Beginn, Fortsetzung und Beendigung des Dialogs) und die DVA Nachrichten von Typ A1, B1 oder C1 (entsprechend den möglichen Antworten).

Der Dialog wird von der DEE initialisiert und zwar mit einer Nachricht vom Typ A. Die Nachricht wird zur DVA gesendet und erzeugt dort mit gewissen Wahrscheinlichkeiten eine Nachricht vom Typ A1, B1 oder C1, die als Antwort zur DEE gesendet wird und dort eine Nachricht vom Typ A, B, C oder Z erzeugt. Wird der Nachrichtentyp Z erzeugt, dann ist der Dialog beendet und es herrscht keine Aktivität an der DEE, sonst wird der Dialog fortgesetzt. Als Nachfolgetyp für den Typ Z ergibt sich entweder A oder Z. Die Übergangswahrscheinlichkeiten der verschiedenen Nachrichtentypen sind in den folgenden zwei Matrizen dargestellt.

I. Matrix der Übergangswahrscheinlichkeiten der Nachrichtenarten A, B, C in die Nachrichtenarten A1, B1, C1

	A1	B1	C1
A	0,70	0,20	0,10
B	0,00	0,80	0,20
C	0,10	0,40	0,50

II. Matrix der Übergangswahrscheinlichkeiten der Nachrichtenarten A1, B1, C1 und Z in die Nachrichtenarten A, B, C und Z.

	A	B	C	Z
Z	0,1	0,0	0,00	0,90
A1	0,01	0,40	0,40	0,19
B1	0,00	0,50	0,20	0,30
C1	0,00	0,10	0,60	0,30

Programm:

Der Dialog ist eine Aufeinanderfolge von Nachrichten verschiedener Nachrichtenarten. Der Dialog wird im Programm als eine Transaction abgebildet, dem im Laufe des Dialogs verschiedene Nachrichtenarten zugeordnet werden. Zwischen zwei Dialogen herrscht keine Aktivität an der DEE. Das wird durch die Erzeugung von Transactions mit dem Nachrichtentyp Z repräsentiert.

Das Programm gliedert sich in folgende Abschnitte:

1. Nichtausführbarer Definitionsteil des Programms
   Sämtliche Veränderliche des Programms wurden im Definitionsteil festgelegt, um leicht Änderungen vornehmen zu können. Die Matrix II der Übergangswahrscheinlichkeiten wurde in die Funktionen 1 - 4 umgesetzt, die Matrix I in die Funktionen 9 - 11. Die Zufallsabweichungen von z.B. Nachrichtenlänge um einen Mittelwert werden mit Hilfe der Funktion 7 erzeugt. Über die Funktionen 5 und 12 wird auf die Nachrichtenlänge, über Funktion 6 auf die Eingabezeit, über Funktion 8 auf die Verarbeitungszeit in der DVA, über Funktion 13 auf die Entstehungszeit der neuen Nachrichtenart in der DVA und über Funktion 14 auf die Zeit,

die der Bediener an der DEE zur Auswertung der Nachrichten von der DVA benötigt, zugegriffen. Die Variablen 1 - 5 dienen der Reihe nach zu folgenden Berechnungen: Funktionsnummer für die Übergangswahrscheinlichkeiten von A1, B1, C1 nach A, B, C, Tablenummer, Sendezeit, Funktionsnummer für die Übergangswahrscheinlichkeiten von A, B, C nach A1, B1, C1 und für die Ausgabezeit auf den Bildschirm.

2. Ausführbarer Teil
Der ausführbare Teil gliedert sich in folgende Abschnitte.

2.1 Erzeugen der Transactions
Die Transactions werden im Generate-Block erzeugt. Facility 1 steuert ihre zeitliche Aufeinanderfolge, da der Generate-Block von einer Transaction nur verlassen werden kann, wenn Facility 3 nicht besetzt ist. Dem Parameter 1 der Transaction wird die Kennziffer für die Nachrichtenart zugewiesen (A = 1, B = 2, C = 3, Z = 4 , A1 = 1, B1 = 2, C1 = 3). Ist die Nachricht vom Typ A (Beginn eines Dialogs), dann werden die Bearbeitungseinheiten durchlaufen, sonst wird Facility 1 freigegeben und die Transaction vernichtet.

2.2 Durchlaufen der Bearbeitungseinheiten
In dem Parameter 2 der Transaction wird die Nachrichtenlänge gespeichert. Die Nachricht wird um die Eingabezeit an der DEE verzögert und gelangt in den Ausgabepuffer der DEE, der durch Storage 1 repräsentiert wird. Die Nachricht wird gesendet. Facility 2 entspricht der Sendeeinheit der DEE. Die Nachricht wird um die Sendezeit verzögert. Storage 1 und Facility 2 werden freigegeben. Die Nachricht gelangt in den Eingabepuffer der DVA (Storage 2), besetzt die Verarbeitungseinheit (Facility 3) und wird um die Verarbei-

tungszeit verzögert. Dann werden Verarbeitungseinheit und Puffer (Storage 2, Facility 3) freigegeben. Die Antwortnachrichten werden erzeugt, indem der Transaction die neuen Nachrichtenarten A1, B1 oder C1 zugeordnet werden. Die Nachricht wird um die Entstehungszeit verzögert, gelangt in den Puffer (Storage 3), besetzt die Sendeeinheit (Facility 4) und wird um die Sendezeit verzögert. Die Nachricht gelangt in den Eingabepuffer der DEE (Storage 4), besetzt die Ausgabeeinheit für den Bildschirm (Facility 5) und wird um die Ausgabezeit verzögert. Die Nachricht verlässt den Eingabepuffer (Storage 2) und gibt die Ausgabeeinheit (Facility 5) frei.

3. Ausgabe

Neben der normalen statistischen Ausgabe in GPSS wurde weitere Ausgabe durch den QUEUE-Block von Storages und Facilities erzeugt. Mit Hilfe des Tabulate-Blocks und der Tables 1 - 16 wurde die Anzahl aller Transactions, die während der Simulation einem bestimmten Nachrichtentyp zugeordnet waren, ausgedruckt. Das ermöglichte eine Überprüfung der Matrix der Übergangswahrscheinlichkeiten.

```
*       SIMULATE
*
*
*
*   SIMULATION  VON  NACHRICHTENQUELLEN  DIE  NUR EINE
*    NACHRICHTENART   ERZEUGEN
*
*
 1      FUNCTION   RN1,C5
C.     0.8   0.16  0.9   0.5   1.    0.84  1.1   1.    1.2
 1      VARIABLE   P2*2000/3
 1      TABLE      M1,100,100,9
*    TERMINAL 1
       GENERATE   15000,5000,0,1000,,5
*    ZUORDNUNG  DER NACHRICHTENART, NACHRICHTENLAENGE  UND  DER
*     ZUR  NACHRICHT  GEHOERENDEN  BEARBEITUNGSZEIT
       ASSIGN     1,1
       ASSIGN     2,500,1
       ASSIGN     3,100,1
       TRANSFER   ,CCM
*    TERMINAL2
       GENERATE   15000,5000,0,1000,,5
*    ZUORDNUNG  DER NACHRICHTENART, NACHRICHTENLAENGE  UND  DER
*     ZUR  NACHRICHT  GEHOERENDEN  BEARBEITUNGSZEIT
       ASSIGN     1,2
       ASSIGN     2,500,1
       ASSIGN     3,100,1
       TRANSFER   ,CCM
*     TERMINAL   3
       GENERATE   15000,5000,0,1000,,5
*    ZUORDNUNG  DER NACHRICHTENART, NACHRICHTENLAENGE  UND  DER
*     ZUR  NACHRICHT  GEHOERENDEN  BEARBEITUNGSZEIT
       ASSIGN     1,3
       ASSIGN     2,500,1
       ASSIGN     3,100,1
*    QUEUE 1  ERZEUGT DEN STATIST.  OUTPUT  DER WARTESCHLANGE VOR
*    STORAGE  1
 COM   QUEUE      1,1
*   STORAGE 1 ENTSPRICHT DEM  KONZENTRATORPUFFER
       ENTER      1,*2
       DEPART     1,1
*    QUEUE  2 ERZEUGT DEN STATIST.  OUTPUT  DER WARTESCHLANGE VOR
*     FACILITY  1
       QUEUE      2,1
*    FACILITY  1 ENTSPRICHT DEM      ENGPASS  DER
*     DADURCH ENTSTEHT  , DASS WAEHREND EINER BESTIMMTEN   ZEITSPANNE
*    NUR EINE EINZIGE NACHRICHT GESENDET WERDEN KANN
       SEIZE      1
       DEPART     2
*    VERZOEGERN  DER NACHR.  UM DIE ZUGEHOERIGE BEARBEITUNGSZEIT
       ADVANCE    *3,1
*    QUEUE  3 ERZEUGT DEN STATIST.  OUTPUT  DER WARTESCHLANGE VOR
*    FACILITY  2
       QUEUE      3,1
*    FACILITY  2 ENTSPRICHT DEM      ENGPASS  DER
*     DADURCH ENTSTEHT  , DASS WAEHREND EINER BESTIMMTEN   ZEITSPANNE
*    NUR EINE EINZIGE NACHRICHT BEARBEIT.WERDEN KANN
       SEIZE      2
       DEPART     3
*    DIE NACHRICHT WIRD UM DIE SENDEZEIT VERZOEGERT
       ADVANCE    V1
*    FREIGEBEN  VON STORAGE UND FACILITIES
       LEAVE      1,*2
       RELEASE    1
```

```
RELEASE     2
TABULATE    1
TERMINATE   1
START       1000
END
```

69 HOLLERITHKARTEN

```
*       SIMULATE
*
*
*
*   SIMULATION EINER NACHRICHTENQUELLE DIE MEHRERE NACHRICHTENARTEN
*    NACH EINER BESTIMMTEN STATISTISCHEN VERTEILUNG ERZEUGT
*
*
 1      FUNCTION   RN1,D3
0.2   1     0.7   2       1.     3
 2      FUNCTION   P1,L3
1      500   2     400     3      300
 3      FUNCTION   P1,L3
1      100   2     80      3      50
 4      FUNCTION   RN1,C5
0.     0.6   0.16  0.9   0.5   1.    0.84  1.1   1.    1.2
        INITIAL    X1,0
 1      VARIABLE   X1+1
 2      VARIABLE   V1(2
        GENERATE   12500,5000,0,1000,,3
*      JEDE 2. TRANSACTION WIRD SOFORT VERNICHTET
        SAVEVALUE  1,V2
        TEST E     0,X1,END
*    ZUORDNUNG DER NACHRICHTENART, NACHRICHTENLAENGE UND DER
*     ZUR NACHRICHT GEHOERENDEN BEARBEITUNGSZEIT
        ASSIGN     1,FN1
        ASSIGN     2,FN2,4
        ASSIGN     3,FN3,4
*    QUELLE 1 ERZEUGT DEN STATIST. OUTPUT DER WARTESCHLANGE VOR
*   STORAGE  1
 COM    QUELLE     1,1
*   STORAGE 1 ENTSPRICHT DEM KONZENTRATORPUFFER
        ENTER      1,*2
        DEPART     1,1
*    QUELLE  2 ERZEUGT DEN STATIST. OUTPUT DER WARTESCHLANGE VOR
*  FACILITY  1
        QUELLE     2,1
*    FACILITY  1 ENTSPRICHT DEM      ENGPASS  DER
*     DADURCH ENTSTEHT  , DASS WAEHREND EINER BESTIMMTEN   ZEITSPANNE
*    NUR EINE EINZIGE NACHRICHT GESENDET WERDEN KANN
        SEIZE      1
        DEPART     2
*    VERZOEGERN  DER NACHR.  UM DIE ZUGEHOERIGE BEARBEITUNGSZEIT
        ADVANCE    *3,1
*    QUELLE  3 ERZEUGT DEN STATIST.  OUTPUT  DER WARTESCHLANGE VOR
*   FACILITY  2
        QUELLE     3,1
*    FACILITY  2 ENTSPRICHT DEM      ENGPASS  DER
*     DADURCH ENTSTEHT  , DASS WAEHREND EINER BESTIMMTEN   ZEITSPANNE
*    NUR EINE EINZIGE NACHRICHT BEARBEIT.WERDEN KANN
        SEIZE      2
        DEPART     3
*    VERZOEGERN  DER NACHR.  UM DIE ZUGEHOERIGE SENDEZEIT
        ADVANCE    V1
        LEAVE      1,*2
        RELEASE    1
        RELEASE    2
 END    TERMINATE  1
        START      1000
        END
```

```
*       SIMULATE
*
*
*
* SIMULATION EINER NACHRICHTENQUELLE DIE EINE LOGISCH UNTEREINANDER
*   VERBUNDENE
*  NACHRICHTENFOLGE  ERZEUGT
*
*
*  UEBERGANGSWAHRSCHEINLICHKEITEN
 1      FUNCTION   RN1,D4
0.10  1      0.5    2      0.7    3      1.0    4
 2      FUNCTION   RN1,D3
0.1   2      0.4    3      1.     4
 3      FUNCTION   RN1,D3
0.4   2      0.5    3      1.     4
 4      FUNCTION   RN1,D2
0.2   1      1.0    4
*  BEARBEITUNGSZEIT
 5      FUNCTION   X1,L4
1      2000  2      1500  3      1000  4      300
 6      FUNCTION   X1,L4
1      0     2      130   3      90    4      300
 7      FUNCTION   X1,L4
1      0     2      1200  3      800   4      300
 8      FUNCTION   X1,L4
1      1700  2      0     3      0     4      1
*  NORMALVERTEILUNG
 9      FUNCTION   RN1,C5
0.     0.6   0.16  0.9   0.5   1.    0.84  1.1   1.    1.2
*   LAENGE DER NACHRICHT
 10     FUNCTION   X1,L4
1      50    2      20    3      10    4      0
*EINGABEZEIT
 11     FUNCTION   X1,L4
1      2500  2      1000  3      500   4      1
*  BERECHNUNG  D.  FKTSNR.  ZUR  BESTIMMUNG  DER  BEARBEITUNGSZEIT
 1      VARIABLE   X1+4
*  BERECHNUNG  DER  TABELNR.
 2      VARIABLE   4*X3-4+X1
*   BERECHNUNG  DER  SENDEZEIT
 3      VARIABLE   P2*20/3
        INITIAL    X1,4
        INITIAL    X2,1
 1      TABLE      X2,1,500,5
 2      TABLE      X2,1,500,5
 3      TABLE      X2,1,500,5
 4      TABLE      X2,1,500,5
 5      TABLE      X2,1,500,5
 6      TABLE      X2,1,500,5
 7      TABLE      X2,1,500,5
 8      TABLE      X2,1,500,5
 9      TABLE      X2,1,500,5
 10     TABLE      X2,1,500,5
 11     TABLE      X2,1,500,5
 12     TABLE      X2,1,500,5
 13     TABLE      X2,1,500,5
 14     TABLE      X2,1,500,5
 15     TABLE      X2,1,500,5
 16     TABLE      X2,1,500,5
        GENERATE   1,1,0,10000,,4
*  FACILITY 3 STEUERT DIE ZEITLICHE AUFEINANDERFOLGE VON NACHRICHTEN
        SEIZE      3
        ASSIGN     2,X1
```

```
*   IN X3  WIRD   X1 GESPEICHERT  NACHRICHTENART DER VORANGEGANGENEN
*   NACHRICHT
          SAVEVALUE   3,X1
*  IN X1   WIRD NEUE NACHR.ART GESPEICHERT
          SAVEVALUE   1,FN*2
*   ZUORDNUNG VON NACHRICHTENART UND  NACHR.LAENGE ZU G. PARAMETERN
*   1 UND 2
          ASSIGN      1,X1
          ASSIGN      2,V1
          SAVEVALUE   2,FN*2
          ASSIGN      2,FN1C,9
*    VERZOEGERN  DER NACHR.  UM DIE ZUGEHOERIGE BEARBEITUNGSZEIT
          ADVANCE     X2,FN9
          TABULATE    V2
          TEST L      4,*1,END
*    QUEUE 1  ERZEUGT DEN STATIST.  OUTPUT  DER WARTESCHLANGE VOR
*    STORAGE  2
 COM      QUEUE       1,1
*STORAGE  1 ENTSPRICHT  DEM  SENDEPUFFER DER  DEE
          ENTER       1,*2
          DEPART      1,1
*    QUEUE  3 ERZEUGT DEN STATIST.  OUTPUT  DER WARTESCHLANGE VOR
*   FACILITY 2
          QUEUE       3,1
*    FACILITY  2 ENTSPRICHT DEM       ENGPASS  DER
*     DADURCH ENTSTEHT  , DASS WAEHREND EINER BESTIMMTEN ZEITSPANNE
*    NUR EINE EINZIGE NACHRICHT GESENDET WERDEN KANN
          SEIZE       2
          DEPART      3
*    DIE NACHRICHT WIRD UM DIE SENDEZEIT VERZOEGERT
          ADVANCE     V3
*    FREIGEBEN  VON STORAGE UND FACILITIES
          LEAVE       1,*2
          RELEASE     2
 END      RELEASE     3
          TERMINATE   1
          START       1000
          END
```

102 HOLLERITHKARTEN

```
*       SIMULATE
*
*
*     SIMULATION VON  2 SICH GEGENSEITIG  STEUERNDEN NACHRICHTEN-
*       QUELLEN  ZUR NACHBILDUNG EINES DIALOGS
*
*
*
*
*VARIABLE
*
*
*BERECHNUNG DER FUNKTI.  NR.  FUER DIE UEBERGANGSWAHRSCH.
*  VON A1,B1,C1  NACH A,B,C
 1      VARIABLE   1+X1
*  BERECH.  DER TABLES
 2      VARIABLE   4*X3-4+X1
*  SENDEZEIT
 3      VARIABLE   P2*2/3
*BERECHNUNG DER FUNKTI.  NR.  FUER DIE UEBERGANGSWAHRSCH.
*  VON A,B,C  NACH  A1,B1,C1
 4      VARIABLE   X1+8
*  AUSGABEZEIT AUF DEM  BILDSCHIRM
 5      VARIABLE   P2*10
*  FUNKTICNEN
*
*
*  BESTIMMT DIE NACHR. ART  NACH ERZEUG   EINER TRANSA. A ODER Z
 1      FUNCTION   RN1,D2
0.10  1      1.0    4
*  ZUR  BESTIMMUNG DER  NACHRICHTENARTA,B,C,Z   IN ABHAENGIGKEIT
*   VOM TYP DER VORANGEHENDEN NACHR. DIE VOM TYP A1,B1 ODER C1 WAR
 2      FUNCTION   RN1,D4
0.01  1      0.41   2      0.81  3      1.0    4
 3      FUNCTION   RN1,D3
0.5   2      0.7    3      1.0    4
 4      FUNCTION   RN1,D3
0.1   2      0.7    3      1.0    4

*  ZUORDNUNG  DER   NACHR.  LAENGE  FUER TYP A, B, C
 5      FUNCTION   X1,L3
1      50     2      20     3      10
*  ZUORDNUNG  DER EINGABEZEIT
 6      FUNCTION   X1,L3
1      2500   2      1000   3      500
*   ZUFALLSABWEICHUNG
 7      FUNCTION   RN1,C5
0.     0.8    0.16   0.9    0.5    1.0    0.84   1.1    1.0    1.2
*  VERARBEITUNGSZEIT IN DER DVA FUER TYP  A,B,C
 8      FUNCTION   X1,L3
1      20     2      15     3      10
*  UEBERGANGSWAHRSCHEINLICHKEITEN  VON A,B,C NACH A1,B1,C1
 9      FUNCTION   RN1,D3
0.7   1      0.9    2      1.0    3
 10     FUNCTION   RN1,D3
```

```
0.0    1      0.8    2      1.0    3
 11     FUNCTION   RN1,D3
0.1    1      0.5    2      1.0    3
*  NACRICHTENLAENGE FUER TYP  A1,B1,C1
 12     FUNCTION   X1,L3
1      100    2      40     3      20
*  ENTSTEHUNGSZEIT  FUER A1,B1,C1
 13     FUNCTION   X1,L3
1      10     2      8      3      5

*  ZEIT FUER BEARBEITUNG DES BEDIENERS AN DER DEE  FUER TYP
*   A1, B1 UND C1.
 14     FUNCTION   P1,L3
1      250    2      100    3      25
*   TABLES
*
*
 1      TABLE      P1,1,4,5
 2      TABLE      P1,1,4,5
 3      TABLE      P1,1,4,5
 4      TABLE      P1,1,4,5
 5      TABLE      P1,1,4,5
 6      TABLE      P1,1,4,5
 7      TABLE      P1,1,4,5
 8      TABLE      P1,1,4,5
 9      TABLE      P1,1,4,5
 10     TABLE      P1,1,4,5
 11     TABLE      P1,1,4,5
 12     TABLE      P1,1,4,5
 13     TABLE      P1,1,4,5
 14     TABLE      P1,1,4,5
 15     TABLE      P1,1,4,5
 16     TABLE      P1,1,4,5
*ERZEUGEN  EINER TRANSACTION
        GENERATE   1,1,0,15000,,4
        SEIZE      1
*  ZUWEISEN DER  PARAMETER  1 ODER 4
        ASSIGN     1,FN1
        TEST E     1,P1,END
        SAVEVALUE  1,P1
        SAVEVALUE  3,4
        TRANSFER   ,CON2
*ZUWEISUNG DES  PARAMETERS 1 IN ABHAENGIGKEIT VON DEN ANTWORTEN
*       A1, B1,C1.
 CON1   SAVEVALUE  3,P1
        ASSIGN     2,V1
        ASSIGN     1,FN*2
        SAVEVALUE  1,P1
        TABULATE   V2
        TEST NE    4,P1,END
*DURCHLAUFEN DER BEARBEITUNGSEINHEITENNACH ZUORDNEN DER NOETIGEN
*       PARAMETER
 CON2   ASSIGN     2,FN5          ZUORDNEN DER NACHRICHTENLAENGE
        ADVANCE    FN6,FN7        EINGABEZEIT AUF DER DEE
        ENTER      1,*2       IN AUSGABEPUFFER DER DEE WERDEN DIE
```

```
*                          ZEICHEN  GESPEICHERT
        SEIZE       2      SENDE EINHEIT DER DEE
        ADVANCE     V3                SENDEZEIT
        LEAVE       1,*2
        RELEASE     2
*  VERARBEITUNG IN DER DVA
        ENTER       2,*2       EINGABEPUFFER DER DVA
        SEIZE       3   VERARBEITUNGSEINHEIT  DER DVA
        ADVANCE     FN8,FN7                   VERARBEITUNGSZEIT
        LEAVE       2,*2
        RELEASE     3
*ERZEUGEN DER ANTWORT  UND ZUWEISEN DER NEUEN PARAMETAER
        ASSIGN      2,V4
        ASSIGN      1,FN*2
        SAVEVALUE   1,P1
        ASSIGN      2,FN12,7              LAENGE  DER NACHRICHT
        ADVANCE     FN13,FN7
        ENTER       3,*2
        SEIZE       4           SENDEEINHEIT   DER  DVA
        ADVANCE     V3       SENDEZEIT
        RELEASE     4
        LEAVE       3,*2
*  BEARBEITUNG IN DER DEE
        ENTER       4,*2             EINGABEPUFFER DER DEE
        SEIZE       5             AUSGABE EINHEIT F. DEN BILDSCHIRM
        ADVANCE     V5                AUSGABEZEIT
        ADVANCE     FN14,FN7
        LEAVE       4,*2
        RELEASE     5
        TRANSFER    ,CON1
 END    RELEASE     1
        TERMINATE   1
        START       15000
        END
```

Simulation der Kernspeicherbelegung in einem Elektronischen Datenverarbeitungssystem

W. Kistler
IBM Deutschland

Gegenstand

der Betrachtungen ist ein stochastisches Simulationsmodell mit analytischen Elementen, das zur Leistungsteigerung eines EDV-Systems entwickelt und eingesetzt wurde. Es ist unter gewissen Voraussetzungen auch in der Planungsphase verwendbar.

Simulationsziel:

Multiprogrammierbetrieb auf einer EDV-Anlage kann mit fester und auch mit variabler Anzahl und Größe der Kernspeicherbereiche (Regions) gefahren werden, die vom Operating System für die Anwendungsprogramme während ihrer Ausführungsphase zur Verfügung gestellt werden. Jede der Alternativen hat einen besonderen Einfluß auf Durchsatz, Verweilzeit der Programme, Hauptspeicherausnutzung und die Zeit, die benötigt wird, um Programmen verschiedener Größe den benötigten Kernspeicherplatz zur Verfügung zu stellen (Core-Allocation). Die genannten Faktoren werden außer von der Hardware-Konfiguration und der Art des Programm-Load auch von der Anzahl der Programme beeinflußt, die gleichzeitig aktiviert sind und die damit um den Hauptspeicherplatz und andere Systemteile konkurrieren.

Diese Anzahl ist zu optimieren, die Vor- und Nachteile von festen bzw. variablen Regions sind zu quantifizieren. Die Ergebnisse der Simulation sind Grundlage für eine Entscheidung über das günstigste Kernspeicherkonzept, nach dem das DV-System gefahren werden kann.

System-Analyse:

Bei einer elektronischen Datenverarbeitungsanlage unter Multiprogramming in Stapelverarbeitung werden den Programmen, die als "Job-Stream" einlaufen, vom Operating-System nach einem bestimmten Verfahren Kernspeicherbereiche zugeteilt, die sie dann während ihrer Ausführungsphase (Laufzeit) belegen. Die mittlere Laufzeit eines repräsentativen Programms ist unter anderem von der Hardware-Konfiguration und von der Anzahl der Konkurrenten um die System-Komponenten abhängig.

Bei einer Konfiguration mit fester Region-Größe (und -Anzahl) werden die Jobs (ein Job ist eine Reihe logisch voneinander abhängiger Programme) in Regions dirigiert, die für das jeweils größte der Programme im Job ausreichen. Bei den anderen Programmen entstehen dann Kapazitätsverluste. Dafür gibt es aber keine Zeitverzögerungen bei der Speicherzuweisung von Folgeprogrammen, da die Regiongröße, die vom vorausgegangenen Programm freigegeben wurde, auch für jedes folgende Programm des Jobs ausreicht. Beim Konzept der variablen Regiongröße wird jedem Programm nur der tatsächlich benötigte Platz zugeteilt. Der Kernspeicher wird damit besser ausgenutzt. Andererseits werden aber die Verweilzeiten der Jobs länger, da es häufig vorkommt, daß Folgeprogramme auf Speicherplatz warten müssen, weil sie größer sind als die freigegebe Region oder weil Programme parallellaufender Jobs bereits länger warten und damit die Region mit höherer Priorität beanspruchen.

Von besonderer Bedeutung sind bei diesen Vorgängen die "Initiators". Sie initiieren die Ausführung der Jobs im System; ihre Anzahl beeinflußt sowohl das Ausmaß der Parallelverarbeitung als auch die Verzögerung der Core-Allocation größerer Programme. Sind mehr Initiators aktiviert, als Programme im Hauptspeicher Platz finden, so wird der Speicher besser genutzt, denn für Lücken, die entstehen würden, weil das nächste Programm zu groß ist, hat dann oft einer der "wartenden" Initiators ein passendes Programm. Andererseits verringert sich damit die Chance für größere Programme, einen genügend großen zusammenhängenden Speicherbereich zu finden; ihre Ausführung wird verzögert und die Joblaufzeiten werden länger.

Die Zuweisung von Hauptspeicherplatz an die Programme durch das Operating System geschieht nach bestimmten Gesichtspunkten, die der Systemliteratur entnommen werden können. Soweit sie für den Simulationsverlauf von Einfluß sind, sind diese Funktionen mit angemessener Auflösung nachzubilden.

Modellbeschreibung, Modellsynthese:

Gegeben ist ein repräsentativer Strom von Programmen, in dem jedes Programm durch seine Kernspeichergröße und Laufzeit gekennzeichnet ist. Den Programmen wird für die Dauer ihrer Laufzeit nach festgelegtem Verfahren im Hauptspeicher-Platz zur Verfügung gestellt. Unter anderem sind die Funktionen "feste-" und "variable Regionaufteilung" und der "Puffereffekt" überzähliger Inititors nachzubilden. Während des Simulationslaufes sollen für jedes Programm Speicherort, Speicherzeit (Beginn, Ende) und Verzögerung der Zuweisung protokolliert werden und am Schluß u. a. die Daten für Produktionszeit (ermöglicht Schluß auf Durchsatz), Speichernutzung und Gesamtverzögerung der Zuweisungen ausgeworfen werden.

Die Laufzeit eines Programms ist eine sehr komplexe Funktion aller Systemkomponenten, die u.a. auch stark von Art und Menge der Parallelverarbeitung (Multiprogramming) abhängt.

Bei der Verarbeitung eines Programm-Load ergeben sich also Laufzeiten, die nur für die gerade gegebenen Verhältnisse typisch sind. Bei der Simulation müßten die Zeiten abgewandelt werden. Dafür wäre es erforderlich, das Datenverarbeitungssystem in vielen Einzelheiten nachzubilden; der Aufwand wäre sehr groß. Wenn aber ein einfacher Algorithmus für die Korrektur der Zeiten gefunden werden kann, läßt sich diese Schwierigkeit umgehen. Für die Modellsynthese sind demnach folgende besondere Probleme zu lösen:

1. Ermittlung der Programmgrößen einer repräsentativen Arbeitslast.
2. Ermittlung der zugehörigen Programmlaufzeiten in einer als Bezugssystem dienenden Elektronischen Datenverarbeitungsanlage.
3. Ermittlung der Korrekturfaktoren zur Anpassung der Laufzeiten an das simulierte System und an das jeweilige Ausmaß der Parallelverarbeitung zum Zeitpunkt des Programmlaufs.

Unsere Simulation bezieht sich auf ein bereits existierendes DV-System, das mit Hilfe des Modells optimiert werden soll. Bestandteil des Operating Systems ist ein Software-Monitor (SMF - System Management Facilities), der für verschiedene Zwecke kontinuierlich Daten sammelt, die Aufschluß über Aktivität und Status der Anlage geben. Darunter finden sich auch Größen und Laufzeiten der ausgeführten Programme. Damit stehen die Daten, die unter Punkt 1 und 2 erwähnt wurden, als Modell-Eingabe zur Verfügung; es muß nur noch eine repräsentative Auswahl getroffen werden.

Die Ermittlung der unter Punkt 3 angeführten Korrekturfaktoren geschah folgendermaßen: Das System, an dem die Daten für die Charakterisierung der Programme gesammelt werden, ist in unserem Fall identisch mit dem, das simuliert werden soll. Infolgedessen kann die Anpassung an veränderte Hard- und Software entfallen. So reduziert sich das Problem auf folgende Frage: Wie ändert sich die Laufzeit eines Programmes in einem Datenverarbeitungssystem in Abhängigkeit von der Anzahl parallel aktivierter Programme, mit denen es um die Systemkomponenten konkurrieren muß. Für das Simulationsziel ist es uninteressant zu wissen, wie der Einfluß der speziellen Eigenschaften (I/O- oder CPU-intensiv ?) der Programme ist, die zum gegebenen Zeitpunkt gerade parallellaufen, es genügen Mittelwerte.Diese Werte können z.B. mit einem Hardware-Meßgerät ermittelt werden, das es gestattet, den Systemdurchsatz (Indikator: CPU-Aktivität im Problem-Status) in Abhängigkeit von der Anzahl "gleichzeitig" aktiver Regions festzustellen. Über eine einfache Umrechnung kann dann die Laufzeit auf den Durchsatz bezogen werden. Die so gewonnenen Zusammenhänge werden nun als Formel oder Tabelle in das Modell eingebaut und repräsentieren den Einfluß aller Systemkomponenten auf die Programmlaufzeit. Mit diesem Verfahren wurde nicht nur die aufwendige Nachbildung zahlreicher Systemfunktionen umgangen, sondern auch eine relativ kurze Laufzeit des programmierten Modells erreicht.

Ein Faktor, der die Laufzeit des Modells stark beeinflußt, ist die Art der Simulation des Zeitablaufs. Man kann die Zeitschritte eines simulierten Taktgebers für die Auslösung von Modellaktivitäten benutzen. Bei genügend kleiner Schrittweite würden aber in diesem Modell viele Takte ablaufen, die keine oder nur unbedeutende Operationen zur Folge hätten.

Günstiger und für die erforderliche Genauigkeit völlig hinreichend erschien es, nur die Ereignisse "Programmende" zu verwenden. Die Wahl ist sinnvoll, denn erst nach einem solchen Ereignis können neue Zuweisungen versucht werden.

Zwar muß dann die Verweilzeit der Programme im Hauptspeicher bereits zum Zeitpunkt ihrer Aktivierung entsprechend der momentanen Situation im voraus bestimmt werden und kann nicht nachträglich aus den laufend wechselnden Umgebungseinflüssen ermittelt werden. Der Fehler kann aber vernachlässigt werden.

Implementierungshilfen:

Für die Programmierung des Modells standen neben den üblichen Sprachen auch spezielle Simulationssprachen wie GPSS (General Purpose Simulation System) und CSS (Computer System Simulator) zur Verfügung. Deren Verwendung hätte aber bedeutet, daß die Handhabung des Modells durch die stapelorientierte Verarbeitung umständlicher geworden wäre. Da aber die beschriebene Modellierungsmethode bewirkt hatte, daß der zu erwartende Programmierungsaufwand in Grenzen blieb, bot sich die Möglichkeit, auf spezielle Simulationssprachen zu verzichten und APL\360 einzusetzen.

APL\360 ist ein time-sharing-orientiertes System für Dialogverarbeitung am Terminal mit einer Programmiersprache, die sich durch einfache Syntax und fähige Sprachelemente auszeichnet und besonders für mathematische Anwendungen geeignet ist. Die Programme werden bei Ausführung interpretativ bearbeitet, eine Umwandlung entfällt. Die Anwendung versprach beachtliche Vorteile:

1. Programmierung und Test können am Terminal wesentlich schneller durchgeführt werden als bei Stapelverarbeitung.
2. Modifizierungen können beschleunigt durchgeführt und sofort überprüft werden.
3. Modelläufe sind schnell hintereinander in großer Zahl mit veränderten Eingaben durchführbar.

Der einzige Nachteil von APL besteht darin, daß es keine File-Verarbeitung hat und deshalb nicht auf die im System verfügbaren SMF-Sätze zur maschinellen Auswahl und Verarbeitung zurückgegriffen werden konnte. Infolgedessen mußten die zahlreichen Daten zur Charakterisierung der Programme, die den Modell-Input bildeten, über die Tastatur des Terminals eingegeben werden.

Test und
Gültigkeitsbestätigung:

Während der Programmierung wurden am Terminal bereits die einzelnen Programmodule ausgetestet. Das fertig programmierte Modell wurde einem Funktionstest unterworfen, dem sich dann eine umfangreiche Verifikationsphase anschloß. Dabei wurden alle möglich erscheinenden Situationen durchgespielt. Da das Modell die Zuweisungen in allen Einzelheiten protokolliert, ließ sich leicht überprüfen, ob die Regeln, nach denen der Vorgang abzulaufen hatte, auch richtig eingehalten wurden. Auch die korrekte Modifizierung der Programmlaufzeiten konnte so einfach kontrolliert werden. Schließlich wurde in einem Lauf mit einer großen Anzahl echter Daten überprüft, ob die Summe aller Laufzeitmodifikationen bei Berücksichtigung der durchschnittlich erzielten Parallelverarbeitung dem Wert entsprach, der nach den oben erwähnten Messungen mit dem Hardware-Monitor zu erwarten war.

Aufwand, Erfolge, Erfahrungen:

Der Aufwand für Entwurf, Implementierung und Test des Simulationsmodells betrug etwa 6 Wochen. Darin sind die Hardwaremessungen nicht enthalten; die Daten, die für dieses Modell benötigt wurden, waren ein Nebenprodukt von Messungen, die für andere Zwecke gemacht wurden. Der Zeitaufwand für die echten Simulationsläufe (ca. 30) belief sich auf eine weitere Woche.

Die Modelläufe ermöglichten eine quantitative Gegenüberstellung der Vor- und Nachteile von festen und variablen Regions. Die speziellen Ergebnisse waren wichtig für Entscheidungen bei unserem System, lassen sich aber naturgemäß nicht auf andere Verhältnisse übertragen. Einige allgemeinere Bemerkungen können aber gemacht werden.

Variable Regions führen zu einem Durchsatzgewinn, der in besonderen Fällen beachtlich sein kann (z.B. bei hoher Kernspeicherbelegung, niedriger CPU- und I/O-Auslastung, sehr unterschiedlichen Programmgrößen), bewirken dann aber andererseits eine Verzögerung der Kernspeicherzuweisung und damit eine Verlängerung der Jobverweilzeiten. Dadurch wird zwar das System nicht stärker belastet, denn die "wartenden" Programme befinden sich ja nicht im Kernspeicher, aber bei Jobs mit größeren Programmen kann die Verzögerung u.U. unzumutbar werden (Terminarbeiten!). Es müssen dann besondere Maßnahmen getroffen werden, die den normalen Betriebsablauf unterbrechen (Verluste) und diesen Programmen den benötigten Platz verschaffen.
Wenn sich zeigt, daß der Durchsatzgewinn gering ist, z.B. weil das gesamte System schon stark ausgelastet ist, könnte u.U. Hauptspeicherplatz bei gleichem Durchsatz eingespart werden. Sind die Verzögerungen nicht akzeptabel, so bieten feste Regions das bessere Konzept.

Im übrigen bietet das Modell die Möglichkeit, die Anzahl der Initiators so zu optimieren, daß ein akzeptables Verhältnis zwischen Durchsatz (oder Kernspeicherbedarf) und Joblaufzeitverlängerung erreicht wird.

Die Einsatz- und Benutzerfreundlichkeit des Modells ist durch die Verwendung von APL besonders groß; ein fremder Anwender ist rasch mit der Handhabung vertraut, nicht zuletzt, weil alle Eingaben über das Terminal einzeln und verständlich angefordert werden. Bei Abbrüchen, die durch Leitungsfehler oder gewisse Hardwarefehler verursacht werden, ist die laufende Simulation nicht verloren, sondern wird in einen Sicherungsbereich gerettet, so daß nach Beseitigung der Abbruchursache mit dem Lauf fortgefahren werden kann.

Zusammenfassung:

Das vorgestellte Simulationsmodell bietet ein Beispiel dafür, wie mit kombiniertem Einsatz verschiedener Methoden (Datenaufzeichnungen und - Auswertungen, Messungen) und einer für den speziellen Fall geeigneten Programmiersprache die Laufzeit des Modells und auch der Aufwand für die Implementierung gering gehalten werden konnten. Systemanalyse und Modelleinsatz erbrachten wichtige Erkenntnisse und Informationen, die durch das Experiment nicht, oder nur mit großem Aufwand zu gewinnen gewesen wären. Sie ermöglichten uns die Entscheidung für das optimale Kernspeicherkonzept auf einer sicheren Basis.

