

# 行列和計算最適化による Strassen 法の高速化

坂本真貴人<sup>†</sup> 藤井昭宏<sup>‡</sup> 小西克巳<sup>‡</sup> 田中輝雄<sup>‡</sup>

工学院大学大学院情報学専攻<sup>†</sup> 工学院大学情報学部<sup>‡</sup>

## 1 はじめに

さまざまな科学技術計算の基本演算である行列積を高速に計算する手法として Strassen 法 [1] がある。これは行列積の計算量を削減する手法であり、再帰的に適用できる。再帰的に適用する回数 (以下, Strassen 段数) を増やすことで計算量をさらに削減できる。

Strassen 法は小行列の行列積を行う部分と行列和を行う部分に分かれている。現在主流のメモリが階層構造の計算機では、行列積の性能が演算律速であるのに対し、行列和の性能はメモリ律速である。したがって Strassen 法を並列化した場合、行列積と行列和の性能差が広がり、行列和部分の計算時間が全計算時間の多くを占めるようになる。

本研究では、共有メモリ並列環境において行列和部分を最適化し、Strassen 法を高速化する手法の提案を行う。

## 2 Strassen 法

$N$ 次正方行列  $A, B, C$  について、行列積  $C = A \times B$  を計算する際に、Strassen 法ではまず図 1 のように行列  $A, B, C$  を四分分割する。次に  $N/2 \times N/2$  行列である小行列  $T, S$  を作り、小行列  $Q$  を計算する。最後に  $Q$  を用いて行列  $C$  を計算する。Strassen 法

$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$ $B, C$ も同様		
$T_1 = A_{12} - A_{22}$	$S_1 = B_{21} + B_{22}$	$Q_1 = T_1 \cdot S_1$
$T_2 = A_{11} + A_{22}$	$S_2 = B_{11} + B_{22}$	$Q_2 = T_2 \cdot S_2$
$T_3 = A_{11} - A_{21}$	$S_3 = B_{11} + B_{12}$	$Q_3 = T_3 \cdot S_3$
$T_4 = A_{11} + A_{12}$	$S_4 = B_{22}$	$Q_4 = T_4 \cdot S_4$
$T_5 = A_{11}$	$S_5 = B_{12} - B_{22}$	$Q_5 = T_5 \cdot S_5$
$T_6 = A_{22}$	$S_6 = B_{21} - B_{11}$	$Q_6 = T_6 \cdot S_6$
$T_7 = A_{21} + A_{22}$	$S_7 = B_{11}$	$Q_7 = T_7 \cdot S_7$
$C_{11} = Q_1 + Q_2 - Q_4 + Q_6$	$C_{12} = Q_4 + Q_5$	
$C_{21} = Q_6 + Q_7$	$C_{22} = Q_2 - Q_3 + Q_5 - Q_7$	

図 1 Strassen 法

において、小行列  $Q$  の計算が行列積部分、小行列  $T, S$ 、行列  $C$  の計算が行列和部分となる。行列積部分に線形代数ライブラリ BLAS [2] のルーチンである DGEMM を用いると高速に計算できる。

通常の方法では、 $N/2 \times N/2$  行列の積が 8 回、和が 4 回である。一方、Strassen 法は積が 7 回、和が 18 回であり、積の回数が 1 回少ない。積の計算量は  $O(N^3)$ 、和は  $O(N^2)$  であるため、 $N$  が十分大きい場合、Strassen 法の方が計算量が少ない。また、小行列  $Q$  の計算に Strassen 法を再帰適用するとさらに計算量を削減できる。

## 3 行列和計算の最適化手法

Strassen 法の行列和部分は、メモリアクセス回数やメモリアクセス時間の削減が重要である。この章では、これらを削減する手法を提案する。

### 3.1 行列和をまとめて計算 (手法 1)

Strassen 法では、通常は図 2 左のように計算する。この場合、小行列  $T, S$  を 1 つずつ計算するため、行列  $A, B$  の同じ要素に対して複数回アクセスする必要がある。

そこで手法 1 では、図 2 右のように小行列  $T, S$  の行列和計算を 1 つのループ文内でまとめて計算する。それにより、行列  $A, B$  の各要素へのアクセスが 1 回で済む。

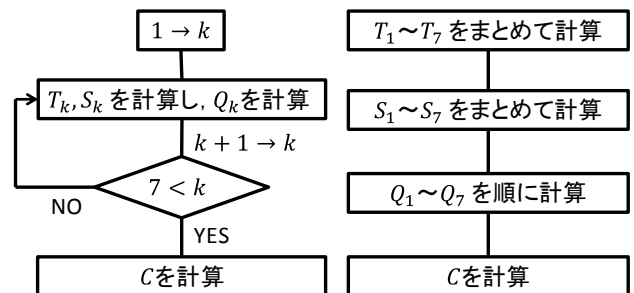


図 2 Strassen 法の計算手順 (左:通常的手法, 右:手法 1)

### 3.2 式の展開 (手法 2 A)

Strassen 段数が 2 段以上の場合、図 3 に示す  $T_{11}$  の式のように最下段の式を展開することで、行列  $A, B$  から最下段の小行列  $T, S$  (2 段の場合  $T_{11} \sim T_{77}, S_{11} \sim S_{77}$ ) を直接計算できる。それによ

Acceleration of Strassen Algorithm from tuning matrix addition

Makito Sakamoto<sup>†</sup>, Akihiro Fujii<sup>‡</sup>, Katsumi Konishi<sup>‡</sup>, and Teruo Tanaka<sup>†</sup>

