常微分方程式と連立方程式のまとめ (後期中間試験に向けて)

山本昌志*

2006 年 11 月 27 日

1　前期末試験の傾向と対策

前期末試験では，常微分方程式の数値解法と連立方程式について，問う．具体的には，次の数値計算法の計算原理とコンピュータープログラムの方法を理解しておくこと．

- 常微分方程式の数値計算法
  - 4 次のルンゲ・クッタ法
  - 高階の常微分方程式
- 連立 1 次方程式 (消去法)
  - ガウス・ジョルダン法
- 連立 1 次方程式 (反復法)
  - ヤコビ法
  - ガウス・ザイデル法
  - SOR 法

以降，学習すべき内容をまとめておくので，よく理解して試験に臨むこと．試験日時と注意事項は，次の通りである．

試験日	12 月 4 日 (月曜日) 9:55〜10:55(60 分)
場所	教室 (PC ルームではない)
注意事項	教科書やノート，プリント類は持ち込み不可

2　常微分方程式

数値計算により，近似解を求める微分方程式は，

$$\frac{dy}{dx} = f(x, y) \qquad \text{初期条件} \quad y(a) = b \tag{1}$$

である．これは問題として与えられ，この式に従う x と y の関係を計算する．

*国立秋田工業高等専門学校　電気工学科

2.1　4 次のルンゲ・クッタ法

前期末試験で出題したオイラー法や 2 次のルンゲ・クッタ法は，パラメーターを増やして誤差項の次数を上げていく方法である．この方法で最良と言われるのが 4 次のルンゲ・クッタ法である．パラメーターを増やして，5, 6, 7, $\cdots$ と誤差項を小さくすることは可能であるが，同じ計算量であれば 4 次のルンゲ・クッタの刻み幅を小さくするほうが精度が良いと言われている．

ということで，皆さんが常微分方程式を計算する必要が生じたときは，何はともあれ 4 次のルンゲ・クッタで計算すべきである．普通の科学に携わる人にとって，4 次のルンゲ・クッタは常微分方程式の最初で最後の解法である．

4 次のルンゲ・クッタの公式は，式 (2) に示す通りである．そして，これは図 1 のように表すことができる．

$$
\begin{cases}
k_1 = hf(x_n, y_n) \\
k_2 = hf(x_n + \dfrac{h}{2}, y_n + \dfrac{k_1}{2}) \\
k_3 = hf(x_n + \dfrac{h}{2}, y_n + \dfrac{k_2}{2}) \\
k_4 = hf(x_n + h, y_n + k_3) \\
x_{n+1} = x_n + h \\
y_{n+1} = y_n + \dfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{cases}
\tag{2}
$$

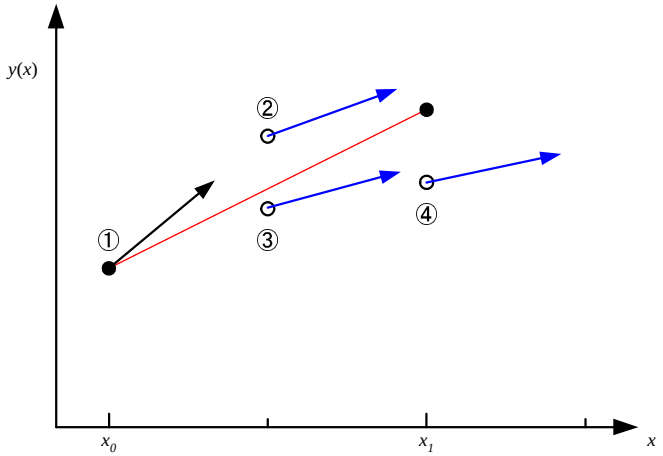

図 1: 4 次のルンゲ・クッタ法．ある区間での y の変化 Δy は，区間内の 4 点の傾きのある種の加重平均に幅 Δx を乗じて，求めている．

2.2 プログラム (4 次のルンゲ・クッタ法)

