Relatório Final do PIBIC 2006/2007

Estimação de Modelos em Espaço de Estado Não-Lineares / Não-Gaussianos via Simulação de Monte Carlo Seqüencial

Aluno: Guilherme Fernandes Sanches

Orientador: Cristiano Augusto Coelho Fernandes

Pontifícia Universidade Católica do Rio de Janeiro
Departamento de Engenharia Elétrica
Sistemas de Apoio à Decisão

Agosto de 2007

Índice:

Introdução página 2

Parte 1 - Modelos Lineares Gaussianos página 2

Parte 2 – Modelo de Tendência Linear página 15

Parte 3 –Modelos em espaço de estado não-lineares página 27

Introdução

O presente trabalho trata da estimação de modelos em espaço de estado, com especial destaque aos não-lineares / não-Gaussianos. Os modelos em espaço de estado são conhecidos na literatura de séries temporais por tratarem as observações como função de componentes não-observáveis (estado), como tendência, sazonalidade e volatilidade. Tal assunto é de extrema relevância para estudantes e pesquisadores das áreas de estatística, econometria e qualquer outro ramo de pesquisa interessado na análise e previsão de séries temporais.

Nosso objetivo central foi estudar uma metodologia de estimação de modelos em espaço de estado que se aplique a modelos não-lineares / não-Gaussianos. Para tais modelos, não é possível utilizar o Filtro de Kalman, método de estimação largamente utilizado para modelos em espaço de estado lineares Gaussianos. Para modelos não-lineares Gaussianos, como uma primeira aproximação, pode-se aplicar o Filtro de Kalman Estendido (FKE). No entanto, verifica-se, entre outras deficiências, que esse método de estimação produz estimadores viesados para o estado. Com o objetivo de sanar estes problemas do FKE, estudamos e implementamos uma metodologia baseada em simulação Monte Carlo, denominada de "Amostragem por Importância", a qual se aplica a modelos não-lineares / não-Gaussianos, produzindo estimadores assintoticamente não-viesados e consistentes. O Filtro de "Amostragem por Importância" também é chamado de Filtro IS, onde IS significa "Importance Sampling".

Parte 1 - Modelos Lineares Gaussianos

Inicialmente, foi implementada, na linguagem de programação *Ox*, uma rotina de estimação de modelos lineares Gaussianos através da metodologia do Filtro de Kalman. Foi gerada, através de simulação, uma série temporal como realização de um processo estocástico denominado de "modelo de nível local". Os resultados da estimação do modelo a partir da série gerada com o mesmo modelo foram satisfatórios. A maximização numérica da função de verossimilhança respondeu bem a diferentes valores iniciais e a diferentes parâmetros populacionais utilizados na simulação do processo.

Segue abaixo a explicação passo-a-passo do que foi feito :

Modelo de nível local :

$y_t = \alpha_t + \varepsilon_t \qquad \varepsilon_t \sim N(0, \sigma_\varepsilon^2)$

$\alpha_{t+1} = \alpha_t + \eta_t \qquad \eta_t \sim N(0, \sigma_\eta^2)$

onde $t = 0, \ldots, n$

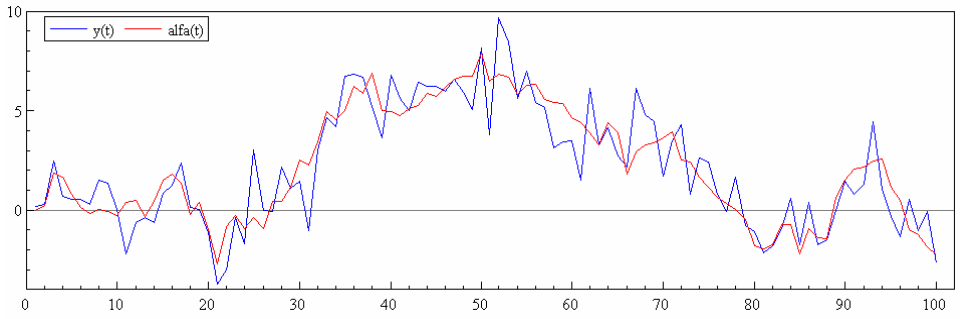
Valores utilizados :

$n = 100$
$\sigma_\varepsilon^2 = 1.5$
$\sigma_\eta^2 = 0.9$
$\alpha_0 = 0.0$

Algoritmo em linguagem Ox para a construção (simulação) do modelo de nível local :

```
Var_Epsilon = 1.5;
Var_Eta = 0.9;

Eta = sqrt(Var_Eta) * rann(n,1);
 // rann(n,1) retorna uma Matriz aleatória Normal(0,1)  n x 1
Epsilon = sqrt(Var_Epsilon) * rann(n,1);

alfa[0][0] = 0.0;

for (i = 0; i<n; i++)
{
        y[i][0] = alfa[i][0] + Epsilon[i][0];
        if (i != n-1)
                alfa[i+1][0] = alfa[i][0] + Eta[i][0];
 }
```

A Figura abaixo apresenta o gráfico de y em função do tempo e de α em função do tempo :



Filtro de Kalman

O Filtro de Kalman é um algoritmo adaptativo cujo objetivo é estimar, seqüencialmente, o estado não-observado de um modelo em espaço de estado. Nos modelos estruturais, que são casos particulares dos modelos em espaço de estado, o vetor de estado corresponde às componentes não-observáveis que descrevem a variação de uma série temporal, como, por exemplo, tendência, sazonalidade e ciclo. A estimativa é feita de forma recursiva à medida que novas observações são reveladas. Esta característica traz eficiência computacional ao método.

Recursões do Filtro de Kalman :

$a_{t+1} = a_t + K_t * v_t$
$P_{t+1} = P_t * (1 - K_t) + \sigma_\eta^2$

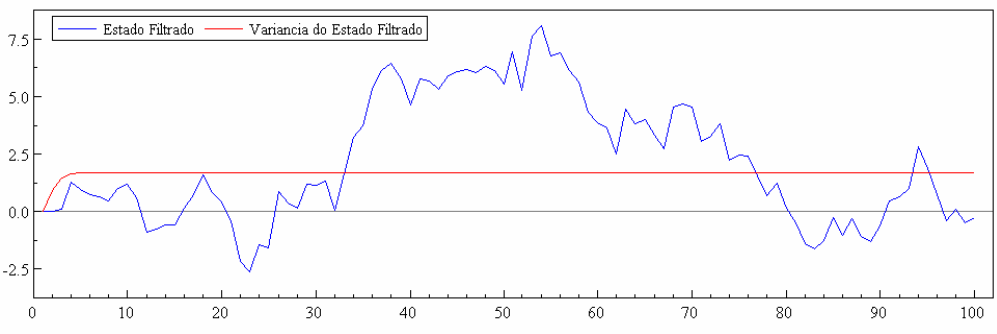
onde :

$v_t = y_t - a_t$
$F_t = P_t + \sigma_\varepsilon^2$
$K_t = P_t / F_t$
$a_0 = 0.0$
$P_0 = 0.0$

Algoritmo em Ox para o cálculo das recursões do Filtro de Kalman :

```
a[0][0] = 0.0;
P[0][0] = 10^6;              // Inicialização Difusa

for (i = 0; i<n; i++)
{
        v[i][0] = y[i][0] - a[i][0];
        F[i][0] = P[i][0] + Var_Epsilon;
        K[i][0] = P[i][0] / F[i][0];
        if (i != n-1)
        {
                a[i+1][0] = a[i][0] + (K[i][0] * v[i][0]);
                P[i+1][0] = P[i][0] * (1 - K[i][0]) + Var_Eta;
        }
}
```

A Figura abaixo apresenta o gráfico do estado filtrado e de sua variância ao longo do tempo :



Suavização :

Recursões da suavização do estado :

$\alpha_t = a_t + P_t * r_{t-1}$
$V_t = P_t - P_t^2 * N_{t-1}$

onde :

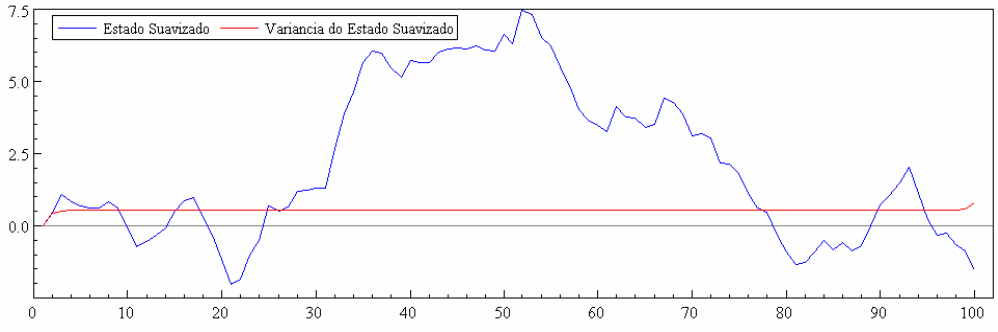
$L_t = 1 - K_t$
$r_{t-1} = (v_t / F_t) + L_t * r_t$
$N_{t-1} = (1 / F_t) + L_t^2 * N_t$

α_t ~ estado suavizado
V_t ~ variância do estado suavizado

Algoritmo em Ox para o cálculo das recursões da suavização do estado :

```
smooth[0][0] = 0.0;
Var_smooth[0][0] = 10^6;        // Inicialização difusa
r[n-1][0] = 0.0;
N[n-1][0] = 0.0;

for (i = n-1; i>0; i--)
{
        r[i-1][0] = (1/F[i][0]) * v[i][0] + L[i][0] * r[i][0];
        smooth[i][0] = a[i][0] + P[i][0] * r[i-1][0];

        N[i-1][0] = (1/F[i][0]) + (L[i][0]^2) * N[i][0];
        Var_smooth[i][0] = P[i][0] - (P[i][0]^2) * N[i-1][0];
}
```

A Figura abaixo apresenta o gráfico do estado suavizado e de sua variância ao longo do tempo :



Queremos, agora, estimar os parâmetros do modelo de nível local. Para isso, usamos novamente o pacote OxMetrics. Fizemos uma rotina que estima os parâmetros do modelo usando o método da máxima verossimilhança. Usamos uma função de maximização do próprio OxMetrics baseada no método de Quasi-Newton.

