数値シミュレーションの基礎(2)

青木 孝
神奈川大学 理学部 情報科学科

Introduction to Numerical Simulation(2)

Takashi Aoki
Department of Information Science, Kanagawa University

Abstract. Introduce to perturbation and condition number for symmetric matrix (e.g. thermal equation).

And also, for symmetric matrix, I explain about Tate-block Gauss. That is valid for division of matrix by cache size.

· Perturbation and condition number(symmetric matrix)

· Tate-block Gauss and cache size

1 はじめに

前回の数値シミュレーションの基礎(1)において、板の温度分布 $u = u(x, y)$ を解く熱方程式：

$$div(-\kappa \nabla u) = f \tag{1.1}$$

を例にとり、一連の数値シミュレーションの手続きを村田流(文献[1])に解説した。その中で、現象のモデル化には、積分形式保存則を満足するように差分化するCV(Control Volume)法が有用であることや、数値解法(ソルバ)において、ガウスの消去法は理論上は解けるが、メモリの制約、計算時間の問題から実務上は使えない事を示した。熱方程式(拡散系のみ)のような対称行列の場合には、実務上は反復法であるICCG法に頼らざるを得ないことを実測で示した。

とは言え、直接法のガウスの消去法は、反復法のデバッグなど、必要となる場合がある。そのための、有用なガウス手法を知っておく事が必要となる。今回は、まず、ガウスの消去法の手法の一つ、「たてブロックガウス」について説明する。これは、村田(文献[1])が開発した手法で、行列を縦にブロック化して前方消去を行なう。特にPCの場合、キャッシュサイズとブロックの大きさを同じにすると、列ガウスに比べ、CPU時間を$\frac{1}{3}$に節約できる。前方消去において、それぞれのブロックの前方消去は、それぞれ独立なので、作業を並列化することもできる。次に、数値シミュレーション(1)で扱った、熱方程式のような対象行列に対する、摂動解析と結果と関係する条件数について説明する。

なお、次回の「数値シミュレーションの基礎(3)」では、熱方程式(拡散系：$-\kappa \nabla u$)に移流項($\mu \vec{b} u$)を加えた移流拡散系の方程式を解くための数値シミュレーションの手法を解説する。この移流拡散方程式：

$$div(-\kappa \nabla u + \mu \vec{b} u) = f \tag{1.2}$$

は、差分化(離散化)した時に、行列が非線形になるため、ICCG 法が使えないので、BCG-Stab 法のようなソルバを使う。この移流項が入るだけで、方程式は拡散項とのからみで、だんぜん解きづらくなる。離散化方程式が安定に解けるように仕向けるために、メッシュサイズが関係する(セルペクレ数として知られる)など、やっかいである。この対策の決定版として指数法があり、半導体デバイス解析の分野では、Shafetter·Gummel 法として呼ばれている。

2 CV 法による熱方程式(純拡散)の離散化

この部分は、前回の「数値シミュレーションの基礎(1)」と同じである。板の温度分布 $u = u(x, y)$ を知りたい。元になるのは、熱方程式となる。

$$div(-\kappa \nabla u) = f \tag{2.1}$$

$\kappa(x, y)$ は熱伝導率、$f(x, y)$ は熱源分布を表わす。(境界条件は、シミュレーション場の所で後述する)。

この方程式(2.1)は、普通には、初期値·境界値問題を設定し、それを差分法によって空間差分化して常微分方程式を作り、台形法(クランク-ニコルソン)によって解く。この時、式(2.1)は、

$$-\kappa(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) = f' \tag{2.2}$$

と書かれることが多い。この式(2.2)は、たとえばκに場所依存性、温度依存性がある場合($\kappa = \kappa(x, y, u)$)には間違いである。κは温度勾配 ∇u と直接くっついていなければならない。

$$\frac{\partial}{\partial x}(-\kappa\frac{\partial u}{\partial x}) + \frac{\partial}{\partial y}(-\kappa\frac{\partial u}{\partial y}) = f' \tag{2.3}$$

この式(2.3)に基づいて離散化したモデルの解は、式 (2.2) の解とは違ったものになる。

なぜ、そのような誤りをするかと言うと、熱方程式のモデル化の元になっている考え方:熱の保存則を忘れて、微分方程式がひとり歩きしてしまったことによる。実は、式(2.1)は、

「領域 Q の表面Γからの流出量+ $\int_\Gamma \boldsymbol{\Phi} \cdot \boldsymbol{n} ds$ は、領域 Q 内での発生量$\int_\Omega f\ dv$と等しい。」

という物理量の保存則を表している。

$$\int_\Gamma \boldsymbol{\Phi} \cdot \boldsymbol{n} ds = \int_\Omega f\ dv \tag{2.4}$$

$\boldsymbol{\Phi}$は単位面積当たり、単位時間当たりの熱量の流量ベクトルを表わし、$\boldsymbol{n}$はΓ上に立てた外向き単位法線ベクトルを表わす。dsは面積要素、dvは体積要素。

式(2.4)を保存則の積分形式という。左辺を、ガウスの発散公式を使って、

$$\int_\Gamma \boldsymbol{\Phi} \cdot \boldsymbol{n} ds = \int_\Omega div\boldsymbol{\Phi} dv \tag{2.5}$$

と変形すると、式(2.4)は、

$$\int_\Omega div\boldsymbol{\Phi} dv = \int_\Omega f\ dv \tag{2.6}$$

となる。ここで、熱は温度勾配 $\nabla u(x,y)$ に比例して、高温部から低温部に流れる：

$$\Phi = -\kappa \nabla u \quad (\kappa \text{は熱伝導率})$$

とすれば、

$$\int_\Omega div(-\kappa \nabla u)\, dv = \int_\Omega f\, dv \tag{2.7}$$

となり、積分をはずせば、式 (2.1) が出る。

村田は、前に述べた、保存則のようなモデルの元になるような考え方を、忘れてしまうような過ちをしないよう、

「積分形式保存則を満足するように差分化する CV(Control Volume) 法」

を奨めている。シミュレーションでは、収束判定値にもよるが、0.001%以下で、保存則を満足させることはたやすい。

具体的には、式 (2.4) に基づき $\Phi = -\kappa \nabla u$ とした次式で、空間離散化を行なう。

$$\int_\Gamma (-\kappa \nabla u) \cdot \boldsymbol{n}\, ds = \int_\Omega f\, dv \tag{2.8}$$

次に、縦 10cm(y 方向) 横 11cm(x 方向) の長方形の例で、離散化の手順を説明する。今 Fig.1 のような 10cm×11cm 四方の板をメッシュに分け、各格子点での温度分布を考える。Fig.1 では、メッシュを横 (x)、縦 (y) 方向とも 1cm きざみで均等にとっているが、実務上、大事な所（熱源の境界など）はメッシュを精しくして解くので、不均等あみ目となる。格子点には、Fig.1 のように通し番号を付ける。全格子点数が方程式の元数となり、Fig.1 では、$N = M(y\text{方向}) \times M1(x\text{ 方向}) = 10 \times 10 = 100$ 点となる。i 番目の格子点と、隣あう十字方向の 4 点はそれぞれ、$i+1$, $i-1$, $i+m$, $i-m$ 点となっている。この例では、境界条件は、Fig.1 の DA, AB, AC 境界を固定境界とし、簡単のために温度 $u = 0[C]$ と与える。上側の DC 境界は、ノイマン境界 ($\kappa \nabla u = 0 : flux$ が 0) とする。

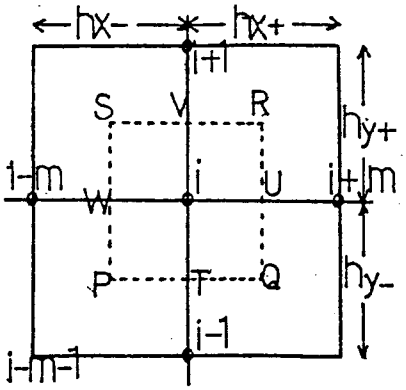


Fig.2. Mesh for CV method

この Fig.1 の各節 (格子) 点 i の i 点回りに、小領域 PQRS(Control Volume という) を作り、その周囲を Γ に見立てて、式 (2.8) の CV 法による離散式を作る (Fig.2)。辺 PQ, QR, RS, SP に立てた外向き単位法線

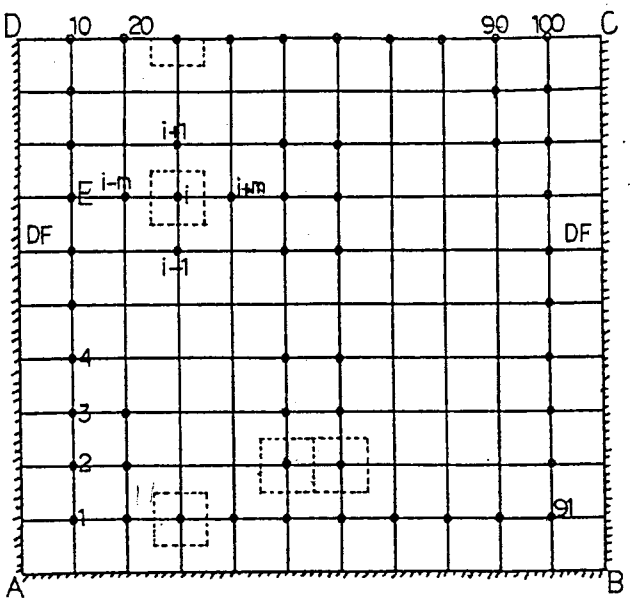


Fig.1. Simulation Field

ベクトルが、$(n_x, n_y)^T = (0,-1)^T,\ (1,0)^T,\ (0,1)^T,\ (-1,0)^T$であることに留意し、また、

$$\nabla u \cdot \boldsymbol{n} = (\frac{\partial u}{\partial x},\ \ \frac{\partial u}{\partial y})^T \cdot (n_x,\ \ n_y)^T$$

であることを使うと、左辺の線積分：

$$\int_\Gamma (-\kappa \nabla u) \cdot \boldsymbol{n}\ ds = \int_{PQ} \ + \ \int_{QR} \ + \ \int_{RS} \ + \ \int_{SP} \tag{2.9}$$

の各項は次のように書ける。

$$\int_{PQ} \doteq -d_P(\frac{u_i - u_{i-1}}{hy_-})(-1)\frac{hx_-}{2} - d_Q(\frac{u_i - u_{i-1}}{hy_-})(-1)\frac{hx_+}{2} \tag{2.10}$$

$$\int_{QR} \doteq -d_Q(\frac{u_{i+m} - u_i}{hx_+})(+1)\frac{hy_-}{2} - d_R(\frac{u_{i+m} - u_i}{hx_+})(+1)\frac{hy_+}{2} \tag{2.11}$$

$$\int_{RS} \doteq -d_R(\frac{u_{i+1} - u_i}{hy_+})(+1)\frac{hx_+}{2} - d_S(\frac{u_{i+1} - u_i}{hy_+})(+1)\frac{hx_-}{2} \tag{2.12}$$

$$\int_{SP} \doteq -d_S(\frac{u_i - u_{i-m}}{hx_-})(-1)\frac{hy_+}{2} - d_P(\frac{u_i - u_{i-m}}{hx_-})(-1)\frac{hy_-}{2} \tag{2.13}$$