実際の微分方程式

$$\begin{cases} \dfrac{dy}{dx} = \sin x \cos x - y \cos x \\ y = 0 \quad (\text{初期条件 } x = 0 \text{ の時}) \end{cases} \tag{3}$$

を 4 次のルンゲ・クッタ法で計算するプログラムを示す．計算結果は，配列「x[]」と「y[]」に格納される．実際にプログラムでは，この結果を利用してグラフにしたりするのであるが，ここでは計算のみとする．

```
#include <stdio.h>
#include <math.h>
#define IMAX 100001
double func(double x, double y);

/*=================================================================*/
/*       main function                                             */
/*=================================================================*/
int main(void){
  double x[IMAX], y[IMAX];
  double final_x, h;
  double k1, k2, k3, k4;
  int ncal, i;


  /*--- set initial condition and cal range ---*/

  x[0]=0.0;
  y[0]=0.0;

  final_x=10.0;
  ncal=10000;

  /* --- size of calculation step --- */

  h=(final_x-x[0])/ncal;


  /* --- 4th Runge Kutta Calculation --- */

  for(i=0; i < ncal; i++){
    k1=h*func(x[i],y[i]);
    k2=h*func(x[i]+h/2.0, y[i]+k1/2.0);
    k3=h*func(x[i]+h/2.0, y[i]+k2/2.0);
    k4=h*func(x[i]+h, y[i]+k3);

    x[i+1]=x[i]+h;
    y[i+1]=y[i]+1.0/6.0*(k1+2.0*k2+2.0*k3+k4);
  }
```

```
  return 0;
}


/*=====================================================================*/
/*       define function                                               */
/*=====================================================================*/
double func(double x, double y){
  double dydx;

  dydx=sin(x)*cos(x)-y*cos(x);

  return(dydx);
}
```

2.3 高階の常微分方程式

2.3.1 1 階の連立微分方程式に変換

ここまで示した方法は，1 階の常微分方程式しか取り扱えないので不便である．そこで，高階の常微分方程式を 1 階の連立微分方程式に直す方法を示す．要するに，高階の常微分方程式を連立 1 階常微分方程式に直し，4 次のルンゲ・クッタ法を適用すれば良いのである．例えば，次のような 3 次の常微分方程式があったする．

$$y'''(x) = f(x, y, y', y'') \tag{4}$$

この 3 階常微分方程式を次に示す式を用いて変換する．

$$\begin{cases} y_0(x) = y(x) \\ y_1(x) = y'(x) \\ y_2(x) = y''(x) \end{cases} \tag{5}$$

この式を用いて，式 (4) を書き直すと

$$\begin{cases} y_0'(x) = y_1(x) \\ y_1'(x) = y_2(x) \\ y_2'(x) = f(x, y_0, y_1, y_2) \end{cases} \tag{6}$$

となる．これで，3 階の常微分方程式が 3 元の 1 階の連立常微分方程式に変換できた．2 階であろうが 4 階 $\cdots$ でも同じ方法で連立微分方程式に還元できる．

2.3.2 練習問題

以下の高次常微分方程式を連立 1 階微分方程式に書き換えなさい.

(1) $y'' + 3y' + 5y = 0$
(2) $y'' + 6y' + y = 0$
(3) $5y'' + 2xy' + 3y = 0$
(4) $y''' + y' + xy = 0$
(5) $5y'' + y' + y = \sin(\omega x)$
(6) $xy'' + y' + y = e^x$
(7) $5y''y' + y' + y = 0$
(8) $y''y' + x^2y'y + y = 0$

3 連立一次方程式 (消去法)

3.1 連立方程式の表現方法

連立 1 次方程式 (Linear Equations) は，次のような形をしている.

