Entwicklungstendenzen im hybriden Rechnen

Future development in hybrid computation

Elektron. Rechenanl. 10 (1968), H. 1, S. 11—17
Manuskripteingang: 27. 12. 1967

von W. GILOI
Technische Universität Berlin
Lehrstuhl u. Institut für Informationsverarbeitung

Anmerkung der Redaktion: Die vorliegende Arbeit ist die deutsche Fassung eines Übersichtsvortrags auf der fünften AICA-Konferenz (Lausanne vom 28. 8. bis 2. 9. 1967), den der Verfasser auf Einladung zu dem gestellten Thema gehalten hat.

1. Brauchen wir noch Hybridrechnen?

Etwa zwei AICA-Konferenzen zurück, 1961 in Opatija, begann eine bis heute währende Diskussion über die Frage, ob der Analogrechner auf längere Sicht überleben würde, oder ob er vollständig durch den Digitalrechner verdrängt würde. Eine AICA-Konferenz zurück, 1964 in Brighton, wurde es offenkundig, daß das Analogrechnen eine Wandlung in Richtung zum „hybriden" erfahren würde; gleichzeitig wurde die Diskussion „analog oder digital?" fortgesetzt, diesmal in einer sehr viel pessimistischeren und defensiveren Stimmung. Von da an gehörte es zum guten Ton aller Computer-Konferenzen, mindestens eine "panel discussion" über dieses Thema zu veranstalten.

Unterdessen ist der Analogrechner immer noch am Leben. Analogrechner-Hersteller brachten bessere (und teurere) Produkte auf den Markt. Wir haben in den letzten Jahren eine Art Boom in Hybridanlagen gehabt, der immerhin interessant genug war, sogar die „großen" Digitalrechner-Hersteller zu dem Versuch zu bewegen, einen Zipfel dieses Geschäfts noch zu erwischen. Trotzdem wird diese „analog oder digital?"-Diskussion weitergehen.

Über den zukünftigen Trend des analogen oder hybriden Rechnens sprechen heißt über die Weiterentwicklung des Digitalrechners sprechen. Die sich stellenden drei Hauptfragen sind hierbei:

1. wird der Digitalrechner in nächster Zukunft (sagen wir bis zur nächsten AICA-Konferenz 1970) schnell genug werden, um eine Echtzeit-Simulation bei allen interessierenden Aufgaben zu ermöglichen;
2. wird er gleichzeitig dem Benutzer (gemeinhin also Ingenieuren) die gleiche Flexibilität und Einfachheit der Programmierung bzw. den gleichen Grad von „man-machine-interaction" bieten wie der Analogrechner; und
3. wenn ja, zu den gleichen Kosten?

Nehmen wir an, der typische Digitalrechner, der heute für Simulationszwecke benutzt wird, hat eine Wortlänge von mindestens 24 bit. Damit beträgt der lokale Rundungsfehler [1] — vergleichbar mit dem statischen Fehler des Analogrechners — weniger als 10^{-7}. Der dynamische Fehler in digitalen Simulationsprogrammen wird durch den Abbrechfehler [1] der im Programm enthaltenen numerischen Integrationsverfahren bestimmt und hängt damit von der verwendeten Formel und der Schrittweite ab. Durch die „quasiparallele" Arbeitsweise des Digitalrechners ist die Integrations-Schrittweite mindestens gleich der Summe aller Ausführungszeiten der durch das Programm innerhalb eines Rechenschrittes vorgeschriebenen Operationen.

Typische Ausführungszeiten von arithmetischen Festkomma-Operationen für mittelgroße Rechner (Einadreß-Maschine, eine Speicherbank) sind zur Zeit

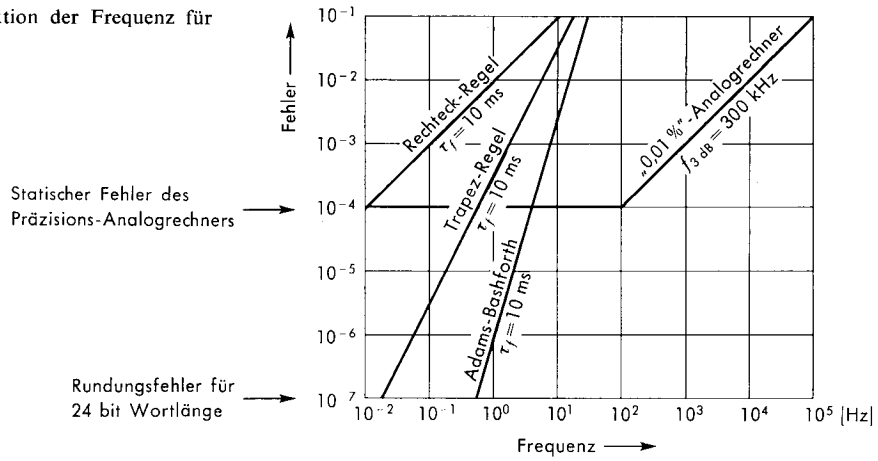
Addition, Subtraktion	2 Speicherzyklen
Multiplikation	4— 8 Speicherzyklen (typisch: 5)
Division	8— 16 Speicherzyklen (typisch: 10)
Integration (je nach verwendeter Formel)	10— 50 Speicherzyklen
Funktion einer Veränderlichen	30—100 Speicherzyklen
Funktion mehrerer Veränderlicher	einige 100 — einige 1000 Zyklen (hängt von der Anzahl der Variablen und der Zahl der Stützstellen pro Variable ab)

Hat der Rechner ein Rechenwerk für arithmetische Gleitkomma-Operationen (gewöhnlich eine Notwendigkeit für Simulationsaufgaben), so ergeben sich etwa die gleichen Werte wie für die entsprechenden Festkomma-Operationen. Es sei noch betont, daß die obengenannten Ausführungszeiten für Integration und Funktionsbildung bereits eine sehr effektive, zeitoptimale Programmierung in der Assemblersprache voraussetzen.

