プログラムのページ

担当　森口繁一

6215. 常微分方程式の数値積分法（自動的にきざみ幅を変化させる PC 法）

高田　勝（九州大学工学部）

1. まえがき

ここに述べようとする方法は，PC法(予測子修正子法[1]，Predictor-Corrector 法の略）の特徴である打切り誤差によって，積分のきざみ幅を自動的に制御して行こうとするものである．

2. 点傾斜法と PC 法との比較[2]

差分近似によって常微分方程式の初期値問題を数値的に解く方法には，よく知られているように，大別して二つの方法が主として用いられている．その一つは Runge-Kutta 法あるいは Runge-Kutta-Gill 法で代表される点傾斜法(point slope method）であって，これはある一点での値がわかっておれば，次の格子点での値が，その中間の点におけるいくつかの値を荷重平均して得られる，したがって，初期値のみわかっておれば直ちに計算をすすめることができ，きざみ幅も容易に変更することができる．また，記憶装置の数も少なくてすみ，プログラミングも比較的容易であるし，きざみ幅が適当に小さければ得られる解も安定である．しかし，計算中に誤差に関する情報が得られないため，折角きざみ幅の変更が容易な利点を持ちながらそれを利用できない．また，一刻み格子点を進めるのに導函数計算の回数が必ず4回必要であるため，それに非常な時間をとられる．実際問題では全体の計算時間の90%以上を導函数の計算に費すことが多いといわれる．もし Runge-Kutta 法あるいは Runge-Kutta-Gill 法で誤差を考えながらきざみ幅変更を行なうとすれば，たとえばきざみ幅 h で2ステップ（$2h$）だけ計算するのに8回，きざみ幅を倍にして $2h$ で1ステップ計算するのに3回，計11回行なってはじめて h の可否が決定できる[3]．したがって，この方法では相当の時間が費やされることになる．

一方 PC 法では，上とちょうど逆の得失を持っている．この方は公式が多点（multi-point）型で，求めようとする点の前のいくつかの値（前歴）を必要とするので，初期値から直ちに計算をはじめることができず，そのための補助手段が必要となる．また前歴を記憶しておくために，記憶装置が余計に要することにもなる．さらに微分を高次の差分でおきかえているため，修正子となる公式は安定性を要求される．もっとも安定性については，安定な公式を使えばすむことで，そうすれば特に問題とはならない．しかしながら，もし予測子と修正子の打切り誤差が同程度(同じ h の次数）で，きざみ幅 h が充分小さいならば，修正値と予測値との差から，修正子の持つ打切り誤差を見積ることができる．つまり Runge-Kutta 法などでは誤差に関する情報を得るためには，わざわざ別に計算せねばならないのに反し，この PC 法では計算の過程中に情報が得られるわけである．しかも，この方法では h を適当に選んで，各ステップでの反復回数を少くし，導函数計算の回数を各ステップ1～2回ですますことができて，そのための計算時間は短かくてすむ．しかしながら，折角誤差に関する情報を得ながらも，前歴を必要とするため，きざみ幅の変更は容易でないので，充分にその利点が生かされていないようである．

ここでは，それらに対する改良と思われる一方法を提案する．

3. 使用公式

この方法では，最初試みに与えたきざみ幅 h_p（たとえばプリント間隔）の適否を判断して，初期値より必要な出発値（前歴）を出す部分と，一般の積分進行の部分とに分けられる．

初期値より出発値を出すには Runge-Kutta 法でもよいが，一般の積分進行部分にも必要な公式のことも考慮に入れて，次のような2段とび法の一種（Clippinger-Dimsdale の方法に用いる公式）を用いる．

$$\left.\begin{aligned}
&\bar{P}: \quad y_1=y_0+hy_0' \\
&\bar{C}: \quad y_{1/2}=(y_0+y_1)/2+h(y_0'-y_1')/8 \\
&\bar{\bar{C}}: \quad y_1=y_0+h(y_0'+4y'_{1/2}+y_1')/6 \\
&T_{\bar{P}}\cong h^2y_0''/2,\ T_{\bar{C}}\cong h^4y^{(\mathrm{IV})}/(24\cdot 2^4),\\
&T_{\bar{\bar{C}}}\cong -h^5y^{(\mathrm{V})}/(90\cdot 2^5)
\end{aligned}\right\} \quad (1)$$

