流体計測特論（スペクトル解析）

機械工学科 飯田 明由

要　約

ランダムデータの解析の基礎となるのがフーリエ解析である．フーリエ解析は任意の周期関数を三角関数の級数として表す方法であり，微分方程式の解としてだけではなく，不規則現象の解析である確率論的な解析に拡張されている．工学的には乱流理論への拡張が最も成功した例であり，乱流の中の渦塊の運動を速度変動のスペクトルとして表した．前回までの講義ではKolmogorovの-5/3乗則などスペクトル解析によって得られた研究成果を解説してきたが，この章ではスペクトル解析を行うためのフーリエ解析，特に実際の計測・解析で利用されるFFT（Fast Fourier Transform）について解説する．

1. フーリエ級数

19世紀の数学者Fourier(1764－1830)がは周期関数を三角関数の級数として表す方法を考案した. 今日フーリエ級数と呼ばれるこの方法は発表当時は学会から正式に認められたものではなかったが, 現在では微分方程式の解法手段としてだけなく, 信号波形, 特にランダム信号の解析に広く利用されている. （Fourierは熱伝導問題を解くためにフーリエ級数を利用した. ）

関数x(t)は区間(-T/2, T/2)を基本とする周期Tの周期関数とする.

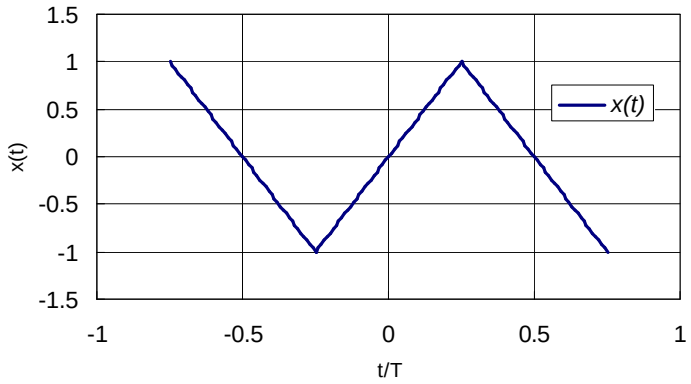


図1　三角系列関数x(t)

$$x(t)=\begin{cases}-2-\dfrac{4}{T}t & \left(-\dfrac{3T}{4}<t<-\dfrac{T}{4}\right)\\ \dfrac{4}{T}t & \left(-\dfrac{T}{4}\le t\le\dfrac{T}{4}\right)\\ 2-\dfrac{4}{T}t & \left(\dfrac{T}{4}<t<\dfrac{3T}{4}\right)\end{cases}$$

この関数を整数個の波を含む正弦関数の過重和で表現するとする. x(t)は奇関数なので, 余弦関数の項は考えなくとも良い.

$$x(t) = b_1 \sin\frac{2\pi t}{T} + b_2 \sin\frac{4\pi t}{T} + b_3 \sin\frac{6\pi t}{T} + \ldots + b_n \sin\frac{2n\pi t}{T} + \ldots$$

両辺にsin(2*npt*/*T*)を掛けて, 積分すると

$$\int_{-T/2}^{T/2} x(t) \sin\frac{2n\pi t}{T}\,dt = b_1 \int_{-T/2}^{T/2} \sin\frac{2\pi t}{T} \cdot \sin\frac{2n\pi t}{T}\,dt + b_2 \int_{-T/2}^{T/2} \sin\frac{4\pi t}{T} \cdot \sin\frac{2n\pi t}{T}\,dt + \ldots + b_n \int_{-T/2}^{T/2} \left(\sin\frac{2n\pi t}{T}\right)^2 dt + \ldots$$

ここで,

$$\sin \int_{-T/2}^{T/2} \sin\frac{2m\pi t}{T} \cdot \sin\frac{2n\pi t}{T}\,dt \begin{cases} = 0 & (m \neq n) \\ = \dfrac{T}{2} & (m = n) \end{cases}$$

であるから, *m*=*n*の時以外は積分値は0となる. このような関係を持つ関数列を直交関数(orthogonal functions)と呼ぶ. *m*=*n*のとき積分値が1となる場合, 正規直交関数(orthonormal functions)という

$$\int_{-T/2}^{T/2} x(t) \sin\frac{2n\pi t}{T}\,dt = 2\left[\int_0^{T/4} \left(\frac{4}{T}t\right) \sin\frac{2n\pi t}{T}\,dt + \int_{T/4}^{T/2} \left(2 - \frac{4}{T}t\right) \sin\frac{2n\pi t}{T}\,dt\right]$$

$$= \frac{2T}{(n\pi)^2} \int_0^{n\pi/2} \zeta \sin\zeta\,d\zeta - \frac{2T}{(n\pi)^2} \int_{n\pi/2}^{n\pi} \zeta \sin\zeta\,d\zeta + \frac{2T}{n\pi} \int_{n\pi/2}^{n\pi} \zeta \sin\zeta\,d\zeta$$

$$\begin{cases} = \dfrac{4T}{(n\pi)^2}(-1)^{n+1} & (n = 1,3,5,\ldots) \\ = 0 & (n = 2,4,6,\ldots) \end{cases}$$

