sdmpパッケージによるMapleの高速化-行列式の計算を例に(Ver.2)

木村欣司(京都大学大学院情報学研究科)

September 27, 2011

前回の補足(I)

前回の「GotoBALS入門」で紹介した内容を反映した世界最速の固有多項式計算プログラムは,
http://www-is.amp.i.kyoto-u.ac.jp/kkimur/charpoly.html
からダウンロードできます

前回の補足(II)

NehalemマイクロアーキテクチャのCPUにおいてGotoBLASを実行させる場合の注意

Intelの最新CPUであるNehalemマイクロアーキテクチャのCPUを搭載したマシンで，GotoBLASを動作させる場合には，Hyper-Thread technologyはoffにするべし

Windowsの場合 BIOSで簡単にoffにできる

MacにおけるHyper-Thread technology をoffにする方法

Mac OSでroot権限を取得し, terminalで, nvram SMT=0 とすると、OFFにできる

今日のお話

数式処理ソフトMapleは，計算速度が。。。
少なくとも，多変数多項式の４則演算については，sdmpパッケージを用いて高速化できる
http://www.cecm.sfu.ca/～rpearcea/
多変数多項式の４則演算だけで，なにか面白いことはできないであろうか?

高速な多変数多項式の４則演算があれば

(1)多変数多項式を成分にもつ行列式の計算を高速化できる

(2)終結式の計算を高速化できる

(3)Cylindrical Algebraic DecompositionのProjectionを高速化できる

(4)Dixonの多重多項式終結式の計算を高速化できる

(5)多変数多項式を成分にもつ連立一次方程式の計算を高速化できる

ただし，プログラミングをする元気があれば。。。

Dixonの多重多項式終結式

$$f(x,y) = x^2 + y^2 - 3, g(x,y) = xy - 1$$

交点を求める

$$\frac{\begin{vmatrix} f(x,y) & g(x,y) \\ f(\alpha,y) & g(\alpha,y) \end{vmatrix}}{x-\alpha}$$

$$= \begin{pmatrix} \alpha & 1 \end{pmatrix} \begin{pmatrix} y & -1 \\ -1 & -y^3+3y \end{pmatrix} \begin{pmatrix} x \\ 1 \end{pmatrix}$$

$$= 0$$

αに関係なく成立する

$$y = \lambda, v = \begin{pmatrix} x \\ 1 \end{pmatrix}$$

とおけば,

$$\begin{pmatrix} \lambda & -1 \\ -1 & -\lambda^3 + 3\lambda \end{pmatrix} v = 0$$

固有多項式

$$\begin{vmatrix} \lambda & -1 \\ -1 & -\lambda^3 + 3\lambda \end{vmatrix} = 0$$

から，yの値が求まる，**この方法は，3変数以上でも適用可能である**

有理式を成分にもつ行列式の計算法

(1)有理式のまま計算する

(2)

$$\begin{vmatrix} \frac{1}{(2+3x)(14+15x)} & \frac{4+5x}{(6+7x)(14+15x)} \\ \frac{8}{9+10x} & \frac{11+12x+13x^3}{(2+3x)(16+17x)} \end{vmatrix} \Rightarrow$$

$$\begin{vmatrix} 6+7x & (2+3x)(4+5x) \\ 8(2+3x)(16+17x) & (9+10x)(11+12x+13x^3) \end{vmatrix}$$

(3)

$$\Rightarrow \begin{vmatrix} a & b(4+5x) \\ 8c & ad(11+12x+13x^3) \end{vmatrix},$$

$$a=\frac{1}{2+3x}, b=\frac{1}{6+7x}, c=\frac{1}{9+10x}, d=\frac{1}{16+17x}$$

多変数多項式を成分にもつ行列式の計算法

1.小行列式展開, $O(2^n)$, 割り算なし

2.久留島-ラプラス展開, ???, 割り算なし

3.fraction-free Gauss消去法, $O(n^3)$, 割り算あり

4.**木村の補間法**, ???, mod 演算あり

5.**Berkowitzの方法**, Fadeevの方法, $O(n^4)$, 割り算なし

単純な多項式の演算の数だけで性能を決めてはならない

fraction-free Gauss消去法

多項式で構成される$n \times n$の行列Cが与えられたとき，$\hat{c}_{0,0}^{(-1)} = 1, \hat{c}_{i,j}^{(0)} = C_{i,j}$として，以下の漸化式に従って計算を行う

