11. 離散フーリエ変換

時間領域における連続関数 x(t)は，周波数領域の連続関数 X(f)を介して，

フーリエ変換(Fourier transform)　：$X(f)=\int_0^{\infty} x(t)e^{-i2\pi ft}dt$

逆フーリエ変換(inverse Fourier transform)：$x(t)=\int_{-\infty}^{\infty} X(f)e^{i2\pi ft}df$

と展開される。x(t)は区間[0,T]以外では 0 とすると，フーリエ変換における無限区間[0,∞]での積分は有限区間[0,T]での積分となる。この区間を N 等分（小区間幅：T_s=T/N，N 等分点：t_j=jT_s（j=0,1,…,N））して複合台形則により数値積分を行い，さらに周波数 f_k=k/T（k=0,1,…,N-1)において評価すると，

$$X(f_k)=T_s\sum_{j=0}^{N-1}x(t_j)\ e^{-i2\pi kj/N}+O\left(T_s^{\,2}\right)$$

を得る。この式を T_s で割って y(f_k)=X(f_k)/T_s とおいたものが，離散フーリエ変換(discrete Fourier transform, DFT)である。x(t_j)=x_j，y(f_k)=y_k と記せば，DFT は，時間領域における時間間隔 T_s（サンプリングタイムとよぶ）の N 点のデータ $\boldsymbol{x}=[x_0,x_1,\cdots,x_{N-1}]^T$ を周波数領域における周波数間隔 f_1=1/T（基本周波数とよぶ）の N 点のデータ $\boldsymbol{y}=[y_0,y_1,\cdots,y_{N-1}]^T$ に変換するものであり，回転因子

$$w_N=e^{-i2\pi/N}$$

を用いて，以下のように書ける。

$$y_k=\sum_{j=0}^{N-1}x_j w_N^{kj}\qquad(k=0,1,\cdots,N-1)$$

行列・ベクトル表示を行うと，

$$\boldsymbol{y}=F_N\boldsymbol{x},\qquad F_N=\begin{bmatrix}1 & 1 & 1 & \cdots & 1\\ 1 & w_N^1 & w_N^2 & \cdots & w_N^{N-1}\\ 1 & w_N^2 & w_N^4 & \cdots & w_N^{2(N-1)}\\ \vdots & \vdots & \vdots & & \vdots\\ 1 & w_N^{(N-1)} & w_N^{2(N-1)} & \cdots & w_N^{(N-1)(N-1)}\end{bmatrix}$$

となる。逆離散フーリエ変換(inverse discrete Fourier transform, IDFT)は，$\boldsymbol{y}$ から $\boldsymbol{x}$ を求めるプロセスであり，次のようになる。

$$x_j=\frac{1}{N}\sum_{k=0}^{N-1}y_k w_N^{-jk}\qquad(j=0,1,\cdots,N-1)$$

これは，行列・ベクトル表示により次のように書ける。

$$\boldsymbol{x}=F_N^{-1}\boldsymbol{y},\qquad F_N^{-1}=\frac{1}{N}\begin{bmatrix}1 & 1 & 1 & \cdots & 1\\ 1 & w_N^{-1} & w_N^{-2} & \cdots & w_N^{-(N-1)}\\ 1 & w_N^{-2} & w_N^{-4} & \cdots & w_N^{-2(N-1)}\\ \vdots & \vdots & \vdots & & \vdots\\ 1 & w_N^{-(N-1)} & w_N^{-2(N-1)} & \cdots & w_N^{-(N-1)(N-1)}\end{bmatrix}$$

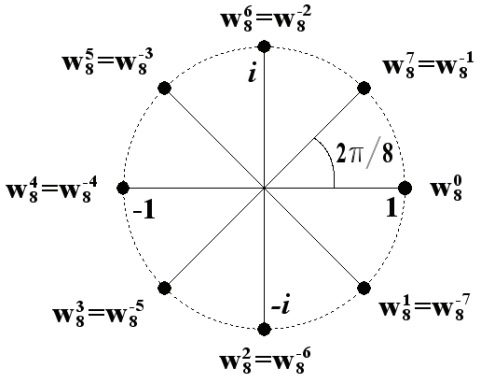
回転因子 w_N は 1 の N 乗根である。その k 乗あるいは-k 乗(k=0,1,…,N-1）もすべて 1 の N 乗根であり，これらはそれぞれ複素平面上の単位円の角度干 2πk/N (k=0,1,…,N-1）における点である。回転因子の演算には，次の性質

