1996 年度 卒業論文

超高速行列演算チップの開発

9320028

木村・齋藤研　中島 瑞樹

電気通信大学 電子工学科 電子デバイス工学講座

指導教官 齋藤 理一郎 助教授

提出日 平成 9 年 2 月 5 日

概要

$Ax = \lambda x$ を満たす数 λ とベクトル x をそれぞれ固有値・固有ベクトルといい科学技術計算などの計算において、必要不可欠になっているが現在, 行列の計算 (固有値・固有ベクトル) は時間がかかる。そこで計算を早くさせるためソフトウエア又はハードウエアによる解決方法があり、ソフトウエアではプログラムのアルゴリズムや並列計算機を使用した並列化プログラムの作成があるが、本研究ではハードウエアによる方法を用いた。そのため、固有値固有ベクトルを求める為だけの専用プロセッサーを仮想し (これを専用チップと定義する) そのプロセッサー上でどのようにしたら固有値固有ベクトルを求めることができるかを考えるのが本研究の目的である。そしてその過程が正しいか確かめるために C 言語を用いプログラムでシミュレーションすることにした。これにより固有値固有ベクトルを求める際の過程がわかり、どの部分が重要であるか、またこの過程は時間がかかるため専用チップ上で行うようにして速く計算させるなど専用チップの制作へスムーズに移行しやすくなる。

ここではまず固有値固有ベクトルを求める方法としてハウスホルダー変換, 2 分法, そして逆反復法について説明し次に専用チップの概要の説明、最後に専用チップ上でこれらの方法によって固有値・固有ベクトルが求まるよう実現できるかを考える。その結果専用チップ上で求まることが確認できた。

目次

1　序論

本研究では行列の固有値, 固有ベクトルを求める方法としてハウスホルダー変換, 2 分法, そして逆反復法によって求めている。この章ではまずハウスホルダー変換, 2 分法, そして逆反復法について説明する。

対称行列の固有値を計算するとき，通常は行列を三重対角行列に変換し，その三重対角行列の固有値を計算する。その理由は，与えられた行列の固有値を一度に求めるよりも，このように三重対角化を中間におく方が，全体として手間が少なくなるからである。ここでは，三重対角化の方法としてハウスホルダ一変換，そしてその固有値を求める方法として二分法を使うプログラムについて説明する。対称行列の固有ベクトルの計算については次章で述べる。

2　ハウスホルダー変換による行列の三重対角化

$n \times n$ 行列 A に関する固有値問題（elgenvalue problem）

$$A\boldsymbol{x} = \lambda\boldsymbol{x} \tag{2.1}$$

を考える。本章では，行列 A は対称であるとしておく。固有値を求めるために，まず A を相似変換

$$\tilde{A} = P^{-1}AP \tag{2.2}$$

によって三重対角行列に変換する。

$$\tilde{A} = \begin{pmatrix} \alpha_1 & \beta_1 & & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \multicolumn{2}{c}{\multirow{2}{*}{0}} \\ & \beta_2 & \alpha_3 & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & \multicolumn{2}{c}{\multirow{2}{*}{0}} & \ddots & \alpha_{n-1} & \beta_{n-1} \\ & & & & \beta_{n-1} & \alpha_n \end{pmatrix} \tag{2.3}$$

三重対角行列（tridiagonal matrix）とは，(2.3) のように，対角成分およびそれに隣接する副対角成分のみが 0 でない行列のことである．相似変換によって固有値

は変わらないから，A の固有値の代わりにより簡単な計算によって $\tilde{A} = P^{-1}AP$ の固有値を求める，というのがここで述べる方法の考え方である。

三重対角化を行う全体の変換行列 P を構成する前に，まず適当な n 次元ベクトル w を使って

$$Q = I - c\boldsymbol{w}\boldsymbol{w}^T \ , c = \frac{2}{\boldsymbol{w}^T\boldsymbol{w}} \tag{2.4}$$

なる $n \times n$ 行列 Q を定義する。この行列は対称な直交行列であること、すなわち $Q^{-1} = Q^{-T} = Q$ をみたすことは容易に確かめられる。この型の行列 Q による相似変換を，ハウスホルダー（Householder）変換という。行列 Q による A のハウスホルダー変換を考え，これを B とおく。

$$B = Q^{-1}AQ \tag{2.5}$$

三重対角化の第 1 のステップは，B の第 1 列の第 3 行目以下の成分がすべて 0 になるようにすることである。そのようにすると，B の第 1 行の第 3 列目より右の成分も自動的に 0 になる。なぜならば，A は対称であるから，(2.5) より B も対称になるからである。つまり，Q すなわち w をうまく選んで，B を次の形にもってゆくわけである。

$$B = \begin{pmatrix} \alpha_1 & \beta_1 & 0 & \dots & 0 \\ \beta_1 & * & * & \dots & * \\ 0 & * & * & \dots & * \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & * & * & \dots & * \end{pmatrix} \tag{2.6}$$

* は一般には 0 でない成分を表す。

いま，A の第 1 列のベクトルを次のように書く。

$$\boldsymbol{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{pmatrix} \tag{2.7}$$

そして，ハウスホルダー変換を定義する (2.4) のベクトル w を

$$\boldsymbol{w} = \begin{pmatrix} 0 \\ a_2 + s \\ a_3 \\ \vdots \\ a_n \end{pmatrix}, \qquad s^2 = \sum_{j=2}^{n} a_j^2 \tag{2.8}$$

に選んでみる。s の符号は $a_2 + s$ の計算で桁落ちの生じないように a_2 と同符号にとる。このとき,(2.4) の c は次のようになる。

$$c = \frac{1}{s^2 + |sa_2|} \tag{2.9}$$

一方，$\boldsymbol{w}$ を (2.8) のようにとると，(2.4) の Q の第 1 列は第 1 成分が 1 である単位ベクトルとなり，$Q^{-1}A$ の右から 0 を乗ずるときその結果の $Q^{-1}AQ$ の第 1 列は $Q^{-1}A$ の第 1 列と変わらない．したがって，$B = Q^{-1}AQ$ の第 1 列のベクトルを$\boldsymbol{b}$ とおくと，

$$(I - c\boldsymbol{w}\boldsymbol{w}^T)\boldsymbol{a} = \boldsymbol{b} \tag{2.10}$$

が成り立つ。これから直ちに

$$\boldsymbol{b} = \begin{pmatrix} a_1 \\ -s \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{2.11}$$

となり，B が (2.8) の形をもつこと，および

$$\begin{cases} \alpha_1 = a_1 \\ \beta_1 = -s \end{cases} \tag{2.12}$$

となることがわかった。実際に (2.5) の変換を計算するときには，次の関係を利用する。

$$B = (I - cww^T)A(I - cww^T) = A - (wq^T + qw^T) \tag{2.13}$$

ただし，最後の式は，まず

$$P = cAw \tag{2.14}$$

を計算してから

$$q = P - \frac{c}{2}wP^Tw \tag{2.15}$$

を求め，この q を使って計算する。ハウスホルダー変換によって行列が (2.6) の形になったから，次に (2.6) の第 1 行と第 1 列を除いた残りの 小行列の部分について同様の変換を行う。この操作を n-2 回くり返せば，もとの行列 A は最後に (2.3) の形をもつ三重対角行列になるわけである。つまり，第 1 行から第 k-1 行までおよび第 1 列から第 K-1 列まで三重対角化の終了した行列を $A^{(k)}$ とするとき (2.8) と同しように選んだベクトル $w^{(k)}$ から行列 $Q^{(k)} = I - c_k w^{(k)} w^{(k)T}$ を構成し，この $Q^{(k)}$ による相似変換

$$A^{(k+1)} = (Q^{(k)})^{-1} A^{(k)} Q^{(k)} \tag{2.16}$$

を行う。これを n-2 回くり返して

$$\tilde{A} = A^{n-1} = p^{-1}AP \tag{2.17}$$

$$P = Q^{n-2}Q^{n-1} \cdot \cdot \cdot Q^{(1)} \tag{2.18}$$

とすればよい。以上の手順をまとめると，次のようになる。ただし， $A^{(k)}$ の第 ij 成分を $a_i j^k$ と書く。

$A^{(1)} = A$ とおき,$k = 1, 2, 3, \cdot \cdot \cdot, n-2$ について次の手順をくり返す。

(i)

$$s_k = \sqrt{\sum_{j=k+1}^{n} a_{jk}^{(k)2}} \tag{2.19}$$

もしも $a_{k+1,k}^{(k)} < 0$ ならば s_k の符号を負にする。

もしも $s_k = 0$ ならば (ii)(iii) を実行せずに次の k へ進む。

(ii)

$$c = \frac{1}{s_k^2 + a_{k+1,k}^{(k)} \times s_k} \tag{2.20}$$

$$\boldsymbol{w}^{(k)} = \begin{pmatrix} 0 \\ \vdots \\ a_{k+1,k}^{(k)} + s_k \\ a_{k+2,k^{(k)}} \\ \vdots \\ a_{n,k}^{(k)} \end{pmatrix} \tag{2.21}$$

(iii)

$$P^{(k)} = c_k A^{(k)} w_{(k)} \tag{2.22}$$

$$q^{(k)} = P_{(k)} - \frac{c_k}{2} w^{(k)} P^{(k)T} w^{(k)} \tag{2.23}$$

$$A^{(k+1)} = A^k - (w^{(k)} q^{(k)T} + q^{(k)} w^{(k)T}) \tag{2.24}$$

2.1　二分法

固有値を求める次のステップは、もとの行列 A の相似変換である (2.3) の三重対角行列 $\tilde{A}$ の固有値を計算することである。この計算方法として、ここではスツルムの定理に基づく二分法を採用する。

三重対角行列 (2.3) に対応して行列 $\lambda I - \tilde{A}$ を考え、その第 k 主小行列式を $p_k(\lambda)$ とおく。

$$\begin{pmatrix} \lambda-\alpha_1 & -\beta_1 & & & \\ -\beta_1 & \lambda-\alpha_2 & \ddots & & 0 \\ & \ddots & \ddots & \ddots & \\ 0 & & \ddots & \lambda-\alpha_{k-1} & -\beta_{k-1} \\ & & & -\beta_{k-1} & \lambda-\alpha_k \end{pmatrix} \tag{2.25}$$

これを最後の行について展開すると

$$p_k(\lambda) = (\lambda-\alpha_k)p_{k-1}(\lambda) - {\beta_{k-1}}^2 p_{k-2} + (\lambda) \tag{2.26}$$

が得られる。これは λ の多項式列 $\{p_k(\lambda)\}$ に関する漸化式であるが、これが $k = 2$ の時にも成立するように便宜的に

$$p_0(\lambda) = 1 \tag{2.27}$$

と定義しておく。 $k = n$ のとき

$$p_n(\lambda) = \mid \lambda I - \tilde{A} \mid \tag{2.28}$$

となるが、この n 次多項式 $p_n(\lambda)$ の根が $\tilde{A}$ の固有値、すなわち A の固有値に等しい。

ここでは証明しないが、上の多項式の列

$$p_n(\lambda), p_{n-1}(\lambda), \cdots, p_1(\lambda)p_0(\lambda) \tag{2.29}$$

の符号の変化の回数に関して、いわゆるスツルム (Sturm) の定理から次の事実が導かれる。すなわち、 λ を固定して (2.29) を左から右へ見ていったときの符号の変化の回数を $N(\lambda)$ とすると、 $N(\lambda)$ は、 λ より大きい固有値の個数に等しい。

行列 A および $\tilde{A}$ は対称であるから、その固有値 $\lambda_i, i = 1, 2, \cdots$ はすべて実数である。固有値は大きい方から $\lambda_1, \lambda_2, \cdots$ の順に並べるものとする。もしも、二つの数

a_{j-1}, b_{j-1} に関して $N(a_{j-1}) \geq k,\ N(b_{j-1}) < k,$ が成立しているとすると、上に述べたことから、大きい方から第 k 番目の固有値 λ_k は $a_{j-1} < \lambda_k < b_{j-1}$ の間に存在する。この範囲を十分に狭くすることができたならば、 λ_k の近似値が求められたことになる。区間の幅を半分にしながら、次第に λ_k の存在する範囲を狭くしてゆく方法が、二分法 (bisection method) である。

2.2　逆反復法

B を $n \times n$ 行列として、その固有値を $\nu_1, \nu_2, \cdots, \nu_n$ とする。これらの固有値は絶対値の大きい方から順にならんでいて、絶対値最大の固有値には縮退はないものとする。つまり

$$| \nu_1 | > | \nu_2 | \geq | \nu_3 | \geq \cdots \geq | \nu_n | \tag{2.30}$$

をみたしているものとする。このとき、適当な初期ベクトル $^{(0)}$ から出発して、

$$^{(l)} = B\,^{(l-1)} \qquad l = 1, 2 \cdots \tag{2.31}$$

を計算してゆくと、よく知られているように、$^{(l)}$ は次第に B の絶対値最大の固有値 ν_1 に対応する固有ベクトルに近づいてゆく。これがいわゆるべき乗法 (power method) の原理である。

さて与えられた $n \times n$ 行列 $\tilde{A}$ の k 番目の固有値を λ_k、それに対応する固有ベクトルを　　とする。いま、μ を λ_k の適当な近似値として、

$$(\mu I - \tilde{A})^{-1} \tag{2.32}$$

なる行列を考えると、

$$(\mu I - \tilde{A})^{-1} \quad = \frac{1}{| \mu - \lambda_k |} \tag{2.33}$$

が成り立つ。近似値 μ は λ_k に十分近く、他の固有値 $\lambda_j, j \neq k$ とは

$$\frac{1}{| \mu - \lambda_k |} > \frac{1}{| \mu - \lambda_j |} \qquad j \neq k \tag{2.34}$$

なる関係になっているとすると、$1/(\mu - \lambda_k)$ は行列 $(\mu I - \tilde{A})^{-1}$ の絶対値最大の固有値になる。したがって、適当な初期値 $^{(l)}$ から出発して $(\mu I - \tilde{A})^{-1}$ にべき情報を適用し、次々に

$$^{(l)} = (\mu I - \tilde{A})^{-1}\,^{(l-1)} \tag{2.35}$$

を計算してゆくと、$^{(l)}$ は次第に固有ベクトル　　に近づいてゆくことになる。これが逆反復法 (inverse iteration) の原理である。

実際の計算では、(2.35) の反復は連立１次方程式

$$(\mu I - \tilde{A})\,^{(l)} = \,^{(l-1)} \tag{2.36}$$

を $^{(l)}$ について解くことによって進めることが出来る。μ が λ_k に十分近いと (2.35) の収束は速くなるが、一方 μ が λ_k ひじょうに近いと、方程式 (2.36) の解はいわゆる不定の状態に近くなる。しかし、(2.36) を解いて得た解は固有ベクトルとしてはあくまで正しいものになっている。

もしも、固有値 λ_k に縮退があると、(2.35) の反復によって $^{(k)}$ は λ_k に対応する複数個の固有ベクトルの 1 次結合に近づく。したがって、縮退がある場合には、λ_k に対応する 2 番目の固有ベクトルを求める反復からは、すでに求めてある固有ベクトルと直交するような初期ベクトルをとって反復を開始する必要がある。

方程式 (2.36) を解くときには、LU 分解を使うとよい。すなわち、$\mu I - \tilde{A}$ をまず

$$\mu I - \tilde{A} = LU \tag{2.37}$$

のように L と U の積に LU 分解する。このとき、(2.36) は、

$$LU^{\,()} \quad = {}^{(l-1)} \tag{2.38}$$

と書くことができる。この方程式は次のように、

$$L \; = {}^{(l-1)} \tag{2.39}$$

$$U^{\,(l)} = \tag{2.40}$$

に分けて解けばよい。

以上の議論は一般の $n \times n$ 行列 $\tilde{A}$ に対するものであるが、以後、$\tilde{A}$ として前に得た三重対角行列を考えることにする。したがって、以下では対称行列 A の固有ベクトルを求めることになる。

逆反復法の手順 (2.38) において $l = 1$ とおくと

$$U^{\,(l)} = L^{-1\,(0)} \tag{2.41}$$

となるが、初期値 $^{(0)}$ が任意にとれることから実は $L^{-1\,(0)}$ を任意にとれることになる。したがって、最初の反復では (2.40) の前進代入の部分 $L^{-1\,(0)}$ は省略することができる。λ_k の近似値 μ が十分に近いとすると 2 回目の反復 $LU^{\,(2)} \; = \; ^{(1)}$ で十分精度の高い固有ベクトルを得ることができる。ただし、λ_k に縮退がある場合には、縮退度が増すにつれて反復回数を多くする必要がある

これまでの段階で計算された固有ベクトル $_k$ は、三重対角行列 $\tilde{A}$ の固有ベクトルである。$\tilde{A}$ は A を相似変換 $\tilde{A} = P^{-1}AP$ を行って得たものであるから、$\tilde{A}$ と A の固有値は同じである。そこで、固有値 λ_k に対応する A の固有ベクトルを $_k$ とすると

$$\tilde{A}(P^{-1}{}_k) = \lambda_k(P^{-1}{}_k) \tag{2.42}$$

なる関係が成立するから、$_k = P^{-1}{}_k$ であることがわかる。したがって、$_k$ は

$$_k = P_k \tag{2.43}$$

によって正しい固有値 $_k$ へ変換しなければならない。この変換は (2.4) と (2.18) の関係を通じて $^{(k)}$ および c_k を使って実行することができる。

2.3　連立１次方程式と LU 分解

2.3.1　ガウス消去法と LU 分解

解くべき連立１次方程式は

$$A = \tag{2.44}$$

であるとする。A は $n \times n$ 行列

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \tag{2.45}$$

で、 および はそれぞれ次のようなベクトルである。

$$= \begin{pmatrix} b_1 \\ b_2 \\ \\ b_n \end{pmatrix} \tag{2.46}$$

$$= \begin{pmatrix} x_1 \\ x_2 \\ \\ x_n \end{pmatrix} \tag{2.47}$$

方程式 (2.44) を最も標準的な方法はガウス消去法であるが、ガウス消去法によって (2.44) を直接解く代わりに、係数行列を左下三角行列 L と右上三角行列 U の積として

$$A = LU \tag{2.48}$$

の形に LU 分解すると便利である。分解のためのアルゴリズムはガウス消去法そのものであり、この分解を経由して方程式 (2.44) を解いても、必要な手間は同じである。

そこで、まずガウス消去法（Gauss elimination）について説明しよう。係数行列 A と右辺のベクトル の成分を

$$a_{ij}{}^{(1)} = a_{ij}, \qquad b_i{}^{(1)} = b_i \tag{2.49}$$

と書くと、方程式 (2.44) は次のようになる。

$$\begin{cases} a_{11}{}^{(1)}x_1 + a_{12}{}^{(1)}x_2 + \cdots + a_{1n}{}^{(1)}x_n = b_1{}^{(1)} & [1] \\ a_{21}{}^{(1)}x_1 + a_{22}{}^{(1)}x_2 + \cdots + a_{2n}{}^{(1)}x_n = b_2{}^{(1)} & [2] \\ a_{31}{}^{(1)}x_1 + a_{32}{}^{(1)}x_2 + \cdots + a_{3n}{}^{(1)}x_n = b_3{}^{(1)} & [3] \\ \cdots\cdots\cdots & \\ a_{n1}{}^{(1)}x_1 + a_{n2}{}^{(1)}x_2 + \cdots + a_{nn}{}^{(1)}x_n = b_n{}^{(1)} & [n] \end{cases} \tag{2.50}$$

上記添字 (1) は、これから第 1 段目の消去が行われることを示す。消去の第 1 段では、 (2.50) の [2] 以下の式から x_1 を含む項を消去する。そのために、まず

$$d_1{}^{-1} = \frac{1}{a_{11}{}^{(1)}} \tag{2.51}$$

とおいて、 [1] に

$$m_{21} = a_{21}{}^{(1)} \times d_1{}^{-1} \tag{2.52}$$

を乗じて [2] から引くと、 [2] の x_1 含む項が消えて

$$(a_{22}{}^{(1)} - m_{21}a_{12}{}^{(1)})x_2 + \cdots + (a_{2n}{}^{(1)} - m_{21}a_{1n}{}^{(1)})x_n = b_2{}^{(1)} - m_{21}b_1{}^{(1)}$$

