Fachberichte Simulation

Herausgegeben von D. Möller und B. Schmidt
Band 12

K. H. Fasol,
K. Diekmann (Hrsg.)

Simulation in der Regelungstechnik

Springer-Verlag
Berlin Heidelberg New York London
Paris Tokyo Hong Kong Barcelona 1990

Rechnerunterstützter Entwurf von Regelungssystemen - - Verfahren und Werkzeuge [+)]

F.E. Cellier, C.M. Rimvall

1 Einführung

In nahezu allen Bereichen wissenschaftlicher und ingenieurwissenschaftlicher Entwicklungen ist das Hilfsmittel der Simulation im Laufe der Zeit von größter Bedeutung und meist sogar unverzichtbar geworden. Korn und Wait (1978) bezeichneten die Simulation treffend als das "Experimentieren mit Modellen". Unter Berücksichtigung dieser Definition besteht also jedes Simulationsprogramm eigentlich aus zwei Teilen: Einerseits aus einer entsprechend kodierten Beschreibung des Modells, die innerhalb des Simulationsprogramms als Modellrepräsentation bezeichnet wird. Andererseits besteht das Programm auch aus einer entsprechend kodierten Beschreibung der mit dem Modell durchzuführenden Experimente, was als Experimentrepräsentation bezeichnet werden kann.

Wenn man die von vielen Autoren beschriebenen Beispiele für Simulationsstudien betrachtet, gleichgültig ob im zeitdiskreten oder kontinuierlichen Bereich, dann stellt man fest, daß diese Darstellungen vorwiegend aus einer sehr eingehenden Beschreibung eines detailliert ausgearbeiteten Modells bestehen, an dem allerdings relativ einfache Experimente durchgeführt werden. Auch die Fallstudien im letzten Teil dieses Buches folgen im wesentlichen diesem Aufbau. Das Experimentieren besteht immer darin, daß, ausgehend von einem vollständigen und konsistenten Satz von Anfangsbedingungen, die zeitlichen Verläufe einzelner Systemgrößen registriert werden. Dieses Vorgehen wird als Ermittlung des Übergangsverhaltens oder des trajektoriellen Verhaltens des Modells bezeichnet. Man kann den Begriff "Simulation" ja tatsächlich recht einfach der Ermittlung des Übergangsverhaltens gleichsetzen, was auch sehr oft getan wird. Dies ist aber nur solange möglich, als eine reine Lösungsfindung gemeint ist und weniger die Gesamtheit aller modellbezogenen Vorgänge. In der Tat sind auch die meisten der gängigen Simulationspakete kaum mehr als wirkungsvolle Werkzeuge zur Berechnung des Übergangsverhaltens.

[+)] Von K.H.Fasol überarbeitete, stark gekürzte und übersetzte Fassung von Cellier, F.E. und Rimvall, C.M. (1987). Computer-Aided Control Systems - Techniques and Tools, in: Systems Modeling and Computer Simulation (N.A.Kheir, ed.). Marcel Dekker, Basel.

Leider stellen sich nur sehr wenige praktische Probleme als reine Simulationsaufgaben dar. So sind z.B. die Anfangswerte sehr häufig nicht alle zum selben Zeitpunkt definiert. Solche Probleme werden allgemein als Randwertprobleme bezeichnet im Gegensatz zu den vorher erwähnten Anfangswertproblemen. Randwertprobleme sind im Sinne des Wortes natürlich keine Simulationsprobleme, obwohl sie durch die Methode des invariant embedding in Anfangswertprobleme umgeformt werden können. Allgemein wird aber für diese Problemklasse eher die sog. shooting technique angewandt bei der man so vorgeht:

1. Annahme eines Satzes von Anfangswerten.
2. Durchführung einer Simulation.
3. Berechnung eines Gütekriteriums wie z.B. der gewichteten Summe der Fehlerquadrate zwischen erwarteten und berechneten Randwerten.
4. Die Prozedur kann abgebrochen werden, sollte sich der Wert des Gütekriteriums als genügend klein ergeben. Anderenfalls werden die unbekannten Anfangswerte als Parameter interpretiert, und das nichtlineare Problem wird durch iterative Parameteroptimierung gelöst, so daß das Gütekriterium minimiert wird.

Man sieht also, daß dieses "Experiment" des "shooting" u.U. eine große Anzahl einzelner Simulationsläufe beinhaltet.

Nehmen wir nun an, um ein anders geartetes Beispiel zu besprechen, die Dynamik eines elektrischen Netzes soll simuliert werden. Die einzelnen elektrischen Komponenten dieses Netzes hätten verschiedene Toleranzen und es sollte ermittelt werden, wie sich das Verhalten des Netzes in Abhängigkeit von diesen Toleranzen ändert. Ein Algorithmus für dieses Problem könnte etwa so lauten:

1. Es werden nur jene Komponenten des Netzes betrachtet, die mit Toleranzen behaftet sind und diese werden als Modellparameter interpretiert. Zunächst werden alle diese Parameter mit ihren Minimalwerten angesetzt.
2. Durchführung von Simulationen unter Variation der Parameter zwischen ihren kleinsten und größten Werten solange bis alle "Worst-case"-Parameterkombinationen erfaßt wurden. Die Ergebnisse sämtlicher Simulationen werden gespeichert.
3. Letztlich werden die Einhüllenden aller möglichen Übergangsverläufe aufgrund der gespeicherten Simulationsergebnisse berechnet und graphisch ausgegeben.

Wie im vorhergehenden Beispiel beinhaltet auch dieses Experiment außerordentlich viele unterschiedliche Simulationsläufe. Hier sind dies genau 2^n Simulationen wenn n toleranzbehaftete Komponenten als Parameter vorhanden sind.

