連立１次方程式 II,

— 反復法 —

桂田 祐史

2002 年 5 月 7 日, 2011 年 11 月 6 日, 2017 年 6 月 4 日

目 次

第 1 章　序 **3**

1.1 反復法とは . . . 3

1.1.1 概説 . . . 3

1.1.2 個人的経験に基づく感想 . . . 3

1.2 なぜ反復法なのか . . . 5

第 2 章　定常反復法 **6**

2.1 短いイントロ . . . 6

2.2 原理 . . . 6

2.2.1 不動点型の方程式 . . . 6

2.2.2 線形漸化式 $x_{k+1} = Mx_k + c$ の収束条件 . . . 7

2.2.3 M, c の選び方 . . . 8

2.3 古典的定常反復法 . . . 9

2.3.1 Jacobi 法 . . . 9

2.3.2 Gauss-Seidel 法 . . . 9

2.3.3 SOR 法 . . . 10

2.3.4 収束を保証する条件 . . . 10

2.4 数値実験: Poisson 方程式に対する差分法 . . . 11

第 3 章　CG 法 — 代表的な非定常反復法 **16**

3.1 序 . . . 16

3.2 素朴な CG 法の原理 . . . 16

3.3 CG 法の計算量 . . . 25

3.4 CG 法の収束 . . . 25

3.5 前処理付き CG 法 . . . 26

3.6 2 次元長方形領域における熱方程式に対する PCG 法 . . . 28

3.6.1 ICCG(1,1) 法 . . . 29

3.6.2 MICCG(1,1) 法 . . . 29

3.7 CG 法の参考文献 (日本語で読めるもの) . . . 29

付 録 A 内積空間、Hilbert 空間の復習 **30**

A.1 Gram-Schimidt の直交化法 . . . 30

A.2 射影定理 . . . 30

付 録 B CG 法のプログラム例 **32**

B.1 MATLAB で遊ぶ . . . 32

B.2 素朴な CG 法の C プログラム . . . 32

B.3 高橋版 . . . 36

付 録 C Lanczos の原理 **38**
C.1 高橋 38
C.2 杉原・室田 [13] 38
C.3 森・杉原・室田 41

第1章　序

ここに書いてあることは、ずいぶん前に書いたことで、その後知ったことも多いし、書き換える必要性を切に感じているのだけれど、時間がない…
もともと素人の書くことであるし、あまり信じないように。
(素人の甘え？それは否めないけれど、玄人が分かるように書いて欲しい。)

1.1　反復法とは

1.1.1　概説

> 大規模な数値シミュレーションに欠かせない連立 1 次方程式の解法として、反復法と総称されるものがある。それについて概観しよう。

連立 1 次方程式 $Ax = b$ の解法には、消去法をその典型とする**直接法** (丸め誤差がなければ、有限回の演算で真の解が求まるような方法) 以外に、**反復法** (**iterative method**) と呼ばれる一群の解法がある。これは、適当な漸化式によって条件

$$\lim_{k\to\infty} x_k = x \equiv \text{“}Ax = b \text{ の真の解”}$$

を満す列 $\{x_k\}_{k\geq 1}$ を順次生成してゆき、x_k が十分真の解に近くなったところで反復を打ち切り、x_k を x の近似解に採用する、という方法である。
無限回の計算をしないと真の解が得られないような方法に何の利点があるか？と不思議に思う人もいようが、

- どうせ丸め誤差のある数値計算をするのならば、直接法といえども、正確な解ではなく、近似解しか得られない。
- 係数行列が素である場合や、あらかじめ良い近似解が分っている場合など、反復法によって、かなり少ない反復回数で十分な精度の近似解が得られることがある。
- 上記二つから想像されるように、反復法によって、直接法よりずっと少ない計算量で、同程度の精度を持った解が得られることがある — この場合は反復法を使わにゃ損である。

ということである。

1.1.2　個人的経験に基づく感想

(書いてあることをうのみにしないように…)

注意 1.1.1 (密かに筆者が考えるに — そのー) 実際に自分で数値計算をしてみるまでは、上のような曖昧な印象しか持っていなかったのだが、世の中どうやらそんな生易しいものではなくて、

反復法でなければやっていけない

というのが本当らしい。すでにこの本でも見たように、多次元問題において陰解法を採用した場合に現れる大規模連立 1 次方程式を解くのは骨で、消去法で解くと (メモリーはもちろんのこと) 手間がかかって陽解法 (前進 Euler 法) をしのぐことすら簡単ではないようである。

反復法を採用することによって初めて数値シミュレーションの世界が開ける

というわけで、連立 1 次方程式の解法に研究者人生を賭けた人達がたくさんいる。そうだったんだ (納得)。 ■

反復法は、以下説明するように、**定常反復法** (線型反復法ともいう) と**非定常反復法**の二つに大別される。

注意 1.1.2 (反復法の二大勢力の微妙な関係？) 人によっては — 具体的には古くからの定常反復法のファンの一部は — 「非定常反復法」は反復法ではない！と主張している。共役勾配法などの非定常反復法は、有限回の演算で厳密解が求まるという直接法的な性質を持っているし、計算手順もあまり簡単な「漸化式」ではないので、定常反復法ほど「反復」という匂いがしない、ということが理由としてあげられている。しかし、かといって「非定常反復法」は反復法ではなく直接法であるかと言うと、そうは言い切れないように思う (真の解が得られるはずの反復回数よりもずっと少ない回数で実際には反復を打ち切るのが普通だから)。筆者としては、名前はどうでも構わない。そんなに目くじら立てるほどのことではないように思うのだが (小声でつぶやく)。

注意 1.1.3 (密かに筆者が考えるに — その二) 定常反復法は多くの本に載っているが、長い間実際に使ってみたことはなかった。世の中には使っている人も結構いるようなので、「なかなか良いものなのかもしれない」と漠然と感じていたが、自分で実験し、真面目に考えるようになって大いに疑問を感じるようになった。

定常反復法は過去の遺物なのではないか？

2 次元長方形領域における熱伝導方程式の初期値境界値問題 (境界条件は Dirichlet) を θ 法で離散化した連立 1 次方程式を解いてみたところ、

- 分割数が小さい場合 (20×20 など) は直接法 (Gauss の消去法に基づく LU 分解を利用) が速い。
- CG 法の反復はかなり速く収束し、分割数が大きくなると直接法よりかなり速く解ける
- SOR 法は速さではいつも最下位

となった。この一つの例で結論することは出来ないが、次のように考えるようになった。

- 定常反復法の利点は
  1. 直接法と比べて少ないメモリーで動く (直接法では不可能なサイズの問題を扱える)
  2. プログラミングが簡単 (行列を作る必要すらない)
  3. 十分誤差の小さい初期値を知っている場合には比較的簡単に良い近似解が得られる[1]

  ということに集約できる。

[1] 例えば非定常問題で時間刻み幅が小さい場合に、前ステップにおける解は良い近似解になっている。しかし刻み幅が小さくなると、行列のスペクトル半径も大きくなりがちなので、この点はあまり簡単ではない。

- 非定常反復法があるので「少ないメモリーで動く」だけでは採用する理由に乏しい。
- 消去法は意外と優れている。メモリーさえ十分あれば、速度では定常反復法につねに勝るし、問題のサイズが小さければ非定常反復法に勝る場合もある。もちろんタフである。

(実際には) 偏微分方程式の数値シミュレーションでは非定常反復法を使うべし

ということを書いたわけだけど、今から見ると向こう見ずと言うか、随分単純化しているね。定常反復法も進歩していて、実際に使い道のある場合があるみたい。一方、正値対称な場合はやはり CG 法の系統が決定版だと思われる。そうでない場合の非定常反復法は、相変わらず、新しい方法がわらわらと湧いてきていて、素人はついていけない…いずれにしても、どういう場合に何を使うのが良いか、ある程度のことは、知っている人は知っているみたいなんだけど、それを解説してくれる文書に出会えていないという状況は変わっていない。

1.2 なぜ反復法なのか

以下の節で示すように、反復法の手続きでは、行列×ベクトルという計算が重要な部分を占めている。微分方程式の数値シミュレーションでは、未知数の個数 N がとても大きい連立 1 次方程式が現われるが、その行列は成分の多くが 0 であるという、いわゆる**疎行列** (**sparse matrix**) であることがほとんどである (そうでないと、そもそもどうやっても解けないことが多い — 例えば、実際のコンピューターで係数行列を記憶することすら出来なくなることもありうる)。この場合、疎性を有効に活用しないと、解けない or 解けても効率が悪い。疎性を利用すると、例えば行列×ベクトルという計算は極めて効率的に遂行可能になるのである。

計算時間、主記憶容量についての考察 最近では $N = 10^8$ という問題も解かれる (矢川・青山 [18] などを見よ)。この場合、係数行列の成分の個数は $N^2 = 10^{16} = 10\,\mathrm{P}$ (ペタ) $= 10,000,000\,\mathrm{G}$ (ギガ) であるから、疎性を利用せずに現在のコンピューターの主記憶中に記憶するのは (0 であることが分かっている部分まで馬鹿正直に記憶するのは) 困難である。計算時間についても、疎性を利用せずに Gauss の消去法で解くと、必要となる乗除算の回数は 約 $N^3/3 = 10^{24}/3$ 回にもなるから、 1 T FLOPS= 10^{12} FLOPS のスーパーコンピューターで計算して $(10^{24}/3)/10^{12}$ 秒 $= 10^{12}/3$ 秒 $= 10,562.69\cdots$ 年かかることになる (まったく非現実的である)[2]。特に Gauss の消去法では、係数行列の疎性をうまく利用できないことがある。上に説明したような反復法では、M を疎になるように取れることが多く、その場合は (M の形は変化しないので) 反復計算は非常に効率的に実行できる、つまり疎性は有効に利用できるわけである。

筆者が数値計算の研究集会に出席するようになった 1985 年頃には、$N = 10^6$ 程度ですごい計算だと思われていたようで、隔世の感がある。 ■

余談 1.2.1 (数学科計算機室でのある実話) 今 (2002 年) から 10 年程前の個人的な話。某友人が $N = 200$ の問題を Gauss の消去法で解いていたのを、これから解説する素朴な CG 法で解くようにしたところ、約 8 倍のスピード・アップになった。その後、係数行列の性質を調べて、係数行列のスケーリングをして調整したところ、さらに 25 倍のスピードアップになり、最初の Gauss の消去法と比べて、約 200 倍の効率向上になった。■

[2] FLOPS=floating point operations per seond. 浮動小数点演算の速度の単位。1 秒間に何回の浮動小数点演算が出来るかを表す。

第2章　定常反復法

桂田 [5] の第 6 章にも、ここに書いてないことを少し書いてある (どこかで時間を確保して整理したい…)。

2.1　短いイントロ

前置きがとても長いので一言先走っておく: 定常反復法とは、連立 1 次方程式を $Ax = b$ の解に収束する列 $\{x_k\}_{k\in\mathbf{N}}$ を、$x_{k+1} = Mx_k + c$ の形の漸化式で生成する算法のことである。時間に余裕のない授業では、不動点定理などと脱線しないで、$x_{k+1} = Mx_k + c$ という漸化式から説明をスタートさせるのだろうか…結局 T 先生の授業に収束 ??

定番の文献は、Varga [17] である。Hadjidimos [4] も詳しい。基本的なことは、森 [6] や森＆杉原＆室田 [8] にも書いてある。最近の話題については、仁木＆河野 [12] に詳しい。

また微分方程式に対する数値解法を扱ったテキストには、短い説明 (およびプログラム) が載っていることが多い。

2.2　原理

2.2.1　不動点型の方程式

少し回り道になるが、**不動点方程式**に対する**逐次近似法**(ちくじきんじほう)について説明しよう。

ある写像 $f\colon X \to X$ を用いて

$$x = f(x) \tag{2.1}$$

の形で表わされる方程式を不動点 (型の) 方程式と呼ぶことにする (この方程式の解 x は、f により動かない点であるから、f の**不動点**(ふどうてん)と呼ばれることによる)。このタイプの方程式はあちこちに登場し、この方程式に解が存在することを主張する定理は**不動点定理**と呼ばれ[1]、非常に重要である。

特に f (または f の巾) が縮小写像[2]である場合は、次の「縮小写像に関する不動点定理」が有名である[3]：任意に取った初期値 $x_0 \in X$ から初めて漸化式

$$x_{k+1} = f(x_k)$$

で列 $\{x_k\}_{k\in\mathbf{N}}$ を定めると、極限

$$x_\infty = \lim_{k\to\infty} x_k$$

[1] f の性質によってさまざまな不動点定理がある。ブラウワーの不動点定理、バナッハの不動点定理、シャウダーの不動点定理, etc.

