5.14 Kernel の基礎(1)　　ver.0.1

標準正規乱数を発生させて、そのヒストグラムを描いてみます。区切りの数は、仮に

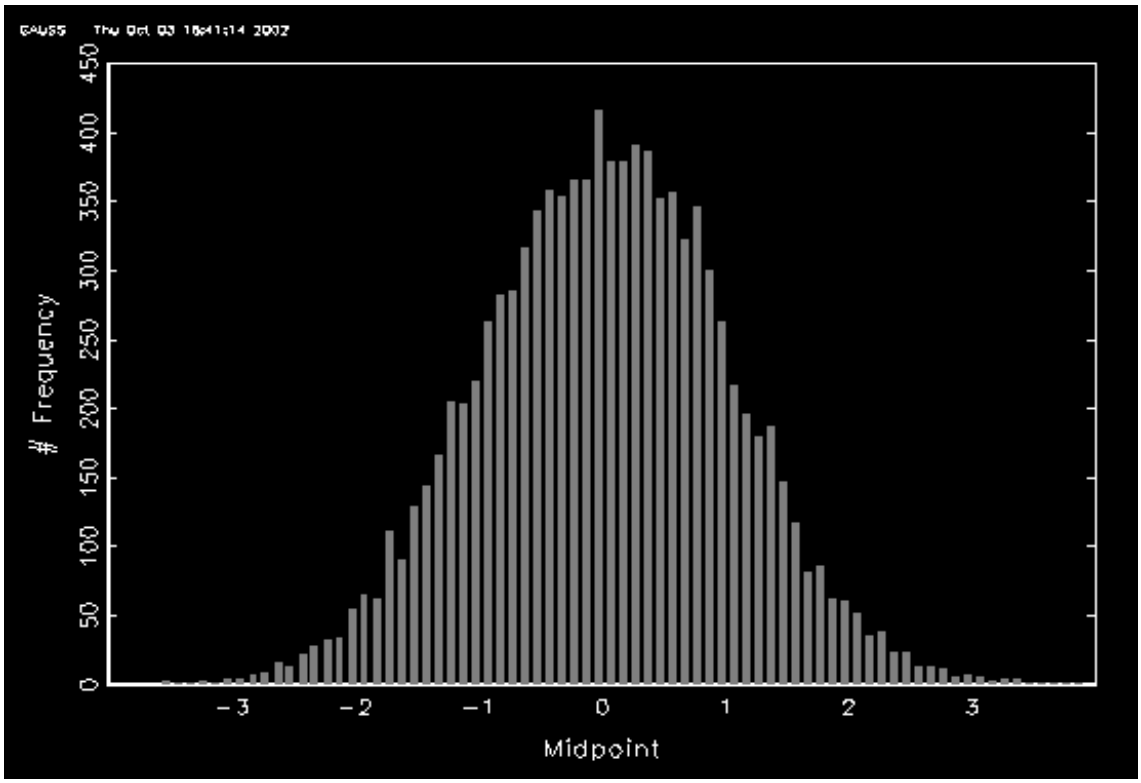
（最高値 - 最低値）÷ヒストグラムの幅

として、近似的に、計算して設定してみます。

プログラム

```
new; cls;
library pgraph;
graphset;
rndseed 1000;

h=0.1;
n=10000;
x=rndn(n,1);
x=sortc(x,1);
a=round((x[n]-x[1])/h);
hist(x,a);
```

グラフ表示



上の計算では、h=0.1 にだいぶん狭い値を設定しています。おおよそのところの分布を表せていると思います。幅hをさらに 0.01 などに小さくすると、真の分布の回りで細かく変動

