査回路の素子の数をぼう大にするかのいずれかであった．いずれにしても不経済であるから，パリティ検査をしている並列式計算機というのは稀れである．しかし，ここに紹介した方法によって，経済的で高速な**第12図**の回路が得られる．この回路も継電器回路では公知のものである[8]．

桁移動も並列式計算機では時間のかかる操作の一つであるが，これも同様に高速化できる．シフトの桁数を2進数で表わしたものの各ビットを最下位から S_1, S_2,……, S_m で，左シフト，右シフトを L, R で示すと，2^m-1 桁まで一挙に何桁でもシフトできる回路は**第13図**のようになる．

6. 結　　言

マンチェスタ大学の高速桁上げ回路を追試して，極めて優れたものであることを示し，その基本素子は接点抵抗3Ω程度，スイッチ時間数十mμs 程度（マイクロ・アロイ・トランジスタの場合）の電子的な継電器であることを考察した．またこれを用いることによって従来継電器式計算機で開発されている種々の回路を，殆んどそのまま電子化できることを指摘し，例としてマンチェスタ大学のものよりさらに高速なる加算器，パリティ検査回路，桁移動回路を示した．

トランジスタのこのような使用法は，今後のディジタル技術の進展に極めて大きな影響を与えるものと予想される．トランジスタとしては，筆者等の知る範囲ではマイクロ・アロイ形がいまのところ最適であるが，欲をいえば，耐圧は 5 V 程度でよいが，対称形で，接合部がさらに薄く，飽和時の抵抗が 1Ω 以下の特別なトランジスタが開発されることが望ましい．

擱筆するに当り，討論に参加された回路課淵一博君並びに実験に尽力された実習生辻信義君（安川電機）に御礼申上げる．

参考文献

1) A. Weinberger and J.L. Smith, IRE Trans. EC-5, No. 2, 65 (1956).
2) 室賀三郎，高島堅助：信学誌，**41**，1132（昭33）．
3) 高橋秀俊，後藤英一 他：電気通信学会電子計算機研究専門委員会資料（昭33年9月）．
4) Digital Computer Lab., University of Illinois, Report 80 (1957).
5) 畔柳功芳：信学誌，**42**，1051（昭34）．
6) 田淵誠一：信学誌，**43**，440（昭35）．
7) M. Goto, Y. Komamiya et al.: Res. Rep. ETL No. 556(1956).
8) Y. Komamiya: ibid No. 580 (1959).

Vol. 1 No.1　　情　報　処　理　　July 1960

プログラムによる Digital Differential Analyzer*

高　田　　勝**

まえがき

力学，自動制御その他あらゆる分野にあらわれる常微分方程式の初期値問題を，アナグロ計算機で解くときは，計算の流れが直視的で物理現象の Simulation も容易であって工学向きである．しかし精度，乗除算特殊関数計算の容易さの点ではディジタル型計算機械に及ばない．

一方，ディジタル型ではそのプログラムに普通は機械語の知識を要し，かつ計算の順序を考えて組まねばならない．もしアナログ計算機の特徴をできるだけ残して，精度のよい計算をディジタル型で行うことができれば非常に便利である．ここではアナログ計算機の微分解析機としての機能を，ディジタル型のものでプログラムによって simulate しようとするものである．

このような Digital Differential Analyzer (D.D.A.) の試みは海外ではすでに一，二試みられているが[1,2]，その内部構造は明らかにされていない．また国内でも外国製の計算機械により，最近いくつかの D.D.A. がプログラムされ実用に供されはじめた[3]．これらはほとんどみな解釈ルーチン方式で，計算の行われる順序に D.D.A. で使う Pseudo code（擬命令）を入

* Programmed Digital Differential Analyzer, by Masaru Takada
日本機械学会第37期定期総会講演会（昭和35年4月）および数理科学総合研究「微分方程式の数値計算」に関するシンポジウム（昭和35年4月）で発表，それらの予稿参照．

