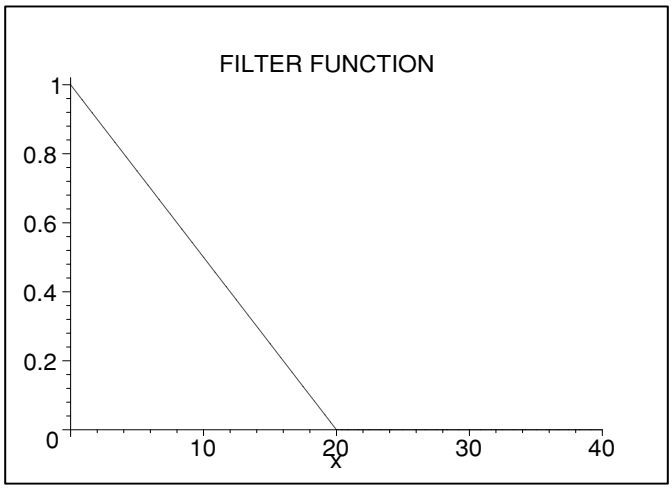
1.6.2 フーリエ変換

フーリエ変換を用いたデータのフィルタリングについての例です．Maple では FFT(高速フーリエ変換) のルーチンが用意されています．詳しくは help を参照下さい．

演習

与えられた 256 個のデータ（ファイル名 DATA101 ）をフーリエ変換し，高周波成分を下図の三角フィルタを用いて取り除き，フーリエ逆変換せよ．上図のノイズ成分はどうなったか．振幅強度 $\sqrt{a_n^2 + b_n^2}$ をプロットせよ（128 チャンネル）．はじめのデータの面積強度とフーリエ変換で求めたデータの面積強度とはどういう関係があるのか．



解説

時系列データ x (t) の解析にはそのフーリエ変換

$$X(f) = \int_{-\infty}^{\infty} x(t) \exp(2\pi i f t) dt \tag{1.3}$$

が最も多用されます．これは，高速フーリエ変換アルゴリズムの発明によって計算機での処理が容易となったことが大きく効いています．時間経過と共に変化する時系列測定データ $x(t)$ に含まれるさまざまな周波数成分のすべてについて，何 Hz の周波数成分がどの程度の割合でその時系列データの中に含まれているのかを明らかにできるからです．時系列データの中のノイズは δ 関数に似ており，そのフーリエ変換は全周波数成分を一様に含んでいます．

一方，測定したいデータは一般に，時間に対してゆっくり変化する量です．測定によって得られたデータの時間軸は連続ではなくとびとびの値をとるので，離散フーリエ変換を取ることになります．この例では時間データは 256 チャンネルのデータです．高周波成分（=ノイズ成分）を取り除くために，データに対して窓関数（低域通過フィルタ）をかけてフーリエ逆変換

$$x(t) = \int_{-\infty}^{\infty} X(f)\exp(2\pi i f t)dt \tag{1.4}$$

することによってフィルターを掛けた後のスムーズなカーブが得られます．低域通過フィルタには，方形窓，修正方形窓，フォンハン窓（2 乗余弦窓），ハミング窓（平たん部のある 2 乗余弦窓）などさまざまなものがあります．

実践

具体的に計算を見ていきましょう．

```
> restart:
> T:=readdata(‘DATA101‘,1):
> Idata:=array([seq(0,i=1..256)]):
> Rdata:=convert(T,array):
```

全パワーを求めておきましょう．

```
> temp:=add(i^2,i=T);
```

$$temp := 534310.3363$$

実際に FFT を実行します．

```
> FFT(8,Rdata,Idata);
```

$$256$$

これだけで終わりです．ここで FFT の第一引数の 8 はデータの個数が 2^8 であることを示しています．フーリエ変換された後のデータ構造を見ておきましょう．

```
> printf("%3d %15.5f %15.5f\n",1,Rdata[1],Idata[1]);
> for i from 2 to 128 do
> printf("%3d %15.5f %15.5f %15.5f %15.5f\n",
> i,Rdata[i],Idata[i],Rdata[258-i],Idata[258-i]);
> od;
```

```
  1      8499.54360         0.00000
  2     -4162.42568       -81.92030     -4162.42568        81.92031
  3      2602.79575       120.20850      2602.79575      -120.20850
```

