6.1 フィルター　トレンドとサイクルの分解　　ver.0.1

厳密には、ある状態が時系列の観測データの先にある場合を予測と呼び、その状態が観測データの終わりの時点と同じ場合をフィルタと呼び、その状態が観測データのどこかに位置する場合を平滑化と呼ぶ。しかしながら、ここでは、そのうちのフィルタと平滑化を大きく分類して１つとして、これにまつわるいろいろな事例を見ていくことにしよう。

実際には単位根の問題や成長率を問題にすべき点から、対数階差を原系列データにほどこしてから分析すべきであるが、以下では線形トレンドなど直感的結果を得るために原系列をあえて最初のうちは使うことにする。いま D:に datafile17.txt があるものとする。

Hodrick-Prescott フィルターと線形トレンドフィルター

Hodrick-Prescott フィルターとは、成長コンポーネントｇと循環コンポーネントｃに分解するには、次のような最小化問題を考える。

$$\underset{\{g_{-1},g_0,g_1,\cdots\cdots,g_T\}}{Min}\{\sum_{t=1}^{T}(y_t-g_t)^2+\lambda\sum_{t=1}^{T}[(g_t-g_{t-1})-(g_{t-1}-g_{t-2})]^2\}$$

λがある値を取るとき、ｇt に関する最小化を行なったものである。このとき FOC は、最初から全体を２分の１しておいて関数の微分の２乗をあらかじめ消しておいて、ｇ側をすべて反対側に移行してマイナスをつけてやれば、

FOC:

$y_1 = g1+\lambda(\ g1\ -2*g2\ +\ g3\ )$
$y_2 = g2+\lambda(-2*g1\ +5*g2\ -4*g3\ +\ g4\ )$
$y_3 = g3+\lambda(\ g1\ -4*g2\ +6*g3\ -4*g4\ +\ g5)$
：　　：
：　　：

とすると、ｙとｇをそれぞれＴ×１の列ベクトルとすると

$y = Ag$

とかける。すなわち、Ａは係数部分だけの行列で

$$A=\begin{pmatrix}1+\lambda & -2\lambda & \lambda & 0 & \cdots & 0 & 0 & 0\\ -2\lambda & 1+5\lambda & -4\lambda & \lambda & & 0 & 0 & 0\\ \lambda & -4\lambda & 1+6\lambda & -4\lambda & & \vdots & \vdots & 0\\ 0 & \lambda & -4\lambda & 1+6\lambda & & 0 & 0 & \vdots\\ \vdots & 0 & 0 & & \ddots & -4\lambda & \lambda & 0\\ 0 & \vdots & \vdots & & -4\lambda & 1+6\lambda & -4\lambda & \lambda\\ 0 & 0 & 0 & & \lambda & -4\lambda & 1+5\lambda & -2\lambda\\ 0 & 0 & 0 & \cdots & 0 & \lambda & -2\lambda & 1+\lambda\end{pmatrix}$$

となりｇの列ベクトルにかかる。このλに関する係数行列Ａの逆行列をとって、

$A^{-1}y = g$

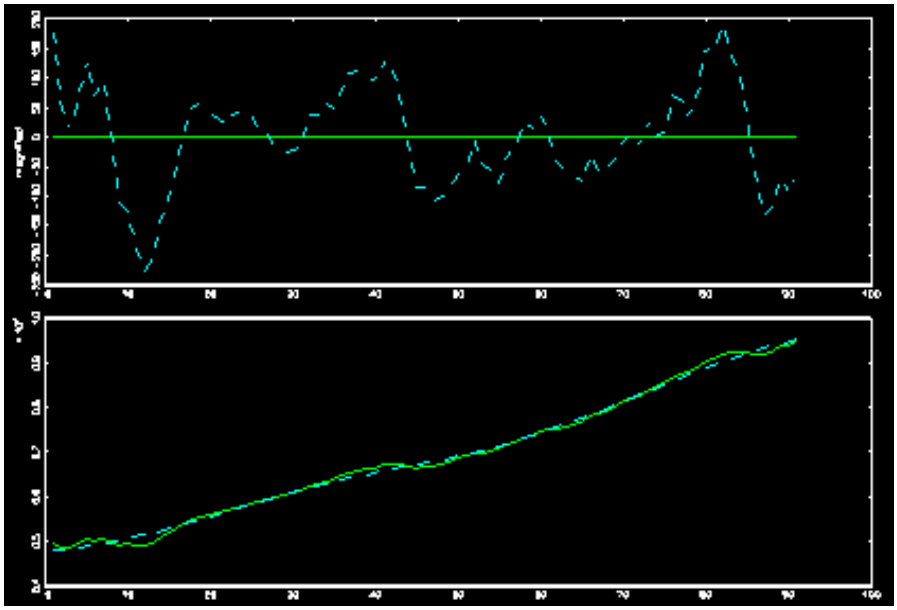
として g の T × 1 の列ベクトルを求めるものが逆行列による Hodrick-Prescott フィルターの導出方法である。以下は、A の行列を作るプログラムで最終的に逆行列計算をしている。

