連立一次方程式 (反復法)

山本昌志*

2005 年 12 月 9 日

1　はじめに

これまで，ガウス・ジョルダン法や LU 分解を用いた連立 1 次方程式の解を求める方法を学習した．これらの方法は，所定の回数計算すれば解が求まる直接法と言われる方法である．この方法は，必ず解が求まる反面，計算時間がかかることが多い．大きな連立方程式を計算するには不向きである．そこで，本日は，計算時間がこれに比べて格段に早い，反復法を学習する．

反復法に先立って，線形代数の復習をする．少しばかり，反復法の説明に必要なのである．反復法の使い方は難しくないので，このプリントを自分でちゃんと読めば理解できるはずである．

2　行列の対角化と応用

2.1　固有値と固有ベクトル

すでに行列の固有値と固有ベクトルについては，学習しているはずであるが，忘れている者も多いと思うので復習をしておく．ただし，ここでは取り扱いの面倒な行列，例えば複数の同じ固有値 (縮退) を持つような行列などは考えないものとする．

行列 $\boldsymbol{A}$ の固有値を λ，固有ベクトルを $\boldsymbol{x}$ とすると，それらには，次の関係がある．

$$\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{x} \tag{1}$$

つまり，行列 $\boldsymbol{A}$ はベクトルを変換するが，それが固有ベクトルの場合，固有値の乗じた変換しかしないのである．要するに，行列 $\boldsymbol{A}$ には特別の方向 $\boldsymbol{x}$ と大きさ λ があるのである．

固有値は，式 (1) を変形して，

$$(\boldsymbol{A} - \lambda\boldsymbol{I})\boldsymbol{x} = 0 \tag{2}$$

から求める．もちろん，この式から $\boldsymbol{x} = 0$ という解もあるが，これはつまらないので興味の対象外である．それ以外の有用な解は，

$$\det(\boldsymbol{A} - \lambda\boldsymbol{I}) = 0 \tag{3}$$

*国立秋田工業高等専門学校　電気工学科

の場合に生じる．この方程式を特性方程式という．$\boldsymbol{A}$ が n 次の正方行列であれば，これは n 次方程式になるので，n 個の解がある．またそれに応じて，n 個の固有ベクトルがある．

このようにして，何がうれしいかというと，線形の連立微分方程式を解いたりするときにこの方法は大変役に立つのである．

2.2 行列の対角化

固有ベクトルを列ベクトルとして，n 個並べる行列 $\boldsymbol{X}$ を考える．即ち，

$$\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \cdots, \boldsymbol{x}_n] \tag{4}$$

である．そして，対角成分に固有値を並べた対角行列

$$\Lambda = \begin{bmatrix} \lambda_1 & & & & \\ & \lambda_2 & & \boldsymbol{0} & \\ & & \lambda_3 & & \\ & \boldsymbol{0} & & \ddots & \\ & & & & \lambda_n \end{bmatrix} \tag{5}$$

