「微分方程式と計算機演習 (前半 8 章) のプログラム

桂田 祐史

2018 年 4 月 8 日, 2018 年 6 月 8 日

随時、修正・更新するので、最新版

http://nalab.mind.meiji.ac.jp/2018/q2/kikuchi-yamamoto-progs/ (HTML)
http://nalab.mind.meiji.ac.jp/2018/q2/kikuchi-yamamoto-progs.pdf (PDF)

もチェックして下さい。

目 次

(元の FORTRAN プログラムでは、単精度浮動小数点数を用いていたが、C プログラムでは、倍精度浮動小数点数を用いている。そのため計算結果に違いが生じることがある。特に反復計算では、EPS の値はかなり小さく取れる。)

1 連立 1 次方程式の解法 (1) 直接法

1.1 p13 (プログラムリスト 1)

13 ページに載っているプログラムで、14 ページの計算実習 1 をやってみよう。

```
% gfortran -O -o p13 p13.f
% ./p13
 INPUT : N=
5
 INPUT : A(  1,   1)=
1
 INPUT : A(  1,   2)=
1
 INPUT : A(  1,   3)=
1
 INPUT : A(  1,   4)=
1
 INPUT : A(  1,   5)=
1
 INPUT : A(  2,   1)=
1
 INPUT : A(  2,   2)=
2
 INPUT : A(  2,   3)=
3
 INPUT : A(  2,   4)=
4
 INPUT : A(  2,   5)=
5
(中略)
 INPUT : A(  5,   5)=
625
 INPUT : F(  1)=
15
 INPUT : F(  2)=
35
```

```
 INPUT : F(  3)=
105
 INPUT : F(  4)=
371
 INPUT : F(  5)=
1449
   I     X(I)      I     X(I)      I     X(I)      I     X(I)      I     X(I)
   1  5.0000E+00   2  4.0000E+00   3  3.0000E+00   4  2.0000E+00   5  1.0000E+00
%
```

$x = (5, 4, 3, 2, 1)$ となっている。

1.2 p20 (プログラムリスト 2)

20 ページに載っているプログラムで、19 ページの計算実習 2 をやってみる。21 ページの [解] と比べてどうか？

```
% gfortran -O -o p20 p20.f
% ./p20
 INPUT : N=
20
 INPUT : B(I),C(I) FOR I=1
2 -1
 INPUT : A(I),B(I),C(I) FOR I=  2
-1 2 -1
 INPUT : A(I),B(I),C(I) FOR I=  3
-1 2 -1
 INPUT : A(I),B(I),C(I) FOR I=  4
(中略)
 INPUT : A(I),B(I),C(I) FOR I= 19
-1 2 -1
 INPUT : A(I),B(I) FOR I=N
-1 2
 INPUT : F(  1)=
1
 INPUT : F(  2)=
0
(中略)
 INPUT : F( 19)=
0
 INPUT : F( 20)=
0
0  I     X(I)      I     X(I)      I     X(I)      I     X(I)      I     X(I)
   1  9.5238E-01   2  9.0476E-01   3  8.5714E-01   4  8.0952E-01   5  7.6190E-01
   6  7.1429E-01   7  6.6667E-01   8  6.1905E-01   9  5.7143E-01  10  5.2381E-01
  11  4.7619E-01  12  4.2857E-01  13  3.8095E-01  14  3.3333E-01  15  2.8571E-01
  16  2.3810E-01  17  1.9048E-01  18  1.4286E-01  19  9.5238E-02  20  4.7619E-02
%
```

結果は一致しているようである。

1.3 p29 (プログラムリスト 3)

29 ページに載っているプログラムで、(やはり) 19 ページの計算実習 2 をやってみる。

```
\begin{verbatim}
% gfortran -O -o p29 p29.f
% ./p29
```

```
 INPUT : N=
20
 INPUT : B(I) FOR I=1
2
 INPUT : A(I),B(I) FOR I=  2
-1 2
 INPUT : A(I),B(I) FOR I=  3
-1 2
 INPUT : A(I),B(I) FOR I=  4
-1 2
(中略)
 INPUT : A(I),B(I) FOR I= 19
-1 2
 INPUT : A(I),B(I) FOR I= 20
-1 2
 INPUT : F(  1)=
1
 INPUT : F(  2)=
0
(中略)
 INPUT : F( 19)=
0
 INPUT : F( 20)=
0
   I     X(I)      I     X(I)      I     X(I)      I     X(I)      I     X(I)
   1  9.5238E-01   2  9.0476E-01   3  8.5714E-01   4  8.0952E-01   5  7.6190E-01
   6  7.1429E-01   7  6.6667E-01   8  6.1905E-01   9  5.7143E-01  10  5.2381E-01
  11  4.7619E-01  12  4.2857E-01  13  3.8095E-01  14  3.3333E-01  15  2.8571E-01
  16  2.3810E-01  17  1.9048E-01  18  1.4286E-01  19  9.5238E-02  20  4.7619E-02
%
```

これも結果は一致しているようである。

2　連立 1 次方程式の解法 (2)

2.1　p39 (プログラムリスト 1)

39 ページのプログラムで、40 ページの計算実習 1 をやってみよう。
まず JACOBI 法の場合は次のようになった。

```
% ./p39
 INPUT : N=
3
 INPUT : M=
100
 INPUT : EPS=
1e-5
 JACOBI : METHOD=0, SOR : =1
 INPUT : METHOD=
0
 INPUT : A(  1,  1)=
2
 INPUT : A(  1,  2)=
-1
 INPUT : A(  1,  3)=
0
 INPUT : A(  2,  1)=
-1
 INPUT : A(  2,  2)=
2
 INPUT : A(  2,  3)=
-1
 INPUT : A(  3,  1)=
0
 INPUT : A(  3,  2)=
-1
 INPUT : A(  3,  3)=
2
 INPUT : F(  1)=
1
 INPUT : F(  2)=
0
 INPUT : F(  3)=
0
 INPUT : X(  1)=
0
 INPUT : X(  2)=
0
 INPUT : X(  3)=
0
 (NITER,ERROR)=           1   1.00000000
 (NITER,ERROR)=           2  0.500000000
 (NITER,ERROR)=           3  0.200000003
 (NITER,ERROR)=           4  0.200000003
 (NITER,ERROR)=           5   9.09090936E-02
(中略)
 (NITER,ERROR)=          31   1.01726291E-05
 (NITER,ERROR)=          32   1.01726291E-05
 (NITER,ERROR)=          33   5.08628909E-06
ONO. OF ITERATOINS= 33   EPS=  1.0000E-05
0  I     X(I)      I     X(I)      I     X(I)      I     X(I)      I     X(I)
   1  7.5000E-01   2  4.9999E-01   3  2.5000E-01
%
```

確かに反復回数 33 回で停止した。真の解と比べてどうだろうか？
次に SOR 法の場合は

```
[chronos:q1/progs/chapter2] mk% ./p39
 INPUT : N=
3
 INPUT : M=
100
 INPUT : EPS=
1E-5
 JACOBI : METHOD=0, SOR : =1
 INPUT : METHOD=
1
 INPUT : OMEGA=
0.3
 INPUT : A(  1,  1)=
2
 INPUT : A(  1,  2)=
-1
 INPUT : A(  1,  3)=
0
 INPUT : A(  2,  1)=
-1
 INPUT : A(  2,  2)=
2
 INPUT : A(  2,  3)=
-1
 INPUT : A(  3,  1)=
0
 INPUT : A(  3,  2)=
-1
 INPUT : A(  3,  3)=
2
 INPUT : F(  1)=
1
 INPUT : F(  2)=
0
 INPUT : F(  3)=
0
 INPUT : X(  1)=
0
 INPUT : X(  2)=
0
 INPUT : X(  3)=
0
 (NITER,ERROR)=           1   1.00000000
 (NITER,ERROR)=           2  0.419448495
 (NITER,ERROR)=           3  0.238088921
(中略)
 (NITER,ERROR)=          87   1.00540037E-05
 (NITER,ERROR)=          88   9.06046444E-06
ONO. OF ITERATOINS= 88   EPS=  1.0000E-05
0  I     X(I)      I     X(I)      I     X(I)      I     X(I)      I     X(I)
   1  7.4995E-01   2  4.9994E-01   3  2.4996E-01
%
```

確かに反復回数 88 回で停止した。

3　2階の常微分方程式の境界値問題

3.1　p58 (プログラムリスト 1)

FINITE DIFFERENCE METHOD FOR -FNU*D2U/DX2=F
58 ページのプログラムで、57 ページの計算実習 1 をやってみよう。

```
% ./p58
 INPUT : N=
10
0MESH SIZE=  1.0000E-01
0** NODAL VALUES OF SOLUTION **
   I     U(I)      I     U(I)      I     U(I)      I     U(I)      I     U(I)
   1  9.0000E-02   2  1.7000E-01   3  2.4000E-01   4  3.0000E-01   5  3.5000E-01
   6  3.9000E-01   7  4.2000E-01   8  4.4000E-01   9  4.5000E-01  10  4.5000E-01
%
```

この結果を 59 ページの表と比べてみよう。

3.2　p65 (プログラムリスト 2)

FINITE DIFFERENCE METHOD FOR -FNU*D2U/DX2+DU/DX=F
65 ページのプログラムで、67 ページの計算実習 2 をやってみよう。

```
% ./p65
 INPUT : FNU=
1
 INPUT : N=
10
 BACKWARD : METHOD=1, CENTRAL : =2, FORWARD : =3
 INPUT : METHOD=
2
0DIFFUSIVITY=  1.0000E+00   MESH SIZE=  1.0000E-01   METHOD= 2
0** NODAL VALUES OF SOLUTION **
   I     U(I)      I     U(I)      I     U(I)      I     U(I)      I     U(I)
   1  3.0045E-02   2  5.2727E-02   3  6.8835E-02   4  7.8861E-02   5  8.3001E-02
   6  8.1161E-02   7  7.2925E-02   8  5.7525E-02   9  3.3777E-02
```

テキストには数値が与えられていないので、グラフを描いて、68 ページの図と比べてみる必要がある。

3.3　p73 (プログラムリスト 3)

73 ページのプログラムで、71 ページの演習問題 1 をやってみよう。

```
% ./p73
 INPUT : N=
10
 INPUT : M=
1000
 INPUT : EPS=
1E-5
 JACOBI : METHOD=0, SOR : =1
 INPUT : METHOD=
0
 (NITER,ERROR)=           1   1.00000000
 (NITER,ERROR)=           2  0.500000000
 (NITER,ERROR)=           3  0.333333373
```

```
 (NITER,ERROR)=           4  0.250000030
 (NITER,ERROR)=           5  0.200000092
(中略)
 (NITER,ERROR)=         580   1.06710586E-05
 (NITER,ERROR)=         581   1.04721094E-05
 (NITER,ERROR)=         582   1.04720166E-05
 (NITER,ERROR)=         583   1.02067979E-05
 (NITER,ERROR)=         584   1.02067088E-05
 (NITER,ERROR)=         585   9.94150014E-06
0NO. OF ITERATIONS= 585   EPS=  1.0000E-05
0MESH SIZE=  1.0000E-01
0** NODAL VALUES OF SOLUTION **
   I     U(I)      I     U(I)      I     U(I)      I     U(I)      I     U(I)
   1  8.9947E-02   2  1.6990E-01   3  2.3985E-01   4  2.9980E-01   5  3.4976E-01
   6  3.8973E-01   7  4.1970E-01   8  4.3968E-01   9  4.4967E-01  10  4.4966E-01
%
```

