数理解析研究所講究録 1295 巻 2002 年 213-219

Risa/Asir の Matrix 演算の新しい実装について

兵頭 礼子* AlphaOmega Inc.
村尾 裕一† 電気通信大学
齋藤 友克‡ AlphaOmega Inc.

Abstract

近年数式処理の分野においても, 高速算法 (例えば FFT など) の利用は常識となっている. しかし, Risa/Asir では, 行列の演算には古典的計算方法が取られており, 特に高速化ははかられていない. 今回, 数値計算分野のアルゴリズムを Risa/Asir に実装したところ, 5 次多項式を各要素に持つ 64×64 行列において 3 倍強の速度で乗算を行うことができた. 他のサイズの行列でも, 古典的アルゴリズムよりも高速に乗算を行うことができる. しかし, 理論的な速度の向上と実際は異なっており今後の課題である.

1 Risa/Asir の行列乗算の実情

Risa/Asir は, 行列の積は定義にしたがった行列の内積演算によるアルゴリズムが実装されている. $l \times m$ の行列 A, と $m \times n$ の行列 B の乗算を行う場合, 積の回数は $l \cdot n \cdot m$ 回, 和は $l \cdot n(m-1)$ 回発生する. よって, 計算量は $n \times n$ の正方行列の積の場合, $O(n^3)$ である.

2 今回 Risa/Asir に実装した方法

2.1 Strassen-Winograd アルゴリズム

$l \times m$ 行列 A, $m \times n$ 行列 B において $A \cdot B$ の演算を行う場合, 行列 A, B を

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

とすると

$$A \cdot B = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & w + v + (A_{12} - s_2)B_{22} \\ w + u + A_{22}(B_{21} - t_2) & w + u + v \end{pmatrix}$$

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

$$
\begin{array}{lcll}
s_1 & = & A_{21}+A_{22} & \\
s_2 & = & s_1-A_{11} & = -A_{11}+A_{21}+A_{22} \\
t_1 & = & B_{12}-B_{11} & \\
t_2 & = & B_{22}-t_1 & = B_{11}-B_{12}+B_{22} \\
u & = & (A_{11}-A_{21})(B_{22}-B_{12}) & \\
v & = & s_1t_1 & = (A_{21}+A_{22})(B_{12}-B_{11}) \\
w & = & A_{11}B_{11}+s_2t_2 & = A_{11}B_{11}+ \\
 & & & (-A_{11}+A_{21}+A_{22})(B_{11}-B_{12}+B_{22})
\end{array}
$$

となる.

Strasse-Winograd の方法での計算量は, 次数が半分の行列の乗算が 7 回, 加算が 15 回発生する.

次数 $m2^k$ の行列の計算量を $T(m,k)$ とすると,

$$
\begin{aligned}
T(m,k) &= 7\times T(m,k-1)+15\times(\text{次数 } m2^{k-1} \text{ の行列の和} : m2^{2(k-1)}t_+) \\
&= 7^k\times T(m,0)+15m^2(2^{2k}-7^k)/(2^2-7)t_+ \\
&= 7^k(m^3t_*+m(m-1)t_+)-5m^2(2^{2k}-7^k)t_+
\end{aligned}
$$

よって,

$$O(7^k)\approx O(7^{\log_2 n})=O(n^{\log_2 7})\approx O(n^{2.8})$$

となる.

このアルゴリズムを, 分割した行列のサイズが 4 行 4 列以下になるまで適用し, 4 行 4 列以下になったところで古典的アルゴリズムを適用して, 演算を行う. 行列が奇数行奇数列であった場合は, 偶数行偶数列の行列になるように要素が全て 0 の行または列を padding して演算を実行するよう実装した.

3 計算速度の比較

3.1 実験データ

計算量の比較のため次の行列を考える. 行列 A, B は大きさ $n\times n$ の正方行列であり各成分はおのおの a_{ij}, $b_{i,j}$ とする.

Case I

$$
\begin{cases}
a_{ij} = (i+1)(j+1)(x^5+x^4+x^3+x^2+x), \\
b_{ij} = (i+1)(j+1)(x^5+x^4+x^3+x^2+x) \\
\qquad 1\le i,j\le n
\end{cases}
$$

Case II

$$
\begin{cases}
a_{ij} = (i+1)(j+1)(x^5+x^4+x^3+x^2+x), \\
b_{ij} = (i+1)(j+1)(y^5+y^4+y^3+y^2+y) \\
\qquad 1\le i,j\le n
\end{cases}
$$

とする. また実験は, AMD athron 1400Mhz (OS:FreeBSD 4.4) によって行った. 演算時間の計測は秒単位である.

3.1.1 Case I の演算時間

古典的アルゴリズム (標準) と Strassen-Winograd アルゴリズムで Case I の場合の $A \times B$ を実行し, 演算時間 (単位は秒) を計測 (表 1) する.

