第１２章　偏微分方程式の差分解法入門

偏微分方程式を解析的に解く方法としてラプラス変換やフーリエ変換の方法が用いられているが，少し複雑な問題になると適用できない場合が多い。その場合は数値的に微分方程式を解く手法が用いられる。偏微分方程式の場合, 差分法, 有限要素法, 境界要素法などの解法があり，複雑な境界の問題などでは，有限要素法がよくもちいられ，化学工学分野でも複雑な形状の物体の熱計算などでよく用いられている。しかし，本講義では一番わかりやすい差分法を扱うことにする。差分法も最近は，複雑な境界を扱う手法が発達してきており, 原理が単純ため, ナヴィエストークス方程式（流動の基本方程式）のような非線形偏微分方程式を扱う手法として見直されている。

本講義では，解の安定性など数学的な難しい部分は直感的な説明にとどめるので詳しい解説は行わないが，詳しいことを知りたい人は次の本をみてほしい。

田辺・高見監修，高見・河村著「偏微分方程式の差分解法」東京大学出版会

１）一次元非定常熱伝導方程式（陽解法）

一次元非定常熱伝導方程式の初期値，境界値問題を考える。

$$\frac{\partial T(x,t)}{\partial t} = D_T \frac{\partial^2 T(x,t)}{\partial^2 x} \tag{12-1}$$

ここで，T，x，t，D_T は，それぞれ温度，座標，時刻，熱拡散率である。この式の導出については，さまざまな成書や HP に書かれている。以前のカリキュラムで私が行っていた 2005 年度の「化学工学数学」の HP が残してあり，（http://www.nuce.nagoya-u.ac.jp/e8/Matsuoka/MathCE/05MathCE.html）この資料の 10/18 の講義資料に導出方法が記載されている。

初期条件は

$$T(x,0) = T_{\mathrm{i}}(x) \tag{12-2}$$

境界条件として

$$T(0,t) = T_{\mathrm{b0}}(t), \quad T(l,t) = T_{\mathrm{be}}(t) \tag{12-3}$$

のものを考える。このように，各時刻の境界の温度の時間依存性が与えられている問題を第 1 種の境界値問題（Dirichlet 問題）という。これを差分法で解くことにする。この問題は，化学工学実験において Excel による解法を学んだと思うが，Octave での解法を学んでみよう。

また，差分法には，陽解法と陰解法があるが，まず，わかりやすい陽解法について解説し，陰解法についてはそのあと解説する。

差分解法であるので式(12-1)を差分化する必要がある。まずは，$|h| << 1$ とき関数 $f(x)$ の Taylor（テイラー）展開は

$$f(x \pm h) \approx f(x) \pm h f'(x) + \frac{h^2}{2} f''(x) + \cdots \tag{12-4}$$

となり，これより

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \tag{12-5}$$

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \tag{12-6}$$

式(12-5)，(12-6)を用いて式(12-1)を差分化する。

$$\frac{T(x,t+\Delta t)-T(x,t)}{\Delta t} \approx D_T \frac{T(x+\Delta x,t)-2T(x,t)+T(x-\Delta x,t)}{\Delta x^2} \tag{12-7}$$

ここで，無次元量

$$\lambda = D_T \frac{\Delta t}{\Delta x^2} \tag{12-8}$$

を考えると式(12-8)は，

$$T(x,t+\Delta t) = \lambda T(x+\Delta x,t) + (1-2\lambda)T(x,t) + \lambda T(x-\Delta x,t) \tag{12-9}$$

と整理できる。この式は，時刻 t における近接した３点の情報から，新しい時刻 $t+\Delta t$ の情報を得るもので前進差分公式と呼ばれる。簡単のために Δt ，Δx は，常に一定とし，$\Delta x = l/N$ とする。温度を２次元の数列 $T_{n,m}$ （$n = 1, 2, \cdots N, N+1$: $m = 1, 2, \cdots,$ ）として考える。

$$T_{n,m} = T((n-1)\Delta x, (m-1)\Delta t) \tag{12-10}$$

となる。式(12-9)は

$$T_{n,m+1} = \lambda T_{n+1,m} + (1-2\lambda)T_{n,m} + \lambda T_{n-1,m} \tag{12-11}$$

式(12-2), (12-3)より

$$T_{n,1} = T_{\mathrm{i}}((n-1)\Delta x) \tag{12-12}$$

$$T_{1,m} = T_{\mathrm{b0}}((m-1)t), \quad T_{N+1,m} = T_{\mathrm{be}}((m-1)t) \tag{12-13}$$