テキストには589回で停止したと書いてあったが、585回で停止し、得られた解もわずかに異なっている。ちなみにCに書き換えたプログラムは587回で停止した。

3.4 p75 (プログラムリスト4)

FINITE DIFFERENCE METHOD FOR -FNU*D2U/DX2+DU/DX=F

75ページのプログラムで、71ページの演習問題2をやってみよう。

```
% ./p75
 INPUT : FNU=
1
 INPUT : N=
10
 BACKWARD : METHOD=1,CENTRAL : =2,FORWARD : =3
 INPUT : METHOD=
2
0DIFFUSIVITY=  1.0000E+00   MESH SIZE=  1.0000E-01   METHOD= 2
0** NODAL VALUES OF SOLUTION **
   I     U(I)      I     U(I)      I     U(I)      I     U(I)      I     U(I)
   1  6.1405E-02   2  1.1875E-01   3  1.7160E-01   4  2.1949E-01   5  2.6189E-01
   6  2.9823E-01   7  3.2787E-01   8  3.5011E-01   9  3.6415E-01  10  3.6915E-01
%
```

これは $N = 10, \nu = 1$ として中心差分近似を用いて計算したものであるが、77ページの表にある結果と一致しているだろうか？

4　固有値問題 (1) — ベキ乗法 —

4.1　p88 (プログラムリスト 1)

88 ページのプログラムで、89–90 ページの計算実習 1 をやってみよう。

```
% ./p88
 INPUT : N=
4
 INPUT : EPS=
1E-5
 INPUT : M=
100
 INPUT : A(  1,  1)=
5
 INPUT : A(  1,  2)=
4
 INPUT : A(  1,  3)=
1
 INPUT : A(  1,  4)=
1
(中略)
 INPUT : A(  4,  1)=
1
 INPUT : A(  4,  2)=
1
 INPUT : A(  4,  3)=
2
 INPUT : A(  4,  4)=
4
 INPUT : X(  1)=
1
 INPUT : X(  2)=
0
 INPUT : X(  3)=
0
 INPUT : X(  4)=
0
 (NITER,EVAL,ERROR=           1   9.60465145       1.00000000
 (NITER,EVAL,ERROR=           2   9.92197895       3.19822803E-02
 (NITER,EVAL,ERROR=           3   9.98053360       5.86688565E-03
 (NITER,EVAL,ERROR=           4   9.99512196       1.45954755E-03
 (NITER,EVAL,ERROR=           5   9.99878025       3.65874090E-04
 (NITER,EVAL,ERROR=           6   9.99969292       9.12694377E-05
 (NITER,EVAL,ERROR=           7   9.99992371       2.30790938E-05
 (NITER,EVAL,ERROR=           8   9.99998188       5.81742370E-06
0NO. OF ITERATIONS=  8   EPS=  1.0000E-05
0EIGENVALUE=  1.0000E+01
0** EIGENVECTOR **
   I     X(I)      I     X(I)      I     X(I)      I     X(I)      I     X(I)
   1  6.3307E-01   2  6.3307E-01   3  3.1499E-01   4  3.1499E-01
[chronos:q1/progs/chapter4] mk%
```

この結果が 90 ページの記述と合っているだろうか？

5　固有値問題 (2) —逆反復法およびシフト法 —

5.1　p98 (プログラムリスト 1)

98 ページのプログラムで、97–100 ページの計算実習 1 をやってみる。

```
% ./p98
 INPUT : N=
4
 INPUT: EPS=
1E-5
 INPUT : M =
100
 INPUT : A(  1,  1)=
5
 INPUT : A(  1,  2)=
4
 INPUT : A(  1,  3)=
1
 INPUT : A(  1,  4)=
1
(中略)
 INPUT : A(  4,  1)=
1
 INPUT : A(  4,  2)=
1
 INPUT : A(  4,  3)=
2
 INPUT : A(  4,  4)=
4
 INPUT:X(  1)=
1
 INPUT:X(  2)=
1
 INPUT:X(  3)=
1
 INPUT:X(  4)=
1
 (NITER,EVAL,ERROR)=            1   8.46153831        1.00000000
 (NITER,EVAL,ERROR)=            2   6.80000019       0.244343832
 (NITER,EVAL,ERROR)=            3   5.61643887       0.210731626
(中略)
 (NITER,EVAL,ERROR)=           16   1.00001359        3.26867419E-04
 (NITER,EVAL,ERROR)=           17   1.00000048        1.31130155E-05
 (NITER,EVAL,ERROR)=           18  0.999999940        5.36441860E-07
0NO. OF ITERATIONS =  18   EPS=  1.0000E-05
0EIGENVALUE=  1.0000E+00
0**EIGENVECTOR**
   I     X(I)      I     X(I)      I     X(I)      I     X(I)      I     X(I)
   1  7.0708E-01   2 -7.0713E-01   3  5.4529E-05   4  3.8838E-05
%
```

これが 97 ページの記述と合っているか。微妙に食い違っている ??

$$A = \begin{pmatrix} 5 & 4 & 1 & 1 \\ 4 & 5 & 1 & 1 \\ 1 & 1 & 4 & 2 \\ 1 & 1 & 2 & 4 \end{pmatrix}$$

$$\boldsymbol{x}_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

から始めると、すぐに $\boldsymbol{\varphi} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} 0 \\ 0 \\ 0 \end{pmatrix}$ に近く。

$$\boldsymbol{x}_0 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

から始めると、本当は $(\varphi, \boldsymbol{x}_0) = 0$ なので、φ に収束するはずはないが、「実際には上記のように反復回数は大きいが正しい値に収束した.」とテキストに書いてあるが、C で書いたプログラムではそうはならなかった。これは単精度と倍精度の違いであろう。試しに double を float にしたら、FORTRAN プログラムと同じ結果を出力した。

5.2 p104 (プログラムリスト 2)

104 ページのプログラムで、103–106 ページの計算実習 2 をやってみよう。

```
% ./p104
 INPUT : N=
4
 INPUT : SHIFT=
0
 INPUT : EPS=
1E-6
 INPUT : M=
100
 INPUT : A(  1,  1)=
5
 INPUT : A(  1,  2)=
4
 INPUT : A(  1,  3)=
1
 INPUT : A(  1,  4)=
1
(中略)
 INPUT : A(  4,  1)=
1
 INPUT : A(  4,  2)=
1
 INPUT : A(  4,  3)=
2
 INPUT : A(  4,  4)=
4
 INPUT : X(  1)=
1
 INPUT : X(  2)=
0
 INPUT : X(  3)=
0
 INPUT : X(  4)=
```

```
0
 (NITER,EVAL,ERROR)=           1   1.10236216       1.00000000
 (NITER,EVAL,ERROR)=           2   1.00199914      0.100162774
 (NITER,EVAL,ERROR)=           3   1.00005829       1.94073329E-03
 (NITER,EVAL,ERROR)=           4   1.00000215       5.61474553E-05
 (NITER,EVAL,ERROR)=           5   1.00000012       2.02655769E-06
 (NITER,EVAL,ERROR)=           6  0.999999881       2.38418608E-07
0NO. OF ITERATIONS=  6   EPS=  1.0000E-06   SHIFT=  0.0000E+00   NPIVOT=  0
0EIGENVALUE=  1.0000E+00
0** EIGENVECTOR **
   I     X(I)      I     X(I)      I     X(I)      I     X(I)      I     X(I)
   1  7.0712E-01   2 -7.0710E-01   3 -1.7818E-05   4 -1.7818E-05
```

ここでは 106 ページの ①の場合を計算している。106 ページの記述と合っているだろうか？

6　常微分方程式の固有値問題

6.1　p125 (プログラムリスト 1)

125 ページのプログラム

6.2　p132 (プログラムリスト 2)

125 ページのプログラム

7　常微分方程式の初期値問題 (1)

7.1　p146 (プログラムリスト 1)

146 ページのプログラムで、147–1??ページの計算実習 2 をやってみる。

```
% ./p146
 INPUT : NFUNC=
1
 EULER : METHOD=1
 HEUN  :       =2
 RUNGE-KUTTA : =3
 INPUT : METHOD=
2
 INPUT : T0=
0
 INPUT : X0=
1
 INPUT : H=
0.1
 INPUT : TMAX=
1
 T=  1.0000E-01  X=  1.1050E+00
 T=  2.0000E-01  X=  1.2210E+00
 T=  3.0000E-01  X=  1.3492E+00
 T=  4.0000E-01  X=  1.4909E+00
 T=  5.0000E-01  X=  1.6474E+00
 T=  6.0000E-01  X=  1.8204E+00
 T=  7.0000E-01  X=  2.0116E+00
 T=  8.0000E-01  X=  2.2228E+00
 T=  9.0000E-01  X=  2.4562E+00
 T=  1.0000E+00  X=  2.7141E+00
%
```

この結果を 148 ページの数表と比べるとどうか。

8　常微分方程式の初期値問題 (2)

8.1　p164 (プログラムリスト 1)

164 ページのプログラムで、166–167 ページの計算実習 1 をやってみよう。

```
% ./p164
 LINEAR SYSTEM        : NFUNC=1
 VAN DER POL EQUATION :      =2
 EXERCISE-2           :      =3
 INPUT : NFUNC=
1
 EULER       : METHOD=1
 HEUN        :       =2
 RUNGE-KUTTA :       =3
 INPUT : METHOD=
1
 INPUT : N=
2
 INPUT : T0=
0
 INPUT : X0(  1)=
1
 INPUT : X0(  2)=
2
 INPUT : H=
0.1
 INPUT : TMAX=
2
 T=  1.0000E-01
   I     X(I)      I     X(I)      I     X(I)      I     X(I)      I     X(I)
   1  1.0000E+00   2  1.9000E+00
 T=  2.0000E-01
   I     X(I)      I     X(I)      I     X(I)      I     X(I)      I     X(I)
   1  9.8000E-01   2  1.7900E+00
(中略)
 T=  1.9000E+00
   I     X(I)      I     X(I)      I     X(I)      I     X(I)      I     X(I)
   1  2.5576E-01   2  3.9084E-01
 T=  2.0000E+00
   I     X(I)      I     X(I)      I     X(I)      I     X(I)      I     X(I)
   1  2.3162E-01   2  3.5320E-01
%
```

これは $x_1(0) = 1$, $x_2(0) = 2$ の場合に、Euler 法で計算したものだが、167 ページの数表と見比べてどうだろうか？

8.2　p170 (プログラムリスト 2)

170 ページのプログラムで – ページの計算実習 2 をやってみよう。

```
% ./p170
 EULER       : METHOD=1
 HEUN        :       =2
 RUNGE-KUTTA :       =3
 INPUT : METHOD=
1
 INPUT : N=
2
 INPUT : A(  1,  1)=
```

```
0
 INPUT : A(  1,  2)=
1
 INPUT : A(  2,  1)=
-1
 INPUT : A(  2,  2)=
0
 INPUT : T0=
0
 INPUT : X0(  1)=
1
 INPUT : X0(  2)=
1
 INPUT : H=
0.05
 INPUT : TMAX=
20
 T=  5.0000E-02
   I     X(I)      I     X(I)      I     X(I)      I     X(I)      I     X(I)
   1  1.0500E+00   2  9.5000E-01
 T=  1.0000E-01
   I     X(I)      I     X(I)      I     X(I)      I     X(I)      I     X(I)
   1  1.0975E+00   2  8.9750E-01
 T=  1.5000E-01
   I     X(I)      I     X(I)      I     X(I)      I     X(I)      I     X(I)
   1  1.1424E+00   2  8.4262E-01
(中略)
 T=  1.9900E+01
   I     X(I)      I     X(I)      I     X(I)      I     X(I)      I     X(I)
   1  2.2530E+00   2 -5.7165E-01
 T=  1.9950E+01
   I     X(I)      I     X(I)      I     X(I)      I     X(I)      I     X(I)
   1  2.2244E+00   2 -6.8430E-01
 T=  2.0000E+01
   I     X(I)      I     X(I)      I     X(I)      I     X(I)      I     X(I)
   1  2.1902E+00   2 -7.9552E-01
%
```