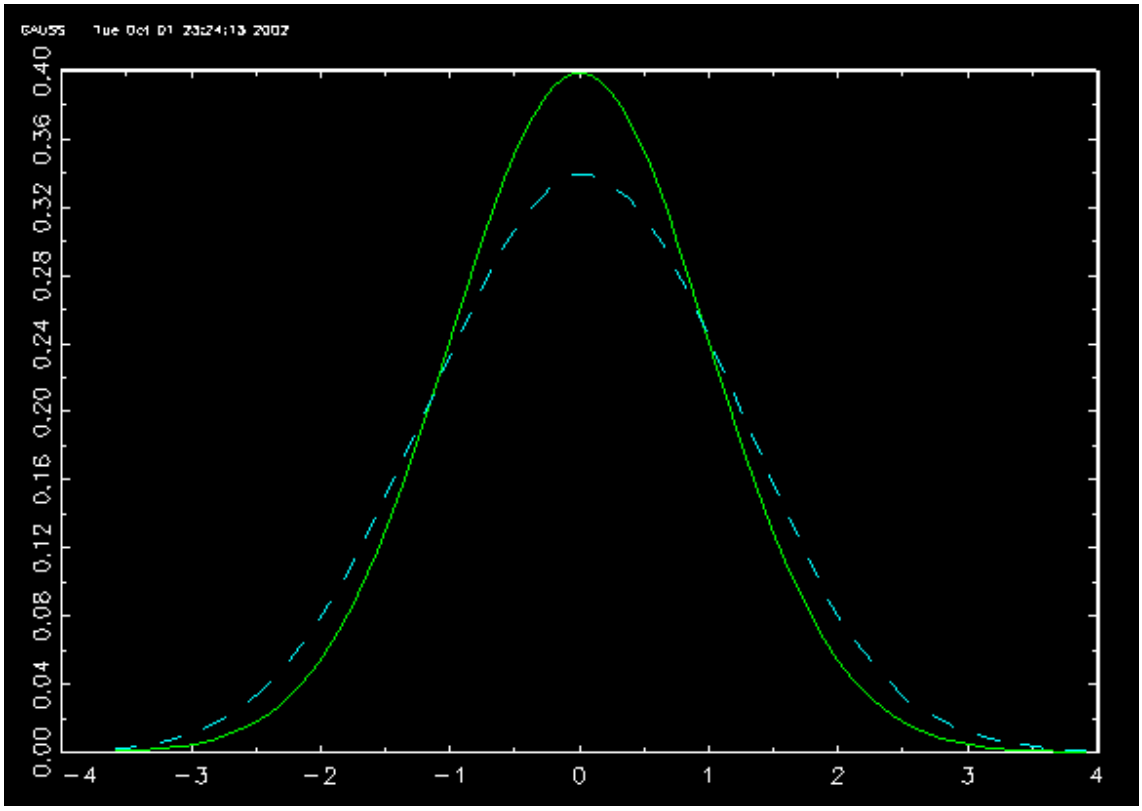
するのが激しくなります。反対に、幅ｈを大きく整数の値、例えば、１などをとるとヒストグラムは「ぼやけた」ような形になります。ヒストグラムは、このように区切りの数を決めて、それぞれの領域を固定して、その中にある度数を計算するわけですが、Kernel の考え方は、大雑把に言って、この領域幅を左から右に徐々に平行移動したものを細かく計算して結果を曲線でつないだものと言えます。

まずは、一番単純な Uniform Kernel を用いてその density をグラフ化します。

プログラム

```
    new; cls;
    library pgraph;
    rndseed 1000;
    call plotdens(10000,1);

proc plotdens(n,h);
    local x,points,dens,kern,i;
    x=rndn(n,1);
    x=sortc(x,1);
    points=seqa(x[1],(x[n]-x[1])/100,101);
    dens=pdfn(points);
    i=1; kern=zeros(101,1);
    do while i<=101;
        /* Here, we use uniform kernel. */
        kern[i]=ukernel(points[i],x,h);
        i=i+1;
    endo;
    xy(points,dens~kern);
    retp(dens~kern);
endp;

proc ukernel(x0,x,h);
    local n,u,kv;
    n=rows(x);
    u=(x0-x)/h;
    kv=0.5*(abs(u).<=1);
    retp( 1/(n*h)*sumc(kv) );
endp;
```

