V. 数値計算による偏微分方程式の解：陰解法と定常解

１２．熱伝導方程式の陰解法

ここでは陰解法の解き方や数値的安定性について学ぶ。

12.1 時間差分の取り方

生成・消滅方程式

$$\frac{d\phi}{dt} = -\lambda\phi \tag{12.1}$$

を考える。今，時間微分を前進差分で近似すると，右辺の ϕ は n ステップ目の値となり，

$$\frac{\phi^{n+1}-\phi^{n}}{\Delta t} = -\lambda\phi^{n} + O\left(\Delta t\right) \tag{12.2}$$

と書ける。右辺の $O(\Delta t)$は無視した項の大きさ(Δt の 1 乗のオーダー)である。未来の値を左辺，過去の値を右辺に集めると

$$\phi^{n+1} = \phi^{n} - \lambda\Delta t\phi^{n} + O\left(\Delta t^{2}\right) = \left(1-\lambda\Delta t\right)\phi^{n} + O\left(\Delta t^{2}\right) \tag{12.3}$$

となる。

一方後退差分に取ると

$$\frac{\phi^{n+1}-\phi^{n}}{\Delta t} = -\lambda\phi^{n+1} \tag{12.4}$$

であり，また，(12.2)と同様 1 次の誤差を持つ。未来の値を左辺，過去の値を右辺に集めると

$$\phi^{n+1} + \lambda\Delta t\phi^{n+1} = \phi^{n} \tag{12.5}$$

から，

$$\phi^{n+1} = \phi^{n} / \left(1+\lambda\Delta t\right) \tag{12.6}$$

となる。この式を Δt が微小としてテイラー展開すると(12.3)に一致する。

12.2 解の安定性と物理的に正しい解

(12.1)の解は時間が無限に経つと 0 に収束する。このような解を持つためには，(12.3)より，

$$\left|1-\lambda\Delta t\right| \le 1 \tag{12.7}$$

すなわち，

$$\Delta t \le 2/\lambda \tag{12.8}$$

でなければならない。解に何らかの誤差が混入した場合，誤差は未来へ(12.3)に従って伝播する。つまり，(12.8)の条件を満たしていれば，誤差は最終的に 0 に収束することになる。この場合には，数値解を安定して求めることが出来る。最終的に誤差が 0 に収束する条件を**数値安定性の条件** (condition for numerical stability)と呼ぶ。時間差分を前進差分に取る場合，安定して解が求められるために条件がつく。

ここで，ϕ が最初に正の場合，ϕ は常に正の値しか取らない。(12.3)の場合には，Δt を大きく取ると ϕ が負の値となってしまうことに注意しよう。物理的に正しく，ϕ が常に正の値を取るためには，

$$1-\lambda\Delta t \ge 0 \tag{12.9}$$

すなわち，

$$\Delta t \le 1/\lambda \tag{12.10}$$

でなくてはならない (ただし，物理的に正しい解が精度の良い解を必ずしも意味しないことには注意すべきである)。このことは，数値安定性の条件を満たしているからと言って，解が必ずしも物理的に正しくないことを意味している。

一方，後退差分に取った場合はどうだろう。この場合には，常に

$$0 \le 1/\left(1+\lambda\Delta t\right) \le 1 \tag{12.11}$$

であり，ϕ は常に正となる。つまり，無条件に安定であると同時に，物理的に正しい解を持つことが分かる。

また，右辺を現在と未来の２つの時間の平均で近似すると，

$$\frac{\phi^{n+1}-\phi^n}{\Delta t} = -\lambda\frac{\phi^{n+1}+\phi^n}{2} + O\left(\Delta t^2\right) \tag{12.12}$$

となる。このような方法を**クランク・ニコルソン法** (Clank-Nicholson method)とよぶ。この近似は，中心差分と同じ２次の精度を持ち，前進差分・後退差分より精度の高い解法が得られる。

$$\phi^{n+1} = \frac{1-\lambda\Delta t/2}{1+\lambda\Delta t/2}\phi^n \tag{12.13}$$

が得られる。また，係数は常に

$$\left|\frac{1-\lambda\Delta t/2}{1+\lambda\Delta t/2}\right| \le 1 \tag{12.14}$$

を満たすので，無条件安定である。しかし，物理的な正の値を持つためには，

$$1-\lambda\Delta t/2 \ge 0 \tag{12.15}$$

を満たさねばならない。これは前進差分の場合の安定性の条件と同じである。

12.3 一次元熱伝導の陰解法と連立一次方程式

1 次元の熱伝導方程式

$$\frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial x^2} \tag{12.21}$$

を差分法により離散化する。時間微分を前進差分にとると，

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \kappa \frac{T_{i-1}^n + T_{i+1}^n - 2T_i^n}{\Delta x^2} \tag{12.22}$$

と書ける。この差分方程式は時間微分にΔt1 次，空間微分にΔx 2 次の誤差を持つ。この場合，未来の値のみを左辺に書くことができる。

$$T_i^{n+1} = T_i^n + \frac{\Delta t \kappa}{\Delta x^2}\left(T_{i-1}^n + T_{i+1}^n - 2T_i^n\right) \tag{12.22}$$

このため，未来の値 T_i^{n+1}をあからさまに計算することができる。このような解法は，**陽解法** (explicit method)と呼ばれる。陽解法が物理的に正しい解を与える条件は，

$$T_i^n \le \frac{T_{i+1}^n + T_{i-1}^n}{2} \tag{12.23}$$

のとき， 1 つ未来の時間でも

$$T_i^{n+1} \le \frac{T_{i+1}^n + T_{i-1}^n}{2} \tag{12.24}$$

となることである。これは，中間の温度が両隣の温度の平均よりも低いときには，未来の時間においてもその温度より低いままであるはず，という条件である。(12.22)を(12.24)の左辺に代入すると，

$$T_i^n + \Delta t \kappa \frac{T_{i+1}^n + T_{i-1}^n - 2T_i^n}{\Delta x^2} \le \frac{T_{i+1}^n + T_{i-1}^n}{2} \tag{12.25}$$