[2] X が距離空間であって、その距離 d について、$d(f(x), f(y)) \le Kd(x, y)$ $(x, y \in X)$ を満す定数 $0 < K < 1$ が存在するとき、f は縮小写像であるという。

[3] バナッハの不動点定理ともいう。常微分方程式の初期値問題の解の一意存在を保証する定理の証明など使われるので、覚えている人もいるだろう。

が存在し、それは方程式 (2.1) の一意な解になる:

$$x_\infty = f(x_\infty).$$

この定理は解の存在を保証するだけでなく、近似解を構成するアルゴリズムまで提供している。つまり十分大きな k に対する x_k は x_∞ の良い近似になるのである。このようにして方程式の近似解を求める方法を、**逐次近似法**という。

2.2.2 線形漸化式 $x_{k+1} = Mx_k + c$ の収束条件

さて、ここで本来の問題に戻る。連立 1 次方程式 $Ax = b$ を

$$x = Mx + c \quad (M \text{ はある } N \text{ 次正方行列, } c \text{ はある } N \text{ 次元ベクトル, いずれも既知})$$

の形をした同値な方程式に変換する (M, c の選び方には色々な流儀があって、それで名前がついている)。これは上に説明した不動点型の方程式であって、逐次近似法の採用が考えられる。つまり初期値 x_0 を適当に選び、以下は漸化式

$$x_{k+1} = Mx_k + c$$

によって列 $\{x_k\}_{k=1}^\infty$ を生成したとき、もしも極限 $x_* = \lim_{k\to\infty} x_k$ が存在するならば、それは方程式の解である:

$$x_* = Mx_* + c, \quad \text{i.e.} \quad Ax^* = b$$

ということである (極限移行をするだけで証明できる)。

それでは、極限 $\lim_{k\to\infty} x_k$ の存在は何を仮定すれば保証されるであろうか？実は M のスペクトル半径が 1 より小さい、すなわち

$$r(M) \overset{\text{def.}}{=} \max\{|\lambda|; \lambda \text{ は } M \text{ の固有値 }\} < 1$$

である場合は、任意の x_0 に対して、$\{x_k\}_{k\in\mathbf{N}}$ の極限が存在することが簡単に分る。

証明

まず

$$r(M) < 1 \Leftrightarrow \lim_{k\to\infty} M^k = O$$

であることに注意する (M の Jordan 標準形を考えることにより分かる。例えば杉浦 [22] を参照せよ。)。

$$x_k = M^k x_0 + (M^{k-1} + M^{k-2} + \cdots + M + I)c$$

であるが、$r(M) < 1$ であることから $I - M$ は正則であることが分かり、

$$x_k = M^k x_0 + (I - M)^{-1}(I - M^k)c$$

となるから、

$$\lim_{k\to\infty} x_k = (I - M)^{-1}c. \quad \blacksquare$$

注意 2.2.1 時々、この逆、すなわち「任意の x_0 に対して、$\{x_k\}_{k\in\mathbf{N}}$ の極限存在するならば、$r(M)<1$」が成り立つと書いてある本があるが、それは厳密には嘘である。自明なケース

$$M=I,\quad c=0$$

などが反例として存在する。「任意の x_0 に対して共通の極限が存在する」と仮定を強くすれば、正しくなる。それを示すには、まず $x_0=0$ の場合を考えて、

$$\lim_{k\to\infty}(M^{k-1}+M^{k-2}+\cdots+M+I)c$$

が収束することから、任意の x_0 に対して

$$\lim_{k\to\infty}M^{k-1}x_0=0$$

が成り立つことが分かる。ゆえに

$$\lim_{k\to\infty}M^{k-1}=O.$$

これから

$$r(M)<1.\blacksquare$$

問 $r(M)<1$ のとき、$f(x)=Mx+c$ について Banach の不動点定理を適用して、$\lim_{k\to\infty}x_k$ の存在を証明せよ。

2.2.3 M, c の選び方

連立 1 次方程式 $Ax=b$ が与えられたとき、前小節の条件を満す M, c をどうやって探せば良いだろうか？A^{-1} の「良い近似」B があったとき、 $Ax=b$ を変形した方程式

$$x=(I-BA)x+Bb$$

において、 $I-BA$ は「小さい」と期待できる。そこで

$$M=I-BA,\quad c=Bb$$

と取ると良いだろう。すると、以下は A の近似逆行列をどうやって探すかが問題となる。

いま $A=(a_{ij})$ に対して、対角部分、狭義の下三角部分、狭義の上三角部分をそれぞれ D, E, F とする。すなわち

$$D=\begin{pmatrix}a_{11}&&&0\\&a_{22}&&\\&&\ddots&\\0&&&a_{nn}\end{pmatrix},$$

$$E=\begin{pmatrix}0&&&&0\\a_{21}&0&&&\\a_{31}&a_{32}&0&&\\\vdots&&\ddots&\ddots&\\a_{n1}&a_{n2}&\cdots&a_{n,n-1}&0\end{pmatrix},\quad F=\begin{pmatrix}0&a_{12}&a_{13}&\cdots&a_{1n}\\&0&a_{23}&&a_{2n}\\&&\ddots&\ddots&\vdots\\&&&0&a_{n-1,n}\\0&&&&0\end{pmatrix}$$

とする。

$B = D^{-1}$ とするのが Jacobi 法、$B = (D+E)^{-1}$ とするのが Gauss-Seidel 法である。一般の A に対して、これらの B が A^{-1} の良い近似となるわけではないが、差分方程式の係数行列は、正定値、狭義対角優であるなどの性質を持っているので、うまく行くのである (詳しいことは後述)。ここでは、A が正定値であるならば

$$a_{ii} = e_i^T A e_i > 0 \quad (i = 1, 2, \cdots, n)$$

であるから、D や $D+E$ は正則であることを注意しておこう。

2.3 古典的定常反復法

以下では

$$a_{ii} \neq 0 \quad (i = 1, 2, \cdots, n)$$

を仮定する。

2.3.1 Jacobi 法

$$Ax = b \quad \Leftrightarrow \quad (D+E+F)x = b \quad \Leftrightarrow \quad Dx = -(E+F)x + b \quad \Leftrightarrow \quad x = -D^{-1}(E+F)x + D^{-1}b$$

そこで

$$M \stackrel{\text{def.}}{=} -D^{-1}(E+F), \quad c \stackrel{\text{def.}}{=} D^{-1}b$$

とする方法が Jacobi 法である。つまり (成分で書くと)

$$\sum_{1 \leq j \leq n} a_{ij} x_j = b_i \quad (i = 1, 2, \cdots, n)$$

において、$a_{ii} \neq 0$ であることに注意して

$$x_i = \left(- \sum_{1 \leq j \leq n, j \neq i} a_{ij} x_j + b_i \right) / (a_{ii})^{-1} \quad (i = 1, 2, \cdots, n)$$

という不動点型の方程式を逐次代入で解く、ということ。

Jacobi 反復法の第 k 段

for i :=1 to n **do** $x_i^{(k+1)} := \dfrac{1}{a_{ii}} \left(- \displaystyle\sum_{j \neq i} a_{ij} x_j^{(k)} + b_i \right)$

2.3.2 Gauss-Seidel 法

$$Ax = b \quad \Leftrightarrow \quad (D+E+F)x = b \quad \Leftrightarrow \quad (D+E)x = -Fx + b \quad \Leftrightarrow \quad x = -(D+E)^{-1}Fx + (D+E)^{-1}b$$

そこで

$$M \stackrel{\text{def.}}{=} -(D+E)^{-1}F, \quad c \stackrel{\text{def.}}{=} (D+E)^{-1}b$$

とする方法が Gauss-Seidel 法である。

Jacobi 法の公式

$$\begin{aligned} a_{ii}x_i^{(k+1)} &= -\sum_{1\le j\le n, j\neq i} a_{ij}x_j + b_i \\ &= -\sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)} + b_i \end{aligned}$$

において、$x_j^{(k)}$ $(j = 1, \cdots, i-1)$ については、新しいステップにおける値 $x_j^{(k+1)}$ が求まっているのだから、これを使うように「改良」しよう、つまり

$$a_{ii}x_i^{(k+1)} = -\sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)} - b_i \quad (i = 1, 2, \cdot, n)$$

を基礎にして計算しよう、というのが Gauss-Seidel 法である。一見 Jacobi 法よりも複雑であるが、プログラム上はかえって単純になる。

Gauss-Seidel 反復法の第 k 段

for i :=1 to n **do** $x_i^{(k+1)} := \dfrac{1}{a_{ii}}\left(-\displaystyle\sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)} + b_i\right)$

2.3.3 SOR 法

$$\xi_{k+1} = D^{-1}(b - Ex_{k+1} - Fx_k) \tag{2.2}$$

$$x_{k+1} = x_k + \omega(\xi_{k+1} - x_k) \tag{2.3}$$

(今年度は書くの略)

$$M \overset{\text{def.}}{=} (I + \omega D^{-1}E)^{-1}[(1-\omega)I - \omega D^{-1}F], \quad c = \omega(D + \omega E)^{-1}b.$$

とするのが **SOR 法** (**逐次過緩和法**, successive overrelaxation method, 外挿 Liebmann 法) である。

ここで ω は**加速パラメーター** (あるいは**緩和パラメーター**) と呼ばれる正定数である。

$\omega = 1$ のとき、これは Gauss-Sidel 法に一致する。より詳しくは、

$$x^{(k+1)} = (1-\omega)x^{(k)} + \omega x_{\text{GS}}^{(k+1)} = u^{(k)} + \omega(x_{\text{GS}}^{(k+1)} - x^{(k)})$$

2.3.4 収束を保証する条件

定義 2.3.1 (狭義優対角行列)

$$|a_{ii}| > \sum_{j\neq i} |a_{ij}| \quad (i = 1, 2, \cdots, n)$$

を満すとき、A は (行に関して) **狭義優対角行列**であるという。

次の定理の証明は、森 [6] などにある (ただし (5) は森 [6] にはない)。

定理 2.3.2 (1) A が狭義対角優であるならば、Jacobi 法の反復行列のスペクトル半径は 1 より小さい。

(2) A が狭義対角優であるならば、Gauss-Seidel 法の反復行列のスペクトル半径は 1 より小さい。

(3) A が正値対称行列であるならば、Gauss-Seidel 法の反復行列のスペクトル半径は 1 より小さい。

(4) A が正値対称行列であるならば、$0 < \omega < 2$ なる加速パラメーターに関して、SOR 法の反復行列のスペクトル半径は 1 より小さい。

(5) A が狭義対角優であるならば、$0 < \omega < 2$ なる加速パラメーターに関して、SOR 法の反復行列のスペクトル半径は 1 より小さい。

A が正定値対称行列であるならば、$1 < \omega < 2$ なる範囲に最適値がある。簡単な問題の場合には、ω の最適な値が分かることもあるが (森・杉原・室田 [8] を見よ)、そういうことは稀である。

2.4 数値実験: Poisson 方程式に対する差分法

実際に、M のスペクトル半径がどのようになるかの例を、MATLAB による数値実験で見てみよう[4]。

以下のプログラムは、効率のことはほとんど考慮していない。

jacobi_m.m

```
function [m,c] = jacobi_m(a,b)
% a=L+D+U
  D=diag(diag(a));
%  U=triu(a,1);
%  L=tril(a,-1);
%  m=-inv(D)*(L+U);
  m=-D\(a-D);
  if (nargin>1) && (nargout>1)
      disp('setting c');
      c=D\b;
  end
```

gs_m.m

```
function [m,c] = gs_m(a,b)
% a=E+D+F
  F=triu(a,1);
  DpE=tril(a);
%  m=-inv(DpE)*F;
  m=-DpE\F;
  if (nargin > 1) && (nargout > 1)
      c=DpE\b;
  end
```

[4] ここで、MATLAB を採用したのは、スペクトル半径を “カンニング” するのに便利であるから (それさえなければ、C 言語でプログラムを書いても良いのだけど…)。

sor_m.m

```
function [m,c] = sor_m(a,omega,b)
% a=L+D+U
  U=triu(a,1);
  L=tril(a,-1);
  D=diag(diag(a));
% [n,dummy]=size(a);
%  m=(eye(n,n)+omega*(D\L))\((1-omega)*eye(n,n)-omega*(D\U));
   m=(D+omega*L)\(-omega*U+(1-omega)*D);
   if (nargin > 2) && (nargout > 1)
       disp('setting c');
       c=omega*((D+omega*L)\b);
   end
```