グラフ表示



上のプログラムでは、最大値と最小値の間を便宜上 100 等分した点で評価をしていて、これを u=(x0-x)/h;として評価点との実際の値の差をバンド幅ｈで割ったものを u として、kernel に代入することによって、平行移動させて、それぞれに対するカーネル値 kv を求めて、さらにこれをカーネル K の中に代入して、

$$f(x) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x0 - x}{h})$$

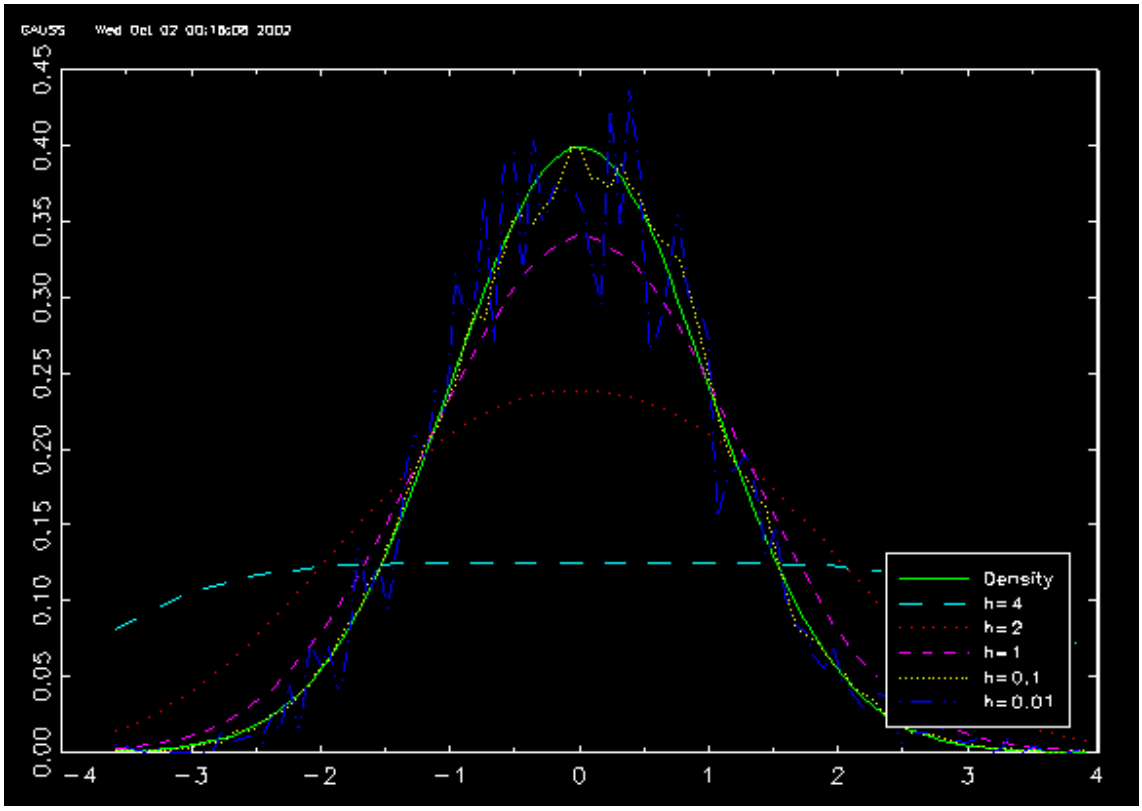
として density を求めます。ここで、K( )は任意の Kernel がきます。例えば、以下の場合は Uniform Kernel を採用しています。ここでは、バンド幅ｈを１としてしまったため、標準正規分布の density に比べてテイルが分厚い形になってしまっています。実際、バンド幅ｈが広すぎるためこういう結果になっています。このことをｈを変化させたグラフで確かめておきます。

プログラム

```
new; cls;
library pgraph; graphset;
rndseed 1000;
call plotdens(10000);


proc plotdens(n);
```

```
    local x,points,dens,kern,i,j,h;
    h={4,2,1,0.1,0.01};
    x=rndn(n,1);
    x=sortc(x,1);
    points=seqa(x[1],(x[n]-x[1])/100,101);
    dens=pdfn(points);
    i=1; kern=zeros(101,rows(h));
    do while i<=101;
        /* Here, we use uniform kernel. */
        j=1;
        do while j<=rows(h);
            kern[i,j]=ukernel(points[i],x,h[j]);
            j=j+1;
        endo;
        i=i+1;
    endo;
    _plegctl=1;
    _plegstr="Density¥000h=4¥000h=2¥000h=1¥000h=0.1¥000h=0.01";
    xy(points,dens~kern);
    retp(dens~kern);
endp;

proc ukernel(x0,x,h);
    local n,u,kv;
    n=rows(x);
    u=(x0-x)/h;
    kv=0.5*(abs(u).<=1);
    retp( 1/(n*h)*sumc(kv) );
endp;
```