Nomeamos a função de "flog_verossimilhanca", por se tratar de uma função que calcula o resultado da função log da máxima-verossimilhança. "vp" é o vetor de parâmetros a ser estimado. No caso, vp[0] é a variância de 'ε' e vp[1] é a variância de 'η'. adFunc é o valor da função a ser retornado. avScore é uma dummy onde zero indica maximização com derivadas numéricas e 1 indica maximização com derivadas analíticas. amHessian é sempre zero para o caso da maximização por Quasi-Newton, já que tal método não utiliza matriz Hessiana.

As variáveis do Filtro de Kalman foram definidas globalmente, de forma que qualquer função pode modificar seus valores. Por isso, elas não entraram como parâmetros da função. Através do uso de um pouco de álgebra, chega-se à equação que define o log da verossimilhança como função das variáveis de saída do Filtro de Kalman :

$$\text{Log(L)} = -(^{N}/_{2}) * \log(2\pi) - (^{1}/_{2}) * \Sigma \left(\log(Ft) + v_t^2/F_t \right)$$

```
flog_verossimilhanca (const vp, const adFunc, const avScore, const amHessian)
{
decl i;

a[0][0] = 0.0;              // Valores iniciais
P[0][0] = 10^6;

for (i = 0; i<n; i++)
        {
                v[i][0] = y[i][0] - a[i][0];
                F[i][0] = P[i][0] + vp[0];
                //vp[0] = variância (a ser) estimada de Epsilon
                K[i][0] = P[i][0] / F[i][0];
                if (i != n-1)
                {
                        a[i+1][0] = a[i][0] + (K[i][0] * v[i][0]);
                        P[i+1][0] = P[i][0]*(1-K[i][0]) + vp[1];
                        //vp[1] = variância (a ser) estimada de Eta
                }
        }

//Função de Log_Verossimilhança :

        adFunc[0] = (-1)*(n/2)*log(2*pi);

        for (i = 0; i<n; i++)
        adFunc[0] = adFunc[0] - (1/2)*( log(F[i][0]) + ( (v[i][0]*v[i][0]) / F[i][0] ) );

        adFunc[0] = adFunc[0]/10000;
        // Dividir por 10.000 melhora a convergência

return 1;
}
```

No exemplo mostrado abaixo, usamos $\sigma^2_\varepsilon = 100$ e $\sigma^2_\eta = 10$ como valores populacionais para gerar as séries de y e alfa (estado). Em seguida, usamos as séries geradas para estimar os parâmetros. Tivemos bons resultados usando valores iniciais de 1.0 para ambos os parâmetros. A função 'MaxBFGS' utiliza o método de Quasi-Newton para maximizar a função 'flog_verossimilhanca'.

```
// Valor inicial para o estado:

alfa[0][0] = 0.0;

// Valores populacionais para Var_Epsilon e Var_Eta :

Var_Epsilon = 100.0;
Var_Eta = 10.0;

Epsilon = sqrt(Var_Epsilon) * rann(n,1);
Eta = sqrt(Var_Eta) * rann(n,1);

for (i = 0; i<n; i++)
{
        y[i][0] = alfa[i][0] + Epsilon[i][0];
        if (i != n-1)
        {
                alfa[i+1][0] = alfa[i][0] + Eta[i][0];
        }
}

vp = zeros(2,1);
vp[0][0] = 1.0;
vp[1][0] = 1.0;

MaxControl(1000, 1, 1);

ir = MaxBFGS(flog_verossimilhanca, &vp, &dfunc, 0, 1);

print("\n", MaxConvergenceMsg(ir), " using numerical derivatives", "\nFunction value = ", dfunc, ";
parametros: ", vp);

}
```

O output do programa é mostrado abaixo :

```
Ox version 4.02 (Windows) (C) J.A. Doornik, 1994-2006
This version may be used for academic research and teaching only

Strong convergence using numerical derivatives
Function value = -0.0385342; parametros:
       81.571
       12.009
```

A expressão “Strong convergence” indica que o processo de maximização foi bem-sucedido. Os valores encontrados foram bastante próximos dos populacionais.

Valor populacional	Valor Estimado (n = 100)
$\sigma^2_\varepsilon = 100.00$	$\sigma^2_\varepsilon = 81.571$
$\sigma^2_\eta = 10.00$	$\sigma^2_\eta = 12.009$

Agora, iremos repetir o procedimento de estimação para amostras maiores. Eis o output do programa com amostra de 200 observações :

```
Ox version 4.02 (Windows) (C) J.A. Doornik, 1994-2006
This version may be used for academic research and teaching only

Strong convergence using numerical derivatives
Function value = -0.0783312; parametros:
       93.020
       16.716
```

Novamente, a expressão “Strong Convergence” sugere um processo de maximização sem complicações. Os valores dos parâmetros estimados aumentaram. A variância estimada de ε se aproximou um pouco mais do valor populacional, mas a variância estimada de η se distanciou de seu valor populacional.

Valor populacional	Valor Estimado (n = 200)
$\sigma^2_\varepsilon = 100.00$	$\sigma^2_\varepsilon = 93.020$
$\sigma^2_\eta = 10.00$	$\sigma^2_\eta = 16.716$

Por fim, veremos o mesmo processo com uma amostra de 500 observações :

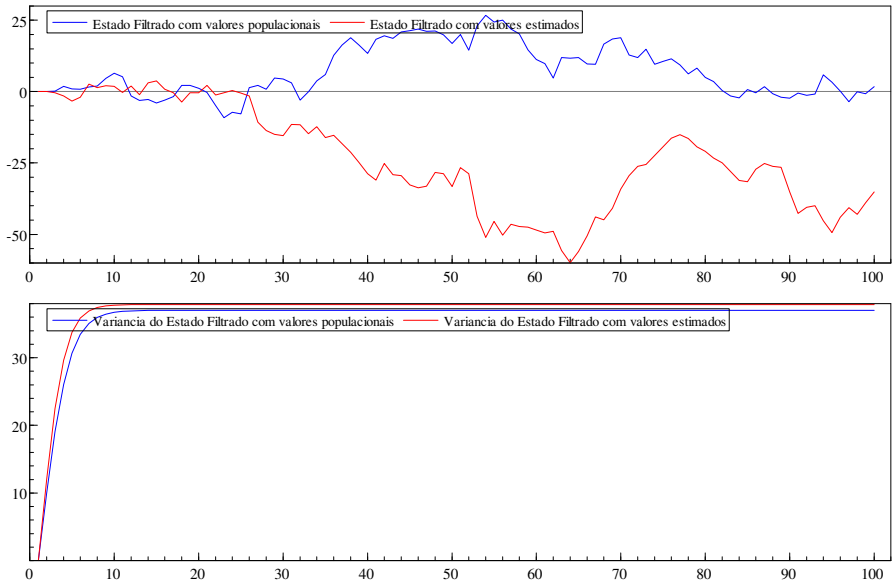
```
Ox version 4.02 (Windows) (C) J.A. Doornik, 1994-2006
This version may be used for academic research and teaching only

Strong convergence using numerical derivatives
Function value = -0.194186; parametros:
       91.560
       14.551
```

Dessa vez, o aumento do número de observações fez reduzir o valor dos parâmetros. A variância estimada de ε ficou mais distante de seu valor populacional, enquanto a variância estimada de η ficou mais próxima de seu valor populacional.

