22.67 - Señales Aleatorias

Instituto Tecnológico de Buenos Aires

Trabajo Práctico Número 3

Filtros de Kalman

Grupo 3

AUTORES:
Gonzalo Davidov
Facundo Farall
Federico Tondi
Nicolás Trozzo
Alan Vekselman

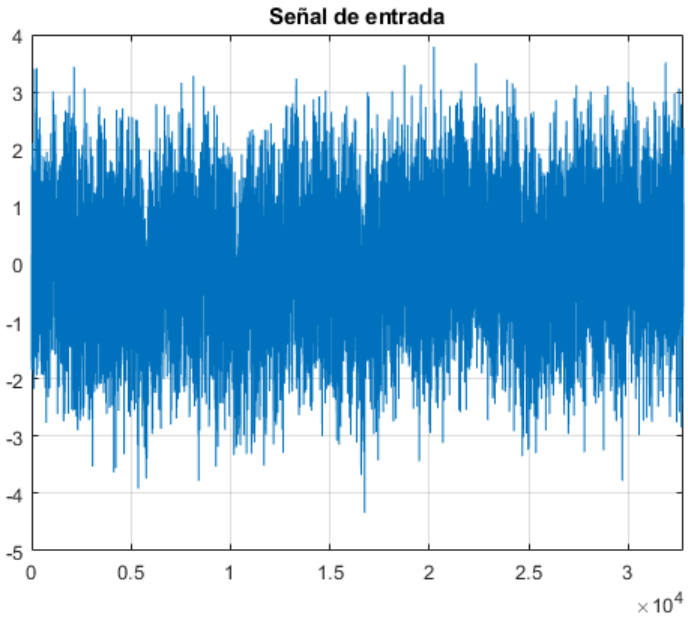
PROFESORES:
Gustavo Hirchoren
Leandro Luo

CIUDAD AUTÓNOMA DE BUENOS AIRES
Febrero 2021

Introducción

En el presente trabajo se busca aplicar un filtro de Kalman a una serie de datos, provistos por los docentes. Estos datos se modelizarán con modelos AR cuyos órdenes varían desde 1 a 9.

```
clear
data = load('h08g1.dat');
n = length(data);
pmax = 9;
figure()
plot(data);
grid on;
title('Señal de entrada');
xlim([0 n])
```

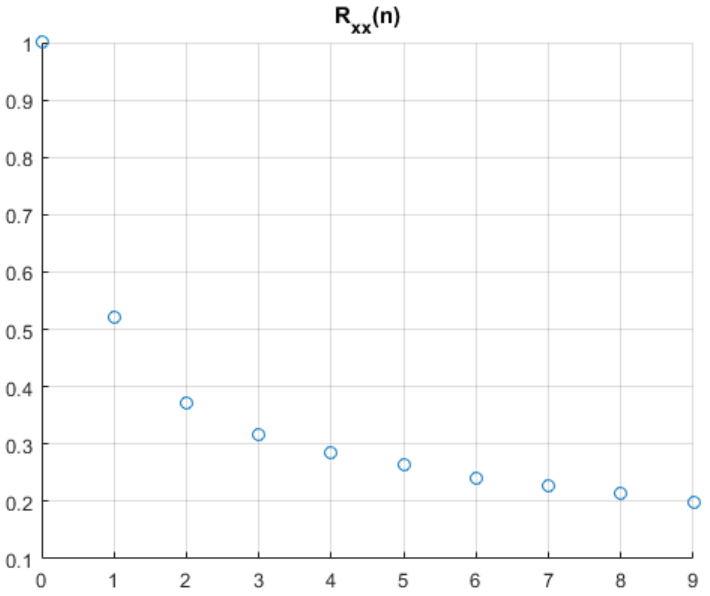


Estimación de Rxx(n)

Se calcula la media y la varianza de los datos, y luego se calcula la autocorrelación mediante el estimador no polarizado:

$$R_{np}(k) = \frac{1}{n-k} \cdot \sum_{i=1}^{n-k} X_i \cdot X_{i+k}$$

```
mu = mean(data);
variance = var(data);
Rxx = zeros(pmax + 1, 1);
for p = 1:(pmax + 1)
    k = p - 1;
    Rxx(p) = 1 / (n - k) * sum(data(1:n-k) .* data(p:n));
end
scatter(0:pmax, Rxx)
title('R_{xx}(n)')
grid
```



