［Scilab で信号処理 2010］

情報工学科　教授　幹　　康
ymiki@cs.takushoku-u.ac.jp



# [4] 信号の観測と窓関数

［以下に示す青色のコマンドは Scilab のコンソール上で実行できます．］

■　グラフィック・ウィンドウの初期設定（[1] 参照）

```
exec( 'DefaultWindow.sce' );
```

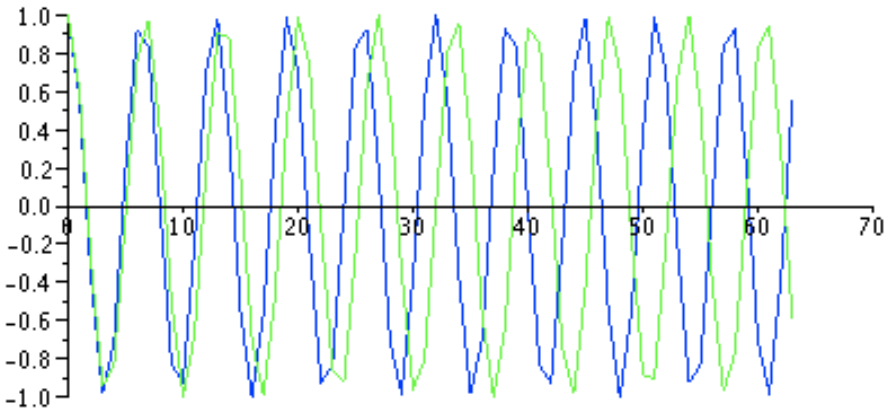
を実行します。

■ 信号の観測
観測する信号の長さを N = 64 点とします。観測時間を T = 1 [s] とすると、サンプリング周期は DT = T / N となります。

いま、
　信号1:周波数 f1 = 10 [Hz]、
　信号2:f2= 9.5 [Hz]
の2つの正弦波を考えます。

```
N = 64; f1 = 10; f2 = 9.5;
n = 0 : N - 1;
x1 = cos(2 * %pi * f1 * n / N);
x2 = cos(2 * %pi * f2 * n / N);
plot2d(n,x1,style=2)
plot2d(n,x2,style=3)
```