Diese beiden Beispiele zeigen, daß die Simulation nicht in einer abgeschlossenen Welt existiert. Eine wissenschaftliche oder technische Studie kann unzählige verschiedene Simulationsläufe und noch vieles andere mehr beinhalten. Leider werden die wenigsten der heute verfügbaren Simulationspakete diesem Bedarf an ausgedehntem Experimentieren gerecht. Wenn auch die Möglichkeiten zur Modellrepräsentation (im Sinne des obigen ersten Absatzes) im Laufe der letzten Jahre ständig verbessert und wirkungsvoller wurden, wurde doch sehr wenig getan, um die softwaremäßigen Möglichkeiten zur Experimentrepräsentation der Simulation auszuweiten (Cellier, 1986). Einige Simulationssprachen, wie z.B. ACSL (Mitchell und Gauthier, 1986) bieten zwar Routinen zur Linearisierung und zur Arbeitspunktbestimmung. Andere, wie etwa DSL/VS (IBM, 1984) bieten eingeschränkte Möglichkeiten zur Frequenzbereichsanalyse wie z.B. die Berechnung des Fourier-Spektrums einer zeitlichen Systemantwort. Uns ist aber kein dzt. verfügbares Simulationssystem bekannt, das ein allgemein anwendbares nichtlineares Programmpaket z.B. für Trajektorienapproximation, Arbeitspunktbestimmung, einen Randwertproblemlöser, usw. als integralen Teil der Software enthält. Ein solches Paket wäre ja auch nur eines von vielen denkbaren nutzbringenden Werkzeugen. Übrigens sind die wenigen verfügbaren, experimentorientierten Softwarewerkzeuge oft auch wenig benutzerfreundlich und sehr spezialisiert, d.h. ihre Anwendbarkeit ist doch eingeschränkt.

Immer wenn wir Softwareingenieure einer solchen Situation begegnen, stellen wir fest, daß mit den Datenstrukturen in der betreffenden Simulationssprache doch etwas nicht in Ordnung sein muß. Tatsächlich führten ja alle Verbesserungen der Modellrepräsentation, wie z.B. die Behandlung von Unstetigkeiten oder die Möglichkeit zur Submodell- (Macro-) Deklaration, zu erweiterten Programmstrukturen, wohingegen die verfügbaren Datenstrukturen immer noch beinahe unverändert sind gegenüber 1967, als die CSSL Spezifikationen (Augustin et al., 1967) zum ersten Mal formuliert wurden.

Wenn wir im Gegensatz zur Simulationssoftware von CAD-Software sprechen, dann denken wir genau an diese erweiterten und verbesserten Möglichkeiten zur Experimentbeschreibung. Die Simulation ist heute einfach nicht mehr der zentrale Teil einer Studie sondern schlicht eines unter den vielen aufrufbaren Softwerkzeugen, die gemeinsam die zum Experimentieren nötige Software bilden. Von nun an wollen wir diese "Computer-aided design software" CAD-Software nennen. Algorithmen für spezielle Anwendungen sollten CAD-Techniken genannt werden und Programme, in denen solche Algorithmen eingebunden werden können, sollen CAD-Werkzeuge (CAD tools) heißen. Da viele der Entwurfswerkzeuge von den jeweiligen Anwendungen abhängen, wollen wir uns hier auf eine spezielle Anwendung konzentrie-

ren. Entsprechend der Zielsetzung dieses Buches, im wesentlichen die Simulation als Werkzeug zum Reglerentwurf darzustellen, wollen wir uns auf den Entwurf von Regelungssystemen festlegen.

Bis vor sehr kurzer Zeit waren die Datenstrukturen innerhalb der Software zum rechnerunterstützten Regelungssystementwurf (Computer-Aided Control System Design, CACSD) ebenso verbesserungswürdig wie jene der reinen Simulationssoftware. Jedoch auch die Programmstrukturen dieser Werkzeuge waren sehr mangelhaft. Der Benutzer wurde durch ein unflexibles Frage-Antwort Protokoll hindurchgeführt. Sobald irrtümlich eine nicht korrekte Spezifikation eingegeben worden war, gab es keine Chance, die Folgen dieses Fehlers zu beseitigen. Die entstandene Abweichung vom entworfenen Weg führte im ungünstigsten Fall zu einem völligen Softwarezusammenbruch, nach dem der Benutzer alle vorher eingetragenen Daten verloren geben mußte und gezwungen war, neu zu beginnen.

Ein wesentlicher Durchbruch konnte mit der Entwicklung von MATLAB (Moler, 1980) erzielt werden. MATLAB ist ein universelles Werkzeug zur Matrix-Manipulation unter einfachster Datenstruktur, die einzige Datenstruktur ist eben jene von Matrizen, und mit einfachster Kodierung. APL bot zwar schon viel früher dieselben Möglichkeiten wie MATLAB, war aber durch eine sehr unzugängliche Syntax gekennzeichnet. Der Benutzer mußte beinahe in derselben Weise denken wie der Rechner das APL-Programm abarbeitete. Bei MATLAB hingegen "denkt" der Rechner ebenso wie der Benutzer. Natürlich war MATLAB als einfache interaktive Sprache für die Matrix-Algebra nicht dazu entwikkelt worden, CACSD Probleme zu lösen, es ist aber offensichtlich jene Art von Werkzeugen, die der Regelungstechniker für die Lösung seiner Probleme braucht. So ist z.B. das Standard Regulator Problem als "Riccati-Entwurf" in wenigen Zeilen eines übersichtlichen Codes formulierbar. Es dauerte daher gar nicht lange, bis etliche CACSD Experten den für sie relevanten Wert dieses zunächst gar nicht für CACSD entworfenen Werkzeugs als Basis für ihre Probleme erkannten. Als Folge entstanden sehr rasch CTRL-C (Systems Control Technology, 1984; Little et al., 1984) MATRIX_x (Integrated Systems, 1984; Shah et al., 1985), IMPACT (Rimvall, 1983; Rimvall und Bomholt, 1985; Cellier und Rimvall, 1989), PC-MATLAB (Little, 1985), MATLAB-SC (Vanbegin und Van Dooren, 1985).