式 (12-13)より，境界値は常にあたえられているので式 (12-11)の差分式は $n = 2, 3, \cdots N$ の範囲で使用する。この差分式に関して次の定理が知られている。

定理　式(12-8)で定義した無次元量 $\lambda = D_T \Delta t/\Delta x^2$ について，$0 < \lambda \le 1/2$ では，$T_{n,m}$ は $\Delta x \to 0$ で真の解に一様に収束する。$\lambda > 1/2$ では，式 (12-11)の差分式は不安定になり，収束しない。

したがって，λ は，$0 < \lambda \le 1/2$ の範囲で選ぶことになる。直感的にいうと $\lambda > 1/2$ では式 (12-11)の $(1-2\lambda)T_{n,m}$ が負となる。温度が正の場合，すべての配列は正でなければならないから，負となる項が存在することにより，式が不安定になる。

また，別の定理により，$\lambda = 1/6$ に選ぶと解の近似次数が $O(\Delta x^4)$ から $O(\Delta x^6)$ となる。これについては，先にあげた 2005 年度「化学工学数学」の 11/8 の配布資料に解説がある。計算量は，λ が大きいほど減るので計算量を小さくしたいときは，$\lambda = 1/2$ にする。$\lambda = 1/2$ とすると式 (12-11)は，

$$T_{n,m+1} = (T_{n+1,m} + T_{n-1,m})/2 \tag{12-14}$$

差分式も簡単になり計算量がさらに減る。また，精度を重視するときには，$\lambda = 1/6$ とする。

さて，式 (12-11)の差分式を Octave で解くことを考える。思いつくのは `for` ループによる繰り返しであるが，６章２節で述べたように `for` ループの使用はできるだけ少なくするのがよい。Octave では（MATLAB や Scilab でも），行列計算は得意であるので，行列を使用するのがよい。式 (12-11)を行列で表すと

$$\begin{pmatrix} T_{1,m+1} \\ T_{2,m+1} \\ \vdots \\ \vdots \\ T_{n,m+1} \\ \vdots \\ \vdots \\ T_{N,m+!} \\ T_{N+1,m+1} \end{pmatrix} = \begin{pmatrix} a_m & & & & & & & & & \\ \lambda & 1-2\lambda & \lambda & & & & & & & \\ & \ddots & \ddots & \ddots & & & & & & \\ & & \ddots & \ddots & \ddots & & & & & \\ & & & \lambda & 1-2\lambda & \lambda & & & & \\ & & & & \ddots & \ddots & \ddots & & & \\ & & & & & \ddots & \ddots & \ddots & & \\ & & & & & & \lambda & 1-2\lambda & \lambda & \\ & & & & & & & & & b_m \end{pmatrix} \begin{pmatrix} T_{1,m} \\ T_{2,m} \\ \vdots \\ \vdots \\ T_{n,m} \\ \vdots \\ \vdots \\ T_{N,m} \\ T_{N,m} \end{pmatrix} \tag{12-15}$$

$T_{1,m+1}$，$T_{N+1,m+1}$ については，境界条件できまるので，両端は，境界条件で決まるので，係数 a_m, b_m は，境界条件で決まる。この問題では，端の温度は時間に依存せず一定なので $a_m = b_m = 1$ である。また，その他の記載されていない要素はすべて 0 である。式(12-11)は，スパース行列だが，対角要素を中心として帯状に 0 でない要素が並び，その他が 0 でない形をしていて特に帯行列（Band Matrix）と呼ばれている。Octave-2.1.50（正確には Octave-2.1.50 の Octave-forge）では，帯行列もスパース行列としてあつかう。スパース行列は，次の sparse 関数で定義される。

関数	
1. S = sparse(A) 2. S = sparse(i,j,s,m,n,nzmax) 3. S = sparse(i,j,s,m,n)	スパース行列を定義する関数 1. 通常の行列で，0 の要素の多い行列と同じ内容を持つスパース行列を定義する関数。 2. i,j は，インデックスを表すベクトル，i と j のベクトル要素の数は等しくなければならない。s は，i,j インデックスに対応する要素のベクトル，もしすべての要素が同じ値のときはスカラーを指定してよい。m,n は行列のサイズを指定する整数。Nzmax は MATLAB との互換のためにと Help に書かれていたが MATLAB の本を見ても省略してあったので 3. の書式でよい。

これだけではわかりずらいので例をあげて説明する。
まずは 1. の例を示す。

```
>> A=eye(5)
A =

     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1

>> SA=sparse(A)
SA =
```

```
Compressed Column Sparse (rows=5, cols=5, nnz=5)
  (1 , 1) -> 1
  (2 , 2) -> 1
  (3 , 3) -> 1
  (4 , 4) -> 1
  (5 , 5) -> 1
```