行列が対称であったり、何か特殊な性質を持っていれば、スペクトル半径の計算が工夫出来るかもしれないが、以下では愚直にすべての固有値を求めて、その絶対値の最大値を計算する。

spectral_radius.m

```
function r=spectral_radius(a)
  r=max(abs(eig(a))); % robust
%  r=abs(eigs(a,1));  % weak
% r=norm(a,2); % if a is symmetric
```

テスト用の行列としては、“人気のある” Poisson 方程式の Dirichlet 境界値問題を差分法で解く場合の、係数行列を使うことにする。2 次元の場合は演習用にとっておいて、ここでは 1 次元の場合

$$-u''(x) = f(x) \quad (x \in (0,1)), \quad u(0) = u(1) = 0$$

を調べる。差分方程式から導かれる連立 1 次方程式の係数行列は

$$h = \frac{1}{N}, \quad A_N = \frac{1}{h^2} \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} \in \mathbf{R}^{(N-1)\times(N-1)}.$$

である。

dirichlet_matrix_1d.m

```
function a=dirichlet_matrix_1d(N)
  h=1/N;
  a=2*eye(N-1,N-1)-diag(ones(N-2,1),1)-diag(ones(N-2,1),-1);
  a=a/h^2;
```

文献によると、SOR 法のパラメーター ω の最適値は

$$\omega_{\mathrm{opt}} = \frac{2}{1 + \sin(\pi h)}$$

であるとか (Yang-Gobbert [19])。

とりあえず $n = 10$ で解き比べ。

test_j_gs_sor1.m

```
% Jacobi 法, Gauss-Seidel 法, SOR 法
n=10
a=dirichlet_matrix_1d(n);
xe=ones(n-1,1);
b=a*xe;

disp('Jacobi 法');
[m,c]=jacobi_m(a,b);
test(a,xe,m,c);

disp('Gauss-Seidel 法');
[m,c]=gs_m(a,b);
test(a,xe,m,c);

disp('SOR 法');
omega=2/(1+sin(pi*1.0/n));
[m,c]=sor_m(a,omega,b);
test(a,xe,m,c);

function test(a,xe,m,c)
  fprintf('spec(m)=%f\n', spectral_radius(m));
  x=0;
  for i=1:1000
      x=m*x+c;
        err=norm(x-xe);
        if mod(i,10)==0
            fprintf('%d error=%e\n', i,err);
        end
        if (err<1e-6)
            fprintf('%d 回, err=%e: 収束しました。\n', i, err);
            break
        end
  end
end
```

$\|x - x_k\| < 10^{-6}$ となったら停止するようにしてある。Jacobi 法が 318 回、Gauss-Seidel 法が 160 回、SOR 法が 32 回で停止した。

test_j_gs_sor2.m

```
disp('Dirichlet condition');
num=10;
jacobi_sp=zeros(num,1);
gs_sp=zeros(num,1);
sor_sp=zeros(num,1);
disp('Jacobi');
for i=1:num
  jacobi_sp(i,1)=spectral_radius(jacobi_m(dirichlet_matrix_1d(2^i)));
end
disp('Gauss-Seidel');
for i=1:10
  gs_sp(i,1)=spectral_radius(gs_m(dirichlet_matrix_1d(2^i)));
end

disp('SOR');
for i=1:num
  h=1/(2^i);
  omega=2/(1+sin(pi*h));
  fprintf('N=%4d, omega=%f\n', 2^i, omega);
  sor_sp(i,1)=spectral_radius(sor_m(dirichlet_matrix_1d(2^i),omega));
end

for i=1:num
  fprintf('%4d %f %f %f\n', 2^i,jacobi_sp(i,1), gs_sp(i,1),sor_sp(i,1));
end
```

```
>> test_j_gs_sor
Dirichlet condition
Jacobi
Gauss-Seidel
SOR
N=   2, omega=1.000000
N=   4, omega=1.171573
N=   8, omega=1.446463
N=  16, omega=1.673514
N=  32, omega=1.821465
N=  64, omega=1.906455
N= 128, omega=1.952093
N= 256, omega=1.975754
N= 512, omega=1.987803
N=1024, omega=1.993883
   2 0.000000 0.000000 0.000000
   4 0.707107 0.500000 0.171573
   8 0.923880 0.853553 0.446463
  16 0.980785 0.961940 0.673514
  32 0.995185 0.990393 0.821465
  64 0.998795 0.997592 0.906455
 128 0.999699 0.999398 0.952093
 256 0.999925 0.999849 0.975754
 512 0.999981 0.999962 0.987803
1024 0.999995 0.999991 0.993883
>>
```

(MacBook Air 2011 で 10 秒程度)

- いずれの場合も M のスペクトル半径は 1 より小さい。
- Jacobi 法よりも Gauss-Seidel 法、Gauss-Seidel 法よりも SOR 法の方が M のスペクトル

半径が小さい。

- N が大きくなると ω_{opt} は 2 に近づく。
- N が大きくなると スペクトル半径は 1 に近づく。

(コンピューターの能力が低くて、小さい N に対して計算するしかなかった頃は、SOR 法がもてはやされたのも納得できるが、N が大きくなると収束がかなり遅くなり、「使い物になるのだろうか?」と心配になってくる。反復法の初期値が十分良ければ大丈夫なのだろうか?)

(工事中)

```
function a=neumann_matrix_1d(N)
  h=1/N;
  a=2*eye(N+1,N+1)-diag(ones(N,1),1)-diag(ones(N,1),-1);
  a(1,1)=1;a(N+1,N+1)=1;
  a=a/h^2;
```

第3章　CG 法 — 代表的な非定常反復法

3.1　序

ここでは係数行列が正値対称行列である場合に有効な**共役勾配法**(きょうやくこうばいほう)(**conjugate gradient method**, CG 法) を紹介する。

係数行列が正値対称である場合には、CG 法は決定版非定常反復法と言ってよいらしい。正値対称でない場合にも似たようなことはできないかと考えて、様々な試みがなされ、色々な方法が提唱されている、ということらしい。つまり非定常反復法は

1. 係数行列が正値対称な場合: CG 法 (前処理手法を併せた PCG 法も含める) がすべて
2. 係数行列が正値対称でない場合: 色々あるが — 決定版なし

という状況にある。

歴史覚え書き　CG 法は Hestenes と Stiefel によって約半世紀前に提案された比較的新しい解法である。

> M. R. Hestenes and E. Stiefel, Methods of conjugate gradients for solving linear systems, Journal of Research of the National Bureau of Standards, Vol. 49 (1952), pp.409–436.

発見当時はかなりもてはやされたが、丸め誤差に弱いという理由で一時は見捨てられかけた。

> J. K. Reid, On the method of conjugate gradients for the solution of large sparse system of linear equations, Academic Press, pp.231–254 (1971)

を契機として、大規模な疎行列用の解法として見直され、1970 年初頭から 1980 年代にかけての、前処理という補助テクニックの研究が成果をあげて (野寺 [9])、現在では微分方程式に伴う連立 1 次方程式用の解法として、最も有力であるとされている。近年は非対称行列に対しても拡張され、盛んに研究されている (例えば野寺 [24], 藤野・張 [23])。共役勾配法の歴史については Golub and Hanson [3] を見よ。

例え話による説明　1 次方程式 $ax = b$ の解は、2 次関数 $\phi(x) = \frac{1}{2}ax^2 - bx$ の最小点として特徴づけられるが、CG 法はこれを多次元化したものと考えられる。

3.2　素朴な CG 法の原理

基本的な問題設定

係数行列 A は N 次正定値対称行列とする。$b \in \mathbf{R}^N$ に対して、連立 1 次方程式

$$Ax = b \tag{3.1}$$

の解 $x^* \stackrel{\text{def.}}{=} A^{-1}b$ を求めることを考える。

注意 3.2.1 (対称な問題に限られていることについて)　1. 「対称な」問題は、案外多く、重要な位置を占めている。例えば、

$$\triangle u = 0 \quad \text{in } \Omega, \quad u = f \quad \text{on } \partial\Omega$$

のような Laplace 方程式の Dirichlet 境界値問題などは、対称な問題である。この種の問題をうまく離散化すると、実対称行列を係数とする連立 1 次方程式が現れる。

2. 係数行列が実対称行列でない場合は、少し面倒になるが、同じような方法がある。

以下では Euclid 内積

$$(x, y) \stackrel{\text{def.}}{=} y^T x = \sum_{j=1}^{N} x_j y_j$$

が重要な役目を果たすので、ノルムも内積と相性の良い Euclid ノルム

$$\|x\| \stackrel{\text{def.}}{=} \sqrt{(x, x)} = (x_1^2 + x_2^2 + \cdots + x_N^2)^{1/2}$$

を用いることにする。

記号の定義 (1)　この稿を通じて用いる (非標準的な) 記号を定義する。

(1) 連立 1 次方程式 (3.1) の解を x^* と書く。すなわち

$$x^* \stackrel{\text{def.}}{=} A^{-1} b.$$

(2) $x, y \in \mathbf{R}^N$ に対して

$$\langle x, y \rangle \stackrel{\text{def.}}{=} (Ax, y),$$

さらに、$x \in \mathbf{R}^N$ に対して

$$|\!|\!| x |\!|\!| \stackrel{\text{def.}}{=} \sqrt{\langle x, x \rangle}$$

とおく。

(3) $\phi\colon \mathbf{R}^N \to \mathbf{R}$ を

$$\phi(x) \stackrel{\text{def.}}{=} \frac{1}{2} |\!|\!| x - x^* |\!|\!|^2 = \frac{1}{2} (A(x - x^*), x - x^*)$$

で定義する。

次の二つの補題は明らかであろう。

補題 3.2.2 $\langle \cdot, \cdot \rangle$ は $\mathbf{R}^N$ の内積になる。それゆえ $|\!|\!| \cdot |\!|\!|$ は $\mathbf{R}^N$ のノルムになる。

証明

線形性は明らか。また対称性は一般に

$$(Ax, y) = y^T A x = (A^T y)^T x = (x, A^T y)$$

が成り立つことと、行列 A の対称性の仮定 $A^T = A$ から

$$\langle x, y \rangle = (Ax, y) = (x, A^T y) = (x, Ay) = (Ay, x) = \langle y, x \rangle$$

と証明できる。一方、実対称であることから適当な実直交行列 U が存在して

$$U^T A U = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_N \end{pmatrix}$$

と対角化できる。$x = Uy$ (つまり $U^T x =: y$) とおくと、

$$\langle x, x \rangle = (Ax, x) = (AUy, Uy) = (U^T AUy, y) = \sum_{j=1}^{N} \lambda_j (y_j)^2$$

となる。 A は正値ゆえ $\lambda_j > 0$ $(j = 1, \cdots, N)$ であるから、$\langle x, x \rangle \geq 0$. さらに

$$x = 0 \Leftrightarrow y = 0 \Leftrightarrow (Ax, x) = 0 \Leftrightarrow \langle x, x \rangle = 0. \blacksquare$$

補題 3.2.3 x^* は ϕ の一意的な最小点である。

証明

まず任意の $x \in \mathbf{R}^N$ について $\phi(x) = \dfrac{1}{2}\|x - x^*\|^2 \geq 0$. そして $x = x^* \Leftrightarrow \|x - x^*\| = 0 \Leftrightarrow \phi(x) = 0$.■

そこで、次のようなことを考える。

傾斜法

$$\phi(x_0) > \phi(x_1) > \phi(x_2) > \cdots$$

なる点列 $\{x_n\}_{n \geq 0}$ で、$\lim_{n \to \infty} x_n = x^*$ なるものを構成し、十分大きな番号 n に対して、x_n を x^* の近似とする。

記号の定義 (2) x_k を x^* の近似と考えるとき、

$$r_k \overset{\text{def.}}{=} b - Ax_k = A(x^* - x_k)$$

を**残差** (**residual**) と呼ぶ。

これから連立 1 次方程式を解くのであるから、x^* は未知であり、x が与えられても、$\phi(x)$ の値はすぐには計算できない。しかし

補題 3.2.4

$$\phi(x) = \frac{1}{2}(Ax, x) - (b, x) + \frac{1}{2}(b, x^*)$$

であり、定数項以外には x^* は含まれていないので、x^* を知らなくても、ϕ の変化量は計算できる。特に

$$\operatorname{grad} \phi(x) = Ax - b.$$

傾斜法の反復の 1 step (x_k から x_{k+1} を求めるところ) を次のように二つの手順に分解する:

逐次最小化法

次の手続きを $k = 0, 1, \cdots$ と行い、1 次元の最小化問題を繰り返し解くことによる傾斜法を**逐次最小化法**と呼ぶ。

1. 探索方向 $p_k \neq 0$ を定める。
2. x_k が求まっているとき、x_{k+1} を

$$x_{k+1} = x_k + \alpha_k p_k \tag{3.2}$$

の形として、α_k を $\phi(x_{k+1})$ が最小となるように選ぶ。

(結局、逐次最小化法では次式が成り立つ: $x_{k+1} = x_0 + \sum_{i=0}^{k} \alpha_i p_i$.)