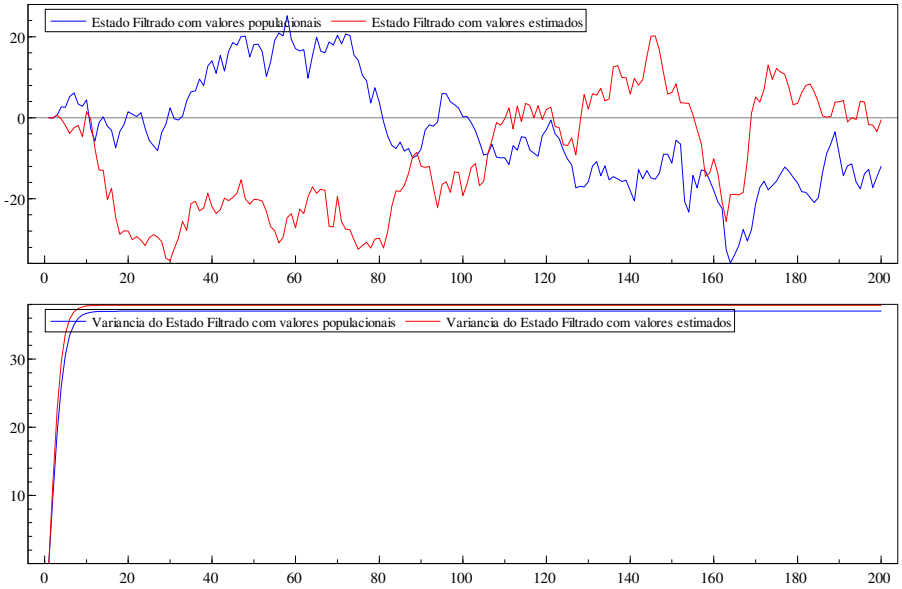
Valor populacional	Valor Estimado (n = 500)
$\sigma^2_\varepsilon = 100.00$	$\sigma^2_\varepsilon = 91.560$
$\sigma^2_\eta = 10.00$	$\sigma^2_\eta = 14.551$

Para visualizar a diferença implicada no modelo pelos parâmetros populacionais e estimados, serão ilustrados, a seguir, gráficos comparando o estado filtrado com os valores populacionais e estimados.

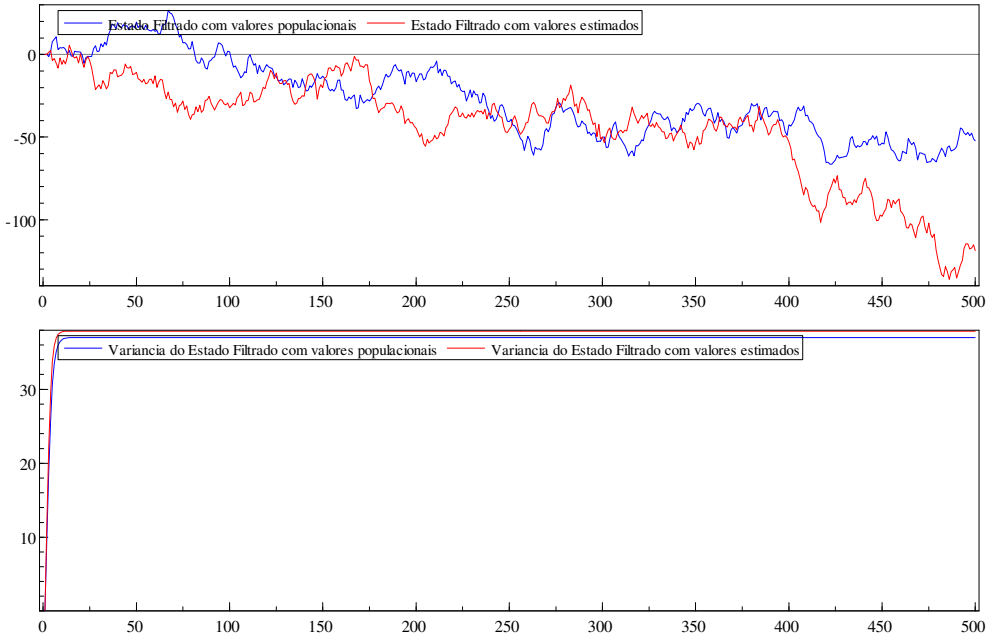
(n = 100)



(n = 200)

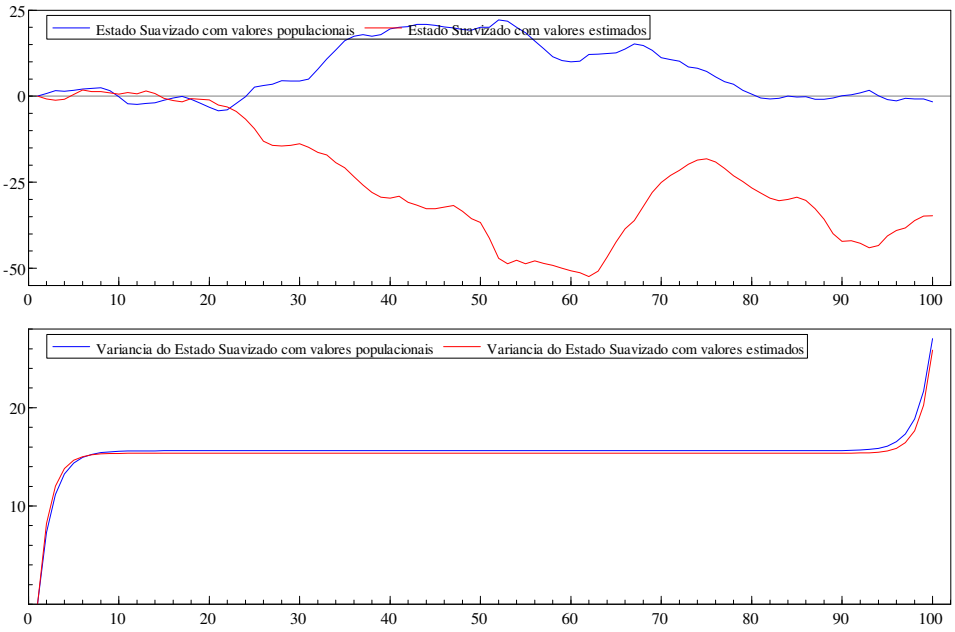

Estado Filtrado com valores populacionais
Estado Filtrado com valores estimados
Variancia do Estado Filtrado com valores populacionais
Variancia do Estado Filtrado com valores estimados

(n = 500)

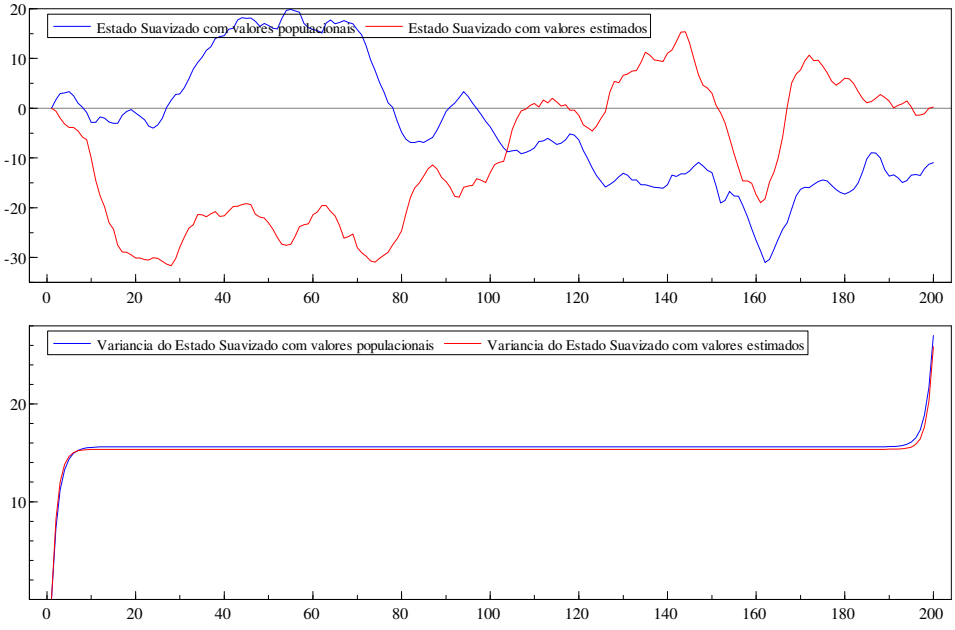

Estado Filtrado com valores populacionais
Estado Filtrado com valores estimados
Variancia do Estado Filtrado com valores populacionais
Variancia do Estado Filtrado com valores estimados

Agora, serão mostrados gráficos dos estados suavizados com os valores populacionais e estimados :

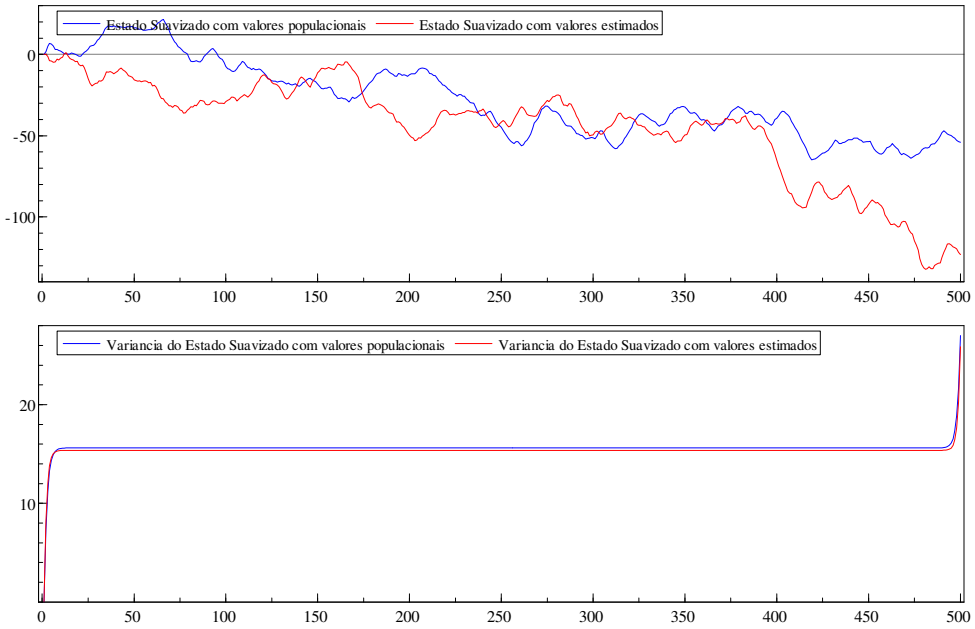
(n=100)



(n=200)



(n=500)


Estado Suavizado com valores populacionais
Estado Suavizado com valores estimados
0
-50
-100
0
50
100
150
200
250
300
350
400
450
500
Variancia do Estado Suavizado com valores populacionais
Variancia do Estado Suavizado com valores estimados
20
10
0
50
100
150
200
250
300
350
400
450
500

Parte 2 – Modelo de Tendência Linear

A seguir, estudaremos o modelo de tendência linear, em que o estado é representado por um vetor de duas posições.