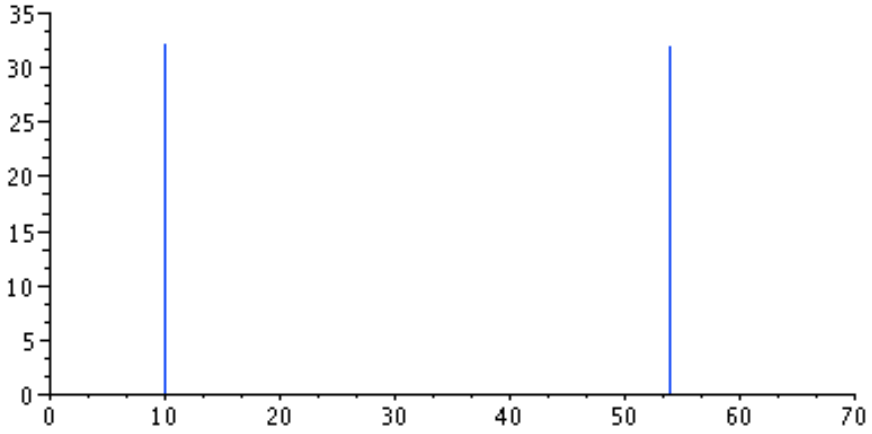
信号1は青、信号2は緑で示します。



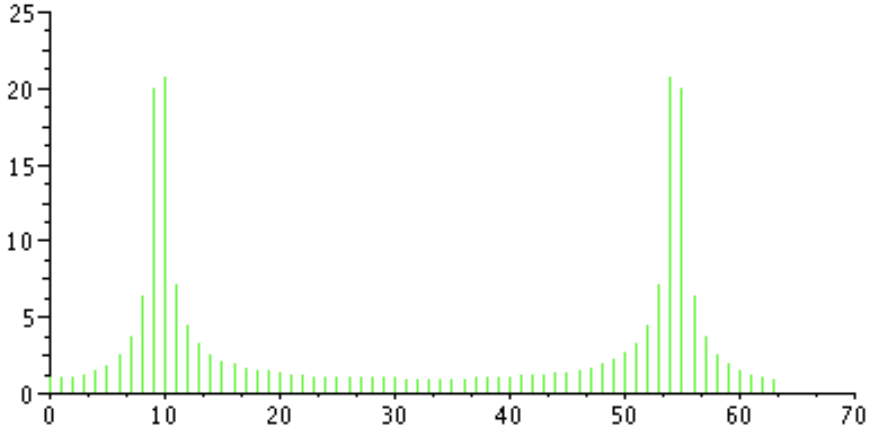

これらのスペクトルはどうなるでしょうか。どちらも正弦波ですから、ラインスペクトルになると思いますか?

```
X1 = fft(x1);
X2 = fft(x2);
scf();
plot2d3(n,abs(X1),style=2)
 scf(); f = gcf(); f.figure_position = [800,360];
plot2d3(n,abs(X2),style=3)
f = gdf(); f.figure_position = [800,40];
```