を考える．

これらの行列から，

$$\boldsymbol{A}\boldsymbol{X} = \boldsymbol{X}\Lambda \tag{6}$$

が直ちに分かる．従って，行列 $\boldsymbol{A}$ は，固有ベクトルからなる行列を用いて

$$\boldsymbol{X}^{-1}\boldsymbol{A}\boldsymbol{X} = \Lambda \tag{7}$$

と対角化できる．この $\boldsymbol{X}$ を $\boldsymbol{A}$ の対角化行列と言い，これにより固有値が並ぶ行列に対角化できる．

この様に行列を変形して，なにがうれしいのか．それは，次に示すように，行列を何回も乗算するときに計算がうんと楽になるのである．

2.3 行列の乗算

先ほどの式は，

$$\boldsymbol{A} = \boldsymbol{X}\Lambda\boldsymbol{X}^{-1} \tag{8}$$

のように書くことができる．次に行列を n 回乗算することを，$\boldsymbol{A}^n$ と書くことにする．通常の記号とおなじである．すると，

$$\begin{aligned} \boldsymbol{A}^n &= \boldsymbol{A}\boldsymbol{A}\boldsymbol{A}\cdots\boldsymbol{A} \\ &= \boldsymbol{X}\Lambda\boldsymbol{X}^{-1}\boldsymbol{X}\Lambda\boldsymbol{X}^{-1}\boldsymbol{X}\Lambda\boldsymbol{X}^{-1}\boldsymbol{X}\Lambda\boldsymbol{X}^{-1}\cdots\boldsymbol{X}\Lambda\boldsymbol{X}^{-1} \\ &= \boldsymbol{X}\Lambda\Lambda\Lambda\cdots\Lambda\boldsymbol{X}^{-1} \\ &= \boldsymbol{X}\Lambda^n\boldsymbol{X}^{-1} \end{aligned} \tag{9}$$

となる．ここで，Λ は対角行列なので，その計算は簡単で，

$$\Lambda^n = \begin{bmatrix} \lambda_1^n & & & & \\ & \lambda_2^n & & 0 & \\ & & \lambda_3^n & & \\ & 0 & & \ddots & \\ & & & & \lambda_n^n \end{bmatrix} \tag{10}$$

となる．これことは，固有値と固有ベクトルを使ってベクトルを表現すると，その n 乗は感単に計算できると言っている．

3　反復法の基礎

ここの説明は，文献 [1] を参考にしました．これは，線形代数を実際にどのように使うか述べられた教科書で，詳しく書かれている．線形代数を実際に使う場合，一読することを進める．初めて私が線形代数の講義を受けたとき，あまりにも抽象的で，さっぱりわからなかった．その後，この教科書を読むことにより，なるほど便利なものであるとやっとわかったのである．

3.1　反復法とは

さて，いままで学習した直接法はしつこく計算すれば，必ず解が求まる．しかし，大きな連立方程式を計算するには不向きである．なぜならば，ガウス・ジョルダン法の計算回数は，方程式の元 n の n^3 に比例するため，大きな行列ではとたんに計算時間が必要になるからである．

実用的なプログラムでは，非常に大きな連立方程式を計算しなくてはならない．たとえば，私の研究室での計算でも 10 万元くらいは計算している．これを，ガウス・ジョルダン法で計算するの時間的にほとんど不可能である．そこで，これよりは格段に計算の速い反復法を用いている．ここでは，反復法を簡単に説明する．

当然ここでも，連立方程式

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{11}$$

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

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

$$\lim_{n\to\infty} \boldsymbol{x}_n = \boldsymbol{x} \tag{12}$$

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

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

$$\boldsymbol{S}\boldsymbol{x}_{k+1} = \boldsymbol{T}\boldsymbol{x}_k + \boldsymbol{b} \tag{13}$$

とすればよい．ここで，$\boldsymbol{x}_k$ が $\boldsymbol{\alpha}$ に収束するとする．すると，式 (13) と式 (11) を比べれば，$\boldsymbol{\alpha}$ と $\boldsymbol{x}$ は等しいことがわかる．すなわち，式 (13) で元の方程式 (11) を表した場合，$\boldsymbol{x}_k$ が収束すれば，必ず真の解 $\boldsymbol{x}$ に

収束するのである．別の解に収束することはなく，真の解に収束するか，発散するかのいずれかである．振動することはないのか?．それはよい質問である．興味がある人が調べてみてほしい．

3.2 解の収束の条件

先の説明で，式 (13) を使った反復法の場合，$\boldsymbol{x}_k$ の収束が重要であることがわかった．ここでは，これが収束する条件を示す．

真の解の場合，式 (13) は

$$\boldsymbol{S}\boldsymbol{x} = \boldsymbol{T}\boldsymbol{x} + \boldsymbol{b} \tag{14}$$

となる．この式 (14) から式 (13) を引くと，となる．

$$\boldsymbol{S}(\boldsymbol{x} - \boldsymbol{x}_{k+1}) = \boldsymbol{T}(\boldsymbol{x} - \boldsymbol{x}_k) \tag{15}$$

