



5ª ESCUELA DE VERANO DE CONTROL TURIX-DYNAMICS

## El filtro de Kalman

Un curso sobre estimación de estados y parámetros

I. Santos-Ruiz & F.R. López-Estrada

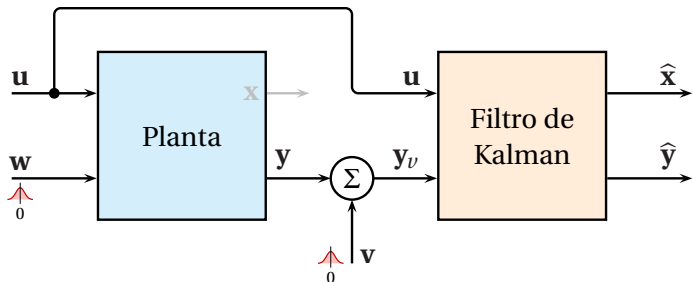
Tecnológico Nacional de México  
Instituto Tecnológico de Tuxtla Gutiérrez

**Turix-Dynamics** Diagnosis and Control Group

- Estimación en condiciones inciertas: variables aleatorias
- Filtro de Kalman
- Implementación del filtro de Kalman
- Filtrado de señales sin modelo

Documentos y código fuente:

<https://github.com/isantosruiz/kalman>



El **filtro de Kalman** (FK) es un algoritmo para estimar los estados/parámetros de un sistema en presencia de incertidumbre. El FK se usa ampliamente en aplicaciones de seguimiento, navegación y control.

## 🚀 MATLAB

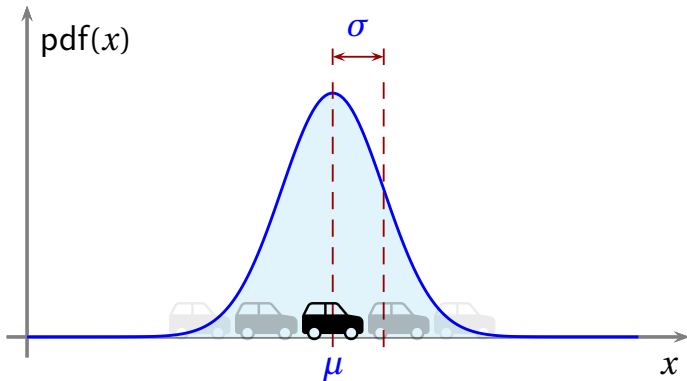
```
A = diag([-1,-2,-3]); B = [1;-1;1]; C = [0,1,0];
continuo = ss(A,B,C,0);
discreto = c2d(continuo,0.01);
A = discreto.a; B = discreto.b; C = discreto.c;
%-----
t = 0:0.01:10;
u = sin(t);
[y,t,x] = lsim(discreto,u,t,[0.3;0.5;0.7]);
y = y + 0.1*randn(size(y));
%-----
kf = KalmanFilter(A,B,C,eye(3)*1e-6,0.01);
xhat = kf.estimate(u,y');
yhat = C*xhat;
subplot(211); plot(t,y,t,yhat,'Linewidth',1)
subplot(212); plot(t,xhat,t,x,'--','LineWidth',1)
```

## Estimación en condiciones inciertas

Un coche partió de  $x = 0$  en  $t = 0$  con velocidad  $v_0 = 2 \text{ m/s}$ . ¿Dónde estará el coche en  $t = 10 \text{ s}$ ?

# Estimación en condiciones inciertas

Un coche partió de  $x = 0$  en  $t = 0$  con velocidad  $v_0 = 2 \text{ m/s}$ . ¿Dónde estará el coche en  $t = 10 \text{ s}$ ?



¿Se puede saber con certeza dónde está el carro?

Una variable aleatoria es un conjunto de posibles valores de un experimento cuyo resultado es incierto. Se describe mediante una **función de densidad de probabilidad** (PDF).

Una **variable aleatoria continua**  $X$  puede tomar cualquier valor  $x$  dentro de un intervalo o dominio específico  $\mathcal{D}$ , como el tiempo de carga de una batería o el tiempo de carrera de maratón. Además,

$$\int_{\mathcal{D}} f(x) \mathrm{d}x = 1.$$

El valor esperado de una variable aleatoria continua con PDF dada por  $f(x)$  se define por:

$$\mathbb{E}[X] = \int_{\mathcal{D}} x f(x) \mathrm{d}x$$

Las PDF se caracterizan por ciertas medidas denominadas **momentos**, que son los valores esperados de las potencias de la variable aleatoria.