このうち、第 2 段では、次のように解は簡単に求まる。実際、

補題 3.2.5 逐次最小化法のうちの第 2 段は次のように解ける。

$$\begin{aligned} \phi(x_k + \alpha p_k) &= \frac{1}{2}\|(x_k + \alpha p_k) - x^*\|^2 \\ &= \frac{1}{2}\left(\|x_k - x^*\|^2 - 2\alpha\langle x^* - x_k, p_k\rangle + \alpha^2\|p_k\|^2\right) \end{aligned}$$

は α に関する 2 次関数だから、

$$\alpha = \alpha_k \overset{\text{def.}}{=} \frac{\langle x^* - x_k, p_k\rangle}{\|p_k\|^2} = \frac{(r_k, p_k)}{(Ap_k, p_k)} \tag{3.3}$$

のとき最小になる。この条件 (3.3) は

$$\langle x^* - x_{k+1}, p_k\rangle = 0$$

と同値である。

証明

前半は簡単なので略。後半は

$$\langle x^* - x_{k+1}, p_k\rangle = \langle x^* - (x_k + \alpha_k p_k), p_k\rangle = \langle x^* - x_k, p_k\rangle - \alpha_k\langle p_k, p_k\rangle$$

による。■

よって、以下は第一段 (探索方向の決定) が問題として残る。一つの素朴な回答として次のものが考えられる。

寄り道 — SD 法

(3.4) $$p_k = -\operatorname{grad}\phi(x_k) = r_k$$

とする逐次最小化法を **steepest descent mathod** (最急降下法, SD 法と略記) と呼ぶ。しかし SD 法は必ずしもうまく行くとは限らない。隣り合う探索ベクトルについては $p_k \perp p_{k+1}$ と直交するが、 $p_0, p_1, p_2, \cdots$ が直交系になるとは限らない。一応

(3.5) $$\phi(x_{k+1}) \le \left(\frac{\kappa-1}{\kappa+1}\right)^2 \phi(x_k) \quad \text{従って} \quad \|x_k - x^*\| \le \left(\frac{\kappa-1}{\kappa+1}\right)^k \|x_0 - x^*\|$$

のような評価が成り立つが[a]、CG 法と比べるとあまり速い収束ではない。ここで κ は、ベクトルのノルムとしてユークリッド・ノルムを採用したときの、A の条件数である ($\kappa = \|A\|\|A^{-1}\|$)。(3.5) の証明は例えば、森・杉原・室田 [8] の演習問題 3.1 を見よ。

[a] これまで誤植をしていた。要注意。

CG 法では $\{p_k\}$ が次に定義する意味で直交関係を満たすようにする (これができるのがミソである):

(3.6) $$\langle p_i, p_j\rangle = 0 \quad (i \ne j) \qquad \text{i.e.} \qquad (p_i, Ap_j) = 0 \quad (i \ne j).$$

このような探索ベクトルを用いる方法を**共役方向法**と呼ぶ。

定理 3.2.6 (共役方向法の性質) A が正値対称行列で、$\{p_j\}_{j=0,1,\cdots,k-1}$ について

(3.7) $$p_j \ne 0 \quad (j = 0, 1, \cdots, k-1),$$

と

(3.8) (**共役直交性**) $$\langle p_i, p_j\rangle = 0 \quad (0 \le i < j \le k-1)$$

が成り立つとするとき、次の (1), (2) が成り立つ。

(1) $p_0, p_1, \cdots, p_{k-1}$ は 1 次独立である。

(2) 任意の $x_0 \in \mathbf{R}^N$ から始めて、$\{p_j\}$ を用いて逐次最小化法を行うとき、

(3.9) $$\phi(x_k) = \min_{x \in S_k} \phi(x),$$

ただし

(3.10) $$S_k \overset{\text{def.}}{=} x_0 + \operatorname{Span}(p_0, p_1, \cdots, p_{k-1}).$$

(3) (2) と同じ仮定の下で、

(3.11) $$(r_k, p_j) = 0 \quad (j = 0, 1, \cdots, k-1).$$

証明

(1)

$$\sum_{j=0}^{k-1} c_j p_j = 0, \quad c_j \in \mathbf{R} \quad (0 \leq j \leq k-1)$$

とする。 p_i との内積を取ると、共役直交性から

$$c_i \langle p_i, p_i \rangle = 0$$

が残るが、$p_i \neq 0$ であるから $\langle p_i, p_i \rangle \neq 0$ で、$c_i = 0$ が導かれる。

(2) $x^* - x_k$ が p_i $(i = 0, 1, 2, \cdots, k-1)$ と共役直交する、つまり x_k が x^* の S_k への (共役直交性の意味での) 直交射影になっていることが分かれば、

$$\|x^* - x_k\| = \min_{x \in S_k} \|x^* - x\|$$

が成立することは明らかである (ピタゴラスの定理より、斜辺は垂辺より長い)。以下、$x^* - x_k$ が p_i $(i = 0, 1, 2, \cdots, k-1)$ と共役直交することを示す。まず $i = k-1$ については、補題 3.2.5 より

$$\langle x^* - x_k, p_{k-1} \rangle = 0.$$

つぎに $i \leq k-2$ については、x_{i+1} を橋渡しに使って、

$$\langle x^* - x_k, p_i \rangle = \langle x^* - x_{i+1} + \sum_{j=i+1}^{k-1} \alpha_j p_j, p_i \rangle = \langle x^* - x_{i+1}, p_i \rangle + \sum_{j=i+1}^{k-1} \alpha_j \langle p_j, p_i \rangle = 0 + \sum_{j=i+1}^{k-1} \alpha_j 0 = 0.$$

(3) $f(t) \overset{\text{def.}}{=} \phi(x_k + t p_j)$ とおくと、$x_k + t p_j \in S_k$ であるから、(2) より $t = 0$ で最小となる。

$$0 = f'(0) = (\operatorname{grad} \phi(x_k), p_j) = -(r_k, p_j). \blacksquare$$

定理 3.2.7 A を正値対称行列とする。共役方向法においては、ある $k \leq N$ に対して $x_k = x^*$ となる。すなわち有限回の逐次最小化によって、解を得る。

証明

$\dim S_k = k$ であるから、$S_0 \subset S_1 \subset \cdots S_{N-1} \subset S_N = \mathbf{R}^N$. ゆえに、$\exists k$ s.t. $x^* \in S_k$. このとき $x_k = x^*$. ■

どうやって、共役直交な探索ベクトルを作るかであるが、CG 法では、残差ベクトルを Gram-Schimidt の直交化法によって、直交化したものを採用する (と考えることができる)。

補題 3.2.8 (CG 法に向けて (1)) A が正値対称行列、$x_0 \in \mathbf{R}^N$ とするとき、

$$\text{(3.12)} \qquad \begin{cases} r_j & \overset{\text{def.}}{=} & b - Ax_j, \\ p_j & \overset{\text{def.}}{=} & r_j - \displaystyle\sum_{i=0}^{j-1} \frac{\langle r_j, p_i \rangle}{\|p_i\|^2} p_i, \qquad (j = 0, 1, \cdots, k) \\ x_{j+1} & \overset{\text{def.}}{=} & x_j + \dfrac{\langle r_j, p_j \rangle}{\|p_j\|^2} p_j. \end{cases}$$

により $\{p_j\}_{j=0,1,\cdots,k}$ が計算できた (つまり $p_j \neq 0$ for $j = 0, 1, \cdots, k-1$) とすると、次の (1), (2) が成り立つ。

(1) $0 \leq i < j \leq k-1$ とするとき $\langle p_i, p_j \rangle = 0$

(2) $r_k = 0 \Leftrightarrow p_k = 0$.

証明

(1) $\{r_j\}$ に Gram-Schmidt の直交化を施したものが $\{p_j\}$ であるから明らか。

(2) ($\Longrightarrow$) は明らかである。 ($\Longleftarrow$) を示す。$p_k = 0$ とすると、

$$r_k = \sum_{i=0}^{k-1} \frac{\langle r_k, p_i \rangle}{\langle p_i, p_i \rangle} p_i$$

一方、定理 3.2.6 (3) により、$(r_k, p_i) = 0$ $(0 \leq i \leq k-1)$ であるから、

$$(r_k, r_k) = \sum_{i=0}^{k-1} \frac{\langle r_k, p_i \rangle}{\langle p_i, p_i \rangle} (r_k, p_i) = \sum_{i=0}^{k-1} \frac{\langle r_k, p_i \rangle}{\langle p_i, p_i \rangle} 0 = 0.$$

ゆえに $r_k = 0$. ■

$r_k = 0$ は $x_k = x^*$ と同値であるから、真の解が得られない限り、この反復は続けられることが分かる。ところが、$\mathbf{R}^N$ には 1 次独立なベクトルは最大で N 個しか取れないから、

$$\exists k \leq N \text{ s.t. } p_k = 0 \quad (\text{i.e. } r_k = 0)$$

となるはずである。つまり最大 N 回反復すれば真の解が得られることが分かる。

記号の定義 (3) $v \in \mathbf{R}^N$, $k \in \mathbf{N}$ に対して、

$$\text{(3.13)} \qquad \mathcal{K}_k(A, v) \overset{\text{def.}}{=} \text{Span}\,(v, Av, A^2v, \cdots, A^{k-1}v)$$

を A によって v から生成される **Krylov 部分空間**と言う。

補題 3.2.9 (CG 法に向けて (2)) A が正値対称行列、$x_0 \in \mathbf{R}^N$ とするとき、

$$\text{(3.14)} \qquad \begin{cases} r_j & \overset{\text{def.}}{=} & b - Ax_j, \\ p_j & \overset{\text{def.}}{=} & r_j - \displaystyle\sum_{i=0}^{j-1} \frac{\langle r_j, p_i \rangle}{\|p_i\|^2} p_i, \qquad (j = 0, 1, \cdots, k) \\ x_{j+1} & \overset{\text{def.}}{=} & x_j + \dfrac{\langle r_j, p_j \rangle}{\|p_j\|^2} p_j. \end{cases}$$

により $\{p_j\}_{j=0,1,\cdots,k}$ が計算できた (つまり $p_j \neq 0$ for $j = 0, 1, \cdots, k-1$) とすると、

(1)

$$\text{(3.15)} \qquad \mathrm{Span}\,(p_0, p_1, \cdots, p_k) = \mathrm{Span}\,(r_0, r_1, \cdots, r_k) = \mathcal{K}_{k+1}(A, r_0).$$

(2)

$$\text{(3.16)} \qquad p_k = r_k + \beta_{k-1} p_{k-1}, \quad \beta_{k-1} \overset{\text{def.}}{=} -\frac{\langle r_k, p_{k-1} \rangle}{\langle p_{k-1}, p_{k-1} \rangle}.$$

証明

(1) 最初の等式は Gram-Schmidt の直交化法においては明らかである。

念のため証明

帰納法を用いる。$r_0 = p_0$ であるから $k = 0$ のときは成立する。以下 $k = j$ のとき成立すると仮定すると、

$$p_{j+1} = r_{j+1} - \sum_{i=0}^{j} \frac{\langle r_j, p_i \rangle}{\langle p_i, p_i \rangle} p_i$$

において、右辺の $\sum$ の部分は

$$\mathrm{Span}\,(p_0, p_1, \cdots, p_j) = \mathrm{Span}\,(r_0, r_1, \cdots, r_j)$$

に属するから、

$$\mathrm{Span}\,(p_0, p_1, \cdots, p_j, p_{j+1}) = \mathrm{Span}\,(r_0, r_1, \cdots, r_j, r_{j+1}).$$

次に二つ目の等式を証明する。まず $k = 0$ のときは明らか。k まで成立すると仮定する。

$$r_{k+1} = b - Ax_{k+1} = b - Ax_0 + A(x_{k+1} - x_0) = r_0 + A(x_{k+1} - x_0),$$

において

$$x_{k+1} - x_0 \in \mathrm{Span}\,(p_0, p_1, \cdots, p_k) = \mathrm{Span}\,(r_0, r_1, \cdots, r_k) = \mathrm{Span}\,(r_0, Ar_0, \cdots, A^k r_0).$$

ゆえに $r_{k+1} \in \mathrm{Span}\,(r_0, Ar_0, \cdots, A^{k+1} r_0)$.

