APLとOR(2)

竹下　亨

10. 級数の計算

数値計算によく出てくる初等関数のいくつかは級数展開の形で表わされ，はじめから何項かまでを計算して許容誤差内の近似値を求めることができる．級数の項に現われる階乗は！(点引用符)を使って！Kと表現する．

[例]

指数関数e^xは$1+\frac{x^2}{2!}+\frac{x^3}{3!}+\frac{x^4}{4!}+\cdots$で表わされるので，$x=0.5$のときに，第3項の値を求めるには次のように書く．

```
      X←.5
      K←3
      (X*K)÷!K
0. 0 2 0 8 3 3 3 3 3 3
```

$k=0$から$k=4$までの項の和は，次のようにして求められる．

```
      X←.5
      K←0 1 2 3 4
      S←+/(X*K)÷!K
1. 6 4 8 4 3 7 5
```

e^xを原始関数*を使って求めよう．

```
      *X
1. 6 4 8 7 2 1 2 7 1
```

もう1つ例をあげてみよう．

[例]

双曲線余弦関数は，

$$1+\frac{x^2}{2!}+\frac{x^4}{4!}+\frac{x^6}{6!}+\frac{x^8}{8!}+\cdots$$

と展開されるが，これを第5項まで求めよう．

```
      X←0.5
      C←2×K←0 1 2 3 4
      +/(X*C)÷!C
1. 1 2 7 6 2 5 9 6 5
```

たけした　とおる　日本アイ・ビーエム・㈱

$(X*K)÷!K$を関数として定義しておいて，先の指数関数の近似値を求めることもできる．

```
      ∇Z←X TERM K
[1]   Z←(X*K)÷!K
[2]   ∇
      +/.5 TERM 0 1 2 3 4
1. 6 4 8 4 3 7 5
```

なお，第8節で述べたように，関数の定義の中で他の定義関数が使える．

[例]

```
      ∇Z←X SUM K
[1]   Z←+/X TERM K
[2]   ∇
      .5 SUM 0 1 2 3 4
1. 6 4 8 4 3 7 5
```

このやり方で，三角関数，双曲線関数などの近似値が求められる．

11. 分岐と反覆

APLではスカラー原始関数が2次元以上の配列に適用され，かつ配列の形で変える関数もあり，また作用子があって，他の言語と比較してループを作る必要性がきわめて少ない．この点がFORTRANプログラマが面くらうことでもあろう．FORTRANで書かれたものを1対1対応させてAPLに変換すると，APLの良さが活かされず効率の悪いプログラムができてしまう．

APLでどうしてもループを作る必要があるときは，制御の項を変えるために分岐(branch)文を使う．常にどこかへ飛ぶとき(無条件分岐)は，→(右矢印)と行き先の行番号(またはそれを求める式)をおく．一般形は次のように書く．

```
      →A
```

[例]

ある整数を読み込んで，この値を2ずつ増加させ，アテンション・キーによって中断されるまで続けるには，

次のように関数を定義する．

```
        ∇Z←F  X
[1]     Z←X
[2]     Z
[3]     Z←2+X
[4]     →2  ……2行目にもどる
```

Xの値を5として，この関数を使ってみる．

```
        F  5
5
7
9
⋮
```

条件分岐を書く方法は何種類もある．たとえば，

$\rightarrow N\times$ 関係… N へ飛ぶか，関数を終了

$\rightarrow N\times\iota$関係… N へ飛ぶか，次へ進む

$\rightarrow$(関係)ρN… N へ飛ぶか，次へ進む

$\rightarrow$(関係)$/N$… N へ飛ぶか，次へ進む

→文番号よりなるベクトル［指標式］……

……指標式の値で示される文番号へ進む

$\rightarrow N+I\times$関係…$N+I$ 行か，N行へ進む

の形がある．行き先が2つのときは，

→(文番号1，文番号2)[1+関係]

