文　献　紹　介

A：数　値　解　析　　B：プログラミング　　C：計算機方式
D：回路および機器　　E：オートマトン　　F：応用その他

A-82. 非線形分解原理

J.L.Sanders: A Nonlinear Decomposition Principle〔J. of O. R. Vol. 23, No. 2, Mar/April, 1965, PP. 266～271〕.

大規模なシステムモデルを線形計画法によって解析しようとする際，次元の非常に高い線形モデルを一挙に解かずにいくつかのサブシステムの最適化とそれらを調整することにより，すなわち大規模のモデルをサブシステムに分解（Decompose）する方法は Dantzig により既に与えられている．本論文はこの分解原理をある種の非線形モデルに対して拡張したものである．いま原問題を二つのサブモデルに分解し，それぞれのサブモデルはラグランジュ定数および助変数をもつが次元は半減される．したがってサブモデルの最適化は各ラグランジュ定数および助変数に対して容易にもとめることができ，原問題の拘束条件に合致するよう，この助変数の値を最峻傾斜法によって逐次近似することにより，大規模のシステムの最適化を計ることができる．

また，この分解原理は，concave な拘束条件および convex な目的関数をもつある種の最適化問題を，convex な拘束条件および convex な目的関数をもつ問題に変換することを可能にする．　（成田正之）

B-83. FORTRAN 対 COBOL

M. D. Fimple: FORTRAN vs. COBOL〔Datamation Vol. 10 No. 8, August. 1964, pp. 34～40〕

1961 年に IBM 7090 を導入したとき，9 PAC と FORTRAN とでプログラムを準備した．その後は，FORTRAN のほうだけが増加している．データ処理は，WDPC 入出力パッケージをつけくわえ，20～50％速くした．1963 年に FORTRA IV が入ったが，実行速度がかなりおそいので採用しなかった．Commercial Translator は好成績だったが開発をつづけないということなので採用しなかった．

COBOL ができたときテストしてみた．ところが別表のように，われわれの FORTRAN より少しおそいことがわかった．FORTRAN か COBOL かは，仕事の種類によるというよりは，機械の種類によるように思われる．

COBOL のソースプログラムは，カードにして 2.5 倍であり，目的プログラムは 20％ ほど長くなった．しかも，やたらにサブルーチンに飛ぶようになっている．翻訳のリストは，コンパイラが勝手にわりつけた名前におきかわっているので，ひどく読みづらい．

筆者は 1959 年にほとんど素人として FORTRAN を習うのに 2 週間かかった．その後いろいろ経験をかさねて後，Commercial Translator では 3 週間かかった．ところが COBOL では 6 週間を要した．それは適当な教育用の入門書がなく，文法書だけであること，独学で，質問できる相手がいなかったこともあるが，言語それ自体もわるい．冗長で，制限が強く，またたいていの英語が固有語（reserved word）になっている．

それやこれやで，将来 COBOL がうんと良くなった証拠が得られるまでは FORTRAN を使いつづけることにきめた．　（西村恕彦）

B-84. COBOL 対 FORTRAN（反論）

J. P. Junker and G. R. Boward: COBOL vs. FORTRAN: A Sequel〔Datamation Vol. 11 No. 4, April. 1965, pp. 65～67〕

かなり大きいデータ処理の問題を依頼され，一人のプログラマが COBOL の勉強からはじめて，プログラムの完成まで約 2 カ月かかった．実行は IBM 7094 で，約 15 万件のレコードを 7.2 分で処理し，予期以上の好成績だった．

ちょうどそのとき，Fimple の報告がでたので，われわれとしてもテストをしてみた．結果は別表のとおりで，バッファリングしない FORTRAN II は問題にならないWDPC パッケージとよく似た CENTAUR システムは，速さの点では COBOL と同等までは行きうる．しかし融通性が劣る．COBOL ではブロッキングの効果ははなはだ大きい．

Fimple の結果は，パッケージを特定の仕事にあう

Fimple	ブロッキング8	ブロッキング1
WDPC FORTRAN	3.4 分/3万件	5.2 分/3万件
COBOL	3.8 分/3万件	5.6 分/3万件

Junker & Boward	ブロッキング 10	ブロッキング1
COBOL	2.3分/5.5万件	4.8分/2.8万件
CENTAUR (Binary)	3.2分/5.5万件	—
CENTAUR (BCD)	5.3分/5.5万件	4.4分/2.8万件
FORTRAN II	15.0分/5.5万件	10.2分/2.8万件

J & B, 入出力のみ	ブロッキング10	—
COBOL	2.1分/5.5万件	—
CENTAUR (BCD)	7.1分/5.5万件	—
FORTRAN II	13.8分/5.5万件	—

ように修正したためではなかろうか．データ処理の仕事一般について，COBOL の能力が WDPC パッケージつき FORTRAN に劣るとは思えない．

7094 では COBOL と，FORTRAN はどちらも，IBSYS の下についているから，仕事に応じて選択できる．この選択は，問題の種類によるのであって，機械の種類によるとはいえない．なほ標準の入出力ファイルは，7094 (SHARE 系) COBOL では，パディングしてないことや，ラベルの形式の点で，7080(GUIDE 系) COBOL と共通性がない．

COBOL はむずかしい言語ではなく，教育コースもいろいろできている．言語の冗長さは文書化の点で有利であり，とくに Data Division ではそうである．COBOL をおぼえれば，同時にデータ処理の概念も理解できるようになる．

筆者の一人は米空軍データサービスセンターのプログラム・ドクメンテーション・ブランチの長である．

(西村恕彦)

B-85. NPL ハイライト

G. Radin and H. P. Rogoway: NPL: Highlights of A New Programming Language [C. ACM, Vol. 8 No. 1, January. 1965, pp. 9～17]