信号1の振幅スペクトル




信号2の振幅スペクトル



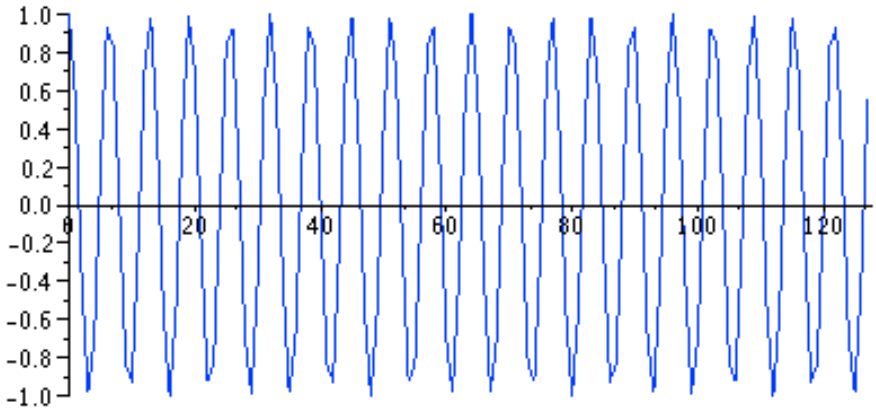

信号2 はラインスペクトルにはなっていません。この違いを明らかにすることが、信号とつきあう出発点になります。

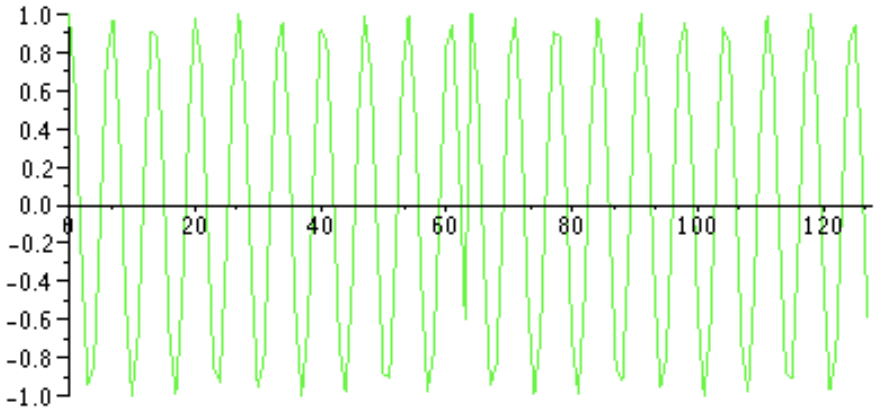
■ 観測信号の取り扱い方（その1）

観測信号は、観測時間を周期とする周期信号と考える。

N 点からなる時系列は周期 N の周期信号であることを想定しています。そこで、これら2つの信号を周期的に接続してみると

```
f = gdf(); f.figure_position = [800,40]; clf();
x1p = {x1,x1};
n2=0:127;
 plot2d(n2,x1p,style=2,rect = [0,-1,128,1])
 scf(); f = gcf(); f.figure_position = [800,320];
x2p={x2,x2};
 plot2d(n2,x2p,style=3,rect = [0,-1,128,1])
f = gdf(); f.figure_position = [800,40];
```




信号1（x1）は連続的に接続されているのに対して、信号2（x2）は接続部分（中央）に不連続が発生しています。後者は、もはや完全な正弦波とは見なせません。従って、そのスペクトルはラインスペクトルにはなりません。言い換えると、このデータ端点における不連続がスペクトルの広がりをもたらすのです。

2つの信号の根本的な違いは、信号の本来の周期と、観測時間とがちょうど整数倍の関係にあるかどうかということになります。

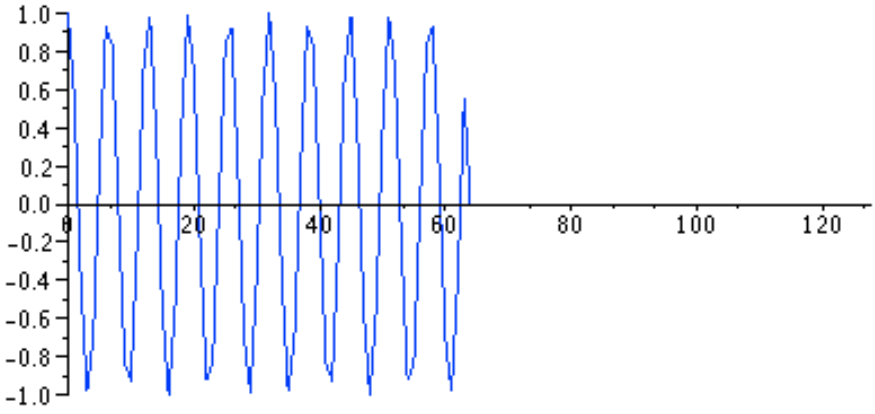
■ 観測信号の取り扱い方（その2）

分析するデータの長さを観測時間より十分長く設定する。

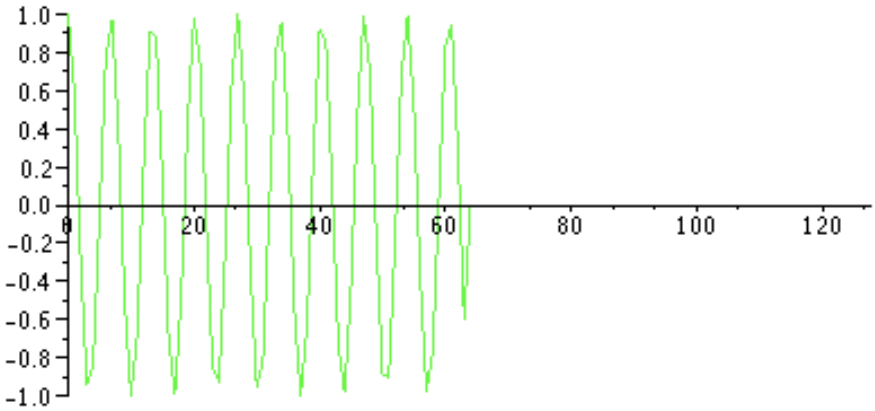
分析するデータの長さをたとえば2倍に設定しておきます。観測時間外は何の情報も得られませんからすべてゼロにします。

```
x3 = resize_matrix(x1,1,128);
x4 = resize_matrix(x2,1,128);
scf();
plot2d(n2,x3,style=2,rect = [0,-1,128,1])
scf(); f = gcf(); f.figure_position = [800,320];
 plot2d(n2,x4,style=3,rect = [0,-1,128,1])
f = gdf(); f.figure_position = [800,40];
```