周期性：$w_N^k=w_N^{k\pm jN}\ (j=0,1,\cdots)$

および N が偶数のときには

原点対称性：$w_N^{k+N/2}=-w_N^k$

がある。N=8 のときの回転因子の例を図示する。



高速フーリエ変換（fast Fourier transform, FFT）とは，上記の変換を効率よく計算するアルゴリズムであり，時間間引き型 FFT と周波数間引き型 FFT がある。

11.1 時間間引き型 FFT

時間間引き型 FFT とは，DFT の右辺（時間領域におけるデータ）を以下のように x_k が偶数と奇数のグループに並べ替えて，データ点数 N の DFT をデータ点数 N/2 の DFT に分解していき，演算量を減らすものである。

$$y_k = \sum_{j=0}^{N-1} x_j w_N^{kj} = \sum_{r=0}^{N/2-1} x_{2r} w_N^{(2r)k} + \sum_{r=0}^{N/2-1} x_{2r+1} w_N^{(2r+1)k} = X_k^{(0)} + w_N^k X_k^{(1)} \qquad (k = 0,1,\cdots,N-1)$$

$$\begin{cases} X_{k'}^{(0)} = \sum_{r=0}^{N/2-1} x_{2r} w_{N/2}^{rk'}, & X_{k'+N/2}^{(0)} = X_{k'}^{(0)} \\ X_{k'}^{(1)} = \sum_{r=0}^{N/2-1} x_{2r+1} w_{N/2}^{rk'}, & X_{k'+N/2}^{(1)} = X_{k'}^{(1)} \end{cases}$$

データ総数が $N=2^m$ の場合，p 段目ではデータ点数 $N_p=2^p$ の DFT グループが $n_g=N/N_p=2^{m-p}$ 個ある。p 段の最初において，各グループ毎に前半には偶数列要素，後半には奇数列要素を順に配置すると，p 段第 i グループ（i=0, 1, …, n_g−1）の前半（偶数列要素）は p-1 段の第 2i グループ要素，後半（奇数列要素）は p-1 段の第 2i+1 グループ要素に相当する。p 段 i グループの第 k 要素を新たに $X^p_{i,k}$ (k=0, 1, …N_p−1) と記すと，$w_{Np}^{k+Np/2}=-w_{Np}^{k}$ より

$$\left.\begin{aligned} X_{i,j}^p &= X_{2i,j}^{p-1} + w_{N_p}^j X_{2i+1,j}^{p-1} \\ X_{i,j+N_p/2}^p &= X_{2i,j}^{p-1} - w_{N_p}^j X_{2i+1,j}^{p-1} \end{aligned}\right\} \qquad \begin{aligned} &(i = 0,\cdots,n_g-1) \\ &(j = 0,\cdots,N_p/2-1) \end{aligned}$$

と分解されることになる。これを p=1 から始め，p=m まで繰り返せばよい。

$N=2^3=8$ の場合の例を示す。最後の第 3 段では，8 点の DFT グループ（第 0 グループ）が 1 組存在し，この順番で周波数領域における値 y_k となっている。

$$\left.\begin{aligned} X_{i,0}^3 &= X_{2i,0}^2 + w_8^0 X_{2i+1,0}^2 \\ X_{i,1}^3 &= X_{2i,1}^2 + w_8^1 X_{2i+1,1}^2 \\ X_{i,2}^3 &= X_{2i,2}^2 + w_8^2 X_{2i+1,2}^2 \\ X_{i,3}^3 &= X_{2i,3}^2 + w_8^3 X_{2i+1,3}^2 \\ X_{i,4}^3 &= X_{2i,0}^2 - w_8^0 X_{2i+1,0}^2 \\ X_{i,5}^3 &= X_{2i,1}^2 - w_8^1 X_{2i+1,1}^2 \\ X_{i,6}^3 &= X_{2i,2}^2 - w_8^2 X_{2i+1,2}^2 \\ X_{i,7}^3 &= X_{2i,3}^2 - w_8^3 X_{2i+1,3}^2 \end{aligned}\right\} \qquad (i=0)$$