Cálculo de parámetros del modelo AR

Ahora se buscan, para cada **p=1,...,9**, los parámetros de un modelo AR(p) que ajuste a los datos provistos. Para ello se utiliza la **ecuación de Yule-Walker:**

$$\vec{\Phi}_p = R_p^{-1} \cdot \vec{r}_{xx}$$

```
% R: matriz que contiene las 9 matrices para calcular cada Phi
% R(p,1:p,1:p) es la matriz que permite calcular Phi_p
R = zeros(pmax, pmax, pmax);
% Phi: matriz que contiene los 9 vectores con los coeficientes de
%      correlacion parcial. Phi(p,1:p) contiene Phi_p
Phi = zeros(pmax);
for p=1:pmax
    % Armamos la matriz R
    for i=1:p
        for j=1:p
            R(p,i,j) = Rxx(abs(i-j)+1) / Rxx(1);
        end
    end
    % Armamos vector rxx con rxx(1) a rxx(p)
    fprintf('Yule-Walker para p = %i', p);
    rxx = Rxx(2:p+1) ./ Rxx(1);
    auxR = squeeze(R(p,1:p,1:p));   % squeeze para que pase de "1xpxp" a "pxp"
    fprintf('R%i = ', p);
    disp(auxR);
    Phi(p, 1:p) = auxR \ rxx;
    fprintf('Phi%i = ', p);
    disp(squeeze(Phi(p,1:p)));
    fprintf(' ')
end
```

```
Yule-Walker para p = 1
R1 =
```

```
     1
Phi1 =
    0.5206

Yule-Walker para p = 2
R2 =
    1.0000    0.5206
    0.5206    1.0000
Phi2 =
    0.4493    0.1371

Yule-Walker para p = 3
R3 =
    1.0000    0.5206    0.3710
    0.5206    1.0000    0.5206
    0.3710    0.5206    1.0000
Phi3 =
    0.4340    0.0871    0.1113

Yule-Walker para p = 4
R4 =
    1.0000    0.5206    0.3710    0.3177
    0.5206    1.0000    0.5206    0.3710
    0.3710    0.5206    1.0000    0.5206
    0.3177    0.3710    0.5206    1.0000
Phi4 =
    0.4249    0.0800    0.0761    0.0813

Yule-Walker para p = 5
R5 =
    1.0000    0.5206    0.3710    0.3177    0.2856
    0.5206    1.0000    0.5206    0.3710    0.3177
    0.3710    0.5206    1.0000    0.5206    0.3710
    0.3177    0.3710    0.5206    1.0000    0.5206
    0.2856    0.3177    0.3710    0.5206    1.0000
Phi5 =
    0.4195    0.0749    0.0707    0.0529    0.0669

Yule-Walker para p = 6
R6 =
    1.0000    0.5206    0.3710    0.3177    0.2856    0.2642
    0.5206    1.0000    0.5206    0.3710    0.3177    0.2856
    0.3710    0.5206    1.0000    0.5206    0.3710    0.3177
    0.3177    0.3710    0.5206    1.0000    0.5206    0.3710
    0.2856    0.3177    0.3710    0.5206    1.0000    0.5206
    0.2642    0.2856    0.3177    0.3710    0.5206    1.0000
Phi6 =
    0.4165    0.0725    0.0675    0.0495    0.0479    0.0452

Yule-Walker para p = 7
R7 =
    1.0000    0.5206    0.3710    0.3177    0.2856    0.2642    0.2407
    0.5206    1.0000    0.5206    0.3710    0.3177    0.2856    0.2642
    0.3710    0.5206    1.0000    0.5206    0.3710    0.3177    0.2856
    0.3177    0.3710    0.5206    1.0000    0.5206    0.3710    0.3177
    0.2856    0.3177    0.3710    0.5206    1.0000    0.5206    0.3710
    0.2642    0.2856    0.3177    0.3710    0.5206    1.0000    0.5206
    0.2407    0.2642    0.2856    0.3177    0.3710    0.5206    1.0000
Phi7 =
    0.4145    0.0704    0.0653    0.0465    0.0447    0.0268    0.0442

Yule-Walker para p = 8
R8 =
```

```
    1.0000    0.5206    0.3710    0.3177    0.2856    0.2642    0.2407    0.2265
    0.5206    1.0000    0.5206    0.3710    0.3177    0.2856    0.2642    0.2407
    0.3710    0.5206    1.0000    0.5206    0.3710    0.3177    0.2856    0.2642
    0.3177    0.3710    0.5206    1.0000    0.5206    0.3710    0.3177    0.2856
    0.2856    0.3177    0.3710    0.5206    1.0000    0.5206    0.3710    0.3177
    0.2642    0.2856    0.3177    0.3710    0.5206    1.0000    0.5206    0.3710
    0.2407    0.2642    0.2856    0.3177    0.3710    0.5206    1.0000    0.5206
    0.2265    0.2407    0.2642    0.2856    0.3177    0.3710    0.5206    1.0000
Phi8 =
    0.4129    0.0695    0.0638    0.0449    0.0424    0.0243    0.0295    0.0354

Yule-Walker para p = 9
R9 =
    1.0000    0.5206    0.3710    0.3177    0.2856    0.2642    0.2407    0.2265    0.2131
    0.5206    1.0000    0.5206    0.3710    0.3177    0.2856    0.2642    0.2407    0.2265
    0.3710    0.5206    1.0000    0.5206    0.3710    0.3177    0.2856    0.2642    0.2407
    0.3177    0.3710    0.5206    1.0000    0.5206    0.3710    0.3177    0.2856    0.2642
    0.2856    0.3177    0.3710    0.5206    1.0000    0.5206    0.3710    0.3177    0.2856
    0.2642    0.2856    0.3177    0.3710    0.5206    1.0000    0.5206    0.3710    0.3177
    0.2407    0.2642    0.2856    0.3177    0.3710    0.5206    1.0000    0.5206    0.3710
    0.2265    0.2407    0.2642    0.2856    0.3177    0.3710    0.5206    1.0000    0.5206
    0.2131    0.2265    0.2407    0.2642    0.2856    0.3177    0.3710    0.5206    1.0000
Phi9 =
    0.4120    0.0686    0.0631    0.0437    0.0412    0.0225    0.0276    0.0239    0.0276
```