となる。次に、 [1] に

$$m_{31} = a_{31}{}^{(1)} \times d_1{}^{-1}$$

を乗じて [3] から引くと、[3] の x_1 含む項が消える。この操作を繰り返し、[n] の x_1 含む項までを消去すると、(2.50) は次の形になる。

$$\begin{cases} {a_{11}}^{(1)}x_1 + {a_{12}}^{(1)}x_2 + \cdots + {a_{1n}}^{(1)}x_n = {b_1}^{(1)} & [1]' \\ {a_{22}}^{(1)}x_2 + \cdots + {a_{2n}}^{(1)}x_n = {b_2}^{(1)} & [2]' \\ {a_{32}}^{(1)}x_2 + \cdots + {a_{3n}}^{(1)}x_n = {b_3}^{(1)} & [3]' \\ \cdots\cdots\cdots & \\ {a_{n2}}^{(1)}x_2 + \cdots + {a_{nn}}^{(1)}x_n = {b_n}^{(1)} & [n]' \end{cases} \tag{2.53}$$

ここで、${a_{ij}}^{(2)}, {b_i}^{(2)}$ は次式で定義される。

$$\begin{cases} {a_{ij}}^{(2)} = {a_{ij}}^{(1)} - m_{i1}{a_{1j}}^{(1)}, & i,j = 2,3,\cdots,n \\ {b_i}^{(2)} = {b_i}^{(1)} - m_{i1}{b_1}^{(1)}, & i = 2,3,\cdots,n \end{cases}$$

次に、消去の第２段では、(2.53) の $[3]'$ 以下の式から x_2 を含む項を消去する。このための手順は、$[2]', \cdots, [n]'$ を新たに与えられた方程式と考えれば、第１段における消去の手順と全く同じである。以下同様にして、$x_3, x_4, \cdots, x_{n-1}$ を含む項の消去を続けることができる。

以上の消去の手順をまとめると、次のようになる。

$$\begin{array}{l} k = 1,2,\cdots,n-1 \\ \quad \left[\begin{array}{l} {d_k}^{(-1)} = 1/{a_{kk}}^{(k)} \\ i = k+1, k+2, \cdots, n \\ \quad \left[\begin{array}{l} m_{i,k} = {a_{ik}}^{(k)} \times {d_k}^{-1} \\ j = k+1, k+2, \cdots, n \\ \quad \left[{a_{ij}}^{(k+1)} = {a_{ij}}^{(k)} - m_{ik} \times {a_{kj}}^{(k)} \right] \\ {b_i}^{(k+1)} = {b_i}^{(k)} - m_{ik} \times {b_k}^{(k)} \end{array} \right. \end{array} \right. \\ {d_n}^{-1} = 1/{a_{nn}}^{(n)} \end{array} \tag{2.54}$$

m_{ik} を乗数（multiplier）という。d_k は対角成分 ${a_{kk}}^{(k)}$ を意味するが、これは以後の計算では常に逆数の形でのみ現れるので、あらかじめ ${d_k}^{-1}$ を計算しておくわけである。

この操作の結果、もとの方程式は次のように右上三角行列 U を係数にもつ形に変形される。

$$U = {}^{(n)} \tag{2.55}$$

$$U = \begin{pmatrix} {a_{11}}^{(1)} & {a_{12}}^{(1)} & \cdots & {a_{1n}}^{(1)} \\ & {a_{22}}^{(2)} & \cdots & {a_{2n}}^{(2)} \\ & & \ddots & \vdots \\ 0 & & & {a_{nn}}^{(n)} \end{pmatrix}, u_{ij} = {a_{ij}}^{(i)} (i \leq j) \tag{2.56}$$

$$^{(n)} = \begin{pmatrix} {b_1}^{(1)} \\ {b_2}^{(2)} \\ \vdots \\ {b_n}^{(n)} \end{pmatrix} \tag{2.57}$$

ここまでの消去の操作を、ガウス消去法における前進消去（forward elimination）という。

2.3.2　LU 分解の行列表現

ところで、すぐに確かめられるように、消去の第 k 段目の操作 (2.54) は、単位行列の第 k 列目の対角線より下に $m_{ik}, i = k+1, \cdots, n$ の符号を変えたものを挿入した行列

$$M_k = \begin{pmatrix} 1 & & & & & \\ & 1 & & & 0 & \\ & & 1 & & & \\ & & -m_{k+1,k} & 1 & & \\ 0 & & -m_{k+2,k} & & & \\ & & \vdots & 0 & \ddots & \\ & & -m_{n,k} & & & 1 \end{pmatrix} \tag{2.58}$$

を使って、

$$A^{(k+1)} = M_k A^{(k)} \tag{2.59}$$

と書くことができる。ここで、$A^{(k)}$ はこれから第 k 段の消去を行う直前の係数行列である。

$$A^{(k)} = \begin{pmatrix} {a_{11}}^{(1)} & {a_{12}}^{(1)} & \cdots & {a_{1k}}^{(1)} & \cdots & {a_{1n}}^{(1)} \\ & {a_{22}}^{(2)} & \cdots & {a_{2k}}^{(2)} & \cdots & {a_{2n}}^{(2)} \\ & & \ddots & \vdots & & \vdots \\ & & & {a_{k,k}}^{(k)} & & {a_{k,n}}^{(k)} \\ & \mathbf{0} & & {a_{k+1,k}}^{(k)} & & {a_{k+1,n}}^{(k)} \\ & & & \vdots & \ddots & \vdots \\ & & & {a_{n,k}}^{(k)} & \cdots & {a_{n,n}}^{(k)} \end{pmatrix} \tag{2.60}$$

一方、$A^{(1)} = A, A^{(n)} = U$ であるから、(2.59) より

$$U = M_{n-1} M_{n-2} \cdots M_1 A$$

すなわち

$$\begin{aligned} A &= (M_{n-1} M_{n-2} \cdots M_1)^{-1} U \\ &= {M_{n-1}}^{-1} {M_{n-2}}^{-1} \cdots {M_1}^{-1} U \end{aligned} \tag{2.61}$$

が成り立つ。ところが、

$${}_k = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ m_{k+1,k} \\ m_{k+2,k} \\ \vdots \\ m_{n,k} \end{pmatrix} \quad \ldots\ldots k+1 \qquad {}_k = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \ldots\ldots k$$

なるベクトルを定義すると、行列 M_k は

$$M_k = I - {}_k {}_k^T$$

と書くことができる。I は単位行列で、上付添字 T は一般に転置を表すものとする。一方、${}_k^T {}_k = 0$ であるから

$$(I - {}_k {}_k^T)(I + {}_k {}_k^T) = I - {}_k({}_k^T {}_k) {}_k^T = I$$

が成り立ち

$$M_k{}^{-1} = I_{kk}{}^T = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & 0 & \\ & & 1 & & & \\ & & -m_{k+1,k} & 1 & & \\ 0 & & -m_{k+2,k} & & & \\ & & \vdots & 0 & \ddots & \\ & & -m_{n,k} & & & 1 \end{pmatrix}$$

であることがわかる。これから、直接計算することにより

$$M_1{}^{-1}M_2{}^{-1}\cdots M_{n-1}{}^{-1} = \begin{pmatrix} 1 & & & & \\ m_{21} & 1 & & & \\ m_{31} & m_{32} & 1 & & \\ \vdots & \vdots & & \ddots & \\ m_{n1} & m_{n2} & \cdots & m_{n,n-1} & 1 \end{pmatrix}, \qquad m_{ij} = \frac{a_{ij}{}^{(j)}}{a_{jj}{}^{(j)}} \tag{2.62}$$

となることを確かめることができる。すなわち、この行列は左下三角行列である。したがって、この行列を

$$L = M_1{}^{-1}M_2{}^{-1}\cdots M_{n-1}{}^{-1} \tag{2.63}$$

とおけば、(2.61) によって、もとの行列 A は

$$A = LU \tag{2.64}$$

の形に、左下三角行列 L と右上三角行列 U の積に LU 分解されたことになる。

2.3.3　LU 分解による連立１次方程式の解

行列 A の LU 分解を実行しておくと、方程式 (2.44) を解くことは簡単である。方程式 (2.44) は

$$LU = \tag{2.65}$$

と書くことができるから、これを解くためには、まず

$$L = \tag{2.66}$$

の解 を求め、次にこの解 を右辺に持つ方程式

$$U \; = \tag{2.67}$$

を解いて を求めればよい。L と U は共に三角行列であるから、解を求める操作はきわめて簡単である。

まず、(2.66) は、(2.62) と (2.63) より、L の対角成分が 1 であることに注意すれば、

$$\begin{cases} y_1 = b_1 \\ y_i = b_i - \displaystyle\sum_{j=1}^{i-1} m_{ij} y_j, \qquad i = 2, 3, \cdots, n \end{cases} \tag{2.68}$$

によって解くことができる。この部分を、前進代入（forward substitution）という。方程式 (2.67) と (2.57) とを比較すれば $= {}^{(n)}$ が成り立ってることがわかる。

次に、(2.67) は、(2.56) より、後ろから順に

$$\begin{cases} x_n = \dfrac{1}{{a_{nn}}^{(n)}} \, y_n \\ x_i = \dfrac{1}{{a_{ii}}^{(i)}} \left(y_i - \displaystyle\sum_{j=i+1}^{n} {a_{ij}}^{(i)} x_j \right), \qquad i = n-1, n-2, \cdots, 1 \end{cases} \tag{2.69}$$

によって解くことができる。この部分を後進代入（backward substitution）という。この段階で、ガウス消去法は完了したことになる。

係数行列 A が同じで右辺のベクトル が異なる連立方程式を幾組も解く場合には、行列 A をあらかじめ LU 分解しておくと、各々の に対して (2.68) と (2.69) を計算するだけできわめて効率良く解を求めることができる。

3　専用チップについて

専用チップの形については図 3.1に示したものを用いる。

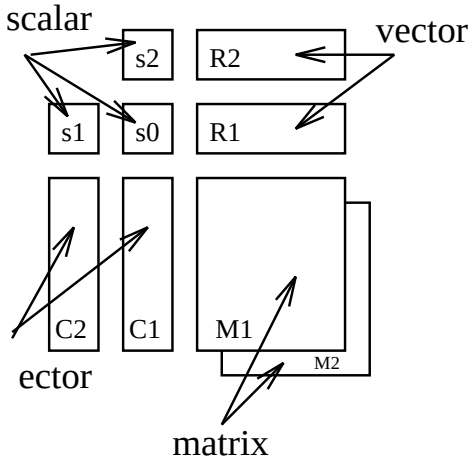


図 3.1: 専用チップの形 [1]

S_0, S_1, S_2 にはスカラーが入り C_1, C_2, R_1, R_2 にはベクトルがそして M_1, M_2 には行列が入る。

以上これらのレジスターに値をいれて, いろいろな過程 (例えば C_1 から C_2 に コピーするなど) を駆使してハウスホルダー変換、 2 分法、逆反復法によって固有値・固有ベクトルを求める。実際の 1 つ 1 つの過程についてはあとの章で述べる。ただし、現段階でプログラム上で新しいレジスターを定義して実行しているのもあるのでその点については今後検討しなければならない。

[1] /home2/students/mizuki/xfig/chip.eps

3.1　専用チップを用いた計算例

例えば $C = AB$ の場合の計算例を以下に示す。

ただし

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} B = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} C = \begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}$$

として考えます。

例えば 2×2 行列のかけ算の場合、普通に計算すると以下のようになる。

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 5+14 & 6+16 \\ 15+28 & 18+32 \end{pmatrix} = \begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}$$

これを専用チップを用いて計算するには以下のような過程で計算する。

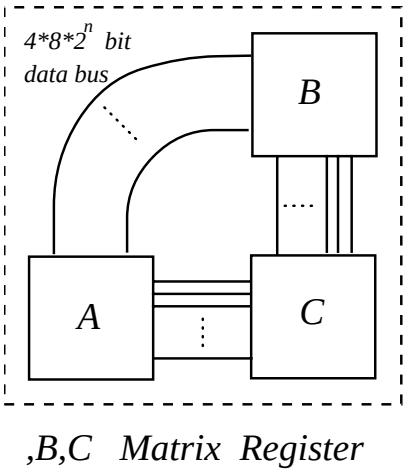


図 3.2: 専用チップの配置例 [2]

まず図 3.2のように専用チップを 2 個用いて掛算を実行する。その結果を下の右側のチップに入れている。それぞれのチップは相互につながっておりデータを伝送させることができる。

次に $C = AB$ の場合の計算を専用チップ上で実現するには以下のことを行う。

[2] /home2/students/mizuki/xfig/chip-cal.eps

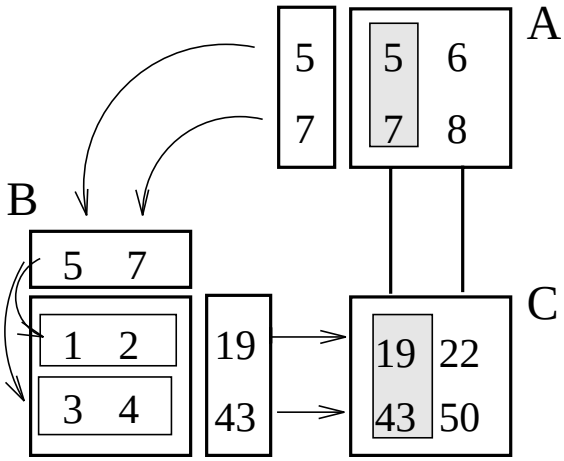


図 3.3: 専用チップの計算例 [3]

(注) ただし図 3.3は図 3.1と少し形が違うが C_1 の位置が移動しているだけなので問題はない。

まず A と B に 行列を入れ結果を C に入れる。

1. Register A の 1 列目をとりだします。そして $C1$ に格納する。
2. Register A の C_1 から Register B の R_1 に移動する。
   ここからは Register B での仕事になり、
3. R_1 と M_1 の行ごとに掛ける。つまりここでは $N \times N$ 行列の場合普通は N^2 回必要になるが専用チップを用いることにより N 回で済むことになる。これがこのチップの特徴であります。
4. M_1 の各行ごとに足し合わせる。
5. 足し合わせた結果を C_1 に入れる。
6. これを C の C_1 に入れる。(または C の M_1 にそのまま入れる。)
   そして 1 から繰り返すが 今度は 2 列目からとりだし 1 から繰り返す。

ここでは６つの過程を用いたがそれぞれ１つ１つの過程を機能を定義する。つまり機能を組み合わせることによりハウスホルダー変換などが実現でき固有値固有ベクトルが求まることとなる。

[3] /home2/students/mizuki/xfig/chip-rei.eps

3.2　計算時間について (実際の数値)

- 行列の足し算の場合

  この場合は行ベクトルを格納するレジスター同士がそれぞれ１成分どうしつながっていれば １回の動作で実行できる。

- 行列のかけ算の場合

  普通 $n \times n$ 行列の場合、 n^3 に比例する時間がかかる。つまり 10000×10000 の行列なら 10^{12} かかることになるがこの専用チップでは n^2 に比例する時間がかかり、大行列ほど時間の短縮になる。 10000×10000 の行列なら 10^8 かかリ,$\frac{1}{10000}$ の時間の短縮となる。

  以上のようにこの専用チップはかけ算が速く計算できるので過程の中にかけ算が多いと計算速度向上が期待できる。

4　過程について

ここでは実際に固有値固有ベクトルを求める際の過程について述べる。次の機能を用いて実現している。

プログラムと関数の名前の見方 (過程のファイル)

4.1　アルファベットの意味

- 始めのアルファベットの意味

  コピー, 移動 c : copy 、 m : move

  ただし move も copy として実行している。

  足し算 a: add

  かけ算 t: times

- 次の m_1(レジスター) などの意味

  次の m_1 や r_1 ,s_0 などは レジスタの位置を表し、足すものなどががそのあとにくる。

又、格納場所は _ (アンダーバー) の後ろの場所になる。

最後の括弧は場所 (何行または何列) を表す。

4.2　実際の例

例えば

(1) am1_m2 は

am1_m2 は m_1 と m_2 を足し 結果を m_2 に格納する。

(2) mc2i_s1(i) i は行

これは C_2i を S_1 に入れるのだが (i) は C_2 の i 行目の成分を S_1 に入れるということを意味している。

(3)mc1i_m1 (i,k,l)

mc1i_m1 (i,k,l) は $c1i$ から m_1 に移動するがそのあとの (i,k,l) の意味は

c_1 の i 行目を m_1 の k 行 l 列に入れるという意味である。

ちなみに i,k は i 行,k 行を意味し j,l は j 列,l 列を表す。(i行, j列 または k行, l列と用いることが多い。)

4.3　特殊なもの

特殊な操作は、操作自体の名前になる。

(例)

平方根 は sqrt

逆数を求めるときは gyaku

値を入れるときは Input

M_1 の対角成分を C_1 に入れる。

C_2 から M_2 の副対角成分に入れる。

などがある。(細かくは実際の過程に述べる。)

4.4　機能一覧表

また i, j などは値を入れる場所 (行や列など) で、次の括弧の中の数値はプログラムの順番です

4.4.1　三重対角行列を求める際 (ハウスホルダー変換) の機能。

mm1_c1i (j) (1) j は列
ms0_c1i (i) (3) i は行
mc1i_c2i (4)
tc2_c1 (5)
ac1i_s0 (6) (13)
sqrt_s0 (7)
mc2i_s1(i) (8) i は行
as1_s0 (9)
ms0_c2i(i) (10)
mc1i_c2i (11-1)
mc2i_r1j (11-2)
tc2i_c1i (12)
ts1_s0 (16)
ts0_r1j (17)
tr1j_m1 (18) (38)
am1_c1i(i) (19)
am1_c1i_2(i) (30)
clr_m1 (20)
mc1i_m1(j) (21) (29) (31)(37)(51)
ts0_c1i (22) (36)
ac1i_s0_2 (24)
mc2i_c1i (11-1)(27) (32) (50)
mc1i_r1j (33) (40)
mm1_c1i(j) (34)
am1_m2 (35)
mm2_m1 (end)
mm2_m3
mm3_m2
mr1j_m2 Q の計算
tc1i_m2
ms0_c1i(i)
ts2_m2
ar2j_m2 (i)

4.4.2　２分法 (固有値を求める) での機能

mm1_m2( ) (0)
ftrace_c2i( ) (1)
tc2i_c2i( ) (2)
ttei_c2i( ) (3)
zettai_m1( ) (4)
max_c1i ( ) (6)
Trace_m2( ) (7)
ms0_r3j ( )
ms0_r4j ( )
ar3j_r4j( )

mr4j_r5j( )

ts1_r5j ( )

mr3j_r4j( )

ttei_r4j( )

ttei_r5j ( ) (8-0)

mc1i_m1 (i,k,l) (8)

mm1_r1j (k,l,j) (9)

gyaku_r1j( ) (11)

tc2i_r1j(i,j) (12)

ar1j_m1 (i,k,l) (13)

ar5j_m1 (j,k,l) (14)

mm1_m1 (j,l) (15)

count (l) (16)

compare ( ) (17)

mr5j_r6j (j) last

4.4.3　逆反復法での機能

ms0_r1j ( ) (2)

tr1j_m1 ( ) (3) and (19)

Trace_m1 ( ) (4)

as0_c1i ( ) (5-2)

gTrace_m1 ( ) (6)

mc2i_m1 (l) (7-0)

mm1_c1i (l) (8)

mm1_r1j (k) (9)

ts0_c1i ( ) (10)

mr1j_s0 (j) (11)

gyaku_s0 ( ) (12)

ts0_r1j ( ) (13)

mr1j_m1 (i) (14)

ms0_c1i (i) (16)

mm1_m2 ( ) (17) and (21)

mc1i_m1 (l) (18)

am2_m1 ( ) (20)

mm1_c2i (j) (22)

mc2i_c1i ( ) (23)

tc1i_c2i ( ) (24)

ac2i_s2 ( ) (25)

gyaku_s2 ( ) (26)

sqrt_s2 ( ) (27)

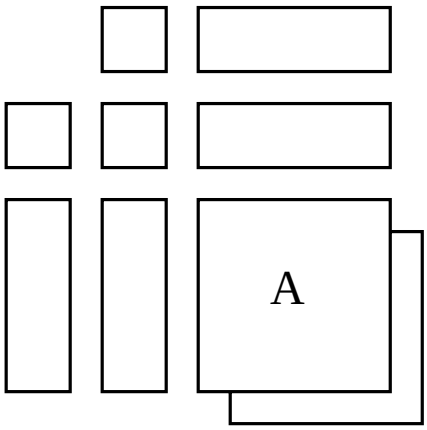
ts2_c1i ( ) (28)