このように，0 でない要素だけがメモリに入り，メモリが節約された形になる。

まずは，要素の数，パラメータλ，$1-2\lambda$ の定義，対角項の定義である。ここでは近似精度の高い$\lambda=1/6$ とした。

```
>> N=5
N = 5
>> lambda=1.0/6
lambda = 0.16667
>> os2lambda=1-2*lambda
os2lambda = 0.66667
>> dia=[1 (ones(1,N-1)*os2lambda) 1]
dia =

  1.00000  0.66667  0.66667  0.66667  0.66667  1.00000

>> Cd=sparse(1:N+1,1:N+1,dia,N+1,N+1)
Cd =

Compressed Column Sparse (rows=6, cols=6, nnz=6)
  (1 , 1) -> 1
  (2 , 2) -> 0.66667
  (3 , 3) -> 0.66667
  (4 , 4) -> 0.66667
  (5 , 5) -> 0.66667
  (6 , 6) -> 1
```

最初の(2:N,2:Nで２行２列目からＮ行Ｎ列目までの対角要素を指定する。第3引数は，両端が 1 であり，それ以外は$1-2\lambda$ の値である。したがって，dia=[1 (ones(1,N-1)*os2lambda) 1]で対応するベクトルをつくり。行列は，区関数より 1 多い要素をもっているので，第4,5引数はともにN+1を指定した。

さて，非対角要素であるがまずは，上側については2行3列，3行4列,... と進んでいくので

```
>> Cu=sparse(2:N,3:N+1,lambda,N+1,N+1)
Cu =

Compressed Column Sparse (rows=6, cols=6, nnz=4)
  (2 , 3) -> 0.16667
  (3 , 4) -> 0.16667
  (4 , 5) -> 0.16667
  (5 , 6) -> 0.16667
```

(2:N,3:N+1 のように第 1 引数と第 2 引数が一つずつずらして定義されている。
下側も同様に

```
>> Cl=sparse(2:N,1:N-1,lambda,N+1,N+1)
Cl =

Compressed Column Sparse (rows=6, cols=6, nnz=4)
  (2 , 1) -> 0.16667
  (3 , 2) -> 0.16667
  (4 , 3) -> 0.16667
  (5 , 4) -> 0.16667
```

となる。Cd, Cu, Cl を加えることにより望んだスパース行列が得られる。

```
>> C=Cd+Cu+Cl
C =

Compressed Column Sparse (rows=6, cols=6, nnz=14)
  (1 , 1) -> 1
  (2 , 1) -> 0.16667
  (2 , 2) -> 0.66667
  (3 , 2) -> 0.16667
  (2 , 3) -> 0.16667
  (3 , 3) -> 0.66667
  (4 , 3) -> 0.16667
  (3 , 4) -> 0.16667
  (4 , 4) -> 0.66667
  (5 , 4) -> 0.16667
  (4 , 5) -> 0.16667
  (5 , 5) -> 0.66667
  (5 , 6) -> 0.16667
  (6 , 6) -> 1

>> full(C)
ans =

  1.00000  0.00000  0.00000  0.00000  0.00000  0.00000
  0.16667  0.66667  0.16667  0.00000  0.00000  0.00000
  0.00000  0.16667  0.66667  0.16667  0.00000  0.00000
  0.00000  0.00000  0.16667  0.66667  0.16667  0.00000
  0.00000  0.00000  0.00000  0.16667  0.66667  0.16667
  0.00000  0.00000  0.00000  0.00000  0.00000  1.00000
```

最後に full(C) で望んだ形になっていることを確認した。
さてこれを踏まえて熱伝導方程式の数値解法のプログラムを作成した。
問題は次のようである。
50°C で熱平衡に達していた熱拡散率 $1.75\mathrm{cm}^2\,\mathrm{s}^{-1}$ の 10 cm 棒を端以外は断熱した状態で時刻 t = 0 s で 30°C の水浴に浸した。温度分布の時間依存性を解析せよ。
スクリプトを以下に示す。