したがって,

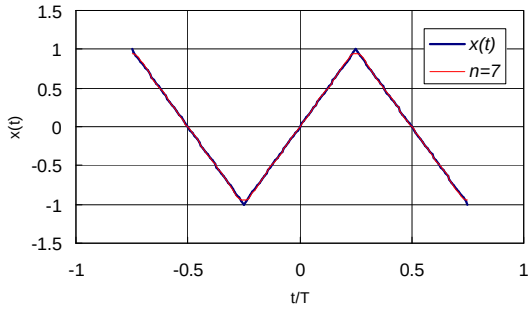
$$b_n = \frac{2}{T} \int_{-T/2}^{T/2} x(t) \sin\frac{2n\pi t}{T}\,dt \qquad \begin{cases} = \dfrac{8T}{(n\pi)^2}(-1)^{n+1} & (n = 1,3,5,\ldots) \\ = 0 & (n = 2,4,6,\ldots) \end{cases}$$

以上の結果から先に与えた周期関数x(t)は

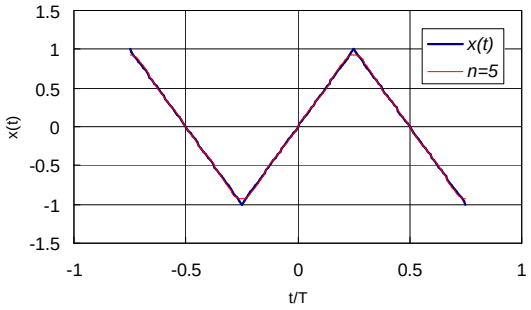
$$x(t) = \frac{8}{\pi^2}\left\{\sin\frac{2\pi t}{T} - \frac{1}{3^2}\sin\frac{6\pi t}{T} + \frac{1}{5^2}\sin\frac{10\pi t}{T} - \frac{1}{7^2}\sin\frac{14\pi t}{T} + \ldots\right\}$$

と正弦関数の和として表すことができる.

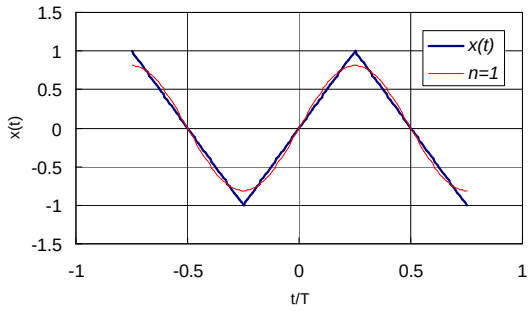
図2(a)～(d)は次数*n*を増やした場合の関数*x*(*t*)の近似値である. 次数*n*を増やすと, 関数*x*(*t*)を精度良く再現できることがわかる.



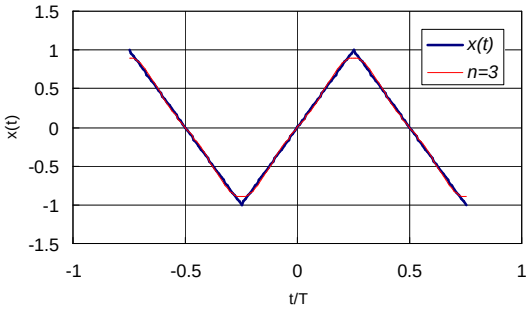
(a) *n*=1



(b) *n*=3



(c) *n*=5



(d) *n*=7

図2．2　三角級数による三角系列関数の近似

一般に区間[-T/2, T/2]を一周期とする周期関数のフーリエ級数は

$$x(t)=\frac{a_0}{2}+\sum_{n=1}^{\infty}\left(a_n\cos\frac{2n\pi t}{T}+b_n\sin\frac{2n\pi t}{T}\right)$$

$$\begin{cases} a_n=\dfrac{2}{T}\displaystyle\int_{-T/2}^{T/2}x(t)\cos\frac{2n\pi t}{T}\,dt \\ b_n=\dfrac{2}{T}\displaystyle\int_{-T/2}^{T/2}x(t)\sin\frac{2n\pi t}{T}\,dt \end{cases}$$

となる.

