1－17

スーパーコンピュータSXシステムにおけるFFTプログラムの高速化

花村光泰* 萬 淳一*
増田典雄** 津和義昭* 渡辺 貞*

Vectorizing the FFTs on Supercomputer SX System

by

Mitsuyasu HANAMURA, Junichi YOROZU,
Yoshiaki TSUWA, Tadashi WATANABE
NEC Corporation
Norio MASUDA
NEC Scientific Information System Development Ltd.

ABSTRACT

Fast Fourier Transform (FFT) has been widely applied to various fields covering science, technology and social science. It's used not only in the spectral analysis of acoustic signals, seismic waves and so on, but also in boundary value problems of PDE occuring in numerical fluid dynamics (for example, as a direct method of solving a Poisson type PDE (RES method, etc.)).

Since the appearance of supercomputers, research of FFT algorithms available for supercomputers has increased rapidly. We have investigated the one-dimensional and two-dimensional FFT algorithms available for the supercomputer SX system. In this paper, we present the algorithms and calculated CPU-time of two-dimensional FFTs. We obtained a time of 9.41 msec in calculating two-dimensional FFT (256 x 256) at single precision using the SX-2.

1. はじめに

高速フーリエ変換（Fast Fourier Transform, FFT）は，数値流体力学において重要な偏微分方程式の境界値問題の解析[1]（例えば，ポアソン方程式の直接解法（RES法[2]など）），音声信号や地震波，脳波，経済変動データなどの各種の信号のスペクトル解析等，理学，工学，社会科学の各分野で利用されている。一方，スーパーコンピュータが本格的に稼働して以来，そのH/W性能を最大限に引き出すFFTのアルゴリズムの研究も盛んである。我々は，スーパーコンピュータSXシステムに最適な1次元，2次元FFTプログラムの研究を行って来た[5]。本論文では，2次元FFTを中心に，そのアルゴリズムとSX-2での実測結果を報告する。

＊日本電気㈱
＊＊日本電気技術情報システム開発㈱

2. 2次元複素フーリエ変換の3つのアルゴリズム

2次元複素フーリエ変換は，次式に相当した変換を行う。

順変換

$$C_{t,j}=\frac{1}{NM}\sum_{k=1}^{N}\sum_{s=1}^{M}c_{s,k}W_N^{(j-1)(k-1)}W_M^{(t-1)(s-1)}$$

式(1)

逆変換

$$c_{t,j}=\sum_{k=1}^{N}\sum_{s=1}^{M}C_{s,k}W_N^{-(j-1)(k-1)}W_M^{-(t-1)(s-1)}$$

式(2)

ここで，$j=1,\cdots N$，$t=1,\cdots M$，また，回転因子，W_N, W_Mは，以下の式で示される（iは，虚数単位）。

$$W_N=\exp\left(-\frac{2\pi i}{N}\right),\ W_M=\exp\left(-\frac{2\pi i}{M}\right)$$

式(1)，(2)において，$c_{s,k}$は，M行N列の入力行列であり，$C_{t,j}$はM行N列のフーリエ係数行列である。

上式をベクトルマシンで並列処理する方法としては，以下の3つの方法が考えられる。

(1)　アルゴリズムA

1次元複素フーリエ変換の多重性に注目し，1次元複素フーリエ変換を同時に処理する方法である。順変換を例にして示す。式(1)を，以下のように変形する。

$$C_{t,j}=\frac{1}{NM}\sum_{k=1}^{N}W_N^{(j-1)(k-1)}\sum_{s=1}^{M}c_{s,k}W_M^{(t-1)(s-1)}$$

式(3)

この式は，M項の1次元複素フーリエ変換をN回実行し，その結果得られた2次元データを入力としてN項の1次元複素フーリエ変換をM回実行することを意味している。このことを概念的に示したものが，図1である。この図1において，左側は，M行N列の入力行列$c_{s,k}$の列方向がM項1次元フーリエ変換を示しており，それを行方向へN回実行する様子を示している一方，右側は，左側の実行で得られた行列に対して，N項1次元フーリエ変換をM回実行することを示している。図2に，プログラムの中心部分を示す。