exdiff.m

```
% Octave script m file
% solve one dimension  diffusion partial differential eq. by explicit method
% main script
hold off;subplot(111);clg;clear all;
%
% Constants for Calculation
D=1.75; % cm^2/s diffusion constant
length=10; % cm
N=100; % Divsion Number of the length
lambda=1/6;
os2lambda=1-2*lambda;
TR=30000; % Time Range
Tint=100; % Time interval
Dint=Tint*10; % Time interval for final graph
SaveFname="Tsave.txt"; % File name to save data
fprintf("Simulation of Temperature distribution in one dimensional thermal
diffusion¥n");
fprintf("  Thermal Diffusion Constant : %7.4f cm^2/s ¥n", D);
fprintf("  Length : %7.4f cm¥n",length);
fprintf("  Number of divided of x range : %d ¥n",N);
fprintf("  Value of parameter lambda : %7.4f ¥n",lambda);
fprintf("  Number of time iteration : %d ¥n", TR);
fprintf("  File name to save data : %s¥n", SaveFname);
%
N1=N+1;
TR1=TR+1;
%
%
DeltaX=length/N;
Deltat=lambda*DeltaX^2/D;
X=[0:DeltaX:length]';
%
% construct a sparse Matrix
dia=[1 (ones(1,N-1)*os2lambda) 1];
Cd=sparse(1:N+1,1:N+1,dia,N+1,N+1);
Cu=sparse(2:N,3:N+1,lambda,N+1,N+1);
Cl=sparse(2:N,1:N-1,lambda,N+1,N+1);
C=Cd+Cu+Cl;
% Initial and boundary condition
Tp=[30 (ones(1,N-1)*50) 30]';  %Initial condition
Tb1=ones(TR1+1,1)*30;
Tb2=Tb1;
%
Tsave = fopen (SaveFname, "w");
% solve PDE by iteration
for m=1:TR1
  if (mod((m-1),Tint) == 0)
   plot(X,Tp, sprintf(";%d: t=%7.4f s;",(m-1),(m-1)*Deltat));
   xlabel("x / cm¥n");  ylabel("T / deg¥n"); axis([0 10 30 50]);
```

```
      drawnow;  % required for Octave 2.9.xxx
      fprintf(Tsave, "%e ",Tp); fputs(Tsave,"¥n");
    end
    Tn=C*Tp;
    Tp=Tn;
  end
  fclose(Tsave);
  input("Press any key to continue");
  Tread = fopen (SaveFname, "r");
  Tx=zeros(1,N1);
  clg;
  xlabel("x / cm¥n");  ylabel("T / deg¥n");  axis([0 10 30 50]);
  hold on
  for m=1:Tint:TR1
     sTx=[ "[" fgets(Tread) "]" ];
     Tx=eval(sTx);
    if (mod((m-1),Dint) == 0)
      clr=mod(floor((m-1)/Dint),5)+1;
     ;
      plot(X,Tx', sprintf("%d;%d: t=%7.4f s;",clr,(m-1),(m-1)*Deltat));
    end
  end
  fclose(Tread);
```

このスプリプトも凝ったので長くなったが流れとしては次のようになる。

1．物性値である熱拡散率と棒の長さを変数に代入する。

2．x の分割数と時間のレンジを考える。前進差分公式を使った方法では，$0 < \lambda \leq 1/2$ と言う制限があり，$\lambda = D_T \Delta t / \Delta x^2$ という条件から x の分割数の２乗に比例して Δt が小さくなる。したがって，メモリの節約のために，すべてのデータを保存せずに，インターバルをとって保存するようにしている。