Mit diesen Zahlen kommt man (grob geschätzt) auf etwa 4000—5000 Speicherzyklen pro Rechenschritt bei einem typischen „100 Verstärker-Problem" (die Schrittzeit kann sich etwa 20% je nach verwendeter Integrationsformel ändern). Die typische Zykluszeit der in Frage kommenden Rechner liegt gegenwärtig etwa bei 1—2 μs (im Augenblick noch typisch: 1,75 μs; in nächster Zeit: ca. 1 μs), so daß sich eine Schrittzeit von etwa 5—10 ms ergibt. (Für einen solchen Rechner errechnet man einen Gibson-Mix von etwa 2—4 μs pro Operation.)

Die anschaulichste und bequemste Methode, den dynamischen Fehler einer numerischen Integration abzuschätzen, ist die der Frequenzanalyse auf der Grundlage der Z-Transformation [2, 3]. Bild 1 zeigt das Ergebnis einer solchen

Bild 1. Dynamischer Rechenfehler als Funktion der Frequenz für Analog- und Digitalrechner.



Fehleranalyse für drei verschiedene Integrationsverfahren (Rechteck-, Trapez- und Adams-Bashforth-Formel) unter der Annahme einer Schrittzeit von 10 ms. Zum Vergleich ist das Fehlerdiagramm der Rechenelemente eines modernen Präzisions-Analogrechners mit eingezeichnet. Wie das Diagramm zeigt, ist das Bandbreite-Fehler-Verhältnis sogar bei der Adams-Bashforth-Methode (die von relativ hoher Fehlerordnung [1] ist und damit ein verhältnismäßig günstiges dynamisches Fehlerverhalten hat) 1—2 Dekaden kleiner als beim breitbandigen Präsisions-Analogrechner. Der Vergleich fällt noch sehr viel ungünstiger für den Digitalrechner bei den einschrittigen Formeln wie der Trapez- oder gar der Rechteckformel aus.

Als Ergebnis können wir festhalten, daß für einen zulässigen dynamischen Fehler von p Prozent die wesentlichen Eigenfrequenzen des in Echtzeit zu simulierenden Systems nicht höher als $100 \cdot \sqrt[3]{\frac{}{p}}$ Hertz sein dürfen (im Falle der Adams-Bashforth-Methode). Manchmal ist diese Einschränkung nicht sehr gravierend; in vielen Fällen schließt sie jedoch eine Echtzeitsimulation auf dem vorausgesetzten Rechner aus. Diese Aussage erhält noch mehr Gewicht durch die Erfahrungstatsache, daß bei größeren Simulationsaufgaben die Schrittzeit selbst bei sehr schnellen Rechnern wesentlich größer als die angenommenen 10 ms ist.

Ein Digitalrechner, der in allen praktisch interessierenden Fällen eine Echtzeit-Simulation ermöglichen würde, müßte also mindestens zehnmal schneller als der oben betrachtete sein. Die bisherige Entwicklung der Rechengeschwindigkeit von Digitalrechnern entsprach recht gut der Faustregel, wonach sich etwa alle fünf Jahre die Rechengeschwindigkeit verzehnfacht. Damit werden wir uns auch in den nächsten Jahren noch in der Situation befinden, daß in vielen Fällen das Hybridrechnen die einzige Möglichkeit der Echtzeit-Simulation bietet. (Die sehr schnellen Multiprozessor-Großrechner mit mehreren, verschränkt arbeitenden Speichermoduln und sehr kurzer Zykluszeit, die in nächster Zukunft auf dem Markt sein werden, scheiden aus Wirtschaftlichkeitsgründen normalerweise für die betrachteten Anwendungen aus.)

Die Schwierigkeiten der rein digitalen Simulation sind durch die noch unbefriedigende Rechengeschwindigkeit der in Frage kommenden Rechner allein noch nicht voll gekennzeichnet. Fast noch größere Probleme entstehen durch die potentielle Instabilität der numerischen Integration der der Simulationsaufgabe zugrunde liegenden Differentialgleichungssysteme.

Wenn die Abbrechfehler das einzige Problem wären, könnte man dieses leicht durch Verwendung mehrschrittiger Integrationsformeln entsprechend hoher Fehlerordnung beseitigen. (Startprobleme sollen in diesem Zusammenhang außer acht gelassen werden.) Das Leiden dabei ist aber, daß das Risiko, durch die Anwendung von Differenzenmethoden Instabilitäten in die Lösung der Differentialgleichungssysteme einzuschleppen, mit der Ordnung der Differenzenoperatoren wächst.

Es ist sehr leicht, aufgrund einer Analyse auf der Basis der Z-Transformation zu sagen, ob eine Integrationsformel selbst stabil oder instabil ist. Die numerische Mathematik kann darüber hinaus auch noch aussagen, unter welchen Bedingungen eine mehrschrittige Integrationsformel, die zur Lösung einer Differentialgleichung erster Ordnung (oder eines Systems von solchen Zustandsgleichungen) herangezogen wird, notwendigerweise instabil wird [1]. Diese Aussage (das Theorem von *Dahlquist*) gibt aber leider keine hinreichende Bedingung für die Stabilität, so daß bis jetzt kein allgemeines Kriterium für die Anwendbarkeit einer bestimmten Integrationsformel existiert.

Bei linearen Zustandsgleichungssystemen könnte man die Wurzeln des zugehörigen charakteristischen Gleichungssystems berechnen, nachdem man den idealen Integrationsoperator durch die Z-Transformierte der benutzten Integrationsformel ersetzt hat. Liegen diese Wurzeln dann außerhalb des Einheitskreises (der Z-Ebene), dann weiß man, daß die Lösung des Zustandsgleichungssystems instabile Komponenten enthält.