グラフ表示



上のように、n=10000 と標準正規分布のσ＝１に対しては、バンド幅ｈ＝１では広すぎてもう少し狭くする必要があることがわかると思います。バンド幅ｈが広いと滑らかになるかわりにテイルが分厚くなり、その反対にバンド幅ｈが狭すぎてしまうとでこぼこが激しくなります。何か最適なバンド幅ｈというものが、だいたいのところデータ数ｎと散らばりσ（あるいは分布の形）に依存して決まってくることがわかると思います。

さらに、今度はΓ分布からの乱数によって、同じことをやってみましょう。

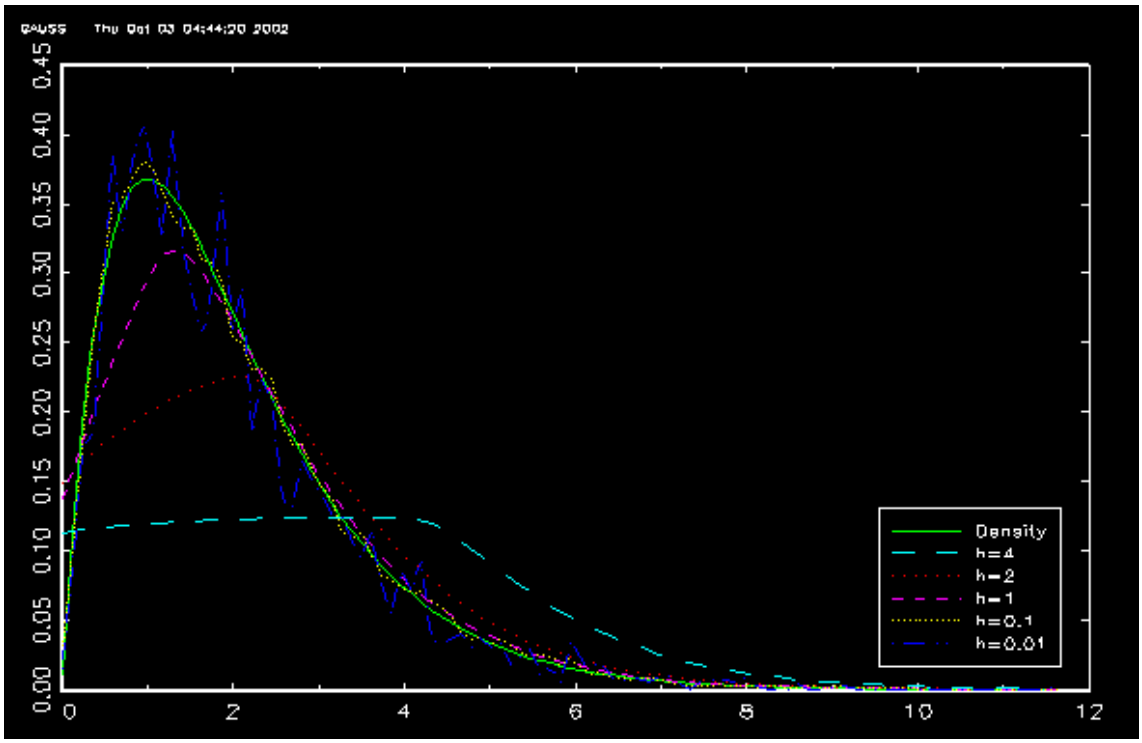
プログラム

```
new; cls;
library pgraph; graphset;
rndseed 1000;
call plotdens(10000);


proc plotdens(n);
local x,points,dens,kern,i,j,h,alpha;
h={4,2,1,0.1,0.01};
alpha=2;
x=rndgam(n,1,alpha);                    @ Now we draw from gamma dist. @
x=sortc(x,1);
```

```
    points=seqa(x[1],(x[n]-x[1])/100,101);
    dens=pdfgam(points,alpha);              @ pdf here is gamma. @
    i=1; kern=zeros(101,rows(h));
    do while i<=101;
        /* Here, we use uniform kernel. */
        j=1;
        do while j<=rows(h);
            kern[i,j]=ukernel(points[i],x,h[j]);
            j=j+1;
        endo;
        i=i+1;
    endo;
    _plegctl=1;
    _plegstr="Density¥000h=4¥000h=2¥000h=1¥000h=0.1¥000h=0.01";
    xy(points,dens~kern);
    retp(dens~kern);
endp;

proc ukernel(x0,x,h);
    local u,kv;
    u=(x0-x)/h;
    kv=0.5*(abs(u).<=1);
    retp( sumc(kv/(rows(x)*h)) );
endp;

proc pdfgam(x,a);
    retp( 1/gamma(a).*x^(a-1).*exp(-x) );
endp;
```