No se observa que se anulen ninguno de los $\phi_{k,k}$, por lo cual no se puede afirmar que alguno de los modelos se haya ajustado correctamente.

Modelo en variables de estado y filtro de Kalman

Para cada modelo AR(p) se plantea el siguiente modelo de estados para el filtro de Kalman:

$$\mathbf{x}_{k+1} = \phi_p \cdot \mathbf{x}_k + \mathbf{w}_k$$

$$z_k = \mathbf{H}_k \cdot \mathbf{x}_k + v_k$$

Donde:

$$\mathbf{x}_k = \begin{pmatrix} x(k) \\ x(k-1) \\ \vdots \\ x(k-p+1) \end{pmatrix}$$

$$\phi_p = \begin{pmatrix} \phi_{p,1} & \phi_{p,2} & \cdots & \phi_{p,p-1} & \phi_{p,p} \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix} \epsilon \mathbb{R}^{pxp}$$

$$\mathbf{w}_k = \begin{pmatrix} w_k \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \epsilon \mathbb{R}^{px1}$$

$$\mathbf{H}_k = (1 \quad 0 \quad \dots \quad 0) \, \epsilon \, \mathbb{R}^{1xp}$$

v_k: ruido blanco gaussiano de varianza σ_v^2

```
vars_len = 10;
vars = logspace(-2,1,vars_len);
var_to_plot = vars_len-2;

nmax = n;
nplot = 100;

mse_input = zeros(pmax,1);
mse_output = zeros(pmax,vars_len);
improvements = [];


for j = 1:vars_len
    varV = vars(j);
    Z = data(1:nmax) + wgn(nmax, 1, 10*log10(varV));

    for p=1:pmax

        Phip = zeros(p);
        Phip(1,:) = Phi(p,1:p);
        Phip(2:end,:) = [ eye(p-1) zeros(p-1,1) ];

        Hk = zeros(1,p);
        Hk(1) = 1;

        varN = variance * (1 - dot(Phi(p,1:p), rxx(1:p)));

        Qk = zeros(p);
        Qk(1,1) = varN;
        Rk = varV;

        Xminus = [ mu ; zeros(p-1,1) ];
        % x's covariance matrix was used for Yule-Walker
        Pminus = squeeze(R(p,1:p,1:p));
        % Run Filter
        output = zeros(nmax,1);
        input = zeros(nmax,1);
        for k=1:nmax
```

Se obtiene la ganancia de Kalman:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1}$$

```
            Kk = Pminus*Hk'*inv(Hk*Pminus*Hk'+Rk);
```