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1N}x_N &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2N}x_N &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \cdots + a_{3N}x_N &= b_3 \\ &\vdots \\ a_{M1}x_1 + a_{M2}x_2 + a_{M3}x_3 + \cdots + a_{MN}x_N &= b_M \end{aligned} \tag{7}$$

式 (7) は行列とベクトルで書くと，式がすっきりして考えやすくなる．書き直すと，

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{8}$$

である．それぞれの行列とベクトルは，

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1N} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2N} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3N} \\ \vdots & & & \ddots & \vdots \\ a_{N1} & a_{N2} & a_{N3} & \cdots & a_{NN} \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_N \end{bmatrix} \tag{9}$$

を表す.

通常，連立 1 次方程式 (7) は

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1N} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2N} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3N} \\ \vdots & & & \ddots & \vdots \\ a_{N1} & a_{N2} & a_{N3} & \cdots & a_{NN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_N \end{bmatrix} \tag{10}$$

と書き表せる．このようにすると，見通しがかなり良くなる.

3.2 ガウス・ジョルダン法の基本的な考え方

ガウス・ジョルダン (Gauss-Jordan) 法というのは，連立方程式 (10) を次にように変形させて，解く方法である．

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} b'_1 \\ b'_2 \\ b'_3 \\ \vdots \\ b'_N \end{bmatrix} \tag{11}$$

この式から明らかに，求める解 $x_i = b'_i$ となる．これをどうやって求めるか?．コンピューターで実際に計算する前に，人力でガウス・ジョルダン法で計算してみる．例として，

$$\begin{cases} 1x + 2y + 3 = 2 \\ 2x + 2y + 3z = 1 \\ 2x + 2y + 1z = -1 \end{cases} \tag{12}$$

をガウス・ジョルダン法で解を求める．

解くべき，方程式

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 2 & 3 \\ 2 & 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ -1 \end{bmatrix}$$

2 行 $-2\times$1 行

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & -2 & -3 \\ 2 & 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ -3 \\ -1 \end{bmatrix}$$

3 行 $-2\times$1 行

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & -2 & -3 \\ 0 & -2 & -5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ -3 \\ -5 \end{bmatrix}$$

$-\frac{1}{2}\times$2 行

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & \frac{3}{2} \\ 0 & -2 & -5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ \frac{3}{2} \\ -5 \end{bmatrix}$$

1 行 $-2\times$2 行

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \frac{3}{2} \\ 0 & -2 & -5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ \frac{3}{2} \\ -5 \end{bmatrix}$$

3 行 $+2\times$2 行

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \frac{3}{2} \\ 0 & 0 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ \frac{3}{2} \\ -2 \end{bmatrix}$$

$-\frac{1}{2}\times$3 行

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \frac{3}{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ \frac{3}{2} \\ 1 \end{bmatrix}$$

2 行 $-\frac{3}{2}\times$3 行

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

これで，ガウス・ジョルダン法による対角化の作業は完了である．これから，$(x_1, x_2, x_3) = (-1, 0, 1)$ と分かる．

3.3 逆行列

ガウス・ジョルダンを使って，逆行列が求められる．以下のようにする．解くべき，方程式

$$\begin{bmatrix}1&2&3\\2&2&3\\2&2&1\end{bmatrix}\left[\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix}\sqcup\begin{pmatrix}y_{11}&y_{12}&y_{13}\\y_{21}&y_{22}&y_{23}\\y_{31}&y_{32}&y_{33}\end{pmatrix}\right]=\left[\begin{pmatrix}2\\1\\-1\end{pmatrix}\sqcup\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}\right]$$

とする．

2 行 -2×1 行

$$\begin{bmatrix}1&2&3\\0&-2&-3\\2&2&1\end{bmatrix}\left[\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix}\sqcup\begin{pmatrix}y_{11}&y_{12}&y_{13}\\y_{21}&y_{22}&y_{23}\\y_{31}&y_{32}&y_{33}\end{pmatrix}\right]=\left[\begin{pmatrix}2\\-3\\-1\end{pmatrix}\sqcup\begin{pmatrix}1&0&0\\-2&1&0\\0&0&1\end{pmatrix}\right]$$