$$\hat{c}_{i,j}^{(k)} = (\hat{c}_{k,k}^{(k-1)} \hat{c}_{i,j}^{(k-1)} - \hat{c}_{k,j}^{(k-1)} \hat{c}_{i,k}^{(k-1)}) / \hat{c}_{k-1,k-1}^{(k-2)}$$

$$k+1 \leq i \leq n, k+1 \leq j \leq n, k = 1, \cdots, n-1$$

行列式は，$\hat{c}_{n,n}^{(n-1)}$に現れる，0による除算を避けるために，ピボット選択を適宜行ってよい，Jacobiの恒等式が理論の背景，**数学的にカッコイイだけ，性能は悪い**

fraction-free Gauss消去法

多項式で構成される$n \times n$の行列Cが与えられたとき，$\tau_{0,0}^{(-1)} = 1, \tau_{i,j}^{(0)} = C_{i,j}$として，以下の漸化式に従って計算を行う

$$\tau_{i,j}^{(k)} = (\tau_{k,k}^{(k-1)} \tau_{i,j}^{(k-1)} - \tau_{k,j}^{(k-1)} \tau_{i,k}^{(k-1)}) / \tau_{k-1,k-1}^{(k-2)}$$
$$k+1 \leq i \leq n, k+1 \leq j \leq n, k = 1, \cdots, n-1$$

行列式は，$\tau_{n,n}^{(n-1)}$に現れる，0による除算を避けるために，ピボット選択を適宜行ってよい，Jacobiの恒等式が理論の背景，**数学的にカッコイイだけ，性能は悪い**

Berkowitzの方法(I)

固有多項式の計算法

$$A = \begin{pmatrix} A_r & S \\ R & a_{r+1,r+1} \end{pmatrix}$$

$$T_A = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ -a_{r+1,r+1} & 1 & \cdots & 0 \\ -RS & -a_{r+1,r+1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -RA_r^{r-2}S & -RA_r^{r-3}S & \cdots & 1 \\ -RA_r^{r-1}S & -RA_r^{r-2}S & \cdots & -a_{r+1,r+1} \end{pmatrix}$$

Berkowitzの方法(II)

[procedure Berkowitz(A)]

if dim A=1 then return $[1, -A_{1,1}]^\top$

else $r \leftarrow (\dim A) - 1$

Decompose $A = \begin{pmatrix} A_r & S \\ R & a_{r+1,r+1} \end{pmatrix}$

Compute T_A

$[X_{r+1}, \cdots, X_1]^\top \leftarrow$ Berkowitz(A_r)

$[X_{r+2}, \cdots, X_1]^\top \leftarrow T_A [X_{r+1}, \cdots, X_1]^\top$

return $[X_{r+2}, \cdots, X_1]^\top$

何を使えばよい?

小次元 ⇒ 1.小行列式展開, 2.久留島-ラプラス展開

多項式の変数の種類が少ない ⇒ 4.木村の補間法

大次元、多項式の変数の種類が多い ⇒ そんなものは，計算できるわけがない

Resultant(終結式)の計算法

1. Collinsのsubresultantによる計算法
2. 行列式を経由した計算法
3. 木村の補間法

行列式を経由した終結式の計算法

$$f = a_2x^2 + a_1x + a_0, g = b_1x + b_0$$

Sylvesterの表現法

$$\mathrm{res}_x(f,g) = \begin{vmatrix} a_2 & a_1 & a_0 \\ b_1 & b_0 & 0 \\ 0 & b_1 & b_0 \end{vmatrix}$$

m次とn次の多項式の終結式の

$(m+n) \times (m+n)$行列式による表現

$$f = a_2x^2 + a_1x + a_0, g = b_1x + b_0$$

Bezoutの表現法

$$\mathrm{res}_x(f,g) = \begin{vmatrix} b_0 & b_1 \\ -b_1a_0 & b_0a_2 - b_1a_1 \end{vmatrix}$$