SIMULATION VON RECHNERNETZEN MIT GETAKTETER INFORMATIONSÜBERNAHME

von U. Herzog, G. Kampe und M. Langenbach-Belz

Institut für Nachrichtenvermittlung und Datenverarbeitung

(Universität Stuttgart)

1. Einleitung

In Rechnernetzen oder rechnergesteuerten Daten- oder Fernsprechnetzen müssen sowohl zwischen den einzelnen Rechnern als auch zwischen einem Rechner und seiner zugehörigen Peripherie Informationen ausgetauscht werden. Dabei müssen die Informationen nacheinander verschiedene Baueinheiten (z.B. Register) durchlaufen, wodurch sie vor ihrer endgültigen Verarbeitung Wartezeiten und weitere Verarbeitungszeiten erfahren. In Realzeitsystemen sind harte Forderungen bezüglich solcher Wartezeiten gestellt, d.h. die Zeit zwischen der Entstehung einer Information und ihrer endgültigen Verarbeitung sollte einen festgelegten Maximalwert nicht bzw. nur in einem geringen Prozentsatz aller Fälle überschreiten. Um zu prüfen, ob die gestellten Forderungen von einem geplanten System erfüllt werden, sowie Engpässe zu finden und zu beseitigen, ist es nötig, den Verkehrsfluß (der Informationen) durch das System zu analysieren und Wartezeiten, Speicherbelastungen etc. zu bestimmen. Die komplexe Feinstruktur solcher Netze und die z.T. auftretenden Abhängigkeiten der Informationen voneinander lassen es oft nicht zu, den Verkehrsfluß mit Hilfe verkehrstheoretischer Verfahren exakt zu berechnen. Deshalb ist man zur Untersuchung des geschilderten Problemkreises neben Näherungsrechnungen größtenteils auf die Simulation angewiesen, d.h. Netzstruktur und Verkehrsfluß werden auf einem Digitalrechner nachgebildet (vgl. auch /2/). Doch auch die Simulation kann nicht in unbegrenztem Ausmaß angewendet werden. So ist es z.B. auch selbst auf den größten Datenverarbeitungsanlagen oft aus Gründen des Speicherplatzbedarfes nicht möglich, die oben beschriebenen Systeme als Ganzes zu simulieren. Außerdem würden auf Grund der Komplexität der Simulationsprogramme die Programmlaufzeiten unvertretbar groß werden. Schließlich sei noch darauf hingewiesen, daß die Simulationsprogramme auch noch ein gewisses Maß an Übersichtlichkeit besitzen sollten (einfachere und verständlichere Dokumentation) und auch mit erträglichem Aufwand sicher ausgetestet werden können. Aus diesen Gründen ist es sinnvoll, ein Gesamtsystem (vollständiges Netz) in einzelne, kleinere Teilsysteme aufzugliedern und dann die einzelnen Teilsysteme zu simulieren. Ein dabei auftretendes Problem ist die Wahl der Schnittstellen zwischen den Teilsystemen. Insbesondere ist die Nachbildung der an den Schnittstellen auftretenden Verkehre von besonderer Bedeutung, da die charakteristischen Eigenschaften dieser Ersatzverkehre (z.B. Ankunftsabstandsverteilungsfunktion) wesentlichen Einfluß auf die charakteristischen Verkehrsgrößen (z.B. Speicherbelastungen, Wartezeiten etc.) der simulierten Teilsysteme ausüben.

Einerseits sollten die Ersatzverkehre möglichst einfach im Simulationsmodell realisiert werden können, andererseits aber sollten sie keine Verfälschungen der Ergebnisse bei der Simulation eines Teilsystems gegenüber der Simulation des Gesamtsystems verursachen.

In den folgenden Abschnitten wird zunächst die prinzipielle Struktur und der Verkehr der untersuchten Gesamt-Netze beschrieben. Anschließend folgt die Einteilung des Gesamt-Netzes in Teilsysteme. Die dabei auftretenden Schnittstellenprobleme werden am Beispiel einer Schnittstelle erläutert. Es wird gezeigt, wie für diese Schnittstelle durch eine systematische Untersuchung ein Ersatzverkehr gefunden wurde, der simulationstechnisch ähnlich einfach zu realisieren ist wie ein Poisson-Verkehr und welcher sehr wirklichkeitstreue Ergebnisse liefert.

Es sei an dieser Stelle bemerkt, daß in dem vorliegenden Bericht nicht detailliert eine Modellbildung gezeigt werden soll (vgl. dazu /1/, /2/, /3/, /4/) oder eine Beschreibung des realisierten, umfangreichen Simulationsprogrammes gegeben werden soll. Vielmehr werden einige prinzipielle Gedanken und Erfahrungen dargelegt, welche während der Erstellung des Programmes auftraten.

2. Struktur und Verkehr der untersuchten Netze

2.1 Struktur

Die prinzipielle Struktur der untersuchten Netze ist in Bild 1 dargestellt.

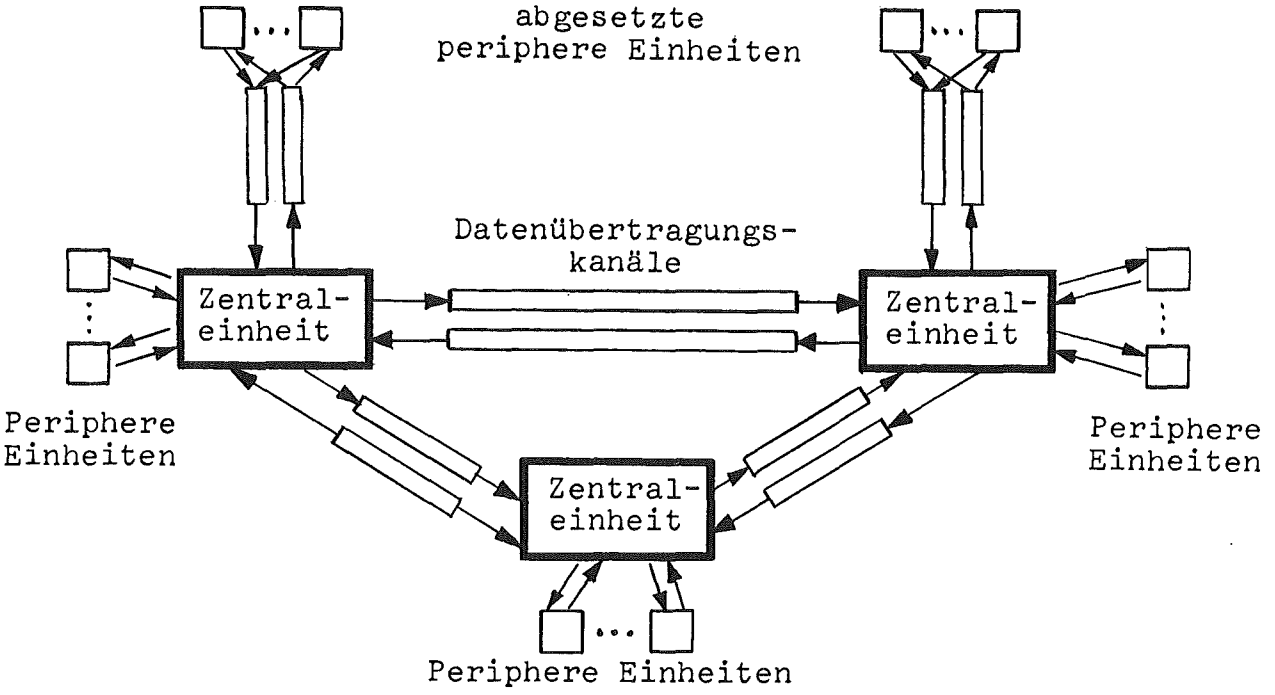


Bild 1: Prinzipielle Struktur der untersuchten Netze

Die in Bild 1 vereinfacht dargestellten Einheiten wie Zentraleinheit (Rechner), Datenübertragungskanäle und periphere Einheiten bestehen in Wirklichkeit aus mehreren Wartespeichern (Registern) und Bedienungseinheiten.

Die Zentraleinheiten sind untereinander jeweils mit einem Paar einseitig gerichteter Datenübertragungskanäle verbunden. Je Zentraleinheit können periphere Einheiten entweder direkt oder über Datenübertragungskanäle (abgesetzte periphere Einheiten) angeschlossen sein.

Als Beispiel für das Auftreten solcher Netzstrukturen in der Praxis können Netze für Steuerinformationen in zukünftigen und z. Zt. in Entwicklung befindlichen rechnergesteuerten Fernsprechnetzen genannt werden. Es kann sich aber auch um Rechnernetze handeln, bei welchen die peripheren Einheiten z.B. Terminals, Hintergrundspeichern oder auch Satellitenrechnern entsprechen können.

2.2 Verkehr

Der Verkehrsfluß in dem Netz wird außer durch die Feinstruktur des Netzes durch den Entstehungsprozeß von Informationen (Anforderungen), den Bedienungsprozeß in den einzelnen Bedienungseinheiten und die Abfertigungsstrategien in den einzelnen Warteschlangen bestimmt. Diese Prozesse bzw. Strategien müssen im Simulationsmodell möglichst genau den tatsächlichen Verhältnissen nachgebildet werden (vgl. /2/).

In dem Netz nach Bild 1 entstehen verschiedene Typen von Anforderungen zufallsmäßig in den peripheren Einheiten. Die Ankunftsabstände zwischen zwei Anforderungen werden durch eine negativ-exponentielle Verteilung beschrieben ("Poisson-Ankünfte"), welche für die meisten in der Praxis auftretenden Fälle eine gute Näherung darstellt (aus Messungen bekannt). Die in der Peripherie entstandenen Anforderungen werden in die Zentraleinheit (Rechner) gebracht, wo sie nach einer evtl. Wartezeit von einer Bedienungseinheit abgearbeitet werden. Die Verarbeitung einer Anforderung in der Bedienungseinheit der Zentraleinheit kann weitere Meldungen an die Peripherie oder an die anderen Zentraleinheiten zur Folge haben, welche ihrerseits wieder weitere Rückmeldungen verursachen können etc. (Folgeinformationen). Durch diese Abhängigkeiten der Informationen voneinander entsteht gewissermaßen ein "rückgekoppelter Informationsfluß" zwischen Zentraleinheit und Peripherie bzw. eine gegenseitige Beeinflussung der Verkehrsflüsse auf den Datenübertragungskanälen.

Die Bedienungseinheit einer Zentraleinheit benötigt zur Verarbeitung der Anforderungen pro Informationstyp ("ursprüngliche Anforderungen" und "Folgeinformationen") eine spezifische, konstante Zeit.

Die Informationsübernahme zwischen den Datenübertragungskanälen bzw. den peripheren Einheiten und den Zentraleinheiten erfolgt in vielen Systemen getaktet, d.h. nur zu bestimmten, festen Zeitpunkten. Es wird also z.B. immer in Taktzeitpunkten von der Zentraleinheit geprüft, ob auf den an sie angeschlossenen Datenübertragungskanälen bzw. in den peripheren Einheiten Anforderungen auf ihre Übernahme in die Zentraleinheit warten. Durch diese Betriebsweise können in den Taktzeitpunkten mehrere Anforderungen gleichzeitig in der Zentraleinheit ankommen (Gruppenankünfte).

3. Aufgliederung in Teilsysteme

Wie schon in Abschnitt 1 näher erläutert wurde, ist es aus verschiedenen Gründen nicht sinnvoll oder sogar unmöglich, das gesamte Netz als Ganzes zu simulieren. Deshalb ist eine Aufgliederung in Teilsysteme erforderlich, für welche jeweils ein eigenes Simulationsmodell bzw. -programm erstellt wird.

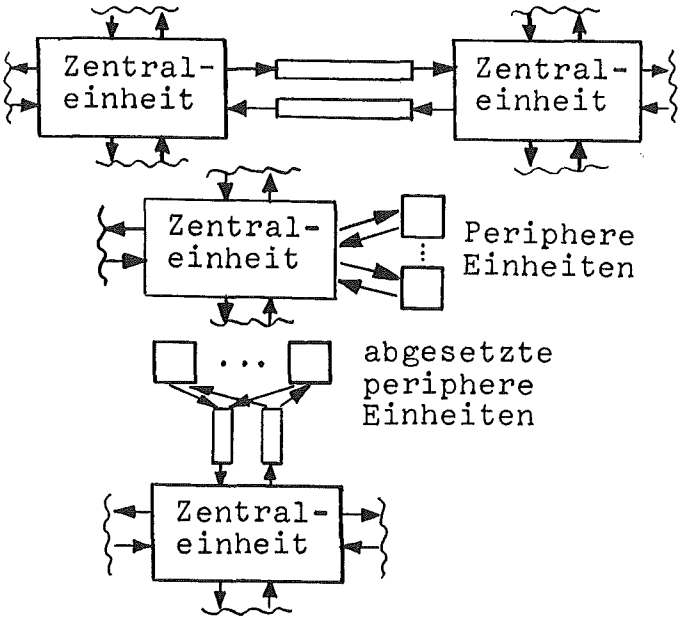
Bei der Aufgliederung in Teilsysteme sollte darauf geachtet werden, daß die einzelnen Teilsysteme so weit wie möglich in sich geschlossene Systeme darstellen, d.h., daß sie möglichst wenig Schnittstellen zu anderen Teilsystemen besitzen. Dadurch wird i.a. gewährleistet, daß ein einzelnes Teilsystem von möglichst wenig Parametern der anderen Teilsysteme abhängig ist.

Bei der Netzstruktur nach Bild 1 und dem in Abschnitt 2.2 beschriebenen Datenverkehr bietet sich deshalb die Einteilung in drei Teilmodelle an, nämlich (vgl. auch Bild 1):

1) Datenverkehr zwischen zwei Zentraleinheiten

2) Datenverkehr zwischen einer Zentraleinheit und direkt angeschlossenen peripheren Einheiten

3) Datenverkehr zwischen einer Zentraleinheit und über Datenübertragungskanäle angeschlossene periphere Einheiten (abgesetzte periphere Einheiten)



Bei jedem dieser Teilmodelle liegt die Schnittstelle zu den anderen Teilmodellen bei der Zentraleinheit (vgl. auch Abschnitt 4). Der von den nicht im Detail mitsimulierten Teilen des Netzes, d.h. von den anderen Teilmodellen, an dieser Schnittstelle ankommende Verkehrsfluß muß durch einen "Ersatzankunftsprozeß" nachgebildet werden, welcher möglichst genau der Wirklichkeit entspricht. Dabei ist von besonderem Interesse, wie sich Vereinfachungen des Ersatzankunftsprozesses (wie z.B. Vernachlässigung der Korrelation der Folgeinformationen) auf die charakteristischen Verkehrsgrößen des Teilmodells auswirken. Für das 1. Teilmodell "Datenverkehr zwischen zwei Zentraleinheiten" wird in Abschnitt 4 am Beispiel der Schnittstelle zwischen Zentraleinheit und Peripherie gezeigt, wie durch systematische Untersuchungen ein geeigneter Ersatzankunftsprozeß gefunden wurde.

Die Simulationsergebnisse aller Teilsysteme müssen schließlich noch auf gegenseitige Übereinstimmung kontrolliert werden. Bei den oben erläuterten Teilmodellen ist z.B. eine von mehreren Möglichkeiten der Kontrolle dadurch gegeben, daß eine Zentraleinheit in allen Teilmodellen enthalten ist. Setzt man nun für alle 3 Teilmodelle gleiches Gesamt-Verkehrsaufkommen voraus, so muß sich bei allen 3 Modellen für die Wartespeicherbelastung, mittlere Wartezeit etc. in der Zentraleinheit ein vergleichbarer Wert ergeben. Andernfalls müssen die Teilmodelle einer Modifikation unterzogen werden.

Aus den Simulationsergebnissen aller Teilsysteme lassen sich dann Schlüsse auf das Gesamtsystem ziehen (wobei allerdings die Abhängigkeit der einzelnen Teilsysteme voneinander nicht außer Acht gelassen werden darf !).

4. Beispiel für die Nachbildung des Verkehrsflusses an einer Schnittstelle eines Teilsystems

In diesem Abschnitt soll für das Teilmodell "Datenverkehr zwischen zwei Zentraleinheiten" am Beispiel der Schnittstelle zwischen Zentraleinheit und Peripherie gezeigt werden, wie ein geeigneter Ersatzankunftsprozeß gefunden wurde (vgl. Bild 2).

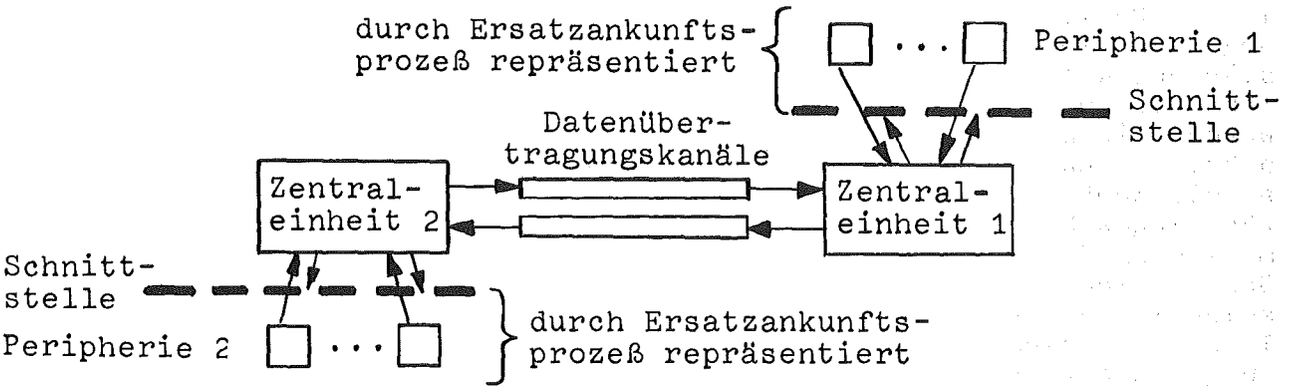


Bild 2: Teilmodell "Datenverkehr zwischen zwei Zentraleinheiten"

Im folgenden wird nun mit Hilfe schrittweise vereinfachter Modelle untersucht, wie der Ersatzankunftsprozeß (vgl. Bild 2) gewählt werden muß, welcher den von der Peripherie kommenden Verkehrsfluß repräsentieren soll. Dabei wird von der Zielvorstellung ausgegangen, den Ersatzprozeß simulationstechnisch möglichst einfach realisieren zu können, andererseits aber dadurch den Verkehrsfluß zwischen den Zentraleinheiten nicht zu verfälschen.

4.1 Ausführliches Schnittstellenmodell (Modell 1)

Um den Einfluß der Vereinfachungen des Ersatzankunftsprozesses von der Peripherie auf die Wartezeiten in der Zentraleinheit zu untersuchen, ist es zunächst einmal nötig, als Ausgangspunkt ein Modell für den Datenverkehr zwischen Peripherie und Zentraleinheit aufzustellen und zu simulieren. Bild 3 zeigt ein Beispiel für ein solches Modell.

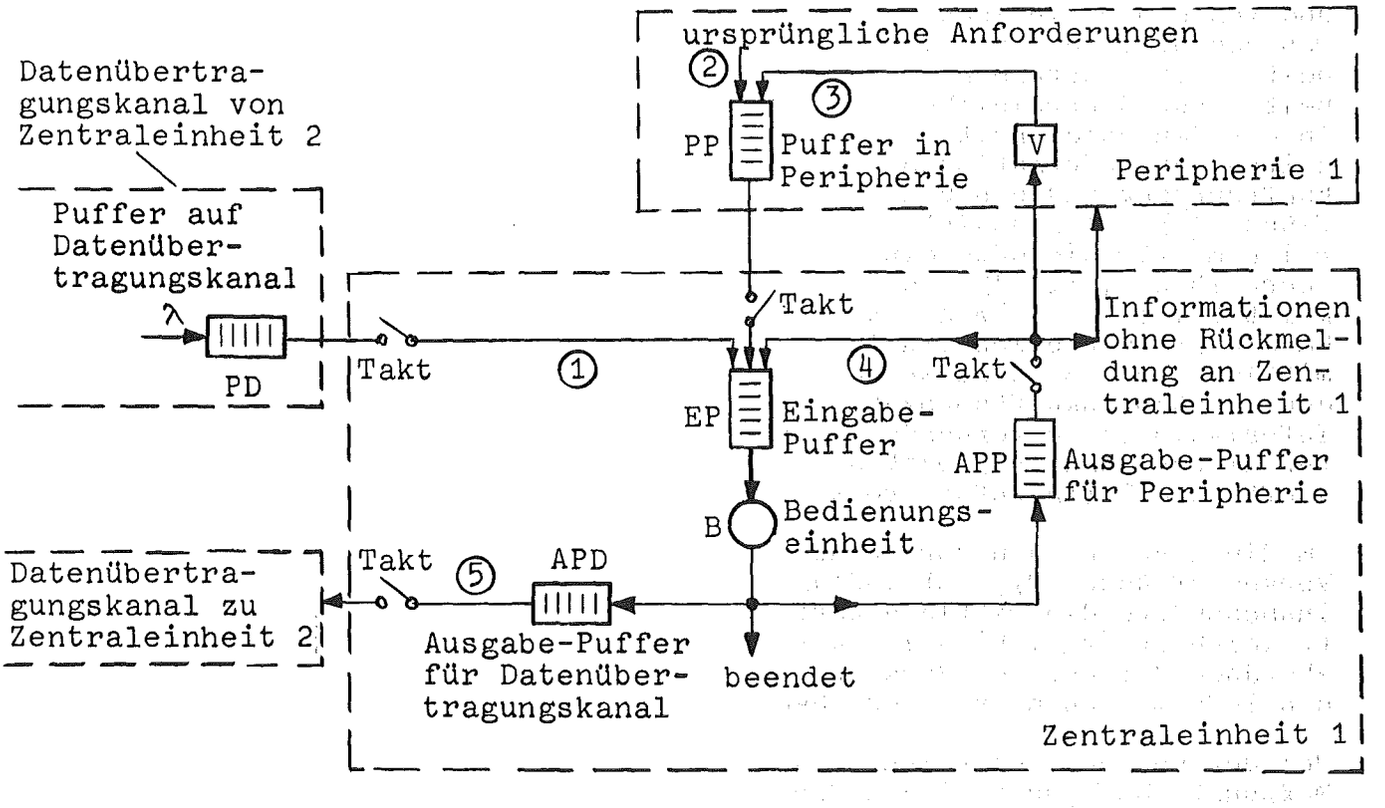


Bild 3: Modell für Datenverkehr zwischen Zentraleinheit und Peripherie (Modell 1). Der taktmäßige Austausch der Informationen zwischen Datenübertragungskanal bzw. Peripherie und Zentraleinheit ist durch die mit "Takt" bezeichneten Schalter dargestellt. Pro Takt wird aus jedem Puffer maximal 1 Anforderung entnommen.

In der Peripherie 1 entstehen Anforderungen (ursprüngliche Anforderungen ② in Bild 3) mit negativ exponentiell verteilten Ankunftsabständen, welche im PP (vgl. Bild 3) bis zu ihrer Abholung durch den Takt warten müssen. Nach ihrer Übernahme in den EP der Zentraleinheit 1 werden sie nach einer evtl. Wartezeit von der Bedienungseinheit B bearbeitet. Hat die Verarbeitung der Anforderung eine Folgeinformation an die Zentraleinheit 2 zur Folge, so wird diese anschließend in den APD eingeschrieben. Entsprechendes geschieht mit einer Folgeinformation an die Peripherie 1, welche in den APP eingeschrieben wird. Schließlich kann es auch vorkommen, daß die verarbeitete Anforderung keine weitere Folgeinformation verursacht. Diese 3 verschiedenen Möglichkeiten sind in Bild 3 durch die Verzweigung unterhalb der Bedienungseinheit angedeutet. Eine ähnliche Verzweigungsstelle befindet sich auch nach der getakteten Entnahme der Informationen aus dem APP. Entweder haben aus dem APP entnommene Informationen keine Rückmeldung mehr an die Zentraleinheit zur Folge, oder sie haben eine direkte Rückmeldung zur Folge, welche in den EP geschrieben wird (z.B. Quittungen, daß bestimmte Informationen die Zentraleinheit verlassen haben) oder aber sie haben erst nach einer bestimmten Verzögerungszeit V in der Peripherie eine Rückmeldung zur Folge, welche in den PP eingeschrieben wird. Damit ist ein rückgekoppeltes Wartesystem entstanden, welches die zeitliche Korrelation der einzelnen Folgeinformationen sowie die taktmäßige Übernahme der Informationen aus der Peripherie berücksichtigt.
Es ist noch zu erwähnen, daß in Bild 3 nur prinzipiell die möglichen Wege des Verkehrsflusses eingezeichnet sind. Bei der Simulation wurde wesentlich detaillierter zwischen verschiedenen Informationstypen unterschieden (bis zu 13 Typen) und deren Weg durch das Modell 1 eindeutig festgelegt.

Wie aus Bild 3 ersichtlich ist, werden alle an der Zentraleinheit 1 eintreffenden Informationen in dem gemeinsamen EP zwischengespeichert und von einer einzigen Bedienungseinheit B nacheinander abgearbeitet. Das bedeutet, daß der interessierende Verkehrsfluß ① von Zentraleinheit 2 zur Zentraleinheit 1 bzw. der Verkehrsfluß ⑤ von Zentraleinheit 1 zur Zentraleinheit 2 durch die Verkehrsflüsse ②, ③ und ④ in der Zentraleinheit 1 beeinflusst wird. So hängen z.B. die Wartezeiten der Informationen des Verkehrsflusses ① im EP u.a. vom Ankunftsprozeß und den Bedienungszeiten der Informationen der Verkehrsflüsse ②, ③ und ④ ab. Als ein B e w e r t u n g s k r i t e r i u m für die Güte eines Ersatzankunftsprozesses, welcher die Verkehrsflüsse ②, ③ und ④ repräsentieren sollte, wurde deshalb die m i t t l e r e W a r t e z e i t von Informationen im EP gewählt. Um die prinzipiellen Einflüsse eines Ersatzankunftsprozesses auf die mittlere Wartezeit im EP zu untersuchen, ist es nicht unbedingt nötig, den exakten Ankunftsprozeß am PD der von der Zentraleinheit 2 kommenden Informationen zu verwenden. Deshalb wurde der Einfachheit halber für diesen Ankunftsprozeß am PD ein Poisson-Ankunftsprozeß mit der Ankunftsrate λ (Ankünfte pro Zeiteinheit) angenommen.

Im Hinblick auf den vorgesehenen Ersatzankunftsprozeß am EP für die Verkehrsflüsse ②, ③ und ④ wurde bei der Simulation des Modells 1 insbesondere die mittlere Wartezeit w der Informationen im EP der Zentraleinheit (vgl. Bild 7, Kurve 1) sowie die Verteilung der Ankunftsabstände der aus den Verkehrsflüssen ②, ③ und ④ am EP eintreffenden Informationen (vgl. durchgezogene Linie in Bild 4) gemessen.

Bei der Messung der Ankunftsabstandsverteilung zeigte sich, daß die Meßpunkte bei ganzzahligen Vielfachen der Taktzeit (vgl. Kreuze in Bild 4) in halblogarithmischer Darstellung in sehr guter Näherung durch eine Gerade verbunden werden können (vgl. gestrichelte Gerade in Bild 4).

Deshalb wurden in erster Näherung anstelle der getakteten Ankünfte die betrachteten Ankunftsabstände als unabhängig voneinander und negativ exponentiell verteilt angesehen. Dadurch entstand das in Abschnitt 4.2 beschriebene, stark vereinfachte Modell.

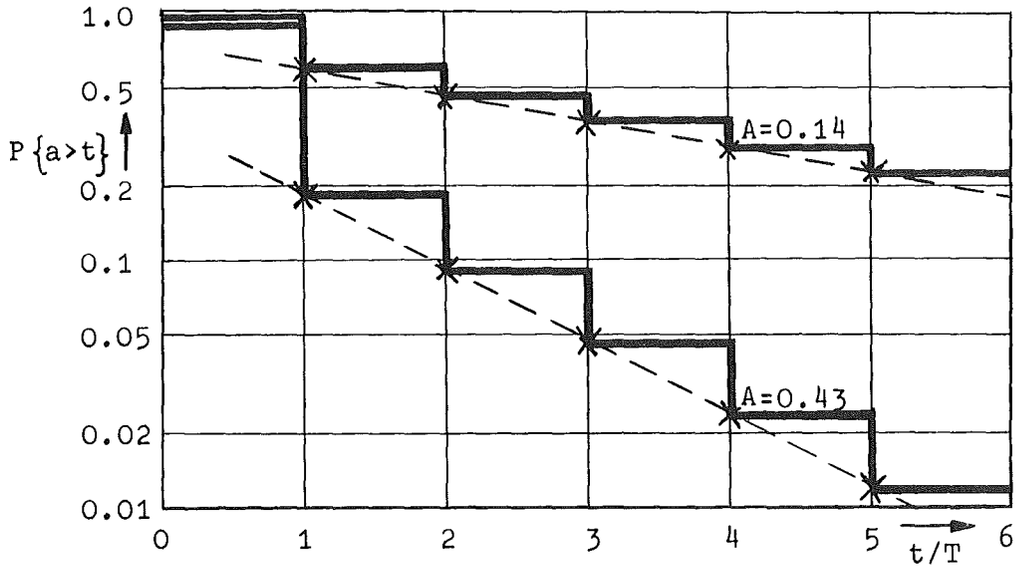


Bild 4: Meßwerte der Ankunftsabstandsverteilung P{a>t} der Informationen aus den Verkehrsflüssen ②, ③ und ④ am EP im Modell 1 (vgl. Bild 3). Taktzeit T=5 Zeiteinheiten. Parameter A = Angebot an Bedienungseinheit = $\sum_i \lambda_i h_i$ Erlang, wobei λ_i = Ankunftsrate von Informationstyp i; h_i = Bedienungsdauer von Informationstyp i.

4.2 Stark vereinfachtes Modell (Modell 2)

In dem stark vereinfachten Modell nach Bild 5 (Modell 2) wird die gesamte in Bild 3 eingezeichnete Rückkopplung weggelassen und der von den Verkehrsflüssen ②, ③ und ④ herrührende Ankunftsprozeß am EP durch einen Poisson-Ankunftsprozeß mit gleicher Ankunftsrate ersetzt (vgl. Ende von Abschnitt 4.1). Somit ist die taktmäßige Informationsübernahme vom PP und APP in den EP vernachlässigt. Für die Informationen des Ersatzankunftsprozesses wird bei der Übernahme vom EP in die Bedienungseinheit mittels einer Zufallszahl entschieden, um welchen Informationstyp es sich handeln soll. Damit ist die zeitliche Korrelation der Folgeinformationen vernachlässigt.

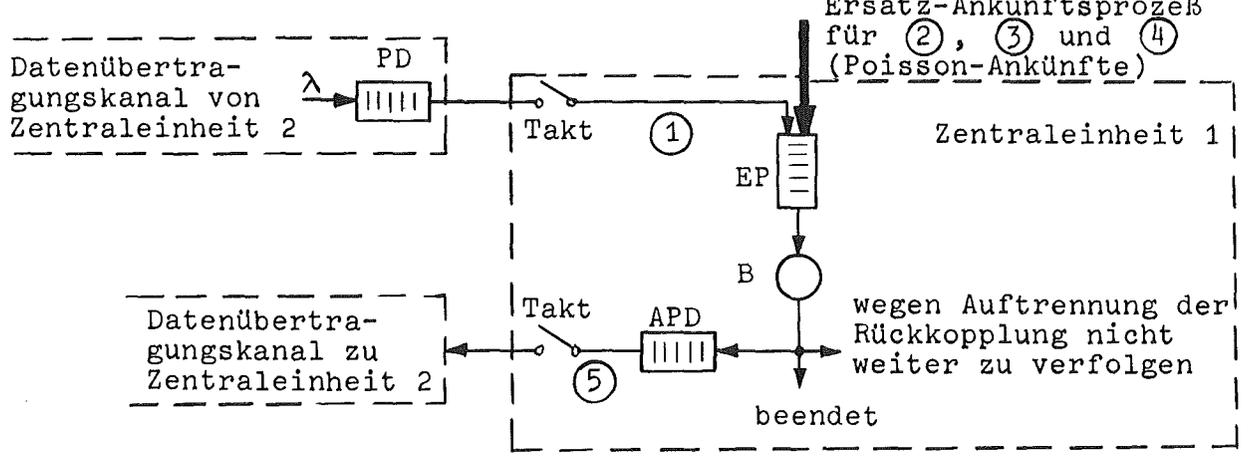


Bild 5: Stark vereinfachtes Modell für den Datenverkehr zwischen Zentraleinheit und Peripherie (Modell 2). Abkürzgn. s. Bild 3.

Ein Vergleich der Simulationsergebnisse für die mittlere Wartezeit w im EP des Modells 1 und des Modells 2 ergab, daß die Näherung mit Modell 2 gegenüber dem Modell 1 nur sehr grobe und z.T. unbefriedigende Ergebnisse lieferte (vgl. Bild 7, Kurve 2). Es mußte also nach einer besseren Näherung als einem Poisson-Ankunftsprozeß gesucht werden. Die ermittelte, bessere Näherung wird im nächsten Abschnitt 4.3 erläutert.

4.3 Verbessertes, vereinfachtes Modell (Modell 3)

Bei dem verbesserten, vereinfachten Modell (Modell 3) wurde die Struktur des Modells 2 beibehalten (vgl. Bild 5) und auch die zeitliche Korrelation der Folgeinformationen innerhalb des Ersatz-Ankunftsprozesses vernachlässigt. Allerdings wurde nun für den Ersatz-Ankunftsprozeß die getaktete Übernahme in den EP berücksichtigt.

Die im Modell 1 gemessene Verteilung der Ankunftsabstände ist eine Treppenfunktion, deren Eckpunkte bei ganzzahligen Vielfachen der Taktdauer in halblogarithmischer Darstellung näherungsweise auf einer Geraden liegen (vgl. durchgezogene Linie in Bild 4). Die Treppenfunktion kommt durch den Takt zustande, da ja immer nur zu Taktzeitpunkten Informationen am Eingabe-Puffer ankommen können und dadurch die Ankunftsabstände nur diskrete Werte, nämlich ganzzahlige Vielfache der Taktdauer, annehmen können. Bemerkenswert ist außerdem, daß die Wahrscheinlichkeit für einen Ankunftsabstand >0 nicht 1, sondern <1 ist! Die Ursache dafür ist, daß pro Takt mehrere Informationen in den Eingabe-Puffer übernommen werden können (Gruppenankünfte).

Für die Verteilung der Ankunftsabstände des Ersatz-Ankunftsprozesses wurde deshalb anstelle des Poisson-Prozesses in Bild 4 eine Treppenfunktion verwendet. Die Realisierung dieser Treppenfunktion im Simulationsprogramm wäre prinzipiell durch die Angabe einer Zuordnungstafel möglich, welche eine gleichverteilte Zufallsvariable in eine beliebig verteilte andere Zufallsvariable transformieren könnte. Die Anwendung dieser Möglichkeit erschien aber aus folgenden 2 Gründen nicht besonders geeignet:

1) Bei jeder Bestimmung eines Ankunftsabstandes hätte ein Zugriff zu der Zuordnungstafel erfolgen müssen. Da ein solcher Zugriff aber relativ viel Rechenzeit in Anspruch nimmt, wäre die Laufzeit des gesamten Simulationsprogrammes erheblich vergrößert worden.

2) Um überhaupt eine Zuordnungstafel eingeben zu können, wären pro Belastungsfall für das System eine große Menge von Wertepaaren (Ankunftsabst./Wahrsch.) für die Treppenfunktion nötig gewesen. Die Schwierigkeit hätte nun darin bestanden, überhaupt zu geeigneten Zahlenwerten für diese Wertepaare zu kommen.

Deshalb wurde angestrebt, die Treppenverteilung simulationstechnisch ähnlich einfach zu erzeugen wie die negativ exponentielle Verteilung. Aus diesen Überlegungen heraus entstand die folgende Näherung:

Wie schon oben erwähnt, liegen die unteren Eckpunkte der Treppenfunktion bei halblogarithmischer Darstellung näherungsweise auf einer Geraden. Deshalb wird als "untere Berandungskurve" der Treppenfunktion eine Exponentialfunktion angenommen. Diese Exponentialfunktion wird aber bei t=0 nicht durch 1, sondern durch q<1 gelegt, um die Gruppenankünfte zu berücksichtigen (vgl. Bild 6).

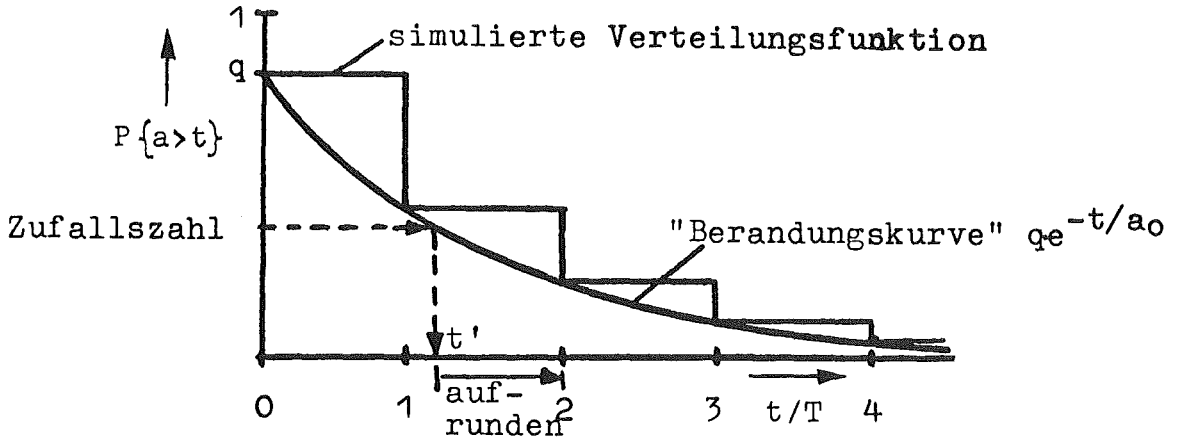


Bild 6: Simulierte Verteilungsfunktion P{a>t} der Ankunftsabstände des Ersatz-Ankunftsprozesses

Die Bestimmung des momentanen Ankunftsabstandes erfolgt nun derart, daß man von einer zwischen 0 und 1 gleichverteilten Zufallszahl ausgeht. Liegt diese Zufallszahl zwischen q und 1 (vgl. Bild 6), so wird der momentane Ankunftsabstand gleich 0 gesetzt (d.h. Gruppenankunft findet statt). Liegt die Zufallszahl zwischen 0 und q, so wird entsprechend wie bei einem Poissonangebot rechnerisch eine zugehörige Zeit t' ermittelt, hier allerdings mittels der Berandungskurve $q \cdot e^{-t/a_o}$ (vgl. gestrichelte Pfeile in Bild 6). Der so gefundene Wert t' wird auf das nächste ganzzahlige Vielfache der Taktzeit aufgerundet und als momentaner Ankunftsabstand verwendet. Dadurch ist die Treppenfunktion simulationstechnisch ähnlich einfach verwirklicht wie beim Poisson-Angebot.