3．初期条件と境界条件を決める。初期条件は次に示す Tp に入れる。

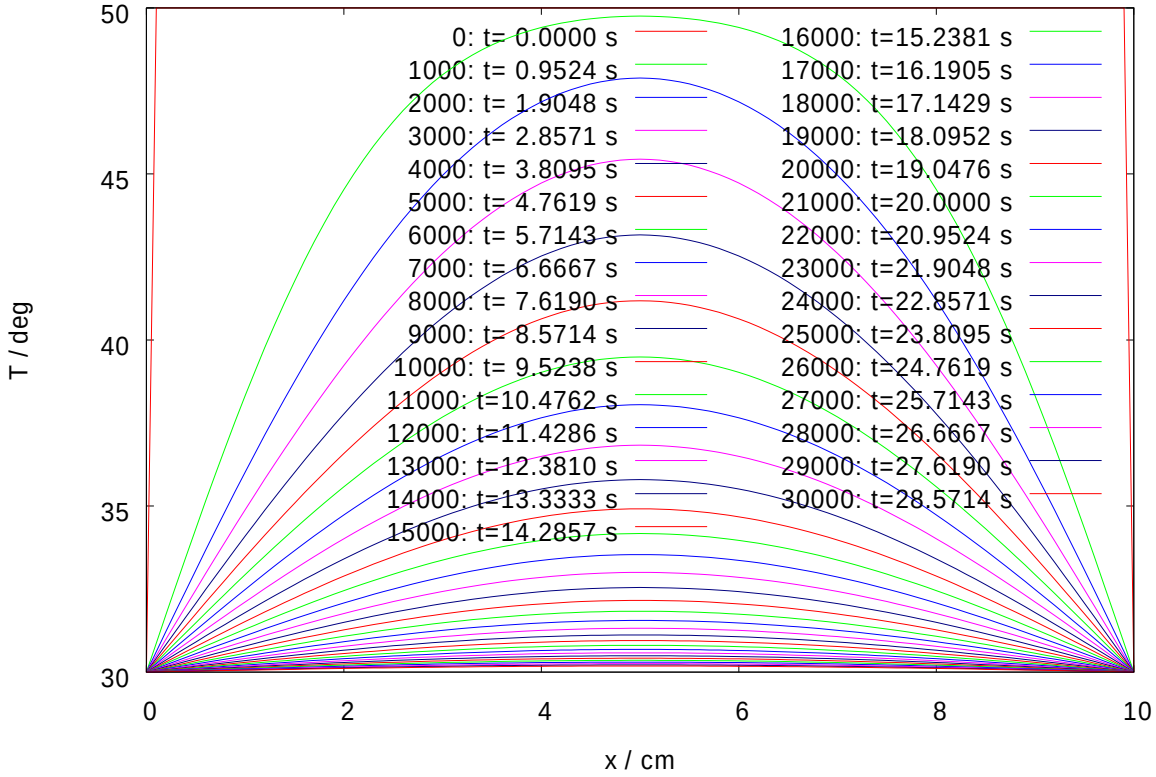
4．for ループで計算する。また，for ループでは，Tn=C*Tp をかけて Tn とし，計算した温度のベクトルを Previous という意味で Tp とし,次の m にまわす。次の m で，また，Tn=C*Tp をかけて Tn とし，これをまた Tp に移すという作業を繰り返して進めていく。あるステップごとに保存するため割り算の剰余を計算する mod を利用して，一定の間隔ごとにデータをセーブする。データのセーブには，C ライクな fopen fprintf, fclose などの関数を利用した。また，グラフもそのとき表示させ，時間の推移とともに温度分布が変わっていく様子を見えるようにした。また，Octave 2.9.12 では，drawnow コマンドを入れる必要があり，軸に対する設定もその都度行う必要があったので，2.9.12 でも動作するようにした。

5．“Press any key to continue”で最後に保存したデータを適当なインターバルで読み出し，最後のグラフを作成した。

Octave の出力画面と最終的なグラフを示す。

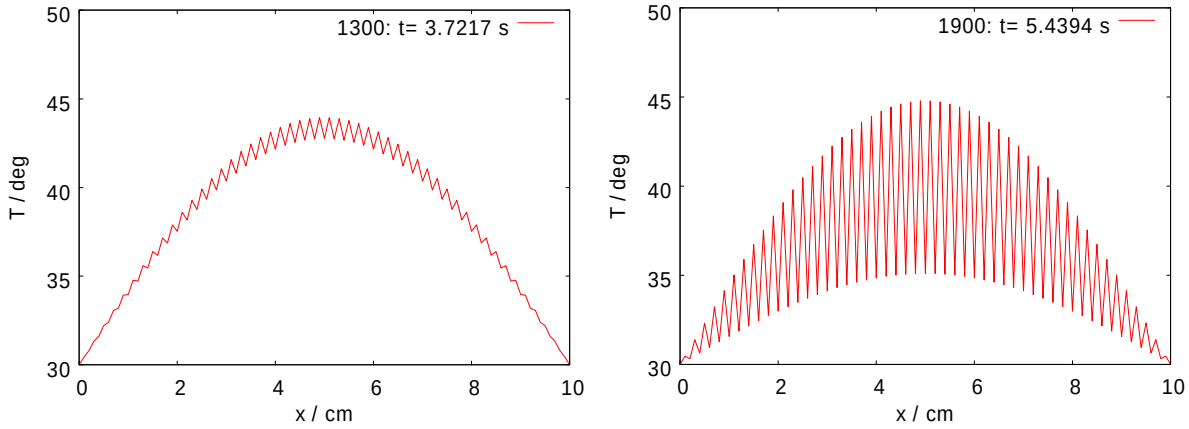
```
>> exdiff
Simulation of Temperature distribution in one dimensional thermal diffusion
  Thermal Diffusion Constant :  1.7500 cm^2/
  Length : 10.0000 cm
  Number of divided of x range : 100
```

```
      Number of time iteration : 30000
      File name to save data : Tsave.txt
   Press any key to continue
```