3 行 -2×1 行

$$\begin{bmatrix}1&2&3\\0&-2&-3\\0&-2&-5\end{bmatrix}\left[\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix}\sqcup\begin{pmatrix}y_{11}&y_{12}&y_{13}\\y_{21}&y_{22}&y_{23}\\y_{31}&y_{32}&y_{33}\end{pmatrix}\right]=\left[\begin{pmatrix}2\\-3\\-5\end{pmatrix}\sqcup\begin{pmatrix}1&0&0\\-2&1&0\\-2&0&1\end{pmatrix}\right]$$

$-\frac{1}{2}\times2$ 行

$$\begin{bmatrix}1&2&3\\0&1&\frac{3}{2}\\0&-2&-5\end{bmatrix}\left[\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix}\sqcup\begin{pmatrix}y_{11}&y_{12}&y_{13}\\y_{21}&y_{22}&y_{23}\\y_{31}&y_{32}&y_{33}\end{pmatrix}\right]=\left[\begin{pmatrix}2\\\frac{3}{2}\\-5\end{pmatrix}\sqcup\begin{pmatrix}1&0&0\\1&-\frac{1}{2}&0\\-2&0&1\end{pmatrix}\right]$$

1 行 -2×2 行

$$\begin{bmatrix}1&0&0\\0&1&\frac{3}{2}\\0&-2&-5\end{bmatrix}\left[\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix}\sqcup\begin{pmatrix}y_{11}&y_{12}&y_{13}\\y_{21}&y_{22}&y_{23}\\y_{31}&y_{32}&y_{33}\end{pmatrix}\right]=\left[\begin{pmatrix}-1\\\frac{3}{2}\\-5\end{pmatrix}\sqcup\begin{pmatrix}-1&1&0\\1&-\frac{1}{2}&0\\-2&0&1\end{pmatrix}\right]$$

3 行 $+2\times2$ 行

$$\begin{bmatrix}1&0&0\\0&1&\frac{3}{2}\\0&0&-2\end{bmatrix}\left[\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix}\sqcup\begin{pmatrix}y_{11}&y_{12}&y_{13}\\y_{21}&y_{22}&y_{23}\\y_{31}&y_{32}&y_{33}\end{pmatrix}\right]=\left[\begin{pmatrix}-1\\\frac{3}{2}\\-2\end{pmatrix}\sqcup\begin{pmatrix}-1&1&0\\1&-\frac{1}{2}&0\\0&-1&1\end{pmatrix}\right]$$

$-\frac{1}{2}\times3$ 行

$$\begin{bmatrix}1&0&0\\0&1&\frac{3}{2}\\0&0&1\end{bmatrix}\left[\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix}\sqcup\begin{pmatrix}y_{11}&y_{12}&y_{13}\\y_{21}&y_{22}&y_{23}\\y_{31}&y_{32}&y_{33}\end{pmatrix}\right]=\left[\begin{pmatrix}-1\\\frac{3}{2}\\1\end{pmatrix}\sqcup\begin{pmatrix}-1&1&0\\1&-\frac{1}{2}&0\\0&\frac{1}{2}&-\frac{1}{2}\end{pmatrix}\right]$$

2 行 $-\frac{3}{2}$×3 行

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \sqcup \begin{pmatrix} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \\ y_{31} & y_{32} & y_{33} \end{pmatrix} \right] = \left[\begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \sqcup \begin{pmatrix} -1 & 1 & 0 \\ 1 & -\frac{5}{4} & \frac{3}{4} \\ 0 & \frac{1}{2} & -\frac{1}{2} \end{pmatrix} \right]$$

これで，ガウス・ジョルダン法による対角化の作業は完了である．これから，$(x_1, x_2, x_3) = (-1, 0, 1)$ と分かる．さらに，逆行列が

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 2 & 3 \\ 2 & 2 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} -1 & 1 & 0 \\ 1 & -\frac{5}{4} & \frac{3}{4} \\ 0 & \frac{1}{2} & -\frac{1}{2} \end{bmatrix}$$