フーリエ級数を用いて関数を表すには項数を非常に多くとる必要があり, 一般に20から30項が必要である. しかし, コンピュータが発達した今日では級数の項数を増やすことはそれほど困難ではなく, また, 後述するFFTにより高速に計算することが可能である.

フーリエ変換で矩形波を表そうとすると, 項数の増加とともに不連続点において振動が発生する. これをGibbs現象という.

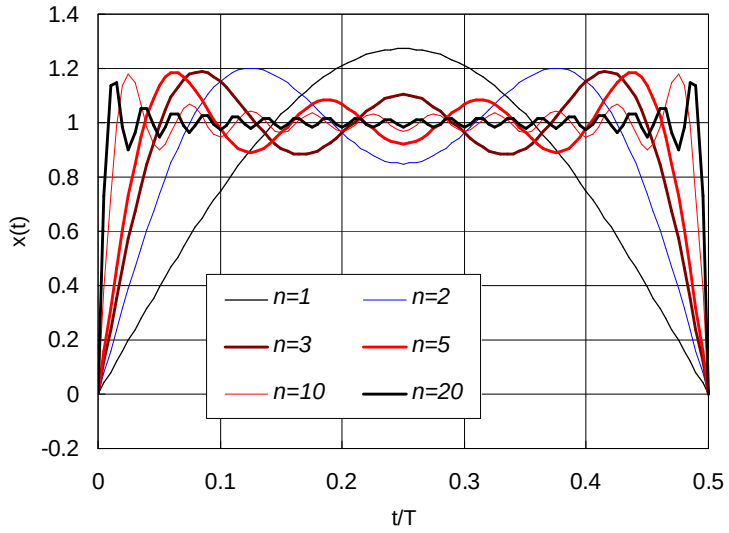


図3　不連続点を持つ関数のフーリエ級数

2. フーリエ変換

基本周波数区間Tが有限値ならば, T/nは第n次高調波の周期であり, その逆数は周波数f_nであり,

$$f_n = \frac{n}{T}$$
$$\delta f = \frac{1}{T}$$

と表すことができる.
三角関数と指数関数の関係

$$\cos\theta = \frac{\left(e^{i\theta} + e^{-i\theta}\right)}{2}$$
$$\sin\theta = \frac{\left(e^{i\theta} - e^{-i\theta}\right)}{2}$$

から,

$$x(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} A_n \exp\left(\frac{i2n\pi t}{T}\right) + \sum_{n=1}^{\infty} B_n \exp\left(-\frac{i2n\pi t}{T}\right)$$
$$\begin{cases} A_n = \dfrac{a_n - ib_n}{2} = \dfrac{1}{T}\displaystyle\int_{-T/2}^{T/2} x(t)\exp\left(-\frac{i2n\pi t}{T}\right)dt \\ A_n = \dfrac{a_n + ib_n}{2} = \dfrac{1}{T}\displaystyle\int_{-T/2}^{T/2} x(t)\exp\left(\frac{i2n\pi t}{T}\right)dt \end{cases}$$

となるので, 周波数を用いてフーリエ級数を表すと

$$x(t) = \lim_{T\to\infty} \sum_{n=-\infty}^{\infty} \left[\frac{1}{T} \int_{-T/2}^{T/2} x(t) \exp(-i2\pi nt/T) dt \right] \exp(i2\pi nt/T)$$

$$= \lim_{T\to\infty} \sum_{n=-\infty}^{\infty} \delta f \left[\int_{-T/2}^{T/2} x(t) \exp(-i2\pi f_n t) dt \right] \exp(i2\pi f_n t)$$

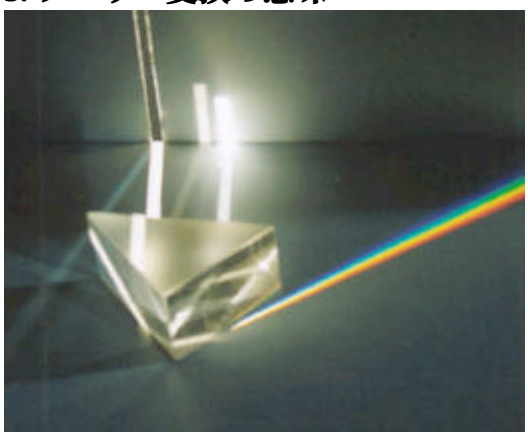
となる. したがって, 区間Tを無限大に置き換えれば,

$$x(t) = \int_{-\infty}^{\infty} X(f) \exp(i2\pi ft) df$$

$$X(f) = \int_{-\infty}^{\infty} x(t) \exp(-i2\pi ft) dt$$

となる. これをフーリエ積分あるいはフーリエ変換と呼ぶ.