2）一次元非定常熱伝導方程式（陰解法）

先ほどのプログラムで $\lambda = 1/2 + 0.001$ としたところこのような時間とともに解が下図のように振動し，いずれは発散してしまう。



したがって，1）取り上げた前進差分公式による方法では，$\lambda \le 1/2$ の条件は厳密に守らなければならない。$\lambda = D_T \Delta t / \Delta x^2$ という条件から x の分割数の 2 乗に比例して Δt が小さくなる。したがって，x 方向の分解能を上げたいときには，進差分公式による方法はあまりよい方法とはいえない。そこで，λ の値にそれほど敏感ではない方法が

あると便利である。まず差分近似式を次のように考える。

$$\frac{T(x,t+\Delta t)-T(x,t)}{\Delta t} \approx D_T \frac{T(x+\Delta x,t+\Delta t)-2T(x,t+\Delta t)+T(x-\Delta x,t+\Delta t)}{\Delta x^2} \tag{12-16}$$

以前使用した式(12-7)は

$$\frac{T(x,t+\Delta t)-T(x,t)}{\Delta t} \approx D_T \frac{T(x+\Delta x,t)-2T(x,t)+T(x-\Delta x,t)}{\Delta x^2} \tag{12-17}$$

であった。左辺の時刻が t ではなく $t+\Delta t$ になっていることに注意しよう。

$$-\lambda T(x+\Delta x,t+\Delta t)+(1+2\lambda)T(x,t+\Delta t)-\lambda T(x-\Delta x,t+\Delta t)=T(x,t) \tag{12-18}$$

式(12-10)と同様に

$$T_{n,m}=T((n-1)\Delta x,(m-1)\Delta t) \tag{12-19}$$

を考えると

$$-\lambda T_{n+1,m+1}+(1+2\lambda)T_{n,m+1}-\lambda T_{n-1,m+1}=T_{n,m} \tag{12-20}$$

初期条件，境界条件は同様により

$$T_{n,1}=T_{\mathrm{i}}((n-1)\Delta x) \tag{12-21}$$

$$T_{1,m}=T_{\mathrm{b0}}((m-1)t),\quad T_{N+1,m}=T_{\mathrm{be}}((m-1)t) \tag{12-22}$$

式(12-20)は，３つの新しい時刻の要素が一つの古い時刻の要素に対応しており，後退差分公式と呼ばれる。これだけでは，何も決まらないようだが，これを組み合わせて連立方程式が構成できる。行列形式で書けば

$$\begin{pmatrix} a_m & & & & & & & & \\ -\lambda & 1+2\lambda & -\lambda & & & & & & \\ & \ddots & \ddots & \ddots & & & & & \\ & & \ddots & \ddots & \ddots & & & & \\ & & & -\lambda & 1+2\lambda & -\lambda & & & \\ & & & & \ddots & \ddots & \ddots & & \\ & & & & & \ddots & \ddots & \ddots & \\ & & & & & & -\lambda & 1+2\lambda & -\lambda \\ & & & & & & & & b_m \end{pmatrix} \begin{pmatrix} T_{1,m+1} \\ T_{2,m+1} \\ \vdots \\ \vdots \\ T_{n,m+1} \\ \vdots \\ \vdots \\ T_{N,m+!} \\ T_{N+1,m+1} \end{pmatrix} = \begin{pmatrix} T_{1,m} \\ T_{2,m} \\ \vdots \\ \vdots \\ T_{n,m} \\ \vdots \\ \vdots \\ T_{N,m} \\ T_{N,m} \end{pmatrix} \tag{12-23}$$

となる。前節と同様，両端は，境界条件で決まるので，係数 a_m, b_m は，境界条件で決まる。前節と同じ問題を考えるとき，端の温度は時間に依存せず一定なので $a_m=b_m=1$ である。この連立方程式を解くことにより，一連の $\{T_{n,m}\}$ から $\{T_{n,m+1}\}$ が決まる。

このように，$\{T_{\mathrm{i,j+1}}\}$ の各要素は上の連立方程式の解を解くことで初めて与えられる。前節の方法では $\{T_{\mathrm{i,j+1}}\}$ の各要素は $\{T_{\mathrm{i,j}}\}$ の３つの要素で連立方程式を解くことなしに与えられていた。前回のように $\{T_{\mathrm{i,j+1}}\}$ の $\{T_{\mathrm{i,j}}\}$ の少数の要素から直接計算できる解法を**陽解法**とよび今回のように連立方程式を解くなどしてはじめて求められる場合を**陰解法**とよぶ。