となる．ここで，$\boldsymbol{x} - \boldsymbol{x}_{k+1}$ や $\boldsymbol{x} - \boldsymbol{x}_k$ は，真の解からの差，すなわち，誤差を示している．k 回目の計算の誤差を $\boldsymbol{e}_k$ とすると，

$$\boldsymbol{e}_{k+1} = \boldsymbol{S}^{-1}\boldsymbol{T}\boldsymbol{e}_k \tag{16}$$

と表すことができる．この誤差ベクトル $\boldsymbol{e}_k$ がゼロに収束すれば，ハッピーなのだ．

ハッピーになるための条件を探すために，計算の最初の誤差を $\boldsymbol{e}_0$ とする．すると，

$$\begin{aligned}
\boldsymbol{e}_{k+1} &= \boldsymbol{S}^{-1}\boldsymbol{T}\boldsymbol{e}_k \\
&= \boldsymbol{S}^{-1}\boldsymbol{T}\boldsymbol{S}^{-1}\boldsymbol{T}\boldsymbol{e}_{k-1} \\
&= \boldsymbol{S}^{-1}\boldsymbol{T}\boldsymbol{S}^{-1}\boldsymbol{T}\boldsymbol{S}^{-1}\boldsymbol{T}\boldsymbol{e}_{k-2} \\
&= \boldsymbol{S}^{-1}\boldsymbol{T}\boldsymbol{S}^{-1}\boldsymbol{T}\boldsymbol{S}^{-1}\boldsymbol{T}\cdots\boldsymbol{S}^{-1}\boldsymbol{T}\boldsymbol{e}_0 \\
&= \left(\boldsymbol{S}^{-1}\boldsymbol{T}\right)^k \boldsymbol{e}_0
\end{aligned} \tag{17}$$

となる．この式の右辺に行列の k 乗の計算がある．このとき，2.3 節で得た結果を利用する．行列 $\boldsymbol{S}^{-1}\boldsymbol{T}$ の固有値と固有ベクトルで作る行列を，Λ と $\boldsymbol{X}$ とすると，式 (17) は

$$\boldsymbol{e}_{k+1} = \boldsymbol{X}\Lambda^k\boldsymbol{X}^{-1}\boldsymbol{e}_0 \tag{18}$$

となる．明らかに，計算回数 k を増やしていくと，誤差のベクトルは Λ^k に依存する．これは，

$$\Lambda^k = \begin{bmatrix} \lambda_1^k & & & \\ & \lambda_2^k & & \boldsymbol{0} \\ & & \lambda_3^k & \\ & \boldsymbol{0} & & \ddots \\ & & & & \lambda_n^k \end{bmatrix} \tag{19}$$

なので，$k \to \infty$ の場合，誤差 $\boldsymbol{e}_k$ がゼロに収束するためには，固有値すべてが $|\lambda_i| < 1$ でなくてはならない．そして，収束の速度は，最大の固有値 $\max|\lambda_i|$ に依存する．この絶対値が最大の固有値をスペクトル半径と言う．

ここで言いたいのは，連立方程式を式 (13) の反復法で計算する場合，結果が真の値に収束するためには，行列 $\boldsymbol{S}^{-1}\boldsymbol{T}$ の最大固有値の絶対値が 1 以下でなくてはならないと言うことである．

最大固有値が 1 以下になる行列の条件を探すことは難しい．また，予め行列 $\boldsymbol{S}^{-1}\boldsymbol{T}$ の最大固有値を計算することも考えられるが，それもかなりの計算量が必要で，反復法を使って計算時間を短縮するメリットが無くなってしまう．このようなことから，反復法はとりあえず試してみて，発散するようであれば他の方法に切り替えるのが良いだろう．後で述べる SOR 法の加速緩和係数 ω を 1 以下にするという方法もある．

4 ヤコビ法

4.1 計算方法

計数行列 $\boldsymbol{A}$ の対角行列を反復計算の行列 $\boldsymbol{S}$ としたものがヤコビ (Jacobi) 法である．ここでも，ヤコビは顔を出す．ヤコビ法では，係数行列を