3. フーリエ変換の意味



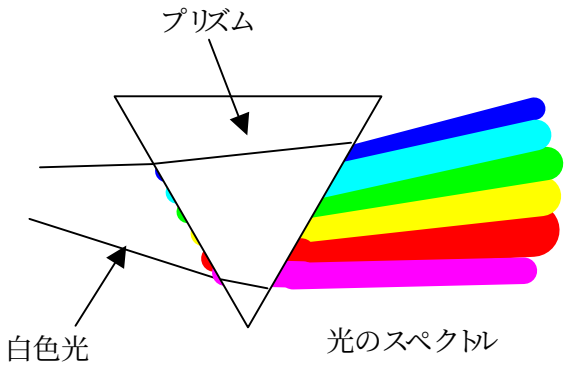


図4　プリズムを通過する白色光

暗室内に白色光（太陽光）を入れ, 三角プリズムに当てると光の波長ごとに屈折率が異なるため, プリズムの背後に波長の長さ順に7色の光の帯が映し出される.
プリズムは白色光の中に含まれる各波長の光の成分毎に光を分解する役割をしている. 分光された光の強さ（明るさ）は白色光に含まれる各波長の光の強さを表している. このようにプリズムは光の波長と強さを分離する装置である. さて, フーリエ変換では信号は周波数と振幅（強度）に分解されていることを思い出そう. すべての光の波長が集まった白色光では各波長の成分の持つ光の色がわからなくなる. これはちょうど, ランダム信号には特徴がないことに似ている. ランダム信号は副速な種々の波長の波の合成と考えられる. 不規則波形はフーリエ変換という数学的プリズムを介して, 信号をスペクトルに分解することに他ならない.

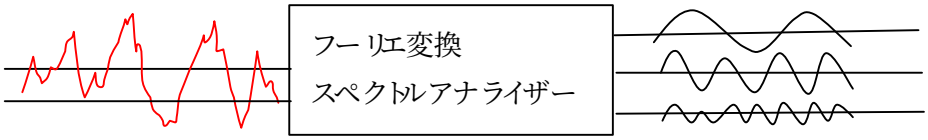


図5　フーリエ変換による信号の周波数分解

フーリエ成分$X(f)$は周期fの波の振幅を表し, $|X(f)|^2$はその強さ, エネルギーを表している. 光の分光との相似性から周波数fの成分波のエネルギーの分布関係$|X(f)|^2$をエネルギースペクトルと定義する. 有限区間Tにおける信号や周期関数の場合, $|X(f)|^2$も有限と考えられるので, エネルギーも有限となり, 物理的にも理解しやすい. しかし, 区間Tが無限の場合, 単位時間当たりの平均エネルギーをとって, パワースペクトル密度(Power Spectrum Density Function)$P(f)$を定義する.

$$P(f)=\lim_{T\to\infty}\left[\frac{1}{T}|X(f)|^2\right]=\lim_{T\to\infty}\left[\frac{1}{T}X(f)X^*(f)\right]$$

関数x(t)が確率過程の場合は, $|X(f)|^2$の期待値についてのパワースペクトル

$$P(f)=\lim_{T\to\infty}E\left[\frac{1}{T}|X(f)|^2\right]$$

を定義することができる.

$$\overline{x^2}=\lim_{T\to\infty}\frac{1}{T}\int_{-T/2}^{T/2}x^2(t)dt=\int_{-\infty}^{\infty}P(f)df$$

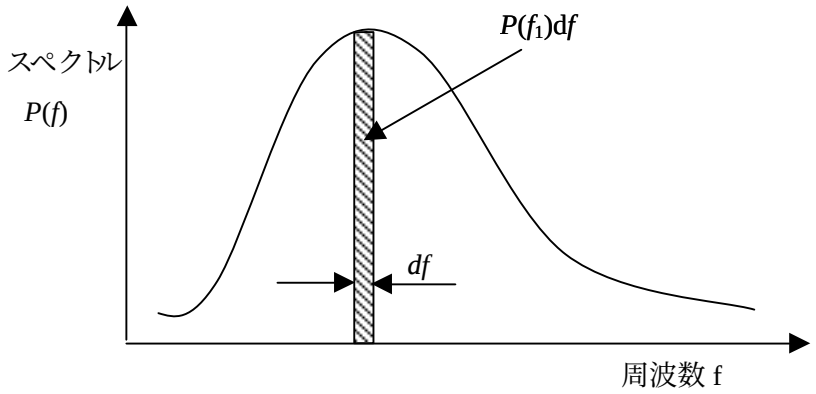


図6 スペクトルの意味とエネルギーとの関係

4. Fast Fourier Transform

フーリエ変換は周期関数を三角関数の級数として表現するだけなく, ランダム信号の特徴を表現することが可能であり, 工学的に有効な数学ツールである, しかし, 信号を20から30項の級数に展開することは容易ではなく, コンピュータが発達する以前は, 電気的なバンドパスフィルターにより信号を (まさにプリズムで分解するように)周波数毎に分析していた. 1965年にCooleyとTukeyは計算時間を驚異的に短縮する方法を開発した.