式(12-23)は３項方程式と呼ばれており，2.1.50 の octave-forge にはその解法が収録されているが，最新の octave-forge には収録されていないようである。したがってここではスパース行列を定義し，スパース行列の左除算で解を求めることにする。詳しい説明は，省略するが，この解法は λ の値によらず安定である。したがって，x 軸方向の分割数によらず時間の刻み幅を任意にとることが可能である。熱拡散方程式は，初期は変動が大きく，後半は変動が少ないことから途中から刻み幅を変更することも可

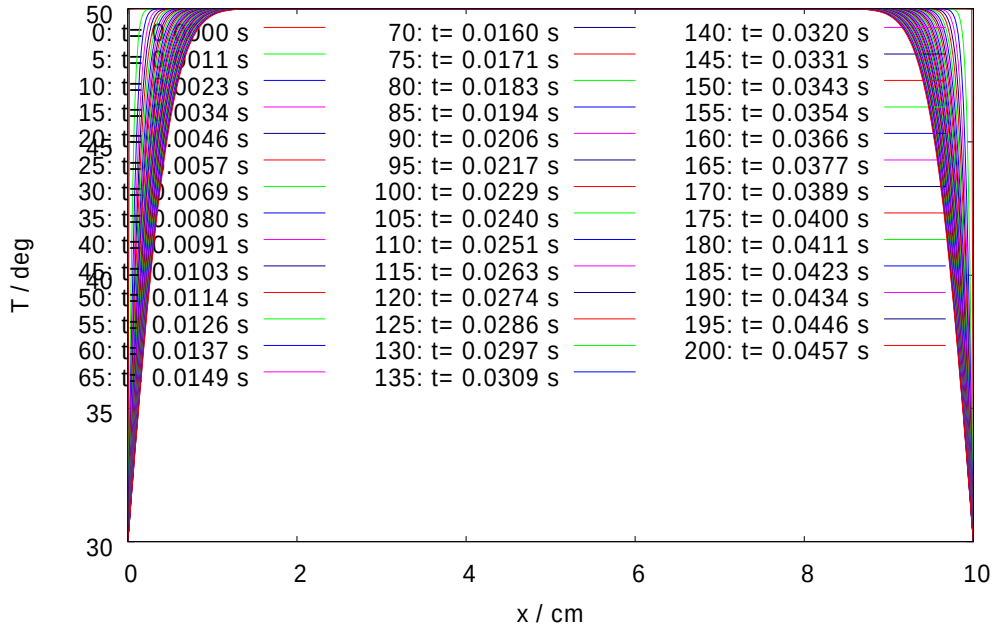
能である。ここでは，陽解法のプログラムを少し改造だけすることにする。基本は何も変わらず,行列が異なることとベクトル Tn と Tp の間の演算が Tn=C*Tp; の積からから Tn=C¥Tp;の左除算に変わるだけである。

imdiff.m

```
% Octave script m file
% solve one dimension  diffusion partial differential eq. by implicit method
% main script
hold off;subplot(111);clg;clear all;
%
% Constants for Calculation
D=1.75; % cm^2/s diffusion constant
length=10; % cm
N=500; % Divsion Number of the length
lambda=1;
op2lambda=1+2*lambda;
TR=200; % Time Range
Tint=1; % Time interval
Dint=Tint*2; % Time interval for final graph
SaveFname="Tsave.txt"; % File name to save data
fprintf("Simulation of Temperature distribution in one dimensional thermal
diffusion¥n");
fprintf("  Thermal Diffusion Constant : %7.4f cm^2/s ¥n", D);
fprintf("  Length : %7.4f cm¥n",length);
fprintf("  Number of divided of x range : %d ¥n",N);
fprintf("  Value of parameter lambda : %7.4f ¥n",lambda);
fprintf("  Number of time iteration : %d ¥n", TR);
fprintf("  File name to save data : %s¥n", SaveFname);
%
N1=N+1;
TR1=TR+1;
%
DeltaX=length/N;
Deltat=lambda*DeltaX^2/D;
X=[0:DeltaX:length]';
%
% construct a sparse Matrix
dia=[1 (ones(1,N-1)*op2lambda) 1];
Cd=sparse(1:N+1,1:N+1,dia,N+1,N+1);
Cu=sparse(2:N,3:N+1,-lambda,N+1,N+1);
Cl=sparse(2:N,1:N-1,-lambda,N+1,N+1);
C=Cd+Cu+Cl;
% Initial and boundary condition
Tp=[30 (ones(1,N-1)*50) 30]';  %Initial condition
Tb1=ones(TR1+1,1)*30;
Tb2=Tb1;
%
Tsave = fopen (SaveFname, "w");
% solve PDE by iteration
```

```
for m=1:TR1
  if (mod((m-1),Tint) == 0)
   plot(X,Tp, sprintf(";%d: t=%7.4f s;",(m-1),(m-1)*Deltat));
   xlabel("x / cm¥n");  ylabel("T / deg¥n"); axis([0 10 30 50]);
    drawnow;  % required for Octave 2.9.xxx
    fprintf(Tsave, "%e ",Tp); fputs(Tsave,"¥n");
  end
  Tn=C¥Tp;
  Tp=Tn;
end
fclose(Tsave);
input("Press any key to continue");
Tread = fopen (SaveFname, "r");
Tx=zeros(1,N1);
clg;
xlabel("x / cm¥n");  ylabel("T / deg¥n");  axis([0 10 30 50]);
hold on
for m=1:Tint:TR1
   sTx=[ "[" fgets(Tread) "]" ];
   Tx=eval(sTx);
  if (mod((m-1),Dint) == 0)
    clr=mod(floor((m-1)/Dint),5)+1;
   ;
    plot(X,Tx', sprintf("%d;%d: t=%7.4f s;",clr,(m-1),(m-1)*Deltat));
  end
end
fclose(Tread);
```

