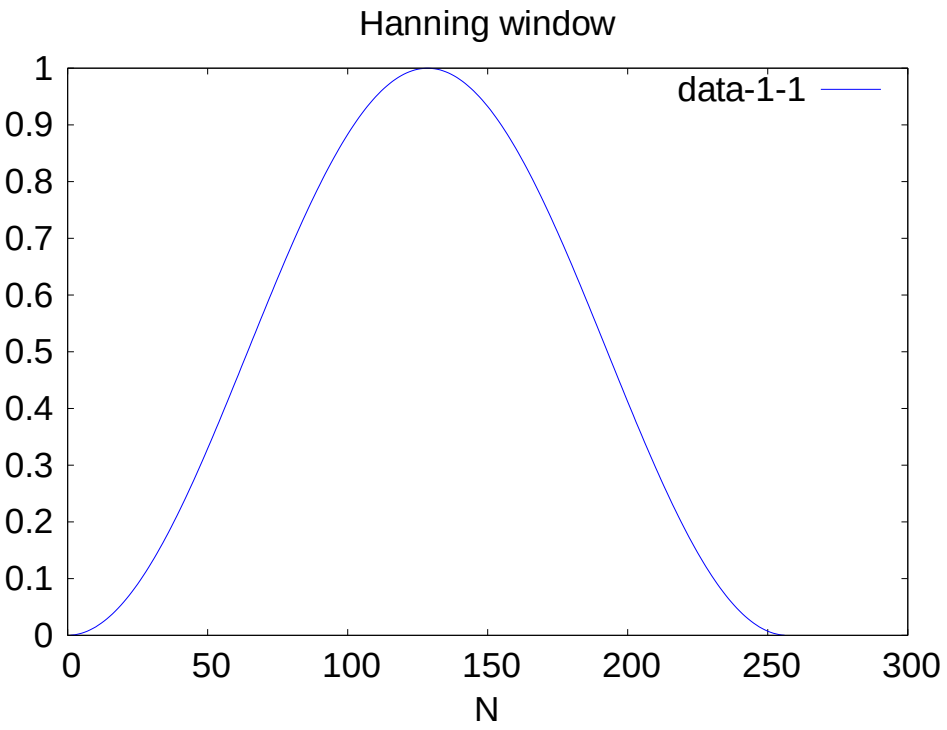
５．窓関数の影響

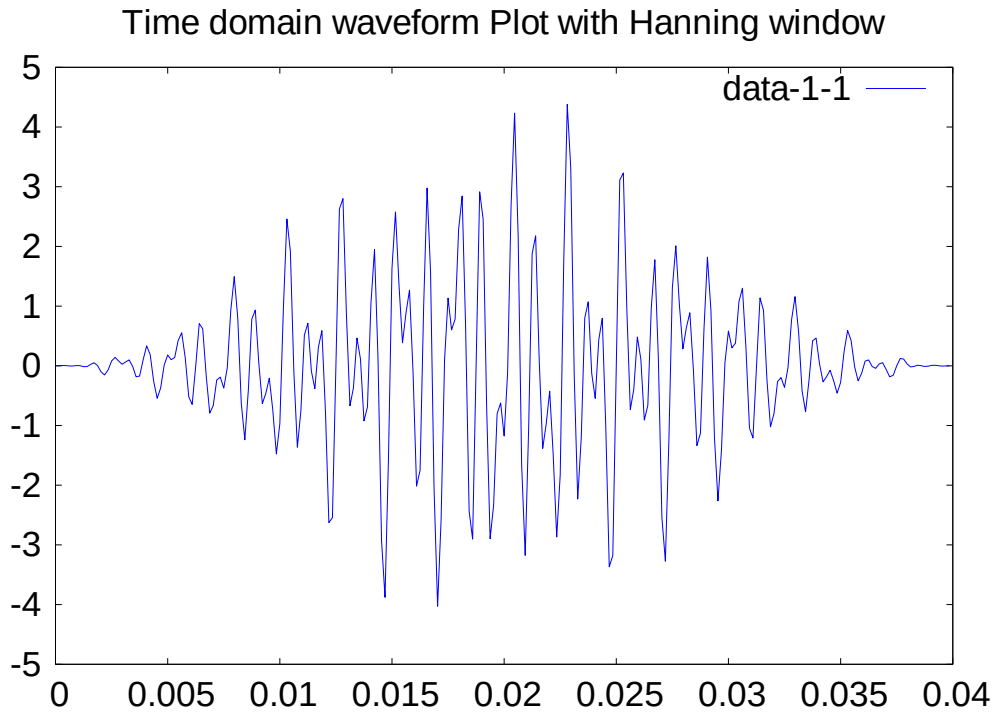
これまでは窓を考慮しない場合（矩形窓の場合）を計算して来ました。デジタルフーリエ変換では N 個のデータを無限に繰り返される N 個の周期データとみなして計算します。この場合 N 個の最初のデータと最後のデータに段差があると、これを元の信号にはなかった急峻な周期信号成分として扱い、周波数刻み（Fs/N）の高調波成分が誤差として多数生じます。この誤差を減らすために両端で信号が小さくなる窓関数を用います。最もよく使用されるのは Hanning 窓です。

```
N=256;
k=[1:N];
w=0.5*(1-cos(2*PI*k/(N+1)));                    //これが Hanning 窓の形を決める式
mgplot_reset(1);
mgplot_title(1,"Hanning window");
mgplot_xlabel(1,"N");
mgplot_ylabel(1,"Amplitude");
mgplot(1,k,w);
```



