ODE ソルバー DOPRI5 コードの使用法

廣田千明 (秋田県立大学システム科学技術学部)

平成 16 年 3 月 31 日

陽的な埋め込み型 Runge–Kutta 公式の一つとして Dormand–Prince 5(4) 公式が知られている [1, pp. 178–179]．この公式の実装である DOPRI5 コードは

http://www.unige.ch/math/folks/hairer/software.html

から Fortran 版，C 版，C++版がダウンロードできる．このコードは刻み幅を適応的に制御する機能を持ち合わせている．刻み幅制御の概念については，三井 [3, pp.51–56] を参照せよ．本稿ではこの DOPRI5 コードの使用法を説明する．ここでは C 版を用いることとし，cprog.tar をダウンロードしたとして説明を続ける．まず，圧縮ファイルを展開する．

```
% tar xvf cprog.tar
cprog/
cprog/aren.c
cprog/dop853.c
cprog/dop853.h
cprog/dopri5.c
cprog/dopri5.h
cprog/ret.c
cprog/retard.c
cprog/retard.h
cprog/vanderp.c
```

圧縮ファイル cprog.tar には DOP853 コードや RETARD コードも含まれているが，DOPRI5 に関係するファイルは，aren.c, dopri5.c, dopri5.h の 3 つである．aren.c は，DOPRI5 を用いて Arenstorf 軌道を計算するプログラムである．dopri5.c は dopri5 関数の定義ファイルであり，dopri5.h はヘッダファイルである．本稿の目的は DOPRI5 コードの使用法を示すことであるので，dopri5.c と dopri5.h に関しては解説しない．プログラムの解説は後で行うこととして，まずプログラムのコンパイル法と実行法について述べる．

コンパイルは

```
% gcc aren.c dopri5.c -lm
```

または

```
% gcc -c dopri5.c
% gcc aren.c dopri5.o -lm
```

とする．a.out という実行ファイルができるので，それを実行すれば良い．実行結果は

```
% ./a.out
x=0.000000  y=0.9940000000 0.0000000000  nstep=0
x=2.000000  y=-0.5798781411 0.6090775251  nstep=60
x=4.000000  y=-0.1983335270 1.1376380857  nstep=73
x=6.000000  y=-0.4735743943 0.2239068118  nstep=91
x=8.000000  y=-1.1745533505 -0.2759466982  nstep=110
x=10.000000  y=-0.8398073466 0.4468302268  nstep=122
x=12.000000  y=0.0131471247 -0.8385751499  nstep=145
x=14.000000  y=-0.6031129504 -0.9912598031  nstep=159
x=16.000000  y=0.2427110999 -0.3899948833  nstep=177
x=xend  y=0.9940021016 0.0000089112
rtol=0.0000001000   fcn=1442   step=240   accpt=216   rejct=22
```

となる．

Arenstorf 軌道を計算するプログラム `aren.c` の解説に移る．Arenstorf 軌道は 2 階の連立微分方程式

$$
\begin{aligned}
y_1'' &= y_1 + 2y_2' - \mu' \frac{y_1+\mu}{D_1} - \mu \frac{y_1+\mu'}{D_2},\\
y_2'' &= y_2 - 2y_1' - \mu' \frac{y_2}{D_1} - \mu \frac{y_2}{D_2}\\
D_1 &= ((y_1+\mu)^2 + y_2^2)^{3/2}, \quad D_2 = ((y_1-\mu')^2 + y_2^2)^{3/2},\\
\mu &= 0.012277471, \quad \mu' = 1-\mu
\end{aligned}
$$

の解 (y_1, y_2) の軌跡である [1, pp. 129–131]．`DOPRI5` コードは 1 階の常微分方程式系を解くコードであるから，この方程式を

$$
\begin{aligned}
Y_0 &= y_1,\\
Y_1 &= y_2,\\
Y_2 &= y_1',\\
Y_3 &= y_2'
\end{aligned}
$$

