任意サンプル数のデータによる二次元高速フーリェ変換法(第2報)

——そのプログラミング——

青木由直*

(昭和48年4月28日受理)

Two-Dimensional Fast Fourier Transform with Data of Arbitrary Sampling Number (2)

—Its Programming—

Yoshinao AOKI*

(Received April 28, 1973)

Abstract

A computer program of a two-dimensional fast Fourier transform is discussed. The program is written according to the algorithm applicable for calculating the two-dimensional discrete Fourier transform with data of arbitrary sampling numbers with respect to each dimension. The execution time is examind for various data of different sampling numbers and the results were discussed with reference to the theory. The Fourier spectra distributions of two-dimensional rectangular functions are calculated as examples for processing the two-dimensional data of arbitrary sampling numbers, resulting in showing the validity of the program as a two-dimensional fast Fourier transform program.

1. まえがき

任意サンプル数のデータによる二次元高速フーリェ変換法(FFT)のアルゴリズムについては,このテーマに関する研究の第1報[1]で論じた。本報告は第1報でのアルゴリズムにしたがって,二次元高速フーリェ変換法のプログラミングをおこなうことについて論じている。第1報においては, FFT の計算速度を複素数同志の乗算回数によって定めていた。実際に計算機で FFT の計算をおこなうためには,与えられた指数に対して指数関数の値を計算して,その値とデータに相当するものの積を計算することが必要となる。ここで計算機の内部記憶容量(コア・メモリ)が充分に大きければ,一度計算した指数関数の値を格納しておき,二回目からこの値を使用する時に,改めて指数関数の値を計算せず,前に格納した値を呼び出して積の計算をおこなえばそれだけ演算時間が短縮される。しかしコア・メモリの容量は限られたものであるので,大きなデメンジョンの二次元のアレイをとって FFT の演算をおこなわせようとすれば,一度計算した値を格納する作業領域をいくらでもとることはできない。そのため常にあいているアレイの番地をみつけ,そこに格納したい値を移すという操作が要求される。この操作のためプログラミングが複雑になったり,演算時間が長くなったりするが,限られたコア・メモリを有効に使用して最大の

* 電子工学科

データ数の FFT 演算をおこなわせるためにはこの移し換え操作は必要なことである。これらの点に関する工夫について検討し Glassman[2] による一次元の FFT のプログラムを参考にしながら二次元の FFT のプログラミングを試みた。

2. フロー・チャート

簡単な例として二次元の一方の座標に関するサンプル数を $M=6$, 他の座標に関するサンプル数を $N=3$ とした場合について考えてみる。サンプルされたデータを $X_{mn}(m=1, 2, \ldots 6; n=1, 2, 3)$, そのフーリェ係数を $Y_{mn}(m=1, 2, \ldots 6; n=1, 2, 3)$ とすれば, フーリェ変換マトリックスの因数分解法により離散的フーリェ変換 (DFT) はつぎのように書ける。

$$\begin{pmatrix} Y_{11} & Y_{12} & Y_{13} \\ Y_{21} & Y_{22} & Y_{23} \\ Y_{31} & Y_{32} & Y_{33} \\ Y_{41} & Y_{42} & Y_{43} \\ Y_{51} & Y_{52} & Y_{53} \\ Y_{61} & Y_{62} & Y_{63} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & V^1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & V^2 \\ 1 & V^3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & V^4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & V^5 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & V^2 & 0 & V^4 & 0 \\ 0 & 1 & 0 & V^2 & 0 & V^4 \\ 1 & 0 & V^4 & 0 & V^2 & 0 \\ 0 & 1 & 0 & V^4 & 0 & V^2 \end{pmatrix} \begin{pmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \\ X_{41} & X_{42} & X_{43} \\ X_{51} & X_{52} & X_{53} \\ X_{61} & X_{62} & X_{63} \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 1 & W^1 & W^2 \\ 1 & W^2 & W^4 \end{pmatrix} \tag{1}$$

ただし式 (1) において

$$V=e^{-i2\pi/6},\quad W=e^{-i2\pi/3}$$

である。式 (1) を計算してゆく過程を具体的に書くと, まずデータを 6×3 個の要素を持つアレイ C_1 に格納する。

$$C_1(m, n)=X_{mn}(m=1, 2, \ldots 6; n=1, 2, 3) \tag{2}$$

つぎに式 (1) においてデータのマトリックスとそのすぐ左隣にあるマトリックスとの掛算をおこなう。得られた値は 6×3 個の要素を持つアレイ C_2 に格納する。

$$\left.\begin{aligned} C_2(1, n)&=C_1(1, n)+C_1(3, n)+C_1(5, n)\\ C_2(2, n)&=C_1(2, n)+C_1(4, n)+C_1(6, n)\\ C_2(3, n)&=C_1(1, n)+C_1(3, n)V^2+C_1(5, n)V^4\\ C_2(4, n)&=C_1(2, n)+C_1(4, n)V^2+C_1(6, n)V^4\\ C_2(5, n)&=C_1(1, n)+C_1(3, n)V^4+C_1(5, n)V^2\\ C_2(6, n)&=C_1(2, n)+C_1(4, n)V^4+C_1(6, n)V^2 \end{aligned}\right\} \tag{3}$$