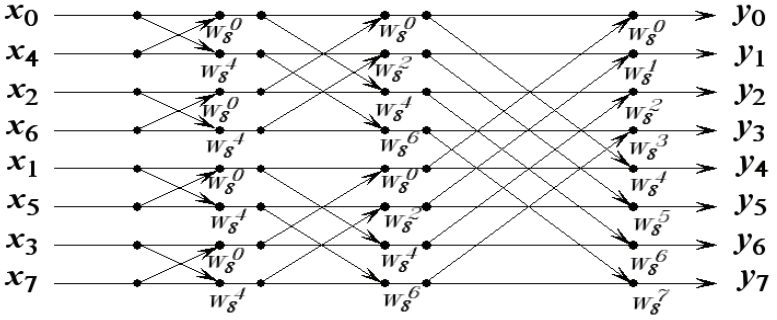
ここで右辺の $(X^2_{0,0}, X^2_{0,1}, X^2_{0,2}, X^2_{0,3})$ は偶数項 (x_0, x_2, x_4, x_6) をデータに持つ第 2 段の第 0DFT グループ，$(X^2_{1,0}, X^2_{1,1}, X^2_{1,2}, X^2_{1,3})$ は奇数項 (x_1, x_3, x_5, x_7) をデータに持つ第 2 段の第 1DFT グループである。このように，第 2 段では，4 点の DFT グループが 2 組存在し，さらに次のように分解される。

$$\left.\begin{aligned} X_{i,0}^2 &= X_{2i,0}^1 + w_4^0 X_{2i+1,0}^1 \\ X_{i,1}^2 &= X_{2i,1}^1 + w_4^1 X_{2i+1,1}^1 \\ X_{i,2}^2 &= X_{2i,0}^1 - w_4^0 X_{2i+1,0}^1 \\ X_{i,3}^2 &= X_{2i,1}^1 - w_4^1 X_{2i+1,1}^1 \end{aligned}\right\} \qquad (i=0,1)$$

右辺の $(X^1_{0,0}, X^1_{0,1})$，$(X^1_{1,0}, X^1_{1,1})$，$(X^1_{2,0}, X^1_{2,1})$，$(X^1_{3,0}, X^1_{3,1})$ はそれぞれの偶数項 (x_0, x_4)，奇数項 (x_2, x_6)，偶数項 (x_1, x_5)，奇数項 (x_3, x_7) をデータに持つ第 1 段の第 0, 1, 2, 3DFT グループである。つまり第 1 段では，2 点の DFT グループが 4 組存在し，さらに次のように分解される。

$$\left.\begin{aligned} X_{i,0}^1 &= X_{2i,0}^0 + w_2^0 X_{2i+1,0}^0 \\ X_{i,1}^1 &= X_{2i,0}^0 - w_2^0 X_{2i+1,0}^0 \end{aligned}\right\} \qquad (i=0,1,2,3)$$

実際の計算は第 1 段から始めて第 2 段を行い，第 3 段で終了する。データ処理の流れは以下のように書ける。



第1段の最初において与えるべきデータ x_k の順番（添え字番号 k の順番）は，0,1,…,7 を表す 2 進数
$(000)_2$, $(001)_2$, $(010)_2$, $(011)_2$, $(100)_2$, $(101)_2$, $(110)_2$, $(111)_2$
のビットを逆転させたもの（以下）となっている。

$(000)_2$	$(100)_2$	$(010)_2$	$(110)_2$	$(001)_2$	$(101)_2$	$(011)_2$	$(111)_2$
=0	=4	=2	=6	=1	=5	=3	=7

【アルゴリズム 11.1.1】

DFT のアルゴリズムは以下のとおり。ただし，Y，X_k，$w_{Np}^{\ j}$ は複素数である。

1) ビット逆転により，$N=2^m$ 点の入力データ x_i(対応する時間は $t_i=iT_s$)の順序を並べ替え，その結果を変数 $X_0, X_1, \cdots, X_{N-1}$ にセットする。

```
for i=0 to N-1          {データ数 N=2^m だけ繰り返す}
    bit  := i               {i を 2 進数で表したときのビットを逆転して変数 bitr にセット}
    bitr := 0;              {C言語の左右シフト演算子<<, >>, 論理積演算子&, 論理和演算子|を用いる}
    for k=1 to m
        bitr := bitr << 1               {bitrを左に1ビットだけシフト}
        bitr := bitr | (bit & 1)        {bitの1ビット目をbitrの1ビット目にセット}
        bit  := bit >> 1                {bitを右に1ビットだけシフト}
    end
    if i<bitr then                      {i＜bitr のときのみ，i番目とbitr番目のデータを入れ換え}
        Y    := X_i                     {注：既に交換済みのものを再度交換すると元に戻るので}
        X_i  := X_bitr
        X_bitr := Y
    end
end
```