Existen dos tipos de momentos:

- El  **$k$ -ésimo momento** es el valor esperado de la  $k$ -ésima potencia de la variable aleatoria:  $\mathbb{E}[X^k]$

El primer momento  $\mathbb{E}[X]$  es la **media** ( $\mu$ ) de la variable aleatoria.

- El  **$k$ -ésimo momento central** es el valor esperado de la  $k$ -ésima potencia de la distribución de la variable aleatoria respecto a su media:  $\mathbb{E}[(X - \mu)^k]$

El segundo momento central  $\mathbb{E}[(X - \mu)^2]$  es la **varianza** ( $\sigma^2$ ) de la variable aleatoria.



**Ejercicio:** Hallar la media (primer momento) y la varianza (segundo momento central) de la variable aleatoria definida por la siguiente PDF triangular en  $\mathcal{D}$ :  $0 \leq x \leq 4$ :

$$f(x) = \begin{cases} x/6 & \text{para } 0 \leq x < 3, \\ (4-x)/2 & \text{para } 3 \leq x \leq 4. \end{cases}$$

## 🚩 MATLAB

```
f = @(x) (x/6).*double(x<3)+(4-x)/2.*double(x>=3);  
fplot(f,[0,4])  
integral(f,0,4) % ¿Es una PDF?  
%-----  
% Cálculo analítico (aproximado numéricamente}  
mu = integral(@(x)x.*f(x),0,4)  
sigma2 = integral(@(x)(x-mu).^2.*f(x),0,4)  
%-----  
% Cálculo por fuerza bruta}  
pdf = makedist('Triangular','A',0,'B',3,'C',4);  
x = random(pdf,1000000,1);  
mu = mean(x)  
sigma2 = var(x)
```

# La PDF normal o “gaussiana”

Normal univariada:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2} \frac{(x-\mu)^2}{\sigma^2}\right)$$

# La PDF normal o “gaussiana”

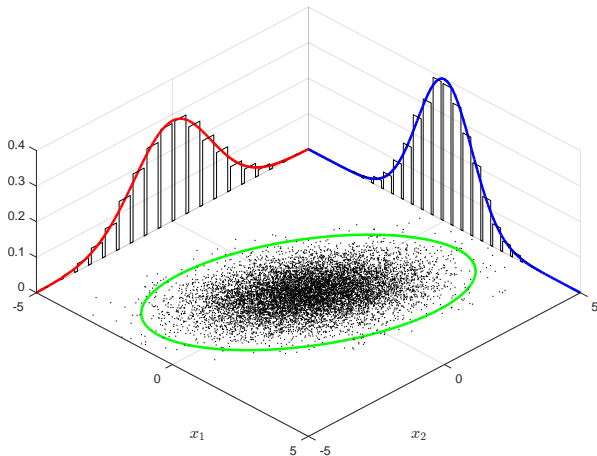
Normal univariada:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2} \frac{(x-\mu)^2}{\sigma^2}\right)$$

Normal multivariada:

$$f(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp\left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^\top \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})\right)$$

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix} \quad \Sigma = \begin{bmatrix} \sigma_{11}^2 & \sigma_{12}^2 & \cdots & \sigma_{1n}^2 \\ \sigma_{21}^2 & \sigma_{22}^2 & \cdots & \sigma_{2n}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1}^2 & \sigma_{n2}^2 & \cdots & \sigma_{nn}^2 \end{bmatrix}$$

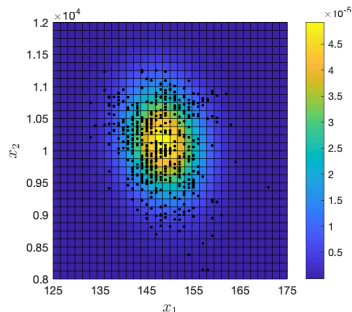
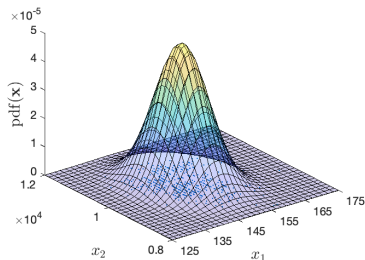


Las proyecciones sobre los ejes de  $x_1$  y  $x_2$  son gaussianas.