ここに解くべき微分方程式を

$$\frac{dy}{dx}=f(x,\ y) \quad (2)$$

とし，（′）は x についての y の導函数を示し，また $y^{(\mathrm{V})}$ は y の x に関する5階の導函数を表わす．その添字は原点 x_0 または $x_{1/2}=x_0+h/2$，$x_n=x_0+nh$（n は整数，この場合は1）における値を表わす．$T_{\bar{P}}$，$T_{\bar{C}}$ などはそれぞれの公式が持つ打切り誤差である．

一般の積分進行には次の公式の組み合せを用いる．

$$\left.\begin{aligned} P:\quad & y_{n+1}=4(y_{n-1}-y_n)+y_{n-1}\\ & \quad +2h(2y_n'+y_{n-1}'),\\ C:\quad & y_{n+1}=2y_n-y_{n-1}\\ & \quad +h(y'_{n+1}-y'_{n-1})/2,\\ & T_P\cong h^4y^{(\mathrm{IV})}/6,\ T_C\cong -h^4y^{(\mathrm{IV})}/12 \end{aligned}\right\}(3)$$

このとき $y^{(\mathrm{V})}$ がおとなしい函数ならば，T_C は次の式で見積ることができる．

$$T_C\cong C_0/3,\ C_0=y_{n+1}^{(1)}-y_{n+1}^{(0)}. \qquad (4)$$

ただしここの $y_{n+1}^{(1)}$ は公式 C による第1回の修正値，$y_{n+1}^{(0)}$ は公式 P による予測値である．

なお，打切り誤差が大きいとき，きざみ幅 h を半分にするためには，出発時に用いる公式のうちの C を内挿公式として用いる．また，多点型公式で問題となる安定性は，ここで用いる公式 C のみについて調べればよいが，これは梯形則と同じ性質を持ち安定であることがわかる．

4. 手順のあらまし

以下記述を簡単にするため，(2)のような一元一階の微分方程式の場合についての手順のあらましを述べる．多元連立あるいは高階の場合については次節のプログラムの項を参照されたい．

出発時

1. 初期値より導函数 y_0' を計算する．
2. $\bar{P}$ で y_1 を予測する．
3. 導函数 y_1' を計算．
4. $\bar{C}$ により $y_{1/2}$ を出す．
5. 導函数 $y_{1/2}'$ を計算．
6. $\bar{\bar{C}}$ により y_1 を出す．
7. y_1 の収束性を見る．収束していたら次へ，未収束ならば3へ．

（収束していないとき）

8. 収束した y_1 が，第1回目のときは無条件に次へ．第2回目以降のときは，きざみを半分にする前に出した $y_{1/2}$ と新しく収束した y_1 とを比べ，収束していない時次へ．収束していたら，出発値の計算は終り．

（$y_{1/2}$ と y_1 が前回の一致していないとき）

9. $y_{1/2}$ を出し，これを記憶するとともに，それを，新しいきざみ幅での y_1 とし，きざみ幅を半分として3へ．

（出発値計算終り）

このようにしてまず出発値 y_1 およびそのときのきざみ幅 h が得られる．このままでは得られた h が必ず h_P の半分以下になるが，もし h_P でも充分なときはここでそのための処置をほどこさずとも，積分進行の段階で直ちに回復されるであろう．