Es wäre einerseits sehr zweckmäßig, wenn eine Matrixnotation, ähnlich wie in MATLAB, innerhalb einer Simulationssprache für die Beschreibung linearer Systeme bzw. Subsysteme verwendet werden könnte. Auch wäre es für Simulationssoftware sehr vorteilhaft, effektivere Integrationsalgorithmen zu verwenden als die konventionellen expliziten Runge-Kutta-, Adams-Bashforth-

oder Gear-Algorithmen. Dies könnte z.B. ein impliziter Adams-Moulton-Algorithmus sein. Lineare (Sub)-systeme könnten durch den Compiler aufgrund einer Matrixnotation sofort automatisch erkannt werden. Andererseits ist es sicher zutreffend, daß die meisten aktuellen CACSD Programme nur eingeschränkte Simulationsmöglichkeiten bieten. Es wäre daher außerordentlich nützlich, das verfügbare Know how über die Simulation dynamischer Systeme und Prozesse in die CACSD-Software einzubringen. Ein Entwurfsvorgang umfaßt natürlich wesentlich mehr als nur Simulationen, er benötigt diese aber unbedingt neben anderen Möglichkeiten. Daher sollte ein flexibles Interface zwischen CACSD-Programm und Simulationssprache vorhanden sein, so daß innerhalb einer komplexen Entwurfsstudie aussagekräftige Simulationsläufe an bestimmten Stufen des Entwurfs diesen wirkungsvoller machen könnten.

Wir befassen uns nachfolgend nur mit digitaler Simulation. Jedoch sind CACSD Algorithmen in unveränderter Weise auch auf die im Beitrag von Troch und Breitenecker besprochene klassische hybride Simulation anwendbar. Der dynamische Prozeß wird als Modellrepräsentation am Analogteil verschaltet wogegen die Experimentrepräsentation am Digitalteil der Hybridanlage programmiert wird. Die eingangs zitierte Definition des Begriffes der Simulation (Experimentieren mit Modellen) wird am klassischen Hybridrechner besonders deutlich.

2 Entwicklung und Klassifizierung von CACSD Methoden

Historisch gesehen entstanden die ersten CACSD Probleme aus den Bedürfnissen des zweiten Weltkriegs als die Militärs nach wirkungsvolleren Waffen fragten. Die Ingenieure ersetzten als Antwort die bislang manuell bedienten durch automatisch operierende, geregelte Waffensysteme. In den Anfängen der Regelungstechnik, in den 30er bis in die 50er Jahre, beschäftigten sich die Ingenieure mit isolierten zeitkontinuierlichen Eingrößensystemen (SISO-Systeme). Der Reglerentwurf fand vorwiegend mit graphischen Frequenzbereichsverfahren statt, die vor allem durch W.R.Evans und H.Nyquist repräsentiert wurden. Als man sich aber immer mehr den Mehrgrößensystemen (MIMO-Systeme) zuwenden mußte, erwiesen sich die klassischen Kennlinien- und Ortskurven-Verfahren als nicht mehr ausreichend, und es war in den 60er Jahren vor allem R.Kalman, der mit der Zustandsraumdarstellung in den Zeitbereich zurückführte. Die für SISO-Systeme auch damals schon vorhandenen Zeitbereichsmethoden konnten, dank der bereits verfügbaren Computer, für MIMO-Systeme von zunächst nicht allzuhoher Ordnung übernommen werden. Was waren aber die wesentlichen Entwicklungen der letzten 15 bis 20 Jahren? Zunächst wurden die ziemlich konso-

lidierten Arbeiten durch verschiedenste, parallel laufende Entwicklungen abgelöst. Für verschiedene Arten von Problemen wurden sehr unterschiedliche, auf die Probleme zugeschnittene Lösungswege eingeschlagen. Einer der wesentlichen Nachteile der früheren Technologien fand sich, ironisch genug, zunächst in der hochgradigen Automatisierung seiner Algorithmen wieder. Eine aufgerufene Subroutine antwortete auf eingegebene Parameterwerte eben mit anderen Parameterwerten. Es fehlte daher an Einsicht, was eigentlich vor sich ging. Häufig ergab es sich, daß eine für die betreffende Aufgabe ungeeignete Regelungsstruktur gewählt worden war und die Parameteroptimierung für diese Struktur daher zum Scheitern verurteilt war. Der Regelungstechniker mußte also Strukturentscheidungen statt lediglich Parameterentscheidungen treffen. Keiner der vor 20 Jahren verfügbaren automatischen Algorithmen war aber zu solchen Strukturentscheidungen fähig.

Aus diesen Gründen gingen nach der anfänglichen Zeitbereichseuphorie viele Wissenschaftler wieder in den Frequenzbereich zurück bzw. kombinierten Zeit- und Frequenzbereichsverfahren und schlugen neue Entwurfswerkzeuge vor wie z.B. das verallgemeinerte Nyquist Diagramm (Rosenbrock, 1969) oder neuartige Systembeschreibungen wie z.B. verschiedene Polynom-Matrizen Darstellungen (Wolowich, 1974; Wonham, 1974). Andere Autoren versuchten Algorithmen zu entwickeln, die als Funktion von Eingangsparametern größere Bereiche von Ausgangsparametern produzierten, dargestellt durch dreidimensionale Graphen im Parameterraum. So etwas wird häufig beim Entwurf sog. robuster Regler getan (Ackermann, 1980), was allerdings meist sehr großen Rechenaufwand erfordert. Ein etwas günstigerer Weg könnte auch in der Anwendung der Empfindlichkeitsanalyse bestehen (Cellier, 1986). Neueste Entwicklungen wenden sich sogar von numerischen Algorithmen wieder völlig ab.