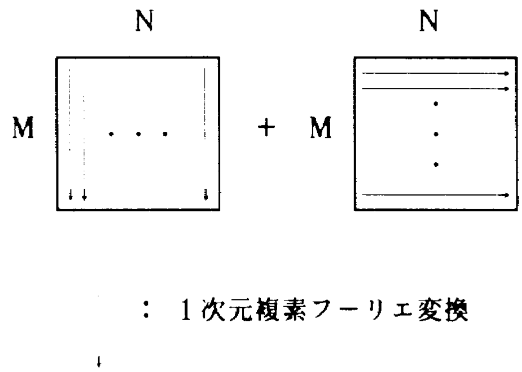


図1　アルゴリズムAの概念図

```
      ISTEP = ALOG2( REAL(N) ) + 0.5
      NN = N/2
      IA = 1
      IB = 1+NN
      JA = 1
      DO 5 IX = 1,M
         DO 1 IS = 1,ISTEP
            LA = 2**(IS-1)
            JB = 1+LA
            I = 0
            J = 0
            DO 20 K = 1,NN,LA
               DO 10 L = 1,LA
                  A2R(IX,JA+J) = AR(IX,IB+I) + AR(IX,IA+I)
                  XR           = AR(IX,IA+I) - AR(IX,IB+I)
                  A2I(IX,JA+J) = AI(IX,IB+I) + AI(IX,IA+I)
                  XI           = AI(IX,IA+I) - AI(IX,IB+I)
                  A2R(IX,JB+J) = WR(K)*XR    - WI(K)*XI
                  A2I(IX,JB+J) = WI(K)*XR    + WR(K)*XI
                  I = I+1
                  J = J+1
   10          CONTINUE
               J = J+LA
   20       CONTINUE
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    1    CONTINUE
    5 CONTINUE
```

図2　アルゴリズムAのプログラムリスト

このプログラムは，図1の右側の処理に対応している。DO 1ループ内が，N項1次元複素フーリエ変換に相当し，基数2の高速フーリエ変換（FFTと略す）を行っている。DO 5ループは，N項1次元FFTをM回実行する部分である。また変数ISTEP N項1次元FFTの段数である。このアルゴリズムAの性能は，1次元FFTの性能に左右されることになる。

(2)　アルゴリズムB[3)]

1次元複素フーリエ変換に共通する部分に注目し，それに並列処理を行う方法である。すなわち，式(3)において，M項の1次元複素フーリエ変換をN回実行する代わりに，N個の共通する部分を順次行うことである。このことを概念的に示したものが図3である。この図の左側は，N個のM項1次元複素フーリエ変換の内，N個の共通する行方向処理を順次列方向に行うことを示している。一方，右側では，

左側の実行で得られた行列に対して，M個の共通する列方向の処理を順次行方向へ繰り返す事を示している。図4にプログラムの中心部分を示す。このプログラムは，図3の右側の処理に対応している。

アルゴリズムA（図2）では最外側DOループであったDO 5ループが，アルゴリズムBでは，最深DOループになっている。すなわちN項1次元FFT（入力行列の行方向または入力列配の第2次元目の添字に対応する方向）に直行する方向（入力行列の列方向または入力行列の第1次元目の添字に対応する方向）をベクトル化のループ（すなわち最深DOループ）とする。これにより，アルゴリズムBの最深DOループ長は各次元のデータ長（MまたはN）になる。更に，1次元FFTの欠点であるメモリアクセス時のバンクコンフリクトが回避可能となる。

(3) アルゴリズムC[4)]

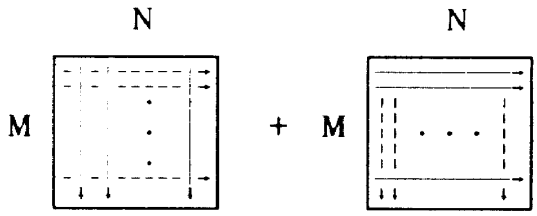
1次元複素フーリエ変換の多重性と処理に共通する部分の両方に注目し，それに並列処理を行う方法である。すなわち，アルゴリズムAとBを融合したものである。図5にプログラムの中心部分を示す。