信号3




信号4

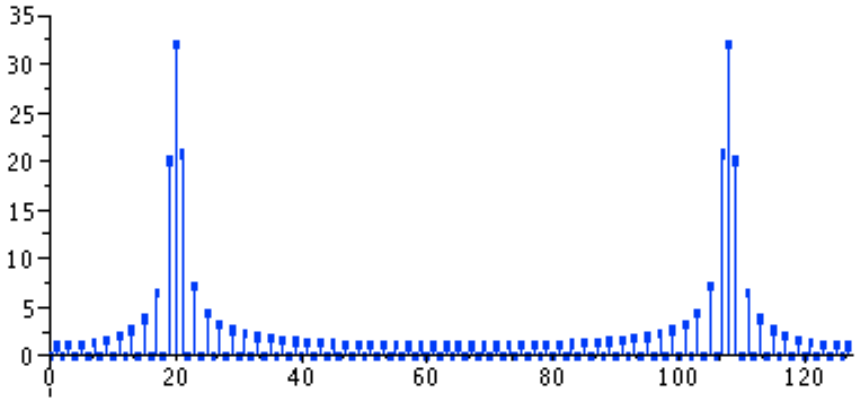



このデータと、前のデータとの違いは、周期的に接続される代わりにゼロが付加されてデータ数が2倍になったことです。
2つの信号のスペクトルを計算してみると次のようになります。

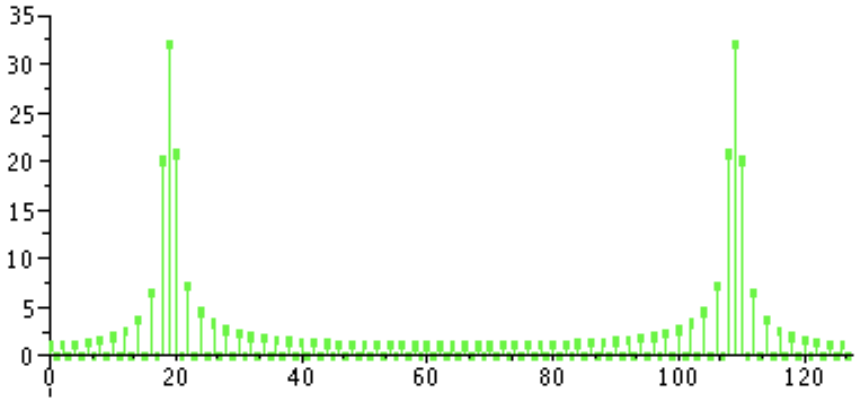
```
X3 = fft(x3);
X4 = fft(x4);
scf();
plot2d3(n2,abs(X3),style=2,rect=[0,-4,128,35])
rects = [n2-0.5;abs(X3)+0.5;ones(n2);ones(n2)];
xrects(rects, ones(n2)*2);
 scf(); f = gcf(); f.figure_position = [800,360];
```

```
plot2d3(n2,abs(X4),style=3,rect=[0,-4,128,35])
rects = [n2-0.5;abs(X4)+0.5;ones(n2);ones(n2)];
xrects(rects, ones(n2)*3);
f = gdf(); f.figure_position = [800,40];
```