とすれば，関係が成り立たない場合に文番号1へ，成り立つときには文番号2へ飛ぶようにできる．

なお，次のように書けば，条件が成り立つときだけ，現在実行中の関数とそれを呼び出したすべての関数を終了させる．

⍎(関係)/'-'

［例］

引数の値が正ならば，その値をそのまま返し，0が負であれば，0をもどす関数は，次のように定義される．

```
        ∇Y←ABSF  X
[1]     Y←X
[2]     →3×X≤0   X≤0が真なら3へ進む
[3]     Y←0
[4]     ∇
```

APL でループを組むには，制御変数を設け，終りの部分に分岐文をおき，これに制御変数(多くはカウンタ)の増加や減少を行ない，その値がある限界値に達したかテストする．また，分岐文をループの頭におくやり方もある．制御変数の更新と終了のテストを1つの文で行なうことが多い．

［例］

元金 P と利率 R が与えられているとき，1年目から10年目までの元利合計が求まるごとに出力する関数を定義してみよう．

```
        ∇P FUKURI R
[1]     I←0
[2]     S←P
[3]     S←□←S×(1+R)……元利合計
[4]     →3×(10≥I←I+1){10以下では3へもどる
[5]     ∇
```

1年目から10年目までのそれぞれの元利合計を要素とするベクトルを効率を気にしないで求めるのであれば，次のように書ける．なお，ι10は1から10までの整数を生成する．

```
        ∇P FUKURI R
[1]     S←P+(1+R)*ι10
[2]     S∇
```

12. 簡単なグラフ

APL には，グラフ用の既製のプログラムが用意されているので，ユーザーはグラフを表示するためにプログラムを作成する必要性は少ないが，文字表示装置で簡単なグラフの描き方を試みることにしよう．

その1つの方法としては，今 X,Y 座標で表わされる測定値（統計値）があるときに Y 座標の各目盛ごとに X 座標方向に見ていって，測定値でその上に乗るものがあればそこに1をおき他には0をおく．これをYの測定値の最大値と等しいY座標から最小値までくり返してゆく．そうしておかれた1をつないでいけば曲線ができる．

棒グラフを描くには，各目盛ごとに測定値と等しいかより小さい場合に1をおき，他には0をおく．こうしてできた0と1の行列の1を見ればよい．0のところは空字を，1のところは指定の文字(たとえば＊や□)でおき換えるのに，空字と指定文字よりなるベクトルに，上記の0と1を要素とする行列を指標としてつける．

説明が少々長くなったが，例をみて理解されたい．

［例］

$y=x^2-7x+6$ の $1\leqq x\leqq 5$ の間の曲線を描こう．

```
        X←1  2  3  4  5
        Y←□←(X-2)×(X-4)
3  0  ¯1  0  3
```

縦軸の目盛のベクトルと2次式の値との「＝」による外積を求めて，目盛の値と2次式の値が等しいところだけが1となり，他は0となるようにする．

```
        R←3  2  1  0  ¯1……縦軸の目盛
        R∘.=Y……縦軸の目盛とyの値の外積，
                一致した点だけが1となる
```

```
 3  1  0  0  0  1
 2  0  0  0  0  0
 1  0  0  0  0  0
 0  0  1  0  1  0
¯1  0  0  1  0  0
```

棒グラフにするには，「＝」の代りに「＜」を使う．

```
      R∘.≤Y
 3  1  0  0  0  1
 2  1  0  0  0  1
 1  1  0  0  0  1
 0  1  1  0  1  1
¯1  1  1  1  1  1
```

上のグラフで，0を空字でおき換え，1を＊でおき換えるには，次のようにする．

```
      ' *'[1+(R∘.=Y)]
 3  *        *
 2
 1
 0     *     *
¯1        *
```

次に，棒グラフで，0の代わりに空字を，1の代りに□をおくことにする．

```
      ' □'[1+(R∘.≤Y)]
 3  □        □
 2  □        □
 1  □        □
 0  □  □     □  □
¯1  □  □  □  □  □
```