この窓関数を元の N 個のデータに掛けると

```
w=hanning(256);     //式を書く代わりに Hanning 窓の関数も利用できます。
xnw=w*xn;           //MaTX では配列の掛け算は要素ごとの掛け算になります。
mgplot_reset(1);
mgplot_title(1,"Time domain waveform Plot with Hanning window");
mgplot(1,t,xnw);        //窓関数を掛けたデータをグラフにします。
```



窓関数を掛けると、偽の高調波成分の影響を抑えられますが、信号が小さくなるので、振幅の補正が必要です。矩形窓に比べてどれだけ小さくなるか比率を求めるには、窓形状の各要素の振幅の和を求め、矩形窓のそれとの比を計算します。（窓の面積の比と言ってもよいでしょう。）矩形窓の各要素の振幅は全部 1 なので、その N 個の和はNになります。

Amplitude Correction Factor=sum(w)/N

Hanning 窓で N=256 の場合、Amplitude Correction Factor(ACF) =0.501953 になりました。すなわち振幅は 0.5 倍になるので、fft の計算の時、0.5 で割って補正します。

次に窓関数によるパワー減衰率を計算します。窓の各要素の自乗の和を求め、矩形窓の値に対する比率を計算します。矩形窓の場合、各要素の振幅が 1 なので、自乗和も N になります。

よって

Power Correction Factor=sum(w*w)/N

Hanning 窓で N=256 の場合、Power Correction Factor(PCF)=0.376465 になりました。すなわちパワーは 0.3765 倍になるので、パワースペクトルを 0.3765 で割って補正すればよいことになります。しかし、既に ACF によって振幅補正済みの fft 結果を使ってパワースペクトルを計算したのであれば、補正量は PCF/ACF^2= sum(w*w)/(sum(w))^2*N となります。
この値は Hanning 窓で N=256 の場合、0.376465/(0.501953)^2=1.49416 になりました。
パワー計算では振幅補正済みパワースペクトルの和を 1.5 で割って補正する必要があるわけです。
振幅を補正した後に更に必要になる補正が何を意味するかというと、等価帯域幅の補正です。矩形窓の場合、一つのサンプルの帯域幅は周波数分解能=周波数刻み=Fs/N に等しいのですが、Hanning 窓ではこれが 1.5 倍に広がっていると解釈できます。この補正量を米国の Web 情報に