信号3の振幅スペクトル



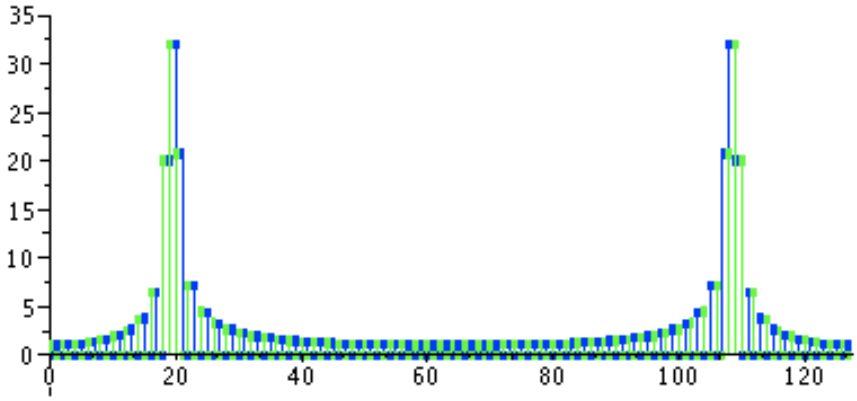

信号4の振幅スペクトル




2つのスペクトルはほとんど同じです。両スペクトルを重ねてみると、ピークの位置が1周波数刻み分ずれているだけです。

```
plot2d3(n2,abs(X3),style=2,rect=[0,-4,128,35])
rects = [n2-0.5;abs(X3)+0.5;ones(n2);ones(n2)];
xrects(rects, ones(n2)*2);
plot2d3(n2,abs(X4),style=3,rect=[0,-4,128,35])
rects = [n2-0.5;abs(X4)+0.5;ones(n2);ones(n2)];
xrects(rects, ones(n2)*3);
```

信号3、4のスペクトルの違い

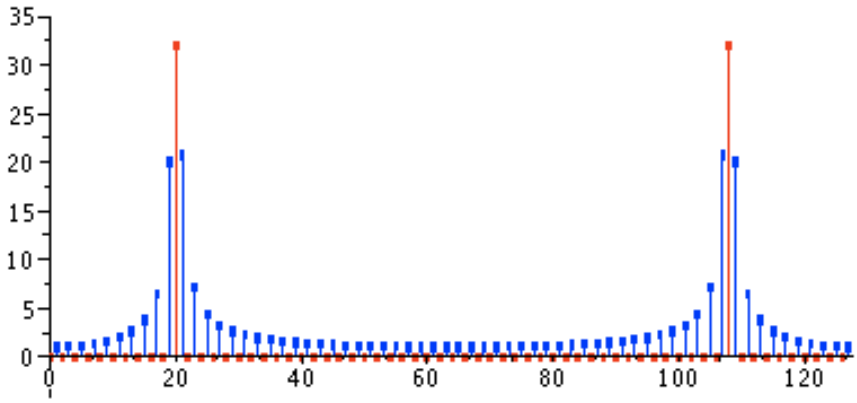



■ 信号の取り扱い方（その1）と（その2）の比較

前節（その1）と（その2）のスペクトルを比較してみると、両者の違いは時系列の点数が2倍になったこと、すなわち、周波数刻みが1/2になったことです。

信号1、3のスペクトルの違い

```
n22=0:2:126;
scf();
plot2d3(n2,abs(X3),style=2,rect=[0,-4,128,35])
rects = [n2-0.5;abs(X3)+0.5;ones(n2);ones(n2)];
xrects(rects, ones(n2)*2);
plot2d3(n22,abs(X1),style=5,rect=[0,-4,128,35])
rects = [n22-0.5;abs(X1)+0.5;ones(n22);ones(n22)];
xrects(rects, ones(n22)*5);
```