Modelo linear Gaussiano geral:

$$y_t = Z_t * \alpha_t + \varepsilon_t \quad , \varepsilon_t \sim N(0, H_t)$$
$$\alpha_{t+1} = T_t * \alpha_t + R_t * \eta_t \quad , \eta_t \sim N(0, Q_t)$$

Modelo de tendência linear:

$$y_t = \mu_t + \varepsilon_t \quad , \varepsilon_t \sim N(0, \sigma_\varepsilon^2)$$
$$\mu_{t+1} = \mu_t + \beta_t + \xi_t \quad , \xi_t \sim N(0, \sigma_\xi^2)$$
$$\beta_{t+1} = \beta_t + \zeta_t \quad , \zeta_t \sim N(0, \sigma_\zeta^2)$$

Em notação de Espaço de Estado, temos:

$$y_t = \begin{pmatrix} 1 & 0 \end{pmatrix} * \begin{pmatrix} \mu_t \\ \beta_t \end{pmatrix} + \varepsilon_t$$

$$\begin{pmatrix} \mu_{t+1} \\ \beta_{t+1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} * \begin{pmatrix} \mu_t \\ \beta_t \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} * \begin{pmatrix} \xi_t \\ \zeta_t \end{pmatrix}, \quad H_t = \begin{pmatrix} \sigma_\varepsilon^2 \end{pmatrix} \qquad Q_t = \begin{pmatrix} \sigma_\xi^2 & 0 \\ 0 & \sigma_\zeta^2 \end{pmatrix}$$

Valores iniciais utilizados:

$\sigma_\varepsilon^2 = 1.5$
$\sigma_\xi^2 = 0.9$
$\sigma_\zeta^2 = 2.0$

$$\begin{pmatrix} \mu_0 \\ \beta_0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Algoritmo em Ox que preenche y(t) e alfa(t):

```
alfa[0] = zeros(2,1);
for (t = 0; t<n; t++)
        {
                y[t] = Z*alfa[t] + Epsilon[t];
                if (t != n-1)
                        alfa[t+1] = T*alfa[t] + R*Eta[t];
        }
```

Recursões do Filtro de Kalman:

$v_t = y_t - Z_t * a_t$ $\qquad$ $F_t = Z_t * P_t * Z_t' + H_t$

$K_t = T_t * P_t * Z_t' * F_t^{-1}$ $\qquad$ $L_t = T_t - K_t * Z_t$

$a_{t+1} = T_t * a_t + K_t * v_t$ $\qquad$ $P_{t+1} = T_t * P_t * L_t' + R_t * Q_t * R_t'$

Algoritmo em Ox que obtém as recursões do Filtro de Kalman :

```
for (t = 0; t<n; t++)
{
v[t] = y[t] - Z*a[t];
F[t] = Z*P[t]*(Z') + H;                    //H = matriz de covariâncias de Epsilon
K[t] = T*P[t]*(Z')*invert(F[t]);
L[t] = T - K[t]*Z;

if (t != n-1)
{
        a[t+1] = T*a[t] + K[t]*v[t];
        P[t+1] = T*P[t]*(L[t]') + R*Q*(R');        //Q = matriz de covariâncias de Eta
}

}
```

Recursões do processo de Suavização:

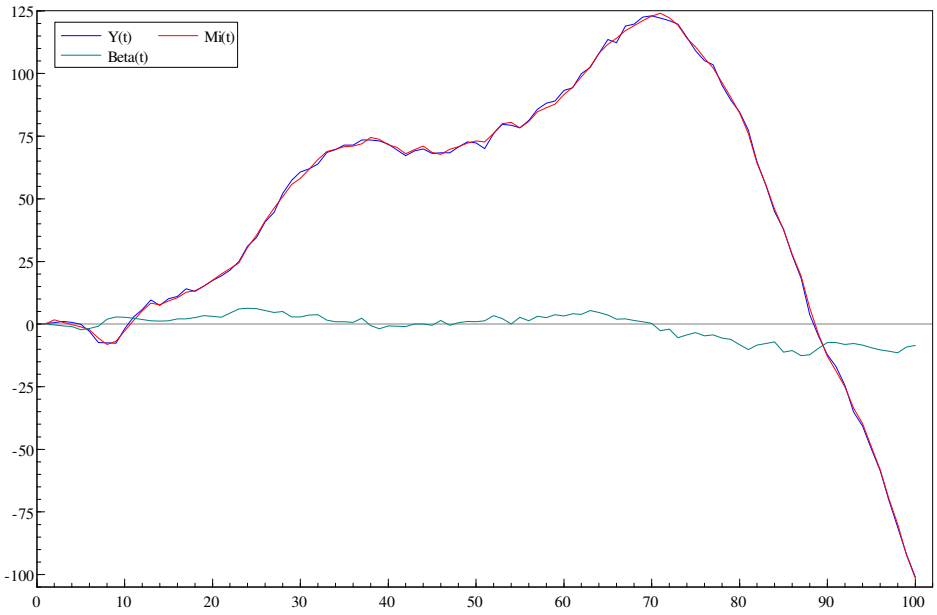
$r_{t-1} = Z_t' * F_t^{-1} * v_t + L_t' * r_t$ $\qquad$ $N_{t-1} = Z_t' * F_t^{-1} * Z_t + L_t' * N_t * L_t$

$\acute{\alpha}_t = a_t + P_t * r_{t-1}$ $\qquad$ $V_t = P_t - P_t * N_{t-1} * P_t$

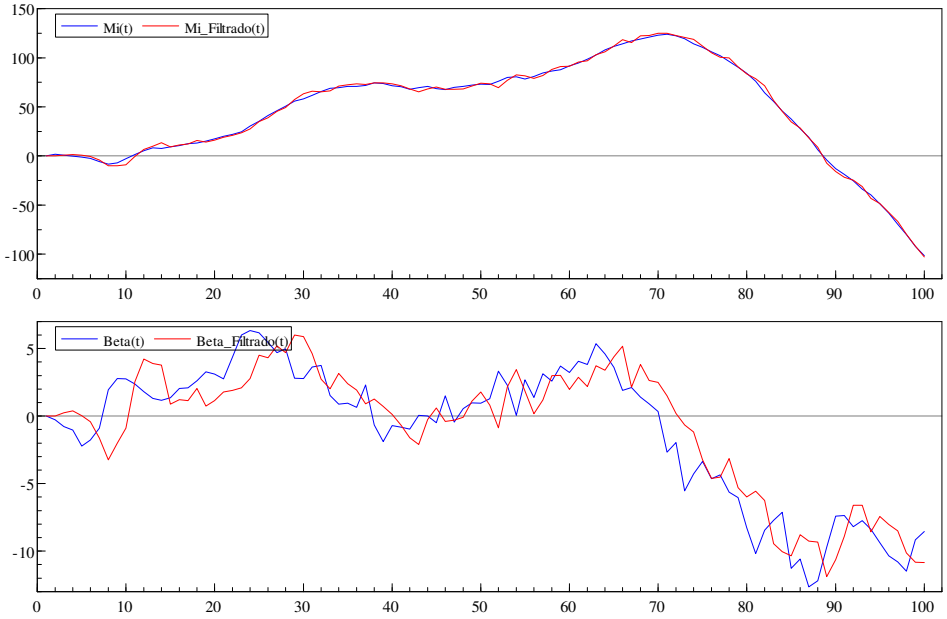
Algoritmo em Ox que obtém as recursões do processo de Suavização:

```
for (t=n-1; t>0; t--)
{
        r[t-1] = (Z')*invert(F[t])*v[t] + (L[t]')*r[t];
        smooth[t] = a[t] + P[t]*r[t-1];
        N[t-1] = (Z')*invert(F[t])*Z + (L[t]')*N[t]*L[t];
        Var_smooth[t] = P[t] - P[t]* N[t-1]*P[t];
}
```