これは Euler 法で解いたものだが、テキストには結果が数値で与えられていないので、グラフを描いて、172 ページの図と比べてみよう。

9 問題点

2018/4/9 8 章まで、ほとんどの問題点は解消した。

- p65.f の結果を可視化しよう。
- p170.f の結果を可視化しよう。

A ソースプログラム

chapter1/p13.c

```
/* 菊地文雄，山本昌宏「微分方程式と計算機演習」，山海堂（1991）
   P. 13 のプログラムを C 言語に書き直した。
  ** GAUSS ELIMINATION METHOD **
  n : NUMBER OF UNKNOWS
  a : COEFFICIENT MATRIX
  f : KNOWN VECTOR & SOLUTION VECTORS
*/

#include <stdio.h>
#include <stdlib.h> // exit()
#define NDIM (100)

void gauss(int ndim, double a[ndim][ndim], double f[], int N);

int main(void)
{
  int n, i, j;
  double a[NDIM][NDIM], f[NDIM];
  printf("INPUT : N=\n"); // 最後の \n は不要かも
  scanf("%d", &n);
  if (n<=1)
    exit(0);
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++) {
      printf("INPUT : A(%3d,%3d)=\n", i+1, j+1); // 最後の \n は不要かも
      scanf("%lf", &a[i][j]);
    }
  for (i = 0; i < n; i++) {
    printf(" INPUT : F(%3d)=\n", i+1); // 最後の \n は不要かも
    scanf("%lf", &f[i]);
  }
  gauss(NDIM,a,f,n);
  for (i = 0; i < 5; i++)
    printf("  I     X(I)    ");
  printf("\n");
  for (i = 0; i < n; i++) {
    printf("%3d%12.4e ",i+1, f[i]); //I3 は %3d, PE12.4 は %12.4e (1P は無視)
    if ((i+1) % 5 == 0 || i+1 == n)
      printf("\n");
  }
}

void gauss(int ndim, double a[ndim][ndim], double f[], int n)
{
  int i, j, k;
  double aa;
  // FORWARD ELIMITNATION
  for (i=0; i<n-1; i++)
    for (j=i+1; j<n; j++) {
```

```
50        aa = a[j][i] / a[i][i];
51        for (k=i+1; k<n; k++)
52      a[j][k] -= aa * a[i][k];
53        f[j] -= aa * f[i];
54      }
55    // BACKWARD SUBSTITUTION
56    f[n-1] /= a[n-1][n-1];
57    for (i=n-2;i>=0;i--) {
58      for (j=i+1; j<n; j++)
59        f[i] -= a[i][j] * f[j];
60      f[i] /= a[i][i];
61    }
62  }
```

chapter1/p20.c

```
1  // 菊地文雄, 山本昌宏「微分方程式と計算機演習」, 山海堂 (1991), P. 20
2  /* ** GAUSS ELIMINATION METHOD **
3     **  FOR TRIDIAGONAL SYSTEM  **
4     N : NUMBER OF UNKNOWS
5     A : LOWER DIAGONAL
6     B : DIAGONAL
7     C : UPPER DIAGONAL
8     F : KNOWN VECTOR & SOLUTION VECTOR
9  */
10
11 #include <stdio.h>
12 #include <stdlib.h>
13 #define NDIM (100)
14
15 void trid(double a[], double b[], double c[], double f[], int n);
16
17 int main(void)
18 {
19   int n, i;
20   double a[NDIM], b[NDIM], c[NDIM], f[NDIM];
21   printf("INPUT : N=\n");
22   scanf("%d", &n);
23   if (n <= 1)
24     exit(1);
25   printf("INPUT : B(I),C(I) FOR I=1\n");
26   scanf("%lf%lf", &b[0], &c[0]);
27   for (i = 1; i < n-1; i++) {
28     printf("INPUT : A(I),B(I),C(I) FOR I=%3d\n", i+1);
29     scanf("%lf%lf%lf", &a[i],&b[i],&c[i]);
30   }
31   printf("INPUT : A(I),B(I) FOR I=N\n");
32   scanf("%lf%lf", &a[n-1], &b[n-1]);
33   for (i=0; i < n; i++) {
34     printf("INPUT : F(%3d)=\n", i+1);
35     scanf("%lf", &f[i]);
36   }
37   trid(a,b,c,f,n);
38   for (i = 0; i < 5; i++)
39     printf("  I     X(I)    ");
40   printf("\n");
41   for (i=0; i<n; i++) {
42     printf("%3d%12.4e ", i+1, f[i]);
43     if ((i+1) % 5 == 0 || i+1 == n)
44       printf("\n");
45   }
46 }
47
48 void trid(double a[], double b[], double c[], double f[], int n)
```

```
49  {
50    int i;
51    double aa;
52    // FORWARD ELIMITNATION
53    for (i=0; i<n-1; i++) {
54      aa = a[i+1] / b[i];
55      b[i+1] -= aa * c[i];
56      f[i+1] -= aa * f[i];
57    }
58    // BACKWARD SUBSTITUTION
59    f[n-1] /= b[n-1];
60    for (i=n-2; i>=0; i--)
61      f[i] = (f[i] - c[i] * f[i+1]) / b[i];
62  }
```

chapter1/p29.c

```
 1  // 菊地文雄, 山本昌宏「微分方程式と計算機演習」, 山海堂 (1991), P. 29
 2  /* ** GAUSS ELIMINATION METHOD FOR **
 3     ** SYMMETRIC TRIDIAGONAL SYSTEM  **
 4     N : NUMBER OF UNKNOWS
 5     A : LOWER DIAGONAL
 6     B : DIAGONAL
 7     F : KNOWN VECTOR & SOLUTION VECTOR
 8  */
 9
10  #include <stdio.h>
11  #include <stdlib.h> // exit()
12  void trisym(double a[], double b[], double f[], int n);
13
14  #define NDIM (100)
15
16  int main(void)
17  {
18    int n, i;
19    double a[NDIM], b[NDIM], c[NDIM], f[NDIM];
20    printf("NPUT : N=\n");
21    scanf("%d", &n);
22    if (n <= 1) exit(1);
23    printf("INPUT : B(I) FOR I=1\n");
24    scanf("%lf", &b[0]);
25    for (i=1; i < n; i++) {
26      printf("INPUT : A(I),B(I) FOR I=%3d\n", i+1);
27      scanf("%lf%lf", &a[i], &b[i]);
28    }
29    for (i=0; i<n; i++) {
30      printf("INPUT: F(%3d)=\n", i+1);
31      scanf("%lf", &f[i]);
32    }
33    trisym(a,b,f,n);
34    for (i = 0; i < 5; i++)
35      printf("  I     X(I)    ");
36    printf("\n");
37    for (i=0; i<n; i++) {
38      printf("%3d%12.4e ", i+1, f[i]);
39      if ((i+1) % 5 == 0 || i+1 == n)
40        printf("\n");
41    }
42    return 0;
43  }
44
45  void trisym(double a[], double b[], double f[], int n)
46  {
47    int i;
```

```
48    double aa;
49    // FORWARD ELIMITNATION
50    for (i=0; i<n-1; i++) {
51      aa = a[i+1] / b[i];
52      b[i+1] -= aa * a[i+1];
53      f[i+1] -= aa * f[i];
54    }
55    // BACKWARD SUBSTITUTION
56    f[n-1] /= b[n-1];
57    for (i=n-2; i>=0; i--)
58      f[i] = (f[i]-a[i+1]*f[i+1])/b[i];
59  }
60
```

chapter2/p39.c

```
 1  // 菊地文雄, 山本昌宏「微分方程式と計算機演習」, 山海堂 (1991), p. 39
 2  /* ** JACOBI OR SOR METHOD **
 3     N : NUMBER OF UNKNOWS
 4     A : LOWER DIAGONAL
 5     F : KNOWN VECTOR & SOLUTION VECTOR
 6     X : SOLUTION VECTOR
 7     N : MAXIMUM NO. OF ITERATIONS
 8     EPS : THRESHOLD VALUE FOR ERROR
 9     METHOD : METHOD NUMBER
10     OMEGA : RELAXATION PARAMETER
11  */
12
13  #include <stdio.h>
14  #include <stdlib.h> // exit()
15  #include <math.h>   // fabs()
16  #define NDIM (100)
17
18  void iter(int ndim,
19        double a[][ndim], double f[], double x[], double xnew[],
20        double eps, double omega, int n, int m, int method, int *niter);
21
22  int main(void)
23  {
24    int n, m, method, i,j,niter;
25    double eps, omega;
26    double a[NDIM][NDIM], f[NDIM], x[NDIM], xnew[NDIM];
27    printf("INPUT : N=\n");
28    scanf("%d", &n);
29    if (n <= 1) exit(1);
30    printf("INPUT : M=\n");
31    scanf("%d", &m);
32    printf("INPUT : EPS=\n");
33    scanf("%lf", &eps);
34    printf("JACOBI : METHOD=0, SOR : =1\n");
35    printf("INPUT : METHOD=\n");
36    scanf("%d", &method);
37    omega=1.0;
38    if (method == 1) {
39      printf("INPUT : OMEGA=\n");
40      scanf("%lf", &omega);
41    }
42    for (i=0; i<n; i++)
43      for (j=0; j<n; j++) {
44        printf("INPUT : A(%3d,%3d)=\n", i+1, j+1);
45        scanf("%lf", &a[i][j]);
46      }
47    for (i=0; i<n; i++) {
48      printf("INPUT : F(%3d)=\n", i+1);
```

```
49      scanf("%lf", &f[i]);
50    }
51    for (i=0; i<n; i++) {
52      printf("INPUT : X(%3d)=\n", i+1);
53      scanf("%lf", &x[i]);
54    }
55    // 試しに表示する
56    printf("n=%d, m=%d, eps=%f, method=%d\n", n, m, eps, method);
57    for (i=0; i<n; i++) {
58      for (j=0; j<n; j++)
59        printf("%f ", a[i][j]);
60      printf("   %f %f", f[i], x[i]);
61      printf("\n");
62    }
63    iter(NDIM,a,f,x,xnew,eps,omega,n,m,method,&niter);
64    printf("NO. OF ITERATIONS=%3d   EPS=%12.4e\n", niter, eps);
65    for (i = 0; i < 5; i++)
66      printf("  I     X(I)    ");
67    printf("\n");
68    for (i=0; i<n; i++) {
69      printf("%3d%12.4e ", i+1, x[i]);
70      if ((i+1) % 5 == 0 || (i+1) == n)
71        printf("\n");
72    }
73  }
74
75  void iter(int ndim,
76        double a[][ndim], double f[], double x[], double xnew[],
77        double eps, double omega, int n, int m, int method, int *niter)
78  {
79    int i, j;
80    double error, xmax, xtemp, dx;
81    *niter = 0;
82    do {
83      error = 0.0;
84      xmax = 0.0;
85      for (i=0; i<n; i++) {
86        xtemp = f[i];
87        for (j=0; j<n; j++)
88      if (j != i) xtemp -= a[i][j] * x[j];
89        xtemp /= a[i][i];
90        if (method == 1)
91      xtemp = omega * xtemp + (1.0 - omega) * x[i];
92        dx = fabs(xtemp - x[i]);
93        if (dx > error) error = dx;
94        if (fabs(xtemp) >  xmax) xmax = fabs(xtemp);
95        if (method == 0) xnew[i] = xtemp;
96        if (method == 1) x[i] = xtemp;
97      }
98      if (method == 0)
99        for (i=0; i<n; i++) x[i] = xnew[i];
100     (*niter)++;
101     error /= xmax;
102     printf("(NITER,ERROR)=%d %g\n", *niter, error);
103     if (*niter == m)
104       return;
105   } while (error > eps);
106 }
```