Andere Probleme und Entwicklungen entstanden mit der "optimalen" Regelung großer Systeme bis zur 200. Ordnung. Man versucht, solche Systeme in kleine Subsysteme zu zerlegen und zunächst jedes für sich zu behandeln. Dies führte zur dezentralisierten Regelung (Athens, 1978) und zur hierarchischen Regelung (Siljak und Sundareshan, 1976). Die Verfügbarkeit relativ billiger Mikrorechner ermöglichte in diesem Zusammenhang die dezentralisierte Subsystemregelung. Dies aber wiederum stimulierte die Entwicklung spezieller zeitdiskreter Regelalgorithmen, die Entwicklung von Algorithmen zur Regelung stark nichtlinearer Systeme bzw. zur Regelung von linearen zeitvarianten Systemen wie selbsteinstellende Regler (Åström, 1980), model-reference adaptiv Regler (Parks, 1966; Monopoli, 1974; Narendra, 1980) und robuste Regler (Ackermann, 1980). Alle diese Probleme bzw. Algorithmen erfordern unabhängig von der Systemordnung hohe numerische Stabilität und numerische Genauigkeit,

womit sich immer noch sehr viele Autoren intensiv befassen. Da jedoch in den einzelnen Kapiteln dieses Buches und auch bei Darstellung der Fallstudien die Regelungssysteme von nicht allzuhoher Ordnung sind und andererseits die CACSD- und Simulationswerkzeuge eher in ihren Anwendungsmöglichkeiten exemplarisch beschrieben werden sollen, muß auf alle diese numerischen Probleme hier nicht mehr näher eingegangen werden.

Klassifizierend kann man zusammenfassen, daß es CACSD Methoden für SISO, MIMO und dezentralisierte Systeme gibt; Methoden für Frequenzbereich und Zeitbereich; Methoden für kontinuierliche Systeme im Unterschied zu Methoden für diskrete Systeme; Methoden für lineare und nichtlineare Systeme und schließlich Verfahren für niedrige, hohe und sehr hohe Ordnungen. Man kann auch zwischen benutzerfreundlichen und -unfreundlichen Algorithmen und schließlich zwischen numerischen und nichtnumerischen Algorithmen unterscheiden. Letztere machen u.a. auch von regelbasiertem Vorgehen Gebrauch, worauf im letzten Abschnitt dieses Beitrags noch kurz eingegangen werden soll.

3 Klassifizierung von CACSD Werkzeugen

Entwurfsprobleme können außerordentlich vielseitig sein; ebenso uneinheitlich sind auch die verfügbaren CACSD Werkzeuge und es ist daher wesentlich, das richtige Werkzeug für das jeweilige Problem zu verwenden. Die CACSD Software kann zunächst nach Bibliotheken von Unterprogrammen und speziellen Entwurfs-Programmpaketen unterschieden werden. Die erste Generation der CACSD Tools umfaßte eher den Bibliothekstyp, während neuere Werkzeuge vor allem dem integrierten Pakettyp zuzuordnen sind. Diese neueren Werkzeuge sind entweder umfassende Tools oder sie sind Entwurfs-Shells. Werkzeuge der ersteren Art bemühen sich, für alle denkbaren Fälle passende Algorithmen bereitzustellen. Dies kann schließlich sehr umfangreiche Pakete mit vielen verschiedenen Eigenschaften und Anwendungsmöglichkeiten ergeben; KEDDC (Schmid, 1979, 1985) ist dafür ein Beispiel (siehe den Beitrag von Schmidt und Dastych in diesem Buch). Entwurfs-Shells bieten uneingeschränkte Sätze von Operationen, womit der Benutzer im Rahmen der CACSD Software seine eigenen Algorithmen kodieren kann. MATLAB (Moler, 1980) ist dafür ein Beispiel. Natürlich ist eine Kombination beider Typen möglich und für den Regelungstechniker sehr nützlich.

CACSD Programme sind entweder für den Batch-Betrieb geschrieben, vollständig interaktiv oder beides. Die interaktive Benutzeroberfläche ist für eine rasche Analyse nützlich und erleichtert beträchtlich das Verständnis der Vorgänge inner-

halb der betreffenden Projekte. Allerdings gibt es viele Ent wurfsprobleme, wie z.B. den optimalen Entwurf nichtlineare Systeme, die einen überaus großen Rechenaufwand verursachen Für solche Probleme eignet sich der Batch-Betrieb am besten.

Maschinencode-orientierte Programme (code-driven) sind kompi liert und implementieren ihre Algorithmen und Operationen i Programmcode. Sie sind natürlich schneller aber weniger flexi bel und weniger leicht zu erweitern. CACSD Programme mit In terpretiercode-Orientierung (data-driven) implementieren Algo rithmen und Operationen als Datenanweisungen, die während de Programmbearbeitung interpretiert werden. Sie sind wirkungs voll für Programmentwicklungen aber weniger für die Anwendung Nach Abschluß des Austestens eines neuen Programms kann e daher vom Interpreter-Code in Maschinencode reimplementier werden.

Benutzeroberflächen können vielfältig sein: Orientiert nac Dialogen, Befehlen, Menues, Formularen, Graphik und Window Techniken. Im ersten der genannten Fälle ist der Ablauf de Programms vollständig festgelegt. Dieser Typ der Benutzerober fläche ist am leichtesten zu implementieren, ist jedoch unfle xibel und für Entwicklungen nicht gut geeignet. CACSD Softwar ist auch häufig befehlsgesteuert mit umfangreicher interakti ver HELP-Unterstützung. Die Alternative der Menue-Steuerun hat manchmal die Nachteile des erforderlichen großen Informa tionsaustausches zwischen Programm und Benutzer, der Langsam keit und der Terminalabhängigkeit. Formulargesteuerte Benut zeroberflächen sind sehr nützlich in der Entwicklungsphas aber auch terminalabhängig und daher ebenfalls in der Portabi lität eingeschränkt. Graphik-Benutzeroberflächen dienten an fänglich lediglich dazu, z.B. Bode-Diagramme oder Trajektorie als Simulationsergebnisse graphisch auszugeben. Zunächst wa auch dies in höchstem Grade terminalabhängig, neben den zahl reichen Graphik-Treibern hat sich jedoch inzwischen GKS al Standard etabliert. Sehr aufwendige Graphikdarstellungen, di hohe Datenübertragungsraten erfordern, sind auf Workstation (z.B. APOLLO, SUN) möglich und werden inzwischen zu hohe Vollkommenheit entwickelt. Ebenso wie die graphische Ausgab besteht heute vor allem auch die Möglichkeit der graphische Eingabe, wie z.B. das Entwickeln von Blockschaltbildern a Bildschirm. Die so entworfenen Regelungssysteme werden durc einen Graphikkompiler in kodierte Modellrepräsentation über setzt (u.a.MATRIX$_x$, Integrated Systems Inc.,1984; Shah et.al. 1985). Besonders zu erwähnen ist auch HIBLITZ (Elmquist, 1982 Elmquist und Mattson, 1986) als besonders komfortable graphi sche Eingabemöglichkeiten. Diese beiden Oberflächen sind je doch nur zwei Beispiele für eine sehr große Anzahl der heut verfügbaren Möglichkeiten.