5　専用チップでの過程

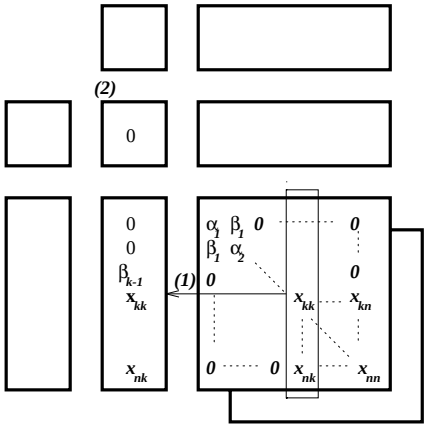
ここからは実際に専用チップ上で固有値固有ベクトルを求める際の過程を説明する。

左側には途中過程を専用チップ上で説明しており、右側には機能を表す。なお右の機能の意味 (例えば $CM_1(k列)C_1$ などは 4.2章で述べたのと同じである。

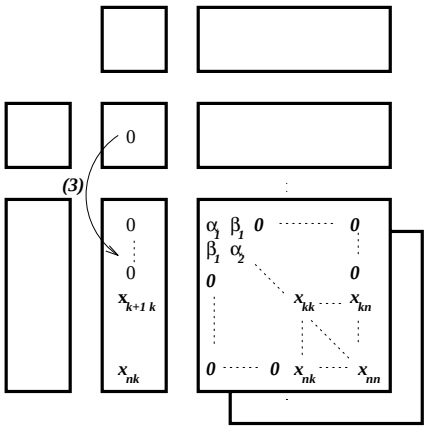
5.1　三重対角化までの過程



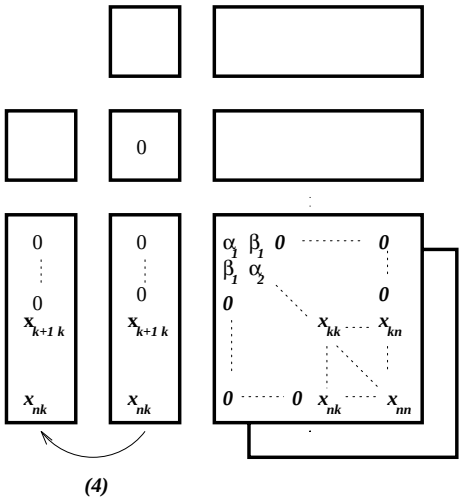
M1,M2 に行列 A を入れる。



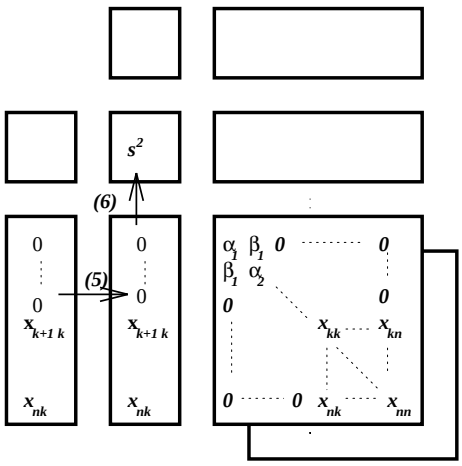
$(1)CM_1(k列)C_1$
M_1 上の k 列目を C_1 に入れる。
$(2)IS_0(0)$ S_0 に 0 を入れる。



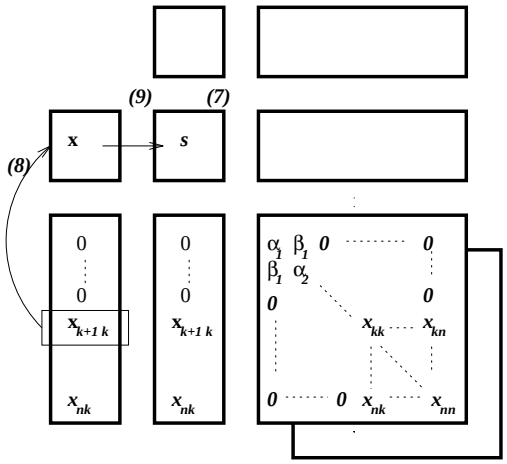
(3)CS_0C_1(i行)
(C_1のi行目成分を0にする)



(4)CC_1C_2
C_1 成分を C_2 へ copy



(5)TC_2C_1
C_2 成分と C_1 成分を掛け合わせ C_1 に格納する。
(6)AC_1S_0
C_1 上の成分を足し 結果を S_0 へ入れる。(S^2 ができる)



(7)$SQRTS_0$

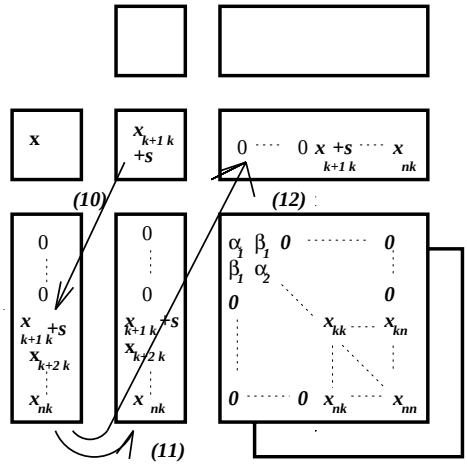
S_0 の平方根をとる。

(8)$CC_2(i+1)S_1$

C_2 の i+1 行目成分を S_1 へ入れる。

(9)AS_1S_0

S_1 と S_0 の足し算を S_0 上で行なう。



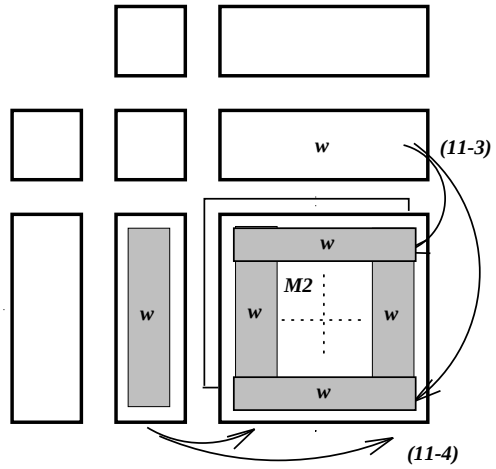
(10)$CS_0C_2(i+1)$

W の完成

(11-1)CC_2C_1

(11-2)CC_2R_1

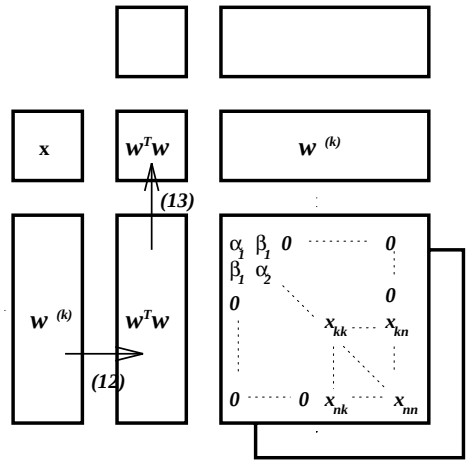
W を C_1R_1 へ入れる。



(11-3)CR_1M_2

(11-4)TC_1M_2

これは逆反復法で用いるため

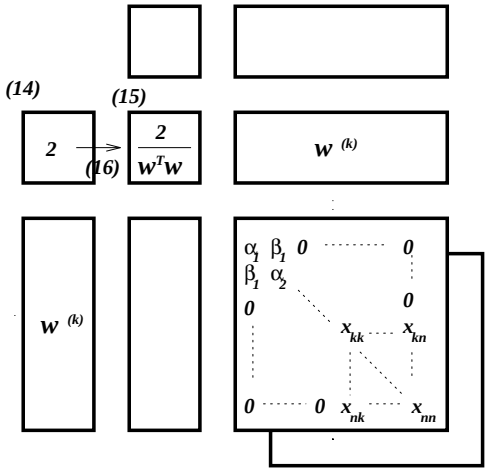


$(12)TC_2C_1$

これより W^TW の計算

$(13)AC_1S_0$

W^TW の計算。



$(14)IS_1(2)$

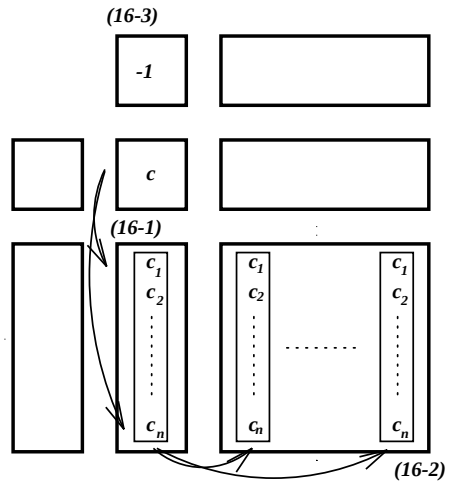
S_1 に 2 を入れる。

$(15)GS_0$

S_0 の逆数をとる。

$(16)TS_1S_0$

$(12) \sim (16)$ で $c = \frac{2}{w^Tw}$ の計算

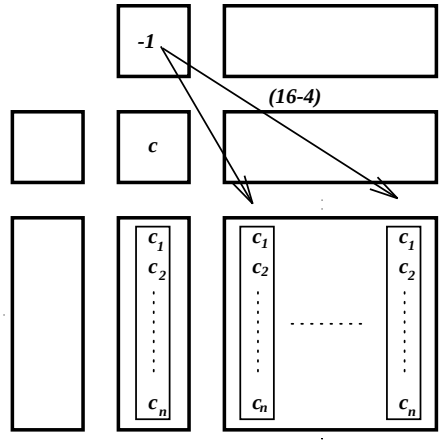


これもあとで逆反復法で用いる。

(16-1)CS_0C_1

(16-2)TC_1M_2

(16-3)$InputS_2(-1)$

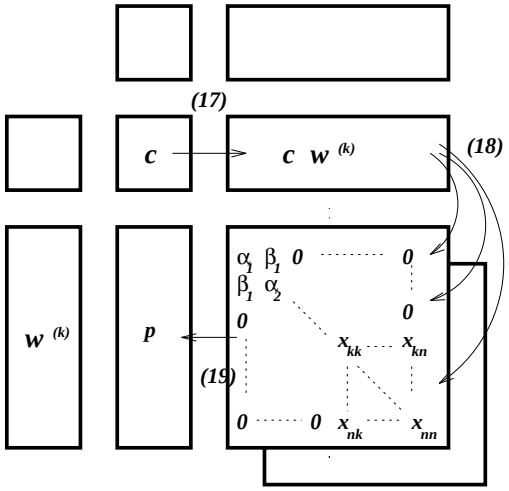


これもあとで逆反復法で用いる。

(16-4)TS_2M_2

(16-5)$InputR_2(1)$

(16-6)AR_2M_2



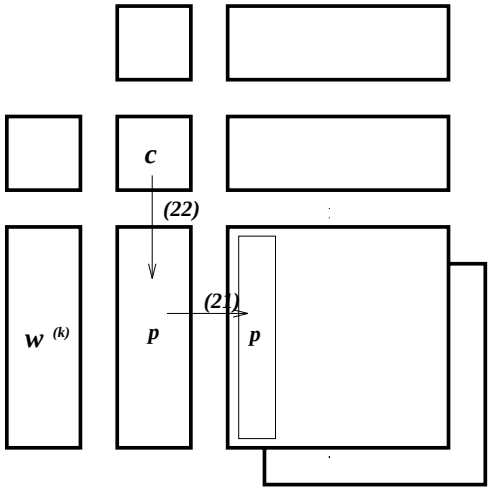
(17)TS_0R_1

CW の計算

(18)TR_1M_1

(19)AM_1C_1

$P = cAaw$ の計算終了



(20)$CLRM_1$

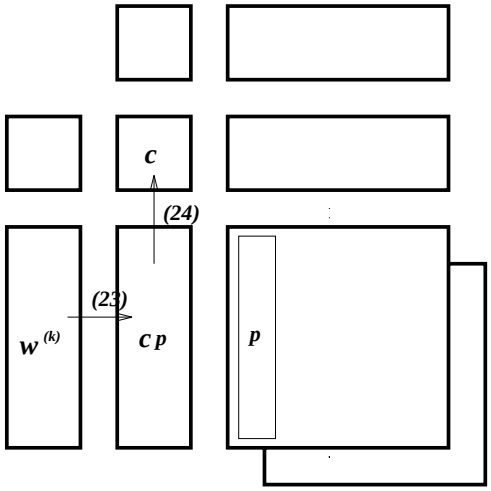
M_1 をすべて 0 にする。

(21)CC_1M_1(1列目)

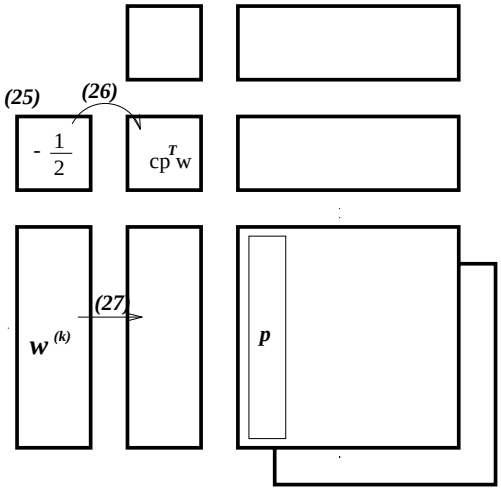
P を M_1 上の任意の位置へ。

(22)TS_0C_1

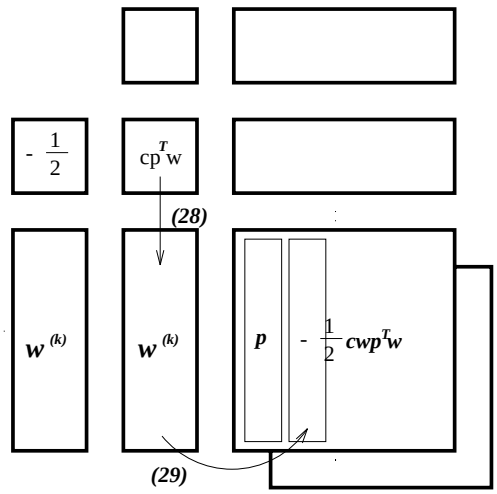
CP^T の計算



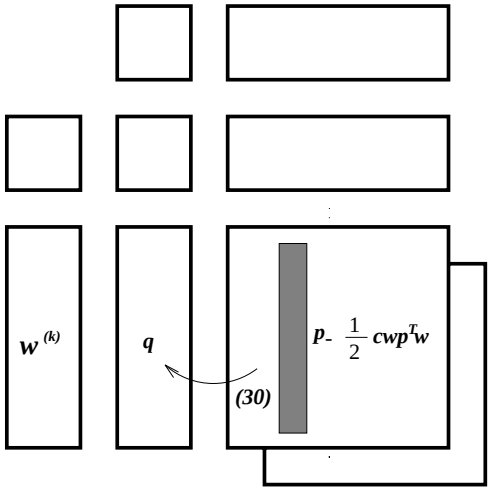
(23)TC_2C_1
CP^Tw の計算 (実はこれはスカラーです。)
(24)AC_1S_0
CP^Tw の計算終了



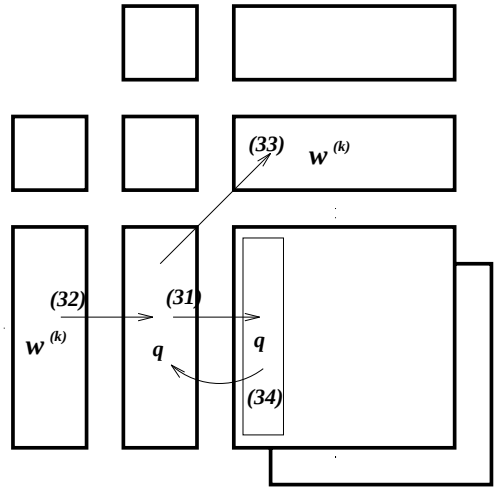
(25) $IS_1(-\frac{1}{2})$
(26) TS_1S_0 $-\frac{1}{2}cp^Tw$ をつくる。
(27) CC_2C_1



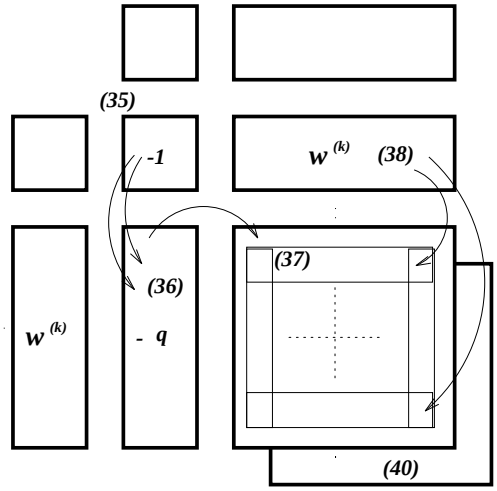
(28)TS_0C_1
(29)$CC_1M_1(2)$ 固定する。



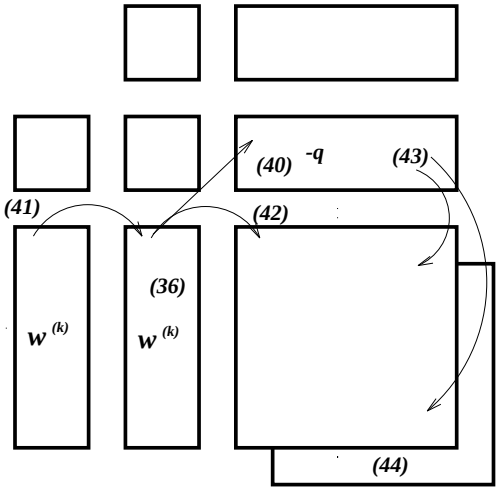
(30)$AM_1C_1q = p - \frac{c}{2}wP^Tw$ の計算終了。



(30-2)$CLRM_1$
(31)CC_1M_1(1列目)
q を M_1 上へ入れる
(32)CC_2C_1
(33)CC_1R_1
(34)$CM_1(1)C_1$



(35)$IS_0(-1)$
(36)TS_0C_1 -q できる
(37)CC_1M_1
(38)TR_1M_1 $-qw^T$ の計算終了。
(39)AM_1M_2　$A - qw^T$ の計算終了。



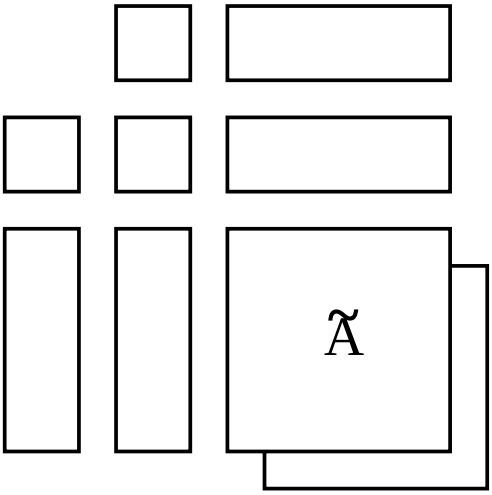
(40)CC_1R_1
(41)CC_2C_1
(42)$CC_1M_1(all)$
(43)TR_1M_1 $-wq^T$ の計算終了。
(44)AM_1M_2

以上を $N-2$ 回繰り返すことにより M_2 の行列が対角化され三重対角化は完成される。[4]

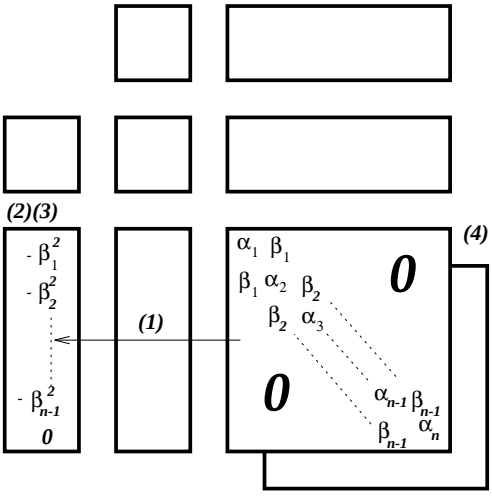
[4] /home2/students/mizuki/xfig/no1.eps ～ no18.eps