より，

$$\frac{\Delta t \kappa}{\Delta x^2}\left(T_{i+1}^n + T_{i-1}^n - 2T_i^n\right) \le \frac{1}{2}\left(T_{i+1}^n + T_{i-1}^n - 2T_i^n\right) \tag{12.26}$$

すなわち，

$$\Delta t \le \frac{\Delta x^2}{2\kappa} \tag{12.27}$$

が得られる。この値は(12.10)と同様，数値的安定性の条件(9.27)の 1/2 となっている。(9.27)は，中間の温度は両隣の温度の周りで振動(オーバーシュート)しながらも，最終的には平均に収束していくという条件である。**問題 IV-1** のプログラムでは，条件(12.27)

を用いて Δt を決めている。

ここで，時間微分を後退差分にとってみよう。このときは，

$$\frac{T_i^{n'} - T_i^{n'-1}}{\Delta t} = \kappa \frac{T_{i-1}^{n'} + T_{i+1}^{n'} - 2T_i^{n'}}{\Delta x^2} \tag{12.28}$$

と書けるが，n' を $n+1$ に置き換えると，

$$\frac{T_i^{n+1} - T_i^{n}}{\Delta t} = \kappa \frac{T_{i-1}^{n+1} + T_{i+1}^{n+1} - 2T_i^{n+1}}{\Delta x^2} \tag{12.29}$$

となる。この式は，右辺に未知の未来の値が入っているから，そのままでは解くことができない。このような方法は**陰解法** (implicit method)とよばれる。この方法は無条件安定である。クランク・ニコルソン法も陰解法の 1 つと考えることができる。未知数を左辺に移項すると，

$$\frac{\Delta t\kappa}{\Delta x^2} T_{i-1}^{n+1} - \left(2\frac{\Delta t\kappa}{\Delta x^2} + 1 \right) T_i^{n+1} + \frac{\Delta t\kappa}{\Delta x^2} T_{i+1}^{n+1} = T_i^{n} \tag{12.30}$$

となる。この式は，$i = 0, nz$ に境界があるとすると，$nz-$ 1 個の温度 T を未知数とする連立一次方程式となっている。両隣だけが，0 でない係数を持っている連立方程式である。つまり，左辺を省略しないで書くと

$$LHS = 0T_1^{n+1} + 0T_2^{n+1} + ... + \frac{\Delta t\kappa}{\Delta x^2} T_{i-1}^{n+1} - 2\left(\frac{\Delta t\kappa}{\Delta x^2} + 1 \right) T_i^{n+1} + \frac{\Delta t\kappa}{\Delta x^2} T_{i+1}^{n+1} +$$

$$... + 0T_{nz-1}^{n+1} \tag{12.31}$$

となっている。連立一次方程式を行列の形

$$\boldsymbol{Ax} = \boldsymbol{b} \tag{12.32}$$

で書くと，係数行列 A は，

$$
\begin{bmatrix}
a_1^P & a_1^D & 0 & \cdots & \cdot & & & & & & \cdots & 0 \\
a_2^U & a_2^P & a_2^D & 0 & \cdots & \cdot & & & & & \cdots & 0 \\
0 & a_3^U & a_3^P & a_3^D & 0 & & & & & & & \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & & & & & & \vdots \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & & & & & \cdot \\
 & & & & & & & & & & & \\
0 & \cdots & \cdot & 0 & a_i^U & a_i^P & a_i^D & 0 & \cdots & \cdot & \cdots & 0 \\
0 & \cdots & \cdot & & 0 & a_{i+1}^U & a_{i+1}^P & a_{i+1}^D & 0 & \cdots & \cdots & 0 \\
\vdots & & & & & \ddots & \ddots & \ddots & \ddots & & & \vdots \\
\cdot & & & & & & \cdot & \cdot & \cdot & \cdot & & \cdot \\
 & & & & & & & & & & & \\
\vdots & & & & & & & & \cdots & 0 & a_{m-1}^U & a_{m-1}^P & a_{m-1}^D \\
0 & \cdots & \cdot & & & & & & \cdots & 0 & a_m^U & a_m^P
\end{bmatrix}
$$

$$\cdots \quad (12.33)$$

となっている。この行列は，対角要素とその両隣だけが 0 でない値を持つ。このような行列は**三重対角行列** (tridiagonal matrix)と呼ばれる。ただし，係数 a_i^U，a_i^P，a_i^D はそれぞれ，T_{i-1}^{n+1}，T_i^{n+1}, T_{i+1}^{n+1} の係数である。すなわち，

$$a_i^U = \frac{\Delta t \kappa}{\Delta x^2} \quad (12.34)$$

$$a_i^P = -\left(2\frac{\Delta t \kappa}{\Delta x^2} + 1 \right) \quad (12.35)$$

$$a_i^D = \frac{\Delta t \kappa}{\Delta x^2} \quad (12.36)$$

である。ここで，格子点数 $nz - 1$ を m とした。右辺のベクトルは，

$$b_i = T_i^n \quad (12.37)$$

である。ここで，行列の対称性に注意しよう。熱伝導方程式のように偶数階の空間微分しか持たない方程式を差分化したときには，係数行列は対称行列となっている。これは式中の x を $-x$ で置き換えても同じ式になることと対応している。T_i^{n+1} を求めるのに三重対角行列を持つ連立一次方程式を解かなければならない。多くの場合，ガウスの消去法が用いられる。

問題 V-1

(1) 熱伝導率一定である 1 次元熱伝導方程式を陰解法による差分方程式が対称な係数

行列を持つことを示せ。

(2) 熱伝導率が変化する場合の熱伝導方程式

$$\frac{\partial T}{\partial t}=\frac{\partial}{\partial x}\left(\kappa\frac{\partial T}{\partial x}\right)$$