Die Frage ist nun nur noch, wie bei einem gegebenen Belastungsfall die Werte q und a_o der Berandungskurve gewählt werden. Dazu können die folgenden Betrachtungen durchgeführt werden:

Der tatsächliche mittlere Ankunftsabstand a_m der Informationen muß durch die Treppenkurve in Bild 6 erzeugt werden. Dieser tatsächliche mittlere Ankunftsabstand a_m läßt sich aber auch formal aus der Berandungskurve berechnen. Er ist:

$$a_m = q \cdot \frac{T}{1-e^{-T/a_o}} \quad \text{, wobei T = Taktzeit.} \qquad (1)$$

Gibt man den tatsächlichen Mittelwert a_m der Ankunftsabstände vor, so ergibt sich der Wert a_o für die Berandungskurve aus Gleichung (1) zu:

$$a_o = \frac{-T}{\ln(1-q \cdot \frac{T}{a_m})} \qquad (2)$$

Zum Vergleich der Ergebnisse des Modells 1 und des oben beschriebenen Modells 3 wurde nun noch pro Belastungsfall der Wert von q jeweils den Meßergebnissen des Modells 1 entnommen (Wahrsch. für einen Ankunftsabstand >0). Es zeigte sich dabei, daß die Ergebnisse des vereinfachten Modells 3 sehr gut mit den Ergebnissen des ausführlichen Modells 1 übereinstimmten (vgl. Bild 7, Kurve 3). Das bedeutet, daß die zeitliche Korrelation zwischen den Ankünften der Folgeinformationen (welche beim vereinfachten Modell ja nicht berücksichtigt war) von vernachlässigbarem Einfluß auf die Ergebnisse ist. Der Haupteinfluß kommt von dem

getakteten Ankunftsprozeß und den dadurch verursachten Gruppenankünften. Dies zeigte sich auch darin, daß die Ergebnisse des Modells 3 stark von dem gewählten Wert von q abhängen. Es sei noch bemerkt, daß für einen vorgegebenen Wert a_m aus Gleichung (2) die Forderung $q<a_m/T$ folgt.

Der oben beschriebene Ersatzankunftsprozeß kann nun in dem Teilmodell "Datenverkehr zwischen zwei Zentraleinheiten" (vgl. Bild 2) eingesetzt werden, indem er dort den Verkehrsfluß von der Peripherie zur Zentraleinheit wirklichkeitstreu ersetzt. Dabei sind pro Belastungsfall für den Ersatz-Ankunftsprozeß nur 2 Parameter nötig, nämlich der tatsächliche mittlere Ankunftsabstand a_m und der Wert q. Der Wert a_o für die Berandungskurve der Treppenfunktion wird dann nach Gleichung (2) innerhalb des Simulationsprogrammes selbst ausgerechnet.

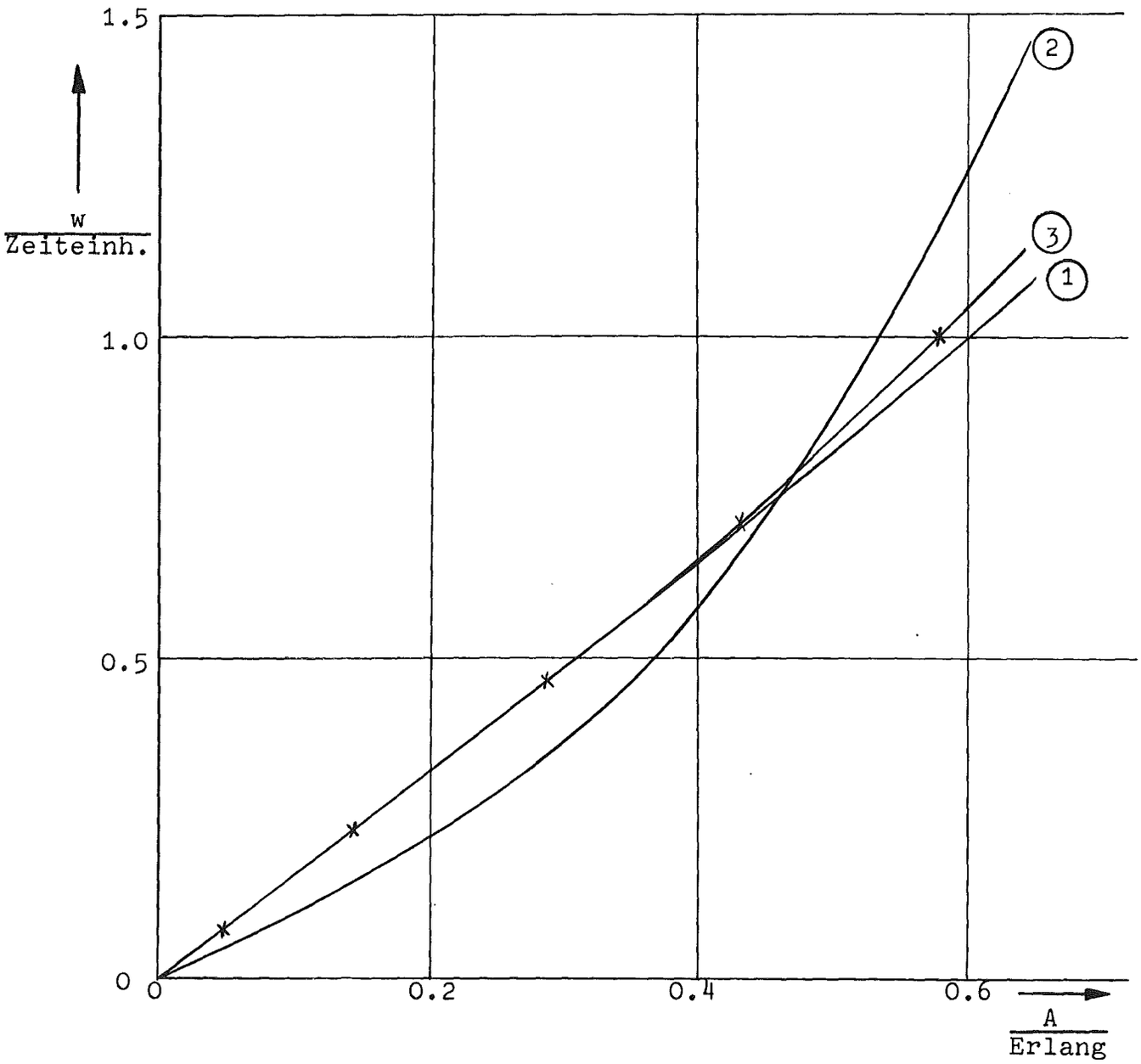


Bild 7: Mittlere Wartezeit w aller Informationen im EP als Funktion des Angebots A an die Bedienungseinheit.
Bedienungsdauer aller Informationstypen einheitlich h = 2 Zeiteinheiten. Taktzeit T = 5 Zeiteinheiten
① : Modell 1 (Abschnitt 4.1)
② : Modell 2 (Abschnitt 4.2)
③ : Modell 3 (Abschnitt 4.3)

5. Zusammenfassung und Ausblick

In dem vorliegenden Bericht wurden einige prinzipielle Überlegungen zur Simulation von Rechnernetzen mit getakteter Informationsübernahme geschildert. Ziel der in SIMSCRIPT (vgl. /1/,/2/) durchgeführten Simulation war es, den Verkehrsfluß in den Netzen zu analysieren (Bestimmung von Wartespeicherbelastungen, Wartezeiten etc.) und Engpässe aufzudecken.

Wegen der Komplexität des betrachteten Netzes sowie einer besseren Übersichtlichkeit der Simulationsprogramme war es nötig, das Gesamtsystem in einzelne Teilmodelle aufzugliedern. Durch die Aufteilung in Teilmodelle ergaben sich Schnittstellenprobleme, insbesondere bezüglich der an diesen Schnittstellen auftretenden Verkehrsflüsse. Am Beispiel einer Schnittstelle eines Teilmodelles wurde gezeigt, wie wichtig die Annahme geeigneter (einfacher, aber genauer) Ersatzankunftsprozesse von nicht im Detail mitsimulierten Teilen des Netzes für die Simulation ist. Dabei wurde mit Hilfe schrittweise vereinfachter Modelle ein Ersatzankunftsprozeß gefunden, welcher einerseits simulationstechnisch sehr einfach zu realisieren ist, der andererseits aber die realen Verhältnisse sehr wirklichkeitstreu nachbildet.

Es wäre zusätzlich noch zu bemerken, daß die gewonnenen Simulationsergebnisse auch noch dazu verwendet wurden, parallel durchgeführte analytische Näherungsrechnungen für den Verkehrsfluß auf ihre Genauigkeit und Gültigkeit hin zu prüfen.

Weiterführende Untersuchungen, welche z.Zt noch im Gange sind, befassen sich insbesondere mit dem Problem der "Verwaltungszeiten" in den Zentraleinheiten. Dabei wird wieder mit Hilfe der Simulation analysiert, wie sich Verwaltungszeiten (z.B. Eingabe/Ausgabe-Verwaltungszeiten in den Taktzeitpunkten) auf den Verkehrsfluß auswirken.

Literatur:

/1/ Kampe,G.: Simscript.
Vieweg-Verlag, Braunschweig, 1971.

/2/ Kampe,G.; Kühn,P., Langenbach-Belz,M. Simulation in der Nachrichtenverkehrstheorie: Problemstellungen und Programmiersprachen. Workshop über Methodik der rechnergestützten Simulation, 10.5.-11.5.73, Karlsruhe.

/3/ Huber, M., Wagner, W.: Simulation von Nachrichtenvermittlungssystemen. aus: Nicht-numerische Informationsverarbeitung, Herausgeber: R. Gunzenhäuser, Springer-Verlag, Wien-New York, 1968.

/4/ Wagner, H., Dietrich, G.: Bestimmung der Verkehrsleistung von Wartesystemen durch künstlichen Fernsprechverkehr. NTZ, 17 (1964) 6 , S. 273-279.

/5/ Kümmerle, K.: Ein Vorschlag zur Berechnung der Vertrauensintervalle bei Verkehrstests. A.E.Ü., 23 (1969) 10, S. 507-511.

Anmerkung:

Den Herren Dipl.-Ing. Otto Neff und Dipl.-Ing. Gebhard Thierer sei an dieser Stelle ein besonderer Dank ausgesprochen, da sie im Rahmen ihrer Diplomarbeiten aktiv an den im vorliegenden Bericht geschilderten Untersuchungen mitgearbeitet haben.

Simulation von Rechnerverbundsystemen - Systemanalyse und Modellsynthese

von Oswald Drobnik

1. Einführung

Ein Rechnerverbundsystem (Rechnernetz) /1/, /2/, /3/ ist eine Menge untereinander verbundener, abhängiger oder unabhängiger Rechnersysteme, die unter der Leitung eines Verbundbetriebssystems (Netzkontrollsystem) miteinander kommunizieren, um gewisse Betriebsmittel wie Programme oder Daten gemeinsam zu benutzen und/oder Belastungsunterschiede auszugleichen und Zuverlässigkeitsanforderungen zu genügen. Viele der beim Aufbau und Betrieb auftretenden Probleme wie etwa Auswahl geeigneter Netzkontrollstrukturen, Bestimmung leistungsfähiger Auftragsvergabealgorithmen, Gewährleistung vorgegebener Auftragsantwortzeiten, Vorhersage des Systemverhaltens bei Ausfall von Netzkomponenten, können auf Grund ihrer Vielschichtigkeit nicht mit rechnerischen (analytischen) Verfahren sondern nur mittels Simulation bearbeitet werden. Ziel dieses Beitrags ist die Betrachtung allgemeiner Aspekte der Simulation von Rechnernetzen zur Entwicklung solcher diskreten Simulationsmodelle, die zur Lösung komplexer Problemklassen konzipiert sind und ohne großen Aufwand zur Untersuchung von Einzelproblemen zugeschnitten werden können.

2. Systemanalyse

Das zu simulierende System setzt sich zusammen aus dem System Rechnernetz und dessen Kommunikation mit der Umwelt, ausgedrückt in der Annahme und Bearbeitung von Aufträgen. Die Systemanalyse beschränkt sich nicht nur auf die Beschreibung der Komponenten des Rechnernetzes (siehe dazu /4/), möglicher Zerlegungen in Teilsysteme und der Wechselwirkungen, sondern beinhaltet auch die Charakterisierung des zu bearbeitenden Auftragsstromes und die Zusammenstellung von Maßgrößen wie Systemverweilzeit von Aufträgen, Durchsatz

usw., die zur Bestimmung der Systemleistungsfähigkeit oder des Systemverhaltens herangezogen werden können.

Von wesentlicher Bedeutung im Umsetzprozeß von einem Systemmodell als Ergebnis der Systemanalyse in ein zu implementierendes Simulationsmodell ist die Beschreibungsart des Systemmodells. Graphenmodelle bieten sich dazu nicht nur wegen ihrer Anschaulichkeit in der Darstellung der Wechselwirkungen funktioneller Einheiten an, sondern vor allem wegen der Existenz formaler Methoden z.B. zum Auffinden eines äquivalenten Minimalsystems oder zur Entscheidung der Determiniertheit des Systems.

Aus auftragsspezifischer Sicht kann man das System beschreiben als ein Netzwerk von Warteschlangen und Bedienungsstationen, welche die funktionellen Einheiten repräsentieren. Das Systemgeschehen stellt sich einem Auftrag dann als Folge von Warte- und Bedienungsphasen dar. Charakterisiert ist das System /5/ durch die Netzstruktur, die Verteilungen der Ankunfts- und Bedienungsprozesse, das Fassungsvermögen der Warteräume, die Warteschlangenabarbeitungsdisziplinen und evtl. ergänzender Beschreibungen der Auftragsbearbeitung. Dieses Modell verdeutlicht insbesonders die Konkurrenz der im System befindlichen Aufträge um die Bedienungsstationen (siehe Anhang, Bild 1).

Zu einer Beschreibung des Rechnernetzes, die sich mehr an den Wechselwirkungen der Komponenten orientiert, gehen wir von der in Bell /1/ vorgeschlagenen Definition des aktiven Rechnernetzes als Menge interaktierender Prozesse aus. Unter Prozeß wird dabei die Ausführung einer Funktion in ihrer Umgebung verstanden (Aktivphase einer funktionellen Einheit) - z.B. die Abarbeitung eines Programmes in einem bestimmten Rechnersystem.

Interaktion der Prozesse äußert sich explizit im Austausch von Nachrichten oder implizit im Wettbewerb um die beschränkt im System vorhandenen Betriebsmittel. Damit ist eine Formu-

lierung des Rechnernetzes als allgemeines Betriebsmittelsystem im Sinne von Holt /6/ möglich, welches sich zusammensetzt aus den wiederverwendbaren Betriebsmitteln (Speichermedien, Dateien) den konsumierbaren Betriebsmitteln (Botschaften zwischen den Prozessen) und Angaben über die Definitions- und Wirkungsbereiche der Prozesse und ihrer logischen Abhängigkeit. Der Einfluß der von außen kommenden Aufträge äußert sich in der Folge der Prozeßaktivierungen.

Für die Darstellung der Wechselwirkungen zwischen Prozessen wurden leistungsfähige Graphenmodelle entwickelt wie

- der Datenflußgraph von Karp/Miller /7/ zur Darstellung der logischen Abhängigkeit zwischen Prozessen bzw. Prozeßstücken,
- der Anforderungsgraph von Hebalkar /8/ zur Darstellung der Konkurrenz parallel ablaufender Prozesse um Betriebsmittel, die nur in beschränkter Anzahl im System vorhanden sind,
- das Petri-Netz /9/, /10/, /11/ zur Beschreibung der Ablaufsteuerung in Prozeßsystemen.

Raubold/Ullrich /12/ haben aus diesen drei Modellen eines abgeleitet, das die Vorteile eines jeden beinhaltet und sich zur Darlegung vor allem qualitativer Zusammenhänge im Systemablaufgeschehen eignet.

Die bisher angeführten Modelle können rekursiv verfeinert oder vergröbert werden und erlauben, den Bausteinen des Modells eines konkreten Systems eine Interpretation bezüglich realer Systemkomponenten zuzuordnen.

Ein Modell, das nicht nur diese Vorteile bietet, sondern zudem in gewissem Sinne vollständig zu nennen ist, indem es auftragsspezifische und prozeßinteraktive Aspekte gleichermaßen berücksichtigt, ist die Formulierung des Systems als Auswertungsnetz gemäß Nutt /13/. Es ist eine Modifikation des Petri-Netzes und soll zur Verdeutlichung des Beispiels für ein Rechnernetz (Anhang, Bild 2) skizziert werden (zur

exakten Formalisierung siehe /13/). Dem Auswertungsnetz liegt ein gerichteter Graph mit zwei Knotentypen zugrunde, den Stellen und den Transitionen, wobei nur Knoten verschiedenen Typs direkt verbunden sein können. Stellen können mit Marken und diese mit Attributen versehen werden. Ausgehend von einer Anfangsmarkierung der Stellen kann zu weiteren Markierungen der Netzstellen entsprechend der Vorschrift der Transitionsdeklarationen übergegangen werden, wobei zur Lösung von Konflikten gewisse Stellen als Entscheider fungieren. Eine Transitionsdeklaration setzt sich zusammen aus dem Transitionsschema, welches den Fluß der Marken von Eingangsstellen der Transition zu deren Ausgangsstellen festlegt, der Zeitdauer der Transition (Konstante oder Funktion) und der Transitionsprozedur, die den Fluß der Marken regelt und für die Attributmodifikation verantwortlich ist. Das Schalten der Transition ist dabei in 4 Phasen zerlegbar:

1. Die Stellen der Transition entsprechen bis auf eine mögliche periphere Entscheidungsstelle dem Transitionsschema (Entscheidungsprozedur ausführen).
2. Alle Stellen einer Transition entsprechen dem Transitionsschema.
3. Das Schalten der Transition ist in Ausführung (während dieser Aktivphase bleibt die Markierung ungeändert).
4. Die Transition ist geschaltet, die Transitionsprozedur wird ausgeführt und die Markierungen werden dem Schema entsprechend geändert.

Die Verweilzeit einer Marke in einer Stelle setzt sich zusammen aus der Zeit, die verstrichen ist vom Setzen der Marke in die Stelle bis zum Beginn der aktiven Phase der Transition und der Transitionszeit, also insgesamt aus einer Warte- und Bedienungszeit.

Interpretiert man Stellen als Zustände von Aufträgen oder funktionellen Einheiten und das Setzen von Marken als Beginn oder Ende der entsprechenden Zustände, so spiegelt die Folge der Markierungen zum einen den Fluß der Aufträge durch das System und zum anderen die zeitlichen Wechselwirkungen der Prozesse wider. Das Verweilen der Marken in den Stellen gibt den gewünschten Aufschluß über Maßgrößen.

3. Modellsynthese

Die Modellsynthese, deren Aufgabe die Transformation des bei der Systemanalyse erhaltenen Modells in ein funktionsfähiges Simulationsmodell unter Berücksichtigung der vorgegebenen Simulationsziele ist, beinhaltet unter anderen Problemen das der Festlegung der geeigneten Modellierungstiefe, also der Bestimmung der Art und Anzahl der im Modell verwendeten Parameter, Variablen sowie Maßgrößen und des Detaillierungsgrades deren wechselseitiger Beziehungen.

Die Modellierungstiefe, die eigentlich nur an den geforderten Simulationszielen orientiert sein dürfte, kann wesentlich beeinflußt werden durch

- vorgegebene Kosten- und Zeitschranken für die Experimente,
- die Fähigkeiten verfügbarer Hilfsmittel wie statistische Testverfahren, Simulationssprachen oder den Rechner, auf dem die Experimente durchgeführt werden,
- die Anforderungen an die Flexibilität des Modells hinsichtlich Strukturverfeinerungen bzw. -vergröberungen,
- die Möglichkeit der Verwendung gesicherter Kenntnisse über qualitative und quantitative Zusammenhänge von System- bzw. Modellkomponenten und Maßgrößen,
- die Eigenschaften der im Modell vorkommenden stochastischen Prozesse.

Um auf solche Einflüsse sinnvoll reagieren zu können, empfiehlt sich die Modellbildung mittels modularer, bezüglich realer Systemkomponenten interpretierbarer Bausteine, die sowohl durch analytische Modelle als auch durch Simulationsmodelle unterschiedlichen Abstraktionsgrades beschrieben werden können. Dies soll an den Beispielen Kommunikationssystem und Rechnersystem verdeutlicht werden, da diesen in einem komplexen System wie dem Rechnernetz eine Rolle als elementarer Baustein durchaus zukommen kann.

Umfangreiche Untersuchungen von Kleinrock /14/ für Message-Switching-Kommunikationssysteme zeigen, daß solche Systeme im Modell durch analytische Modelle dargestellt werden können, so z.B. durch die funktionale Beschreibung der Verweilzeit einer Nachricht in Abhängigkeit von Aufgabe- und Bestimmungsort, Umfang der Nachricht und wenigen für das Kommunikationssystem charakteristischen Parametern wie etwa Kapazität oder Aufteilung der Nachricht in Versandeinheiten.

Bezüglich analytischer Modelle für Rechnersysteme ist man leider noch nicht soweit. Es existieren zwar sehr viele brauchbare Modelle für Teilbereiche (siehe /15/, /16/, /17/) wie Modelle für den Prozessorverkehr, den Speicherverkehr und den Verkehr zwischen verschiedenartigen Funktionseinheiten wie Prozessor - Peripherie oder Prozessor - Arbeitsspeicher. Komplette analytische Modelle für den Allgemeinfall fehlen jedoch völlig und sind nur herleitbar für ganz spezielle, in Wirklichkeit selten gegebene Voraussetzungen wie etwa die statistische Identität der Aufträge /17/. Möglichkeiten, um dennoch verwendbare Rechnermodelle zu erhalten, bestehen zum einen in der Wahl eines passenden Modells für die hervorstechendste Eigenschaft des Rechnersystems /17/, zum andern in der stufenweisen Entwicklung eines Rechnersystems aus Modellen für Teilbereiche zu einem Modell für das Gesamtsystem, wobei vorteilhaft Modellauslegung und verwendete Wahrscheinlichkeitsverteilungen aufeinander abgestimmt werden können /16/.

4. Abschließende Bemerkungen

Die Geschlossenheit des Auswertungsnetzes von Nutt in der Beschreibung der kausalen Abhängigkeit oder Unabhängigkeit von Ereignissen, die bei Prozeßinteraktionen unter Beeinflussung durch Auftragsspezifikationen auftreten, weist diese Modellierungsart als eindrucksvolles Hilfsmittel zur Konstruktion von Systemmodellen sowie Simulationsmodellen aus. Sie unterstützt die Implementierung von Simulationsmodellen in einer der Simulationssprachen (GPSS, SIMSCRIPT, SIMULA 67) und erleichtert damit auch die Modellverifikation.

Anhang:

An folgendem einfachen Rechnernetz soll in groben Zügen die Systemmodellierung mittels Warteschlangenmodell (Bild 1) und Auswertungsnetz (Bild 2) veranschaulicht werden.

Zwei Rechnersysteme (R_1,R_2) arbeiten unter einem zentral organisierten Netzkontrollsystem, das die Komponenten Eingabeeinheit (E), Scheduler (S), Kommunikationssystem (K) und Ausgabeeinheit (A) besitzt. Die Eingabeeinheit nimmt den Auftrag an und leitet ihn, falls das zur Auftragsbearbeitung heranzuziehende Rechnersystem bereits in der Auftragsbeschreibung festgelegt ist, direkt dem Kommunikationssystem zu. Andernfalls wird zuerst der Bearbeitungsort des Auftrags im Scheduler bestimmt. Das Ergebnis des im Rechnersystem ausgeführten Auftrags wird über das Kommunikationssystem der Ausgabeeinheit übergeben.

Bild 1 zeigt die verschiedenen Stadien der Auftragsbearbeitung durch das Netz.

In Bild 2 wurde das Systemgeschehen präzisiert durch die Darstellung der Konkurrenz der Prozesse Scheduling und Updating um eine Betriebsmittelzustandsliste, die der Scheduler für die Auftragsvergabe benötigt und die gemäß dem Zustandswechsel der Rechnersysteme einem Updating unterworfen wird. Aus Gründen der Übersichtlichkeit wurde der Kommunikationssystemeinfluß unterdrückt.

Interpretation der Stellen:

b_1 (b_{10}): Auftrag in Warteschlange (WS) der Eingabeeinheit (Ausgabeeinheit)

b_2 (b_{11}): Auftrag in Bearbeitung durch Eingabeeinheit (Ausgabeeinheit)

b_3 : Auftrag in WS des Schedulers

b_4 : Auftrag belegt Scheduler, und Scheduler benötigt Zugriff auf Zustandstabelle

b_5 : Auftrag in Bearbeitung durch Scheduler, und Scheduler hat Zugriff auf Zustandstabelle

$b_6(b_7)$: Auftrag in WS von Rechner 1 (Rechner 2)

$b_8(b_9)$: Auftrag in Bearbeitung durch Rechner 1 (Rechner 2)

b_{12} : Auftrag hat Netz verlassen

$c_1(c_{10})$: Eingabeeinheit (Ausgabeeinheit) verfügbar

c_2 : Scheduler verfügbar

c_3 : Zustandstabelle verfügbar

c_4 : Updating ist durchzuführen (es wird Zugriff auf Zustandstabelle benötigt)

c_5 : Updating wird durchgeführt (Zustandstabelle für Updating verfügbar)

c_6 : Updating beendet

c_7 : Aufforderung zum Updating der Zustandstabelle von Rechner 1 oder Rechner 2

$c_8(c_9)$: Rechner 1 (Rechner 2) verfügbar

r_1 : Entscheidet anhand der Auftragsbeschreibung, ob Auftrag Rechner 1, Rechner 2 oder Scheduler zugeleitet wird

r_2 : Entscheidet über die Zuteilung des Zugriffs auf die Zustandstabelle

r_3 : Entscheidet, ob Auftrag Rechner 1 oder Rechner 2 zugeführt wird

Während die Stellen b_2, b_4, b_5 und c_1, c_2, c_3, c_5, c_6, c_{10} nur mit maximal einer Marke besetzt sein dürfen, können die restlichen mit mehreren Marken gemäß ihrer Kapazität belegt werden. So bedeuten bei c_8 oder c_9 mehrere Marken die Möglichkeit, der Markenanzahl entsprechend viele Aufträge dem Rechner zur gleichzeitigen Bearbeitung zu übergeben; bei b_1, b_3, b_6, b_7, b_{10} kann damit die Größe des Warteraumes ausgedrückt werden.

Zur Anfangsmarkierung werden die Stellen c_1, c_2, c_3, c_6, c_8, c_9 und c_{10} besetzt, d.h. alle funktionellen Einheiten sind verfügbar. Das Eintreten von Aufträgen in das Netz entspricht dem Setzen von Marken in b_1.

Interpretation der Transitionen:

Die Voraussetzungen, unter denen eine Transition geschaltet werden kann, sind im Beispiel aus ihrer Funktion unmittelbar ersichtlich und werden deshalb nicht mit aufgeführt.

a_1 : Eingabeeinheit wird von Auftrag belegt

a_2 : Eingabeeinheit wird verfügbar, Auftrag wird gemäß r_1 dem Rechner 1, Rechner 2 oder dem Scheduler zugeführt

a_3 : Scheduler wird von Auftrag belegt

a_4 : gemäß r_2 wird Zustandstabelle an Scheduler oder Updater vergeben

a_5 : Scheduler und Zustandstabelle werden verfügbar, Auftrag wird gemäß r_3 Rechner 1 oder Rechner 2 übergeben

a_6 : Updating beendet, Zustandstabelle wird verfügbar

a_7 : Updatingforderungen von Rechner 1 oder Rechner 2 liegen vor, Zugriff auf Zustandstabelle ist erforderlich

$a_8(a_9)$: Rechner 1 (Rechner 2) werden gemäß Markenanzahl in $c_8(c_9)$ mit Aufträgen belegt, Updatingforderungen werden weitergeleitet

$a_{10}(a_{11})$: Updatingforderungen werden weitergeleitet, Auftrag wird Ausgabeeinheit übergeben, $c_8(c_9)$ erhält zusätzliche Marke

a_{12} : Ausgabeeinheit wird von Auftrag belegt

a_{13} : Auftrag vom Netz bearbeitet, Ausgabeeinheit wieder verfügbar

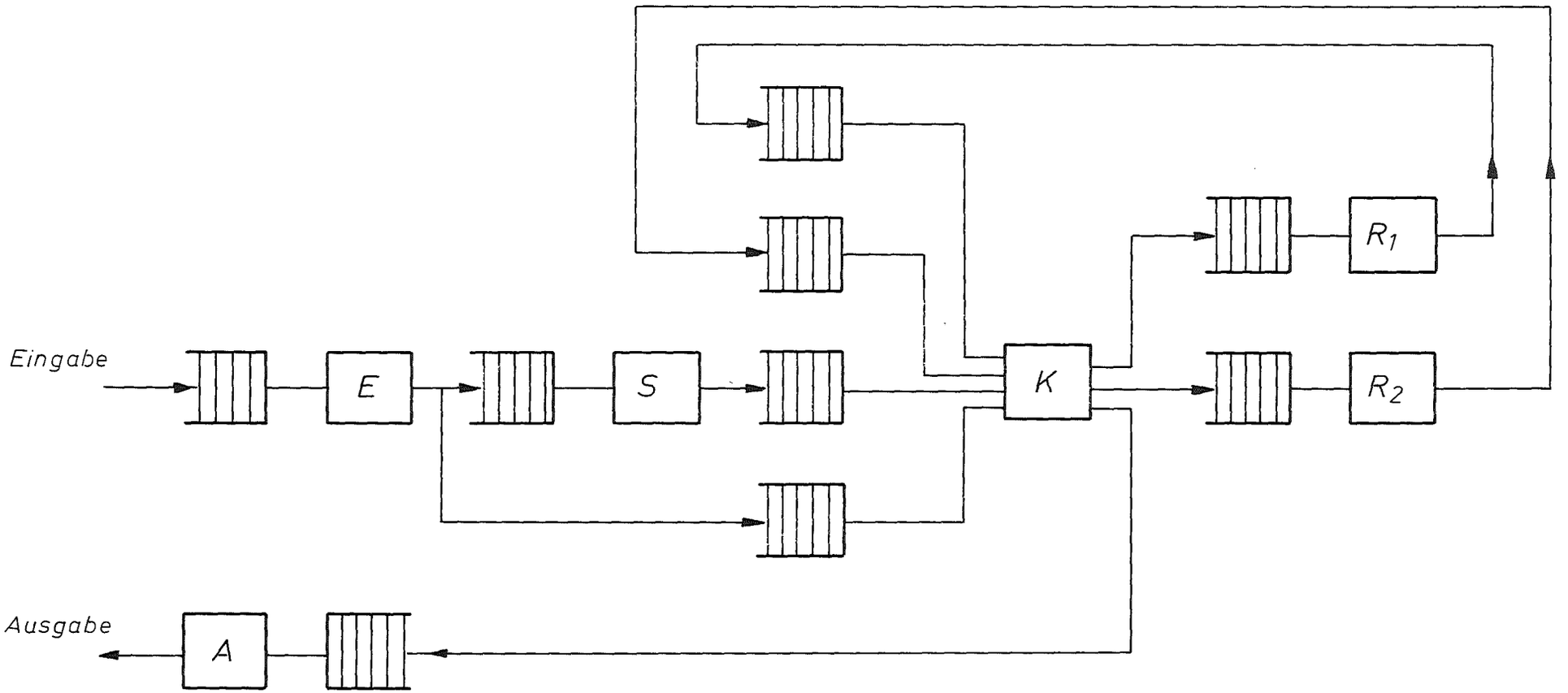


Bild 1 *Beispiel für die Modellierung eines Rechnernetzes mittels Warteschlangenmodell.*

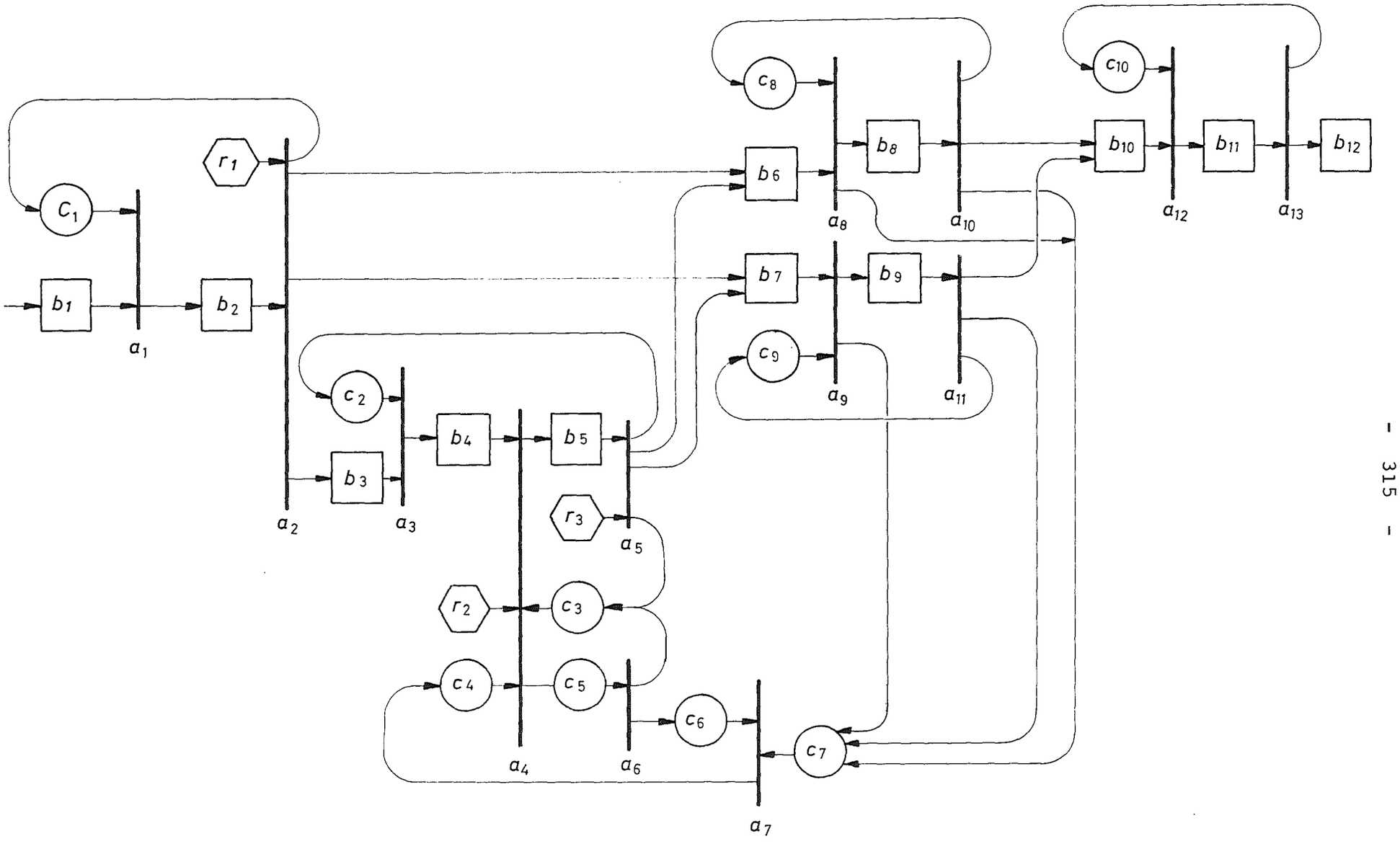


Bild 2 Beispiel für die Modellierung eines zentral organisierten Rechnernetzes mittels Auswertungsnetz

Literatur:

/1/ C.G. Bell, A.N. Habermann, J. McCredie, R. Rutledge, W. Wulf
Computer Networks
Computer Science Review 1969, Carnegie Mellon University, pp. 30-49

/2/ D.J. Farber
Networks: An Introduction
Datamation, April 1972, pp. 36-39

/3/ E. Holler
Betriebsmittelvergabe in heterogenen Rechnernetzen bei dezentralisierter Netzwerksteuerung
Nachrichtentechnische Fachberichte Band 44/1972, VDE-Verlag Berlin, pp. 96-105

/4/ O. Drobnik, E. Holler, F. Schumacher, F. Sell
Statusbeschreibung und Statusüberwachung in einem Rechnernetz
Gesellschaft für Kernforschung Karlsruhe, KFK-Ext. 13/72-5

/5/ T.L. Saaty
Elements of Queueing Theory
McGraw Hill Book Company, New York, 1961

/6/ R.C. Holt
Some Deadlock Properties of Computer Systems
Computing Surveys, Vol.4, Nr.3, September 1972

/7/ R.M. Karp, R.E. Miller
Parallel Program Schemata
Journal of Computer and System Sciences, Nr.3, 1969, p. 147

/8/ P.G. Hebalkar
Coordinated Sharing of Resources in Asynchronous Systems
Record of the Project MAC Conference on Concurrent Systems and Parallel Computation, ACM, New York, 1970, pp. 151-168

/9/ J. Dennis
Concurrency
Advanced Course on Software Engineering, Technische Universität München, Februar 1972

/10/ A.W. Holt, F. Commoner
Events and Conditions
wie /8/, pp. 3-52

/11/ H.J. Genrich
Einfache nicht-sequentielle Prozesse
BMBW-GMD, Bonn, Nr.37, 1971

/12/ E. Raubold, G. Ullrich
Graphenmodelle zur Beschreibung der Wechselwirkung zwischen asynchronen Prozessen
wie /3/, pp. 207-216

/13/ G.J. Nutt
Evaluation Nets for Computer System Performance Analysis
AFIPS, FJCC 1972, PP. 279-286

/14/ L. Kleinrock
Computer Network Research
Advanced Research Projects Agency
Semiannual Technical Report, Juni 1971

/15/ U. Herzog, P. Kühn, A. Zeh
Klassifizierung und Analyse von Verkehrsmodellen
für das Ablaufgeschehen in Rechnersystemen
wie /3/, pp. 181-198

/16/ A. De Cegama
A Methodology for Computer Model Building
AFIPS, FJCC 1972, pp. 299-310

/17/ S.R. Kimbleton
Performance Evaluation - A Structured Approach
AFIPS, SJCC 1972, pp. 411-416

Simulationsmodell eines Rechnerverbundsystems: Experimententwurf und Validation

F. Schumacher

1. Einleitung

Rechnernetze lassen sich allgemein als Warteschlangensysteme darstellen (s. vorhergehenden Vortrag Bild 1). Bei dem Experimententwurf und der Validation von Simulationsmodellen solcher stochastischen Systeme treten neben der Auswahl einer geeigneten Simulationssprache vor allem statistische Problemstellungen (z.B. Stationarität, Einschwingverhalten, Stichprobenumfang, ...) in den Vordergrund, die wegen der Komplexität von Rechnerverbundsystemen - nur in wenigen Spezialfällen besitzt man Kenntnisse über die systeminhärenten Strukturen - schwer zu lösen sind.
Weiterhin sind die üblichen, auf der Unabhängigkeit von Zufallsgrößen beruhenden statistischen Verfahren wegen der Autokorrelation der durch die Simulation generierten Daten ohne Voruntersuchungen meistens nicht anwendbar.