chapter3/p58.c

```
1  // 菊地文雄, 山本昌宏「微分方程式と計算機演習」, 山海堂 (1991), p. 58
2  /* ** FINITE DIFFERENCE METHOD **
3     **      FOR -D2U/Dx2=F      **
```

```
 4     N : NUMBER OF DIVISIONS
 5     H : MESH SIZE
 6     A,B,C : COEFFICIENT MATRIX
 7     F : FREE TERM & SOLUTION VECTOR
 8  */
 9
10  #include <stdio.h>
11  #include <stdlib.h>
12  #define NDIM (100)
13
14  void trid(double a[], double b[], double c[], double f[], int n);
15
16  int main(void)
17  {
18    int n, i;
19    double h, h2;
20    double a[NDIM], b[NDIM], c[NDIM], f[NDIM], x[NDIM];
21    printf("INPUT : N=\n");
22    scanf("%d", &n);
23    if (n <= 1) exit(0);
24    h = 1.0 / n;
25    h2 = h * h;
26    for (i=0; i<n; i++) {
27      a[i] = -1.0 / h2;
28      b[i] = 2.0 / h2;
29      c[i] = -1.0 / h2;
30      f[i] = 1.0;
31    }
32    b[n-1] = 1.0 / h2;
33    f[n-1] = 0;
34    trid(a,b,c,f,n);
35    printf("MESH SIZE=%12.4e\n", h);
36    printf("** NODAL VALUES OF SOLUTION **\n");
37    for (i = 0; i < 5; i++)
38      printf("  I     U(I)    ");
39    printf("\n");
40    for (i=0; i<n; i++) {
41      printf("%3d%12.4e ", i+1, f[i]);
42      if ((i+1) % 5 == 0 || (i+1) == n)
43      printf("\n");
44    }
45  }
46
47  void trid(double a[], double b[], double c[], double f[], int n)
48  {
49    int i;
50    double aa;
51    // FORWARD ELIMITNATION
52    for (i = 0; i < n-1; i++) {
53      aa = a[i+1] / b[i];
54      b[i+1] -= aa * c[i];
55      f[i+1] -= aa * f[i];
56    }
57    // BACKWARD SUBSTITUTION
58    f[n-1] /= b[n-1];
59    for (i=n-2; i>= 0; i--)
60      f[i] = (f[i] - c[i] * f[i+1]) / b[i];
61  }
```

chapter3/p65.c

```
 1  // 菊地文雄, 山本昌宏「微分方程式と計算機演習」, 山海堂 (1991), p. 65
 2  /* ** FINITE DIFFERENCE METHOD **
 3     ** FOR -FNU D2U/Dx2+DU/DX=F **
```

```
 4     FNU : DIFFUSIVITY
 5     N : NO. OF DIVISIONS
 6     METHOD : METHOD NUMBER
 7     H : MESH SIZE
 8     A,B,C : COEFFICIENT MATRIX
 9     F : FREE TERM & SOLUTION VECTOR
10  */
11
12  #include <stdio.h>
13  #include <stdlib.h>
14  #define NDIM (100)
15  void trid(double a[], double b[], double c[], double f[], int n);
16
17  int main(void)
18  {
19    int n, i, method;
20    double fnu, h, h2; // fnu は nu でも良いかも。
21    double a[NDIM], b[NDIM], c[NDIM], f[NDIM], x[NDIM];
22    printf("INPUT : FNU=\n");
23    scanf("%lf", &fnu);
24    printf("INPUT : N=\n");
25    scanf("%d", &n);
26    if (n <= 1) exit(1);
27    printf("BACKWARD : METHOD=1, CENTRAL : =2, FORWARD : =3\n");
28    printf("INPUT : METHOD=\n");
29    scanf("%d", &method);
30    h = 1.0 / n; // (double) n とする必要もないでしょう。
31    h2 = h * h;  // C に冪乗演算子はないので、乗算で代用
32    for (i=0; i<n-1; i++) {
33      a[i] = - fnu / h2;
34      b[i] = 2.0 * fnu / h2;
35      c[i] = - fnu / h2;
36      if (method == 1) {
37        // BACKWARD
38        a[i] -= 1.0 / h;
39        b[i] += 1.0 / h;
40      }
41      else if (method == 2) {
42        // CENTRAL
43        a[i] -= 0.5 / h;
44        c[i] += 0.5 / h;
45      }
46      else {
47        // FORWARD
48        b[i] -= 1.0 / h;
49        c[i] += 1.0 / h;
50      }
51      f[i] = 1.0;
52    }
53    trid(a,b,c,f,n-1);
54    printf("DIFFUSIVITY=%12.4e   MESH SIZE=%12.4e   METHOD=%2d\n",
55       fnu, h, method);
56    printf("** NODAL VALUES OF SOLUTION **\n");
57    for (i = 0; i < 5; i++)
58      printf("  I     U(I)    ");
59    printf("\n");
60    for (i=0; i<n-1; i++) {
61      printf("%3d%12.4e ", i+1, f[i]);
62      if ((i+1) % 5 == 0 || (i+1) == n-1)
63      printf("\n");
64    }
65  }
66
```

```
67  void trid(double a[], double b[], double c[], double f[], int n)
68  {
69    int i;
70    double aa;
71    // FORWARD ELIMITNATION
72    for (i=0; i<n-1; i++) {
73      aa = a[i+1] / b[i];
74      b[i+1] -= aa * c[i];
75      f[i+1] -= aa * f[i];
76    }
77    // BACKWARD SUBSTITUTION
78    f[n-1] /= b[n-1];
79    for (i=n-2; i>= 0; i--)
80      f[i] = (f[i] - c[i]*f[i+1]) / b[i];
81  }
```

chapter3/p73.c

```
 1  // 菊地文雄, 山本昌宏「微分方程式と計算機演習」, 山海堂 (1991), p. 73
 2  /*  ** FINITE DIFFERENCE METHOD **
 3      **      FOR -D2U/Dx2=F      **
 4      ** BY JACOBI OR SOR METHOD  **
 5      N : NO. OF DIVISIONS
 6      M : MAXIMUM NO. OF ITERATIONS
 7      EPS : THRESHOLD VALUE FOR ERROR
 8      METHOD : METHOD NUMBER
 9      OMEGA : RELAXATION PARAMETER
10      H : MESH SIZE
11      F : FREE TERM
12      U : SOLUTION VECTOR
13  */
14
15  #include <stdio.h>
16  #include <stdlib.h>  // exit()
17  #include <math.h>    // fabs()
18
19  #define NDIM (100)
20
21  void iter(double f[], double u[], double eps, double h2,
22        double omega, int N, int M, int METHOD,
23        int *niter);
24
25  int main(void)
26  {
27    int n, m, method, i, niter;
28    double eps, omega, h, h2;
29    double f[NDIM],u[NDIM];
30    printf("INPUT : N=\n");
31    scanf("%d", &n);
32    if (n <= 1) exit(1);
33    printf("INPUT : M=\n");
34    scanf("%d", &m);
35    printf("INPUT : EPS=\n");
36    scanf("%lf", &eps);
37    printf("JACOBI : METHOD=0, SOR : =1\n");
38    printf("INPUT : METHOD=\n");
39    scanf("%d", &method);
40    if (method == 1) {
41      printf("INPUT : OMEGA=\n");
42      scanf("%lf", &omega);
43    }
44    h = 1.0 / n;
45    h2 = h * h;
46    for (i = 0; i < n; i++) {
```

```
47      u[i] = 0.0;
48      f[i] = 1.0;
49    }
50    f[n-1] = 0.0;
51    iter(f, u, eps, h2, omega, n, m, method, &niter);
52    printf("NO. OF ITERATIONS=%4d   EPS=%12.4e\n", niter, eps);
53    printf("MESH SIZE=%12.4e\n", h);
54    printf("** NODAL VALUES OF SOLUTION **\n");
55    for (i = 0; i < 5; i++)
56      printf("  I     U(I)    ");
57    printf("\n");
58    for (i=0; i<n; i++) {
59      printf("%3d%12.4e ", i+1, u[i]);
60      if ((i+1) % 5 == 0 || (i+1) == n)
61        printf("\n");
62    }
63  }
64
65  void iter(double f[], double u[], double eps, double h2,
66        double omega, int n, int m, int method,
67        int *niter)
68  {
69    int i;
70    double error, umax, uleft, unew, du;
71    for (i = 0; i < n; i++)
72      printf("%f %f\n", u[i], f[i]);
73    *niter = 0;
74    do {
75      error = 0.0;
76      umax = 0.0;
77      uleft = 0.0;
78      for (i = 0; i < n; i++) {
79        if (i != n-1) unew = (h2 * f[i] + uleft + u[i+1]) / 2.0;
80        if (i == n-1) unew = uleft;
81        if (method == 1) unew = omega * unew + (1.0 - omega) * u[i];
82        du = fabs(unew - u[i]);
83        if (du > error) error = du;
84        if (fabs(unew) > umax) umax = fabs(unew);
85        if (method == 0) uleft = u[i];
86        if (method == 1) uleft = unew;
87        u[i] = unew;
88      }
89      (*niter)++;
90      error /= umax;
91      printf("(NITER,ERROR)=%d %g\n", *niter, error);
92      if (*niter == m) return;
93    }
94    while (error > eps);
95  }
```

chapter3/p75.c

```
1  // 菊地文雄, 山本昌宏「微分方程式と計算機演習」, 山海堂 (1991), p. 75
2  /* ** FINITE DIFFERENCE METHOD **
3     ** FOR -FNU*D2U/DX2+DU/DX=F **
4     FNU : DIFFUSIVITY
5     N : NO. OF DIVISIONS
6     METHOD : METHOD NUMBER
7     H : MESH SIZE
8     A,B,C : COEFFICIENT MATRIX
9     F : FREETERM & SOLUTION VECTOR
10 */
11
12 #include <stdio.h>
```

```
13  #include <stdlib.h> // exit()
14  #define NDIM (100)
15
16  void trid(double a[], double b[], double c[], double f[], int n);
17
18  int main(void)
19  {
20    int n, method, i;
21    double fnu, h, h2;
22    double a[NDIM], b[NDIM], c[NDIM], f[NDIM];
23    printf("INPUT : FNU=\n");
24    scanf("%lf", &fnu);
25    printf("INPUT : N=\n");
26    scanf("%d", &n);
27    if (n <= 1) exit(1);
28    printf("BACKWARD : METHOD=1,CENTRAL : =2,FORWARD : =3\n");
29    printf("INPUT : METHOD=\n");
30    scanf("%d", &method);
31    h = 1.0 / n;
32    h2 = h * h;
33    for (i = 0; i < n-1; i++) {
34      a[i] = - fnu / h2;
35      b[i] = 2.0 * fnu / h2;
36      c[i] = - fnu / h2;
37      if (method == 1) {
38        // BACKWARD
39        a[i] -= 1.0 / h;
40        b[i] += 1.0 / h;
41      }
42      else if (method == 2) {
43        // CENTRAL
44        a[i] -= 0.5 / h;
45        c[i] += 0.5 / h;
46      }
47      else {
48        // FORWARD
49        b[i] -= 1.0 / h;
50        c[i] += 1.0 / h;
51      }
52      f[i] = 1.0;
53    }
54    a[n-1] = - fnu / h2;
55    b[n-1] = fnu / h2;
56    f[n-1] = 0.0;
57    if (method == 2) f[n-1] = 0.5;
58    if (method == 3) f[n-1] = 1.0;
59    trid(a, b, c, f, n);
60    printf("DIFFUSIVITY=%12.4e   MESH SIZE=%12.4e   METHOD=%2d\n",
61       fnu, h, method);
62    printf("** NODAL VALUES OF SOLUTION **\n");
63    for (i = 0; i < 5; i++)
64      printf("  I     U(I)    ");
65    printf("\n");
66    for (i=0; i<n; i++) {
67      printf("%3d%12.4e ", i+1, f[i]);
68      if ((i+1) % 5 == 0 || (i+1) == n)
69        printf("\n");
70    }
71  }
72
73  void trid(double a[], double b[], double c[], double f[], int n)
74  {
75    int i;
```

```
76    double aa;
77    // FORWARD ELIMINATION
78    for (i=0; i < n-1; i++) {
79      aa = a[i+1] / b[i];
80      b[i+1] -= aa * c[i];
81      f[i+1] -= aa * f[i];
82    }
83    // BACKWARD SUBSTITUTION
84    f[n-1] /= b[n-1];
85    for (i = n-2; i >= 0; i--)
86      f[i] = (f[i]-c[i]*f[i+1]) / b[i];
87  }
```