こちらは、０よりも大きい範囲で乱数が作成されます。上のプログラムでは、乱数とその PDF をそれぞれΓ分布のものに変えて、α＝２のときの分布にもとづくものをプロットしています。GAUSS にはΓの PDF は存在しないので、簡易な procedure を末尾に pdfgam として付け加えています。

グラフ表示



バンド幅の設定

いろいろな方法があるが、一番簡単な方法は、ガウス型 Kernel のときのバンド幅を

$$h_g=1.059\,\sigma\,n^{-0.2}$$

とする方法で、これをもとに他の型の Kernel をそれぞれの定数ｃをかけて

$$h=c\,h_g$$

として補正した値を出すものである。なお、このコンスタントｃは

Epanechnikov	2.214
Biweight	2.623
Triweight	2.978
Uniform	1.740

であることが知られている。もちろん、ガウス型の時は１である。これとは別に Silverman の方法として

$$h=0.9\min[\sigma,(\text{第３四分位点}-\text{第１四分位点})/1.34]\,n^{-0.2}$$

を計算する方法もある。以下ではこの方法を用いて自動的にバンド幅を計算しておこう。

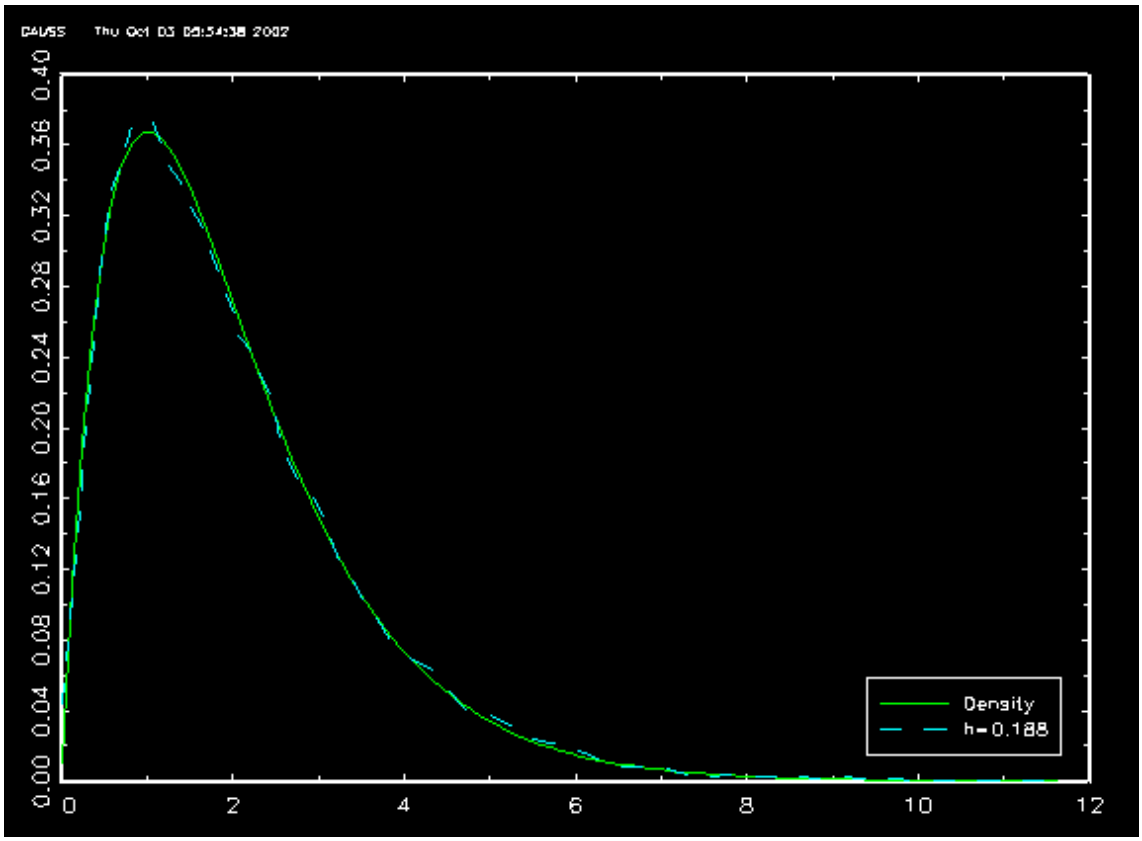
プログラム

```
new; cls;
library pgraph; graphset;
rndseed 1000;
```

```
    call plotdens(10000);

proc plotdens(n);
    local x,points,dens,kern,i,j,h,alpha;
    alpha=2;
    x=rndgam(n,1,alpha);
    x=sortc(x,1);
    /* h=1.059*stdc(x)*n^(-0.2)*1.740;       */
    h=0.9*minc(stdc(x)|((x[round(0.75*n)]-x[round(0.25*n)])/1.34))*n^(-0.2);
    points=seqa(x[1],(x[n]-x[1])/100,101);
    dens=pdfgam(points,alpha);
    i=1; kern=zeros(101,rows(h));
    do while i<=101;
        /* Here, we use uniform kernel. */
        j=1;
        do while j<=rows(h);
            kern[i,j]=ukernel(points[i],x,h[j]);
            j=j+1;
        endo;
        i=i+1;
    endo;
    _plegctl=1;
    _plegstr="Density¥000h=" $+ftos(h,"%*.*lf",4,3); @ print h @
    xy(points,dens~kern);
    retp(dens~kern);
endp;

proc ukernel(x0,x,h);
    local u,kv;
    u=(x0-x)/h;
    kv=0.5*(abs(u).<=1);
    retp( sumc(kv/(rows(x)*h)) );
endp;

proc pdfgam(x,a);
    retp( 1/gamma(a).*x^(a-1).*exp(-x) );
```