一般の積分進行の時

1. 導函数計算（最初は出発時に得られた y_1 について，一般には得られた y_n について．要すれば印刷）
2. y_{n-1}, y_n および導函数の値より P を用いて $y_{n+1}^{(0)}$ を出す．
3. 導函数 y'_{n+1} を計算．
4. C により修正値 $y_{n+1}^{(k)}$（$k=0, 1, 2, \cdots\cdots$）を出す．
5. 予測値との差 $C_0=y_{n+1}^{(1)}-y_{n+1}^{(0)}$ あるいは，くりかえしのときは $C_k=y_{n+1}^{(k+1)}-y_{n+1}^{(k)}$ を出す．
   もし最初の修正ならば次へ，それ以外の時は8へ．
6. 小さい正の数 ε_1 と C_0 とを比較．$|C_0|\leqq\varepsilon_1$ ならば，新しく得た x_{n+1} での値を x_n での値とし，きざみ幅 h を2倍にして1へ．$|C_0|>\varepsilon_1$ なら次へ．
7. $|C_0|$ と $\varepsilon_2(\gg\varepsilon_1$，たとえば $\varepsilon_2\cong\varepsilon_1\times 100)$ と比べ，$|C_0|\geqq\varepsilon_2$ ならば9．へ．$|C_0|<\varepsilon_2$ ならば次へ．
8. $|C_0|$ または $|C_k|$ と $\varepsilon(\varepsilon_1<\varepsilon<\varepsilon_2)$ を比べ，$|C_0|\leqq\varepsilon$ ならば，x_n での値を新しい x_{n-1} での値とし，いま得られた x_{n+1} での値を新しい x_n での値として1へ．$|C_k|>\varepsilon$ ならば3へ．
9. y_{n+1} を出すことを止め，y_n と y_{n-1} から，その中間点 $x_{n-1/2}$ における値を，出発時に用いた $\bar{C}$ で出す．きざみ幅 h を半分にして，内挿値を新しい y_{n-1} として2へ．

収束判定用の定数 ε_1, ε_2, および ε については上記の手順では $\varepsilon_1=0.1\varepsilon$, $\varepsilon_2=10\varepsilon$ の程度に選んだが，これは一応の目途として採用したものである．連立の場合は ε を各従属変数について原則として違った値をとることが望ましい．

なお，この方法では打切り誤差は $O(h^4)$ であって，Runge-Kutta 法などに比べ若干公式の精度は悪い．

5. プログラム

上記の手順をもとにして，連立方程式の場合についての ALGOL プログラムを作った．これを表に示す．また流れ図を 282 ページに示す．それらについての概略の説明を加える．なお用意すべきデータは，元数 N, 原点 x_0, 印刷間隔 h_p, 終点 x_{end}, 初期値 $y_{0i}(i=1, 2, \cdots\cdots)$ とそれに対応する収束判定用定数 ε_i であってこの順に用意する．結果は，まず $x_0(h_p)x_{end}$ を出し，次に x_0 における初期値，対応する導函数の値を印刷する．きざみ幅 h がきまれば MESIIWIDTH $H=h$ を出し，x_1 における値を印刷した後は x_0 での値からはじめて h_p ごとにその点における y と導函数を印刷するよう計画した．なお，きざみ幅が変るごとに新しい h を印刷する．これらの印刷は手続とした．また内挿公式 $\bar{C}$ は INTERPOL, 終点の判定は END TEST, 函数計算は FKT とし，どれもパラメータのない手続の形としている．FKT では Z[I] と F[I] を使用する．なお取扱う元数は一応 20 元までとした．