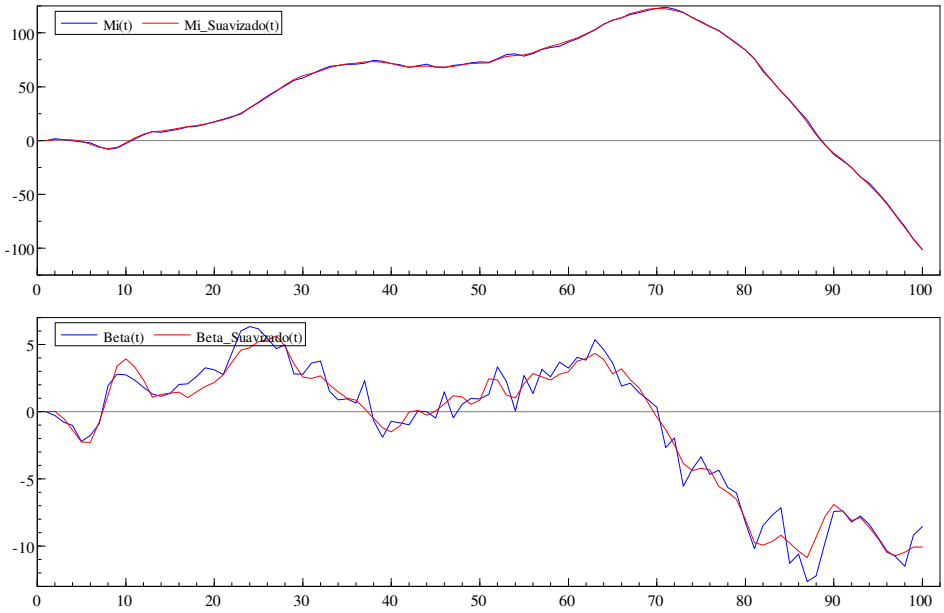
Abaixo, é apresentado o gráfico de Y(t) (observável) e do estado alfa(t), desmembrado nas componentes Mi(t) e Beta(t).



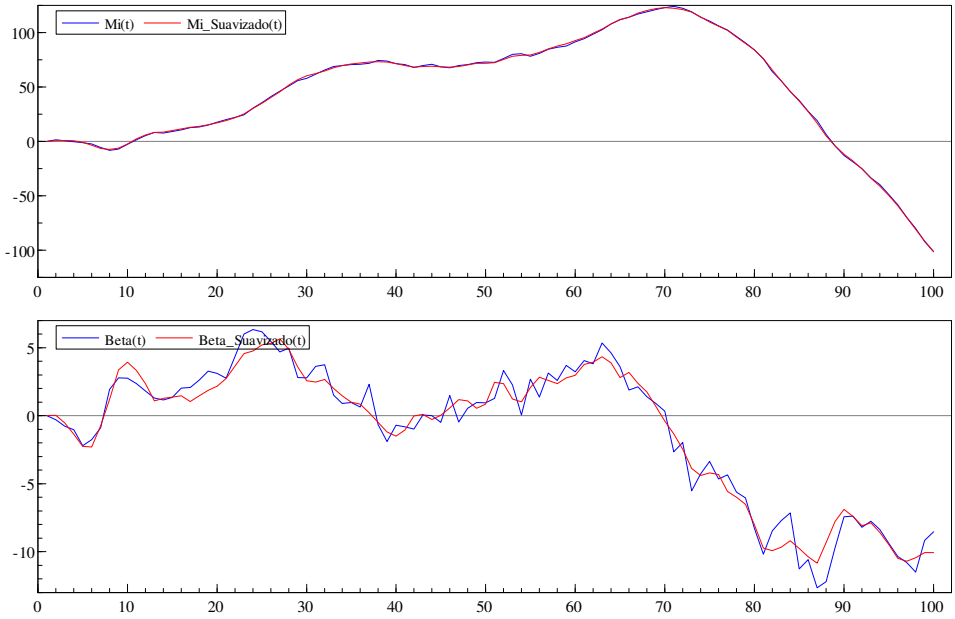
A seguir, são comparados os valores “reais” (simulados) do estado e seus respectivos valores filtrados.



Abaixo, são comparados os valores “reais” (simulados) do estado e seus respectivos valores suavizados.



Agora, são comparados os estados filtrado e suavizado:



Estimação dos parâmetros:

A estimação dos parâmetros do modelo de tendência linear foi realizada através do método de "máxima verossimilhança".

$L(y) = p(y_1 , y_2 , ... , y_n) = p(y_1)*p(y_2)* \Pi\, p(y_t \mid Y_{t-1})$
$\text{Log } L(y) = \Sigma \log p(y_t \mid Y_{t-1})$

Sabemos que $E(y_t \mid Y_{t-1}) = Z_t*a_t$. Fazendo $v_t = y_t - Z_t*a_t$, $F_t = Var(y_t \mid Y_{t-1})$ e substituindo $N(Z_t*a_t , F_t)$ em $p(y_t \mid Y_{t-1})$, obtivemos um método recursivo para o cálculo da máxima verossimilhança, demonstrado no quadro abaixo.

Algoritmo em Ox que define a função de log_verossimilhança, usando como parâmetros as variâncias a serem estimadas:

```
H = zeros(1,1);
H[0][0] = vp[0][0];
Q = zeros(2,2);
Q[0][0] = vp[1][0];
Q[1][1] = vp[2][0];

H2 = zeros(1,1);
H2[0][0] = exp(2*H[0][0]);              // Transformação exponencial
Q2 = zeros(2,2);
Q2[0][0] = exp(2*Q[0][0]);
Q2[1][1] = exp(2*Q[1][1]);

for (t = 0; t<n; t++)
{
        v[t] = y[t] - Z*a[t];
        F[t] = Z*P[t]*(Z') + H2;
        //H2 = matriz de covariâncias (estimadas) de Epsilon
        K[t] = T*P[t]*(Z')*invert(F[t]);
        L[t] = T - K[t]*Z;

        if (t != n-1)
        {
                a[t+1] = T*a[t] + K[t]*v[t];
                P[t+1] = T*P[t]*(L[t]') + R*Q2*(R');
                //Q2 = matriz de covariâncias (estimadas) de Eta
        }
}

//Função de Log_Verossimilhança :

adFunc[0] = (-1)*(n*p/2)*log(2*pi);

for (t = 2; t<n; t++)
        adFunc[0] = adFunc[0] - (1/2)*( log(determinant(F[t])) + (v[t]')*invert(F[t])*v[t] );

adFunc[0] = adFunc[0]/10000;                    // Dividir por 10.000 melhora a
convergência

return 1:
```

Observe que foi feita uma transformação exponencial nos parâmetros, que são os elementos das matrizes H e Q. Tal transformação se tornou necessária para assegurar valores positivos para as variâncias estimadas.

Assim como na estimação dos parâmetros do modelo de nível local, foi utilizado, para o modelo de tendência linear, o método de maximização "Quasi-Newton".

Para simular as séries "y" e "alfa", utilizamos os seguintes valores "populacionais":

Var_Epsilon = 1.5;
Var_Csi = 0.9;
Var_Zeta = 2.0;

Para realizar a maximização, utilizamos os seguintes valores iniciais :

vp[0][0] = 0.0001; (Representa a variância de Epsilon)
vp[1][0] = 0.0001; (Representa a variância de Csi)
vp[2][0] = 0.0001; (Representa a variância de Zeta)

Com uma amostra de 100 valores simulados de y, conseguimos o seguinte resultado no processo de estimação :

n = 100)

```
Strong convergence using numerical derivatives
Valor da funcao =
    -0.023465
; parametros:
      0.25286
       3.2174
       1.2695
```

Na ordem, os parâmetros estimados são : Variância de Epsilon, Variância de Csi e Variância de Zeta.

Veremos o que acontece com o valor dos parâmetros quando aumentamos o tamanho da amostra. Primeiro, para 200 :

n = 200)

```
Strong convergence using numerical derivatives
Valor da funcao =
    -0.046864
; parametros:
  1.2191e-005
       3.8903
       1.0635
```

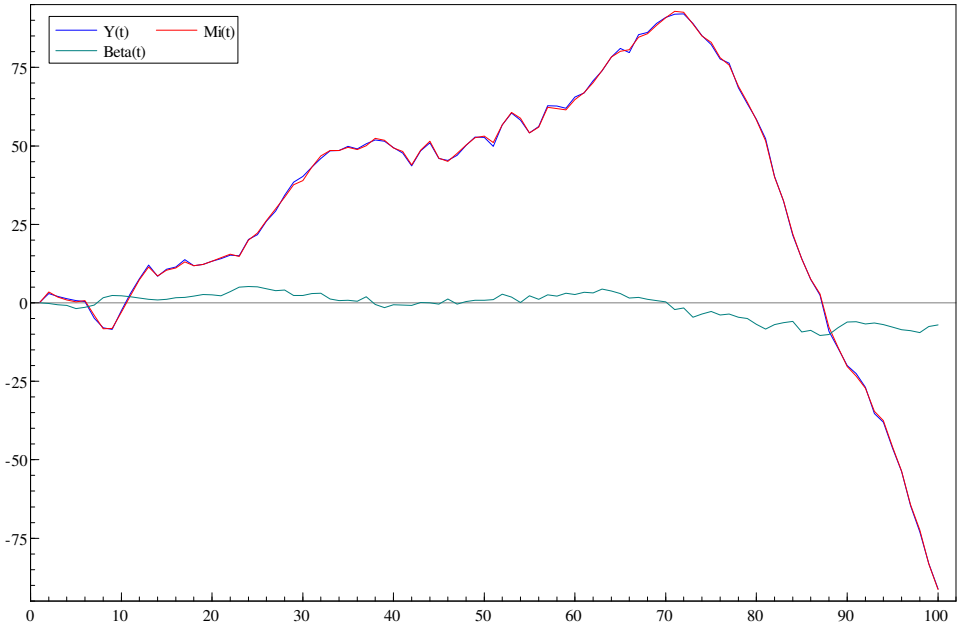
Por último, utilizaremos uma amostra de tamanho 500:

n = 500)