2) 偶数列と奇数列への分解の統合を m 段繰り返す。次に示すのはアルゴリズムの原型であり，$w_{Np}^{\ j}$ の計算を効率よく行うべく更に改良できる。

```
for p=1 to m                    {第 p 段：p=1,…,m}
    Np := 2^p                       {Np: 各グループのデータ点数}
    Ng := N/Np                      {ng: グループ数}
    for i=0 to ng-1
        for j=0 to Np/2-1
            k := Np i + j
            l := k+Np/2
            Y := w_Np^j X_l
            X_l := X_k-Y
            X_k := X_k+Y
        end
    end
end
```

この結果，X_k が DFT の y_k（対応する周波数は $f_k=k/(NT_s)$）となる。
　なお，IDFT も全く同じアルゴリズムにより計算できる。DFT との相違は回転因子 w_N を w_N^{-1} に置き換え，最後に(1/N)を乗じることのみである。

【プログラム 11.1.1】

C言語のコンパイラによっては，複素数を扱えないものもある。ここでは，構造体によりcomplex型を定義する。

```
1 /*  FFT (Fast Fourier Transform) program */
2 /*      with Complex type supplied       */
```

```
 3
 4 #include <stdio.h>
 5 #include <math.h>
 6
 7 #define PI    3.14159265358979323
 8 #define NMAX  8192              /* NMAX = 2^13 =8192 */
 9
10 typedef struct{                /* 複素数を扱うためのcomplex型構造体を定義 */
11         double re;
12         double im;
13 }complex;
14
15 complex comp_set(double x, double y)   /* 複素数の実部と虚部のセット */
16 {
17      complex z;
18
19      z.re = x;
20      z.im = y;
21      return(z);
22 }
23
24 complex comp_add(complex x, complex y)   /* 複素数の加算 */
25 {
26      complex z;
27
28      z.re = x.re + y.re;
29      z.im = x.im + y.im;
30      return(z);
31 }
32
33 complex comp_sub(complex x, complex y)   /* 複素数の減算 */
34 {
35      complex z;
36
37      z.re = x.re - y.re;
38      z.im = x.im - y.im;
39      return(z);
40 }
41
42 complex comp_mul(complex x, complex y)   /* 複素数の乗算 */
43 {
44      complex z;
45
46      z.re = x.re * y.re - x.im * y.im;
47      z.im = x.re * y.im + x.im * y.re;
48      return(z);
49 }
50
51 complex comp_div(complex x, complex y)   /* 複素数の割算 */
52 {
53      complex z;
```

```
54      double   c;
55
56      c = y.re * y.re + y.im * y.im;
57      z.re = ( x.re * y.re + x.im * y.im)/c;
58      z.im = (-x.re * y.im + x.im * y.re)/c;
59      return(z);
60  }
61
62
63  void fft_time(complex *x, int n, int M, int idft)
64  {
65      complex w,wj,y;
66      float   dth;
67      int     i,j,k,l,p,bit,bitr,sign,np,np2,ng;
68
69      for(i=0;i<n;i++){
70                                  /* bit逆転: i ==> bitr */
71          bit =i;
72          bitr=0;
73          for(k=0;k<M;k++){
74             bitr = bitr << 1;          /* bitrを左に1ビットだけシフト */
75             bitr = bitr | (bit & 1);   /* bitの1ビット目をbitrの1ビット目にセット */
76             bit  = bit >> 1;           /* bitを右に1ビットだけシフト */
77          }
78
79          if(i<bitr){             /* i < bitr のときのみ, データを入れ換える */
80             y      = x[i];
81             x[i]   = x[bitr];
82             x[bitr]= y;
83          }
84      }
85
86      if(!idft){              /*  DFT */
87          sign=-1;
88      }else{                  /* IDFT */
89          sign= 1;
90      }
91
92      np=1;
93      for(p=1;p<=M;p++){
94          np *= 2;            /* p段目のデータ点数 : np = 2^p  */
95          np2 = np/2;
96          ng  =n/np;          /* p段目のグループ数 : ng = n/np */
97
98          dth =PI/np2;                    /* dtheta = 2PI/np */
99          w  = comp_set( cos(dth), sign*sin(dth) );
100         wj = comp_set( 1.0e0, 0.0e0 );
101
102         for(j=0;j<np2;j++){
103             for(i=0;i<ng;i++){
104                 k   =np*i+j;
```

```
105                 l   =k+np2;
106                 y   =comp_mul( wj, x[l] );
107                 x[l]=comp_sub( x[k], y );
108                 x[k]=comp_add( x[k], y );
109             }
110             wj = comp_mul( wj, w );
111         }
112     }
113
114     if(idft){                       /* IDFT */
115         for(i=0;i<n;i++){
116             x[i].re /= (float)n;
117             x[i].im /= (float)n;
118         }
119     }
120 }