プログラムの始めは立札 START で示しており，出発値の計算は立札 ENDSTART の直前までである．詳細はプログラムと流れ図で読んでいただきたい．

表 プログラム

```
begin
  real   X, HP, XEND, H;
  array  Y0, Y1, YH, Z, F0, F1, F, EPS
         [1:20];
  integer I, N, D, E, S1, S2, COUNT;
  switch SW0:=MID, YHALF;
         SW1:=SIMP, NEXT;
         SW2:=RAISE2, TESTYH;
         SW3:=CONTINUE, END;
         SW4:=COTEST, LARGTEST;
         SW5:=HTWICE, ORDIN;
  procedure INTERPOL;
    begin for I:=1 step 1 until N do
      begin Z[I]:-(Y0[I]+Z[I])×0.5
                 +(F0[I]-F[I])×H×0.125;
            F1[I]:=F[I]
      end
    end;
  procedure ENDTEST;
    begin if H>0 & X≧XEND or H<0
          & X≦XEND
          then E:=2 else E:=1
    end;
  procedure PRINTH;
    begin CRLF; PRINTSTRING ('MESH # WI
                DTH ## H=');
          PRINTREAL(H);
          PRINTSTRING(':'); CRLF
    end;
  procedure READINITIAL;
    begin
      READREAL(X); READREAL(HP);
      READREAL(XEND);
      for I:=1 step 1 until N do
      READREAL(Y0[I]);
      for I:=1 step 1 until N do
      READREAL(EPS[I])
    end READINITIAL;
  procedure PRINTX;
    begin CRLF;
      PRINTSTRING('X='); PRINTREAL(X)
    end PRINTX;
  procedure PRINTYF;
    begin integer C; CRLF; C:=5;
      for I:=1 step 1 until N do
        begin if C=0 then begin C:=5;
        CRLF end else C:=C-1;
        PRINTREAL(Z[I]); PRINTSTRING('#');
        PRINTREAL(F[I]);PRINTSTRING('##')
        end
    end PRINTYF;
  procedure FKT;
      function subroutine use Z[I], F[I]
  START: READINTEGER(N); READINITIAL;
    for I:=1 step 1 until N do Z[I]:=Y0[I];
    H:=HP; CRLF; PRINTX;
    PRINTSTRING('('); PRINTREAL(HP);
    PRINTSTRING(')'); PRINTREAL(XEND);
    CRLF; PRINTX;
    S1:=S2:=1; FKT; PRINTYF;X:=X+H;
    for I:=1 step 1 until N do
      begin Y1[I]:=Z[I]:=F[I]×H+Y0[I];
            F0[I]:=F[I]
      end PREDICT Y1;
  REP1: FKT;
  INTER: INTERPOL; go to SW0[S1];
  YHALF: for I:=1 step 1 until N do
            Y1[I]:=YH[I]:=Z[I];
  MID:   X:=X-H/2.0; FKT; go to SW1[S1];
```

```
NEXT: H:=H/2.0; S1:=1; go to INTER;
SIMP: D:=0;
  for I:=1 step 1 until N do
  begin Z[I]:=(F[I]×4.0+F0[I]+F1[I])
              ×H/6.0+Y0[I];
  if ABS(Z[I]−Y1[I])≧EPS[I]then D:=1;
    Y1[I]:=Z[I]
  end SIMPSON;
  if D≠0 then go to REP2; go to SW2[S2];
  RAISE2: S2:=2;
  RAISE1: S1:=2;
REP2: X:=X+H/2.0; go to REP1;
TESTYH: for I:=1 step 1 until N do
    if ABS(Z[I]−YH[I])>EPS[I]then D:=1;
  if D=0 then go to ENDSTART;
  go to RAISE1;
ENDSTART: PRINTH; CRLF; X:=X+H/20;
  FKT;
WRITE: PRINTX; PRINTYF;
  COUNT:=ENTIER(HP/H)−1;
PREDICT: S1:=1; for I:=1 step1 until N do
  begin  Z[I]:=(F[I]×2.0+F0[I])×H×2.0
                +Y0[I]×5.0−Y1[I];
         F1[I]:=F[I]
  end;
  X:=X+H;
CORRECT: FKT; D:=0;
  for I:=1 step 1 until N do
    begin real Y2, TRUNC, DELTA;
          Y2:=(F[I]−F0[I])×H×0.5
      +Y1[I]×2.0−Y0[I]; TRUNC:=
      ABS(Y2−Z[I]); DELTA:=EPS[I];
      Z[I]:=Y2;  go to SW4[S1];
    COTEST:
      if TRUNC≦DELTA×0.1 then go to
      LOOPEND; S1:=2;
    LARGTEST: if TRUNC≧DELTA×10.0
    then go to HALV;
      if TRUNC>DELTA then D:=1;
    LOOPEND:
    end CORRECT;
  if D≠0 then go to CORRECT;
    go to SW5[S1];
HALVTWICE: if ABS(H)≧ABS(HP) or COUNT≠
    <COUNT÷2)×2
    then go to ORDIN;
  for I:=1 step 1 until N do Y1[I]:=Z[I];
  COUNT:=COUNT÷2; H:=H×2.0;
    PRINTH;
REPEAT: FKT; ENDTEST; go to SW3[E];
CONTINUE: if COUNT=0 then go to WRITE;
    COUNT:=COUNT−1; go to PREDICT;
ORDIN: for I:=1 step 1 until N do
    begin  Y0[I]:=Y1[I]; F0[I]:=F1[I];
           Y1[I]:=Z[I]
    end; go to REPEAT;
HALV: for I:=1 step 1 until N do
  begin Z[I]:=Y1[I]; F[I]:=F1[I] end;
  INTERPOL; H:=H/2.0; X:=X−H; FKT;
  for I:=1 step 1 until N do
  begin Y0[I]:=Z[I]; F0[I]:=F[I];
        F[I]:=F1[I]
  end;
  X:=X+H;  COUNT:=COUNT×2;
     PRINTH;
  go to PREDICT;
END: CRLF; CRLF; PRINTX; PRINTYF;
       CRLF; CRLF;
    PRINTSTRING('END');
    STOP:
end PROGRAMME;
```