Dieses Verfahren würde für ein komplexes Differentialgleichungssystem höherer Ordnung recht mühsam sein, es sei denn, man hat ein Rechenprogramm dafür zur Verfügung. Nun sind solche Programme geschrieben worden, so daß damit die Stabilitätsanalyse der numerischen Integration linearer Zustandsgleichungssysteme wesentlich einfacher wird [4]. Nach Kenntnis des Verfassers gibt es aber bisher noch keine solchen Verfahren bzw. zugehörigen Programme, die auch im Falle nichtlinearer Differentialgleichungssysteme funktionieren würden. Versuche, auf dem Wege der Linearisierung die lineare Stabilitätsanalyse auf nichtlineare Gleichungssysteme anzuwenden, schlugen bisher fehl.

Wir haben diese Tatsachen betont, um deutlich werden zu lassen, daß nicht nur der Analogrechner seine Fehlerprobleme hat. Die Anwendung des Digitalrechners zur Echtzeit-Simulation bedeutet in vielen Fällen, die (statischen) Fehler des Analogrechners (mit allen dadurch verursachten Skalierungsproblemen) gegen die u. U. noch unangenehmere Problematik der numerischen Integration einzutauschen.

Ferner darf noch ein weiterer Gesichtspunkt in dieser Diskussion nicht vergessen werden. Ingenieure sind meistens nur

gelegentliche Benutzer elektronischer Rechenanlagen und keine sehr geübten und erfahrenen Programmierer. Im Falle des Analogrechners ist die Art der Programmierung nahezu identisch mit der Denkungsweise eines Ingenieurs und der Art, in der er seine Aufgaben zu formulieren pflegt. Dies resultiert nicht nur aus der Natur der analogen Rechenschaltung als eine unmittelbare Entsprechung zum Blockdiagramm des simulierten Systems, sondern der Analogrechner hat auch die Denkungsweise selbst stark beeinflußt. In der Ingenieurausbildung ist im vergangenen Jahrzehnt die einseitige Betonung der Operatorenkalküle im Frequenzbereich (wie z. B. der Laplacetransformation) einer allgemeineren Betrachtungsweise physikalischer Systeme durch Differentialgleichungen (Zustandsgleichungen) gewichen. Ich wage zu sagen, daß dies nicht ohne die Existenz des Analogrechners geschehen wäre.

Schließlich kommt der Analogrechner (mit all seinen Unvollkommenheiten) fast in idealer Weise der Forderung nach, die *R. W. Hamming* in die Worte faßte: *The Purpose of Computing is Insight, Not Numbers.* Dies wird durch den hohen Grad von „man-machine-interaction" erreicht, den der Analogrechner bietet. (So wie der Digitalrechner oft bei wissenschaftlichen Aufgaben benutzt wird, produziert er häufig ein Maximum an Zahlen und ein Minimum an Einsicht. Diese Tatsache mag die Minderwertigkeitskomplexe der Analogrechner-Benutzer etwas lindern.)

Als Ergebnis unserer Diskussion „analog oder digital?" halten wir fest, daß in vielen Fällen die rein digitale Echtzeit-Simulation zur Zeit und in naher Zukunft noch nicht möglich ist.

In diesen Fällen bietet die hybride Simulation das Optimum an Leistungsfähigkeit, Verfügbarkeit und Wirtschaftlichkeit. Aber auch in den Fällen, in denen ein mittelgroßer Digitalrechner geeignet ist, werden wir zwar auf die analoge „hardware" verzichten, nicht jedoch auf die analogrechnergemäße Programmierung und nicht auf eine entsprechende „interaction" zwischen Mensch und Maschine.

Man mag gegen die Einbeziehung einer analogrechnergemäßen Programmierung in die Definition des hybriden Rechnens Einwände erheben, und man mag dies als einen faulen Trick betrachten, um letzteres vor dem Schicksal zu bewahren, eine vorübergehende Erscheinung zu sein. Wir sehen jedoch keinen Grund dafür, die Definition des Hybridrechnens nicht auch auf die „software"-Eigenschaften beider Rechner auszudehnen. Die Erfahrungen der Vergangenheit haben uns schließlich gelehrt, daß „software"-Eigenschaften mindestens ebenso wichtig wie „hardware"-Eigenschaften sind.

2. Das Spektrum des Hybridrechnens

Nachdem wir somit hybrides Rechnen als eine Verbindung analoger und digitaler Programmierungseigenschaften und/oder „hardware" definiert haben, können wir daran gehen, die sich hier bietenden Möglichkeiten zu klassifizieren (Bild 2).

Der erste Schritt vom reinen Analogrechner zum Hybridrechner stellt der Analogrechner mit digitaler Steuerung dar. Diese besteht gemeinhin in digitalen Zeitgebern und dem sogenannten Digitalzusatz, der einen beliebig benutzbaren Vorrat von logischen Verknüpfungselementen enthält. Es wäre präziser, hier von einem „iterativen" Analogrechner zu sprechen, denn die Funktion der digitalen Komponenten besteht genau darin, die Programmierung von Iterationsverfahren zu ermöglichen. Die eigentlichen Rechenoperationen werden nach wie vor ausschließlich von den analogen Rechenelementen ausgeführt.

Auf der anderen Seite besteht der erste Schritt vom reinen Digitalrechnen zum Hybridrechnen in einer analogrechnergemäßen Programmierung des Digitalrechners, realisiert durch sogenannte „blockorientierte" Programmiersprachen. Alle mathematischen Operationen werden hier ausschließlich digital ausgeführt.

Der nächste Schritt in Richtung zum echten hybriden Rechnersystem würde auf der einen Seite darin bestehen, den Analogrechner nun auch mit speziellen digitalen Recheneinheiten auszustatten, wie z. B. digitale Multiplizierer, Funktionsgeber oder Funktionsspeicher. Solche Systeme gibt es nicht und wird es nach unserer Meinung auch nicht geben (erste Versuche in dieser Richtung zeigten schon vor Jahren, daß dieser Weg unverhältnismäßig aufwendig ist), so daß wir diese Möglichkeit von der weiteren Diskussion ausschließen können.