chapter4/p88.c

```
 1  // 菊地文雄, 山本昌宏「微分方程式と計算機演習」, 山海堂 (1991), p. 88
 2  /* ** POWER METHOD **
 3     N : ORDER OF MATRIX
 4     A : COEFFICIENT MATRIX
 5     EPS : THRESHOLD VALUE FOR ERRORS
 6     M : MAXIMUM NO. OF ITERATIONS
 7     EVAL : EIGENVALUE
 8     X : EIGENVECTOR
 9     NITER : NO. OF ITERATIONS
10  */
11
12  #include <stdio.h>
13  #include <stdlib.h>
14  #include <math.h> // sqrt(),fabs()
15  #define NDIM (100)
16
17  void mpr(int ndim, double a[][ndim], double x[], int n, double y[]);
18  void inpr(double x[], double y[], int n, double *s);
19
20  int main(void)
21  {
22    int n, m, i, j, niter;
23    double eps, eval, s, enew, error;
24    double a[NDIM][NDIM], x[NDIM], y[NDIM];
25    // INPUT
26    printf("INPUT : N=\n");
27    scanf("%d", &n);
28    if (n <= 1) exit(1);
29    printf("INPUT : EPS\n");
30    scanf("%lf", &eps);
31    printf("INPUT : M=\n");
32    scanf("%d", &m);
33    for (i = 0; i < n; i++)
34      for (j = 0; j < n; j++) {
35        printf("INPUT : A(%3d,%3d)=\n", i+1, j+1);
36        scanf("%lf", &a[i][j]);
37      }
38    for (i = 0; i < n; i++) {
39      printf("INPUT : X(%3d)=\n", i);
40      scanf("%lf", &x[i]);
41    }
42    // ITERATION
43    niter = 0;
44    eval = 0.0;
45    mpr(NDIM,a,x,n,y);
46    do {
47      inpr(y,y,n,&s);
48      s=sqrt(s);
49      for (i=0; i<n; i++)
```

```
50        x[i] = y[i] / s;
51      mpr(NDIM,a,x,n,y);
52      inpr(x,y,n,&enew);
53      error = fabs((enew-eval)/enew);
54      eval = enew;
55      niter++;
56      printf("(NITER,EVAL,ERROR=%d %g %g\n", niter,eval,error);
57      if (niter == m) break;
58    }
59    while (error > eps);
60    // OUTPUT
61    printf("NO. OF ITERATIONS=%3d   EPS=%12.4e\n", niter, eps);
62    printf("EIGENVALUE=%12.4e\n", eval);
63    printf("** EIGENVECTOR **\n");
64    for (i = 0; i < 5; i++)
65      printf("  I     X(I)    ");
66    printf("\n");
67    for (i=0; i<n; i++) {
68      printf("%3d%12.4e ", i+1, x[i]);
69      if ((i+1) % 5 == 0 || (i+1) == n)
70        printf("\n");
71    }
72  }
73
74  void mpr(int ndim, double a[][ndim], double x[], int n, double y[])
75  {
76    int i, j;
77    // MATRIX MULTIPLICATION
78    for (i=0; i<n; i++) {
79      y[i] =0.0;
80      for (j=0; j<n; j++)
81        y[i] += a[i][j] * x[j];
82    }
83  }
84
85  void inpr(double x[], double y[], int n, double *s)
86  {
87    int i;
88    // INNER PRODUCT
89    *s = 0.0;
90    for (i = 0; i < n; i++)
91      *s += x[i] * y[i];
92  }
```

chapter5/p104.c

```
1   // 菊地文雄, 山本昌宏「微分方程式と計算機演習」, 山海堂 (1991), p. 104
2   /* ** SHIFT METHOD **
3      N : ORDER OF MATRIX
4      A : COEFFICIENT MATRIX
5      SHIFT : SHIFT PARAMETER
6      EPS : THRESHOLD VALUE FOR ERROR
7      M : MAXIMUM NO. OF ITERATIONS
8      EVAL : EIGENVALUE
9      X : EIGENVECTOR
10     NITER : NO. OF ITERATIONS
11     NPIVOT : NO. OF NEGATIVE PIVOTS
12  */
13
14  #include <stdio.h>
15  #include <stdlib.h>
16  #include <math.h>
17
18  void gauss(int ndim, double a[][ndim], double f[], int n, int id);
```

```
19  void inpr(double x[], double y[], int n, double *s);
20
21  #define NDIM (100)
22
23  int main(void)
24  {
25    int n, m, i, j, niter, npivot;
26    double shift, eps, eval, s, t, enew, error;
27    double a[NDIM][NDIM], x[NDIM], y[NDIM];
28    // INPUT
29    printf("INPUT : N=\n");
30    scanf("%d", &n);
31    if (n <= 1) exit(1);
32    printf("INPUT : SHIFT=\n");
33    scanf("%lf", &shift);
34    printf("INPUT : EPS=\n");
35    scanf("%lf", &eps);
36    printf("INPUT : M=\n");
37    scanf("%d", &m);
38    for (i=0; i<n; i++) {
39      for (j=0; j<n; j++) {
40        printf("INPUT : A(%3d,%3d)=\n", i+1, j+1);
41        scanf("%lf", &a[i][j]);
42        if (j == i) a[i][i] -= shift;
43      }
44    }
45    for (i=0; i<n; i++) {
46      printf("INPUT : X(%3d)=\n", i+1);
47      scanf("%lf", &x[i]);
48    }
49    // ITERATION
50    niter = 0;
51    eval=shift;
52    do {
53      for (i=0; i<n; i++)
54        y[i] = x[i];
55      gauss(NDIM,a,y,n,niter);
56      inpr(x,y,n,&s);
57      inpr(y,y,n,&t);
58      enew=s/t+shift;
59      t=sqrt(t);
60      for (i=0; i<n; i++)
61        x[i] = y[i] / t;
62      error=fabs((enew-eval)/enew);
63      eval=enew;
64      niter++;
65      printf("(NITER,EVAL,ERROR)=%d %g %g\n", niter, eval, error);
66      if (niter == m) break;
67    }
68    while (error > eps);
69    npivot=0;
70    for (i=0; i<n; i++)
71      if (a[i][i] < 0.0) npivot++;
72    // OUTPUT
73    printf("NO. OF ITERATIONS=%3d   EPS=%12.4e   SHIFT=%12.4e   NPIVOT=%3d\n",
74       niter, eps, shift, npivot);
75    printf("EIGENVALUE=%12.4e\n", eval);
76    printf("** EIGENVECTOR **\n");
77    for (i = 0; i < 5; i++)
78      printf("  I     X(I)    ");
79    printf("\n");
80    for (i=0; i<n; i++) {
81      printf("%3d%12.4e ", i+1, x[i]);
```

```
82      if ((i+1) % 5 == 0 || (i+1) == n)
83        printf("\n");
84    }
85  }
86
87  void gauss(int ndim, double a[][ndim], double f[], int n, int id)
88  {
89    int i, j, k;
90    double aa;
91    //FORWARD ELIMINATION
92    for (i=0; i<n-1; i++) {
93      for (j=i+1; j<n; j++) {
94        aa = a[j][i] / a[i][i];
95        if (id == 0) {
96      for (k=i+1; k<n; k++)
97        a[j][k] -= aa * a[i][k];
98        }
99        f[j] -= aa * f[i];
100     }
101   }
102   // BACKWARD SUBSTITUTION
103   f[n-1] /= a[n-1][n-1];
104   for (i=n-2; i>=0; i--) {
105     for (j=i+1; j<n; j++)
106       f[i] -= a[i][j] * f[j];
107     f[i] /= a[i][i];
108   }
109 }
110
111 void inpr(double x[], double y[], int n, double *s)
112 {
113   int i;
114   // INNER PRODUCT
115   *s = 0.0;
116   for (i=0; i<n; i++)
117     *s += x[i] * y[i];
118 }
```

chapter5/p98.c

```
1   // 菊地文雄, 山本昌宏「微分方程式と計算機演習」, 山海堂 (1991), p. 98
2   /* ** INVERSE ITERATION METHOD **
3      N : ORDER OF MATRIX
4      A : COEFFICIENT MATRIX
5      EPS : THRESHOLD VALUE FOR ERROR
6      M : MAXIMUM NO. OF ITERATIONS
7      EVAL : EIGENVALUE
8      X : EIGENVECTOR
9      NITER : NO. OF ITERATIONS
10  */
11
12  #include <stdio.h>
13  #include <stdlib.h>
14  #include <math.h>
15
16  void gauss(int ndim, double a[][ndim], double f[], int N, int ID);
17  void inpr(double x[], double y[], int n, double *s);
18
19  #define NDIM (100)
20
21  int main(void)
22  {
23    int n, m, i, j, niter;
24    double eps, eval, s, t, enew, error;
```

```
25    double a[NDIM][NDIM], x[NDIM], y[NDIM];
26    // INPUT
27    printf("INPUT : N= \n");
28    scanf("%d", &n);
29    if (n <= 1) exit(1);
30    printf("INPUT: EPS=\n");
31    scanf("%lf", &eps);
32    printf("INPUT : M= \n");
33    scanf("%d", &m);
34    for (i = 0; i < n; i++) {
35      for (j=0; j < n; j++) {
36        printf("INPUT : A(%3d,%3d)=\n", i+1, j+1);
37        scanf("%lf", &a[i][j]);
38      }
39    }
40    for (i=0; i<n; i++) {
41      printf("INPUT:X(%3d)=\n",i+1);
42      scanf("%lf", &x[i]);
43    }
44    // ITERATION
45    niter = 0;
46    eval = 0.0;
47    do {
48      for (i=0; i<n; i++)
49        y[i] = x[i];
50      gauss(NDIM,a,y,n,niter);
51      inpr(x,y,n,&s);
52      inpr(y,y,n,&t);
53      enew = s / t;
54      t = sqrt(t);
55      for (i=0; i<n; i++)
56        x[i] = y[i] / t;
57      error = fabs((enew-eval)/enew);
58      eval = enew;
59      niter++;
60      printf("(NITER,EVAL,ERROR)=%d %g %g\n", niter, eval, error);
61      if (niter == m) break;
62    }
63    while (error > eps);
64    // OUTPUT
65    printf("NO. OF ITERATIONS = %3d   EPS=%12.4e\n", niter, eps);
66    printf("EIGENVALUE=%12.4e\n", eval);
67    printf("**EIGENVECTOR**\n");
68    for (i = 0; i < 5; i++)
69      printf("  I     X(I)    ");
70    printf("\n");
71    for (i=0; i<n; i++) {
72      printf("%3d%12.4e ", i+1, x[i]);
73      if ((i+1) % 5 == 0 || (i+1) == n)
74        printf("\n");
75    }
76  }
77
78  void gauss(int ndim, double a[][ndim], double f[], int n, int id)
79  {
80    int i, j, k;
81    double aa;
82    // FORWARD ELIMINATION
83    for (i=0; i<n-1; i++) {
84      for (j=i+1; j<n; j++) {
85        aa = a[j][i] / a[i][i];
86        if (id == 0) {
87      for (k=i+1; k<n; k++)
```

```
88        a[j][k] -= aa * a[i][k];
89        }
90        f[j] -= aa * f[i];
91      }
92    }
93    // BACKWARD SUBSTITUTION
94    f[n-1] /= a[n-1][n-1];
95    for (i=n-2; i>= 0; i--) {
96      for (j=i+1; j<n; j++)
97        f[i] -= a[i][j] * f[j];
98      f[i] /= a[i][i];
99    }
100 }
101
102 void inpr(double x[], double y[], int n, double *s)
103 {
104   int i;
105   // INNER PRODUCT
106   *s = 0.0;
107   for (i=0; i<n; i++)
108     *s += x[i] * y[i];
109 }
```