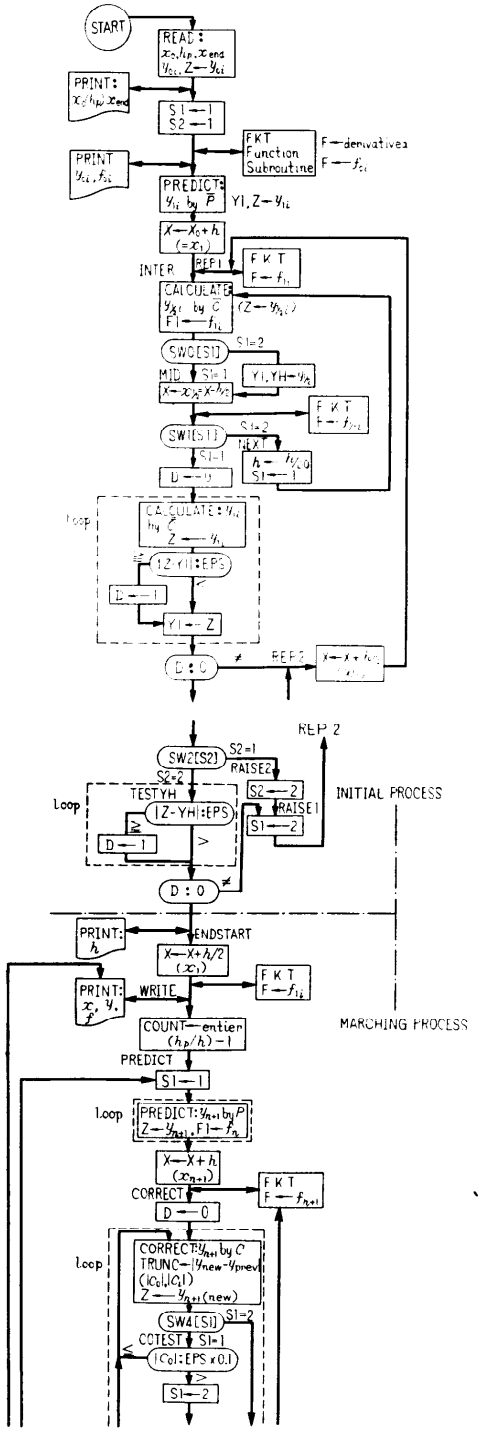
6. 結　　語

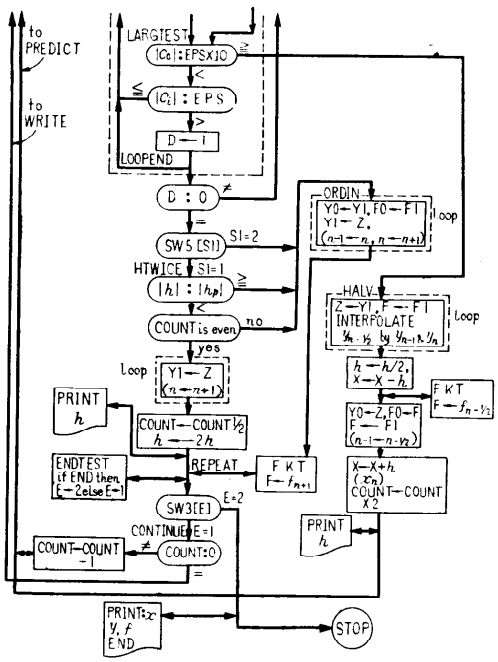
この考えは一昨年秋頃出したもので，これを昨年電気試験所数学研究室の戸田英雄氏がまず HIPAC 101 B（日本科学技術研修所計算センター所属）で SIP によりプログラムし，テストした[5]．次いで，東大数物系研究科応用物理大学院修士コースの中川友康君が OKITAC-5090 に対し，OKISIP でプログラムし，実用の段階にまで仕上げた．特にこのような形で報告する必要はないと考えて放っていたが，最近この方法について問合せがよくあるので，ALGOL の形でプログラムし報告する次第である．ただし，この形のままではまだテストされていないことをおことわりしておく．

面倒なプログラムをよくして下さった戸田英雄氏，中川友康君には深く感謝する．

参 考 文 献

1) 森口，高田：数値計算法 II（岩波講座現代応用数学，昭 33），§19.
2) Takata, M.: Proc. of 9 th Jap. Nat. Congr. for Appl. Mech., 1959, p. 463.
3) Anderson, W.H.: Comm. of ACM, **3**, 6 (1960) pp. 355~360.
4) Kunz, K.S.: Numerical Analysis, McGraw-Hill, 1957, p. 206.
5) 高田，戸田：CP 委員会資料（日本科学技術連盟，1961 年 5 月）


START
READ:
PRINT:
S1←1
S2←1
FKT
Function
Subroutine
F←derivatives
PRINT