Als Gegenstück wäre der Digitalrechner mit eingebauter analoger „hardware" zu betrachten (so wie man z. B. eine spezielle digitale „hardware" für arithmetische Gleitkommaoperationen haben kann). Ein bekanntes Beispiel ist das von *Karplus* entwickelte Verfahren zur hybriden iterativen Lösung von partiellen Differentialgleichungen. Hierbei wird die analoge Komponente durch ein Netzwerk aus passiven Elementen gebildet, das unter der Kontrolle des Digitalrechners steht. Ich kenne keine weiteren Beispiele für ein solches System, aber die Möglichkeit ist an sich interessant und eine weitere Diskussion wert.

Das Gegenstück zum hybriden Rechnersystem, in dem autonome Analog- und Digitalrechner miteinander gekoppelt sind, ist das einzelne hybride Rechenelement, in welchem analoge und digitale Komponenten und/oder eine analoge und digitale Wertedarstellung kombiniert werden. Beispiele

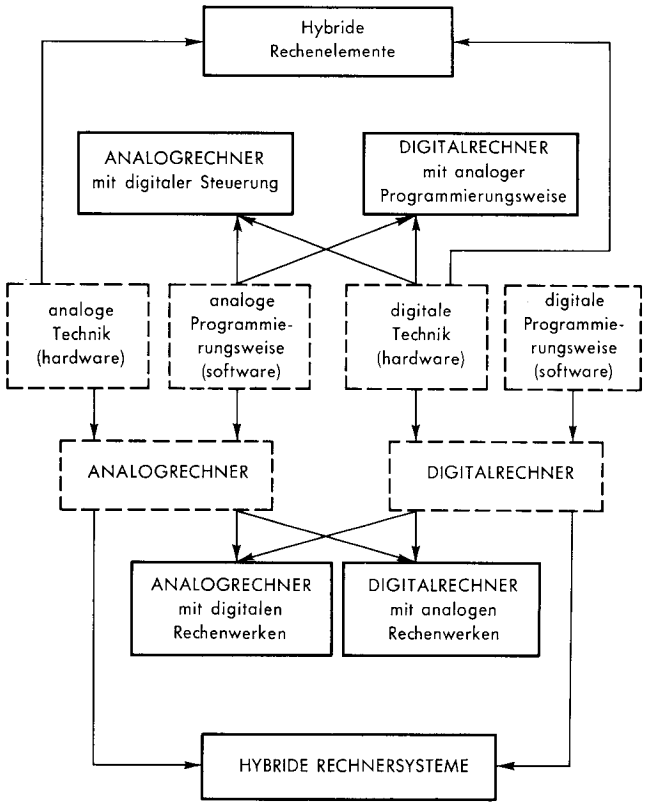


Bild 2. Das Spektrum der hybriden Rechentechnik.

sind die von *Skramstad* vorgeschlagenen CADDA-Elemente [5] oder das AMBILOG-System [6].

Diese hier aufgestellte Klassifizierung wird es uns erleichtern, im folgenden mögliche zukünftige Entwicklungen des hybriden Rechnens aufzuzeigen. Dabei wollen wir uns an das Schema von Bild 2 halten.

3. Trends

3.1 Einzelne Hybridelemente und Spezialrechner

Nach unserer Meinung werden hybride Rechenkomponenten eine wachsende Bedeutung beim Aufbau kleiner Spezialrechner erlangen. Beispiele sind: Rechner für Navigationsaufgaben, Fahrzeugsteuerung, Prozeßsteuerung und militärische Anwendungen. Diese Komponenten werden auf der Basis von logischen Schaltkreisen, analogen Schaltern, Umsetzern (A/D- und D/A-), Zählern (mit oder ohne Rückführung durch ein Schaltnetz) usw. arbeiten. Sie können analoge, binär-inkrementelle und binär-parallele Wertedarstellungen in sich vereinigen. Einfache Beispiele sind der multiplizierende Digital-Analog-Umsetzer (MDAU), hybride oder digitale Funktionsgeber oder das AMBILOG-System [6]. Eine andere Möglichkeit ist die der gemischten Wertedarstellung (CADDA) nach *Skramstad* [5, 7, 8]. Diese Komponenten haben sich mittlerweile jedoch als zu aufwendig im Vergleich zu ihrer Leistung erwiesen, so daß wir nicht glauben, daß sie noch einmal eine Bedeutung erlangen werden.

Mit den von uns entwickelten PHENOS („**p**recise **h**ybrid **e**lements for **n**onlinear **o**perations") [9] haben wir ein neues System von hybriden Rechenelementen vorgestellt. Wir glauben, daß wir damit einen höheren Grad von Vielseitigkeit und Flexibilität und bessere Leistungsdaten erreichen als bei allen bisherigen analogen oder hybriden Elementen. In unserer auf dieser Konferenz präsentierten Veröffentlichung sprechen wir über die besonderen Vorteile, die ein mit PHENOs bestückter Analogrechner zu bieten hat, aber wir glauben nicht, daß in Zukunft unbedingt jeder Analogrechner mit diesen Komponenten ausgestattet sein wird. Wie wir bereits betonten, werden diese Elemente vorwiegend zur Ergänzung hybrider Rechnersysteme oder zum Aufbau von Spezialrechnern dienen.

3.2 Digitalrechner mit analogrechnergemäßer Programmierung

Blockorientierte Programmiersprachen, die eine analogrechnergemäße Programmierung des Digitalrechners ermöglichen, gibt es bereits in einer großen Vielfalt. Die bekanntesten Systeme wie z. B. MIDAS, DSL 90 oder 360/CSMP werden in vielen Simulationszentren verwendet. Einige Programmiersysteme haben auch Stapelverarbeitungs-Fähigkeiten, so daß sie sich gut in die übliche Organisationsform eines Rechenzentrums einfügen lassen. Allerdings verliert man dann die sehr wichtige Eigenschaft einer direkten „man-machine-interaction".

