效果的인 複素매트릭스 곱셈 알고리즘에 關한 硏究

金 斗 京*

A Study on the Effective Complex Matrix Multiplication Algorithm

*Kim, Doo-gyung**

Summary

In digital processing, inner products, vector-scalar and vector-matrix multiplication are sometimes encounted with complex entries. A well-known algorithm for complex multiplication which requires three real multiplications and five real additions is observed.

This extends its applicability to complex matrices. The computational cost function is based on the number of equivalent real additions, with every real multiplication as equivalent to r real additions.

As examined in this paper, the computational savings are shown to approach 1/4.

序 論

알고리즘(algorithm)은 一般的으로 所要 計算時間(complexity), 記憶場所 使用量, 正確性, 簡潔性을 重要視한다. 특히 수 많은 計算을 要求하는 매트릭스 곱셈(matrix multiplication)인 경우는 所要 計算時間을 計算하는 것이 가장 重要한 要素가 된다. 따라서 알고리즘을 選擇할 때 效率性을 簡潔性보다 重要하게 考慮해야 한다(강맹규, 1988).

複素數 x와 y의 곱에서 x=a+jb, y=c+jd일 때 xy=(a+jb)(c+jd)=(ac-bd)+j(ad+bc)이므로 ac-bd와 ad+bc의 計算을 必要로 한다. 만일 直接的으로 計算한다면 이것은 네번의 실곱셈(real multiplication)과 두번의 실덧셈(real addition)을 必要로 한다. Golub에 의하여 (1)식으로 使用될 수 있다는 것이 알려져 있다.

$$
\begin{aligned}
xy &= (ac-bd)+j(ad+bc) = \\
&(a(c-d)+(a-b)d)+j(b(c+d)+(a-b)) \cdots\cdots (1)
\end{aligned}
$$

이것은 세번의 실곱셈과 다섯번의 실덧셈을 必要로 한다. 이 恒等式은 (1)식에서 Winograd에 의해서 세련되게 論議됨으로서 多項式 곱셈의 效率的인 알고리즘 應用은 特別한 경우에 利用될 수 있다. 一般的으로 덧셈보다 곱셈에 計算時間이 더 所要되므로 곱셈을 적게하고 덧셈이 많은 알고리

* 電子計算所

즘이 더 效率的이다(Adlyt, 1988).

실곱셈이 r번 실덧셈과 計算的으로 同等하다고 놓자 (1)식의 應用에서는 r>3일 때 重要하다. Moharir에 의해서 된 것처럼 分散된 計算과 個別的인 VLSI Chip들에서 利用할 수 있는 向上된 計算 能力의 出現에 의해서 r의 값은 어떤 경우에 일정한 값에 接近한다. 이것은 計算 費用에서 重要한 要素가 I/O(input/output)와 데이터 操作의 應用 分野이다(Moharir, 1985).

특히 곱셈이 덧셈보다 費用이 많이 드는 分野는 매트릭스 演算이다.nxn 실매트릭스에서 곱셈은 O(n^3)計算을 必要로 하고 반면에 덧셈은 O(n^2)만이 必要하다. (1)식은 복소매트릭스에도 應用할 수 있다(Kung, 1979).

$$xy=((a-b)c+b(c-d))+j((a+b)d+b(c-d)) \quad \cdots\cdots (2)$$

直接 計算에 의한 方法보다는 (1)식과 (2)식에 의하여 變形된 計算이 훨씬 所要 計算時間을 적게 必要로 한다.

따라서 본 硏究에서는 複素매트릭스 곱셈의 處理時間을 줄일 수 있는 알고리즘을 表現하고자 한다.

硏 究 方 法

1. 알고리즘 分析의 必要性

一般的으로 알고리즘의 評價 基準은 所要 計算時間(computation Time), 記憶容量 使用量(memory space usage), 正確性(correctness), 簡潔性(simplicity)가 基本 要素이다. 實際的인 處理에서는 所要 計算時間과 記憶容量 使用量을 重要視하는 데 그 理由는 많은 使用者가 컴퓨터를 同時에 使用하게 되면 記憶容量이 不足하고 計算時間이 오래 걸리게 된다.

