

## 代数方程式を解く

総合的なプログラム・パッケージについて。

早大・理工 松村 重宏

### 1. はじめに

現在、何らかの計算システムを有する計算センターには、サブルーチン・ライブラリまたはパッケージと呼ばれているものが用意されている。しかし、その多くは、単体で用いられるサブルーチンの集合体であり、パッケージ全体でひとつの、あるシステムを組織するものではない。この報告の目的は、サブルーチン・パッケージを、単なるプログラムの集合体にとどまらせることなく、代数方程式を解くことに関するプログラムを有機的に組み立て、代数方程式のすべての根を求める総合的なプログラム・パッケージを紹介するところにある。

### 2. 設計方針

このパッケージは RÖÖTS-PACK と呼び、Bairstow-Hitchcock法に関する一連の数値実験により得られた結果を参考にして作成した。RÖÖTS-PACKは、サブルーチンを単体で使用で

きるといふ従来のパッケージの機能に加え、

1. 使用法が解法に依存しない

2. 信頼度の高い解を与え、同時にその評価をする。

ということを作成の目標においた。本来、パッケージは、入出力は、できるだけ簡素化するべきものである。しかし、その与える解は信頼のおけるものでなければならぬ。

RÖÖTS-PACKのプログラム本体は、7のFÖRTRAN IV言語と、3つのIBM System/370アセンブラ言語で書かれており、FÖRTRANプログラムから、CALL文によって制御の受け渡しをすることができる。RÖÖTS-PACKで採用している、あるサブルーチンでは、係数配列、解の入る配列以外にも、計算のための配列を必要とするものがある。FÖRTRANでは、整合寸法機能と呼ばれるものがついてはいるが、これは単に、配列の重ねあわせの機能に過ぎない。知っての通り、FÖRTRANの記憶域はすべてSTATICであるため、計算用の配列を別に必要とするサブルーチンを使用する場合、ユーザーは、あらかじめ配列を用意しておかなければならない。RÖÖTS-PACKでは、仮想主記憶の制御を行ない、自動的に記憶域の割り当て、並びに開放を行なうため、メイン・エントリーより制御の受け渡しをするがぎり、CALLするプログラムで、それらを特別に用意する必要はない。FÖRTRANよりこのパッケージを使用するに

は、

CALL ROTPAC(N,A,X,R)

N:方程式の次数

A:係数配列

X:解の入る配列

R:評価値の入る配列

と書きさえすればよい。ただし、ROTPAC (ROOTS-PACKのメインエントリー) を引用する前に、引数の型宣言をしておかなければならないのは言うまでもない。

### 3. 解法

#### 3.1 Bairstow-Hitchcock法に関する考察

Bairstow-Hitchcock法(以下B-H法と略す。)は、多項式

$$P(x) = x^n + a_1 x^{n-1} + \dots + a_n$$

を2次式  $x^2 + px + q$  で整除して、次のようにおく。

$$P(x) = (x^2 + px + q)(x^{n-2} + b_1 x^{n-3} + \dots + b_{n-2}) \\ + b_{n-1}(x+p) + b_n$$

$b_{n-1}, b_n$  はともに  $p, q$  の関数で、

$$b_{n-1}(p, q) = 0.0$$

$$b_n(p, q) = 0.0$$

を満たす  $p, q$  の値を2変数Newton法により求めようとするものである。実係数代数方程式の根を求める問題に対して、

B-H法は、実数で反復内のすべての計算を行なうことができ、しかも収束回数も2次であるため、有効な方法であると考えられる。しかし、 $p, q$ の初期値の設定が悪い時などは収束しないと思われがちである。しかし、B-H法が収束しない原因は別の所にある。

・方程式を解く前に

$$\text{B-H法は} \quad \begin{cases} p_{i+1} = p_i + \Delta p_i \\ q_{i+1} = q_i + \Delta q_i \end{cases}$$

により順次  $p, q$ の値を修正し  $\max(|\Delta p|, |\Delta q|) \leq \epsilon$  により収束判定をする。少し極端な例を考えてみよう。求めるべき根のオーダーが  $10^{15}$  といった方程式に対し、倍精度(10進16桁)の演算を行ない、収束判定の定数  $\epsilon = 10^{-15}$  と設定したとしよう。従来の考え方ならば、初期値の設定さえ良ければ収束するはずである。しかし、実際の計算では、桁の制限や丸め誤差の限界があり、このような根をもつ方程式に対しても運よく収束判定条件を満たす場合もあるだろうが、多くの場合それは望めない。また、それは無駄である。例えば、仮数部が16桁の浮動小数点演算で実行する場合、根  $x$  の指数部を  $E(x)$  で示すとすると、計算結果の丸め誤差は  $0.5 \times 10^{E(x)-16}$  で表わされる。つまり、これより厳しい収束判定条件を設けても意味がない訳である。反対に求めるべき根のオーダーが  $10^{-20}$  といっ

E による極端に小さい問題に対し、前と同じように  $\epsilon=10^{-15}$  と設定し、収束判定を行なえば、本来は根としてはならないような数値であっても、収束判定をパスしてしまう恐れもある。このようなニセの根により、順次次数低下を行なっても、最終的に得られる数値は本来の根とは似ても似つかぬものになってしまう。形のうえでは収束した状態ではあるが、得られた解の信頼度はゼロである。収束性に関する話は多く耳にするが、収束判定に関する話はそれほど多くない。しかし、当然のことであるが、それはある場面に於いては、収束に関して、あるいは解の信頼性に関して非常に重要な要素となる。反復の停止条件は、方程式あるいは根に対して何らかの意味を持つものでなければならぬ。まやみに厳し過ぎても良くないし、またゆるすぎても良くない。そういう意味で、停止条件としては相対誤差評価を用いるのが良い。RÖÖTS-PACK に用いた B-H 法では、

$$\max(|\Delta p|, |\Delta q|) \leq \max(|p|, |q|) \times 10^{-12}$$

を停止条件とした。

しかし、実際問題として、これだけでは不十分である。と、言うのは、前述した極端な問題では、オーバーフローエラー又はアンダーフローエラーを起こすことが考えられるからである。そのような事態の起こることが予想される問題では、

係数変換(スケーリング)を行なうべきであろう。与えられた多項式の係数を  $x=ky$  とおいて変換する。 $k$  の値は, 計算機の内部表示のことを考えれば,  $k=16$  又は  $k=16^4$  と定めるのがよい。この変換により, 変換後の係数は  $b_m = k^m a_m$  ( $m=1, \dots, n$ ) となる。強力な解法を研究するのも, 数値計算の一つの方向であるが, 解く以前の環境の整備も重要であろう。