として，1 階の常微分方程式系

$$
\begin{aligned}
Y_0' &= Y_2,\\
Y_1' &= Y_3,\\
Y_2' &= Y_0 + 2Y_3 - \mu' \frac{Y_0+\mu}{D_1} - \mu \frac{Y_0+\mu'}{D_2},\\
Y_3' &= Y_1 - 2Y_2 - \mu' \frac{Y_1}{D_1} - \mu \frac{Y_1}{D_2}\\
D_1 &= ((Y_0+\mu)^2 + Y_1^2)^{3/2}, \quad D_2 = ((Y_0-\mu')^2 + Y_1^2)^{3/2},\\
\mu &= 0.012277471, \quad \mu' = 1-\mu
\end{aligned}
\tag{1}
$$

に変換する．このとき，方程式の本数は 4 本になる．プログラム `aren.c` 上では，6〜8 行目に

aren.c
```
6  #define  ndgl      4
7  #define  nrdens    2
8  #define  licont    nrdens
```

と定義されており，方程式の本数は `ndgl` で表される．また dense output の機能を用いて計算ステップ以外の

点での数値解を求める場合，その成分の個数を nrdens と定義している．ここで，dense output について簡単に説明する．通常，数値解は計算ステップ $x_n(n = 0, 1, 2, \ldots)$ 上で計算され，計算ステップの間の解は求められない．しかし，dense output が可能な公式では，計算ステップの間の解を少ない計算コストで計算することができる．解の外形を表示したいときに一定間隔で数値解を出力する必要があり，そのような場合に便利である．詳しくは [1, pp. 188–194] を参照のこと．

微分方程式 (1) の右辺は，void 型の関数 faren として，プログラム aren.c の 13〜30 行目に定義されている．

aren.c

```
13 void faren (unsigned n, double x, double *y, double *f)
14 {
15   double amu, amup, r1, r2, sqr;
16
17   amu = 0.012277471;
18   amup = 1.0 - amu;
19   f[0] = y[2];
20   f[1] = y[3];
21   sqr = y[0] + amu;
22   r1 = sqr*sqr + y[1]*y[1];
23   r1 = r1 * sqrt(r1);
24   sqr = y[0] - amup;
25   r2 = sqr*sqr + y[1]*y[1];
26   r2 = r2 * sqrt(r2);
27   f[2] = y[0] + 2.0 * y[3] - amup * (y[0]+amu) / r1 - amu * (y[0]-amup) / r2;
28   f[3] = y[1] - 2.0 * y[2] - amup * y[1] / r1 - amu * y[1] / r2;
29
30 } /* faren */
```

ここで，μ は変数 amu に格納され，μ' は変数 amup に格納されている．また，変数 r1, r2 には，それぞれ D_1, D_2 が格納されている．

プログラム aren.c の 33〜49 行目に定義されている solout 関数は数値解を出力する関数である．

aren.c

```
33 void solout (long nr, double xold, double x, double* y, unsigned n, int* irtrn)
34 {
35   static double xout;
36
37   if (nr == 1)
38   {
39     printf (format99, x, y[0], y[1], nr-1);
40     xout = x + 2.0;
41   }
42   else
43     while (x >= xout)
44     {
45       printf (format99, xout, contd5(0,xout), contd5(1,xout), nr-1);
46       xout += 2.0;
47     }
48
49 } /* solout */
```

このプログラムでは独立変数 x が 2 増えるごとに dense output の機能を用いて結果を出力している (contd5 関数により出力される)．この関数内では nr $-$ 1 がステップ数を表している．

aren.c

```
59   iout = 2;
```

iout は contd5 関数を用いて計算ステップ以外の点での数値解を計算するかどうかを指定するものである．iout の設定によって以下のように動作が変化する．

iout	
0	solout 関数は呼び出さない
1	solout 関数は計算ステップ上の解の出力のみに用いられる (contd5 関数を呼び出すことはできない)
2	contd5 関数を用いて計算ステップ間の解を計算することが可能

プログラムの 60 から 65 行目では初期値を設定している．

aren.c

```
60   x = 0.0;
61   y[0] = 0.994;
62   y[1] = 0.0;
63   y[2] = 0.0;
64   y[3] = -2.00158510637908252240537862224;
65   xend = 17.06521656015796255889172062 49;
```