長さNの時系列信号$x_0,x_1\ldots\ldots x_{N-1}$の有限余弦フーリエ変換及び有限正弦フーリエ変換は以下のように表すことができる.

$$\begin{cases} A_k = \dfrac{2}{N}\displaystyle\sum_{j=0}^{N-1} x_j \cos\frac{2\pi kj}{N} & (k = 0,1,\ldots N/2) \\ B_k = \dfrac{2}{N}\displaystyle\sum_{j=0}^{N-1} x_j \sin\frac{2\pi kj}{N} & (k = 1,2,\ldots N/2-1) \end{cases}$$

ここで

$$\left.\begin{aligned} C_0 &= A_0/2 \\ C_{N/2} &= A_{N/2} \\ C_k &= (A_k - iB_k)/2 \\ C_{N-K} &= (A_k + iB_k)/2 \end{aligned}\right\} \quad 0 < k < \frac{N}{2}$$

とすると,

$$C_k = \frac{2}{N}\sum_{j=0}^{N-1} x_j \exp\left(-i\frac{2\pi kj}{N}\right) \quad (j = 0,1,2\ldots N-1)$$

逆変換は

$$x_j = \sum_{k=0}^{N-1} C_k \exp\left(i\frac{2\pi kj}{N}\right) \quad (j = 0,1,2\ldots N-1)$$

である.
N＝8のとき,

$$W = \exp\left(i\frac{2\pi}{8}\right) = \cos\frac{\pi}{4} + i\sin\frac{\pi}{4}$$

とおけば,$W^2=i$,$W^4=-1$, $W^5=-W$, $W^6=--i$,$W^7=-W^3$より

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & W & W^2 & W^3 & W^4 & W^5 & W^6 & W^7 \\ 1 & W^2 & W^4 & W^6 & W^8 & W^{10} & W^{12} & W^{14} \\ 1 & W^3 & W^6 & W^9 & W^{12} & W^{15} & W^{18} & W^{21} \\ 1 & W^4 & W^8 & W^{12} & W^{16} & W^{20} & W^{24} & W^{28} \\ 1 & W^5 & W^{10} & W^{15} & W^{20} & W^{25} & W^{30} & W^{35} \\ 1 & W^6 & W^{12} & W^{18} & W^{24} & W^{30} & W^{36} & W^{42} \\ 1 & W^7 & W^{14} & W^{21} & W^{28} & W^{35} & W^{42} & W^{49} \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & W & i & W^3 & -1 & -W & -i & -W^3 \\ 1 & -i & -1 & -i & 1 & i & -1 & -i \\ 1 & W^3 & -i & -W & -1 & W^3 & -i & -W \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -W & i & -W^3 & -1 & W & -i & W^3 \\ 1 & -i & -1 & i & 1 & -i & -1 & i \\ 1 & -W^3 & -i & -W & -1 & W^3 & i & W \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \end{bmatrix}$$

行列の列を入れ替えて, 偶数項と奇数項に分解する.

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} D_0 + D_4 \\ D_1 + WD_5 \\ D_2 + iD_6 \\ D_3 + W^3 D_7 \\ D_0 - D_4 \\ D_1 - WD_5 \\ D_2 - iD_6 \\ D_3 - W^3 D_7 \end{bmatrix}$$

$$\begin{bmatrix} D_0 \\ D_1 \\ D_2 \\ D_3 \end{bmatrix} = \mathbf{T} \begin{bmatrix} C_0 \\ C_2 \\ C_4 \\ C_6 \end{bmatrix}, \qquad \begin{bmatrix} D_4 \\ D_5 \\ D_6 \\ D_7 \end{bmatrix} = \mathbf{T} \begin{bmatrix} C_1 \\ C_3 \\ C_5 \\ C_7 \end{bmatrix}$$

$$\mathbf{T} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}$$

となる. フーリエ変換は簡単な加算計算に置き換えられる.
標本数がNのとき, 元のフーリエ変換に必要な演算数はN2のオーダーである. 一方, 上記の演算方法の場合, N= P･Qとおくと, 演算回数は PN (logpN) となるので, もとの方法と比較すると演算回数比は $\frac{P}{\log_2 P}\frac{\log_2 N}{N}$ となる.　Nが十分大きければP=4のとき最も計算効率がよくなる.

