Strassenの行列積アルゴリズムのSX-3上でのインプリメント

3Q−1

竹内純一[1]，萬淳一[2]
[1]NEC C&C情報研究所，[2]NEC C&C汎用アプリケーション技術本部

1 はじめに

通常N次正方行列の積を計算するには，$2N^3$回の浮動小数点演算が必要となるが，1969年にV.Strassenは，分割統治法(divide and conqure)という方略を再帰的に適用することにより，$4.7N^{2.81}$回の浮動小数点演算で行列積が求められることを示した[1, 2, 3].

このアルゴリズムは計算量のオーダを下げるが，係数が増加するため行列のサイズがある程度大きくないと実際には計算量が減らない．また，アルゴリズムが自然なものに比べ複雑であり，誤差の点で不利であること等の理由から，ほとんど実用に供されることがなかった．しかしながら，近年計算機のハードウェアが発達し，大規模な行列の演算が実用的な時間で計算出来るようになるとともに，実用化される例が現れている．

こうしたことから，NECのスーパーコンピュータSX-3上でも，Starassenのアルゴリズムをインプリメントしてみた．一般に，SX-3のようなベクトル計算機の場合，行列のサイズが小さいとベクトル化の利点が活かしにくいため，演算速度が下がる．このため，行列を小行列に分割して計算するStrassenのアルゴリズムでは，必要な演算量を減らしても実行時間が減るとは限らない．

しかしながら，評価実験の結果，Strassenのアルゴリズムにより，行列積の演算時間が最大約15%削減されることが分かった．特に2048次の行列では，見かけ上約5.9 GFLOPSの演算速度を達成した．また，誤差の面でも実用上問題が無いことが分かった．

2 Strassenのアルゴリズム

ここでは，Strassenの行列積アルゴリズムを簡単に説明する．

A, Bを$N\times N$の正方行列とし，それぞれのij成分を，a_{ij}, b_{ij}と書く．$C = A\times B$を計算するには，通常，$c_{ij} = \sum_{k=1}^{N} a_{ik}b_{kj}$という定義式に素直に従ったアルゴリズムを用いる．この場合，1つの成分を求めるのにN回の積と，$N-1$回の和を計算することになり，$2N^3 - N^2$回の演算が必要になる．これに対し，Strassenが考案した方法は，次の様なものである．

Nが偶数である仮定する．このとき，A, Bをそれぞれ$\frac{N}{2}\times\frac{N}{2}$の小行列に分割することができる(次式)．

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \times \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \quad (1)$$

これらの小行列について，

$$M_1 = (A_{12} - A_{22})(B_{21} + B_{22}) \quad (2)$$
$$M_2 = (A_{11} + A_{22})(B_{11} + B_{22}) \quad (3)$$
$$M_3 = (A_{11} - A_{21})(B_{11} + B_{12}) \quad (4)$$
$$M_4 = (A_{11} - A_{12})B_{22} \quad (5)$$
$$M_5 = A_{11}(B_{12} - B_{22}) \quad (6)$$
$$M_6 = A_{22}(B_{21} - B_{22}) \quad (7)$$
$$M_7 = (A_{21} - A_{22})B_{11} \quad (8)$$
$$C_{11} = M_1 + M_2 - M_4 + M_6 \quad (9)$$
$$C_{12} = M_4 + M_5 \quad (10)$$
$$C_{21} = M_6 + M_7 \quad (11)$$
$$C_{22} = M_2 - M_3 - M_5 - M_7 \quad (12)$$

として，Cを求めることが出来る．これは，行列の演算が環の演算則則を満たすという条件だけから容易に確かめられる([1]参照)．

ここでは，$\frac{N}{2}\times\frac{N}{2}$正方行列について，7回の積と18回の和を計算している．和の演算が多いが，和の回数はNの2乗にしか比例しないため，行列のサイズが大きくなると問題にならなくなる．すなわち，この方法は自然なアルゴリズムに対してほぼ$\frac{7}{8}$に演算量を減らすことが出来るのである．なお，以下この計算法をStrassenの手続きと呼ぶことにする．