Se actualizan las estimaciones con las mediciones Z(k). Ésta es la salida del filtro:

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^-)$$

```
            Xk = Xminus + Kk*(Z(k) - Hk*Xminus);
```

Se obtiene la covarianza del error para la nueva estimación:

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-$$

```
            Pk = (eye(p)-Kk*Hk)*Pminus;
```

Se proyecta hacia adelante. Ésta es la salida si se utiliza como predictor:

$$\hat{\mathbf{x}}_{k+1}^- = \phi_k\hat{\mathbf{x}}_k$$

$$\mathbf{P}_{k+1}^- = \phi_k\mathbf{P}_k\phi_k^T + \mathbf{Q}_k$$

```
            Xminus = Phip*Xk;
            Pminus = Phip*Pk*Phip' + Qk;
            input(k) = Z(k);
            output(k) = Xk(1);
        end
```

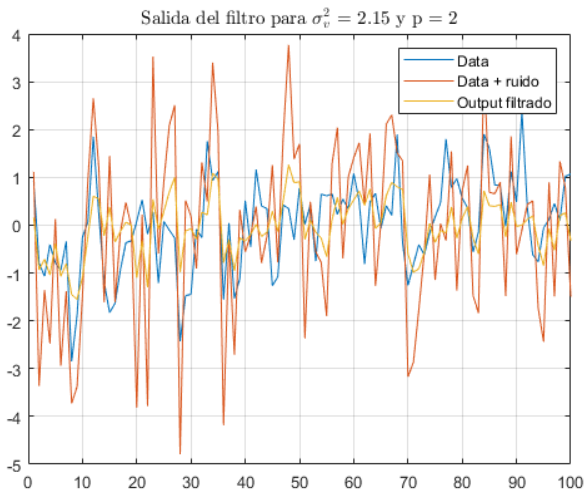
Se calcula el error cuadrático de la entrada con ruido, y la salida filtrada, ambos con respecto a los datos originales.

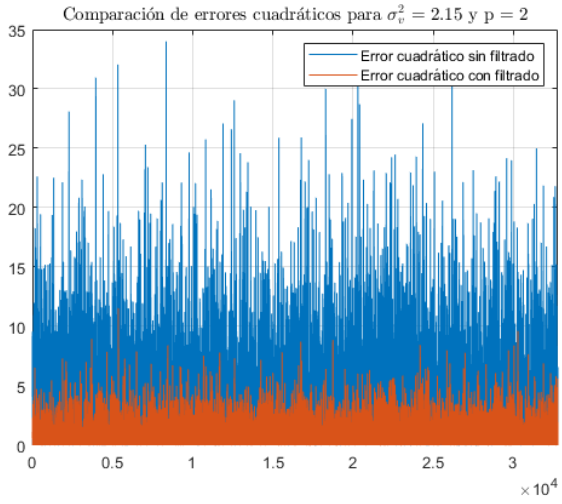
```
        err_input = (data(1:nmax) - input).^2;
        err_output = (data(1:nmax) - output).^2;
        mse_input(p) = mean(err_input);
        mse_output(p,j) = mean(err_output);
```

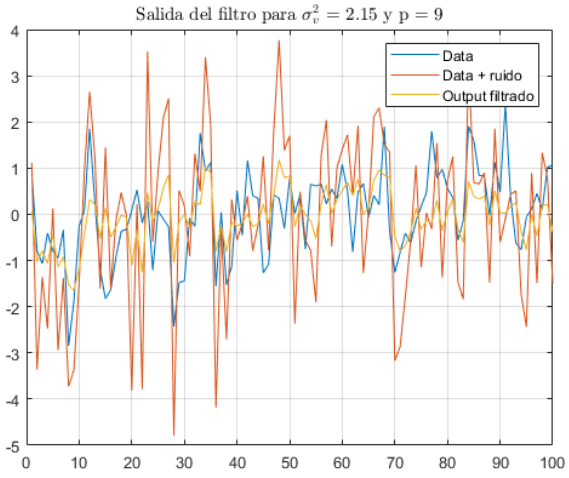
Se elige una de las varianzas para graficar de forma representativa.