の有限体積法に基づいて離散化した式

$$\frac{T_i^{n+1}-T_i^n}{\Delta t}=\frac{1}{\Delta x}\left(\kappa_{i+1/2}\frac{T_{i+1}^{n+1}-T_i^{n+1}}{\Delta x}-\kappa_{i-1/2}\frac{T_i^{n+1}-T_{i-1}^{n+1}}{\Delta x}\right)$$

の係数行列の成分 a_i^U，a_i^P，a_i^D を求めよ。また，係数行列が対称となるか確かめよ。

12.4 境界条件の扱い

境界条件は，次のように扱う。境界条件を差分方程式に取り込んだ連立一次方程式を作る。今，格子点数を $nz-1$ (m)個とすると，格子点番号 i は境界の格子点も合わせると 0 から m+1 までである。ここで，$i=0$ と $i=m+1$ は境界上の値である。温度などの値そのものが与えられる境界条件(ディリクレ境界条件)の場合には，

$$a_1^U T_0^{n+1}+a_1^P T_1^{n+1}+a_1^D T_2^{n+1}=T_1^n \tag{12.41}$$

より，

$$a_1^P T_1^{n+1}+a_1^D T_2^{n+1}=T_1^n-a_1^U T_0^{n+1} \tag{12.42}$$

となる。一方，熱流量などの微分値が与えられる境界条件(ノイマン型境界条件)の時は，境界での微分値を

$$G=-\frac{q}{k}=\frac{T_{m+1}^{n+1}-T_m^{n+1}}{\Delta z} \tag{12.43}$$

とする。この式から，境界の値は，

$$G=\frac{q}{k}=\frac{T_{m+1}^{n+1}-T_m^{n+1}}{\Delta x} \tag{12.44}$$

と表される。これを i=m (nz-1)の差分方程式に

$$a_m^U T_{m-1}^{n+1}+a_m^P T_m^{n+1}+a_1^D T_{m+1}^{n+1}=T_m^n \tag{12.45}$$

に代入すると，

$$a_m^U T_{m-1}^{n+1}+a_m^P T_m^{n+1}+a_1^D\left(T_m^{n+1}+G\Delta x\right)=T_m^n \tag{12.46}$$

より，

$$a_m^U T_{m-1}^{n+1}+\left(a_m^P+a_1^D\right)T_m^{n+1}=T_m^n-a_1^D G\Delta x \tag{12.47}$$

が得られる。具体的には

$$\frac{\Delta t\kappa}{\Delta x^2}T_{m-1}^{n+1}-\left(\frac{\Delta t\kappa}{\Delta x^2}+1\right)T_m^{n+1}=T_m^n-\frac{\Delta t\kappa}{\Delta x}G=T_m^n-\frac{\Delta t\kappa}{\Delta x}\frac{q}{k} \tag{12.48}$$

である。

12.5 2次元および3次元の場合

2次元以上の変数は，普通2つまたは3つの添え字を用いて表す。例えば，

$T_{i,j}$（T_{ij}）

$T_{i,j,k}$（T_{ijk}）

などである。熱伝導方程式のように空間2階微分を持つ場合，2次元の差分方程式は

$$\cdots+a_{i,j}^{S}T_{i,j-1}+\cdots+a_{i,j}^{W}T_{i-1,j}+a_{i,j}^{P}T_{i,j}+a_{i,j}^{E}T_{i+1,j}+\cdots+a_{i,j}^{N}T_{i,j+1}+\cdots=b_{i,j}$$

…(12.49)

のように書ける。ここで，上付き添え字 $EWNS$ は(i, j)から見て東西南北の方向を表す。

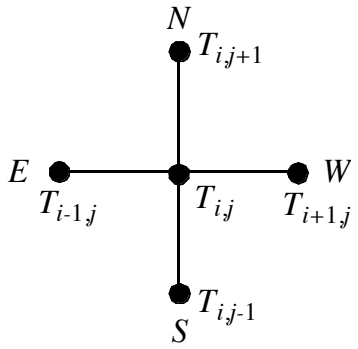

図 12.1 2次元のラプラシアンにおける格子点配置

5つの点がすべて境界を含まない場合, 陰解法による熱伝導方程式の差分方程式において係数はそれぞれ,

$$a_{i,j}^{S}=a_{i,j}^{W}=a_{i,j}^{E}=a_{i,j}^{N}=\frac{\Delta t\kappa}{\Delta x^2} \tag{12.50}$$

$$a_{i,j}^{P}=-\left(a_{i,j}^{S}+a_{i,j}^{W}+a_{i,j}^{E}+a_{i,j}^{N}+1\right)=-\left(4\frac{\Delta t\kappa}{\Delta x^2}+1\right) \tag{12.51}$$

である。ただし，

$$\Delta x=\Delta z=const. \tag{12.52}$$

としている。

このままだと, 差分方程式を連立一次方程式と対応させにくいかもしれない。そこで, ２次元の添え字を１次元の添え字に置き換えて考える。境界条件を与える格子を除いた内部の格子点が $m = r \times s$ 個(x 方向×y 方向)である場合,

$$k = r(j-1)+i \tag{12.53}$$

のように置き換える。そうすると,

$$\cdots + a_k^S T_{k-r} + \cdots + a_k^W T_{k-1} + a_k^P T_k + a_k^E T_{k+1} + \cdots + a_k^N T_{i+r} + \cdots = b_k \tag{12.55}$$