<sup>†</sup>Major in Informatics, Kogakuin University

<sup>‡</sup>Faculty of Informatics, Kogakuin University

$$A = \begin{pmatrix} A'_{11} & A'_{12} \\ A'_{21} & A'_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{pmatrix}$$

$$T_1 = A'_{12} - A'_{22}$$

$$T_{11} = (T_1)_{12} - (T_1)_{22}$$

$$= (A'_{12} - A'_{22})_{12} - (A'_{12} - A'_{22})_{22}$$

$$= (A'_{12})_{12} - (A'_{22})_{12} - (A'_{12})_{22} + (A'_{22})_{22}$$

$$= A_{14} - A_{34} - A_{24} + A_{44}$$

図3 Strassen法の式の展開(2段)

り中間の小行列は不要になる。また、最下段の行列和を手法1のようにまとめて計算することで、行列A,Bへのメモリアクセスを最小化できる。

### 3.3 式の展開とループ分割(手法2B)

式を展開することで、理想的にはメモリアクセス回数は削減される。しかし実際は1つのループ文で読み込むデータが多いと、レジスタの退避、回復の発生によるメモリアクセス回数の増加や、キャッシュヒット率低下によるメモリアクセス時間の増加が起こる。

そこで、1つのループ文を複数に分割することでこれらを回避する。今回は、小行列T,Sをそれぞれ1度に7つずつ計算するように分割する。

## 4 数値実験

数値実験では、通常の方法(BLASのDGEMM)とStrassen法で行列積 $C = A \times B$ の性能を比較する。実験環境はCPUがIntel Core i7 3770K 3.5GHz(4コア)、BLASは高速なBLASであるOpenBLAS[3]を用いる。手法1の適用前と適用後の比較を図4に、手法1, 2A, 2Bをそれぞれ適用し、比較した結果を図5に示す。

図4を見ると、手法1を適用するとStrassen段数1段, 2段で共に性能が向上している。手法1適用前は1段の性能がDGEMMを上回る行列サイズが $N = 4500$ 程度であったのに対し、適用後は $N = 3500$ 程度まで小さくできた。

図5では、手法2Bを適用することで2段の

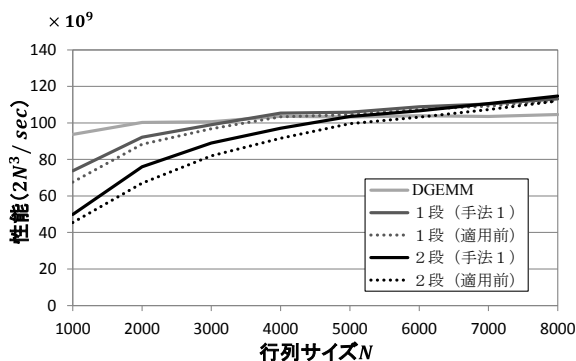


図4 手法1適用前と適用後の性能比較

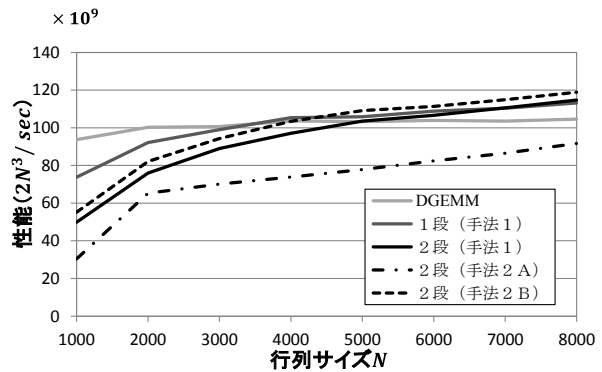


図5 手法1, 2A, 2Bを適用した性能

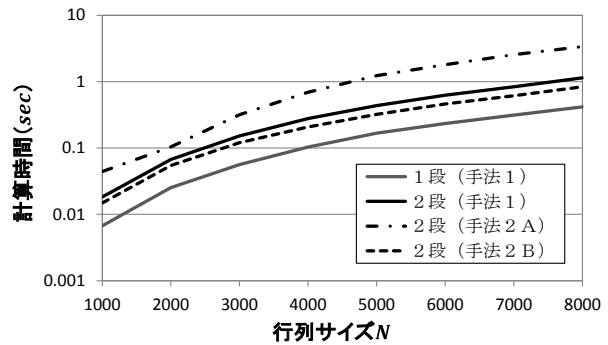


図6 行列和部分の計算時間

性能が手法1より向上している。2段の性能が1段を上回る行列サイズが、手法1では $N = 7500$ 程度であったのが $N = 4500$ 程度まで小さくできた。手法2Aを適用した2段は手法1より性能が低下していた。これは図6に示すように行列和部分の計算時間が増加したためである。式の展開のみではメモリアクセス回数やメモリアクセス時間が手法1より増加していると考えられる。

## 5 まとめ

本研究では、Strassen法における行列和計算を最適化する手法を提案し、全体を高速化した。

実験の結果、行列和計算をまとめることでStrassen段数1段の性能が通常の方法を上回る行列サイズ $N$ を約1000小さくし、式の展開とループ分割を適用することで2段の性能が1段を上回る $N$ を約3000小さくできた。

今後はより最適なループ分割の仕方やキャッシュサイズを考慮した最適化手法を見つけたい。

### 参考文献

[1] V. Strassen, Gaussian Elimination is not Optimal, Numer. Math., vol. 13, pp.354-356, Aug. 1969.  
 [2] BLAS (Basic Linear Algebra Subprograms), <http://www.netlib.org/blas/>.  
 [3] OpenBLAS Homepage, <http://www.openblas.net/>.