数理解析研究所講究録 1335 巻 2003 年 28-32

多項式表現と行列演算の改良

兵頭 礼子
HYODO NORIKO*
AlphaOmega Inc.

村尾 裕一
MURAO HIROKAZU†
電気通信大学

齋藤 友克
SAITO TOMOKATSU‡
AlphaOmega Inc.

概　要

行列成分が多項式である行列の積演算を高速化することを目的とした実験報告である．実験対象アルゴリズムとしては，[1] において実装実験をおこなった Strassen-Winograd アルゴリズムと行列の定義にしたがった基本的な積和演算のアルゴリズムである．本報告は，Risa/Asir の多項式の内部表現を Hash 関数を利用した構造に変え，行列の積演算の高速化をめざす．

1　始めに

数値計算において最も基本的かつ有用なアルゴリズムは，『線形計算』である．一方，数式処理システムにおいて線形計算は，比較的軽視されていたアルゴリズムである．軽視されている原因は，線形演算のアルゴリズムが数式処理の他のアルゴリズムと比べると，単純であり計算量的には軽いアルゴリズムであると考えられてきた点である．

しかし，要素が多項式であることが前提である数式処理システムの場合，単純な演算回数を低減させるだけの計算アルゴリズムの適用は，システムの高速化がはかれるか否かにつて十分な検討を必要とする．要素が多項式である場合，計算に必要な資源を考慮すると数式処理における線形計算アルゴリズムは，改良の余地のない単純なアルゴリズムと考えることはできない．特にシステム資源の問題は，行列の積の問題のみならず数式処理固有の問題であり，数式処理システム全般の問題である．

本論は，数式処理システムにおける行列演算，特に行列の積に関する必要資源の解析をおこなった．この解析に基づき，行列演算の高速化をはかることを目標とする実験報告である．

本研究の最終的な目標は，数式処理システムにおいて，大きなサイズの行列を自然に取り扱うことである．当面の目的は，

- 演算に必要な資源の低減化，
- 演算速度をを高速化，

をすることである．

2　単純な解決法

数値計算において，行列の積の高速化技法としては，Strassen-Winograd アルゴリズム（もしくはその改良型）が良く知られた手法である．高速化の実態は，行列の分割による各成分に対する積演算回数の低減である．これにより積に関する計算量は，$O(n^3)$ から $O(n^{2.8})$ に低減する [2][3]．(アルゴリズムの改良によりこの計算量は，理論的にさらに低下させることが可能であることが知られている．)

このような高速アルゴリズムを導入した場合，加法演算の回数は増加する．このことは，積演算が加法演算より重い処理である数値計算の場合は，アルゴリズムの高速化と言う目的に対し，合理的な選択である．また，記憶領域の必要量も増加する．しかし，単位長演算である数値計算の場合，あまり問題になる増加ではない．

実際の行列に対する Strassen-Winograd アルゴリズムの適用は，アルゴリズムのオーバーヘッドが大きいため，行列のサイズが 8 程度から実際の計算時間の低減に効くといわれている．

*noriko@a2z.co.jp
†murao@cs.uec.ac.jp
‡saito@a2z.co.jp

3 数式処理（Risa/Asir）に適用

Risa/Asir では，行列の積は内積の定義にしたがい行と列の内積演算により実装されている．Risa/Asir の制御変数によるアルゴリズムの切替えを導入し，Strassen-winograd アルゴリズムが古典的アルゴリズムと同様に利用できるようにした．

しかし，デフォルトのアルゴリズムを Strassen-Winograd アルゴリズムとすることは，4 節で示すように問題がある．

Risa/Asir での動作の制御

Risa/Asir での行列の積演算の制御は，以下のようにした．

- 制御は，Risa/Asir の組み込み関数 `ctrl` による．
- 制御変数は，`StrassenSize` である．

  `ctrl("StrassenSize",0)` 内積のみを利用するアルゴリズムを実行する．
  （Risa/Asir のデフォルト）

  `ctrl("StrassenSize",n)` アルゴリズムの適用は

  - 行列サイズが n 以上の場合に Strassen-Winograd アルゴリズムを適用
  - それ未満の場合は，内積のみを利用する．

4 Risa/Asir に実装した場合の問題点

Risa/Asir に実装して計算実験を繰り返した結果，理論値と解離した結果がでることがある．変動要因としては，行列の成分により実行時間が予想を越えて変動することより，数式処理システムに Strassen-Winograd アルゴリズムを適用したと考えられる．