このプログラムは，N項1次元FFTをM回実行する処理（図3の概念図の右側の処理）に対応している。また，アルゴリズムB（図4）のDO 5ループとDO 10ループを融合（すなわち，2重DOループの1重DOループ化）し，DO 5ループを形成している。これにより，最深DOループの平均ループ長は，各次元のデータ長に（ISTEP/2）をかけた長さになる。

3. 3つのアルゴリズムにおける最深DOループの平均DOループ長

3つのアルゴリズムの性能を比較するために，各々の最深DOループの平均DOループ長を計算し，表1に示す。

一般に，ベクトルマシンにおいて，平均DOループ長が長い程，高速性能が得られる。表1では，各次元でデータ長が等しい場合（$N=M$）と，異なる場合（$N=128$に固定）について示した。$N=M$の場合，アルゴリズムCは，小さなデータ長に対して，長い平均DOループ長となり，他のアルゴリズムより優れた性能が予測される。一方，$N\neq M$の場合，特にその傾向が強く，$M=32$で平均DOループ長が200を越えており，$N=M$の場合より，アルゴリズムCが優利であると考えられる。これらの傾向は，$N=128$以外についても同様である。また，注1に，計算に用いた1次元FFTのアルゴリズム（これら



：1次元複素フーリエ変換の方向

——→：最深DOループの方向

図3 アルゴリズムBの概念図

```
      ISTEP = ALOG2( REAL(N) ) + 0.5
      NN = N/2
      IA = 1
      IB = 1+NN
      JA = 1
      DO 1 IS = 1,ISTEP
        LA = 2**(IS-1)
        JB = 1+LA
        I = 0
        J = 0
        DO 20 K = 1,NN,LA
          DO 10 L = 1,LA
            DO 5 IX = I,M
              A2R(IX,JA+J) = AR(IX,IB+I) + AR(IX,IA+I)
              XR           = AR(IX,IA+I) - AR(IX,IB+I)
              A2I(IX,JA+J) = AI(IX,IB+I) + AI(IX,IA+I)
              XI           = AI(IX,IA+I) - AI(IX,IB+I)
              A2R(IX,JB+J) = WR(K)*XR    - WI(K)*XI
              A2I(IX,JB+J) = WI(K)*XR    + WR(K)*XI
    5       CONTINUE
            I = I+1
            J = J+1
   10     CONTINUE
          J = J+LA
   20   CONTINUE
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    1 CONTINUE
```

図4 アルゴリズムBのプログラムリスト

```
      ISTEP = ALOG2( REAL(N) ) + 0.5
      NN = N/2
      DO 1 IS = 1,ISTEP
        LA = 2**(IS-1)
        JB = 1+LA
        LAM = LA*M
        J = 0
        DO 20 K = 1,NN,LA
          JAI = J*LAM
          JBI = JAI+M*NN
          JAJ = 2*JAI
          JBJ = LAM+JAJ
          J   = J+1
          DO 5 IR = 1,LAM
            A2R(IR+JAJ) = AR(IR+JBI) + AR(IR+JAI)
            XR          = AR(IR+JAI) - AR(IR+JBI)
            A2I(IR+JAJ) = AI(IR+JBI) + AI(IR+JAI)
            XI          = AI(IR+JAI) - AI(IR+JBI)
            A2R(IR+JBJ) = WR(K)*XR   - WI(K)*XI
            A2I(IR+JBJ) = WI(K)*XR   + WR(K)*XI
    5     CONTINUE
   20   CONTINUE
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    1 CONTINUE
```