Es verbleibt schließlich noch, die Windowtechnik zu erwähnen, bei der der Bildschirm in verschiedene "Fenster" aufgespalten wird. Jedes Window kann entweder alphanumerische oder graphische Informationen sowie alle der oben erwähnten Kommunikationsmöglichkeiten bieten, wobei sich die einzelnen Windows z.T. auch überdecken können. Im Prinzip werden hier mehrere Bildschirme auf einem realen Bildschirm gleichzeitig dargestellt. Ein solches Windowmanagement verlangt Schirme mit sehr hoher Auflösung.

Die hier eben vorgenommene Klassifikation der CACSD Tools würde nun in konsequenter Weise eine eingehende Einordnung und übersichtliche Besprechung der wichtigsten am Markt verfügbaren Software nach den eben besprochenen Merkmalen verlangen, was jedoch weit über die Aufgabe und Zielsetzung dieses einführenden Beitrags hinausginge. Eine solche detaillierte Übersicht über eine repräsentative Auswahl von CACSD Werkzeugen wird von Cellier und Rimvall (1987) geboten, wo 20 verschiedene CACSD Programmpakete diskutiert und miteinander verglichen werden.

Die vielen, hier angedeuteten unterschiedlichen Typen der Programmorganisationen, der Schnittstellen, des Datenaustauschs, usw., diese so weitgehende augenblickliche Diversifikation im Bereich der CACSD Software läßt hoffen, daß Bemühungen um eine Standardisierung einmal erfolgreich sein werden.

4 Ausblick

Wie wird sich das Gebiet der CACSD Software in nächster Zeit entwickeln? Um in Beantwortung dieser Frage zu verstehen was wir erreichen wollen, müssen wir zunächst feststellen, auf welchem Stand wir uns derzeit befinden. Bis vor nicht allzulanger Zeit sprach man nur über Programmentwicklung; darauf lag der Schwerpunkt, Daten waren weniger wichtig. Zwischen dem Programm als abgespeichertem, unveränderlichen Code und den Daten als ein Teil des Speichers, der seinen Inhalt während der Programmbearbeitung ändert, wurde streng unterschieden.

Mit der neueren Generation von CACSD Tools entfernte man sich von dieser Einstellung sehr rasch. Aktuelle CACSD Programme sind tatsächlich Programmiersprachen für sich, d.h. der Anwender ist nicht länger auf den Rechnerhersteller hinsichtlich der Verfügbarkeit der Sprachen angewiesen, sondern er entwikkelt seine eigenen, speziell anwendungsorientierten Sprachen. Diese Entwicklung besteht einfach darin, daß von der zu lösenden Aufgabe ein immer geringerer Teil in einem Code festgelegt wird, während ein immer größerer Anteil parametrisiert, d.h.

datengesteuert ist. Die Daten selbst erreichten im Laufe de Zeit schließlich einen solchen Grad an Komplexheit, daß dere zweckmäßige Organisation notwendig wurde. Die Schnittstell zum Benutzer, früher als unwichtiges Detail angesehen, wurd zur zentralen Frage dahingehend, ob ein bestimmtes CACSD Werk zeug als gut oder schlecht zu beurteilen ist. Diese Frag wurde sogar wichtiger als jene nach der Reichhaltigkeit a angebotenen Algorithmen. Dies resultierte in einem enorme Zuwachs an Flexibilität der CACSD Tools. Wegen des erhöhte Aufwands für die Benutzeroberfläche mußte diese Flexibilitä jedoch zunächst mit einer erhöhten Rechenzeit bezahlt werder Mit der Verfügbarkeit immer effizienterer Rechner konnte die ses Opfer aber doch leicht erbracht werden. Es traf aber auc oft zu, daß Kompilieren und Linken eines Simulationsprogramm zehnmal so lange dauerte als die eigentliche Ausführung de Programms selbst. Mit der Verfügbarkeit neuer, voll datenge steuerter (direct executing) Simulationssprachen wie z.B SIMNON (Elmqvist, 1975, 1977) oder DESCTOP und DESIRE (Korn 1985, 1987, 1989) kann man Simulationsresultate in kürzeste Zeit erhalten. Auch wenn das eigentliche Simulationsprogram halb so schnell läuft als bei entsprechender Kompilierung entschädigt doch die wesentlich erhöhte Flexibilität (z.B Modelländerbarkeit) mit geringerem Zeitaufwand am Terminal Solche hochflexiblen Simulationswerkzeuge sind genau das, wa als Bestandteil eines CACSD Pakets gebraucht wird.

Man ist jedoch im Augenblick im Begriff, auch noch weiter Schritte zu tun. Gemeint sind die Entwicklung von Multiwir dow-Oberflächen, von objektorientierter Programmierung, vo sprachenerkennenden Editoren, von CAD Datenbanken, usw. Kar man dies alles auf der Ebene von Programmiersprachen in An griff nehmen? Sind das nicht eher Fragen, die auf der Eber eines unterlagerten Betriebssystems zu diskutieren sind? Is denn die Feststellung der Notwendigkeit einer CAD Datenbank um Modelle und Dateien zu speichern, nicht etwa einfach di Feststellung, daß das betreffende Betriebssystem für dies Aufgaben nicht ausreicht? Interaktive Sprachen sind doch ei gentlich selbst spezielle, wenn auch sehr primitive, anwer dungsorientierte Betriebssysteme. Wir sind in der Tat de Ansicht, daß künftige Programmsysteme die jetzt nocht strikt Unterscheidung zwischen Sprachen und dem Betriebssytem, in da sie eingebunden sind, verwischen werden. So z.B. ist jetz bereits die ADA Umgebung grundsätzlich nichts anderes als di teilweise Spezifikation eines Betriebssystems, in das die AD Sprache eingebettet ist. Dies trifft auch für CACSD Werkzeug zu; es könnte hier einiges vom ADA Konzept oder vergleichbare Konzepten übernommen werden. Ein reiches Betätigungsfeld steh hier notwendigerweise offen.