実際の計算手順：

(1) データ数の決定：データ数Nを2のべき乗に選ぶ　$N=2^p$
データが2のべき乗になっていない場合は，データの一部を削るか，データの末尾に0を加えていく．

(2) ウィンドウ関数：フーリエ変換は本来，無限に長い区間Tに関する積分であるから，有限長のデータではデータの端部でデータが周期的に変化しないための誤差を生じる．データの端部における誤差を無くすために，ウィンドウ関数を用いる．データの始めと終わりをなだらかに0に近づけることで，無限に長い領域に0というデータがあるように修正する．
一般にはHanningWindowが使われることが多い．（連続信号）
パルス信号の場合はRectangular Windowを用いる．

$$x(j) = x(t)Wc(t)$$

Hanning　　$Wc = 0.5\left\{1 - \cos\left(\dfrac{2\pi i}{N-1}\right)\right\}$　　$\mu = \left[\dfrac{1}{T}\int_{-T/2}^{T/2} W^2 dt\right] = \dfrac{2}{3}$

Rectangular　　$Wc = 1$

(3) パワースペクトルの算出：フーリエ係数からパワースペクトルを推定する．

$$\tilde{P}\left(\frac{k}{T}\right) = \frac{\Delta t}{N}\left[A_k{}^2 + B_k{}^2\right]\mu$$

(4) 平滑化：一般にスペクトル推定値は推定誤差が大きく，激しく振動する．スペクトルの分散を抑えるためには，平滑化処理が必要である．

統計平均　同一条件でM回の平均を取る．
分割平均　データをl個の部分に分割して各区間のスペクトルの平均値を取る．
周波数平滑　周波数空間での重み平均を取る．

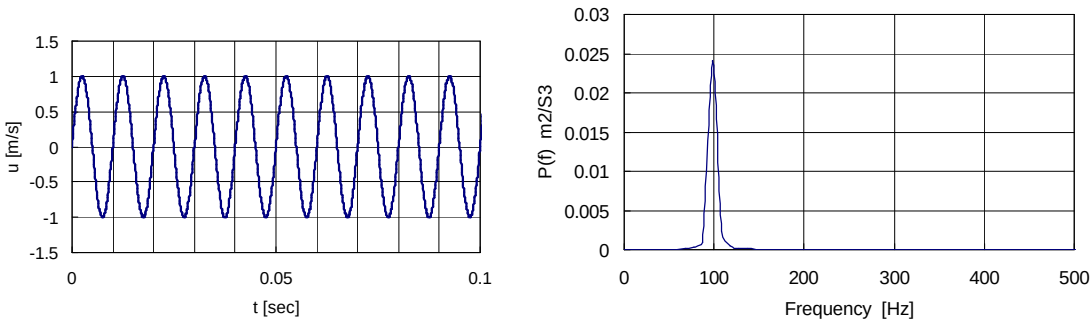
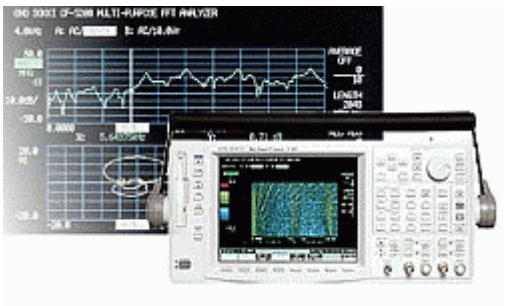