Vor der Diskussion einzelner Probleme sei noch angemerkt, daß sich unser Simulationsmodell momentan in der Implementierungsphase befindet und folgende Überlegungen zur Auswahl einer bestimmten Sprache und von Entwurfs- sowie Validationsmethoden geführt haben.

2. Auswahl einer Simulationssprache

Für die Simulation von Warteschlangensystemen wird meistens die Ereignis- der Zeitfolgesimulation wegen günstigerer Mo-

dellierungsmöglichkeiten und Ersparnis von Rechenzeit vorgezogen, so daß Sprachen wie CSMP oder DYNAMO von vornherein für den Vergleich ausgeklammert wurden.

Algorithmische Sprachen wie ALGOL, FORTRAN oder PL/1 entfielen wegen der Komplexität des zu behandelnden Problems und des daraus resultierenden übermäßigen Aufwands. Aus Gründen der Verfügbarkeit und allgemeinen Anwendbarkeit stellten wir daher für die Auswahl die jeweils neuesten Versionen der Simulationssprachen GPSS, SIMSCRIPT und SIMULA gegenüber (siehe Anhang).

Die Tabelle zeigt eine Überlegenheit von SIMSCRIPT II und SIMULA 67 in Bezug auf GPSS V. Dabei ist jedoch anzumerken, daß im Hintergrund des Vergleichs die Simulation von komplexen Systemen, wie es ein Rechnernetz darstellt, stand.

Obwohl SIMSCRIPT II im Hinblick auf die simulationsspezifischen Anforderungen (Punkt 1-7) SIMULA 67 überlegen ist, entschieden wir uns für SIMULA 67, weil zum einen für unsere Zwecke die in SIMULA 67 angebotenen Verteilungsfunktionen genügen und zum anderen wegen der Autokorrelation die Punkte 5 und 6 wenig ins Gewicht fallen, da unabhängig von den Systemroutinen in GPSS V und SIMSCRIPT II umfangreiche Datenauswertungsprozeduren in das Simulationsprogramm eingebaut werden müssen. Weiterhin entfiel für uns das gewichtige Argument der schweren Erlernbarkeit. Insgesamt erschien uns gerade für die Simulation von Mehrrechnersystemen der Prozeßbegriff und das CLASS-Konzept von SIMULA 67 als eine geeignete Unterstützung sowohl für die Konzipierung als auch für die Implementierung des Simulationsmodells.

3. Stationarität und Einschwingverhalten des Simulationsmodells

Eine Grundannahme unserer Untersuchungen bildet die "normale" Betriebsweise eines Rechnernetzes, d.h. das System soll sich im Gleichgewicht (steady state) befinden. Eine wichtige Bedingung für das Erreichen dieses Zustandes besteht bei Warte-

schlangesystemen darin, daß die Bedienungsrate einer Servicestation größer ist als deren Ankunftsrate. In einem Netz von Warteschlangen und Bedienungsstationen lassen sich aber in Bezug auf die internen Ankunftsströme a priori keine genauen Aussagen über die Stationarität (im weiteren Sinne) treffen. Es müssen also innerhalb von Pilotläufen Tests auf Stationarität wie z.B. der Run- oder Trendtest /1/ vor dem entgültigen Lauf durchgeführt werden. Zuvor kann man sich aber durch Aufzeichnen einzelner Systemgrößen über der Zeit einen groben Einblick in das Systemverhalten verschaffen (z.B. Überlauf von Warteschlangen).

Wichtiger ist das Plotten jedoch für die Analyse des Einschwingverhaltens, die zur Eliminierung der Verzerrung von Messungen durch die Anfangswerte - in unserem Fall leere Warteschlangen und untätige Bedienungsstationen - notwendig ist.
Hierzu existieren grundsätzlich zwei Vorgehensweisen, zum einen startet man mit bestimmten für den Lauf repräsentativen Anfangswerten (entfällt für uns wegen Unkenntnis über das Systemverhalten), zum anderen unterdrückt man für die Messungen die ersten Werte des Laufs. Diese zweite und wohl gebräuchlichste Alternative läßt sich leicht nach oben genanntem Plotten anwenden. Anzumerken ist jedoch, daß dazu mehrere Pilotläufe mit jeweils verschiedenen Zufallszahlen durchzuführen sind. Für eine weitere Diskussion dieser beiden Punkte siehe auch /1,2,3,6/.

4. Simulationslaufzeit und Datenanalyse

Meistens läßt sich das Simulationsproblem auf statistische Fragestellungen wie z.B. den Vergleich von Verteilungsparametern einzelner Systemgrößen reduzieren, für die von der Statistik zahlreiche erprobte Verfahren zur Bestimmung des Stichprobenumfangs und zur Datenanalyse angeboten werden. Nur besitzen diese Verfahren i.a. die Voraussetzung der statistischen Unabhängigkeit.

In unserem Fall liegen aber je nach Systemauslastung zum Teil hoch autokorrelierte Zeitreihen vor. In Bild 1 ist als Beispiel für ein einfaches Warteschlangensystem (Poissonankunftsstrom, exponentiell verteilte Bedienungszeit und Fifo-Disziplin) das Korrelogramm der Warteschlangenlänge für eine Ankunftsrate von 4.5 Aufträgen/Std. und eine Bedienungsrate von 5 Aufträgen/Std. aufgezeichnet (nach /7/).

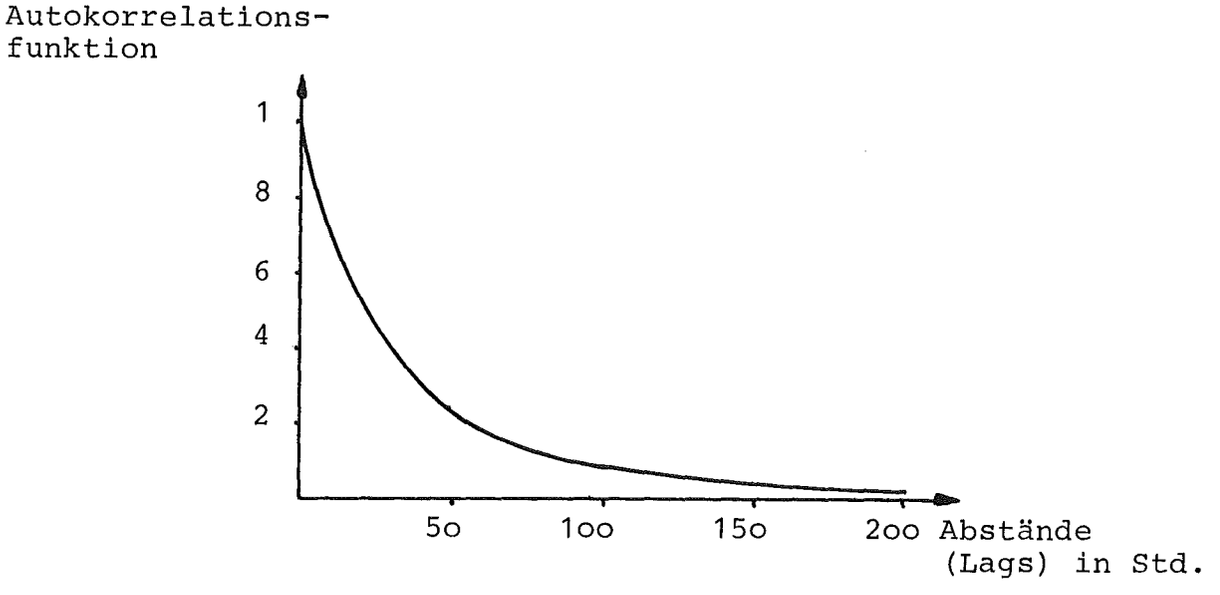


Bild 1. Korrelogramm der Warteschlangenlänge eines einfachen Warteschlangensystems.

Zur Eliminierung der Autokorrelation werden von der Literatur verschiedene Lösungsmöglichkeiten angeboten, die im folgenden kurz gegenübergestellt werden.

Die Grundlage für fast alle dieser Verfahren bildet die Erweiterung des zentralen Grenzwertsatzes von Diananda /11/, der für eine Folge von Zufallsgrößen eines stochastischen Prozesses unter der Voraussetzung der Stationarität im weiteren Sinne, des Verschwindens der Autokorrelation nach endlich vielen Schritten und eines endlichen dritten Moments eine mit wachsendem Stichprobenumfang asymptotische Annäherung des Mittelwertes der Stichprobe an die Normalverteilung beweist.

Die einfachste Vorgehensweise besteht nun darin, die Korrelation nicht zu beachten und den Mittelwert $\bar{x}$ sowie die Varianz des Mittelwertes $var(\bar{x})$ einer Stichprobe mit dem Umfang N mittels den üblichen Formels zu berechnen, also

$$(1) \qquad \bar{x} = \sum_{i=1}^{N} x_i/n$$

$$(2) \qquad var(\bar{x}) = \sigma^2/n \simeq s^2/N = 1/N \cdot \sum_{i=1}^{N} (x_i-\bar{x})^2/(N-1)$$

wobei σ^2 die Varianz der Grundgesamtheit und s^2 deren Schätzung. Da aber in Wahrheit

$$(3) \qquad var_N(\bar{x}) = \sigma^2/N \ (1+2/N \cdot \sum_{k=1}^{N} (N - k)\rho(k) \ )$$

mit der Autokorrelationsfunktion $\rho(k) \geq 0$ $(k=1,2,...N)$, führt diese Methode zu einer Unterschätzung der Varianz des Mittelwertes mit der Konsequenz, daß entweder der Stichprobenumfang zu klein gewählt wird oder das Vertrauensintervall zu günstig ausfällt.

Eine der wohl meist verwendeten Vorgehensweisen, die u.a. von Conway in /6/ vorgeschlagen und ausführlich diskutiert wird, basiert auf der Einteilung eines Simulationslaufs in äquidistante Intervalle, deren Mittelwerte als voneinander unabhängig betrachtet und zur Abschätzung der Varianz des Gesamtmittelwertes herangezogen werden. Die Länge eines Intervalls wird zwar im vorweg festgelegt, soll aber nach Conway im Rahmen von Pilotläufen durch Korrelationsmessungen überprüft und je nach Ergebnis geändert werden. Die Anzahl der Intervalle wird dann in Abhängigkeit vom Rechenaufwand auf der einen und von der gewünschten Anzahl der Freiheitsgrade auf der anderen Seite bestimmt.

Obwohl diese Methode einen relativ geringen Aufwand benötigt und bei kleineren Simulationsstudien aus unserer Erfahrung

durchaus mit Erfolg anwendbar ist, birgt sie doch gewichtige Nachteile. Als erstes ist die Intervalleinteilung, da sie im vorweg erfolgt, meistens willkürlich und im Vergleich zu anderen Methoden (s. unten) mit einem Datenverlust behaftet, zumal über höhere Momente wie z.B. die Varianz kaum Aussagen getroffen werden können. Zum anderen erfolgt die Messung der Autokorrelation i.a. über die Autokorrelationsfunktion, die aber nur asymptotisch unkorreliert ist.

Eine interessante Weiterführung der Conway'schen Methode, jedoch mit ähnlichen Fehlern behaftet, wurde von Mechanic und McKay /1o/ entwickelt und praktisch sowie theoretisch ausführlich untermauert. Ausgehend vom Satz von Diananda und der Formel (3) geben die Autoren ein Verfahren an, bei dem der Simulationslauf in Intervalle verschiedenen Umfangs eingeteilt wird, deren Mittelwerte dann als Eingangsgrößen für eine Folge von Kenngrößen benutzt werden. Es wird gezeigt, daß diese Kenngrößen ein Maß für die Autokorrelation zwischen den Mittelwerten einer Intervallreihe bilden, deren Wert mit wachsender Intervallänge gegen Null konvergiert. Unter Vorgabe eines Vertrauensintervalls liefert das Verfahren dann eine minimale Intervallänge, bei der die Korrelation zwischen den Mittelwerten verschwindet.

Der große Vorteil gegenüber dem Conway'schen Verfahren liegt in der im gewissen Sinne adaptiven Bestimmung der Intervallänge und somit in der Vermeidung von Pilotläufen und eines Informationsverlustes durch Überdimensionierung der Intervallänge. Einen Einblick in die Struktur des stochastischen Prozesses liefert dieses Verfahren jedoch auch nicht und versagt z.B. bei oszillatorischem Systemverhalten. Auch wird die Simulationslauflänge im vorweg festgelegt.

Eine der für die Analyse von autokorrelierten Zeitreihen heutzutage geeignetsten Methoden ist die Spektralanalyse, über die

eine ausgiebige Literatur existiert (siehe als Referenz /4,5/) und die für Simulationsanwendungen von Fishman und Kiviat /7,8/ aufbereitet wurde. Obwohl sie in der Anwendung ziemlich aufwendig und zu ihrem Verständnis eine längere Einarbeitungszeit notwendig ist, liefert sie umfangreiche Informationen über die Autokorrelation und die Varianz einer Zeitreihe sowie etwaige oszillatorische Verhaltensweise von Systemgrößen, zum anderen lassen sich nach ihrer Anwendung die üblichen, auf die Unabhängigkeit beruhenden, statistischen Verfahren zur weiteren Datenanalyse anwenden.

Als letztes Verfahren sei an dieser Stelle noch die bekannte Methode der Wiederholungsläufe angeführt:

Sobald sich das Simulationsmodell im eingeschwungenen Zustand befindet, wird der Lauf gestoppt und mit anderen Zufallszahlen wiederholt. Die jeweiligen Endwerte (keine kumulativen Werte) werden dann zur Abschätzung des Mittelwertes herangezogen.

Obwohl das Verfahren unabhängige Stichprobenwerte liefert, ist es wegen der in jedem Lauf enthaltenen Einschwingperiode überaus aufwendig und deshalb nur für Unterschungen von instationären Systemen und der Einschwingperiode zu empfehlen. Weiterhin kann man mit dieser Methode im Gegensatz zur Spektralanalyse nur geringe Informationen über das Zeitverhalten des Prozesses gewinnen.

Als Folgerung aus diesen Überlegungen haben wir uns für die spektralanalytische Methode entschieden, da sie unserem Warteschlangenmodell mit seiner Komplexität auf der einen Seite und der umfassenden Informationsmöglichkeit der Spektralanalyse auf der anderen am ehesten entspricht. Für etwaige Untersuchungen von instationären Systemen werden wir das Verfahren der Wiederholungsläufe benutzen.

Mit den hier angeschnittenen Problemen ist der Rahmen der zum Experimententwurf gehörenden Fragestellungen noch nicht abge-

steckt. Zu anderen Problemkreisen wie z.B. Faktordesign oder Varianzreduzierungstechnik sei auf die Literatur verwiesen /1,2,4,5/. Für unser Modell sind aber obige Probleme entscheidend, zumal die Anwendung von Varianzreduzierungsmethoden für ein komplexes Warteschlangensystem schwierig und im Vergleich zum Nutzen überaus fraglich ist (s. auch /2/ S. 495).

5. Validation des Simulationsmodells

Die Validation, obwohl im Rahmen der beiden Beiträge an letzter Stelle, ist an sich während jeder der in den Titeln der Vorträge genannten Phasen durchzuführen. Doch wegen fehlender Möglichkeiten und geringeren Aufwands wird sie erst nach Implementierung des Modells durchgeführt und setzt sich dann aus zwei Komponenten zusammen:

Zum einen ist zu prüfen, ob sich das Simulationsmodell so verhält, wie es sich der Implementierer vorstellt und zum anderen ist die Gültigkeit der Abbildung des realen Systems zu bestätigen (häufig in der Literatur (siehe z.B. /9/) durch die Differenzierung Verifikation und Validation ausgedrückt).

Zum ersteren gehört vor allem neben einer Bestätigung der Programmstruktur mit Hilfe eines hypothetischen Trace die Anwendung der in den beiden vorhergehenden Abschnitten genannten Verfahren, also Prüfung der Stationarität, des Einschwingverhaltens und der statistischen Datenauswertungsverfahren.

Im Anschluß oder auch paralell zur Verifikation tritt die Phase der Validation. Da sich unser Modell auf ein System bezieht, welches in der Realität nur in speziellen Fällen exisitert, gestaltet sich dieser Schritt als überaus schwierig, zumal allgemein die Validation ein kaum gelöstes Problem darstellt - es läßt sogar noch philosophische Betrachtungen zu (siehe /5/).

Unser Standpunkt ist der, daß ein Modell erst dann validiert ist, wenn sich seine Aussagen in der Praxis, also am realen System bewährt haben. Wir werden also versuchen, die Aussagen unseres Modells so weit wie möglich zu validieren. Dazu gehen wir in folgenden Schritten vor:

-Vergleich von Teilkomponenten des Modells (Rechnersystem, Kommunikationssystem) mit existierenden Systemen, mit mathematischen Modellen und mit anderen Simulationsstudien.

-Vergleich des Gesamtmodells mit mathematischen Modellen unter vereinfachten Modellstrukturen.

Zur Durchführung dieser Vergleiche bietet sich wieder die Spektralanalyse als eine der geeignetsten und informationsmäßig umfassendsten Methoden an (siehe auch /9/).

6. Schlußbemerkungen

Zusammenfassend läßt sich feststellen, daß für die rechnergestütze Simulation auf dem Gebiet der Systemanalyse und Modellsynthese sowie auf dem Sprachensektor eine Vielzahl von geeigneten Verfahren vorliegt, jedoch bezüglich des Experimententwurfs und der Validation zwar die Statistik einige Ansätze für die Simulationsanwendung bereithält, diese aber praktisch wenig erprobt sind und somit die Simulation für den Anwender noch ein überaus erfahrungsorientiertes Hilfsmittel darstellt.

Literaturverzeichnis

/1/ Bendat J.S. und Piersol A.G.:
Measurement and Analysis of Random Data.
Wiley, 1966.

/2/ Mihram G.A.:
Simulation: Statistical Foundations and Methodology.
Academic Press, 1972.

/3/ Gordon G.:
System Simulation.
Prentice Hall, 1969.

/4/ Naylor T.H. (ed.):
The Design of Computer Simulation Experiments.
Duke University Press, Durham, 1969.

/5/ Naylor T.H.:
Computer Simulation Experiments with Models of
Economic Systems.
Wiley, 1971.

/6/ Conway R.W.:
Some Tactial Problems in Digital Simulation.
Management Science, Vol. 1o, No. 1, October 1963.

/7/ Fishman G.S. und Kiviat P.J.:
The Analysis of Simulation-Generated Time Series.
Management Science, Vol. 13, No. 7, March 1967.

/8/ Fishman G.S.:
Problems in the Statistical Analysis of Simulation
Experiments: The Comparsion of Means and the Lenth
of Sample Records.
Communication of the ACM, Vol. 1o, No. 2, Febr. 1967.

/9/ Fishman G.S. und Kiviat P.J.:
The statistis of discrete-event simulation.
Simulation, April 1968.

/1o/ Mechanic H. und McKay W.:
Confidence Intervals for Averages of Dedendent Data in Simulations II.
Technical Report No. ASSD 17-2o2, IBM, Yorktown Heights, N.Y., 1966.

/11/ Diananda P.H.:
Some Probability Limit Theorems with Statistical Applications.
Proc. Camb. Phil. Soc., 1953, 239-246.

/12/ Kubosch S.:
The Structures of the NCC SIMULA COMPILERS and Bench Mark Comparisons with other Major Languages.
2. GI-Jahrestagung 1972.

Anforderungen an eine

A n h a n g

Simulationssprache	GPSS V(G)	SIMSCRIPT II(SII)	SIMULA 67(S 67)
1.Aufbau und Verknüpfung von Datenstrukturen	in G sehr speziell;von Möglichkeiten her S II und S 67 gleich, doch in S 67 wegen CLASS-Konzept einfacher und klarer als in S II.		
	C +	B	A
2.Verwaltung der Simulationszeit	in G unübersichtlich und rechenzeitaufwendig; in S II und S 67 ähnliches Konzept.		
	B	A	A
3.Erzeugung von Zufallszahlen und Verteilungen	Verteilungen in GPSS nur "per Hand" implementierbar; S II verfügt über mehr Systemfunktionen als S 67.		
	C	A	B
4.Datenanalysefunktionen und Routinen zur Datenausgabe	in G zwar starr festgelegt, doch umfangreich; in S II umfassende Unterstützung vorhanden, während S 67 nur über wenige Grundfunktionen verfügt.		
	B	A	C
5.Dateneingaberoutinen, Initialisierung und Wiederstart.	ähnlich 4.		
	B	A	C
6.Fehlerdiagnose- und Testroutinen	in S 67 nicht vorhanden; S II und G ähnliches Konzept.		
	A	A	B

+ Um eine grobe Wertung vorzunehmen, wurden die Relationen A besser B besser C eingeführt.

Anforderungen an eine

Simulationssprache	GPSS V(G)	SIMSCRIPT II(SII)	SIMULA 67(S 67)
7.Modellkonzeptstärke und Anwendungsbreite	G für Job-Shop-Probleme konzipiert; S 67 wegen CLASS-Konzept stärker und flexibler als S II, in Anwendungsbreite jedoch gleich.		
	C	B	A
8.Erlernbarkeit	S 67 schwer erlernbar; für Anfänger G wohl am leichtesten.		
	A	B	C
9.Modellierungs- und Programmaufwand	G zwar starr und deshalb für größere Anwendungen schwierig handhabbar, doch leicht codierbar; S II umfangreicher als S 67.		
	C	B	A
1o.Compilier- und Laufzeiteffektivität	nach einem SIMULA-Implementierer /12/!:		
	C	B	A
11.Systemzuverlässigkeit und -unterstützung des Herstellers	S II und S 67 werden von keinem Computerhersteller unterstützt; für S 67 existiert in Oslo Komitee, für S II Simulations Associates, Inc.; für einen IBM 36o/37o Benutzer:		
	A	C	B
12.Dokumentation, Beispiele	S II und S 67 Syntax in BNF-ähnlicher Form beschrieben; bzgl. Beispielen alle drei gleich.		
	B	A	A

ZUR GUELTIGKEITSBESTIMMUNG DISKRETER SIMULATIONSMODELLE

von

Heinz Beilner
Institut fuer Informatik
Universitaet Stuttgart

Vortrag im Rahmen des workshop
Methodik der rechnergestuetzten Simulation
Karlsruhe, 10.-11.5.1973

Kurzfassung:
Der Beitrag diskutiert die prinzipiellen Probleme, die im Zusammenhang mit der Frage der Vertrauenswuerdigkeit von Simulationsmodellen und deren Ergebnissen auftreten. Gueltigkeitsbestaetigung, als Phase in der Entwicklungsgeschichte eines Simulators verstanden, wird abgesetzt gegen eine ihr vorausgehende Calibrierungsphase und eine ihr folgende Experimentierphase. Die wesentlichen Grundhaltungen gegenueber dem Problem der Gueltigkeitsbestaetigung werden eroertert und Hinweise gegeben, wie dies Problem in einer Reihe konkreter Simulationsprojekte angegangen wurde.

1. Einleitung

Simulationsmodelle haben sich in den letzten 15 Jahren zu einem anerkannten Hilfsmittel bei der Planung, Verbesserung und Leistungsbestimmung komplexer Systeme entwickelt. Die Entwicklung war begleitet von haeufigen Mahnungen an die Adresse der Modellkonstrukteure, nie zu vergessen, dass die Verwendbarkeit ihrer Simulationsmodelle steht und faellt mit deren Faehigkeit, realistische Einblicke in das Funktionsverhalten der nachgebildeten realen Systeme zu ermoeglichen. Dieses Ziel ist in keinem Fall zu erreichen durch den blossen Bau eines Simulators. Vielmehr muessen zusaetzlich Anstrengungen unternommen werden, sich von der Vertrauenswuerdigkeit der Simulatoraussagen in ausreichendem Masse zu ueberzeugen. Erst dann laesst sich u.U. die Berechtigung ableiten, Ergebnisse von Simulatorexperimenten (anstelle der Ergebnisse entsprechender Experimente in der realen Welt) zur Grundlage von Entscheidungen bezueglich der realen Systeme zu machen.

Wie skeptisch man gegenueber Modellen sein sollte, die keiner ausreichenden Pruefung ihrer Vertrauenswuerdigkeit unterzogen wurden, ist in der Literatur deutlich (z.T. recht drastisch) aufgezeigt; hier nur 3 Zitate:
Ernst<5>: "Can our models be trusted, or are they involved exercises in self-deception?"
Fishman/Kiviat<6>: "Large scale models that are not amenable to validation often lead to perplexing, if not misleading, results."
Schrank/Holt<15>: Es muss etwas getan werden "to prevent the construction of models from being exercises in science fiction."

Erfreulicherweise ist in letzter Zeit ein erhoehtes Interesse an der Gueltigkeitsbestaetigung von Simulationsmodellen zu verzeichnen. Wenn auch immer noch eine gewisse Sorglosigkeit bei einer Reihe von Modellkonstrukteuren nicht zu verkennen ist, so stoesst sie doch auf energischen Widerspruch (siehe z.B. Blackmore's<2> aeusserst kritischen Vortrag auf der 1972 SCSC, der ausschliesslich den Versaeumnissen in punkto Gueltigkeitsbestaetigung gewidmet war, die im Rahmen einiger Vortraege auf der 1971 SCSC zu beobachten waren). Wesentlich positiver noch ist zu bewerten, dass in juengster Vergangenheit in einer ganzen Reihe einschlaegiger Veroeffentlichungen methodische Schwierigkeiten der Gueltigkeitsbestaetigung samt entsprechender Loesungsvorschlaege ausgiebig zur Sprache kamen (siehe Schrank/Holt<15>, Naylor/Finger<12>, v.Horn<7>, Nolan<14>); die bevorstehende 1973 SCSC hat "validation" sogar zu einem ihrer zentralen Themen erklaert.

Im vorliegenden Beitrag wird versucht, einige grundsaetzliche Fragen zu eroertern, die mit der Gueltigkeitsbestaetigung von Simulationsmodellen verknuepft sind, sowie Hinweise zu geben, in welcher Weise diese Fragen in diversen konkreten Simulationsprojekten angegangen worden sind. Dabei wird die Gueltigkeit der Ergebnisse von Simulationsmodellen im Zentrum

des Interesses stehen, philosophisch gesehen also ein positivistischer Standpunkt eingenommen werden. Das soll nicht heissen, dass die Bestaetigung der Hypothesen und Annahmen, auf die das Modell sich abstuetzt, als unwesentlich angesehen wird (vgl. auch Naylor/Finger<12>); vielmehr erhoeht eine derartige Bestaetigung die Wahrscheinlichkeit der Gueltigkeit von Simulatorergebnissen. Doch ist die Bestaetigung der Grundannahmen allein nicht hinreichend, um den Einsatz eines Simulators zu rechtfertigen. Die Benutzung eines Simulators, die Uebertragung von Simulatoraussagen auf die reale Welt, kann sich letzten Endes nur auf die Glaubwuerdigkeit der Ergebnisse des Simulators berufen.

2. Gueltigkeitsbestaetigung - eine Entwicklungsphase und ihre Abgrenzung

Simulator und zu simulierendes System sind nicht identisch. Es ist daher auch nicht zu erwarten, dass sie sich identisch verhalten. Eine realistische Einschaetzung der Verhaltensunterschiede zwischen Modell und realer Welt bildet infolgedessen die Basis jedes Versuchs, ein wie auch immer geartetes Vertrauen in die Resultate eines Modells aufzubauen. Des weiteren wird ein solches Vertrauen sicher umso groesser sein, je geringer die Verhaltensunterschiede sind. Verringerung der Verhaltensunterschiede Realitaet/Modell einerseits und Einschaetzung der Verhaltensunterschiede beim fertigen Modell andererseits sind auf den ersten Blick sehr aehnliche Aufgaben. Wie im folgenden gezeigt werden soll, ist es dennoch geboten, sie im Laufe der Entwicklung eines Simulators deutlich zu trennen und in zwei verschiedenen Entwicklungsphasen zu behandeln:

* einer Calibrierungsphase, die auf eine Reduzierung der Verhaltensunterschiede Realitaet/Modell abzielt und in Verfolgung dieses Ziels den Simulator in geeigneter Weise veraendert,
* einer Phase der Gueltigkeitsbestaetigung, deren Ziel eine Abschaetzung der Verhaltensunterschiede Realitaet/Modell ist, wie sie im Verlauf des Einsatzes des Modells, d.h. im Verlauf der abschliessenden Experimentierphase zu erwarten sind und in deren Verlauf der Simulator nicht veraendert wird.

In beiden Phasen hat man es mit dem nicht zu unterschaetzenden Problem zu tun, wie Verhaltensunterschiede zwischen realer Welt und Modellwelt gemessen werden koennen. Dieses gemeinsame Problem darf die Trennlinie zwischen Calibrierung und Gueltigkeitsbestaetigung nicht verwischen: In der Calibrierungsphase suchen wir eine Version des Modells zu finden, die fuer eine bestimmte Menge ausgewaehlter Systemsituationen das Verhalten des realen Systems moeglichst gut annaehert (wir wollen fuer den Augenblick die Verfuegbarkeit entsprechender Vergleichsdaten als gegeben ansehen). Mit anderen Worten, wir passen das Verhalten des Modells dem Verhalten der realen Welt nach Moeglichkeit an, dies aber notgedrungen fuer eine begrenzte Menge von Systemsituationen, die lediglich einen sehr beschraenkten Teil der grossen Menge moeglicher Systemsituationen abdeckt. Nun ist es aber geradezu Ziel jedes

Experimentierens mit dem Modell, das Verhalten des Systems in neuen, von den Calibrierungssituationen abweichenden Situationen vorauszusagen. Es gibt keinen Grund fuer die Annahme, dass Verhaltensaehnlichkeit fuer eine beschraenkte Menge von Situationen (z.B. fuer Calibrierungssituationen) sich automatisch auf andere Punkte im Situationsraum (z.B. auf Experimentiersituationen) uebertraegt. Ausserdem laeuft man bei jeder Calibrierungsanstrengung Gefahr, das Modell "ueberanzupassen", d.h. spezifische Eigenheiten der Calibrierungssituationen bei der Veraenderung des Modells in dieses mit einzubauen. Aus diesen Gruenden kann die Calibrierungsphase allein keine ausreichenden Aussagen darueber liefern, wie die Verhaltensunterschiede Realitaet/Modell fuer Situationen ausserhalb des Calibrierungsbereiches ausfallen werden. Gueltigkeitsbestaetigung, d.h. Abschaetzung der Verhaltensunterschiede fuer die Experimentierphase, ist die Aufgabe einer zusaetzlichen Entwicklungsphase, die jedem Calibrierungsprozess - ja, jeder Aenderung am Simulator - folgen muss und in deren Verlauf die Verhaltensunterschiede Modell/Realitaet fuer zusaetzliche, unabhaengige, bisher nicht betrachtete Situationen geprueft werden muss.

Sicherlich kann die Phase der Gueltigkeitsbestaetigung Unzulaenglichkeiten des Simulators aufdecken, die bis dahin unbekannt waren. Dadurch moegen weitere Aenderungen am Simulator noetig werden, die ihrerseits u.U. sogar eine neue Calibrierungsphase nach sich ziehen. Auch aus der Experimentierphase koennen sich derartige Anstoesse entwickeln. Es waere geradezu ungewoehnlich, wenn wir nicht im Verlauf der Entwicklung eines Simulators zu jedem Zeitpunkt bereits wuessten, wie die naechsten Simulatorverbesserungen im Hinblick auf eine bessere Abbildung der Realitaet aussehen sollten. Calibrierung, Gueltigkeitsbestaetigung und Experimente werden sich insbesondere bei laengerdauernder Benutzung eines Simulators zyklisch abwechseln - weshalb es umso noetiger wird, sich der unterschiedlichen Ziele dieser 3 Entwicklungsphasen stets bewusst zu sein, bzw. diese Unterschiede nicht zu verwischen und damit Gefahr zu laufen, zweifelhafte Schlussfolgerungen zu ziehen.

3. Zur Bestimmung der Verhaltensunterschiede zwischen realer Welt und Modellwelt

Wie schon angedeutet, gehen sowohl Calibrierung als auch Gueltigkeitsbestaetigung davon aus, dass die Verhaltensunterschiede zwischen realer Welt und Simulationswelt in irgendeiner Weise gemessen werden koennen - was trivialerweise nur dann moeglich ist, wenn ueber das Verhalten der realen Welt genuegend Kenntnisse fuer einen solchen Vergleich bestehen. Der Stand der Kenntnisse ueber das Verhalten des realen Systems kann, abhaengig vom konkreten Projekt, sehr verschieden sein. Es gibt Faelle mit umfassenden Kenntnissen, so z.B. wenn das Simulationsprojekt darauf abzielt, die Leistungsfaehigkeit eines Systems zu verbessern, das bereits geraume Zeit im Betrieb ist und ueber dessen Verhalten

regelmaessig Daten aufgezeichnet worden sind. Die Kenntnisse ueber das Verhalten des realen Systems koennen aber durchaus geringer sein bis hin zu Faellen, wo so gut wie nichts ueber das tatsaechliche Verhalten des zu simulierenden Systems bekannt ist, wie beispielsweise bei der Planung eines voellig neuartigen Systems.

Das Problem der Beschaffung von Daten, ein Systemverhalten beschreibend, mit dem das Simulatorverhalten verglichen werden koennte, ist im wesentlichen auf drei verschiedene Arten angegangen worden:

* Der Vergleich mit dem Verhalten eines existierenden Systems, falls moeglich, scheint bei weitem die besten Moeglichkeiten zu bieten. Bei diesem Ansatz wird als treibender input fuer den Simulator der treibende input des entsprechenden realen Systems verwendet (bzw. dessen Abstraktion in Form einer passenden Aufzeichnung des treibenden realen inputs) und der Simulator-output mit dem zugehoerigen output des realen Systems verglichen. Die Begriffe input und output koennen hier je nach konkretem Projekt in verschiedenen Bedeutungen auftauchen. Input kann z.B. die Arbeitslast in Systemen von job-shop-Typ kennzeichnen, aber auch die Wertefolge von Strategie- und Entscheidungsvariablen bei oekonomischen Systemen. Entsprechend ist der Begriff output zu interpretieren. In jedem Falle bezeichnet input jene exogenen Variablen/Ereignisse, die System und Modell "antreiben", output jene endogenen Variablen/Ereignisse, die die "Antwort" von System und Modell beschreiben.

  Dieser Ansatz ist erfolgreich verwendet worden z.B. von
  - Boughton/Naylor<4>, die im Verlauf einer Simulation des monetaeren Sektors der USA eine 7 Jahre ueberstreichende Aufzeichnung des tatsaechlichen Systemverhaltens benutzten,
  - Beilner/Waldbaum<1>,<16> und Noe/Nutt<13>, die im Verlauf von Rechensystemsimulationen ihre Simulatoren mit Aufzeichnungen realer Arbeitslasten (traces) betreiben und dadurch minutioese Vergleiche zwischen dem Verhalten der realen Rechensysteme und dem der Simulatoren anzustellen vermoegen.

* Der Vergleich des Simulatorverhaltens mit dem Verhalten gewisser mathematischer Modelle, die analytische Loesungen zulassen, ist ein weiterer moeglicher Ansatz. Dieser Weg ist eingeschlagen worden, wo ein Vergleich mit einem realen Systemverhalten nicht durchfuehrbar war (sei es, dass kein entsprechendes reales System existierte, oder auch, dass keine Aufzeichnungen ueber reales Systemverhalten zugaenglich waren); fallweise wurde die Methode auch zusaetzlich zu Vergleichen mit dem realen Systemverhalten eingesetzt. Da mathematische Modelle komplexer Systeme haeufig genug nur dann einer analytischen Loesung zufuehrbar sind, wenn Modellstruktur und Arbeitsbedingungen fuer das Modell erheblichen, teils sogar unrealistischen, Einschraenkungen unterworfen werden, und da gerade diese Tatsache der eigentliche Grund dafuer ist, dass in vielen Faellen

Simulationsmodelle die einzig moeglichen realistischen Modelle gewisser komplexer Systeme darstellen, mag die hier aufgezeigte Methode in gewisser Weise widerspruechlich erscheinen und bedarf einer Klaerung: Die Methode geht davon aus, dass das Simulationsmodell eines bestimmten Systems nicht nur unter realistischen Betriebsannahmen zufriedenstellend arbeiten muss, sondern auch unter den einschraenkenden Annahmen, die ein entsprechendes mathematisches Modell analytisch loesbar machen. Der Simulator wird durch normalerweise einfach vorzunehmende Aenderungen an Struktur und input in direkten Bezug zu dem analytisch loesbaren Modell gebracht; ein Vergleich der beiden Modelle wird damit zu einer sinnvollen Aufgabe.

In der Literatur finden sich Berichte ueber den Einsatz dieser Methode z.B. bei

- Blatny/Clark/Rourke<3>, die beim Simulieren eines time-sharing Systems das Verhalten ihres Simulators unter speziellen Bedingungen mit mehreren analytisch abgeleiteten Verhaltensweisen vergleichen,
- Naylor<11>, der beim Simulieren eines Betriebs das Verhalten seines Simulators in Spezialfaellen mit dem Verhalten eines analytisch behandelbaren Modells vergleicht.

* Schliesslich gibt es noch die anscheinend hoffnungslosen Faelle, wo weder ein reales System noch ein analytisch loesbares Modell existiert, mit deren Verhalten das Simulatorverhalten verglichen werden koennte. Eine gewisse Hilfestellung kann hier gegeben werden von eventuell vorhandenen aehnlichen Systemen, fuer die gueltige Simulatoren vorliegen. Ernst<5> erwaehnt eine derartige "reference to a past clean validation against reality of the model type which the conceptual modeller is using" und Nolan<14> berichtet ueber "validating, in a relative sense, a proposed system against an existing one" im Verlauf der Simulation eines Flugzeugwartungssystems.

Moeglicherweise bietet ein Ansatz, der bei Fishman/Kiviat<6> angedeutet ist, die Moeglichkeit zu einer allgemeineren Loesung des Problems. Ohne genauere Ausfuehrung des Ansatzes scheint dort vorgeschlagen zu sein, ein bootstrapping-aehnliches Vorgehen zu waehlen, bei dem mit Simulationsmodellen einfacher, in ihrer Gueltigkeit bestaetigbarer Systeme begonnen wird (die natuerlich in gewissem Sinne dem zu untersuchenden System aehnlich sein muessen) und in evt. mehreren Zwischenschritten das endgueltige Modell erreicht wird; jeder Schritt erstellt ein jeweils komplizierteres Modell und bestaetigt dessen Gueltigkeit in Bezug auf das Modell des vorangegangenen Schrittes. Ein derartiges Vorgehen wird, jedenfalls in Annaeherung, von Marte, Kuespert und Walke (<9>,<17>) praktiziert, die im Verlauf der Modellierung eines time-sharing Systems wiederholt zwischen der Verwendung von analytischen und Simulationsmodellen wechseln.