プログラム

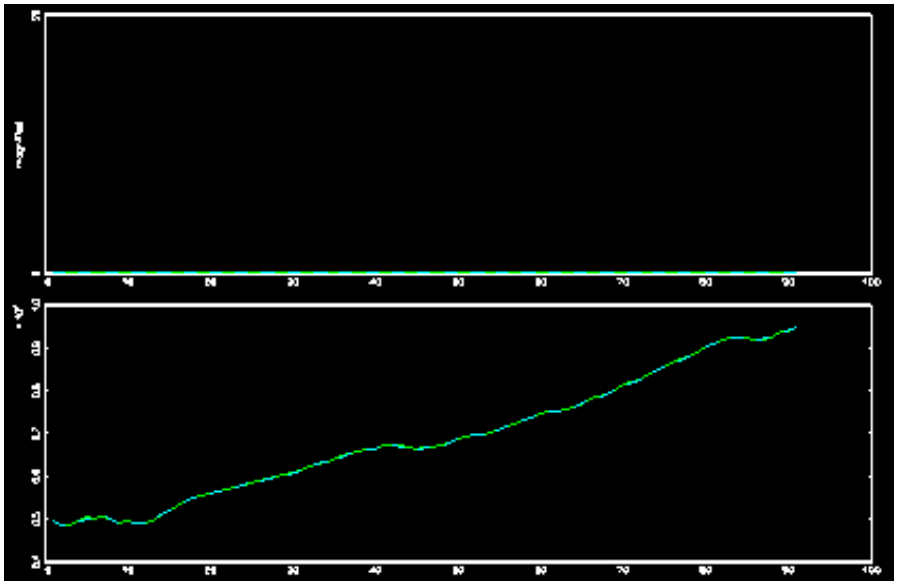
```
new; cls;
load data[91,2]=D:¥datafile17.txt;
y=data[.,2];

{g,c}=hp(y,1600);
library pgraph;
begwind;
window(2,1,1);
setwind(1);
    graphset;
    ylabel("magnified");
    xy(seqa(1,1,rows(y)),zeros(rows(y),1)~c);
setwind(2);
    graphset;
    xy(seqa(1,1,rows(y)),y~g);
endwind;

proc(2)=hp(y,lambda);
    local t,a,i,g,c;
    t=rows(y);
    a=zeros(t,t);
    i=3;
    do while i<=t-2;
        a[i,i-2]=lambda;
        a[i,i-1]=-4*lambda;
        a[i,i]=1+6*lambda;
        a[i,i+1]=-4*lambda;
        a[i,i+2]=lambda;
        i=i+1;
    endo;
    a[1,1]=1+lambda;     a[t,t]=1+lambda;
```

```
    a[1,2]=-2*lambda;    a[t,t-1]=-2*lambda;
    a[1,3]=lambda;       a[t,t-2]=lambda;
    a[2,1]=-2*lambda;    a[t-1,t]=-2*lambda;
    a[2,2]=1+5*lambda; a[t-1,t-1]=1+5*lambda;
    a[2,3]=-4*lambda;    a[t-1,t-2]=-4*lambda;
    a[2,4]=lambda;       a[t-1,t-3]=lambda;
    g=inv(a)*y;
    c=y-g;
    retp(g,c);
endp;
```

グラフ表示



グラフ表示（λ＝0のケース）



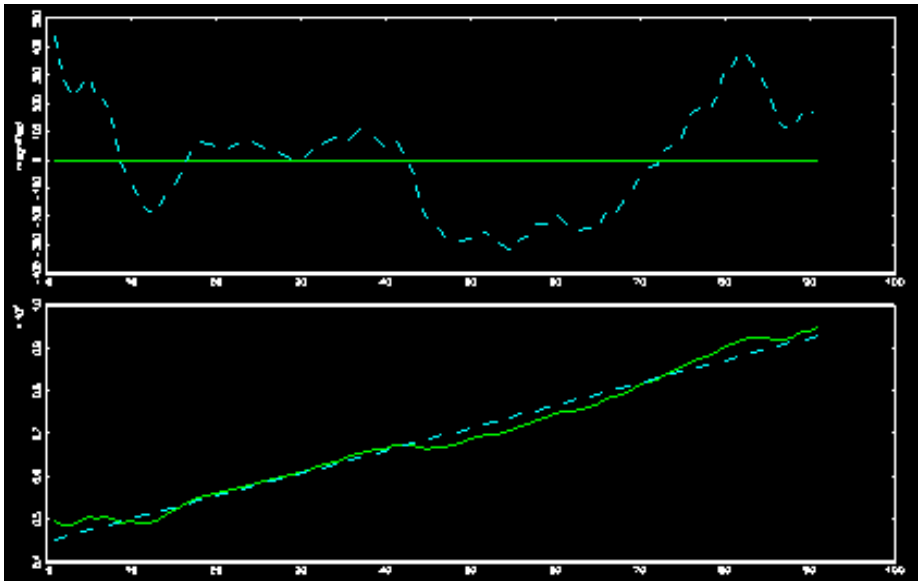
上のグラフは四半期データに対するλ＝1600 をλ＝０に変更したものである。理論上、原系列に等しくなるはずであるが、実際 cyclical コンポーネントは０となっていて、原系列に一致した結果となっている。

プログラム

```
new; cls;
load data[91,2]=D:¥datafile17.txt;
y=data[.,2];

{g1,c1}=hp(y,1e+10);
{g2,c2}=ltrend(y);
library pgraph;
begwind;
window(2,1,1);
setwind(1);
    graphset;
    ylabel("magnified");
    xy(seqa(1,1,rows(y)),zeros(rows(y),1)~c1);
setwind(2);
    graphset;
    xy(seqa(1,1,rows(y)),y~g1);
endwind;
print g2-g1;

proc(2)=hp(y,lambda);
    local t,a,i,g,c;
    t=rows(y);
    a=zeros(t,t);
    i=3;
    do while i<=t-2;
        a[i,i-2]=lambda;
        a[i,i-1]=-4*lambda;
        a[i,i]=1+6*lambda;
        a[i,i+1]=-4*lambda;
        a[i,i+2]=lambda;
        i=i+1;
```

```
    endo;
    a[1,1]=1+lambda;      a[t,t]=1+lambda;
    a[1,2]=-2*lambda;     a[t,t-1]=-2*lambda;
    a[1,3]=lambda;        a[t,t-2]=lambda;
    a[2,1]=-2*lambda;     a[t-1,t]=-2*lambda;
    a[2,2]=1+5*lambda; a[t-1,t-1]=1+5*lambda;
    a[2,3]=-4*lambda;     a[t-1,t-2]=-4*lambda;
    a[2,4]=lambda;        a[t-1,t-3]=lambda;
    g=(inv(a))*y;
    c=y-g;
    retp(g,c);
endp;

proc(2)=ltrend(y);
    local t,x,b,g,c;
    t=rows(y);
    x=ones(t,1)~seqa(1,1,t);
    b=inv(x'x)*x'y;
    g=x*b;
    c=y-g;
    retp(g,c);
endp;
```