のようになって連立一次方程式らしく見えるようになる。これは k 行目の式である。係数行列だけ書き出すと,

$$\begin{bmatrix}
a_1^P & a_1^E & 0 & \cdots & & a_1^N & 0 & \cdots & \cdot & & & & & & & & \cdots & 0 \\
a_2^W & a_2^P & a_2^E & 0 & \cdots & & a_2^N & & & & & & & & & & & \vdots \\
0 & \ddots & \ddots & \ddots & & & & \ddots & & & & & & & & & & \cdot \\
\vdots & & \cdot & \cdot & \cdot & & & & \cdot & & & & & & & & & \\
0 & \cdots & 0 & a_r^W & a_r^P & 0 & 0 & \cdots & \cdot & a_r^N & & & & & & & & \\
a_{r+1}^S & \cdot & \cdots & 0 & 0 & a_{r+1}^P & a_{r+1}^E & 0 & \cdots & \cdot & a_{r+1}^N & & & & & & & \\
0 & \ddots & & & & & \ddots & \ddots & & & & \ddots & & & & & & \\
\vdots & & a_{2r+1}^S & \cdot & \cdots & 0 & 0 & a_{2r+1}^P & a_{2r+1}^E & 0 & \cdots & & a_{2r+1}^N & & & & & \\
\cdot & & & \ddots & & & & \ddots & \ddots & \ddots & & & & \ddots & & & & \\
 & & & & \cdot & & & & \cdot & \cdot & \cdot & & & & \cdot & & & \\
 & & & & & a_k^S & \cdot & \cdots & 0 & a_k^W & a_k^P & a_k^E & 0 & \cdots & \cdot & a_k^N & & \\
 & & & & & & \ddots & & & & \ddots & \ddots & \ddots & & & & \ddots & \\
 & & & & & & & \cdot & & & & \cdot & \cdot & & & & \cdot & \vdots \\
 & & & & & & & & \cdot & & & \cdot & \cdot & \cdot & & & \cdot & 0 \\
 & & & & & & & & & a_{m-r}^S & \cdot & \cdots & 0 & a_{m-r}^W & a_{m-r}^P & 0 & 0 & \cdots & 0 & a_{m-r}^N \\
 & & & & & & & & & & \cdot & & & & \cdot & \cdot & \cdot & & & \cdot \\
 & & & & & & & & & & & \cdot & & & & \cdot & \cdot & \cdot & & \vdots \\
 & & & & & & & & & & & & \ddots & & & & \ddots & \ddots & \ddots & 0 \\
\vdots & & & & & & & & & & & & & a_{m-1}^W & \cdot & \cdots & 0 & a_{m-1}^W & a_{m-1}^P & a_{m-1}^E \\
0 & \cdots & \cdot & & & & & & & & & \cdots & 0 & a_m^S & \cdot & \cdots & 0 & a_m^W & a_m^P &
\end{bmatrix} \tag{12.56}$$

となっている。行列は $m \times m$ の正方行列であるが, 対角要素から r だけ離れたところまでが, 0 でない値を持っている。このような行列は**帯行列** (band matrix)と呼ばれる。ただし, 0 でない値を持つのは, 帯の中でも対角要素と両隣, r だけ離れた列, の５つだけである。計算機プログラムの配列変数に格納にする場合は, $(r+1) \times m$ の２次元配列に係数行列を格納する。

12.6 定常状態の場合・楕円型方程式

熱伝導方程式において定常状態の場合は,

$$\frac{\partial T}{\partial t} = 0 \tag{12.61}$$

とすれば良い。この場合，方程式は楕円型となり，ポワソン方程式に一致する。2次元の場合の差分方程式は，

$$0 = \kappa \frac{T_{i,j-1} + T_{i-1,j} + T_{i+1,j} + T_{i,j+1} - 4T_{i,j}}{\Delta x^2} + \frac{H_{i,j}}{C_p} \tag{12.62}$$

となる。係数は

$$a_{i,j}^S = a_{i,j}^W = a_{i,j}^E = a_{i,j}^N = \frac{\kappa}{\Delta x^2} \tag{12.63}$$

$$a_{i,j}^P = -\left(a_{i,j}^S + a_{i,j}^W + a_{i,j}^E + a_{i,j}^N\right) = -4\frac{\kappa}{\Delta x^2} \tag{12.64}$$

となる。

１３．連立一次方程式の解法：直接法

連立方程式の解法には，元数によって決まっている計算手順の回数で厳密に解が得られる**直接法** (direct method)と，同じ操作を繰り返し収束したらそこで計算をやめる**反復法** (iterative method)と２通りの方法がある。ここでは，直接法から紹介する。

13.1 ガウスの消去法

連立一次方程式を解くためのもっとも一般的な方法は**ガウスの消去法** (Gaussian elimination method)である。この方法は単純な操作の繰り返しであるので，コンピュータ向きの方法であり，数値計算でもよく用いられ，とくに密行列（要素がゼロであるところが少ない行列）に向く方法である。この方法は直接法の１つである。

(1) 前進消去

$k = 1$ から m - 1 まで以下の処理を行う

第 k 行の $a_{k\,k+1}$ から $a_{k\,m}$ までを $a_{k\,k}$ で割る（消去済みのところはすでに０）

...[1]

右辺ベクトル b_k を $a_{k\,k}$ で割る

$i = k + 1$ から m まで以下の処理を行う

第 k 行の (上記の新) $a_{i\,k}$ 倍を第 i 行の $a_{i\,k+1}\cdots a_{i\,m}$ から引く

(上記の新) b_k の $a_{i\,k}$ 倍を b_i から引く

この前進消去により行列Aは左下半分のすべての要素が０となっている。

(2) 後退代入

$x_m = b_n / a_{mm}$

$k = m - 1$ から 1 まで以下の処理を行う。

$$x_k = b_k - \sum_{i=k+1}^{m} a_{ki} x_i$$

前進消去のときにピボット $a_{k\,k}$ が 0 だと割ることができないし，0 に近い値となると数値計算の精度が非常に悪くなる。このため，[1]の処理を行う前に，行交換を行い，ピボットを最大にする操作を行う。つまり，$a_{k\,k}$, $a_{k+1\,k}$, ..., $a_{m\,k}$ の中から絶対値最大のものを選び，その行を第 k 行に持ってくる。これを部分ピボット選択という。

また，前進消去のときの左辺の行列への操作と，右辺ベクトルへの操作は独立しているので，別々に行うことも可能である。ふつうのガウスの消去法のプログラムはそのように書かれているものが多い。つまり，