変数 xend には x の最終値を設定する．

プログラムの 66〜70 行目では許容誤差を設定している．

aren.c

```
66   itoler = 0;
67   rtoler = 1.0E-7;
68   atoler = rtoler;
```

全成分に対して同じ許容誤差を用いる場合，許容誤差は

$$\texttt{rtoler} * \max(|y_{n,i}|, |y_{n+1,i}|) + \texttt{atoler}$$

で定義される [1, pp. 167–168]．ここで，$y_{n,i}$ は第 n ステップでの第 i 成分の数値解を現す．ここで，$\texttt{atoler} = 0$ なら上記の値は相対誤差を与えたことになり，$\texttt{rtoler} = 0$ のときは絶対誤差を与えたことになる．また itoler の値により次のように許容誤差の与え方が変わる．

itoler	
0	全成分に対して同じ許容誤差を用いる ($\texttt{rtoler} * \mathrm{abs}(y[i]) + \texttt{atoler}$)
1	各成分によって異なる許容誤差を用いる ($\texttt{rtoler}[i] * \mathrm{abs}(y[i]) + \texttt{atoler}[i]$)

aren.c

```
69   icont[0] = 0;
70   icont[1] = 1;
```

icont は dense output の機能を用い計算ステップ以外の点で数値解を計算する成分を指定している．この場合，contd5$(0, s)$ で $Y_0(s)$ の数値解が出力され，contd5$(1, s)$ で $Y_1(s)$ の数値解が出力される．

プログラム 72 行目で dopri5 関数が呼び出されている．

aren.c

```
72   res = dopri5 (ndgl, faren, x, y, xend, &rtoler, &atoler, itoler, solout,
iout, stdout, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0, 0, 0, ndgl, NULL, licont);
```

関数の引数を理解するために dopri5 関数の宣言を以下に与える．

dopri5.h

```
extern int dopri5
 (unsigned n,       /* dimension of the system <= UINT_MAX-1*/
  FcnEqDiff fcn,    /* function computing the value of f(x,y) */
  double x,         /* initial x-value */
  double* y,        /* initial values for y */
  double xend,      /* final x-value (xend-x may be positive or negative) */
  double* rtoler,   /* relative error tolerance */
  double* atoler,   /* absolute error tolerance */
  int itoler,       /* switch for rtoler and atoler */
  SolTrait solout, /* function providing the numerical solution during integration */
  int iout,         /* switch for calling solout */
  FILE* fileout,    /* messages stream */
  double uround,    /* rounding unit */
  double safe,      /* safety factor */
  double fac1,      /* parameters for step size selection */
  double fac2,
  double beta,      /* for stabilized step size control */
  double hmax,      /* maximal step size */
  double h,         /* initial step size */
  long nmax,        /* maximal number of allowed steps */
  int meth,         /* switch for the choice of the coefficients */
  long nstiff,      /* test for stiffness */
  unsigned nrdens, /* number of components for which dense outpout is required */
  unsigned* icont, /* indexes of components for which dense output
                      is required, >= nrdens */
  unsigned licont  /* declared length of icon */
 );
```

引数を簡単に説明すると以下のようになる．

fileout	出力先ストリーム
uround	丸めの単位．1.0 + uround > 1.0 を満たす最小の数．デフォルトは 2.3e-16
safe	安全係数 [1, p. 168]．デフォルトは 0.9
beta	安定係数 [2, pp. 24–31]．デフォルトは 0.04
hmax	刻み幅の最大値．デフォルトは xend − x
nmax	ステップ数の最大値．デフォルトは 100000 ステップ
meth	この値を変更することで他の解法に切替えできるようにプログラムされているが現在は未使用
nstiff	stiffness の判定基準．デフォルトは 1000 ステップ

またこの関数の戻り値は，以下の通りである．

1	computation successful
2	computation successful interrupted by solout
-1	input is not consistent
-2	larger nmax is needed
-3	step size becomes too
-4	the problem is probably stff (interrupted)

プログラム 74〜76 行目は結果の表示である．

aren.c

```
74   printf ("x=xend  y=%12.10f %12.10f\r\n", y[0], y[1]);
75   printf ("rtol=%12.10f   fcn=%li   step=%li   accpt=%li   rejct=%li\r\n",
76   rtoler, nfcnRead(), nstepRead(), naccptRead(), nrejctRead());
```