グラフ表示（λが十分に大きいケース）



画面表示は省略するが、Hodrick-Prescott フィルターと線形トレンドのフィルターとの差をとると、その結果は、ほぼ０のまわりの数になっている。ただし、計算精度を超えてあまりにも大きな数にするとプログラムの行列計算自体が意味を持たなくなり、その極限が線形のフィルターであることは示せないが、理論上はλ＝∞のケースでは、線形トレンドと同じことになる。この場合も、計算精度いっぱいにλを大きくすると、線形トレンドとの差がほぼ０のまわりに散らばる結果となる。λの値は滑らかさの度合いと考えられ、

λ＝	０	そのまま
	１６００	四半期
	１４４００	月次
	∞	線形トレンド

というふうに、λが大きくなるにつれてもとの系列を滑らかに両端から引っ張られて最終的にはその極限で一直線の線形トレンドになる。なお、Ravn-Uhlig によると、四半期のλ＝1600 を基準にして、$\lambda \times n^{1/4}$ の補正関係式が与えられていて、それぞれの期のλは

１６００×$(1/4)^4$＝６．２５	年次
１６００×　３＝１２９６０	月次

とされている。年次データにはλ＝４で行なっている論文も見られる。年次がλ＝１００でやるのには大きすぎると考えるべきではあるが、Hodrick-Prescott 原論文による限りでは、λの値によってそう結果が変化するものではないことも事実である。

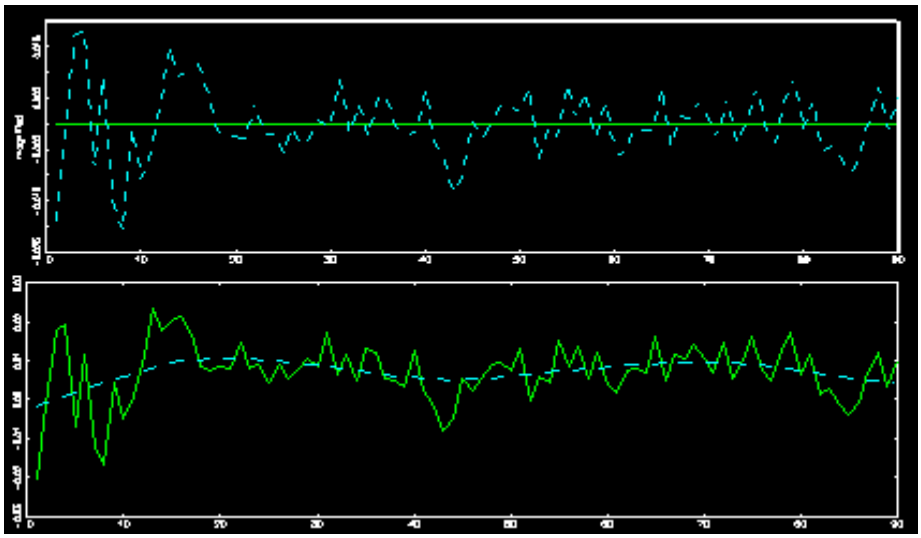
なお、プログラムの後半に置いた ltrend は定数項つきの一直線の線形トレンドを推定する procedure である。定数項には１ばかりからなる列データが、トレンド推定には１からそのデータの個数分までの数列がくる。Hodrick-Prescott フィルターを全く同じようにして使うことが可能である。

慣習的な対数階差をとる方法にするには、

```
n=rows(y);
y=ln(y);
y=y[2:n,.]-y[1:n-1,.];
```

の３行を冒頭のｙの式の後に加えればよい。その場合の Hodrick-Prescott フィルターのグラフ結果は次のようになる。

グラフ表示



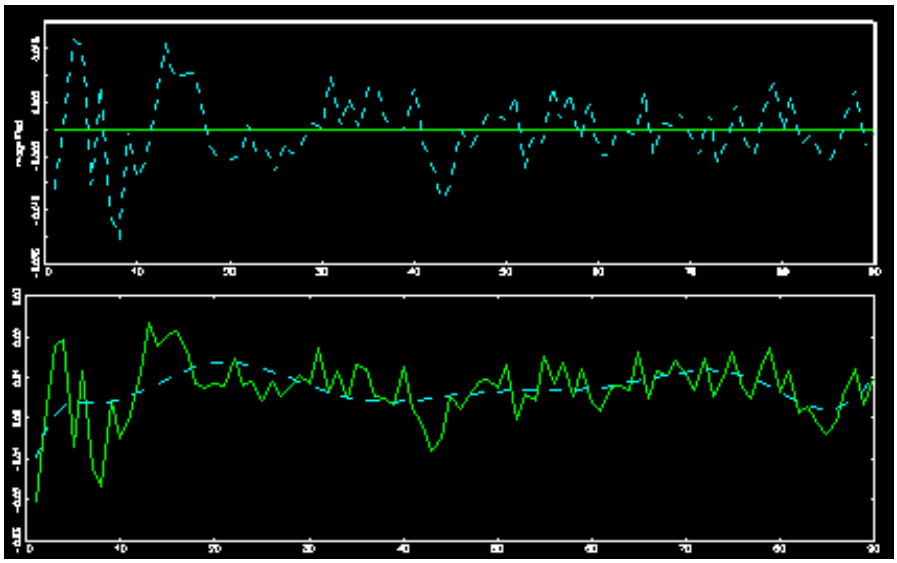
Polynomial トレンドフィルター

プログラム

```
new; cls;
load data[91,2]=D:¥datafile17.txt;
y=(data[.,2]);
n=rows(y);
y=ln(y);
y=y[2:n,.]-y[1:n-1,.];

{g,c}=ptrend(y,10);
library pgraph;
begwind;
window(2,1,1);
setwind(1);
    graphset;
    ylabel("magnified");
    xy(seqa(1,1,rows(y)),zeros(rows(y),1)~c);
setwind(2);
    graphset;
    xy(seqa(1,1,rows(y)),y~g);
endwind;
```

```
proc(2)=ptrend(y,order);
    local t,xt,x1,i,x,b,c,g;
    t=rows(y);
    xt=seqa(1,1,t);
    x1=seqa(1,1,t);
    i=2;
    do while i<=order;
        xt=xt~(x1^i);
        i=i+1;
    endo;
    x=ones(t,1)~xt;
    b=inv(x'x)*x'y;
    g=x*b;
    c=y-g;
    retp(g,c);
endp;
```