m次とn次$(m > n)$の多項式の終結式の$m \times m$行列式による表現

1.小行列式展開，2.久留島-ラプラス展開の重要な応用

非凸の多項式型の最適化問題 $\subset$ Quantifier Elimination

Q.E.は，Cylindrical Algebraic Decompositionで解ける

C.A.D.は，Resultant(終結式)を必要とする

小行列式展開法

$$
\begin{vmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \\ d_1 & d_2 & d_3 & d_4 \end{vmatrix} =
\begin{aligned}
&+a_1(b_2(\underline{c_3d_4 - d_3c_4}) - c_2(\underline{b_3d_4 - d_3b_4}) \\
&+d_2(\underline{b_3c_4 - c_3b_4})) - b_1(a_2(\underline{c_3d_4 - d_3c_4}) \\
&-c_2(\underline{a_3d_4 - d_3a_4}) + d_2(\underline{a_3c_4 - c_3a_4})) \\
&+c_1(a_2(\underline{b_3d_4 - d_3b_4}) - b_2(\underline{a_3d_4 - d_3a_4}) \\
&+d_2(\underline{a_3b_4 - b_3a_4})) - d_1(a_2(\underline{b_3c_4 - c_3b_4}) \\
&-b_2(\underline{a_3c_4 - c_3a_4}) + c_2(\underline{a_3b_4 - b_3a_4}))
\end{aligned}
$$

計算結果の使いまわし(bottom up)をするとスピードはあがる，メモリは溢れる!

久留島-ラプラス展開

$$L_1 = a_{4,4}a_{5,5} - a_{4,5}a_{5,4}, L_2 = a_{4,3}a_{5,5} - a_{4,5}a_{5,3}, L_3 = a_{4,3}a_{5,4} - a_{4,4}a_{5,3},$$
$$L_4 = a_{4,2}a_{5,5} - a_{4,5}a_{5,2}, L_5 = a_{4,2}a_{5,4} - a_{4,4}a_{5,2}, L_6 = a_{4,2}a_{5,3} - a_{4,3}a_{5,2},$$
$$L_7 = a_{4,1}a_{5,2} - a_{4,2}a_{5,1}, L_8 = a_{4,1}a_{5,5} - a_{4,5}a_{5,1}, L_9 = a_{4,1}a_{5,4} - a_{4,4}a_{5,1},$$
$$L_{10} = a_{4,1}a_{5,3} - a_{4,3}a_{5,1}$$

$$\begin{aligned}
\det(A) = &+(a_{1,1}a_{2,2} - a_{1,2}a_{2,1})(a_{3,3}L_1 - a_{3,4}L_2 + a_{3,5}L_3) \\
&-(a_{1,1}a_{2,3} - a_{1,3}a_{2,1})(a_{3,2}L_1 - a_{3,4}L_4 + a_{3,5}L_5) \\
&+(a_{1,1}a_{2,4} - a_{1,4}a_{2,1})(a_{3,2}L_2 - a_{3,3}L_4 + a_{3,5}L_6) \\
&-(a_{1,1}a_{2,5} - a_{1,5}a_{2,1})(a_{3,2}L_3 - a_{3,3}L_5 + a_{3,4}L_6) \\
&+(a_{1,2}a_{2,3} - a_{1,3}a_{2,2})(a_{3,1}L_1 - a_{3,4}L_8 + a_{3,5}L_9) \\
&-(a_{1,2}a_{2,4} - a_{1,4}a_{2,2})(a_{3,1}L_2 - a_{3,3}L_8 + a_{3,5}L_{10}) \\
&+(a_{1,2}a_{2,5} - a_{1,5}a_{2,2})(a_{3,1}L_3 - a_{3,3}L_9 + a_{3,4}L_{10}) \\
&+(a_{1,3}a_{2,4} - a_{1,4}a_{2,3})(a_{3,1}L_4 - a_{3,2}L_8 + a_{3,5}L_7) \\
&-(a_{1,3}a_{2,5} - a_{1,5}a_{2,3})(a_{3,1}L_5 - a_{3,2}L_9 + a_{3,4}L_7) \\
&+(a_{1,4}a_{2,5} - a_{1,5}a_{2,4})(a_{3,1}L_6 - a_{3,2}L_{10} + a_{3,3}L_7)
\end{aligned}$$

計算結果の使いまわしができる

計算結果の使いまわし＝テーブル参照

一度計算した結果を，使いまわす伝統的な方法

Lisp言語による数式処理の実装
Lisp言語のHash関数によるテーブル参照

Hash関数を使って，小行列式展開を実装する必要が本当にあるのか?