表 1: 演算時間の比較 (Case I)

Size	標準法	Strassen-Winograd	比
8	0.01598	0.01526	1.047
16	0.181	0.1026	1.764
32	1.137	0.5701	1.994
64	10.24	2.761	3.709
128	94.78	12.83	7.387
256	864.1	60.18	14.36

3.1.2 Case II の演算時間

Case I と同様に Case II の場合の古典的アルゴリズムと Strassen-Winograd アルゴリズムで $A \times B$ の演算を実行し, 計算速度を計測 (表 2) する. 表 1, 表 2 から,

表 2: 演算時間の比較 (Case II)

Size	標準法	Strassen-Winograd	比
8	0.01154	0.01178	0.9796
16	0.1105	0.08877	1.245
32	1.025	0.5214	1.966
64	9.235	2.684	3.441
128	75.95	13.03	5.829
256	953.1	66.13	14.41

Strassen-Winograd アルゴリズムを適用した場合, 古典的アルゴリズムで演算を行った場合よりも演算時間が短くなっている. しかし, 演算時間の比が, 計算量の比とは明らかにかけ離れた値になっている. よって, 計算量以外の要因が, 演算時間の短縮につながっていると考えられる.

4 演算回数, 計算量の比較

4.1 Case I の場合の演算回数

演算の高速化が計算量以上にはかられたため, 計算量以外の要因を考える. Case I の場合の行列の乗算を行う際に, 行列要素どうしの演算の加算, 乗算の発生回数をカウントした. 以下の表 3 に示す. 行列要素どうしの演算回数では, 演算回数は古典的アルゴリズムに比べ減少しているものの, 計算量の理論通りの減少率となり, 古典的アルゴリズムと Strassen-Winograd アルゴリズムの演算速度の差につながるような関係は見付からなかった.

表 3: 行列要素どうしの演算回数 (Case I)

Size	標準的アルゴリズム			Strassen-Winograd		
	加算	乗算	合計	加算	乗算	合計
8	448	512	960	624	448	1072
16	3840	4096	7936	5520	3136	8656
32	31744	32768	64512	43248	21952	65200
64	258048	262144	520192	321168	153664	474832
128	2080768	2097152	4177920	2321904	1975648	4297552
256	16711680	16777216	33488896	16548240	7529536	24077776

そこで, さらに行列要素の各項レベルでの演算回数を調べる. つまり, 加算が発生した場合は, 加算した結果の項の数, 乗算の場合は掛け合わせる各要素の項の数を乗算した数をそれぞれカウントし, 実際の項レベルの演算回数 (表 4) を調べる. 各サイ

表 4: Strassen-Winograd アルゴリズムの項レベル演算回数 (Case I)

Size	アルゴリズム	加算	乗算	合計	比
8	標準法	4608	12800	17408	
	Strassen-Winograd	5680	11200	16880	1.031
16	標準法	36864	102400	139264	
	Strassen-Winograd	39712	62400	102112	1.364
32	標準法	294912	819200	1114112	
	Strassen-Winograd	236848	302400	539248	2.066
64	標準法	2359296	6553600	8912896	
	Strassen-Winograd	1290160	1339200	2629360	3.340
128	標準法	18874368	52428800	71303168	
	Strassen-Winograd	6641776	5572800	12214576	5.838
256	標準法	150994944	419430400	570425344	
	Strassen-Winograd	32973392	22161600	55134992	10.35

ズの行列の乗算で, 旧アルゴリズムと Strassen-Winograd アルゴリズムの演算回数の比は, 8×8, 16×16, 32×32, 64×64, 128×128, 256×256 行列の順に, 1.031, 1.364, 2.066, 3.340, 5.838, 10.35 となる.

4.2 Case II の場合の演算回数

Case II の場合の, 古典的アルゴリズムと Strassen-Winograd アルゴリズムで $A\times B$ の演算を実行し, Case I の場合と同様に演算回数を計測 (表 5) し比較する. 各サイズの行列の乗算で, 標準的アルゴリズムと Strassen-Winograd アルゴリズムの計算量の比は, 8×8, 16×16, 32×32, 64×64, 128×128, 256×256 行列の順に, 0.9907, 1.266, 1.859, 2.965, 4.990, 8.693 となる.

表 5: Strassen-Winograd アルゴリズムの演算回数 (Case II)

Size	アルゴリズム	加算	乗算	合計	比
8	標準法	12800	12800	25600	
	Strassen-Winograd	14640	11200	25840	0.991
16	標準法	102400	102400	204800	
	Strassen-Winograd	99360	62400	161760	1.266
32	標準法	819200	819200	1638400	
	Strassen-Winograd	579120	302400	881520	1.859
64	標準法	6553600	6553600	13107200	
	Strassen-Winograd	3080880	1339200	4420080	2.965
128	標準法	52428800	52428800	104857600	
	Strassen-Winograd	15442800	5572800	21015600	4.990
256	標準法	419430400	419430400	838860800	
	Strassen-Winograd	74336080	22161600	96497680	8.693

以上 Case I, Case II の結果から, 行列の各要素の項レベルでの演算回数が演算時間の比に比較的近いものとなり, 項レベルの演算回数が演算の高速化に影響を与えるものと考えられる.