121
122
123 void main(void)
124 {
125     complex x[NMAX];
126     float   time_smpl,sp,time[NMAX],freq[NMAX],single_re,single_im;
127     int   n,M,i,spectrum,idft=0;
128     char  fname[3][30];             /* ファイル名用領域          */
129     FILE  *fpin, *fpout;            /* ファイルポインターの宣言 */
130
131     printf("Fast Fourier Transform¥n");
132     printf("* Number of data  n=2^M : n, M ? ");     /* データ点数 : n = 2^M */
133     scanf("%d%d", &n, &M);
134                                     /* ファイル名の入力 */
135     printf("input  file name for data ==> ");
136     scanf("%s",fname[0]);           /* idft=0のときは後者(DFT)を, それ以外のときは前者(IDFT)を出力 */
137     printf("output file name for %s complex data  ==> ", idft?"IDFT":"DFT");
138     scanf("%s",fname[1]);
139     printf("output file name for %s spectrum data ==> ", idft?"IDFT":"DFT");
140     scanf("%s",fname[2]);
141                                         /* ファイルオープンとエラー処理 */
142     if( (fpin=fopen(fname[0],"r"))==NULL ){
143         printf("ファイルをオープンできません。¥n");
144         exit(1);
145     }
146
147     for(i=0;i<n;i++){
148         if(!idft){                                  /*  data for DFT */
149             fscanf(fpin, "%g%g",&time[i], &single_re);
150             single_im=0.0;
151         }else{                                      /*  data for IDFT */
152             fscanf(fpin, "%g%g%g",&time[i], &single_re, &single_im);
153         }
154         x[i]=comp_set(single_re, single_im);
155     }
```

```
156     fclose(fpin);
157     printf("¥n* input data were read from file: %s¥n",fname[0]);
158                                                     /* set of sampling time */
159     if(!idft){                      /*  DFT: time[] ... time */
160         time_smpl = time[1]-time[0];
161     }else{                          /* IDFT: time[] ... frequency */
162         time_smpl = 1.0/((time[1]-time[0])*n);
163     }
164
165     fft_time(x, n, M, idft);                            /* FFT */
166
167     printf("!!! %s finished !!!¥n", idft?"IDFT":"DFT");
168
169
170     if(!idft){                    /*  DFT: set frequency */
171         for(i=0;i<n;i++)
172             freq[i]=(float)i/(time_smpl * n);
173     }else{                        /* IDFT: set time */
174         for(i=0;i<n;i++)
175             freq[i]=(float)i*time_smpl;
176     }
177                                                 /* 変換結果を複素数でファイル出力 */
178     if( (fpout=fopen(fname[1],"w"))==NULL ){
179         printf("ファイルをオープンできません。¥n");
180         exit(1);
181     }
182     for(i=0;i<n;i++)
183         fprintf(fpout,"%13.6g¥t%13.6g¥t%13.6g¥n",freq[i],x[i].re,x[i].im);
184     fclose(fpout);
185     printf("* complex output were written to file: %s¥n",fname[1]);
186                                                 /* 出力スペクトルの種類の選択 */
187     printf("¥ninput spectrum No.: 0.phase, 1.amplitude, 2.power spectrum ? ");
188     scanf("%d",&spectrum);
189                                                 /* 変換結果を各種スペクトルでファイル出力 */
190     if( (fpout=fopen(fname[2],"w"))==NULL ){
191         printf("ファイルをオープンできません。¥n");
192         exit(1);
193     }
194     for(i=0;i<n;i++){
195         if(spectrum==0){
196             sp = atan2( x[i].im, x[i].re );
197         }else if(spectrum==1){
198             sp = sqrt( x[i].re*x[i].re + x[i].im*x[i].im );
199         }else if(spectrum==2){
200             sp = x[i].re*x[i].re + x[i].im*x[i].im;
201         }
202         fprintf(fpout,"%13.6g¥t%13.6g¥n",freq[i],sp);
203     }
204     fclose(fpout);
205     printf("* spectrum output were written to file: %s¥n",fname[2]);
206 }
```

【実行例】

時間 t[s]における変位 x[mm]が sin 波による合成（下式）により与えられるとする。

$$x(t) = a_0 + a_1 \sin(2\pi\nu_1 t) + a_2 \sin(2\pi\nu_2 t) + a_3 \sin(2\pi\nu_3 t) + a_4 \sin(2\pi\nu_4 t)$$