endp;

グラフ表示



なお、上のケースの場合、前者の近似的な値を使って計算すると、バンド幅は上の２倍程度になって、でこぼこさは消えるが真の PDF と若干ずれてくる。

様々な Kernel

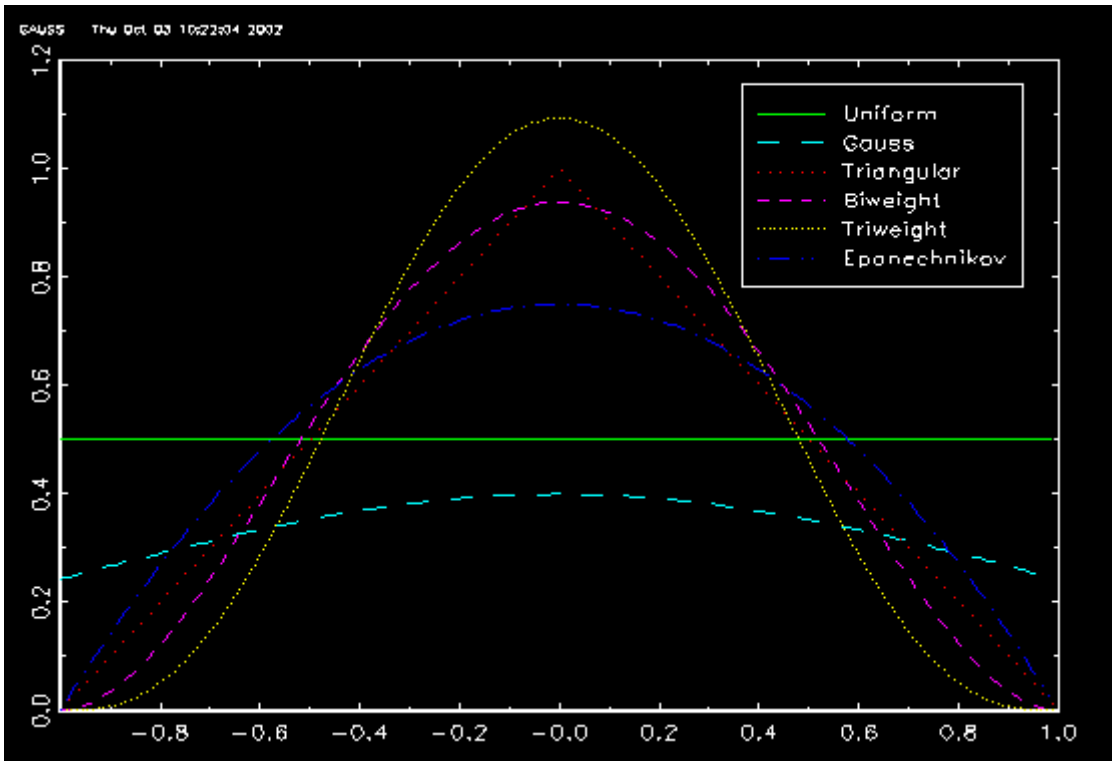
カーネルを使う前に、一度－１から１までの間の種々の Kernel を図示しておこう。