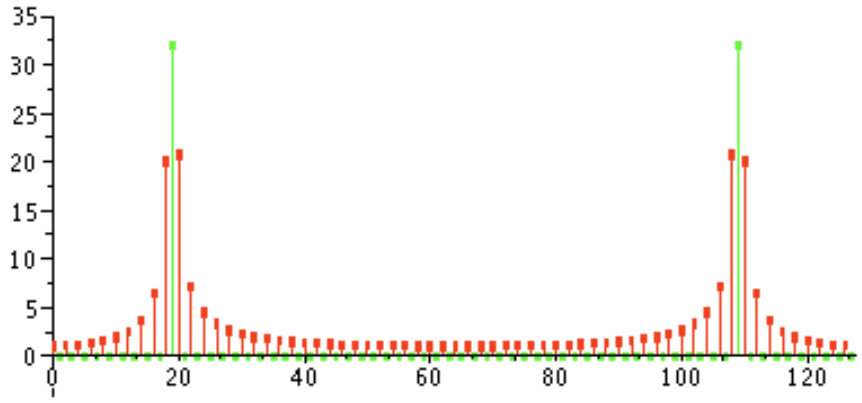



信号1のスペクトル（赤）は、信号3のスペクトル（青）を1つおきに取り出したものになっています。つまり、周波数刻みが2倍になっています。

信号2、4のスペクトルの違い

```
scf();
```

```
plot2d3(n2,abs(X4),style=3,rect=[0,-4,128,35])
rects = [n2-0.5;abs(X4)+0.5;ones(n2);ones(n2)];
xrects(rects, ones(n2)*3);
plot2d3(n22,abs(X2),style=5,rect=[0,-4,128,35])
rects = [n22-0.5;abs(X2)+0.5;ones(n22);ones(n22)];
xrects(rects, ones(n22)*5);
```

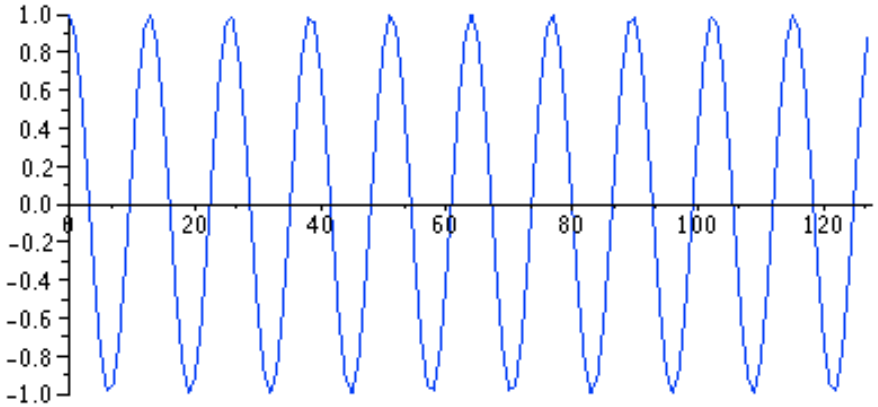



信号2のスペクトル（赤）は、信号4のスペクトル（緑）を1つおきに取り出したものになっています。つまり、周波数刻みが2倍になっています。前のスペクトルとの違いは、ピークの位置が1刻み分ずれているために、サンプルされる位置がずれ、結果的に、スペクトルの形状が大きく変化して表示されているのです。周波数刻みが細かくなると、見えなかったものが見えてくるということです。

■ 信号の観測と窓関数

観測信号を格納するために用意した配列のサイズより観測信号の長さが短い場合があります。
下に信号 y(n) を示します。

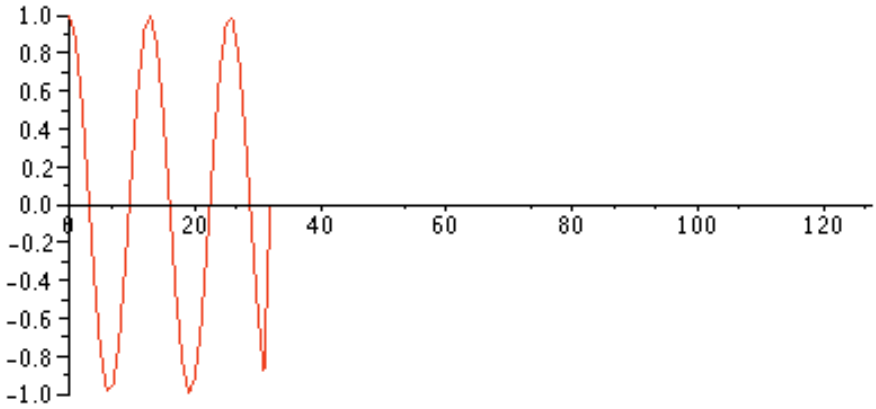
```
n2 = 0:127;
y = cos(2*%pi*n2/12.8);
plot2d(n2,y,style=2,rect=[0,-1,128,1])
```