図5 アルゴリズムCのプログラムリスト

表1．3つのアルゴリズムの平均DOループ長

データ長 M	N=Mの場合			N=128の場合		
	アルゴリズム A	アルゴリズム B	アルゴリズム C	アルゴリズム A	アルゴリズム B	アルゴリズム C
4	2	4	4	33	66	71
8	4	8	12	34	68	110
16	8	16	32	36	72	156
32	16	32	80	40	80	216
64	32	64	192	48	96	304
128	64	128	448	64	128	448
256	128	256	1024	96	192	704
512	256	512	2304	160	320	1184
1024	512	1024	5120	288	576	2112

(注1) FFTのアルゴリズムは、Pease+DIF（アルゴリズムA）、Stockham+DIF（アルゴリズムB）、Stockham+DIF（アルゴリズムC）である。

(注2) 計算式
アルゴリズムA…（N+M）／4
アルゴリズムB…（N+M）／2
アルゴリズムC…（Nlog2M+Mlog2N）／4

のアルゴリズムについては，参考文献[5]を参照）を，注2に，計算に用いた式を示す。

4. SX-2による実測結果

図6および図7に，3つのアルゴリズムのSX-2による実測結果を示す。

図6は，各次元でデータ長が等しい場合（$N=M$）を，図7は，各次元でデータ長が異なる場合（$N=128$に固定）を示している。両図ともに，単精度計算であり，単位はMFLOPSである。前項で述べた平均DOループ長の予測通り，図6において，全データ長の範囲でアルゴリズムAは最も遅く，一方，データ長が128以下では，アルゴリズムCが最高速となっている。また，256以上で，アルゴリズムBとCの処理時間は同程度となる。SX-2では，演算およびデータアクセスが256要素単位（すなわち，SX-2のベクトルレジスタの最大要素数）で実行されるので，256の整数倍のデータ長においては，類似の処理（アルゴリズムBとCの中心部分の演算およびデータアクセス形式は同じ）が整数倍くり返されるだけとなり，アルゴリズムBとCの処理時間は殆んど同じとなる。

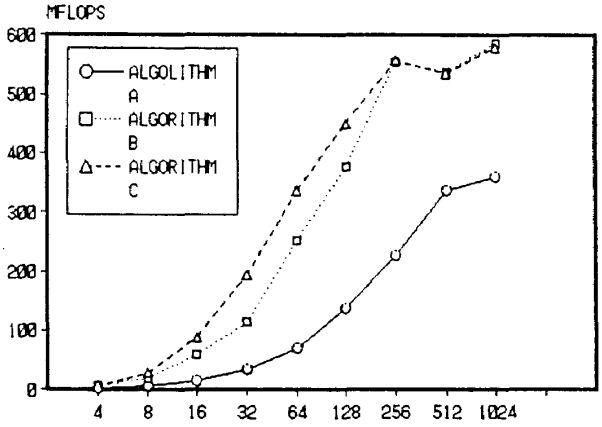


図6　3つのアルゴリズムのMFLOPS値（$N=M$）

(注1) 初期化の処理時間を含まない。
(注2) 正規化の処理時間を含まない。
(注3) MFLOPSの計算は、以下の式で行った。

M*M*(log2M/2)*2*10
/CPUtime(μsec)

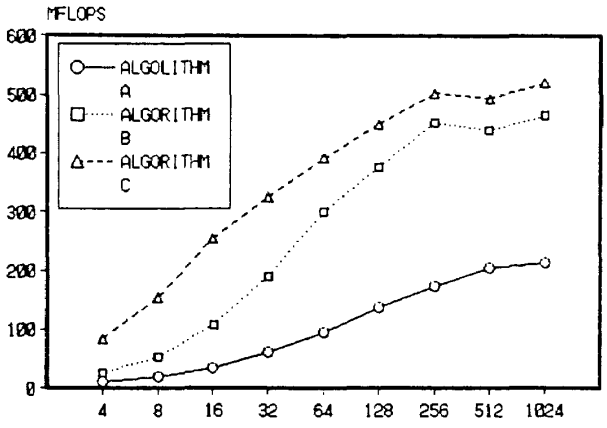


図7　3つのアルゴリズムのMFLOPS値（$N=128$に固定）

(注1) 初期化の処理時間を含まない。
(注2) 正規化の処理時間を含まない。
(注3) MFLOPSの計算は、以下の式で行った。

(N*M*(log2M/2)
+M*N*(log2N/2))*10
/CPUtime(μsec)