13. 逆行列と連立1次方程式の解

逆数を求めるには，÷の記号を用いた逆数関数があるが，逆行列を求めるには÷と□を重ね合せた⊞（ドミノ記号）を使った逆行列関数を使う．

[例]

```
      A←4 5 2 2 5 4 4 4 1
      A
 4  5  2
 2  5  4
 4  4  1
      ⊞A
¯5.5   1.5   5
  7    ¯2    ¯6
¯6     2     5
```

次に，$Ax=b$ で表わされる連立1次方程式の解を得るのに，B ⊞ A と書けばよい．ここで A は次の連立1次方程式の左辺の x の係数よりなる行列で，B は右辺の定数を要素とするベクトルである．

$$a_{11}x_1+a_{12}x_2+\cdots+a_{1n}x_n=b_1$$
$$\vdots$$
$$a_{n1}x_1+a_{n2}x_2+\cdots+a_{nn}x_n=b_n$$

[例]

$$2x_1-x_2+3x_3=7$$
$$x_1+2x_2+x_3=8$$
$$3x_1+x_2-x_3=10$$

を解くのに，A には x の係数が，B には右辺の値が入っているとすると，次により解が求まる．

```
      B⊞A
3  2  1
```

14. 最小2乗近似

$X\leftarrow B$ ⊞ A は，A が特異点でないときに，

$$\wedge/B=A+.\times X$$

となるようなベクトル X を生ずる．A が特異点であると，そのような X の値は得られぬが，B と $A+.\times X$ の差が最小2乗法の意味で最小となるような X を決める．すなわち，$+/(B-A+.\times X)*2$ の値を最小とするものである．

$y=f(x)$ の関係があるときに，そのグラフ上の N 個の点の座標を (x_1,y_1)，(x_2,y_2)，……，(x_N,y_N) とし，x_1，x_2，……，x_N よりなるベクトルを X，y_1，y_2，……，y_N よりなるベクトルを Y とすると，最小2乗法によって多項式の係数は，次元が D の場合に，

$$Y \boxplus X\circ.*0,\iota D$$

で表わされる．($0,\iota D$ で $0,1,\cdots\cdots,D$ を要素とするベクトルが得られる．)

[例]

$f(x)=x^3$ として，$x=1,2,3,4$ の点に対して，$y=c_1+c_2x+c_3x^2$ で近似するときの，係数 c_1,c_2,c_3 を求める．

```
      X←1 2 3 4
      Y←X*3……Y の値は 1  8  27  64
      c←Y⊞X∘.*0,ι2
1 0.5   ¯1 6.7    7.5
```

15. 簡単な報告書

入力データとともに計算結果を見やすいように小数点の位置をそろえたり，適当なスペースを取りたい．またこれを本体とし，その上に標題や日付，ページ番号，頭書き，各列の上や行の左に見出し，脚書きなどを加えて報告書の形にしたいことがよくある．

このような場合に，APL では，先に紹介した書式関数と連結の関数がよく使われる．表をいくつかの（文字データよりなる）ベクトルや行列を張り合わせて作ると思えばよい．この点は他の言語と異なるやり方である．

連結(concatenate)関数は，(コンマ記号)で表わされ，ベクトルとベクトルとを連結して，1つのベクトルにする．ベクトルと行列，行列と行列の連結にはどの方向のベクトルに沿って連結するかを軸で示す．連結する配列の一方がスカラーのときは，相手に合うような長さのベクトルに拡張される．

[例]

```
      V←2 5 3
      W←1 6
      V, W
2 5 3 1 6
```

```
      PQR
M が  STU, N が ABC という配列であるとする．
      VWX
```

```
      M, [1] N          M, [2] N
PQR               PQRA
STU               STUB
VWX               VWXC
ABC
```