倣って ENBW_cf（Equivalent Noise Bandwidth correction factor）と書きました。

では MaTX を使って確認してみましょう。

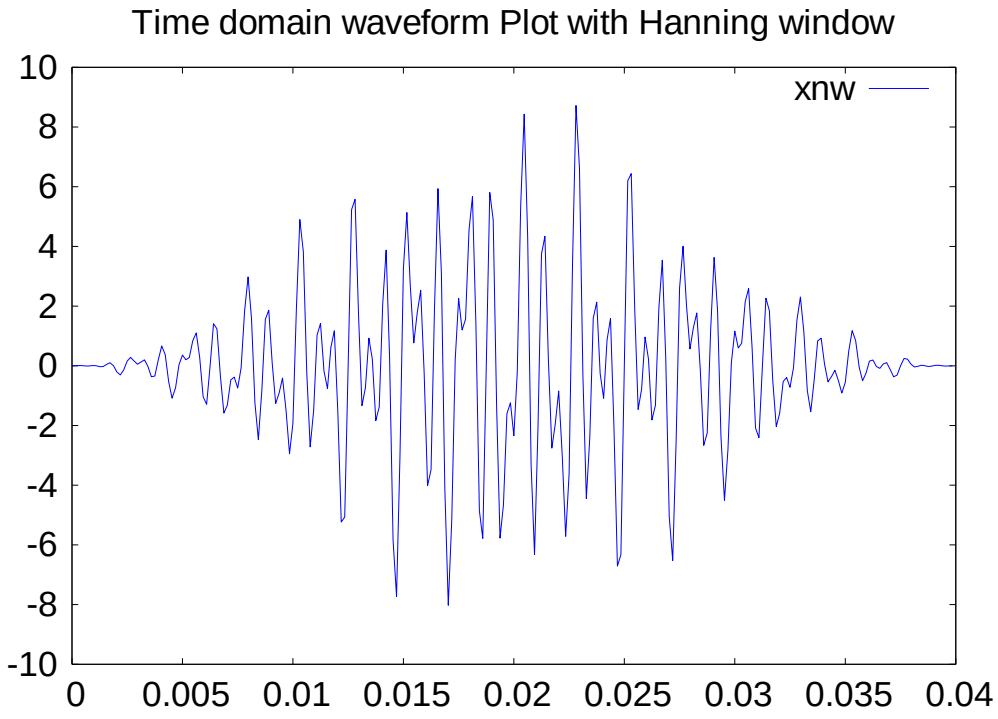
```
Fs=6400.0;              // sampling frequency
N=256;                  //Sample number
Freq_step=Fs/N
t=[0:N-1]/Fs;           //サンプリング時刻列
f=Freq_step*[0:N-1];    //周波数刻みの列
k=[1:N];
w=0.5*(1-cos(2*PI*k/(N+1)));    //Hanning 窓

sum_sq_w=sum(w*w)
sq_sum_w=(sum(w))^2
Amp_cf=sum(w)/N
Power_cf=sum_sq_w/N
ENBW_cf=sum_sq_w/sq_sum_w*N
```

ここまでで、補正係数の答えが次のように得られます。

```
Freq_step = 25
sum_sq_w = 96.375
sq_sum_w = 16512.2
Amp_cf = 0.501953                振幅補正係数
Power_cf = 0.376465
ENBW_cf = 1.49416                帯域幅補正係数
```

窓関数を掛けた信号列を振幅補正係数（Amplitude Correction Factor）で割って補正すると信号波形は前掲のものと同じ形ですが、振幅が２倍に大きくなっています。(Hanning 窓の場合)