(2) $j \leq k-2$ ならば、(1) により $p_j \in \mathcal{K}_{j+1}(A, r_0)$. ゆえに

$$Ap_j \in \mathcal{K}_{j+2}(A, r_0) \subset \mathcal{K}_k(A, r_0) = \mathrm{Span}\,(p_0, p_1, \cdots, p_{k-1}).$$

(3.11)「r_k は p_j $(0 \leq j \leq k-1)$ と直交」により、$(r_k, Ap_j) = 0$. 言い替えると $\langle r_k, p_j \rangle = 0$. ゆえに p_k の定義式 ((16) の第 2 式) の $\sum$ のうち、最後の項のみが残る。 ■

では、CG 法の計算手順をまとめておこう。

CG 法のアルゴリズム

初期ベクトル $\boldsymbol{x}_0$ をとる ; 目標とする相対残差 ε を決める ;
$\boldsymbol{r}_0 := \boldsymbol{b} - A\boldsymbol{x}_0$; $\boldsymbol{p}_0 := \boldsymbol{r}_0$;
for $k := 0, 1, \cdots$ **until** $\|\boldsymbol{r}_k\| \leq \varepsilon\|\boldsymbol{b}\|$ **do**
begin
$\alpha_k := \dfrac{(\boldsymbol{r}_k, \boldsymbol{p}_k)}{(\boldsymbol{p}_k, A\boldsymbol{p}_k)}$;
$\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$;
$\boldsymbol{r}_{k+1} := \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k$;
$\beta_k := -\dfrac{(\boldsymbol{r}_{k+1}, A\boldsymbol{p}_k)}{(\boldsymbol{p}_k, A\boldsymbol{p}_k)}$;
$\boldsymbol{p}_{k+1} := \boldsymbol{r}_{k+1} + \beta_k \boldsymbol{p}_k$;
end

以上の議論の流れからは直接用いなかったが、次の性質も重要である。

補題 3.2.10 (残差ベクトルの直交性) $\{r_k\}$ は直交性「$i \neq j \Longrightarrow (r_i, r_j) = 0$」をもつ。

証明

$j > i$ とすると、r_j は p_ℓ $(\ell = 0, 1, \ldots, j-1)$ と直交することは既に示した ((3.11) を見よ)。一方で、$r_i \in \mathrm{Span}(r_0, r_1, \ldots, r_i) = \mathrm{Span}(p_0, p_1, \ldots, p_i) \subset \mathrm{Span}(p_0, p_1, \ldots, p_{j-1})$ であるから、$(r_j, r_i) = 0$. ■

(旧版) 証明

(この証明をチェックせよ！) 帰納法で証明する。$i \neq j$ かつ $i \leq k$, $j \leq k$ に対して成り立つと仮定する。そのとき、 $i \leq k$ に対して

$$(r_{k+1}, r_i) = 0,$$

が成り立つことを示せばよい。

$i < k$ のとき

$$\begin{aligned}
(r_{k+1}, r_i) &= (r_i, r_{k+1}) \\
&= (r_i, r_k - \alpha_k A p_k) && (r_{k+1} = r_k - \alpha_k A p_k) \\
&= -\alpha_k (r_i, A p_k) && (\text{帰納法の仮定より } (r_i, r_k) = 0) \\
&= -\alpha_k (p_i - \beta_{i-1} p_{i-1}, A p_k) && (r_i = p_i - \beta_{i-1} p_{i-1}) \\
&= 0 && ((p_i, A p_k) = 0,\ (p_{i-1}, A p_k) = 0).
\end{aligned}$$

$i = k$ のときには

$$
\begin{aligned}
(r_{k+1}, r_k) &= (r_k - \alpha_k A p_k, r_k) && (r_{k+1} = r_k - \alpha_k A p_k) \\
&= (r_k, r_k) - \alpha_k (r_k, A p_k) \\
&= (r_k, r_k) - \alpha_k (p_k - \beta_{k-1} p_{k-1}, A p_k) && (r_k = p_k - \beta_{k-1} p_{k-1}) \\
&= (r_k, r_k) - \alpha_k (p_k, A p_k) && ((p_{k-1}, A p_k) = 0) \\
&= (r_k, r_k) - (p_k, r_k) && (\alpha_k = (r_k, p_k)/(p_k, A p_k)) \\
&= (r_k - p_k, r_k) \\
&= -\beta_{k-1} (p_{k-1}, r_k) && (p_k = r_k + \beta_{k-1} p_{k-1}) \\
&= -\beta_{k-1} (p_{k-1}, r_{k-1} - \alpha_{k-1} A p_{k-1}) && (r_k = r_{k-1} - \alpha_{k-1} A p_{k-1}) \\
&= -\beta_{k-1} \left[(p_{k-1}, r_{k-1}) - \alpha_{k-1} (p_{k-1}, A p_{k-1}) \right] && (\alpha_{k-1} = (r_{k-1}, p_{k-1})/(p_{k-1}, A p_{k-1})) \\
&= 0. \quad \blacksquare
\end{aligned}
$$

3.3 CG 法の計算量

前節で示したアルゴリズムの中の 1 ステップ中の乗算の回数を数えよう。

ベクトルのノルムの計算	1 回	N 回の乗算
行列 × ベクトル	1 回	N^2 回の乗算
ベクトルの内積	3 回	$3N$ 回の乗算
ベクトル + スカラー × ベクトル	3 回	$3N$ 回の乗算
合計		$N^2 + 7N$ 回の乗算

反復前にも 1 回の「行列 × ベクトル」があり、反復は最大 N 回であるから、全部で

$$N^2 + N(N^2 + 7N) = N^3 + 8N^2$$

回の乗算で十分であることが分かる。

これは Gauss の消去法や Cholesky 分解法の $N^3/6 + O(N^2)$ とオーダーは同じであるが[1]、6 倍程度の計算量になってしまっている。しかし、実際には CG 法は N 回よりもずっと少ない反復回数で収束するような問題に適用されているので、十分実用的ということになるのである。

3.4 CG 法の収束

ところで、問題によっては N よりもずっと小さな反復回数で、x^* に十分近い近似解が得られることがある。実際、A の条件数[2]を κ とするとき、

$$\phi(\boldsymbol{x}_k) \le 4 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^{2k} \phi(\boldsymbol{x}_0) \quad \text{つまり} \quad \|\boldsymbol{x}_k - x^*\| \le 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^{k} \|\boldsymbol{x}_0 - x^*\| \tag{3.17}$$

が成立するので (Axelsson [1])、κ が十分小さいならば、$\phi(\boldsymbol{x}_k)$ は k があまり大きくないうちに小さくなる (つまり $\boldsymbol{x}_k$ は良い近似解になる)。この種の問題では、実際には k が N よりもずっと小さいところで計算を打ち切ることになる。

[1] Gauss の消去法は、一般の密行列に対しては $N^3/3 + O(N^2)$ の計算量となるが、正値対称行列に対しては約半分の計算量で十分である。

[2] ここではノルムとして普通のユークリッドを採用する。

余談 3.4.1 とあるところで、某専門家と会って話をしていたときのこと。上のような誤差評価式はあるけれど、CG 法は多くの場合に超収束を引き起こすことが大事なんだ、とボソッと漏らした。外野からは CG 法の理論はどんどん整理されてきている (やり過ぎではないか？と感じるくらい) ように見えるが、まだまだ大事なことの究明が残されているのかも知れない。ひょっとすると、非対称問題向けに決定版解法となるものがあって、発見されるのを待っているのかも。ミーハー (野次馬) としては楽しい分野である。 ■

注意 3.4.1 (条件数は最大最小固有値の比) ノルムとしてユークリッド・ノルム

$$\|x\| = \sqrt{\sum_{i=1}^{N} x_k^2}$$

を採用するとき、正値対称行列 A の条件数 κ は、

$$\kappa = \frac{\max \sigma(A)}{\min \sigma(A)}, \quad \sigma(A) = A \text{ の固有値全体}$$

を満たすから、「条件数が小さい」とは、「固有値のばらつきが小さい」ことである、と言い替えられる。

3.5 前処理付き CG 法

与えられた問題のままでは、条件数が大きい場合、そのまま前小節で説明した素朴な CG 法を適用すると、精度の高い近似解を得るには、大きな反復回数が必要になってしまう。この場合、問題を適当に変形することによって、それと等価で、条件数の小さな問題を導き (このことを **前処理** (**preconditioning**) という)、その問題に CG 法を適用して解を求める、というのが**前処理付き共役勾配法** (**preconditioned conjugate gradient method**, 略して PCG 法) の基本的なアイディアである。

前処理には色々な方法があるので、PCG 法も色々な方法の総称である。

与えられた問題に対して、具体的にどのような前処理を採用すべきかは、係数行列 A の性質に強く依存するが、Cholesky 分解 $A = LL^T$ を適当にさぼった不完全 Cholesky 分解を採用した ICCG 法などが、ポピュラーである。

より具体的な説明をしよう。C を N 次正則行列とするとき、(3.1) は

$$C^{-1}A(C^T)^{-1}C^T x = C^{-1}b \tag{3.18}$$

と同値である。そこで、

$$\begin{cases} \tilde{A} \stackrel{\text{def.}}{\equiv} C^{-1}A(C^T)^{-1}, \\ \tilde{b} \stackrel{\text{def.}}{\equiv} C^{-1}b \end{cases}$$

とおけば、

$$\tilde{A}\tilde{x} = \tilde{b}, \tag{3.19}$$

$$C^T x = \tilde{x} \tag{3.20}$$

という問題に帰着できる (まず (3.19) を解いて $\tilde{x}$ を求め、次に (3.20) を解いて x を求める)。ここで C は**前処理行列**と呼ばれるが、これには、次のような性質をもつものが都合がよいことになる。

(i) CC^T が A に近い (言い替えると $\tilde{A}$ が単位行列に近い[3])。

(ii) 与えられたベクトル v に対して、$C^{-1}v$ や $(C^T)^{-1}v$ を計算するのが簡単。

より具体的には、それぞれ次のようにする。

(i) 「A の**不完全 Cholesky 分解**を利用する」 — A が正値実対称行列であることから、

$$A = LL^T \tag{3.21}$$

となるような左下三角行列 L が存在する。(3.21) を A の **Cholesky 分解**という。与えられた行列を Cholesky 分解するためのアルゴリズムは色々知られているが、その計算を「適当に」さぼることにより、

$$A \simeq CC^T \tag{3.22}$$

を満たす左下三角行列 C を得ることが出来る。この (3.22) を A の不完全 Cholesky 分解と呼ぶ。この C を前処理行列とする。

(ii) 「疎な三角行列を選ぶ」 — 常識的なことであるが、$C^{-1}v$ を計算するには方程式 $Cu = v$ を解き、$(C^T)^{-1}v$ を計算するには方程式 $C^Tw = v$ を解くのが賢明である。この計算は C が三角行列であれば非常に簡単に実行できるが[4]、さらに C が疎であれば計算量が大幅に少なくてすむ。

一言でまとめると、

> 与えられた行列 A の Cholesky 分解を適当にさぼって実行することによって、
> $A \simeq CC^T$ を満たす疎な三角行列 C を求め、前処理行列に採用する。

このさぼり方の指針は「あらかじめ零／非零パターンを指定する」というものである。つまり、

$$G_A = \{(i,j) \in \{1,2,\cdots,N\}^2; a_{ij} \neq 0\}$$

として、

$$G_A \subset G$$

なる集合 G を固定して、A を Cholesky 分解するときに、$\hat{L}$ の要素のうち、添字が G に属するものだけを計算し、他のものは 0 とすることで、L, D を計算し、$C = LD^{1/2}$ とする。

不完全 Cholesky 分解の仕方 (G の選び方) には色々あるが、それについては、次節で述べることにして、ここで PCG 法のアルゴリズムをまとめておこう。原理的には 「˜」つきの方程式 (3.19) に CG 法を適用して $\tilde{x}$ を求め、最後に $(C^T)^{-1}$ をかけて $x = x^*$ とするわけだが、最初から「˜」なしの変数を主役にして記述すると、

[3]単位行列の条件数は 1 であり、望み得る最小の値である。単位行列に近ければ、条件数は小さいと期待できる、というわけ。

[4]LU 分解を学んだときに習った。

PCG 法のアルゴリズム (素朴版)

初期ベクトル $\boldsymbol{x}_0$ をとる ; 目標とする相対残差 ε を決める ;
$\boldsymbol{r}_0 := \boldsymbol{b} - A\boldsymbol{x}_0$; $\boldsymbol{p}_0 := (CC^T)^{-1}\boldsymbol{r}_0$;
for $k := 0, 1, \cdots$ **until** $\|\boldsymbol{r}_k\| \leq \varepsilon\|\boldsymbol{b}\|$ **do**
begin
$\alpha_k := \dfrac{(\boldsymbol{r}_k, \boldsymbol{p}_k)}{(\boldsymbol{p}_k, A\boldsymbol{p}_k)}$;
$\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k\boldsymbol{p}_k$;
$\boldsymbol{r}_{k+1} := \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k$;
$\beta_k := -\dfrac{((CC^T)^{-1}\boldsymbol{r}_{k+1}, A\boldsymbol{p}_k)}{(\boldsymbol{p}_k, A\boldsymbol{p}_k)}$;
$\boldsymbol{p}_{k+1} := (CC^T)^{-1}\boldsymbol{r}_{k+1} + \beta_k\boldsymbol{p}_k$;
end