따라서 컴퓨터로 알고리즘을 遂行하기 전에 미리 손으로 알고리즘을 遂行하여(hand simulation) 豫測하는 것이 바람직하다. 또한 이러한 分析을 통하여 프로그램의 어느 部分이 가장 많은 計算時間과 記憶場所를 必要로 하는 지를 判斷하여 效率的인 알고리즘을 選擇하여 使用하는 것이 必要하다. 그러나 알고리즘의 比較는 相對的인 效率性을 가지고 論議할 수 밖에 없다. 所要 計算時間의 測定은 컴퓨터 機種, 프로그램 言語, 프로그램 作成者에 의하여 좌우되지 않는 基本 演算의 回數를 基準으로 하여 推定値를 算定하는 것이 普通이다. 알고리즘의 簡潔性은 알고리즘의 正確性을 쉽게 證明할 수 있고 프로그램의 作成, 修正 作業을 빠르게 할 수 있다.

그러자 자주 反復 使用되는 알고리즘이라면 所要 計算時間을 적게 使用하는 알고리즘의 效率性을 簡潔性보다 重要視하여 새로운 알고리즘의 發見을 追求해야 한다.

2. 正方形 매트릭스에서 nxn 곱셈 매트릭스

매트릭스 a, b을 곱하여 그 結果를 c에 저장하는 경우 알고리즘은 다음과 같이 表現된다.

```
Repeat for I=1, N
    Repeat for J=1, N
        Sum=0
        Repeat for K=1, N
        Sum=Sum+A(I, K) * B(K,J)

        End Repeat
        C(I, J)=Sum
    End Repeat
End Repeat
```

I에 N번 指定 演算 J와 C에 N^2번 指定 演算 K에 N^3번의 指定 演算 그리고 Sum에 N^2+N^3의 指定 演算이 遂行된다. 따라서 總 指定 演算은 $N+3N^2+2N^3$이다. 이 경우 complexity는 $n+3n^2+2n^3$이다.

(1)식과 (2)식의 x와 y를 nxn 複素매트릭스라 하면 x와 y의 直接 곱셈은 A번의 실덧셈과 M번의 실곱셈을 必要로 한다.

$A=2n^2+4(3n^3+3n^2+n)=8n^3+14n^2+4n$
$M=8n^3$ ……………………(3)

다른 한편 (1)식과 (2)식을 使用하면

$A'=5n^2+3(2n^3+3n^2+n)=6n^3+14n^2+3n$
$M'=6n^3$ ………………(4)

直接 計算에 의해서 同等한 덧셈에서 xy를 計算한 費用은 다음과 같다.

$C=8n^3(1+r)+14n^2+4n$ ……… (5)

(1)식과 (2)식을 使用하여 計算하면 다음과 같다.

두 식을 比較하면

$S=(C-C')/C$
$=(2n^2(1+r)+n)/(8n^3(1+r)+14n^2+4n)$
$=(n(1+r)/(4n^2(1+r)+7n+2)$ … (7)

(7)식은 (1)식과 (2)식이 複素 正方形 매트릭스에 使用될 때 計算 費用이 相對的으로 減少하는 것을 나타낸다.

$R=1-S=C'/C=(6n^3(1+r)+14n^2+3n)/$
$(8n^3(1+r)+14n^2+4n)$ ………… (8)

R은 두 費用의 比率을 나타낸다. n가 크다면 $2n^2$으로 나누면

$R=(3n(1-r)+14)/(4n(1+r)+14)$ …… (9)

$n(1+r)$가 增加하면 R은 3/4에 接近한다.

3. 一般的인 複素매트릭스의 경우

複素매트릭스 x는 pxn차원이고 y는 nxm의 一般的인 경우 直接 計算에 의하면

$A=2pm+4(2pmn+3pm+p)=8pmn+14pm+4p$
$M=8pmn$ ………………………… (10)

非正方形 매트릭스의 경우 (1)식과 (2)식은 다른 結果를 가져온다. (1)식을 使用하면

$A'=2pn+mn+2pm+3(2pmn+3pm+p)$
$=6pmn+11pm+2pn+mn+3p$ ……… (11)
$M'=6pmn$ ………………… (12)

(2)식을 使用하면

$A''=pn+2mn+2pm+3(2pmn+3pm+p)$
$=6pmn+11pm+2pn+mn+3p$ ……(13)

$M''=6pmn$ ………………………… (14)

(11)式에서 A′의 表現式은 하나의 pn이 (13)식에서 하나의 mn으로 代置된다.

그러므로 (1)식은 p<m인 매트릭스 짝에 대해서 使用되어야 하며 (2)式은 p>m인 경우에 使用해야 한다.