** 東大工学部応用物理

れる必要があり，それが一語一語機械語に翻訳されながら計算されるようになっている．

ここで述べるプログラムの特徴は，Assembler 方式であって，入れられた擬命令から，計算順序を判定しながら自動的に機械語に翻訳し，計算は普通の場合と同じ速さで行われること，および浮動小数点演算ができることである．したがって，使用者は解こうとする微分方程式

$$\frac{dx^i}{dt}=f^i(x^1, x^2, \cdots\cdots, t)\quad i=1, 2, \cdots\cdots (0.1)$$

の変数に適当な成分番号を割当てて，式のとおりに擬命令を組立てて入れてやればよく，計算順序を考える必要も，スケーリングの必要もない．

このようにすれば，プログラムはアナコンと同様あるいはそれ以上の容易さででき，内部計算はディジタル型で微分方程式を数値計算することと全く同じになる．

1．主な機能と制限

使用した機械は電子工業振興協会設置の JEIDAC-101 で，容量が 2,000 語の磁気ドラム式記憶装置と 40 語の一時高速書込み読出し装置をもち，1 語は10進法11桁の数値と符号の12ビットからなる．これは 6 ビットの二つの命令としても使われる．数値は固定小数点方式で使うときは小数点が符号のつぎつまり第 1 の数字の前にあるものとみなされ，浮動小数点方式であればつぎの形をとる．

$$\overbrace{\pm\square\square}^{2}\overbrace{\times\times\times\times\times\times\times\times\times}^{9}$$

ここで□□は指数部，10^a の a に 50 を代数的に加えたもの，つまり $50+a$．　×××…… は数値部で 9 桁．小数点は数値部の最初の数字の前にある．

先に述べたように，D.D.A. での演算中の数値はすべて浮動小数点方式で取扱われるので，読みこみには上の形式で入れてやる必要がある．

アナログ計算機と同じように次の成分をもつ．

積分器，積和器，除算器，陰関数発生器，関数発生器，遅延器，リレー，定数（ポテンショメーター），プリンタ．

これらの数は一定しておらず，全擬命令の和が 100 以下，全定数の数が 50 以下，という制限をみたせば任意でよい．その他若干の制限があるが，最もきびしいのはこれらの制限である．

以上は機械語になおす上での制限で，演算上の収束性，精度よりする制限は別問題である．

また印刷は欄指定，固定小数点形式（桁数指定も可能）か，浮動小数点式かの指定ができる．時間おくれは全体のおくれ数が 300 ステップ以下ならば任意の数ができる．

数値積分は梯形則を修正子とするつぎの予測子修正子法によった．ただし h はきざみ幅である．

$$P\quad\begin{cases}x_1=x_0+h\dot{x}_0 & (t=h,\ \text{あるいはきざみ幅変更のとき})\\ x_{n+1}=x_{n-1}+2h\dot{x}_n+T,\quad T=\dfrac{h^3}{3}\dddot{x},\end{cases}$$

$$C\quad x_n{}^{+}{}_1=x_n+\frac{h}{2}(\dot{x}_n+\dot{x}_{n+1})+T,\quad T=-\frac{h^3}{12}\dddot{x}$$

（1・1）

この方式では途中の打切り誤差を見積ることも可能である．

2．擬命令，文法

ここで用いる擬命令は±の符号のほか11桁の数字よりなり，次の形をもつ．

$$\pm a_0a_1a_2\ b_0b_1b_2\ \alpha\ c_0c_1c_2\ r$$

はじめの 3 桁，$a(a_0a_1a_2)$ はその成分の機能を表わす $a_0(0\sim9)$ とその番号を示す 2 桁の数 a_1a_2 とからなり，$b(b_0b_1b_2)$ と $c(c_0c_1c_2)$ は，ともにその成分 a への入力となる部分の成分およびその番号を示す．α は 0 または 1 で，0 のときは普通 $c=0$，$\alpha=1$ のときは b 部と c 部の積 bc を入力とするか，あるいはその他 a_0 によってちがった意味をもつ．r はリレーに a の出力をつなぐときに用い，1～9の値をとる．具体的には第 2・1 表のようになる．なお $a_0=3$ は未定で，使用者の工夫にまかした空地である．