ただし式 (3) において $n=1, 2, 3$ であり, 式 (3) は 6×3 個のマトリックス要素をあらわしている。つぎに式 (3) で計算されたマトリックスとさらにその左隣にあるマトリックスの掛算をおこなう。得られた値は 6×3 個の要素を持つアレイ C_3 に格納する。

$$\left.\begin{aligned} C_3(1, n)&=C_2(1, n)+C_2(2, n)\\ C_3(2, n)&=C_2(3, n)+C_2(4, n)V^1\\ C_3(3, n)&=C_2(5, n)+C_2(6, n)V^2\\ C_3(4, n)&=C_2(1, n)+C_2(2, n)V^3\\ C_3(5, n)&=C_2(3, n)+C_2(4, n)V^4\\ C_3(6, n)&=C_2(5, n)+C_2(6, n)V^5 \end{aligned}\right\} \tag{4}$$

式 (4) においても $n=1, 2, 3$ である。つぎに式 (4) で計算されたマトリックスと式 (1) の最右にあるマトリックスの掛算をおこない, 得られた値を 6×3 個の要素を持つアレイ C_4 に格納する。

$$\left.\begin{aligned} C_4(m,\ 1)&=C_3(m,\ 1)+C_3(m,\ 2)+C_3(m,\ 3)\\ C_4(m,\ 2)&=C_3(m,\ 1)+C_3(m,\ 2)W^1+C_3(m,\ 3)W^2\\ C_4(m,\ 3)&=C_3(m,\ 1)+C_3(m,\ 2)W^2+C_3(m,\ 3)W^4 \end{aligned}\right\} \quad (5)$$

ただし式 (5) において $m=1,\ 2,\ \ldots 6$ である。式 (2)～(5) までを計算してゆくようにプログラミングをおこなえば FFT のアルゴリズムにしたがって DFT の係数が計算できる。ここで各アレイ C_1, C_2, C_3, C_4 を別々にとると，サンプル数 M, N が比較的小さな値でも，アレイのデメンジョンの総数は計算機のコア容量をこえてしまう。そこで実際のプログラミングにあたっては $C(M,\ N)$ と $CC(M,\ N)$ の $M\times N$ 個の要素をもつ二種類のアレイを使用して演算をおこなっている。例えば式 (2) の C_1 を C に格納して，式 (3) の計算された C_2 の値を CC に格納する。つぎに CC に格納された値を C に移し，式 (4) で計算された値を CC に格納する。このようにして計算してゆくと二種類の二次元アレイで計算をおこなってゆけるが，計算された値を移し換えてゆく必要があるのでその分だけ演算時間が長くなる。つぎに演算時間を短かくするために工夫しなければならぬ事は，一度計算した $V^1=\exp[-i2\pi/6]$, $V^2=\exp[-i2\pi/3]$, ... 等の値を二回目に使用する時は再び計算しないように別のところに格納しておくことである。ただし格納のために大きなデメンジョンの二次元アレイをとることは前述のようにコア容量から制限されてくるので，一次元のアレイに格納して二回目に使用する時に呼び出して計算できるようにしている。例えば式 (3) の $C_2(3,\ n)$ を計算する場合，$n=1$ として計算したときの V^2, V^4 の値を一次元のアレイに格納しておき，$n=2$ の場合の計算のときはこれらの格納した値を呼び出して演算をおこなう。$n=3$ の場合も同様である。この場合一次元のアレイにしか V^2, V^4 等の計算値を格納できぬので $C_2(4,\ n)$ を計算する場合 $n=1$ として再び V^2, V^4 の値を計算せねばならぬ。このことのため計算時間の無駄があるが，二次元アレイの要素数，したがってデータの個数をできる限り大きくとるためにはやむをえぬことである。

以上の事を考慮して二次元 FFT のプログラミングをおこなったが，そのフロー・チャートを図1に示す。サブルーチン FFT の引数の個数は7個で，$IFT=0$ でフーリェ変換，$IFT=1$ でフーリェ逆変換をおこなう。引数 MD, ND は，二次元のそれぞれの座標 (t および v 座標) に関するサンプル数をあらわす。MF, NF はそれらのサンプル数を因数分解したときの因数の個数であり，IFX, IFY はそれらの因数である。図1において $JFT=1$ であればデータのマトリックスの左側にある変換マトリックスとの掛算 ($[Y]=[V][X][W]$ と DFT をあらわせば，$[V][X]$ の計算) を $JFT=2$ であれば計算されたマトリックス $[V][X]$ と右側のマトリックスの掛算 ($([V][X])[W]$ の計算) をおこなっている。①のループではデータのマトリックスの一行目と変換マトリックスの一列目の掛算をおこなっており，その際計算された $\exp[-i2\pi m/M]$ $(m=1,\ 2,\ \ldots M)$ や $\exp[-i2\pi n/N](n=1,\ 2,\ \ldots N)$ の値を一次元のアレイ W に格納しておく。このようにして②のループでデータのマトリックスの二行目以下と変換マトリックスの一列目の掛算をおこなう際に W に格納された値を呼び出して計算をおこなっている。ループ③でデータのマトリックスと変換マトリックスの掛算の一列目の計算がおわることになる。ループ④⑤でデータのマトリックスと因数分解された変換マトリックスの第一番目のものとの掛算が完了する。ここでまだ計算せねばならぬ変換マトリックスが残っていれば，ループ⑥でアレイ CC に格納されている計算値をアレイ C に移し換えてループ⑧で最初に戻って計算を続ける。二次元の一方の座標に関する計算が終了すれば，ループ⑦でその座標に関するサンプル数 ANN で除してその値をアレイ C に格納する。つぎに⑨で最初に戻り，もう一方の座標に関して今までと同様の計算を繰返す。計算が終了すればメイン・プログラムに戻る。このフロー・チャートにしたが