同じ行に文字データと数値結果を組み合わせて出力するには，後者を1項書式関数を使って文字データに変えておいてから，前者に連結する．

[例]

ANSER IS の後に X の値(35.2とする)を表示したいのであれば，次のようにする．

```
      'ANSWER IS ', ⍕X
ANSWER IS 35.2
```

次に，行と列に見出しをつけた報告書を，書式と連結の関数を使って作成してみよう．

[例]

3字よりなる月の名前の英略語を左側に縦に並べるのに，12字のベクトルから変形関数を使って，4×3の行列を得る．

```
      M←4 3⍴'JAN……APR'
      M
JAN
FEB
MAR
APR
      Y← 79 80 81 82
```

R には，5桁，小数点以下3桁，負の符号が含まれる要素の4×4の数値の行列とする．この各要素には，9の長さのフィールドを与え，小数点以下2桁とする．Y もこれに合わせて，フィールドの長さを9字分にしておく，月の略名の上に空白をおく．これをまとめると次のような文を与えることになる．

```
      T←(9 0⍕Y), [1] 9 2⍕R
      (' ', [1] M), T
          79          80
JAN   −583.23      283.63
FEB     36.56    −386.25
MAR   −325.12      89.65
APR    234.53     389.23
```

右側省略

なお列の見出しの年を2字分左に寄せなければ(9 0⍕Y)の代りに(2⌽9 0⍕Y)とすればよい．

16. プログラムの読み方と書き方

APL のプログラムは読みにくいといわれることがあるが，その理由は，情報の密度が高いこと，ギリシャ文字を含む数学的記号が使われていること，関数の種類が多いこと，配列を演算の対象としていることなどである．また右から左へ演算することにも初心者は抵抗を感じるかもしれない．楽に読めるようになるには，行列に慣れること，スカラーから配列指向に切り換えること，左から右へ読むときはトップダウン指向で進むこと，そして，よく使われる演算のステップ(式)は，熟語や慣用句として覚えてしまうとよい．

わかりやすく，使いやすいプログラムの書き方としては，次のようなことをまずお推めしたい．

(1) 名前の使い方

分かりやすい名前を使う．局所的変数にはなるべく英字1字を使い，機能別に区分する．大域的変数には，なるべく短く，意味のある略称を使う．

(2) モジュール構造

- 全体の論理を構造化する．プログラムをモジュール(定義関数)の集合として設計する．
- 1個のモジュールは最大40～50個の文とする．

(3) プログラムの文の構造

- 文の長さは最大60～70字とし，1行に収める
- 処理内容が不明瞭になったり，実行時間が増えるような合成は避ける

(4) 分岐の構造

- 条件のテストはかっこでくくり，見やすくする
- 分岐先の選択は，APL 式の演算の結果が1のとき分岐とする
- 分岐先は，文番号でなく，文の名札で示す
- 多重分岐は，基本的構造を拡張して作る

17. 適用分野

当初は大学や研究所で主として使われ始め，また言語表現が数学の記法の延長ということから，APL はもっぱら科学技術計算用であるという印象を与えていたが，対話方式の問題解決として，企画・管理部門から一般の

現業部門の管理者スタッフや末端の担当者により，予測・計画，設計，見積，実績の評価・分析等々の日常業務に広く活用されている．それも今日では，一時的，非定型的業務から定常的な大規模なオンライン・データ処理にも使われつつある．

その応用面の具体例としては，たとえば，1981年6月に日本IBMが開催したAPLシンポジウムでは，銀行経営情報IR，設計用汎用会話処理システム，原価見積業務，セールスマン組織構造の予測シミュレーション，損保業務統計・予測，販売計画フォロー・システム，市街地再開発事業システム，四媒体広告統計，銀行本部情報システム，銀行企画部門の分析・予測業務，営業店事務指導，社外情報データ・バンク・システム，定期ポジション把握システムなどのアプリケーションが発表されている．