NPL は現代のいろいろの計算機，モニタにみあった能力をもち，科学，事務，実時間，システムの各プログラミングに使える必要なあらゆる仕様を含んでいる反面，特定の分野の仕事なら小さな部分集合だけでプログラムでき，マニュアルや教育もそのようにわけて構成できる (modularity)．選択仕様を書かずにおくと，自動的に標準仕様が想定されることも，プログラミングを楽にする (default interpretation)．算法言語というよりは実用的なプログラム言語である．

プログラムのブロック構造は，サブルーチン (Procedure)，名前の局所化 (Begin)，コントロール(Do)，番地割付の自由さのためであってそれぞれの限定詞と End とでかこまれる．

変数の割付は，その名前の有意味範囲により外部までひろがったものと局所的なものとがあり，また割付の時期はロード時，呼出時，実行時が区別される．

データの型は連鎖と数がある．連鎖は文字とビット，定長と変長の組合わせがある数は2進と10進，固定小数点と浮動小数点，実数と複素数の組合わせがある．これらのあいだは自由に混用し，また変換できる．データの集りには配列と構造とがあり，演算はいちいちの要素についても，また全体についても可能である．

入出力データの形式は，通常の Fortran 風にプログラムで制御されるもののほかに，データの名前や分離符で区切られた列もあつかえる．入出力関係の多様さと自由さは NPL で最も成功している部分の一つである．

手続や入出力動作を呼ぶときに，実行のきっかけだけ与えておいてけぼりにし，相互に時分割で同時処理できる．これらのあいだの時間的な調整ができる．また金物の割込信号を再生しあるいは模擬して，他方ではこれを割出でうけとる．同時動作の指定をプログラマの手のとどくところにおいたことは従来の言語にみられなかった特徴で，実時間処理などの進んだプログラミングに有用であろう．

プログラママクロのために，言語は2レベルにわけれる．上位のプログラムは翻訳時に実行され，その出力の文字連鎖がソースプログラムとなって普通に翻訳される．

NPL は新しいプログラミング言語であるから，今後も SHARE と IBM とで改訂が進められるだろう．ここに述べたのは第3版である．なお Iverson の著書を参考文献としてあげる．

(西村恕彦)

B-86. 記号表の構成

A. Batson: The Organization of Symbol Tables [C. ACM, Vol. 8 No 2, February. 1965, pp. 111～112]

B 205 で ALGOL コンパイラを作った．名前を定義された順に表におさめ，索表を逆順にすると，ブロック構造の処理に都合がよい．しかし平均して表の半分以上を毎度走査することになる．また30語ある固

有語（reserved word）のチェックも問題である．

そこで IBM 650 SOAP で最初に採用された方式をとった．すなわち，名前の値を乱数化して添字とし，ルック・アットする．あとは見つかるまで，あるいは空白の欄まで順次走査する．これだと最後に見つかったものが目指す名前である，400 語の大きさの表に100語ほど入った状態で20倍ほどはやく処理できた．

（西村恕彦）

B-87．可変長データの保護と操作のための符号構成

C. V. Ramamoorthy: Code Structures for Protection and Manipulation of Variable-Length Items〔C. ACM, Vol. 8 No. 1, January, 1965, pp. 35～38〕

可変長データを表現するのに分離符を入れる方式をとったとき，機械的な故障で機能符号と情報符号とがまぎれるとこまるから，両者のあいだのハミング距離を離しておくとよい．

6ビット1字で情報をあらわすのに，奇パリティの32符号を英字用に，偶パリティの32符号を数字用にわりつけると，従来よりも強い検査がおこなえる．偶パリティの符号を BCD で解釈すると，同じディジットであってゾーンの裏返しのものが対になる．こうして数字について2種類の表現を用意して，分離符，位取りなどに利用する．　（西村恕彦）

B-88．ディジタルシステムをシミュレートするためのプログラム言語

R. M. McClure: A Programming Language for Simulating Digital System.〔J. ACM, Vol. 12, No. 1, January 1965, pp. 14～22〕

ディジタル・システムにおける大きな問題の一つは，雑音問題，不完全回路，布線ミスなどのデバッグである．この論文は，論理設計のデバッグをシミュレートしやすいように，FORTRAN 形式で書くことができるシミュレーション・プログラムである．

このプログラムを作るおもな仮定としては，

（1） 記憶素子（F・F）は J-K, R-S, T, D 型の完全同期型である．

（2） システムは理想化されており，入力，出力，遅延，遷移時間は無視できる．

（3） 入力形式は，設計技術者に便利なように作る．

（4） 論理設計を試験する能力は，hardware において試験した時と同じ能力を持っていること．

（5） データの入力と出力は，できるだけ簡単であること．

などである．

Compiler の構成について詳しいことは，のちに紹介される予定であるが，おもな特徴としては，

（1） top down parse algorithm を使用した．syntax directed compiler を使用している．

（2） 出力は Irons の使用した方法と同じ，parse history から作り出される．

（3） コンパイルした出力は，parsing と交互に行なわれるので，コンパイルできるプログラムの長さに制限はない．

以上の3点である．このプログラムを使用した結果．だいたい初期に仮定した速さと，正確さで，シミュレートできたようである．　（岡　知範）

B-89．可変長レコードに対する最適な固定セル長

E. Wolman: A Fixed Optimum Cell-Size for Records of Variable Lengths.〔J. ACM, Vol. 12, No. 1, January, 1965, pp. 53～70〕.

可変長の通信文を取り扱かうデータ通信系を考える場合，通信文を蓄積する領域をどのように構成するかが大きな問題となる．

この論文では，データ領域をあらかじめ c 語のセルに分割しておき，データの大きさに応じ，このセルをリスト構造によって結んでこれを蓄積する方法を考える．そしてメモリースペースの利用度の面から最も効率の高い最適のセルの大きさについて論ずる．