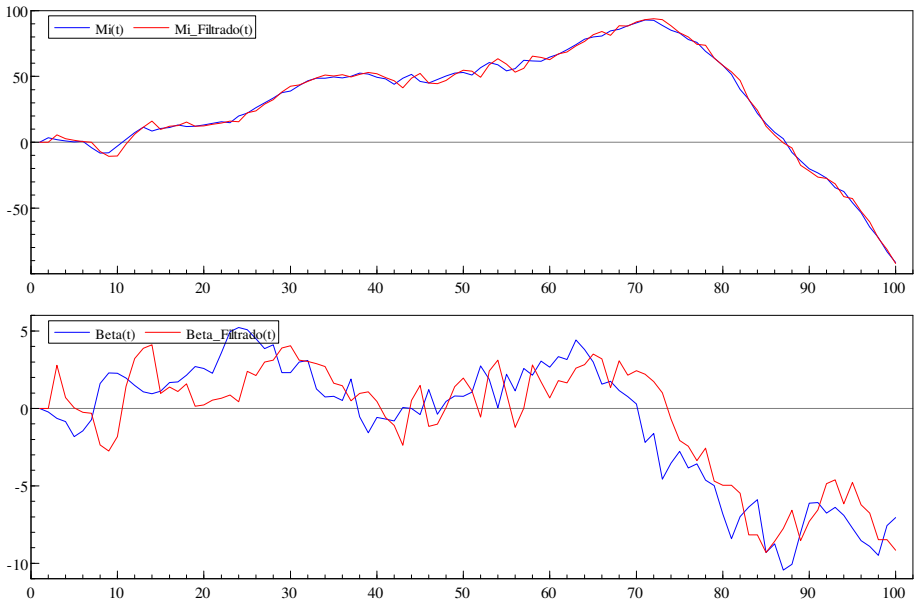
```
Strong convergence using numerical derivatives
Valor da funcao =
     -0.12263
; parametros:
      0.33019
       3.9153
       1.3611
```

Não é surpresa que os valores obtidos com a maior amostra sejam os mais próximos dos valores populacionais.

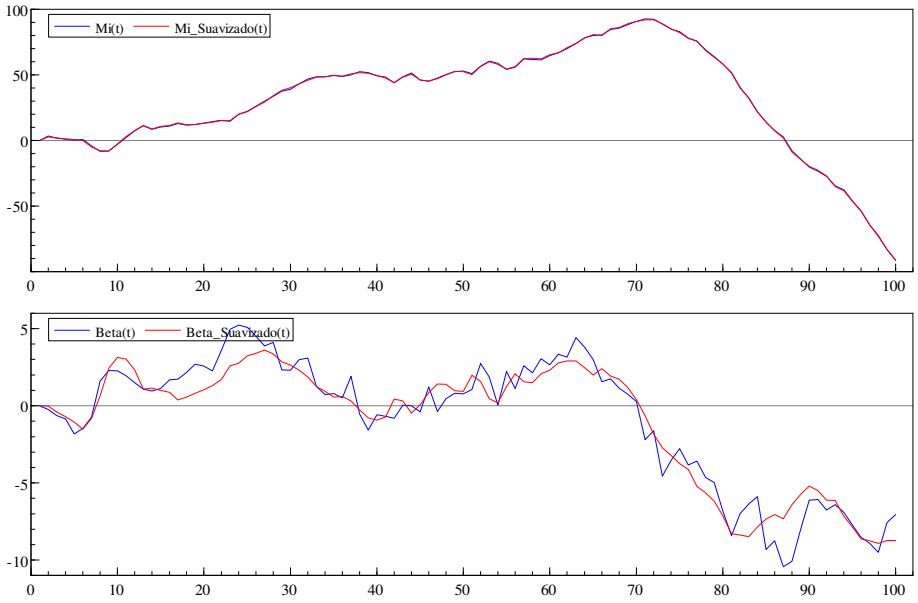
Em seguida, utilizaremos os valores estimados (com n = 500) para plotar gráficos da série “y” e do estado “alfa”.



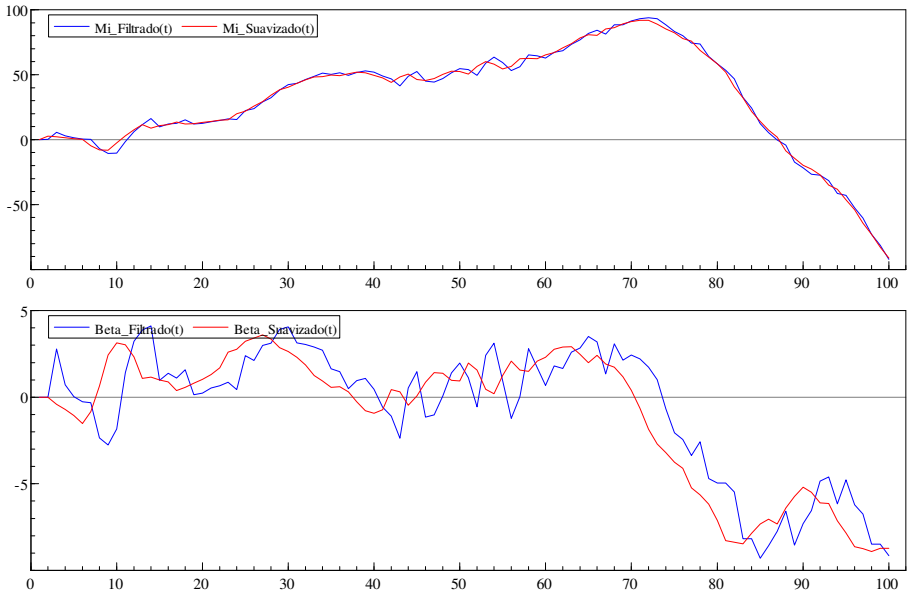
Agora, veremos uma comparação do estado simulado com o estado filtrado :



Estado simulado e estado suavizado :



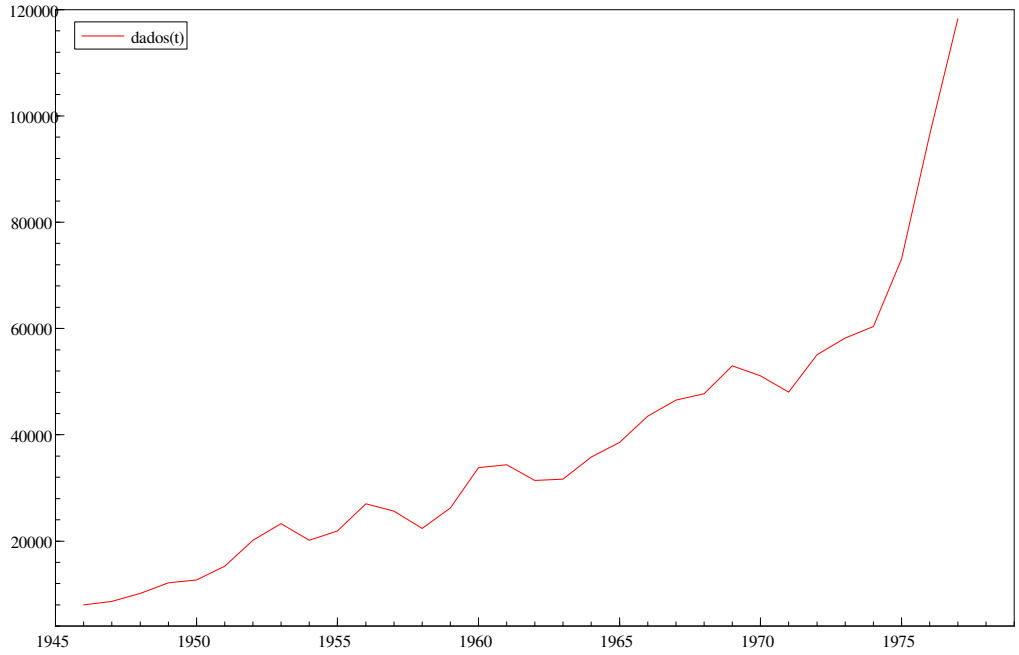
Por fim, uma comparação entre os estados filtrado e suavizado:



Aplicação do Modelo de Tendência Linear

O modelo de tendência linear apresentado neste trabalho apresenta diversas aplicações em econometria, sobretudo em relação a séries macroeconômicas. Analisaremos, a seguir, o modelo aplicado a uma **série brasileira de exportações*.