5.2　二分法の過程

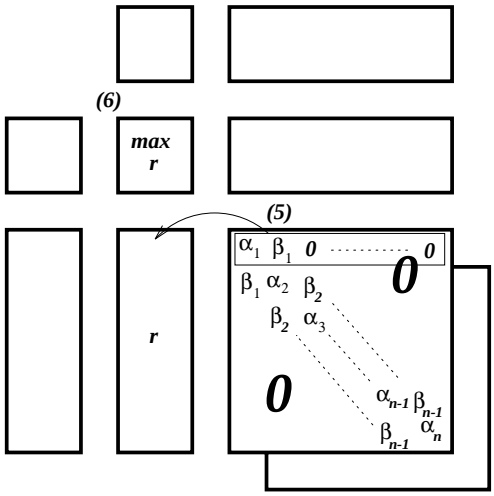
ここで k は 1 から 25 まで動き kk は 1 から N まで動く。なぜ k が 25 なのかというと 2 分法では 1 回ごとに幅が $\frac{1}{2}$ ずつ狭くなるのでこの回数が精度に影響する。いまは 25 回としているが 50 回以上 (できれば 64 回くらい) がよい。
kk は N 回でこれは行列の行数 (または列数) である。



(0) M_1, M_2 に三重対角化された $\tilde{A}$ が入る。

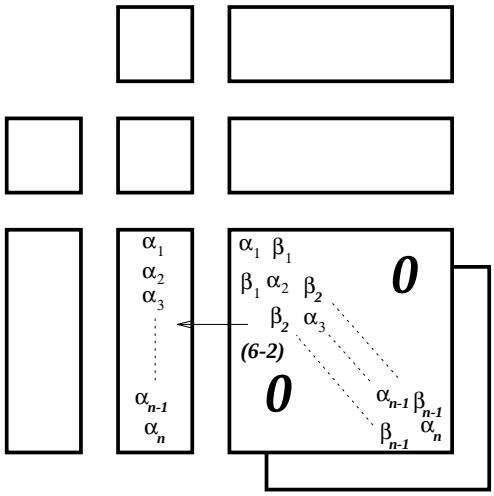


(1) $(FM_1)C_2$　副対角成分を C_2 に入れる。
(2) TC_2C_2
(3) $T(-1)C_2$　-1 を C_2 に掛け結果を C_2 に入れる。
(4) ZM_1　M_1 の絶対値をとる。



(5)AM_1C_1

(6-1)$Max(C_1)S_0$ C_1 の最大値をとり S_0 に入れる。



(6-2)$(TM_1)C_1$

ここ (7) からは仮に R_3, R_4, R_5 を用いている。これは 2 分法の上限下限を表している。

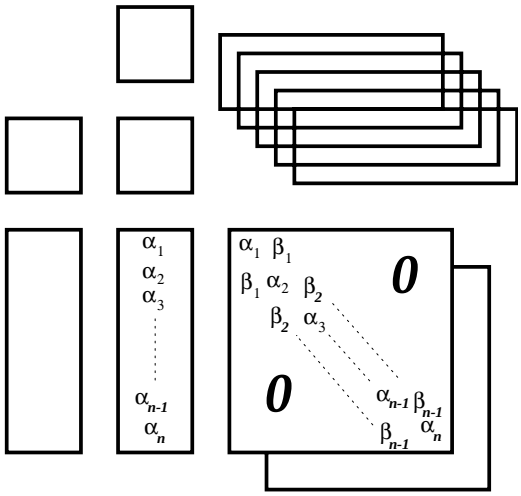
(7-1)CS_0R_4

(7-2)$InputS_1(-1)$

(7-3)TS_1R_4

(7-4)MS_0R_3

(7-5)AR_3R_4



(7-6)MR_4R_5

(7-7)$InputS_1(0.5)$

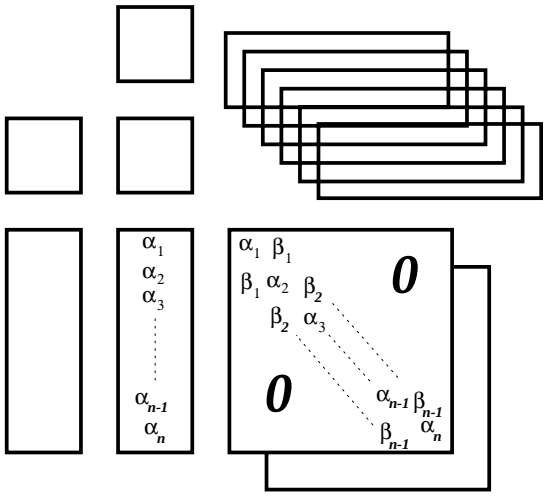
(7-8)TS_1R_5

(7-9)MR_3R_4

(7-10)$T(-1)R_4$

$-1 \times R_4$ の計算

以上ではじめの範囲 ($-r$ から r) が確定する。



(8-0)$T(-1)R_5(all)$

(8-1)$MC_1i(1)M_1(1,k)$ k は移動する。

(8-2)$AR_5(1)M_1(1,kk)$ kk は移動する。

(9)$MM_1(k-1,kk)R_1(kk)$

(10)$MC_1(k)M_1(k,kk)$

(11)$GR_1(kk)$ R_1 の逆数をとる

(12)TC_2(k-1)$R_1(kk)$ kk は列です

(13)$AR_5(1)M_1(1,kk)$ kk は列で 移動する。

(14)MM_1(k-1,kk)$R_1(kk)$ M_1 の (k-1 行 kk 列) の値を R_1 の kk 列に入れる。

　なお (9) 〜 (14) までは N 回繰り返す。(これは 1 列目から N 列目までということである。)

(15)$MC_1(k)M_1(k,kk)$

(16)$Count$ これは M_1 のそれぞれの列ベクトルの値が正である数を数えて結果を R_1 に入れる。

(17)$Compare$ R_1 と R_2 の比較をする。 R_2 には左から 1、 2、 3、と順に値が入っている。この結果

$r1j[j] >= r2j[j]$ なら

$r4j[j] = r5j[j]$そして$r5j[j] = \dfrac{r4[j] + r3[j]}{2}$

となり、

そうでないなら $r3j[j] = r5j[j]$ そして $r5j[j] = \dfrac{r4j[j] + r3j[j]}{2}$

となる。

　このあと

(18)$MR_5R_6(kk)$ kk は移動する。

(19)$InputR_4(0)$

(20)MR_6R_3

(21)$\dfrac{R_3(kk+1)+R_4(kk+1)}{2}$を$R_5(kk+1)$へ($kk$ は列を表す。)

　このあと (8-0) へいき kk 回繰り返す。

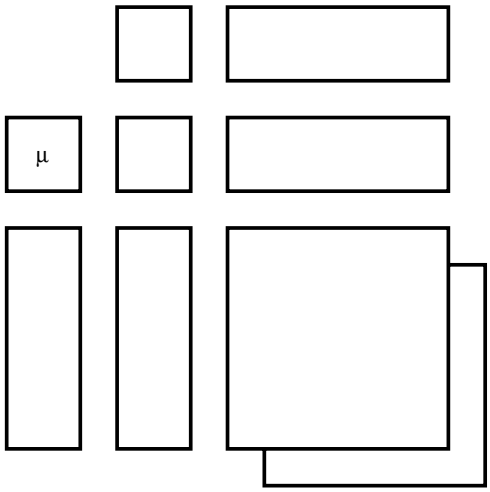
そしてそれぞれの列で計算するので N 回繰り返す。

以上の過程で R_6 に固有値が入る。

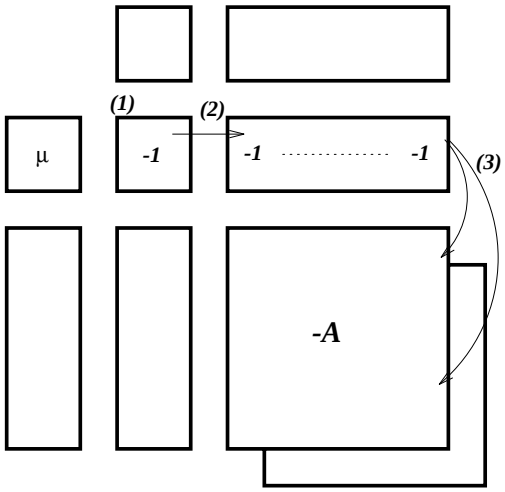
(ただし 2 分法は R_6 まであるのでさらに改良する可能性がある。) [5]

[5] /home2/students/mizuki/xfig2/no1.eps 〜 no6.eps

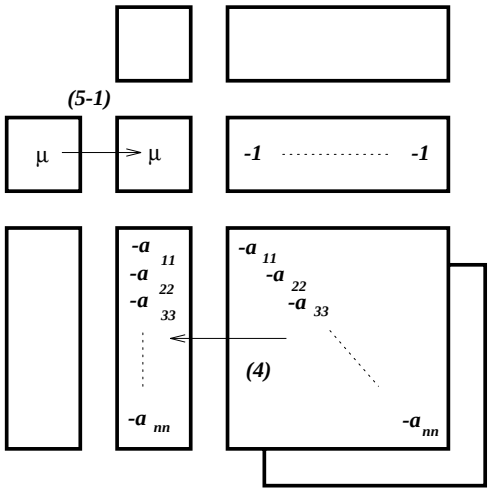
5.3　逆反復法の過程



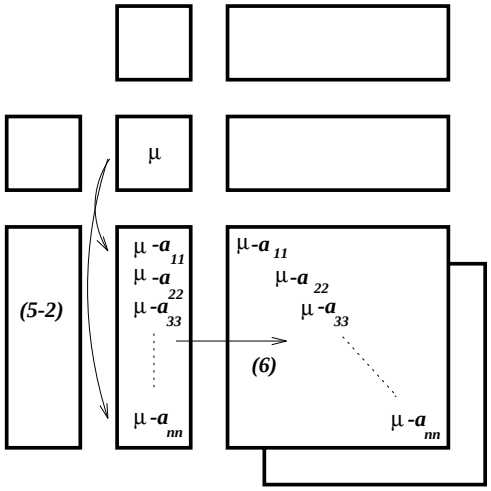
(0)M_1 に行列を入れる。この行列を A とする。



(1)$IS_0(-1)$
(2)CS_0R_1
(3)TR_1M_1
M_1 には $-A$ がはいる

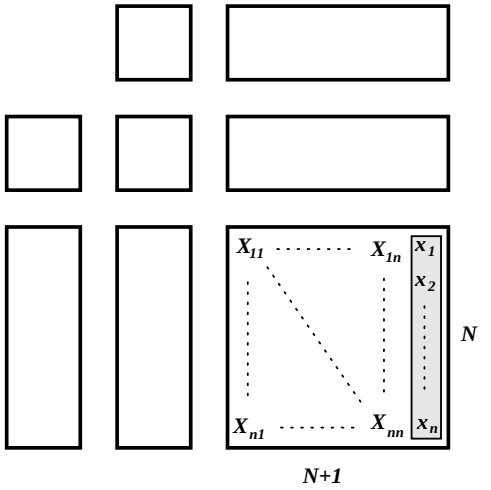


(4)MTM_1C_1 M_1 の対角成分を C_1 に入れる。
(5-1)CS_1S_0



(5-2)AS_0C_1

(6)C_1 を M_1 の対角成分に入れる。

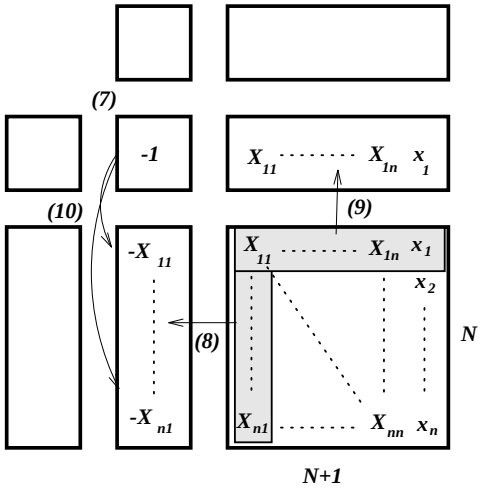


ここで $X = \mu I - A$ とおき成分を X_{11}で表す。

また

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

を任意の初期ベクトルとする。

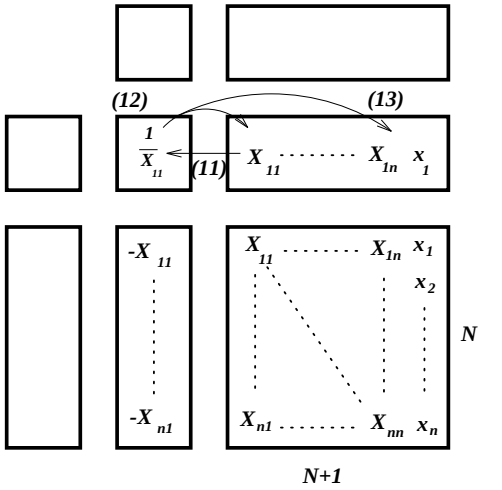


(7)$IS_0(-1)$

(8)$CM_1(j$列$)C_1$

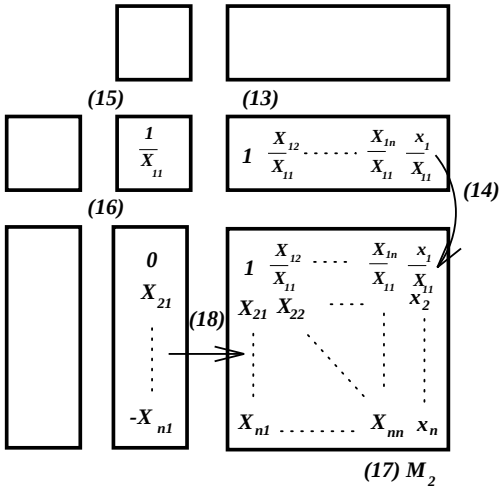
(9)$CM_1(i$行$)R_1$

(10)TS_0C_1



(11)$CR_1(j$列$)S_0$

(12)GS_0



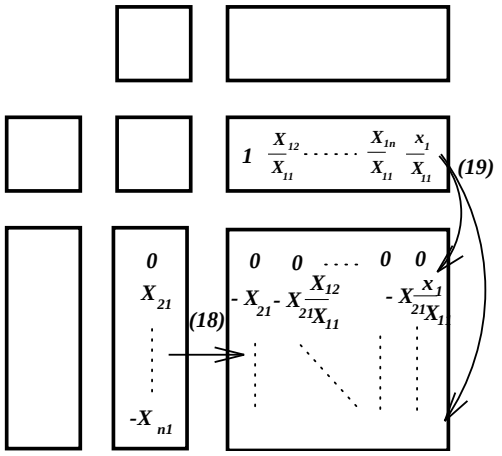
(13)TS_0R_1

(14)$CR_1M_1(j$列$)$

(15)$IS_0(0)$

(16)$CS_0C_1(i$行$)$

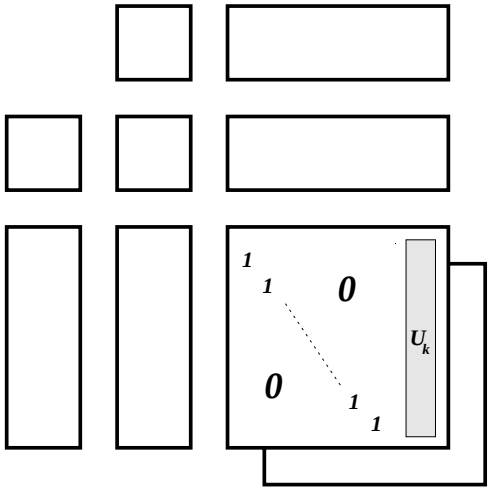
(17)CM_1M_2



(18)CC_1M_1

(19)TR_1M_1

(20)AM_2M_1



(7) 〜 (21) を N 回繰り返す。この結果 $n+1$ 列目にできるベクトル U_k が固有ベクトルとなる。

結果 $n+1$ 列目にできるベクトル U_k が固有ベクトルとなり、これをとりだせば良い。[6]

[6]以上の図は /home2/students/mizuki/xfig3/no1.eps から no10.eps

6　考察

前章までで述べたように多くの機能を現段階では用いねばならないが実現できることがわかった。しかしまだ問題点が多くあり改良しなければならない。

まずシミュレーションプログラムについてはさらなる検討の余地がある。特に無駄な部分を取りシミュレーションプログラム自体の時間の短縮が必要である。

実際に計算させたところハウスホルダー変換では 128 × 128 行列で 7.12 秒なのに対し 逆行列での計算 (逆反復法) がかなりの時間 (128 × 128 行列で 284 秒) を要し、改良しなければならない。(しかしこれはファイルアクセスの時間も含んでいる。)

また 2 分法でのプログラムはレジスターの数が多く定義 (R_1 から R_6 の 6 つ) している。これは途中結果を１回１回格納しなければならないためだか、なるべくレジスターを有効に利用し少なくなるようにしなければならない。あまりに多過ぎるとチップをつくるのが困難になってしまう可能性がある。

7　結論

この専用チップを用いた結果、現段階では多くの過程が必要だが専用チップを用いて固有値固有ベクトルを求めることが可能である。

そしてチップのさらなる効率化を考えハード化することにより固有値固有ベクトルを求める際専用チップを用い計算時間が短縮すると考えられる。

謝辞

本研究及び論文作成に当たり、終始御懇切なる御指導、御鞭撻を賜わりました指導教官である齋藤理一郎助教授に衷心より御礼の言葉を申しあげます。
また、本研究を進めるにあたり、熱心な御指導をいただくとともに種々の御高配を賜わりました木村忠正教授、湯郷成美助教授に深謝の意を表します。
また、研究活動をともにし、多くの援助をいただいた八木将志氏に深謝いたします。
そして、数々の御援助、御助言をしていただいた中平政男氏、竹谷隆夫氏、はじめ木村・齋藤研究室の大学院生、卒研生の方々、高田さんをはじめ (株) 画像技研の方々に感謝します。