ここで、T 点での$\frac{\partial u}{\partial y}$は$\frac{u_i - u_{i-1}}{hy_-}$、W 点での$\frac{\partial u}{\partial x}$は$\frac{u_i - u_{i-m}}{hx_-}$などと、各点での$u_i$を使って差分化している。また、伝導度$\kappa(x,y)$は、メッシュで区分けした面内は一定と考えて差分化する。たとえば、P点での$\kappa(x,y) \equiv d_P$は、4 点$i,\ i-m,\ i-m-1,\ i-1$が囲む面内一定として、プログラムレベルでは$DF(i)$に格納するなど工夫する。配列DFの範囲は、$DF(N+M)$まで必要となる。

これら(2.10)～(2.13)式を、$u_{i-m},\ u_{i-1},\ u_i,\ u_{i+1},\ u_{i+m}$項毎に整理すれば、第$i$番方程式：

$$a_{i,i-m}u_{i-m} + a_{i,i-1}u_{i-1} + a_{i,i}u_i + a_{i,i+1}u_{i+1} + a_{i,i+m}u_{i+m} = \ f_i \tag{2.14}$$

の係数(左辺)は、次式となる（式(2.8)の右辺=f_iは後述）。

$$a_{i,i} = \frac{1}{2}\left[d_P\left(\frac{hx_-}{hy_-} + \frac{hy_-}{hx_-}\right) + d_Q\left(\frac{hx_+}{hy_-} + \frac{hy_-}{hx_+}\right) + d_R\left(\frac{hx_+}{hy_+} + \frac{hy_+}{hx_+}\right) + d_S\left(\frac{hx_-}{hy_+} + \frac{hy_+}{hx_-}\right)\right]$$

$$a_{i,i-m} = -\frac{1}{2}\left[d_S\frac{hy_+}{hx_-} + d_P\frac{hy_-}{hx_-}\right]$$

$$a_{i,i-1} = -\frac{1}{2}\left[d_P\frac{hx_-}{hy_-} + d_Q\frac{hx_+}{hy_-}\right]$$

$$a_{i,i+1} = -\frac{1}{2}\left[d_R\frac{hx_+}{hy_+} + d_S\frac{hx_-}{hy_+}\right]$$

$$a_{i,i+m} = -\frac{1}{2}\left[d_Q\frac{hy_-}{hx_+} + d_R\frac{hy_+}{hx_+}\right] \tag{2.15}$$

式(2.14)の左辺は、式(2.10)～(2.13)を式(2.9)に代入したもので、式(2.14)は、行列方程式になっている($\mathbf{A}\ (a_{k,j})\mathbf{U}(u_k) = \mathbf{F}(f_k)$)。**Fig.1 の通し番号の節点** $i\ (=1 \rightarrow 100)$ は、

$$i = m \times (j-1) + k \qquad (j = 1, m1,\ k = 1, m;\ m1 \text{ は } x \text{ 方向の点の数})$$

などのように指定する。Fig.1 の場合は、はなはだ簡単に、$hx = hx_+ = hx_- = 1[cm]$, $hy = hy_+ = hy_- = 1[cm]$ である。今、$d_P = d_Q = d_R = d_S = 1[\frac{cal}{C \cdot sec \cdot cm}]$ とすれば、式(2.15)はそれぞれ、

(2.16) $a_{i,i} = 4.0 \quad a_{i,i-m} = -1.0 \quad a_{i,i-1} = -1.0 \quad a_{i,i+1} = -1.0 \quad a_{i,i+m} = -1.0$

となる。問題を簡単にするのは、$hx = hy = 1[cm]$ とすることより、むしろ $hx = hy$の均等あみ目とすることが主役である。その時、

$$a_{i,i} = \frac{1}{2}\left[2d_P + 2d_Q + 2d_R + 2d_S\right]$$

$$a_{i,i-m} = -\frac{1}{2}\left[d_S + d_P\right] \qquad a_{i,i-1} = -\frac{1}{2}\left[d_P + d_Q\right]$$

$$a_{i,i+1} = -\frac{1}{2}\left[d_R + d_S\right] \qquad a_{i,i+m} = -\frac{1}{2}\left[d_Q + d_R\right]$$

などとなる。実務上、避けて通れない不均等あみ目 ($hx_+ \neq hx_-$, $hy_+ \neq hy_-$) は、行列 $\mathbf{A}(a_{i,j})$ を解きづらくする一因となり、あみ目の取り方が、解いた結果に影響する。

村田の奨めるあみ目の取り方は、Fig.1 の $1[cm]$ を分割する分割数を MJとして、$1[cm]$ 毎に MJの値を変えてやる方法である。$1[cm]$ 毎に、$hx = \frac{1.0}{MJ}$が変わることになる。メッシュを倍に精しくしたければ、$MJ = 2$ とすればよい。プログラムレベルでは、節点 i 番 (j列に当たる) の、たとえば hx_-, hx_+を、それぞれ $DHX(J)$, $DHX(J+1)$ に格納しておく (y も同様)。$1cm$ 毎の境界では、$hx_- \neq hx_+$となるわけである。Fig.1 で、MJを導入して均等あみ目 ($hx_- = hx_+$) とすると、

$M = MJ \cdot 10 \qquad M1 = MJ \cdot 11 - 1 \qquad N = M \cdot M1$

というあみ目になる。行列 $\mathbf{A}$ は、$n \times n$ 行列なので、メモリサイズは、$8 \times n^2$ Byte 必要。

これら係数も、**上側のノイマン境界の節点 (CD 上) では、左辺の積分区間を Fig.2 の下半分だけにする必要があり、扱いが異なる。**線積分は、

(2.17) $$\int_{\Gamma\ CD} (-\kappa \nabla u) \cdot \boldsymbol{n}\ ds = \int_{WP} + \int_{PQ} + \int_{QU}$$

となり、CD 上の節点では、係数は (2.15) と同様の手順で、

$$a_{i,i} = \frac{1}{2}\left[d_P\left(\frac{hx_-}{hy_-} + \frac{hy_-}{hx_-}\right) + d_Q\left(\frac{hx_+}{hy_-} + \frac{hy_-}{hx_+}\right)\right]$$

$$a_{i,i-m} = -\frac{1}{2}\left[d_P \frac{hy_-}{hx_-}\right]$$

$$a_{i,i-1} = -\frac{1}{2}\left[d_P \frac{hx_-}{hy_-} + d_Q \frac{hx_+}{hy_-}\right]$$

$$a_{i,i+1} = 0.0$$

(2.18) $$a_{i,i+m} = -\frac{1}{2}\left[d_Q \frac{hy_-}{hx_+}\right]$$

となる。$hx = hy = 1[cm]$、$d_P = d_Q = 1[\frac{cal}{C \cdot sec \cdot cm}]$ の下では、

(2.19) $$a_{i,i} = 2.0 \quad a_{i,i-m} = -0.5 \quad a_{i,i-1} = -1.0 \quad a_{i,i+1} = 0.0 \quad a_{i,i+m} = -0.5$$

となる。ここでのシミュレーションでは、$\kappa = 1.0$ の一定では、面白くないので、Fig.1 の両側の伝導率κに段差をつける。

$i = 1 \rightarrow 10$ $\quad \kappa_i(x,y) = DF$(一定値)

$i = 101 \rightarrow 110$ $\quad \kappa_i(x,y) = DF$(一定値)

それ以外は、$\kappa_i(x,y) = 1.0$ とする。

u_iは節点 1～100 までだが、κ_iは 1～110 まであることに注意。この変数 DFを、$DF = 1.0[\frac{cal}{C \cdot sec \cdot cm}]$ にすれば、$\kappa = 1$ 一定の固定境界、$DF = 0.0$ にすれば、両サイドはノイマン境界 $(-\kappa \nabla u = 0)$ と同じになる。$i = 1 \rightarrow 10$ の各係数は、

$$a_{i,i} = \frac{1}{2}[DF \cdot 2 \ + \ 2 \ + \ 2 \ + \ DF \cdot 2] = [2 \ + \ 2DF]$$

(2.20) $$a_{i,i+1} = -\frac{1}{2}[1 \ + \ DF] \qquad a_{i,i+m} = 0.0$$

となる。

また、節点iに隣あう各$i+1$, $i-1$, $i+m$, $i-m$のいずれかに、固定境界がある場合には、違った扱いをする必要がある。例えば、左側に固定境界$u = u_0(= 0)$を持つE点(Fig.1)の場合には、i番方程式が、

(2.21) $$a_{i,i-m}u_0 + a_{i,i-1}u_{i-1} + a_{i,i}u_i + a_{i,i+1}u_{i+1} + a_{i,i+m}u_{i+m} = f_i$$

となる。この時、未知数項でない$a_{i,i-m}u_0$は、右辺に移項してf_iの仲間に入れる。したがって、

(2.22) $$a_{i,i-1}u_{i-1} + a_{i,i}u_i + a_{i,i+1}u_{i+1} + a_{i,i+m}u_{i+m} = f_i - a_{i,i-m}u_0$$

を解くことになるので、係数は、

$a_{i,i-m} = 0.0$

$a_{i,i-1}$, $a_{i,i}$, $a_{i,i+1}$, $a_{i,i+m}$は、式(2.15)と同じと設定する。$u_0 = 0$ ならば、なおさら簡単で、右辺に手を加える必要はなくなる。

以上より、式(2.9)の左辺は、次の行列の積で表せる(Fig.3)。今、この $\mathbf{A}$ が、$a_{i,j} = a_{j,i}$の対称性（隣あう Control Volume 境界で、連続であることからも明らか）を利用すると、各iについて、$a_{i,i}$, $a_{i,i+1}$, $a_{i,i+m}$だけ作れば十分だということになる。この時、

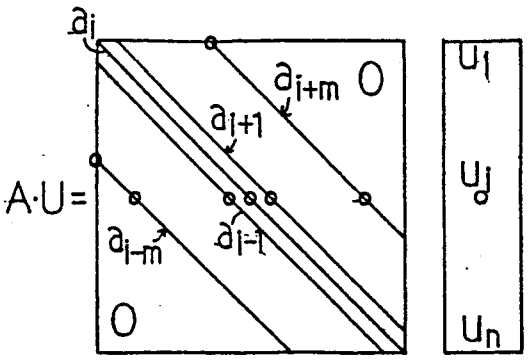


Fig.3. Band Matrix

(2.23) $$a_{i-m,i}u_{i-m} + a_{i-1,i}u_{i-1} + a_{i,i}u_i + a_{i,i+1}u_{i+1} + a_{i,i+m}u_{i+m} = f_i$$