4.3 分割終了条件

Strassen-Winograd アルゴリズムは, 行列を 4 分割し再帰的に Strassen-Winograd アルゴリズムを呼び出し, 実行している. そこで, 各要素に Case I の場合の 128×128 行列で, Strassen-Winograd アルゴリズムから古典的アルゴリズムに切り替わるサイズを 64,32,16,8,4 と変化させた場合の演算速度と演算回数を比較し, 分割を終了するサイズを決定する. Strassen-Winograd アルゴリズムと旧アルゴリズムとの演算時間

表 6: 分割サイズによる計算量と演算速度の比較 (Case I)

Size	加算	乗算	合計	秒
	18874368	52428800	71303168	102.1
64	16936960	45875200	62812160	81.57
32	12605440	31948800	44554240	53.29
16	9015040	19353600	28368640	32.6
8	7089088	10713600	17802688	19.53
4	6641776	5572800	12214576	12.73

の比 (表 6) は 64×64, 32×32, 16×16, 8×8, 4×4 に分割サイズを変化させるにつれて, 0.88, 0.71, 0.64, 0.63, 0.686 となる. また, 演算回数の比 (図 3) は同様に, 0.80, 0.65, 0.61, 0.60, 0.65 となる.

演算回数は, 4×4 行列に分割するまで減少し続けており, また, 演算時間も減少し続けている. よって, 分割は 4×4 行列になるまで実行することとした.

5 まとめ

今回, これまで Risa/Asir に実装されていた, 行列の定義に基づく行列の乗算と Strassen-Winograd アルゴリズムの演算時間の比較を行った. これまでの計算量の理論値を越えた高速化が計られており, 直接の理由については今だ不明である. しかし, 行列の要素に含まれる項レベルの演算量の比較から, 多項式の項レベルの計算量が大きな要因となっていることが推測できる. 数式処理の場合では, 従来の計算量ではなく, メモリアクセス等の空間計算量のような新たな尺度を用いる必要があるのではないかと思われる.

参 考 文 献

[Bailey, 1988] Bailey, D. H. (1988). Extra high speed matrix multiplication on the Cray-2. *SIAM J. Sci. Stat. Comp.*, 9(3):603–607.

[Brent, 1970] Brent, R. P. (1970). Algorithms for matrix multiplication. Technical Report CS 157, Stanford University.

[Bunch and Hopcroft, 1974] Bunch, J. R. and Hopcroft, J. E. (1974). Triangular factorization and inversion by fast matrix multiplication. *Mathematics of Computation*, 28(125).

[Cohen and Roth, 1976] Cohen, J. and Roth, M. (1976). On the implementation of Strassen's fast multiplication algorithm. *Acta Informatica*, 6:341–355.

[Coppersmith and Winograd, 1990] Coppersmith, D. and Winograd, S. (1990). Matrix multiplication via airthmetic progressions. *Journal of Symbolic Computation*, 9:251–280.

[Cray SciLib, 1990] Cray SciLib (1990). *(UNICOS) Math and Scientific Library Reference Manual (SR-2081), Release 6.0.* Cray Research Inc.

[Douglas et al., 1994] Douglas, C. C., Heroux, M., Slishman, G., and Smith, R. M. (1994). GEMMW: A portable level 3 BLAS Winograd variant of Strassen's matrix matrix multiply algorithm. *Journal of Computational Physics*, 10:1–10.

[Higham, 1990] Higham, N. J. (1990). Exploiting fast matrix multiplication within the level-3 BLAS. *ACM Transactions on Mathematical Software*, 16:352–368.

[Huss-Lederman et al., 1996a] Huss-Lederman, S., Jacobson, E. M., Johnson, J. R., Tsao, A., and Turnbull, T. (1996a). Implementation of Strassen's algorithm for matrix multiplication. In *Supercomputing '96 Conference Proceedings.*

[Huss-Lederman et al., 1996b] Huss-Lederman, S., Jacobson, E. M., Johnson, J. R., Tsao, A., and Turnbull, T. (1996b). Strassen's algorithm for matrix multiplication: Modeling, analysis, and implementation. Technical Report CCS-TR-96-147, Center for Computing Sciences.

[IBM ESSL, 2000] IBM ESSL (–2000). *Engineering and Scientific Subroutine Library: Guide and Reference.* IBM.
`http://www.rs6000.ibm.com/resource/aix_resource/sp_books/essl/`.

[Pan, 1978] Pan, V. Y. (1978). Strassen algorithm is not optimal. trilinear technique of aggregating, uniting and canceling for constructing fast algorithms for matrix multiplication. In *Proc. 19th FOCS*, pages 166–176.

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

[Strassen, 1986] Strassen, V. (1986). The asymptotic spectrum of tensors and the exponent of matrix multiplication. In *Proc. 27th FOCS*, pages 49–54.

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