数理解析研究所講究録
1147巻 2000年 127-131

対称行列の固有値に対する簡便な精度保証法とその実装

電気通信大学・情報工学科　山本野人(Nobito Yamamoto)

概要

対称行列の固有値を、その大きさの順位まで込めて精度保証付きで算定する方法の提案する。これはLDL分解を用いた2分法とその誤差解析に基づくもので、特に帯行列に対しては記憶容量が節約できる簡明な方法である。

はじめに

精度保証付き計算を普及させるには、専門家による技法の構築もさることながら、専門外の人々がたやすく利用できる条件を調えることも必要である。ここで提案する方法は、

- 背景の理論が初等的
- 計算機への実装が容易
- 苛酷な条件のもとでなければ、十分実用になる

という特徴をもつので、**特に精度保証の専門家以外の方**が固有値の限界を厳密に知る必要がある場合には利用価値があると思われる。さらに、

- 固有値の大きさによる順位を確定できる
- 固有値が多重度をもっている場合や、いくつかの固有値がClusterをなしている場合でも使える
- 一般固有値問題に対しても応用できる
- 行列が区間値をもつ場合に対しても拡張がたやすい
- 帯行列に対しては記憶容量が節約できる

という利点をもつが、精度の点でほかの方法と比べてかなり落ちる場合があるのが欠点である。

方法の原理

$A: n\times n$ の実対称行列, $\tilde{\rho}: A$ の固有値 ρ の近似値, $I: n$ 次の単位行列とするとき、 $M=A-\tilde{\rho}I$ の負の固有値の個数がわかれば、$\tilde{\rho}$ を上限とする固有値および下限とする固有値の順位がわかる。したがって、2分法的な反復を行なえば、固有値の存在する範囲を特定することができる。M の負の固有値の個数については、シルベスターの慣性律から直接導かれる次の定理を用いて数えることとする。

定理 1.
任意の実対称行列 M が対角行列 D と下三角行列 L によって

$$M = LDL^T$$

と表されるとしよう。このとき、M, D の固有値のうち負のものの個数は一致する。

さて、このことを精度保証付きで行なうためには LDL^T 分解の際の丸め誤差の事後評価が必要となる。そのために、

$$\|M-LDL^T\|_2 \leq \max_{1\leq i\leq n}\sum_{1\leq j\leq n}|M_{ij}-(LDL^T)_{ij}|$$

として次の定理を用いる：

Weylの単調性の定理
n 次実対称行列 A, B, C が $A=B+C$ をみたすとする。それぞれの固有値を小さい順に並べたものを $\lambda_i(A), \lambda_i(B), \lambda_i(C), (i=1,\cdots,n)$ とすると、任意の i について

$$|\lambda_i(A)-\lambda_i(B)| \leq \|C\|_2 \tag{1}$$

が成り立つ。

以上のことを踏まえて、この方法の根拠となる定理を述べよう。

主定理

行列 A のある固有値の近似値 $\tilde{\rho}$ と正の定数 δ_1, δ_2 によって

$$\begin{aligned} Y_1 &:= A-(\tilde{\rho}-\delta_1)I \\ Y_2 &:= A-(\tilde{\rho}+\delta_2)I \end{aligned}$$

と定める。I は単位行列である。この Y_1 と Y_2 に対して、下三角行列 L_1, L_2 および対角行列 D_1, D_2 を選んで、

$$\begin{aligned} \varepsilon_1 &:= \|Y_1 - L_1 D_1 L_1^T\|_2 \\ \varepsilon_2 &:= \|Y_2 - L_2 D_2 L_2^T\|_2 \end{aligned}$$

を算定する。(実際には Y_1 および Y_2 のそれぞれの近似的な LDL^T 分解を用いて、その差の ∞-norm によって評価される)。

さて、D_1 が $k-1$ 個、D_2 が $k+r$個の負の要素を持つとしよう ($k>0, r\geq 0$)。このとき A の第 k 固有値 ρ_k から第 $k+r$固有値 ρ_{k+r} までが、区間

$$[\tilde{\rho}-\delta_1-\varepsilon_1 \quad , \quad \tilde{\rho}+\delta_2+\varepsilon_2]$$

の中に存在する。

この定理を用いた実際の計算手順は次のようになる。行列 A の近似固有値 $\tilde{\rho}$ が得られたとき、これが何番目の固有値をどの程度近似するかが知りたいとすると、