chapter6/p125.c

```
1  // 菊地文雄, 山本昌宏「微分方程式と計算機演習」, 山海堂 (1991), p. 125
2  /* **        EIGEN ANALYSIS OF       **
3     ** ORDINARY DIFFERENTIAL EQUATION **
4     N : NO. OF DIVISIONS
5     SHIFT : SHIFT PARAMETER
6     EPS : THRESHOLD VALUE FOR ERROR
7     M : MAXIMUM NO. OF ITERATIONS
8     EVAL : APPROXIMATE EIGENVALUE
9     H : MESH SIZE
10    A,B,C : COEFFICIENT MATRIX
11    U : EIGENVECTOR
12    NITER : NO. OF ITERATIONS
13    NPIVOT : NO. OF NEGATIVE PIVOTS
14 */
15
16 #include <stdio.h>
17 #include <stdlib.h>
18 #include <math.h>
19
20 void shinv(double a[], double b[], double c[],
21        double x[], double y[],
22        double eps, double h, double shift,
23        int n, int m, double *eval, int *niter, int *npivot);
24 void trid(double a[], double b[], double c[], double f[], int n, int id);
25 void inpr(double x[], double y[], int n, double *s);
26 double rnd(int *i);
27
28 #define NDIM (100)
29
30 int main(void)
31 {
32   int n, m, i, niter, npivot;
33   double shift, eps, h, h2, eval;
34   double a[NDIM], b[NDIM], c[NDIM], u[NDIM], v[NDIM];
35   // INPUT
36   printf("INPUT : N=\n");
37   scanf("%d", &n);
38   if (n <= 1) exit(1);
39   printf("INPUT : SHIFT=\n");
```

```
  scanf("%lf", &shift);
  printf("INPUT : EPS=\n");
  scanf("%lf", &eps);
  printf("INPUT : M=\n");
  scanf("%d", &m);
  // CALCULATION OF COEFFICIENT MATRIX
  h = 1.0 / n;
  h2 = h * h;
  for (i=0; i<n-1; i++) {
    a[i] = -1.0 / h2;
    b[i] = 2.0 / h2 - shift;
    c[i] = -1.0 / h2;
  }
  // CALCULATION OF EIGENPAIR
  shinv(a,b,c,u,v,eps,h,shift,n-1,m,&eval,&niter,&npivot);
  // OUTPUT
  printf("NO. OF ITERATIONS=%3d   EPS=%12.4e   SHIFT=%12.4e   NPIVOT=%3d\n",
     niter, eps, shift, npivot);
  printf("EIGENVALUE=%12.4e\n", eval);
  printf("** NODAL VALUES OF EIGENFUNCTION **\n");
  for (i = 0; i < 5; i++)
    printf("  I     U(I)    ");
  printf("\n");
  for (i=0; i<n-1; i++) {
    printf("%3d%12.4e ", i+1, u[i]);
    if ((i+1) % 5 == 0 || (i+1) == n-1)
      printf("\n");
  }
}

void shinv(double a[], double b[], double c[],
       double x[], double y[],
       double eps, double h, double shift,
       int n, int m, double *eval, int *niter, int *npivot)
{
  int j, i;
  double s, t, enew, error;
  // STARTING QUANTITIES
  j=1;
  for (i=0; i<n; i++)
    x[i] = rnd(&j);
  *niter=0;
  *eval=shift;
  // ITERATION
  do {
    for (i=0; i<n; i++)
      y[i] = x[i];
    trid(a,b,c,y,n,*niter);
    inpr(x,y,n,&s);
    inpr(y,y,n,&t);
    enew=s/t+shift;
    t=sqrt(h*t);
    for (i=0; i<n; i++)
      x[i] = y[i] / t;
    error=fabs((enew-*eval)/enew);
    *eval=enew;
    (*niter)++;
    printf("(NITER,EVAL,ERROR)=%d %g %g\n", *niter, *eval, error);
    if (*niter == m) break;
  }
  while (error > eps);
  *npivot=0;
  for (i=0; i<n; i++)
```

```
103     if (b[i] < 0.0) (*npivot)++;
104 }
105
106 double rnd(int *i)
107 {
108   // PSEUDO-RANDOM NUMBERS
109   *i = 12869 * (*i) + 6925;
110   *i = *i % (1<<15);
111   return  (double) *i / (1 << 15);
112 }
113
114 void trid(double a[], double b[], double c[], double f[], int n, int id)
115 {
116   int i;
117   double aa;
118   // FORWARD ELIMINATION
119   for (i=0; i<n-1; i++) {
120     aa = a[i+1] / b[i];
121     if (id == 0) b[i+1] -= aa * c[i];
122     f[i+1] -= aa * f[i];
123   }
124   // BACKWARD SUBSTITUTION
125   f[n-1] /= b[n-1];
126   for (i=n-2; i>=0; i--)
127     f[i] = (f[i] - c[i]*f[i+1]) / b[i];
128 }
129
130 void inpr(double x[], double y[], int n, double *s)
131 {
132   int i;
133   // INNER PRODUCT
134   *s=0.0;
135   for (i=0; i<n; i++)
136     *s += x[i] * y[i];
137 }
```

chapter6/p132.c

```
 1 // 菊地文雄, 山本昌宏「微分方程式と計算機演習」, 山海堂 (1991), p. 132
 2 /* **        EIGEN ANALYSIS OF       **
 3    ** ORDINARY DIFFERENTIAL EQUATION **
 4    N : NO. OF DIVISIONS
 5    SHIFT : SHIFT PARAMETER
 6    EPS : THRESHOLD VALUE FOR ERROR
 7    M : MAXIMUM NO. OF ITERATIONS
 8    EVAL : APPROXIMATE EIGENVALUE
 9    H : MESH SIZE
10    A,B,C : COEFFICIENT MATRIX
11    U : EIGENVECTOR
12    NITER : NO. OF ITERATIONS
13    NPIVOT : NO. OF NEGATIVE PIVOTS
14 */
15
16 #include <stdio.h>
17 #include <stdlib.h>
18 #include <math.h>
19
20 void shinv(double a[], double b[], double c[],
21            double x[], double y[],
22            double eps, double h, double shift,
23            int n, int m, double *eval, int *niter, int *npivot);
24 double rnd(int *i);
25 void trid(double a[], double b[], double c[], double f[], int n, int id);
26 void inpr(double x[], double y[], int n, double *s);
```

```
27
28  #define NDIM (100)
29
30  int main(void)
31  {
32    int n, m, i, niter, npivot;
33    double  shift, eps, h, h2, eval;
34    double a[NDIM], b[NDIM], c[NDIM], u[NDIM], v[NDIM];
35    // INPUT
36    printf("INPUT : N=\n");
37    scanf("%d", &n);
38    if (n <= 1) exit(1);
39    printf("INPUT : SHIFT=\n");
40    scanf("%lf", &shift);
41    printf("INPUT : EPS=\n");
42    scanf("%lf", &eps);
43    printf("INPUT : M=\n");
44    scanf("%d", &m);
45    // CALCULATION OF COEFFICIENT MATRIX
46    h = 1.0 / n;
47    h2 = h * h;
48    for (i=0; i<n-1; i++) {
49      a[i] = -1.0 / h2;
50      b[i] = 2.0 / h2 - shift;
51      c[i] = -1.0 / h2;
52    }
53    a[n-1] = - 2.0 / h2;
54    b[n-1] = 2.0 / h2 - shift;
55    // CALCULATION OF EIGENPAIR
56    shinv(a,b,c,u,v,eps,h,shift,n,m,&eval,&niter,&npivot);
57    // OUTPUT
58    printf("NO. OF ITERATIONS=%3d   EPS=%12.4e   SHIFT=%12.4e   NPIVOT=%3d\n",
59           niter, eps, shift, npivot);
60    printf("EIGENVALUE=%12.4e\n", eval);
61    printf("** NODAL VALUES OF EIGENFUNCTION **\n");
62    for (i = 0; i < 5; i++)
63      printf("  I     U(I)    ");
64    printf("\n");
65    for (i=0; i<n; i++) {
66      printf("%3d%12.4e ", i+1, u[i]);
67      if ((i+1) % 5 == 0 || (i+1) == n)
68        printf("\n");
69    }
70  }
71
72  void shinv(double a[], double b[], double c[],
73             double x[], double y[],
74             double eps, double h, double shift,
75             int n, int m, double *eval, int *niter, int *npivot)
76  {
77    int j, i;
78    double s, t, enew, error;
79    // STARTING QUANTITIES
80    j=1;
81    for (i=0; i<n; i++)
82      x[i] = rnd(&j);
83    *niter=0;
84    *eval=shift;
85    // ITERATION
86    do {
87      for (i=0; i<n; i++)
88        y[i] = x[i];
89      trid(a,b,c,y,n,*niter);
```

```
90      inpr(x,y,n,&s);
91      inpr(y,y,n,&t);
92      enew=s/t+shift;
93      t=sqrt(h*t);
94      for (i=0; i<n; i++)
95        x[i] = y[i] / t;
96      error=fabs((enew-*eval)/enew);
97      *eval=enew;
98      (*niter)++;
99      printf("(NITER,EVAL,ERROR)=%d %g %g\n", *niter, *eval, error);
100     if (*niter == m) break;
101   }
102   while (error > eps);
103   *npivot=0;
104   for (i=0; i<n; i++)
105     if (b[i] < 0.0) (*npivot)++;
106 }
107
108
109 double rnd(int *i)
110 {
111   // PSEUDO-RANDOM NUMBERS
112   *i = 12869 * (*i) + 6925;
113   *i = *i % (1<<15);
114   return  (double) *i / (1 << 15);
115 }
116
117 void trid(double a[], double b[], double c[], double f[], int n, int id)
118 {
119   int i;
120   double aa;
121   // FORWARD ELIMINATION
122   for (i=0; i<n-1; i++) {
123     aa = a[i+1] / b[i];
124     if (id == 0) b[i+1] -= aa * c[i];
125     f[i+1] -= aa * f[i];
126   }
127   // BACKWARD SUBSTITUTION
128   f[n-1] /= b[n-1];
129   for (i=n-2; i>=0; i--)
130     f[i] = (f[i] - c[i]*f[i+1]) / b[i];
131 }
132
133 void inpr(double x[], double y[], int n, double *s)
134 {
135   int i;
136   // INNER PRODUCT
137   *s=0.0;
138   for (i=0; i<n-1; i++)
139     *s += x[i] * y[i];
140   *s += x[n-1] * y[n-1] / 2.0;
141 }
```