と分かる．

3.4 ガウス・ジョルダン法の C 言語の関数

ピボット選択は行わないで，逆行列も求めないのガウス・ジョルダン法で連立方程式を計算するプログラムを示す．このプログラムの動作は，次の通りである．

- 仮引数「n」は，解くべき連立方程式の未知数の数である．
- 仮引数の配列「a」と「b」は，係数行列 $\boldsymbol{A}$ と非同次項 $\boldsymbol{b}$ である．値は，呼び出し元からにアドレス渡しで送られる．
  - 係数行列は，配列「a[1][1]」〜「a[n][n]」に格納されている．
  - 非同次項は，配列「b[1]」〜「b[n]」に格納されている．
- 連立方程式の解 $\boldsymbol{x}$ は，配列「b[1]」〜「b[n]」に格納される．
- このプログラムでの処理が終了すると，配列「a[1][1]」〜「a[n][n]」は単位行列になる..

```
/* ========== ガウスジョルダン法の関数 =================*/
void gauss_jordan(int n, double a[][100], double b[]){

   int ipv, i, j;
   double inv_pivot, temp;

   for(ipv=1 ; ipv <= n ; ipv++){

      /* ---- 対角成分=1(ピボット行の処理) ---- */
      inv_pivot = 1.0/a[ipv][ipv];
      for(j=1 ; j <= n ; j++){
         a[ipv][j] *= inv_pivot;
      }
      b[ipv] *= inv_pivot;
```

```
        /* ---- ピボット列=0(ピボット行以外の処理) ---- */
        for(i=1 ; i<=n ; i++){
           if(i != ipv){
              temp = a[i][ipv];
              for(j=1 ; j<=n ; j++){
                 a[i][j] -= temp*a[ipv][j];
              }
              b[i] -= temp*b[ipv];
           }
        }

     }
  }
```

4 連立一次方程式 (反復法)

実用的なプログラムでは，非常に大きな連立方程式を計算しなくてはならない．数百万元に及ぶことも珍しくない．これを，ガウス・ジョルダン法で計算するの時間的にほとんど不可能である．そこで，これよりは格段に計算の速い方法が用いられる．ここでは，その一つとして反復法を簡単に説明する．

当然ここでも，連立方程式

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{13}$$

を満たす $\boldsymbol{x}$ を数値計算により，求めることになる．ここで，真の解 $\boldsymbol{x}$ とする．

ここで，ある計算により n 回目で求められたものを $\boldsymbol{x}^{(n)}$ とする．そして，計算回数を増やして，

$$\lim_{n\to\infty} \boldsymbol{x}^{(n)} = \boldsymbol{x} \tag{14}$$

になったとする．この様に計算回数を増やして，真の解に近づける方法を反復法という．

この様な方法は，行列 $\boldsymbol{A}$ を $\boldsymbol{S} - \boldsymbol{T}$ と分解するだけで，容易に作ることができる．たとえば，

$$\boldsymbol{S}\boldsymbol{x}^{(k+1)} = \boldsymbol{T}\boldsymbol{x}^{(k)} + \boldsymbol{b} \tag{15}$$

とすればよい．ここで，$\boldsymbol{x}^{(k)}$ が $\boldsymbol{\alpha}$ に収束するとする．すると，式 (15) と式 (13) を比べれば，$\boldsymbol{\alpha}$ と $\boldsymbol{x}$ は等しいことがわかる．すなわち，式 (15) で元の方程式 (13) を表した場合，$\boldsymbol{x}^{(k)}$ が収束すれば，必ず真の解 $\boldsymbol{x}$ に収束するのである．別の解に収束することはなく，真の解に収束するか，発散するかのいずれかである．

4.1 ヤコビ法

係数行列 $\boldsymbol{A}$ の対角行列を反復計算の行列 $\boldsymbol{S}$ としたものがヤコビ (Jacobi) 法である．係数行列を