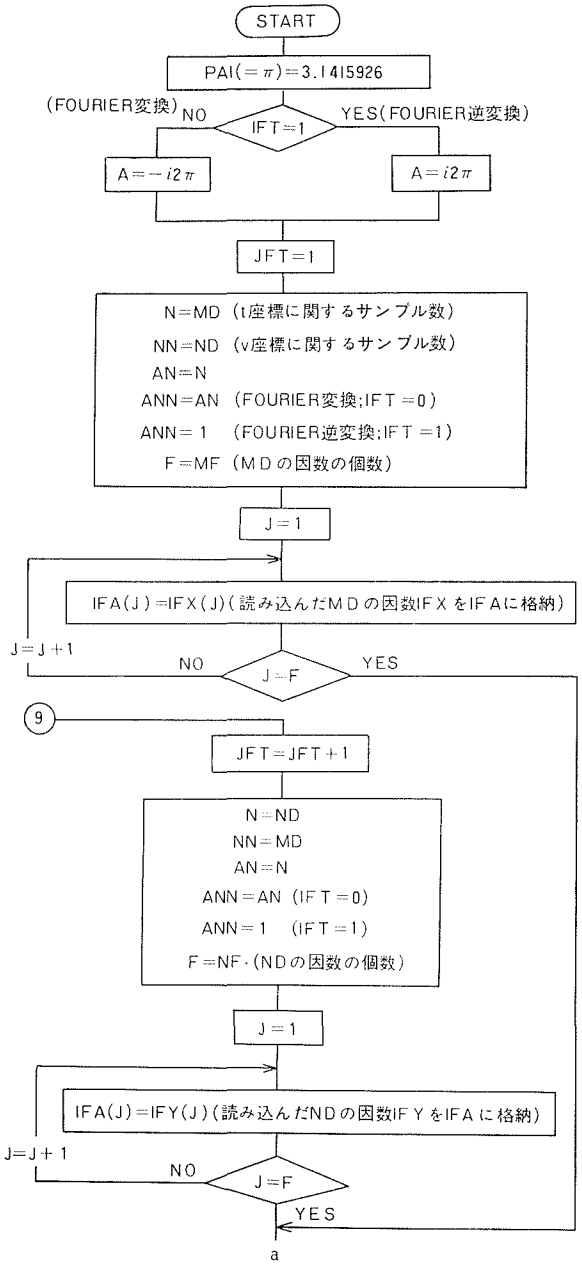
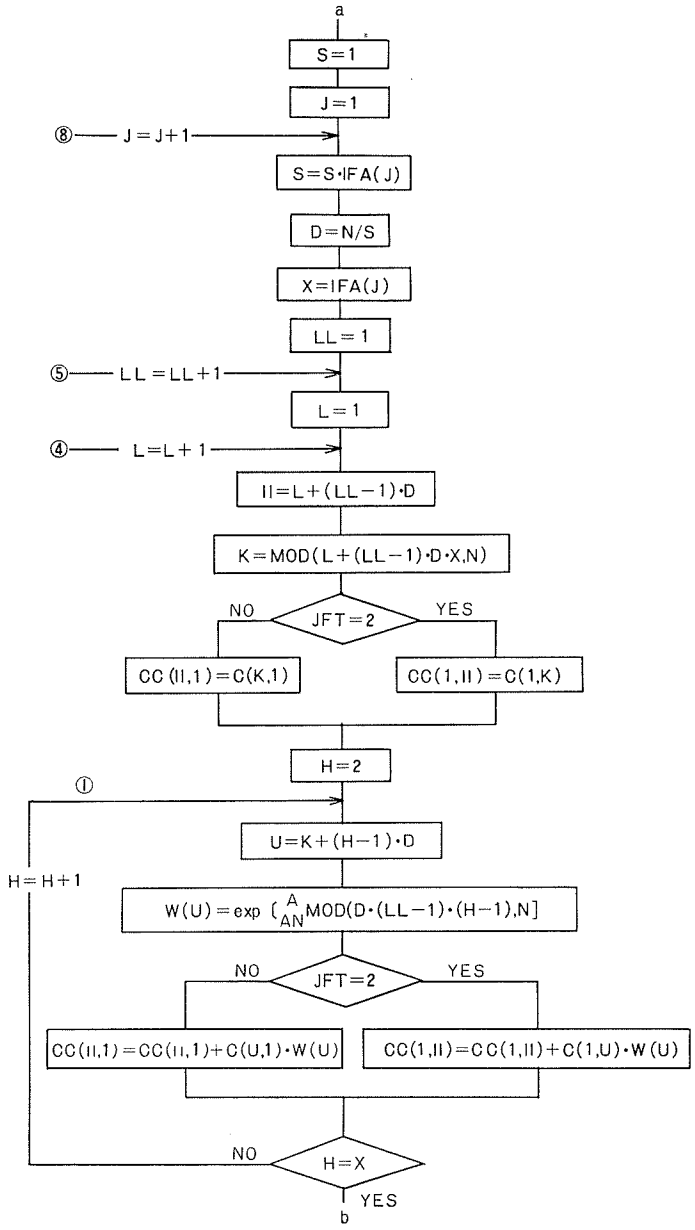
