数式処理 *Bulletin of JSSAC (200X)*
Vol. XX, No. X, pp. 1 - 8

数値線形代数における前処理と数式処理における固有多項式計算の高速実装について

木村 欣司 *

京都大学大学院情報学研究科

数値線形代数における前処理として, 数値不安定性のため, 現代ではどなたも利用していないが, ガウスの消去法により, 直接法として上 Hessenberg 行列への変換を行うアルゴリズムが存在する。 一方で, 数式処理における固有多項式計算では, 有限体を用いるために, 剰余算の少ないクリロフ部分空間法を用いたものが有効である。 上 Hessenberg 行列への変換を可能にするものと, ブロック コンパニオン行列への変換を許すことで, より高速化したものが考えられる。 数値計算における直接法として上 Hessenberg 行列への変換をおこなうアルゴリズムの演算の順序を変更することにより, 数式処理におけるクリロフ部分空間法を用いた上 Hessenberg 行列への変換と完全に一致することを紹介する。

1 既存の固有多項式計算プログラム

LinBox:Exact computational linear algebra(http://www.linalg.org/) というプログラムが, 整数行列の固有多項式を高速に計算できるプログラムとして有名である。しかし, このプログラムは, probabilistic アルゴリズムを採用しており, 正しい計算結果が得られない場合がある。これから紹介するアルゴリズムおよびその実装は, すべて deterministic である。そのような条件のもとで, 両者の実装を比較していることを注意する。

2 BLAS

Basic Linear Algebra Subprograms（BLAS）は, ベクトルと行列に関する基本線型代数操作を実行するライブラリ API のデファクトスタンダードである。ベクトルの内積, 密行列とベクトルの乗算, 密行列と密行列の乗算などが行える。倍精度浮動小数点数においても, IEEE754 の規格より仮数部 52bit が存在する為 (economized form), $2^{53}-1$ までは正確に整数を表現できる。さらに, 1.0×2^{53} であるから, 倍精度浮動小数点数の規格では, 0 から 2^{53} までのすべての整数を正確に表現できる。いま, 行列サイズを N とするとき, $0 < p \leq \sqrt{\frac{2^{53}}{N}}$ を満たす素数 p を法とする有限体 $\mathbb{Z}/p\mathbb{Z}$ を考える。BLAS が内積, 行列ベクトル乗算, 行列行列乗算を計算

*kkimur@amp.i.kyoto-u.ac.jp

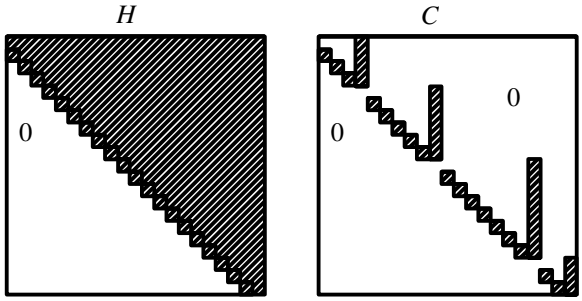
した後, p で剰余算を行うことで有限体 $\mathbb{Z}/p\mathbb{Z}$ を実現できる。よって, 厳密計算を行う数式処理においても, CPU の限界性能を追求する BLAS の恩恵に受け, 高速な計算を行うことが可能である。LinBox は, この BLAS を活用し, 高速な固有多項式の計算を行っている。

3 固有多項式を計算するための数学の基礎

$N \times N$ の有限体 $\mathbb{Z}/p\mathbb{Z}$ 上の行列 A が, 行列 V による相似変換

$$V^{-1}AV = H \text{ または } V^{-1}AV = C$$

により, 上 Hessenberg 行列 (H), 複数個のコンパニオン行列 (C) に変換されたとき, H, C から固有多項式を求めることは容易に可能である。上 Hessenberg 行列と複数個のコンパニオン行列は, 以下のような行列である。