Blockorientierte Programmiersysteme, die dem Benutzer den gleichen unmittelbaren Zugriff zum Rechner und die gleiche Einfachheit der Programmierung wie ein Analogrechner bieten, werden auf lange Sicht der Weg sein, die analoge Simulation zu ersetzen. Wie wir bereits ausführlicher begründet haben, ist der Digitalrechner zur Zeit allerdings hierfür noch nicht schnell genug. Aber auch jetzt schon stellt die blockorientierte Programmierung ein sehr wertvolles Hilfsmittel dar.

Ich glaube nicht, daß schon in nächster Zukunft der Digitalrechner mit analogrechnergemäßer Programmierung die hybriden Rechenanlagen ersetzen kann. Nach meiner Meinung können aber sehr kleine und billige Digitalrechner mit spezieller Programmierung schon in Kürze zu einer ernsthaften Konkurrenz für die kleinen und mittleren Analogrechner werden.

Diese Behauptung wird durch die Tatsache gestützt, daß verschiedene Hersteller in nächster Zeit kleine Digitalrechner auf den Markt bringen werden, deren Verkaufspreis bei etwa 50000,— DM beginnt. Der Hauptzweck dieser Rechner mag sein, das Kernstück einer Buchungsmaschine oder eines Bank-Terminals zu bilden oder zu Aufgaben der Datenerfassung, Prozeßsteuerung, usw. eingesetzt zu werden. Alle diese Anwendungen haben gemeinsam, daß dazu der Rechner keine Allzweck-Programmierung (wie bei wissenschaftlichen Aufgaben notwendig) zu haben braucht. Dies ist der Hauptgrund dafür, daß solche Rechner billig sein können, Wenn man nun einen Rechner dieser Klasse mit einem einfachen, blockorientierten, interpretierend arbeitenden Programmierungssystem ausstattet, wird er kaum mehr kosten als ein kleiner Tisch-Analogrechner, während er die Rechenkapazität eines mittleren Analogrechners mit — sagen wir — 50 Rechenverstärkern bietet. Natürlich wird ein solcher „digitaler Analogrechner" wesentlich langsamer als ein wirklicher Analogrechner sein, aber für diesen Nachteil (der in den meisten Fällen in Kauf genommen werden kann) wird sich der Benutzer den großen Vorteil einhandeln, keine mühsame Skalierungen mehr vornehmen zu müssen (da alle Variablen als Gleitkomma-Zahlen dargestellt werden können). Durch eine bescheidene Hardware-Ergänzung kann der gleiche Grad von „interaction" wie beim echten Analogrechner erreicht werden.

3.3 Analoge „hardware" im Digitalrechner

Selbstverständlich wird der Digitalrechner von morgen im allgemeinen keine analoge „hardware" eingebaut haben. Für spezielle Zwecke jedoch ist es ganz interessant zu untersuchen, welche Möglichkeiten sich hier bieten. Die Methode von Karplus, ein Netzwerk aus passiven Elementen zur diskreten Lösung partieller Differentialgleichungen einzubauen, wurde bereits erwähnt. Im Gegensatz zu dem üblichen hybriden Rechnersystem läuft hier die Programmierung der Aufgabe und die Durchführung der Rechnung ausnahmslos unter der Kontrolle des digitalen Programms ab. Der Benutzer braucht nicht einmal zu wissen, daß gewisse Unterprogramme durch analoge Rechenschaltungen implementiert werden.

Manche mathematischen Aufgaben erfordern zu ihrer Lösung eine große Zahl von Rechenschritten, sei es, daß das Problem nur iterativ gelöst werden kann oder sei es z. B. in den Fällen der Parameter-Optimisierung oder der statistischen Analyse. Selbst auf schnellen Digitalrechnern kann die Lösung solcher Probleme sehr viel Rechenzeit erfordern. Gerade für diese Aufgaben können daher hybride Rechnersysteme äußerst nützlich sein, um so mehr, wenn die Genauigkeitsanforderungen bescheiden sind (Näherungslösungen können jederzeit mit guter Konvergenz digital verbessert werden). Trotzdem wird man aber in den wissenschaftlichen Rechenzentren deshalb niemals geneigt sein, an die vorhandenen Digitalrechenanlagen noch einen Analogrechner anzuhängen. Einer der Gründe hierfür ist die relativ komplizierte Programmierung des Hybridsystems, aber eine viel elementarere Ursache liegt in der bei diesem Personenkreis vorherrschenden instinktiven Abneigung gegen Verstärker, Dioden, Steckschnüre und anderes elektrotechnisches Zeug. Die einzige Methode, um solche Leute an den Segnungen des Hybridrechners teilhaben zu lassen, würde darin bestehen, in den

Digitalrechner geeignete analoge Unterprogramme einzubauen und dies vor dem Benutzer zu verbergen.

In einem speziellen Forschungsvorhaben, dem wir den Namen AIDER (**a**nalog **i**mplemented **d**ifferential **e**quation solving **r**outine) gaben, untersuchen wir die Möglichkeiten, die sich zum Einbau sehr schneller analoger Unterprogramme zur Lösung gewöhnlicher Differentialgleichungssysteme in den Digitalrechner bieten. Das Hauptproblem besteht in diesem Falle in der erforderlichen automatischen Herstellung des analogen Unterprogramms (automatic patching). Würden wir versuchen, dies auf die gleiche Weise durchzuführen wie es normalerweise manuell getan wird, so hätten wir die Telefonzentrale einer Kleinstadt einzubauen. Nun ist der