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & 0 & 0 & \dots & 0 \\ 0 & a_{22} & 0 & \dots & 0 \\ 0 & 0 & a_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a_{nn} \end{bmatrix} + \begin{bmatrix} 0 & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & 0 & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & 0 & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & 0 \end{bmatrix} \tag{20}$$

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

$$\boldsymbol{S}^{-1} = \begin{bmatrix} a_{11}^{-1} & 0 & 0 & \dots & 0 \\ 0 & a_{22}^{-1} & 0 & \dots & 0 \\ 0 & 0 & a_{33}^{-1} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a_{nn}^{-1} \end{bmatrix} \tag{21}$$

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

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

とすると，k+1 番目の解は

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

と計算できる．これが，ヤコビ法である．行列の形で表すと

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{D}^{-1}\left\{\boldsymbol{b} - (\boldsymbol{A} - \boldsymbol{D})\,\boldsymbol{x}^{(k)}\right\} \tag{23}$$

となる．ここで，$\boldsymbol{D}$ は係数行列 $\boldsymbol{A}$ の対角成分から作った対角行列である．

4.2 収束条件

$\boldsymbol{A}$ が対角優位な行列の場合，ヤコビ法の $\boldsymbol{S}^{-1}\boldsymbol{T}$ の最大固有値は 1 以下になることが分かっている[1]．対角優位行列ならば，ヤコビ法は収束するのである．十分条件ではあるが，これは使える．なぜならば，自然科学の計算でお目にかかる多くの行列はこの性質を満たしているからである．

5 ガウス・ザイデル法

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

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

$$\begin{aligned} x_i^{k+1} = a_{ii}^{-1}\Big\{b_i - (a_{i1}x_1^{(k+1)} + a_{i2}x_2^{(k+1)} + a_{i3}x_3^{(k+1)} + \cdots + a_{ii-1}x_{i-1}^{(k+1)} + \\ a_{ii+1}x_{i+1}^{(k)} + a_{ii+2}x_{i+2}^{(k)} + a_{ii+3}x_{i+3}^{(k)} + \cdots + a_{in}x_n^{(k)})\Big\} \end{aligned} \tag{24}$$

とするのである．実際の計算では，k+1 番目の解は

$$\begin{aligned} x_1^{(k+1)} &= a_{11}^{-1}\left\{b_1 - \left(a_{12}x_2^{(k)} + a_{13}x_3^{(k)} + a_{14}x_4^{(k)} + \cdots + a_{1n}x_n^{(k)}\right)\right\} \\ x_2^{(k+1)} &= a_{22}^{-1}\left\{b_2 - \left(a_{21}x_1^{(k+1)} + a_{23}x_3^{(k)} + a_{24}x_4^{(k)} + \cdots + a_{2n}x_n^{(k)}\right)\right\} \\ x_3^{(k+1)} &= a_{33}^{-1}\left\{b_3 - \left(a_{31}x_1^{(k+1)} + a_{32}x_2^{(k+1)} + a_{34}x_4^{(k)} + \cdots + a_{3n}x_n^{(k)}\right)\right\} \\ &\vdots \\ x_n^{(k+1)} &= a_{nn}^{-1}\left\{b_n - \left(a_{n1}x_1^{(k+1)} + a_{n2}x_2^{(k+1)} + a_{n3}x_3^{(k+1)} + \cdots + a_{nn-1}x_{n-1}^{(k+1)}\right)\right\} \end{aligned} \tag{25}$$

と計算できる．これが，ガウス・ザイデル法である．

このガウス・ザイデル法は，k 番目と k+1 番目の解を混ぜて使うという，大胆なことをやっているが，研究の結果，収束条件はヤコビ法とほとんど同じと言うことである．ヤコビ法と比べてどちらが良いかというと