整数行列の固有多項式を計算する場合には, 中国剰余定理を用いる為, 有限体 $\mathbb{Z}/p\mathbb{Z}$ 上の固有多項式の計算が重要になる。

3.1 固有多項式の係数の上界

中国剰余定理を用いて計算を行うためには, 固有多項式の係数の上界が必要となる。そのための定理を次に紹介する。

2 つの量を定義する。q は, 多変数多項式である。

$$\begin{aligned} \|q\|_1 &= \sum_{\alpha_1,\cdots,\alpha_s} |c(\alpha_1,\cdots,\alpha_s)|, \\ \|q\|_2 &= \sqrt{\sum_{\alpha_1,\cdots,\alpha_s} c(\alpha_1,\cdots,\alpha_s)^2}, \end{aligned}$$

ここで, $q = \sum_{\alpha_1,\cdots,\alpha_s} c(\alpha_1,\cdots,\alpha_s) x_1^{\alpha_1} \cdots x_s^{\alpha_s} \in \mathbb{Z}[x_1,\cdots,x_s]$。

$A = (a_{i,j}) \in \mathbb{Z}^{N\times N}$ についての Hadamard の上界は,

$$|\det(A)| \le \min\left\{ \prod_{i=1}^{N} \sqrt{\sum_{j=1}^{N} |a_{i,j}|^2}, \prod_{j=1}^{N} \sqrt{\sum_{i=1}^{N} |a_{i,j}|^2} \right\}$$

となる。$A = (a_{i,j}) \in \mathbb{Z}[x_1,\cdots,x_s]^{N\times N}$ についての Goldstein と Graham の上界は,

$$\|\det(A)\|_2 \le \min\left\{ \prod_{i=1}^{N} \sqrt{\sum_{j=1}^{N} \|a_{i,j}\|_1^2}, \prod_{j=1}^{N} \sqrt{\sum_{i=1}^{N} \|a_{i,j}\|_1^2} \right\}$$

となる。例えば, B の計算結果の係数の上界は,

$$B = \begin{vmatrix} a & 2b \\ 3c & 4d + f \end{vmatrix} = \underline{4}ad + \underline{1}af - \underline{6}bc$$

Goldstein と Graham の上界より, $\min(\sqrt{1+4}\sqrt{9+25}, \sqrt{1+9}\sqrt{4+25}) < 13.1$ となる。確かに, これは係数の上界になっている。すなわち, 行列式で定式化できる対象については, すべて中国剰余定理が使える。固有多項式は行列式で定式化できる為, 係数の上界は上記の定理を用いて容易に計算可能である。

3.2 数値解析における現状

	直接法	クリロフ部分空間法
H	Householder 変換または ガウス消去型	Arnoldi 法 (直交行列で空間を張る)
C	Danilevsky 法	数値的に発散, 利用不能

3.3 数式処理における現状

	直接法	クリロフ部分空間法
H	ガウス消去型のみ	下三角行列で空間を張る
C	Danilevsky 法	Dumas らが研究

数式処理におけるこれらの算法は, BLAS を有効に活用できない。しかし, 行列サイズが小さくかつ要素に入る整数の桁長が長い場合には, ここでの議論は有用である。

4 クリロフ部分空間法による H への変換

4.1 クリロフ部分空間法

初期ベクトル v から生成されるベクトルにより, 張られる空間

$$\mathrm{span}\{v, Av, A^2v, \cdots, A^{n-1}v\}$$

を利用する方法を, クリロフ部分空間法という。この概念を利用せずに, Hessenberg 行列, コンパニオン行列に相似変換する方法を直接法という。

4.2 クリロフ部分空間法による H への変換

$$A \begin{pmatrix} v_1 & v_2 & \cdots & v_n \end{pmatrix} = \begin{pmatrix} v_1 & v_2 & \cdots & v_n \end{pmatrix} H$$

$$H = \begin{pmatrix} h_{1,1} & h_{1,2} & \cdots & \cdots \\ h_{2,1} & h_{2,2} & \cdots & \cdots \\ & \vdots & \vdots & \cdots \end{pmatrix}$$

ここで, $v_j = (v_{i,j})$

$$v_{i,j} = \begin{cases} 0, \ (1 \leq i \leq j-1) \\ 1, \ (i = j) \\ v_{i,j} \end{cases}$$