プログラム

```
new; cls;
library pgraph;
graphset;
u=seqa(-1,0.01,200);
/* Kernel Values */
kv1=0.5*(abs(u).<=1);
kv2=(1/sqrt(2*pi))*exp(-0.5*u^2);     /* Same as kv2=pdfn(u); */
kv3=(1-abs(u)).*(abs(u).<=1);
kv4=(15/16)*((1-u^2)^2).*(abs(u).<=1);
kv5=(35/32)*((1-u^2)^3).*(abs(u).<=1);
```

```
kv6=0.75*(1-u^2).*(abs(u).<=1);
/* Graphic settings */
xscale={-1,1}; yscale={0,1.2};
scale(xscale,yscale);
_plegctl={2,5,6,4.3};
_plegstr="Uniform¥000Gauss¥000Triangular¥000Biweight¥000Triweight
¥000Epanechnikov";
xy(u,kv1~kv2~kv3~kv4~kv5~kv6);
```

グラフ表示



いま、上のそれぞれの Kernel のタイプを用いて、もう少し滑らかでないデータをｎを小さくすることで作成して、density をそのヒストグラムとともに描いてみることにする。基準となるポイントを最小のポイントから最大のポイントまで少しずつ動かすときに、その動かす間隔を今までよりも密にするために 100 等分ではなくて 1000 等分してやることにしよう（実際には、この間隔の大きさにより uniform の kernel でグラフに違いが見られる）。下のプログラムでは、procedure の外部でｘをシミュレートして作成したものをデータにしてヒストグラムと density を求める。ｘのところには、ｎ×１の実際のデータが来てもプログラムは動くようになっている。