Welche Möglichkeiten könnten nun zukünftige CACSD Tools bieten? Vor allem ist hier zunehmende Flexibilität hinsichtlich der Datenschnittstellen zu erhoffen bzw. zu erwarten. Die höchste Vollendung datengesteuerten Programmierens wäre eine Sprache, in der es im wesentlichen überhaupt keinen Unterschied mehr zwischen Kodierung und Daten gibt. Jede innerhalb der betreffenden Sprache durchführbare Operation wird als eine Eintragung in eine Datenbank formuliert und kann auf diese Weise jederzeit geändert werden. Eine solche Umgebung wird bereits von LISP geboten. Die einzigen primitiven Operationen in LISP bestehen hauptsächlich im Einfügen oder Entfernen von Eintragungen in Listen, und die Operationen sind selbst wiederum als Listeneintragungen erklärt. Die erste Notierung ist der Operator und alle weiteren sind die Parameter. Deshalb weisen allerdings LISP Programme sehr lange Ausführungszeiten auf, wobei ein numerischer Algorithmus um mehrere Größenordnungen langsamer bearbeitet wird als in einer konventionell kompilierten Sprache. Auch ist LISP in der Spezifikation von Algorithmen oft unhandlich. Trotz allem bietet LISP heute in mehrerlei Hinsicht die größtmögliche Flexibilität, so auch im Hinblick auf nichtnumerische Datenverarbeitung und kann deshalb doch mit einiger Aussicht mit konventionellem Programmieren in Konkurrenz treten. Daher werden Anstrengungen unternommen, einige der speziellen Nachteile von LISP zu reduzieren. So können z.B. inkrementale Compiler die Laufzeit um eine Größenordnung vermindern. Ein LISP Interpreter ist außerordentlich einfach und kann in etwa 600 Zeilen kodiert werden. Im Hinblick darauf ist es sinnvoll, diesen Task teilweise als Hardware zu implementieren. Solche LISP Rechner sind verfügbar und können zumindest einen Teil der Ineffizienz von LISP überwinden.

Im Hinblick auf die Benutzerschnittstelle könnten viele der Probleme mit LISP durch eine neuerliche Veränderung des "Weltbildes" vermieden werden, so etwa durch PROLOG, das im Gegensatz zu LISP ausführungsorientiert ist. Hier versetzt sich der Anwender in die Situation der Daten mit der Frage "Was muß mit mir als nächstes geschehen?" Das ausführungsorientierte Programm muß natürlich sequentiell abgearbeitet werden, weshalb PROLOG zunächst allerdings noch etwas ineffizienter als LISP sein wird. Jedoch kann PROLOG ziemlich leicht in LISP implementiert werden. Die Ergebnisse, die mit PROLOG erzielt werden, sind einerseits kürzere und besser lesbare Programme aber andererseits Verlust an Flexibilität. Diese neuen Sprachen werden jedoch in Kürze eine neue Generation von CACSD Werkzeugen hervorrufen, deren Stärke im nichtnumerischen Entwurf liegen wird. Dies bedeutet den Entwurf parametrisierter Regelungssysteme, wobei während des Entwurfsprozesses einige Parameter als Unbekannte behandelt werden. Vermutlich wird der Bearbeiter dadurch mehr Einblick in die Vorgänge innerhalb

seines Systems erhalten. Solche nichtnumerischen CACSD Algo rithmen sind noch in ihren ersten Anfängen; wir wissen noc nicht, wohin diese neuen Konzepte führen werden, und wir wis sen nicht, wie das Problem der Formelexplosion vermieden wer den kann: Wie kann man parametrische Antworten ohne endlos Formulierungen erhalten? Neuere Entwicklungen könnten bei de Beantwortung solcher Fragen helfen (Birdwell et al., 1985).

Auf einem anderen Gebiet könnte die Entwicklung durch dies Konzepte vorangetrieben werden, nämlich durch die Integratic von CACSD Software mit Expertensystemen. Dies sind Programme die Sätze von parametrisierten Regeln anwenden bzw. auswerter in die entsprechende Parameterwerte eingegeben werden. Di verfügbaren Parameterwerte sind das "Wissen" des Systems. Jed Auswertung kann neues Wissen und schließlich neue Regel kreieren, die laufend aktualisiert werden. Expertensystem sollten aber nicht, wie noch häufig angenommen, reine Frage Antwort Programme mit verhältnismäßig eingeschränkten Möglich keiten sein, sondern die Benutzerschnittstelle, durch di neues Wissen in die Wissensbasis des Systems eingegeben wird sollte vom Mechanismus der Regelanwendung und -auswertun vollständig entkoppelt sein. Tatsächlich sind Systeme, auf di diese Beschreibung zutrifft, in Form mancher Sprachen, wi z.B. MATLAB für lineare Algebra, schon viel weiter entwickel als man annehmen könnte. Ist denn nicht auch ein CACSD Too ein "Expertensystem" für den Reglerentwurf? Man wird natür lich solche Werkzeuge nicht als Expertensysteme bezeichner doch bieten sie bereits mehr Expertensystemtechniken an als a ihnen in der Regel genutzt wird.