#### ・収束しない問題

B-H法が収束しない原因は,  $p, q$  の初期値の与え方に依存するものではないようである。根の状態と収束の状態を比べてみると, 複素平面内に, 絶対値が0に近い順に実根, 実根...と存在する時は例外なく収束する。収束しない問題は, 根の状態が, 実根, 虚根, ... となっているものである。すべてが収束しない訳ではないが, 収束しない問題はすべてこのタイプのようなものである。このような収束しない方程式に対して反復ごとの  $p, q$  の値の変化の状態をみると, 振動している。Eに振動といっても, 単一振動ではなく, 1変数Newton法の谷越えの時起こる振動によく似ている。また, おもしろいことに, この振動状態で, 適当な  $p, q$  の値を引用し, それらを係数とする2次方程式  $x^2 + px + q = 0.0$  を解いてやると, 原点に近い1実根と, 虚根の実部に値が似ている実根が得られる。

#### ・収束改善策

B-H法において、収束しない状態となるのは、根のペアの組み方が問題である。上の述べた例の他に、根のオーダーが極端に違うようなものに対しても、収束しないということは十分考えられることである。実験では、収束しなかった方程式に対して、0(原点)を根として方程式に付加して、次数を1つあげてB-H法を適用したところ、収束しなかったものが、見事に収束した。次ページの問題は収束しなかったものの一つである。一番初めの根が分離できない状態である。その時の $p, q, \epsilon$ 等の値も出力してみた。 $\epsilon = \text{EPSILON}$ ,  $\text{MOVE}$ は補正量、 $B$ は $f_{n-2}$ と $f_{n-1}$ の値、 $\text{INDEX}$ と $\text{SCALE}$ はスクーリングに関する情報である。その下の2つの数値は、この状態で $x^2+px+q=0.0$ を解いた時の根である。下の方の数値と解 $X(1)$ は同じ値になっていることに注目していただきたい。

次数を1つあげた方程式を

$$\psi(x) = x^{n+1} + a_1 x^n + \dots + a_n x = 0.0$$

とすると、 $\psi(x)$ の根は、多重根を含め $(n+1)$ 個であるが、そのうち1つは必ず $0.0$ である。このことにより、B-H法の2次因数を $(x^2+px)$ とおき、 $\psi(x)$ を整除する。しかし、 $\psi(x) = xP(x)$ であるため、 $(x^2+px)$ で $\psi(x)$ を整除することは、 $(x+p)$ で $P(x)$ を整除することに等しい。つまり、 $\psi(x)$ に対するB-H法は、 $P(x) = 0.0$ に対する1変数Newton法に等しい。一見、次数を上

$$X + A(1)*X**(N-1) + \dots + A(N) = 0.0$$

10

A( 1 ) = -0.2000000000000000D 05  
 A( 2 ) = 0.9999999999999999D 08  
 A( 3 ) = -0.2140000000000000D 09  
 A( 4 ) = 0.2260000000000000D 08  
 A( 5 ) = -0.7320000000000000D 06  
 A( 6 ) = 0.3410000000000000D 04  
 A( 7 ) = 0.1240000000000000D 03  
 A( 8 ) = -0.3530000000000000D-04  
 A( 9 ) = 0.2440000000000000D-11  
 A( 10 ) = 0.1150000000000000D-19

◀係数

CONVERGENCE:

THE 1-TH ROOT HAS NOT BEEN DEFINED AFTER 100 ITERATIONS. - P,Q は、収束しなかった時の値.

THE LAST VALUE OF P, Q, EPS, B & SCALE:

P	Q	EPSILON	MOVE	B	SCALE
0.1321757D-06	-0.6045124D-15	0.143D-18	0.112D-07	-0.175D-04	0.895D-14
INDEX	SCALE	収束判定値	移動(補正)量		おま。
0	0.10000E 01				

ROOTS OF THE EQUATION EXPRESSED IN THE FORM OF  $X^2 + P*X + Q = 0.0$  :

0.1366011251568242D-06 , 0.0	*I )
-0.4425384117348719D-08 , 0.0	*I )

1  $x^2 + px + q = 0.0$  を解いた時の解

SUPPORTING PROGRAM ENTERED.

SYSTEM SET UP A DUMMY ROOT GIVEN BY ZERO.

EXECUTION CONTINUED.

ROOTS OBTAINED BY BAIRSTOW-HITCHCOCK METHOD:

X( 1 ) = ( -0.4425384117348719D-08 , 0.0	*I )
X( 2 ) = ( 0.1445505580294644D-06 , 0.7860179854626189D-08	*I )
X( 3 ) = ( 0.1445505580294644D-06 , -0.7860179854626189D-08	*I )
X( 4 ) = ( 0.3466272177458383D-01 , 0.0	*I )
X( 5 ) = ( -0.9680734485987065D-02 , 0.0	*I )
X( 6 ) = ( 0.4227538726046039D-01 , 0.5721434661662218D-02	*I )
X( 7 ) = ( 0.4227538726046039D-01 , -0.5721434661662218D-02	*I )
X( 8 ) = ( 0.9852623371074486D 04 , 0.0	*I )
X( 9 ) = ( 0.2031338311967048D 01 , 0.0	*I )
X(10) = ( 0.1014523575756706D 05 , 0.0	*I )

げることは、感質的にも、精度の点において不利のようであるが、係数そのものを書き替えるゆりではないので、もとの精度を維持でき、しかも収束性を高めるという点において、非常に有効な手段であることが、種々の実験によって確かめられた。

#### ・多重根に対する B-H 法

多重根に対して、B-H 法は必ずしも精度のよい解を与えるものではない。と言うより、特別な考慮をしてやらなければ、14桁、15桁といった高精度は、とても保証できない。下に示す数値は  $x^6 - 12x^5 + 63x^4 - 190x^3 + 358x^2 - 400x + 200 = 0.0$

に対する B-H 法による解である。真の解は、 $(2, 1 \pm 2i, 3 \pm i)$  で 2 が 2 重根になっている。この 2 根は  $X(1), X(2)$  に相当する。実用上は、この程度の精度で問題は生じないであろう。2 重根に対しては、収束判定条件をかなり厳しくしても反復回数に制限を設けなければ収束するようであるし、その時得られる値は得心のゆくものであるが、今度は、B-H 法における重根判定に問題が残るところである。出力 2 に示す数値は、 $x^7 - 14x^6 + 87x^5 - 316x^4 + 738x^3 - 1116x^2 + 1000x - 400 = 0.0$

に対する B-H である。真の解は、 $(2, 1 \pm 2i, 3 \pm i)$  で 2 が 3 重根になっている。これに対応するものは  $X(1), X(2), X(3)$  であるが、精度が 2 重根に対して落ちていゝ上に、収束改善策を施さな

TS OBTAINED BY BAIRSTOW-HITCHCOCK METHOD:

- X( 1) = ( 0.2000000051619139D 01 , 0.0 \*I )
- X( 2) = ( 0.1999999948380866D 01 , 0.0 \*I )
- X( 3) = ( 0.1000000000000000D 01 , 0.2000000000000000D 01 \*I )
- X( 4) = ( 0.1000000000000000D 01 , -0.2000000000000000D 01 \*I )
- X( 5) = ( 0.2999999999999996D 01 , 0.999999999999982D 00 \*I )
- X( 6) = ( 0.2999999999999996D 01 , -0.999999999999982D 00 \*I )

▲出力1.  $x^6 - 12x^5 + 63x^4 - 190x^3 + 358x^2 - 400x + 200 = 0.0$   
 のB-H法による解。真の解は、 $(2, 1 \pm 2i, 3 \pm i)$  で X(1), X(2)が"重根 2"に対応する。

▼出力2.  $x^7 - 14x^6 + 87x^5 - 316x^4 + 738x^3 - 1116x^2 + 1000x - 400 = 0.0$   
 のB-H法による解。真の解は $(2, 2, 2, 1 \pm 2i, 3 \pm i)$  で、X(1), X(2), X(3)が"3重根 2"に対応する。収束改善策を施さなければ収束しない。

TS OBTAINED BY BAIRSTOW-HITCHCOCK METHOD:

- X( 1) = ( 0.1999982893617020D 01 , 0.0 \*I )
- X( 2) = ( 0.2000008553191473D 01 , 0.1481529170570842D-04 \*I )
- X( 3) = ( 0.2000008553191473D 01 , -0.1481529170570842D-04 \*I )
- X( 4) = ( 0.1000000000000002D 01 , 0.2000000000000002D 01 \*I )
- X( 5) = ( 0.1000000000000002D 01 , -0.2000000000000002D 01 \*I )
- X( 6) = ( 0.3000000000000018D 01 , 0.1000000000000000D 01 \*I )
- X( 7) = ( 0.3000000000000018D 01 , -0.1000000000000000D 01 \*I )

▼出力3 (出力1)と同じ問題を複素Newton法(MATRP)で補正して得られた解。

ITS OBTAINED BY MATRP :

- X( 1) = ( 0.1999999999999994D 01 , 0.0 \*I )
- X( 2) = ( 0.1999999999999994D 01 , 0.0 \*I )
- X( 3) = ( 0.1000000000000000D 01 , 0.2000000000000000D+01 \*I )
- X( 4) = ( 0.1000000000000000D 01 , -0.2000000000000000D+01 \*I )
- X( 5) = ( 0.2999999999999997D 01 , 0.999999999999964D+00 \*I )
- X( 6) = ( 0.2999999999999997D 01 , -0.999999999999964D+00 \*I )

ければ収束しない。別の見方をすれば、前述した収束改善策は、多重根を持つ問題に対しては、多重根のうち一つを分離させる効果をもつと言える。B-H法において、 $\Delta p$ と $\Delta q$ は

$$\Delta p = (b_{n-1} \times C_{n-2} - b_n \times C_{n-3}) / \Delta$$

$$\Delta q = (b_n \times C_{n-2} - b_{n-1} (C_{n-1} - b_{n-1})) / \Delta$$

$$\Delta = C_{n-2}^2 - C_{n-1} \times C_{n-3}$$

により決定されるが、与えられた方程式が3重根またはそれ以上の重根を持つとき、 $\Delta = 0.0$  となってしまう。2重根の場合も、うまく2重根同士をペアに組めなかつたならば、 $\Delta = 0.0$  となる。原理的には解が求まらないはずである。しかし、幸いにも、我々は今までそのような事態に出会ったことはない。B-H法で求まった解がかなりの誤差をもっているとしても、その解が得られた反復の列を見ても、 $\max(|\Delta p|, |\Delta q|)$  がゼロになっていたりする。このあたり計算機による多項式の計算の限界であるのかもしれない。しかし、B-H法にせよ、DKA法にせよ、その計算の限界によって、ある程度救われている面を持っていることは否めない事実であろう。

### 3.2 Durand-Kerner-Aberth法 (DKA法)

DKA法は、適当な初期値  $z_1^{(0)}, \dots, z_n^{(0)}$  から出発し、

$$z_i^{(v+1)} = z_i^{(v)} - \frac{P(z_i^{(v)})}{\prod_{\substack{j=1 \\ j \neq i}}^n (z_i^{(v)} - z_j^{(v)})} \quad (i=1, \dots, n) \quad (v \geq 0)$$

による反復を行ない,  $\nu \rightarrow \infty$  のとき,  $x_i^{(\nu)} \rightarrow \alpha_i$  ( $i=1, \dots, n$ )

$P(\alpha_i) = 0.0$  とする方法である。DKA法は、大域収束性があり、他の方法と異なり、誤差が累積しないという極めてすぐれた特徴をもつ。この方法は、 $n$ 元連立非線型方程式を解くNewton法であり、局所的2次収束する。しかし、次数の高いものに対しては、最初のうち収束が極端に遅いという性質を併せもつ。RÖÖTS-PACKでは、誤差が累積しない、局所2次収束という点で、精度の改善に用いている。

### 3.3 複素Newton法

多重根に対してB-H法による解が高い精度を保証できないことは前述した。DKA法において、 $P(x) = 0.0$ の根に多重根があっても、近似根は相異なるため、重根を持つ方程式に対しても収束することが確かめられている。実用上は、まったく支障はないが、高い精度はDKA法でも保証できない。ここで述べるNewton法はRÖÖTS-PACKで用いているもので、多重根に対しても有効なNewton法である。 $P(x)$ を $\alpha$ 中心にTaylor展開すると、

$$P(x) = P(\alpha) + \sum_{i=1}^{\infty} \frac{P^{(i)}(\alpha)}{i!} (x-\alpha)^i$$

$P(x) = 0.0$ において、 $x = \alpha$ が $m$ 重根であるとするとき、

$$P(\alpha) = P'(\alpha) = \dots = P^{(m-1)}(\alpha) = 0.0$$

となり、 $x = \alpha$ の近傍で

$$P(x) = \frac{P^{(m)}(\alpha)}{m!} (x-\alpha)^m + O((x-\alpha)^{m+1})$$

このことから、 $x-\alpha \approx (P(x))^{\frac{1}{m}}$ 。つまり、1変数Newton法を $m$ 重根に適用した場合、根の精度は、残差の精度の $\frac{1}{m}$ しか得ることができない。方程式  $P(x)=0.0$  が2重根 $\alpha$ をもつ時、 $P(x)$ の導関数  $P'(x)$ による  $P'(x)=0.0$ という方程式は $\alpha$ を単根にもつ。一般に  $P(x)=0.0$  が $m$ 重根 $\alpha$ をもつ時  $P^{(m-1)}(x)=0.0$  は $\alpha$ を単根にもつ。ROOTS-PACK で用いたNewton法は、方程式  $P(x)=0.0$  が、 $m$ 重根を持つとき

$$x_{i+1} = x_i - \frac{P^{(m-1)}(x_i)}{P^{(m)}(x_i)}$$

によるNewton法を行なうものである。第 $k$ 次導関数の計算は組立除法によっている。すなわち、 $P(x)$ の原点を移動し、

$$P(x) = (x-d)^m + r_1(x-d)^{m-1} + \dots + r_n$$

とした時  $r_k = \frac{P^{(n-k)}(\alpha)}{(n-k)!}$  となる。

$$x_{i+1} = x_i - \frac{r_{n+1-m}}{r_{n-m} \times m}$$

よって

とすればよい。多重根の判定は問題の残るところであるが、

今のところ  $\left| \frac{P^{(m-1)}(x_i)}{P^{(m)}(x_i)} \right| \geq |x_i| \times \rho$  ( $\rho=10^{-8}$ )  
によっている。