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & 0 & 0 & \dots & 0 \\ 0 & a_{22} & 0 & \dots & 0 \\ 0 & 0 & a_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a_{nn} \end{bmatrix} + \begin{bmatrix} 0 & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & 0 & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & 0 & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & 0 \end{bmatrix} \tag{16}$$

と分解し，右辺第 1 項が行列 $\boldsymbol{S}$ で第 2 項が $-\boldsymbol{T}$ となる．$\boldsymbol{x}^{(k+1)}$ の解の計算に必要な $\boldsymbol{S}$ の逆行列は，それが対角行列なので，

$$\boldsymbol{S}^{-1} = \begin{bmatrix} a_{11}^{-1} & 0 & 0 & \dots & 0 \\ 0 & a_{22}^{-1} & 0 & \dots & 0 \\ 0 & 0 & a_{33}^{-1} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a_{nn}^{-1} \end{bmatrix} \tag{17}$$

と簡単に求めることができる．$k+1$ 番目の近似解は $\boldsymbol{x}^{(k+1)} = \boldsymbol{S}^{-1}\left(\boldsymbol{b} + \boldsymbol{T}\boldsymbol{x}^{(k)}\right)$ となるなので，容易に求めることができる．実際，k 番目の解を

$$x_1^{(k)},\, x_2^{(k)},\, x_3^{(k)},\, \cdots,\, x_n^{(k)}$$

とすると，k+1 番目の解は

$$\begin{aligned} x_1^{(k+1)} &= a_{11}^{-1}\left\{b_1 - \left(a_{12}x_2^{(k)} + a_{13}x_3^{(k)} + a_{14}x_4^{(k)} + \cdots + a_{1n}x_n^{(k)}\right)\right\} \\ x_2^{(k+1)} &= a_{22}^{-1}\left\{b_2 - \left(a_{21}x_1^{(k)} + a_{23}x_3^{(k)} + a_{24}x_4^{(k)} + \cdots + a_{2n}x_n^{(k)}\right)\right\} \\ x_3^{(k+1)} &= a_{33}^{-1}\left\{b_3 - \left(a_{31}x_1^{(k)} + a_{32}x_2^{(k)} + a_{34}x_4^{(k)} + \cdots + a_{3n}x_n^{(k)}\right)\right\} \\ &\vdots \\ x_n^{(k+1)} &= a_{nn}^{-1}\left\{b_n - \left(a_{n1}x_1^{(k)} + a_{n2}x_2^{(k)} + a_{n3}x_3^{(k)} + \cdots + a_{nn-1}x_{n-1}^{(k)}\right)\right\} \end{aligned} \tag{18}$$

と計算できる．これを繰り返して連立方程式の解を求める方法が，ヤコビ法である．

4.2 ガウス・ザイデル法

ヤコビ法の特徴では，$\boldsymbol{x}^{(k+1)}$ の近似値は，すべてその前の値 $\boldsymbol{x}^{(k)}$ を使うことにある．大きな行列を扱う場合，全ての $\boldsymbol{x}^{(k+1)}$ と $\boldsymbol{x}^{(k)}$ を記憶する必要があり，大きなメモリーが必要となり問題が生じる．今では，個人で大きなメモリーを使い計算することは許されるが，ちょっと前まではできるだけメモリーを節約したプログラムを書かなくてはならなかった．

そこで，$\boldsymbol{x}^{(k+1)}$ の各成分の計算が終わると，それを直ちに使うことが考えば，メモリーは半分で済む．即ち，$x_i^{(k+1)}$ を計算するときに，

$$x_i^{(k+1)} = a_{ii}^{-1}\left\{b_i - (a_{i1}x_1^{(k+1)} + a_{i2}x_2^{(k+1)} + a_{i3}x_3^{(k+1)} + \cdots + a_{ii-1}x_{i-1}^{(k+1)} + a_{ii+1}x_{i+1}^{(k)} + a_{ii+2}x_{i+2}^{(k)} + a_{ii+3}x_{i+3}^{(k)} + \cdots + a_{in}x_n^{(k)})\right\} \tag{19}$$