となる。ここで、補助ベクトル等を導入して工夫すると、

PCG 法のアルゴリズム (実用版)

初期ベクトル $\boldsymbol{x}_0$ をとる ; 目標とする相対残差 ε を決める ;
$\boldsymbol{r}_0 := \boldsymbol{b} - A\boldsymbol{x}_0$; $\boldsymbol{p}_0 := (CC^T)^{-1}\boldsymbol{r}_0$; $\rho := (\boldsymbol{r}_0, \boldsymbol{p}_0)$;
for $k := 0, 1, \cdots$ **until** $\|\boldsymbol{r}_k\| \leq \varepsilon\|\boldsymbol{b}\|$ **do**
begin
$\boldsymbol{q} := A\boldsymbol{p}_k$; $\nu := (\boldsymbol{p}_k, \boldsymbol{q})$; $\alpha_k := \rho/\nu$;
$\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k\boldsymbol{p}_k$;
$\boldsymbol{r}_{k+1} := \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k$;
$\boldsymbol{q} := (CC^T)^{-1}\boldsymbol{r}_{k+1}$;
$\mu := (\boldsymbol{q}, \boldsymbol{r}_{k+1})$; $\beta_k := \mu/\rho$; $\rho := \mu$;
$\boldsymbol{p}_{k+1} := \boldsymbol{q} + \beta_k\boldsymbol{p}_k$;
end

3.6 2 次元長方形領域における熱方程式に対する PCG 法

前節で述べたように、一口に PCG 法といっても、不完全 Cholesky 分解の仕方 (前処理行列 C の取り方) が決まっているわけではない。いろいろな方法が提案されているが、現状では決定版は見つかっていない (そんなものはないのかもしれない)。扱う問題ごとに、「この取り方ならばうまく行く」という経験が蓄積されていて、それに従って問題を解いているのが実情である。

ここでは、2 次元の長方形領域における熱方程式の初期値境界値問題を差分法で解く場合に現れる連立 1 次方程式を考える。格子点番号を図のように x 軸方向、y 軸方向の順につけ、x 方向の格子点数を m とする ($m = x$ 軸方向の分割数 $+1$)。このとき係数行列 A の非零要素の添字の集合は

$$G_A = \{(i, j) \in \mathbf{Z}^2; 1 \leq i, j \leq N, j = i, i \pm 1, i \pm m\}$$

となる。A の対角要素を a_i, 副対角要素を b_i, c_i とする。ただし、i は行番号を表す。すなわち

$$\begin{aligned} a_i &= A \text{の第 } (i, i) \text{ 要素}, \\ b_i &= A \text{の第 } (i, i+1) \text{ 要素}, \\ c_i &= A \text{の第 } (i, i+m) \text{ 要素}. \end{aligned}$$

3.6.1 ICCG(1,1) 法

これは $G = G_A$ とするものである。

$$A = U^T DU - R \simeq U^T DU$$

としたときの U の対角要素を $\tilde{a}_i$, 副対角要素を $\tilde{b}_i$, $\tilde{c}_i$ とし、D の対角要素を $\tilde{d}_i$ とすると、

$$\begin{cases} \tilde{d}_i^{-1} &= \tilde{a}_i = a_i - b_{i-1}^2 \tilde{d}_{i-1} - c_{i-m}^2 \tilde{d}_{i-m} \\ \tilde{b}_i &= b_i \\ \tilde{c}_i &= c_i \end{cases}$$

ただし、添字が範囲外のもの (0 以下になったり、$N+1$ 以上になったもの) は 0 とみなす。例えば、

$$\tilde{d}_1^{-1} = \tilde{a}_1 = a_1, \quad \tilde{d}_2^{-1} = \tilde{a}_2 = a_2, -b_1^2 \tilde{d}_1.$$

この前処理を用いた PCG 法が ICCG(1,1) 法である。

3.6.2 MICCG(1,1) 法

これは ICCG(1,1) に補正 (modification) を加えたもので、

$$\begin{cases} \tilde{d}_i^{-1} &= \tilde{a}_i = a_i - b_{i-1}^2 \tilde{d}_{i-1} - c_{i-m}^2 \tilde{d}_{i-m} - \alpha(b_{i-1}c_{i-1}\tilde{d}_{i-1} + b_{i-m}c_{i-m}\tilde{d}_{i-m}) \\ \tilde{b}_i &= b_i \\ \tilde{c}_i &= c_i, \end{cases}$$

ただし α は 1 より小さな定数である。普通 $\alpha = 0.95$ くらいでよい。

3.7 CG 法の参考文献 (日本語で読めるもの)

まず、何と言っても野寺 [9], [24] をあげておく。

名取 [25] は、いわゆる教科書風の本で、この分野ではとても珍しい。プログラムは載っていないが、サービス精神に溢れていて、分かりやすい。

森・杉原・室田 [8] も原理的なところをよく解説した教科書である。

戸川 [15] は、日本におけるテキストとしては先駆的な本である。今でも、説明が参考になる点が多く、重宝する。戸川 [16] は、非常に面白い。前処理付き共役勾配法が現れる以前の解説書・報告書として出色であり、現在でも必読書である (これを読むことで、他の本を読んでいる際に浮かんだ多くの疑問を解消することができた)。

森 [7] には、さまざまな基本的な数値計算アルゴリズムの簡単な解説とプログラムが載っている。PCG については、一般の行列 A に対して $G = G_A$ とするプログラムが載っている。

村田・名取・唐木 [11] では、空間 2 次元、3 次元の偏微分方程式の問題を差分法で解く場合に現れる連立 1 次方程式に対する各種の解法を扱っていて、どの手法がよいか、実験結果も載っている。非対称な問題向けのアルゴリズムの解説もある。実際に偏微分方程式の数値シミュレーションをする場合は、一度は目を通しておくべきである。この本の改訂版が出ると良いのだが…

小国 [10] は、現在邦書のうちで大型疎行列を係数行列とする連立 1 次方程式のもっとも詳しい本 (理論を解説したというよりは、実戦的であることを目指している)。これも何が書いてあるか一度は目を通しておくべき本である。

正値対称でない係数行列に対する非定常反復法については、まとまった書籍としては、藤野・張 [23] や野寺 [24], Dongarra 他 [2] などがある。手短な解説として、張 [14] なども勧めたい。

付 録A　内積空間、Hilbert 空間の復習

A.1　Gram-Schimidt の直交化法

(準備中 — 線形代数の本を見よう)

A.2　射影定理

Hilbert 空間 X の部分集合 K, $x \in X$ に対して、x に最も近い $y \in K$、

$$y \in K \quad \text{s.t.} \quad \|x - y\| = \inf_{h \in K} \|x - h\|$$

を満たす y を見い出す問題を考える。この y を x の K の上への射影と呼ぶ。

定理 A.2.1 (射影定理) X が Hilbert 空間, K を X の空でない凸閉集合, $x \in X$ とするとき、x の K の上への射影 $y \in K$ が一意的に存在する。また y は

$$\mathrm{Re}\,(x - y, v - y) \leq 0 \quad (v \in K)$$

で特徴づけられる。特に $K = V$ (閉部分空間) の場合は、この不等式は

$$(x - y) \perp V \quad \text{i.e.} \quad (x - y, v) = 0 \quad (v \in V)$$

と同値である。すなわち、y は x の V の上への正射影 (x から V に下ろした垂線の足) である。

(証明は、藤田・黒田・伊藤 [21] 等を参照せよ。)

注意 A.2.2 (正射影の定義) y が部分空間 V への、x の正射影であるとは、

$$y \in V, \quad (x - y) \perp V$$

が成り立つことである。 ■

注意 A.2.3 $K = x_0 + V = \{x_0 + v; v \in V\}$ (x_0 は X の元、V は X の閉部分空間) の場合も、x の K への正射影は一意的に存在し、それは

$$y \in V, \quad (x - y) \perp V$$

で特徴づけられる。 ■

$\{\varphi_j\}$ を X の直交系とするとき、任意に固定した k に対して、

$$V_k \overset{\text{def.}}{=} \mathrm{Span}(\varphi_1, \cdots, \varphi_k)$$

とおき、$x \in X$ の V_k への正射影を x_k とすると、(良く知られているように)

$$x_k = \sum_{j=1}^{k} c_j \varphi_j, \quad c_j = \frac{(x, \varphi_j)}{(\varphi_j, \varphi_j)} \quad (j = 1, \cdots, k)$$

である。

特に、各 j について、c_j は φ_j のみで定まっていることに注意しよう。特に

(1) c_j は k $(\geq j)$ によらない (Fourier 係数の最終性)。

(2) $x_{k+1} = x_k + c_{k+1}\varphi_{k+1}$ であるが、 c_{k+1} は最小問題

$$\min_{c \in \mathbf{R}} \|x - (x_k + c\varphi_{k+1})\|$$

の唯一の解として得られる。

付 録B　CG 法のプログラム例

B.1　MATLAB で遊ぶ

```
% cg1.m --- CG 法で A x = b を解くための関数 cg1()
%  2003/5/15 作成, 5/21 修正
%
% 【使い方】
%    x = cg1(A, b, 初期ベクトル, 相対残差)

% 【例】
%  まず問題を用意
%   n=2;a=[2 1;1 3];exact=ones(n,1);b=a*exact
%  初期ベクトルを 0 ベクトル、相対残差を 10^{-12} にして実行
%   cg1(a,b,zeros(n,1),1e-12)
%

function x = cg1(a,b,x0,epsilon)
  x = x0;
  r = b - a * x;
  p = r;
  eps_b = epsilon * norm(b);
  k = 0;
  while norm(r) > eps_b
    ap = a * p;
    pap = ap' * p;
    alpha = p'*r/pap;
    x = x + alpha*p;
    r = r - alpha*ap;
    beta = - (ap'*r)/pap;
    p = r + beta * p;
    k=k+1;  % 最初 k++ としていましたが (Octave なら動く)、MATLAB ではダメです
    % 以下の 3 行は中間結果の表示用（不要ならば削除すればよい）
    k
    x
    r
  end
```