Neben der Frage, mit was denn das Simulatorverhalten sinnvollerweise verglichen werden kann, liegt ein weiteres erhebliches Problem in der Frage, wie ein derartiger Vergleich vorzunehmen ist (vereinfachend wird im folgenden das System, mit dessen Verhalten das Simulatorverhalten verglichen wird, das "reale" System genannt; damit soll nicht gesagt sein, dass nicht, wie in den letzten Abschnitten gezeigt, andere Systeme an seine Stelle treten koennen). Die Frage des "wie vergleichen?" umschliesst in Wahrheit zwei verschiedene Problemkreise, naemlich zum einen, welche Daten ein Systemverhalten charakterisieren und daher als Basis der Verhaltensvergleiche dienen sollten, und zum anderen, mithilfe welcher Methoden der Vergleich durchgefuehrt werden kann.

Es ist hier nicht der Ort, in Einzelheiten zu gehen - kompetente Veroeffentlichungen sind leicht zugaenglich (vgl. Naylor/Finger<12>). Moege es genuegen, festzustellen, dass es fuer keines der beiden Probleme "die" Antwort gibt, dass die Antworten vielmehr problemabhaengig sind. Unterschiedliche Masse und Methoden werden zum Einsatz kommen bei ereignisgepraegten mikroskopischen Untersuchungen (wo etwa die Zeitpunkte gewisser Ereignisse von Interesse sein koennen) im Unterschied zu flussgepraegten makroskopischen Untersuchungen (wo etwa die Wertefolgen gewisser zentraler Variabler zu beachten sein koennten). Systeme, bei denen man von stationaerem Verhalten ausgehen kann, werden verschieden behandelt werden von solchen, bei denen diese Annahme nicht gerechtfertigt ist (wie z.B. bei batch-Rechensystemen).

Kurz angerissen werden sollen hier aber zwei allgemeinere Punkte. Der erste betrifft Methoden zum Vergleich charakteristischer Daten. Hier kommt die Statistik ins Spiel. Nun hat jeder statistische Test seine Voraussetzungen und sollte nur nach gewissenhafter Pruefung der Erfuelltheit dieser Voraussetzungen angewendet werden. Das ist recht offensichtlich bei parametrischen Tests (die leidige "Normalitaet") - es ist nicht so offensichtlich bei den sogenannten nichtparametrischen Tests (doch haben auch sie Voraussetzungen wie "symmetrische Verteilung" oder zumindest "zufaellige Stichprobe") oder bei Methoden wie etwa der Spektralanalyse (die von "schwacher Stationaritaet" ausgeht). Dass hier viel gesuendigt wird, ist ein offenes Geheimnis - vielfach ist eine Bestaetigung von Testvoraussetzungen gar nicht durchfuehrbar - und es ist sicher die Muehe wert, zumindest zusaetzlich zu formalen Tests, einen Alternativansatz in die Ueberlegungen mit einzubeziehen, der manchmal als "Turing's Test" apostrophiert wird (vgl. v.Horn<7>, Hutchinson<8>), ein Ansatz, der den Menschen selbst zur Entscheidungsinstanz darueber macht, wie gut reale Welt und Simulationswelt in ihren Verhalten uebereinstimmen.

Der zweite Punkt, der hier zur Sprache kommen soll, bezieht sich auf die Typen von Daten, die als charakteristisch fuer die Verhalten von realer Welt und Simulator angesehen werden und daher als Basis eines Verhaltensvergleiches dienen sollen. Diese Daten sollten nicht isoliert betrachtet werden, sondern immer in

Beziehung zu den erstrebten Faehigkeiten eines Simulators, unter denen wohl seine Voraussagekraft die wertvollste ist. Nun gibt es bei jedem Simulationsprojekt gewisse spezifische Variable (bzw. Funktionen solcher Variabler), deren zukuenftige Entwicklung von besonderem und zentralem Interesse ist. Oft stellen diese Variablen/Funktionen etwa Masse fuer die Leistungsfaehigkeit der Systeme dar. Aus diesem Blickwinkel ist es nur natuerlich, eine Verhaltensaehnlichkeit zwischen realem System und Simulator gerade in Bezug auf diese zentralen Variablen/Funktionen zu verlangen. Sicherlich erhoeht jede Art von Verhaltensaehnlichkeit das Vertrauen in das Simulationsmodell - doch sollte offensichtlich (insbesondere in trade-off-Faellen) das Hauptgewicht des Verhaltensvergleiches auf der Aehnlichkeit der zentralen Variablen/Funktionen liegen. So gesehen, zahlt es sich nicht aus, allzuviel Arbeit in die Erarbeitung der "richtigen" Masse fuer Verhaltensunterschiede zwischen realen Systemen und Simulationsmodellen zu stecken - die Bestimmung zentraler Guetemasse ist in den meisten Faellen eine Frage des Managements des untersuchten Systems, eine politische Frage, und liegt damit ausserhalb des Arbeitsbereiches des Modellkonstrukteurs.

4. Gueltigkeitsbestaetigung - prinzipielle Ansaetze

Ernst<5>: "Validation is the determination of the domain of situations for which the model performs within a given accuracy, with an established calibration."
v.Horn<7>: "Validation is the act of increasing to an acceptable level the confidence that an inference about a simulated process is correct for the actual process."
Diese ausgezeichneten, jedoch offensichtlich unterschiedlichen, Definitionen beleuchten die beiden verschiedenen Standpunkte, von denen aus das Gueltigkeitsproblem betrachtet werden kann.

Ernst bezieht sich auf den formalen Aspekt der Gueltigkeitsbestaetigung, indem er die Vorstellung eines Situationsraumes einfuehrt, zerlegt in zwei disjunkte Teilraeume, von denen einer all jene Situationen umfasst, in denen der Simulator den gegebenen Genauigkeitsforderungen genuegt (der Simulator liefert "gueltige" Ergebnisse), waehrend der andere all jene Situationen einschliesst, fuer die die Simulatorergebnisse als ungueltig gelten. Die Calibrierungssituationen haben wir uns irgendwo innerhalb des Gueltigkeitsteilraumes vorzustellen. Gueltigkeitsbestaetigung nun wuerde daraus bestehen, zusaetzliche Situationen darauf zu ueberpruefen, ob sie zu den Gueltigkeitssituationen zu zaehlen sind oder nicht. Um die Definition aber wirklich brauchbar zu machen, waere eine Extra- (bzw. Inter-) polationsmethode zu entwickeln, die es erlaubte, von der Gueltigkeit des Simulators in ueberprueften Situationen auf seine Gueltigkeit in bisher ungeprueften Situationen zu schliessen. Es ist vorstellbar, dass derartige Methoden bezueglich numerischer Systemparameter entwickelt werden koennten. Es faellt jedoch schwer, sich vorzustellen, wie eine allgemeine derartige Methode aussehen sollte, anwendbar etwa auch fuer einen Fall, wo unterschiedliche

Entscheidungsstrategien mithilfe eines Simulators zu beurteilen sind und Aussagen ueber die Gueltigkeit dieser Beurteilung gefordert werden. Eine wesentliche Implikation der Ernst'schen Definition sollte aber auf jeden Fall im Gedaechtnis bleiben: Es ist die Feststellung, dass Gueltigkeit eines Simulators eine lediglich relative Angelegenheit ist. Ein Simulator ist nicht entweder gueltig oder ungueltig, sondern er liefert gueltige Resultate fuer eine gewisse Menge von Situationen, ungueltige fuer die restlichen Situationen - und dies in Bezug auf vorgegebene Genauigkeitsforderungen.

v.Horns Definition besteht nicht auf einem formalen Beweis der Simulatorgueltigkeit an sich - sie laesst auch ein (subjektives) Vertrauen gelten, dass die Simulatoraussagen in bestimmten Situationen gueltig sein werden. Gewiss muss auch dieses Vertrauen sich auf die formale Ueberpruefung der Gueltigkeit von Simulatorergebnissen in Situationen ausserhalb des Calibrierungsbereichs stuetzen. Doch kann das Konzept eines stetig sich entwickelnden subjektiven Vertrauens in die Ergebnisse eines laengerfristig im Einsatz befindlichen Simulators dazu beitragen, sich aus der misslichen Lage zu befreien, die durch die beiden folgenden Einstellungen gekennzeichnet ist:
"Die Gueltigkeit eines Simulators kann erst dann als erwiesen gelten, wenn Empfehlungen seitens des Simulationsmodells (abgeleitet aus Experimenten mit dem Simulator) in der realen Welt verwirklicht worden sind und eine Ueberpquefung der vom Simulator vorausgesagten wahrscheinlichen Auswirkungen der Systemveraenderungen mit den tatsaechlich eingetretenen Auswirkungen in Einklang stehen".
"Es ist zu gefaehrlich (kostspielig, usw.), Simulatorempfehlungen in das reale System zu uebernehmen, bevor die Gueltigkeit des Simulators gezeigt ist".

In jedem Falle laesst sich ein Vertrauen in die Ergebnisse eines Simulators nur aufbauen, wenn die Gueltigkeit der Ergebnisse fuer Situationen ausserhalb des Calibrierungsbereiches gezeigt werden konnte, d.h. nur mithilfe gesonderter Simulatorlaeufe eigens zum Zwecke der Gueltigkeitsbestaetigung. Man kann sicher derartige Gueltigkeitstestsituationen entwerfen, anschliessend den Simulator dazu benutzen, das Verhalten des realen Systems fuer diese Situationen vorauszusagen, schliesslich das reale System entsprechend aendern und das Verhalten des realen Systems mit den Voraussagen des Simulators vergleichen, somit entscheiden, ob die Voraussagen gueltig waren. Ein solcher Ansatz zur Gueltigkeitsbestaetigung wird aber haeufig auf Widerstand stossen (wie zuvor: Gefaehrlichkeit, Kosten, usw.) - eine Gueltigkeitsbestaetigung ohne direkte Einbeziehung des realen Systems waere sicherlich willkommener.

Historische Gueltigkeitsbestaetigung ist ein solcher Weg, Gueltigkeitstests ohne Einbeziehung des realen Systems durchzufuehren. Er setzt voraus, dass Aufzeichnungen ueber das Verhalten des Systems in der Vergangenheit verfuegbar sind, mit denen das Verhalten des (evt. entsprechend veraenderten) Simulators verglichen werden kann. Die Methode der historischen

Gueltigkeitsbestaetigung kann eingesetzt werden, die Gueltigkeit sowohl bezueglich des treibenden inputs als auch bezueglich der Systemstruktur zu pruefen:

* Input-Gueltigkeit kann geprueft werden, indem man den Simulator mit inputs betreibt, die waehrend der Calibrierungsphase nicht benutzt worden sind und anschliessend das Simulatorverhalten mit dem entsprechenden Verhalten der realen Welt vergleicht, um so zu ermitteln, ob diese Verhalten aehnlich genug sind, d.h., ob die Ergebnisse des Simulators fuer neue input-Situationen gueltig sind.

* Struktur-Gueltigkeit laesst sich pruefen, wenn Aufzeichnungen ueber das Verhalten des realen Systems vorliegen fuer Zeitspannen, in denen das reale System sich strukturell unterschied von der Version, die in der Calibrierungsphase betrachtet wurde. Solche Zeitspannen koennen gekennzeichnet sein etwa durch das Vorhandensein anderer Systemkomponenten (Menge und/oder Typ), aber auch durch das Bestehen anderer Arbeitsstrategien, usw. Die Struktur des Simulators ist in diesen Faellen geeignet (d.h. im Hinblick auf die strukturellen Unterschiede) zu aendern und das Simulatorverhalten mit Aufzeichnungen ueber das Verhalten der realen Welt in diesen andersartigen Strukturs ituationen zu vergleichen.

Historische Simulation ist ein aeusserst wertvolles Hilfsmittel, das zu Zwecken der Gueltigkeitsbestaetigung eingesetzt werden sollte wo immer moeglich. Um dieses guenstige Hilfsmittel nicht zu verspielen, sollte man darauf achten, auf keinen Fall alle Aufzeichnungen ueber historisches Systemverhalten schon in der Calibrierungsphase aufzubrauchen (auch diese Phase vermag ja historisches Systemverhalten guenstig einzusetzen), sondern eine genuegende Menge dieser Aufzeichnungen fuer Zwecke der Gueltigkeitsbestaetigung aufzusparen.

5. Schlussbemerkungen

In diesem Beitrag wurde versucht, einige der schwierigen Fragen zu beleuchten, die im Verlaufe des Aufbaus von Vertrauen in die Ergebnisse von Simulationsmodellen auftreten. Gueltigkeitsbestaetigung als eine Phase im Verlauf der Entwicklung eines Simulators wurde abgesetzt gegen eine der Gueltigkeitsbestaetigung vorausgehende Calibrierungsphase und eine der Gueltigkeitsbestaetigung folgende Experimentierphase. Verschiedene Grundeinstellungen zu den Problemen der Gueltigkeitsbestaetigung wurden diskutiert und Hinweise gegeben, wie diese Probleme in konkreten Simulationsprojekten angegangen wurden. Der Stand der Methodik auf diesem Gebiet ist bei weitem nicht befriedigend. Beachtet man aber, in welchem Masse die Notwendigkeit der Gueltigkeitsbestaetigung ins allgemeine Bewusstsein gerueckt ist und beruecksichtigt man die vielfaeltigen Bemuehungen um dieses Problem, dann kann man mit einiger Hoffnung auf baldige Fortschritte in die Zukunft blicken.

Literaturverzeichnis:

<1>Beilner,H. und Waldbaum,G.: Statistical Methodology for Calibrating a Trace-Driven Simulator of a Batch Computer System.
in Freiberger,W. (ed.):
Statistical Computer Performance Evaluation
Academic Press, 1972

<2>Blackmore,W.R.: Pitfalls, Pratfalls and Pleas in Monte Carlo Simulation
Proc. of the 1972 Summer Computer Simulation Conference, San Diego

<3>Blatny,J.,Clark,S.R. und Rourke,T.A.: On the Optimization of Performance of Time-Sharing Systems by Simulation
CACM 15(1972,nr.6

<4>Boughton,J.M. und Naylor,T.H.: A Model of the United States Monetary Sector.
Kapitel 15 in <10>

<5>Ernst,H.A.: Problems in Computing Systems Performance Modelling: Characterization, Calibration and Validation.
IBM Research Report RC 3319, 1971

<6>Fishman,G.S. und Kiviat,P.J.: Digital Computer Simulation: Statistical Considerations.
RAND Memorandum RM-5387-PR, 1967

<7>v.Horn,R.L.: Validation of Simulation Results
Management Science, 17(1971), nr.5

<8>Hutchinson,G.K.: The Use of Simulation in the Design of a Multi Computer Operating System
Technical Report, University of Wisconsin, Milwaukee

<9>Kuespert,H.J. und Marte,G.: Optimale Rechenzeitzuteilung bei einem Teilnehmerrechensystem mit jeweils einer Aufgabe im Arbeitsspeicher
Elektronische Rechananlagen 12(1970), nr.3

<10>Naylor,T.H.(ed.): Computer Simulation Experiments with Models of Economic Systems.
J. Wiley & Sons, 1971

<11>Naylor,T.H.: Analysis of Variance
Kapitel 7 in <10>

<12>Naylor,T.H. und Finger,J.M.: Validation
Kapitel 5 in <10>

<13>Noe,J.D. und Nutt,G.J.: Validation of a Trace-Driven CDC 6400 Simulation.
Proc. of the SJCC, 1972

<14>Nolan,R.L.: Verification/Validation of Computer Simulation Models
Proc. of the 1972 Summer Computer Simulation Conference, San Diego

<15>Schrank,W. und Holt,Ch.: Critique of Verification of Computer Simulation Models
Management Science 14(1967), nr.2

<16>Waldbaum,G. und Beilner,H.: SOUL - A Simulation of OS Under LASP
Proc. of the 1972 Summer Computer Simulation Conference, San Diego
<17>Walke,B.: Die mittlere Verweilzeit in Teilnehmerrechensystemen bei optimaler Rechenzeit-, Arbeitsspeicher- und Transportkanalzuteilung
Nachrichtentechnische Fachberichte vol.44(1972), VDE Verlag Berlin

Der Stichprobenumfang bei Monte-Carlo-Simulation

M. Helm

Einleitung

Während auf dem Gebiet der Monte-Carlo-Simulation eine große Anzahl vorzüglicher Sprachen existiert, stößt man in der Literatur relativ selten auf völlig einwandfrei durchgeführte Simulationsstudien. Der Grund hierfür ist das Fehlen von Programmsystemen, welche dem Benutzer eine mathematische einwandfreie Analyse der eingegebenen Daten sowie der durch die Simulation erhaltenen Resultate ermöglichen und so eine Ergänzung zur eigentlichen Sprache darstellen würden.

So steht der Anwender von Simulationssprachen diesen oft sehr diffizilen Problemen der mathematischen Statistik manchmal völlig hilflos gegenüber und es kann die Situation eintreten, daß die mit großem Aufwand durch Simulation erarbeiteten Aussagen irreführend sind, weil sie nicht sorgfältig genug in der Realität verankert wurden.

Es soll daher ein Programmsystem vorgestellt werden, das zunächst auf der aus dem Gebiet der Zeitreihenanalyse wohlbekannten und von G.S. FISHMAN [1], [2], [3], [4] in die Simulation eingeführten Methode des autoregressiven Modells aufbaut. Es soll dem Benutzer ermöglicht werden vom Bildschirm eines Terminals aus die Entwicklung der Genauigkeit der relevanten Modellgrößen zu verfolgen und so aktiv im Dialog mit dem Modell die Simulation solange laufen zu lassen, bis die erforderliche Genauigkeit erreicht ist bzw. eine Fortführung des Laufs aus wirtschaftlichen Gründen nicht mehr sinnvoll erscheint.

Dazu wird im Abschnitt 1) die Abschätzung der Toleranzintervalle von Mittelwerten besprochen. In Abschnitt 2) wird auf die Behandlung des Einschwingvorganges eingegangen, in 3) die Wirkungsweise des Dialogprogramms erläutert und schließlich in 4) eine kritische Wertung der Methode durchgeführt und auf weitere zu untersuchende Punkte hingewiesen.

1. Die Bestimmung des Stichprobenumfangs

Die in diesem Abschnitt besprochene Fragestellung ist: Wielange muß eine Simulation laufen, bis die vom Benutzer geforderte Genauigkeit des Mittelwertes einer Modellvariablen erreicht ist?

Dazu sollen die wichtigsten Punkte der Theorie zusammengestellt werden. Eine ausführliche Darstellung findet man bei G.S. Fishman [1].

Durch die Simulation möge eine Realisierung $\hat{X}$ aus dem Ensemble eines stochastischen Prozesses $\{X\}$ erzeugt worden sein (z. B. sei $\{X\}$ die Länge der Warteschlange vor einem Schalter, die in regelmäßigen Intervallen gemessen wird usw.).

Nun soll $\{X\} = \{ X_t \; ; \; t = 0, \pm 1, \pm 2, \cdots \pm \infty \}$ kovariant stationär sein, in dem Sinne daß:

$$\mu = E[X_t] < \infty \qquad 1.1)$$

und

$$R_{t,s} = R_{t-s} = E[(X_s - \mu)(X_t - \mu)] < \infty \qquad 1.2)$$

für alle t und s.

Dabei ist μ der Erwartungswert und R_{t-s} die Autokorre-

lationsfunktion des Prozesses.

In vielen Fällen interessiert der Erwartungswert, für den man innerhalb vorgegebener Toleranzgrenzen einen Schätzwert angeben will.

Dieser Schätzwert für μ ist:

$$\bar{X}_N = \frac{1}{N} \sum_{t=1}^{N} X_t \qquad 1.3)$$

Man kann zeigen (Satz von S.N. Bernstein), daß die Eigenschaften 1.1) und 1.2) sicherstellen, daß $\bar{X}_N$ für $N \to \infty$ normal verteilt mit dem Erwartungswert μ und der Varianz V_N ist.

Daher gilt die Beziehung: 1.4)

$$P(\bar{X}_N - Q_\beta \cdot \sqrt{V_N} \leq \mu \leq \bar{X}_N + Q_\beta \cdot \sqrt{V_N}) = 1-\beta$$

wobei $1-\beta$ der Konfidenzlevel und Q_β die zu diesem Level gehörige Quantile der (0,1)-Normalverteilung ist.

Damit ist die Fragestellung ein Toleranzintervall für $\bar{X}_N$ zu bestimmen, auf die Ermittlung eines Schätzwertes für V_N reduziert.

Nun gilt für $N \to \infty$

$$V_N = VAR[\bar{X}_N] = \frac{m}{N}$$

wobei

$$m = \sum_{s=-\infty}^{+\infty} R_s \qquad 1.5)$$

Die Autokorrelationsfunktionen

$$R_s \qquad s = 0, \pm 1, \pm 2, \cdots \pm \infty$$

sind für eine begrenzte Realisierung $\hat{X}$ recht schlecht zu schätzen. Es wird daher die Hypothese aufgestellt,

daß die allgemeine Darstellung des Prozesses $\{X\}$ als unendlicher autoregressiver Prozeß durch eine endliche approximiert werden kann d. h., daß für das folgende Modell gilt:

$$\sum_{s=0}^{p} b_s (X_{t-s} - \mu) = Y_t \qquad 1.6)$$

Dabei soll Y_t weißes Rauschen sein, d. h.

$$E[Y_t] = 0$$

$$E[Y_t \cdot Y_s] = \begin{cases} \sigma^2 & t = s \\ 0 & sonst \end{cases} \qquad 1.7)$$

Man kann nun zeigen, daß für die Größen aus 1.5), 1.6) und 1.7) der folgende Zusammenhang gilt:

$$m = \frac{\sigma^2}{\left(\sum_{s=0}^{p} b_s\right)^2} \qquad 1.8)$$

Man kann mit Hilfe der Methode der kleinsten Quadrate Schätzwerte für die Koeffizienten b_s angeben, deren Fehler mit $1/\sqrt{N}$ abnehmen. Mit 1.8) haben wir also einen brauchbaren Schätzwert für die Varianz des Mittelwertes V_N gefunden.

Nun wird 1.5) in 1.4) eingesetzt und das Toleranzintervall in Einheiten des Erwartungswertes gemessen. Man setzt also $Q\sqrt{m/N} = c\mu$ wobei c die prozentuale Genauigkeit des Mittelwertes darstellt. Damit erhält man für den erforderlichen Stichprobenumfang den Schätzwert

$$\hat{N} = m\left(\frac{Q}{c\mu}\right)^2 \qquad 1.9)$$

2. Der Einschwingvorgang

Bei keinem Simulationsexperiment ist 1.1) erfüllt, da man stets von einem willkürlich gewählten Anfangszustand X_0 ausgeht und der stationäre Zustand (wenn überhaupt) erst in der Grenze $t \to \infty$ erreicht wird.

Dieser Sachverhalt läßt sich durch die folgende Beziehung ausdrücken:

$$\mu = \lim_{t \to \infty} \mu_t = \lim_{t \to \infty} E[X_t | X_0] \qquad 2.1)$$

wobei $E[X_t | X_0]$ der bedingte Erwartungswert ist.

Um den Einfluß des Anfangzustandes X_0 auszuschalten, versucht man nun 2.1) näherungsweise zu erfüllen, indem man die ersten K Meßwerte verwirft und als Schätzwert des Erwartungswertes verwendet:

$$\bar{X}_{N,K} = \frac{1}{N-K} \sum_{j=K+1}^{N} X_j \qquad 2.2)$$

Das Problem liegt nun darin, ein K zu bestimmen, so daß einerseits die Verzerrung des Mittelwertes durch den willkürlichen Anfangszustand klein bleibt, andererseits der Verlust an Meßwerten, der zu einem Anwachsen der Varianz von $\bar{X}_{N,K}$ führt, nicht zu groß wird (siehe auch [2]).

In [1] wird für K der folgende heuristische Vorschlag gemacht:

Wir betrachten zwei Prozesse:
Einen unabhängigen Y_t (siehe 1.7)) mit der Varianz σ^2

und einen autokorrelierten X_t . Man bildet nun den Mittelwert $\bar{X}_N$, der die Varianz m/N hat. Nun kann man sich die Frage stellen, wieviele Werte N des abhängigen Prozesses man kumulieren muß, damit der entstandene Mittelwert dieselbe Varianz wie der unabhängige Prozeß hat; N ist also bestimmt durch $m/N = \sigma^2$.

Man kann dies auch so interpretieren, daß man angibt, auf wieviele abhängige Punkte des Prozesses ein unabhängiger kommt und vermuten, daß der Einfluß des Anfangswertes nach diesen Punkten klein ist. Man setzt also für K in 2.2)

$$K \approx \left[\frac{m}{\sigma^2} \right] \qquad 2.3)$$

3. Das Programm COSIM

Auf der Basis der in [1] und [2] beschriebenen Theorie wurde das Programmsystem COSIM entwickelt, das es gestattet, vom Datensichtgerät aus den Verlauf einer Simulation hinsichtlich ihrer Genauigkeit aktiv zu steuern. Im Wesentlichen läuft dieser Dialog folgendermaßen ab:

Der Benutzer gibt zunächst an, wieviele Meßwerte einer bestimmten Leibgröße bis zur ersten Unterbrechung des Laufs gesammelt werden sollen. Ist diese Zahl erreicht, wird die Simulation unterbrochen und der Benutzer beginnt die Toleranzintervalle der ihn interessierenden Größen zu analysieren. Es wird ermittelt, wieviele Werte für

jede einzelne dieser Größen zum Erreichen der Genauigkeit noch nötig sind und er entscheidet, welche Größe als neuer Leitwert dienen soll. Er gibt an, welcher Bruchteil der vorgeschlagenen Anzahl noch erzeugt werden soll und läßt die Simulation bis zur nächsten Unterbrechung weiterlaufen.

Die Simulation läuft in dieser iterativen Weise von Unterbrechung zu Unterbrechung, wobei die Entwicklung der Toleranzintervalle auch graphisch am Bildschirm dargestellt wird. Ist die vorgegebene Genauigkeit erreicht, bzw. der Genauigkeitsgewinn pro erzeugten Meßwert so gering, daß die Fortsetzung der Simulation nicht mehr wirtschaftlich erscheint, so bricht der Benutzer den Lauf ab. Dabei werden sämtliche gesammelte Rohdaten für weitere Untersuchungen abgespeichert.

(Technisches: COSIM läuft als Dialogsystem im BSV-Time sharing - Betriebssystem auf der 4004-151 bzw. im PBS auf allen 4004-Maschinen. Es ist in der GPSS-artigen Sprache SIAS implementiert)

4. Schlußbemerkung

Die von FISHMAN vorgeschlagene Methode ist zweifellos in vielen Fällen brauchbar. Man kann jedoch verschiedene Punkte einer Kritik unterziehen, die zu weiteren Untersuchungen anregen sollte:

Nämlich:

1) Es ist keinesfalls zu erwarten, daß das unter 1.6) vorgeschlagene autoregressive Modell auf alle durch Simulationen erzeugten Zeitreihen anwendbar ist.

2) Kein Prozeß, der mit willkürlichen Anfangswerten startet, ist im Endlichen stationär. Bei vielen Prozessen wird die unter 2.) beschriebene Bestimmung von K zu untragbar hohen Verlusten an Rechenzeit führen.

3) Wenn der Prozeß tatsächlich als autoregressiv erkannt wird, erlaubt diese Information wesentlich weiterreichende Fragestellungen zu beantworten, als lediglich die Absicherung des Mittelwertes.

Zusammenfassend kann man also sagen, daß mit den Arbeiten von G.S. FISHMAN ein erster, wichtiger Schritt auf diesem Gebiet getan wurde, daß zu diesem Problemkreis jedoch noch zahlreiche Untersuchungen notwendig sind.

Literatur:

1) G.S. FISHMAN
Estimating Sample Size in Computing Simulation Experiments
Management Science 18 p. 21 (1971)
2) G.S. FISHMAN
BIAS-Considerations in Simulation Experiments
Operations-Research p. 785 (1972)
3) G.S. FISHMAN
Digital Computer Simulation:
Input-Output Analysis
RAND-Memorandum RM-5540-PR (1968)
4) G.S. FISHMAN P.J. KIVIAT
The Analysis of Simulation-Generated Time Series
Management-Science 13 o. 525 (1967)

Fachgespräch über Methodik der rechnergestützten Simulation, IDT - GfK Karlsruhe, 10./11. Mai 1973

ÜBER EIN ADAPTIVES MODELL ZUR LAGESIMULATION EINES SATELLITEN

Dr. R. Kammüller, Institut für Informatik III,
Universität Karlsruhe

1. EINLEITUNG

An einem Fallbeispiel aus der Raumfahrt soll hier gezeigt werden, wie Simulationsaufgaben von hohen Qualitätsansprüchen gelöst werden können.

Die Technologie des zu simulierenden Systems wird lediglich einleitend und,soweit zum Verständnis erforderlich, berührt. Der Beitrag befaßt sich zunächst damit, die Probleme und Schwierigkeiten aufzuzeigen, die sich aus der Aufgabenstellung ergeben haben, um dann exemplarisch die gewählte Lösung für dieses Simulationsprojekt vorzustellen.

Das Projekt umfaßte die Entwicklung und den Einsatz eines digitalen Simulators zur Generierung der Winkellage eines künstlichen Erdsatelliten. Dieser Satellit gehörte zu einer Serie kleiner Forschungssatelliten, die von der Europäischen Raumfahrt-Organisation ESRO in den Jahren 1967- 1971 gebaut wurden und mit Unterstützung der NASA von Kalifornien aus gestartet wurden.

Die gewählten Flugbahnen waren polare Bahnen von mittlerer Höhe (Apogäum und Perigäum in der Größenordnung von 400 bzw. 2000 km) mit Umlaufzeiten von ca. 100 Minuten; die Masse dieser Satelliten lag zwischen 70 und 100 kg.

Zur Orientierung des Flugkörpers im Raum, d.h. zur Steuerung seiner "Winkellage" wurde ein einfaches aber sehr zuverlässiges magnetisches Prinzip verwendet. Wie in Fig. 1 gezeigt, wurde ein festmontierter Dauermagnet benutzt, um die Referenzachse des Satelliten jeweils parallel zur Richtung des lokalen Erdfeldvektors auszurichten. Entsprechend dem Dipol-Charakter des Erdmagnetfeldes wechselt die Richtung des Feldvektors entlang der Umlaufbahn.

Durch das Richtmoment des Steuermagneten $\vec{M}$ wird der Satellit zur Ausführung einer periodischen "Purzelbaum-Bewegung" gezwungen. Neben dem hierzu erforderlichen Steuerungsmoment wirken eine Reihe von sehr komplexen Störmomenten auf den Flugkörper ein. Diese Störungen sind bedingt durch die Restatmosphäre, das Magnetfeld und die Inhomogenität des Gravitationsfeldes der Erde. Zum Abbau des Rest-Dralls und zur Dämpfung der Ausgleichsschwingungen wurde ein Hysterese-Dämpfer aus ferromagnetischen Stäben eingebaut.

Das Lagesteuerungssystem mußte so dimensioniert sein, daß der Ausrichtungsfehler β (Fig. 2) zwischen Referenzachse z und lokalem Feldvektor H innerhalb vorgegebener Grenzen (Größenordnung Winkelgrade) blieb. Um die Winkellage des Satelliten zu beschreiben, benutzt man die in der Kreiseltheorie üblichen Euler Winkel (Ψ, Θ, Φ). Sie definieren entsprechend Fig. 2 die Lage der Satellitenachsen (X, Y, Z) relativ zu den als raumfest angenommenen Achsen (x, y, z).

Soviel zur Technologie des Systems; eine ausführliche Beschreibung des Satelliten und seines dynamischen Verhaltens findet sich in [1],[2],[3].

2. AUFGABENSTELLUNG

Der oben beschriebene Satellitentyp war mit einem Meßsystem ausgerüstet, welches den Sonnenstand und die Richtung des Magnetfeldes relativ zum Satelliten-Achsensystem kontinuierlich messen konnte. Diese Meßwerte genügten, um die Winkellage des Flugkörpers in eindeutiger Weise zu berechnen.

Befand sich der Satellit im Empfangsbereich einer Bodenstation, so konnten die Meßdaten über das Telemetrie-System in "real time mode" abgerufen werden. Während der übrigen

Flugzeit (90 - 95% eines Umlaufes) wurden die Meßwerte auf Magnetband zwischengespeichert und bei Überfliegen der nächsten Bodenstation im "play-back mode" übergeben. Dieser Bandspeicher stellte das empfindlichste Element in der Telemetrie-Kette dar und hatte die geringste Lebensdauer von allen Bauelementen des Satelliten.

Die Aufgabe bestand nun darin, dafür zu sorgen, daß bei Ausfall des on-bord Bandrekorders die nicht erfaßtbaren Lagedaten durch Simulàtion generiert werden konnten. Die Hauptschwierigkeiten der Aufgabe lagen darin, daß der Satellit ein äußerst empfindliches dynamisches System hinsichtlich Störungen der Rotationsbewegung darstellte (Gyroscopischer Effekt). Andererseits waren gerade die Parameter der wichtigsten Störmomente, wie Dichte und Temperaturverteilung in der Restatmosphäre nur unzureichend bekannt. Schließlich sollte sich der Simulationsvorgang über relativ lange Zeitintervalle (2 - 3 Stunden Flugzeit) erstrecken bei Genauigkeitsanforderungen in der Größenordnung von wenigen Winkelgraden.

3. LÖSUNGSWEG

Da sich die oben genannten Genauigkeiten mit Analogrechnern nicht erreichen ließen, kam nur eine digitale Simulation in Frage. Da keine Realzeit-Bedingungen gestellt waren, konnte ein Integrations-Algorithmus von hoher Genauigkeit bei maximaler Wortlänge gewählt werden.

Es konnte weiterhin davon ausgegangen werden, daß vor Eintreten einer Störung am Bandspeicher noch genügend "gesunde"Meßdaten an die Bodenstation übergeben worden waren. Es lag nahe, die mit Unsicherheiten behafteten Parameter zunächst anhand dieser Meßdaten genauer zu bestimmen. Andererseits war die Zahl dieser Parameter so groß (bis zu

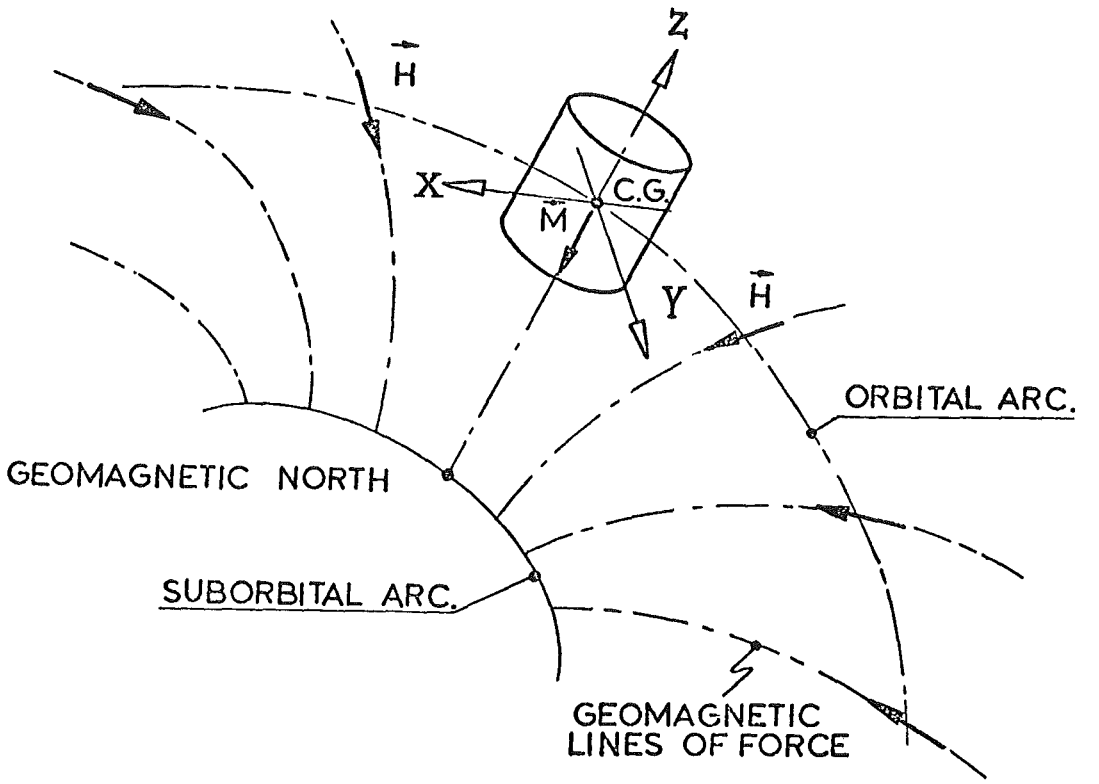


Fig 1: Orientierung eines magnetisch gesteuerten Satelliten in polarer Umlaufbahn.

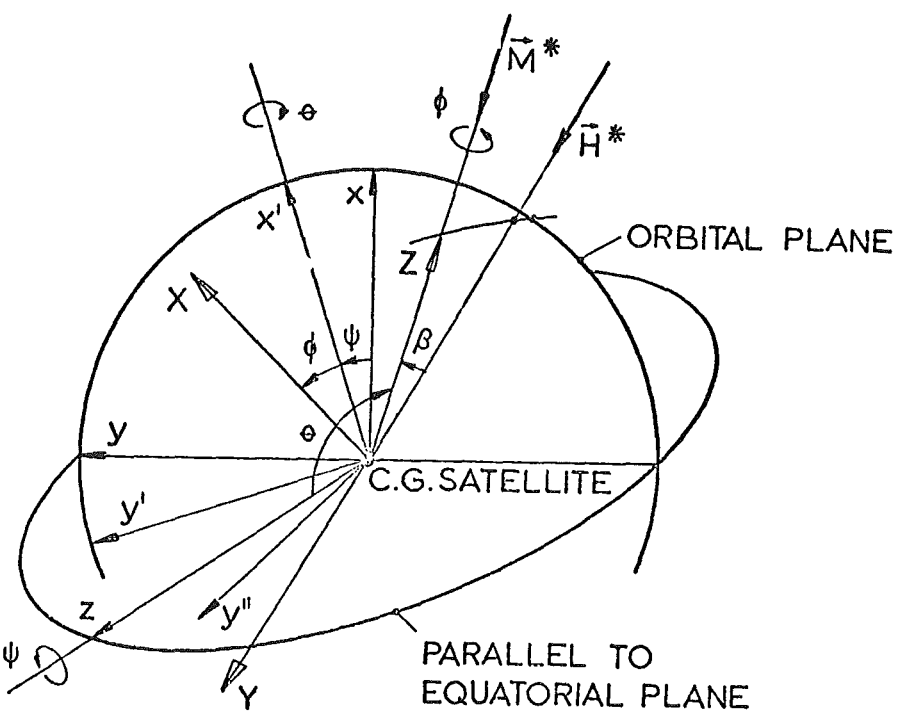


Fig 2: Definition der Euler-Winkel (Ψ, Θ, Φ) und des Ausrichtungsfehlers β.