左辺 A に対して，

$k = 1$ から $m - 1$ まで以下の処理を行う

　第 k 行の $a_{k\,k+1}$ から $a_{k\,m}$ までを $a_{k\,k}$ で割る（消去済みのところはすでに 0）

　$i = k + 1$ から m まで以下の処理を行う

　　第 k 行の (上記の新) $a_{i\,k}$ 倍を第 i 行の $a_{i\,k+1} \cdots a_{i\,m\,n}$ から引く

右辺 $\boldsymbol{b}$ に対して，

$k = 1$ から $m - 1$ まで以下の処理を行う

　b_k を $a_{k\,k}$ で割る

　$i = k + 1$ から m まで以下の処理を行う

　(上記の新) b_k の $a_{i\,k}$ 倍を b_i から引く

とする。これは，行列 A を下三角行列 L と上三角行列 U に分解

$$Ax = \boldsymbol{b} \quad \rightarrow \quad LUx = \boldsymbol{b}$$

としたあと，前進消去

$$Ux = L^{-1}\boldsymbol{b}$$

と後退代入

$$x = U^{-1}L^{-1}\boldsymbol{b}$$

を行っていることと同じである。このようなガウスの消去法のプログラムの例を**問題**

V-II に示す。

13.2 対称行列の場合

この場合には行列の対称性を利用して計算回数を半分に減らすことが可能である。このような方法の代表的なものに修正コレスキー分解法がある。修正コレスキー分解法では行列 A を

$$A = LDL^T$$

のように分解する。この分解を行った後に，右辺ベクトルの前進・後退代入

$$x = (DL^T)^{-1}L^{-1}\boldsymbol{b}$$

により解を求める。

13.3 特殊な係数行列を持つ連立一次方程式の解法

拡散方程式（定常解や陰解法）やポワソン(Poisson)方程式を差分化したときに得られる特別な形の連立一次方程式を解く方法として次のような方法がある。

一次元の熱伝導方程式の場合

行列 A が対角要素とその両隣だけがゼロでない値を持つ三重対角行列になる。要素数が少なく，計算量，メモリー容量ともに少ないのでガウスの消去法が用いられる。

二次元の熱伝導方程式の場合

行列 A は帯行列になる。反復法が多くの場合用いられる。大きなメモリー容量が必要だが，直接法も十分可能である。境界条件だけが変わる場合のような，同じ行列に対して右辺だけを変えて繰り返し計算を行うときには有利になる。

二次元の熱伝導方程式の場合

行列 A は帯行列になる。計算量とメモリー容量の関係から反復法が用いられる。

13.4 ガウスの消去法による三重対角行列の解法

三重対角行列の構造を利用すると効率の良いガウスの消去法が得られる

前進消去

左辺 A に対して，

$k = 1$ から m - 1 まで以下の処理を行う

$a_{k\,k+1}$ を $a_{k\,k}$ で割る

$a_{k\,k+1}$ の（上記の新）$a_{k+1\,k-1}$ 倍を $a_{k+1\,k+1}$ から引き，新しい $a_{k+1\,k+1}$ とする

右辺 $\boldsymbol{b}$ に対して，

k =1 から m - 1 まで以下の処理を行う

b_k を $a_{k\,k}$ で割る

（上記の新）b_k の $a_{k+1\,k}$ 倍を b_{k+1} から引き，新しい b_{k+1} とする

後退代入

$x_m = b_m / a_{mm}$

$k = m$ - 1 から 1 まで以下の処理を行う。

$x_k = b_k - a_{k\,k+1} x_{k+1}$

行列 A の要素はほとんど 0 なので

$a_{k\,k-1}$ → a(k,1)

$a_{k\,k}$ → a(k,2)

$a_{k\,k+1}$ → a(k,3)

のように配列にいれると記憶容量を節約できる。対称行列の場合は

$a_{k\,k-1}$ → a(k,1)

$a_{k\,k}$ → a(k,2)

だけでよい。このようなやりかたで行列を扱う三重対角行列用のガウスの消去法のプログラム例を

http://home.hiroshima-u.ac.jp/nakakuki/others/Toys.html

に示す。

13.5 帯対称行列に対する修正コレスキー法

２次元以上の熱伝導の差分方程式の係数行列は前に見たように, 帯対称行列となっている。帯行列は対角要素に近いある幅の範囲だけが 0 でない要素を持ち，それ以外は 0 である行列である。つまり 0 でない要素は斜めの帯状に分布する。この構造を利用すると，０でない要素がある範囲(帯幅という)だけを配列に記憶し，そこだけ計算すればよい。

k = 1 から n まで

a^S_k → a(1,k)

a^W_k → a(r,k)

a^P_k → a(r+1,k)

のように配列に入れる。それ以外の a(i,j)には 0 を入れておく。ただし，下の境界と接する部分，つまり，i=1～r の下三角行列が左にはみ出す部分(a^W_1 等)は 0 を代入する。

またこのとき，境界条件から生じる項は上三角行列の時と同様右辺に移項する。

問題 V-2

２次元のラプラス方程式

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial z^2} = 0$$

を考える。下のような矩形領域を 4×4 に分割した格子を設定し，内部の 2×2 点の値を求めたい。外周の格子点にはディリクレ型境界条件で，ϕの値が図に書かれた数字のように与えられている。ここで，格子の間隔は$\Delta x=\Delta z=1$ とする。

(1) 内部の４つの格子点における差分方程式を書き，連立一次方程式となることを確かめよ。

(2) 連立一次方程式をガウスの消去法を用いて解き，与えられた境界条件でのラプラス方程式の解を求めよ。手計算と添付のプログラム(**問題 V-2 プログラム**)を用いる計算と２通りの方法で計算せよ。