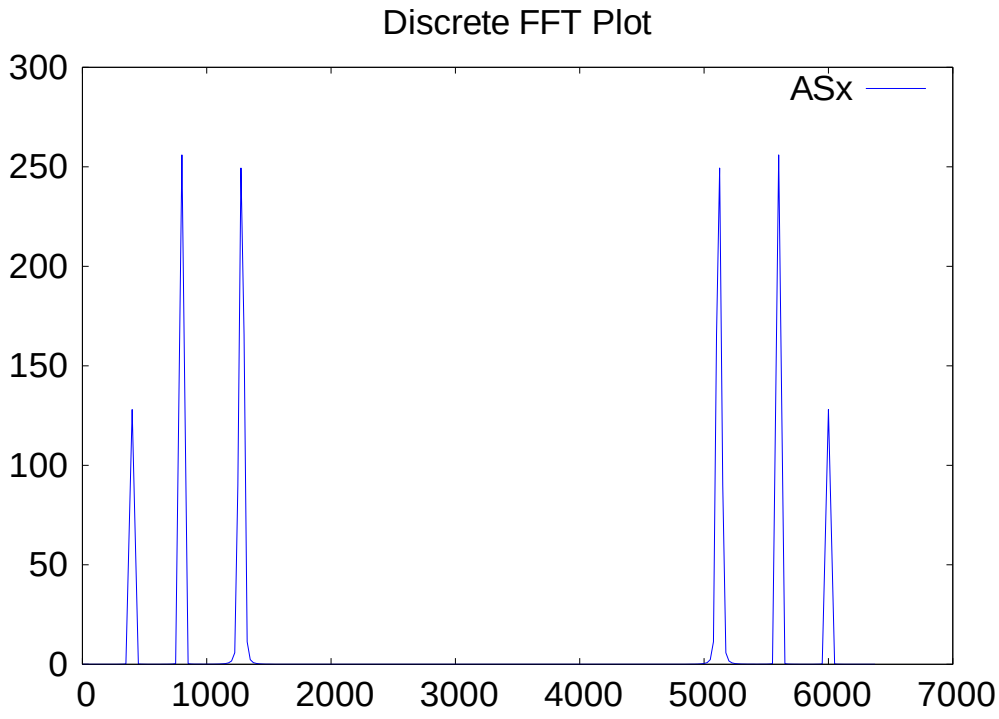
```
xn=sin(2*PI*400*t)+2*sin(2*PI*800*t)+2*sin(2*PI*1280*t);  //信号列
xnw=w*xn(1:256)/Amp_cf;       //窓による振幅減少を補正
                              //次に gnuplot を呼び出して xnw をグラフにします。
mgplot_reset(1);
mgplot_title(1,"Time domain waveform Plot with Hanning window");
mgplot(1,t,xnw,{"xnw"});
```



これをデジタルフーリエ変換して絶対値を表示すると、既に xn を xnw に振幅補正をしてあるので、Hanning 窓を通しても周波数成分の振幅は矩形窓のときと同じになっています。

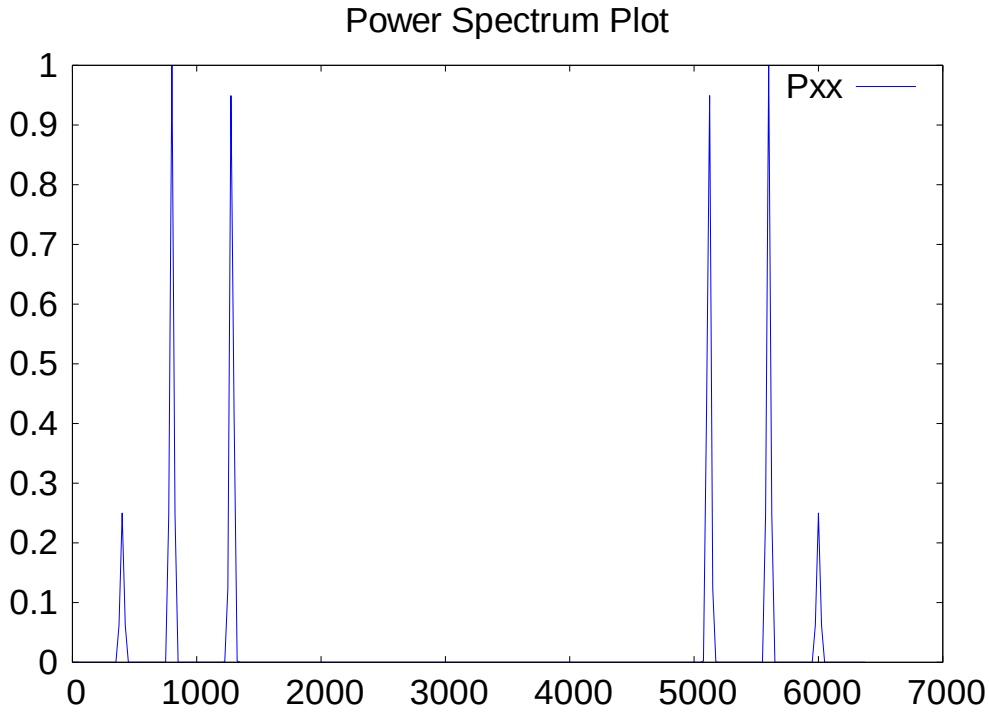
```
Sx=fft(xnw,N);
ASx=abs(Sx);                    //周波数成分の絶対値
mgplot_reset(2);
mgplot_title(2,"Discrete FFT Plot");
mgplot(2,f,ASx,{"ASx"});        //ASX をグラフにします。
                    //周波数成分の振幅は 0-p 値の N/2 倍に見える。
```