を解くことになる。プログラムレベルでは、

$a_{i,i} = AA(I),\ a_{i,i+1} = AB(I),\ a_{i,i+m} = AC(I)$

と配列を用意し、それぞれの範囲を $AA(N),\ AB(-M:N),\ AC(-M:N)$ (−M:N は−M〜N までの事) とし、

$$AB(N) = 0.0, \quad A(0) = 0$$

(2.24) $do\ \ I = N - M + 1, N\ \ [\ AC(I) = 0.0\ ] \quad do\ \ I = 1 - M, 0\ \ [\ AC(I) = 0.0\ ]$

と初期設定しておく工夫だけで、**A·U** が次式のように楽に計算できる。

$do\ \ I = 1, N$

[fi=AC(I-M)*U(I-M)+AB(I-1)*U(I-1)+AA(I)*U(I)+AB(I)*U(I+1)+AC(I)*U(I+M)]

(2.24a)

節点 1 について見ておく。節点 1 では、

$$a_{1-m}u_0 + a_{1-1}u_0 + a_1u_1 + a_{1+1}u_{1+1} + a_{1+m}u_{1+m} = f_1 \tag{2.25}$$

である。$a_{1-m} = 0,\ \ a_{1-1} = 0$ としなければいけない。実際には、式 (2.24a) から、

AC(1-M)*U(1-M)+AB(0)*U(0)+AA(1)*U(1)+AB(1)*U(2)+AC(1)*U(1+M)= fi

(2.25a)

となる。u_iの範囲は、$U(-M:N+M)$ で、$U(-M)$〜$U(0) = 0.0$、$U(N+1)$〜$U(N+M) = 0.0$ と始末する。$a_{1-m} = 0,\ \ a_{1-1} = 0$ は、式 (2.24) から満足している。

後回しにした、右辺の始末は、

$$f_i = \int_\Omega f\ dv$$

の面積分になる。Fig.2 の面 PQRS から、発生または吸収される熱量を 4 つの部分の合算で求める。伝導率κの場合のように、面 (i, i-m, i-m-1, i-1) 内は一定の熱源をもつとして離散化するので、この面の熱源を、$f_P[\frac{cal}{sec\cdot cm^3}]$ とすれば、$\frac{1}{4}$面 (iWPT) に当たる f_iへの寄与は、$f_P\frac{hx_-}{2}\frac{hy_-}{2}[\frac{cal}{sec}]$ (厚さが $1[cm]$ あると思う) となる。

したがって、節点 i での f_iは、

$$f_i = f_P\frac{hx_-}{2}\frac{hy_-}{2} + f_Q\frac{hx_+}{2}\frac{hy_-}{2} + f_R\frac{hx_+}{2}\frac{hy_+}{2} + f_S\frac{hx_-}{2}\frac{hy_+}{2} \qquad [\frac{cal}{sec}] \tag{2.26}$$

と求まる。$hx = hy = 1$ ならば、$f_i = \frac{1}{4}(f_p + f_Q + f_R + f_S)$。

当然、上側のノイマン境界上の節点 (CD 上) では、積分区間が下半分しかないので、

$$f_{i\ CD} = f_P\frac{hx_-}{2}\frac{hy_-}{2} + f_Q\frac{hx_+}{2}\frac{hy_-}{2} \qquad [\frac{cal}{sec}] \tag{2.27}$$

となる。

式 (2.26) は、$hx = hy$均等あみ目で、$f_P = f_Q = f_R = f_S = fc$ ならば、

$$f_i = \frac{hx\ hy}{4}(f_p + f_Q + f_R + f_S) = fc\cdot hx\ hy$$

と簡単になる。

今回のシミュレーションでは、熱源を節点 i に対して、

i=42 to 44, i=52 to 54 では： $f_i = +0.2$ $[\frac{cal}{sec}]$

i=46 to 48, i=56 to 58 では： $f_i = -0.2$ $[\frac{cal}{sec}]$

と与える。プログラムレベルでは、面 (i, i-m, i-m-1, i-1) の熱源密度を FP(I) の配列に格納しておく。FP の範囲は、FP(N+M) まで必要で、伝導率κと同じ扱いをする。離散化時には、各節点 i での温度 u_iや、メッシュで区切った面での密度として扱う$\kappa(d_P)$, f_Pなどの物理量には、それぞれの扱いに特に注意を必要とする。

また、はじめに対称の熱源 f_iを与えるにはそれなりの意味がある。もし、離散化が間違っていれば、解は非対称になり、そのデバッグになる。

3 縦ブロックガウス (GLUB)

かくして、式(2.23) により、

$$\mathbf{A}(a_{i,j}) \cdot \mathbf{U}(u_i) = \mathbf{F}(f_i) \tag{3.1}$$

の n 行 n 列の連立一次方程式を解くという問題に帰着したわけである。

また、**A** は、各行 (i) につき、$\mid a_{i,i} \mid \geq \sum_{j \neq i} \mid a_{i,j} \mid$ になっており ((2.16) を見よ)、行対角優位になっている。**A** が、行対角優位の時には、ピボット (部分軸) 選択なしのガウス (消去法) で、上三角化でき、解けることが分かっている。部分軸選択があるなしで、計算時間が主部だけで倍違う。

ここでは、「縦ブロックガウス (GLUB)」について説明する。仮想メモリ方式計算機:「プログラムは、あたかもディスクのメモリ容量が主メモリにあるかのように思ってプログラムできるように、ハードウェアと OS の方で面倒を見る」の仕組みでは、ディスクと主メモリの間の情報転送が頻繁に起きないように、プログラムするのが望ましい。(転送は、連続する番地の内容をページと呼ばれる単位に区切って行なわれる。このことをページスワップという。1 ページは普通 500 長語、4KB。)

配列 **A**(i,j) において、**A**(i,j) がディスク上にある時は、当然それを載せたページを主メモリに呼び出さねばならない。その時のページ交換の規則は、「主メモリ上の最後に使われてから最も時間の経過したページをディスクに追い出して、その後に必要なページを入れる。」である。列ガウスと行ガウスの違い等での、添字の参照の仕方で、ページスワップを頻繁に引こ起こす (特に行ガウス) ことになる。そこで、そのことを考慮して、村田 (文献 [6][9]) が「縦ブロックガウス」を考案した。主メモリの大きさ (超える時) や、ページスワップ (サイズ) のことを考えて、2 次元配列をいくつかのブロックに分けて、掃き出しを行なうのである。

以下、文献 [6] による。主メモリ容量を大巾に超えるような大次元問題のためには、列ガウスでももうひと工夫が要る。それが、表記の縦ブロックガウスである。いま行列の元数を n、主メモリ容量を M 語とし、$M \ll n^2$とする。このとき行列を縦長のいくつかのブロックに分ける (Fig.4)。整数 m を、

$n \cdot m + r < M$ ($n \cdot m$ は 1 ブロックの語数、r はプログラムやワークエリアの語数)

となるように選び、m 個の列ごとにまとめてひとつのブロックとする。ブロック番号を kb とする。

kb=1 に対しては (ブロック 1 内の要素だけを対象として)

k の do ループは m まで, に注意

```
ir=0
do  k=1,m
    amax=|a(k,k)| ; ip(k)=k
    do  i=k+1,n
        aik=|a(i,k)|
        if  aik>amax then
            ip(k)=i ; amax=aik
    if  amax>eps then
        if  ip(k)≠k then
            a(k,k) ⇔ a(ip(k),k)
        do  i=k+1,n
            a(i,k)=-a(i,k)/a(k,k)
        do  j=k+1,m          (mまで, に注意)
            if  ip(k)≠k then
                a(k,j) ⇔ a(ip(k),j)
            do  i=k+1,n
                a(i,j)=a(i,j)+a(i,k)·a(k,j)
    else
        ir=1 ; return
```

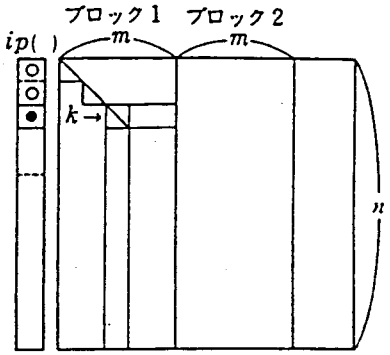
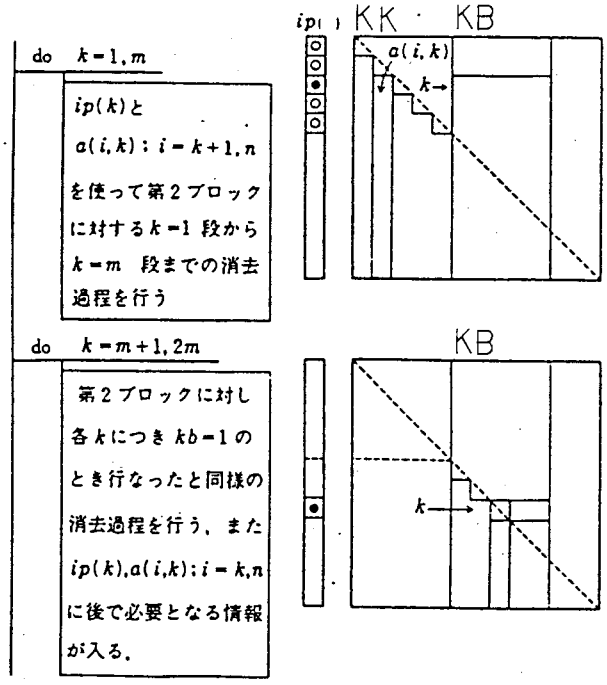