- メモリーの節約を考えた場合，ガウス・ザイデル法に軍配が上がる．

[1] Gershgorin の定理を使う．

- 計算速度では，ガウス・ザイデル法の方が早いと思われる．

となる．ヤコビ法を使うよりは，ガウス・ザイデル法を使う方が良いであろう．

6 SOR 法

ここでは，より高速な逐次加速緩和法 (SOR 法:Successive Over-Relaxation) について説明する．このでの説明は，文献 [2] を参考にした．この教科書には，行列の計算テクニックが多く欠かれているので便利で，このような計算をする人は参考書として持っておくのが良いだろう．

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

具体的な計算手順は，次のようにする．ここでは，ガウス・ザイデル法の式 (26) を用いて，得られた近似解を $\tilde{x}_i^{(k+1)}$ としている．

$$
\begin{aligned}
\tilde{x}_1^{(k+1)} &= a_{11}^{-1}\left\{b_1 - \left(a_{12}x_2^{(k)} + a_{13}x_3^{(k)} + a_{14}x_4^{(k)} + \cdots + a_{1n}x_n^{(k)}\right)\right\} \\
x_1^{(k+1)} &= x_1^{(k)} + \omega\left(\tilde{x}_1^{(k+1)} - x_1^{(k)}\right) \\
\tilde{x}_2^{(k+1)} &= a_{22}^{-1}\left\{b_2 - \left(a_{21}x_1^{(k+1)} + a_{23}x_3^{(k)} + a_{24}x_4^{(k)} + \cdots + a_{2n}x_n^{(k)}\right)\right\} \\
x_2^{(k+1)} &= x_2^{(k)} + \omega\left(\tilde{x}_2^{(k+1)} - x_2^{(k)}\right) \\
\tilde{x}_3^{(k+1)} &= a_{33}^{-1}\left\{b_3 - \left(a_{31}x_1^{(k+1)} + a_{32}x_2^{(k+1)} + a_{34}x_4^{(k)} + \cdots + a_{3n}x_n^{(k)}\right)\right\} \\
x_3^{(k+1)} &= x_3^{(k)} + \omega\left(\tilde{x}_3^{(k+1)} - x_3^{(k)}\right) \\
&\vdots \\
\tilde{x}_n^{(k+1)} &= a_{nn}^{-1}\left\{b_n - \left(a_{n1}x_1^{(k+1)} + a_{n2}x_2^{(k+1)} + a_{n3}x_3^{(k+1)} + \cdots + a_{nn-1}x_{n-1}^{(k+1)}\right)\right\} \\
x_n^{(k+1)} &= x_n^{(k)} + \omega\left(\tilde{x}_n^{(k+1)} - x_n^{(k)}\right)
\end{aligned}
\tag{26}
$$

これが，SOR 法である．

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

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

7　初期値と計算の終了

良い初期値が与えられれば，計算は早く収束するだろう．ただ，良い初期値というものがなかなか分からない．問題を考えて，あまり見当違いのない初期値を与えるのが良いだろう．収束は早いので，初期値を複雑にしない方が良い．

次に計算の終了判定であるが，十分，真の解に近づいたときに，計算をうち切るのであるが，その見極めが重要である．ここでは，2つの方法をしてしておく．収束判定のパラメーターとして，十分小さい ε をつかう．

まず，はじめに示すのが，平均的な修正量を考える場合である．以下の条件が成立したときに計算を止める．

$$\frac{\sum_{i=1}^{n}\left|x_i^{(k+1)}-x_i^{(k)}\right|}{\sum_{i=1}^{n}\left|x_i^{(k+1)}\right|}<\varepsilon \tag{27}$$

次に最大の修正量を考える場合である．これは，以下の条件が成立したときに計算を止める．

$$\max\left|\frac{x_i^{(k+1)}-x_i^{(k)}}{x_i^{(k+1)}}\right|<\varepsilon \tag{28}$$

参考文献

[1] Gilbert Strang. 線形代数とその応用. 産業図書株式会社, 1992.

[2] 戸川隼人. マトリックスの数値計算. オーム社, 1990.