数値行列の場合

行列成分が数値である場合の実験結果である．この実験の目的は，Risa/Asir の制御構造等の処理が通常の数値システムとなんら変わりがないことの確認である．

行列成分に，最小単位桁数の数値を入れて計算実験をした．この実験は，また，入力データを調整して最終結果が Risa/Asir の数の表現長が最小単位桁数になるものを選んだ．

表 1: Risa/Asir の行列演算速度（成分が数の場合）

	アルゴリズムの切替えタイミング			
行列サイズ	0	1	2	4
8 × 8	0.0002	0.000768	0.00033	0.000338
16 × 16	0.001423	0.005326	0.001977	0.001607
32 × 32	0.013516	0.034233	0.014642	0.009624
64 × 64	0.175903	0.235951	0.097103	0.062697
128 × 128	1.54873	1.61777	0.641289	0.389518
256 × 256	16.281	11.0553	4.26196	2.43897

アルゴリズムの切替えタイミングは，行列がそのサイズ以下にまで分割された場合，通常の内積演算による行列の積を実行する意味である．

この実験結果から言える点は，Risa/Asir において行列成分が，すべて数値である場合は，概ね理論値程度であることである．

極端に遅い事例

数式処理システムの特徴である，多項式を成分に持つ行列の積に関しては，数値成分行列とは異なった実験結果となった．理論値と比べ計算時間が誤差の範囲を越えて速い場合と遅い場合がある．

速い事例は，行列成分である多項式が密である場合である．(ここで密とは，与えられた多項式の最高次の単項式から最低次の項までほとんど単項式として存在する場合である．)

例えば，$n \times n$ 行列の成分 $a_{i,j}$ が，

$$a_{i,j} = \sum_{k=0}^{n} \sum_{l=0}^{n} c_{k,l} x^k y^l, \quad c_{k,l} \neq 0$$

である場合は，Strassen-Winograd アルゴリズムは，内積演算による行列の積と比較して高速であり，かつ計算量から導き出した計算時間と比べても高速である（[1] 参照）．

一方，理論値と比べ格段に計算が遅い事例が存在する．例えば，行列成分が全て異なる変数からなる行列の積，

$$\begin{pmatrix} a_1 & a_2 & a_3 & a_4 \\ a_5 & a_6 & a_7 & a_8 \\ a_9 & a_{10} & a_{11} & a_{12} \\ a_{13} & a_{14} & a_{15} & a_{16} \end{pmatrix} \begin{pmatrix} b_1 & b_2 & b_3 & b_4 \\ b_5 & b_6 & b_7 & b_8 \\ b_9 & b_{10} & b_{11} & b_{12} \\ b_{13} & b_{14} & b_{15} & b_{16} \end{pmatrix} \tag{1}$$

ここで a_i, b_j $i,j = 1,\cdots,16$ は，最高次 1 次の多項式である．

表 2 に示すように従来のアルゴリズムと比較して 10 倍以上の時間がかかる．行列の成分である多項式が疎である場合にこの現象が起る．

表 2: 理論値と比較して実行時間が極端に遅い事例

行列サイズ	内積による積	Strassen-Winograd 法
4 × 4	0.000261 秒	0.002474 秒
8 × 8	0.003957 秒	0.05579 秒

計算速度の理論的な正確な分析は困難である．

5 数式処理の場合の問題点

Strassen-Winograd アルゴリズムを多項式に対して適用した場合，基本演算レベルでは計算量が低下しているが，実行時間が増大する場合が存在する．また，計算量が低下する以上に計算時間が短縮される事例もある．この問題を解決するためには，新たな概念が必要と考える．

項演算量

数式処理システムの場合，計算量として従来の積の回数や分岐の回数という量は，本事例のような場合には，計算時間の判断材料としての計算量としては問題が[1)]あることが解った．そこで，この現実の計算時間と計算量の矛盾をを解決するために，のような仮説を設定し，この問題を検証する．