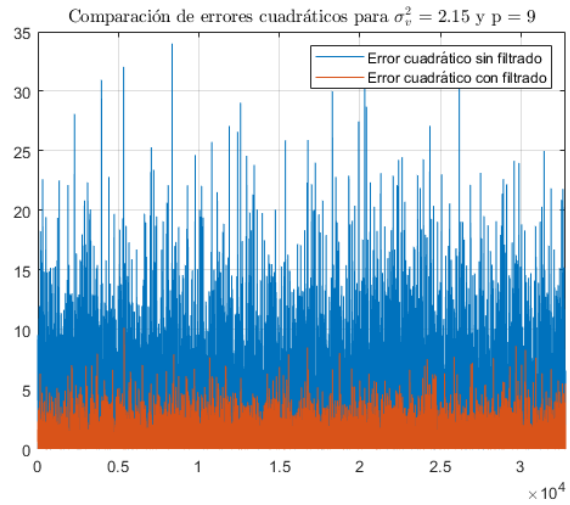
```
        if (varV == vars(var_to_plot) && (p == 2 || p == 9))
            figure()
            plot([data((nmax-nplot):nmax) input((nmax-nplot):end) output((nmax-
nplot):end)]);
            legend({'Data', 'Data + ruido', 'Output filtrado'});
            title(sprintf('Salida del filtro para $$\\sigma_v^2$$ = %.2f y p = %i', varV,
p), 'interpreter', 'latex');
            xlim([0 nplot]);
            grid on;

            figure()
            plot([err_input err_output]);
            legend({'Error cuadrático sin filtrado', 'Error cuadrático con filtrado'});
            title(sprintf("Comparaci\\'on de errores cuadr\\'aticos para $$\\sigma_v^2$$ =
%.2f y p = %i", varV, p), 'interpreter', 'latex');
            xlim([0 nmax]);
            grid on;
```

```
        end
    end
    if (varV == vars(var_to_plot))
        figure()
        plot([mse_input mse_output(:,j)], '-o');
        title(sprintf('MSE a la entrada vs MSE filtrado con $$\\sigma_v^2$$ = %.2f', varV),
'interpreter', 'latex');
        grid on;
        xlabel('p');
        ylabel('MSE');
        legend({'MSE_{in}', 'MSE_{out}'}, 'Location', 'best');
        ylim([0 ceil(1.2*max(mse_input))])
    end

    % Se guarda la informacion de en cuantas veces el filtro mejoro el MSE, con respecto al
MSE de la señal con ruido.
    improvements = [improvements mean((mse_input - mse_output(:,j)) ./ mse_input)];
end
```

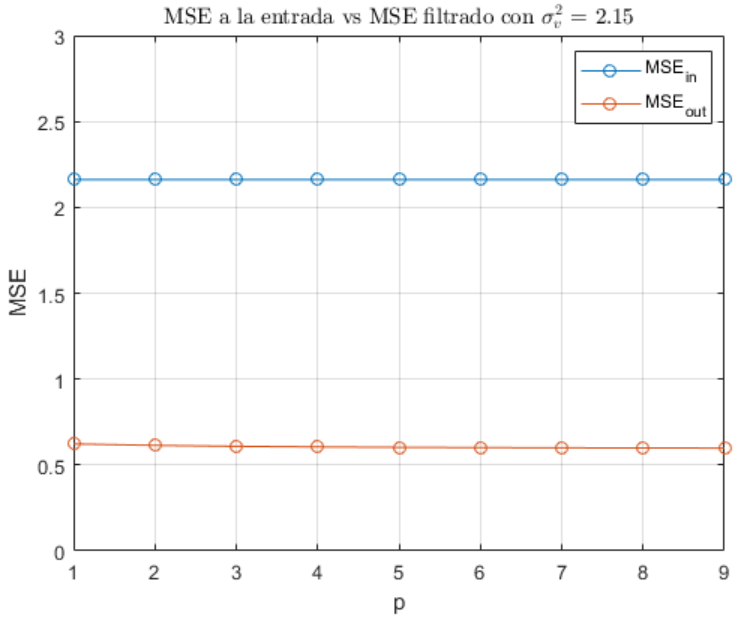








Si se observan los datos originales, la entrada ruidosa, y la salida filtrada en el tiempo, se puede apreciar que el filtro está actuando correctamente, dado que la salida filtrada se asemeja más a los valores originales. Esto se condice con lo observado en el gráfico del error cuadrático, el cual presenta una mejora considerable.