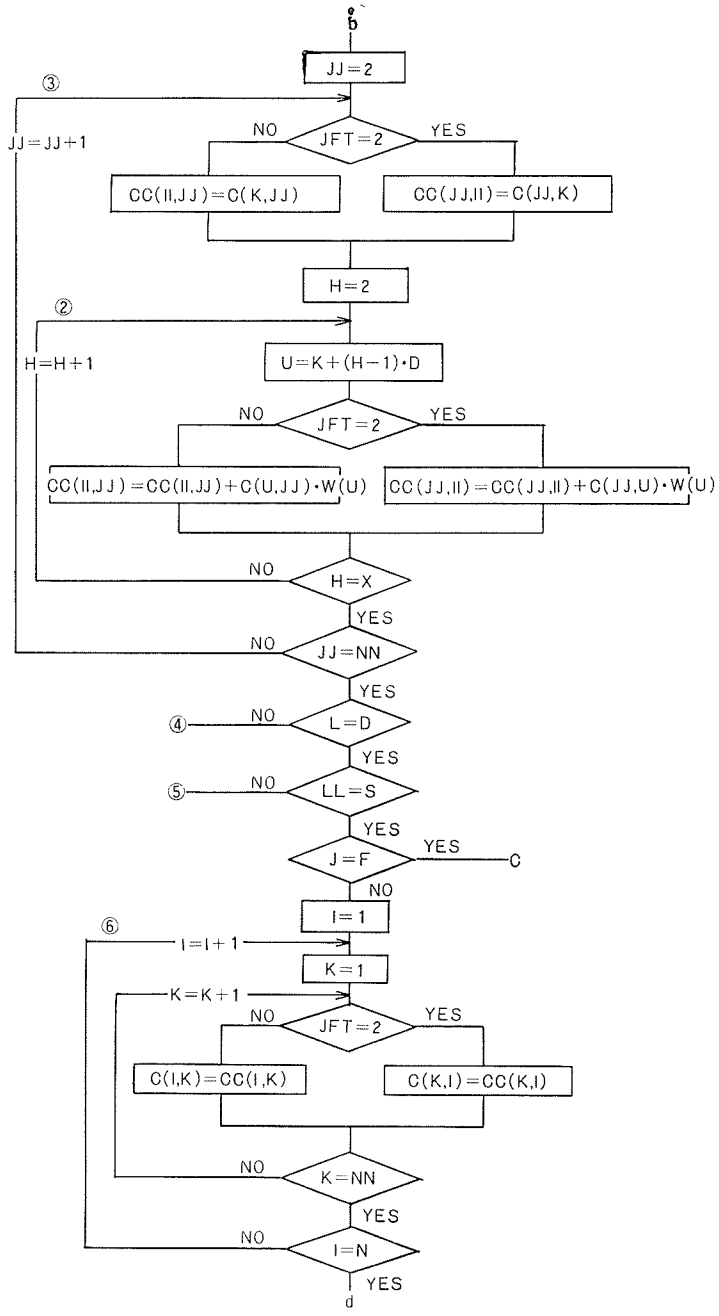
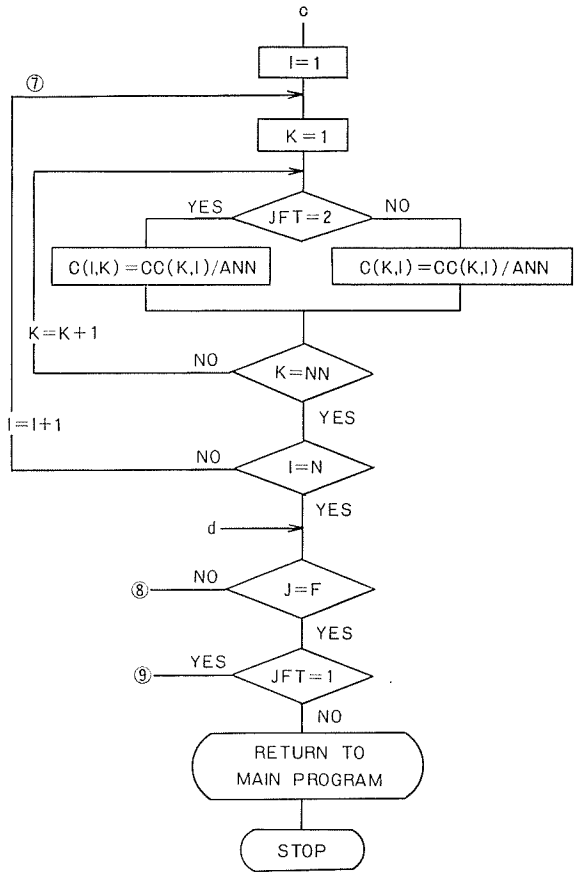