ここに登場する関数は以下の通りである．

nfcnRead	関数評価回数を出力する関数
nstepRead	ステップ数を出力する関数
naccptRead	アクセプトの回数を出力する関数
nrejctRead	リジェクトの回数を出力する関数

数値例として，微分方程式の初期値問題

$$\frac{dy}{dx} = -y, \quad y(0) = 1$$

を考える．まず最初に dense output の機能を用いて，0.1 刻みに解を出力するプログラムをプログラムリスト 2(exam1.c) に与える．実行結果は以下のようになる．

```
% gcc exam1.c dopri5.c -lm
% ./a.out
x=0.000e+00 y=1.000e+00 step=0
x=1.000e-01 y=9.048e-01 step=2
x=2.000e-01 y=8.187e-01 step=3
x=3.000e-01 y=7.408e-01 step=4
x=4.000e-01 y=6.703e-01 step=5
x=5.000e-01 y=6.065e-01 step=6
x=6.000e-01 y=5.488e-01 step=6
x=7.000e-01 y=4.966e-01 step=7
x=8.000e-01 y=4.493e-01 step=7
x=9.000e-01 y=4.066e-01 step=8
x=1.000e-00 y=3.679e-01 step=9
rtol=1.000e-07   fcn=56   step=9   accpt=9   rejct=0
```

次にステップサイズを表示するプログラムをプログラムリスト 3 (exam2.c) に与える．実行結果は以下の通りである．

```
% gcc exam2.c dopri5.c -lm
% ./a.out
step=1 x=1.821e-02 h=1.821e-02
step=2 x=1.013e-01 h=8.310e-02
step=3 x=2.054e-01 h=1.041e-01
step=4 x=3.381e-01 h=1.328e-01
step=5 x=4.831e-01 h=1.449e-01
step=6 x=6.389e-01 h=1.558e-01
step=7 x=8.009e-01 h=1.621e-01
step=8 x=9.685e-01 h=1.676e-01
step=9 x=1.000e+00 h=3.147e-02
```

刻み幅は `hRead` 関数で，x の値は `xRead` 関数で知ることができる．

参考文献

[1] Hairer, E., Nørsett, S.P., and Wanner, G., Solving Ordinary Differential Equations I, 2nd rev. ed., Springer, Berlin, 2000.

[2] Hairer, E. and Wanner, G., Solving Ordinary Differential Equations II, 2nd rev. ed., Springer, Berlin, 1996.

[3] 三井斌友，微分方程式の数値解法 I，岩波講座応用数学 [方法 3]，岩波書店，1993.

プログラムリスト 1　aren.c

```
 1  #include <math.h>
 2  #include <stdio.h>
 3  #include "dopri5.h"
 4
 5
 6  #define  ndgl      4
 7  #define  nrdens    2
 8  #define  licont    nrdens
 9
10  char format99[] = "x=%f  y=%12.10f %12.10f  nstep=%li\r\n";
11
12
13  void faren (unsigned n, double x, double *y, double *f)
14  {
15    double amu, amup, r1, r2, sqr;
16
17    amu = 0.012277471;
18    amup = 1.0 - amu;
19    f[0] = y[2];
20    f[1] = y[3];
21    sqr = y[0] + amu;
22    r1 = sqr*sqr + y[1]*y[1];
23    r1 = r1 * sqrt(r1);
24    sqr = y[0] - amup;
25    r2 = sqr*sqr + y[1]*y[1];
26    r2 = r2 * sqrt(r2);
27    f[2] = y[0] + 2.0 * y[3] - amup * (y[0]+amu) / r1 - amu * (y[0]-amup) / r2;
28    f[3] = y[1] - 2.0 * y[2] - amup * y[1] / r1 - amu * y[1] / r2;
29
30  } /* faren */
31
32
33  void solout (long nr, double xold, double x, double* y, unsigned n, int* irtrn)
34  {
35    static double xout;
36
37    if (nr == 1)
38    {
39      printf (format99, x, y[0], y[1], nr-1);
40      xout = x + 2.0;
41    }
42    else
43      while (x >= xout)
44      {
45        printf (format99, xout, contd5(0,xout), contd5(1,xout), nr-1);
46        xout += 2.0;
47      }
48
49  } /* solout */
50
51
52  int main (void)
53  {
54    double   y[ndgl];
55    unsigned icont[licont], i;
56    int      res, iout, itoler;
57    double   x, xend, atoler, rtoler;
58
59    iout = 2;
```

```
60    x = 0.0;
61    y[0] = 0.994;
62    y[1] = 0.0;
63    y[2] = 0.0;
64    y[3] = -2.00158510637908252240537862224;
65    xend = 17.06521656015796255889172062 49;
66    itoler = 0;
67    rtoler = 1.0E-7;
68    atoler = rtoler;
69    icont[0] = 0;
70    icont[1] = 1;
71
72    res = dopri5 (ndgl, faren, x, y, xend, &rtoler, &atoler, itoler, solout, iout, stdout,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0, 0, 0, ndgl, NULL, licont);
73
74    printf ("x=xend  y=%12.10f %12.10f\r\n", y[0], y[1]);
75    printf ("rtol=%12.10f   fcn=%li   step=%li   accpt=%li   rejct=%li\r\n",
76    rtoler, nfcnRead(), nstepRead(), naccptRead(), nrejctRead());
77
78    return 0;
79
80  } /* main */
```