chapter7/p146.c

```
1  // C  菊地文雄, 山本昌宏「微分方程式と計算機演習」, 山海堂 (1991), p. 146
2  /* ** EULER,HEUN,AND RUNGE-KUTTA METHODS **
3     ** FOR ORDINARY DIFFERENTIAL EQUATIONS **
4     NFUNC : FUNCTION NUMBER
5     METHOD : METHOD NUMBER
6     T : TIME VARIABLE
7     X : APPROXIMATE VALUE OF SOLUTION
8     (T0,X0) : INITIAL VALUE OF (T,X)
9     H : TIME INCREMENT
```

```
10     TMAX : UPPER LIMIT OF TIME
11  */
12
13  #include <stdio.h>
14  #include <stdlib.h>
15  #include <math.h>
16
17  double f(double t, double x, int nfunc);
18
19  int main(void)
20  {
21    int nfunc, method, nstep;
22    double t0, x0, h, Tmax, t, x, fk1, fk2, fk3, fk4;
23    printf("INPUT : NFUNC=\n");
24    scanf("%d", &nfunc);
25    printf("EULER : METHOD=1\n");
26    printf("HEUN  :       =2\n");
27    printf("RUNGE-KUTTA   =3\n");
28    printf("INPUT : METHOD=");
29    scanf("%d", &method);
30    printf("INPUT : T0=\n");
31    scanf("%lf", &t0);
32    printf("INPUT : x0=\n");
33    scanf("%lf", &x0);
34    printf("INPUT : h=\n");
35    scanf("%lf", &h);
36    printf("INPUT : Tmax=\n");
37    scanf("%lf", &Tmax);
38    t=t0;
39    x=x0;
40    nstep=0;
41    do {
42      if (method == 1) {
43        x += h*f(t,x,nfunc);
44      }
45      else if (method == 2) {
46        fk1=f(t,x,nfunc);
47        fk2=f(t+h,x+h*fk1,nfunc);
48        x += h*(fk1+fk2)/2.0;
49      }
50      else {
51        fk1=f(t,x,nfunc);
52        fk2=f(t+h/2.0,x+h*fk1/2.0,nfunc);
53        fk3=f(t+h/2.0,x+h*fk2/2.0,nfunc);
54        fk4=f(t+h,x+h*fk3,nfunc);
55        x += h*(fk1+2.0*fk2+2.0*fk3+fk4)/6.0;
56      }
57      nstep++;
58      t=t0+h*nstep;
59      printf("T=%12.4e X=%12.4e\n", t, x);
60    }
61    while ((h > 0.0 && t+h < Tmax+0.1*h) || (h < 0.0 && t+h > Tmax+0.1*h));
62    return 0;
63  }
64
65  double f(double t, double x, int nfunc)
66  {
67    if (nfunc == 1) return x;
68    if (nfunc == 2) return 1.0 - x * x;
69    if (nfunc == 3) return x * x;
70    return 0.0;
71  }
```

chapter8/p164.c

```
 1  // 菊地文雄, 山本昌宏「微分方程式と計算機演習」, 山海堂 (1991), p. 164
 2  /* **  EULER, HEUN, AND RUNGE-KUTTA  **
 3     ** METHODS FOR SYSTEM OF ORDINARY **
 4     **    DIFFERENTIAL EQUATIONS-1    **
 5     NFUNC : FUNCTION NUMBER
 6     METHOD : METHOD NUMBER
 7     N : NUMBER OF UNKNOWN FUNCTIONS
 8     T : TIME VARIABLE
 9     X : APPROXIMATE VALUE OF SOLUTION
10     (T0,X0) : NITIAL VALUE OF (T,X)
11     H : TIME INCREMENT
12     TMAX : UPPER LIMIT OF TIME
13  */
14
15  #include <stdio.h>
16  #include <stdlib.h>
17
18  double func(double t, double x[], double f[], int nfunc);
19
20  int main(void)
21  {
22    int nfunc, method, n, i, nstep;
23    double t0, h, Tmax, t;
24    double x[10], u[10];
25    double fk1[10],fk2[10],fk3[10],fk4[10];
26    printf("LINEAR SYSTEM        : NFUNC=1\n");
27    printf("VAN DER POL EQUATION :      =2\n");
28    printf("EXERCISE-2           :      =3\n");
29    printf("INPUT : NFUNC=\n");
30    scanf("%d", &nfunc);
31    printf("EULER       : METHOD=1\n");
32    printf("HEUN        :       =2\n");
33    printf("RUNGE-KUTTA :       =3\n");
34    printf("INPUT : METHOD=\n");
35    scanf("%d", &method);
36    printf(" 'INPUT : N=\n");
37    scanf("%d", &n);
38    printf(" 'INPUT : T0=\n");
39    scanf("%lf", &t0);
40    for (i=0; i<n; i++) {
41      printf("INPUT : X0(%3d)=\n", i+1);
42      scanf("%lf", &x[i]);
43    }
44    printf(" 'INPUT : H=\n");
45    scanf("%lf", &h);
46    printf(" 'INPUT : TMAX=\n");
47    scanf("%lf", &Tmax);
48    // CALCULATION
49    t=t0;
50    nstep=0;
51    do {
52      if (method == 1) {
53        func(t,x,fk1,nfunc);
54        for (i=0; i<n; i++)
55      x[i] += h * fk1[i];
56      }
57      else if (method == 2) {
58        func(t,x,fk1,nfunc);
59        for (i=0; i<n; i++)
60      u[i] = x[i] + h * fk1[i];
61        func(t+h,u,fk2,nfunc);
62        for (i=0; i<n; i++)
```

```
63      x[i] += h * (fk1[i] + fk2[i]) / 2.0;
64      }
65      else {
66        func(t,x,fk1,nfunc);
67        for (i=0; i<n; i++)
68      u[i] = x[i] + h * fk1[i] / 2.0;
69        func(t+h/2.0,u,fk2,nfunc);
70        for (i=0; i<n; i++)
71      u[i] = x[i] + h * fk2[i] / 2.0;
72        func(t+h/2.0,u,fk3,nfunc);
73        for (i=0; i<n; i++)
74      u[i] = x[i] + h * fk3[i];
75        func(t+h,u,fk4,nfunc);
76        for (i=0; i<n; i++)
77      x[i] += h * (fk1[i] + 2.0 * fk2[i] + 2.0 * fk3[i] + fk4[i]) / 6.0;
78      }
79      nstep++;
80      t = t0 + h * nstep;
81      // OUTPUT
82      printf("T=%12.4e\n", t);
83      for (i = 0; i < 5; i++)
84        printf("  I     X(I)    ");
85      printf("\n");
86      for (i=0; i<n; i++) {
87        printf("%3d%12.4e ", i+1, x[i]);
88        if ((i+1) % 5 == 0 || (i+1) == n)
89      printf("\n");
90      }
91    }
92    while ((h > 0.0 && t+h < Tmax+0.1*h) || (h < 0.0 && t+h > Tmax+0.1*h));
93  }
94
95  double func(double t, double x[], double f[], int nfunc)
96  {
97    if (nfunc == 1) {
98      f[0] = -4.0 * x[0] + 2.0 * x[1];
99      f[1] = -3.0 * x[0] + x[1];
100   }
101   else if (nfunc == 2) {
102     f[0] = x[1];
103     f[1] = (1.0 - x[0] * x[0]) * x[1] - x[0];
104   }
105   else if (nfunc == 3) {
106     f[0] = x[1];
107     f[1] = -x[0] - 0.2 * x[1];
108     if (x[1] < 0.0) f[1] = f[1] - 1.0;
109     if (x[1] > 0.0) f[1] = f[1] + 1.0;
110   }
111   return 0.0; // ここに来ない
112 }
```

chapter8/p170.c

```
1  // 菊地文雄, 山本昌宏「微分方程式と計算機演習」, 山海堂 (1991), p. 170
2  /* **  EULER, HEUN, AND RUNGE-KUTTA  **
3     ** METHODS FOR SYSTEM OF ORDINARY **
4     **    DIFFERENTIAL EQUATIONS-2    **
5     NFUNC : FUNCTION NUMBER
6     METHOD : METHOD NUMBER
7     N : NUMBER OF UNKNOWN FUNCTIONS
8     A : OEFFICIENT MATRIX
9     T : TIME VARIABLE
10    X : APPROXIMATE VALUE OF SOLUTION
11    (T0,X0) : NITIAL VALUE OF (T,X)
```

```
12     H : TIME INCREMENT
13     TMAX : UPPER LIMIT OF TIME
14  */
15
16  #include <stdio.h>
17  #define NDIM (10)
18
19  void mpr(int ndim, double a[][ndim],double x[], int n, double y[]);
20
21  int main(void)
22  {
23    int method, n, i, j, nstep;
24    double t0, h, Tmax, t;
25    double a[NDIM][NDIM],x[NDIM],u[NDIM];
26    double fk1[NDIM],fk2[NDIM],fk3[NDIM],fk4[NDIM];
27    printf("EULER       : METHOD=1\n");
28    printf("HEUN        :       =2\n");
29    printf("RUNGE-KUTTA :       =3\n");
30    printf("INPUT : METHOD=\n");
31    scanf("%d", &method);
32    printf("INPUT : N=\n");
33    scanf("%d", &n);
34    for (i=0; i<n; i++) {
35      for (j=0; j<n; j++) {
36        printf("INPUT : A(%3d,%3d)=\n", i+1, j+1);
37        scanf("%lf", &a[i][j]);
38      }
39    }
40    printf("INPUT : T0=\n");
41    scanf("%lf", &t0);
42    for (i=0; i<n; i++) {
43      printf("INPUT : X0(%3d)=\n",i+1);
44      scanf("%lf", &x[i]);
45    }
46    printf("INPUT : H=\n");
47    scanf("%lf", &h);
48    printf("INPUT : TMAX=\n");
49    scanf("%lf", &Tmax);
50    // CALCULATION
51    t=t0;
52    nstep=0;
53    do {
54      if (method == 1) {
55        mpr(NDIM,a,x,n,fk1);
56        for (i=0; i<n; i++)
57      x[i]=x[i]+h*fk1[i];
58      }
59      else if (method == 2) {
60        mpr(NDIM,a,x,n,fk1);
61        for (i=0; i<n; i++)
62      u[i]=x[i]+h*fk1[i];
63        mpr(NDIM,a,u,n,fk2);
64        for (i=0; i<n; i++)
65      x[i]=x[i]+h*(fk1[i]+fk2[i])/2.0;
66      }
67      else {
68        mpr(NDIM,a,x,n,fk1);
69        for (i=0; i<n; i++)
70      u[i]=x[i]+h*fk1[i]/2.0;
71        mpr(NDIM,a,u,n,fk2);
72        for (i=0; i<n; i++)
73      u[i]=x[i]+h*fk2[i]/2.0;
74        mpr(NDIM,a,u,n,fk3);
```

```
75        for (i=0; i<n; i++)
76      u[i]=x[i]+h*fk3[i];
77        mpr(NDIM,a,u,n,fk4);
78        for (i=0; i<n; i++)
79      x[i]=x[i]+h*(fk1[i]+2.0*fk2[i]+2.0*fk3[i]+fk4[i])/6.0;
80      }
81      nstep++;
82      t=t0+h*nstep;
83      // OUTPUT
84      printf("T=%12.4e\n", t);
85      for (i = 0; i < 5; i++)
86        printf("  I     x[i]    ");
87      printf("\n");
88      for (i=0; i<n; i++) {
89        printf("%3d%12.4e ", i+1, x[i]);
90        if ((i+1) % 5 == 0 || (i+1) == n)
91      printf("\n");
92      }
93    }
94    while ((h > 0.0 && t+h < Tmax+0.1*h) || (h < 0.0 && t+h > Tmax+0.1*h));
95  }
96
97  void mpr(int ndim, double a[][ndim],double x[], int n, double y[])
98  {
99    int i, j;
100   // MATRIX MULTIPLICATION
101   for (i=0; i<n; i++) {
102     y[i]=0.0;
103     for (j=0; j<n; j++)
104       y[i] += a[i][j] * x[j];
105   }
106 }
```