B.2　素朴な CG 法の C プログラム

```
/*
 * cg2.C --- 入出力に iostream を使う
 *    g++ -O -o cg2 cg2.C  -L/usr/local/lib -lmatrix
 */

#include <iostream.h>
#include <cmath>
extern "C" {
#include <matrix.h>
}

matrix zeros(int m, int n)
{
```

```
  int i, j;
  matrix tmp;
  tmp = new_matrix(m, n);
  for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
      tmp[i][j] = 0.0;
  return tmp;
}

void zeros(int m, int n, matrix z)
{
  int i, j;
  for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
      z[i][j] = 0.0;
}

vector zeros(int n)
{
  int i;
  vector tmp;
  tmp = new_vector(n);
  for (i = 0; i < n; i++)
    tmp[i] = 0.0;
  return tmp;
}

void zeros(int n, vector z)
{
  int i;
  for (i = 0; i < n; i++)
    z[i] = 0.0;
}

void print_vector(int n, vector x)
{
  int i;
  for (i = 0; i < n; i++) {
    cout << " " << x[i];
    if (i % 5 == 4)
      cout << endl;
  }
  if (i % 5 != 0)
    cout << endl;
}

void print_matrix(int m, int n, matrix a)
{
  int i, j;
  for (i = 0; i < m; i++) {
    for (j = 0; j < n; j++) {
      cout << a[i][j];
      if (j % 5 == 4)
        cout << endl;
    }
    if (j % 5 != 0)
      cout << endl;
  }
}

double dotproduct(int n, vector x, vector y)
{
  int i;
```

```
  double sum;
  sum = 0;
  for (i = 0; i < n; i++)
    sum += x[i] * y[i];
  return sum;
}

double norm(int n, vector x)
{
  return sqrt(dotproduct(n, x, x));
}

void multiply_mv(int m, int n, vector Ab, matrix A, vector b)
{
  int i;
  for (i = 0; i < m; i++)
      Ab[i] = dotproduct(n, A[i], b);
}

void copy_vector(int n, vector y, vector x)
{
  int i;
  for (i = 0; i < n; i++)
    y[i] = x[i];
}

void cg(int n, matrix A, vector b, vector x, double eps, int *maxiter)
{
  int i, k;
  double eps_b, pAp, alpha, beta;
  vector r, p, Ap;

  r = new_vector(n);
  p = new_vector(n);
  Ap = new_vector(n);
  /* eps_b := ε ||b|| */
  eps_b = eps * norm(n, b);
  /* r := A x */
  multiply_mv(n, n, r, A, x);
  /* r := b - A x */
  for (i = 0; i < n; i++)
    r[i] = b[i] - r[i];
  /* p := r */
  copy_vector(n, p, r);
  /* 反復 */
  for (k = 0; k < n; k++) {
    /* Ap := A * p */
    multiply_mv(n, n, Ap, A, p);
    /* pAp := (p, Ap) */
    pAp = dotproduct(n, p, Ap);
    /* α = (r,p)/(p,Ap) */
    alpha = dotproduct(n, r, p) / pAp;
    /* x, r の更新 */
    for (i = 0; i < n; i++) {
      /* x = x + α*p */
      x[i] += alpha * p[i];
      /* r = r - α*A*p */
      r[i] -= alpha * Ap[i];
    }
    /* ||r||<ε||b|| ならば反復を終了 */
    if (norm(n, r) < eps_b)
      break;
    /* β := - (r, Ap) / (p, Ap) */
```

```
    beta = - dotproduct(n, r, Ap) / pAp;
    /* p := r + β*p */
    for (i = 0; i < n; i++)
      p[i] = r[i] + beta * p[i];
  }
  if (maxiter != NULL)
    *maxiter = k;

  free_vector(r);
  free_vector(p);
  free_vector(Ap);
}

int main()
{
  int n, N, i, niter;
  matrix a;
  vector x, b;
  double c1, c2;
  /* 問題のサイズを決める */
  cout << "c1, c2, N=";
  cin >> c1 >> c2 >> N;
  n = N / 2;
  N = 2 * n;
  /* 行列 A, ベクトル x, b を記憶する変数の準備 */
  a = zeros(N, N);
  x = new_vector(N);
  b = new_vector(N);
  /* 係数行列を決める */
  for (i = 0; i < N; i++) {
    a[i][i] = 4.0;
    if (i != 0)
      a[i][i-1] = -1.0;
    if (i != N-1)
      a[i][i+1] = -1.0;
  }
  a[n-1][n] = 0.0;
  a[n][n-1] = 0.0;
  /* */
  for (i = 0; i < n; i++) {
    if (i != 0)
      a[i][i-1] *= c1;
    a[i][i] *= c1;
    a[i][i+1] *= c1;
  }
  for (i = n; i < N; i++) {
    a[i][i-1] *= c2;
    a[i][i] *= c2;
    if (i != N-1)
      a[i][i+1] *= c2;
  }
  /* 係数行列を表示する */
  if (N < 10) {
    cout << "a=" << endl;
    print_matrix(N, N, a);
  }
  /* 解 x を決める */
  for (i = 0; i < N; i++)
    x[i] = i;
  /* 解に対応する右辺 b を計算する */
  multiply_mv(N, N, b, a, x);
  /* b を表示する */
  if (N < 10) {
```

```
    cout << "b=" << endl;
    print_vector(N, b);
  }
  /* 初期ベクトル x を零ベクトルにする */
  zeros(N, x);
  /* CG 法で解く */
  cg(N, a, b, x, 1e-12, &niter);
  /* 解を表示する */
  cout << "x=" << endl;
  print_vector(N, x);
  cout << "反復回数=" << niter << endl;

  return 0;
}
```

B.3 高橋版

細かい工夫だが、次のアルゴリズムは演算回数が少ない。

CG 法のアルゴリズム (高橋秀俊版)

初期ベクトル $\boldsymbol{x}_0$ をとる ; 目標とする相対残差 ε を決める ;
$\boldsymbol{r}_0 := \boldsymbol{b} - A\boldsymbol{x}_0$; $\beta_0 := 1/(\boldsymbol{r}_0, \boldsymbol{r}_0)$; $\boldsymbol{p}_0 := \beta_0 \boldsymbol{r}_0$;
for $k := 0, 1, \cdots$ **until** $\|\boldsymbol{r}_k\| \leq \varepsilon \|\boldsymbol{b}\|$ **do**
begin

$$\alpha_k := \frac{1}{(\boldsymbol{p}_k, A\boldsymbol{p}_k)};$$
$$\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k;$$
$$\boldsymbol{r}_{k+1} := \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k;$$
$$\beta_k := \frac{1}{(\boldsymbol{r}_{k+1}, \boldsymbol{r}_{k+1})};$$
$$\boldsymbol{p}_{k+1} := \boldsymbol{p}_k + \beta_k \boldsymbol{r}_{k+1};$$

end

このアルゴリズムの中の 1 ステップ中の乗算の回数を数えよう。

ベクトルのノルムの計算	1 回	N 回の乗算
行列 × ベクトル	1 回	N^2 回の乗算
ベクトルの内積	2 回	$2N$ 回の乗算
ベクトル + スカラー × ベクトル	3 回	$3N$ 回の乗算
合計		$N^2 + 6N$ 回の乗算

本文で示したアルゴリズムでは、反復 1 回あたりの計算量は $N^2 + 7N$ 乗算であったから、少し効率化されていることが分かる。

反復前にも 1 回の「行列 × ベクトル」,「内積」,「スカラー倍」があり、反復は最大 N 回であるから、全部で

$$(N^2 + N + N) + N(N^2 + 6N) = N^3 + 7N^2 + 2N$$

回の乗算で十分であることが分かる。

```
/*
 * takahashi.C
 */

void cg(int n, matrix A, vector b, vector x, double eps, int *maxiter)
```

```
{
  int i, k;
  double eps_b, pAp, alpha, beta;
  vector r, p, Ap;

  r = new_vector(n);
  p = new_vector(n);
  Ap = new_vector(n);
  /* eps_b := ε ||b|| */
  eps_b = eps * norm(n, b);
  /* r := A x */
  multiply_mv(n, n, r, A, x);
  /* r := b - A x */
  for (i = 0; i < n; i++)
    r[i] = b[i] - r[i];
  /* β := 1 / (r,r); p := β r */
  beta = 1.0 / dotprod(n, r, r);
  multiply_sv(n, p, beta, r);
  /* 反復 */
  for (k = 0; k < n; k++) {
    /* Ap := A * p */
    multiply_mv(n, n, Ap, A, p);
    /* pAp := (p, Ap) */
    pAp = dotproduct(n, p, Ap);
    /* α = 1/(p,Ap) */
    alpha = 1.0 / pAp;
    /* x, r の更新 */
    for (i = 0; i < n; i++) {
      /* x = x + α*p */
      x[i] += alpha * p[i];
      /* r = r - α*A*p */
      r[i] -= alpha * Ap[i];
    }
    /* ||r||<ε||b|| ならば反復を終了 */
    if (norm(n, r) < eps_b)
      break;
    /* β := 1 / (r, r) */
    beta = 1.0 / dotprod(n, r, r);
    /* p := r + β*p */
    for (i = 0; i < n; i++)
      p[i] += beta * r[i];
  }
  if (maxiter != NULL)
    *maxiter = k;

  free_vector(r);
  free_vector(p);
  free_vector(Ap);
}
```

付 録C　Lanczos の原理

これが CG 法の肝だと思われるのだが…

C.1　高橋

高橋 [20]

C.2　杉原・室田 [13]

次に掲げるのは、杉原・室田 [13] の第 7 章「最小 2 乗近似と直交多項式」の補題 7.3 である。

命題 C.2.1 (Lanczos の原理) E を内積 $(,)$ の定義された内積空間とし、$A\colon E \to E$ を対称線形作用素とする。$u_0 \in E$ を取り、

$$u_n = A^n u_0 \quad (n \in \mathbf{N})$$

で u_n を定めるとき、$u_0,\ u_1,\ \cdots,\ u_N$ が線型独立とする。このとき Gram-Schmidt の直交化

$$\begin{aligned} v_0 &= u_0, \\ v_n &= u_n - \sum_{j=0}^{n-1} \frac{(u_n, v_j)}{(v_j, v_j)} v_j \quad (n = 1, 2, \cdots, N) \end{aligned}$$

を適用して得られる直交系 $\{v_n; n = 0, 1, \cdots, N\}$ は、次の 3 項漸化式を満たす。

$$v_{n+1} = (A - \alpha_n)v_n - \beta_n v_{n-1} \quad (n = 0, 1, \cdots, N-1). \tag{C.1}$$

ただし

$$\begin{aligned} \beta_0 v_{-1} &= 0, \\ \alpha_n &= \frac{(Av_n, v_n)}{(v_n, v_n)} \quad (n = 0, 1, \cdots, N-1), \\ \beta_n &= \frac{(v_n, v_n)}{(v_{n-1}, v_{n-1})} \quad (n = 1, 2, \cdots, N-1). \end{aligned}$$

補題 C.2.2 (Gram-Schmidt の直交化) 内積空間 E の元 $u_0, u_1, \cdots, u_N$ が 1 次独立であるとするとき、

$$\text{(C.2)} \qquad \begin{cases} v_0 := u_0, \\ v_n := u_n - \displaystyle\sum_{i=0}^{n-1} \frac{(u_n, v_i)}{(v_i, v_i)} v_i \quad (1 \leq n \leq N) \end{cases}$$

で $v_0, v_1, \cdots, v_N$ を定めると、任意の n について

$$\text{(C.3)} \qquad \mathrm{Span}(u_0, u_1, \cdots, u_n) = \mathrm{Span}(v_0, v_1, \cdots, v_n)$$

であり、

$$(v_n, v_m) = 0 \quad (0 \leq m < n \leq N).$$

証明

まず (C.3) を n についての帰納法で示す。$n = 0$ のときは、$v_0 = u_0$ であるから明らかに成立する。n まで成り立つとすると、帰納法の仮定から $v_0, \cdots, v_n$ は 1 次独立になるので、特に $v_i \neq 0$ $(0 \leq i \leq n)$ で、

$$\text{(C.4)} \qquad v_{n+1} = u_{n+1} - \sum_{i=0}^{n} \frac{(u_n, v_i)}{(v_i, v_i)} v_i$$

が意味を持つ (分母は 0 にならない)。帰納法の仮定から

$$\sum_{i=0}^{n} \frac{(u_n, v_i)}{(v_i, v_i)} v_i \in \mathrm{Span}(v_0, \cdots, v_n) = \mathrm{Span}(u_0, \cdots, u_n)$$

であるから、

$$v_{n+1} \in \mathrm{Span}(u_0, \cdots, u_n, u_{n+1}).$$

ゆえに

$$\mathrm{Span}(v_0, \cdots, v_n, v_{n+1}) \subset \mathrm{Span}(u_0, \cdots, u_n, u_{n+1}).$$

一方、(C.4) を移項して

$$u_{n+1} = v_{n+1} + \sum_{i=0}^{n} \frac{(u_n, v_i)}{(v_i, v_i)} v_i \in \mathrm{Span}(v_0, \cdots, v_{n+1}).$$

ゆえに

$$\mathrm{Span}(u_0, \cdots, u_n, u_{n+1}) \subset \mathrm{Span}(v_0, \cdots, v_n, v_{n+1}).$$

以上から

$$\mathrm{Span}(u_0, \cdots, u_n, u_{n+1}) = \mathrm{Span}(v_0, \cdots, v_n, v_{n+1}).$$

さて、

$$(v_n, v_m) = \left(u_n - \sum_{i=0}^{n-1} \frac{(u_n, v_i)}{(v_i, v_i)} v_i, v_m \right) = (u_n, v_m) - \sum_{i=0}^{n-1} \frac{(u_n, v_i)}{(v_i, v_i)} (v_i, v_m) = (u_n, v_m) - (u_n, v_m) = 0. \blacksquare$$

補題 C.2.3 $u_0 \in E$ に対して、$u_n := A^n u_0$ $(n = 0, 1, \cdots, N)$ で u_n を定めたとき、u_0, u_1, $\cdots$, u_N が 1 次独立であったとするとき、Gram-Schmidt の直交化

$$\text{(C.5)} \qquad \begin{cases} v_0 := u_0, \\ v_n := u_n - \displaystyle\sum_{i=0}^{n-1} \frac{(u_n, v_i)}{(v_i, v_i)} v_i \quad (1 \leq n \leq N) \end{cases}$$

で v_n を定めると、次が成り立つ。

(1) $0 \leq n \leq N$ なる任意の n に対して、最高次の係数が 1 の n 次多項式 $p_n(x)$ が存在して、$v_n = p_n(A)u_0$.