メモリーの無駄はリストとして各セルをリンクするのに要するセル名（セルの先頭アドレス）を示す部分と最後のセルで使われない部分とからなる．

データ長を l，使用されるセルの数を $n(l)$，セル名の長さを b，セルの大きさを c とすれば，このデータに対する無駄なスペース $w(c)$ は

$$w(c)=\{n(l)c-l\}+b\{n(l)+1\}$$

で表わされる．l が既知の分布に従がう確率変数と考え $w(c)$ の平均をとれば，1データ当たりの無駄の期待値が得られる．この期待値を最少とする c の値を，最適なセル長と定義する．

l が指数分布に従がう場合，最適なセル長 c はほぼ $\sqrt{2bL}$ で与えられる．ここに L は l の平均値である．

l が一般の連続分布に従がう場合にも，指数分布と

ほぼ同様のセル長が得られる.

しかし l が離散的な分布の場合には, $\sqrt{2bL}$ のセル長がほぼ最適となる場合もそうでない場合もあることが, 例証されている. (新井克彦)

B-90. 同じ長さのテープファイルのマージについて

S. Glicksman: Concerning the Merging of Equal Length Tape Files〔J. ACM, Vol. 12, No. 2, April, 1965, pp. 254～258〕

k 本の等長のテープファイルを併合して一組のファイルを作る場合, その所要時間を最小にする方法について論ずる.

この問題については, Burge によって解が得られているが, この論文では (S. r) チェインの概念を導入し, 同じような結論を導いている.

S を k 個の要素からなる集合とし, その disgoint な部分集合による S の分割を考える.

分割の数が k から漸次減少し, 最後には S そのものに到る分割の列 (S_0. S_j,……, S_t) において , S_t の任意の部分集合 I が高々 r 個の S_{i-1} の部分集合を含んでいるものを (S. r) チェインと定義する. 各要素が (S. r) チェインの中で異なる部分集合 (少なくとも2要素を含む) に含まれている数を, その要素の位数, 要素の位数の合計を (S. r) チェインの位数と定義する.

定義から明らかなようにある (S. r) チェインは, 一つの併合パターンを表わし, その位数はそのパターンの所要リール時間を表わす. 位数最小の (S. r) チェインが, 最適併合パターンである.

この論文では, $n=\min\{n;y^n\geq k\}$, $b=[(r^n-1)/(r-1)]$ とするとき, b 個のファイルが n-1 回併合され k-b 個のファイルが n 回併合される場合が最適であることを証明している.

このことは Burge の示した次の結果と一致する.

$$A=(r-1)\bmod(k-1)$$

として A が0か否かにより最初 k または $A+1$ ウェイで併合し以後 k ウェイで併合するのが最適併合パターンである. (新井克彦)

B-91. Syntax-directed Compiler を使った解析微分法

Herbert Schorr: Analytic differentiation using a syntax-directed compiler〔Computer J., Vol. 7 No. 4, January. 1965, pp. 290～298〕

この論文は解析微分法に使用する言語の syntactic definition について述べたものである. definition は BNF (Backus normal form) で与えられており, これに syntax-directed compiler (By E.T.Irons) を使った translation が, 付け加えられている. この transration によって, この言語で記述された任意の代数式の導関数が得られる.

まず基本的な BNF の specification (第1表) によって, 得られた結果が数学的に洗練された型でないと, もとの言語の syntactic definition の変更を導くことになるので, 適当な definition を追加してその結果を改良する(第2表). その方法は次のようにして行なわれる. 第2表を使って得た導関数は正しいが, o*x 等の余分の項が付加されて出てくることがある. これが常数の導関数が0であることから生じたものであると <constant expression> を作り, この項が無視されるよう適当な definition を formation rules の各々に付け加える.

たとえば

<expression>↑<constant expression>::
=<expression>
{$\rho_3\uparrow\rho_1|\rho_{3\cdot 2}*\rho_1*\rho_3\uparrow(\rho_1-1)$}

が第2表に追加されると

(sin (2 x))↑b の結果はいままでの

(sin(2*x))↑b(b*2cos(2*x)/(sin(2*x)+O*ln(sin(2*x)) の代わりに

(2*cos(2*x))*b*(sin(2*x))↑(b−1)

が得られる. このように specification に適当な definition を与えて拡張したものを Appendix として末尾に付記してあるが, 言語の拡張の主眼は次の四つの事柄を導くためのものであり, それぞれについて説明が与えられている.

(1) 高次導関数
(2) 偏微分係数
(3) 陰関数の導関数
(4) 一時に関数の導関数をいくつでも得られること.

例題として y=1/x+sin(a*x↑2) の微分を求めるためのダイヤグラムが与えられている.

(長谷文子)

C-92. 動的番地割付への Content-Address メモリーの応用

Yaohan Chu: Applicaion of Content-Addressed Memory For Dynamic Storage Allocation〔RCA Rev., March, 1965, pp. 140～152〕

この論文は小容量の content-addressed memory（以下 CA メモリーという）を用いた動的番地割付の一例について述べたものである．番地割付の対象となる高速ランダムアクセスメモリー（HS メモリー）は，語とマクロ語の二重構造を持っていて，この中マクロ語を単位として番地割付動作が実行される．一般にプログラムは一連のマクロ語より構成されることになるので，HS メモリ中には，多くのマクロ語の鎖ができる．この配列状態が CA メモリーに登録される．この論文では Executive Program の制御の下に，マクロ語の割付と解放を行ない，命令実行時のアドレス変換が CA メモリーへの問合わせにより，自動的に行なわれる過程について詳しく説明している．