図7 スペクトルの測定例

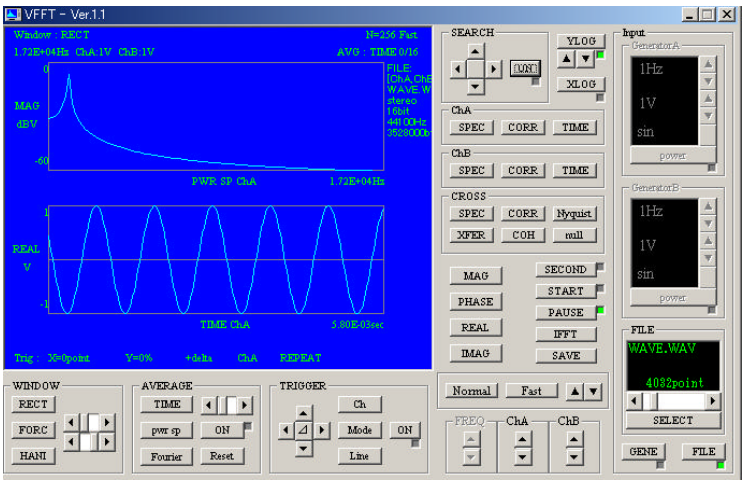
FFTソフト等

FFT解析用のハードウェアとしては，かつては小野測器製のCFシリーズが広く使われていたが，現在では，パソコンで直接信号を取得し，ソフトウェアでFFT処理する方法が一般的である．FFTのプログラム自体は簡単なので，A/Dコンバータがあれば自作も可能である．

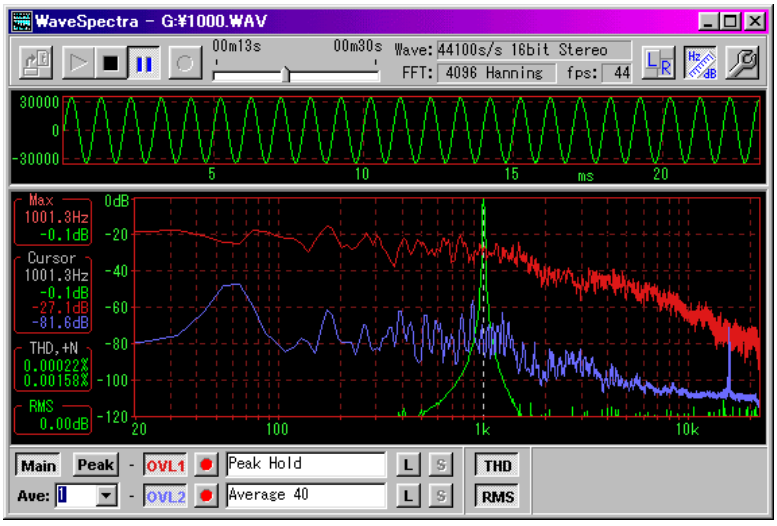


各種のフリーソフトも作られているので，実験データの整理等には，これらのソフトを利用しても良い．
VirtualFFTアナライザは小野測器のCFシリーズのインターフェースを模擬した教育用ソフトでFFTアナライザ（波形解析装置）の操作方法や概念を学習することができる．以下のサイトからダウンロードが可能である．