グラフ表示



上のケースは多項トレンドの次数を 10 と定めたものである。次数選択は敢えてつけていない。なお、当然のことながら次数(Order)を 1 とすると線形トレンドと全く同一となる。プログラムの性質上、次数は、このままの計算では計算精度を超えて無限大に高次のタイム項が指数的に増すのでスケーリングをしないのであれば、10 前後までしかあげられない。

Baxter-King フィルター

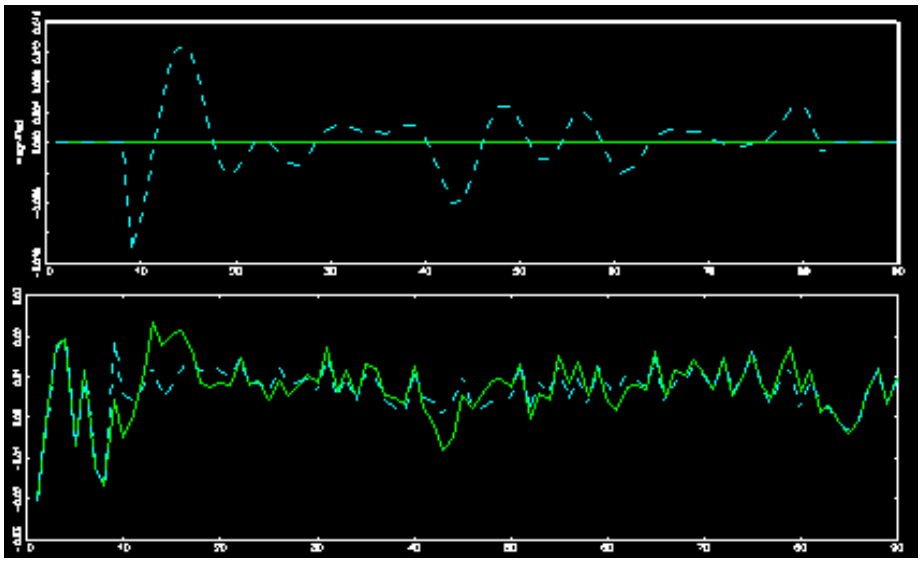
このフィルターは Band Pass フィルターと呼ばれるものの一種で、普通とらえるべき６四半期から３２四半期のサイクルを捕らえるために、その下限（高バンド）と上限（低バンド）の数を６と３２と設定しておいて、それに対する三角関数の定数を次数のｋ＋１だけ求めておいて、それをもとに中央からはじめてｋ次分の左右対称な次数の移動移動平均すなわち、２ｋ＋１の期間の対称中央移動平均を行なう。以下のプログラムでは、ｋ＝８にしてあるが、ｋ＝１２でやられることも多い。なお、procedure の前半はそのｋ＋１の定数を高バンド数と低バンド数から計算している。通常は 1000 以上に設定されることはないので、phi の If 分岐は必要なければ取り除いてもよい。後半は、ｋ＋１個の定数を中央に対称に２ｋ＋１分ならべたベクトルを原系列にかけて中央を対称にする逐次移動平均をやっているだけである。ウエイトベクトルは、If 分岐で 1000 以上にならないかぎり、通常は足して合計が０になるものであって、これを逐次移動平均するために原系列にかけるとサイクルが取り出される。Hodrick-Prescott フィルターの場合には、逆行列を原系列にかけるとサイクルではなくトレンドが取り出されるので、違いに注意されたい。

プログラム

```
new; cls;
load data[91,2]=D:¥datafile17.txt;
y=(data[.,2]);
n=rows(y);
y=ln(y);
y=y[2:n,.]-y[1:n-1,.];

{g,c}=bk(y,6,32,8);
library pgraph;
begwind;
window(2,1,1);
setwind(1);
    graphset;
    ylabel("magnified");
    xy(seqa(1,1,rows(y)),zeros(rows(y),1)~c);
setwind(2);
    graphset;
    xy(seqa(1,1,rows(y)),y~g);
endwind;
```

```
proc(2)=bk(y,high,low,k);
    local t,a,i,phi,theta,b,c,g;
    t=rows(y);
    a=zeros(k+1,1);
    a[1]=(2*pi/high-2*pi/low)/pi;
    i=1;
    do while i<=k;
        a[i+1]=(sin(i*2*pi/high)-sin(i*2*pi/low))/(i*pi);
        i=i+1;
    endo;
    if low>1000;
        phi=1;
    else;
        phi=0;
    endif;
    theta=a[1]+2*sumc(a[2:k+1]);
    theta=phi-theta/(2*k+1);
    a=a+theta;
    b=zeros(2*k+1,1);
    b[k+1]=a[1];
    i=1;
    do while i<=k;
        b[k+1-i]=a[i+1]; b[k+1+i]=a[i+1];
        i=i+1;
    endo;
    c=zeros(t,1);
    i=k+1;
    do while i<=t-k;
        c[i]=b'y[i-k:i+k,.];
        i=i+1;
    endo;
    g=y-c;
    retp(g,c);
endp;
```