この条件を満足した時、導関数を1つあげることにしているが、この点に関しては、ここからの考察が必要となろう。

出力 3 は  $x^6 - 12x^5 + 63x^4 - 190x^3 + 358x^2 - 400x + 200 = 0.0$

を B-H 法で求め Newton 法で修正した解である。前にも書いたが真の解は  $(2, 1 \pm 2i, 3 \pm i)$  で 2 が 2 重根である。出力では、 $X(1)$  と  $X(2)$  がそれに対応している。

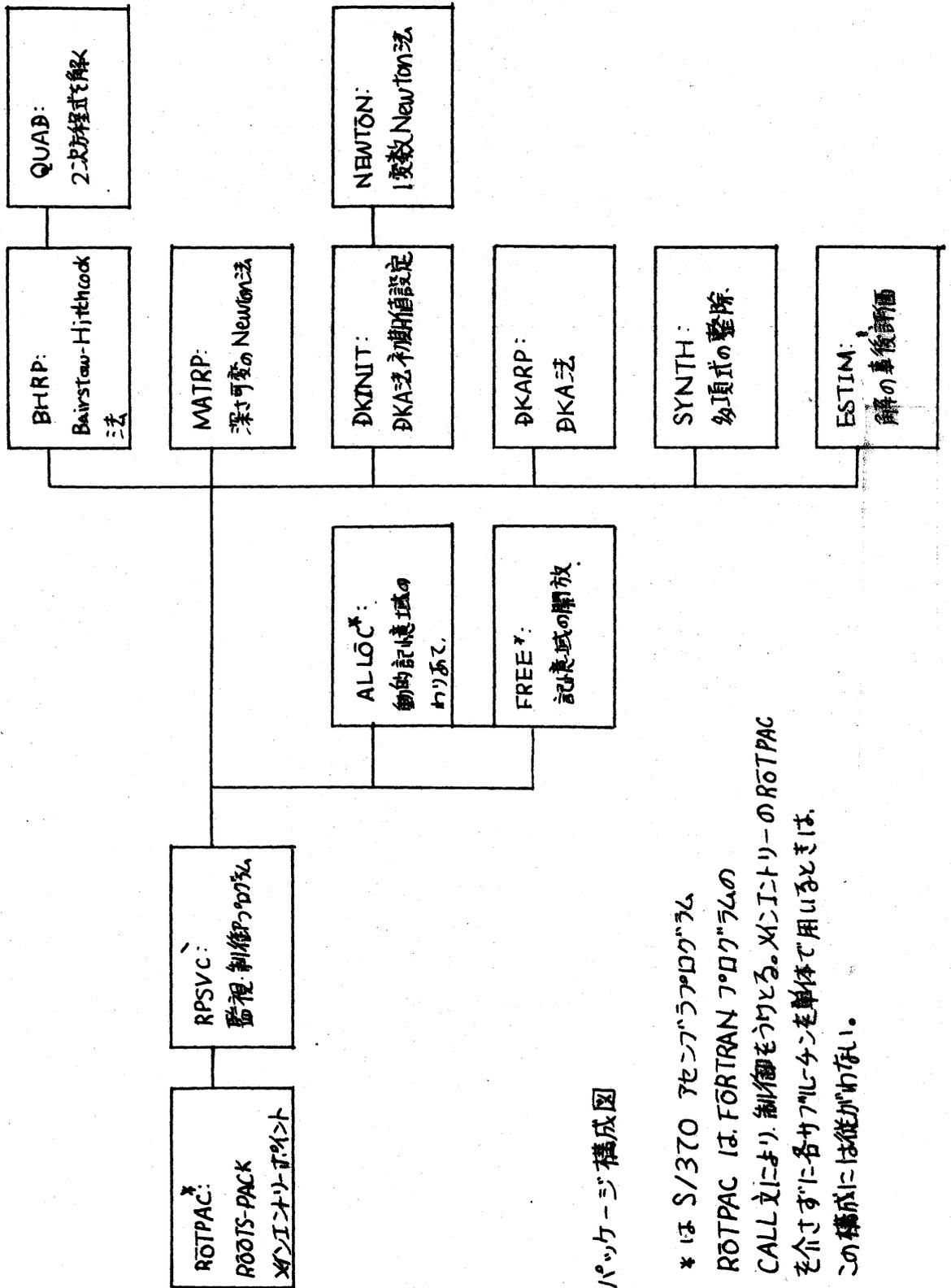
#### 4. 誤差の処理

RÖÖTS-PACK の中心となっている解法は B-H 法であるが、B-H 法等の減次操作を伴うもので気になるのは、次数低下のくり返しによる誤差の累積の危険性をもつということである。誤差の混入、伝播、拡大について考慮するならば、絶対値の小さいものから求めてやれば、析落ちの危険も少なくなり、誤差の拡大も最小に抑えられる。RÖÖTS-PACK では解の事後評価を行ない、必要であれば精度の改善を行なうため、それ程気にする必要もないが、できるならば、絶対値の最も小さいものに収束するような値を B-H 法の初期値として与えることが望ましい。数値実験では、ZERO-DIV. I ラーの生じる以外は、 $p, q$  の初期値として 0.0 を用いたが、B-H 法に関する限り、この値は初期値の設定という問題を解決しているように思われる。と言うのは、初期値として 0.0 を与えた場合、ほとんどの問題で原点に近い 2 根が組みあわせられてゆくことが実験によりわかったからである。減次操作による精度の低下は、析落ち等の他に、解そのもののもつ誤差によっても生じる。前

述べたように B-H 法は、3 重根に対しては収束しにくく、得られた解の精度も悪い。精度の悪い近似解で減次すれば、以下の解に対する影響は極めて大きいものとなるであろう。しかし、今までの実験では、減次操作を繰り返す度に、解の精度が次第に低下してくるような現象に出会ったことはない。精度の悪い近似解が得られた後の 2, 3 のものは影響を受けるが、再び精度をもちなおすようである。精度の悪い近似解が出現したならば、その後の 2, 3 の解で精度のくずれを吸収しているような現象がおこるのである。