さらに，もしNが2の冪乗であるとすれば，$\frac{N}{2}$も偶数であるから，7つの小行列の積に関して上述の手続きを再び使うことが出来る．この手続きは小行列のサイズが$O(1)$になるまで繰り返すことが出来る(これをStrassenアルゴリズムと呼ぶことにする)．その回数は$O(\log N)$であるから，$(\frac{7}{8})^{\log N}$倍に演算量を減らせる．ここで，

$$N^3 \times \left(\frac{7}{8}\right)^{\log N} = N^3 \times \frac{N^{\log 7}}{N^{\log 8}} = N^{\log 7} \cong N^{2.81} \quad (13)$$

であるから，$O(N^{2.81})$の演算数で行列積を求められることがわかる．

Nが2の冪でないときは，奇数次の行列は一つ大きいサイズの行列に埋め込んでからStrassenの手続きを行うという手続き(一般化Strassenの手続きと呼ぶ)を再帰的に適用すればよい．この場合でもオーダーは同じである．

3 評価するマシンについて

SX-3は，パイプライン方式のベクトルプロセッサを最大4台擁するスーパーコンピュータである．SX-3のプロセッサは，1台で1クロックあたり，16回の浮動小数点演算を行える．また，マシンサイクルは2.9$nsec.$である．よって，1プロセッサでのピーク性能は5.5GFLOPSである．また，ベクトルレジスタの長さは256である．

実験に用いたマシンはモデル24で，2台のプロセッサを持つが，本実験は2台が独立に動くモードで行った．従って，CPUタイムについては，ほぼ1プロセッサのモデルと同じ性能を示す．

また，プログラム言語はFORTRANを用い，コンパイラによる自動ベクトル化を行った．

4 評価実験

4.1 演算速度の評価

はじめに，一度だけ一般化Strassenの手続きを用いるプログラムの評価をした．

N次行列の場合の演算数を$2N^3$と見積もり，これをCPUタイムで除したものをFLOPS*で表すとする．Strassenの手続き(1段)の場合，これに$\frac{7}{8}$をかけたものが実際のFLOPS値ににほぼ等しくなる．

表1は実験結果で，単位はMFLOPS*である．この結

Implementation of Strassen's Matrix-matrix Multiplication Algorithm on SX-3. Jun-ichi Takeuchi[1] and Jun-ichi Yorozu[2]. [1]C&C Information Technology Research Laboratories, NEC Corp., [2]C&C Application Software Engenering Division, NEC Corp..

size	Strassen(A)	Normal(B)	B/A
256	2467	4021	1.63
512	4496	4583	1.01
768	4184	4766	1.14
1024	5173	4858	0.94
1280	4765	4915	1.03
1536	5401	4932	0.91
1792	5058	4969	0.98
2048	5526	4990	0.90

表 1: 一般化 Strassen の手続き (1 段)

果によると，768 次以下のサイズの行列には，Strassen の手続きを用いても効果がないことがわかる．より詳しい実験によると，769 次以上の行列には効果がある．また，1280 次付近にも効果のない場合がある．また，サイズが 256 の偶数倍の場合に効果が大きく，奇数倍ではあまり効果がないことが見てとれる．

これは，SX-3 のベクトルレジスタの大きさが 256 であることによる．つまり，SX-3 ではベクトル長が 256 のベクトル演算まで，1 つのベクトル命令で行えるため，ベクトル長が 256 の整数倍になるときに，最も効率が良い．ところが，256 の奇数倍のサイズの行列に Strassen の手続きを用いると，半端なサイズの行列の積を通常のアルゴリズムで計算することになり，ベクトル化の効率が悪くなるのである．しかし例えば 769 次といった，もともと半端なサイズの行列の場合は，通常アルゴリズムもベクトル化効率が悪くなっているため，Strassen の手続きは効果がある．