Wie kann man also das Konzept der Expertensysteme durch CACS Software vorteilhaft ausnutzen? Zunächst sollten die dzt statischen Fehlermeldungen, die Help- und Tutorial-Funktione der CACSD Tools dynamisch gemacht werden. Ansätze dazu wurde schon früher z.B. durch IBM geboten; IMPACT geht in dies Richtung, und eine andere Möglichkeit wurde von Munro et a (1986) beschrieben; siehe auch Taylor und Frederick (1984) Cellier und Rimvall (1987) erwähnen eine von Åström und Ljur eben in Ausarbeitung begriffene Funktion, die nicht erst ein getretene Fehler meldet, sondern die schon unmittelbar be fehlerhaften Eingaben vor deren negativen Auswirkungen warnt Eine ähnliche Funktion könnte in einen sprachenerkennende Editor einbezogen werden. So könnte ein CACSD Programm sowol auf syntaktische als auch auf semantische Korrektheit schon i einem frühen Stadium überprüft werden.

Eine der vielen anderen Möglichkeiten wäre auch die Bild schirmanzeige der Daten zur selbständigen Anwahl der jeweil sinnvollsten Algorithmen. So könnten z.B. die strukturelle Eigenschaften einer Matrix festgestellt und dafür, als Ergeb

nis dieses Tests, z.B. der günstigste Invertierungsalgorithmus ausgewählt werden. Sehr viel verfügbares Wissen in verschiedener Richtung wird derzeit von vielen CACSD Werkzeugen noch nicht ausgenutzt. Hier liegt ein Anreiz.

Schließlich kann man erwarten, daß aus der Technologie der Expertensysteme sogar neue Regelungsalgorithmen resultieren. Die heute verfügbaren Algorithmen sind für die dezentrale Regelung von Subsystemen hervorragend ausgereift. Sie sind jedoch nicht so sehr geeignet für die globale Steuerung und vor allem die Überwachung komplexer Systeme wie z.B. eines Kernkraftwerks. Eine globale Strategie muß die Prozeßdiagnose und die Entscheidung über notwendige Schritte übernehmen. Dies wird noch vorwiegend von Operateuren beinahe besser vorgenommen, die aber in ihren Köpfen keine Riccatigleichungen lösen können. Sie treffen ihre Entscheidungen aufgrund von qualitativen, d.h. hochgradig diskretisierten Informationen, die mittels eines gedanklichen Prozeßmodells erarbeitet werden, also durch Prozeßsimulation. Diese Möglichkeiten qualitativer Simulation und darauf beruhendem regelbasierten Regelungssystementwurf wurde von Cellier und Zeigler (1987), Vesanterä und Cellier (1989) untersucht.

Zusammenfassend und abschließend kann festgestellt werden, daß CACSD unverändert ein aktives Forschungs- und Entwicklungsgebiet darstellt, aus dem noch viele Ergebnisse erhofft bzw. erwartet werden. Die Simulation innerhalb von CACSD Systemen ist ein unabdingbares, weiterhin verbesserungswürdiges Werkzeug zum Reglerentwurf. Dieser einführende Beitrag und die nachfolgenden exemplarisch besprochenen Werkzeuge könnten die eine oder andere Anregung vermitteln.

Literaturhinweise

[1] Ackermann, J. (1980). Parameter space design of robust control systems. IEEE Trans. Automatic Control AC-25, pp.1058-1072.

[2] Agathoklis, P., Cellier, F.E., et al. (1979). INTOPS, educational aspects of using computer-aided design in automatic control, in: Proceedings of the IFAC Symposium on Computer-Aided Design. Zürich, Switzerland, August 29-31, 1979 (M.A. Cuenod, ed.), Pergamon Press, Oxford, pp. 441-446.

[3] Åström, K.J. (1980). Self-tuning regulators-design principles and applications, in Applications of Adaptive Control (K.S. Narendra and R.V. Monopoli, eds.), Academic Press, New York.

[4] Åström, K.J. (1985). Computer-aided tools for control system design, in: Computer-Aided Control Systems Engineering (M.Jamshidi, C.J.Herget, eds.), North-Holland Publishing, Amsterdam, pp. 3-40.

[5] Athens, M., ed. (1978). On large scale systems and decentralized control, IEEE Trans. Automatic Control, AC-23.

[6] Augustin, D.C., et al. (1967). The SCi continous systems simulation language (CSSL), Simulation, 9, pp.281-303.

[7] Birdwell, J.D. et al. (1985). Expert systems techniques and future trends in a computer-based control system analysis and design environment, in: Proceedings of the 3rd IFAC Symposium on Computer-Aided Design in Control and Engineering Systems (CADCE'85), Kopenhagen, 1985, Pergamon Press, Oxford, pp. 1-8.

[8] Cellier, F.E. (1986). Enhanced run-time experiments in continous system simulation languages, in: Proceedings of the 1986 SCSC Multiconference (F.E. Cellier, ed.), SCS Publishing, San Diego, CA, pp. 78-83.

[9] Cellier, F.E., Rimvall, C.M. (1983). Computer-aided control systems design, Invited Survey Paper, in: Proceedings of the Winter Simulation Conference (ESC'83), Aachen, FRG (W. Ameling, ed.), Springer-Verlag, Lecture Notes in Informatics, New York, pp. 1-21.

[10] Cellier, F.E., Rimvall, C.M. (1987). Computer-Aided Control Systems. Techniques and Tools, in: Systems Modelling and Computer Simulation (N.A. Kheir, ed.). Marcel Dekker, Basel, pp. 631-679.

[11] Cellier, F.E., Zeigler, B.P.(1987). AI's Role in Control of Systems: structural and behavioural knowledge, Proc. ESM'87, Europäische Simulations Multikonferenz, Wien.

[12] Cellier, F.E., Rimvall, C.M. (1989). Matrix encironments for continous system modeling and simulation, Simulation, 52:4, pp.141-149.

[13] Elmquist, H. (1975). SIMNON - An Interactive Simulation Program for Nonlinear Systems-User's Manual, Report CODEN: LUTFD2/(TFRT-7502). Dept. of Automatic Control, Lund Institute of Technology, Lund, Sweden.

[14] Elmquist, H. (1977). SIMNON-An interactive simulation language for nonlinear systems, in: Proccedings International Symposium SIMULATION '77, Montreux (M. Hamza, ed.), Acta Press, Anaheim, CA, pp.85-90.