この表で少し説明を補足しておく．

4-code　$4\,a_1a_2\ b_0b_1b_2 1c_0c_1c_2\ r$

では b として $f(x)=0$ を Newton-Raphson 式に

$$x^{(k+1)}=x^{(k)}-f(x^{(k)})/f'(x^{(k)})$$

で計算するときの第 2 項 f/f' をとる．c にははじめの試みの値が入る．これは最初に他の定数とともに読みこんだ 8-code が割当てられ，途中では各ステップでの一つ前の x の値が入るようになっている．収束の判定はあとで述べる収束判定用の小さな値 δf を用いている．

7-code　$+70\ r\ b_0b_1b_2 1c_0c_1c_2\ r'$,

$-70\ r\ b_0b_1b_2 1c_0c_1c_2\ r'$

リレーはこれにつながれている成分の内容の値如何でその出力を b としたり c としたりする．たとえば，

第　1　表

成分別	擬命令	機能
	入力 b　α　入力 c　r	
積分器	± 0 a_1a_2　$b_0b_1b_2$　1　$c_0c_1c_2$　r	$\pm\int bc\,dt$
($a_0=0$)	± 0 a_1a_2　$b_0b_1b_2$　0　0　0　0　r	$\pm\int b\,dt$
積和器	± 1 a_1a_2　$b_0b_1b_2$　1　$c_0c_1c_2$　r	±bc
($a_0=1$)	± 1 a_1a_2　$b_0b_1b_2$　0　0　0　0　r	±b
除算器	± 2 a_1a_2　$b_0b_1b_2$　1　$c_0c_1c_2$　r	±b/c
($a_0=2$)		
(未　定)	± 3 a_1a_2　$b_0b_1b_2$　0　0　0　0　r	未定
($a_0=3$)		
関数発生器(陰関数型)	＋ 4 a_1a_2　$b_0b_1b_2$　1　$c_0c_1c_2$　r	$f(x)=0$ より x を出す
($a_0=4$)		別記
関数発生器	± 5 a_1a_2　$b_0b_1b_2$　0　0　0　f　r	$\pm f(b)$　f: function の種類 (0, 1, 2……)
($a_0=5$)		
遅延器	＋ 6 ××　$b_0b_1b_2$　0　$c_0c_1c_2$　r	$+b(t-c\Delta t)$　$\Sigma(c)_i\leqq 300$
($a_0=6$)		
リレー	＋ 7 0　r　$b_0b_1b_2$　1　$c_0c_1c_2$　r′	(r)≧0: 出力は b　r, r′: 1～9
		＜ : 〃　c　(r): つながれた成分の内容
($a_0=7$)	－ 7 0　r　$b_0b_1b_2$　1　$c_0c_1c_2$　r′	(r)=0: 〃　b
		≠ : 〃　c
独立変数	＋ 8 0　0　0 0 0　0　$c_0c_1c_2$　r	独立変数　c: プリント回数指定 (cきざみごとにプリント)
($a=800$)		
定数	＋ 8 a_1a_2　0 0 0 0　$c_0c_1c_2$　r	定数　c: 定数の数　$c\leqq 50$
($a_0=8$)		
プリンター	＋ 9 a_1a_2　$b_0b_1b_2$　□　$d_0d_1d_2$　d_3	(b) を印刷　a_1a_2: 欄指定
($a_0=9$)		d_0d_1: 小数点以上の桁数
		d_2d_3: 全桁数
		詳細別記

7-code の前の方のどこかに

$\bar{a}_0\bar{a}_1\bar{a}_2$　$\bar{b}_0\bar{b}_1\bar{b}_2$　1　$\bar{c}_0\bar{c}_1\bar{c}_2$　r