また，同年10月にサンフランシスコで開かれたAPL 81コンファレンスで発表された論文の中には，パターン照合，分散処理，生保商品の利益予測，マクロ経済の教育，計算機援助教育，確率と統計，テキスト処理と写植，対話式報告書作成，テンソルと多重線形代数，ORにおける利用——対話式動的プログラミング・モデル，解析的待合せネットワーク・モデル，統合化回路設計システム等々が含まれている．

さらに，1982年7月に日本IBMが主催したシンポジウムには，株式情報分析，構造解析，電力需要予測，エンジン組立作業配分，医学研究と医学教育などが提出されている．なお，APLで書かれた既製のアプリケーション・パッケージには，次の種類が含まれている．

数学および統計，経営事務，経営科学，シミュレーションとOR，プロジェクト管理，図形処理，プログラム開発ツール，テキスト処理．

18. 今後の見通し

言語の成熟度や普及度をみるのに，標準化の動きが注目される．APLの標準化はAPL 79コンファレンスで発表された“Development of an APL Standard”がベースとなってANSI(米国規格協会) ISO(国際標準化機構)によって進められている．すなわち，1979年11月にチューリンで開かれたISO/TC 97の総会でAFNOR(仏規格協会)が幹事役に決まり，“APL Experts Group”が作られ，最初の作業案を1980年3月に完成し，同年6月，翌年4月，10月の会議を経て，2番目の改訂版が12月に出されている．近くAPL E. G.は作業グループに昇格し，ISO原案が審議されることになろう．ANSIのほうは1980年5月にX3J10の第1回会議を開いてから本年4月までに最終原稿を作り，1983年中にANSI規格が発行されると見られている．

既存のAPL言語を標準化する動きと並行して，この言語をさらに拡張させる試みが進められており，新しい機能の中には，入れ子の配列(一般的配列)，新しい作用子と関数群，現在の作用子と関数の拡張，複素数，例外処理，ファイル・システムなどが含まれている．これらについては，1979年3月にIversonが情報処理学会での学術講演「作用子」の中で触れており，くわしくは，同博士の“Operators and Functions”，Research Report 7091 (#30399) 4126/78, IBM Research Divisionに記述されている．IBMのT. J. Watson研究所ではAPL 2という実験システムが動いているし，上記の機能のいくつかはI. P. SharpやSTSSで実働化されているとも伝えられている．

Computer Worldの1982年2月15日号によると，APLがアプリケーションの開発に信じられないほどの生産性を実証してきたし，またAPLが使えるデータ処理環境が増えてきていると報じられている．標準化や言語の拡張と言語処理系を支えるシステム環境の整備により，ますます多方面に利用されてゆくと予想される．

参考文献

以上APLの特徴をかいつまんで解説した．さらにくわしく言語を勉強したい読者には下記を推せんしたい．

まず，公式的な文法書としては，IBM社発行の「APL Language」GC 26-3847およびその邦訳「APL言語」NGC 26-3847がある．

英語の参考書としてはIversonは次をあげている．

Gilman, L. and Rose A. J.: *APL An Interactive Approach*. John Wiley & Sons

Polivka, R. P. and Pakin, S: *APL; The Language and Its Usage*, Prentice-Hall

Le Page, W: *Applied APL Programming*, Prentice-Hall

Harms, E. and Zabinski, M. P.: *Introduction to APL and Computer Programs*, John Wiley & Sons

日本で出されたAPL関係書物には次がある．

長田純一，内山 昭：APLSV，丸善(1975)

長田純一，内山 昭：APL入門，丸善(1976)

竹下 亨：APLプログラミング入門，オーム社(1978)

竹下 亨：プログラミング言語APL，共立出版(1981)

竹下 亨：APL 8週間，共立出版(1982)

日本APL協会編(訳)：APL—微積分のアルゴリズム的方法(Orth, D. L.: Calculus in new keyの訳)