仮説

「数式処理の計算量としては，積に関する計算量よりも項をどのくらい生成し操作したかの方が実際の計算時間に良く対比する．」

この概念は，現時点ではまだ実験的かつ定性的なものである．しかし，アルゴリズムを詳細に検討すれば定量的な記述が可能であると考える．この計算量を項演算量と呼ぶことにする．正確な定義は，現時点では与えることができない．

[1)]現実の計算時間と解離していると言う意味である．

仮説の正当性の検証のため，基礎資料として次の基本データを採取した．基本データとしては，項レベルの和，積の回数と和，積を選んだ．

このデータを採取するため実行プログラムにトラップを仕掛け計測した．計測のターゲットとしての行列は，第 4 節の (1) の事例における行列とする．多項式演算の項レベルまでの回数を求める．

表 3: 項レベルの計算量 4 × 4 の場合

	内積計算のみ	Strassen-Winograd 法
和の回数	64	1040
積の回数	64	224
和の領域	128	5104
積の領域	64	224

表 4: 項レベルの計算量 8 × 8 の場合

	内積計算のみ	Strassen-Winograd 法
和の回数	512	42968
積の回数	512	6272
和の領域	1024	893880
積の領域	512	6272

以上の実験により，Strassen-Winograd アルゴリズムは，項を処理する計算が内積をもちいる従来の計算アルゴリズムと比べ多大に増大していることがわかる．また，項の増大と計算時間は，ほぼ比例していることも示すことができる．

6 新しい多項式表現の実装

項の処理が実際の計算時間と比例していることを着目すると，多項式の内部表現と計算時間は，多大な関係があることがわかる．

Risa/Asir においては，多項式の内部表現を再帰的表現として表している．しかし，項の処理が計算時間に寄与する割合が高いのであれば，多項式を再帰表現とするよりも，項に着目した実装をした方が良いのではないかと考えられる．

Risa/Asir において，項に着目した実装としては，グレブナ基底の為の分散表現多項式がある．これを行列の積の場合に適用できるように修正を行った．この実装によると行列の積は本来の実装と比べると 1 割程度遅くなった．

Hash による多項式の実装

Risa/Asir に新たな多項式表現を導入し，巾積による多項式表現を考えた．各項は，Hash 表に登録され表を引くことにより実際の項と対応付けた．

Hash 関数の仕様 Hash 関数の要求する仕様としては，積を容易に計算するために，Hash 値の積が項の積の Hash 値とシノニムを除いて同じになるように定めた．

項の表現 項は，数値としての係数と Hash 値のペアにより表現した．

多項式の表現 多項式は，項のリストとして表現した．

実装の結果

第 4 節の (1) の事例における行列に対し速度実験をおこなった.

表 5: Hash と従来の形式の計算例（16 × 16 行列の場合）

	内積計算のみ	Strassn-Winograd 法
従来の表現	0.02102 秒	0.6442 秒
Hash 表現	0.008129 秒	0.4563 秒

この結果より直ちに Hash による内部表現が良いか否かの判断はできないが，内積のみの行列の積ならびに Strassen-Winograd アルゴリズムの双方で十分な計算速度の改良がみられる.

7 結論

今回の実装はテスト的な実装である．実装のクオリティに関して問題が多いため，実行時間等は参考程度に考える必要がある．しかし，計算速度の向上は，無視できないものがあると考える.

今後の課題であるが，Hash 関数の決定は，実験的かつ理論的な考察をして選ぶ必要がある．しかし，今回利用した程度の簡単な Hash 関数であっても表 5 に示したように十分な速度向上があることより，実用に耐える Hash 関数の構築は，今後の大きな課題である.

参 考 文 献

[1] 兵頭 礼子, 村尾 裕一, 齋藤 友克 (2002). *Risa/Asir の Matrix* **演算の新しい実装について 数理解析研究所講究録** *1295 Computer Algebra – Alogrithms, Implementations and Applications,* **京都大学数理解析研究所**, *2002,213–219*

[2] Strassen, V. (1969). Gaussian elimination is not optimal. *Numer. Math.*, 13:354–356.

[3] Winograd, S. (1971). On multiplication of 2 × 2 matrices. *Linear Algebra and its Applications*, 4:381–388.