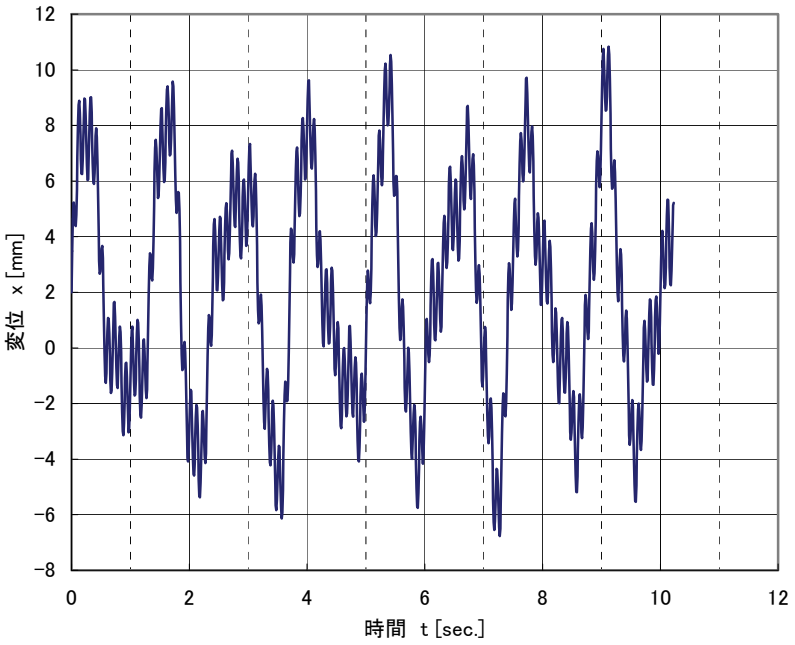
ここで，各振幅と周波数は以下のように設定する。

$$a_0 = 2$$
$$a_1 = 5 \qquad \nu_1 = 0.8$$
$$a_2 = 2 \qquad \nu_2 = 1.35$$
$$a_3 = 1 \qquad \nu_3 = 3$$
$$a_4 = 1.5 \qquad \nu_4 = 10$$

サンプリングタイム T_s を 0.01[s]として N=2^{10}=1024 個のデータを（エクセルなどにより）計算してテストデータとし，ファイル test1.txt に格納する。テストデータの時間と変位を以下に図示する。

ファイル test1.txt の中身（時間と変位）

0	2
0.01	3.489723
0.02	4.634185
0.03	5.217003
0.04	5.230414
0.05	4.875495
0.06	4.482857
0.07	4.384178
0.08	4.783693
0.09	5.67879
0.1	6.860047
0.11	7.990741
0.12	8.735432
:	:
:	:
10.2	3.649625
10.21	4.547769
10.22	5.13804
10.23	5.215783



DFT の結果を以下に示す。

```
-------------------------実行開始-----------------------
Fast Fourier Transform
* Number of data  n=2^M : n, M ? 1024 10                    ……入力データ点数1024は2の10乗
input  file name for data ==> test1.txt                     ……入力データファイル名
output file name for DFT complex data  ==> test1_complex.txt …出力データ(複素数)ファイル名
output file name for DFT spectrum data ==> test1_spectrum.txt …出力データ(スペクトル)ファイル名

* input data were read from file: test1.txt
!!! DFT finished !!!
* complex output were written to file: test1_complex.txt

input spectrum No.: 0.phase, 1.amplitude, 2.power spectrum ? 1 ……出力スペクトルの選択
* spectrum output were written to file: test1_spectrum.txt
-------------------------おしまい-----------------------
```