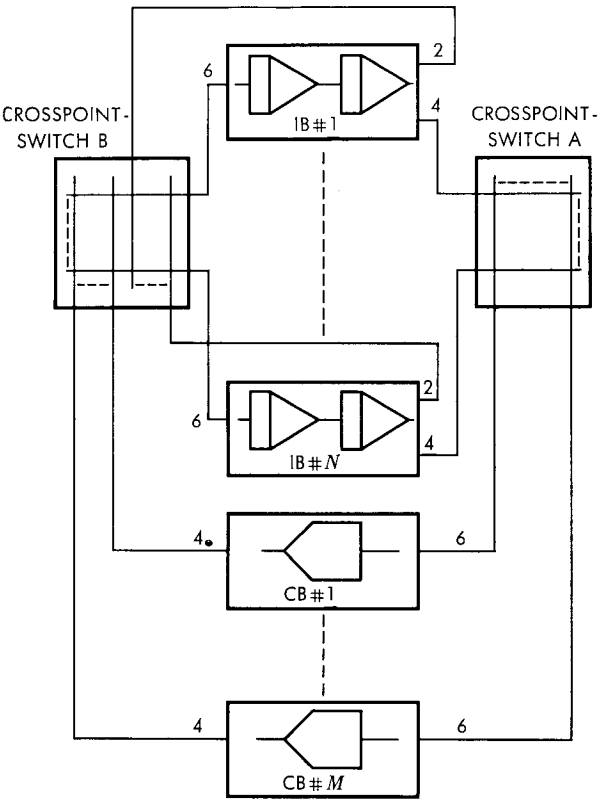


Bild 3. Struktur von AIDER.

INTEGRATIONSBLOCK:

2 Integrierer
2 Summierer
6 Servo-Potentiometer

KOPPELZWEIG:

2 Multiplizierer/Dividierer
1 digital einstellbarer Funktionsgeber
1 Standardeinheit für unstetige Funktionen
1 digital einstellbares Potentiometer

CROSSPOINT-SWITCHES A und B:

für $N = 10$ und $M = 9$
$< 1\,000$ Schaltpunkte

Bild 4. Struktur von AIDER.

bisher übliche Weg des Aufsteckens einer Rechenschaltung ein Relikt aus der Zeit, als die Rechenelemente teuer und die Steckverbindungen im Vergleich dazu billig waren. Mittlerweile, im Zeitalter der integrierten Schaltungen, haben sich diese Verhältnisse fast umgekehrt. Die Bilder 3 und 4 zeigen schematisch den „hardware"-Teil von AIDER. Wie Bild 3 zeigt, haben wir eine Anzahl von „Integrationsblocks" (IB), von denen jeder 2 analoge Integrierer, 2 Summierer und 6 Servo-Potentiometer enthält. Jeder IB kann eine (lineare oder nichtlineare) Differentialgleichung 2. Ordnung mit gegebenen Konstanten und Anfangsbedingungen lösen. Zusätzlich haben wir in diesem System „Kopplungsblocks" (CB), von denen jeder eine gewisse Zahl von nichtlinearen Rechenelementen enthält, die die nichtlinearen Rückführungszweige innerhalb eines IB bilden können (lineare Rückführungszweige sind im IB enthalten) oder zur (nichtlinearen) Verkopplung der einzelnen IBs dienen. Bevor die Rechnung beginnt, erhält jeder CB eine „Statusinformation" (ein oder mehrere 24 bit-Worte) vom Digitalrechner, wodurch Funktionsgeber eingestellt werden oder im Falle der „diskontinuierlichen Funktionen" der Funktionstyp einschließlich seiner Unstetigkeitsstellen definiert wird. (Solche Funktionen sind z. B. Begrenzer, tote Zone, Hysterese, etc.). Durch 2 „crosspoint"-Schalter kann das gesamte System programmiert werden. In dem im Aufbau befindlichen System, das Differentialgleichungssysteme bis zur 20. Ordnung lösen können soll, werden weniger als 1000 Kreuzungspunkte benötigt, ein durchaus vernünftiger Aufwand.

Ich wage gegenwärtig nicht zu sagen, ob ein solches System je praktische Bedeutung erlangen wird, aber dies ist für uns auch ein sekundärer Gesichtspunkt. Die Schwierigkeit eines solchen Systems liegt weniger in der technischen Realisierung als in dem dafür benötigten, sehr komplizierten Programm. Aber gerade diese Programmierung bringt eine ganze Zahl von Fragen von prinzipieller Bedeutung mit sich, die dieses Projekt für uns interessant machen. Im ersten Stadium des Experimentierens versuchen wir, soweit wie möglich den „hardware"-Teil auf der vorhandenen hybriden Rechenanlage zu programmieren.

3.4 Hybride Rechnersysteme

Ich habe bereits zum Ausdruck gebracht, daß meiner Meinung nach auf der einen Seite hybride Rechnersysteme noch für einige Zeit die einzige Lösung für viele Echtzeit-Simulationsaufgaben sein werden, auf der anderen Seite aber nicht in den allgemeinen wissenschaftlichen Rechenzentren Fuß fassen werden. Wir sollten uns auch bewußt sein, daß eine solche Kombination von Digitalrechner und Analogrechner zwar die Chance bietet, die Fähigkeiten beider Arten zu vereinigen, daß man auf der anderen Seite aber mit Sicherheit auch die Programmierungsschwierigkeiten beider Rechnerarten vereinigt.

Diese Situation kann nur dadurch gemildert werden, daß eine komfortable und leistungsfähige Programmausstattung, speziell für das Hybridrechnen konzipiert, vorhanden ist. Bild 5 gibt eine Aufzählung der einzelnen Programm-Teile oder Programme, die in der Programmausstattung eines Hybridsystems zum Teil vorhanden sein müssen, zum Teil vorhanden sein sollten. Alle Programme in der Liste der absolut notwendigen Programme und das erste in der Liste der wünschenswerten Programme sind in dem hybriden Programmiersystem enthalten, das wir für unsere Rechenanlage entwickelt haben. Wir werden auf dieser AICA-Konferenz darüber berichten [10]. (Natürlich existieren auch schon ähnliche Programmiersprachen für andere Rechner-