という同じ r をつけたものがあると，この ā の内容の如何で，b あるいは c がリレーの出力となる．r は 9 個だけ使用できる．このリレーをうまくつかえばヒステリシスも作ることができる．

9-code　欄指定 a_1 a_2 で次のような形式に印刷される．

00　01　02……05
06……10
……

なお □ は b 部を続けていくつ印刷するかを示すもので，例えば

9010003　$d_0d_1d_2d_3$

では，成分 000 等を 01 欄から逆順に次のようにうつ．

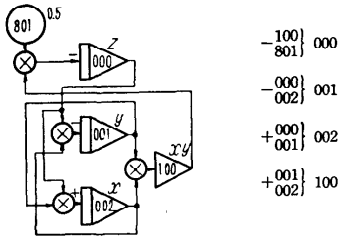
(003)　(002)　(001)　(000)

また d=0000 では浮動小数点形式で出す．固定小数点の指定で，小数以上の桁数が指定値を上まわれば浮動小数点形式で印刷する．

これらの擬命令を a_0 の小さいものから大きい順に，同じ a_0 のものでは番号 a_1a_2 の大きいものを（最大のものを一番）後にするだけでよい．そして一番おしまいに 0Z とおく．なお，定数，初期値，$a_0=5$ のサブルーチン（たとえば平方根，立方根）などは別途に読みこませる．数値はすべて浮動小数点形式で，機械特有の制御指令をつけるが，これは後で例によって説明する．

〔例〕　$\dot{x}=yz$, $\dot{y}=-zx$, $\dot{z}=-(1/2)xy$,
$t=0$ で $x=0$, $y=1$, $z=1$.

きざみ幅 $h=0.0025$ にとり，20 ステップごとにプリントする．t は 0 欄，x, y, z はこの順に第 1, 2, 3



第 1 図　ブロックダイアグラム

欄に，t は小数以上2桁，全体を5桁，その他は小数以上2桁，全体を9桁印刷することにする．積分器の割当ては z, y, x の順にそれぞれ 000, 001, 002 を当てる．$\dot{z}$ には積和器を要する．

定数 0.5 には 801 をあてると第1図のようなアナログ式のブロックダイヤグラムができ，これに対してその右のような擬命令で，代用のダイヤグラムができる．これは式そのものを表わしていると見てよい．これから擬命令は直ちに次のようにかくことができる．

```
−000 100 1 801 0
−001 000 1 002 0
 002 000 1 001 0
 100 001 1 002 0
 800 000 0 040 0
 801 000 0 001 0
 900 800 0 020 5
 901 000 2 020 9
          0Z
```

もし同じ要素に入力となるものがいくつもあるときはそれをならべる．

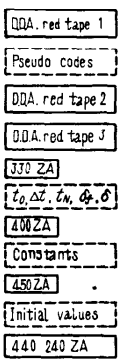
例へば，この例で，$\dot{z}=-(1/2)xy+z$ となっていたとすると，第1行は

```
−000 100 1 801 0
 000 000 0 000 0
```

とすればよい．

3. 使 用 法

初期値や定数の入れ方など実際に使うときのテープの配列は第2図に示すとおりである．図中□はいわゆる red tape で，すべての問題に共通であり，⋯の部分が各問題に応じて作るものである．記号を説明するとつぎのとおりである．



第2図
テープの編集

t_0（原点），Δt（$=h$，数値積分のきざみ幅），t_N（積分の終点），δ_f（収束判定用の数値で浮動小数点表示），δ（同じく固定小数点表示）．

ZA とあるのは機械の制御指令で，××××ZA は以下の数値または命令を××××番地以降に読みこんで記憶せよという意味をもつ．

また，t_0, Δt などの数値はすべて ZD または ZI という制御指令をそのすぐ後につける．ZD では，その前に書かれた数値が，記憶装置の頭の方につめて入れられ，ZI では後の方からつめて入れられる．