25 Parameter), daß nur ein systematisches oder besser noch automatisches Verfahren zur richtigen Anpassung der Parameter führen würde.

Aus diesen Gründen wurde versucht, ein Modell zu entwerfen, das sich selbsttätig dem zeitlich nächstliegenden Meßdatensatz anpaßt (adaptives Modell). Erst bei Erreichen eines gewissen optimalen Modellzustandes wird der eigentliche Simulationsvorgang gestartet.

4. DAS BASISMODELL ESIDYN

Der Kern des adaptiven Modells besteht aus dem Basismodell, welches die mathematische Beschreibung der Satelliten-Dynamik und die auf ihn einwirkenden Störungen umfaßt. Das Blockdiagramm des Basismodells ist in Fig. 3 gezeigt, die dazugehörigen Modellgleichungen sind in vektorieller Form in Fig. 4 zusammengefaßt. Das Programm ist in FORTRAN geschrieben und benötigt 65 K Speicherplatz. Der Aufbau des Programms ist hierarchisch geordnet.

Das Unterprogramm DIFFEQ fungiert als unterstes Steuerprogramm, welches selbständig die Unterprogramme für die Störmomente, Umlaufbahn und geomagnetisches Feld steuert und kontrolliert. Das ergibt sich zwangsläufig aus dem Aufbau des Gleichungssystems in Fig. 4. DIFFEQ enthält die Bewegungsgleichungen (1) und die kinematischen Gleichungen (2), die zusammen ein Differentialgleichungssystem 6. Ordnung ergeben.

Weiterhin führt DIFFEQ alle Transformationen von Vektoren zwischen dem satellitenfesten Koordinaten-System (XYZ) und dem raumfesten Koordinatensystem (xyz) durch, wie sie durch Gleichung (3) und deren inverse Form gegeben sind.

Die Unterprogramme DEMAG, AERDYN und GRAV liefern die Momentenvektoren, die durch den Hysteresis-Dämpfer, Luftwiderstand und Gravitationsfeld verursacht werden. Die Routine BEHA dient zur Simulation des Hysterese-Effektes in dem ferromagnetischen Material der Dämpferstäbe.

DIFFEQ selbst wird durch die Integrationsroutine RKINT gesteuert. RKINT führt die numerische Integration der Differentialgleichungen nach dem Runge-Kutta-Merson-Verfahren durch. Die Schrittweite wird hierbei automatisch angepaßt, um eine vorgegebene Genauigkeit zu erreichen. Dieses Verfahren wurde ausgewählt, weil es als Einschrittverfahren bekanntlich selbststartend ist. Diese Eigenschaft ist für den Aufbau eines adaptiven Modells von großer Bedeutung.

Das Basismodell wird eingebaut in ein Programmsystem, welches den Anpassungsprozeß der Modellparameter steuert. Während dieses Prozesses muß das Basismodell mehrere hundert mal initialisiert werden, wobei eine einfache Startprozedur der Integrationsroutine erheblichen Rechenzeitgewinn bedeutet.

5. DAS ANPASSUNGSVERFAHREN

Wie eingangs erwähnt, sollen bis zu 25 Parameter des Basismodells durch Anpassung an ein Kollektiv von Meßdaten bestimmt werden. Diese Meßdaten bestanden aus dem in Abständen von 12,8 sec gemessenen Sonnenwinkel und der Magnetfeldrichtung, die nach Übermittlung an die Bodenstation sofort in die Lagewinkel (Ψ,θ,Φ) umgerechnet wurden.
Da die unbekannten Parameter einer gewissen zeitlichen Drift unterlagen, hat es sich als zweckmäßig erwiesen, als Kollektiv die Lagemeßdaten des letzten noch ungestört vermessenen Umlaufs zu verwenden. Dies ergab somit ein Kollektiv von rund 500 Meßpunkten.
Die Anpassung des Modells stellt ein Problem der nicht-

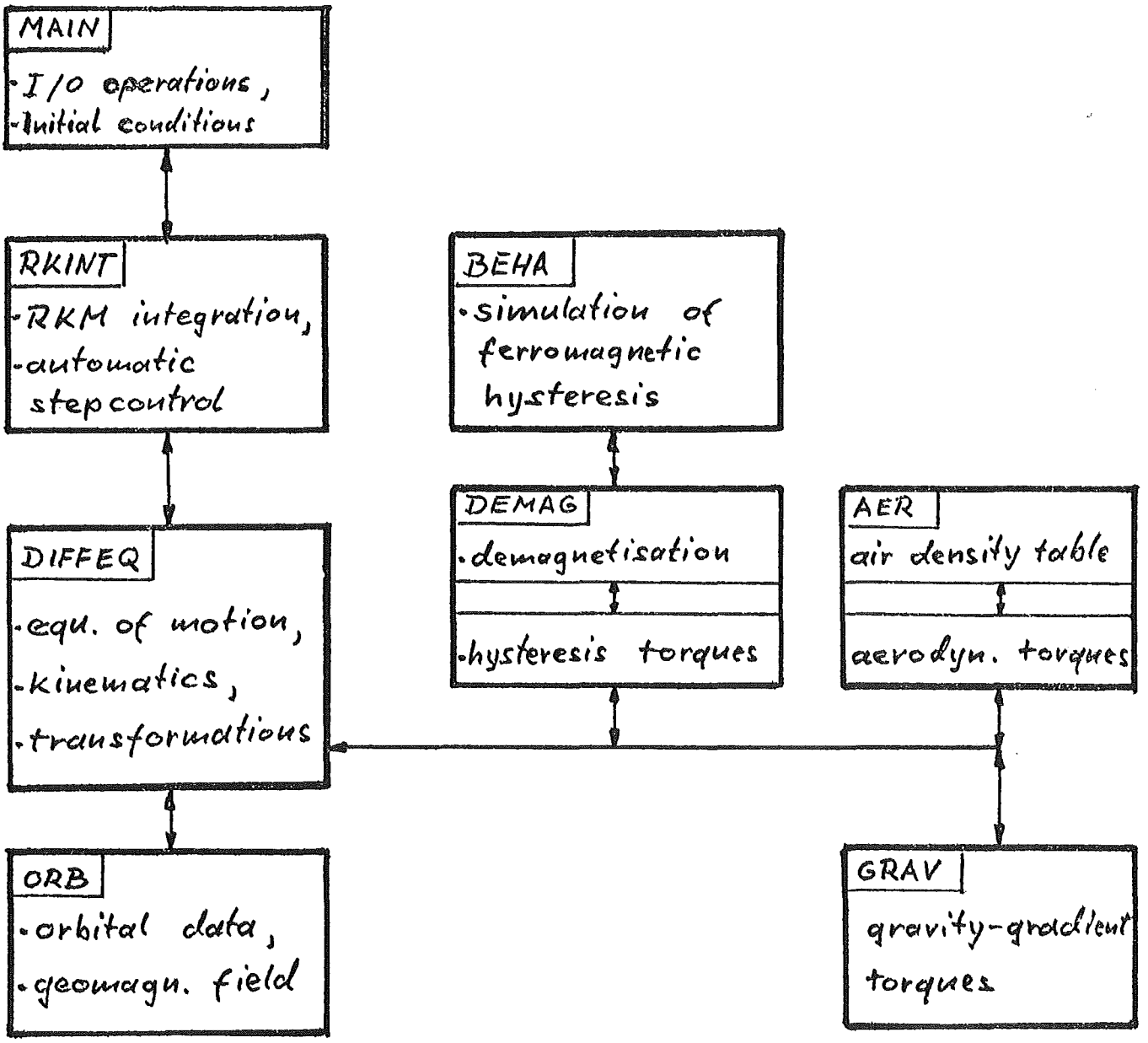


Fig 3 : Blockdiagramm des Basismodells ESIDYN

1. Bewegungsgleichungen

$$\vec{\vec{I}} \cdot \dot{\vec{\omega}} + \vec{\omega} \times \vec{\vec{I}} \cdot \vec{\omega} = \vec{T}_s + \sum^{p} \vec{T}_p \qquad (1):$$

$\vec{\vec{I}}$ Trägheits Tensor

$\vec{\omega}$ Winkelgeschwindigkeit in Bezug auf das Satellitensystem

$\vec{T}_s$ Steuermoment

$\vec{T}_p$ Störmomente

2. Kinematische Beziehungen

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = A \begin{bmatrix} \dot{\psi} \\ \dot{\Theta} \\ \dot{\Phi} \end{bmatrix}$$

A = orthogon. Transformations matrix der Geschwindigkeiten (2)

$A = [a_{ij}(\Theta\Phi)] \quad i,j = 1,2,3$

3. Vektor Transformation

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = B \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

B - orthogon. Transformationsmatrix der Vektoren

$B = [b_{ij}(\Psi,\Theta,\Phi)] \quad i,j = 1,2,3$

4. Momente

$\vec{T}_s = \vec{M} \times \vec{H}$ Steuermoment, ($\vec{H}$ lokaler Feldvektor)

$\vec{T}_1 = \vec{M}_D(H) \times \vec{H}$ Dämpfungsmoment

$\vec{T}_2 = \varrho F_{ae} \cdot \vec{c}$ Luftwiderstandsmoment

$\vec{T}_3 = F_G (\vec{e}_v \times \vec{\vec{I}} \cdot e_v)$ "gravity gradient" Moment

$\vec{c} = [c_x, c_y, c_z]^T$ aerodynamische Koeffizienten

F_{ae} - Funktion der Satellitenform u. der Geschwindigkeit

F_G - Funktion des Gravitationsfeldes

$\vec{e}_v$ - Einheitsvektor in Richtung Erdmittelpunkt

Fig 4: Modellgleichungen des Basismodells

linearen Parameter Abschätzung (non-linear estimation) dar, zu dessen Lösung hauptsächlich zwei Methoden eingesetzt werden. Das erste Verfahren geht auf Gauß zurück.
Die Variablen Y(b) des Modells werden in Bezug auf die Parameter b_i in eine Taylor Reihe entwickelt

$$\vec{Y}(t_i,\vec{b}+\vec{\delta}) = \vec{Y}_o(t_i,\vec{b}) + \sum^{i}(\frac{\partial\vec{Y}}{\partial b_j})(\delta)_j = \vec{Y}_o+\vec{P}\vec{\delta} \quad (5.1)$$

wobei $\vec{\delta}$ eine kleine Korrektur des Parametervektors $\vec{b}$ darstellt, und $\vec{P}$ die Matrix des Gradienten $(\delta\vec{Y}/\partial b_j)$ der Modellfunktion
Die Gauß'sche Fehlerfunktion

$$F(\delta) = \sum^{i} [\vec{Y}^*(t_i) - \vec{Y}(t_i)]^2 \quad (5.2)$$

mit dem Meßwertvektor $\vec{Y}^*(t_i)$ und den Modellvariablen $\vec{Y}(t_i)$ wird als Kriterium für die Anpassung des Modells verwendet. Nach einsetzen der linearisierten Modellfunktion $\vec{Y}(t_i,\vec{b}+\vec{\delta})$ stellt Gl. (5.2) eine quadratische Funktion der Parameter-Korrektur $\vec{\delta}$ dar. Die optimalen Modellparameter sind dann gefunden, wenn durch Variation von $\vec{\delta}$ die Fehlerfunktion $F(\vec{\delta})$ ihr absolutes Minimum erreicht hat. Die Bedingungen für $\vec{\delta}_{min}$ lauten bekanntlich

$$\vec{P}^T\vec{P}\vec{\delta}_{min} = \vec{P}^T(\vec{Y}^* - \vec{Y}_o) \quad (5.3)$$

und stellen ein lineares Gleichungssystem zur Bestimmung der Parameter-Korrekturen $\vec{\delta}$ dar.

Das zweite Verfahren ist unter dem Namen Gradienten Verfahren oder "steepest descent method" bekannt. Hierbei ergibt sich der Vektor $\vec{\delta}$ der Parameter Korrektur durch Bildung des negativen Gradienten der Fehlerfunktion $F(\vec{b})$.

$$\vec{\delta} = -(\frac{\partial F}{\partial b_1}, \frac{\partial F}{\partial b_2}, \cdots \frac{\partial F}{\partial b_N})^T = -2P^T(\vec{Y}^* - \vec{Y}_o) \quad (5.4)$$

In anderen Worten, man sucht in Richtung des steilsten Abstieges das Minimum der Fehlerfunktion zu erreichen.

Die Gauß´sche Methode hat den Vorteil schneller Konvergenz in gutartigen Fällen. Sie führt jedoch durch die notwendige Linearisierung, Gl. (5.1), sehr leicht zu instabilem Verhalten bei Funktionen, die nicht überall konvex sind.
Die Gradienten-Methode dagegen konvergiert in allen Fällen anfänglich recht gut. Bei Annäherung an das gesuchte Minimum wird die Konvergenzgeschwindigkeit jedoch untragbar langsam.

Von W. Marquardt wurde eine Kombination beider Methoden entwickelt, die alle Vorteile vereint und die Nachteile weitgehend vermeidet [4]. Das Verfahren nennt sich "maximum neighbourhood method" und beruht darauf, daß bei der Suche nach dem Minimum von $F(\vec{\delta})$ eine Begrenzung des Bereichs für $\vec{\delta}$ vorgenommen wird. Durch diese Nebenbedingung ergibt sich die Bestimmungsgleichung für $\vec{\delta}_{min}$ als

$$(\vec{P}^T\vec{P} + \frac{1}{a}\vec{E})\vec{\delta}_{min} = P^T(\vec{Y}^* - \vec{Y}_o) \qquad (5.5)$$

Der skalare Faktor a bestimmt die Größe des Bereichs $||\vec{\delta}||$. ($\vec{E}$ - Einheitsmatrix)
Ein Vergleich von Gl. (5.3) mit Gl. (5.4) zeigt, daß für große Werte des Faktors a die Gauß´sche Methode angenähert wird. Für kleine Werte von a nähert sich Gl. (5.5) einem Ausdruck, der Gl. (5.4) proportional ist, d.h. die Gradienten Methode wird approximiert.
Hieraus leitet sich die Strategie für die Bemessung des Faktors a ab:
Zu Beginn des Anpassungsverfahrens wird ein kleiner Wert für a gewählt. Dadurch startet das Verfahren mit der Gradienten Methode und ist selbst bei ungünstigen Anfangswerten der Parameter b_j noch konvergent.

Gegen Ende des Anpassungsprozesses wird a sehr groß gewählt. In der Nähe des Minimums ist $F(\vec{b})$ mit Sicherheit konvex und das Gauß'sche Verfahren mit seiner schnellen Konvergenz kann gefahrlos eingesetzt werden.

6. DAS ADAPTIVE MODELL ESADAP

Der Marquardt'sche Algorithmus wurde verwendet um das adaptive Modell zu erstellen. In Figur 5 ist der Aufbau des digitalen Programms ESADAP gezeigt. Das Hauptprogramm MAIN steuert die I/O Operationen. Es veranlaßt den Transfer der Meßwerte vom Band auf den Plattenspeicher und steuert die Ausgabe der optimalen Modellparameter sowie der Simulationsergebnisse. Das Hauptprogramm startet und beendet außerdem den eigentlichen Simulationsvorgang durch direkten Verkehr mit dem Basismodell ESIDYN.

Das Unterprogramm REGR steuert und kontrolliert den Anpassungsprozess der Modellvariablen an die Meßwerte. Die wichtigste Aufgabe vom REGR ist die Überwachung des Suchvorganges nach Min $F(\vec{b})$ entsprechend der oben beschriebenen Strategie der "maximum neighbourhood method". Hierzu gehört die Wahl eines geeigneten Faktors a, die eine wiederholte Berechnung der Fehlerfunktion $F(\vec{b})$ notwendig macht. Weiterhin berechnet REGR die Korrekturvektoren $\vec{\delta}_{min}$ durch Lösung des Gleichungssystems (5.5).
Die hierzu erforderliche Berechnung der Matrix P wird im Unterprogramm FCODE vorgenommen. Die Komponenten von P sind die Ableitungen $(\partial\vec{Y}|\partial b_j)$ der Modellfunktion nach den momentanen Parameterwerten, in unserem Fall also die Gradieten $(\partial\Psi|\partial b_j)$, $(\partial\theta|\partial b_j)$ und $(\partial\Phi|\partial b_j)$.
Diese Differentiale werden in FCODE durch endliche Differenzenausdrücke der Form $[\Psi(b_j+\Delta b_j) - \Psi(b_j)]/\Delta b_j$ angenähert.
Die Werte der Modellfunktion $\Psi(t_i,b_j)$, $\theta(t_i,b_j)$, $\Phi(t_i,b_j)$

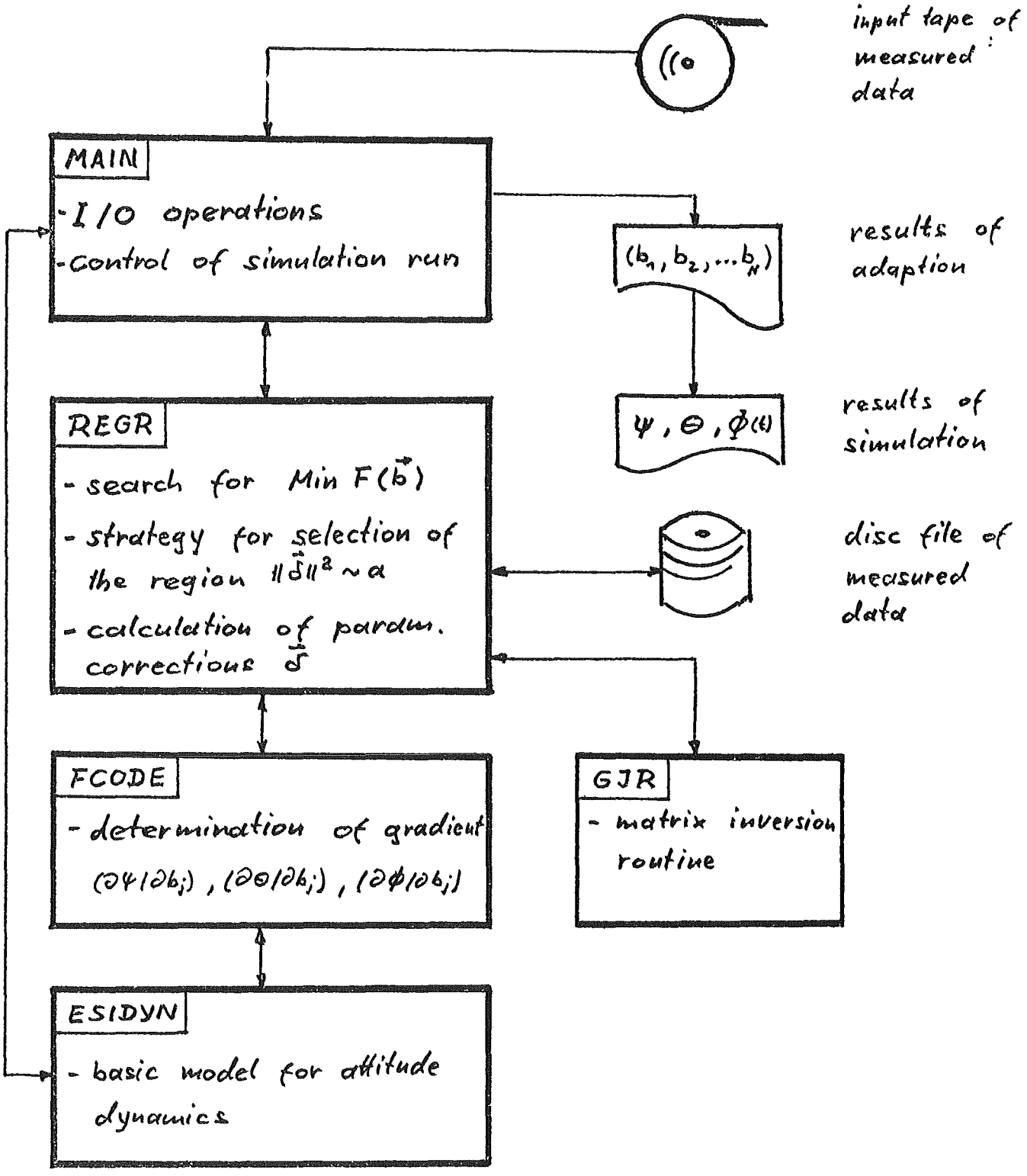


Fig 5: Blockdiagramm des adaptiven Modells ESADAP

zu einer vorgegebenen Zeit t_i und mit vorgegebenen Parametern b_j erhält man durch Aufruf des Unterprogrammes ESIDYN. Dieser Programmteil umfaßt das in Abschnitt 4. erläuterte Basismodell.
Das Unterprogramm GJR schließlich dient dazu, die mehrfach notwendigen Matrix-Inversionen durchzuführen, die nach dem Gauß-Jordan-Rutishauser Verfahren durchgeführt werden.

Der Ablauf des Programms findet in zwei Stufen statt. Er beginnt mit dem Anpassungsprozess des Basismodells, d.h. mit der Suche nach den optimalen Modellparametern durch REGR. Der Suchvorgang ist beendet, wenn die Fehlerfunktion $F(\vec{b})$ einen vorgegebenen Minimalwert ε unterschritten hat. Jetzt werden die gefundenen optimalen Modellparameter an das Hauptprogramm zurückgegeben.
Nach dem Anpassungsprozess startet das Hauptprogramm in direktem Aufruf den Simulationsvorgang. Hierzu werden dem Basismodell die Anfangswerte der Lagewinkel und deren Gradienten sowie die optimalen Parameter und die Simulationsdauer übergeben. Nach beendeter Simulation sorgt das Hauptprogramm wiederum für die Ausgabe der berechneten Lagewinkel auf Liste oder Band.
Das gesamte Modell ESADAP ist für doppelte Wortlänge programmiert und beansprucht zusammen mit dem Basismodell ESIDYN 160K Speicherplatz.
Der Rechenzeitbedarf hängt in erster Linie von der Zahl der Parameter b_j und der Zahl der Meßwerte ab.
Als Beispiel seien folgende Zahlen genannt:
Für ein Kollektiv von 300 Meßpunkten benötigte das Programm auf einer IBM 360/65 bei

6 Parametern	31 min	CPU-Zeit
16 Parametern	60 min	CPU-Zeit
25 Parametern	101 min	CPU-Zeit

Literatur:

[1] ESRO-I Spacecraft Description, TR-10, ESRO Paris

[2] R. Kammüller, Nonlinear Resonant Rollmotion, AIAA-Journal, April 1971

[3] R. Kammüller, Roll Resonance and Roll Control, AIAA-Journal, Feb. 1972

[4] D. Marquard, Algorithm for least-square estimation, J.S.I.Appl. Math., June 1963

SIMULATION DES REGLER MENSCH-VERHALTENS IN MENSCH-FAHRZEUG-SYSTEMEN

W. Berheide und G. Johannsen

Forschungsinstitut für Anthropotechnik
5309 Meckenheim

1. Einleitung

Die Gesamtsystemleistung hochentwickelter Fahrzeuge der Gegenwart und Zukunft hängt entscheidend von den Fähigkeiten und Grenzen des Menschen ab. Daraus ergeben sich für die Entwicklung neuer Mensch-Fahrzeug-Systeme z.B. folgende Fragestellungen: Welche Aufgaben können dem Menschen durch die Automatik abgenommen oder erleichtert werden ? In welchen Fällen ist der Mensch nicht ersetzbar ? Verschiedene Alternativen der Aufgabenteilung zwischen Mensch und Automatik sollten in einer möglichst frühen Entwurfsphase bewertet werden können. Eine dem Systemingenieur vertraute analytische Behandlung dieses Problems wird möglich, wenn das Verhalten aller Untersysteme des geschlossenen Mensch-Maschine-Regelkreises, auch das des Systems Mensch, quantitativ beschreibbar ist.

Seit einigen Jahren wird an der Entwicklung von Regler Mensch-Simulationsmodellen gearbeitet [1], [2], [3]. Am Forschungsinstitut für Anthropotechnik wird z.Zt. mit dem Aufbau eines Programmsystems begonnen, das die systematische Weiterentwicklung und den Vergleich verschiedener Simulationsmodelle erleichtern soll. Außerdem sollen die Modelle für ein breites Spektrum von Fahrzeugführungsaufgaben validiert werden. Über den Stand der vorläufigen Planungen wird in dieser Arbeit berichtet.

2. Einführendes Beispiel zur Modellentwicklung

Die generelle Aufgabenstellung bei dem vorliegenden Simulationsproblem wird durch das im Bild 1 aufgezeichnete Beispiel verdeutlicht.

Der Mensch-Fahrzeug-Regelkreis (oberer Teil des Bildes), in dem der Mensch selbst die Führung eines Fahrzeuges übernimmt, wird mit dem Regler Mensch-

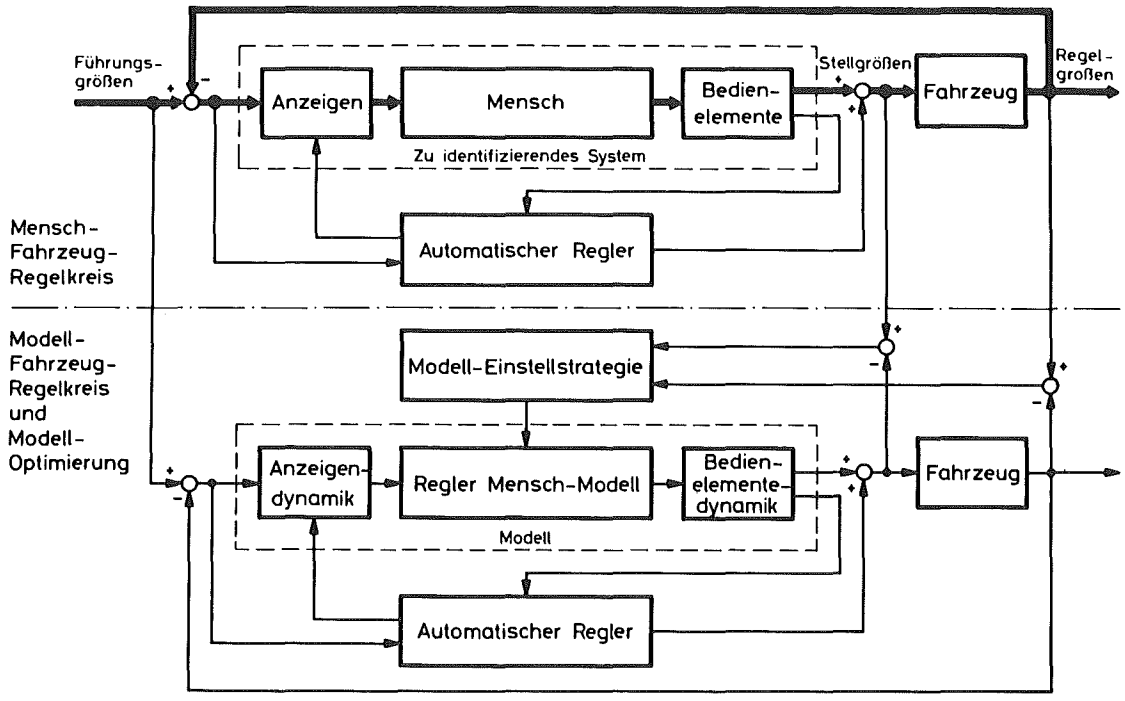


Bild 1. Simulation des Regler Mensch-Verhaltens

Modell-Fahrzeug-Regelkreis (unterer Teil des Bildes) verglichen. Aus der Differenz der Stell- und Regelgrößen wird mittels einer Modell-Einstellstrategie entsprechend gewählter Gütekriterien eine Veränderung der Modellparameter abgeleitet. In den Modell-Fahrzeug-Regelkreis werden die gleichen Führungsgrößen eingegeben wie in den Mensch-Fahrzeug-Regelkreis. Beide Regelkreise besitzen die gleiche Struktur. Während in Experimenten mit dem Menschen das Fahrzeug ein Echtsystem oder ein simuliertes System sein kann, wird im Modell-Fahrzeug-Regelkreis fast immer die Simulation vorherrschen.

3. Verwendetes Rechnersystem

Für die Durchführung der Aufgaben steht ein hybrides Rechnersystem zur Verfügung. Ein Analogrechner EAI 680 ist über Interface und Koppler der Fa. Dornier mit einem Siemens Prozeßrechner 306 gekoppelt. Für die Auslegung des in dieser Arbeit beschriebenen Programmsystems sind im wesentlichen die Eigenschaften des verwendeten Digitalrechners bestimmend. Dem Hauptvorteil einer relativ kurzen Zykluszeit von 0,6 µs stehen folgende Nachteile gegenüber. Interruptprogramme werden rein softwaremäßig über ein Koordinierungsprogramm für

schnelle Alarme mit Programmwechselzeiten in der Größenordnung von 200 µs verwaltet. Das eigentliche Organisationsprogramm belegt zusammen mit dem Koordinierungsprogramm und der notwendigen Hybrid-Systemsoftware etwa die Hälfte des 32 K (24 bit) Kernspeichers. Programmwechselzeiten, die über das Organisationsprogramm verwaltet werden, liegen im Bereich von 1 ms.

Bei dem vorliegenden Problem sollen die Multiprogrammingeigenschaften des vorhandenen Digitalrechners zumindest in zeitkritischen Fällen möglichst wenig genutzt werden, da das Rechnersystem eine relativ langsame Programmwechselstrategie besitzt. Außerdem wird bei Multiprogramming der verfügbare Arbeitsspeicher nicht ökonomisch genutzt, da zumindest in FORTRAN zu jedem einzelnen Programm umfangreiche Systemsoftware, z.B. für Ein-, Ausgaben, gebunden wird. FORTRAN soll aber wegen der angestrebten Benutzerfreundlichkeit in hohem Maße für die einzelnen Programme verwendet werden. Der geringe verfügbare Arbeitsspeicher legt eine möglichst intensive Nutzung der Overlay-Technik in weniger zeitkritischen Teilen des Programmablaufs nahe.

4. Vorgehensweise bei der Analyse und Simulation des Regler Mensch-Verhaltens

Die Verhaltensanalyse und -simulation des Reglers Mensch wird in drei Phasen vorgenommen:

1) Versuchsdurchführung und Datenerfassung,
2) Versuchsauswertung und
3) Modellentwicklung und -validierung.

Den drei Phasen der Verhaltensanalyse und -simulation entsprechen drei Ebenen des Programmsystems, die modular aufgebaut sind. Die Versuchsauswertung wird digital vorgenommen. Die beiden anderen Ebenen enthalten hybride Programmteile. Während Experimente mit Versuchspersonen jedoch Echtzeit-Programmabläufe sein müssen, laufen Modelloptimierungen in möglichst stark gerafftem Zeitmaßstab ab, um Rechenzeit zu sparen.

4.1 Versuchsdurchführung und Datenerfassung

In der ersten Phase werden Experimente mit Versuchspersonen durchgeführt. Ein

Steuerprogramm mit Versuchsablaufsteuerung hat das Kommando über die einzelnen Programmteile (s. Bild 2). Dieses Programm wird für jede Versuchsreihe vom Experimentator neu geschrieben oder aus vorhandenen Programmteilen zusammengestellt. Die Versuchsablaufsteuerung übernimmt dabei das Setzender Anfangsbedingungen und Betriebsarten am Analogrechner, gibt Hinweise an den Versuchsleiter und die Versuchsperson über Bildschirm oder Fernschreiber, startet die Simulationsprogramme und steuert bzw. überwacht den zeitlichen Ablauf des Versuches.

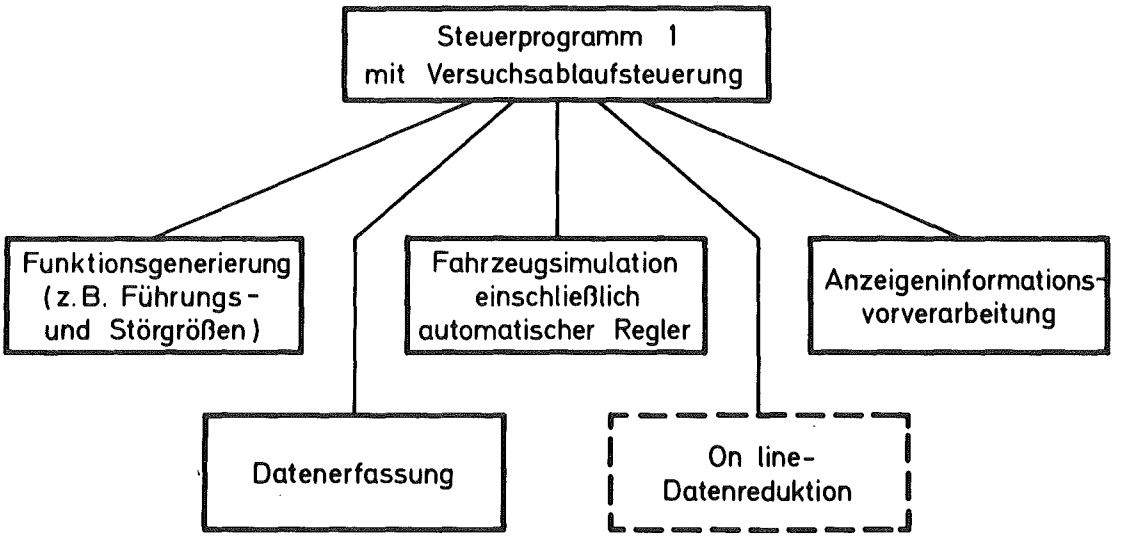


Bild 2. Versuchsdurchführung und Datenerfassung

Führungs- und Störgrößen können deterministische oder stochastische Vorgänge sein. Die stochastischen Prozesse werden entweder durch analoge Rauschgeneratoren oder digitale Pseudo-Rauschgeneratoren erzeugt. Während die analoge Erzeugung der Führungs- und Störgrößen im Rahmen der Fahrzeugsimulation erfolgen kann, übernehmen die Funktionsgenerierungsprogramme diese Aufgabe im Digitalbereich.

Die Fahrzeugsimulation ist ein Hybridprogramm. Aus methodischen und Kostengründen wird die Fahrzeugdynamik meistens mit einem Rechner simuliert. Eine an das Rechnersystem angeschlossene Versuchskabine enthält Anzeigegeräte und Bedienelemente an den Schnittstellen zwischen Mensch und Fahrzeug. Programme

zur Anzeigeninformationsvorverarbeitung werden für die Bilderzeugung und für Berechnungen abgeleiteter Größen, z.B. Voranzeigen, verwendet.

Bei der Datenerfassung besteht das Problem, in möglichst kurzer Zeit viele Daten aus diskreten und kontinuierlichen Prozessen zu speichern. In besonders zeitkritischen Fällen wird man die Daten direkt auf den Plattenspeicher bringen. Soweit die Zeit reicht, wird man Datenreduktion und -umwandlung on-line durchführen.

4.2 Versuchsauswertung

Die Versuchsauswertung erfolgt größtenteils off-line. Es soll aber daran gedacht werden, diese Programmphase später bei vergrößertem Kernspeicher und bei weniger zeitkritischen Experimenten teilweise mit in die Versuchsdurchführung einzubeziehen.

Der Kern dieses Programmsystems sind die Auswertungen (s. Bild 3). Es besteht aus Programmen für statistische Verfahren, z.B. Varianzanalyse, und aus Programmen zur Signal- und Systemanalyse stochastischer Prozesse.

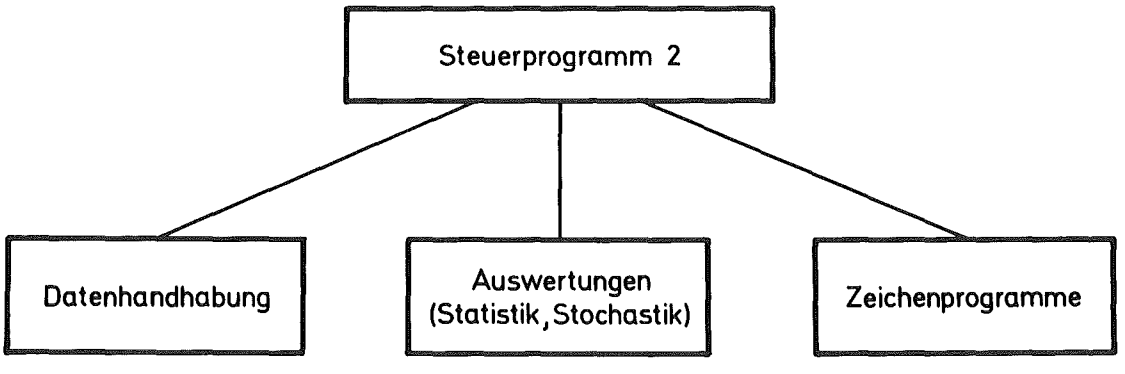


Bild 3. Versuchsauswertung

Mit Hilfe von Korrelationsverfahren wird eine statistische Analyse von Signalen vorgenommen. Die Signaleigenschaften werden durch Kenngrößen und Kennfunktionen wie Korrelations-, Verteilungs- und Leistungsdichtefunktionen beschrieben. Hieraus kann eine lineare Systembeschreibung, z.B. für den Regler

Mensch, im Frequenzbereich hergeleitet werden. Ferner werden die nichtlinearen Systemeigenschaften untersucht. Die Datenein-, -ausgabe und -aufbereitung erfolgt über Datenhandhabungsprogramme. Zur Ergebnisdarstellung werden verschiedene Zeichenprogramme geschrieben.

4.3 Modellentwicklung und -validierung

Die eigentliche Modellentwicklung wird im allgemeinen ebenfalls off-line erfolgen (s. Bild 4). Modelle verschiedenen Komplexitätsgrades und unterschiedlicher Struktur werden anhand der vorhandenen Versuchsdaten validiert und teilweise miteinander verglichen.