参考文献

[1] 数値計算の手順と実際　高田勝 春海佳三朗　コロナ社

[2] FORTRAN77 数値計算プログラミング　森正武　岩波書店

[3] C 言語によるプログラミング (基礎編・応用編)　内田智史　オーム社

[4] C 言語による最新アルゴリズム辞典　奥村春彦　技術評論社

A　付録・シミュレーションプログラム

このプログラムは hhc.c , 2bun.c ,gyaku.c の 3 つのプログラムからなり,hhc.c はハウスホルダー変換の実行 2bun.c は 2 分法の実行 gyaku.c は逆反復法を実行する。
なお hhc.c の実行ファイルを実行すると al.dat ,be.dat というファイルができる。これは三重対角行列の対角成分 (al.dat) 副対角成分 (be.dat) が入っている。
2bun.c の実行ファイルを実行すると al.dat と be.dat を読み込み koyuu.dat という固有値がはいったファイルができる。
最後に gyaku.c の実行ファイルを実行すると koyuu.dat を読んで koyuuvec.dat という固有ベクトルが入ったファイルができる。

```
/*-------------------------------------------
      ハウスホルダー変換プログラム
      hhc.c   1997 1 23  Mizuki Nakajima
-------------------------------------------*/
/*-----------------------------------------
       include
-----------------------------------------*/
#include<stdio.h>
#include<math.h>
#include<time.h>
#define   N   4    /* 行列 N 行 N 列 の配列を定義する  */
#include"register.h"  /* 専用チップのレジスタの定義      */
/* #define DBGPRN */ /* 途中経過を出力するか？
                      #define DBGPRN をコメントアウトすると出力しない  */
/*-----------------------------------------*/
     int i,j  ; /* i 行 j 列 */
     int k    ;
     int s = 1; /* s = start */
/*---------------------------------------
    プロトタイプ宣言
-----------------------------------------*/
     void input ( void );/* この関数を用いる */
     void input_m2 ( void );
     void input_test ( void ); /* test */
/*-------------------------------------------
  以下は
     input_c1i などの関数は c1i に値 0 を入れる関数
     print_m1  などの関数は m1 に値 0 を入れる関数
     _ の後には格納場所が入る
     これは後の 2 分法、逆反復法も同様である
-------------------------------------------------*/
     void input_c1i( void );
     void input_c2i( void );
     void input_r1j( void );
     void input_r2j( void );
     void print_m1( void );
     void print_m2( void );
     void print_c1i( void );
     void print_c2i( void );
     void print_r1j( void );
     void print_r2j( void );
/*-------------------------------------------------
  以下は
  ハウスホルダー変換により三重対角化するため
  一つの過程を一つの関数として表示し
  関数を組み合わせてハウスホルダー変換を実現している
---------------------------------------------------*
     void m1_c1i ( int j );  /* (1) j は列 */
     void s0_c1i ( int i );  /* (3) i は行 */
     void c1i_c2i( void  );  /* (4)       */
     void tc2_c1 ( void  );  /* (5)       */
     double ac1i_s0 ( void  );/* (6) (13)  */
     void sqrt_s0( void );   /* (7)       */
     void mc2i_s1 ( int i ); /* (8) i は行 */
     void as1_s0 ( void );   /* (9)       */
     void ms0_c2i( int i );  /* (10)      */
     void mc1i_c2i( void );  /* (11-1)    */
     void mc2i_r1j( void );  /* (11-2)    */
     void tc2i_c1i( void );  /* (12)      */
      void ts1_s0 ( void );  /*  (16)      */
     void ts0_r1j ( void ); /*  (17)     */
     void tr1j_m1 ( void ); /*  (18) (38)  */
     double am1_c1i ( int i ); /*  (19)   */
     double am1_c1i_2 ( int i ); /*  (30)   */
     void clr_m1 ( void );  /* (20) */
     void mc1i_m1 ( int j ); /* (21) (29) (31)(37)(51)*/
     void ts0_c1i ( void ); /* (22) (36) */
      /* (23) はある */
     double ac1i_s0_2( void ); /* (24) */
```

```
     void mc2i_c1i ( void );  /* (11-1)(27) (32) (50)*/
     void mc1i_r1j( void ); /* (33) (40)*/
     void mm1_c1i ( int j ); /* (34) */
     void am1_m2 ( void ); /* (35) */
     void mm2_m1( void ); /* end */
     void mm2_m3( void );
     void mm3_m2( void );
     void mr1j_m2 ( void ); /*  Q の計算 */
     void tc1i_m2 ( void );
     void ms0_c1i( int i );
     void ts2_m2 ( void ) ;
     void ar2j_m2 ( int a );
/*----------------------------------------
          main ( )
----------------------------------------*/
void  main( void )
{
    FILE *q  ;
    FILE *fp ;
    FILE *fp1;
#ifdef DBGPRN
    time_t  t;
    t = time( NULL );
    printf("%s\n",ctime( &t ));
#endif
    /* 専用レジスタ clear */
    s0 = 0;
    s1 = 0;
    s2 = 0;
    /* 専用レジスタに数値代入 */
    input( ); /* m1 m2 に数値代入 */
    input_c1i( );
    input_c2i( );
    input_r1j( );
    input_r2j( );
#ifdef DBGPRN
    printf("もとの行列表示\n");
    printf("s0 =  %f\n",s0);
    printf("s1 =  %f\n",s1);
    printf("s2 =  %f\n",s2);
    print_m1( );
    print_m2( );
    print_c1i( );
    print_c2i( );
    print_r1j( );
    print_r2j( );
#endif
/*----------------------------------
     3重対角化 start
----------------------------------*/
/* k を動かす k=1 から  k を用いるのは 1,3, です */
   for ( k = 1 ; k <= (N-2) ; k++){
     m1_c1i ( k );    /* (1) m1_c1 移動 k は列 */
     s0 = 0 ;        /* (2) clr_s0 */
     s0_c1i(  k  );   /* (3)-1 */
#ifdef DBGPRN /* 確認 */
    printf("(3) 終了後 \n");
    printf("s0 = %f\n",s0);
    printf("s1 = %f\n",s1);
    printf("s2 = %f\n",s2);
    print_m1( );
    print_m2( );
    print_c1i( );
    print_c2i( );
    print_r1j( );
    print_r2j( );
#endif
    c1i_c2i(  );   /* (4) mc1i_c2i */
    tc2_c1 (  );   /* (5) tc2_c1 */
    s0 = ac1i_s0(  );/* (6) ac1_s0 */
#ifdef DBGPRN /* 確認 */
    printf("(6) 終了後 \n");
    printf("s0 = %f\n",s0);
    printf("s1 = %f\n",s1);
    printf("s2 = %f\n",s2);
    print_m1( );
    print_m2( );
    print_c1i( );
    print_c2i( );
```

```
    print_r1j( );
    print_r2j( );
#endif
    sqrt_s0(  );    /* (7)        */
#ifdef DBGPRN /* 確認 */
    printf("(7)\n");
    printf("s0 = %f (s の答え)\n",s0);
    printf("(8)start\n");
    printf("k= %d \n",k);
#endif
    mc2i_s1( k+1 );  /* (8) */
    as1_s0(  );     /*  (9) */
#ifdef DBGPRN
    printf("(9) \n");
    printf("s0 =  %f\n",s0);
    printf("s1 =  %f\n",s1);
    printf("s2 =  %f\n",s2);
    print_m1( );
    print_m2( );
    print_c1i( );
    print_c2i( );
    print_r1j( );
    print_r2j( );
#endif
#ifdef DBGPRN
    printf("符号確認 \n");
    printf("s0 =  %f\n",s0);
#endif
    ms0_c2i( k+1 ); /* (10)   */
#ifdef DBGPRN
    printf(" w の値\n");
    print_c2i( );
#endif
    mc2i_c1i( );    /* (11-1) */
    mc2i_r1j( );    /* (11-2) */
#ifdef DBGPRN /* 確認 */
    printf("(11-2)\n");
    printf("s0 = %f\n",s0);
    printf("s1 = %f\n",s1);
    printf("s2 = %f\n",s2);
    print_m1( );
    print_m2( );
    print_c1i( );
    print_c2i( );
    print_r1j( );
    print_r2j( );
#endif
/*-------------------Q の計算 (No 1) ----------------------------*/
    mr1j_m2( ); /*  Q の計算 (1) */
    tc1i_m2( ); /*  Q の計算 (2) w * w^T の計算終了 */
/*-------------------Q の計算 end (No 1) ------------------------*/
    tc2i_c1i( ); /* (12-1) */
    s0 = ac1i_s0( );/* (13)  w^T * w の計算終了 */
#ifdef DBGPRN /* 確認 */
    printf("(13)\n");
    printf("s0 = %f\n",s0);
    printf("s1 = %f\n",s1);
    printf("s2 = %f\n",s2);
    print_m1( ) ;
    print_m2( ) ;
    print_c1i( );
    print_c2i( );
    print_r1j( );
    print_r2j( );
#endif
    s1 = 2 ;      /* (14) */
    if ( s0 == 0 ){
        printf("s0 = 0 です。 0 で割れません !! \n");
    }
    else{
        s0 = 1 / s0 ; /* (15) */
    }
#ifdef DBGPRN /* 確認 */
    printf("(15)\n");
    printf("s0 = %f\n",s0);
#endif
    ts1_s0( );  /* (16)  c の計算 終了 */
/*-------------------Q の計算 (No 2) ----------------------------*/
```

```
    for ( i = 1 ; i <= N ; i++){
        ms0_c1i( i ); /* (3) */
    }
    tc1i_m2( ); /* (4) */
    s2 = -1   ; /*  (5)  */
    ts2_m2(  ); /*  (6)*/
    for ( i = 1 ; i <= N ; i++ ){
        r2j[i] = 1 ;  /*  (7)  */
        ar2j_m2( i ); /*  (8)  */
#ifdef DBGPRN
    printf(" (7) Input r2j(%d) \n ",i);
    printf(" (8) ar2j(%d)_m2(%d) \n ",i,i);
    print_r2j( );
#endif
    for( j = 1 ; j <= N ; j++){
       r2j[j] = 0 ;
    }
  }
/*    printf(" result \n");
*    print_m1( );
*    print_m2( );
*/
/*-------------------Q の計算 (No 2) end-------------------------*/
 if((q=fopen("q.dat","a")) == NULL){
        printf("Can not open file q.dat!! \n");
        exit(1);
    }
    fprintf (q," k = %d\n",k);
       for ( i = s; i <= N ; i++ ){
           for ( j = s ; j <= N ; j++ ){
               fprintf( q, "%10.6lf",m2[i][j] );
           }
           fprintf(q,"\n");
       }
    fclose(q);
    ts0_r1j( ); /* (17) */
#ifdef DBGPRN /* 確認 */
    printf("(17)\n");
    printf("s0 = %f\n",s0);
    printf("s1 = %f\n",s1);
    printf("s2 = %f\n",s2);
    print_m1( );
    print_m2( );
    print_c1i( );
    print_c2i( );
    print_r1j( );
    print_r2j( );
#endif
    tr1j_m1( ); /* (18) */
    for ( i = s; i <= N; i++){
        c1i[i] = am1_c1i( i ); /* (19) P=cAW の計算終了 */
       /*  printf(" c1i[%d]= %f \n",i,c1i[i]); */
    }
#ifdef DBGPRN /* 確認 */
    printf("(19)\n");
    printf("s0 = %f\n",s0);
    printf("s1 = %f\n",s1);
    printf("s2 = %f\n",s2);
    print_m1( );
    print_m2( );
    print_c1i( );
    print_c2i( );
    print_r1j( );
    print_r2j( );
#endif
    clr_m1( ); /* (20) */
    j = s; /* j は固定 (一番左端の列) */
    mc1i_m1 ( j ); /* (21 ) */
    ts0_c1i ( ); /* (22) */
#ifdef DBGPRN /* 確認 */
    printf("(22)\n");
    printf("s0 = %f\n",s0);
    printf("s1 = %f\n",s1);
    printf("s2 = %f\n",s2);
    print_m1( );
    print_m2( );
    print_c1i( );
```

```
    print_c2i( );
    print_r1j( );
    print_r2j( );
#endif
    tc2i_c1i ( ); /* (23) */
    s0 = ac1i_s0_2(  ) ;
#ifdef DBGPRN /* 確認 */
    printf("(24)\n");
    printf("s0 = %f\n",s0);
    printf("s1 = %f\n",s1);
    printf("s2 = %f\n",s2);
    print_m1( );
    print_m2( );
    print_c1i( );
    print_c2i( );
    print_r1j( );
    print_r2j( );
#endif
   s1 = -0.5   ;  /* (25) */
   ts1_s0( )   ;  /* (26) */
   mc2i_c1i ( );  /* (27) */
#ifdef DBGPRN /* 確認 */
    printf("(27)\n");
    printf("s0 = %f\n",s0);
    printf("s1 = %f\n",s1);
    printf("s2 = %f\n",s2);
    print_m1( );
    print_m2( );
    print_c1i( );
    print_c2i( );
    print_r1j( );
    print_r2j( );
#endif
    ts0_c1i( ) ; /* (28) */
#ifdef DBGPRN /* 確認 */
    printf("(28) 終了後 \n");
    printf("s0 = %f\n",s0);
    printf("s1 = %f\n",s1);
    printf("s2 = %f\n",s2);
    print_m1( );
    print_m2( );
    print_c1i( );
    print_c2i( );
    print_r1j( );
    print_r2j( );
#endif
    j=s+1; /* j は任意 (ただし j は (21) でいれた値 +1 です !!)(一番左端の列) */
    mc1i_m1( j ); /* (29) */  /* 注意　j の値 */
#ifdef DBGPRN /* 確認 */
    printf("(29) 終了後 \n");
    printf("s0 = %f\n",s0);
    printf("s1 = %f\n",s1);
    printf("s2 = %f\n",s2);
    print_m1( );
    print_m2( );
    print_c1i( );
    print_c2i( );
    print_r1j( );
    print_r2j( );
#endif
       for( i= s; i <=N ; i++){
          c1i[i] = am1_c1i_2( i ); /* (30-1) */
          /* printf(" c1i[%d] = %f \n",i,c1i[i]); */
       }
#ifdef DBGPRN /* 確認 */
    printf("(30-1) 終了後 \n");
    printf("s0 = %f\n",s0);
    printf("s1 = %f\n",s1);
    printf("s2 = %f\n",s2);
    print_m1( );
    print_m2( );
    print_c1i( );
    print_c2i( );
    print_r1j( );
    print_r2j( );
#endif
```

```
    clr_m1( );/* (30-2) */
    j = s       ; /* j は固定 (一番左端の列) */
    mc1i_m1( j ); /* (31) */
    mc2i_c1i(  );/* (32) */
    mc1i_r1j( );/* (33) */
#ifdef DBGPRN /* 確認 */
    printf("(33)end \n");
    printf("s0 = %f\n",s0);
    printf("s1 = %f\n",s1);
    printf("s2 = %f\n",s2);
    print_m1( );
    print_m2( );
    print_c1i( );
    print_c2i( );
    print_r1j( );
    print_r2j( );
#endif
    j = s ;
    mm1_c1i( j );  /* (34) j 固定 */
#ifdef DBGPRN /* 確認 */
    printf("(34)\n");
    print_m1( );
    print_c1i( );
#endif
    s0 = -1; /* (35) */
    ts0_c1i ( ); /* (36) */
#ifdef DBGPRN /* 確認 */
    printf("(36)\n");
    printf("s0 = %f\n",s0);
    printf("s1 = %f\n",s1);
    printf("s2 = %f\n",s2);
    print_m1( );
    print_m2( );
    print_c1i( );
    print_c2i( );
    print_r1j( );
    print_r2j( );
#endif
    for ( j =s; j <=N ; j++){
        mc1i_m1( j ); /* (37) */
    }
#ifdef DBGPRN /* 確認 */
    printf("(37)\n");
    print_c1i( );
    print_m1( );
#endif
    tr1j_m1(  ) ;/* (38) */
#ifdef DBGPRN /* 確認 */
    printf("(38)\n");
    print_r1j( );
    print_m1( );
#endif
    if ( k == 1 ) {
        input_m2( );
    }
    else {
        mm3_m2( );
    }
    am1_m2 (  ) ; /* (39) */
#ifdef DBGPRN /* 確認 */
    printf("(39)\n");
    print_m1( );
    print_m2( );
#endif
#ifdef DBGPRN /* 確認 */
    printf("(39)\n");
    printf("s0 = %f\n",s0);
    printf("s1 = %f\n",s1);
    printf("s2 = %f\n",s2);
    print_m1( );
    print_m2( );
    print_c1i( );
    print_c2i( );
    print_r1j( );
    print_r2j( );
#endif
```

```
    mc1i_r1j( ); /* (40) */
#ifdef DBGPRN /* 確認 */
    printf("(40)\n");
    print_c1i( );
    print_r1j( );
#endif
    mc2i_c1i( ); /* (50) or (41)*/
#ifdef DBGPRN /* 確認 */
    printf("(50) -->(41)\n");
    print_c2i( );
    print_c1i( );
#endif
    for ( j =s; j <=N ; j++){
    mc1i_m1 ( j ); /* (51) 本当は (42)*/
    }
#ifdef DBGPRN /* 確認 */
    printf("(51) -->(42)\n");
    print_c1i( );
    print_m1 ( );
#endif
    tr1j_m1( ); /* (52) */
#ifdef DBGPRN /* 確認 */
    printf("(52) -->(43)\n");
    print_r1j( );
    print_m1 ( );
#endif
    am1_m2 ( ); /* (53) 本当は (44) */
#ifdef DBGPRN /* 確認 */
    printf("(53)-->(44)\n");
    printf("s0 = %f\n",s0);
    printf("s1 = %f\n",s1);
    printf("s2 = %f\n",s2);
    print_m1( );
    print_m2( );
    print_c1i( );
    print_c2i( );
    print_r1j( );
    print_r2j( );
#endif
#ifdef DBGPRN
    printf("結果\n");
    print_m2( );
#endif
   for ( i = s; i<=N ;i++){
       al[i] = m2[i][i] ;
       /* printf ("al[%d]= %lf \n",i,al[i]); */
   }
   for ( i =s; i<=N-1 ;i++){
      be[i] = m2[i+1][i] ;
      /* printf ("be[%d]= %lf \n",i,be[i]); */
   }
    mm2_m3( );
    mm2_m1( ); /* m2 m1 copy  */
#ifdef DBGPRN /* 確認 m2 and m1 */
    print_m2( );
    print_m1( );
#endif
    s = s + 1 ;
 /* 専用レジスタ clear */
    s0 = 0;
    s1 = 0;
    s2 = 0;
    input_c1i( );
    input_c2i( );
    input_r1j( );
    input_r2j( );
    }      /*  for end  */

  if((fp=fopen("al.dat","w")) == NULL){
       printf("Can not open flie al.dat!! \n");
       exit(1);
    }
    if((fp1=fopen("be.dat","w")) == NULL){
       printf("Can not open file be.dat!! \n");
       exit(1);
    }
    for ( i = 1; i <=N ; i++ ){
       fprintf(fp, "%f  \n ",al[i]);
```

```
        fprintf(fp1,"%f  \n ",be[i]);
    }
    fclose(fp);
    fclose(fp1);

}      /* main end */
/*********************************************************
*        行列    input 関数
*********************************************************/
void input ( void )
{
    m1[1][1] = 4; m1[1][2] = 3; m1[1][3] = 2; m1[1][4] = 1;
    m1[2][1] = 3; m1[2][2] = 3; m1[2][3] = 2; m1[2][4] = 1;
    m1[3][1] = 2; m1[3][2] = 2; m1[3][3] = 2; m1[3][4] = 1;
    m1[4][1] = 1; m1[4][2] = 1; m1[4][3] = 1; m1[4][4] = 1;
    m2[1][1] = 4; m2[1][2] = 3; m2[1][3] = 2; m2[1][4] = 1;
    m2[2][1] = 3; m2[2][2] = 3; m2[2][3] = 2; m2[2][4] = 1;
    m2[3][1] = 2; m2[3][2] = 2; m2[3][3] = 2; m2[3][4] = 1;
    m2[4][1] = 1; m2[4][2] = 1; m2[4][3] = 1; m2[4][4] = 1;

}
/*********************************************************
*        行列    input_m2 関数
*********************************************************/
void input_m2 ( void )
{
    m2[1][1] = 4; m2[1][2] = 3; m2[1][3] = 2; m2[1][4] = 1;
    m2[2][1] = 3; m2[2][2] = 3; m2[2][3] = 2; m2[2][4] = 1;
    m2[3][1] = 2; m2[3][2] = 2; m2[3][3] = 2; m2[3][4] = 1;
    m2[4][1] = 1; m2[4][2] = 1; m2[4][3] = 1; m2[4][4] = 1;

}
/*********************************************************
*        行列    input 関数
*********************************************************/
void input_test ( void )
{
    for ( i=1;i<=N ;i++){
        for(j=i;j<=N;j++){
            m1[i][j]=(i+j)*(i+j);
            m2[i][j]=(i+j)*(i+j);
        }
    }
    for ( i=2;i<=N;i++){
        for (j =1; j <=i-1;j++){
            m1[i][j]=m1[j][i];
    m2[i][j]=m2[j][i];
            }
      }
#ifdef DBGPRN
    for ( i=1;i<=N ;i++){
        for(j=i;j<=N;j++){
            print_m1( );
            print_m2( );
        }
    }
#endif
}
/*********************************************************
*      <  行列 r1j   input 関数   >                      *
*********************************************************/
 void input_r1j ( void )
{
    int a,b;
    for(a=1;a<=N;a++){
    for(b=1;b<=N;b++){
             r1j[a]= 0;
        }
    }
}
/*********************************************************
*      <  行列 r2j   input 関数   >                      *
*********************************************************/
 void input_r2j ( void )
{
    int a,b;
    for(a=1;a<=N;a++){
    for(b=1;b<=N;b++){
              r2j[a]= 0;
        }
    }
}
/*********************************************************
      <  行列 c1i   input 関数   >
```

```
*******************************************************/
 void input_c1i ( void )
{
    int a,b;
    for(a=1;a<=N;a++){
    for(b=1;b<=N;b++){
                c1i[a]=0;
        }
    }
}
/*********************************************************
*      <  行列 c2i   input 関数   >                      *
*********************************************************/
 void input_c2i ( void )
{
    int a,b;
    for(a=1;a<=N;a++){
    for(b=1;b<=N;b++){
             c2i[a]=0;
        }
    }
}
/*************************************************************
*                    <   print_r1j 関数 >                    *
*************************************************************/
void  print_r1j ( void )
{
    int a;
    for(a=s;a<=N;a++){
            printf(" r1j[%d]=%f  ",a,r1j[a]);
    }
    printf("\n\n");
}
/*************************************************************
*                    <   print_r2j 関数 >                    *
*************************************************************/
void  print_r2j ( void )
{
    int a;
    for(a=s;a<=N;a++){
            printf(" r2j[%d]=%f  ",a,r2j[a]);
    }
    printf("\n\n");
}
/*************************************************************
*                    <   print_c1i 関数 >                    *
*************************************************************/
void print_c1i ( void )
{
    int a;
    for(a=s;a<=N;a++){
            printf(" c1i[%d]=%f \n",a,c1i[a]);
    }
    printf("\n");
}
/*************************************************************
*                    <   print_c2i 関数 >                    *
*************************************************************/
void print_c2i ( void )
{
    int a;
    for(a=s;a<=N;a++){
            printf(" c2i[%d]=%f \n",a,c2i[a]);
    }
    printf("\n");
}
/*************************************************************
*                    <   print_m1 関数 >                     *
*************************************************************/
 void print_m1 ( void )
{
    int a,b;
    for(a=s;a<=N;a++){
        for(b=s;b<=N;b++){
            printf("m1[%d][%d]=%f ",a,b,m1[a][b]);
        }
    printf("\n");
    }
    printf("\n");
}
/*************************************************************
*                    <   print_m2 関数 >                     *
*************************************************************/
 void print_m2 ( void )
{
```

```
    int a,b;
    for(a=s;a<=N;a++){
        for(b=s;b<=N;b++){
            printf("m2[%d][%d]=%f ",a,b,m2[a][b]);
        }
    printf("\n");
    }
    printf("\n");
}
/*---------------- 関数 ----------------------------*/
/*****************************************************
(1)       m1_c1i ( int b )  m1 から c1i へ移す (copy)
*****************************************************/
void m1_c1i ( int b )
{
int a;
   for(a=s;a<=N;a++){
       c1i[a]=m1[a][b];
       }
}
/*****************************************************
(3)       s0_c1i ( int a  ) 移動 (copy)
*****************************************************/
void s0_c1i ( int a )
{
       c1i[a] = s0;
}
/*****************************************************
(4)      c1i_c2i (   ) 移動
*****************************************************/
void c1i_c2i ( void )
{
       int a;
       for ( a = s; a <=N ; a++ ){
           c2i[a] = c1i[a];
       }
}
/*************************************************
(5)    tc2_c1( )  (times( ))   掛け算関数
*************************************************/
void tc2_c1(  )
{
    int a; /* a は行 */
  for( a = s ; a <=N ; a++ ){
           c1i[a] = c2i[a] * c1i[a] ;
     }
}
/*************************************************************
         <  c1i の 1 行目から N 行目まで足しあわせる。>
(6)   double ac1i_s0( )   ( sum_gyou(  ) )
*************************************************************/
double ac1i_s0( )
{
     int a;
        double t = 0 ; /* t = total */
            for ( a = s ; a <=N ; a++){
           t = t + c1i[a];
            }
        return t;
}
/*****************************************************
(7)            sqrt_s0( )
*****************************************************/
void sqrt_s0 ( void )
{
      s0 = sqrt( s0 );
}
/*****************************************************
(8)           mc2i_s1 ( int a ) 移動 (copy)
*****************************************************/
void mc2i_s1 ( int a )
{
#ifdef DBGPRN
      printf(" (8) \n");
      printf("a = %d \n",a);
#endif
      s1 = c2i[a];
if ( s1 < 0 ){
        s0 = -s0 ;
        }
#ifdef DBGPRN
      printf("s1 =  %f\n",s1);
#endif
}
/*****************************************************
```

```
(9)          as1_s0 ( void ) 足し算関数
*****************************************************/
void  as1_s0 ( void )
{
    s0 = s0 + s1 ;
#ifdef DBGPRN
    printf("s0 の値 = %f\n",s0);
#endif
}
/*****************************************************
(10)        ms0_c2i ( a ) 移動 (copy)
*****************************************************/
void ms0_c2i ( int a )
{
       c2i[a] = s0;
}
/*****************************************************
(11-1)            mc1i_c2i (   ) 移動
*****************************************************/
void mc1i_c2i ( void )
{
       int a;
       for ( a = s; a <=N ; a++ ){
           c2i[a] = c1i[a];
       }
}
/*****************************************************
(11-2)           mc2i_r1j (   ) 移動 (copy)
*****************************************************/
void mc2i_r1j ( void )
{
       int a;
       for ( a = s; a <=N ; a++ ){
           r1j[a] = c2i[a];
       }
}
/*********************************************
(12) (23) tc2i_c1i( )(times( )) 掛け算関数
*********************************************/
void tc2i_c1i( void )
{
    int m; /* m は行 */
  for( m = s ; m <=N; m++){
           c1i[m] = c2i[m] * c1i[m] ;
          }
#ifdef DBGPRN
          printf("(12) or (23)\n");
          print_c1i( );
#endif
}
/************************************************************
(13)   ac1i_s0( )  (6) にある (ただし (6) は c1i_s0!!!)
************************************************************
double ac1i_s0(  )
{
     int m;
        double total = 0  ;
            for ( m = s ; m <=N ; m++){
          total = total + c1i[m] ;
            }
        return total;
}  */
/*********************************************
(16)  ts1_s0   s1 に times を掛け s0 に格納
*********************************************/
void ts1_s0 ( void )
{
    s0 = s1 * s0  ;
#ifdef DBGPRN
    printf("(16)or(26)\n");
    printf("s0= %f \n",s0);
#endif
}
/*********************************************
(17)       ts0_r1j ( )
*********************************************/
void ts0_r1j( void )
{
    int b;
        for(b=s;b<=N;b++){
            r1j[b]=s0 * r1j[b];
        }
}
/************************************************************
(18)   tr1j_m1(  )   掛け算関数
************************************************************/
void tr1j_m1 ( void )
```

```
{
    int a,b;/* a 行 b 列 */
    for(a=s;a<=N;a++){
       for(b=s;b<=N;b++){
           m1[a][b] = r1j[b] * m1[a][b];
       }
    }
}
/*******************************************************
(19)(30)  行列 m1 で 関数 doble am1_c1i( int a )(横にたす)
*******************************************************/
double am1_c1i( int a )
{
     int b;         /*  列が動く */