さらに

$$v_{i,1} = \begin{cases} 1, \ (i = 1) \\ 0, \end{cases}$$

とすると,

$$Av_1 = h_{1,1}v_1 + h_{2,1}v_2 \tag{1}$$

$$Av_2 = h_{1,2}v_1 + h_{2,2}v_2 + h_{3,2}v_3 \tag{2}$$

$$\vdots \tag{3}$$

から順に決まる。式 (1)-式 (3) の左辺は, 明らかに内積の形をしている為, 少なく剰余算で実行可能である。さらに, 式 (1)-式 (3) を変形すると

$$\begin{aligned} Av_1 - h_{1,1}v_1 &= h_{2,1}v_2 \\ Av_2 - h_{1,2}v_1 - h_{2,2}v_2 &= h_{3,2}v_3 \\ &\vdots \end{aligned}$$

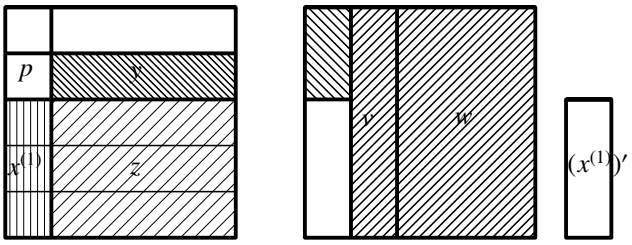
となり, 反復毎に新しい Av_j が生成され, そのベクトルからすでに求まっているベクトル $v_k(1 \leq k \leq j)$ の成分を抜き取り一次独立なベクトルを生成しているとみなすことができる。すなわち, 内積形式 LU 分解の演算を行うことに他ならない。以上より, 剰余演算の回数は, $O(n^2)$ で済む。しかし, $v_{j,j} = 1$ にできない場合があるので, その場合には pivot 選択をする。pivot 選択をしても 1 にできない場合がありその場合には以下のような計算を行う。pivot 選択をしても 1 にできない場合とは, すべてのベクトルの成分が 0 であることに他ならない。Av_j として, $v_k(1 \leq k < j)$ と従属なベクトルが生成された場合には, $h_{j+1,j} = 0$ として, 新たに初期ベクトルを,

$$v_{i,j} = \begin{cases} 0, \ (1 \leq i \leq j-1) \\ 1, \ (i = j) \\ 0, \end{cases}$$

として, 再スタートすればよいことがわかる。

5 直接法による H への変換

5.1 数値計算における直接法による H への変換



1.$\left(x^{(1)}\right)' \leftarrow x^{(1)}/(-p)$
2.$z' \leftarrow z + \left(x^{(1)}\right)' y$
3.$v' \leftarrow v + w\left(x^{(1)}\right)'$
1,2,3 の操作を繰り返すと, 固有値を保存したままで H に変形できる。この方法には, 問題点がある。与えられた行列 $A = A^{(0)}$ に対して,

$$\begin{aligned} A^{(1)} &= T_1 A^{(0)} \\ A^{(2)} &= A^{(1)}(T_1)^{-1} \\ A^{(3)} &= T_2 A^{(2)} \\ A^{(4)} &= A^{(3)} T_2^{-1} \\ &\vdots \end{aligned}$$

このように左右から作用を施すと, $A^{(0)}$ の全要素が, 有限体 $\mathbb{Z}/p\mathbb{Z}$ 上の代表元そのものであったとしても, $\mathbb{Z}$ 上で計算した後, 剰余を計算しない限り, $A^{(1)}$ は, 多くの場合に, もはや有限体 $\mathbb{Z}/p\mathbb{Z}$ 上の代表元ではなくなる (桁が増える)。すなわち, $O(n^3)$ 回の剰余演算が必要になる。