グラフ表示



ただし、最初と最後のそれぞれｋ期だけは計算されていない。必要であればカットされたい。ｋの長さはデータの全体の長さを損ねない範囲で長くなればなるほど、そのサイクルにあてはまりはよくなるはずであるが、前後それぞれｋ期分だけ計算はなされないことになる。もともと、移動平均をもとにするフィルターは Cyclical コンポーネントに見せかけのダイナミクスを生み出す傾向にあるということが報告されていて、その点は避けがたい注意点である。ただし、このフィルターでも６期から３２期の外はカットすることができるからサイクルを取り出すのみであれば季節調整は必要はない。しかしながら、もとデータは季節調整をしていないのであれば、少なくともトレンドの方からは４期以内の季節変動は分離できない。その場合には、中央化移動平均を行なえば純粋なトレンド＋サイクルが得られるから、そこからはじめれば次のようになる。

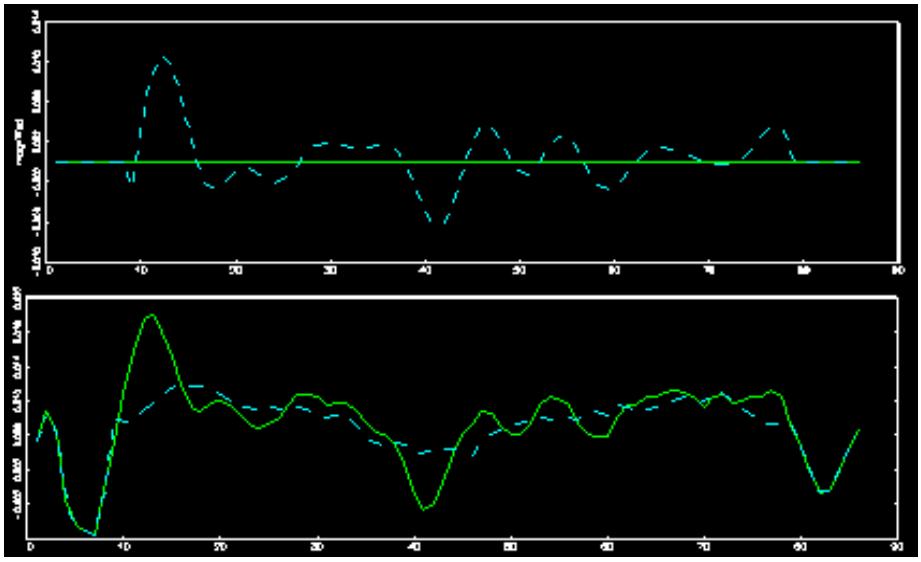
プログラム

```
new; cls;
load data[91,2]=D:¥datafile17.txt;
y=(data[.,2]);
y=cma(y);
n=rows(y);
y=ln(y);
y=y[2:n,.]-y[1:n-1,.];

{g,c}=bk(y,6,32,8);
library pgraph;
begwind;
window(2,1,1);
```

```
setwind(1);
    graphset;
    ylabel("magnified");
    xy(seqa(1,1,rows(y)),zeros(rows(y),1)~c);
setwind(2);
    graphset;
    xy(seqa(1,1,rows(y)),y~g);
endwind;

proc(2)=bk(y,high,low,k);
    local t,a,i,phi,theta,b,c,g;
    t=rows(y);
    a=zeros(k+1,1);
    a[1]=(2*pi/high-2*pi/low)/pi;
    i=1;
    do while i<=k;
        a[i+1]=(sin(i*2*pi/high)-sin(i*2*pi/low))/(i*pi);
        i=i+1;
    endo;
    if low>1000;
        phi=1;
    else;
        phi=0;
    endif;
    theta=a[1]+2*sumc(a[2:k+1]);
    theta=phi-theta/(2*k+1);
    a=a+theta;
    b=zeros(2*k+1,1);
    b[k+1]=a[1];
    i=1;
    do while i<=k;
        b[k+1-i]=a[i+1]; b[k+1+i]=a[i+1];
        i=i+1;
    endo;
    c=zeros(t,1);
    i=k+1;
```

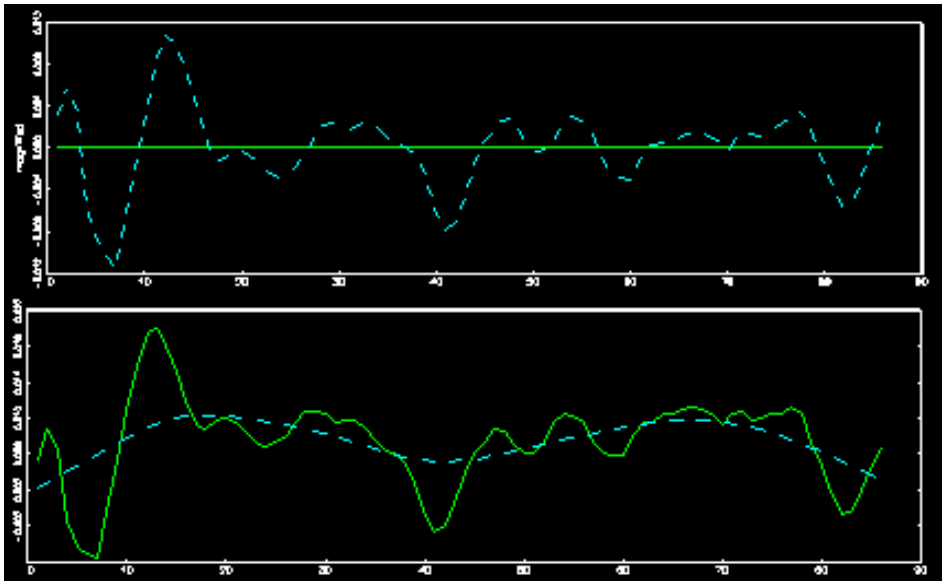
```
    do while i<=t-k;
        c[i]=b'y[i-k:i+k,.];
        i=i+1;
    endo;
    g=y-c;
    retp(g,c);
endp;

proc cma(y);
    local t,k,m,i,m1,m2;
    t=rows(y);
    k=cols(y);
    m=zeros(t-4,k);
    i=1;
    do while i<=t-4;
        m1=(y[i,.]+y[i+1,.]+y[i+2,.]+y[i+3,.])/4;
        m2=(y[i+1,.]+y[i+2,.]+y[i+3,.]+y[i+4,.])/4;
        m[i,.]=(m1+m2)/2;
        i=i+1;
    endo;
    retp(m);
endp;
```