〔例〕 前の例では $t_0=0$，$\Delta t=0.0025$，$t_N=1.0$ とし，数値計算上の判断から $\delta_f=10^{-8}$，$\delta=10^{-9}$* としたとすると，

```
330 ZA ZD 4825 ZD 511 ZD 431 ZD 100 ZI
```

となる．また定数 0.5 と，初期値は

```
400 ZA 505 ZD
450 ZA 511 ZD 511 ZD ZD
```

とする．図中 440 240 ZE とあるのは，読込み終れば一度止り，Restart ボタンをおせば 240 番地(積分ルーチンの格納開始番地）から演算がはじまる意味である．

330 ZA 以下のデータのテープと擬命令は別のテープにしておく．そこで，まず red tape 1 を読みこむこんで，擬命令のテープをかけ 700 番地からの命令によりこの擬命令を読みこむ．次に red tape 2 を読みこんで 800 番地からはじめると擬命令から f(x, t) のプログラムの編集が行われ 660 番地で止る．そこで Restart ボタンをおせばタイプのコードを編集して全部の準備が終る．そこで再び red tape 3 およびデータあるいは関数計算に必要なサブルーチンを読みこんでいよいよ演算が開始できる．なお遅延器を使うときはデータテープの一番おしまいに，その初期値を浮動小数点表示で ××××Z××××Z⋯⋯ のように 11 桁までの数値と Z とを使つた遅延器の数だけパンチしておく必要がある．初期値などを変更したいときは，330 ZA 以下を変更すればよいが，Pseudo code をかえるときははじめからやり直す必要がある．

4. 内部構造のあらまし（付録流れ図参照）

この D.D.A, の内部は次の三つに大別される，

（1） 擬命令の読みこみと Table I, II の作製．

（2） 計算順序（編集順序）の判定と，プログラム編集．

（3） 数値積分のルーチン．

(1) は擬命令を 1800～1899 番地に入れ，積分器等使用成分の数などデータをとって 1770～1799 に記憶する．つぎに各成分の各番号ごとに実際の番地××××を割当てて，次のような形で 1900 番地以下に表 (II) として記憶する．この各語の内容を A で表わす．

$$\underbrace{\times\times\times\times}_{[A]_0}\ \underbrace{0000}_{[A]}\ \underbrace{000}_{\text{語数}}$$

積分器 000 からはじまり，000 に対しては，450+

* δ_f と δ とは必ずしも一致させていない．まず δ で収束の判別をし，非常に小さい0に近い値の収束判別に δ_f を用いる．

$3n$（n は元数，つまり使用積分器の数）が割当てられ，以下 001 に対しては $450+3n+1$ …… とする．これらのアドレスが各語の先頭に上のように××××（これを $[A]_0$ で表わす）としておかれる．そして編集されるごとに，その成分が編集されて植えつけられたプログラム上の番地（4桁，これを $[A]_1$ の記号で示す）と使った語数（3桁）とを，のこりの0の所に書きこむ．なお 7-code に対するものは表をつくった直後編集のはじまる前に次の形にしておく．

A：×××× ×××× ×××
- $[A]_0$ ― 7-code の出力位置
- $[\bar{A}]_0$ ― $\bar{a}$ の出力位置（実際の $\bar{a}$ の番地）
- 7-code への入力となる成分 $\bar{a}$（末尾に r をつけた成分）

なお Table I には各成分種類別に，その先頭番号のものに関する情報が入れてある Table II の番地を書きこむ．編集に際しては a_0 に応じてこの Table I を引き，つぎにその内容を a_1a_2 で修飾して Tabl II を引く．