図1 二次元 FFT プログラムのフロー・チャート


a
S=1
J=1
⑧ J=J+1
S=S·IFA(J)
D=N/S
X=IFA(J)
LL=1
⑤ LL=LL+1
L=1
④ L=L+1
II=L+(LL−1)·D
K=MOD(L+(LL−1)·D·X,N)
NO
JFT=2
YES
CC(II,1)=C(K,1)
CC(1,II)=C(1,K)
H=2
①
U=K+(H−1)·D
H=H+1
W(U)=exp [A/AN MOD(D·(LL−1)·(H−1),N]
NO
JFT=2
YES
CC(II,1)=CC(II,1)+C(U,1)·W(U)
CC(1,II)=CC(1,II)+C(1,U)·W(U)
NO
H=X
YES
b


b
JJ=2
③
JJ=JJ+1
NO
JFT=2
YES
CC(II,JJ)=C(K,JJ)
CC(JJ,II)=C(JJ,K)
H=2
②
H=H+1
U=K+(H-1)・D
NO
JFT=2
YES
CC(II,JJ)=CC(II,JJ)+C(U,JJ)・W(U)
CC(JJ,II)=CC(JJ,II)+C(JJ,U)・W(U)
NO
H=X
YES
NO
JJ=NN
YES
④
NO
L=D
YES
⑤
NO
LL=S
YES
J=F
YES
C
NO
I=1
⑥
I=I+1
K=1
K=K+1
NO
JFT=2
YES
C(I,K)=CC(I,K)
C(K,I)=CC(K,I)
NO
K=NN
YES
NO
I=N
YES
d



って作製したプログラムをつぎに示す。

3. プログラム

任意サンプル数のデータによる二次元高速フーリェ変換法の計算機プログラムを図2に示す。データのサンプル数は M (プログラムでは MD)=48, N(プログラムでは ND)=72 に選んである。したがって M の因数の個数は $MF=5$ で因数は $IFX=2, 2, 2, 2, 3$ である。N の因数の個数は $NF=5$ で因数は $IFY=2, 2, 2, 3, 3$ である。作成したプログラムの演算時間を調べるため，サンプル数を変えて計算した結果を図3に示す。図3において実線が演算時間を示し，フオトランのコンパイリング等の時間も含めた総演算時間を一点鎖線で示した。図3より明らかなようにサンプル数の因数が多くなる程 FFT の効力があらわれてきて演算時間が短縮されてくる。これらの事については第1報ですでに論じているが，理論値との比較を簡単に述べておく。第1報の式 (36) より演算時間は次式のものに比例する。