この場合，割付けられたプログラムは，HS メモリー中ではとびとびに配置されるが，他のプログラムからは保護されている．この方法の特徴は，命令で用いられているアドレスの floating code が，番地割付動作によっては全然変化しないことで，その floating code の HS メモリーの絶対番地への変換は，命令実行直前に行なわれることである．この動的番地割付の方法を採用すると，executive control は処理が容易となる．

このメモリーシステムの構造を**第1図**に示す．

HS メモリーはコア，1語 48 bits. サイクルタイム 2 μsec，容量 32,768 語（256 マクロ語）．

CA メモリーは磁性薄膜，1語 32 bits. サイクルタイム 0.5 μsec. 容量 512 語，1語の全ビットまたは部分的な associative read が可能である．

CA メモリーの一語の構造は**第2図**の如くである．

A……対応する HS メモリー中のマクロ語のアドレス

B……Aに続く次のマクロ語のアドレス

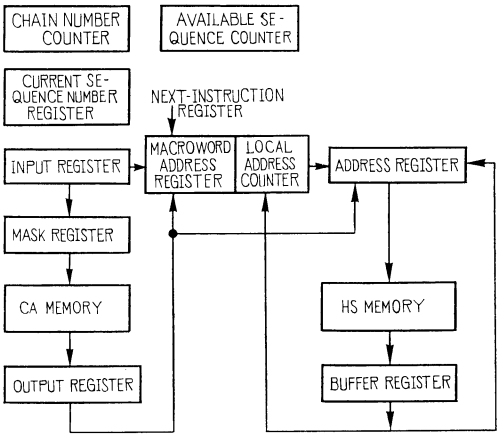
C……1本のプログラムを構成するマクロ語の順番

sequence……各プログラムに付けられた番号

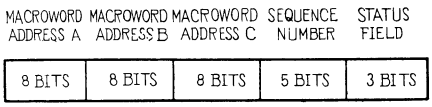
Status……マクロ語の状態表示

（割付済，解放状態，予約中など）

なお，このメモリーシステムの memory protection は CA メモリーの status bit により容易に実行できるほか，Branch, Indexing, Indirect addressing Interruption などの動作は支障なく実行できる．



第1図



第2図

（鍵山圭一郎）

C-93. IBM システム 360 チャンネル設計方式

A. Padegs: The Structure of System/360 Part IV-Channel Design Considerations〔IBM Sys. J., Vol 3. No. 2, 3, 1964, pp. 165～180〕

システム 360 のチャンネルは入出力装置と CPU の同時動作および多重入出力動作を可能にすることを目的として設計されている．このような目的に従い，multi progrom を可能にするため，入出力動作はすべて supervisory program の制御により行なわれ，channel hardware と supervisory prograph により入出力装置の割付け，各プログラムの相互干渉の保護，主記憶装置の保護などの機能を備えている．

システム 360 の入出力動作は Instruction, Command, order により制御される．入出力動作のための Instruction には START I/O, HALT I/O, TEST CHANNEL, TEST I/O の4種類があり，前者の二つで入出力動作の開始および停止を行ない，後者の二つ

で割込み信号の検出およびその内容の解語を行なっている.

Instruction は CPU でデコードされチャンネルプログラムを開始させる. START I/O によりチャンネルは channel command Word (CCW) を受けとり, 入出力動作に必要な情報を得る. CCW の中の, command はチャンネルでデコードされ, 読取りまたは書込みなどの動作を指定する. 入出力装置特有の命令たとえば REWIND などは Order と呼ばれ入出力制御装置におくられ, ここで解語され動作を行なう. 入出力動作の能率を向上させるため, チャンネルは data chaining, command chaining を可能にする機能を備えている. またチャンネルは割込機能により入出力動作の各種状態信号を CPU に送ることができる.

チャンネルには selector channel と multiplexor channel があり前者は高速入出力動作のために設計され data trausfer の間チャンネルは一つの入出力装置を制御する. 後者は多くの低速入出力装置を同時に働かせるものでチャンネル機能を各入出力装置が時分割で使用する. この multiplexor channel には, burst と multiplex との二つの mode があり, 前者は selector と同じ動きをする. CPU とチャンネルの間の data tansfer はワード単位で行なわれ, チャネルと入出力制御装置の間はバイト単位でデータが転送される. チャンネルと入出力制御装置の接続はすべて標準化され, どのような入出力制御装置も接続することが可能になっている. (田中千代治)

D-94. 組合わせ回路および順序回路の Hazard の検出

E. B. Eichelberger: Hazard Detecion in Combinational and Sequential Switching Circuits [IBM J.Res. and Dev. Vol. 9, No. 2, March, 1965, pp. 90～99]

この論文では3値論理代数を使って, 組合わせ回路および順序回路の hazard を検出する方法をのべている.

この方法は次の二つの点で unique である.

(1) 入力変数が同時に二つまたはそれ以上変化した場合にも適用できる.

(2) 組合わせ回路および順序回路の両方に適用できる.

(3) 2,000 個以上の論理ゲートを含む回路を解析するためのプログラムを作って計算機で簡単にしらべることができる.

まずこの方法でとりあっかう各論理ゲートは, 遅れのない理想的な素子と, それらの入力端子に遅延素子がついたものを考える, しかも遅延時間はすべて一定値 $d_{\max}$ より少さいものとする. hazard の定義として, M 個の入力が同時に変化した場合に, 最初の入力状態と最後の入力状態で出力が変化しなくて, しかも途中のある入力状態で noise の出るものを M-hazard という. そのうち真理値表の上で noise の出る組合わせのあるものを function hazard といい, そういう組合わせはないけれども, その関数の実現方法によっては, 出るような場合を logic hazard として区別する.