⋮

中略

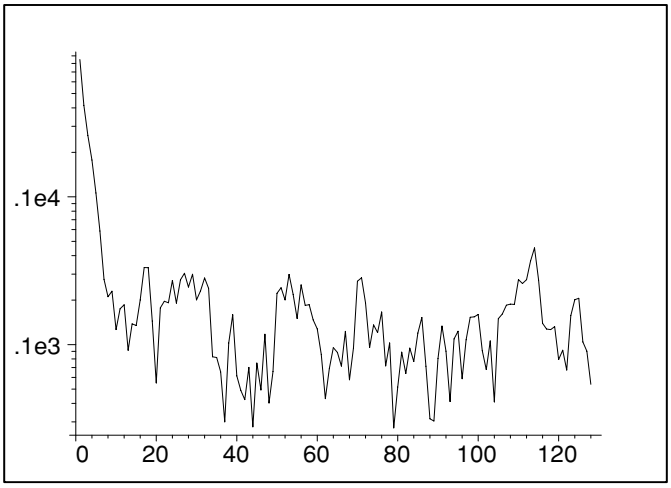
⋮

```
125      -160.89158      -128.35496      -160.89158       128.35496
126        91.48342        49.80754        91.48342       -49.80754
127        36.31291       -82.64862        36.31291        82.64862
128        53.48855         6.15179        53.48855        -6.15179
```

波数空間での強度を logplot で見てみましょう.

```
> Adata:=[seq([i,sqrt(Idata[i]^2+Rdata[i]^2)],i=1..128)]:
> with(plots):
> logplot(Adata);
```

```
Warning, the name changecoords has been redefined
```



離散的な形の Parseval の定理が成り立っていることも確認できます.

```
> add(Rdata[i]^2+Idata[i]^2,i=1..256)/256;
```

$$534310.3320$$

先程求めた実空間での全パワーの結果と一致しています. 次にフィルター関数を作りましょう.

```
> filter:=x-> piecewise(x>=0 and x<=20,(1-x/20) );
> plot(filter(x),x=0..40,title="FILTER FUNCTION");
```

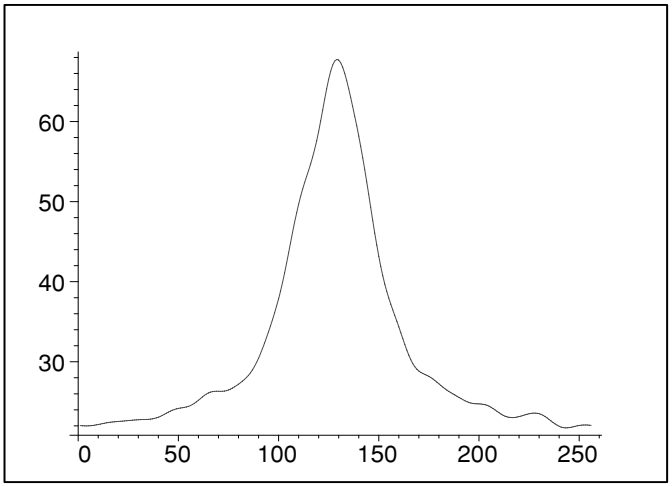
$$filter := x \rightarrow \mathrm{piecewise}(0 \le x \ \mathbf{and}\ x \le 20,\ 1 - \frac{1}{20}\,x)$$

この結果が本節の初めに示したフィルター関数の図です. フィルターを通したデータを作ります.

```
> FRdata:=array([seq(Rdata[i]*filter(i),i=1..256)]):
> FIdata:=array([seq(Idata[i]*filter(i),i=1..256)]):
```

逆フーリエ変換を実行すると,

```
> iFFT(8,FRdata,FIdata);
> Adata:=[seq([i,FRdata[i]],i=1..256)]:
> plot(Adata);
```



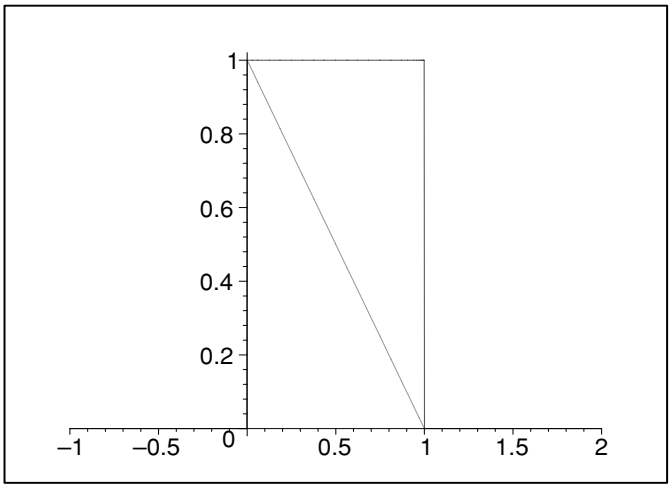
ノイズが取り除かれているのが分かるでしょう.