B　C ユーザーのための FORTRAN 速習

B.1　おまけ: FORTRAN プログラムのコンパイル・実行

テキストに載っている FORTRAN プログラムを実行することも可能 (割と簡単) である。

(今回、C プログラムに書き換えるにあたり、実行結果をもとの FORTRAN プログラムのそれと比較して、チェックした。)

B.1.1　gfortran の導入

現象数理学科 Mac では、**MacPorts** をインストールしてあるので、MacPorts が使える状態になっている (はずである[1])。その場合、MacPorts を使って **GCC** (GNU compiler collection) をインストールするのがお勧めである (GCC には gcc (GNU C compiler) 以外に gfortran (GNU Fortran compiler) が含まれている)。

長い間 MacPorts を更新をしていない場合は、更新することを勧める。

[1] macOS をアップデートした場合は、使えない状態になっている可能性もある。思い当たる節があれば相談すること。

MacPorts の更新

ネットに接続した状況で、ターミナルで

```
sudo port selfupdate
sudo port upgrade outdated
```

とする。ひょっとすると結構長い時間がかかるかもしれない。

それがしてある状況では、例えば

GCC version 6 のインストール

ネットに接続した状況で、ターミナルで

```
sudo port install gcc6
```

とすると GCC version 6 がインストールされ、gcc-mp-6, gfortran-mp-6 というコマンドが使えるようになる。

毎度 gfortran-mp-6 のようなコマンドを打つのは面倒なので、alias 定義をすることを勧める。例えば

.profile 等に設定

```
alias gfortran='gfortran-mp-6'
```

と設定しておくと、単に gfortran と入力するだけで gfortran-mp-6 が実行される。ついでに (後述する) 最適化の指定 -O もつけるとよいかも (この辺は色々な工夫が考えられる)。

```
alias gfortran='gfortran-mp-6 -O'
```

B.1.2 gfortran の使い方

例えば 13 ページのプログラムは p13.f という名前をつけてある。

```
gfortran p13.f
```

とすると、a.out という実行形式が出来る。

```
gfortran -o p13 p13.f
```

とすると、p13 という実行形式が出来る。

最適化の指定 -O もつけると良いかもしれない。

```
gfortran -O -o p13 p13.f
```

B.2 菊地・山本 [1] のプログラムを C 言語に書き換える — どうやったか

- 変数や関数の名前が大文字であるのを小文字にする。— これは必要のないことではあるが、C のプログラムの習慣にはそったものにしたいよね。
- 古い FORTRAN では、いわゆる固定形式しか使えず、普通は各行の 7 ~ 72 桁目に命令を書く。そのためにプログラムで、1 つの文が複数行になってしまっている場合が多い。

```
      WRITE(*,*) 'BACKWARD : METHOD=1, ',
     1           'CENTRAL : =2, ',
     2           'FORWARD : =3'
```

(6 桁目にブランクでない文字 1,2 を置いている、継続行であることを意味する。)

は次のように 1 行にすることが出来る。

```
      printf("BACKWARD : METHOD=1, CENTRAL : =2, FORWARD : =3\n");
```

- FORTRAN の固定形式では、行の先頭の文字が **C** である行は、注釈行である。C 言語に直すには、// を使うのが簡単であろう。

```
C 菊地文雄，山本昌宏「微分方程式と計算機演習」，山海堂（1991），P．13
```

```
// 菊地文雄，山本昌宏「微分方程式と計算機演習」，山海堂（1991），P．13
```

- オリジナルの FORTRAN プログラムは、“IJKLMN ルール” による**暗黙の変数宣言**を使って書かれている。これは宣言せずに変数を使うことが出来る、ただし変数名の先頭の文字が I,J,K,L,M,N であれば INTEGER (整数型), そうでなければ REAL (単精度浮動小数点数型) となる、というものである。現在では、暗黙の変数宣言は推奨されない習慣とされていて、FORTRAN でも、プログラム中に `implicit none` と書き、暗黙の変数宣言をしない場合が多い。

  ともあれ、C 言語では、変数宣言は絶対にサボれないので、ちゃんと書く必要がある。IJKLMN ルールを使っているので、先頭の文字が I,J,K,L,M,N のいずれかの場合は int 型に、そうでない場合は double 型にした。(もとの FORTRAN プログラムは単精度 (REAL) なので、double でなく float にすべきかもしれないが、C 言語で float の利用はあまり推奨できないので double にした。)

```
(変数宣言なし)
```

を

```
int i,j,k;
double aa;
```

と書き換える。配列変数については、(成分の型は IJKLMN ルールで定まるにしても) 大きさを DIMENSION 文で

```
      DIMENSION A(NDIM,NDIM),F(NDIM)
```

のように指定してある。これは

```
int n, i, j;
double a[NDIM][NDIM], f[NDIM];
```

と書き換えれば良い。

- FORTRAN では、デフォールトで配列の添字が 1 から始まるが、C 言語では 0 からなので、そこを直した (画面表示は +1 して元のプログラムに)。

  例えば、B(1) は b[0] に, A(N) は a[n-1] に直す。また

```
      DO 2 I=1,N
        DO 1 J=1,N
          READ(*,*) A(I,J)
    1   CONTINUE
    2 CONTINUE
```

  は

```
      for (i=0; i<n; i++)
        for (j=0; j<n; j++)
          scanf("%lf"< &a[i][j];
```

  に直す。

- 出力の書式 (FORMAT 文) の書き換えがちょっと難しい。これは今回は無視して良い (自分で FORTRAN プログラムを書き換える必要が生じない限り学ぶのはサボって良い)。

  – I は %d にする。

  – F は %f, E は %e にする。浮動小数点数を * で表示するときは %g が良いかも。1PE12.4 というのが多いが %12.4e とした (P がどういう意味か説明は省略するが、1PE の場合は %e にして良い)。

  – X は空白を意味する。1X は “ ” (1 文字のブランク), 3X は “ ” (3 文字のブランク)

  – H 変換は、もともとはライン・プリンターの制御文字を埋め込むために出来たものであろう (と桂田は理解している)。それは画面出力では意味がないので、その書き換えは “ゆるく” やっている。

- ベクトル、行列の扱いについて。FORTRAN プログラムで

```
      PARAMETER (NDIM=100)
      DIMENSION A(NDIM,NDIM),F(NDIM)
```

  のような変数宣言をしている。今回は機械的に

```
#define NDIM (100

  double a[NDIM][NDIM], f[NDIM];
```

  のように書き換えておいた。これは C 言語のプログラムとしては、やや不自然になってしまうので、少し説明しておく。

  次元 N 以上の数 NDIM を取って、大きめの変数を用意しておく、ということである。古い FORTRAN では、動的な変数宣言は出来ず、プログラムをコンパイルする時点で、変数の大きさを決めておく必要があるので、このようなプログラムになっている (そうするより仕方がない)。最近のプログラミング言語では、動的な変数宣言が出来るのが普通なので、古めかしい書き方であると言える。

  例えば C 言語であれば、ベクトルを記憶する変数 f は

```
#include <stdlib.h> // malloc() のため

   int N;
   double *f;

   // N を決めてから
   f = malloc(sizeof(double) * N);
   if (f == NULL) { } // エラー処理
```

のように準備すれば良い。行列の場合は少し面倒だが可能である。

- サブルーチン (C 言語の void 型の関数に相当する) への配列変数の渡し方についても、注意が必要である。

```
      SUBROUTINE GAUSS(A,F,N,NDIM)
      DIMENSION A(NDIM,NDIM),F(NDIM)
```

意味は比較的容易に想像できるであろう。素朴に考えると

```
void gauss(double a[ndim][ndim],f[ndim],int n, int ndim)
```

のような書き換えが思い浮かぶかもしれない。しかし、これは文法エラーになる。

```
void gauss(int ndim,double a[ndim][ndim],double f[ndim],int n)
```

のように a や f が出て来る前に int ndim がないとダメである。配列の最初の次元の大きさは省略できるので

```
void gauss(int ndim,double a[][ndim],double f[],int n)
```

のように書ける。だから、例えば 1 次元配列の場合は

```
// n 次元ベクトル x,y の内積を計算する関数
double naiseki(double x[], double y[], int n)
```

のように [] の中に何も書かないで済ませられる。

細かい注意　実は古い C 言語では、

```
void gauss(double a[][NDIM],f[],int n)

(ただし NDIM は定数)
```

のように、[] 内の数は定数にしなければならなかった。上のプログラムでは ndim という引数を用いているが、それは “整合配列引数” という、特殊な機能を用いている。その機能は, C99 規格には入ったが、より新しい規格ではオプション扱いになった。現在普及している多くの C コンパイラーではコンパイル出来るが、将来はどうなるか分からない。

- DO 文は for 文にする。繰り返す範囲を文番号を用いて、DO 文番号 制御変数=出発値,... のように指定してあるのを読み取って、必要に応じて、かっこ { } で適当にブロックを

作る。例えば

```
      DO 1 K=I+1,N
1     A(J,K)=A(J,K)-AA*A(I,K)
```

は

```
      for (k=i+1;k<n;k++)
        a[j][k] -= aa * a[i][k];
```

とすれば良い (かっこはつけても良いが、なくても構わない)。

```
    DO 3 I=1,N-1
      DO 2 J=I+1,N
        AA=A(J,I)/A(I,I)
        DO 1 K=I+1,N
1       A(J,K)=A(J,K)-AA*A(I,K)
        F(J)=F(J)-AA*F(I)
2     CONTINUE
3   CONTINUE
```

は

```
    for (i=0;i<n-1;i++) {
      for (j=i+1;j<n;j++) {
        aa = a[j][i] / a[i][i];
        for (k=i+1;k<N;k++)
          a[j][k] -= aa * a[i][k];
        f[j] -= aa * f[i];
      }
    }
```

とすれば良い (一番外側のカッコは不要である)。

B.3 菊地・山本 [1] プログラム を Fotran 90 に書き換える

固定形式はやめて自由形式にする。

- ファイル名末尾を .f から.f90 に変える。
- 行頭の C を ! にする。
- 比較を ==, <, <=, >, >=, /= に書き換える。
- DO 文を文番号を使わずに DO, END DO を使うように書き換える。
- FORMAT も WRITE(某,FORMAT 文の文番号) でなく、WRITE(某, 書式)　を用いる。

参考文献

[1] 菊地文雄, 山本昌宏：微分方程式と計算機演習, 山海堂 (1991).