実際の信号は連続的に変化するものであるから, 分岐のある場合には, その信号を受けているある素子では, 1, また別の素子では0と受けとられる場合が起きる, そこでそのようなことの起きるある範囲を考えて, そこでは論理変数が第3の値1/2をとるものとする. そして最初に与えられた論理関数 f から, 入力変数が1/2になっているものをのぞいて, 出力が1か0に確定しない場合は, その関数の出力も1/2とするような3値の論理関数を考える. この3値の論理関数 f^* を使って hazard 検出のための次の定理が証明されている.

次のように考えられる入力状態AからBへ変化する場合に,

$A=(a_1\cdots a_r,\cdots a_{p+1},\cdots a_n)$,

$B=(b_1\cdots b_p,a_{p+1}\cdots a_n)$,

$A/B=(1/2,\cdots 1/2,\ a_{p+1},\cdots a_n)$.

$f^*(A)=f^*(B)\neq 1/2$ でしかも $f^*(A/B)=1/2$ の場合 f は M-hazard がある.

次に同様にして順序回路を解析する方法がのべられている. (相馬 嵩)

D-95. 超伝導記憶について

L. L. Burns: Cryoelectric Memories [Proc, IEEE. Vol. 52, No. 10, Oct, 1964, pp. 1164～1176]

この論文は, まず超伝導記憶は冷却装置が高価であるために, フェライトコアで実現可能な程度の容量では有効ではないが, 1～10 μsec のサイクルタイムで10億ビットぐらいの記憶容量を可能にするものは, 超伝導以外に考えられないと述べ, 実際には 16 K 語, 各50ビットの記憶装置について検討している.

クライオトロン型の記憶は広い面積が必要になるので適当でなく，連続膜を使用する持続電流方式が好ましい．しかし，これにも欠点があり，駆動電流やスイッチング磁場の大きさをコントロールするのが困難である．駆動回路には，薄膜クライオトロンが考えられている．

鉛で作られる駆動線の幅は，1～10 mil について検討されており，細い方が駆動電流が小さくて済むが出力電圧も低下する．錫で作られる記憶素子は膜の厚さが 1,000～10,000 Å のものが考えられ，膜の厚い方が出力電圧は大きい．

読取方式には，ジグザグ状のストリップを用いる方法と，キャビテイによる方法とがあるが，前者は 16 K 語を直列に接続すると信号遅れが 40 nsec もある．一方，後者は出力電圧が非常に小さくなる．

記憶素子そのものの動作時間はナノセコンド以下とも言われているが，実際には 10 nsec 程度であろう．しかし，駆動線の選択に時間がかかり，読み書きサイクルは普通の回路で 3 μsec，複雑な回路で 1 μsec 程度になるであろう．

製作は 5×10^{-8} torr 程度の高真空中で蒸着によって行なわれる．蒸着のステップ数は 10～30 で，相互間の精度が非常に要求されるので歩留りが悪い．

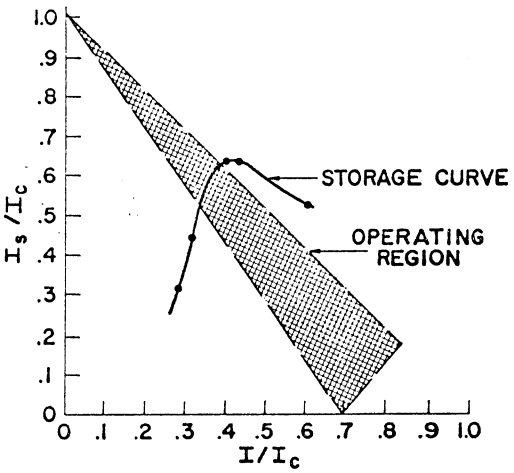
冷却装置は現在高価であるが，量産に入ると 20,000～30,000 ドルぐらいになるであろう．また消費電力は 1～10 ワットであろう．　　（石立　喬）

D-96．超伝導連続膜記憶素子の動作

J. D. Barnard, R. H. Blumberg and H. L. Caswell: Operation of the Cryogenic Continuous Film Memory Cell.〔Proc. IEEE. Vol. 52, No. 10, Oct., 1964, pp. 1177～1181〕

超伝導記憶素子のなかでは連続膜を用いた持続電流方式が最も実現可能性が高いようである．そこで，この論文はこの方式に関して，理想的な特性の場合と実際の特性の場合とについて動作領域を図示し，いかに超伝導記憶素子の動作可能領域が狭く，製造困難であるかを示している．

理想的な場合には，駆動電流を I，記憶されている電流を I_s，遷移電流を I_c とすると，動作原理から**第1図**のような関係が得られる．図で記憶曲線とは，実際の素子について測定した I に対する I_c と I_s の関係を図に表わしたもので，これから $0.33<I/I_c<0.38$ の範囲が動作可動であることがわかる．



第1図

実際の場合には Imax や Imin が半選択電流の回数によって変化する．このことは，半選択電流によって，記憶されている電流の一部が失なわれることを意味する．これを防止するには，内部に十分多数の物理的空所（voids）のある連続膜を用いるとよい．空所は決して超伝導状態にならないので，そこは磁束を強力に引きつけておく場所となる．実験によると，半選択電流を 2,000 回印加すると，10^7 回加えた結果と等しくなる．

超伝導と常伝導との状態の遷移曲線が実際には急峻でないために，動作可能領域はさらに狭められ，場合によっては，動作可能領域がなくなる．

多数の素子を用いて記憶装置を形成するには，各素子の熱的電気的タイムコンスタントが揃っていることが必要である．均一性を高めるには，駆動線の幅の寸法精度をたとえば $\pm2\mu$ にする必要などがあり，まず実現不可能である．連続膜を均一に作ることも非常に困難で，わずかな表面状態の違いが特性を大きく変化させてしまう．　　（石立　喬）

E-97．閾素子による対称ブール函数実現のためのグラフの利用

C. L. Sheng: A Graphical Interpretation of Realization of Symmetric Boolean Functions with Threshold Elements,〔IEEE Trans. EC, EC-14, No. 1, Febrary, 1965, pp. 8～18〕