1. 正の定数 δ_1, δ_2 を定める。
2. Y_1, Y_2 を算定する (浮動小数点演算)。
3. $Y_1 = L_1 D_1 L_1^T, Y_2 = L_2 D_2 L_2^T$ と分解 (浮動小数点演算)。
4. 丸めの方向を制御しながら $\varepsilon_1, \varepsilon_2$ を次の形で精度保証付きで計算する。

$$\begin{aligned} \varepsilon_1 &:= \|A-(\tilde{\rho}-\delta_1)I - L_1 D_1 L_1^T\|_2 \\ \varepsilon_2 &:= \|A-(\tilde{\rho}+\delta_2)I - L_2 D_2 L_2^T\|_2 \end{aligned}$$

5. D_1, D_2 の負の要素の個数を調べ、定理に従って $\tilde{\rho}$ の近傍にある固有値の番号を定める。

注意

- δ_1, δ_2 の決め方は、例えば、隣の近似固有値までの距離の 1/3 に取る。
- 得られた $\varepsilon_1, \varepsilon_2$ の値が大き過ぎる場合や、計算不能に陥る場合には、δ_1, δ_2 の値を取り直してやってみる。
- D_1, D_2 は 1 または -1 のみから成る対角行列としたほうが丸め誤差に関して得策である。
- 同じように見えるが、$L\,D\,L^T$ 分解のかわりに LU分解を使ってはならない (再構成した行列を対称とみなす根拠がなくなるから)。
- 帯行列に対しては下三角行列 L を全部記憶しておかなくてよい。バンド幅が M とすると、$(M+1)\times(M+1)$ の記憶領域で対応できる。これは Hauseholder 法 (帯行列であっても $n\times n$ の記憶領域が必要) +二分法を精度保証付きでやる方法より有利な点である。

応用

- 行列が区間値の場合
  区間の中の代表値 (ふつうは近似的中心) に対して 1 から 3 までを行ない、$\varepsilon_1, \varepsilon_2$ の計算で区間値を用いればよい。

- 一般固有値問題について

$$Ax = \rho Bx, \qquad B \text{ は正値対称}$$

に対してはあらかじめ B の最小固有値を下から評価しておく。これを λ_B とする。1 から 3 までを I のかわりに B を用いて行ない、$\varepsilon_1, \varepsilon_2$ のかわりにこれらを λ_B で割った値を用いればよい。

数値例

1. まず、この方法の robustness を見るために、テスト行列として単純固有値のクラスターをもつものに対し適用してみる。行列のデータは次のとおり：

行列の大きさ	$n = 720$
バンド幅	$2m+1 = 39$
クラスターの数	143
1 つのクラスターにつき	4 個の固有値
固有値間の距離	約 10^{-10}
最大固有値	4.28136987787859 (近似値)
最小固有値	$1.58194270717581 \times 10^{-11}$ (近似値)

行列 A_1 のデータ

行列の大きさ	$n = 720$
バンド幅	$2m+1 = 39$
クラスターの数	71
1 つのクラスターにつき	9 個の固有値
固有値間の距離	約 10^{-10}
最大固有値	3.91447949480503 (近似値)
最小固有値	$7.15482294457147 \times 10^{-12}$ (近似値)

行列 A_2 のデータ

計算は以下のように行なった。

- すべての近似固有値は Householder and bisection method によって計算した。
- 定数 δ_1 および δ_2 は近似固有値間の距離の 1/3 に取った。
- この方法によって真の固有値を含む区間が得られ、かつ ε_i $(i = 1, 2)$ がともに小さくて隣合う区間同士が接することがないかどうかを調べた。これが実現していれば得られた区間によって固有値が単離できていることになる。

結果は、すべての固有値について、その真の値を含む区間が得られた。それぞれのテスト行列に関する固有値単離の成功率を次の表に示す。

行列 A_1	86.4902 %
行列 A_2	90.3899 %

この例で扱った行列はかなり条件が厳しいものなので、この結果からは、我々の方法は現実の問題に対する精度保証用のツールとして十分実用になると考えられるであろう。

2. この方法の利点のひとつは、帯行列を扱う場合は使用するメモリーが節約できることにある。大きさが $n \times n$ でバンド幅が $2m+1$ の行列に対して、$n \times (m+1)$ の大きさの配列を行列記憶用に、また $(m+1) \times (m+1)$ の大きさの配列を作業用に必要とするのみである。

以下に比較的大きなサイズの帯行列を生じる問題を考える。

Find u such that

$$\begin{cases} -\triangle u = g & \text{in } \Omega, \\ \dfrac{\partial u}{\partial n} = 0 & \text{on } \partial\Omega. \end{cases}$$