とするのである．実際の計算では，$k+1$ 番目の解は

$$\begin{aligned}
x_1^{(k+1)} &= a_{11}^{-1}\left\{b_1 - \left(a_{12}x_2^{(k)} + a_{13}x_3^{(k)} + a_{14}x_4^{(k)} + \cdots + a_{1n}x_n^{(k)}\right)\right\} \\
x_2^{(k+1)} &= a_{22}^{-1}\left\{b_2 - \left(a_{21}x_1^{(k+1)} + a_{23}x_3^{(k)} + a_{24}x_4^{(k)} + \cdots + a_{2n}x_n^{(k)}\right)\right\} \\
x_3^{(k+1)} &= a_{33}^{-1}\left\{b_3 - \left(a_{31}x_1^{(k+1)} + a_{32}x_2^{(k+1)} + a_{34}x_4^{(k)} + \cdots + a_{3n}x_n^{(k)}\right)\right\} \\
&\vdots \\
x_n^{(k+1)} &= a_{nn}^{-1}\left\{b_n - \left(a_{n1}x_1^{(k+1)} + a_{n2}x_2^{(k+1)} + a_{n3}x_3^{(k+1)} + \cdots + a_{nn-1}x_{n-1}^{(k+1)}\right)\right\}
\end{aligned} \tag{20}$$

と計算できる．これを繰り返して連立方程式の解を求める方法が，ガウス・ザイデル法である．

4.3 SOR 法

ガウス・ザイデル法をもっと改善する方法がある．ガウス・ザイデル法の解の修正は，$x_{k+1} - x_k$ であったが，これをもっと大きなステップにしようというのである．通常の場合，ガウス・ザイデル法では近似解はいつも同じ側にあり，単調に収束する．そのため，修正を適当にすれば，もっと早く解に近づく．修正幅を，加速緩和乗数 ω を用いて，$\omega(x_{k+1} - x_k)$ とする事が考えられた．これが，逐次加速緩和法 (SOR 法: Successive Over-Relaxation) である．

具体的な計算手順は，次のようにする．ここでは，ガウス・ザイデル法の式 (20) を用いて，得られた近似解を $\tilde{x}_i^{(k+1)}$ としている．

$$\begin{aligned}
\tilde{x}_1^{(k+1)} &= a_{11}^{-1}\left\{b_1 - \left(a_{12}x_2^{(k)} + a_{13}x_3^{(k)} + a_{14}x_4^{(k)} + \cdots + a_{1n}x_n^{(k)}\right)\right\} \\
x_1^{(k+1)} &= x_1^{(k)} + \omega\left(\tilde{x}_1^{(k+1)} - x_1^{(k)}\right) \\
\tilde{x}_2^{(k+1)} &= a_{22}^{-1}\left\{b_2 - \left(a_{21}x_1^{(k+1)} + a_{23}x_3^{(k)} + a_{24}x_4^{(k)} + \cdots + a_{2n}x_n^{(k)}\right)\right\} \\
x_2^{(k+1)} &= x_2^{(k)} + \omega\left(\tilde{x}_2^{(k+1)} - x_2^{(k)}\right) \\
\tilde{x}_3^{(k+1)} &= a_{33}^{-1}\left\{b_3 - \left(a_{31}x_1^{(k+1)} + a_{32}x_2^{(k+1)} + a_{34}x_4^{(k)} + \cdots + a_{3n}x_n^{(k)}\right)\right\} \\
x_3^{(k+1)} &= x_3^{(k)} + \omega\left(\tilde{x}_3^{(k+1)} - x_3^{(k)}\right) \\
&\vdots \\
\tilde{x}_n^{(k+1)} &= a_{nn}^{-1}\left\{b_n - \left(a_{n1}x_1^{(k+1)} + a_{n2}x_2^{(k+1)} + a_{n3}x_3^{(k+1)} + \cdots + a_{nn-1}x_{n-1}^{(k+1)}\right)\right\} \\
x_n^{(k+1)} &= x_n^{(k)} + \omega\left(\tilde{x}_n^{(k+1)} - x_n^{(k)}\right)
\end{aligned} \tag{21}$$