#### 5. RÖÖTS-PACK の構成

RÖÖTS-PACK のパッケージ構成図を図 1 に示す。RÖÖTS-PACK は FORTRAN プログラムの CALL 文によって制御権を受けると、メインエントリーの RÖTPAC は dummy array のポインタの初期設定を行なう。その後、RÖÖTS-PACK の監視コントロールを行なう RPSVC へさらに制御をわたす。RPSVC 以下のすべてのサブルーチンの情報は全て RPSVC へ集められ、制御される。RPSVC は初めに仮想主記憶の制御を行ない、必要な長さだけの記憶域を確保する。解法は、B-H 法、DKA 法、Newton 法を必要に応じて使い分け、精度の改善をはかる必要があるれば、それを行なう。与えられたすべての方程式は、まず B-H 法へ通され、収束しなかつた場合や、精度が悪い場合には、しかるべき



・パッケージ構成図

\* は S/370 アセンブラプログラム  
 ROTPAC は FORTRAN プログラムの  
 CALL文により、制御をうつりうる。メインリリーの ROTPAC  
 を介さずに各サブルーチンを単体で用いるときは、  
 この構成には従がわない。

処置がとられる。収束しなかつた問題に対して、収束改善策を施してもなおかつ収束しなかつた問題に出会ったことはないが、万一来備え、Aberthの方法によつて初期値を設定し、DKA法が用いられるようになっている。

#### ・プログラム概略

RÖTPAC (n,A,X,R)      RÖOTS-PACK のメインエントリー。次の RPSVC で仮想主記憶の制御を行なうため、dummy array のポインタの初期化をする。つまり、FORTRAN プログラム内で割りあてられた配列を使うため、ポインタを '1' にして、パラメータリストを書き替える作業をしている。ここでは、(array, n, A, X, R) というパラメータリストを作成し、次の RPSVC へ制御を渡す。

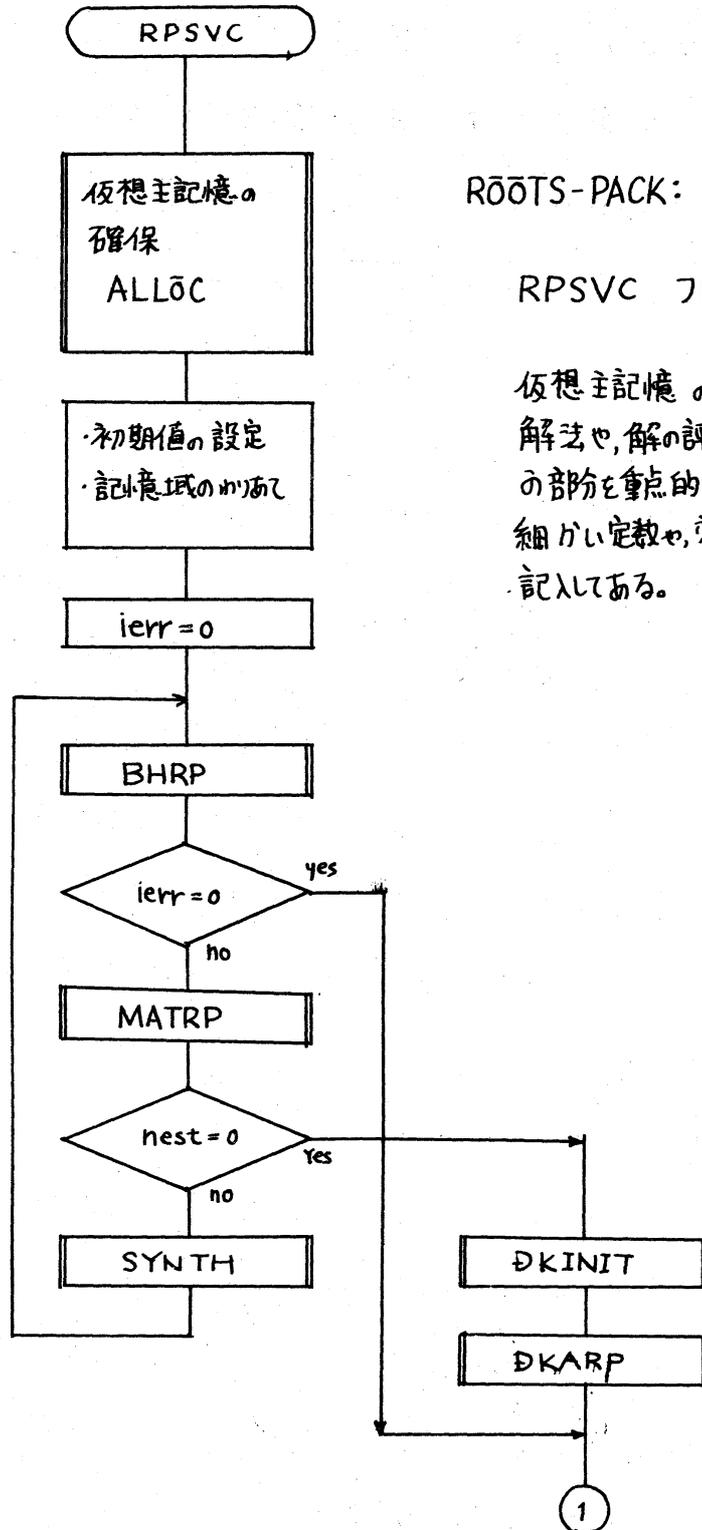
RPSVC (ARRAY, n, A, X, R)      RÖOTS-PACK 監視制御プログラム。

```

C ::::: ROOTS SYSTEM SUBROUTINE PACKAGE ::::::::::::::::::::
C ::::: ROOTS-PACK SUPERVISOR & CONTROLLER ::::::::::::::::::::
      SUBROUTINE RPSVC (ARRAY, N, A, X, R)
        COMPLEX*16  X(N), CDUMMY
        REAL*8      A(N), R(N)
        INTEGER*4   ADDR16, ADDR1, ADDR2, BOUND1, BOUND2
        LOGICAL*1   ARRAY(1), CONVRP, DKASUP, ALLDEF

```

必要な配列の長さを LENGTH バイトであるとすると、RPSVC は ALLOC を呼ぶ (つまり CALL ALLOC (LENGTH, ADDR16) ) ことにより、主記憶内 ADDR16 番地から LENGTH バイトの記憶域を確保し、その制御範囲に入れるのである。ただし、dummy array の ARRAY は、



RÖÖTS-PACK:

RPSVC フローチャート (1)

仮想主記憶の制御以外の部分、  
解法や、解の評価、精度の改良のアルゴリズム  
の部分を重点的に示した。  
細かい定数や、変数は全て省いて  
記入してある。