PREDICT:
Y1, Z←
X←X0+h
INTER
REF1
CALCULATE
SW0[S1]
S1=2
MID
S1=1
Y1, YH←
SW1[S1]
NEXT
S1=1
D←1
Loop
CALCULATE
Z←
|Z-Y1|:EPS
D←1
Y1←Z
D:0
REP2
SW2[S2]
S2=1
RAISE2
S2=2
TEST YH
|Z-YH|:EPS
RAISE1
S1←2
INITIAL PROCESS
ENDSTART
X←X+h/2
WRITE
COUNT←entier
PREDICT
S1←1
Loop
PREDICT:
X←X+h
CORRECT
D←0
Loop
CORRECT:
TRUNC←
SW4[S1]
S1=2
COTEST
S1=1
|C0|:EPS×0.1
S1←2
MARCHING PROCESS


to PREDICT
to WRITE
LARGTEST
|C0|:EPS×10
|C0| : EPS
D←1
LOOPEND
D:0
ORDIN
Y0←Y1, F0←F1
Y1←Z.
loop
SW5[S1]
S1=2
HTWICE
S1=1
|h| : |hp|
COUNT is even
no
yes
HALV
Z←Y1, F←F1
INTERPOLATE
loop
Loop
Y1←Z
PRINT
h
h←h/2,
X←X-h
FKT
COUNT←COUNT½
h←2h
Y0←Z, F0←F
F←F1
ENDTEST
if END then
E←2 else E←1
REPEAT
FKT
X←X+h
COUNT←COUNT
×2
SW3[E]
E=2
CONTINUE
E=1
PRINT
h
COUNT←COUNT
-1
COUNT:0
PRINT:
Y, f
END
STOP

流 れ 図