以上の結果から，1536 次以上の行列には 2 段の Strassen が，768 次以上の行列には 1 段の Strassen が有効であると考えられ，その様に動作するプログラムを作成し，計測した．結果は表 2の通りである． 1280 次付近

size	Strassen(A)	Normal(B)	B/A
512	4588	4581	1.00
768	4756	4756	1.00
1024	5169	4849	0.94
1280	4746	4862	1.02
1536	5404	4943	0.91
1792	5407	4967	0.92
2048	5883	4971	0.84

表 2: Strassen アルゴリズム (最大 2 段)

では，若干 Normal に劣る場合があるが，ほぼ数 % から 15% 高速に演算を実行することが分かった．特に，2 段の Strassen の手続きを行う 2048 次の場合は，見かけ上 SX-3 のピーク性能を越える 5.9GFLOPS* を達成した．

4.2 演算誤差の評価

Strassen のアルゴリズムと，通常のアルゴリズムの丸め誤差の比較を行った．

2048 次の行列積について，通常のアルゴリズム (4 倍精度)，通常のアルゴリズム，1 段の Strassen，2 段の Strassen，3 段の Strassen に，同一のデータを入力して結果を比較した．$C = A \times B$ としたとき，次のデータを用いた．

$$a_{ij} = \begin{cases} \frac{\sin j}{\sqrt{j}}, & j \le \frac{N}{2} \\ -\frac{\sin(N-j+1)}{\sqrt{N-j+1}}, & \text{otherwise.} \end{cases} \quad (14)$$

$$b_{ij} = \begin{cases} \sqrt{i}\log i, & i \le \frac{N}{2} \\ \sqrt{N-i+1}\log(N-i+1), & \text{otherwise.} \end{cases} \quad (15)$$

ここで，$N = 2048$ である．容易にわかるように，$c_{ij} = 0$ が成り立つ．また，$a_{ik} \cdot b_{kj}$ の絶対値は $O(1)$ である．すなわち，このデータは比較的絶対値が大きい数値を加えて結果が 0 になるため，誤差が出やすいデータと言える．とくに，Strassen のアルゴリズムは通常のアルゴリズムに較べて和の量が増えるので不利である．

実験では，データは 4 倍精度で作成したのち，倍精度に型変換したものを用いている．結果は表 3の様になった．最も絶対値が大きかった成分の絶対値を示している．

4 倍精度	Normal	1 段	2 段	3 段
6.5×10^{-33}	1.6×10^{-15}	9.3×10^{-13}	5.5×10^{-12}	7.3×10^{-12}

表 3: 誤差の比較

4 倍精度の結果から，このデータは実際に結果が 0 になるべきものであることがわかる．したがって，Normal の結果に対し 1 段の結果は 3 桁ほど悪い．しかし，1 段と 2 段の比較と，2 段と 3 段の比較では 1 桁は変わらない．これは，データが行列 A の左半分で正の値をとり，右半分で負の値をとることと関係している．Strassen の手続きは行列を 4 つの小行列に分割するため，このような結果が出ているのである．したがって，自然なデータの場合には 3 桁もの違いは出ないと思ってよかろう．

5 おわりに

SX-3 上で Strassen の行列積アルゴリズムをインプリメントし，計算速度および計算誤差を計測し，評価を行った．その結果，ベクトル計算については従来に対し計算時間を最大約 15% 削減することが分かった．また，計算誤差についても，通常アルゴリズムにくらべ若干精度が落ちるものの，実用上問題ない程度であることが分かった．

今後の課題としては，マルチタスキング処理を用いるプログラムの最適化，およびその評価実験を行うことが挙げられよう．

謝辞

本実験を行うにあたりお世話頂いた方々や，本稿作成につき貴重なコメントを下さった方々，特に，NEC の津和義昭氏，小久保達信氏，大野和彦氏，安倍直樹氏に深謝します．

参考文献

[1] Aho A.V.,Hopcroft J.E., and Ulman J.D.(1974) *The Design and Analysis of Computer Algorithm.* 230-232. Addison-Wesley.

[2] Wilf H.S.(1986) *Algorithms and Complexity.* Prentice-Hall.

[3] 野崎昭弘.(1987) アルゴリズムと計算量 (計算機科学 / ソフトウェア技術講座 5). 共立出版.