(2) $p_0(x), \cdots, p_N(x)$ は 1 次独立で、さらに任意の n $(0 \leq n \leq N)$ に対して、

$$\mathrm{Span}(p_0(x), p_1(x), \cdots, p_n(x)) = \mathrm{Span}(1, x, \cdots, x^n).$$

(3) $0 \leq n \leq N-1$ なる任意の n に対して、$n+1$ 個の定数 $c_0, \cdots, c_n$ が存在して

$$p_{n+1}(x) = x p_n(x) + \sum_{i=0}^{n} c_i p_i(x)$$

が成り立つ。

証明

(1) 帰納法による。$n = 0$ に対しては、$p_0(x) = 1$ とすると、$v_0 = u_0 = p_0(A)u_0$. $n-1$ まで成り立つとすると、$p_n(x)$ を

$$p_n(x) = x^n - \sum_{i=0}^{n-1} \frac{(u_n, v_i)}{(v_i, v_i)} p_i(x)$$

で定めればよい。実際、帰納法の仮定から右辺の $\sum$ の部分は $n-1$ 次以下の多項式であり、$p_n(x)$ は最高次の係数が 1 の n 次多項式になる。また

$$p_n(A)u_0 = A^n u_0 - \sum_{i=0}^{n-1} \frac{(u_n, v_i)}{(v_i, v_i)} p_i(A)u_0 = u_n - \sum_{i=0}^{n-1} \frac{(u_n, v_i)}{(v_i, v_i)} v_i = v_n. \blacksquare$$

(2) まず 1 次独立性を示そう。

$$c_0 p_0(x) + c_1 p_1(x) + \cdots + c_N p_N(x) = 0$$

とすると、

$$c_0 v_0 + c_1 v_1 + \cdots + c_N v_N = 0$$

が導かれ、v_0, v_1, $\cdots$, v_N の 1 次独立性から

$$c_0 = c_1 = \cdots = c_N = 0$$

が得られる。1 次独立性が証明できれば、明らかな包含関係

$$\mathrm{Span}(p_0(x), p_1(x), \cdots, p_n(x)) \subset \mathrm{Span}(1, x, \cdots, x^n)$$

の両辺の次元が等しいことから、実は等式になることが分かる。

(3) $p_{n+1}(x) - xp_n(x)$ は n 次以下の多項式になるので、(2) より $p_0(x), p_1(x), \cdots, p_n(x)$ の線型結合で表わせる。 ■

命題 C.2.1 の証明

上の補題の (3) より

$$v_{n+1} = p_{n+1}(A)u_0 = Ap_n(A)u_0 + \sum_{i=0}^{n} c_i p_i(A)u_0 = Av_n + \sum_{i=0}^{n} c_i v_i. \tag{C.6}$$

すると $0 \leq j \leq n-2$ なる j に対して

$$Av_j \in A\,\mathrm{Span}(v_0, \cdots, v_{n-2}) = A\,\mathrm{Span}(u_0, \cdots, u_{n-2}) \subset \mathrm{Span}(u_0, \cdots, u_{n-1}) = \mathrm{Span}(v_0, \cdots, v_{n-1})$$

であるから、$Av_j \perp v_n$ である。ゆえに A の対称性から

$$(Av_n, v_j) = (v_n, Av_j) = 0.$$

(C.6) と v_j との内積を取ると

$$0 = (v_{n+1}, v_j) = (Av_n, v_j) + \sum_{i=0}^{n-1} c_i (v_i, v_j) = 0 + c_j (v_j, v_j).$$

ゆえに $c_j = 0$. こうして (C.6) の項のほとんどは 0 で、残ったところは

$$v_{n+1} = Av_n + c_n v_n + c_{n-1} v_{n-1}.$$

v_n との内積を取ると

$$0 = (Av_n, v_n) + c_n (v_n, v_n) \quad \therefore \quad c_n = -\frac{(Av_n, v_n)}{(v_n, v_n)}.$$

v_{n-1} との内積を取ると

$$0 = (Av_n, v_{n-1}) + c_{n-1} (v_{n-1}, v_{n-1}) \quad \therefore \quad c_{n-1} = -\frac{(Av_n, v_{n-1})}{(v_{n-1}, v_{n-1})}.$$

これから

$$v_{n+1} = \left(A - \frac{(Av_n, v_n)}{(v_n, v_n)} \right) v_n - \frac{(Av_n, v_{n-1})}{(v_{n-1}, v_{n-1})} v_{n-1}. \blacksquare$$

C.3 森・杉原・室田

森・杉原・室田 [8] の第 3 章「連立 1 次方程式の解法 III (共役勾配法)」の定理 3.8 である。

命題 C.3.1 A を n 次対称行列, $U_k(\lambda)$ をちょうど k 次の多項式 $(k = 0, 1, \cdots, \overline{n}, \overline{n} \leq n)$, u_0 を n 次元ベクトルとする。$u_k = U_k(A)u_0$ が直交条件

$$(u_i, u_j) = 0 \quad (0 \leq i < j \leq \overline{n})$$

を満たし、かつ $u_k \neq 0$ $(k = 0, 1, 2, \cdots, \overline{n} - 1)$ ならば、ある実数 $\xi_k \neq 0$, η_k, ζ_k $(k - 0, 1, \cdots, \overline{n} - 1)$ が存在して、3 項漸化式

$$U_{k+1}(\lambda) = \xi_k \lambda U_k(\lambda) + \eta_k U_k(\lambda) + \zeta_k U_{k-1}(\lambda) \quad (k = 0, 1, \cdots, \overline{n} - 1) \tag{C.7}$$

が成り立つ。ただし $U_{-1}(\lambda) = 0$ とする。

証明

まず $0 \le k \le \overline{n}$ なる任意の k に対して

$$\mathrm{Span}(U_0(\lambda), U_1(\lambda), \cdots, U_k(\lambda)) = \mathrm{Span}(1, \lambda, \cdots, \lambda^k)$$

である。実際、左辺 $\subset$ 右辺は明らかで、$U_0(\lambda), \cdots, U_k(\lambda)$ の 1 次独立性から (それは次数から明らか) 等号が成立することが分かる。ゆえに

$$\mathrm{Span}(u_0, u_1, \cdots, u_k) = \mathrm{Span}(u_0, Au_0, \cdots, A^k u_0)$$

である。よって、$0 \le i \le \overline{n} - 1$ なる i に対して、

$$\begin{aligned} Au_i &\in A\,\mathrm{Span}(u_0, u_1, \cdots, u_i) = A\,\mathrm{Span}(u_0, Au_0, \cdots, A^i u_0) \\ &= \mathrm{Span}(Au_0, A^2 u_0, \cdots, A^{i+1} u_0) \subset \mathrm{Span}(u_0, Au_0, \cdots, A^{i+1} u_0) \\ &= \mathrm{Span}(u_0, u_1, \cdots, u_{i+1}). \end{aligned}$$

ゆえに

$$Au_i \in \mathrm{Span}(u_0, u_1, \cdots, u_{i+1}) \quad (0 \le i \le \overline{n} - 1).$$

さて、

$$\xi_k := \frac{U_{k+1}(\lambda) \text{ の最高次の係数}}{U_k(\lambda) \text{ の最高次の係数}}$$

とおくと、$U_{k+1}(\lambda) - \xi_k \lambda U_k(\lambda)$ は k 次以下の多項式であるから、

$$\exists c_0, c_1, \cdots, c_k \quad \text{s.t.} \quad U_{k+1}(\lambda) - \xi_k \lambda U_k(\lambda) = \sum_{i=0}^{k} c_i U_i(\lambda).$$

これから

$$u_{k+1} - \xi_k A u_k = \sum_{i=0}^{k} c_i u_i. \tag{C.8}$$

$i \le k - 2$ なる i に対して、A の対称性から

$$(\xi_k A u_k, u_i) = \xi_k (u_k, A u_i).$$

ここで $Au_i \in \mathrm{Span}(u_0, u_1, \cdots, u_{i+1}) \subset \mathrm{Span}(u_0, u_1, \cdots, u_{k-1})$ であるので、Au_i は u_k とは直交する。ゆえに

$$(\xi_k A u_k, u_i) = 0.$$

(C.8) と u_i との内積を取ると

$$0 - 0 = \left(\sum_{j=0}^{m} c_j u_j, u_i \right) = c_i (u_i, u_i)$$

であるから、$c_i = 0$. こうして (C.8) の右辺で残るのは、$i = k - 1, k$ の項だけである:

$$u_{k+1} - \xi_k A u_k = c_{k-1} u_{k-1} + c_k u_k.$$

この式と u_k, u_{k-1} との内積を取ると、

$$-\xi_k (A u_k, u_k) = c_k (u_k, u_k), \quad -\xi_k (A u_k, u_{k-1}) = c_{k-1} (u_{k-1}, u_{k-1})$$

であるから

$$c_k = -\xi_k \frac{(A u_k, u_k)}{(u_k, u_k)}, \quad c_{k-1} = -\xi_k \frac{(A u_k, u_{k-1})}{(u_{k-1}, u_{k-1})}. \blacksquare$$

関連図書

[1] O. Axelsson. *Solution of linear systems of equations*. Lecture Notes in Mathematics, No.572. Springer Verlag, 1977.

[2] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H.A. van der Vorst. *Templates for the solution of linear systems: Building blocks for iterative methods*. SIAM, 1994. 長谷川 里美、長谷川 秀彦、藤野 清次 訳、反復法 Templates, 朝倉書店 (1996).

[3] G. H. Golub and D. P. O'Leary. Some history of the conjugate gradient and lanczos algorithms: 1948–1976. *SIAM Review*, Vol. Vol. 31, pp. 50–102, 1989.

[4] A Hadjidimos. Successive overrelaxation (sor) and related methods. *Journal of Computational and Applied Mathematics*, Vol. 123, No. 1, pp. 177–199, 2000.

[5] 桂田祐史. 連立 1 次方程式 iii. http://nalab.mind.meiji.ac.jp/~mk/labo/text/linear-eq-3.pdf, 2002〜.

[6] 森正武(まさたけ). 数値解析. 共立出版, 1973. 第 2 版が 2002 年に出版された。

[7] 森正武. FORTRAN 77 数値計算プログラミング. 岩波書店, 1986, 1987.

[8] 森正武, 杉原正顯(まさあき), 室田一雄(むろたかずお). 線形計算. 岩波講座 応用数学. 岩波書店, 1994.

[9] 野寺隆志(のでらたかし). 大型行列に対する PCG 法. Seminar on Mathematical Science (Keio University), No.7. マテマティカ, 1983.

[10] 村田健郎, 三好俊郎, J. J. ドンガラ, 長谷川秀彦. 行列計算ソフトウェア. 丸善, 1991.

[11] 村田健郎(けんろう), 名取 亮(まこと), 唐木幸比古(からきゆきひこ). 大型数値シミュレーション. 岩波書店, 1990.

[12] 仁木 滉(にきひろし), 河野敏行(こうのとしゆき). 楽しい反復法. 共立出版, 1998.

[13] 杉原正顯(まさあき), 室田一雄(むろた). 数値計算法の数理. 岩波書店, 1994.

[14] 張 紹良(チャンシャオリャン). 大規模連立 1 次方程式の解法 — クリロフ部分空間法. 応用数理, Vol. Vol.8, No.4, , 1998.

[15] 戸川隼人(はやと). マトリクスの数値計算. オーム社, 1971.

[16] 戸川隼人. 共役勾配法. 教育出版, 1977.

[17] R. S. Varga. *Matrix Iterative Analysis*. Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1962. second ed., Springer (2000). (初版邦訳) 渋谷 政昭他訳, 計算機による大型行列の反復解法, サイエンス社 (1972).

[18] 矢川元基, 青山裕司. 有限要素固有値解析. 森北出版, 2001.

[19] Shiming Yang and Matthias K. Gobbert. The optimal relaxation parameter for the sor method applied to the poisson equation in any space dimensions. *Applied Mathematics Letters*, Vol. 22, No. 3, pp. 325–331, 2009.

[20] 高橋秀俊. Lanczos の原理と数値解析. 数理科学, Vol. No.157, pp. 25–31, 1976.

[21] 藤田宏, 黒田成俊, 伊藤清三. 関数解析. 岩波書店, 1991. 岩波講座 基礎数学 (1978 年) の書籍化.

[22] 杉浦光夫, 横沼健雄. Jordan 標準形・テンソル代数. 岩波書店, 1990. これは、杉浦 光夫, Jordan 標準形と単因子論 I, II, 岩波講座 基礎数学 (1976,1977) が元になっている。

[23] 藤野清次, 張紹良. 反復法の数理. 朝倉書店, 1996.

[24] 野寺隆志. 大規模数値計算の最先端技術　インターネット時代の数学ツール 2　「新 数値計算」. 共立出版, 1999.

[25] 名取亮. 線形計算. 朝倉書店, 1993.