A. Basis-Programm-System (obligatorisch)

1. Makro-Befehle für alle Steuerungsfunktionen und den Daten-Transfer
   - Betriebsartensteuerung des Analog-Rechners
   - Einstellen von Servopotentiometern
   - Anwahl und Abfragen der Rechenelemente-Ausgänge
   - Setzen und Abfragen von Steuerleitungen
   - Einstellen von Zeitgebern
   - Daten-Transfer D → A, A → D
2. Erweiterung der Assembler-Sprache um diese Makro-Befehle
3. Erweiterung der Compiler-Sprachen (ALGOL, FORTRAN) um diese Makro-Befehle
4. On-Line-Steuerungsprogramm, das es dem Benutzer ermöglicht, auf der Grundlage der Makro-Befehle in das System einzugreifen (in conversational mode)
5. Erweiterung von Austest- und Diagnose-Routinen
6. Spezielle Unterprogramme
   - Table-look-up für die Erzeugung von Funktionen einer oder mehrerer Variabler
   - Pseudo-Zufalls-Zahlen-Geber
   - Totzeitgenerierung
   - Parameter-Optimierung und Berechnung statistischer Parameter
   - Fehlerkorrigierende digitale Filter, etc.

B. Spezial-Routinen und Komfort-Hilfsprogramme (wünschenswert)

1. Automatisches Statisches Prüfen
2. Automatischer Entwurf der Analogschaltung und deren Skalierung
3. Diagnose-Programm, das das Stabilitätsverhalten und die dynamischen Fehler von Differential-Gleichungen unter Verwendung verschiedener Integrationsformeln untersucht

Bild 5. Programmausstattung für hybride Rechnersysteme.

systeme als das unsere.) Ein typischer Vertreter des Programms B. 2 ist das wohlbekannte APACHE-Programm. Im ersten Teil dieses Vortrags habe ich bereits über die (bis jetzt noch ungelösten) Schwierigkeiten gesprochen, die darin liegen, ein Programm aufzustellen, das eine Analyse der dynamischen Fehler und der Stabilität bei der numerischen Lösung von Differentialgleichungen für beliebige Integrationsmethoden durchführt.

Von der nächsten Zukunft erwarte ich die Entwicklung noch komfortablerer Programmiersysteme für hybride Rechenanlagen, als das hier skizzierte (auch bei uns gehen die Arbeiten in dieser Richtung weiter). Die „hardware"-Struktur der hybriden Rechnersysteme aber wird eher noch stärkeren Änderungen unterworfen sein, die durch die im folgenden Abschnitt diskutierten möglichen Weiterentwicklungen der technischen Realisierung der Analogrechner verursacht werden.

3.5 (Hybride) Analogrechner

Wie bereits ausgeführt, werden kleine und billige Digitalrechner mit einem speziellen Programmiersystem in Konkurrenz zu den kleinen bis mittleren Analogrechnern in den Fällen treten, wo die Rechengeschwindigkeit von sekundärer Bedeutung ist. Die einzige Chance für den Analogrechner-Hersteller, dieser Bedrohung zu entgehen, besteht darin, billigere Rechner zu bauen. Zu diesem Zweck kann nötigenfalls auch ein gewisses Maß an Präzision geopfert werden (ohnehin bei kleinen Analogrechnern von sekundärer Bedeutung). Schließlich sollten die rapide gefallenen Preise für Halbleiter allgemein und für integrierte Schaltung im besonderen gelegentlich auch einmal zu sinkenden Rechnerpreisen führen. Andererseits wird bei dieser Klasse von Analogrechnern sich die Art der technischen Realisierung der Rechenelemente wahrscheinlich wenig ändern.

Dagegen wird die Technik und Struktur der mittleren bis großen Analogrechner — seien sie Teil eines Hybridsystems oder sogenannte „hybride" Analogrechner — unserer Meinung nach erheblichen Änderungen unterworfen sein. Ich wage zu behaupten, daß die Zeit gekommen ist, zwei Dinge völlig neu zu betrachten: erstens die Art, in der bisher die analogen Rechenelemente durch logische Verknüpfungselemente ergänzt werden und zweitens einige der grundlegenden Arbeitsprinzipien der analogen Komponenten.

Beginnen wir mit dem Digitalzusatz. Meiner Meinung nach war die Idee, die bei seiner Einführung zugrunde lag, wirklich gut. Es war die Idee, nicht einen ganzen Zusatzschrank kaufen zu müssen, um eine Handvoll logischer Schaltkreise und einige Flipflops unterzubringen, und dafür das Äquivalent eines hübschen kleinen Digitalrechners entrichten zu müssen. Da zur damaligen Zeit (1964) niemand sagen konnte, wieviel Elemente von jeder Sorte im allgemeinen benötigt werden würden (oder, um es provozierender zu sagen, wozu der Digitalzusatz überhaupt gut sei), war es die beste Lösung, logische Schaltungen steckbar und die Elemente selbst auswechselbar zu machen.

Durch den enormen Preisrückgang bei integrierten Schaltungen, den wir inzwischen erfahren haben, befinden wir uns aber nun wieder in der Situation, daß man für den Digitalzusatz des Analogrechners fast das Äquivalent eines der neuen, kleinen und extrem billigen Digitalrechner zu zahlen hat. Meiner Meinung nach ist es lächerlich, — sagen wir — 40000,— DM für eine Handvoll logischer Schaltkreise zu zahlen, wenn zur gleichen Zeit zwei NOR-Schaltungen oder ein Flipflop für 1,70 DM zu haben sind. Was folgt daraus? Entweder wird der Digitalzusatz durch einen dieser kleinen und billigen Digitalrechner ersetzt werden, die zu vergleichbaren Kosten einen sehr viel höheren Komfort und vermehrte Möglichkeiten bieten werden, oder wir müssen bessere und weniger aufwendigere Lösungen für den Einbau digitaler Steuerelemente in den Analogrechner finden. Meiner Meinung nach besteht diese Lösung in einer besseren Integration der logischen Schaltkreise in das System von Analogkomponenten. Ich glaube, ich sollte dies noch etwas näher erläutern.