グラフ表示



もちろん、Hodrick-Prescott フィルターにもこのことはあてはまるので、同じことをやって結果だけを示すと次のようになる。

グラフ表示(Hodrick-Prescott フィルターのケース)



つまり、中央化移動平均では、最初の４四半期を平均したものと１期ずらした次の４四半期を平均したものとの２つをさらに平均したものを最初から数えて３期目のトレンド＋サイクルの成分とすることを繰り返すものである。

ここで、少し季節調整のプログラムに立ち入ることにしよう。以下は、この中央化移動平均をさらに進めて、原系列から４期以内の（１年以内の）季節変動を除いて、トレンド＋サイクル＋不規則変動の３つの成分からなる系列に調整しようというものである。以下四半期データのみを扱うものとして手順を説明する。

加法モデルのケース

Ｙ＝Ｔ＋Ｃ＋Ｓ＋Ｉからなる原系列から中央化移動平均によって、Ｔ＋Ｃを取り出す。次に、ＹからＴ＋Ｃを引いてＳ＋Ｉを取り出す。ここには不規則変動成分Ｉが含まれているので、四半期ごと、つまり４つおきのデータの平均をとってそれを季節変動成分Ｓとする。原系列ＹからＳを引けば、最終的に季節調整済みのデータＴ＋Ｃ＋Ｉになる。

乗法モデルのケース

Ｙ＝Ｔ＊Ｃ＊Ｓ＊Ｉからなる原系列から中央化幾何平均によって、Ｔ＊Ｃを取り出す。次に、Ｙ÷（Ｔ＊Ｃ）を計算して比率にするため必要なら 100 倍する。これをＳ＊Ｉとする。ここから四半期ごとの幾何平均をとって季節変動指数として、さらにこの合計が 400 になるように調整する。原系列Ｙをこの調整された季節変動指数で割ってやって最終的に季節調整済みのデータＴ＊Ｃ＊Ｉが得られる。この場合、前に 100 倍してあるので必要であればここでも 100 倍する。なお、乗法モデルでは幾何平均を使うべきではあるが、簡易に通常の算術平均で同じ計算をすることもある。以下では、その２つも区別して示す。

プログラム

```
new; cls;
load data[91,2]=D:¥datafile17.txt;
y=data[.,2];
library pgraph;
graphset;
print "Seasonal Component(if any):";;
print (y-addsadj(y))~(y-mulsadj(y))~(y-geosadj(y));

proc addsadj(y);
    local t,k,m,i,m1,m2,si,s1,s2,s3,s4,n,s,adjy;
    t=rows(y);
    k=cols(y);
    m=zeros(t-4,k);
    i=1;
    do while i<=t-4;
        m1=(y[i,.]+y[i+1,.]+y[i+2,.]+y[i+3,.])/4;
        m2=(y[i+1,.]+y[i+2,.]+y[i+3,.]+y[i+4,.])/4;
        m[i,.]=(m1+m2)/2;
        i=i+1;
    endo;
    si=y[3:t-2,.]-m;
    s1=0; s2=0; s3=0; s4=0;
    i=1;
    do while i<=t-7;
        s3=s3+si[i,.];
        s4=s4+si[i+1,.];
        s1=s1+si[i+2,.];
        s2=s2+si[i+3,.];
        i=i+4;
    endo;
    n=floor((t-4)/4);
    s1=s1/n; s2=s2/n; s3=s3/n; s4=s4/n;
    s=s1|s2|s3|s4;
    i=1;
    do while i<=n+1;
```

```
        s=s|(s1|s2|s3|s4);
        i=i+1;
    endo;
    s=s[1:t];
    adjy=y-s;
    retp(adjy);
endp;

proc mulsadj(y);
    local t,k,m,i,m1,m2,si,s1,s2,s3,s4,n,s,adjy;
    t=rows(y);
    k=cols(y);
    m=zeros(t-4,k);
    i=1;
    do while i<=t-4;
        m1=(y[i,.]+y[i+1,.]+y[i+2,.]+y[i+3,.])/4;
        m2=(y[i+1,.]+y[i+2,.]+y[i+3,.]+y[i+4,.])/4;
        m[i,.]=(m1+m2)/2;
        i=i+1;
    endo;
    si=y[3:t-2,.]./m*100;
    s1=0; s2=0; s3=0; s4=0;
    i=1;
    do while i<=t-7;
        s3=s3+si[i,.];
        s4=s4+si[i+1,.];
        s1=s1+si[i+2,.];
        s2=s2+si[i+3,.];
        i=i+4;
    endo;
    n=floor((t-4)/4);
    s1=s1/n; s2=s2/n; s3=s3/n; s4=s4/n;
    s=s1|s2|s3|s4;
    s=400*s/sumc(s);
    i=1;
    do while i<=n+1;
```

```
        s=s|(s1|s2|s3|s4);
        i=i+1;
    endo;
    s=s[1:t];
    adjy=y./s*100;
    retp(adjy);
endp;

proc geosadj(y);
    local t,k,m,i,m1,m2,si,s1,s2,s3,s4,n,s,adjy;
    t=rows(y);
    k=cols(y);
    m=zeros(t-4,k);
    i=1;
    do while i<=t-4;
        m1=(y[i,.].*y[i+1,.].*y[i+2,.].*y[i+3,.])^(1/4);
        m2=(y[i+1,.].*y[i+2,.].*y[i+3,.].*y[i+4,.])^(1/4);
        m[i,.]=(m1.*m2)^(1/2);
        i=i+1;
    endo;
    si=y[3:t-2,.]./m*100;
    s1=1; s2=1; s3=1; s4=1; @ Not 0 but all 1 for multiplication. @
    i=1;
    do while i<=t-7;
        s3=s3.*si[i,.];
        s4=s4.*si[i+1,.];
        s1=s1.*si[i+2,.];
        s2=s2.*si[i+3,.];
        i=i+4;
    endo;
    n=floor((t-4)/4);
    s1=s1^(1/n); s2=s2^(1/n); s3=s3^(1/n); s4=s4^(1/n);
    s=s1|s2|s3|s4;
    s=400*s/sumc(s); @ Partially an arithmetic mean to use 400. @
    i=1;
    do while i<=n+1;
```

```
        s=s|(s1|s2|s3|s4);
        i=i+1;
    endo;
    s=s[1:t];
    adjy=y./s*100;
    retp(adjy);
endp;
```