(2) では，入力部の b_0, c_0 が 0, 6, 8 であるものが先ず編集可能であるから，これらをふくむ a が第一に編集されて，その a が既知として a_p の表中に登録される．もちろん同じ a 中のすべての b, c が既知として登録されたものでなければならない．編集された擬命令は消去されて未編集のものと区別される．この編集は使用成分中 $a_0=7$ 以前のもので最大のものからはじめて $a_0=0$ の手前までをくりかえしスィープして可能なものから逐次編集する．$a_0=1\sim7$ がすべて既知となると，$a_0=0$ の編集にうつる．これらの命令は1001地番以降に植えつけられ，全部を編集し終れば積分ルーチンとのつなぎ命令をこの編集プログラムの両端に植えつけて終了する．ここで (SCC) が 660 となって停止するが，再びスタートするとタイプコードを 350 番地以降に編集し，すべての編集が終る．

もし $a_0=4$ の命令があると，相続く2回のスィープでの未編集コードの数 n_u が同じになるので，そのときは仮にこの 4-code を既知として登録し，陰関数型の方程式の根の推定値を入れる命令を編集する．この後で最初に編集された 4-code をふくむ擬命令の編集位置（アドレス）を記憶しておく．これが反復計算で収束判定の結果もどるべき位置となる．そして編集を続けて f/f' の成分のものが編集されると，4-code の収束判定部分が編集されることになる．

以上が編集のあらましである．

(3) は数値積分ルーチンで，Master, Predictor, Corrector の3部分からなる．Master では必要な初期値その他の数値移動が行われて後一般の積分ルーチンに入る．

以上が内部のあらましである．面倒であったのは，Table の作製，計算順序の判定，タイプのルーチンであった．あらすじの設計に約3週間，細部設計と，'ハンダ附け'に約1.5箇月，'ハンダ附けの修正'，など全部で3.5箇月で一応でき上った．

しかし化学反応の問題などは平衡の条件が陰関数型となるので，4-code を附け加えてそれに対処したりなどしたのでさらに時間がかかった．編集に必要な命令数約 1,200（長語で 600），他に表や作業用地として 100～300 長語を要している．その他積分器本体とプリンター用に約 430（長語で 215）の命令を必要としている．これら所謂 red tape の読みこみには約1分30秒を要する．

5. むすび

この D.D.A. を用いて，いろいろの例題の他に実際問題として化学反応（U_{235} の抽出）の問題を解いた．これは陰関数型の関数計算を必要とする5元連立常微分方程式の初期値問題で，擬命令は印刷命令をふくめ67で，機械による編集に1分35秒かかり，編集された機械語での長さは104長語になった．20ステップごとに計算結果を6項目宛プリントさせて，1回に計算時間2分10秒，プリントに約20秒かかった．

この D.D.A. によるときは式の中の変数や定数を成分番号でおきかえ，それを書き並べればよく，アナログ計算機使用者に限らず誰でも容易に使えると思う，既在は計算時間の点でアナログ式に劣るが，将来高速度の機械が得られた場合には非常な利点を発揮するものと考えられる．

このプログラム作製に当っては逐次追加した code があるので a_0 の割当て方をもっと合理的にすべき点があるし，加減算の多いものでは命令編集上 no effect の命令が多くなるので，プログラム圧縮のプログラムがのぞましい．そのほか Runge-Kutta-Gill 法による積分ルーチンも計画しているが，このまま充分実用になるので，実際問題への応用を主としている．なお，3-address 方式の代数式計算にも用いることができる．

このプログラムの完成には機械時間を随分とって日本電子工業振興協会の方々に非常に御迷惑をおかけした．ここに深く謝意を表する次第である．

参考文献

1) R.G. Selfridge, Proc. of Western Computer Conference (1955), p. 82～84.