$$F(M, N)=MN\left(\sum_{i=1}^{m} p_i+\sum_{i=1}^{n} q_i\right) \tag{6}$$

ここで M, N はサンプル数，p_i, q_i はそれぞれのサンプル数の因数である。式 (6) により演算時間の理論値を計算し，$M=32, N=32$ の場合の実際の演算時間と一致させるように比例定数を選ぶと理論値は図3の鎖線のようになる。これより実際の演算時間はサンプル数が異なれば，ほ

```
C     SUBROUTINE OF TWO DIMENSIONAL FOURIER TRANSFORM
      SUBROUTINE FFT(IFT,MD,ND,MF,NF,IFX,IFY)
      COMMON C,CC
      DIMENSION IFX(5),IFY(5),IFA(5)
      COMPLEX C(48,72),CC(48,72),W(72),A,AA,B
      INTEGER D,F,H,S,U,X
      PAI=3.1415926
      IF(IFT.EQ.1) GO TO 10
      A=(0.0,-2.0)*PAI
      GO TO 20
   10 A=(0.0,2.0)*PAI
   20 CONTINUE
      JFT=1
      N=MD
      NN=ND
      AN=N
      ANN=AN*(1-IFT)+1.0*IFT
      F=MF
      DO 30 J=1,F
   30 IFA(J)=IFX(J)
      GO TO 60
   40 JFT=JFT+1
      N=ND
      NN=MD
      AN=N
      ANN=AN*(1-IFT)+1.0*IFT
      F=NF
      DO 50 J=1,F
   50 IFA(J)=IFY(J)
   60 CONTINUE
      S=1
      DO 1600 J=1,F
      S=S*IFA(J)
      D=N/S
      X=IFA(J)
      DO 800 LL=1,S
      DO 700 L=1,D
      II=L+(LL-1)*D
      K=MOD(L+(LL-1)*D*X,N)
      IF(JFT.EQ.2) GO TO 70
      CC(II,1)=C(K,1)
      GO TO 80
   70 CC(1,II)=C(1,K)
   80 CONTINUE
      DO 100 H=2,X
      U=K+(H-1)*D
      IG=MOD(D*(LL-1)*(H-1),N)
      G=IG
      AA=A/AN
      B=G*AA
      W(U)=CEXP(B)
      IF(JFT.EQ.2) GO TO 90
      CC(II,1)=CC(II,1)+C(U,1)*W(U)
      GO TO 100
   90 CC(1,II)=CC(1,II)+C(1,U)*W(U)
  100 CONTINUE
      DO 600 JJ=2,NN
      IF(JFT.EQ.2) GO TO 200
      CC(II,JJ)=C(K,JJ)
      GO TO 300
  200 CC(JJ,II)=C(JJ,K)
  300 CONTINUE
      DO 500 H=2,X
      U=K+(H-1)*D
      IF(JFT.EQ.2) GO TO 400
      CC(II,JJ)=CC(II,JJ)+C(U,JJ)*W(U)
      GO TO 500
  400 CC(JJ,II)=CC(JJ,II)+C(JJ,U)*W(U)
  500 CONTINUE
  600 CONTINUE
  700 CONTINUE
  800 CONTINUE
      IF(J.EQ.F) GO TO 1200
      DO 1100 I=1,N
      DO 1000 K=1,NN
      IF(JFT.EQ.2) GO TO 900
      C(I,K)=CC(I,K)
      GO TO 1000
  900 C(K,I)=CC(K,I)
 1000 CONTINUE
 1100 CONTINUE
      GO TO 1600
 1200 CONTINUE
      DO 1500 I=1,N
      DO 1400 K=1,NN
      IF(JFT.EQ.2) GO TO 1300
      C(I,K)=CC(I,K)/ANN
      GO TO 1400
 1300 C(K,I)=CC(K,I)/ANN
 1400 CONTINUE
 1500 CONTINUE
 1600 CONTINUE
      IF(JFT.EQ.1) GO TO 40
      RETURN
      END
```

図2 任意サンプル数の二次元 FFT のプログラム

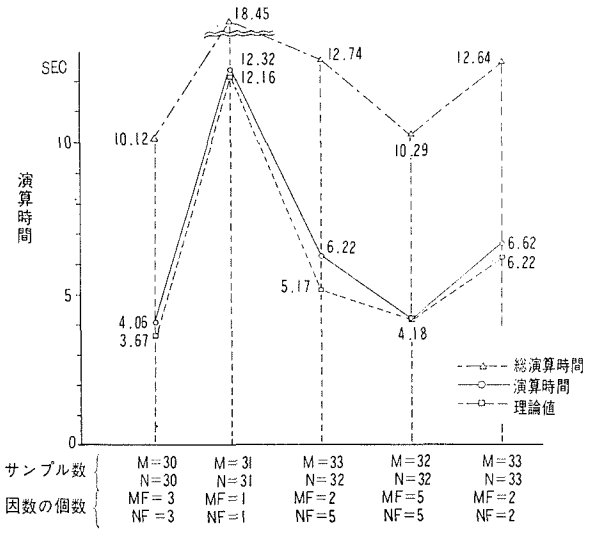


図3 実際の演算時間と理論値との比較

ぼ理論通りの変化をしているのがわかる。

ここで作成を試みた二次元 FFT のプログラムは最善のものとはいい難い。例えばサンプル数が $2^n \times 2^n (n=1, 2, ...)$ に限定された二次元 FFT において中村氏が作成したプログラム[3] では，$M=N=32$ の場合演算時間は 3.21 秒，総演算時間は 9.58 秒であるから本報告のプログラムより能率良く作られていると考えられる。さらに中村氏のプログラムでは FFTの 演算には 2 種類の二次元のアレイは必要でなくデータの個数だけの要素を持つ一種類の二次元アレイで演算をおこなっているのでこの点でもすぐれている。本報告で作成を試みたプログラムをさらに改良してゆくことについては今後の課題としたい。

4. フーリェ・スペクトルの並べ換え

作成したプログラムによる二次元高速フーリェ変換の一例として，二次元の矩形関数のフーリェ変換をおこなってみた。サンプル点の数は 48×72 個で，データとしては関数値の存在する点として 9 点を選びその値を 1.0 とし，残りのサンプル点では 0.0 となるようにして計算をおこなった。得られたフーリェ変換の絶対値でスペクトル分布を表示したのが図 4 である。図 4 においては 5 段階のレヴェルを用いてスペクトルを表示している。一度フーリェ変換したものをフーリェ逆変換で元のデータを再現したものを図 5 に示す。図 5 は明らかに元の関数をあらわしており，元のデータで 0.0 であったところは 10^{-7} 以下の値で再現されている。図 4 のスペクトル分布で注意せねばならぬことは，FFT 特有の折返しによるスペクトル分布があらわれていることである。したがって正しいスペクトルを求めようとするなら図 4 のスペクトルを並べ換える必要がある。その並べ換え方を図 6 に示す。M(奇数) $\times N$(偶数) のアレイを考えると，図 6-a において (1, 1) に直流分のスペクトルがあらわれるので，これを $((M+1)/2, N/2)$ に移すように 図6-a のアレイの①の部分と④の部分を移し換える。同様に図 6-a の③の部分の $((M+3)/2, (N+2)/2)$ の点を②の部分の $(1, N)$ の点に移し換えるように②の部分と③の部分を移し換える。このような操作によって図 4 のスペクトル分布を図 6-b のように並べ換えたものを図 7 に示す。二次元のフーリェ・スペクトル座標を s, u とすれば，図 7 には二次元の矩形関数である図 5 のフーリェ