ここに Ω は $(0,0), (1,0)$ および $(0,1)$ に頂点をもつ直角三角形とする。この問題を有限要素法で解くことにしよう。

S_h : 一様三角形分割されたメッシュの上での1次要素からなる有限要素空間

N : Ω の一辺の分割数

とし、ここでは $N = 140$ に取る。S_h の次元は $n = 10011$ になる。
行列 A はその要素を

$$A_{ij} = (\nabla\phi_i, \nabla\phi_j), \qquad i,j = 1,\cdots,n,$$

と定めることで定義される。ここに ϕ_i は S_h の基底, $(\cdot,\cdot)$ は L^2 内積である。
目的を **小さい方から数えて2番目の固有値** を真に含む区間を得ることとする。

- A の最小固有値は 0 である。
- 近似固有値は LDL^T 分解を2分法的に繰り返して求めた。これは村田の方法として知られている。

行列のデータは次のとおり。

行列の大きさ	$n = 10011$
バンド幅	$2m+1 = 283$ $(m = N+1)$
近似固有値 $\tilde{\rho}$	$4.8592964434113 \times 10^{-4}$
δ_1	1.0×10^{-16}
δ_2	1.0×10^{-16}

結果は次のようになった。

D_1 の負の要素数	1
D_2 の負の要素数	2
ε_1	$3.819858925699894 \times 10^{-13}$
ε_2	$3.879345846297622 \times 10^{-13}$
第2固有値の下限	$4.859296439590441 \times 10^{-4}$
第2固有値の上限	$4.859296447291646 \times 10^{-4}$
CPU Time	2357.9 sec

すなわち、この方法は比較的サイズの大きな帯行列に対しても有効である。

3. さて、次に一般固有値問題を扱うことにしよう。

$$Ax = \rho Bx,$$

ここに A は上述の例と同じ行列であり、行列 B は以下のように定められる。

$$B_{ij} = (\phi_i, \phi_j), \qquad i,j = 1,\cdots,n.$$

B は A とおなじバンド幅の帯行列となる。
ここでは、やはり第2固有値を求めることにする。最小固有値はこの場合も 0 となる。

- 予め、行列 B の最小固有値の下限 λ_B を我々の方法で厳密に求めた。
  $\lambda_B = 3.479325304159183 \times 10^{-6}$.
- 目的の第2固有値の近似値は $\tilde{\rho} = 9.87001850948186$.
- $\delta_1 = \delta_2 = 1.0 \times 10^{-15}$ に取った。

結果は次のようになった。

D_1 の負の要素数	1
D_2 の負の要素数	2
ε_1	$1.279393915300903 \times 10^{-12}$
ε_2	$1.239274369160737 \times 10^{-12}$
$\eta_1 = \varepsilon_1/\lambda_B$	$3.677132212304256 \times 10^{-7}$
$\eta_2 = \varepsilon_2/\lambda_B$	$3.561823804400553 \times 10^{-7}$
第2固有値の下限	9.87001814176864
第2固有値の上限	9.87001886566424
CPU Time	2356.9 sec

このように一般固有値問題に対しても適用できる。CPU time については次の注意を参照されたい。

Fortran90 での丸めの扱いについて

ここで述べた方法では、本質的な区間演算は必要でなく、内積演算に対して丸めの影響を考慮した限界が得られればよい。C 言語でプログラムをする場合には丸めの方向を制御する ieee_flag などのコマンドを用いれば簡単にできる。

C 言語と違って Fortran には演算の丸め方向を制御するコマンドがない。このため、演算のたびに結果を 1bit ずつずらす方法をとることになるが、これは演算速度を遅くするだけでなく、かなり overestimate した結果になることがある。演算速度については、現状では Fortran プログラムから C の命令を呼び出すなどの対策を取らざるを得ないと考えている。

ここで示した例では、プログラムはすべて Fortran90 で書かれ 500MHz のマシンで計算されているが、以上のことを考えあわせると、C 言語でプログラム作成した場合には CPU time をかなり減じることができるものと期待される。

References

1. J.H.Wilkinson, **The Algebric Eigenvalue Problem** Oxford Univ. Press, 1965

2. F.Chatelin, **行列の固有値**, 伊理正夫・由美訳, シュプリンガー・フェアラーク東京, 1993

3. 村田健郎, **対称帯行列固有値解析**, 数理研講究録 422, 1981

4. 小国力編, **行列計算ソフトウェア**, 丸善, 1991