Fig.4. Block division for Matrix

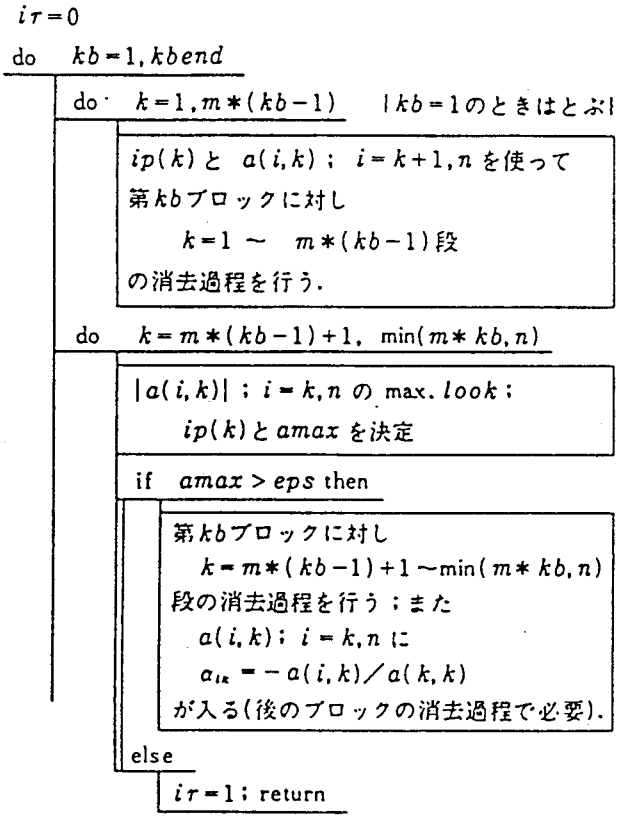
kb=2 に対しては (kb=3,···も同様だが)

k=1 から m までと,m+1 から 2m までの二つの部分に分けて、



以上、理解を容易にするための、kb=1 と kb=2 のときを特に取り上げて解説した。kb=1 から kb=kbend までの全過程は、次のようになる。

＜縦ブロックガウス GLUB ＞



以上、文献 [6] によった。パソコンの場合には、配列サイズが実メモリを超えた場合には使いものにならないので、ページスワップ同様の現象は、配列サイズが実メモリに収まるとして、キャッシュメモリオーバーヘッドで現れる。キャッシュメモリとは、主メモリと演算制御装置との間にある（主メモリよりも数倍速い小容量の）メモリの事で、たとえば、とびとびの番地参照だと、キャッシュメモリと主メモリ間のデータ転送が頻繁に起こり、CPU タイムの増加（キャッシュメモリオーバーヘッド）を著しくする。今、計測中のパソコンでは、2 次キャッシュメモリは 256KB で、このキャッシュに、K 段を掃きだすのに必要な 2 ブロック (KK と KB) が入りきると効率が良いことになる。MJ(メッシュの精しさ)=5 とすると、元数 $N = (MJ \times 10) \times (MJ \times 11 - 1) = 2700$ となり、まとめて処理する列数 m(MM)=5 とした時には、$2700 \times 5 \times 8\ Byte = 108KB$が 1 ブロックのサイズになる。その時、2 ブロック=216KB で、計測中 PC のキャッシュに入るので、MM=5 が一番効率が良いと予想できる。この時、配列 A(i,j) は、全体が$\frac{2700}{5} = 540$ に分割される。

実際に、横軸に m(MM) ($1 \leq MM \leq N$ で、MM=N の時、通常の列ガウスとなる。) をとり、縦軸に CPU(秒) 時間をとると、Fig.5 となる。Fig.5 で、下側のプロットは MJ=5 (元数 N=2700) の場合で、上側は MJ=6 (MM=4 の時、256KB キャッシュメモリに収まる。元数 N=3900) の場合で、ほぼ予想通りの結果となる。

計算は、日立 FLORA330, 933MHz, PentiunIII, 2 次キャッシュ256KB, システムバスクロック 133MHz で行なった。数値結果は Table 1 による。Fig.5 によれば、キャッシュサイ

ズに入るようなMMでブロックを切ってやれば、列ガウスのほぼ$\frac{1}{3}$のCPU時間で計算することができる。

MJ=5の時は、MM=6で最小CPU時間となり、 98(sec):列ガウス比 [0.3]

MJ=6の時は、MM=5で最小CPU時間となり、357(sec):列ガウス比 [0.37]

前半の過去のKK=1 ~ $(KB-1)$段までの掃き出しをKB段に施す部分と、後半のKB段自身を掃き出す部分をサブルーチン化すると、最小CPU時間となるMMは、計算通りの値となる。後でプログラムリストを示すが、縦ブロックガウス(GLUBはSUB版, GLUBMはSUBなし)のKKブロックとKBブロックへの処理をそれぞれサブルーチン：(GLUBAとGLUBB)にした場合には、

MJ=5の時は、MM=5で最小CPU時間となり、122(sec)。(サブなしより20 %増)

MJ=6の時は、MM=4で最小CPU時間となり、457(sec)。(サブなしより22 %増)

Table 1 CPU Time(sec) performance for Block size MM(MJ=5,6)

MJ	1	2	3	4	5	6	7	8	10	100	N
5	301	175	133	113	103	98	100	107	128	336	334
6	908	532	408	368	357	389	444	494	634	1037	1007

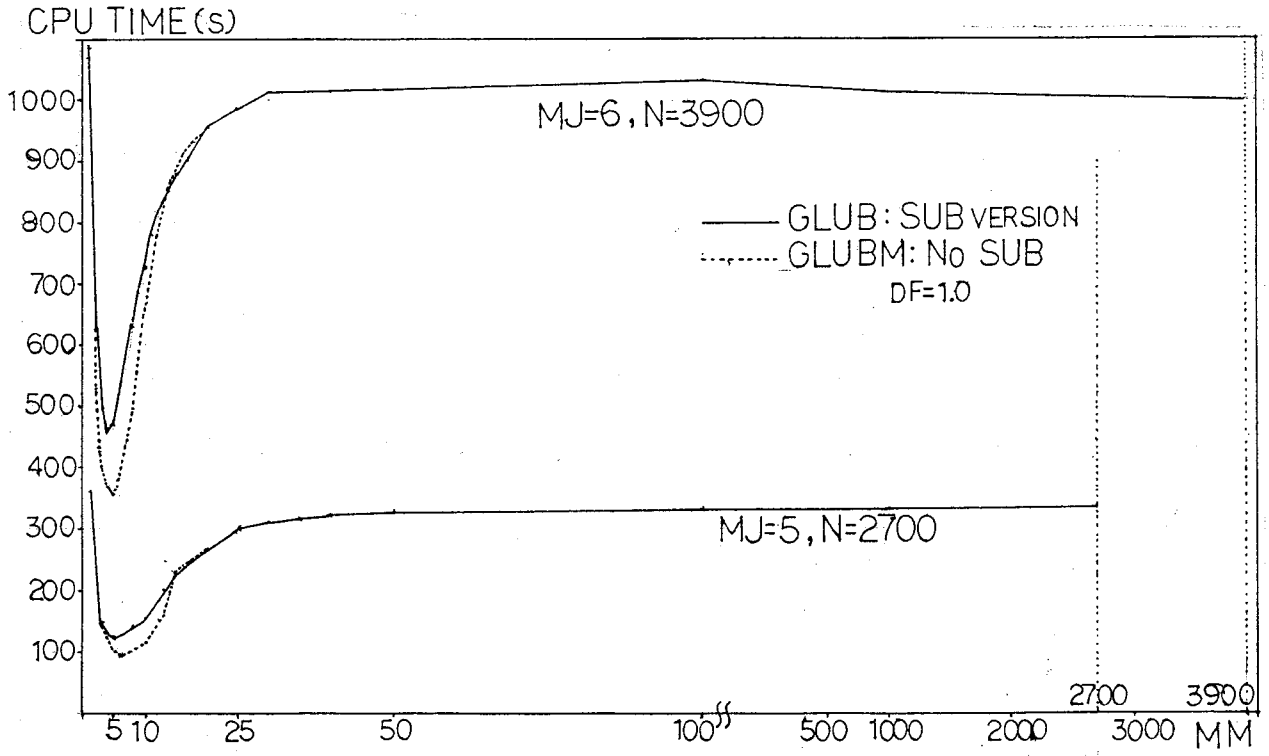


Fig.5. CPU Time(sec) performance for Block size MM(MJ=5,6; GLUB, GLUBM)