上では３つまとめて示しているが、実際にはデータが季節調整がなされていない場合に３つのうちのどれかかを行なえばよい。1 つ目は加法モデルによるもの。２つ目と３つ目は乗法モデルによるもので、３つ目は 400 でならすところ以外は厳密に幾何平均で計算している。上の画面表示は省略するが、このデータが仮に季節調整をしていないとすれば、このデータの場合、季節成分は１期目はほぼ０で、2 期目と４期目はプラスで、３期目だけが大幅にマイナスになるという季節成分のパターンがある。実際に季節調整済みにするには

```
y=addsadj(y);
```

というふうにまだ変換を何もしていないところに１行を置き、該当する関数を末尾につければよい。ただし、こうするとトレンド＋サイクル以外に不規則変動成分も含まれる。

Christiano-Fitzgerald フィルター

Baxter-King フィルターにはデータの前後それぞれｋ期ずつの欠損値が生じる欠点があったが、Chrisitino-Fitsgerald フィルターにはそれがない。途中までは、Baxter-King のものと同じ Band-Pass を計算する。下限の高バンドと上限の低バンドを通常通り６と３２に設定すれば同じである。そのあと、後半でＴ×Ｔの行列展開をしてすべてのディメンションで原系列にかけ合わせてサイクルを取り出す。ただし、Ｔ×Ｔの最初と最後の列にはプログラムの最後で補正を行なう部分がある。

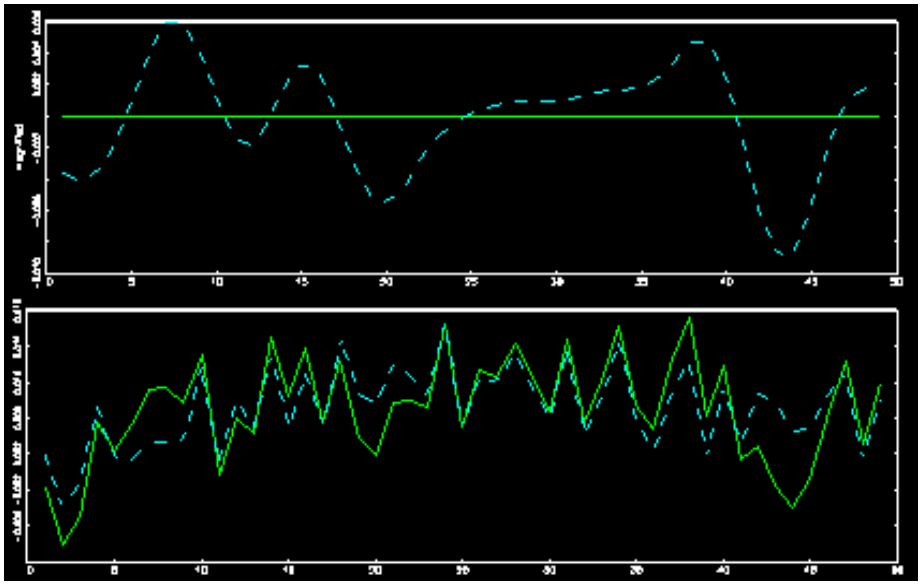
プログラム

```
load data[91,2]=D:¥datafile17.txt;
y=data[42:91,2];                    @ Last 50 @
n=rows(y);
y=ln(y);
y=y[2:n,.]-y[1:n-1,.];
/*
**  This procedure requires lots of memory to expand matrix.
**  Cut data to the last 50 of them for GAUSS Light version.
**  This procedure replicates the official version of Christiano-
**  Fitzgerald Filter by Eduard Pelz but slightly modified to
**  compare with Baxter-King Filter. The results should be the
```

```
**    same as the official program I have made sure.
*/
{g,c}=cf(y,6,32);
library pgraph;
begwind;
window(2,1,1);
setwind(1);
    graphset;
    ylabel("magnified");
    xy(seqa(1,1,rows(y)),zeros(rows(y),1)~c);
setwind(2);
    graphset;
    xy(seqa(1,1,rows(y)),y~g);
endwind;

proc(2)=cf(y,high,low);
    local t,drift,x,i,B,A,bhatu,bhatd,c,g;
    t=rows(y);
    drift=(y[t,.]-y[1,.])/(t-1);
    x=y;
    i=1;
    do while i<=t;
        x[i,.]=y[i,.]-(i-1)*drift;
        i=i+1;
    endo;
    B=zeros(t,1);
    B[1]=(2*pi/high-2*pi/low)/pi;
    i=1;
    do while i<=t-1;
        B[i+1]=(sin(i*2*pi/high)-sin(i*2*pi/low))/(i*pi);
        i=i+1;
    endo;
    A=zeros(2*t,2*t);
    i=1;
    do while i<=t;
        A[i:i+t-1,i]=B; A[i,i:i+t-1]=B';
```

```
        i=i+1;
    endo;
    A=A[1:t,1:t];
    bhatu=zeros(t,1);
    bhatd=zeros(t,1);
    bhatu[1]=B[1]/2;
    bhatd[t]=B[1]/2;
    i=1;
    do while i<=t-1;
        bhatu[i+1]=bhatu[i]-B[i];
        bhatd[t-i]=bhatu[i]-B[i];
        i=i+1;
    endo;
    A[1:t,1]=bhatu;
    A[1:t,t]=bhatd;
    c=A*x;
    g=y-c;
    retp(g,c);
endp;
```