        double t = 0 ; /*  t = total */
for( b = s ; b <=N ; b++ ){
              t = t + m1[a][b];
                 /* printf(" b=%i and t = %f \n" ,b,t); */
        }
        return t;
}
/*******************************************************
(30)  行列 m1 で 関数 doble am1_c1i_2( int a )(横にたす)
*******************************************************/
double am1_c1i_2( int a )
{
     int b;         /*  列が動く */
        double t = 0 ; /*  t = total */
for( b = s ; b <=N ; b++ ){
              t = t + m1[a][b];
                 /* printf(" b=%i and t = %f \n" ,b,t); */
        }
        return t;
}
/*************************************************
(20)          clr_m1 ( )
*************************************************/
void clr_m1( void )
{
    int a,b;
    for(a=s;a<=N;a++){
        for(b=s;b<=N;b++){
            m1[a][b] = 0;
        }
    }
}
/*************************************************
(21)   mc1i_m1 ( int b )   c1i から m1 へ移す
*************************************************/
void mc1i_m1 ( int b )
{
int a;
for(a=s;a<=N;a++){
    m1[a][b]=c1i[a];
        }
}
/*************************************************
(22)    ts0_c1i ( )
*************************************************/
void ts0_c1i( void )
{
    int b;
        for(b=s;b<=N;b++){
            c1i[b]=s0 * c1i[b];
        }
}
/*************************************************************
         <  c1i で 1 行目から N 行目まで足しあわせる。>
(24)   double ac1i_s0_2( )   ( sum_gyou(  ) )
*************************************************************/
double ac1i_s0_2( )
{
     int a;
        double total = 0 ; /* total = total */
            for ( a = s ; a <=N ; a++){
               /* printf("c1i[%d]=%f\n",a,c1i[a] );*/
                total = total + c1i[a] ; /* ここがおかしい 11/9  */
                                         /* 解決 11/11 13:46 */
                /* printf("total = %f\n",total); */
           }
           return total;
}
/*************************************************
(27)               mc2i_c1i (   ) 移動
*************************************************/
```

```
void mc2i_c1i ( void )
{
       int a;
       for ( a = s; a <=N ; a++ ){
           c1i[a] = c2i[a];
       }
}
/*****************************************************
(33)      mc1i_r1j (   ) 移動 (copy)
*****************************************************/
void mc1i_r1j ( void )
{
       int a;
       for ( a = s; a <=N ; a++ ){
           r1j[a] = c1i[a];
       }
}
/*****************************************************
(34)       mm1_c1i ( int b )  m1 から c1i へ移す (copy)
*****************************************************/
void mm1_c1i ( int b )
{
int a;
   for( a = s ; a <=N ; a++ ){
     c1i[a] = m1[a][b];
           }
}
/*************************************************
(39)              am1_m2
*************************************************/
void  am1_m2( void )
{
    int a,b;
    for(a=s;a<=N;a++){
        for(b=s;b<=N;b++){
            m2[a][b] = m1[a][b] + m2[a][b] ;
     }
   }
}
/*************************************************
(end)             mm2_m1
*************************************************/
void  mm2_m1( void )
{
    int a,b;
    for(a=s;a<=N;a++){
        for(b=s;b<=N;b++){
            m1[a][b] = m2[a][b] ;
     }
   }
}
/*************************************************
(end)             mm2_m3
*************************************************/
void  mm2_m3( void )
{
    int a,b;
    for(a=s;a<=N;a++){
        for(b=s;b<=N;b++){
            m3[a][b] = m2[a][b] ;
     }
   }
}
/*************************************************
(end)             mm3_m2
*************************************************/
void  mm3_m2( void )
{
    int a,b;
    for(a=s;a<=N;a++){
        for(b=s;b<=N;b++){
            m2[a][b] = m3[a][b] ;
     }
   }
}
/*************************************************************
(Q の計算)(1)   mr1j_m2( void )
*************************************************************/
void mr1j_m2 ( void )
{
    int a,b;  /* a 行 b 列 */
#ifdef DBGPRN
    printf("Q の計算  mr1j_m2( void ) \n");
    print_r1j( );
    print_m2 ( );
#endif
```

```
    for(a=1;a<=N;a++){
       for(b=1;b<=N;b++){
           m2[a][b] = r1j[b] ;
       }
    }
#ifdef DBGPRN
    print_m2 ( );
#endif
}
/*************************************************************
Q の計算 (2)and (4)   tc1i_m2(  )   掛け算関数
*************************************************************/
void tc1i_m2 ( void )
{
    int a,b;/* a 行 b 列 */
#ifdef DBGPRN
    printf("Q の計算 (2)or (4)  tc1i_m2( void ) \n");
    print_c1i( );
    print_m2 ( );
#endif
    for(a=1;a<=N;a++){
       for(b=1;b<=N;b++){
           m2[a][b] = c1i[a] * m2[a][b];
       }
    }
#ifdef DBGPRN
    print_m2 ( );
#endif
}
/*************************************************
 Q の計算 (3)    ms0_c1i ( int a ) 移動 (copy)
*************************************************/
void ms0_c1i ( int a )
{
#ifdef DBGPRN
    printf("Q の計算 (3) ms0_c1i ( int a )  \n");
    printf("s0 = %f \n",s0);
    print_c1i( );
#endif
       c1i[a] = s0;
#ifdef DBGPRN
     print_c1i( );
#endif
}
/*************************************************************
 Q の計算 (6)   ts2_m2(  )   掛け算関数
*************************************************************/
void ts2_m2 ( void )
{
    int a,b;/* a 行 b 列 */
#ifdef DBGPRN
    printf("Q の計算 (6)   ts2_m2( void ) \n");
    printf("s2 = %f \n ",s2);
    print_m2 ( );
#endif
    for(a=1;a<=N;a++){
       for(b=1;b<=N;b++){
           m2[a][b] = s2 * m2[a][b];
       }
    }
#ifdef DBGPRN
    print_m2 ( );
#endif
}
/*************************************************
Q の計算 (8)             ar2j_m2
*************************************************/
void  ar2j_m2( int a )
{
    int b;
#ifdef DBGPRN
    printf("Q の計算 (8)  ar2j_m2 \n");
    print_r2j ( );
    print_m2 ( );
#endif
            for(b=s;b<=N;b++){
                m2[a][b] = r2j[b] + m2[a][b] ;
         }
#ifdef DBGPRN
     print_m2 ( );
#endif
}

/*-------------------------------------------
         2 分法 プログラム
      2bun.c   1997 1 31 Mizuki Nakajima
```

```
------------------------------------------*/
/*----------------------------------------
        include
------------------------------------------*/
#include<stdio.h>
#include<math.h>
#define  N  4  /* N x N 行列 */
#include"register.h"
/*----------------------------------------
        define
------------------------------------------*/
/*  #define DBGPRN  */
/*  #define RESULT  */
     int i,j; /* i行 j列 */
     int c,cnt,k,kk,z;
/*----------------------------------------
    プロトタイプ宣言
------------------------------------------*/
     void input ( void );
     void input_m1 ( void );
     void input_m2 ( void );
     void input_c1i( void );
     void input_c2i( void );
     void input_r1j( void );
     void input_r2j( void );
     void print_m1 ( void );
     void print_m2 ( void );
     void print_c1i( void );
     void print_c2i( void );
     void print_r1j( void );
     void print_r2j( void );
     void print_r3j( void );
     void print_r4j( void );
     void print_r5j( void );
     void print_r6j( void );
     void mm1_m2( void ); /* (0) */
     void ftrace_c2i( void )  ;  /* (1)  */
     void tc2i_c2i( void )    ;  /* (2)  */
     void ttei_c2i( void )    ;  /* (3)  */
     void zettai_m1( void )   ;  /* (4)  */
     void abort( void )       ;  /* (4-1)*/
     double sum_gyou ( int k );  /* (5-0)*/
     void  sum_wa_c1i( void ) ;  /* (5)  */
     double max_c1i ( void )  ;  /* (6)  */
     void Trace_m2( void )    ;  /* (7)  */
     void ms0_r3j ( void );
     void ms0_r4j ( void );
     void ar3j_r4j( void );
     void mr4j_r5j( void );
     void ts1_r5j ( void );
     void mr3j_r4j( void );
     void ttei_r4j( void );
 void ttei_r5j ( void )              ;/* (8-0) */
 void mc1i_m1  ( int i,int k,int l ) ;/*  (8)  */
 void mm1_r1j  ( int k,int l,int j ) ;/*  (9)  */
 void gyaku_r1j( void )              ;/* (11)  */
  void tc2i_r1j( int i,int j )       ;/* (12)  */
  void ar1j_m1 ( int i,int k,int l ) ;/* (13)  */
  void ar5j_m1 ( int j,int k,int l)  ;/* (14)  */
  void mm1_m1  (int j,int l)         ;/* (15)  */
  void count   ( int l )             ;/* (16)  */
  void compare ( void )            ;/* (17)  */
  void mr5j_r6j  ( int j )           ;/* last */
/*----------------------------------------
          main ( )
------------------------------------------*/
void  main ( void )
{
   FILE *fp2 ;
   FILE *fp3 ;
   FILE *fpout;
   /* 専用レジスタ clear    */
    s0 = 0;
    s1 = 0;
    s2 = 0;
    /* 専用レジスタに数値代入 */
    /* input (   ); */
/* ハウスホルダー変換の結果ファイルをとりだす（対角成分）   */
    if((fp2=fopen("al.dat","r")) == NULL){
        printf("input file al.dat が open できません !! \n");
        exit(1);
```

```
    }
    cnt = 1;
    while ( fscanf(fp2,"%lf",&m1[cnt][cnt]) == 1 ){
   /*     printf("m1[%d][%d] = %lf\n",cnt,cnt,m1[cnt][cnt]);  */
        cnt = cnt + 1 ;
    }
    fclose(fp2);
/* ハウスホルダー変換の結果ファイルをとりだす (副対角成分)   */
    if((fp3=fopen("be.dat","r")) == NULL){
        printf("input file be.dat が open できません !! \n");
        exit(1);
    }
    cnt = 1;
    while ( fscanf(fp3,"%lf",&m1[cnt+1][cnt]) == 1 ){
        m1[cnt][cnt+1] = m1[cnt+1][cnt];
   /*     printf("m1[%d][%d] = %lf\n",cnt,cnt+1,m1[cnt][cnt+1]); */
   /*     printf("m1[%d][%d] = %lf\n",cnt+1,cnt,m1[cnt+1][cnt]); */
        cnt = cnt + 1 ;
    }

    fclose(fp3);
    mm1_m2(  ); /*  */
    input_c1i( );
    input_c2i( );
    input_r1j( );
    input_r2j( );
/*    print_m1( );
*    print_m2( );
*/
#ifdef DBGPRN
    printf("もとの行列表示\n");
    printf("s0 =  %12.10lf\n",s0);
    printf("s1 =  %12.10lf\n",s1);
    printf("s2 =  %12.10lf\n",s2);
    print_m1( );
    print_m2( );
    print_c1i( );
    print_c2i( );
    print_r1j( );
    print_r2j( );
#endif

/*----------------------------------
     2 分法 start
----------------------------------*/
    ftrace_c2i ( );  /* (1) */
#ifdef DBGPRN /* 確認 */
    printf("(1) 終了後 \n");
    print_m1( );
    print_m2( );
    print_c1i( );
    print_c2i( );
#endif
    tc2i_c2i(  ) ; /* (2) */
    ttei_c2i(  ) ; /* (3) */
    zettai_m1(  ); /* (4)  */
#ifdef DBGPRN
    printf("(4) 結果\n");
    printf("s0 =  %12.10lf\n",s0);
    printf("s1 =  %12.10lf\n",s1);
    printf("s2 =  %12.10lf\n",s2);
    print_m1( );
    print_m2( );
    print_c1i( );
    print_c2i( );
    print_r1j( );
    print_r2j( );
#endif
    sum_wa_c1i( ); /* am1_c1i( );  (5) */
#ifdef DBGPRN
    printf("(5) 結果\n");
    printf("s0 =  %12.10lf\n",s0);
    print_m1( );
    print_c1i( );
#endif
    s0 =  max_c1i ( );   /* (6) */
  for ( z = 1 ; z <=N ; z++ ){
    r3j[z]=  s0;
```

```
    r4j[z]= -s0;
  }
#ifdef DBGPRN
    printf("(6) 結果 \n");
    printf("s0 =  %12.10lf\n",s0);
    print_r3j( );
    print_r4j( );
#endif
    Trace_m2 (  ) ; /* (7) 対角成分を c1i に格納 */

  /* r5j の計算  */
    ms0_r3j ( );
    ms0_r4j ( );
    ttei_r4j( );
    ar3j_r4j( );
    mr4j_r5j( );
    s1 = 0.5   ;
    ts1_r5j ( );
    mr3j_r4j( );
    ttei_r4j( );
#ifdef DBGPRN
    printf("------------r3j,r4j,and r5j-------- \n");
    print_r3j( );
    print_r4j( );
    print_r5j( );
#endif
   /* r5j の計算 end  */
/*****************for loop start!! ( kk )************************/
  for ( kk = 1 ; kk <=N ; kk++) {
/*****************for loop start!!*******************************/
  for ( k = 1 ; k <=25  ; k++){

     /* q1 の計算  */
       ttei_r5j(  ) ; /* (8-0) */
#ifdef DBGPRN
    printf("(8-0)  \n");
    print_r5j( );
#endif
       mc1i_m1 ( 1,1,kk ) ; /* (8-1) */
#ifdef DBGPRN
    printf("(8-1)  \n");
    print_c1i( );
    print_m1( );
#endif
       ar5j_m1(kk,1,kk); /* (8-2) */
#ifdef DBGPRN
    printf("(8-2)  \n");
    print_r5j( );
    print_m1( );
#endif
    /* Qc の計算 c は count */
/*************NO 3 for loop start ***********************************/
   for ( c = 2 ; c <=N ; c++){
    mm1_r1j ( c-1,kk,kk ) ; /* (9) */
#ifdef DBGPRN
    printf("**************test r3j,r4j,r5j**************** \n");
    print_r3j( );
    print_r4j( );
    print_r5j( );
    printf("-----------from (9) to (14) ----------------------\n");
    printf("(9) %d kaime \n",c-1);
    print_m1( );
    print_r1j( );
#endif
    mc1i_m1 ( c,c,kk ) ; /* (10) */
#ifdef DBGPRN
    printf("(10) %d kaime \n",c-1);
    print_m1( );
    print_c1i( );
#endif
    gyaku_r1j( ); /* (11) */
#ifdef DBGPRN
    printf("(11) %d kaime \n",c-1);
    print_r1j( );
#endif
    tc2i_r1j( c-1,kk ); /* (12) */
#ifdef DBGPRN
    printf("(12) %d kaime \n",c-1);
    print_c2i( );
```

```
    print_r1j( );
    print_m1( );
#endif
    ar1j_m1( kk, c, kk );/* (13) */
#ifdef DBGPRN
    printf("(13)%d kaime \n",c-1);
    print_r1j( );
    print_m1( );
    print_r5j( );/* 確認 */
#endif
    ar5j_m1( kk,c,kk ); /*  (14)  */
#ifdef DBGPRN
    printf("(14)%d kaime \n",c-1);
    print_m1( );
    print_r5j( );
#endif
 } /*************NO 3 for loop end ***********************************/
 ttei_r5j(  ) ; /* (15-0) */
#ifdef DBGPRN
   printf(" (15 の前) \n");
   print_m1( );
#endif
   /* count 機能  (16) */
   for ( z = 1 ; z <=N ; z++){
#ifdef DBGPRN
      printf(" %d retume \n",z);
#endif
      count ( z ); /* (16) */
   }
#ifdef DBGPRN
   printf(" (16) \n");
   print_r1j( );
#endif
    /* 比較する  */
    compare( ); /* (17) */
#ifdef DBGPRN
    printf("最終結果\n");
    printf("s0 =  %12.10lf\n",s0);
    printf("s1 =  %12.10lf\n",s1);
    printf("s2 =  %12.10lf\n",s2);
    print_m1( );
    print_m2( );
    print_c1i( );
    print_c2i( );
    print_r1j( );
    print_r2j( );
    print_r3j( );
    print_r4j( );
    print_r5j( );
#endif

#ifdef DBGPRN
    printf(" 固有値結果 %d kaime \n " ,k );
    print_r3j( );
    print_r4j( );
    print_r5j( );
    print_r6j( );
#endif
   } /* for loop k end */
/****************** for loop k end ***********************************/
    mr5j_r6j( kk );
    for ( z = 1; z <=N ; z++){
        r4j[z] = 0;
    }
    r3j[kk+1] = r6j[kk] ;
    r5j[kk+1] = ( r3j[kk+1] + r4j[kk+1] ) * 0.5 ;
#ifdef DBGPRN
    printf("--------------test-------------\n");
    print_r3j( );
    print_r4j( );
    print_r5j( );
    print_r6j( );
    printf("------------test end----------------\n");
#endif
} /* for loop end kk この後8-0から繰り返す */
/********************* for loop end kk ************************/
#ifdef RESULT
 printf("最終結果(固有値) \n");
```

```
 print_r6j( );
#endif
    if(( fpout = fopen( "koyuu.dat","w" )) == NULL )
    {
         printf( "Can not open output file koyuu.dat!!\n");
         exit(1);
    }
    for ( i= 1; i <= N ; i++){
        fprintf(fpout,"%12.10lf\n",r6j[i]);
    }
    fclose(fpout);
}        /* main end */
/*********************************************************
*        行列    input 関数
*********************************************************/
void input ( void )
{
m1[1][1] = 4; m1[1][2] = -3.741657; m1[1][3] = 0 ;  m1[1][4] = 0    ;
m1[2][1] = -3.741657; m1[2][2] = 0.4999989 ; m1[2][3] = 0.4629118; m1[2][4] = 0 ;
m1[3][1] = 0; m1[3][2] = 0.4629118; m1[3][3] = 0.6666635 ;  m1[3][4] = -0.0890879;
m1[4][1] = 0; m1[4][2] = 0    ; m1[4][3] = -0.0890879;  m1[4][4] = 0.3333329 ;
m2[1][1] = 4; m2[1][2] = -3.741657; m2[1][3] = 0 ;  m2[1][4] = 0    ;
m2[2][1] = -3.741657; m2[2][2] = 0.4999989 ; m2[2][3] = 0.4629118; m2[2][4] = 0 ;
m2[3][1] = 0; m2[3][2] = 0.4629118; m2[3][3] = 0.6666635 ;  m2[3][4] = -0.0890879;
m2[4][1] = 0; m2[4][2] = 0    ; m2[4][3] = -0.0890879;  m2[4][4] = 0.3333329 ;
}
/*********************************************************
*        行列 m1   input 関数
*********************************************************/
void input_m1 ( void )
{
    static int i,j;
    for(i=0;i<=N;i++){
    for(j=0;j<=N;j++){
            m1[i][j]=i+j-1; /* i+j+1 は対角行列 */
        }
    }
}
/*********************************************************
*        行列 m2   input 関数                          *
*********************************************************/
  void input_m2 ( void )
{
    int i,j;
    for(i=0;i<=N;i++){
    for(j=0;j<=N;j++){
            m2[i][j]=i+j-1; /* i+j+1 は対角行列 */
        }
    }
}
/*********************************************************
*      <  行列 r1j   input 関数   >                     *
*********************************************************/
 void input_r1j ( void )
{
    int i,j;
    for(i=0;i<=N;i++){
    for(j=0;j<=N;j++){
          /*  r1j[i]=i+3; */
              r1j[i]= 0;
        }
    }
}
/*********************************************************
*      <  行列 r2j   input 関数   >                     *
*********************************************************/
 void input_r2j ( void )
{
    int i,j;
    for(i=0;i<=N;i++){
    for(j=0;j<=N;j++){
         /*   r2j[i]=i+3; */
              r2j[i]= 0;
        }
    }
}
/*********************************************************
*      <  行列 c1i   input 関数   >                     *
*********************************************************/
 void input_c1i ( void )
{
    int i,j;
```

```
    for(i=0;i<=N;i++){
    for(j=0;j<=N;j++){
          /*  c1i[i]=i+3; */
              c1i[i]=0;
        }
    }
}
/*********************************************************
*      <  行列 c2i   input 関数   >                      *
*********************************************************/
 void input_c2i ( void )
{
    int i,j;
    for(i=0;i<=N;i++){
    for(j=0;j<=N;j++){
          /*  c2i[i]=i+3;  */
              c2i[i]=0;
        }
    }
}
/*************************************************************
*                    <   print_r1j 関数 >                    *
*************************************************************/
void  print_r1j ( void )
{
    int k;
    for(k=1;k<=N;k++){
            printf(" r1j[%d]=%12.10lf  ",k,r1j[k]);
    }
    printf("\n\n");
}
/*************************************************************
*                    <   print_r2j 関数 >                    *
*************************************************************/
void  print_r2j ( void )
{
    int k;
    for(k=1;k<=N;k++){
            printf(" r2j[%d]=%12.10lf  ",k,r2j[k]);
    }
    printf("\n\n");
}
/*************************************************************
*                    <   print_r3j 関数 >                    *
*************************************************************/
void  print_r3j ( void )
{
    int k;
    for(k=1;k<=N;k++){
            printf(" r3j[%d]=%12.10lf  ",k,r3j[k]);
    }
    printf("\n\n");
}
/*************************************************************
*                    <   print_r4j 関数 >                    *
*************************************************************/
void  print_r4j ( void )
{
    int k;
    for(k=1;k<=N;k++){
            printf(" r4j[%d]=%12.10lf  ",k,r4j[k]);
    }
    printf("\n\n");
}
/*************************************************************
*                    <   print_r5j 関数 >                    *
*************************************************************/
void  print_r5j ( void )
{
    int k;
    for(k=1;k<=N;k++){
            printf(" r5j[%d]=%12.10lf  ",k,r5j[k]);
    }
    printf("\n\n");
}
/*************************************************************
*                    <   print_r6j 関数 >                    *
*************************************************************/
void  print_r6j ( void )
{
    int k;
    for(k=1;k<=N;k++){
            printf(" r6j[%d]=%12.10lf  ",k,r6j[k]);
    }
    printf("\n\n");
```

```
}
/************************************************************
*                    <   print_c1i 関数 >                   *
************************************************************/
void print_c1i ( void )
{
    int k;
    for(k=1;k<=N;k++){
            printf(" c1i[%d]=%12.10lf \n",k,c1i[k]);
    }
    printf("\n");
}
/************************************************************
*                    <   print_c2i 関数 >                   *
************************************************************/
void print_c2i ( void )
{
    int k;
    for(k=1;k<=N;k++){
            printf(" c2i[%d]=%12.10lf \n",k,c2i[k]);
    }
    printf("\n");
}
/************************************************************
*                    <   print_m1 関数 >                    *
************************************************************/
 void print_m1 ( void )
{
    int k,l;
    for(k=1;k<=N;k++){
        for(l=1;l<=N;l++){
            printf("m1[%d][%d]=%12.10lf ",k,l,m1[k][l]);
        }
    printf("\n");
    }
    printf("\n");
}
/************************************************************
*                    <   print_m2 関数 >                    *
************************************************************/
 void print_m2 ( void )
{
    int k,l;
    for(k=1;k<=N;k++){
        for(l=1;l<=N;l++){
            printf("m2[%d][%d]=%12.10lf ",k,l,m2[k][l]);
        }
    printf("\n");
    }
    printf("\n");
}
/*---------------- 関数 ----------------------------*/
/************************************************
(0)              mm1_m2
************************************************/
void  mm1_m2( void )
{
    int a,b;
    for(a=1;a<=N;a++){
        for(b=1;b<=N;b++){
            m2[a][b] = m1[a][b] ;
     }
   }
}

/************************************************************
(1)            ftrace_c2i ( ) 副対角成分を c2i[N] に格納
************************************************************/
void  ftrace_c2i( void )
{
int k;
for(k=1;k<=N;k++){
c2i[k]=m1[k][k+1];
    if( k == N ) {
            c2i[k] = 0 ;
        }
    }
}
/*******************************************************
(2)              tc2i_c2i ( ) (2 乗する)
*******************************************************/
 void tc2i_c2i ( void )
{
    int j;
    for(j=1;j<=N;j++){
```

```
    c2i[j] = c2i[j] * c2i[j] ; /* c2i[j] のもとの内容は消えるので注 !! */
    }
}
/*********************************************
(3)       ttei_c2i ( )
*********************************************/
void ttei_c2i( void )
{
    int j;
        for(j=1;j<=N;j++){
            c2i[j] = -1 * c2i[j] ;
        }
}
/*************************************************
(4)             zettai_m1( void )
*************************************************/
void zettai_m1( void )
{
    int i,j;
    for(i=1;i<=N;i++){
        for(j=1;j<=N;j++){
            m1[i][j] = fabs ( m1[i][j] );
        }
    }
}
/*************************************************************
 (5-0)        <  行列 m1 で   sum_gyou 関数 (横にたす)  >
*************************************************************/
double sum_gyou ( int k )
{
    int l;
    double t=0;
/*    printf("いま %d 行目を横にたしています \n",k);  */
if (  k <=  N  ) {
        for(l=1;l<=N;l++){
            t = t + m1[k][l];
        }
        return t;
        }
else {
        abort(  );
     }
}
/*************************************************************
(5)      sum_wa_c1i(   )
   m1 の 1 行目からN行目までの行成分を足したものが c1i に入る。
*************************************************************/
void  sum_wa_c1i( void )
{
     int i;
     for(i=1;i<=N;i++){
         c1i[i]=sum_gyou( i );
     }
}
/*********************************************************
(6)                 double max_c1i ( void )
*********************************************************/
double max_c1i ( void )
{
    double max;
    int i;
    max = c1i[1];
    for( i = 1 ; i <=N ;i++){
        if ( c1i[i] > max ) max = c1i[i];
#ifdef DBGPRN
         printf ("max = %12.10lf \n",max );
#endif
    }
    return max;
}
/*************************************************************
(7)             Trace_m2 ( ) 対角成分を格納する
*************************************************************/
void  Trace_m2 ( void )
{
int k;
for(k=1;k<=N;k++){