これを繰り返して連立方程式の解を求める方法が，SOR 法である．

ここで，問題なのが加速緩和係数 ω の値の選び方である．明らかに，それが 1 の場合，ガウス・ザイデル法となりメリットは無い．また，1 以下だと，ガウス・ザイデル法よりも収束が遅い．ただし，ガウス・ザイデル法で収束しないような問題には使える．

従って，1 以上の値にしたいわけであるが，余り大きくすると，発散するのは目に見えている．これについては，2 を越えると発散することが分かっている．最適値となると，だいたい 1.9 くらいが選ばれることが多い．

4.4 プログラム例 (ガウス・ザイデル法)

ガウス・ザイデル法のような反復法は大きな連立方程式の計算に敵している．しかし，ここではその計算原理を分かり易くするため，次の連立方程式を計算する．

$$\begin{bmatrix} 3 & 2 & 1 \\ 1 & 4 & 1 \\ 2 & 2 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 10 \\ 12 \\ 21 \end{bmatrix} \tag{22}$$

式 (20) の漸化式に従うプログラム例を以下に示す．

```
#include <stdio.h>
#include <math.h>
#define N (3)                  // 連立方程式の大きさ
#define EPS (1e-15)            // 計算誤差の許容値

int main(void){
  double a[N+1][N+1], x[N+1], b[N+1];
  double dx, absx, sum, new;
  int i,j;

  a[1][1]=3.0;  a[1][2]=2.0;  a[1][3]=1.0;  // 係数行列
  a[2][1]=1.0;  a[2][2]=4.0;  a[2][3]=1.0;
  a[3][1]=2.0;  a[3][2]=2.0;  a[3][3]=5.0;

  b[1]=10.0;                              // 同次項
  b[2]=12.0;
  b[3]=21.0;

  x[1]=0.0;                               // 近似解の初期値
  x[2]=0.0;
  x[3]=0.0;

  do{                                     // 反復計算のループ
    dx=0.0;
    absx=0.0;

    for(i=1;i<=N;i++){
      sum=0;
```

```
      for(j=1;j<=N;j++){
if(i != j){
  sum+=a[i][j]*x[j];
}
      }

      new=1.0/a[i][i]*(b[i]-sum);   // 反復計算後の近似解
      dx+=fabs(new-x[i]);           // 近似解の変化量を加算
      absx+=fabs(new);              // 近似解の総和計算
      x[i]=new;                     // 新しい近似解を代入
    }

  }while(dx/absx > EPS);            // 計算終了条件

  for(i=1;i<=N;i++){
    printf("x[%d]=%25.20f\n",i,x[i]);
  }

  return 0;
}
```

5 まとめ

この試験範囲で理解すべきことをまとめると，以下のようになる．

- 常微分方程式
  - 4 次のルンゲクッタ法の漸化式くらいは，暗記すること．イメージが湧けば，そんなに難しくはない．
  - 4 次のルンゲクッタ法のプログラムの内容が理解できること．
  - 高階の微分方程式を連立の 1 階の微分方程式に直せること．
- 連立 1 次方程式 (消去法)
  - ガウス・ジョルダン法で連立方程式が計算できること．
  - ガウス・ジョルダン法で逆行列が計算できること．
  - ガウス・ジョルダン法の C 言語の関数が理解できること．
- 連立 1 次方程式 (反復法)
  - ガウス・ザイデル法の漸化式くらい導けること．
  - ガウス・ザイデル法で C 言語のプログラムがかけること．