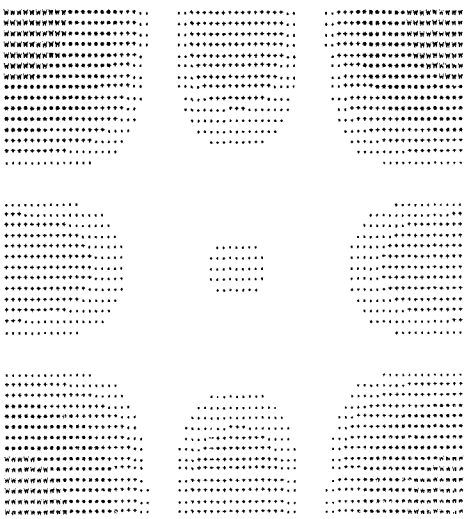


図 4 二次元 FTF により計算された二次元矩形関数のフーリェ・スペクトル分布

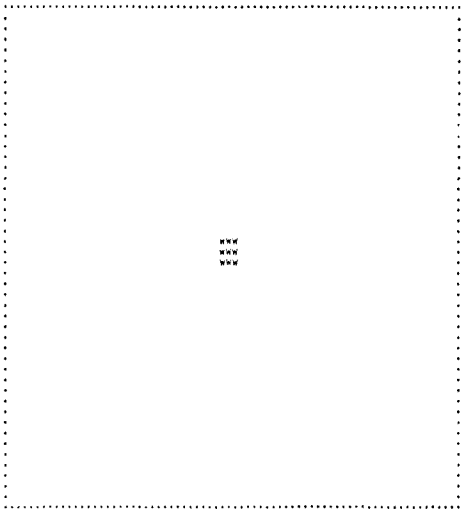


図 5 フーリェ逆変換で再現された元次の二元矩形関数

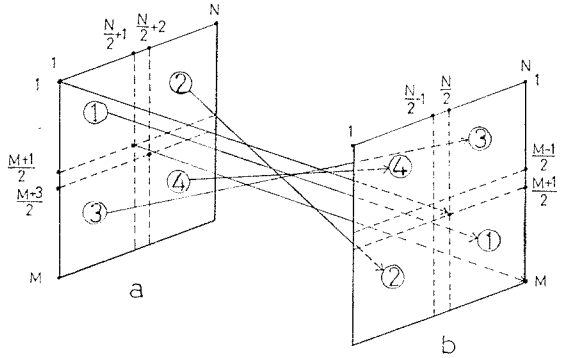


図6 フーリェ・スペクトルの並べ換えの方法

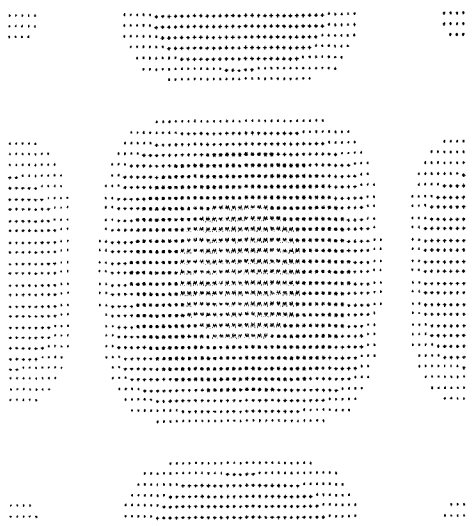


図7 並べ換えられた図5の関数のフーリェ・スペクトル分布

変換である $\sin s \sin u/su$ の関数の一部分（絶対値で表示）があらわれてきている。

5. サンプリング間隔とサンプル数

二次元の直角座標を t, v, これらの座標に関するサンプリング間隔を Δt, Δv, 開口（サンプリング範囲）を L_t, L_v, さらにサンプル数を M, N とすればこれらの間にはつぎの関係がある。

$$\left.\begin{aligned} M&=\frac{L_t}{\Delta t}\\ N&=\frac{L_v}{\Delta v}\end{aligned}\right\} \qquad (7)$$