Neben den parameterfreien Modellen, die sich direkt aus der Versuchsauswertung ergeben, oder Modellen, die mit Methoden der optimalen Regelungstheorie

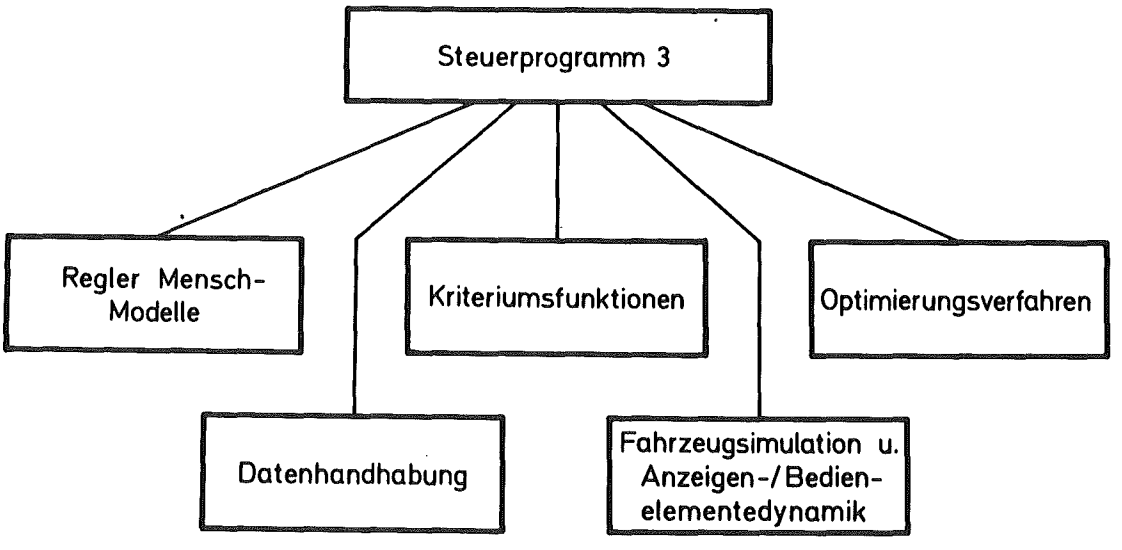


Bild 4. Modellentwicklung und -validierung

errechnet werden, gibt es solche, deren Parameter mit Hilfe einer Einstellstrategie optimiert werden. Diese Modelle können aus Digitalprogrammen bestehen oder auch analoge Elemente enthalten. Mit Optimierungsprogrammen (z.B. kontinuierliche Parameteroptimierung, diskrete Gradientenverfahren, Suchverfahren) werden verschiedene Einstellstrategien realisiert. Dabei werden verschiedene Kriteriumsfunktionen, wie z.B. mittleres Fehlerquadrat, Effektivwertfehler, eingesetzt.

Die Funktionsblöcke Datenhandhabung und Fahrzeugsimulation entsprechen etwa den in den beiden anderen Phasen gezeigten Programmen. Die Fahrzeugsimulationsprogramme enthalten, wie auch aus Bild 1 hervorgeht, jetzt zusätzlich die Anzeigen- und Bedienelementedynamiken und gegebenenfalls Funktionsgenerierungen.

5. Probleme bei der Implementierung des Programmsystems

Der Programmablauf in den Phasen 1 und 3 verdient besondere Beachtung, da Hybrid-Programmteile enthalten sind. Es laufen ein oder mehrere Simulationsprogramme als Taktprogramme mit hoher Priorität. Zur Vorbereitung dieser Taktprogramme gehört ein Aufbau der Befehle für die hybride Simulation und das Freigeben der entsprechenden Takteingänge. Diese Aufgabe übernimmt das Steuerprogramm vor dem eigentlichen Programmlauf. Da die hybride Software, die hierfür benutzt wird, recht umfangreich ist, empfiehlt es sich, diesen Vorbereitungsteil in Overlaystruktur ablaufen zu lassen. Die Taktprogramme sind eigenständige Hauptprogramme und können mit vorhandenen Unterprogrammen gebunden werden.

Im Rahmen des Steuerprogramms läuft während der Simulations- und eigentlichen Programmphase das Datenerfassung- bzw. Datenhandhabungsprogramm als Hintergrundprogramm. Es wird jeweils vom Taktprogramm unterbrochen und läuft, wenn es nicht Datenbearbeitungen ausführt, in einer Warteschleife. Beim Datenerfassen wird ein Wechselpuffer benutzt. In den einen Puffer werden die Daten aus dem Experiment übernommen, während vom anderen die Daten auf die Platte transferiert werden. Datenreduktions- bzw. Datenhandhabungsprogramme kann das Datenerfassungsprogramm nach Belieben aufrufen.

Im Programmsystem gibt es eine Reihe von Unterprogrammen, z.B. Optimierungsverfahren, Regler Mensch-Modelle, Funktionsgenerierungsprogramme, die beliebig aufrufbar in einer Library auf dem Plattenspeicher stehen sollen. Dies sind zum größten Teil FORTRAN-Unterprogramme oder Assemblerprogramme, die mit FORTRAN-Aufrufen angesprochen werden können. Aus der Forderung, gleichartige Programme innerhalb eines Funktionsblocks, z.B. Optimierungsprogramme, beliebig austauschen zu können, folgt, daß diese Programme gleiche Anzahl

und Gleichartigkeit der Parameter besitzen müssen. Die Parameter von "Unter-unterprogrammen", die auch wieder beliebig ausgetauscht werden können, müssen mit übergeben werden.

Aus der Forderung nach variablen Versuchs- bzw. Auswertungsintervallen ergibt sich, daß die Felder der Programme variabel sein müssen. Das ist der Hauptgrund dafür, daß weitgehend die Unterprogrammtechnik mit Übergabe aktueller Parameter angewendet und auf COMMON-Bereiche verzichtet wird. Da bei der Parameterübergabe pro Parameter etwa 20 μs Rechenzeit vergeht, wird daran gedacht, die Übergabe zu modifizieren oder möglichst viele Parameter in einem Feld gemeinsam zu übergeben.

Es wird erwartet, daß der beschriebene modulare Aufbau des Programmsystems, unter Berücksichtigung des zur Verfügung stehenden Rechnersystems, eine optimale Anpassung an die wechselnden Fragestellungen des Forschungsgebietes Regler Mensch darstellt. Zugleich hoffen wir, daß der Stand unserer derzeitigen Überlegungen Anregungen für die Vorgehensweise bei verwandten Simulationsaufgaben geben kann. Gleichzeitig hoffen wir, für unsere weiteren Arbeiten Anregungen aus diesem Workshop gewinnen zu können.

Literatur :

[1] Mc Ruer, D.T. and Jex, H.R. — A review of quasi-linear pilot models. IEEE Trans. Human Factors in Electronics, vol. HFE-8, pp. 231-249, Sept. 1967.

[2] Kleinman, D.L, Baron, S., and Levison W.H. — An optimal control model of human response, Part I : Theory and validation. Automatica, vol. 6, pp. 357-369, May 1970.

[3] Johannsen, G. — Development and optimization of a nonlinear multiparameter human operator model. IEEE Trans. Syst. Man Cybern., vol. SMC-2, pp. 494-504, Sept. 1972.

G.-U. Weigand
Abt.: EP24
i/Hs. TC GmbH.
7750 K o n s t a n z
Max-Stromeyer-Straße 116

Konstanz,
11. April 1973

"Bestimmung des statistischen Fehlers im Simulationsmodell eines Teilnehmerrechensystems"

Inhalt:

1. Einleitung

Das programmierte Simulationsmodell BS3SIM dient der Untersuchung des Teilnehmerrechensystems 440. Es wird in der augenblicklichen Ausbaustufe bei der

* Planung der Konfiguration des TR440
  und
* Untersuchung von Betriebsmittel-Zuteilungsstrategien

eingesetzt.

Als Optimierungsziele werden im Abschnittsbetrieb (Batch) der Auftragsdurchsatz und im Gesprächsbetrieb (Timesharing) die Konsolreaktionszeit herangezogen.

In Zeichnung 1 sind die Verkehrswege des Modells dargestellt. Sie erkennen die Betriebsmittel Rechnerkerne, Transportkanäle und Arbeitsspeicher. Die verschiedenen Hintergrundspeicher erscheinen nicht, weil sie im Gegensatz zum Arbeitsspeicher im Modell unendliche Kapazität haben.

Im folgenden beschränke ich mich auf die Betrachtung des reinen Abschnittsbetriebs und den dafür wesentlichen Auftragsdurchsatz. Verdrängungen wegen Arbeitsspeicherengpasses werden nicht zugelassen, d.h. Abschnitte, die bei Bearbeitungsbeginn im Arbeitsspeicher gemäß ihres Bedarfs eingeplant worden sind, können während der Bearbeitung weder Arbeitsspeicher anfordern noch aufgeben.

Da die langsame Peripherie, wie zum Beispiel Kartenleser und Schnelldrucker, verhältnismäßig wenige Hintergrundtransporte verursacht, ist das Spooling des TNS440 nicht in das Modell aufgenommen worden.
Damit zerfällt ein Abschnitt in nur zwei mögliche Bearbeitungsphasen, in:

* Teilaufgaben
  und
* Ergänzungstransporte.

Die Teilaufgaben benötigen zur Erledigung einen Rechnerkern, die Ergänzungstransporte einen Transportkanal. Die beiden Bearbeitungsphasen Teilaufgabe und Ergänzungstransport wechseln einander ab bis der Abschnitt beendet ist.

Die Definition des Durchsatzes D ist:
Anzahl der bearbeiteten Abschnitte N geteilt durch die dafür benötigte Zeit T(N);
(1) $D = N/T(N)$
Diese Definition von D reicht nicht aus. Es müssen noch Ausführungsbestimmungen hinzukommen, wie N und T(N) zu ermitteln sind:

* der Rechner bzw. sein Modell sind bei Messungen bzw. Simulationen in der Auftragssättigung zu betreiben
* die Anlaufphase ist vor Beginn der Messung bzw. des Simulationsexperiments abzuwarten
* N ist so groß wie möglich zu wählen, um den statistischen Fehler so klein wie möglich zu halten.

Ganz allgemein kann D als eine Funktion von drei Eingangsgrößen aufgefaßt werden; diese sind

* die Betriebssystemkonstruktion
* die Hardwarekonstruktion
* das Jobprofil.

Bei dem Simulationsmodell BS3SIM vereinfachen sich diese Eingangsparameter zu

* den Betriebsmittelzuteilungsstrategien
* der Konfiguration des Rechners (ausgenommen die lang-

same Peripherie)

* einer Beschreibung des Jobprofils durch Verteilungen
  1. der Dauer der Teilaufgaben
  2. der Dauer der Ergänzungstransporte
  3. der Kanalzugriffe.

Ich komme nun zu dem eigentlichen Thema dieses Vortrags: Der Berechnung des statistischen Fehlers des durch Simulation gewonnenen Durchsatzes D. Die Lösung des Problems wird durch die folgenden Modellvereinfachungen ermöglicht:

1. Die Dauer der Teilaufgaben t_{RK} und der Ergänzungstransporte t_{TK} sei exponentiell verteilt. Die exponentielle Verteilung hat die Eigenschaft, daß die Verteilung der Restzeit gleich der ursprünglichen Verteilung ist:

   $$P\{T \leq t\} = F(t) = 1 - e^{-\alpha t}$$
   $$= P\{T - T_o \leq t \mid T \geq T_o\} = F(t)$$

   Die Folge davon ist, daß die Reihenfolge der Bedienung der Teilaufgaben, d.h. die Rechnerkernzuteilungsstrategie, keinen Einfluß auf den Durchsatz ausübt.

2. Es wird angenommen, daß im Arbeitsspeicher immer eine vorgebbare Anzahl von Abschnitten geladen ist. Das beinhaltet,
   - daß das Modell in der Auftragssättigung betrieben wird, d.h. bei Abschnittsende wird sofort ein neuer begonnen;
   - daß eine feste Anzahl von Programmplätzen zur Verfügung steht, von denen jeder Abschnitt genau einen einnimmt.

2. Das vereinfachte Rechnermodell

In diesem Beitrag werde ich mich auf ein Rechnermodell beschränken mit den

aktiven Betriebsmitteln

* 1 Rechnerkern RK
* 1 Transportkanal TK

und den

passiven Betriebsmitteln

* 1 endlicher Arbeitsspeicher ASP_{ABW} mit ABW Programmplätzen, entsprechend der Anzahl der Abwickler im BS3
* 1 unendlicher Hintergrundspeicher HSP_{∞}.

Die Anwendbarkeit der im folgenden entwickelten Methode wird durch eine andere Zusammenstellung der aktiven Betriebsmittel nicht berührt.

Die Bedienungszeit t_{RK} des Rechnerkerns (Dauer einer Teilaufgabe) besitze die Verteilung

$$P\{t_{RK} \leq t\} = 1 - e^{-\mu t}$$

und die Bedienungszeit t_{TK} des Transportkanals (Dauer eines Ergänzungstransports) die Verteilung

$$P\{t_{TK} \leq t\} = 1 - e^{-\lambda t}$$

Wir definieren eine Menge Z von Zuständen des Modells:

$$Z = \{E_o, \ldots, E_{ABW}\}$$

wobei der Zustand E_i bedeutet, daß i Plätze der Warteschlange vor dem Rechnerkern besetzt sind. Der Rechnerkernplatz selbst gehöre mit zur Warteschlange.
Im Zustand E_o steht der Rechnerkern leer, im Zustand E_{ABW} der Transportkanal. Bei allen Zuständen E_i, $1 \leq i \leq ABW-1$, ist sowohl der Rechnerkern als auch der Transportkanal belegt.

Die Zeitachse wird durch Ereignisse, die den Zustand des Modells verändern, in Zeitintervalle mit den Teilungspunkten t_K zerlegt.

E sei die Abbildung von der Zeit t in die Menge Z der Zustände. Die Zeitintervalle $\Delta t(K) = t_{K+1} - t_K$ sind Zufallsvariable, deren Verteilung vom Zustand des Rechnermodells abhängt:

1. Im Zeitpunkt t_K werde der Zustand $E_o = E(t_K)$ erreicht. Die Verteilung von $\Delta t(K)$ ist dann
   $P\{\Delta t(K) \leq t \mid E(t_K) = E_1\} = 1 - e^{-\lambda t}$

2. Im Zeitpunkt t_K werde der Zustand $E_i = E(t_K)$, $1 \leq i < ABW$, erreicht. Die Verteilung von $\Delta t(K)$ ist dann
   $P\{\Delta t(K) \leq t \mid E(t_K) = E_i\} = 1 - e^{-(\mu+\lambda)t}$

3. Im Zeitpunkt t_K werde der Zustand $E_{ABW} = E(t_K)$ erreicht. Die Verteilung von $\Delta t(K)$ ist dann
   $P\{\Delta t(K) \leq t \mid E(t_K) = E_{ABW}\} = 1 - e^{-\mu t}$

Da die Restrechenzeit $t_{Rest} = t_{RK} - t_o$ die gleiche Verteilung wie t_{RK} besitzt, können Unterbrechungen bei der Teilaufgabenbearbeitung durch andere Teilaufgaben zugelassen werden, ohne die Verhältnisse zu ändern.

Wir werden nun die Übergangswahrscheinlichkeiten
$P_{ij}(K) = P\{E(t_{K+1}) = E_j \mid E(t_K) = E_i\}$
bestimmen.

Es liege der Zustand E_o vor. Dann ist nur der Transportkanal TK belegt und er generiert beim nächsten Ereignis unabhängig von der Schrittzahl K sicher eine neue Teil-

aufgabe; auf den Zustand E_o folgt mit der Wahrscheinlichkeit 1 der Zustand E_1:
$P_{o\ 1} = 1$ unabhängig von K.

Es liege der Zustand E_i, $1 \leq i < ABW$, vor. Rechnerkern RK und Transportkanal TK sind gleichzeitig belegt. Die Wahrscheinlichkeit, daß der Rechnerkern bzw. Transportkanal in dem Zeitintervall (t, t+dt) fertig wird, ist μ dt bzw. λ dt. Die Wahrscheinlichkeit, daß in diesem Zeitintervall irgendein Ereignis stattfindet, ist $(\mu + \lambda)$dt. Die bedingte Wahrscheinlichkeit, daß in diesem Zeitintervall bei einem Ereignis der Rechnerkern (Transportkanal) eine Teilaufgabe fertigstellt (generiert) ist $\frac{\mu}{\lambda+\mu}$ ($\frac{\lambda}{\lambda+\mu}$); also folgt auf den Zustand E_i mit der Wahrscheinlichkeit $\frac{\mu}{\lambda+u}$ ($\frac{\lambda}{\lambda+\mu}$) der Zustand E_{i-1} (E_{i+1}) :

$$P_{i\ i-1} = \frac{\mu}{\lambda+\mu}$$

unabhängig von K

$$P_{i\ i+1} = \frac{\lambda}{\lambda+\mu}$$

Es liege der Zustand E_{ABW} vor. Damit ist nur der Rechnerkern belegt. Mit der Wahrscheinlichkeit 1 wird beim nächsten Ereignis eine Teilaufgabe fertiggestellt; auf E_{ABW} folgt mit Sicherheit E_{ABW-1} :

$$P_{ABW\ ABW-1} = 1$$

unabhängig von K

Offensichtlich sind alle übrigen Übergangs-Wahrscheinlichkeiten unabhängig von K gleich 0.

Man erkennt nun, daß es sich um eine Markovkette mit der Übergangsmatrix $P = (p_{ij})$ für alle Schritte K handelt.

$$(2)\quad P = \begin{pmatrix} 0 & 1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ m & 0 & n & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & m & 0 & n & \dots & 0 & 0 & 0 & 0 \\ & & & \vdots & & & & & \\ 0 & 0 & 0 & 0 & \dots & m & 0 & n & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & m & 0 & n \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 1 & 0 \end{pmatrix}$$

$$m = \frac{\mu}{\lambda+\mu} \qquad n = \frac{\lambda}{\lambda+\mu}$$

3. Der stationäre Zustand des Rechnermodells

Der Markovprozeß mit der Übergangsmatrix P ist periodisch mit der Periode t = 2. Die Menge der Zustände zerfällt damit in zwei disjunkte Gruppen G_1, G_2 mit folgender Eigenschaft:

Befindet sich der Prozeß nach n Schritten im Zustand $E_K \in G_1$, dann befindet er sich nach n+1 Schritten mit Sicherheit in einem Zustand $E_1 \in G_2$, nach n+2 Schritten mit Sicherheit in einem Zustand $E_m \in G_1$ usw. Es ist ersichtlich, daß man in G_1 die geraden, in G_2 die ungeraden Zustände zusammenfassen kann.

Wir bezeichnen mit $P_{ij}^{(n)}$ die Elemente der Matrix P^n. $P_{ij}^{(n)}$ ist die Wahrscheinlichkeit, daß der Prozeß, wenn er sich anfangs im Zustand E_i befand, nach n Schritten den Zustand E_j erreicht. Wir stellen fest, daß alle Zustände der Markovkette rekurrent sind, d.h. es gibt keine noch so große positive ganze Zahl $N < \infty$, so daß für alle $n \geq 0$

$$P_{jj}^{(n+N)} \geq 0$$

ist.

Damit gilt (siehe W. FELLER: An Introduction to Probability Theory and its Applications Vol.I, Seite 405):

Sei $E_j \in G_\alpha$, dann ist

(3) $$P_{jK}^{(2n)} = \begin{cases} 2\,U_K & \text{wenn } E_K \in G_\alpha \\ 0 & \text{wenn } E_K \notin G_\alpha \end{cases}$$

wobei die U_K durch $\sum_K U_K = 1$ normierte Lösung der Gleichung

(4) $$U_K = \sum_\nu U_\nu P_{\nu K}$$ sind.

Die Lösung von (3) ergibt bei ABW = 2, 3 und 4 Programmplätzen im Arbeitsspeicher

(5)ABW = 2 :
$$U_o = m$$
$$U_1 = 1/2$$
$$U_2 = n$$

(6)ABW = 3 :
$$U_o = m^2 / (m^2 + n^2 + 1)$$
$$U_1 = m / (m^2 + n^2 + 1)$$
$$U_2 = n / (m^2 + n^2 + 1)$$
$$U_3 = n^2 / (m^2 + n^2 + 1)$$

(7)ABW = 4 :
$$U_o = m^2 / (m^3 + m^2n + mn+n^2+n^3)$$
$$U_1 = m^2n / (m^3 + m^2n + mn+n^2+n^3)$$
$$U_2 = mn / (m^3 + m^2n + mn+n^2 + n^3)$$
$$U_3 = n^2 / (m^3 + m^2n + mn+n^2+n^3)$$
$$U_4 = n^3 / (m^3 + m^2n + mn+n^2+n^3)$$

4. Der Durchsatz und der statistische Fehler

Wir wollen nun sehen, wie der Durchsatz aus der relativen Belegung B_{RK} bzw. B_{TK} der Betriebsmittel RK bzw. TK berechnet werden kann. Wir hatten bei der Definition (1) des Durchsatzes vorgeschrieben, daß das Modell in der Auftragssättigung betrieben werden muß. Da alle Abschnitte das gleiche Rechenzeit- und Transportzeitverhalten haben, ist Abschnittsende bzw. -anfang eine willkürliche Schnittstelle.

Man kann daher statt des Durchsatzes von Abschnitten den Durchsatz von Teilaufgaben betrachten. Aus dem Teilaufgabendurchsatz erhält man den Auftragsdurchsatz, indem man durch die mittlere Anzahl von Teilaufgaben eines Abschnitts teilt.

Der Teilaufgabendurchsatz ist

(8) $$d = \frac{n}{t(n)}$$

$n \mathrel{\hat{=}}$ Anzahl der Teilaufgaben

$t(n) \mathrel{\hat{=}}$ zur Bearbeitung von n Teilaufgaben benötigte Zeit

Wenn n genügend groß ist, dann gilt

$$n \cdot E(t_{RK}) = \sum_{K=1}^{n} t_{RK}(K)$$

Daraus folgt:

(9) $$d = \frac{\sum_{K=1}^{n} t_{RK}(K)}{t(n)} \cdot \frac{1}{E(t_{RK})} = B_{RK} \cdot \frac{1}{E(t_{RK})}$$

Da auf jede Teilaufgabe ein Ergänzungstransport folgt, kann n in (8) auch als die Anzahl von Ergänzungstransporten interpretiert werden. Damit gilt auch

$$(10) \quad d = B_{TK} \cdot \frac{1}{E(t_{TK})}$$

Wir definieren die Zufallsvariablen

$\delta^i(n)$:

$$\delta^i(n) = \begin{cases} 1 \text{ wenn } E(t) = E_i \\ 0 \text{ wenn } E(t) \neq E_i \end{cases} \qquad t_n < t \leq t_{n+1}$$

Bei einem Simulationsexperiment mit N Ereignissen entsteht der Wert für B_{RK} nach folgender Formel

$$(11) \quad B_{RK}(N) = 1 - \frac{\sum_{n=1}^{N} \delta^0(n)\,\Delta t(n)}{\sum_{i=0}^{ABW} \sum_{n=1}^{N} \delta^i(n)\,\Delta t(n)}$$

Die Wahrscheinlichkeitsverteilung der Zufallsvariablen $\delta^j(n)$ ist

$$(12) \quad P\{\delta^j(2n) = 1\} = P_{ij}^{(2n)} \qquad E_i, E_j \in G_\alpha$$
$$P\{\delta^j(2n) = 0\} = 1 - P_{ij}^{(2n)} \qquad \alpha = 1,2$$

Man erkennt, daß die Zufallsvariablen $\delta^i(n)$ bei festem i und n=1, 2, ..., N nicht alle voneinander unabhängig sind. Wenn im Schritt n festgestellt wird, daß $\delta^i(n)$ gleich 1 ist, dann ist damit die Verteilung $P_j(n)$ der Zustände Z gleich δ_{ij} (= Kronecker-Symbol).

Sei $E_i \in G_\alpha$. Die "Erinnerung" an die Verteilung der Zustände aus G_α ist erst dann verlorengegangen, wenn man so viele Perioden K abwartet, daß die Verteilung $P_{ij}(2.K)$ der Zustände G unabhängig von der Verteilung δ_{ij} ist.

Unser Ziel ist es, in der Berechnung von B_{RK} nur statistisch voneinander unabhängige Summanden zu haben. Wir stellen zunächst fest, daß es von den Zufallsvariablen

$\Delta t(n)$ drei Typen gibt:

1. Die $\Delta t(n)$ besitzen bei leerem Rechnerkern die Verteilung $1 - e^{-\lambda t}$:

$$\Delta t(n) = t_{A(n)}$$

2. Die $\Delta t(n)$ besitzen bei belegtem Rechnerkern und Transportkanal die Verteilung $1 - e^{-(\mu+\lambda)t}$

$$\Delta t(n) = \Delta t_{B(n)}$$

3. Die $\Delta t(n)$ besitzen bei leerem Transportkanal die Verteilung $1 - e^{-\mu t}$

$$\Delta t(n) = t_{C(n)}$$

Daraus folgt

$$(13)\quad B_{RK}(N) = 1 - \frac{A(N)}{A(N) + B(N) + C(N)}$$

$$A(N) = \sum_{n=1}^{N} \delta^{o}(n)\, \Delta t_{A(n)} \;/\; N$$

$$B(N) = \sum_{i=1}^{ABW-1} \sum_{n=1}^{N} \delta^{i}(n)\, \Delta t_{B(n)} \;/\; N$$

$$C(N) = \sum_{u=1}^{N} \delta^{ABW}(n)\; t_{C(n)} \;/\; N$$

Wir werden nun die statistisch abhängigen Summanden in den Ausdrücken für A(N), B(N) und C(N) entfernen und berücksichtigen, daß über eine Periode gemittelt wird. Es ergibt sich dann

$$(14)\quad A(N.K) = \sum_{n=1}^{N} \frac{\delta^{o}(n.K) + \delta^{o}(n.K+1)}{2 \cdot N}\, \Delta t_{A(n.K)}$$

$$(15)\quad B(N.K) = \sum_{i=1}^{ABW-1} \sum_{n=1}^{N} \frac{\delta^{i}(n.K) + \delta^{i}(n.K+1)}{2 \cdot N}\, \Delta t_{B(n.K)}$$

$$(16)\quad C(N.K) = \sum_{n=1}^{N} \frac{{}^{ABW}(n.K) + {}^{ABW}(n.K+1)}{2 \cdot N}\, \Delta t_{C(n.K)}$$

O.B.d.A. kann $\delta^{j}(n.K+1) = 0$ angenommen werden. Durch die Mitteilung über die Periode braucht nicht mehr zwi-

schen den Gruppen G_1 und G_2 unterschieden zu werden.
In den Ausdrücken (14) (15) (16) ist es nur noch erforderlich, die Zufallsvariable $\xi^i(n) = \frac{\delta^i(n.K) + \delta^i(n.K+1)}{2}$ zu betrachten.
Die Verteilung dieser Zufallsvariablen ist

(17) $P\{\xi^i(n) = 1\} = U_i$
$P\{\xi^i(n) = 0\} = 1-U_i$

Für N gegen unendlich erhält man

(18) $\lim_{N \to \infty} A(N.K) = U_o \; E(\Delta t_A)$

(19) $\lim_{N \to \infty} B(N.K) = \sum_{i=1}^{ABW_{-1}} U_i \; E(\Delta t_B)$

(20) $\lim_{N \to} C(N.K) = U_{ABW} \; E(\Delta t_C)$

Wir bezeichnen mit
ΔA, ΔB und ΔC die statistischen Fehler von A, B und C.
Der statistische Fehler von B_{RK} ist damit

(21) $$\Delta B_{RK} = \left|\frac{\partial B_{RK}}{\partial A}\right| \Delta A + \left|\frac{\partial B_{RK}}{\partial B}\right| \Delta B + \left|\frac{\partial B_{RK}}{\partial C}\right| \Delta C$$

Für den relativen Fehler von d bzw. D ergibt sich, wenn man beachtet, daß $E(t_{RK})$ aus den Eingangsdaten der Simulation ermittelt wird und damit keinen Fehler aufweist:

(22) $$\frac{\Delta d}{d} = \frac{\Delta D}{D} \leq \left|\frac{1}{A+B+C}\right| \Delta A + \left|\frac{A}{(A+B+C)(B+C)}\right| (\Delta B + \Delta C)$$

Für die statistischen Fehler ΔA, ΔB und ΔC folgt aus (18) (19) und (20)

(23) $\Delta A = \Delta U_{ABW-1} \cdot E(\Delta t_A) + U_o \, \Delta E(\Delta t_A)$

(24) $\Delta B = \sum_{i=1} (\Delta U_i \cdot E(\Delta t_B) + U_i \cdot \Delta E(\Delta t_B))$

(25) $\Delta C = \Delta U_{ABW} \cdot E(\Delta t_C) + U_{ABW} \cdot \Delta E(\Delta t_C)$

Die statistischen Fehler

$$\Delta U_i = \left| \frac{\sum_{n=1}^{N} \xi^i(n)}{N} - U_i \right|$$

(26)

$$\Delta E(\Delta t_{A,B,C}) = \left| \frac{\sum_{n=1}^{L_{A,B,C}} \Delta t_{A,B,C}(n)}{L_{A,B,C}} - E(\Delta t_{A,B,C}) \right|$$

$$L_A = \left[U_o \cdot N \right]$$

(27)

$$L_B = \left[\sum_{i=1}^{ABW-1} U_i \cdot N \right]$$

$$L_C = \left[U_{ABW} \cdot N \right]$$

können mit der TSCHEBYSCHEW'schen Ungleichung abgeschätzt werden.

Aus der TSCHEBYSCHEW'schen Ungleichung folgt für paarweise unabhängige Zufallsvariable X(1),...,X(N), die die gleiche Verteilung besitzen, daß für beliebiges positives ε gilt:

$$P\{|\eta_N - a| > \varepsilon\} \leq \frac{\sigma^2}{\varepsilon^2} \cdot \frac{1}{N},$$

wobei $\eta_N = \frac{\sum_{n=1}^{N} X(n)}{N}$, a der Mittelwert, σ die Streuung der Variablen X(i) ist.

Die Mittelwerte und Streuungen der verschiedenen Zufallsvariablen sind in folgender Tabelle zusammengefaßt:

(28)

Zufallsvariable	Δt_A	Δt_B	Δt_C	ξ^i
Mittelwert	$1/\lambda$	$1/(\lambda+\mu)$	$1/\mu$	U_i
Streuung	$1/\lambda^2$	$1/(\lambda+\mu)^2$	$1/\mu^2$	$U_i(1-U_i)$

Wir geben eine Wahrscheinlichkeit P_{GRENZ} vor, mit der die Fehler ΔU_i und $\Delta E(\Delta t_{A,B,C})$ größer als gewisse positive

Zahlen ε sein dürfen:

$$P_{GRENZ} \approx \frac{\sigma^2}{\varepsilon^2} \cdot \frac{1}{N} \Rightarrow \varepsilon = \sigma / \sqrt{N \cdot P_{GRENZ}}$$

Für die verschiedenen Zufallszahlen ergeben sich daraus die folgenden Werte für ε :

(29)

	Δt_A	Δt_B	Δt_C	ξ^i
ε	$\frac{1}{\lambda \sqrt{L_A \cdot P_{GRENZ}}}$	$\frac{1}{(\lambda+\mu)\sqrt{L_B \cdot P_{GRENZ}}}$	$\frac{1}{\mu \sqrt{L_C \cdot P_{GRENZ}}}$	$\sqrt{\frac{U_i(1-U_i)}{N \cdot P_{GRENZ}}}$

(Bedeutung von $L_{A,B,C}$: siehe (27)

Der relative Fehler des Durchsatzes D kann nun wie folgt berechnet werden:

Man ermittelt die Anzahl K der Perioden bis die Markovkette praktisch stationär ist. Dann löst man (4) und erhält nach der Normierung der Lösung die stationäre Verteilung der Zustände. Wenn die Anzahl S der Schritte in dem Simulationsexperiment bekannt ist, dann erhält man N durch Division von S mit 2K.Mit (22), (23), (24), (25), (28) und (29) ermittelt man eine Fehlergrenze F_{GRENZ} für $\Delta D / D$. Sie wird höchstens mit der Wahrscheinlichkeit P_{GRENZ} überschritten. F_{GRENZ} ist proportional zu

$$1 / \sqrt{N \cdot P_{GRENZ}} .$$

(30) $$\boxed{F_{GRENZ} = R \cdot \frac{1}{\sqrt{N \cdot P_{GRENZ}}}}$$

R ist eine Funktion von μ, λ und U_i.

Eine Unsicherheit bei der Fehlerberechnung besteht in der Wahl der Anzahl K der Perioden, nach denen man die Markovkette für stationär ansieht.

Eine wichtige Einschränkung der Formel (30) ist die, daß F_{GRENZ} aufgrund der linearen Näherung (21) nur den Fehler 1. Ordnung berücksichtigt. Man sieht jedoch, daß die Fehler n-ter Ordnung den Faktor $(1/N)^n$ enthalten, so daß ab $N > 100$ Fehler höherer als 1. Ordnung vernachlässigt werden können.

5. Berechnungen

I. Durchsatzgewinn D/D_o

Durch die mathematische Lösung des Rechnermodells ist es auch möglich geworden, den relativen Gewinn des Durchsatzes bei Übergang von Singleprogramming auf Multiprogramming zu berechnen.

Bei unverändertem Jobprofil ergibt sich

(31) $$D/D_o = B_{RK} / B^o_{RK}$$

D = Durchsatz bei Multiprogramming
D_o = Durchsatz bei Singleprogramming
B_{RK} = relative Rechnerkernbelegung bei Multiprogramming
B^o_{RK} = relative Rechnerkernbelegung bei Singleprogramming

In der Zeichnung 2 ist D / D_o als Funktion der Rechenintensität RI (RI-mittlere Rechenzeit / mittlere Transportzeit) dargestellt (dazugehörige Wertetabelle siehe Tabelle 1).

II. Anzahl der Schritte bis zur Stationarität der Markovkette

Die Verteilung $\vec{p}(n)$ der Zustände nach n Schritten erhält man aus der Anfangsverteilung $\vec{p}(0)$ mit der Formel

$$\vec{p}(n) = (P^n)^T \vec{p}(0)$$

Nach (3) und (4) gibt es zu jeder noch so kleinen positiven reellen Zahl ε eine ganze positive Zahl K, so daß für $p_i(2 \cdot K) \neq 0$:

$$(32) \quad \left| \frac{p_i(2.K) - 2 \cdot U_i}{2 \cdot U_i} \right| \leq \varepsilon$$

und für $p_j(2.K+1)^i \neq 0$

$$(33) \quad \left| \frac{p_j(2.K+1) - 2 \cdot U_j}{2 \cdot U_j} \right| \leq \varepsilon$$

Als Anfangsverteilung sind die Verteilungen $p_j = \delta_{ij}$ geeignet, weil bei der Beobachtung des Modellzustandes zu gewissen Zeitpunkten genau ein Zustand E_i erreicht worden ist.
In Tabelle 2 werden Werte für K bei $\varepsilon \leq 10^{-1}$ und der Anfangsverteilung $p_i = \delta_{1i}$ in Abhängigkeit von der Recheninтensität RI (RI = mittlere Rechenzeit / mittlere Transportzeit) und der Anzahl ABW der Programmplätze angegeben.

III. Anzahl der Ereignisse; Simulationszeit

Die Anzahl S der Ereignisse bestimmt den statistischen Fehler des Durchsatzes.
S ist gegeben durch

$$(34) \quad S = 2 \cdot K \cdot N,$$

wobei K die Anzahl der Perioden ist, bis die Markovkette für stationär angesehen werden kann, und N die Anzahl der

Beobachtungszeitpunkte, bei denen Information gesammelt wird. Der mittlere Abstand $E(\Delta t)$ zwischen den Ereignissen kann mit der folgenden Formel berechnet werden:

$$(35)\quad E(\Delta t) = \frac{1}{\lambda} U_o + \frac{1}{\lambda+\mu} \sum_{i=1}^{ABW-1} U_i + \frac{1}{\mu} U_{ABW}$$

Für den Mittelwert $E(T)$ der Simulationszeit T ergibt sich damit

$$(36)\quad E(T) = S \cdot E(\Delta t)$$

Aus der Tabelle 3 sind die Werte des Proportionalitätsfaktors R der Formel (30) zu entnehmen.
Die Tabelle 4 enthält die Anzahl S der Ereignisse, nach denen der über B_{RK} berechnete Durchsatz D höchstens mit der Wahrscheinlichkeit $P_{GRENZ} = 10^{-2}$ einen größeren statistischen Fehler als $F_{GRENZ} = 10^{-1}$ aufweist. Da bei der Abschätzung des statistischen Fehlers von D eine lineare Näherung benutzt wurde, müssen alle Felder mit $N < 100$ freibleiben, da in diesem Falle Fehler höherer als 1. Ordnung eine Rolle spielen können. In Tabelle 5 sind die entsprechenden Mittelwerte $E(T)$ der Simulationszeit T angegeben; in die Berechnung von $E(T)$ geht neben der Rechenintensität $RI = \lambda/\mu$ noch λ oder μ ein; wir sind von dem Wert $1/\lambda = 50$ ausgegangen, da ein Hintergrundtransport zur Trommel im Mittel bei 50 msec. liegt. Formal betrachtet ist die Zeiteinheit beliebig.

6. Diskussion der Ergebnisse

Zu 5.I.: Mit der mathematischen Behandlung des Rechnermodells ist ein Weg aufgezeigt, wie man den Auftragsdurchsatz für beliebige Rechnermodelle mit der einzigen Einschränkung, daß die Bedienungszeiten exponentiell verteilt sind, berechnen kann. Es ist nur erforderlich, eine geeignete Menge von Zuständen zu finden, deren Verteilung

vollständig ist ($\Sigma p_i = 1$). Zur Demonstration betrachten wir ein Rechnermodell mit 1 Rechnerkern und 2 Transportkanälen und einer Warteschlange für jedes Betriebsmittel. Die Menge der Zustände würde dann folgendermaßen aussehen:

A_i bzw. B_i, bzw. C_i steht für"die RK-Warteschlange bzw. die TK-Warteschlange 1 bzw. die TK-Warteschlange 2 ist mit i-1 Aufträgen besetzt".

Die Menge der Zustände besteht dann aus den Elementen

$E_l = i.j.k = A_i$ und B_j und C_k; i, j, k = 1,..., ABW+1

Bei der Aufstellung der Übergangsmatrix kann wie bei dem einfacheren Modell dieser Untersuchung vorgegangen werden; Verteilung der Bedienungszeiten und die Zugriffswahrscheinlichkeiten liefern die Übergangswahrscheinlichkeiten P_{ij} vom Zustand E_i in die Zustände E_j.

Zu 5.III.: Bei der Durchsatzbestimmung durch Simulation nach (9) sind bei EA-intensiven Jobmixen erhebliche Simulationszeiten erforderlich, wenn man sicher gehen will, daß der Fehler mit der hohen Wahrscheinlichkeit von mindestens o.99 10 % nicht überschreitet. Mit (10) ist eine andere Bestimmung des Durchsatzes möglich.

Wir bezeichnen mit E_o^*, ..., E_{ABW}^* die Zustände der Transportkanalwarteschlange, d.h. E_i liegt vor, wenn die Warteschlange mit i Transportaufträgen besetzt ist.

Ähnlich wie für B_{RK} erhält man

(37) $$B_{TK} = \left(\frac{1}{\lambda+\mu}\sum_{i=1}^{ABW-1} U_i^* + \frac{1}{\mu} U_{ABW}^*\right) \Big/ \left(\frac{1}{\lambda} U_o + \frac{1}{\lambda+\mu}\sum_{i=1}^{ABW-1} U_i^* + \frac{1}{\mu} U_{ABW}^*\right)$$

Wegen $U_i^* = U_{ABW-i}$ folgt

$$(38)\quad B_{TK} = \frac{\frac{1}{\lambda+\mu}\sum_{i=1}^{ABW-1} u_i + \frac{1}{\mu}u_0}{\frac{1}{\lambda}u_{ABW} + \frac{1}{\lambda+\mu}\sum_{i=1}^{ABW-1} u_i + \frac{1}{\mu}u_0}$$

(38) unterscheidet sich von dem entsprechenden Ausdruck B_{RK} nur dadurch, daß u und vertauscht sind. Daraus folgt

$$(39)\quad B_{RK}(RI) = B_{TK}(1/RI)$$

$$(40)\quad \Delta B_{RK}(RI) = \Delta B_{TK}(1/RI)$$

Berechnet man daher d bei RI = 0.25 mit (10), so wird man denselben statistischen Fehler erhalten wie bei der Berechnung von d mit (9) bei RI = 4.0.
Bei RI = 1 sind die statistischen Fehler von (9) und (10) gleich groß. Wenn als Zeiteinheit 1 msec. zugrunde gelegt wird, ergibt sich unter den oben genannten Voraussetzungen ungefähr eine Simulationszeit von zwei Stunden. Dies stellt den ungünstigsten Fall dar, wenn d bei RI $\leq$ 1 mit (10), bei RI$>$1 mit (9) berechnet wird.