RPSVC内では1バイトの論理型であると宣言してあるため、このプログラム内では、割りあてた記憶を実数型としては使用できない。実数型の配列として使用するには、さらにサブルーチンを叫ばなくてはならない。RPSVCでは、次のようなサブルーチンを用いている。このサブルーチンは、概に用意してある。A, Xの他に、倍精度複素型のZ, 論理型のCONVを新たに用意しなければ利用することはできない。

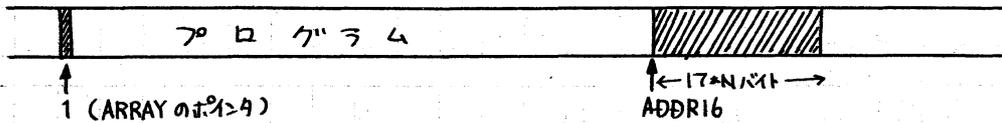
```
C ::::: DURAND-KERNER-ABERTH METHOD FOR SOLVING ALGEBRAIC EQUATIONS :::
SUBROUTINE DKARP(N,A,X,Z,CONV,ITRLMT)
COMPLEX*16 X(N),Z(N)
COMPLEX*16 U,V,CD,DX
REAL*8 A(N)
REAL*8 THRESH,ROOF,EPS,DUMMY
REAL*8 DREAL,DIMAG
LOGICAL*1 CONV(N),ALLDEF
```

N(方程式の次数)が例えば10であったとすると、RPSVC内でDKARPを用いるには次のようにすればよい。

```
LENGTH = 16*N + N
CALL ALLOC(LENGTH,ADDR16)
ADDR1 = ADDR16 + 16*N
CALL DKARP(N,A,X,ARRAY(ADDR16),ARRAY(ADDR1),100)
CALL FREE(LENGTH,ADDR16)
```

実引数と仮引数の型が異なっているが、ここでは単にアドレスの受け渡しをしているだけで、エラーは起きない。上の例は、ARRAYという配列のADDR16番地から16\*NバイトをZ用に、ADDR1番地からNバイトをCONV用に割りあてた例である。ADDR16の内容

は知ることができない。主記憶内は下のよう分割



されている。ARRAY(1)はプログラムの先頭に1バイトだけ確保されているが、ARRAY(2)は恐らくプログラム本体とオーバーラップしている。ADDR16の値はオーバーラップしている配列が配列として使用できる最初の番地なのである。

ALLOC (length, addr) 動的記憶域割当てルーチン。仮想主記憶の制御を行ない、lengthバイトの記憶域を確保する。

FREE (length, addr) 動的記憶域開放ルーチン。仮想主記憶の制御を行ない、ALLOCで割りあてた addr番地から、長さlengthバイトの記憶域を開放する。

BHRP (n, A, X, ierr) 単体で使用する時は ierr=0としてから、CALLしなればならない。出力時、通常は ierr=0となって制御が返されるが、エラー時は ierr=j となって出力される。この時 X(j) 以降の解は不定である。

MATRP (n, A, α, Z, nest) 複素 Newton法サブルーチン。単体で利用する場合、ユーザーは、αに初期値を与えておかなければならない。Zは計算用の配列。nestには根の多重度が入る。反復が収束しなかった場合、

nest=0 とセットされて出力される。

DKARP (n, A, X, Z, CONV, iters) DKA 法サブルーチン。単体で使用する場合は、CALL する前に  $CONV(i) = .false.$  ( $i=1, \dots, n$ ) と初期値を与えておかなければならない。すべての解を得られた時は  $CONV(i) = .true.$  となって制御が返される。もし、 $CONV(i) = .false.$  ならば、 $X(i)$  は不定である。iters はユーザーが指定する反復回数。

DKAINIT (n, A, X) Aberthの方法によるDKA法の初期値設定ルーチン。配列Aの内容は出力時も破壊されない。

SYNTH (n, A,  $\alpha$ , iroot)  $\alpha$ による多項式の整除を行なう。 $\alpha$ が虚数の時は、自動的に共役数を組みあわせて、2次式による整除を行なう。irootは整除した式の次数、すなわち、2次式による整除であれば  $iroot=2$  である。nは整除された後の式の次数が入り、て出力される。

ESTIM (n, A, X, R) Smithの定理により、得られた解を事後評価するルーチン。 $R(i)$  は  $X(i)$  の評価値。

QUAD (p, q,  $y_1, y_2$ ) 2次方程式  $x^2 + px + q = 0.0$  を解くサブルーチン。絶対値の小さい解が  $y_1$  に入る。

NEWTON (n, A,  $\alpha$ , iters) 実Newton法サブルーチン。単体で用いる場合は、ユーザーは  $\alpha$  の初期値を設定しておかな

ければならない。itersは解を得るのに必要とした反復数が入る。収束しなかった場合は、iters=0とセットされて出力される。

\*変数    n: 方程式の次数 (integer \* 4)  
           A: 係数を含む配列 (real \* 8)  
           X: 解が入る配列 (complex \* 16)  
           R: 解の評価値が入る配列 (real \* 8)  
           ARRAY: dummy array (logical \* 1)  
           Z: 計算用配列 (complex \* 16)  
           CONV: 収束判定用配列 (logical \* 1)  
           α: MATRP, SYNTH では (complex \* 16)  
               NEWTON では (real \* 8)  
           y<sub>1</sub>, y<sub>2</sub>: (complex \* 16)

#### ・プログラム

以下に、ROOTS-PACK のプログラムのうち、一部を付記しておく。言語は、FORTRAN IV (G or G1) と IBM System/370 アセンブラ言語である。GETMAIN, FREEMAIN は仮想主記憶の制御に関するマクロ命令である。

FILE: ROOTSPK VMASM A VM/370 VERSION 05 PLC.12 LTR.512 WASEDA UNIV.

```

*****
*
*   ROOTS SYSTEM SUBROUTINE PACKAGE SYSTEM/370 ASSEMBLER PROGRAMS
*
*****
* MACRO DEFINITION
  MACRO
    REGEQU
R0    EQU    0
R1    EQU    1
R2    EQU    2
R3    EQU    3
R4    EQU    4
R5    EQU    5
R6    EQU    6
R7    EQU    7
R8    EQU    8
R9    EQU    9
R10   EQU    10
R11   EQU    11
R12   EQU    12
R13   EQU    13
R14   EQU    14
R15   EQU    15
  MEND
* SAVE REGISTERS & SET R12 AS BASE REGISTER
  MACRO
&LABEL  SAVEREG  &AREA
        STM     14,12,12(13)
        BALR   12,0
        USING  *,12
        ST     13,&AREA+4
        LR     15,13
        LA     13,&AREA
        ST     13,8(15)
  MEND
* RESTORE REGISTERS & RETURN CONTROL TO CALLER
  MACRO
&LABEL  RETREG  &AREA
        L      13,&AREA+4
        LM     14,12,12(13)
        BR     14
  MEND
*
* CONTROL SECTION BEGINS
*
* USAGE: ROTPAC(N,A,X,R)
ROTPAC  CSECT
        EXTRN RPSVC
        REGEQU
        SAVEREG  SAVEAREA
* GENERATE NEW PARM-LIST
        MVC     NAXR(16),0(R1)
        LA     R1,PARMLIST
        L      R15,=A(RPSVC)
*GENERATE PARMLIST
*R1 = ADDR OF PARMLIST
*LOAD ADDR OF 'RPSVC'