次に、プログラムリストを示す。

```
IMPLICIT REAL*8 (A-H,O-Z)
REAL*4 CPUT
PARAMETER(MJ=5,M=MJ*10,M2=MJ*11-1,N=M*M2,EPS=1.0D-7)
DIMENSION A(N,N),AA(N),AB(-M:N),AC(-M:N),F(N), IP(N),WK(N)
```

```
CC*
      DF=1.0D0
      MM=5
CC*
      WRITE(6,*) ' % DF= ',DF, ' , MM= ',MM
      WRITE(6,*) ' % M= ',M,' M2= ',M2,' N= ',N
      FMJ= DFLOAT(MJ)
      DHX=1.0D0/FMJ
      DHY=1.0D0/FMJ
C
      DO 10 I=1,M-1
        AA(I)=2.0D0*(DF+1.0D0)
        AB(I)=-(1.0D0+DF)/2.0D0
        AC(I)=-1.0D0
   10 CONTINUE
      AA(M)=DF+1.0D0
      AB(M)=0.0D0
      AC(M)=-0.5D0
C
      DO 30 J=1,M2-2
        DO 20 K=1,M-1
          AA(M*J+K)=4.0D0
          AB(M*J+K)=-1.0D0
          AC(M*J+K)=-1.0D0
   20   CONTINUE
        AA(M*J+M)=2.0D0
        AB(M*J+M)=0.0D0
        AC(M*J+M)=-0.5D0
   30 CONTINUE
C
      DO 40 I=N-M+1, N-1
        AA(I)=2.0D0*(DF+1.0D0)
        AB(I)=-(1.0D0+DF)/2.0D0
        AC(I)=0.0D0
   40 CONTINUE
      AA(N)=DF+1.0D0
      AB(N)=0.0D0
      AC(N)=0.0D0
C
      DO 50 I=1,N
        A(I,I)=AA(I)
  50  CONTINUE
      DO 52 I=1,N-1
        A(I,I+1)=AB(I)
        A(I+1,I)=AB(I)
  52  CONTINUE
      DO 54 I=1,N-M
        A(I,I+M)=AC(I)
        A(I+M,I)=AC(I)
  54  CONTINUE
C
*C    Uhen ***
      DO 60 J= 5*MJ-1, 6*MJ-1
        DO 70 I=0,2*MJ
          F(M*J +2*MJ+I)= 0.2D0*(DHX*DHY)
          F(M*J +6*MJ+I)=-0.2D0*(DHX*DHY)
   70   CONTINUE
   60 CONTINUE
C
      CALL CLOCK0
C
*C    AR*U=F wo Toku; Kotae --> F ***
      CALL GLUBM( N, MM, A, IP, EPS, IR )
CC    CALL GLUB ( N, MM, A, IP, EPS, IR, WK )
CC    CALL GLU  (N,A, IP,EPS,IR)
      IF( IR.EQ.0 )  THEN
        CALL GSLV (N, A, F,IP)
      ENDIF
C
```

```
      CALL CLOCK(CPUT)
      UMIN=F(1)
      JUMIN=1
      KUMIN=1
      UMAX=F(1)
      JUMAX=1
      KUMAX=1
      DO 130 J=1,M2
        DO 120 K=1,M
          I=M*(J-1)+K
          IF(F(I).LT.UMIN) THEN
            UMIN=F(I)
            JUMIN=J
            KUMIN=K
          ENDIF
          IF(F(I).GT.UMAX) THEN
            UMAX=F(I)
            JUMAX=J
            KUMAX=K
          ENDIF
  120   CONTINUE
  130 CONTINUE
C
      WRITE(6,*) ' % MJ= ',MJ
      WRITE(6,*) ' % CPUT= ',CPUT
      WRITE(6,*) ' % UMIN,(x,y)= ',UMIN,'(',JUMIN,',',KUMIN,')'
      WRITE(6,*) ' % UMAX,(x,y)= ',UMAX,'(',JUMAX,',',KUMAX,')'
      WRITE(6,*) ' % IR= ',IR
C
      WRITE(6,*) ' z=[ '
C
CC    DO 80 K= M,MJ,-MJ
CC      WRITE(6,2000) (F(M*J+K),J=MJ-1,MJ*10-1,MJ),' ; '
CC 80 CONTINUE
*C    MATLAB You ***
      DO 80 K= MJ,M,MJ
        WRITE(6,2000) (F(M*J+K),J=MJ-1,MJ*10-1,MJ),' ; '
   80 CONTINUE
 2000 FORMAT(1H ,10F6.2,A3)
C
      WRITE(6,*) ' ]; '
      WRITE(6,*) ' contour(0:1:9, 0:1:9, z,10);'
C
      STOP
      END
CCCC
CCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE GLUBM ( N, M, A, IP, EPS, IR )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(N,N), IP(N)
C
      IR = 0
      KBE = (N-1)/M+1
      DO 1000 KB = 1, KBE
C
        KB1 = (KB-1)*M+1
        DO 100 K = 1, M*(KB-1)
          IPK = IP(K)
C
C*        DO 110 J = KB1,KB*M
*                          ---- Bug ***
          DO 110 J = KB1, MIN(KB*M,N)
            IF( IPK.NE.K )  THEN
               W = A(IPK,J)
               A(IPK,J) = A(K,J)
               A(K,J) = W
            END IF
```

```
C
            T = A(K,J)
            DO 120 I = K+1, N
  120         A(I,J) = A(I,J)+A(I,K)*T
  110     CONTINUE
  100   CONTINUE
CC
CC
        DO 200 K = M*(KB-1)+1, MIN(M*KB,N)
C
          AMAX = ABS(A(K,K))
          IPK = K
          DO 210 I = K+1, N
            AIK = ABS(A(I,K))
            IF( AIK.GT.AMAX )  THEN
               IPK = I
               AMAX = AIK
            END IF
  210     CONTINUE
          IP(K) = IPK
C
          IF( AMAX.GT.EPS )  THEN
             IF( IPK.NE.K )  THEN
                W = A(IPK,K)
                A(IPK,K) = A(K,K)
                A(K,K) = W
             END IF
C
C ****************************************
             DO 220 I = K+1, N
  220          A(I,K) = -A(I,K)/A(K,K)
C ****************************************
C
             DO 230 J = K+1, MIN(M*KB,N)
               IF( IPK.NE.K )  THEN
                  W = A(IPK,J)
                  A(IPK,J) = A(K,J)
                  A(K,J) = W
               END IF
C
               T = A(K,J)
               DO 240 I = K+1, N
  240            A(I,J) = A(I,J)+A(I,K)*T
  230        CONTINUE
C
          ELSE
             IR = IR+1
C            IP(K)=0
             RETURN
          END IF
  200   CONTINUE
C
 1000 CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCC
      SUBROUTINE GLUB ( N, M, A, IP, EPS, IR, WK )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(N,N), IP(N), WK(N)
C
      IR = 0
      KBEND = (N-1)/M+1
      DO 100 KB = 1, KBEND
C
        KB1 = (KB-1)*M+1
        DO 110 KK = 1, KB-1
          KK1 = (KK-1)*M+1
          CALL GLUBA ( N, M, A(1,KK1), A(1,KB1), IP, KK, KB, WK )
  110   CONTINUE
```

```
C
        CALL GLUBB ( N, M, A(1,KB1), IP, EPS, IR, KB, WK )
  100 CONTINUE
      RETURN
      END
C
      SUBROUTINE GLUBA ( N, M, AA, AB, IP, KK, KB, WK )
      IMPLICIT REAL*8 (A-H,O-Z)
CC    DIMENSION AA(N,*), AB(N,*), IP(N), WK(N)
      DIMENSION AA(N,M), AB(N,M), IP(N), WK(N)
C
      DO 100 K = (KK-1)*M+1, KK*M
        K0 = K-(KK-1)*M
        IPK = IP(K)
C
        DO 110 I = K+1, N
  110     WK(I) = AA(I,K0)
        DO 120 J0 = 1, MIN(KB*M,N)-(KB-1)*M
C
          IF( IPK.NE.K )  THEN
             W = AB(IPK,J0)
             AB(IPK,J0) = AB(K,J0)
             AB(K,J0) = W
          END IF
C
          T = AB(K,J0)
          DO 130 I = K+1, N
  130       AB(I,J0) = AB(I,J0)+WK(I)*T
  120   CONTINUE
  100 CONTINUE
      RETURN
      END
C
      SUBROUTINE GLUBB ( N, M, AB, IP, EPS, IR, KB, WK )
      IMPLICIT REAL*8 (A-H,O-Z)
CC    DIMENSION AB(N,*), IP(N), WK(N)
      DIMENSION AB(N,M), IP(N), WK(N)
C
      JB = M*(KB-1)
      DO 200 K = JB+1, MIN(M*KB,N)
C
C     pivoting
        K0 = K-JB
        AMAX = ABS(AB(K,K0))
        IPK = K
        DO 210 I = K+1, N
          AIK = ABS(AB(I,K0))
          IF( AIK.GT.AMAX )  THEN
             IPK = I
             AMAX = AIK
          END IF
  210   CONTINUE
        IP(K) = IPK
C
        IF( AMAX.GT.EPS )  THEN
           IF( IPK.NE.K )  THEN
              W = AB(IPK,K0)
              AB(IPK,K0) = AB(K,K0)
              AB(K,K0) = W
           END IF
C
C ****************************************
           DO 220 I = K+1, N
             AB(I,K0) = -AB(I,K0)/AB(K,K0)
  220        WK(I) = AB(I,K0)
C ****************************************
C
```

```
            DO 230 J0 = K-JB+1, MIN(M*KB,N)-JB
              IF( IPK.NE.K )  THEN
                 W = AB(IPK,J0)
                 AB(IPK,J0) = AB(K,J0)
                 AB(K,J0) = W
              END IF
C
              T = AB(K,J0)
              DO 240 I = K+1, N
  240           AB(I,J0) = AB(I,J0)+WK(I)*T
  230       CONTINUE
C
         ELSE
            IR = IR+1
C           IP(K)=K
C           DO 250 I=K+1,N
C 250       AB(I,K0)=0.0D0
            RETURN
         END IF
  200 CONTINUE
      RETURN
      END
CCC
      SUBROUTINE GLU (N, A, IP, EPS, IR)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(N,N), IP(N)
C
C forward elimination for A
      IR = 0
      DO 100 K = 1, N
C
C     pivoting
         AMAX = ABS(A(K,K))
         IPK = K
         DO 110 I = K+1, N
           AIK = ABS(A(I,K))
           IF( AIK.GT.AMAX )  THEN
              IPK = I
              AMAX = AIK
           END IF
  110    CONTINUE
         IP(K) = IPK
C
         IF( AMAX.GT.EPS )  THEN
            IF( IPK.NE.K )  THEN
               W = A(IPK,K)
               A(IPK,K) = A(K,K)
               A(K,K) = W
            END IF
C
            DO 120 I = K+1, N
              A(I,K) = -A(I,K)/A(K,K)
  120       CONTINUE
C
            DO 130 J = K+1, N
              IF( IPK.NE.K )  THEN
                 W = A(IPK,J)
                 A(IPK,J) = A(K,J)
                 A(K,J) = W
              END IF
C
              T = A(K,J)
              DO 140 I = K+1, N
                A(I,J) = A(I,J)+A(I,K)*T
  140         CONTINUE
  130       CONTINUE
```

```
C
          ELSE
             IR = IR+1
             RETURN
          END IF
  100 CONTINUE
      RETURN
      END
C
      SUBROUTINE GSLV ( N, A, B, IP )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(N,N), B(N), IP(N)
C
C forward elimination for B
      DO 100 K = 1, N
C
        IF( IP(K).NE.K )  THEN
           W = B(IP(K))
           B(IP(K)) = B(K)
           B(K) = W
        END IF
C
        T = B(K)
        DO 110 I = K+1, N
  110     B(I) = B(I)+A(I,K)*T
  100 CONTINUE
C
C backward substitution for B
      B(N) = B(N)/A(N,N)
      DO 200 K = N-1, 1, -1
        T = B(K+1)
        DO 210 I = 1, K
  210     B(I) = B(I)-A(I,K+1)*T
        B(K) = B(K)/A(K,K)
  200 CONTINUE
      RETURN
      END
CCC
      SUBROUTINE CLOCK0
      DIMENSION IA(3)
      COMMON /CLO/ IB(3)
C
        CALL ITIME(IA)
        DO 200 I=1,3
  200     IB(I)=IA(I)
CC        WRITE(6,*) (IB(I),I=1,3)
       RETURN
       END
CCC
      SUBROUTINE CLOCK(CPUT)
      DIMENSION IA(3)
      COMMON /CLO/ IB(3)
C
        CALL ITIME(IA)
        IJI= IA(1)-IB(1)
        IFUN=IA(2)-IB(2)
        IBYO=IA(3)-IB(3)
        CPUT=IJI*3600.0 +IFUN*60.0 +IBYO*1.0
CC        WRITE(6,*) (IB(I),I=1,3),(IA(I),I=1,3),CPUT
C
        CALL ITIME(IA)
        DO 200 I=1,3
  200     IB(I)=IA(I)
      RETURN
      END
```

4 摂動解析 (連立方程式の) と条件数

$\mathbf{A}(a_{i,j}) \cdot \mathbf{U}(u_i) = \mathbf{F}(f_i)$ の n 行 n 列の連立一次方程式において、右辺 $\mathbf{F}$ や $\mathbf{A}$ がちょっと変わった時、解 $\mathbf{U}$ がどのように変わるか調べることを摂動解析 (あるいは感度解析) という。これは、計算機による有限桁計算に伴う誤差の問題とは独立の重要課題である (文献 [6])。ここでの熱方程式 ($\mathbf{A}$ が対称) を題材にとり、村田の文献 [6] にそっくりならって摂動解析をしてみる。

いま、Fig.1 の両側の伝導率κに段差をつけるための DF 値 (MJ=1 の時)：

$i = 1 \quad \rightarrow 10 \qquad \kappa_i(x,y) = DF$(一定値)

$i = 101 \rightarrow 110 \qquad \kappa_i(x,y) = DF$(一定値)