A amostra é composta por 32 observações. São dados anuais de exportações brasileiras contabilizadas em dólares, a partir do ano de 1947 até 2005. Segue o gráfico da série original :



* - Série obtida no site www.ipeadata.gov.br

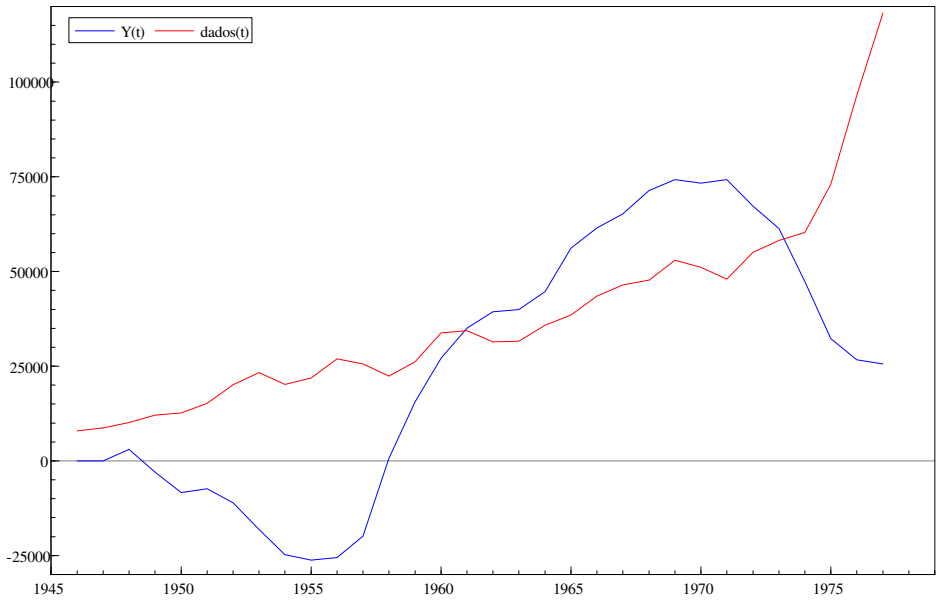
Utilizando tais dados, estimamos os parâmetros através do método de máxima verossimilhança. Eis o output do programa :

```
Strong convergence using numerical derivatives
Valor da funcao =
    -0.029926
; parametros:
       1792.2
       142.55
  2.3925e+007
```

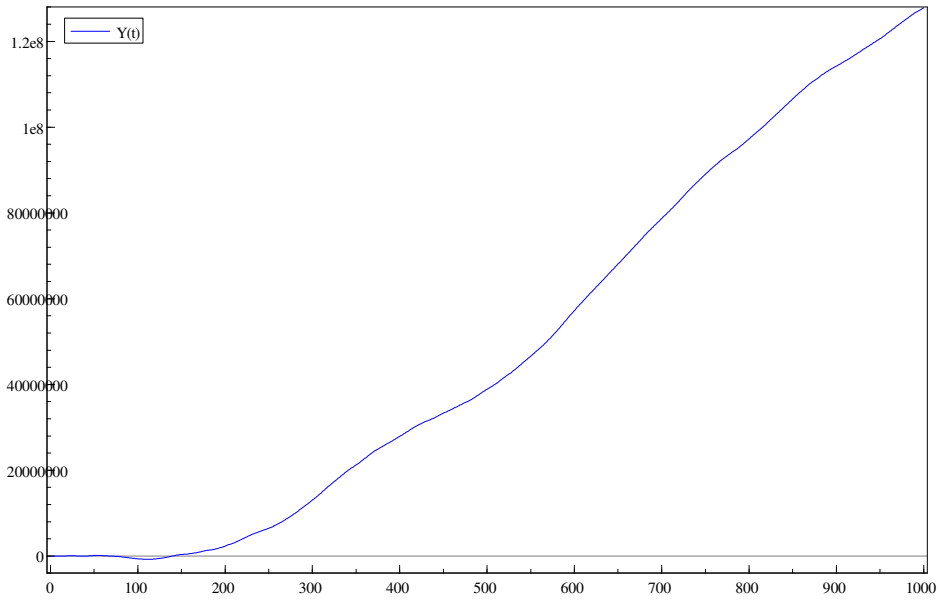
O processo de maximização da função de máxima verossimilhança foi um sucesso, e obtivemos, a partir de então, os valores estimados dos parâmetros :

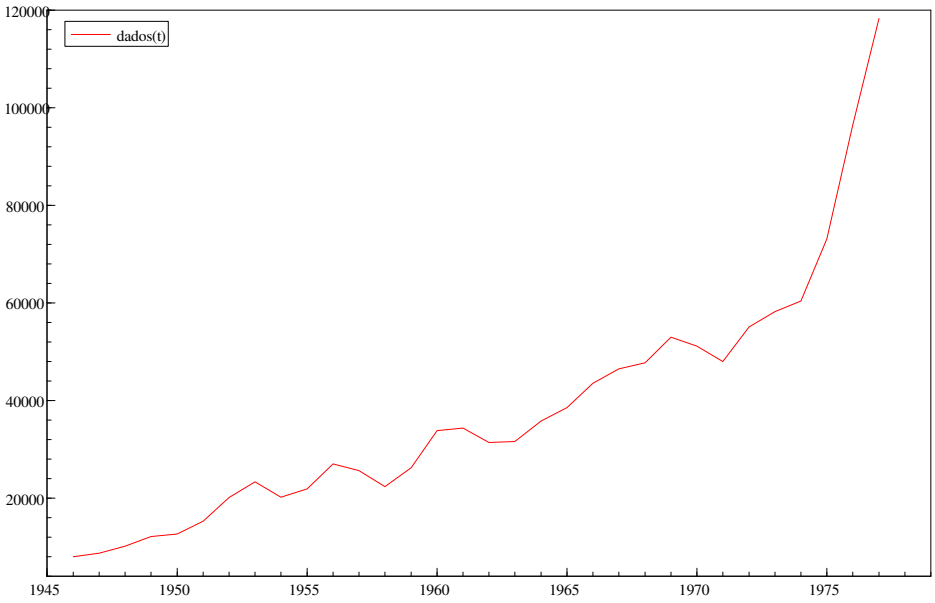
Var_Epsilon = 1792.2
Var_Csi = 142.55
Var_Zeta = 2.3925 * (10^7)

Utilizando tais valores, simulamos uma série Y e a comparamos com a série original. Isso é ilustrado no seguinte gráfico :



Ao aumentarmos o tamanho da amostra usada na simulação de Y, verificamos uma melhor aproximação dos "shapes" das séries real e simulada :





Parte 3 –Modelos em espaço de estado não-lineares

A última parte deste trabalho envolve a estimação de modelos em espaço de estado não-lineares através da metodologia conhecida por "Amostragem por Importância". Nesta abordagem, as densidades condicionais do estado, a densidade preditiva das observações e os respectivos momentos destas densidades são estimados por algoritmos que utilizam simulação de Monte Carlo para resolver numericamente as integrais associadas aos cálculos destas densidades e momentos. Uma particular dificuldade ao se estimar tais modelos é a ausência de uma função de máxima verossimilhança analítica, uma vez que a função de verossimilhança é construída a partir da densidade preditiva, e esta, como previamente apontada, não possui forma analítica. Desta forma, se faz necessário um otimizador numérico que forneça os parâmetros maximizadores da função de verossimilhança. Neste trabalho, utilizamos o método de maximização de "Quasi-Newton". A otimização numérica revelou-se estável para diferentes valores iniciais e para diferentes parâmetros populacionais utilizados na geração do estado.

A "Amostragem por Importância" pode ser entendida como uma técnica de redução de variância que pode ser utilizada na Simulação de Monte Carlo. A idéia principal por trás dela é que certos valores de entrada das variáveis aleatórias, em uma simulação, têm maior impacto sobre os parâmetros que estão sendo estimados do que outros. Quando esses valores "importantes" são enfatizados pela amostragem com maior freqüência, a variância do estimador pode ser reduzida.

A metodologia básica na "Amostragem por Importância" é escolher uma distribuição que destaque os valores "importantes". O uso de distribuições "viesadas" resultará em viesidade dos estimadores se elas forem aplicadas diretamente na simulação. No entanto, no Filtro IS, os valores de saída recebem pesos a fim de corrigir o uso da distribuição "viesada", assegurando que o novo estimador seja não-tendencioso. Cada peso é construído por uma razão entre a densidade de probabilidade verdadeira e a densidade de importância.

A questão fundamental na implementação do Filtro IS é a escolha de uma densidade de importância que destaque as regiões mais freqüentes da distribuição verdadeira. Escolher uma boa densidade de importância é a arte do Filtro IS. No caso deste trabalho, foi utilizada uma densidade de importância conhecida como "mistura de normais". Tal densidade se dá pela soma de duas densidades de probabilidade normais com médias e variâncias diferentes.