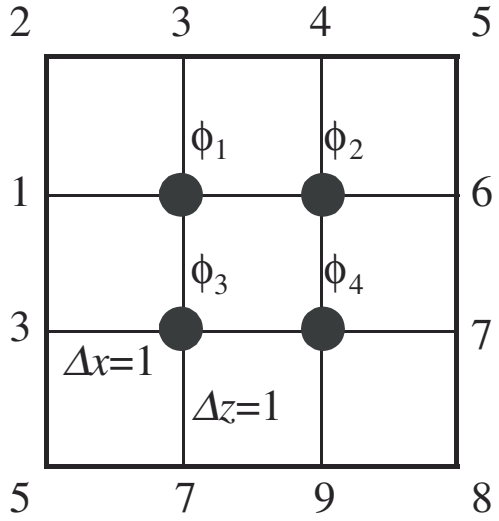

図 13.1 問題の格子点配置

問題の出展：戸川隼人著「数値解析とシミュレーション」共立全書

１４．反復法による楕円型方程式の数値解法

３次元の問題のように大規模な問題では，ガウスの消去法では計算量が多くなりすぎて解けないことが多くなる。この場合，反復法が用いられる。反復法には，ここで紹介する線形反復法と，共役勾配法のようなベクトルの直交性に注目した方法と２通りの方法がある。

14.1 緩和法によるポワソン方程式の解法

ここでは話を簡単にするためにポワソン方程式

$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial z^2}=\rho \tag{14.1}$$

の差分方程式

$$\frac{\phi_{i,j-1}+\phi_{i-1,j}+\phi_{i+1,j}+\phi_{i,j+1}-4\phi_{i,j}}{\Delta x^2}=\rho_{i,j} \tag{14.2}$$

を解くことを考える。簡単化のために$\Delta x=\Delta y$とした。この式を変形すると

$$\phi_{i,j}=\left(\phi_{i,j-1}+\phi_{i-1,j}+\phi_{i+1,j}+\phi_{i,j+1}+\Delta x^2\rho_{i,j}\right)/4 \tag{14.3}$$

となる。この式に単純反復法を適用する。この様な方法を**線形反復法**あるいは**緩和法** (relaxation method)と呼ばれる。$(i,j)=(1,1)$から計算を始めて，
$(i,j)=(m,n)$まで計算をして反復１回が終わりである。つまり，

$$\phi_{i,j}^{n+1}=\left(\phi_{i,j-1}^{n}+\phi_{i-1,j}^{n}+\phi_{i+1,j}^{n}+\phi_{i,j+1}^{n}+\Delta x^2\rho_{i,j}\right)/4 \tag{14.4}$$

のように反復計算する。この緩和法を**ヤコビ法** (Jacobi method)という。

このとき，$\phi_{i,\,j-1}$と$\phi_{i-1,\,j}$は$\phi_{i,\,j}$を計算するときにはすでに計算済みの点であるので，$\phi_{i-1,\,j}$と$\phi_{i-1,\,j}$に更新されたばかりの値を使うと収束が速くなるし，$\phi_{i,j}$を記憶する配列が一つですむ。このやり方を**ガウス・ザイデル法** (Gauss-Seidel method)という。つまり，

$$\phi_{i,j}^{n+1}=\left(\phi_{i,j-1}^{n+1}+\phi_{i-1,j}^{n+1}+\phi_{i+1,j}^{n}+\phi_{i,j+1}^{n}+\Delta x^2\rho_{i,j}\right)/4 \tag{14.5}$$

のように反復する。プログラム中では変数をスイープするときに上書きするとガウス-ザイデル法となり，上書きしないように書くとヤコビ法である。ヤコビ法では並列計算に向いているのに対し，ガウス・ザイデル法は直前に計算した値への依存性があるので，そのままでは並列計算に向いていない。並列計算には Red-Black 法などが必要になる。

上の式は変形すると

$$\phi_{i,j}^{n+1} = \phi_{i,j}^{n} + \left(\phi_{i,j-1}^{n+1} + \phi_{i-1,j}^{n+1} + \phi_{i+1,j}^{n} + \phi_{i,j+1}^{n} - 4\phi_{i,j}^{n} + \Delta x^2 \rho_{i,j}\right)/4 \qquad (14.6)$$

である。

SOR 法 (Successive Over Relaxation Method)はガウス・ザイデル法の反復を加速した改良版である。(14.6)の代わりに，加速係数ωを導入して，

$$\phi_{i,j}^{n+1} = \phi_{i,j}^{n} + \omega\left(\phi_{i,j-1}^{n+1} + \phi_{i-1,j}^{n+1} + \phi_{i+1,j}^{n} + \phi_{i,j+1}^{n} - 4\phi_{i,j}^{n} + \Delta x^2 \rho_{i,j}\right)/4 \qquad (14.7)$$

という式で反復を行う。ただし，収束を得るために

$$(1.0 \le)\omega < 2.0 \qquad (14.8)$$

でなければならない。ωはメッシュの数により最適値が決まる。SOR 法のプログラム例を**問題 V-3** に示す。このプログラムでポワソン方程式を解くように設定されているが，係数のところを工夫すれば熱伝導方程式の陰解法にも利用できる。線形反復法 (緩和法) では，行列の形を意識しなくてよく，簡潔である。

問題 V-3

(1) **問題 V-2** の解を Gauss-Seidel 法あるいは SOR 法により求めよ。プログラムはダウンロードしたものを元に作成せよ。

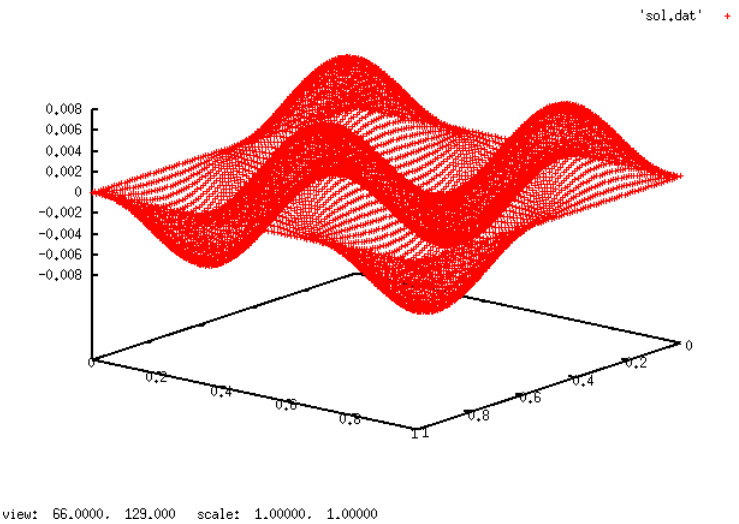

図 14.1 プログラムそのままの実行した結果 (`gnuplot> splot 'sol.dat'`でプロット)