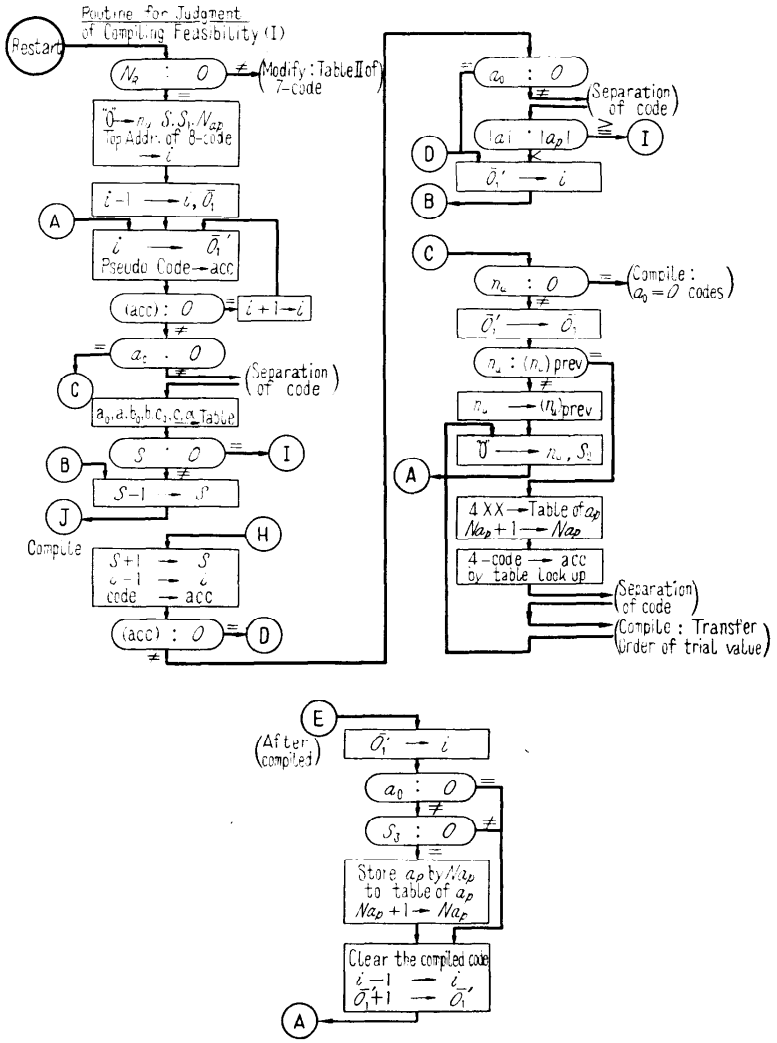
2) F. Lesh, Journ. of A.C.M. 5 (1958), p. 281～288.

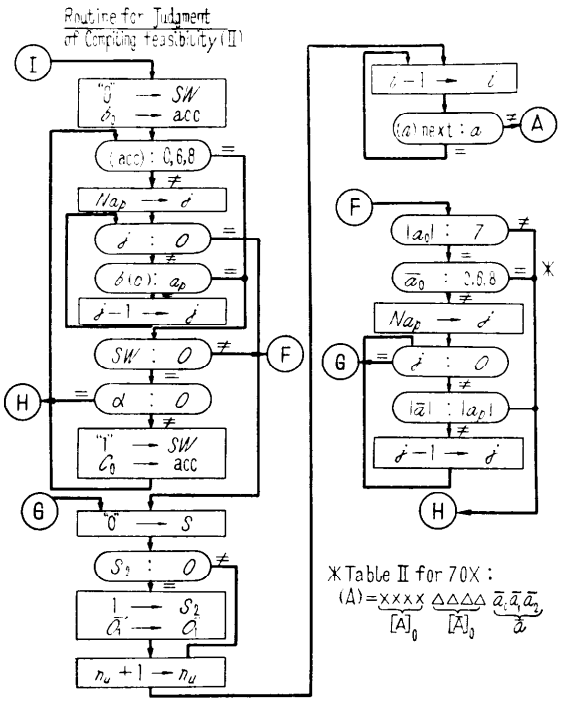
3) 玄地 宏，青木克忠：微分方式の数値計算に関するシンポジウム提出資料，1960年4月，その他，田村康男氏他の資料.