5.2 直接法における演算順序の変更

与えられた行列 $A = A^{(0)}$ に対して,

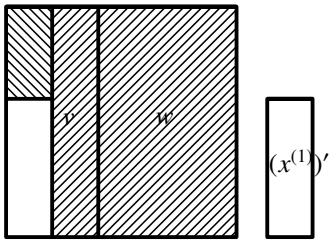
$$\begin{aligned} A^{(1)} &= A^{(0)}(T_1)^{-1} \\ A^{(2)} &= A^{(1)}(T_2)^{-1} \\ &\vdots \\ A^{(n-3)} &= A^{(n-4)}(T_{n-3})^{-1} \\ A^{(n-2)} &= T_1 A^{(n-3)} \\ A^{(n-1)} &= T_2 A^{(n-2)} \\ &\vdots \\ A^{(2n-6)} &= T_{n-3} A^{(2n-7)} \end{aligned}$$

と演算の順序を変更し, 計算に関係する領域だけに注目すると,

$$\begin{aligned} A^{(1)} &= A^{(0)}(T_1)^{-1} \\ A^{(2)} &= A^{(0)}(T_2)^{-1} \text{ で代用できる} \\ &\vdots \\ A^{(n-3)} &= A^{(0)}(T_{n-3})^{-1} \text{ で代用できる} \end{aligned}$$

これは, クリロフ部分空間法における $A^k v$ の演算に等しい。

各ステップにおける計算を見直す。



この演算においては, $\left(x^{(1)}\right)'$ に相当するベクトルが分かればよい。しかし, $\left(x^{(2)}\right)'$ を決定するには, 右からの作用が必要となる。すなわち,

$$A^{(0)}\text{の 1 列目} \to x^{(1)}\text{を決定} \tag{4}$$

$$T_1(A^{(1)}\text{の 2 列目}) \to x^{(2)}\text{を決定} \tag{5}$$

$$T_2T_1(A^{(2)}\text{の 3 列目}) \to x^{(3)}\text{を決定} \tag{6}$$

$$T_3T_2T_1(A^{(3)}\text{の 4 列目}) \to x^{(4)}\text{を決定} \tag{7}$$

$$\cdots \tag{8}$$

という演算が必要になる。これは, 「ベクトル $v_k(1 \le k \le j)$ の成分を抜き取り一次独立なベクトルを生成するための内積形式 LU 分解」と完全に一致する。H を作る立場で考える。

$$T_1(A^{(0)}\text{の 1 列目}) = (*, *, 0)^\top$$

よって, $T_2, \cdots, T_{n-3}$ の作用によって不変。

$$T_2T_1(A^{(1)}\text{の 2 列目}) = (*, *, *, 0)^\top$$

よって, $T_3, \cdots, T_{n-3}$ の作用によって不変。上記の性質を考慮すると, 式 (4)-式 (8) から得られる $\left(x^{(k)}\right)'$ は, 左からの作用だけでなく右から作用も完全に含んでいる。以上より, 直接法による H への変換において剰余算を最大限取り除いたものは, クリロフ部分空間法による H への変換に一致する。

6 主結果

数値解析におけるガウス消去型を使った直接法による H への変換において, 剰余算を減らすように工夫すると, そのアルゴリズムは, 数式処理における下三角行列で部分空間を張るクリロフ部分空間法による H への変換に帰着する。その帰結として, クリロフ部分空間法ではなく直接法と解釈することができる為, 実装において A, V, H という 3 の行列をメモリに確保する必要はなく, A の領域さえあれば実装可能であり, その分, キャッシュのヒット率が向上する。

7 性能評価 (1)

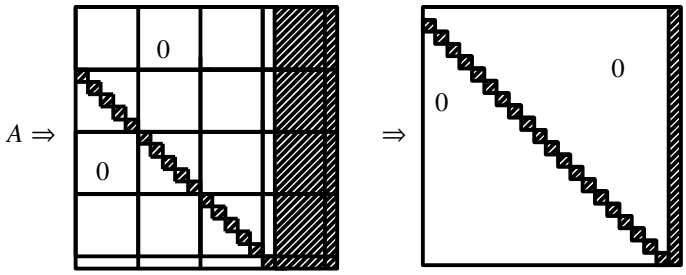
各要素が [1, 10] の乱数の完全密行列に対して, 逐次計算によって固有多項式を計算することでその性能を評価する。LinBox は, 固有多項式の計算が最も高速な LinBox 1.1.6 を用いる。Kimura I は, クリロフ部分空間法による H への変換を利用した計算法である。

n	LinBox without NTL	LinBox with NTL	算法 (Kimura I)
250	2.034	0.781	0.530
500	20.541	8.652	8.468
750	79.458	40.437	42.628
1000	213.544	118.066	134.601
1250	486.344	384.952	334.824