1280Hz 成分は周波数刻みの中心からずれているので、小さくなりますが、矩形窓より Hanning 窓の方が周波数刻み毎の通過帯域幅が広いので、矩形窓のときと比べると振幅減少はわずかです。



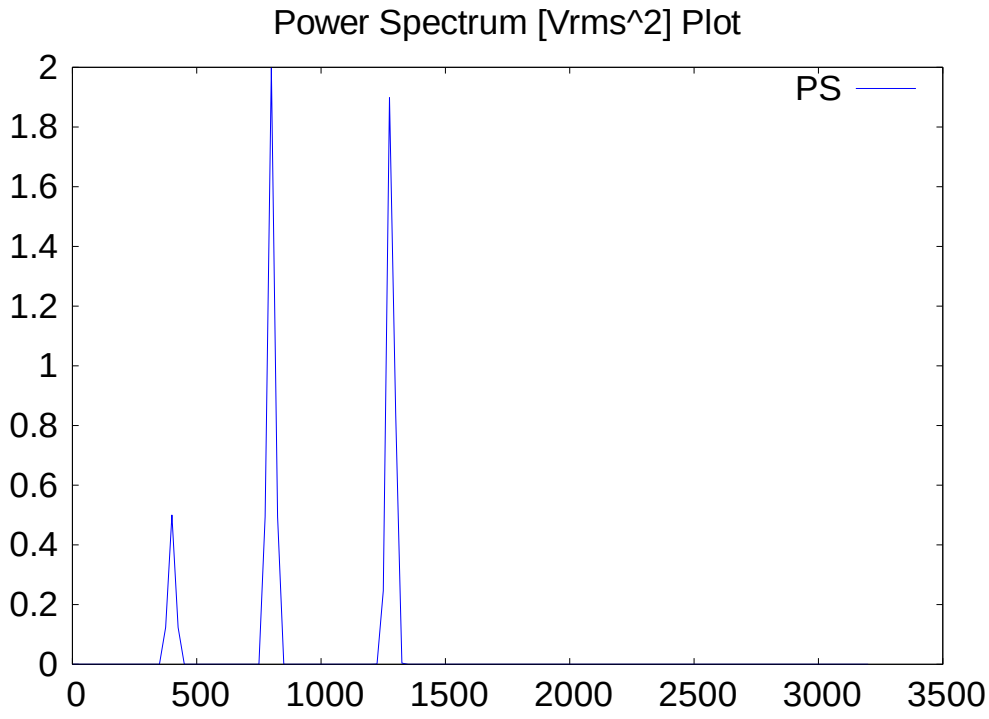
パワースペクトルは振幅補正のおかげで矩形窓と同じスケールになっています。

```
Pxx=Re(Sx*conj(Sx))/(N^2);              //N^2 でノーマライズしたパワースペクトル
mgplot_reset(3);
mgplot_title(3,"Power Spectrum Plot");
mgplot(3,f,Pxx,["Pxx"]);
      //周波数成分の振幅は(0-p 値の N/2 倍)の自乗の1／N^2 になる。
      //結局 0-p 値の自乗の 1/4 になる
```