陰解法の利点を生かし，刻み幅を小さく取り短時間でのシミュレーションをしている。



刻み幅や時間間隔などを自由に取れるので各自でいろいろと条件を変えて試してほしい。

その他, 境界条件として端点での温度勾配が与えられるような第２種の境界条件や, 境界条件が時間変化する場合など興味深い応用例があるが，時間の都合で割愛する。

おわりに

私が初めて Octave に触れたのは，2004 年の夏ごろである。ある物質の上に別の硬さの層状物質がのっている場合，そこで伝わる表面波の速度を計算する必要が生じた。計算式は複雑で，行列計算と複素数の計算を伴った問題であった。大変な計算になると思い，その当時の最新の数値計算の事情を調べていて，フリーの Octave というツールを知った。そのプログラムのプロトコルを Octave で作成し，Octave で解くことが可能であることがわかった。しかし，その問題は，複雑なので計算時間がかかりすぎるということで，Octave そのものをビルドするときに作成される liboctave という C++のクラスライブラリ（よくわからないと思うが C++上で動く拡張パッケージと思ってくれればいい。）があることを知り，Octave で作成したプログラムを liboctave を組み込んだ C++でプログラムで書き直した。基本的には似た構文でかけるので，移植は比較的簡単であった。一からそのプログラムを組んでいたらうまくいったどうかを考えたときやはり Octave のおかげであると思った。解析は，うまくいき，それを用いて乾燥中のゲルの表面を伝播する表面波の解析をおこなって，実験結果の解釈を行い，論文を書いた。以来，Octave は私の研究生活に欠かせないものとなった。

そのころ，新しいカリキュラムを組むことになり４年前期のアドバンスなコンピュータ利用の選択科目を開設しようということになった。わたしは，いろいろと悩んだすえ，Octave を使った数値計算入門をやってはと思い，本講義の準備を進めた。その過程の中で，この演習で何度も使った「化学工学プログラミング演習」に出会い，この内容を Octave で書き直せば，その当時は最先端であった化学工学の計算問題がごく普通の問題として現代では解けるということが示せると確信した。

今回，自転車操業で資料を作ってきたので完成度とかの面で問題があったと思うが，考え方自体は，間違っていなかったと思う。「はじめに」でのべたが，２１世紀になった今，数値計算のプロが格闘してきた成果がフリーウェアとして登場し，少しの努力をすれば使えるツールとして存在するのである。２１世紀を生きるエンジニアとなる現在の学生諸君は，このような新しい武器を手に新たな問題に挑んでほしい。そのために，本講義資料が，少しでもお役に立てば幸いである。

なお，本講義資料は Octave の機能の一部を紹介したに過ぎない。Octave には，化学工学では重要な制御論のパッケージや線形計画法のパッケージなども含まれている。また，線形代数では重要な固有値問題などは本講義では取り扱わなかった。その点は，各自 Octave のマニュアルを読んで勉強してほしい。マニュアルは，英語であるが，この程度の英語を読めて，ツールとして使えることも２１世紀を生きるエンジニアには要求されていることである。また，近いうちに Octave は Ver. 3 になることがアナウンスされ，プレリリースとして Ver.2.9 シリーズが公式リリースとなった，MATLAB との互換性がより高まり，さまざまな機能が強化されている。また，Octave は，UNIX をベースにしたソフトだが Microsoft の C++でコンパイルされた，Windows Native の Octave も登場しており，Windows ユーザーでもさらに使いやすくなることであろう。

Octave が皆さんの研究に役立つことを祈念して。

2007 年 7 月 13 日
松岡　辰郎