画面表示（最後の５０データのみ）



こちらは、あらかじめドリフト項を取り除く作業をしていることを除けば、基本的には Baxter-King と同じ Band Pass フィルターの計算をしている。しかしながら、こちらはすべてのデータをフィルタリングしている。そのため、行列展開があってメモリを食う。

データが季節調整済みでないと仮定して、中央化移動平均でトレンド＋サイクルを純粋に取り出して、同じ分析を行なうと同様に次のようになる。

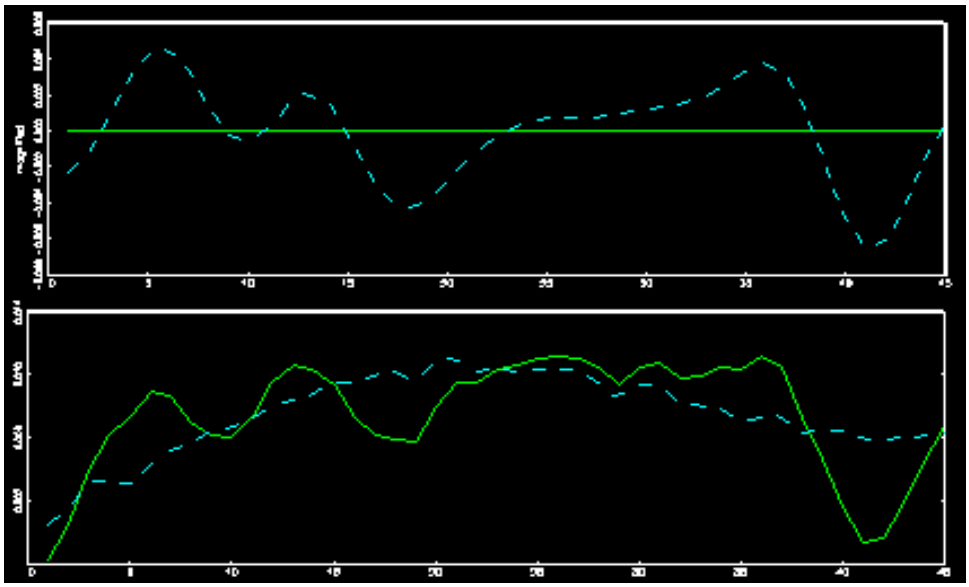
プログラム

```
new; cls;
load data[91,2]=D:¥datafile17.txt;
y=data[42:91,2];                    @ Last 50 @
y=cma(y);
n=rows(y);
y=ln(y);
y=y[2:n,.]-y[1:n-1,.];
/*
**    This procedure requires lots of memory to expand matrix.
**    Cut the data to the last 50    of them for GAUSS Light.
**    This procedure replicates the official version of Christiano-
**    Fitzgerald Filter by Eduard Pelz but slightly modified to
**    compare with Baxter-King Filter. The results should be the
**    the same as the official program I have made sure.
*/
{g,c}=cf(y,6,32);
library pgraph;
begwind;
window(2,1,1);
setwind(1);
    graphset;
    ylabel("magnified");
    xy(seqa(1,1,rows(y)),zeros(rows(y),1)~c);
setwind(2);
    graphset;
    xy(seqa(1,1,rows(y)),y~g);
endwind;

proc(2)=cf(y,high,low);
    local t,drift,x,i,B,A,bhatu,bhatd,c,g;
    t=rows(y);
    drift=(y[t,.]-y[1,.])/(t-1);
    x=y;
```

```
    i=1;
    do while i<=t;
        x[i,.]=y[i,.]-(i-1)*drift;
        i=i+1;
    endo;
    B=zeros(t,1);
    B[1]=(2*pi/high-2*pi/low)/pi;
    i=1;
    do while i<=t-1;
        B[i+1]=(sin(i*2*pi/high)-sin(i*2*pi/low))/(i*pi);
        i=i+1;
    endo;
    A=zeros(2*t,2*t);
    i=1;
    do while i<=t;
        A[i:i+t-1,i]=B; A[i,i:i+t-1]=B';
        i=i+1;
    endo;
    A=A[1:t,1:t];
    bhatu=zeros(t,1);
    bhatd=zeros(t,1);
    bhatu[1]=B[1]/2;
    bhatd[t]=B[1]/2;
    i=1;
    do while i<=t-1;
        bhatu[i+1]=bhatu[i]-B[i];
        bhatd[t-i]=bhatu[i]-B[i];
        i=i+1;
    endo;
    A[1:t,1]=bhatu;
    A[1:t,t]=bhatd;
    c=A*x;
    g=y-c;
    retp(g,c);
endp;
```

```
proc cma(y);
    local t,k,m,i,m1,m2;
    t=rows(y);
    k=cols(y);
    m=zeros(t-4,k);
    i=1;
    do while i<=t-4;
        m1=(y[i,.]+y[i+1,.]+y[i+2,.]+y[i+3,.])/4;
        m2=(y[i+1,.]+y[i+2,.]+y[i+3,.]+y[i+4,.])/4;
        m[i,.]=(m1+m2)/2;
        i=i+1;
    endo;
    retp(m);
endp;
```

グラフ表示



繰り返すが、この場合のデータはメモリ消費の理由から、最後の５０個だけを処理している。これまでのグラフのおおよそ後半部分と考えてもらいたい。なお、cma の procedure という四半期データを対象にした単純な中央化移動平均は、もうすでに季節調整を行なったデータに行なってはならないことは言うまでもない。ただし、季節調整済みであろうがなかろうが、グラフの上半分のサイクルのところのグラフは Baxter-King フィルターのケースもこの Christiano-Fitzgerald フィルターも分離することは可能であり、おおよそではあるがそれぞれ似た波形になる。これは Band Pass 系フィルターの場合には設定した下限と上限の期の外側はサイクルとして認識しないためである。