LinBox with NTL よりも, キャッシュヒット率が高い為, 次元が大きくなると, LinBox with NTL に勝つようになる。次元が大きくなるに連れて, LinBox without NTL と算法 (Kimura I) の実行時間は近づいてくる。よって, 別のアルゴリズムによるさらなる速度向上が必要になる。

8 BLAS を用いた高速化

$N \times N$ の実非対称行列 A を, 行列積を利用して帯コンパニオン行列 C に変換し, C から Wiedemann algorithm における最小多項式の計算法を用いて固有多項式を計算する。概念的に図示すると, 以下のようになる。



8.1 数式処理におけるブロックコンパニオン行列への変換法

以下のどちらかの方法を用いて, おこなう。(1)Dumas のアルゴリズムのブロック化, または, (2) ブロック Berlekamp-Massey アルゴリズム。ここでは, (1) のアルゴリズムを採用する。

$N \times NB(NB < N)$ の縦長の乱数行列 V を用意する。C を帯コンパニオン行列, C' を帯コンパニオン行列の右の帯の部分とする。N と $NB \times l$ が等しくない場合には, 微調整が必要であるが, $N = NB \times l$ の場合には,

$$\left(\begin{array}{cccc} V & AV & \cdots & A^{l-1}V \end{array} \right) C = \left(\begin{array}{cccc} AV & A^2V & \cdots & A^lV \end{array} \right) = A \left(\begin{array}{cccc} V & AV & \cdots & A^{l-1}V \end{array} \right)$$

この形の連立一次方程式を解くと, A と同じ固有値を持つ帯コンパニオン行列 C が手に入る。ただし, 任意の A に対して, 常にこの形を満たす C が存在するわけではないことを注意する。与えられた問題は, 以下の連立一次方程式を解く問題に帰着された。

$$\left(\begin{array}{cccc} V & AV & \cdots & A^{l-1}V \end{array} \right) C' = A^lV$$

8.2 帯コンパニオン行列から固有多項式を得る方法

以下のどちらかの方法を用いて, おこなう。(1) スカラー Berlekamp-Massey アルゴリズム (Wiedemann algorithm における最小多項式の計算法), または, (2)N 回の行列式計算 (N サンプリングポイント) と補間。ここでは, (1) のアルゴリズムを採用する。

9 性能評価 (2)

各要素が [1, 10] の乱数の完全密行列に対して, 逐次計算によって固有多項式を計算することでその性能を評価する。LinBox は, 固有多項式の計算が最も高速な LinBox 1.1.6 を用いる。Kimura II は, 上記の BLAS を用いた計算法である。

n	LinBox without NTL	LinBox with NTL	算法 (Kimura II)
250	2.034	0.781	0.958
500	20.541	8.652	9.393
750	79.458	40.437	39.995
1000	213.544	118.066	115.538
1250	486.344	384.952	277.168

算法 (Kimura II) は, 算法 (Kimura I) よりもさらにキャッシュヒット率が高い為, 次元が大きくなると, LinBox with NTL に大幅に勝つようになる。次元が大きくなるに連れて, LinBox without NTL と算法 (Kimura II) の実行時間は近づいてくる。しかし, 算法 (Kimura I) の場合よりもその近づき方は緩やかになる。

10 計算機環境

CPU は Intel(R) Core(TM) i7 X 980 3.33GHz, 1CPU(6Core), メモリは 24Gbyte, OS は Fedora 13, コンパイラは gcc 4.4.5, BLAS は GotoBLAS2-1.13 を用いた。