式 (7) よりつぎのことがいえる。サンプル数を多くしてゆくことは，i) サンプリング間隔を小さくし密にサンプリングをおこなう，ii) 開口を大きくとる，の二つの場合が考えられる。図7のフーリェ・スペクトルを計算した際には，実際にサンプリングをおこなってデータを得ているわけではなく，式 (7) のサンプル数 M, N のみが与えられているので典型的な場合として，i) $\Delta t\left(=\frac{1}{M}=\frac{1}{48}\right)>\Delta v\left(=\frac{1}{N}=\frac{1}{72}\right)$ で $L_t=L_v=1$ の場合か，ii) $\Delta t=\Delta v=1$ で $L_t(=M=48)<L_v$ $(=N=78)$ の場合が考えられる。i) の場合にはフーリェ・スペクトルは t, v 座標に対応するスペクトル座標 s, u に関して同じ間隔 $1/L_t=1/L_v$ の細かさで求められるが，スペクトル分布の範囲は s 座標に関しては $1/\Delta t=48$, u 座標に関しては $1/\Delta v=72$ と u 座標の方が広い範囲にわたって求まる。しかしこの場合元の二次元矩形関数は t 座標方向に伸び，v 座標方向には縮まった長方形となるので，そのスペクトル分布は s 座標に関してはせばまり，u 座標に関しては拡った二次元の sinc 関数となるので，結局スペクトル分布としては s 座標も u 座標も同じ形のものが表示されることになる。ii) の場合にはスペクトル分布の範囲は s 座標 u 座標ともに $1/\Delta t=1/\Delta v$ と同じ範囲のものが計算されるが，スペクトルは s 座標に関しては $1/L_t=1/48$ の間隔で，u 座標に関しては $1/L_v=1/72$ とより細かな間隔で求まっていることになる。このように二次元の各座標に関するサンプル数が異なっている場合，サンプル数を多くしてより広い範囲のフーリェ・スペクトル分布を求めるか，またはより細かなサンプリング間隔でフーリェ・スペク

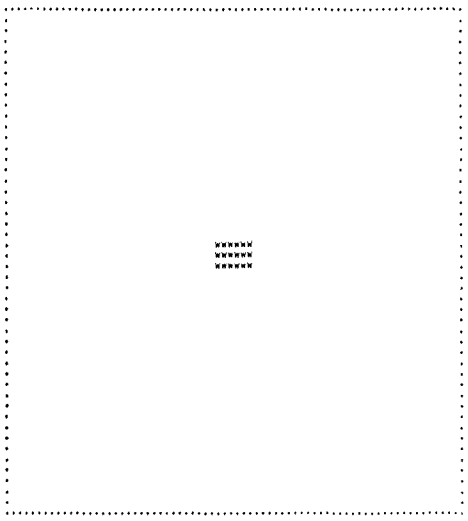


図8　二次元矩形関数

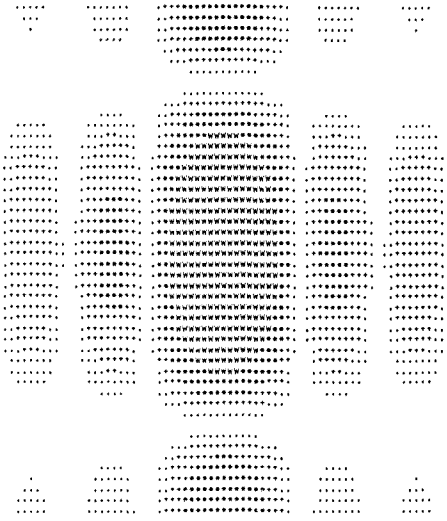


図9　図8の関数のフーリェ・スペクトル分布

トルを計算するかは実際のサンプリングの仕方によってきまる。

別の例として図5の二次元矩形関数の関数値の存在する範囲を横方向のみに2倍に拡げた図8に示す二次元矩形関数のフーリェ変換を求めてみる。図7のフーリェ・スペクトルを求めたときと同様にして図8のフーリェ・スペクトルを計算し表示したものが図9である。この場合も図8の関数は $\Delta t > \Delta v$ でサンプルされたか $\Delta t = \Delta v$ でサンプルされたか区別がつかないが，いずれの場合にせよフーリェ・スペクトルは u 座標に関しては s 座標よりもより高周波成分が得られることは明らかであり，図9にもそれが示されている。

6. あ と が き

本テーマの第1報で論じた任意サンプル数のデータを処理できる二次元高速フーリェ変換法のアルゴリズムに従って，そのプログラミングを試みた。ここで作成したプログラムは最善のものとはいい難いが，一応二次元 FFT のプログラムとして使用できるものである。二次元の各次元のサンプル数が異なる場合サンプリングの仕方と計算されたフーリェ・スペクトル分布のあらわれ方がどのように関連しているかについても簡単に論じたが，このことについては実際のデータを処理する場合により注意を払わねばならぬ問題である。本テーマに関する第3報以後では二次元 FFT の応用例について論ずる予定である。

文　　献

2) 青木由直：工学部研究報告，67，(昭48)，pp. 73-82.
2) Glassman, J. A.: IEEE Trans. on Computers, vol. C-19 (1970), 2, pp. 105-116.
3) 実際にプログラムが記載された論文はないが，このプログラムを使用しての研究としては，Nakamura, S. and C. Yoshimoto：応用電気研究所報告，vol. 22, (1970), 3, 4, pp. 144-156；Aoki, Y.: IEEE Trans. on Audio and Electroacoustics, AU-18 (1970), 3, pp. 258-267.