出力スペクトルファイル（test1_spectrum.txt）から振幅スペクトルをエクセルに読み込み，図示する。

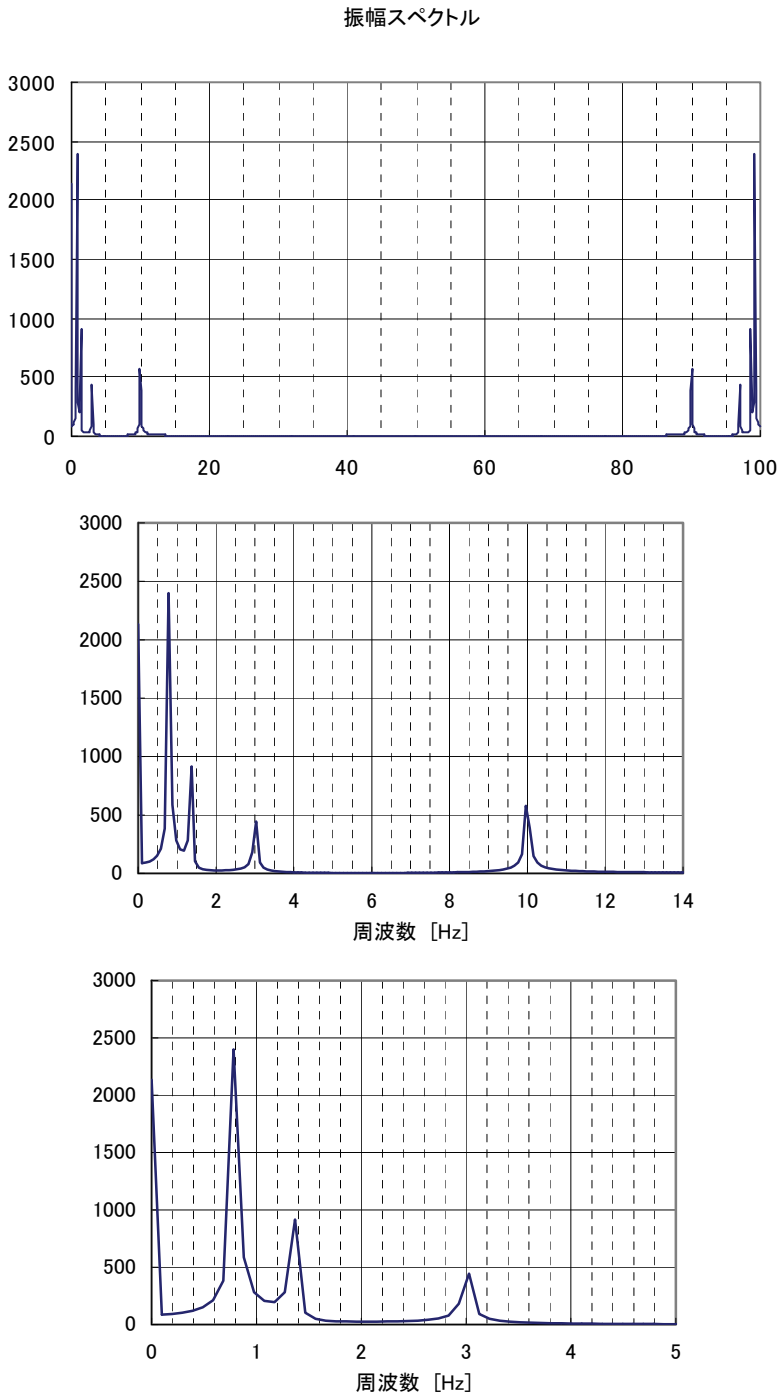
この変換におけるサンプリングタイムは T_s=0.01[s]，サンプリング点数は N=1024，サンプリング区間（サンプリングを行った全時間）は $T=T_s \times N$=10.24[s]である。

・したがって基本周波数は $f_1=1/T$=0.097656[Hz]であり，ファイルのデータをチェックすると出力周波数は基本周波数の倍数 kf_1（k=0,1,…N-1)であることが確認される。
・サンプリング周波数は $f_s=1/T_s$=100[Hz]であり，結果の周波数の範囲は[0, f_s)となっていることが確認される。
・振幅スペクトル図には，中央（$f=f_s/2$）に関して対称となる特性が現れている。
・サンプリング定理によると，ある波形に含まれる最大周波数を f_{max} とすると，その波形を復元するには $2f_{max}$ よりも高い周波数 f_s でサンプリングする必要がある。したがって計算された周波数の範囲[0, f_s)のうち意味を持つのは，[0, $f_s/2$)の範囲である。

グラフには，周波数 0.8，1.35，3，10 近傍にピークがあり，入力波が適正にフーリエ変換されたことがわかる。

ファイル test1_spectrum.txt の中身
（周波数と振幅）

0	2134.07
0.097656	87.6652
0.195312	92.7724
0.292969	102.508
0.390625	119.471
0.488281	149.884
0.585938	211.732
0.683594	384.064
0.78125	2398.48
0.878906	587.984
0.976562	281.205
1.07422	205.96
1.17188	195.394
1.26953	282.767
1.36719	916.352
1.46484	104.201
1.5625	49.6984
:	:
:	:
:	:
98.7305	282.767
98.8281	195.394
98.9258	205.96
99.0234	281.205
99.1211	587.984
99.2188	2398.48
99.3164	384.064
99.4141	211.732
99.5117	149.884
99.6094	119.471
99.707	102.508
99.8047	92.7724
99.9023	87.6652