この信号の一部分を観測したとします（z(n)）。

```
x = {ones(1:32),zeros(1:96)};
z=y.*x;
```

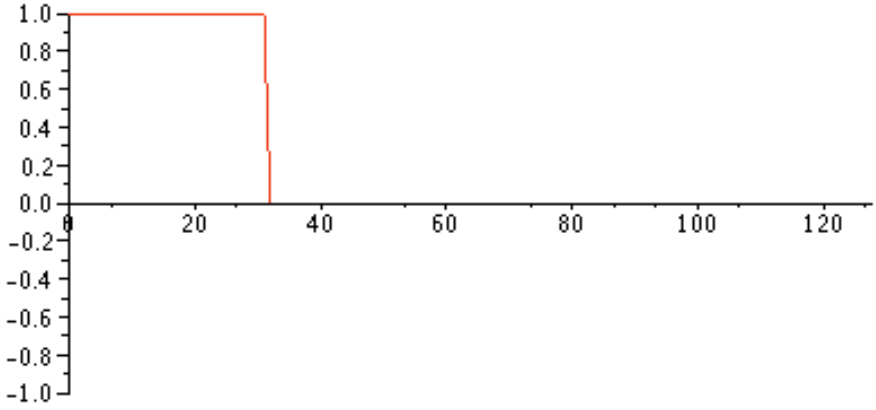
```
plot2d(n2,z,style=5,rect=[0,-1,128,1])
```




このデータは本来の信号に下に示す「窓」(window)あるいは「窓関数」(window function)と呼ばれるデータ(x(n))を掛けたものと考えることができます。

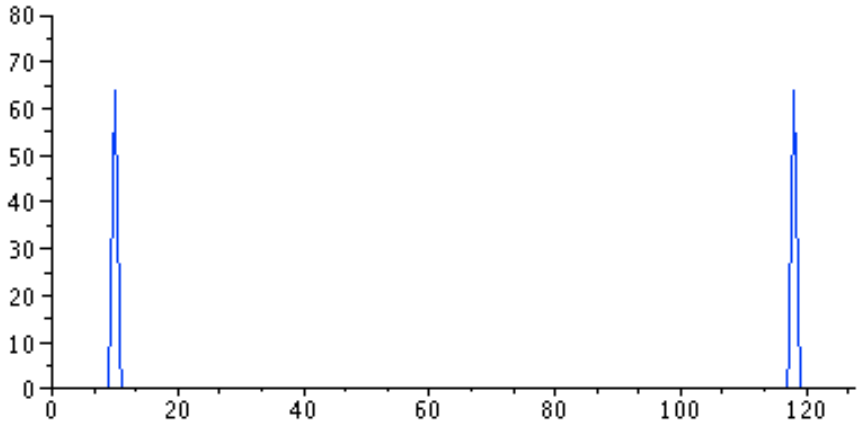
z(n) = y(n) * x(n)

```
plot2d(n2,x,style=5,rect=[0,-1,128,1])
```