5 × 5から3 × 3の小行列式を抜き出す

左のindexから，右の"データ位置"を出す

3	4	5	1
2	4	5	2
1	4	5	3
2	3	5	4
1	3	5	5
1	2	5	6
2	3	4	7
1	3	4	8
1	2	4	9
1	2	3	10

計算によるデータ位置への到達

$w[k]$は, index, 例(2 3 4), $c[i][j] = \binom{i}{j-1}$

```
n=5; i=3; u=1; b1=n-1;
for(k=i;k>0;k--){
  for(b2=b1;b2>=w[k];b2--){
    u=u+c[b2][k];
  }
  b1=b2-1;
}
```

多変数多項式のメモリにおける表現方法

再帰表現 ⇒ 因数分解で効率的

分散表現 ⇒ 多項式の積で効率的

geobucket表現⇒ グレブナ基底の計算で効率的

因数分解においては，再帰表現が絶対的に効率的，分散表現から再帰表現への変換が必要

行列式の展開による計算では，“多変数多項式の積” の計算が必要

再帰表現 $\Rightarrow$ １変数多項式の場合の筆算方式の再帰的実行

分散表現 $\Rightarrow$ **heapの利用(sdmpの方法)**

分散表現 $\Rightarrow$ **分割統治法(Singularの方法)**

$$(f_1 + f_2) \times (g_1 + g_2) =$$
$$f_1 g_1 + f_1 g_2 + f_2 g_1 + f_2 g_2$$

再帰的に実行

分散表現とは?

多変数多項式を，係数と指数の組の結合されたリストで持つ

$$f = 5x^2y + 6x + 7y + 8$$

$$(5, [2, 1]) \to (6, [1, 0]) \to (7, [0, 1])$$

$$\to (8, [0, 0])$$

分散表現における多変数多項式の積(heapの利用)

X_j, Y_jは，ある順序でそれぞれすでにソートされている，$m \leq n$のとき

$$
\begin{aligned}
& (X_1 + X_2 + \cdots + X_m)(Y_1 + Y_2 + \cdots + Y_n) \\
= \; & X_1(Y_1 + Y_2 + \cdots + Y_n) + \\
& X_2(Y_1 + Y_2 + \cdots + Y_n) + \\
& \cdots \\
& X_m(Y_1 + Y_2 + \cdots + Y_n)
\end{aligned}
$$

待ち行列理論

$$(X_1Y_1 + X_1Y_2 + \cdots + X_1Y_n) +$$
$$(X_2Y_1 + X_2Y_2 + \cdots + X_2Y_n) +$$
$$\cdots$$
$$(X_mY_1 + X_mY_2 + \cdots + X_mY_n)$$

$Y_1, Y_2, \cdots, Y_n$は，すでにソートされているために，X_jの作用によっても，その順序は不変，m本の待ち行列ができていると思える

積の結果も，元の順序でソートする
＝m本の待ち行列に対して，１個のサーバーを置き，待ち行列の先頭の位置の人の中で，順序が一番大きな人を探す
＝1個のサーバーの前に，中間的な2分木を用意する，２分木のrootの人はぬける，ぬけたことにより，２分木の作り直す，rootの位置の人が属する待ち行列の２番目の人は，中間的な2分木の中に入る，入ったことにより，２分木の作り直す,それを繰り返す

多変数多項式の積における再帰表現と分散表現の比較

sdmpの論文の結果参照

多変数多項式の積では，分散表現のほうが再帰表現よりも優れている

heapの究極の利用

$$u = \begin{pmatrix} a_1 & -b_1 & c_1 \end{pmatrix}$$

$$v = \begin{pmatrix} \begin{vmatrix} b_2 & b_3 \\ c_2 & c_3 \end{vmatrix} & \begin{vmatrix} a_2 & a_3 \\ c_2 & c_3 \end{vmatrix} & \begin{vmatrix} a_2 & a_3 \\ b_2 & b_3 \end{vmatrix} \end{pmatrix}$$