Der Preis, den wir heute für den Digitalzusatz zahlen, ist nur zum kleinen Teil der Preis für die logischen Schaltkreise. Der größte Teil der Kosten wird verursacht durch Gestell, Einschübe, Steckeinheiten, Programmierfeld, Programmierbrett usw. Was machen wir damit? Wir programmieren Steuerprogramme, die den Analogrechner in die Lage versetzen, iterativ zu arbeiten. (Selbstverständlich läßt sich mit dem Digitalzusatz auch noch etwas mehr anfangen. Im Falle statistischer Untersuchungen, zum Beispiel, kann man Ereignis-Zähler, binäre Rauschgeneratoren und anderes programmieren; aber alle diese Aufgaben könnten auch spezielle Elemente übernehmen.)

Iterative Rechenverfahren erfordern die individuelle Steuerung von einzelnen Integrierern (oder von Integrierern, die als Speicher mißbraucht werden). Sie erfordern Komparatoren, die die Einhaltung bestimmter Bedingungen bei den analogen Rechengrößen überwachen und durch eine boolesche Variable anzeigen, und schließlich Elemente für die Speicherung dieser booleschen Variablen. Es können ferner Zähler benötigt werden, die die Zahl der durchgeführten Rechenschritte ermitteln, usw. Nun, warum versieht man dann nicht die (elektronische) Steuerung der Integrierer generell mit drei Flipflops zum Speichern der einzelnen Steuerbefehle (für die 3 Betriebsarten)? Diese Flipflops könnten ihrerseits immer mit Eingangskonjunktionen versehen sein, so daß ein Steuerbefehl logisch bedingt werden kann. Warum verbindet man nicht generell jeden Komparator mit einem Flipflop, um seine boolesche Ausgangsgröße zu speichern (oder zurückzusetzen)? Warum enthält das Bediengerät des Rechners nicht immer auch einen Zähler für die Anzahl der Rechenschritte? Durch solche Maßnahmen und durch eine Steuerung, die etwas flexibler und „more sophisticated" als üblich ist, könnte man in den meisten Fällen den Digitalzusatz (und damit seine Kosten) einsparen, und dieser Weg wäre auch für kleine Rechner gangbar.

Die zweite Vorhersage, nach der einige der Grundprinzipien, auf denen die analogen Rechenelemente basieren, durch andere ersetzt werden, resultiert zunächst aus der allgemeinen Überlegung, daß die bisherigen Prinzipien zu einer Zeit erfunden wurden, als die elektronischen Komponenten (z. B. die Operationsverstärker) die aufwendigeren Teile des Rechners waren und die passiven Komponenten die billigeren. Heute hat sich dieses Verhältnis umgekehrt, so daß wir nach neuen Prinzipien der technischen Realisierung Ausschau halten sollten, die zwar einen größeren elektronischen Aufwand bedingen können, dafür aber voluminöse Kondensatoren, Thermostate, temperaturempfindliche Dioden usw. überflüssig machen.

Der konkretere Hintergrund für unsere Vorhersage wird aber gebildet durch die Tatsache, daß wir uns solche neuen Prinzipien für die Realisierung von Integrierern, Multiplizierern, Funktionsgebern bereits ausgedacht haben (diese Dinge haben nichts mit „PHENO" zu tun). Es ist noch zu früh, hier bereits im Detail über Ergebnisse und Leistungsdaten zu reden, aber wir können jetzt schon sagen, daß die neuen Rechenelemente beträchtlich besser als die bisher bekannten sein werden (wir streben eine Erhöhung des Bandbreite/Fehler-Verhältnisses um etwa eine Zehnerpotenz an).

Durch die Verwendung integrierter Schaltkreise sind heute die Signalpegel in den Digitalrechnern weitgehend vereinheitlicht worden. Dadurch wird es für die Analogrechner-Hersteller jetzt möglich, ein „control interface" in den Analogrechner zu integrieren (das ein Betriebsarten-Eingaberegister, ein Anwahl-Adreßregister, Einzelbit-Ein/Ausgabe etc. enthält). Wenn der Integrierer von morgen aufgrund der andersartigen Integrationsmethode von Hause aus ein digitales Register enthalten wird, braucht nicht mehr viel von den heute üblichen umfangreichen und teuren Koppelwerken in den hybriden Rechnersystemen übrigzubleiben. Wir wollen sehen, was die Zukunft bringen wird.

Literatur

[1] *Henrici, P.*, Discrete variable methods in ordinary differential equations. John Wiley & Sons, New York 1962.

[2] *Hamming, R. W.*, Numerical methods for scientists and engineers. McGraw-Hill, New York 1962.

[3] *Gilliland, M. C.*, Technical comment. SIMULATION, Vol. 8, no. 6 (1967), pp. 308—309.

[4] *Schweizer, G., Seelmann, H.*, Möglichkeiten für die Anwendung von Verfahren des Abtasttheorems zur Untersuchung komplexer Systeme bei digitaler und hybrider Simulation. Dornier-Report 5614/A23.

[5] *Skramstad, H. K.*, A combined analog-digital differential analyzer. Proc. Eastern Joint Computer Conf. 1961.

[6] *Hagan, T. G.*, AMBILOG Computers: Hybrid machines for measurement-system calculation tasks. Proc. 17th Annual ISA Conf., New York, (Oct. 1962).

[7] *Schmid, H.*, Combined analog-digital computing elements. Proc. Western Joint Computer Conf., 1961.

[8] *O'Grady, P.*, A simplified hybrid differential analyzer. M. S. Thesis, University of Arizona, 1964.

[9] *Giloi, W., Sommer, H.*, PHENO — A new concept of hybrid computing elements. To be presented on the 5th Conference of AICA, Lausanne, Aug. 28th — Sept. 2nd, 1967.

[10] *Beckert, D., Liebig, H., Wiesenthal, P.*, Programmierung von hybriden Rechensystemen auf der Grundlage problemorientierter Sprachen. To be presented on the 5th Conference of AICA, Lausanne, Aug. 28th — Sept. 2nd, 1967.