それ以外は、$\kappa_i(x,y) = 1.0$ とする。

を、DF=1.0, 0.1 の 2 通りに変え ($\mathbf{A}$ を変えること)、熱源 $\mathbf{F}$ を正と負の両熱源とも 0.5 %正方向に (MJ=1 の時)：

$i = 42 \rightarrow 44$, $i = 52 \rightarrow 54 \qquad f_i' = 0.2 + 0.001 \quad [\frac{cal}{sec}]$

$i = 46 \rightarrow 48$, $i = 56 \rightarrow 58 \qquad f_i' = -0.2 + 0.001 \quad [\frac{cal}{sec}]$

ふらせた「一様正 (f+−)」ふらせたものと、「正負交互 (f++)」に：

$i = 42 \rightarrow 44$, $i = 52 \rightarrow 54 \qquad f_i' = 0.2 + 0.001 \quad [\frac{cal}{sec}]$

$i = 46 \rightarrow 48$, $i = 56 \rightarrow 58 \qquad f_i' = -0.2 - 0.001 \quad [\frac{cal}{sec}]$

ふらせた時に、それぞれの場合で解 u_{max}, u_{min}がどう変わるかを見る。結果は $MJ = 1$ とした時、Table 2 となる。この表で、$\mathbf{A} \cdot \mathbf{u} = \mathbf{F}$ と摂動入り式: $\mathbf{A} \cdot \mathbf{u_s} = \mathbf{F_s}$に対して、$\mathbf{\Delta u} = \mathbf{u_s} - \mathbf{u}$ として、相対誤差 $\mathbf{RERR} = \frac{\|\mathbf{\Delta u}\|_2}{\|\mathbf{u}\|_2}$を計算する。ここで $\| \mathbf{u} \|_2 = \langle \vec{u}, \vec{u} \rangle^{\frac{1}{2}} = (\sum_{i=1}^{n} u_i^2)^{\frac{1}{2}}$。また、f+0 は元の式の解、f+−は f を正負交互にふらせた場合、f++は f を一様正にふらせた場合の解を示す。

Table 2 Perturbation Analysis for f +0, +−0.001, ++0.001 (DF=1, 0.1; MJ=1)

f+Δ	DF=1 u_{min}	DF=1 u_{max}	RERR	DF=0.1 u_{min}	DF=0.1 u_{max}	RERR
f+0	-0.35256	0.21374	0.0	-0.48888	0.16532	0.0
f+−	-0.35433	0.21481	0.50 %	-0.49133	0.16615	0.50 %
f++	-0.34978	0.21605	1.08 %	-0.48410	0.16777	1.34 %

DF=1 の時 (MJ=1)、摂動なしの元の $\mathbf{u}$ に対して、$\| \mathbf{u} \|_2$= 1.977。一方 DF=0.1 の時には、$\| \mathbf{u} \|_2$= 5.414 となる。

この時、DF=1(左)、DF=0.1(右) で、右辺 f を、元の式、正負交互、一様正にふらせた時の温度分布 u の等高線は、Fig.6 となる。

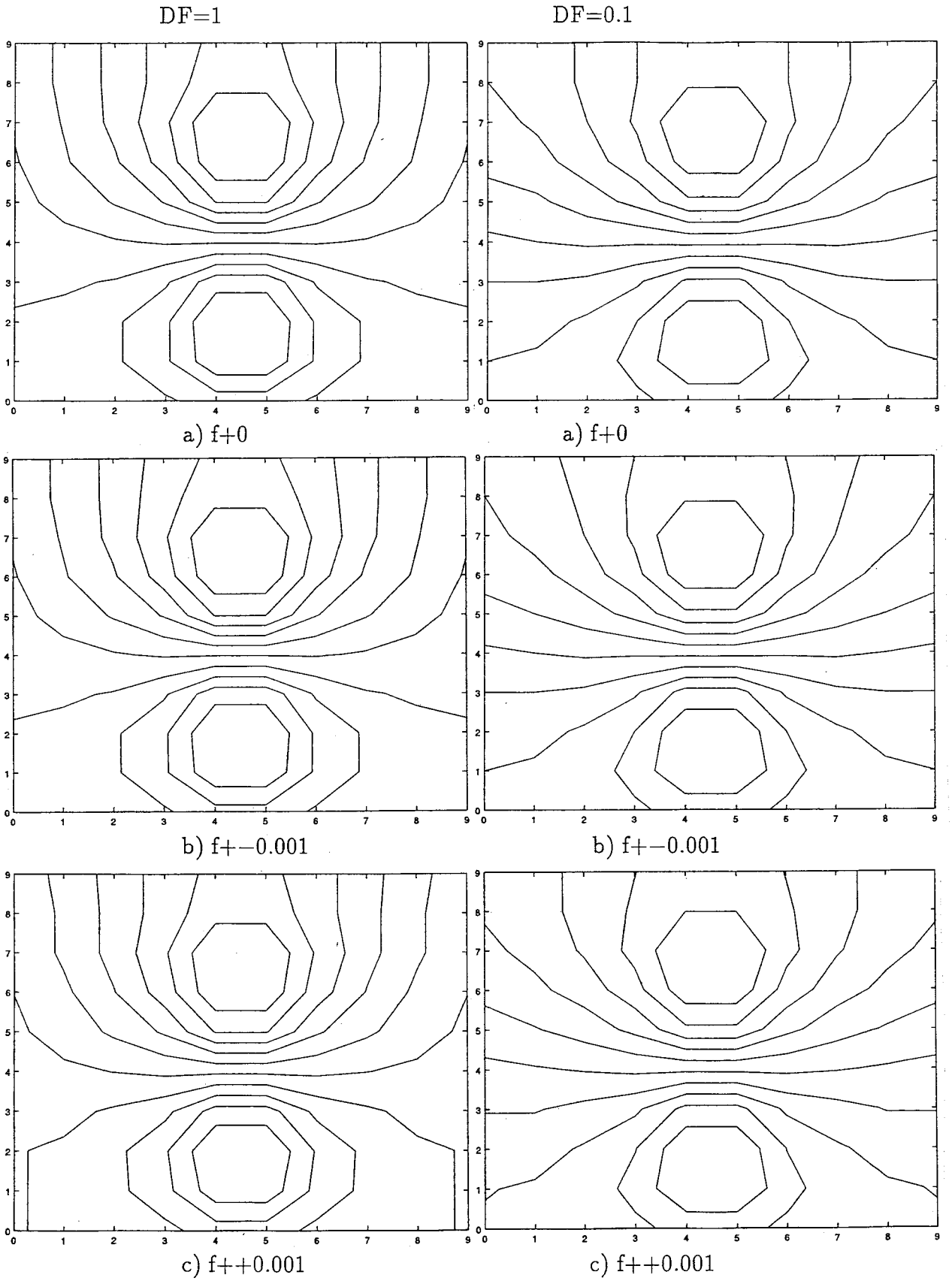


Fig.6. u_i distribution by f+0, f+−0.001, f++0.001 (DF=1, DF=0.1)

Table 2 によれば、

(1) DF 値が0.1 の時の方が、1.0 の時より相対誤差 RERR が (特に一様正で) 大きい (**u** の変動が大きい)。

(2) 同じ DF 値に対しては、熱源 **f** を一様に正方向にふらせた方が正負交互にふらせた時よりも相対誤差 RERR が大きい。

この違いは、Fig.6 のように等高線上では、大きく現れない (DF=1.0 も 0.1 も)。この理由を解説する (文献 [6] そのまま)。

方程式：

$\mathbf{Au} = \mathbf{F}$ と $\mathbf{A}(\mathbf{u} + \delta\mathbf{u}) = \mathbf{F} + \delta\mathbf{F}$

から (辺々引き算して)、

$\mathbf{A}\delta\mathbf{u} = \delta\mathbf{F}$ よって $\delta\mathbf{u} = \mathbf{A}^{-1}\delta\mathbf{F}$

ここでノルム (1 ノルムでも 2 ノルムでも) をとって、

$$\| \delta\mathbf{u} \| \leq \| \mathbf{A}^{-1} \| \cdot \| \delta\mathbf{F} \| \tag{4.1}$$

また、$\mathbf{Au} = \mathbf{F}$ のノルムをとって、

$$\| \mathbf{A} \| \cdot \| \mathbf{u} \| \geq \| \mathbf{F} \| \tag{4.2}$$

式 (4.1)(4.2) から、

$$\frac{\| \delta\mathbf{u} \|}{\| \mathbf{A} \| \cdot \| \mathbf{u} \|} \leq \frac{\| \mathbf{A}^{-1} \| \cdot \| \delta\mathbf{F} \|}{\| \mathbf{F} \|} \tag{4.3}$$

よって、次の公式が得られる。

$$\frac{\| \delta\mathbf{u} \|}{\| \mathbf{u} \|} \leq \| \mathbf{A} \| \cdot \| \mathbf{A}^{-1} \| \cdot \frac{\| \delta\mathbf{F} \|}{\| \mathbf{F} \|} \tag{4.4}$$

ここに現れた、$\| \mathbf{A} \| \cdot \| \mathbf{A}^{-1} \|$ のことを行列 **A** の条件数という。特に **A** 自身が対称正定値の時には、2 ノルムにおいて、

$\| \mathbf{A} \|_2 = \lambda_{max}(\mathbf{A})$ ： **A** の最大固有値

となるので、条件数は、はぎれよく、

$$\| \mathbf{A} \|_2 \cdot \| \mathbf{A}^{-1} \|_2 = \frac{\lambda_{max}(\mathbf{A})}{\lambda_{min}(\mathbf{A})} \tag{4.5}$$

としてよい。

式 (4.4) から、条件数が大きい時、右辺 **F** の相対的変動に対し、解 **u** の相対的変動が大きくなり得るのが分かる。これが先の (1) の理由である。式 (4.5) の計算によれば、DF=1 の時、条件数は33、DF=0.1 の時には、条件数は87 となっている。対称正定値の **A** の $\lambda_{max}(\mathbf{A})$ はベキ乗法で概算値が計算できる。一方、対称正定値の **A** の $\lambda_{min}(\mathbf{A})$ は、$\frac{1}{\lambda_{min}(\mathbf{A})} = \lambda_{max}(\mathbf{A}^{-1})$ より、$\mathbf{A}^{-1}$ に対するベキ乗法で概算値が計算できる。このベキ乗法を使用した、条件数の計算プログラムは後で示す。

ところで条件数だけでは、(2) の事実の説明がつかない。(2) の説明は次のようにする (文献 [6] そのまま)：右辺 $\mathbf{F}$ と$\delta\mathbf{F}$ をそれぞれ $\mathbf{A}$ の固有ベクトルによって展開し、

$$\mathbf{F} = \beta_1 \boldsymbol{v}_1 + \beta_2 \boldsymbol{v}_2 + \cdots + \beta_n \boldsymbol{v}_n$$

$$\delta\mathbf{F} = \delta_1 \boldsymbol{v}_1 + \delta_2 \boldsymbol{v}_2 + \cdots + \delta_n \boldsymbol{v}_n$$