```

FILE: ROOTSPK VMASM A VM/370 VERSION 05 PLC.12 LTR.512 WASEDA UNIV.

```

BALR R14,R15                                *SUBROUTINE CALL
* RESTORE REGISTERS
  RETREG SAVEAREA
  LTORG
* SAVE AREA
SAVEAREA DS 18F
* PARAMETER LIST
PARMLIST DS 0F                                *PARAMETER LIST
ARRAY DC F'1'                                *SET ADDR OF DUMMY ARRAY
VAXR DS 4F                                    *REAL PARMLIST
*
  END
*
* VIRTUAL STORAGE CONTROLLER
* USAGE: ALLDC ( LENGTH, ADDR )
ALLOC CSECT
* SAVE REGISTERS
  SAVE (14,12)
  BALR 2,0
  USING *,2
* GET VIRTUAL STORAGE
  LM 3,4,0(1)                                *LOAD ADDR
  L 3,0(3)                                    *R3 = LENGTH
  GETMAIN R,LV=(3)                            *GET VIRTUAL STORAGE
  ST 1,0(4)                                    *STORE ADDR
* RESTORE REGISTERS
  RETURN (14,12)
  END
*
* USAGE: FREE ( LENGTH, ADDR )
FREE CSECT
* SAVE REGISTERS
  SAVE (14,12)
  BALR 2,0
  USING *,2
* FREE VIRTUAL STORAGE
  LM 3,4,0(1)                                *LOAD ADDRS OF PARAMETERS
  L 3,0(3)                                    *R3 = LENGTH
  L 4,0(4)                                    *R4 = ADDR
  FREEMAIN R,LV=(3),A=(4) *FREE VIRTUAL STORAGE
* RESTORE REGISTERS
  RETURN (14,12)
  END
*
* GET REAL PART OF X DECLARED AS COMPLEX*16
DREAL CSECT
  L 1,0(1)
  LD 0,0(1)
  BR 14
  END
*
* GET IMAGINAL PART OF X DECLARED AS COMPLEX*16
DIMAG CSECT
  L 1,0(1)
  LD 0,8(1)

```

FILE: ROOTSPK VMASM A VM/370 VERSION 05 PLC.12 LTR.512 WASEDA UNIV.

BR 14  
END

26

FILE: BHRP VMF0RT A VM/370 VERSION 05 PLC.12 LTR.512 WASEDA UNIV.

```

C ::::: ROOTS SYSTEM SUBROUTINE PACKAGE ::::::::::::::::::::::::::::
C ::::: BAIRSTOW-HITCHCOCK METHOD ::::::::::::::::::::::::::::::::::::
      SUBROUTINE BHRP(N,A,X,IERR)
        COMPLEX*16 X(N)
        COMPLEX*16 Y1,Y2,CD,U,V
        REAL*8 A(N)
        REAL*8 P,Q,D,DP,DQ,PB,QB,PF,QF,B1,B2,B3,C1,C2,C3
        REAL*8 EPS,THRESH,BASE,BASIS,DUMMY
        REAL*8 DABS,DMAX1,CDABS
        LOGICAL*1 EFFECT,LAST,ZRDIV,SUPPRT
C ::::: EXECUTION BEGINS :::::
      IF ( N .LE. 0 ) RETURN
C ::::: INITIALIZE :::::
      LAST = .FALSE.
      BASE = 1.0D-12
      COUNT = 0
      ITRLMT = 100
      M = N
      ID = 1
      SUPPRT = IERR .EQ. 0
      IF ( .NOT. SUPPRT ) ID = IERR
      IERR = 0
      G = 2.0 / FLOAT( N )
      PF = 1.0D0
      QF = 1.0D0
      IF ( A(1) .NE. 0.0D0 ) PF = A(1) * G
      IF ( A(N) .NE. 0.0D0 ) QF = DABS( A(N) ) ** G
C ::::: BRANCH :::::
      250 IF ( M - 2 ) 580, 470, 300
C ::::: DEFINE P & Q :::::
      300 EFFECT = .FALSE.
          P = 0.0D0
      320 Q = 0.0D0
          DQ = 0.0D0
          K = M - 1
C ::::: ITERATION :::::
      360 ZRDIV = .FALSE.
          BASIS = BASE
          MORE = ITRLMT + 1
          DO 420 ITER = 1, ITRLMT
            THRESH = DMAX1( DABS( P ) , DABS( Q ) ) * BASIS
      370          B2 = 0.0D0
              C2 = 0.0D0
              B3 = 1.0D0
              C3 = 1.0D0
              DO 380 I = 1, K
                B1 = B2
                C1 = C2
                B2 = B3
                C2 = C3
                B3 = A(I) - P * B2 - Q * B1
                C3 = B3 - P * C2 - Q * C1
      380          CONTINUE
              B1 = 0.0D0
              IF ( .NOT. EFFECT ) B1 = A(M)

```

FILE: BHRP VMFORT A V4/370 VERSION 05 PLC.12 LTR.512 WASEDA UNIV.

```

      B1 = B1 - P * B3 - Q * B2
      D = C2 * C2 - C1 * ( C3 - B3 )
      IF ( D .NE. 0.000 ) GO TO 390
C     ::::: ZERO-DIVIDE CONDITION RAISED :::::
      IF ( EFFECT ) GO TO 390
      IF ( ZRDIV ) GO TO 450
      ZRDIV = .TRUE.
      P = PF
      Q = QF
      GO TO 370
390   DP = ( B3 * C2 - B1 * C1 ) / D
      DQ = ( B1 * C2 - B3 * ( C3 - B3 ) ) / D
      EPS = DMAX1( DABS( DP ) , DABS( DQ ) )
      IF ( ITER .GT. MORE ) GO TO 500
400   IF ( EPS .GT. THRESH ) GO TO 410
      MORE = MIN0( ITER + 5 , ITRLMT - 10 )
      PB = P
      QB = Q
      IF ( BASIS .LT. 1.00-30 ) GO TO 500
      BASIS = BASIS * 1.00-02
      GO TO 400
410   P = P + DP
      Q = Q + DQ
420 CONTINUE
C     ::::: THE GREATEST ALGORITHM :::::
      CALL QUAD(P,Q,X(ID),X(ID+1))
      IF ( ( .NOT. SUPPRT ) .OR. ( EFFECT ) ) GO TO 460
      EFFECT = .TRUE.
      P = -X(ID)
      M = M + 1
      GO TO 320
C     ::::: THE LAST TWO ROOTS FOUND :::::
470   PB = A(M-1)
      QB = A(M)
      LAST = .TRUE.
C     ::::: QUADRATIC EQUATION :::::
500   CALL QUAD(PB,QB,X(ID),X(ID+1))
      IF ( .NOT. EFFECT ) GO TO 540
      X(ID) = X(ID+1)
      ID = ID - 1
540   ID = ID + 2
      COUNT = 0
      M = M - 2
      IF ( LAST ) GO TO 590
C     ::::: SYNTHETIC DIVISION :::::
      B2 = 0.000
      B3 = 1.000
      DO 570 I = 1, M
          B1 = B2
          B2 = B3
          B3 = A(I) - P * B2 - Q * B1
          A(I) = B3
570 CONTINUE
      GO TO 250
C     ::::: ONE ROOT FOUND :::::