Tabelle 1

RI	D / D_o bei 2 Programmplätzen	3	4	5	6	7
0.2	1.16	1.19	1.20	1.20	1.20	1.20
0.4	1.26	1.34	1.38	1.39	1.40	1.40
0.6	1.31	1.44	1.51	1.55	1.57	1.58
0.8	1.33	1.49	1.58	1.64	1.68	1.71
1.0	1.33	1.50	1.60	1.67	1.71	1.75
2.0	1.29	1.40	1.45	1.48	1.49	1.49
3.0	1.23	1.30	1.32	1.33	1.33	1.33
4.0	1.19	1.24	1.25	1.25	1.25	1.25

Tabelle 2

ABW \ RI	0.2	0.4	0.6	0.8	1.0	2.0	3.0	4.0
2	1	1	1	1	1	1	1	1
3	7	9	10	11	11	11	10	8
4	11	16	19	22	20	18	16	13
5	12	20	28	35	34	32	23	18
6	16	27	39	42	52	39	27	22
7	18	29	50	64	69	58	34	26

Werte der Anzahl K der Perioden, nach denen die Markovkette "hinreichend" stationär ist.

Tabelle 3

ABW \ RI	0.2	0.4	0.6	0.8	1.0	2.0	3.0	4.0
2	4.1	2.8	1.8	1.2	0.83	0.2	$7.8 \cdot 10^{-2}$	$3.9 \cdot 10^{-2}$
3	3.8	2.5	1.6	1.0	0.66	0.13	$4.3 \cdot 10^{-2}$	$1.9 \cdot 10^{-2}$
4	3.7	2.4	1.5	0.91	0.56	$8.6 \cdot 10^{-2}$	$2.4 \cdot 10^{-2}$	$9.4 \cdot 10^{-3}$
5	3.7	2.4	1.4	0.85	0.50	$5.9 \cdot 10^{-2}$	$1.4 \cdot 10^{-2}$	$4.7 \cdot 10^{-3}$
6	3.7	2.4	1.4	0.81	0.45	$4.1 \cdot 10^{-2}$	$7.9 \cdot 10^{-3}$	$2.3 \cdot 10^{-3}$
7	3.7	2.4	1.4	0.78	0.42	$2.9 \cdot 10^{-2}$	$4.5 \cdot 10^{-3}$	$1.2 \cdot 10^{-3}$

Werte des Proportionalitätsfaktors R in der Formel

$$F_{GRENZ} = R \cdot \frac{1}{\sqrt[7]{P_{GRENZ} \cdot N}}$$

Tabelle 4

ABW \ RI	0.2	0.4	0.6	0.8	1.0	2.0	3.0	4.0
2	$3.4 \cdot 10^5$	$1.6 \cdot 10^5$	$6 \cdot 10^4$	$3 \cdot 10^4$	$1.4 \cdot 10^4$	$8 \cdot 10^2$	$1.2 \cdot 10^2$	$1.5 \cdot 10$
3	$2.0 \cdot 10^6$	$1.1 \cdot 10^6$	$5.1 \cdot 10^5$	$2.2 \cdot 10^5$	$9.1 \cdot 10^4$	$3.7 \cdot 10^3$	$3.7 \cdot 10^2$	$5.8 \cdot 10$
4	$3.0 \cdot 10^6$	$1.8 \cdot 10^6$	$8.6 \cdot 10^5$	$3.6 \cdot 10^5$	$1.3 \cdot 10^5$	$2.7 \cdot 10^3$	$1.8 \cdot 10^2$	----
5	$3.3 \cdot 10^6$	$2.3 \cdot 10^6$	$1.1 \cdot 10^6$	$5.0 \cdot 10^5$	$1.7 \cdot 10^4$	$2.2 \cdot 10^3$	$9.0 \cdot 10$	----
6	$4.4 \cdot 10^6$	$3.1 \cdot 10^6$	$1.5 \cdot 10^6$	$8.4 \cdot 10^5$	$2.1 \cdot 10^4$	$1.3 \cdot 10^3$	----	----
7	$4.1 \cdot 10^6$	$3.3 \cdot 10^6$	$2.0 \cdot 10^6$	$7.8 \cdot 10^5$	$2.4 \cdot 10^5$	$9.8 \cdot 10^2$	----	----

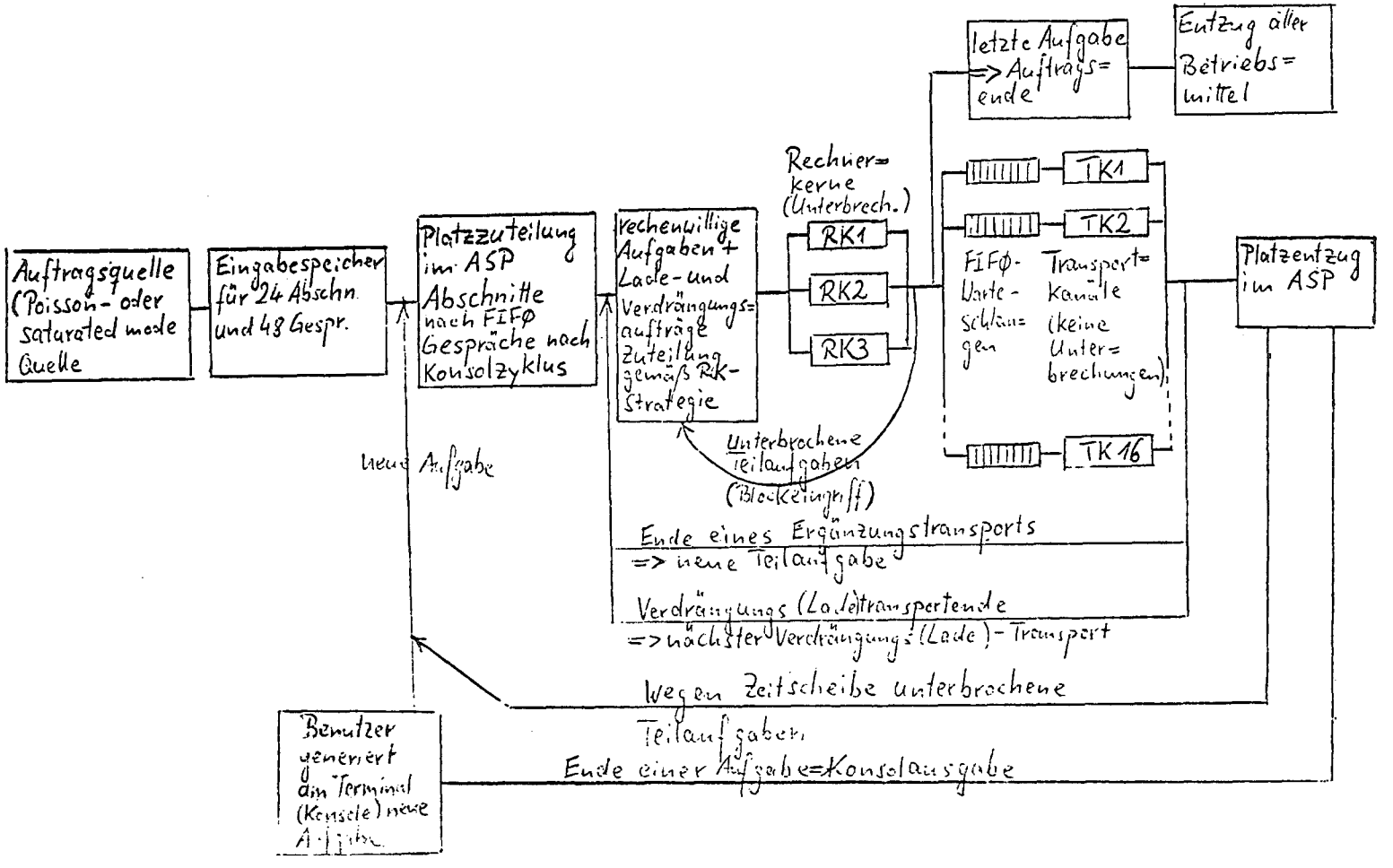
Anzahl S der Ereignisse, bis der statistische Fehler von D höchstens mit der Wahrscheinlichkeit $P_{GRENZ} = 10^{-2}$ größer ist als $F_{GRENZ} = 10^{-1}$.

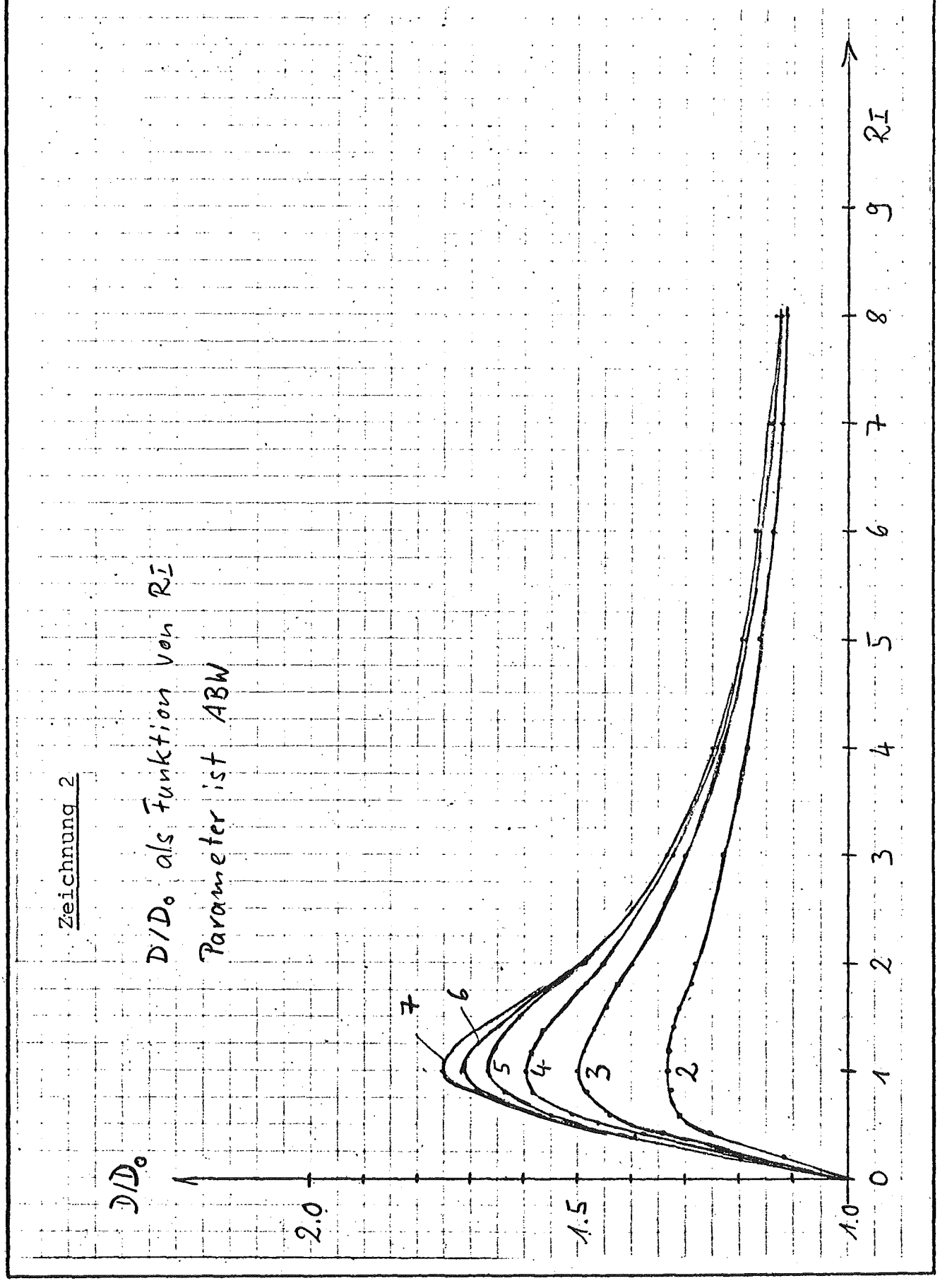
Tabelle 5

ABW \ RI	0.2	0.4	0.6	0.8	1.0	2.0	3.0	4.0
2	$8.7\ 10^{6}$	$4.5\ 10^{6}$	$1.9\ 10^{6}$	$1.0\ 10^{6}$	$5.3\ 10^{5}$	$4.6\ 10^{4}$	$9.7\ 10^{3}$	$1.5\ 10^{3}$
3	$5.0\ 10^{7}$	$2.9\ 10^{7}$	$7.8\ 10^{6}$	$6.6\ 10^{6}$	$3.0\ 10^{6}$	$2.0\ 10^{5}$	$2.9\ 10^{4}$	$5.8\ 10^{3}$
4	$7.5\ 10^{7}$	$4.5\ 10^{7}$	$2.2\ 10^{7}$	$1.0\ 10^{7}$	$4.0\ 10^{6}$	$1.4\ 10^{5}$	$1.4\ 10^{4}$	
5	$8.2\ 10^{7}$	$5.8\ 10^{7}$	$2.9\ 10^{7}$	$1.4\ 10^{7}$	$5.1\ 10^{5}$	$1.1\ 10^{5}$	$6.0\ 10^{3}$	
6	$1.1\ 10^{8}$	$7.8\ 10^{7}$	$3.7\ 10^{7}$	$2.3\ 10^{7}$	$6.0\ 10^{5}$	$6.5\ 10^{4}$	-----	
7	$1.0\ 10^{8}$	$8.3\ 10^{7}$	$5.0\ 10^{7}$	$2.0\ 10^{7}$	$7.0\ 10^{6}$	$4.9\ 10^{3}$	-----	

Mittelwert E(T) der Simulationszeit T, wenn der statistische Fehler von D $F_{GRENZ} = 10^{-1}$ höchstens mit der Wahrscheinlichkeit $P_{GRENZ} = 10^{-2}$ überschreitet.
$1/\lambda = 50$

Zeichnung 1 Verkehrwege des BS3SIM




Zeichnung 2
D/D₀ als Funktion von RI
Parameter ist ABW
D/D₀
2.0
1.5
1.0
0
1
2
3
4
5
6
7
8
9
RI
7
6
5
4
3
2

Erfahrungen mit programmrealisierten Simulationsmodellen und gemessenen Eingangsdaten für Verhaltensanalysen von Informationsverarbeitungssystemen

und

Vorstellungen für den stufenweisen Ausbau von Betriebssystemfunktionen mit Hilfe von Verhaltensanalysen und Simulation von Teilsystemen.

von Erich F. Pantele

Bericht zum Projekt BSM.
Arbeitsgruppe für Betriebssysteme
Rechenzentrum der Technischen Universität München

0. Einführung und Übersicht

In diesem Referat werden zwei Problemstellungen diskutiert, welche im Rahmen des Forschungsprojekts BSM (Betriebssystem München) bearbeitet wurden bzw. bearbeitet werden.

(1.) Ein Erfahrungsbericht über die Untersuchung des TR440 Adreßtransformationsmechanismus mittels Simulation. Dabei werden im einzelnen erläutert:

(1.1.) Die Gründe für die Auswahl des Verfahrens

(1.2.) Probleme mit der Auswahl des Datenmaterials und Experimentdauer

(1.3.) Die Gültigkeitsbetrachtung der Ergebnisse

(2.) Vorbereitungen für die stufenweise Leistungsverbesserung des Betriebssystems BSM, wobei folgende Vorstellungen besonders hervorgehoben werden:

(2.1.) Die Einplanung von Meßeinrichtungen bei der Entwicklung des Betriebssystems

(2.2.) Systematik in der Zerlegung in Teilsysteme

(2.3.) Systematik in der Abwicklung der Verbesserungsphasen.

Am Ende werden die für derartige Arbeiten charakteristischen Probleme noch einmal zusammen bewertet.

1. Ein Erfahrungsbericht über die Untersuchung des TR44o Adreßtransformationsmechanismus mittels Simulation.

Die Ausgangssituation:

Der Telefunken-Rechner TR44o enthält eine Hardware-Einrichtung in Form von 4 elektronischen Assoziativregistern zur Beschleunigung des Arbeitsspeicher-Adressiervorganges. Diese Assoziativregister enthalten in eingeschwungenem Zustand die vier zuletzt benutzten verschiedenen Referenzpaare Seitennummer (des virtuellen Adreßraumes einer programmrealisierten Funktionseinheit) und der Kachelnummer (des realen Arbeitsspeichers). Diese vier S-K-Referenzpaare sind geweils eine Teilmenge der gesamten Abbildungsfunktion: Seitennummern des Adreßraumes→Kachelnummern der realen Arbeitsspeicherlage, die in Form einer Seiten-Kachel-Tabelle vorliegt.
Die Leistung des Rechners (hier: durchschnittliche Befehlsausführungsgeschwindigkeit) hängt relativ stark von der relativen Häufigkeit ab, mit welcher beim Speicheradressiervorgang die benötigte Abbildung Seitennummer→Kachelnummer in den (schnellen) Assoziativregistern gefunden wird (und nicht ein zusätzlicher Arbeitsspeicherzugriff zum Auslesen der S-K-Zuordnung aus der S-K-Tabelle nötig wird).

Das Untersuchungsziel ist die Prüfung der Wirksamkeit der Assoziativregister für die Beschleunigung der Adressiervorgänge und qualitative und quantitative Aussagen über Einflußparameter.(Lit.5)

1.1. Die Gründe für die Auswahl des Simulationsverfahrens.

Von den zwei Alternativen: Direktuntersuchung mit Hardware-Einrichtungen oder Messung und Simulation mit Software-Verfahren wurde das Software-Verfahren aus folgenden Gründen vorgezogen:

- Manipulationen an der Hardware des Rechners zur Anbringung der Meßsonden bringen zu große Sicherheitsrisiken für den Betrieb bzw. zu hohe Realisierungskosten.
- Mit Messungen an der Hardware lassen sich nicht die notwendigen Feststellungen über Kenngrößen des "Auftragsprofils" auf dem Untersuchungsobjekt treffen, bei einer softwaremässigen Gewinnung des Auftragsprofils lassen sich jedoch bequem Kenngrößen und Korrelationen herausfinden.
- Eine Veränderung des Untersuchungsobjektes zur Simulation (z.B. Änderung der Anzahl der Register) läßt sich an der Hardware sehr schwer durchführen - umso bequemer jedoch an einem programmierten Simulationsmodell.

Somit wurde die Simulation mit einem programmrealisierten Simulationsmodell, in welchem die Anzahl der Register veränderbar war, durchgeführt, in dem darauf ein "Auftragsprofil" in Form von Adressreferenzsequenzen gegeben wurde,

welche mit einem Software-Registrierverfahren am realen Objekt gewonnen wurden.
Von diesem Auftragsprofil konnten vor und während der Gewinnung durch Registrieren und während der Simulation charakteristische Kenngrößen, wie

- Programmgröße in Seiten
- dynamsiches Adreßreferenzverhalten des Programmlaufes in seinem Adreßraum (Lokalität)
- Häufigkeit der Einschwingvorgänge wegen Ungültigsetzung der Register-Inhalte bei Wechsel des Adressierungsmodus bzw. Regiewechsel zu Adreßräumen anderer Funktionseinheiten im Multiprogramming.

festgestellt werden.

1.2. Probleme mit der Auswahl des Datenmaterials und mit der Experimentdauer.

Die Erzeugung eines repräsentativen Auftragsprofils für das Simulationsmodell war die schwierigste Aufgabe.
Am realen Objekt konstituiert sich das Auftragsprofil auf dem Adressierverfahren aus dem Gesamteinfluß von vielen Maschinenbefehlssequenzen von verschiedenen Compilern generiert, von verschiedenen Benutzerproblemen mit verschiedenen Programmierstils, Algorithmen, Programmgrößen, Datenmaterial etc.

Für die Gewinnung der Adreßreferenzsequenzen wurden daher Programme gerechnet, die wenigstens vom Ansatz her an den realen Verhältnissen des Benutzerbetriebes orientiert waren.

Das Verhältnis von 2,5 Std. Simulations-Rechenzeit pro 1o sec Originalrechenzeit erlaubt nur minimalste Stichprobenauswahl aus dem Rechenbetrieb.

Anm.: 2,5 Std. Simulations-Rechenzeit setzen sich zusammen aus 1 Stunde Registrierzeit für Adreßreferenzsequenzen und 1,5 Std. Rechenzeit des Simulationsprogrammes mit den Daten.

Die Stichprobenauswahl konnte sich daher nur an der Benutzungshäufigkeit von Systemoperatoren, Sprachencompiler für die Benutzerprogramme und dem Speicherbedarfszahlen an der über 3 Wochen beobachteten Realität orientieren.

Da aus Rechenzeitgründen keine Benutzerprogramme mit längerer Rechenzeit (>2osec) berücksichtigt werden konnten, müssen bei der Ergebnisinterpretation entsprechende Einschränkungen gemacht werden!

1.3. Die Gültigkeitsbetrachtung der Ergenisse.

Die Ergebnisinterpretation muß die in 1.2. erwähnten Einschränkungen in der Stichprobenauswahl berücksichtigen.
Im vorliegenden Fall wurden 64 Programmläufe ausgewertet, die zusammen etwa 35 Mio Adreßtransformationen beinhalten. Das entspricht einer Originalrechenzeit von ca. 7o sec.

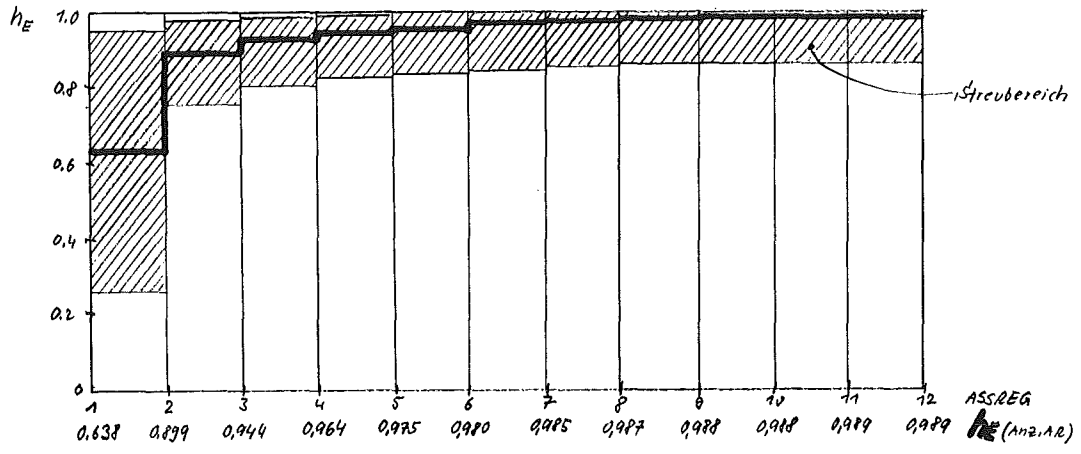
Die systematische Auswahl der Programmläufe ist keine Stichprobenauswahl im Sinne der Statistik und erlaubt daher keine statistische Interpretation der Ergebnisse.

Es werden daher nur Mittelwert, Minimal- und Maximalwert der relativen Häufigkeit des Eintretens des Ereignisses E="Die S-K-Zuordnung wurde in den ASS.Reg.gefunden" in Abhängigkeit von der Anzahl von ASS.Reg. für die ausgewerteten Programmläufe dargestellt.

n Programme, z = Speicherzugriff: $z_1^1, z_2^1, \ldots, z_x^1; \ldots; z_1^n, z_2^n, \ldots z_y^n$.

$$E_j^i \Leftrightarrow z_j^{(i)} \in E_j; \qquad \chi_{E_j^i} = \{1\,|\,0\};$$

$$h_E = \frac{\sum_{i,j} \chi_{E_j^i}}{\sum_{i,j} z};$$



Als Verfahrensbewertung läßt sich feststellen:

1. Das Modell-Programm hat sich sehr gut bewährt, da es neben der dargestellten relativen Häufigkeit für E gleichzeitig noch andere Zusammenhänge wie
   - h_E, abhängig von der Speichergröße des Programmes
   - h_E, abhängig vom Lokalitäts-Charakter des Programmlaufes
   - h_E, abhängig von der Dichte der Adressier-Modus-Unterbrechungen durch Systemaufrufe

   als Ergebnisse lieferte.
   Die große Laufzeit für einen Simulationsschritt ist jedoch ein großer Nachteil.

2. Das Verfahren für die Gewinnung der Simulationsdaten ist sehr nachteilig, da es außer der Gewinnung realer Speicherzugriffssequenzen nicht möglich war
   - nach statistischen Gesichtspunkten Stichproben beliebiger Länge aus beliebigen Programmläufen zu beliebiger Zeit zu nehmen (nur Gesamt-Operatorläufe),
   - und diese Datengewinnung neben dem voll laufenden Rechenbetrieb auf der Maschine durchzuführen (nur im isolierten Betrieb).

2. Vorbereitungen für stufenweise Leistungsverbesserung des Betriebssystemes BSM.

Die Ausgangssituation:

BSM ist ein Betriebssystem, welches aus einem hardwarenahen Basissystem und einer Menge von selbständigen Funktionseinheiten (Bearbeitern) besteht, durch die die Systemdienste erbracht werden. Diese Funktionseinheiten können auf der Zweiprozessoranlage TR44o asynchron parallel arbeiten. In der Menge der Funktionseinheiten ist eine Schichtenstruktur definiert, so daß eine gewisse Hierarchie in der Verfügbarkeit über Systemdienstleistungen besteht. In der obersten Hierarchiestufe sind die Betriebsführung und die Benutzeraufträge eingeordnet. (Lit.1,2).

In diesem Betriebssystem, welches sich im Implementierungsstadium befindet, sind an verschiedenen Stellen Algorithmen für Arbeitsstrategien enthalten. (z.B. Zugriffsalgorithmen zu rotierenden Speichern, Seitenaustauschalgorithmus für demand paging, Betriebsmittelbelegungs- und Betriebsplanungsalgorithmen). Solche Arbeitsstrategien sind im ersten Ansatz i.A. relativ einfach ausgeführt.

Arbeitsziel ist es, mit Hilfe von charakteristischen Auftragslasten (Benchmark-Programmen) eine Auftragsumgebung, deren Kenngrößen bekannt sein sollen, für das System bzw. Teilsysteme zu simulieren und das Systemverhalten zu beobachten. Anhand dieser Beobachtungen sind die Arbeitsalgorithmen des Systems im Sinne der Betriebspolitik zu verbessern.

2.1. Die Einplanung von Meßeinrichtungen bei der Entwicklung des Betriebssystemes.

Ein leistungsfähiges Meßsystem als integrierter Bestandteil des Betriebssystem ist eine notwendige Voraussetzung für bequemes Gewinnen von Information über das Verhalten des Systemes.
Dafür sind in BSM ein sog. Zentralprotokollvermittler (als zentrale Sammelstelle für Informationsablagen) und verschiedene Möglichkeiten zur bequemen Informationsabgabe, wie

- Ablaufregistrierung,
- Dienst für Direktablage von Kurzinformation,
- Dienst für Dumps,
- Ein- und Ausschaltmöglichkeit für Test- und Meßroutinen, die Bestandteile von Systemteilen sind,

vorgesehen.
Diese Einrichtungen gestatten es, Informationen über zu beobachtende Ereignisse in der Reihenfolge ihres Eintreffens zu registrieren und auf externe Datenträger auszuschleusen. Die sequentiell gespeicherte Menge von selbstbeschreibenden Informationselementen kann von einem Auswertungsprogramm in verschiedenen, einstellbaren Versionen ausgewertet werden. (Lit.6)

2.2. Systematik in der Zerlegung in Teilsysteme.

Für die Untersuchung funktioneller Zusammenhänge in dem Betriebssystem wird das System zweckmässigerweise in Teilsystemen betrachtet. Die Schichtengrenzen im System bieten geeignete Abgrenzungen für die jeweils darunterliegenden "Pseudomaschinen" $\mathfrak{M}$ als Teilsysteme. Für jede $\mathfrak{M}^i$ ist die darueberliegende Pseudomaschine $\mathfrak{M}^{i+1}$ nur in Form eines charakteristischen Auftragsprofiles darzustellen.

Die Pseudomaschine $\mathfrak{M}^i$ selbst ist im Sinne definierter Aufgabenstellungen unter Anwendung der für sie charakteristischen Auftragslast zu beobachten und ggf. zu verbessern.
Neben der Schichtenteilung des Systemes ist auch eine laterale Teilung innerhalb einer $\mathfrak{M}^i$ sinnvoll, wenn nur ein Ausschnitt des Dienstleistungsangebotes der Verbesserung unterzogen werden soll, und dabei Teile von $\mathfrak{M}^i$ nicht betroffen sind.

2.3. Systematik in der Abwicklung der Verbesserungsphasen.

Für die Durchführung eines Verbesserungsschrittes ist die Kenntnis

A - der System-Parameter, für die Verhaltensregulierung des (Teil-)Systemes,

B - des Soll-Verhaltens des (Teil-)Systemes,

C - der charakteristischen Parameter der Auftragslast,

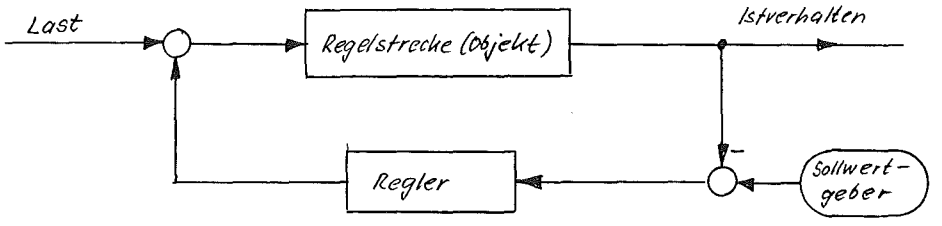
D - des Ist-Verhaltens des (Teil-)Systemes,

Voraussetzung.

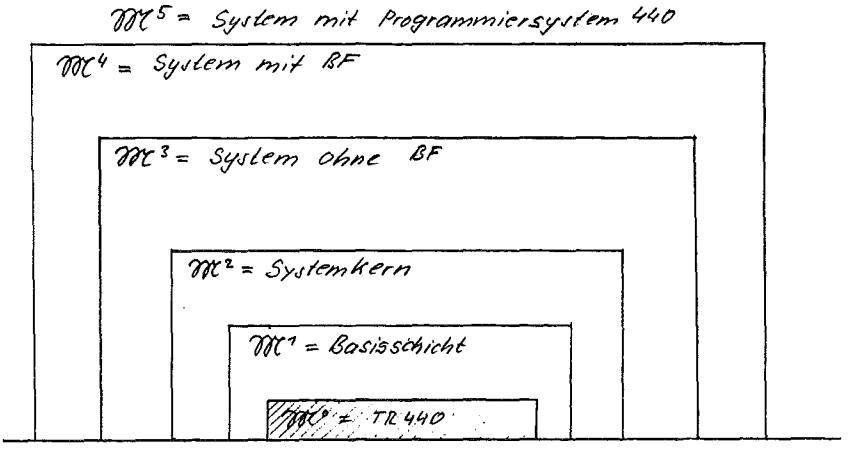
Für eine Verbesserung des Objektes $\mathfrak{M}$ gilt folgendes:

1. Die Systemparameter von $\mathfrak{M}$ werden mit ihrem qualitativen und quantitativen Einfluß auf das Objektverhalten durch Simulation mit speziellen Auftragslasten bekannter Charakteristik festgestellt.
2. Das Soll-Verhalten des Objektes muß als Arbeitsziel definiert sein.
3. Es muß eine hinreichend allgemeine Auftragslast für das Objekt (durch Messung) gewonnen werden.
4. Mit der allgemeinen Auftragslast wird durch Messung das Ist-Verhalten des Objektes festgestellt und
5. aus dem Vergleich von Ist- und Sollwert muß die Konsequenz für die Einstellung der Systemparameter bzw. Änderung des Objektes gezogen werden.

Anm.: P.4.und 5. stellen eine einfache Regelung dar



Das Betriebssystem, das nach der Fertigstellung des ersten Prototyps zu verbessern sein wird, kann aufgrund der Schriftstruktur in eine Folge von geschachtelten "Pseudomaschinen" $\mathcal{M}^i$ zerlegt werden,
z.B.:



Für jede der $\mathcal{M}^i$ können die obigen Punkte 1 mit 5 für eine Verbesserung angewandt werden.
Die Verbesserung des Gesamtsystems geschieht von "innen nach außen" im Sinne einer Kaskadenregelung der Regelungstheorie..
Dabei ist zu beachten, daß bei einer Änderung von $\mathcal{M}^i$ alle $\mathcal{M}^1$ bis $\mathcal{M}^{i-1}$ nachverbessert werden müssen, da sich bei der systemeigenen Auftragsrichtung ihre Auftragslast und damit ihr Verhalten geändert haben könnte.

3. Zusammenfassung

Am Beispiel der ersten Untersuchung sollte gezeigt werden, welche Probleme auftauchen durch Fehlen geeigneter Meßeinrichtung zur Datengewinnung für Simulationen und sonstige Beobachtungen.

Am zweiten Beispiel sollte gezeigt werden, daß eine geeignete Systemstruktur und die frühzeitige Einplanung von Meßeinrichtungen die Verhaltensanalysen,Simulati onen und Verbesserungen an sehr komplexen Objekten erleichtern können.

Literatur-Referenzen:

(1) G.Goos, J.Jürgens, K.Lagally: The Operating System BSM, viewed as a Community of Parallel Processes; März 72, Bericht 72o8, MI-TU-München.

(2) K.Lagally: Aufruf von Systemleistungen in einem schichtenweise gegliederten Betriebssystem. 2.GI-Jahrestagung, Okt.72, Karlsruhe.

(3) G.Mersmann: Planung von Meßverfahren in einem Teilnehmerrechensystem. NTG/GI-Fachtagung "Rechner- und Betriebssystem: Analyse, Simulation und Entwurf", April 72, Darmstadt.

(4) G.A.Mihram: Simulation, statistical Foundations and Methodology; Academic Press 1972.

(5) E.F.Pantele: Ein Meßverfahren und einige Messungen über das dynamsiche Verhalten von Programmläufen im Arbeitsspeicher einer digitalen Rechenanlage. NTG/GI-Fachtagung "Rechner- und Betriebssystem: Analyse, Simulation und Entwurf", April 72, Darmstadt.

(6) E.F.Pantele: Einplanung eines leistungsfähigen Software-Meßsystems bei der Entwicklung eines Betriebssystems. 2.GI-Jahrestagung, Okt.72, Karlsruhe.

Ergebnis der Teilnehmerumfrage zum Simulationsworkshop

Die Teilnehmerumfrage hatte den Sinn, einen Überblick über die Zusammensetzung der Gesprächsrunde und insbesondere Auskunft über die Arbeitsumgebung und die bisherigen Erfahrungen der Teilnehmer mit der Simulation bzw. mit Simulationshilfsmitteln zu geben.

Die hier nun vorgelegten Umfrageergebnisse können natürlich lediglich für den Simulationsworkshop repräsentativ sein, und auch dies nur in beschränktem Maße, da ein Teil der Fragebogen überhaupt nicht oder nur teilweise ausgefüllt wurde (s.u.).

Außerdem ist zu berücksichtigen, daß beim Workshop z.T. ganze Arbeitsgruppen vertreten waren, was sich in mehreren identisch beantworteten Fragebogen niederschlug und zu einer Verzerrung der Ergebnisse führte (vor allem Punkt 4).

Trotzdem glauben wir, daß die Umfrageergebnisse für die Teilnehmer des Workshops eine gewisse Übersicht bieten können, und diesen deshalb nicht vorenthalten werden sollten.

1.

Anzahl der Teilnehmer: 45
abgegebene Fragebogen: 34

2.

Übersicht über den Teilnehmerkreis

Arbeiten mit diskreten Systemen : 25
Arbeiten mit kontinuierlichen Systemen: 13

Tätigkeitsbereich der Teilnehmer

Hochschule : 13
Industrie : 13 (davon 9 Rechnerhersteller)
Forschungszentrum: 8
Sonstige : 4

3.

Aufteilung der Teilnehmer nach Erfahrungszeitraum

Jahre / Art	0	1	1,5	2	2,5	3	4	4,5	5	6-12
kontin.	4	0	0	2	0	3	0	0	2	2
diskret	2	5	2	5	1	3	2	1	4	0

Gesamt:

kontinuierlich: 13
diskret : 25

Sprache \ Jahre	keine Angaben	0,5	1	1,5	2	2,5	3	3,5	4	4,5	5	6-12	gesamt
ALGOL									2	1	2		5
ANAGOL			1		1								2
APL \ 360				1									1
ASIM	1	1	1										2
ASSEMBLER	1						1						2
CSMP							1				1		2
DYNAMO		1	2										3
FORTRAN	4		2				1			1	1	2	11
GPSS	2	2	2		2	1					1		11
SIMSCRIPT I.5					1		1		3		1		6
SIMSCRIPT II		1											1
SIMULA I						2							2
SIMULA 67	3	2	3										8

Die Tabelle ist nicht vollständig. Weitere von einzelen Teilnehmern angegebene Sprachen waren

BASIC, DYSYS, GASP, PL/1

sowie Eigenentwicklungen wie z.B.

SAMO.

Als Auswahlkriterium wurde angegeben:

Maschinenkonfiguration und Compiler: 15 mal

(geplante) Anwendung : 17 mal

4.

Übersicht über die den Teilnehmern zur Verfügung stehenden Anlagen:

TR 44o	15	IBM/37o-195	2
TR 4	1o	Siemens 4oo4/151	3
TR 86	1	Siemens 3o6	5
CDC 66oo	12	Electrologica X 8	4
CDC 65oo	1	Univac 11o8	3
IBM/36o-4o	1	PDP-1o	1
IBM/36o-5o	2	EAI 68o	1
IBM/36o-65	6	P 14oo	2
IBM/36o-67	1	HRS 86o	2
IBM/36o-85	1	HSI SS 1oo	2
IBM/37o-155	1	SDS/Beckmann EASE	1
IBM/37o-165	8		

5.

Anlaß für die Arbeiten auf dem Gebiet der Simulation

1. Persönliche Initiative:	6
2. laufendes Projekt :	17
3. sowohl 1. als auch 2. :	1o
4. keine Angaben :	1