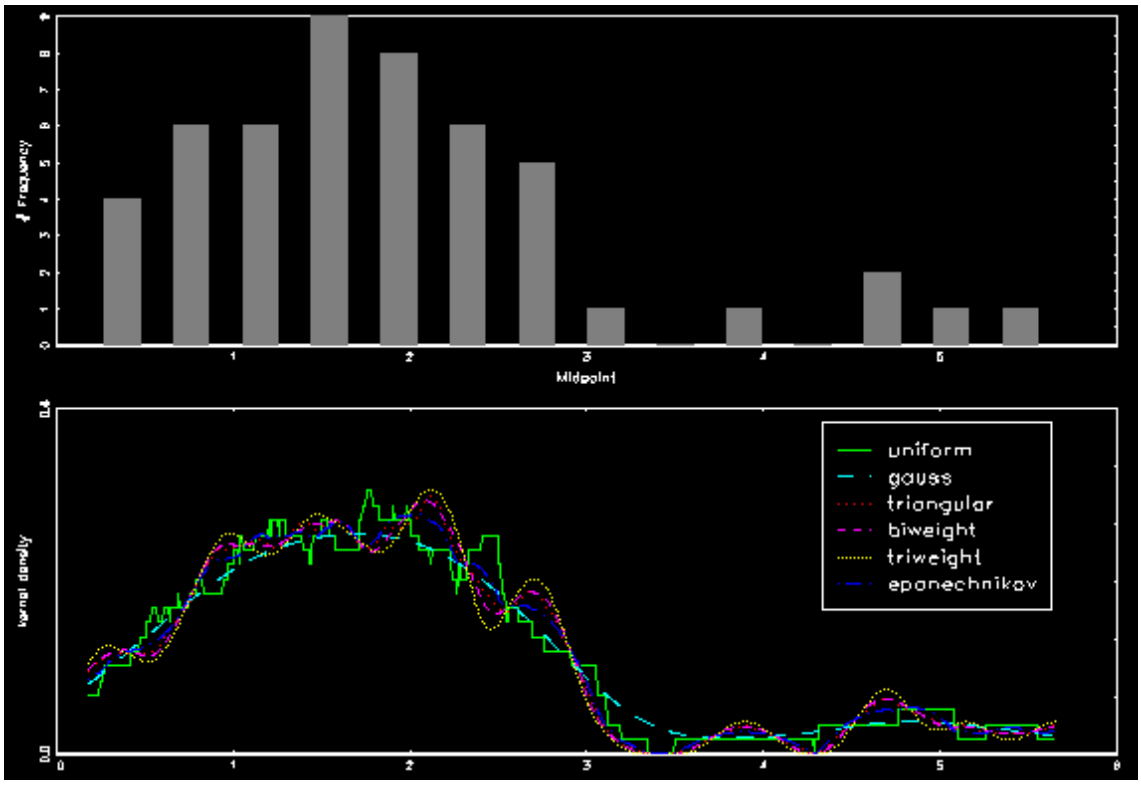
プログラム

```
new; cls;
```

```
library pgraph;
graphset;
rndseed 88;
alpha=2; n=50;
x=rndgam(n,1,alpha);
call plotdn(x);

proc plotdn(x);
    local n,h,points,i,kern;
    n=rows(x);
    x=sortc(x,1);
    h=0.9*minc(stdc(x)|((x[round(0.75*n)]-x[round(0.25*n)])/1.34))*n^(-0.2);
    points=seqa(x[1],(x[n]-x[1])/1000,1001);
    i=1; kern=zeros(1001,6);
    do while i<=1001;
        kern[i,1]=kernels(points[i],x,h,"uniform");
        kern[i,2]=kernels(points[i],x,h,"gauss");
        kern[i,3]=kernels(points[i],x,h,"triangular");
        kern[i,4]=kernels(points[i],x,h,"biweight");
        kern[i,5]=kernels(points[i],x,h,"triweight");
        kern[i,6]=kernels(points[i],x,h,"epanechnikov");
        i=i+1;
    endo;
    begwind;
    window(2,1,0);
    setwind(1);
        xtics(0,ceil(x[n]),1,5);
        hist(x,ceil((x[n]-x[1])/h));
    setwind(2);
        graphset;
        _plegctl={2,9,6.5,3};
        _plegstr="uniform¥000gauss¥000triangular¥000biweight¥000triweight¥000ep
anechnikov";
        xtics(0,ceil(x[n]),1,5);
        ytics(0,0.6,0.6,6);
        ylabel("kernel density");
```

```
        xy(points,kern);
    endwind;
    retp(kern);
endp;

proc kernels(x0,x,h,types);
    local u,kv;
    u=(x0-x)/h;
    if types$=="uniform";
        kv=0.5*(abs(u).<=1);
    elseif types$=="gauss";
        kv=(1/sqrt(2*pi))*exp(-0.5*u^2);
    elseif types$=="triangular";
        kv=(1-abs(u)).*(abs(u).<=1);
    elseif types$=="biweight";
        kv=(15/16)*((1-u^2)^2).*(abs(u).<=1);
    elseif types$=="triweight";
        kv=(35/32)*((1-u^2)^3).*(abs(u).<=1);
    elseif types$=="epanechnikov";
        kv=0.75*(1-u^2).*(abs(u).<=1);
    else;
        errorlog "ERROR:    Type must be either of the following:";
        errorlog "uniform/gauss/triangular/biweight/triweight/epanechnikov";
    endif;
    retp( sumc(kv/(rows(x)*h)) );
endp;

proc pdfgam(x,a);
    retp( 1/gamma(a).*x^(a-1).*exp(-x) );
endp;
```

グラフ表示



グラフの上段はヒストグラム、下段は便宜上同じ長さのバンド幅を用いた場合の各 kernel による density のグラフである。もちろん、nをもっと大きくしてやれば、もとになる分布自体が滑らかなΓ分布にしたがうものになるので、これまでのグラフのように各 kernel の density は真の density に一致してくると同時に、お互いの density も一致してくる。この場合は、それをわざと粗い散らばりを作成したものである。