とおく。このとき $\mathbf{u} = \mathbf{A}^{-1}\mathbf{F}$, $\delta\mathbf{u} = \mathbf{A}^{-1}\delta\mathbf{F}$ から、

$$\| \delta\mathbf{u} \|_2^2 = \langle \delta\mathbf{u}, \delta\mathbf{u} \rangle = \frac{\delta_1^2}{\lambda_1^2} + \cdots + \frac{\delta_n^2}{\lambda_n^2}$$

$$\| \mathbf{u} \|_2^2 = \langle \mathbf{u}, \mathbf{u} \rangle = \frac{\beta_1^2}{\lambda_1^2} + \cdots + \frac{\beta_n^2}{\lambda_n^2} \quad ; \ \lambda_1(\text{大}) > \cdots > \lambda_n(\text{小})$$

この 2 式から、

{$\mathbf{F}$ が高次の$\boldsymbol{v}_i$成分を優勢に含み、
$\delta\mathbf{F}$ が低次の$\boldsymbol{v}_i$成分を優勢に含む} とき、$\frac{\|\delta\mathbf{u}\|_2}{\|\mathbf{u}\|_2}$ が大。

今の場合、$\delta\mathbf{F}$ において、一様正の方が (正負交互に比べ)、最小固有値に対応する$\boldsymbol{v}_n$を濃厚に含んでいる。

一様正に対応する解 $\mathbf{u}_s$の変動が大きい、という先の (2) の理由は、そのためと理解できる。極端に、

$\mathbf{F} = \beta_1 \boldsymbol{v}_1$, $\delta\mathbf{F} = \delta_n \boldsymbol{v}_n$ (ただし $\lambda_1(\text{大}) > \cdots > \lambda_n(\text{小})$ とした。)

のとき、

$$\frac{\| \delta\mathbf{u} \|_2}{\| \mathbf{u} \|_2} = \frac{\lambda_1}{\lambda_n} \frac{| \delta_n |}{| \beta_1 |} = \frac{\lambda_1}{\lambda_n} \frac{\| \delta\mathbf{F} \|_2}{\| \mathbf{F} \|_2} \tag{4.6}$$

となる。この時が最悪である。$\frac{\lambda_1}{\lambda_n}$は、ちょうど行列 $\mathbf{A}$ の条件数になっている。

$\delta\mathbf{F}$ による、$\frac{\|\delta\mathbf{u}\|_2}{\|\mathbf{u}\|_2}$ の変動は、条件数が大きい方が大きくなる。また、正負交互の$\delta\mathbf{F}$ でも、相対誤差がある程度あることから、$\mathbf{F}$ が最大固有値に対応する固有ベクトル$\boldsymbol{v}_1$を多く含んでいると想像できる。

なお、対称正定値行列 $\mathbf{A}$ の最大固有値$\lambda_{max}(\mathbf{A})$ は、ベキ乗法によって (概算でよければ) 簡単に計算される (文献 [6] そのまま)。その原理は次の通り：任意の初期ベクトル$\boldsymbol{x}^{(0)}$を選ぶ。ここで$\lambda_{max}(\mathbf{A})$ を求めるための初期ベクトルは、

$$\boldsymbol{x}^{(0)} = (1, -1, 1, -1, \cdots, -1, 1)^T$$

を使うとよい。これは弦の基本振動を思いうかべればよい。

変動が激しい (正負交互)→高次モードを濃厚に含む→ 最大固有値$\lambda_{max} = \lambda_1$に対応する固有ベクトル$\boldsymbol{v}_1$に近い

という図式である。

$\boldsymbol{x}^{(0)}$ を固有ベクトル $\{\boldsymbol{v}_i : \ i = 1, \cdots, n\}$ によって展開するとき、

$$\boldsymbol{x}^{(0)} = c_1 \boldsymbol{v}_1 + c_2 \boldsymbol{v}_2 + \cdots + c_n \boldsymbol{v}_n$$

となっているとする。$\mathbf{A}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$ に留意し、

$$\boldsymbol{x}^{(l)} = \mathbf{A}^l \boldsymbol{x}^{(0)}$$

を考えると、$\mathbf{A}^l \boldsymbol{v}_i = \lambda_i^l \boldsymbol{v}_i$ を使って、

$$\boldsymbol{x}^{(l)} = c_1 \lambda_1^l \boldsymbol{v}_1 + c_2 \lambda_2^l \boldsymbol{v}_2 + \cdots + c_n \lambda_n^l \boldsymbol{v}_n$$

となる。いま、$\lambda_{max} = \lambda_1 > \cdots > \lambda_n = \lambda_{min}$, $c_1 \neq 0$ とすると、

$$\boldsymbol{x}^{(l)} = c_1\lambda_1^l\left[\boldsymbol{v}_1 + \frac{c_2}{c_1}\left(\frac{\lambda_2}{\lambda_1}\right)^l\boldsymbol{v}_2 + \cdots + \frac{c_n}{c_1}\left(\frac{\lambda_n}{\lambda_1}\right)^l\boldsymbol{v}_n\right]$$

これで分かるように、$l \to \infty$ のとき$\boldsymbol{x}^{(l)}$ は$\boldsymbol{v}_1$の向きに'方向収束'する。そこで、各$l \to 1 \to l$ 毎に、

$$\tilde{\boldsymbol{x}}^{(l)} = \boldsymbol{x}^{(l-1)} \diagup \| \boldsymbol{x}^{(l-1)} \|_2$$

$$\boldsymbol{x}^{(l)} = \mathbf{A}\tilde{\boldsymbol{x}}^{(l)} \tag{4.7}$$

$$\lambda^{(l)} = \langle \tilde{\boldsymbol{x}}^{(l)}, \boldsymbol{x}^{(l)} \rangle$$

を行うと、$l \to \infty$ のとき

$\tilde{\boldsymbol{x}}^{(l)} \to \boldsymbol{v}_1$, $\lambda^{(l)} \to \lambda_1 (= \lambda_{max})$

となる。概算でよいから、λ_{max}のためには、反復2回で十分のはず。以下に、λ_{max}を計算するプログラムを示す。

```
      SUBROUTINE RAMMAX(N,M,X,X1,RMAX,AA,AB,AC)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION AA(N),AB(-M:N),AC(-M:N),
     *          X(-M:N+M),X1(N)
C
      DO 300 I=1,N
        IF(MOD(I,2).EQ.1) THEN
          X1(I)=1.0D0
        ELSE
          X1(I)=-1.0D0
        ENDIF
  300 CONTINUE
C
      LMAX=2
      DO 310 L=1,LMAX
        XNRM=0.0D0
        DO 320 I=1,N
  320     XNRM=XNRM+X1(I)*X1(I)
        XNRM=DSQRT(XNRM)
C
        DO 330 I=1,N
  330     X(I)=X1(I)/XNRM
        DO 340 I=1,N
  340     X1(I)=AC(I-M)*X(I-M)+AB(I-1)*X(I-1)
     *         +AA(I)*X(I)+AB(I)*X(I+1)+AC(I)*X(I+M)
C
        RMAX=0.0D0
        DO 350 I=1,N
  350     RMAX=RMAX+X(I)*X1(I)
C
  310 CONTINUE

      RETURN
      END
```

一方、最小固有値$\lambda_{min}(\mathbf{A})$については、

$1 \diagup \lambda_{min}(\mathbf{A}) = \lambda_{max}(\mathbf{A}^{-1})$

であることを利用して、$\mathbf{A}^{-1}$に対するベキ乗法、すなわち(4.7)式の代わりに、

$$\boldsymbol{x}^{(l)} = \mathbf{A}^{-1}\tilde{\boldsymbol{x}}^{(l)} \tag{4.8}$$

を行なうと考えて、それを、

$$solve \quad \mathbf{A}\boldsymbol{x}^{(l)} = \tilde{\boldsymbol{x}}^{(l)}$$

によって実行する。$l = 1, 2, \cdots$ に対し、同じ $\mathbf{A}$ に対して解くわけゆえ、初めに、(前版の)数値シミュレーションの基礎 (1)「3 (対称) 帯ガウス (BSGLU)」の BSGLU を 1 回行なうだけで、後は右辺$\tilde{\boldsymbol{x}}^{(l)}$ に対する BSGSLV を繰り返すだけである。この手法を逆反復法といい、最小固有値$\lambda_{min}(\mathbf{A})$ の簡単な (概算) 計算に用いる。

この時、$\lambda_{min}(\mathbf{A})$ を求めるための初期ベクトル$\boldsymbol{x}^{(0)}$は、

$$\boldsymbol{x}^{(0)} = (1, 1, 1, 1, \cdots, 1, 1)^T$$

が良い。弦の基本振動との類推で、

変動が穏やか→低次モードを濃厚に含む→ 最小固有値$\lambda_{min} = \lambda_n$に対応する固有ベクトル$\boldsymbol{v}_n$に近い

という図式である。概算でよいから、λ_{min}のためには、反復 1 回で十分のはず。以下に、λ_{min}を計算するプログラムを示す。プログラム中で、X(I) は$\tilde{\boldsymbol{x}}^{(l)}$ に、X1(I) は$\boldsymbol{x}^{(l)}$ に対応している (I=l)。

```
      SUBROUTINE RAMMIN(N,M,X,X1,RMIN,AA,AB,AC,AR,EPS,IR)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION AA(N),AB(-M:N),AC(-M:N),AR(0:M,N),
     *          X(-M:N+M),X1(N)
C
      DO 300 I=1,N
  300   X1(I)=1.0D0
C
      LMIN=1
      DO 310 L=1,LMIN
        XNRM=0.0D0
        DO 320 I=1,N
  320     XNRM=XNRM+X1(I)*X1(I)
        XNRM=DSQRT(XNRM)
C
        DO 330 I=1,N
  330     X(I)=X1(I)/XNRM
        DO 340 I=1,N
  340     X1(I)=X(I)
        CALL BSGSLV(N,M,AR,X1)
C
        RMIN=0.0D0
        DO 350 I=1,N
  350     RMIN=RMIN+X(I)*X1(I)
C
  310 CONTINUE
CC
      RMIN=1.0D0/RMIN
C
      RETURN
      END
```