c1i[k]=m2[k][k];
}
}
 /* r5j の計算  */
/************************************************
              ms0_r3j( )
************************************************/
void ms0_r3j( void )
{
```

```
     for ( j =1; j<=N;j++){
        r3j[j]= s0;
     }
}
/************************************************
              ms0_r4j( )
************************************************/
void ms0_r4j( void )
{
     for ( j =1; j<=N;j++){
        r4j[j]= s0;
     }
}/*******************************************************
              ar3j_r4j( )
*******************************************************/
void ar3j_r4j( void )
{
      for ( j = 1;j <=N ; j++){
         r4j[j]=r3j[j]+r4j[j];
      }
}
/*******************************************************
              mr4j_r5j( )
*******************************************************/
void mr4j_r5j( void )
{
      for ( j = 1;j <=N ; j++){
         r5j[j] = r4j[j];
      }
}
/************************************************
()       ttei_r4j ( )
************************************************/
void ttei_r4j( void )
{
    int j;
        for(j=1;j<=N;j++){
            r4j[j] = -1 * r4j[j] ;
        }
}
/************************************************
              ts1_r5j( )
************************************************/
void ts1_r5j( void )
{
    for ( j = 1 ;j <=N ;j++){
        r5j[j] = s0 * r5j[j];
    }
}
/************************************************
          void mr3j_r4j( void )
************************************************/
void mr3j_r4j( void )
{
      for ( j = 1;j <=N ; j++){
         r4j[j] = r3j[j];
      }
}
/************************************************
(???)      void ttei_c1i ( void  )
************************************************/
void ttei_c1i( void )
{
    int j;
        for(j=1;j<=N;j++){
            c1i[j] = -1 * c1i[j] ;
        }
}
/************************************************
(8-0)(15-0)       ttei_r5j ( )
************************************************/
void ttei_r5j( void )
{
    int j;
        for(j=1;j<=N;j++){
            r5j[j] = -1 * r5j[j] ;
        }
}
/*****************************************************
(8)   mc1i_m1 ( i,k,l )       c1i から m1 へ移す
*****************************************************/
void mc1i_m1 ( int i,int k, int l )
{
```

```
    m1[k][l]=c1i[i] ;
}
/*****************************************************
(9)   mm1_r1j ( int k,int l,int j )  m1 から r1j へ移す
*****************************************************/
void mm1_r1j ( int k,int l,int j )
{
     r1j[j]=m1[k][l];
}
/****************************************
(11)     逆数 gyaku_r1j ( void )
      ベクトル r1j の逆数
****************************************/
void gyaku_r1j( void )
{
     int j;
     for(j=1;j<=N;j++){
         if ( r1j[j] == 0 ){
             r1j[j] = 0 ;
         }
         else
         r1j[j] = 1/r1j[j] ; /* 逆数 をとる */
     }
}
/*****************************************************
 (12)       tc2i_r1j ( int i,int j )
*****************************************************/
 void tc2i_r1j( int i,int j )
{
        r1j[j] = c2i[i] * r1j[j] ;
}
/*****************************************************
 (13)       ar1j_m1 ( int i,int k,int l )
*****************************************************/
 void ar1j_m1( int i,int k,int l )
{
      m1[k][l] = r1j[i]+m1[k][l];
}
/*****************************************************
(14)           ar5j_m1( int j,int k,int l)
*****************************************************/
void ar5j_m1( int j,int k,int l )
{
    m1[k][l] = r5j[j] + m1[k][l];
}
/*****************************************************
               mr5j_r4j( )
*****************************************************/
void mr5j_r4j( void )
{
      for ( j = 1;j <=N ; j++){
         r4j[j] = r5j[j];
      }
}
/*****************************************************
               mr5j_r3j( )
*****************************************************/
void mr5j_r3j( void )
{
      for ( j = 1;j <=N ; j++){
         r3j[j] = r5j[j];
      }
}
/*****************************************************
(15)    void mm1_m1(int j,int l);
*****************************************************/
void mm1_m1( int j,int l )
{
     for ( i = 1;i<=N;i++){
         m1[i][l] = m1[i][j];
     }
}
/*****************************************************
(16)           void count ( int l )
*****************************************************/
void count ( int l )
{
     int k,count = 0 ;
     for ( k =1 ; k <=N ; k++){
      if ( m1[k][l] > 0 ){
            count = count +1 ;
         }
     }  /* for end */
     r1j[l] = count;
```

```
#ifdef DBGPRN
        printf("count = %d \n",count );
#endif
}
/*****************************************************
(17)          比較 関数 compare ( void )
*****************************************************/
void  compare ( void )
{
/*   r1j と r2j との比較をする。 r1j には N が入り r2j には
 *  1 から N までが入る。これはおおきいほうから何番目という意味である。
 */
int j; /* j は列 */
for( j = 1 ; j <=N ; j++ ){
r2j[j] = j;
}
        for( j = 1 ; j <=N ; j++ ){
            r5j[j] = ( r4j[j] + r3j[j] ) / 2 ;
#ifdef DBGPRN
printf("r5j[%d] = %12.10lf \n",j,r5j[j]);
#endif
        }
      for( j = 1 ; j <=N ; j++ ){
            if ( r1j[j] >= r2j[j] ){
#ifdef DBGPRN
            printf("r1j[%d] が r2j[%d] より大きい \n",j,j);
#endif
            r4j[j] = r5j[j] ;
            r5j[j] = ( r4j[j] + r3j[j]) / 2 ;
#ifdef DBGPRN
            printf("r4j[%d]=%12.10lf,r3j[%d]=%12.10lf,r5j[%d]=%12.10lf\n",j,r4j[j],j,r3j[j],j,r5j[j]);
#endif
            } /* if end */
            else  {
#ifdef DBGPRN
            printf("r2j[%d] が r1j[%d] より大きい \n",j,j);
#endif
            r3j[j] = r5j[j];
            r5j[j] = ( r4j[j] + r3j[j]) / 2;
#ifdef DBGPRN
     printf("r4j[%d]=%12.10lf,r3j[%d]=%12.10lf,r5j[%d]=%12.10lf\n",j,r4j[j],j,r3j[j],j,r5j[j]);
#endif
            }/* else if end */
     }/* for end */
}
/*****************************************************
                mr5j_r6j( int j )
*****************************************************/
void mr5j_r6j( int j )
{
         r6j[j] = r5j[j];
}
/*-------------------------------------------
         逆反復法 プログラム
      gyaku.c   1997 1 30 Mizuki Nakajima
-------------------------------------------*/
/*-------------------------------------------
        include
-------------------------------------------*/
#include<stdio.h>
#include<math.h>
#define  N  5  /* 注意 */
#include"register.h"
/*-------------------------------------------
    たとえば 4 行 4 列の場合 N = 5 とします。
    N = 5 としているのは連立方程式を解くとき
   結果を 5 列目に入れるためです。
    よって  N = 5 の場合
    4 x 4 の行列を解くこととなります。
-------------------------------------------*/
/*-------------------------------------------
        define
-------------------------------------------*/
/* #define DBGPRN */
/* #define RESULT */
     int i,j; /* i 行 j 列 */
     int c,cc,cnt  ; /* c,cc,cnt は動く count */
/*-------------------------------------------
    プロトタイプ宣言
-------------------------------------------*/
     void input    ( void );
     void input_test  ( void );
     void print_m1 ( void );
     void print_m2 ( void );
```

```
    void print_c1i( void );
    void print_c2i( void );
    void print_r1j( void );
    void print_r2j( void );
    void print_kekka ( void );
    void ms0_r1j   ( void  ); /* (2)   */
    void tr1j_m1   ( void  ); /* (3)   and  (19)  */
    void Trace_m1  ( void  ); /* (4)   */
    void as0_c1i   ( void  ); /* (5-2) */
    void gTrace_m1 ( void  ); /* (6)   */
    void mc2i_m1   ( int l ); /* (7-0) */
    void mm1_c1i   ( int l ); /* (8)   */
    void mm1_r1j   ( int k ); /* (9)   */
    void ts0_c1i   ( void  ); /* (10)  */
    void mr1j_s0   ( int j ); /* (11)  */
    void gyaku_s0  ( void  ); /* (12)  */
    void ts0_r1j   ( void  ); /* (13)  */
    void mr1j_m1   ( int i ); /* (14)  */
    void ms0_c1i   ( int i ); /* (16)  */
    void mm1_m2    ( void  ); /* (17)  and  (21)  */
    void mc1i_m1   ( int l ); /* (18)  */
    void am2_m1    ( void  ); /* (20)  */
    void mm1_c2i   ( int j ); /* (22)  */
    void mc2i_c1i  ( void  ); /* (23)  */
    void tc1i_c2i  ( void  ); /* (24)  */
    void ac2i_s2   ( void  ); /* (25)  */
    void gyaku_s2  ( void  ); /* (26)  */
    void sqrt_s2   ( void  ); /* (27)  */
    void ts2_c1i   ( void  ); /* (28)  */
/*----------------------------------------
          main ( )
----------------------------------------*/
void  main ( void )
{
   for ( cnt = 1 ; cnt <=N-1 ; cnt++){
    FILE *fpin ;
    FILE *fpout;
    if(( fpin = fopen("koyuu.dat","r")) == NULL ){
        printf("input file koyuu.dat が open できません !! \n");
        exit(1);
    }
    cc = 1 ;
    while ( fscanf(fpin,"%lf\n",&koyuu[cc]) == 1 ){
    /*   printf("koyuu[%d] = %f\n",cc,koyuu[cc]); */
         cc = cc + 1 ;
    }
    fclose(fpin);
     input  (  )  ;  /* 元の行列を入れる */
   /*  input_test  (  )  ;   /* 元の行列を入れる 仮 */
      s1 = koyuu[cnt]; /* (固有値) */
    /*  printf(" s1= %lf \n",s1); */
   /*  s1 = 0 ; /* 仮  */
    s0 = -1      ; /* (1) */
    ms0_r1j(  )  ; /* (2) */
    tr1j_m1(  )  ; /* (3) */
    Trace_m1(  ) ; /* (4) */
    s0 = s1      ; /* (5-1) */
    as0_c1i (  ) ; /* (5-2) */
    gTrace_m1 ( ); /* (6) これで入 I-A ができる */
    /* ----- 初期 vector----- */
        for(i = 1 ; i <= N-1 ; i++ ){
            m1[i][N] = 1;
        }
    for ( c = 1; c <= N-1 ; c++ ) {
        s0 = -1      ; /* (7) */
        mm1_c1i ( c ); /* (8) この c 列は移動する */
        mm1_r1j ( c ); /* (9) この c 行は移動する */
        ts0_c1i (   ); /* (10) */
        mr1j_s0 ( c ); /* (11) この c 列は移動する */
        gyaku_s0 (  ); /* (12) */
        ts0_r1j (   ); /* (13) */
        mr1j_m1 ( c ); /* (14) この c 行は移動する */
        s0 = 0       ; /* (15) */
        ms0_c1i( c ) ; /* (16) この c 行は移動する */
     mm1_m2 (   ) ; /* (17) */
```

```
        for ( j = 1 ; j <= N ; j++){
              mc1i_m1 ( j ); /* (18) */
        }
#ifdef DBGPRN
   printf(" (18) mc1i_m1 ( j ); \n ");
   print_c1i ( );
   print_m1( );
#endif
     tr1j_m1(   ); /* (19) */
     am2_m1 (   ); /* (20) */
     /*   mm1_m2 (   ); /* (21) */
    } /* for end */
       mm1_c2i( N ); /* (22) */
#ifdef DBGPRN
        printf(" 固有 vector 結果 \n");
        print_c2i( );/* test print */
#endif
    mc2i_c1i (  );  /* (23) */
    tc1i_c2i (  );  /* (24) */
    ac2i_s2  (  );  /* (25) */
    gyaku_s2 (  );  /* (26)  */
    sqrt_s2  (  );  /* (27) */
    ts2_c1i  (  );  /* (28) */
#ifdef RESULT
        printf(" 固有 vector 最終結果 \n");
        print_c1i( );/* test print */
#endif
    if(( fpout = fopen( "koyuuvec.dat","a" )) == NULL )
    {
         printf( "Can not open output file koyuuvec.dat!!\n");
         exit(1);
    }
    fprintf(fpout," No %d 's eigenvalue =>%lf \n",cnt,koyuu[cnt]);
    for ( i= 1; i <= N-1 ; i++){
        fprintf(fpout,"%lf\n",c1i[i]);
    }
    fclose(fpout);
  }/* cnt for loop */
}/* main end */
/*********************************************************
*        行列    input 関数
*********************************************************/
void input ( void )
{
    m1[1][1] = 4;     m1[1][2] = 3; m1[1][3] = 2;  m1[1][4] = 1;
    m1[2][1] = 3;     m1[2][2] = 3; m1[2][3] = 2;  m1[2][4] = 1;
    m1[3][1] = 2;     m1[3][2] = 2; m1[3][3] = 2;  m1[3][4] = 1;
    m1[4][1] = 1;     m1[4][2] = 1; m1[4][3] = 1;  m1[4][4] = 1;
}
/*********************************************************
*        行列    input_test 関数
*********************************************************/
void input_test ( void )
{
    m1[1][1] = 1;     m1[1][2] = 1; m1[1][3] = 0;  m1[1][4] = 0;
    m1[2][1] = -1;     m1[2][2] = 1; m1[2][3] = 0;  m1[2][4] = 0;
    m1[3][1] = 0;     m1[3][2] = 0; m1[3][3] = 1;  m1[3][4] = 0;
    m1[4][1] = 0;     m1[4][2] = 0; m1[4][3] = 0;  m1[4][4] = 1;
}
/*---------------- 関数 ----------------------------*/
/************************************************
(2)              ms0_r1j( void )
************************************************/
void ms0_r1j( void )
{
     int j;
     for ( j = 1 ; j <= N ; j++ ){
        r1j[j] = s0;
     }
#ifdef DBGPRN
    printf(" (2) ms0_r1j( void ) \n");
    printf("s0 = %lf \n",s0);
    print_r1j( );
#endif
}
/************************************************************
(3) and (19)  tr1j_m1( void )   掛け算関数
************************************************************/
void tr1j_m1 ( void )
{
    int k,l; /* k 行 l 列 */
```

```
#ifdef DBGPRN
    printf(" (3) or (19) tr1j_m1( void ) \n");
    print_r1j( );
    print_m1 ( );
#endif
    for( k = 1 ; k <= N-1 ; k++ ){
       for( l = 1 ; l <= N ; l++ ){
           m1[k][l] = r1j[l] * m1[k][l] ;
       }
    }
#ifdef DBGPRN
    print_m1 ( );
#endif
}
/************************************************************
(4)             Trace_m1 ( void ) 対角成分を格納する
************************************************************/
void  Trace_m1 ( void )
{
int k;
for( k = 1 ; k <= N-1 ; k++){
c1i[k] = m1[k][k];
}
#ifdef DBGPRN
    printf(" (4)  Trace_m1( void ) \n");
    print_m1( );
    print_c1i( );
#endif
}
/*******************************************************
 (5-2)         as0_c1i( void ) 足し算関数
*******************************************************/
void  as0_c1i ( void )
{
     int i;
#ifdef DBGPRN
    printf(" (5-2) as0_c1i( void ) \n");
    printf("s0 = %lf \n",s0 );
    print_c1i( );
#endif
     for(i = 1 ; i <= N-1 ; i++ ){
         c1i[i] = s0 + c1i[i];
     }
#ifdef DBGPRN
    print_c1i( );
#endif
}
/************************************************************
(6)              gTrace_m1 ( void ) 対角成分へ格納する
************************************************************/
void  gTrace_m1 ( void )
{
int k;
for( k=1;k<=N-1;k++ ){
     m1[k][k] = c1i[k];
}
#ifdef DBGPRN
     printf(" (6)  gTrace_m1( void ) \n");
     print_c1i( );
     printf("対角成分を参照 \n");
     print_m1( );
#endif
}
/*****************************************************
(7-0)    void mc2i_m1 ( int l ) ;
*****************************************************/
void mc2i_m1 ( int l )
{
int k;
#ifdef DBGPRN
    printf("(7-0) void mc2i_m1( %d ) \n",c);
    print_c2i ( );
    print_m1  ( );
#endif
          for(k=1;k<=N;k++){
    m1[k][l] = c2i[k] ;
          }
#ifdef DBGPRN
     print_m1( );
#endif
}
/*****************************************************
(8)    mm1_c1i ( int l )       m1 から c1i へ移す (copy)
*****************************************************/
void mm1_c1i ( int l )
```

```
{
int k;
   for( k = 1 ; k <= N-1 ; k++ ){
    c1i[k] = m1[k][l];
           }
#ifdef DBGPRN
    printf(" (7) Input_s0(-1)  \n");
    printf(" (8) mm1_c1i( %d ) \n",c);
    print_m1( );
    print_c1i( );
#endif
}
/*********************************************************
(9)  mm1_r1j ( int k )  m1 から r1j へ移す (copy) k は N-1 まで !!
*********************************************************/
void mm1_r1j ( int k )
{
int l;
#ifdef DBGPRN
    printf(" (9) mm1_r1j ( %d )  \n",c);
    print_m1( );
    print_r1j( );
#endif
        if ( k <= N-1 ){
   for( l = 1 ; l <= N ; l++ ){
    r1j[l] = m1[k][l];
           }
#ifdef DBGPRN
    print_r1j( );
#endif
       } /* if end */
}
/*******************************************************
(10)     ts0_c1i ( void )
*******************************************************/
void ts0_c1i( void )
{
    int j;
#ifdef DBGPRN
    printf(" (10) ts0_c1i( void ) \n");
    print_c1i( );
#endif
    for(j=1 ; j<=N-1 ;j++){
        c1i[j] = s0 * c1i[j] ;
    }
#ifdef DBGPRN
    printf("s0 = %lf \n",s0 );
    print_c1i( );
#endif
}
/*************************************************
(11)        mr1j_s0 ( int j ) 移動
*************************************************/
void mr1j_s0 ( int j )
{
#ifdef DBGPRN
    printf("  (11) mr1j_s0( %d ) \n" ,c );
    printf("s0 = %lf \n",s0);
#endif
       s0 = r1j[j];
#ifdef DBGPRN
    print_r1j( );
    printf("s0 = %lf \n",s0);
#endif
}
/*****************************************
(12)   逆数 gyaku_s0 ( void )
*****************************************/
void gyaku_s0( void )
{
#ifdef DBGPRN
    printf("  (12) gyaku_s0( void ) \n" );
    printf("s0 = %lf \n",s0);
#endif
   if ( s0 != 0 )
         s0 = 1 / s0 ; /* 逆数 をとる */
   else
         s0 = 0;
#ifdef DBGPRN
     printf("s0 = %lf \n",s0);
#endif
}
/*********************************************
(13)      ts0_r1j ( void )
*********************************************/
void ts0_r1j( void )
```

```
{
    int j;
#ifdef DBGPRN
    printf("  (13) ts0_r1j( void ) \n" );
    printf("s0 = %lf \n",s0);
    print_r1j( );
#endif
        for(j=1;j<=N;j++){
            r1j[j] = s0 * r1j[j];
        }
#ifdef DBGPRN
    printf("s0 = %lf \n",s0);
    print_r1j( );
#endif
}
/*****************************************************
(14)   mr1j_m1 ( int i )  r1j から m1 へ移す
*****************************************************/
void mr1j_m1 ( int i )
{
int l;
#ifdef DBGPRN
   printf(" (14) mr1j_m1( %d ) \n",c );
   print_r1j( );
   print_m1 ( );
#endif
for( l = 1 ; l <= N ; l++ ){
    m1[i][l] = r1j[l] ;
      }
#ifdef DBGPRN
    print_m1( );
#endif
}
/************************************************
(16)              ms0_c1i( int i )
************************************************/
void ms0_c1i( int i )
{
        c1i[i] = s0;
#ifdef DBGPRN
    printf(" (15) Input_s0 ( 0 )  \n");
    printf(" (16) ms0_c1i( %d ) \n",c );
    printf("s0 = %lf \n");
    print_c1i( );
#endif
}
/************************************************
(17) and (21)            mm1_m2 ( void )
************************************************/
void  mm1_m2( void )
{
    int i,j;
#ifdef DBGPRN
    printf(" (17) or (21) mm1_m2( void )\n");
    print_m1( );
#endif
    for(i=1;i<=N-1;i++){
        for(j=1;j<=N;j++){
            m2[i][j] = m1[i][j]  ;
     }
   }
#ifdef DBGPRN
    print_m2( );
#endif
}
/*****************************************************
(18)       mc1i_m1 ( int l )
*****************************************************/
void mc1i_m1 ( int l )
{
int k;
   for(k=1;k<=N-1;k++){
    m1[k][l] = c1i[k] ;
          }
}
/************************************************
(20)           am2_m1( void )
************************************************/
void  am2_m1( void )
{
    int i,j;
#ifdef DBGPRN
    printf(" (20) am2_m1( void ) %d loop \n",c);
    print_m2( );
    print_m1( );
```

```
#endif
   for(i=1;i<=N-1;i++){
        for(j=1;j<=N;j++){
            m1[i][j] = m2[i][j] + m1[i][j];
    }
   }
#ifdef DBGPRN
   print_m1( );
   printf(" \n %d loop end \n \n ",c);
#endif
}
/*******************************************************
(22)    void mm1_c2i( int j );
*******************************************************/
void mm1_c2i( int j )
{
     for ( i = 1 ; i <= N ; i++ ){
         c2i[i] = m1[i][j];
     }
#ifdef DBGPRN
    printf("(22) void mm1_c2i( %d ) \n",c);
    printf(" vector \n");
    print_c2i( );
    printf("-------------------------------------------------------\n");
#endif
}
/*****************************************************
(23)       mc2i_c1i ( void )
*****************************************************/
void mc2i_c1i(    )
{
int k;
#ifdef DBGPRN
    printf("(23) void mc2i_c1i( void ) \n");
    print_c2i( );
#endif
   for(k=1;k<=N;k++){
    c1i[k] = c2i[k] ;
           }
#ifdef DBGPRN
    print_c1i( );
#endif
}
/***********************************************
(24)      tc1i_c2i ( void )
***********************************************/
void tc1i_c2i( void )
{
    int j;
#ifdef DBGPRN
    printf("  (24) tc1i_c2i( void ) \n" );
    print_c1i( );
#endif
        for(j=1;j<=N;j++){
             c2i[j]=c1i[j]*c1i[j];
        }
#ifdef DBGPRN
    print_c2i( );
#endif
}
/***********************************************
(25)              ac2i_s2( void )
***********************************************/
void ac2i_s2( void )
{
     int j;
#ifdef DBGPRN
    printf(" (25) ac2i_s2( void ) \n");
    printf("s2 = %lf \n",s2);
    print_c2i( );
#endif
     for ( j = 1 ; j <= N ; j++ ){
        s2 = s2 + c2i[j];
     }
#ifdef DBGPRN
    printf("s2 = %lf \n",s2);
#endif
}
/***************************************
(26)   逆数 gyaku_s2 ( void )
***************************************/
void gyaku_s2( void )
{
#ifdef DBGPRN
```

```
    printf("  (26) gyaku_s2( void ) \n" );
    printf("s2 = %lf \n",s2);
#endif
   if ( s2 != 0 )
         s2 = 1 / s2 ; /* 逆数 をとる */
   else
         s2 = 0;
#ifdef DBGPRN
     printf("s2 = %lf \n",s2);
#endif
}
/*****************************************
(27)   void sqrt_s2 ( void )
*****************************************/
void sqrt_s2( void )
{
#ifdef DBGPRN
    printf("  (27) void sqrt_s2( void ) \n" );
    printf("s2 = %lf \n",s2);
#endif
   if ( s2 != 0 )
         s2 = sqrt( s2 );
   else
         s2 = 0;
#ifdef DBGPRN
     printf("s2 = %lf \n",s2);
#endif
}
/*********************************************
(28)      ts2_c1i ( void )
*********************************************/
void ts2_c1i( void )
{
    int j;
#ifdef DBGPRN
    printf("  (28) ts2_c1i( void ) \n" );
    printf("s2 = %lf \n",s2 );
    print_c1i( );
#endif
        for(j=1;j<=N;j++){
             c1i[j] = s2 * c1i[j];
        }
#ifdef DBGPRN
    print_c1i( );
#endif
}
/*--------------------関数終了--------------------------*/
/*************************************************************
*                    <   print_r1j 関数 >                    *
*************************************************************/
void  print_r1j ( void )
{
    int k;
    for(k=1;k<=N;k++){
            printf(" r1j[%d]=%lf ",k,r1j[k]);
    }
    printf("\n\n");
}
/*************************************************************
*                    <   print_r2j 関数 >                    *
*************************************************************/
void  print_r2j ( void )
{
    int k;
    for(k=1;k<=N;k++){
            printf(" r2j[%d]=%lf  ",k,r2j[k]);
    }
    printf("\n\n");
}
/*************************************************************
*                    <   print_c1i 関数 >                    *
*************************************************************/
void print_c1i ( void )
{
    int k;
    for(k=1;k<=N-1;k++){
            printf(" c1i[%d]=%lf \n",k,c1i[k]);
    }
    printf("\n");
}
/*************************************************************
*                    <   print_c2i 関数 >                    *
*************************************************************/
void print_c2i ( void )
```

```
{
    int k;
    for(k=1;k<=N-1;k++){
            printf(" c2i[%d]=%lf \n",k,c2i[k]);
    }
    printf("\n");
}
/*************************************************************
*                   <   print_m1 関数 >                      *
*************************************************************/
 void print_m1 ( void )
{
    int k,l;
    for(k=1;k<=N-1;k++){
        for(l=1;l<=N;l++){
            printf("m1[%d][%d]=%lf ",k,l,m1[k][l]);
        }
    printf("\n");
    }
    printf("\n");
}
/*************************************************************
*                   <   print_m2 関数 >                      *
*************************************************************/
 void print_m2 ( void )
{
    int k,l;
    for(k=1;k<=N-1;k++){
        for(l=1;l<=N;l++){
            printf("m2[%d][%d]=%lf ",k,l,m2[k][l]);
        }
    printf("\n");
    }
    printf("\n");
}
/*************************************************************
*         <   print_kekka 関数 >
*************************************************************/
 void print_kekka ( void )
{
    int k,l;
    l = N  ;
    for(k=1;k<=N-1;k++){
         printf("m1[%d][%d]=%lf \n",k,l,m1[k][l]);
    }
    printf("\n");
}
```