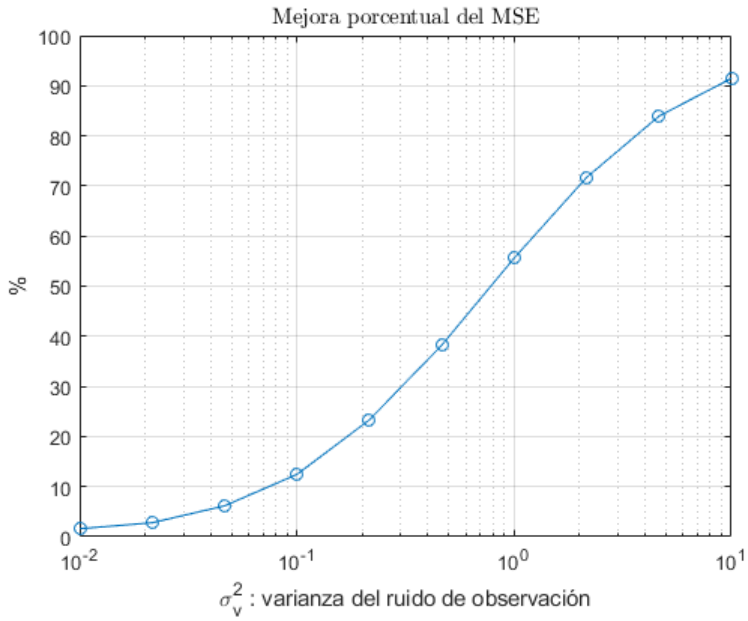
Respecto de la variación del orden del modelo AR, se observa en todos los órdenes una mejora considerable del MSE en contraste al de la entrada. Sin embargo, no es tan apreciable la mejora conforme aumenta el orden del modelo. Esto puede deberse al hecho de que para ningún orden se cumple la condición de $\phi_{k,k} \approx 0$, para $k > p$, es decir, no se ajustan bien.

```
T = array2table(mse_output,'VariableNames',string(vars),'RowName',string(1:pmax));
disp(T);
```

	0.01	0.021544	0.046416	0.1	0.21544	0.46416	1	2.1544	4.6416	10
1	0.0099095	0.020986	0.043698	0.088273	0.16719	0.29281	0.45535	0.6226	0.77434	0.87297
2	0.0099066	0.020986	0.04364	0.088123	0.16657	0.29081	0.45009	0.61425	0.76635	0.8651
3	0.0099065	0.020973	0.043604	0.088004	0.16611	0.28919	0.44691	0.6088	0.76037	0.85931
4	0.0099054	0.020969	0.043593	0.08794	0.1659	0.28838	0.44521	0.6053	0.75658	0.8555
5	0.0099028	0.020963	0.043584	0.087876	0.16579	0.28791	0.44402	0.60271	0.75398	0.85273
6	0.0099025	0.020961	0.043579	0.087859	0.16572	0.28767	0.44342	0.60127	0.75248	0.85104
7	0.0099035	0.020956	0.043574	0.087843	0.16562	0.28731	0.44286	0.59996	0.75101	0.8496
8	0.0099033	0.020955	0.043576	0.087827	0.16555	0.28711	0.44237	0.59902	0.75003	0.84849
9	0.0099034	0.020954	0.043575	0.087827	0.16554	0.28695	0.442	0.59837	0.74926	0.8476

Las columnas de la tabla representan las varianzas del ruido de observación, y las filas, el orden del modelo utilizado. Se puede observar que, al aumentar el ruido de observación, aumenta el MSE a la salida. Además, como se observó en el gráfico anterior, conforme se aumenta el orden del modelo, no hay una mejora significativa del MSE a la salida.

```
figure()
semilogx(vars, improvements .* 100, '-o')
grid on
title('Mejora porcentual del MSE', 'interpreter', 'latex');
xlabel('\sigma_v^2 : varianza del ruido de observación')
ylabel('%')
```



Cuando la varianza del ruido es muy baja, no hay mucho margen para la mejora con el filtro, por lo que la mejora porcentual del MSE no es amplia. Sin embargo, al aumentar la varianza, la influencia del filtro de Kalman se hace mucho más notoria, llegando a mejoras del 90%.

Conclusión

En este trabajo se pudo aplicar el filtro de Kalman con éxito en todos los casos debido a que siempre hubo una mejora del MSE de la salida respecto del de la entrada. No se logró apreciar una mejora significativa del filtrado con el aumento del orden del modelo. Esto puede deberse a que ninguno de los órdenes de los modelos ajustaba por completo a los datos.