とすると

$$\begin{vmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{vmatrix}$$

$$= a_1 \begin{vmatrix} b_2 & b_3 \\ c_2 & c_3 \end{vmatrix} - b_1 \begin{vmatrix} a_2 & a_3 \\ c_2 & c_3 \end{vmatrix} + c_1 \begin{vmatrix} a_2 & a_3 \\ b_2 & b_3 \end{vmatrix}$$

$$= u \cdot v$$

成分同士の掛け算

$u[1]$と$v[1]$，$u[2]$と$v[2]$，$u[3]$と$v[3]$において，個別に待ち行列を生成するのではなく

内積を一つの演算と思い

$u[1]$と$v[1]$，$u[2]$と$v[2]$，$u[3]$と$v[3]$の積における待ち行列を始めに一気に生成し，内積全体で，1つのサーバーを用意すればよい

当面は

sdmpパッケージによるMaple上での多変数多項式を成分にもつ行列式の計算の実装を，お使いください

http://www-is.amp.i.kyoto-u.ac.jp/kkimur/susemi_etc.html

将来は

1.sdmpよりも高速な多変数多項式の積の実装

2.内積を1つの演算とした場合の多変数多項式を成分にもつ行列式の計算の実装

を，私が作りますので，ご期待ください

sdmpパッケージを利用するための最重要事項

sdmpの高速性のキモは, 多変数多項式のexponent vectorのいくつかの成分を1 wordに詰め込むこと(packオプションをつかう)

$$f = 5x^2y + 6x + 7y + 8$$

$$(5, [2, 1]) \to (6, [1, 0]) \to (7, [0, 1])$$

$$\to (8, [0, 0])$$

32bitのマシン，pack＝2

上位16bit	下位16bit
2	1

しかし，動的に，exponent vectorを延ばす機能は持っていない

⇒　行列式の計算結果の次数に対する上界計算アルゴリズムが必要

sdmpパッケージを利用するための数学の問題

$$\begin{vmatrix} 1+x & 2+2x \\ 2+2x & 4+4x \end{vmatrix} = 0$$

小行列式の次数が全体の次数よりも大きい場合があるので，“行列式の計算結果の次数に対する上界” というよりも，

$$\begin{vmatrix} a_1 + a_2 x & a_3 + a_4 x \\ a_5 + a_6 x & a_7 + a_8 x \end{vmatrix}$$

与えられた多項式に対するgeneric positionの多項式の “行列式の計算結果の次数に対する上界” を考えることにする(すべての小行列式の次数の上界の最大値を計算できれば。。。)

だれでも思いつく行列式の計算結果の次数に対する上界計算アルゴリズム

各行の最大次数の総和と各列の最大次数の総和の小さいほう

計算量のオーダーは，$O(n^2)$

賢い方法

多項式係数を無視した場合の多項式行列の行列式の最大次数の問題は,
Linear Assignment Problemに帰着する

Hungarian algorithm

与えられた行列式から

$$\begin{pmatrix} 5 & 2 & 4 \\ 1 & 2 & 1 \\ 4 & 2 & 3 \end{pmatrix}$$

という行列を考える，この行列に対して，Hungarian algorithm を適用するとこで，$O(n^3)$でタイトなboundを手に入れることができる

cf. http://en.wikipedia.org/wiki/Hungarian_algorithm

まだ，maple上では実装していません，ごめんなさい

最後に

小次元 $\Rightarrow$ 1.小行列式展開，2.久留島-ラプラス展開

http://www-is.amp.i.kyoto-u.ac.jp/

kkimur/susemi_etc.html

多項式の変数の種類が少ない $\Rightarrow$ 4.木村の補間法

http://www-is.amp.i.kyoto-u.ac.jp/

kkimur/susemi_interpolation.html

別の機会に，「木村の補間法入門」という話題で，お話することを約束します