```

FILE: BHRP VMFORT A VM/370 VERSION 05 PLC.12 LTR.512 WASEDA UNIV.

```
580 X(ID) = -A(1)
      ID = ID + 1
590 ID = ID - 1
700 RETURN
C ::::: ERROR EXIT :::::
450 ID = -ID
460 IERR = ID
      RETURN
      END
```

FILE: MULTRP VMF0RT A VM/370 VERSION 05 PLC.12 LTR.512 WASEDA UNIV.

```

C ::::: ROOTS SYSTEM SUBROUTINE PACKAGE ::::::::::::::::::::
C ::::: MATSUMUTA'S ALGORITHM: NEWTON-ITERATION FOR MULTIPLE ROOTS :::::
SUBROUTINE MATRP(N,A,X,B,NEST)
  COMPLEX*16 B(N)
  COMPLEX*16 X,SAVE,DX,C,VALUE,DCMPLX
  REAL*8 A(N)
  REAL*8 THRESH,EPS,BASE
  REAL*8 ROOF,OLDEPS,HEAVEN
  REAL*8 DREAL,DIMAG
  INTEGER*4 DASH,STATUS,STHOLD,HOLD
  LOGICAL*1 FLAG,PERMIT,FIX
C ::::: INITIALIZE :::::
  FIX = NEST .NE. 0
  IF ( .NOT. FIX ) NEST = 1
  ITRLMT = 30
  STATUS = 0
  STHOLD = 7
  SAVE = X
C ::::: ITERATION BEGINS :::::
  DO 200 ITER = 1, ITRLMT
    HEAVEN = CDABS( X )
    ROOF = CDABS( X ) * 1.0D-08
    THRESH = CDABS( X ) * 1.0D-12
C ::::: DEFINE F(X) :::::
    C = 1.0D0
    DO 110 I = 1, N
      C = A(I) + X * C
      B(I) = C
110    CONTINUE
C ::::: DEFINE F'(X) :::::
    DO 140 DASH = 1, NEST
      K = N - DASH
      C = 1.0D0
      DO 130 I = 1, K
        C = B(I) + X * C
        B(I) = C
130      CONTINUE
140      CONTINUE
    DX = B(K+1) / ( B(K) * DFLOAT(NEST) )
    EPS = CDABS( DX )
    IF ( ( EPS .LE. THRESH ) .AND. ( ITER .GE.2 ) ) GO TO 220
C ::::: TEST AS IF F'(X) EQUAL ZERO OR NOT :::::
    PERMIT = ( ITER .GT. 10 ) .AND. ( STATUS .GT. STHOLD )
    & .AND. ( EPS .GT. ROOF ) .AND. ( .NOT. FIX )
    & .AND. ( EPS .LT. HEAVEN )
    IF ( .NOT. PERMIT ) GO TO 150
    STATUS = 0
    NEST = NEST + 1
    STATUS = STATUS + 1
150    IF ( EPS .LT. HEAVEN ) GO TO 170
        STATUS = 0
        STHOLD = 10
        IF ( NEST .GE. 2 ) NEST = NEST - 1
        DX = 0.0D0
170    X = X - DX

```

FILE: MULTRP VMF0RT A VM/370 VERSION 05 PLC.12 LTR.512 WASEDA UNIV.

```
                IF (DABS(DREAL(X)).LE.THRESH) X=DCMPLX(0.0D0,DIMAG( X ))
                IF (DABS(DIMAG(X)).LE.THRESH) X=DCMPLX(DREAL( X ),0.0D0)
200      CONTINUE
210      NEST = 0
           X = SAVE
220      RETURN
           END
```

## 6. おまじ

下に示したプログラムは RÖÖTS-PACK の使用例である。

```

C ::::: THE ALGEBRAIC EQUATION      ::::::::::::::::::::::::::::::::::::
  COMPLEX*16 X(52)
  REAL*8 A(52),R(52)
C ::::: EXECUTION BEGINS ::::::::::::::::::::::::::::::::::::::::::::
  10 READ(5,1000,END=450) N,(A(I),I=1,N)
  WRITE(6,1002) N,(I,A(I),I=1,N)
C -----
  CALL ROTPAC(N,A,X,R)
  WRITE(6,1006)
  WRITE(6,1007) (I,X(I),R(I),I=1,N)
C -----
  GO TO 10
450 STOP
1000 FORMAT(I2/(4E20.12))
1002 FORMAT('1','ROOTS SYSTEM SUBROUTINE PACKAGE: ( VERSION 1.0 )'
&/ ' ', 'X**N + A(1)*X**(N-1) + ... + A(N) = 0.0'
&/ ' ', 'N =', I3
&/ ' '/(6X, 'A(', I3, ' ) =', D24.16))
1006 FORMAT('0','ROOTS OBTAINED BY ROTPAC:')
1007 FGMAT(('0',15X, 'X(', I2, ') = (', D24.16, ' ', ', D24.16, ' *I )'
&, 10X, 'N*|P(X)/DPI =', D9.2))
  END

```

現在、RÖÖTS-PACKは CMS において、個人用ディスクで使用している。オブジェクトで30k足らずで、個人用の割りあてりシリンダに納め、他にワークエリアが充分とれるほど小さい。今後、RPSVCの改良や、重根の判定等、多くの課題が残されているが、RÖÖTS-PACKが、今後のサブルーチン・パッケージのあり方に対する一つの提案となれば幸である。

## ・参考文献

(1) 情報処理 6月号(1977) 代数方程式を解く Durand-Kerner法

と A berth法 山本哲郎 (p.566)

(2) 数値解析の基礎 (日新出版) 篠原能材