問題 V-1　ガウスの消去法プログラム

メインプログラム　leqsys.f90

```
! ------------------------------------------------------------------------
! *   A sample program to solve linear equation systems                  *
! *               by Gaussian elimination method                         *
! ------------------------------------------------------------------------
      program leqsys

      implicit none
      real(8), allocatable:: a(:,:), b(:), x(:)
      integer, allocatable:: ip(:)
      integer:: n,
      integer:: i,j,k

!   number of dimension

      read(5,*) n

!   allocate dimension variables

      allocate ( a(n,n), b(n), x(n) )
      allocate ( ip(n) )

!  *****     Set coefficient     *****

      a(1,1) =  1.0d0
      a(1,2) =  2.0d0
      a(1,3) =  1.0d0
      a(2,1) =  3.0d0
      a(2,2) =  2.0d0
      a(2,3) =  0.0d0
```

```
      a(3,1) =  2.0d0
      a(3,2) = 10.0d0
      a(3,3) =  4.0d0

!  *****     Set RHS vector     *****

      b(1)   =  3.0d0
      b(2)   =  3.0d0
      b(3)   = 10.0d0

      do k = 1,n
        write (*,*) (a(k,j),j=1,n),b(k)
      end do

!  *****     decomposition: A -> LU     *****

      call       gausslu( a, ip, n )

!  *****     Substitution: x = (LU)**(-1)*b     *****

      call       gausssv( a, b, x, ip, n )

!  *****     Write result     *****

      do k = 1,n
        write (*,*) 'x(',k,' ) = ',x(k)
      end do

      stop
      end program leqsys
```

サブルーチンプログラム-1　gausslu.f90

```
! ------------------------------------------------------------------------
! *   Gaussian elimination method for a asymmetric dense matrix        *
! ------------------------------------------------------------------------
      subroutine gausslu(a, ip, n)

      implicit none
      real(8), parameter:: eps = 1.0d-37
      real(8):: a(n,n)
      real(8):: al
      integer:: ip(n)
      integer:: i, j, k, l
      integer:: lv, isingul
!
!    Initialize indices for pivoting
!
      do k = 1,n
        ip(k) = k
      end do

! ====================   k-loop   ==================================
      do k = 1, n-1

!  ****    partial pivot selection    ****

        l = k
        al = abs(a(ip(l),k))

        do i = k+1,n
          if ( abs(a(ip(i),k)) .gt. al ) then
            l = i
            al = abs(a(ip(l),k))
```

```
          end if
        end do

!  ****    row exchange    ****

        if ( l .ne. k ) then
          lv = ip(k)
          ip(k) = ip(l)
          ip(l) = lv
        end if

        if ( abs(a(ip(k),k)) .lt.eps )
          isingul = 9
          exit

!  ****    Gaussian elimination    ****

        piv = 1.0d0 / a(ip(k),k)

        do j = k+1, n
          a(ip(k),j) = a(ip(k),j) * piv
        end do
        do i = k+1, n
          do j = k+1, n
            a(ip(i),j) = a(ip(i),j) - a(ip(i),k) * a(ip(k),j)
          end do
      end do

      end do
! ====================   k-loop end  ================================

!  ****    singular matrix    ****
```

```
      if ( isingul.eq.9 ) then
        write (6,fmt) k
        n = -k
      end if

      return
      end subroutine gausslu
```

サブルーチンプログラム-2　gausssv.f90

```
! ---------------------------------------------------------------------
! *   solve decomposed eqation by forward and backward substitution    *
! ---------------------------------------------------------------------
      subroutine gausssv ( a, b, x, ip, n )

      implicit none
      real(8):: a(n,n), b(n), x(n)
      real(8):: SAx
      integer:: ip(n)
      integer:: n
      integer:: i,j,k

!  ****    forward elimination    ****

      do k = 1,n
        x(k) = b(ip(k))
      end do

      do k = 1, n-1
        x(k) = x(k) / a(ip(k),k)
        do i = k+1,n
          x(i) = x(i) - a(ip(i),k) * x(k)
        end do
```

```
      end do

!  ****    Backward substitution    ****

      x(n) = x(n) / a(ip(n),n)

      do k = n-1, 1, -1
        SAx = 0.0d0
        do i = k+1, n
          SAx = SAx + a(ip(k),i) * x(i)
        end do
        x(k) = x(k) - SAx
      end do

      return
      end subroutine gausssv
```

ガウスの消去法により，密行列を係数行列に持つ連立一次方程式を解くプログラムである。ピボット選択を行っている。メインルーチンで係数行列 A および右辺ベクトル $\boldsymbol{b}$ を与える。二次元配列 a が係数行列，一次元配列 b が右辺ベクトルである。サブルーチン gausslu で A の LU 分解(前進消去)，gausssv で $\boldsymbol{b}$ の前進消去および後退代入を行う。分解後の U の値が配列 a の上半分に，解 $\boldsymbol{x}$ が b に上書きされる。

前進消去の演算の仕方にはいろいろな方法があるが，ガウスの消去法の説明で取り上げた方法と同じであるクラウト法を用いた。ここで紹介した方法より高速な演算を行える方法もあり，実用的なプログラムではそれらの方が良く用いられている。