一方，図7において，各次元でデータ長が異なる場合，全データ長の範囲でアルゴリズムAが最低速となり，アルゴリズムCが最高速となる。特に，Mが小さい場合にその効果が大きいことがわかる。また，$N=128$以外に変化させた場合でも，比率は異なるが，同様の傾向を示すことがわかっている。

5. 1次元複素フーリエ変換との結合アルゴリズム

各次元のデータ長が極端に異なる場合について，

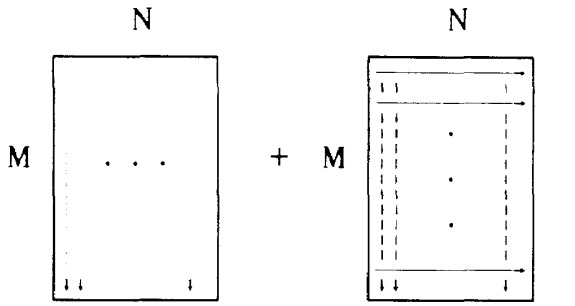


図8　アルゴリズムAとCの結合アルゴリズムの概念図

アルゴリズムCに1次元複素フーリエ変換を組み合わせることを考えてみた。図8に，この場合の概念図を示す。

この概念図は，NがMより極端に小さい時，左側の処理（すなわち，M項1次元複素フーリエ変換をN回実行する部分）に，アルゴリズムAを適用し，右側の処理（すなわち，N項1次元複素フーリエ変換をM回実行する部分）に，アルゴリズムCを適用することを示している。この結合アルゴリズムは左側の処理にアルゴリズムCを適用した際の最深DOループ長の減少（平均DOループ長$=N*(\log 2M/2)$）を防止することが可能となる。表2に，Mを128に固定し，Nを4から1024まで変化させてSX-2により測定した結果を示す（単精度計算，単位はmsec）。

この表より，Nが8までは，結合アルゴリズムが優位であるが，それ以上のデータ長では，アルゴリズムCが優位となることがわかる。このように，各次元でデータ長が極端に異なる場合は，アルゴリズムAとの結合アルゴリズムが有効であることがわかる。$M=128$以外でも，結合アルゴリズムが優位となるデータ長の範囲は異なるが，同様の傾向が見られる。

6. 結　　論

2次元複素フーリエ変換を並列処理する場合における3つの基本的なアルゴリズムを示した。その内，

表2．結合アルゴリズムの実測結果（M＝128）
（単精度、msec）

データ長 N	アルゴリズム C	アルゴリズム A+C
4	0.275	0.155
8	0.331	0.310
16	0.443	0.600
32	0.754	1.237
64	1.36	2.44
128	2.55	5.07
256	4.89	10.2
512	10.6	21.2
1024	21.4	42.6

（注1）初期化の処理時間を含まない。
（注2）正規化の処理時間を含まない。

SXシステムにおいては，アルゴリズムCが最適であることを示した。また，各次元でデータ長が極端に異なる場合は，アルゴリズムAとアルゴリズムCを結合したアルゴリズムが，SXシステムに最適であることを示した。

最後に，多くの有益な助言をして頂いた，日本電気(株)基本ソフトウエア開発本部片山　博氏に感謝します。

参　考　文　献

1) Swartztrauber P.N.: Parallel Computations, 51, 1982.
2) Swartztrauber P.N.: SIAM Rev., 19, 490, 1977.
3) Petersen W.P.: Commun. ACM, 26, 1008, 1983.
4) Wang H.H.: BIT, 20, 233, 1980.
5) 花村，萬，津和，片山，渡辺：情処学会第29回全国大会，133(1984)．花村，萬，宮平，津和，宍戸：情処学会第31回全国大会，73(1985).