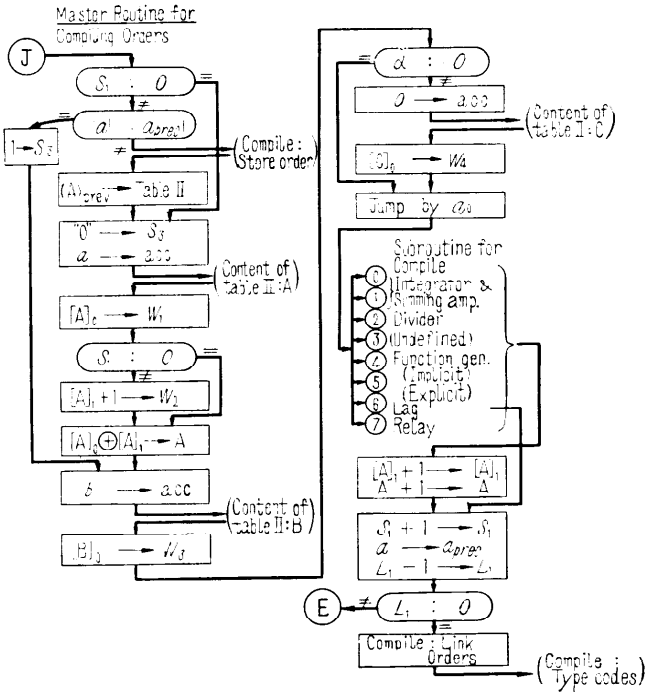
〔附録〕 流 れ 図.

主な記号の説明をすると

N_R はリレーの数，N_u は編集不能コードの数，S, S_1, S_2, S_3 等は判断のためのスイッチ，Na_p は編集され登録された a（これを a_p）の数，$(N_u)_{prev}$ は前回での 7～1 コードの1回のスィープで，解読不能であったコードの数（これが0でなくて続けて等しいときは 4-code がある），A は先にのべた Table II の内容で，その各部は4節でのべたとおり，b, c に対するものはそれぞれ B, C で表わす．L_1 は 0～7 コードの全数.




Routine for Judgment of Compiling feasibility (II)
I
"0" → SW
δ_2 → acc
(acc) : 0,6,8
Nap → i
i : 0
δ(i) : ap
i − 1 → i
SW : 0
F
H
α : 0
"1" → SW
C_0 → acc
G
"0" → S
S_2 : 0
1 → S_2
C_1 → C_1
n_u + 1 → n_u
i − 1 → i
(a) next : a
A
F
|a_0| : 7
$\bar{a}_0$: 0,6,8
Nap → i
G
i : 0
|$\bar{a}$| : |a_0|
i − 1 → i
H
*Table II for 70X :
(A) = XXXX ΔΔΔΔ $\bar{a}_i\bar{a}_j\bar{a}_k$
$[A]_0$ $[A]_0$ $\bar{a}$


Master Routine for Compiling Orders
J
S_1 : 0
a : a_{prev}
1 → S_1
Compile : Store order
$(A)_{prev}$ → Table II
"0" → S_3
a → acc
Content of Table II : A
$[A]_c$ → W_1
S_1 : 0
$[A]_1$ + 1 → W_2
$[A]_2 \oplus [A]_1$ → A
b → acc
Content of Table II : B
$[B]_3$ → W_3
α : 0
0 → acc
Content of Table I : C
$[C]_2$ → W_4
Jump by a_0
Subroutine for Compile
0 Integrator & Summing amp.
1 Divider
2 (Undefined)
3 Function gen. (Implicit)
4 (Explicit)
5 Lag
6 Relay
7
$[A]_1$ + 1 → $[A]_1$
Δ + 1 → Δ
S_1 + 1 → S_1
a → a_{prev}
L_1 − 1 → L_1
E
L_1 : 0
Compile : Link Orders
Compile : Type codes