Utilizamos, como fim de aplicação, o seguinte modelo, com distúrbios normais e relação não-linear entre a série observável e o estado:

$$y_t = 1/(1+\exp(-\alpha_t) + \varepsilon_t \quad , \varepsilon_t \sim N(0, H_t)$$
$$\alpha_{t+1} = \alpha_t + \eta_t \quad , \eta_t \sim N(0, Q_t)$$

Algoritmo em Ox para construção do modelo:

```
estado_inicial = 0.0;
for (t = 0; t<n; t++)
{
        if (t==0)
                alfa[t] = estado_inicial + Eta[t];
        else
                alfa[t] = T[t-1] + R[t-1]*Eta[t];
        R[t] = unit(1);
        T[t] = alfa[t];
        Z[t] = 1.0/(1+exp(-alfa[t]));
        y[t] = Z[t] + Epsilon[t];
}
```

Para cada t pertencente a {0, 1, ..., n-1}, onde n-1 é o tamanho da amostra simulada da variável observável, executa-se os procedimentos de I a VI, descritos a seguir :

Obs: M é o número de iterações na Simulação de Monte Carlo.

I) Filtro de Kalman Estendido:

```
Z_chapeu[t] = pow(1+exp(-a_prev[t]), -2) * exp(-a_prev[t]);
v[t] = y[t] - (1.0/(1+exp(-a_prev[t])));
F[t] = Z_chapeu[t]*P_prev[t]*(Z_chapeu[t]') + H2;
K[t] = T_chapeu[t]*P_prev[t]*(Z_chapeu[t]')*(1/F[t]);
L[t] = T_chapeu[t] - K[t]*Z_chapeu[t];
a_atual[t] = a_prev[t] + K[t]*v[t];
P_atual[t] = P_prev[t] - P_prev[t]*(Z_chapeu[t]')*(1/F[t])*Z_chapeu[t]*P_prev[t];
y_prev[t] = 1.0/( 1+exp(-a_prev[t]) );
if (t != n-1)
{
T_chapeu[t+1] = unit(1);
R_chapeu[t+1] = unit(1);
a_prev[t+1] = a_atual[t];
P_prev[t+1] = T_chapeu[t]*P_prev[t]*(L[t]') + R_chapeu[t]*Q2*(R_chapeu[t]');
}
```

II) Geração de M partículas do estado, utilizando a densidade de importância:

```
p = 0.5;
c = 16.0;
for (i=0; i<M; i++)
{
D = ranbinomial(M, 1, 1, p);
if (D[i][0] == 1)
  alfa_IS[i][t] = a_prev[t] + sqrt(c*P_prev[t])*rann(1,1);
else
  alfa_IS[i][t] = a_atual[t] + sqrt(c*P_atual[t])*rann(1,1);
}
```

III) Construção dos pesos:

III.1) Pesos previstos

```
w_atual_inicial = 1.0;
mi_1 = a_prev[t];
sigma_2_1 = c*P_prev[t];
mi_2 = a_atual[t];
sigma_2_2 = c*P_atual[t];

for (i=0; i<M; i++)
{
normal_1[i][t] = (1.0/sqrt(2*pi*sigma_2_1)) * exp( (-pow(alfa_IS[i][t]-mi_1, 2)) /
(2*sigma_2_1) );
normal_2[i][t] = (1.0/sqrt(2*pi*sigma_2_2)) * exp( (-pow(alfa_IS[i][t]-mi_2, 2)) /
(2*sigma_2_2) );
dens_importancia[i][t] = (1/2)*normal_1[i][t] + (1/2)*normal_2[i][t];

w_prev[i][t] = 0.0;

for (j=0; j<M; j++)
{

if (t==0)
{
mi = estado_inicial;
sigma_2 = Q2[0][0];
dens_alfa_cond[j][t] = (1.0/sqrt(2*pi*sigma_2)) * exp( (-pow(alfa_IS[i][t]-mi, 2)) /
(2*sigma_2) );
w_prev[i][t] += (dens_alfa_cond[j][t]/dens_importancia[i][t]) * w_atual_inicial;
}

else
{
mi = alfa_IS[j][t-1];
sigma_2 = Q2[0][0];
dens_alfa_cond[j][t] = (1.0/sqrt(2*pi*sigma_2)) * exp( (-pow(alfa_IS[i][t]-mi, 2)) /
(2*sigma_2) );
w_prev[i][t] += (dens_alfa_cond[j][t]/dens_importancia[i][t]) * w_atual[j][t-1];
}

}

w_prev[i][t] /= M;

}
```

III.2) Pesos atualizados:

```
for (i=0; i<M; i++)
{
aux = 0.0;
for (j=0; j<M; j++)
{
mi = 1.0/(1+exp(-alfa_IS[j][t]));
sigma_2 = H2[0][0];
dens_y_cond[j][t] = (1.0/sqrt(2*pi*sigma_2)) * exp( (-pow(y[t]-mi, 2)) / (2*sigma_2) );
aux += dens_y_cond[j][t]*w_prev[j][t];
}
aux /= M;
w_atual[i][t] = ( dens_y_cond[i][t]*w_prev[i][t] ) / aux;
}
```

IV) Estimação do estado previsto e atualizado e de suas respectivas variâncias:

```
a_IS_prev[t] = 0.0;
a_IS_atual[t] = 0.0;
P_IS_prev[t] = 0.0;
P_IS_atual[t] = 0.0;

for (i=0; i<M; i++)
{
a_IS_prev[t] += alfa_IS[i][t]*w_prev[i][t];
a_IS_atual[t] += alfa_IS[i][t]*w_atual[i][t];
P_IS_prev[t] += (alfa_IS[i][t] - a_IS_prev[t]) * (alfa_IS[i][t] - a_IS_prev[t])' *
w_prev[i][t];
P_IS_atual[t] += (alfa_IS[i][t] - a_IS_atual[t]) * (alfa_IS[i][t] - a_IS_atual[t])' *
w_atual[i][t];
}
a_IS_prev[t] /= M;
a_IS_atual[t] /= M;
P_IS_prev[t] /= M;
P_IS_atual[t] /= M;
```

V) Obtenção das densidades atualizada e prevista:

```
for (i=0; i<M; i++)
{
densidade_atual[i][t] = w_atual[i][t]*dens_importancia[i][t];
densidade_prev[i][t] = w_prev[i][t]*dens_importancia[i][t];
}
```

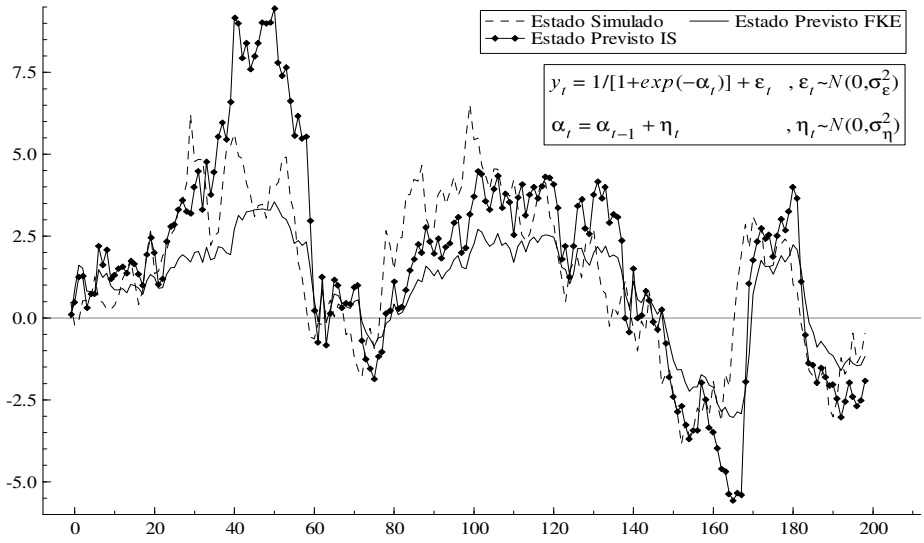
VI) Construção de y_IS_prev:

```
y_IS_prev[t] = 0.0;
for (i=0; i<M; i++)
  y_IS_prev[t] += dens_y_cond[i][t]*w_prev[i][t];
y_IS_prev[t] /= M;
```

VII) Estimação dos Parâmetros:

```
if (t>1)
  adFunc[0] += log(y_IS_prev[t]);
```

A figura abaixo mostra uma comparação entre o estado simulado a partir de um modelo não-linear pré-especificado, o estado previsto pelo Filtro de Kalman Estendido e o estado previsto pelo Filtro IS. Nas três simulações foi utilizada uma amostra de tamanho n=200 para a variável observável y e, no filtro IS, foram geradas 200 partículas do estado na Simulação de Monte Carlo.



A técnica de "Amostragem por Importância" permite a estimação de uma série de modelos em espaço de estado não-lineares / não-Gaussianos com os quais pesquisadores das áreas de econometria e estatística se deparam constantemente. Um exemplo interessante da aplicação do Filtro IS é a estimação de modelos de volatilidade estocástica, estudados nos campos da Economia Financeira e da Matemática Financeira. Tais modelos são, essencialmente, não-lineares e, em sua maioria, não-Gaussianos.

Referências

1 - Durbin, J. and Koopman, S. J. **Time Series Analysis by State Space Methods**. First Edition. Oxford Statistical Science Series.

2 - Tanizaki, Hisashi. **Nonlinear Filters**. Second Edition. Springer.

3 - Doornik, J. A. and Ooms, Marius. **Introduction to Ox**. Disponível em: http://www.doornik.com/ox Acesso em: 05 de julho de 2007.

4 – Srinivasan, Rajan. **Importance sampling - Applications in communications and detection.** Springer-Verlag.