閾素子の組合わせで対称函数を実現する問題を，折線で囲まれた図形を平行四辺形に分割する問題に置き

換えることにより，最小個数に近い閾素子で実現する方法を述べてある．ただしグラフを利用して見やすくした点は，独創的であるが，実現方法の骨格は既に，Kautz らにより示されている．

対称函数を実現する閾素子の重みは同一素子については，すべて等しくなくてはならない．独立変数の正値のものの個数を横軸に，重み和を縦軸にとったグラフを描くと，閾素子1個の場合は原点を通り勾配1の直線分が得られる．求むる対称函数が，Shannon 記法で表現されていれば，その指標に従ってどの変数が閾値線の上になければならないかがただちにわかる．これを満すように，先の直線を下に下げれば，一般にジグザグな折れ線を得る．この下げる操作が，ある閾素子の出線に負の重みをつけて他の閾素子の入線となることに相当する．このジグザグ折線と元の直線で囲んだ領域は，平行四辺形に分割できるが，この分割の仕方および大きさが，それぞれ閾素子の接続の仕方および重みの値を表わすことになる．平行四辺形の選び方は三つの簡単な制約に従っており，それ以外には任意性があるため，解は一意的でないし，必らずしも最適解ではない．視覚をうまく利用した図形分割によって，30変数をやや上まわる範囲までは，最適解が容易に得られると著者は指摘する．

上の方法でジグザグ折線がP個の歯形を持ったとすると，これを実現する閾素子の最小個数は，$[\log_2 P]+1$ であることは既に Kautz らが指摘している．また最大個数はPである．この間にあって，なるべく素子数を減らすための著者の工夫は，

1) ある素子の禁止重みによってグラフ上の影響，すなわち平行四辺形の高さは等しい．

2) それらの各々は，より大きな平行四辺形で分離されていなくてはならない．

3) 縦軸に平行な一つの直線が上の条件を満す平行四辺形の二つ以上を横切ってはならない．

以上である．これらは図形上で領域を区切る操作を実際に行なうには見やすい条件である．しかし著者も指摘するごとく，100変数以上となると，最適解を得るのは非常にむずかしくなろう．もっとも，現在のところ，かような対称函数が何に利用されるか明らかでない． （伊達 悖）

E-98. 3値論理回路の論理設計

M. Yoeli and G. Rosenfeld: Logical Design of Ternary Switching Circuits 〔IEEE Trans. EC, EC-14, No, 1, February, 1965, pp. 19～29〕

3値論理は自動制御システムや符号検査・訂正回路などの論理回路において有用であることが期待されていながらいまだに実用化されていない．これは第1にスイッチング素子および記憶素子として適当なものがなかったこと，第2に論理設計理論が確立されていないことが原因であるが，素子としては最近でもいくつかの例が発表されている．ここでは論理設計における簡単化の手順を最も普通の和積形式の拡張として展開している．

基本演算としては，従来どおり $x+y=\max(x,y)$ と $x.y=\min(x,y)$ の Post 演算のほかに第1表の4種の1変数関数（および定数1）を用いる．これらはいずれも，ダイオードまたはトランジスタで容易に実現できる関数である．

第1表

x	x^0	x^2	x^{01}	x^{12}
0	2	0	2	0
1	0	0	2	2
2	0	2	0	2

そして任意の3値関数を次のような加法標準形式に展開する．

$$f(x_1,\cdots,x_n)=g(x_1,\cdots,x_n)+1\cdot h(x_1,\cdots,x_n)$$

ただし $g(x_1,\cdots,x_n)=\sum_{f(\alpha)=2} x_1^{a}\cdots x_k^{a}$

$$h(x_1,\cdots,x_n)=\sum_{f(\alpha)=1} x_1^{a}\cdots x_k^{a}$$

$\alpha=(a_1,\cdots,a_n)$

$a_i\in\{0,1,2\}$; $i=1,\cdots,n$

なお x^1 は $x^{01}\cdot x^{12}$ の形で実現する．

関数の包含関係，項と主項（いずれもg型とh型とがある）などを2値の場合の拡張として定義して，図表による簡単化，Quine の方法，Scheiman の方法（展開定理の繰返し適用による近似的方法）などの2値的方法を3値の場合に翻訳して応用する手順を示し例を挙げている．当然のことながら手順は2値の場合よりも複雑となり，実際的には機械化の容易な手順を選ぶ必要があろう． （苗村憲司）

E-99. 線型順序回路の解析と合成

A. Gill: Analysis and Synthesis of Stable linear Sequential Circuits 〔J. ACM. Vol. 12, No. 1, January 1965, pp. 141～149〕

線形順序回路（L.S.C.）は誤り符号の訂正，計算機

制御，乱数発生などに応用されている．この文献では，いわゆる L.S.C. のうちの Autonomous stable L.S.C.につきその性質，実現するための必要十分条件，シフト・レジスタによる回路の構成などにつきのべている．この種 L.S.C. は特に誤り符号の訂正回路としてよく用いられるものである．stable な L.S.C. とは初期状態の如何を問わず，かならず Null state に達するような L.S.C. をいう．L.S.C. の特性行列を A で示せば $A^n=0$ を満足する最小の整数 h が存在するような L.S.C. である．すなわち特性行列がベキ零行列である場合の L.S.C. である．いま A が有理標準形の行列とすれば，つぎのように示すことができる．

$$A=\begin{pmatrix} M_1 & 0 \cdots\cdots 0 \\ 0 & M_2 \cdots 0 \\ \vdots & \vdots \\ 0 & 0 \cdots\cdots M_w \end{pmatrix}$$