http://rd.vector.co.jp/soft/win95/edu/se056666.html



WINDOWSの音源ボードと組み合わせて利用できるWS及びWG（信号発生器）を使うと音の周波数とスペクトルの関係がわかりやすい．このソフトも書きのサイトからダウンロードできる．
http://www.ne.jp/asahi/fa/efu/soft/ws/ws.html



```
#include        <stdio.h>
#include        <math.h>
#define MAXDATA 10000
#define NF      4
#define DT      0.0004
#define N       2048
#define WT      1.0/0.875
#define PI      3.14156

int ipow2(),inv2pow(),atoi(),zero_y(),exit();
int sines_2(),fft(),saw_wave();
double atof();

void main(argc,argv)
int argc;
char *argv[];
{
        FILE  *fp;
        int n,l,i,mf,nf,sample;
        char filename[80],temp[256];
        double  x[MAXDATA],  y[MAXDATA];
        double  sx1[MAXDATA], sy1[MAXDATA], sp1[MAXDATA];
        double  sx2[MAXDATA], sy2[MAXDATA], sp2[MAXDATA];
        double  sx[MAXDATA],  sy[MAXDATA];
        double  fsp1[MAXDATA], fsp2[MAXDATA];
        double  fsx[MAXDATA],  fsy[MAXDATA];
        double
        sumxy[MAXDATA],sump1[MAXDATA],sump2[MAXDATA];
        double  dummyx[MAXDATA];
        double  dummyy[MAXDATA];
        double
        ave,sum,dt,TN,OBT,FRE[MAXDATA],t_tmp[MAXDATA];

        switch(argc)
        {
        case 0:
                exit(0);
        break;

        case 1:
                fp=fopen("time.d","w");
                dt=DT;
                l=inv2pow(N);
                n=ipow2(l);
                nf=NF;
                for(i=0;i<n;i++)
                {


        dummyx[i]=sin(2.0*PI*500.0*dt*(double)i) ;
                        dummyy[i]=0.0;

        fprintf(fp,"%e %e\n",(double)i*dt,dummyx[i]);
                }
                fclose(fp);

                strcpy(filename,"fft.d");
                break;

        default:
        /*
        USAGE:  fft  filename  Sampling_Frequency  FFT_Length
FillterLength
        */
        fp=fopen(argv[1],"r");
        nf=NF;
        i = 0;
        for ( ;; )
        {
                if ( fgets(temp,78,fp)==NULL )
                {
                        fclose(fp);
                        break;
                }
                if  (  temp[0]  ==  ';'  ||  temp[0]  <=  31  ||
temp[1] == 0 ) continue;
                if ( temp[0] == 'b' ) break;
                sscanf(temp,"%lf %lf",&t_tmp[i],&dummyx[i]);
                i++;
        }
        n=i;
        l=inv2pow(n);
```

```
        n=ipow2(l);

        dt=t_tmp[1]-t_tmp[0];
        break;
        }


        for(i=0;i<n;i++)
        {
                x[i]=dummyx[i];
                y[i]=0.0;
        }

        /*hanning(x,y,n);*/
        fft(x,y,l,-1.0);

        /*
                printf("decomposition FFT data coresspond to
original\n");
        */

        for(i=0;i<n;i++)
        {
                sp1[i] =  (x[i]*x[i]+y[i]*y[i]);
        }

        OBT=1.0/(dt*(double)n);

        fp=fopen(argv[2],"w");
        for(i=0;i<n;i++)
        {
                FRE[i]=OBT*(double)i ;

        fprintf(fp," %e %e %e %e\n",FRE[i],sp1[i]/OBT*WT,x[i
],y[i]);
        }
        fclose(fp);

        fft(x,y,l,1.0);
        for(i=0;i<n;i++)
        {
                printf("%e %e\n",dt*(double)i,x[i]);
        }

}

int hanning(x,y,n)
double *x,*y;
int n;
{
        int i;

        for(i=0;i<n;i++)
        {
                x[i]                *=                0.5*(1.0-
cos(2.0*PI*(double)i/(double)n));
                y[i]                *=                0.5*(1.0-
cos(2.0*PI*(double)i/(double)n));
        }

}

int filter(z,fz,n,nf)
double *z,*fz;
int n,nf;
{
        int i,k,k1,k2;
        int mf,m2,m;
        double sum;
        m2=nf*2;

        for(k=mf=0;k<n/2;k+=m2,mf++)
        {
                sum=0;
                for(i=0;i<nf;i++)
                {
                        k1=k-i;
                        k2=k+i;

                        if(k1<=0)
                                k1= -k1;
```

```
                        if(k2>=n/2)
                                k2= n/2-(k2-n/2);


                        sum+=(z[k1]+z[k2]);

                }
                fz[mf]=sum/(double)nf;

        }
        return mf;
}

int fft(x,y,l,f)
double *x,*y,f;
int l;
{
        int i,i0,i1,j,l1,n,ns,n1,k,ipow2();
        double s,c,s1,c1,sc,x1,y1,t;

        n=ipow2(l);
        n1=n/2;
        sc=PI;

        j=0;
        for(i=0;i<n-1;i++)
        {
                if(i<=j)
                {
                        t=x[i];
                        x[i]=x[j];
                        x[j]=t;

                        t=y[i];
                        y[i]=y[j];
                        y[j]=t;
        }
                k=n/2;
                while(k<=j)
                {
                        j=j-k;
                        k /= 2;
                }
                j=j+k;
  }

  ns=1;
  while(ns<=n/2)
  {
        c1=cos(sc);
        s1=sin( f * sc);
        c=1.0;
        s=0.0;
        for(l1=0;l1<ns;l1++)
        {
                for(i0=l1;i0<n;i0+=(2*ns))
                {
                        i1              =i0+ns;
                        x1              =x[i1]*c-y[i1]*s;
                        y1              =y[i1]*c+x[i1]*s;
                        x[i1]   =x[i0]-x1;
                        y[i1]   =y[i0]-y1;
                        x[i0]   =x[i0]+x1;
                        y[i0]   =y[i0]+y1;
                }
                t=c1*c-s1*s;
                s=s1*c+c1*s;
                c=t;
        }
        ns=2*ns;
        sc=sc/2.0;
  }

  if(f<0.0)
  for(i=0;i<n;i++)
  {
        x[i] /= (double)n;
        y[i] /= (double)n;
  }
}

int zero_y(y,n)
```

```
double *y;
int n;
{
        while(n!=0)
        {
                *y++=0.0;
                n--;
        }
}

int ipow2(l)
int l;
{
        int n;
        n=1;
        while (l !=0)
        {
                n *= 2;
                l--;
        }
        return n;
}

int inv2pow(n)
int n;
{
        int l;

        l= -1;
        while(n!=0)
        {
                n/=2;
                l++;
        }
        return l;
}
```