プログラムリスト 2　exam1.c

```
 1  #include <math.h>
 2  #include <stdio.h>
 3  #include "dopri5.h"
 4
 5
 6  #define  ndgl      1
 7  #define  nrdens    1
 8  #define  licont    nrdens
 9
10  void func (unsigned n, double x, double *y, double *f)
11  {
12  f[0] = -y[0];
13  }
14
15  void solout (long nr, double xold, double x, double* y, unsigned n, int* irtrn)
16  {
17    static double xout,y1;
18
19    if (nr == 1)
20    {
21  printf ("x=%9.3e y=%9.3e step=%d\n",x, y[0],nr-1);
22      xout = x + 0.1;
23    }
24    else
25      while (x >= xout)
26  {
27  printf ("x=%9.3e y=%9.3e step=%d\n",xout, contd5(0,xout),nr-1);
28  xout += 0.1;
29  }
30
31  } /* solout */
32
33  int main (void)
34  {
35    double   y[ndgl];
36    unsigned icont[licont], i;
37    int      res, iout, itoler;
38    double   x, xend, atoler, rtoler;
39
40    iout = 2;
41    x = 0.0;
42    y[0] = 1.0;
43    xend = 1.0;
44    itoler = 0;
45    rtoler = 1.0E-7;
46    atoler = rtoler;
47    icont[0] = 0;
48
49    res = dopri5 (ndgl, func, x, y, xend, &rtoler, &atoler, itoler, solout, iout,
50  stdout, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0, 0, 0, nrdens, NULL, licont);
51
52    printf ("rtol=%9.3e   fcn=%li   step=%li   accpt=%li   rejct=%li\r\n",
53  rtoler, nfcnRead(), nstepRead(), naccptRead(), nrejctRead());
54
55    return 0;
56
57  } /* main */
```

プログラムリスト 3　exam2.c

```
 1  #include <math.h>
 2  #include <stdio.h>
 3  #include "dopri5.h"
 4
 5
 6  #define  ndgl      1
 7  #define  nrdens    1
 8  #define  licont    nrdens
 9
10  void func (unsigned n, double x, double *y, double *f)
11  {
12  f[0] = -y[0];
13  }
14
15  void solout (long nr, double xold, double x, double* y, unsigned n, int* irtrn)
16  {
17    static double xout,y1;
18
19    if (nr>1)
20  printf ("step=%d x=%9.3e h=%9.3e\n",nr-1, xRead(),hRead());
21
22  } /* solout */
23
24  int main (void)
25  {
26    double   y[ndgl];
27    unsigned icont[licont], i;
28    int      res, iout, itoler;
29    double   x, xend, atoler, rtoler;
30
31    iout = 1;
32    x = 0.0;
33    y[0] = 1.0;
34    xend = 1.0;
35    itoler = 0;
36    rtoler = 1.0E-7;
37    atoler = rtoler;
38    icont[0] = 0;
39
40    res = dopri5 (ndgl, func, x, y, xend, &rtoler, &atoler, itoler, solout, iout,
41  stdout, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0, 0, 0, nrdens, NULL, licont);
42
43    return 0;
44
45  } /* main */
```