そして各部分行列 M_i は

$$M_i=\begin{pmatrix} 0 & 1 & 0 \cdots\cdots 0 \\ 0 & 0 & 1 \cdots\cdots 0 \\ \vdots & \vdots & \vdots \cdots\cdots \vdots \\ 0 & 0 & 0 \cdots\cdots 1 \\ 0 & 0 & 0 \cdots\cdots 0 \end{pmatrix}$$

の形で示される．

stable L.S.C. の state graph はその根が null state である木として表わされる．つぎに i steps で null state に達する state の集合を Ni で示せば

$$Ni=p^{m_1+2m_2+\cdots+(i-1)m_{i-1}+i(m_i+m_{i+1}+\cdots m_h)} -(N_0+N_1+\cdots+N_{i-1})\ (i=1, 2, \cdots, h)$$

となる．

ここで h は部分行列 M_i の最大の正方行列の辺とする．

また $Ki=\log_p(1+N_1+\cdots+N_2)$ で示せば，実現可能な stable L.S.C. の必要十分条件は

（1） K_i が負でない整数であること

$(i=1, 2, \cdots\cdots, h)$

（2） $K_1 \leqq K_2 \leqq 2K_1$

$K_2 \leqq K_3 \leqq 2K_2K_1$

$K_3 \leqq K_4 \leqq 2K_3-K_2$

$\vdots$

$K_{h-1} \leqq K_h \leqq 2K_{h-1}-K_{h-2}$

なる不等式が成立つことである．上でのべた m_i は，（m_i は $i\times i$ の部分行列の個数）

$$m=D\ K$$

ただし

$$K=\begin{pmatrix} K_1 \\ K_2 \\ \vdots \\ K_h \end{pmatrix}=\begin{pmatrix} \log_p(1+N_1) \\ \log_p(1+N_1+N_2) \\ \cdots \\ \log_p(1+N_1+\cdots+N_h) \end{pmatrix}$$

$$m=\begin{pmatrix} m_1 \\ \vdots \\ m_h \end{pmatrix}$$

$$D=\begin{pmatrix} 2 & -1 & 0 & 0 \cdots\cdots 0 & 0 \\ -1 & 2 & -1 & 0 \cdots\cdots 0 & 0 \\ \cdots & -1 & 2 & -1 \cdots\cdots 0 & 0 \\ & & \cdots\cdots & & \\ 0 & 0 & \cdots\cdots & -1 \quad 2 & -1 \\ 0 & 0 & & 0 \quad -1 & 1 \end{pmatrix}$$

で求めることができる．このようにして，求められた L.S.C. は i 個の遅延素子を持った m_i 個の非循環シフト・レジスタで構成できる．ここに K_h は回路で用いられる遅延素子の総数を示す．文献では，構成の一例として GF(2) 上の 5-level tree で示される L.S.C. につき説明している．　（関口正一）

E-100．チューリングマシンの計算機による研究

S. Lin and T. Rado: Computer Study of Turing Machine Problems 〔J. ACM. Vol. 12 No. 2, April, 1965, pp. 196～212〕

この論文は，Turing machine に関する Busy Beaver Problem で state 数が3のものを計算機を使って解いている．問題を標準化するために，0，1の2文字からなる Turing machine を考え，次のようなカードで表わしている．

card 1			
0	1	1	2
1	1	1	3

card 2			
0	1	0	1
1	1	1	2

card 3			
0	1	0	2
1	1	1	0

E-100

各カードが現在の state を表わし，第1欄は現在読んでいる文字，次は書く文字，その次は，シフト命令（0は左，1は右シフト），最後は，次の state を表わす，state 0 は stop state である．問題はこのような3枚のカードで表わされる Turing machine を，最初すべて0が書かれた，両方向に無限に長い Tape からはじめて，ループにならずに止まるものの中で，止った時に Tape 上にある1の数 $\Sigma(3)$ の最大なものを求めることである．この問題はループにならないで

止まるテューリングマシンの中で止まるまでに行なうシフト数の最大もの SH(3) を求めることと同等である．この論文で $\Sigma(3)=6$, $SH(3)=21$ と求められた．

実際の方法として，まず $[4(3+1)]^6$ 個の Turing machine のうち，対称性その他の条件で，82,944 について考えた．まず SH(3) は 21 と予想されていたので 21 回シフトを行なうまでに止ったものをしらべ，残ったものについて繰返しループの型によって分類し，40 種類が得られ，それぞれの動作をしらべて，それらがすべて止まらないことを確かめた．$\Sigma(3)=6$ になるものは 5 個でシフト数が 14 のものが一つ，13 のものが二つ，12 のものが一つ，11 のものが一つあった．また，その中 6 個の 1 が連続しないものが，一つあった．SH(3)=21 のものは一つ，20 のものが二つあった．

このような特別な問題を取上げた動機は，$\Sigma(4)$, $\Sigma(5)$ など……の個々の数についての "effectivc calculability" の形式的な概念が，現在まだ得られてないので，$\Sigma(3)$ や SH(3) を求めて，それについての何か手がかりが得られないかということである．この論文か BB-n enthusiast(n-state Busy Beaver Problem) の参考になれば幸いである． （相馬 嵩）

F-101. 最適制御系

J. M. Nightingale: Practical Optimizing System [Control Engrg. Vol. 11, No. 12, December 1964, pp. 76～81]

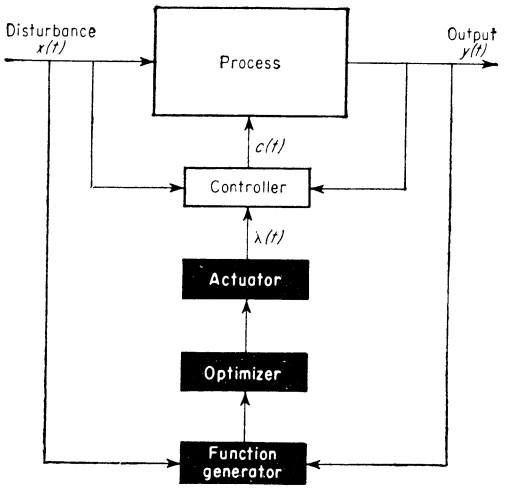


Fig. 1. To optimize the process, the outer loop automatically adjusts λ (t), a parameter in the controller law.