[15] Elmquist, H. (1982). A graphical approach to documentation and implementation of control systems, in: Proceedings of the 3rd IFAC/IFIP Symposium on Software for Computer Control (SOCOCO '82), Madrid. Pergamon Press.

[16] Elmquist, H. Mattson, S.E. (1986). A Simulator for dynamic systems using graphics and equations for modelling, in: Proceedings of the 3rd Symposium on Computer-Aided Control System Design, Washington , DC.

[17] IBM (1984). Dynamic Simulation Language/VS (DSL/VS). Language Reference Manual, Program Number 5798-PXJ, Form SH20-6288-0, IBM Corp., Cottle Road, San Jose, CA.

[18] Integrated Systems, Inc. (1984). $Matrix_X$ Users's Guide, $Matrix_X$ Reference Guide, $Matrix_X$ Training Guide, Command Summary, and on-line Help. Integrated Systems, Inc., Palo Alto, CA.

[19] Kheir, N.A., ed. (1987). Systems Modeling and Computer Simulation. Marcel Dekker, Basel.

[20] Korn, G.A. (1985). A new interactive environment for computer-aided experiments, Simulation 45(6), 303-305.

[21] Korn, G.A. (1987). Control-System simulation on small personal-computer workstations, Int. J. Modeling and Simulation 8(4).

[22] Korn, S.A. (1989). Interactive Dynamic System Simulation. McGraw Hill, New York

[23] Korn, G.A., Wait, J.V. (1978). Digital Continous System Simulation, Prentice-Hall, Englewood Cliffs, NJ. 212pp.

[24] Little, J.N. (1985). PC-MATLAB, User's Guide. Reference Guide and On-line HELP, BROWSE, and Demonstrations. The MathWorks, Inc., Sherborn, MA.

[25] Little, J.N. et al. (1984). CTRL-C and matrix environment for the computer-aided design of control systems, in: Proceedings of the 6th International Conference on Analysis and Optimization (INRIA), Nice, France, Lecture Notes in Control and Information Sciences, Vol. 63, Springer-Verlag, New York.

[26] Mitchell, E.E.L, Gauthier, J.S. (1986). ACSL Advanced Continous Simulation Language-User/Guide Reference Manual, Mitchell & Gauthier, Assoc., Concord, MA.

[27] Moler, C. (1980). MATLAB User's Guide, Dept. of Computer Science, Univ. of New Mexico, Albuquerque, NM. 40 pp.

[28] Monopoli, R.V. (1974). Model reference adaptive control with an augmented error signal. IEEE Trans. Automatic Control AC-19, 474-484.

[29] Munro, N., Palaskas, Z., Frederick, D.K. (1986). An adaptive CACSD dialogue facility, in: Proceedings of the 3rd Symposium on Computer-Aided Control System Design, Washington D.C.

[30] Narendra, K.S. (1980). Recent developments in adaptiv control, in: Methods and Applications in Adaptive Contro (H.Unbehauen, ed.). Springer-Verlag, New York.

[31] Parks, P. (1966). Ljapunow redesign of model referenc adaptive control systems. IEEE Tans. Autom. Control. 1 pp. 362-368.

[32] Rimvall, C.M. (1983). IMPACT, Interactive Mathematica Program for Automatic Control Theory, User's Guide, Dept of Automatic Control, Swiss Federal Institute of Techno logy, ETH-Zentrum, Zürich, Switzerland, 208 pp.

[33] Rimvall, C.M., Bomholt, L. (1985). A flexible man-machin interface for CACSD applications, in: Proceedings of th 3rd IFAC Symposium on Computer-Aided Design in Contro and Engineering Systems (CADCE'85), Copenhagen, 1985 Pergamon Press, Oxford, pp. 98-103.

[34] Rosenbrock, H.H. (1969). Design of Multivariable contro systems using the inverse Nyquist array, Proc. IEE 116 pp.1929-1936.

[35] Schmid, Ch. (1979). KEDDC, User's Manual and Programmer' Manual. Lehrstuhl für Elektrische Steuerung und Regelung Ruhr-Universität Bochum.

[36] Schmid, Ch. (1985). KEDDC - A computer-aided analysis an design package for control systems, in: Computer-Aide Control Systems Engineering (M. Jamshidi, Herget, C.J. eds.), North-Holland, Amsterdam, pp. 159-180.

[37] Shah, S., et al. (1985). Matrix$_x$ Control Design and Mode Building CAE Capability, in: Computer-Aided Control Sy stems Engineering, (M. Jamshidi, Herget, C.J., eds.) North-Holland, Amsterdam, pp. 181-207.

[38] Siljak, D.D., Sundareshan, M.K. (1976). A multileve optimization of large-scale dynamic systems, IEEE Trans Automatic Control AC-21, pp.70-84.

[39] Systems Control Technology (1984). CTRL-C, A Language fo the Computer-Aided Design of Multivariable Control Sy stems, User's Guide. Systems Control Technology, Pal Alto, CA.

[40] Taylor, J.H., Frederick, D.K. (1984). An expert syste architecture for computer-aided control engineering Proc. IEEE 72(12), pp.1795-1805.

[41] Vanbegin, M, Van Dooren, P. (1985). MATLAB-SC, App. B Numerical Subroutines for Systems and Control Problems Technical Note N168, Philips Research Laboratories, Bos voorde, Belgium, 40 pp.

[42] Van den Bosch, P.P.J. (1985). Interactive computer-aided control system analysis and design, in: Computer-Aided Control System Engineering, (M. Jamshidi, Herget, C.J, eds.), North-Holland, Amsterdam, pp. 229-242.

[43] Vesanterä, P.J., Cellier, F.E. (1989): Building intelligence into an autopilot using qualitative simulation to support global decision making, Simulation, 52:3, pp. 111-121.

[44] Wolovich, W.A. (1974). Linear Multivariable Systems, Springer-Verlag, New York.

[45] Wonham, W.M. (1974). Linear Multivariable Systems: A Geometric Approach, Springer-Verlag, New York.