したがって、条件数 COND は、次のように求まる (正負交互の場合を、対称帯ガウス BSGLU で解いている例)。

```
*C    Uhen ***
      DO 60 J= 5*MJ-1, 6*MJ-1
        DO 70 I=0,2*MJ
          F(M*J +2*MJ+I)= 0.2D0*(DHX*DHY)
          F(M*J +6*MJ+I)=-0.2D0*(DHX*DHY)
   70   CONTINUE
   60 CONTINUE
```

```
C
      CALL CLOCK0
C
*C    AR*U=F wo Toku; Kotae --> F ***
      CALL BSGLU(N,M,AR,EPS,IR)
      CALL BSGSLV(N,M,AR,F)
C
C     (SETUDOU)
      DO 62 J= 5*MJ-1, 6*MJ-1
        DO 72 I=0,2*MJ
          FS(M*J +2*MJ+I)= (0.2D0+ 0.001D0)*(DHX*DHY)
          FS(M*J +6*MJ+I)=(-0.2D0- 0.001D0)*(DHX*DHY)
   72   CONTINUE
   62 CONTINUE
C
      CALL BSGSLV(N,M,AR,FS)
C
      DFNRM=0.0D0
      FNRM=0.0D0
      DO 570 I=1,N
        DFNRM= DFNRM+ (FS(I)-F(I))**2
  570   FNRM=FNRM + F(I)**2
      GOSA=DSQRT(DFNRM)/DSQRT(FNRM)
C
      WRITE(6,*) ' %% GOSA, FNRM = ',GOSA, FNRM

*     Eigen value ***
*     (MAX)
      CALL RAMMAX(N,M,X,X1,RMAX,AA,AB,AC)
C
      WRITE(6,*) ' %% RAMDA.MAX= ',RMAX
CC      DO 180 K=10,1,-1
CC 180    WRITE(6,2000) (X(10*J+K),J=0,9),' ; '
CC
*     (MIN)
      CALL RAMMIN(N,M,X,X1,RMIN,AA,AB,AC,AR,EPS,IR)
C
      WRITE(6,*) ' %% RAMDA.MIN= ',RMIN
CC      DO 280 K=10,1,-1
CC 280    WRITE(6,2000) (X(10*J+K),J=0,9),' ; '
CC
      COND=RMAX/RMIN
      WRITE(6,*) ' %% COND.NUM= ',COND
```

結果は、Table 3 のようになる。

Table 3 Condition Number for DF=1, 0.1 (MJ=1)

	DF=1	DF=0.1
λ_{max}	4.13 (2 times)	3.76 (2 times)
λ_{min}	0.121 (1 times)	4.273D-2 (1 times)
Condition Number	33.9	87.9

まじめに条件数を求めるならば、λ_{max}, λ_{min}の収束状況を、収束判定値を決めて決定しなければならない。また、λ_{max}, $1/\lambda_{min}$を求めるための初期値 $x^{(0)}$ は乱数を使って作り、アルゴリズムは次のようになる (プログラムはICCG 法を使って書いてある)。

λ_{max}のための$\boldsymbol{x}^{(0)}$ 初期値を作る (乱数:CALL RANDOM)
$\lambda_{max}^{(0)} = 0$
$|\lambda_{max}^{(l)} - \lambda_{max}^{(l-1)}|/\lambda_{max}^{(l)} >$ EPSEIG(=1.0D-3)

$\tilde{\boldsymbol{x}}^{(l)} = \boldsymbol{x}^{(l-1)}/\|\boldsymbol{x}^{(l-1)}\|_2$ 作る
$\boldsymbol{x}^{(l)} = \boldsymbol{A}\tilde{\boldsymbol{x}}^{(l)}$ 作る
$\lambda_{max}^{(l)}$(=LAM1)=$< \tilde{\boldsymbol{x}}^{(l)}, \boldsymbol{x}^{(l)} >$

$\frac{1}{\lambda_{min}}$のための$\boldsymbol{x}^{(0)}$ 初期値を作る (乱数:CALL RANDOM)
$\frac{1}{\lambda_{min}^{(0)}} = \lambda_{max}$
$|\frac{1}{\lambda_{min}^{(l)}} - \frac{1}{\lambda_{min}^{(l-1)}}|/\frac{1}{\lambda_{min}^{(l)}} >$ EPSEIG(=1.0D-3)

$\tilde{\boldsymbol{x}}^{(l)} = \boldsymbol{x}^{(l-1)}/\|\boldsymbol{x}^{(l-1)}\|_2$ 作る
$\boldsymbol{x}^{(l)} = \boldsymbol{A}^{-1}\tilde{\boldsymbol{x}}^{(l)}$ [solve $\boldsymbol{A}\boldsymbol{x}^{(l)} = \tilde{\boldsymbol{x}}^{(l)}$] を解く (CALL ICCG12)
$\frac{1}{\lambda_{min}^{(l)}}$(=LAM2)=$< \tilde{\boldsymbol{x}}^{(l)}, \boldsymbol{x}^{(l)} >$

cond=$\lambda_{max} \cdot \frac{1}{\lambda_{min}}$ (=LAM1$\times$ LAM2: 条件数)

プログラムレベルでは、ICCG 法を使って次のようになる。先のアルゴリズムに相当する所がサブルーチン:CONDCG

```
      CALL CONDCG( N, M, AA, AB, AC, AE, BW, DI,
     +             P, R, W, X2, X1, AP, COND )
         WRITE(*,'(1H ,A,3D15.5)') ' COND :', COND
      WRITE(6,*) ' %% RAMDA.MAX= ',COND(1)
      WRITE(6,*) ' %% RAMDA.MIN= ',1.0D0/COND(2)
      WRITE(6,*) ' %% COND.NUM= ',COND(3)
```

になる。

```
      SUBROUTINE CONDCG( N, M1, AA, AB, AC, AE, BW, DINV, P, R, W,
     +                   X2, X1, AP, COND )
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION AA(N), AB(-M1:N), AC(-M1:N), AE(-M1:N), DINV(-M1:N),
     +          P(-M1:N+M1), W(-M1:N+M1), X2(N), X1(-M1:N+M1),
     +          AP(N), R(N), BW(-M1:N), COND(3)
      REAL *8 LAM1, LAM2
      DATA KKE1, KKE2, EPSEIG, UP / 50, 20, 1.0D-3, 0.95 /
*
      CALL ICDCMP( N, M1, AA, AB, AC, AE, BW, DINV, UP )
*
```

```
      LAM1 = 0.0D0
      CALL RANDOM( N, X2 )
      DO 100 K0 = 1, KKE1
        S = 0.0
        DO 110 I = 1, N
          X1(I) = X2(I)
  110     S = S + X1(I)*X1(I)
        DO 120 I = 1, N
  120     X1(I) = X1(I)/SQRT(S)
        CALL MAKEAX( N, M1, AA, AB, AC, X1, X2 )
        S = 0.0
        DO 130 I = 1, N
  130     S = S + X1(I)*X2(I)
        RERR = ABS(S-LAM1)/S
        LAM1 = S
        IF( RERR.LT.EPSEIG )  GO TO 1001
  100 CONTINUE
*
 1001 CALL RANDOM( N, X1(1) )
CC    WRITE(*,*) ' MAX : ', K0, LAM1
      LAM2 = LAM1
      RERR = 1.0
      DO 200 K0 = 1, KKE2
        S = 0.0
        DO 210 I = 1, N
          X2(I) = X1(I)
  210     S = S + X2(I)*X2(I)
        DO 220 I = 1, N
          X2(I) = X2(I)/SQRT(S)
  220     X1(I) = X2(I)*LAM2
        BNORM = 1.0D0
        EPSL1 = MAX( MIN( RERR*1.0D-2, 1.0D-3 ), 1.0D-8 )
        CALL ICCG12( AA, AB, AC, AE, BW, DINV, P, R, W, X2, X1, AP,
     +               EPSL1, ERR, M1, N, KCOUNT, BNORM )
        S = 0.0
        DO 230 I = 1, N
  230     S = S+X1(I)*X2(I)
        RERR = ABS(S-LAM2)/S
        LAM2 = S
        IF( RERR.LT.EPSEIG )  GO TO 2001
  200 CONTINUE
*
 2001 COND(3) = LAM1*LAM2
      COND(1) = LAM1
      COND(2) = LAM2
CC    WRITE(*,*) ' MIN  : ', K0, LAM2
CC    WRITE(*,*) ' COND : ', 0, COND(3)
      RETURN
      END
CCC*
      SUBROUTINE RANDOM( N, Y )
      REAL *8 Y(N)
      DATA IBP31M,BNM31 /2147483647,Z39200000/
      DATA IVURN,IFP11  /584287,48828125/
*
      DO 100 I = 1, N
      IVURN = IVURN*IFP11
      IF( IVURN.LT.0 ) IVURN = (IVURN+IBP31M)+1
      X = FLOAT(IVURN)*BNM31
      Y(I) = X
  100 CONTINUE
      RETURN
      END
```

このサブルーチン CONDCG を使うと、Table 3 は次表になる。

Table 4 Condition Number for DF=1, 0.1 (MJ=1) BY CONDCG

	DF=1	DF=0.1
λ_{max}	7.52 (20 times)	7.629 (40 times)
λ_{min}	0.0951 (5 times)	0.0368 (4 times)
Condition Number	(33→) 79	(87→) 207

たとえば、DF=1 では、λ_{max}のために反復回数が 2 回から 20 回へ、λ_{min}のためには反復回数が 1 回から 5 回へ変わり、条件数は 33 から 79 へ変わる。

また、MJ を増やすと条件数も増える。

Table 5 Condition Number for DF=1, 0.1 BY CONDCG

MJ	DF=1(Condition Number,λ_{max},λ_{min})	DF=0.1
1	79(7.52,0.0951)	207(7.629,0.0368)
2	308(7.77,0.0251)	618(7.77,0.0125)
3	684(7.80,0.0114)	1186(7.80,0.00658)

参考文献

[1] 小国 力編著, 村田 健郎、三好 俊郎、ドンガラ J,J、長谷川 秀彦著、行列計算ソフトウェア (WS、スーパーコン、並列計算機), 丸善,pp.252-275,Nov.1991

[2] 村田 健郎,CV 法と、手作り数値シミュレーションシステムの奨め, 日本計算工学会, 計算工学 vol.3,No.1,pp.44-53,1998

[3] 村田 健郎,CV 法と、手作り数値シミュレーションシステムの奨め (第 2 回)-非線形純拡散問題と割線反復法-, 日本計算工学会, 計算工学 vol.3,No.3

[4] 村田 健郎,CV 法と、手作り数値シミュレーションシステムの奨め (第 3 回)-移流拡散系の離散化：特に指数法について-, 日本計算工学会, 計算工学 vol.3,No.4, pp.246-253,Dec.1998

[5] 村田 健郎,CV 法と、手作り数値シミュレーションシステムの奨め (第 4 回)-連立非線形移流拡散系：半導体デバイス解析の場合-, 日本計算工学会, 計算工学 vol.4,No.2,1999

[6] 村田 健郎,「BASIC 数学」連載：線形数値計算法とその応用, 現代数学社,1992

[7] 村田 健郎, 線形代数と線形計算法序節, サイエンス社,1986

[8] 小国 力著,MATLAB と利用の実際 [第 2 版], サイエンス社,2001

[9] 村田 健郎, 名取 亮, 唐木 幸比古著, 大型数値シミュレーション, 岩波書店,pp.51-137,1990