1.6.3 畳み込み (コンボリューション)

波長 t の真のスペクトルが下図の箱形関数 $x(t)$ のようなかたちをしている. 分光器の装置関数が下図の三角関数 $h(t)$ で表されるとき，観測されるスペクトル

$$f(t) = \int_{-\infty}^{\infty} x(t-\tau)h(\tau)d\tau \tag{1.5}$$

はどのような形状となるか.



解説

時刻 t の観測値 $x(t)$ を観測しようとするとき，その t の瞬間のみではなく，t の過去と未来が重み $h(t)$ で見えることがあります．このとき実測されるスペクトルは $x(t)$ と $h(t)$ の畳み込み (convolution),

$$f(t) = \int_{-\infty}^{\infty} x(t-\tau)h(\tau)d\tau \tag{1.6}$$

となります．スリットを持つ分光器によって，ある波長 t のスペクトル強度 $x(t)$ を観測するとき，分光器はその前後の波長 $t-\tau$ の光も重み $h(t-\tau)$ で見ていることを意味しています．畳み込みを間単に $x*h$ と表すと，そのフーリエ変換は

$$F(x*h) = F(x)F(h) \tag{1.7}$$

となります．すなわち convolution のフーリエ変換は，フーリエ変換の積になるわけです．目的のスペクトル $x(t)$ を求めるためには，わり算 $F(x*h)/F(h)$ の結果をフーリエ逆変換すればよいことがわかります．convolution の形の積分方程式を解くより，わり算の方が計算上簡単な操作ですむ場合が多く，これを deconvolution といいます．

実践

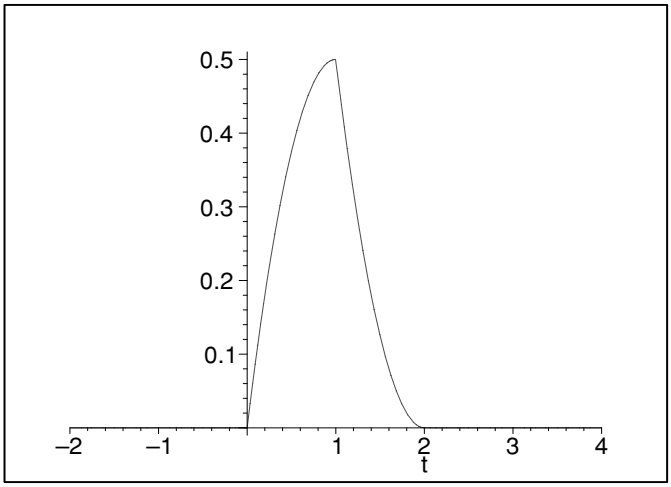
```
> restart;
> fun:=x->piecewise(x>=0 and x<=1,1);
> h:=x->piecewise(x>=0 and x<=1, 1-x);
> plot({fun,h},-1..2);
```

$$fun := x \rightarrow \text{piecewise}(0 \le x \textbf{ and } x \le 1,\ 1)$$

$$h := x \rightarrow \text{piecewise}(0 \le x \textbf{ and } x \le 1,\ 1-x)$$

前掲の関数が描かれます．convolution を実際に計算すると

```
> plot(int(fun(x)*h(t-x),x=-1..1),t=-2..4);
```



と求まります.

1.6.4 非線形最小二乗法

非線形の最小二乗法を用いてデータフィットをしてくれる関数は default では Maple には用意されていないようです[3]. そこで, 実際に非線形最小二乗法のプログラムを作成し, 実際のデータへのフィッティングを試みてみましょう.

問題

与えられたデータ (ファイル名'DATA101') を,

$$f(x) = a + \frac{b}{c + (x - d)^2} \tag{1.8}$$

なるローレンツ型関数にカーブフィッティングするプログラムを作成し, そのパラメータを決定せよ. ここで a はバックグラウンドの強度, b はローレンツ関数の強度, c は線幅, d はピーク位置を表す.

[3]公式のサポートではありませんが, 世界中の科学者, 技術者が Maple で開発した library を公開している The Maple Application Center (http://www.mapleapps.com/) の Data Analysis のカテゴリーに, 後述する Levenberg-Marguqrdt 法を用いた Data fitting の汎用プログラム Generalized Weighted Non-Linear Regression Using the Levenberg-Marquardt Method by David Holmgren (http://www.mapleapps.com/categories/data_analysis_stats/data/html/genfit_6.html) があります.