直接 計算法에서
$C=8pmn(1+r)+14pm+4p$ …… (15)

(1)식과 (2)식을 適當히 選擇하면

$C'=6pmn(1+r)+11pm+mn+pn+3p+n(\min(p,m))$ … (16)

(15)식과 (16)식으로 부터

$R=C'/C=(6pmn(1+r)+11pm+mn+pn+3p+n(\min(p,m)))/(8pmn(1+r)+14pm+4p)$
………………………… (17)

$pmn(1+r)$를 增加시키면 (17)식은 3/4에 接近한다.

結果 및 考察

(9) 식에서 $R=(3n(1+r)+14)/(4n(1+r)+14)$ 이므로 r과 n값에 따라서 R의 값을 Fortran Program으로 구하면 Table.1과 같이 計算된다.

Table 1. R as a function of r and n.

r \ n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	.91	.87	.84	.83	.81	.81	.80	.79	.79	.79	.78	.78	.78	.78	.78
2	.88	.84	.82	.81	.80	.79	.79	.78	.78	.78	.77	.77	.77	.77	.77
3	.87	.83	.81	.79	.79	.78	.78	.77	.77	.77	.77	.77	.77	.76	.76
4	.85	.81	.80	.79	.78	.78	.77	.77	.77	.77	.76	.76	.76	.76	.76
5	.84	.81	.79	.78	.78	.77	.77	.77	.77	.76	.76	.76	.76	.76	.76
6	.83	.80	.79	.78	.77	.77	.77	.76	.76	.76	.76	.76	.76	.76	.76
7	.83	.79	.78	.77	.77	.77	.76	.76	.76	.76	.76	.76	.76	.76	.76
8	.82	.79	.78	.77	.77	.77	.76	.76	.76	.76	.76	.76	.76	.76	.76
9	.81	.79	.78	.77	.77	.76	.76	.76	.76	.76	.76	.76	.76	.76	.76
10	.81	.78	.77	.77	.76	.76	.76	.76	.76	.76	.76	.76	.76	.76	.76

따라서 r과 n값이 1일 때도 R의 값은 0.91이며 r=10, n=15이면 R=0.76이 되는 것을 알 수 있다. r과 n값이 상당히 커지면 0.75에 收斂하는 것을 알 수 있다. 즉 매트릭스 곱셈에서 處理時間을 25% 줄일 수 있다는 것을 意味한다.

摘　　要

디지탈 信號處理(digital signal processing)에서 내적, 벡터-스칼라 및 벡터-매트릭스 곱셈에서 複素項(complex entry)들이 存在할 때 計算節約을 25% 할 수 있는 알고리즘은 computer 處理에서 매우 重要한 意味를 갖는다. 이 硏究를 다상 filter, filter bank, radar, 通信 應用分野 및 디지탈 信號處理에서 效果的인 알고리즘과 結合시키면 計算時間을 節約할 수 있으므로 效用性이 增大될 수 있다.

특히 離散 푸리에變換(discrete fourier transform)에서 複素項의 對稱性을 利用하여 複素

項을 줄이면 演算處理하는 시스토릭 어레이프로세서(systolic array processor)의 設計 費用을 節減시키는 데 이 硏究를 使用할 수 있다.

參 考 文 獻

Adlyt F. July 1988. Efficient Complex Matrix Multiplication. IEEE Trans Computer, Vol 37, No 7, 877-879.

Agarwal R.C., and Cooley J.W. 1977. New Algorithm for Digital Convolution. ASSP-25 IEEE, 392-410.

Cooley J.W., and Turkey J.W. 1965. An Algorithm for the Machine Calculation of Complex Fourier Transfrom. Math., Computer, 297-301.

강맹규. 1988. 데이터 構造論. 홍릉 科學出版社, 2-40.

Kung H.T. Jan. 1979. Let's Design Algorithms for VLSI Systems. Proc., of Conference On VLSI, 65-90.

Kung H.T. Jan. 1982. Why Systolic Architecture? IEEE Computer, 65-90.

Lin W.T., and Hwang J.P. Feb. 1988. A High Speed Shuffle Bus for VLSI Arrays. IEEE Jounal of Solid-State Circuit.

Moharir P.S. May 1985. Extending the Scope of Golub's Method beyond complex Multiplication. IEEE Trans Computer Vol C-34, 484-487.

Oppenheim A.V., and Schafer R.W. 1980. Digital Signal Processing. Prentice-Hall Co.

Strassen V. 1969. Gaussian Elimination is not Optimal. Numer., Math., 354-3356.

Winograd S. 1980. Arithmetic Complexity of Computations. Philadelphia. PA Soc., for Indust., Appl Math., SIAM.

Yeh H.G. 1985. Processor Elements and Systolic Arrays. Conference Record of the 19th Asilomar Con. on Circuit. Systems and Computer.