これよりヘッダファイル

```
/*--------------------------------------
             register.h
---------------------------------------*/
 /* ここで専用レジスタの定義をしている
     全て配列として格納している        */
 double s0;
 double s1;
 double s2;
 double p[N+1];
 double al[N+1];
 double be[N+1];
 double koyuu[N+1];
 double c1i[N+1];
 double c2i[N+1];
 double r1j[N+1];
 double r2j[N+1];
 double r3j[N+1];
 double r4j[N+1];
 double r5j[N+1];
 double r6j[N+1];
 double m1[N+1][N+1];
 double m2[N+1][N+1];
 double m3[N+1][N+1];
```

これより結果ファイル

```
--------al.dat---------
4.000000
 5.000000
 0.666667
 0.333333
```

このファイル (al.dat) は上の行から１行１列，２行２列.....N 行 N 列までの対角成分を表示している。

```
--------be.dat---------
-3.741657
 0.462910
 -0.089087
```

```
 0.000000
```
このファイル（be.dat）は上の行から１行２列，２行３列.....N-1 行 N 列までの
副対角成分を表示している。
```
--------koyuu.dat-------
8.2908591133
1.0000004114
0.4260221535
0.2831185415
```
このファイルは固有値（上から順に大きい固有値を表示している）
```
-------koyuuvec.dat------
 No 1 's eigenvalue =>8.290859
-0.656539
-0.577350
-0.428525
-0.228013
 No 2 's eigenvalue =>1.000000
-0.577350
-0.000000
0.577350
0.577351
 No 3 's eigenvalue =>0.426022
0.428525
-0.577350
-0.228014
0.656538
 No 4 's eigenvalue =>0.283119
0.228013
-0.577350
0.656538
-0.428526
```
このファイルは上の行から固有値の大きい順に
それぞれの固有値に対する固有ベクトルを表示している。
以上で固有値固有ベクトルが求まることがわかった。