プロセスを最適制御するためには，通常のプロセスの制御ループに，アダプテイブ・ループを重畳するがループ相互間の干渉による．不安定や応答の問題がでてくる．第 1 図に最適制御系の一般形を示す．図において，λ は調節するパラメータである．これの調節ループは，外乱，プロセスの特性などの変化を検出し，あらかじめ，きめた制御目標を得るように λ を調節する．普通制御目標としては，適当な指標 $F(\lambda,t)$ を最大または最小にすることがとられる，最適制御系の設計の基本点としては

（1） 適当な指標の選択
（2） 指標の測定法をどうするか，
（3） 最適化の方法
（4） 安定性

などを考えなければならない．

ここでは第 2 図のような制御系を例にとり設計の一

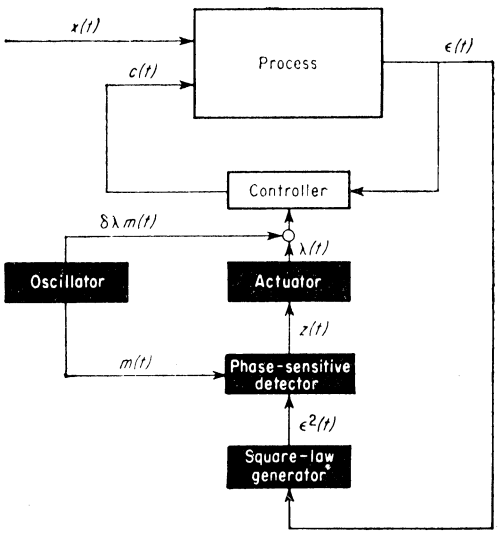


Fig. 2. If a small perturbation $\delta\lambda$ m (t) is added to λ, thesignotth gradient $\partial F/\partial \lambda$ can be determined by synchronous detection.

例を示している．この系は，制御目標は指標を最小にするものとし，指標の符号は勾配測定法で行なっている．発振器は周期的な外乱発生器であり，この信号とプロセス入力とを同期整流して，勾配を求める．また仮定として $\varepsilon(t)$ はガウス分布であり $\varepsilon(t)$ の variance を最小にすることを目標とし，エラーの 2 乗平均は定常状態で λ の双曲線函数で示されるものとしている．これらの仮定のもとに，著者は設計をし

（1） 外乱（発振器出力）の周波数は制御ループの遮断周波数に比して十分小さいこと，実用的にはこの比が1：10以上が必要であること

（2） パラメータ・ループの帯域は外乱周波数より十分小であること

などをのべている．これらの制限を満足するには，パラメータ・ループの帯域を制御ループの帯域の 1/100 くらいにとらねばならぬので，応答が非常に悪くなり過渡応答で問題が出てくるとしている．

（関口正一）

F-102. DDC の Back-up 方法

C. M. Cundall: Backup Methods for DDC 〔Control Engrg. Vol. 12, No. 4, April, 1965, pp. 98～103〕

DDC は従来の計算機制御に用いられてきた計算機よりも，はるかに高い信頼性を要求される．従来の計算機制御の場合には，もし計算機が故障してもプラントの制御は PID 制御装置などで行なわれているので大きな問題はないが，DDC の場合はその故障は殆んどプラントの停止ということになる．ここに DDC の，Backup 装置の必要性がある．どのような Backup 装置を用いるかは，DDC の故障率，修理に要する時間およびプラントの影響の度合などによって，決定される．本文には，Ferranti-Angus 計算機を用いて，Imperial Chemical Industry において DDC を実際に使用した場合の経験から，四つの Backup 方法が説明されている．

DDC で共通に用いられている部分はすべて Check Program（約500語）によって1秒に1回 Check されている．入力装置，たとえば ADC は模擬入力によって Check され，出力装置は，操作器の実際の動きを DDC に読みこみ，それを計算結果と比較して，Check する．この Check 方法によって，DDC に故障が発見されると DDC はプラントからすぐに切離され，それと同時に，操作器には Backup 装置が接続されてプラントの制御は続けられる．Backup 装置を分類すると

（1） Manual Backup; オン・オフ信号で制御弁を制御する最も簡単な方法

（2） Analog Backup; PID 制御装置または特殊制御装置にそのループの制御を切換える方法．この場合は Backup 装置の setpoint と DDC のそれが一致していなければならないこと Back-up 装置の積分項の値が，DDC により計算された操作器の位置と等しくなくてはならないことなど注意しなければならない面が多い．

（3） Simple Digital Backup; の比例制御だけを行なう簡単な Digital Controller による方法で，この Controller に用いられているコア記憶装置には，比例利得，Setpoint，切換えのときに必要な操作器の位置が記憶されており，この内容は DDC の出力によって書きかえることができるようになっている．

（4） Dual Computer Backup; 2台の計算部を並列に用いており，1台の MTBF 1,000 時間，修理に要する時間を2時間とすると，この方法により，MTBF を約25年とすることができる．

ICI での実際使用から，Analog Bockup（1ループ当たり 1,200 ドル）を 25％のループに用意しておけば，DDC の故障時にも充分制御を行なうことができること，Manual Backup（1ループ当たり 200 ドル）はすべてのループに用意しておきたいことが明らかにされた．また Dual Computer Backup は費用の点で問題であると説明している．　（福本昌而）