プログラムは，

http://home.hiroshima-u.ac.jp/nakakuki/Lectures/Geophys.%20ex..html

からダウンロードできる。

問題 V-3　SOR 法のプログラム

SOR 法によりポワソン方程式を解くプログラムである。メインルーチン，サブルーチン 1，モジュール 1 の 3 つのプログラム単位から構成されている。

コンパイルは

```
f95 -c sormod,f90
f95 -o s.out s*.f90
```

の 2 段階で行うこと。

メインプログラム　sor_main.f90

```
! ------------------------------------------------------------------------
! *     A sample program to solve Poisson equation by SOR method       *
! ------------------------------------------------------------------------
   program sormain

      use sormod

      implicit none
      integer:: i, j
      integer:: mxm1, mym1
      integer:: nitr
      real(8),parameter:: ax = 1.0d0, ay = 1.0d0
      real(8),parameter:: pi = 4.0d0 * atan(1.0d0)
      real(8):: dx, dy, xx, yy
      real(8):: omeg, eps
      character(40):: fmt, ofile

!  ****    set variables    ****

      fmt = '(100e8.2)'
      ofile='sol.dat'
```

```
      mxm1 = mx - 1
      mym1 = my - 1
      dx   = ax / dble(mx)
      dy   = ay / dble(mx)

!  ****    set coefficient matrix    ****

      do j = 1, mym1
        do i = 1, mxm1
          as(i,j) =  1.0d0 / dy**2
          aw(i,j) =  1.0d0 / dx**2
          ap(i,j) = -2.0d0 / dx**2 - 2.0d0 / dy**2
          ae(i,j) =  1.0d0 / dx**2
          an(i,j) =  1.0d0 / dy**2
        end do
      end do

!  ****    set boundary condition: xb = 0    ****

      do j = 0,my
        x(0, j) = 0.0d0
        x(mx,j) = 0.0d0
      end do

      do i = 0,mx
        x(i,0)  = 0.0d0
        x(i,my) = 0.0d0
      end do

!  ****    set RHS and initial value    ****

      do j = 1, mym1
```

```
        yy = dx * dble(j)
        do i = 1, mxm1
          xx = dx * dble(i)
          b(i,j) = -sin( 2.0d0*pi*xx/ax ) * sin( 3.0d0*pi*yy/ay )
!         write (6,*) b(i,j)
        end do
      end do

      do j = 1, mym1
        do i = 1, mxm1
          x(i,j) = 0.0d0
        end do
      end do

!   ****    solve by SOR method    ****

      omeg = 1.5d0
      nitr = 1000
      eps  = 1.0d-6

      call       SOR(omeg,eps,nitr,mxm1,mym1)

      write (*,*) ' iteration times   = ',nitr
      write (*,*) ' residual norm     = ',eps

!   ****    write unkown x    ****

      do j = 0,my
        write (6,fmt) ( x(i,j), i = 0,mx,1 )
      end do

      open (10,file=ofile)
      do j = 0,my
```

```
        yy = dx * dble(j)
        do i = 0, mx
          xx = dx * dble(i)
          write (10,*) xx,yy,x(i,j)
        end do
      end do

      stop

   end program sormain
```

サブルーチンプログラム-1　sor.f90

```
! ------------------------------------------------------------------------
! *   solve linear equations by SOR method for 2-dimensional           *
! *                finite difference method                            *
! ------------------------------------------------------------------------
   subroutine SOR(omeg,eps,nitr,mxm1,mym1)

      use sormod

      implicit none
      integer:: i,j,itr
      integer:: nitr,mxm1,mym1
      real(8)::  omeg,eps
      real(8)::  bnorm,rnorm,gosa,zansa
      real(8),allocatable:: dds(:,:)

      allocate ( dds(mxm1,mym1) )

!  ****    norm of RHS vector for convergent criterion    ****

      bnorm = 0.0d0
```

```
      do j = 1, mym1
        do i = 1, mxm1
          bnorm = bnorm + b(i,j)**2
        end do
      end do

!  ****    inverse of coefficient of x(i,j): 1/ap    ****

      do j = 1, mym1
        do i = 1, mxm1
          dds(i,j) = 1.0d0 / ap(i,j)
        end do
      end do

!  ========    iteration    ========================================================

      do itr = 1,nitr

         rnorm = 0.0d0

         do j = 1, mym1
           do i = 1, mxm1
             gosa = b(i,j) - ( as(i,j)*x(i,j-1) + aw(i,j) * x(i-1,j)              &
                             +ae(i,j)*x(i+1,j) + an(i,j) * x(i,j+1)               &
                             +ap(i,j)*x(i,j) )
             rnorm = rnorm + gosa**2
             x(i,j) = x(i,j) + omeg * gosa * dds(i,j)
           end do
         end do

         zansa = sqrt(rnorm/bnorm)
         if ( zansa .lt. eps ) exit
```

```
      end do

!  ========    end of iteration    ============================================

!  ****    end of SOR method    ****

      nitr = itr
      eps  = zansa
      return

   end subroutine sor
```

モジュールプログラム　sormod.f90

```
!  ****    Global variables    ****

   module sormod

      implicit none
      integer, parameter:: mx = 10,  my = 10
      real(8):: ae(0:mx,0:my), aw(0:mx,0:my), an(0:mx,0:my), as(0:mx,0:my),  &
                ap(0:mx,0:my)
      real(8):: x(0:mx,0:my),  b(0:mx,0:my)

   end module sormod
```

SOR 法により，ポワソン方程式から作られる連立一次方程式を解くプログラムである。２次元の差分法そのままの形で配列変数 ae，aw，an，as，ap に係数を記憶している。*ewns* は(i, j)の点 p からみて東西南北である。モジュールを用いて，メインルーチンとサブルーチンの両方にある係数 A や解 $\boldsymbol{x}$，右辺 $\boldsymbol{b}$ の配列変数をグローバル変数として扱っている。モジュール内とそれ以外に implicit none が指定されていることに注意せよ。use はローカル変数の implicit none よりも前に書かなければならない。

注意:ダウンロード版のプログラムは, tar ファイルにアーカイブされているので解凍すること。

ホームディレクトリ上で

```
tar xvf sor.tar
```

と入力すると SOR というディレクトリができ，その中にプログラムが入っている。