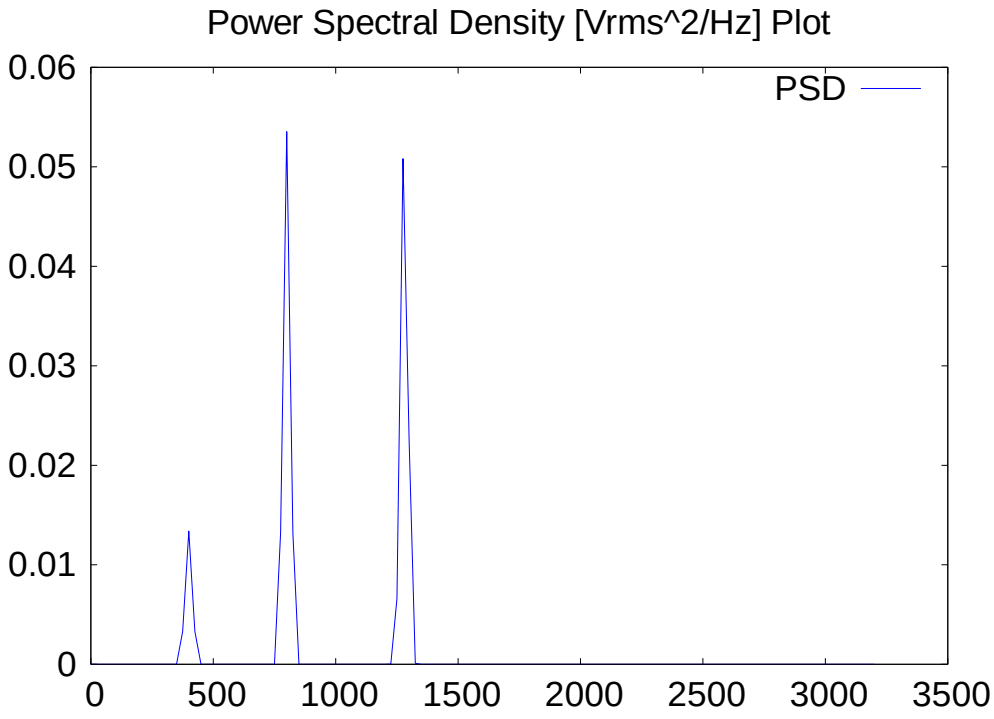
片側パワースペクトルも同様に振幅補正により Vrms^2 単位のスケールになっています。

```
PS=2*Pxx(1:N/2+1); //PS は片側パワースペクトル
PS(1)=Pxx(1); //DC 成分は2倍にしない
PS(N/2+1)=Pxx(N/2+1); //ナイキスト周波数成分は2倍にしない
//PS は FFT アナライザの表示に対応した片側パワースペクトル
//単位は Vrms^2
mgplot_reset(4);
mgplot_title(4,"Power Spectrum [Vrms^2] Plot");
mgplot(4,f(1:N/2+1),PS,["PS"]);
```



Vrms²/Hz 単位に関しては Hanning 窓によって等価帯域幅が ENBW_cf(Equivalent Noise Band Width correction factor)倍だけ周波数分解能より広がっているので、周波数分解能ではなく、等価帯域幅＝周波数分解能×補正係数で割って変換します。

```
PSD=PS/(Freq_step*ENBW_cf);
mgplot_reset(5);
mgplot_title(5,"Power Spectral Density [Vrms^2/Hz] Plot");
mgplot(5,f(1:N/2+1),PSD,{"PSD"});
```



パワーを時間軸信号に窓関数を掛けた信号から計算するには、時間軸データでも ENBM_cf で補正が必要です。

```
Pxn=sum(xnw*xnw)/N/ENBW_cf
```

パワースペクトルの周波数成分の和からパワーを計算する場合も ENBW_cf で補正が必要です。

```
Pps=sum(Pxx)/ENBW_cf
```

Vrms^2/Hz 単位のデータの和からパワーを求める場合は（ENBW_cf は使わず）周波数分解能を掛けます。
単位周波数(1Hz)当たりの Vrms^2 を N/2 個集めると N/2[Hz]当たりの Vrms^2 になります。これに Δf を掛けると(N/2)* Δf=Fs/2 より、片側パワースペクトル密度の全帯域（ナイキスト周波数まで）のパワーになり、トータルパワーを計算したことになります。

```
Ppsd=sum(PSD)*Freq_step
```

これら Pxn、Pps、Ppsd の計算値はいずれも同じになりました。

Pxn = 4.5
Pps = 4.5
Ppsd = 4.5

実効値が√2、√2、1/√2 の 3 つの正弦波の合成なので、全パワーの理論値は実効値の自乗の和で 4.5 でした。この例では誤差を生じていません。