確認のため，更にプログラムの 127 行における変数 idft の初期設定を idft=1 とし，IDFT の計算を行う。入力データには先ほどの DFT で得られた結果（複素数）のファイルを用いると，DFT での最初の入力波形が復元されるはずである。

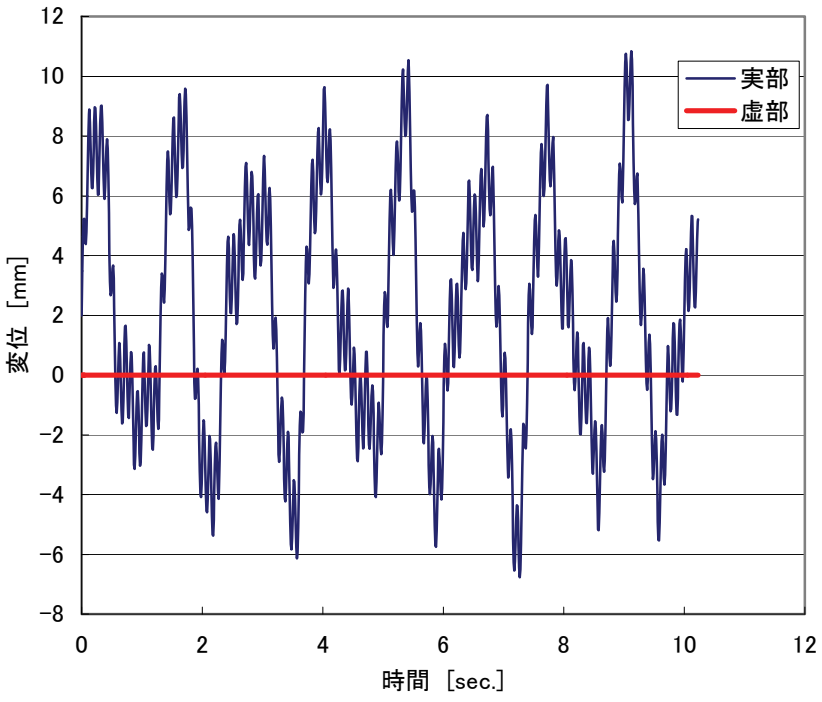
```
-------------------------実行開始-----------------------
Fast Fourier Transform
* Number of data  n=2^M : n, M ? 1024 10                    ……入力データ点数1024は2の10乗
input  file name for data ==> test1_complex.txt             ……入力データファイル名
output file name for IDFT complex data  ==> test1_IDFTcomplex.txt  …出力データ(複素数)ファイル名
output file name for IDFT spectrum data ==> test1_IDFTspectrum.txt
                                                            ……出力データ(スペクトル)ファイル名
* input data were read from file: test1_complex.txt
!!! IDFT finished !!!
* complex output were written to file: test1_IDFTcomplex.txt

input spectrum No.: 0.phase, 1.amplitude, 2.power spectrum ? 1
* spectrum output were written to file: test1_IDFTspectrum.txt
-------------------------おしまい-----------------------
```

DFT の結果データ（test1_complex.txt）を入力して IDFT を行った結果，複素数の実部には最初の入力波形がほぼ復元されており，複素数の虚部は全てほぼ零に近い値となっており，逆フーリエ変換が適正に行われたことがわかる。

ファイル test1_IDFTcomplex.txt の中身
（時間と変位：複素数の実部と虚部）

0	1.99999	-6.87E-07
0.01	3.48972	-6.62E-07
0.02	4.63418	-4.05E-07
0.03	5.217	-1.70E-08
0.04	5.23041	1.50E-07
0.05	4.87549	-1.01E-07
0.06	4.48285	-6.55E-07
0.07	4.38417	-1.30E-06
0.08	4.78369	-2.06E-06
0.09	5.67879	-2.74E-06
0.1	6.86005	-3.11E-06
0.11	7.99074	-3.02E-06
0.12	8.73543	-2.69E-06
:	:	:
:	:	:
:	:	:
10.2	3.64962	1.55E-06
10.21	4.54776	9.96E-07
10.22	5.13803	2.37E-07
10.23	5.21577	-3.98E-07



参考文献

高倉葉子：数値計算の基礎―解法と誤差―，コロナ社（2007）
Heath, Michael T.: Scientific Computing, An Introductory Survey, McGraw-Hill (2002)