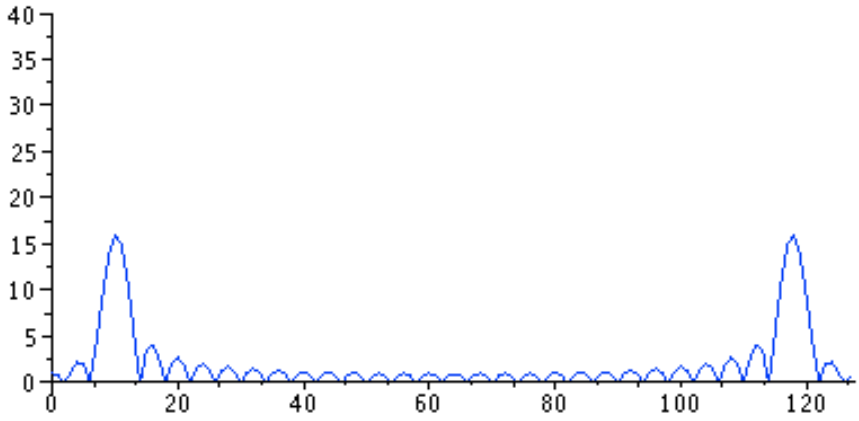

これらのスペクトルを見てみましょう。
原信号 y(n) のスペクトル Y(k)

```
Y = fft(y);
plot2d(n2,abs(Y),style=2,rect=[0,-1,128,10])
```



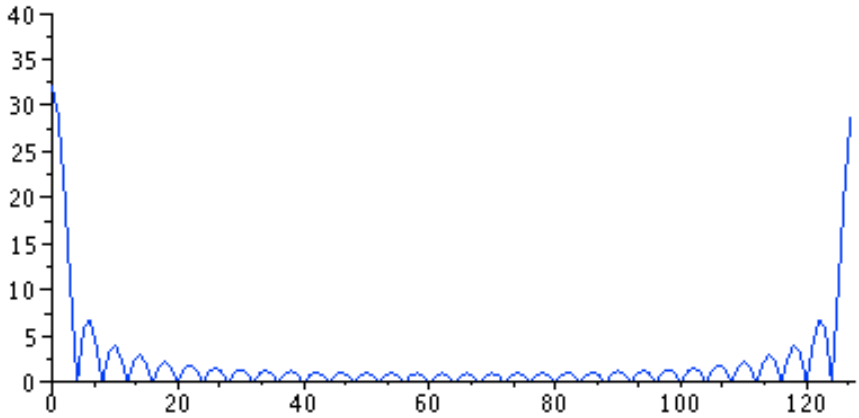

観測信号 z(n) のスペクトル Z(k)

```
scf();
Z = fft(z);
plot2d(n2,abs(Z),style=2,rect=[0,-1,128,40])
```




窓関数 x(n) のスペクトル X(k)

```
scf();
X = fft(x);
plot2d(n2,abs(X),style=2,rect=[0,-1,128,40])
```




以上のように、本来、ラインスペクトルを持っていた原信号のスペクトル Y(k) は、一定の観測時間に対応する窓関数 x(n) の影響を受けて、窓関数のスペクトル X(k) の形状が観測信号のスペクトル Z(k) 上に反映することがわかります。

```
n2 = 0:127;
y = cos(2*%pi*n2/12.8);
for k = 1:4,
scf(0);f = gcf(); f.figure_position = [800,560];
scf(1);f = gcf(); f.figure_position = [800,300];
scf(2);f = gcf(); f.figure_position = [800,40];
for i = 1:7,
      j = 2^i;
      x = {ones(1:j),zeros(1:128-j)};
      scf(0); clf(0);
      plot2d(n2,x,style=2,rect=[0,-1,128,1]);
      X = fft(x);
      scf(1); clf(1);
      plot2d(n2,abs(X),style=2,rect=[0,-1,128,80]);
      scf(2); clf(2);
      Z = fft(x.*y);
      plot2d(n2,abs(Z),style=2,rect=[0,-1,128,80]);
      sleep(300);
end
end
```

窓関数（ウィンドウ）のフーリエ変換はスペクトルの周波数分解能を与えることになります。窓が広くなれば分解能は向上し、逆に、狭くなると分解能は低下します。