# Estimando una PDF a partir de muestras

## MATLAB

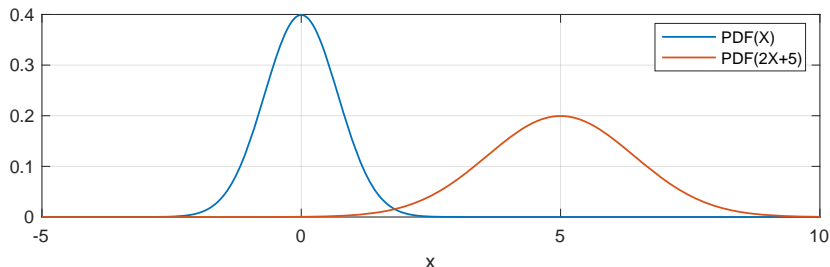
```
[x,t] = chemical_dataset;  
x = x(1:2,:)' ;  
mu = mean(x)  
Sigma = cov(x)
```



Si  $X$  es una variable aleatoria normal,  $X \sim \mathcal{N}(\mu, \sigma^2)$ , entonces  $Y := aX + b$  también es aleatoria normal,  $Y \sim \mathcal{N}(\mu + b, \sigma^2 a^2)$ .

# Transformación lineal de variables aleatorias

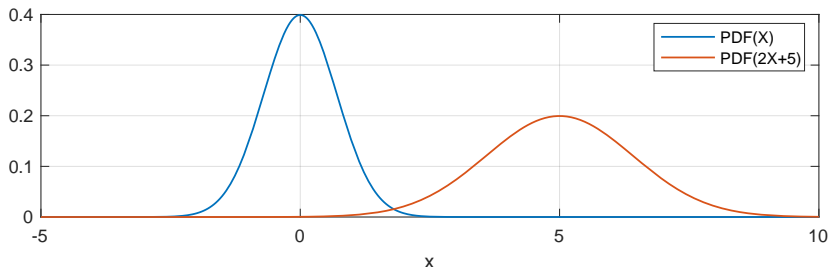
Si  $X$  es una variable aleatoria normal,  $X \sim \mathcal{N}(\mu, \sigma^2)$ , entonces  $Y := aX + b$  también es aleatoria normal,  $Y \sim \mathcal{N}(\mu + b, \sigma^2 a^2)$ .





# Transformación lineal de variables aleatorias

Si  $X$  es una variable aleatoria normal,  $X \sim \mathcal{N}(\mu, \sigma^2)$ , entonces  $Y := aX + b$  también es aleatoria normal,  $Y \sim \mathcal{N}(\mu + b, \sigma^2 a^2)$ .

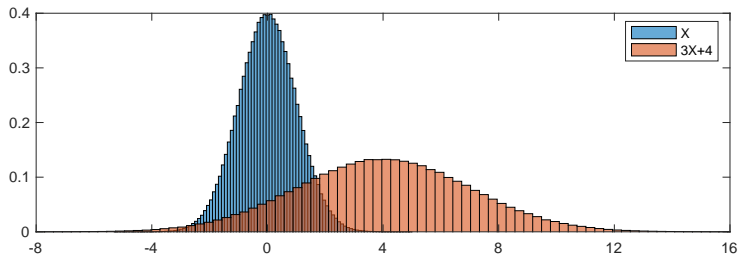


**Ejercicio:** Generar 1 000 000 muestras pseudoaleatorias con distribución normal estandarizada (i.e.  $\mu = 0, \sigma^2 = 1$ ) y almacenarlas en un arreglo  $x$ . Luego, calcular  $y = 3x + 4$ , obtener su media y su varianza. Finalmente, graficar los histogramas de  $x$  e  $y$ .

# Transformación lineal de variables aleatorias

## MATLAB

```
x = randn(1000000,1);  
mu_x = mean(x)  
sigma2_x = var(x)  
y = 3*x+4;  
mu_y = mean(y)  
sigma2_y = var(y)  
histogram(x,100,'Normalization','pdf'); hold on  
histogram(y,100,'Normalization','pdf'); hold off
```



Consideremos el promediado de  $n$  números que se obtienen secuencialmente en el tiempo:

$$p_n = \frac{1}{n} \sum_{k=1}^n x_k$$

Consideremos el promediado de  $n$  números que se obtienen secuencialmente en el tiempo:

$$p_n = \frac{1}{n} \sum_{k=1}^n x_k = \frac{1}{n} \left( x_n + \sum_{k=1}^{n-1} x_k \right)$$

Consideremos el promediado de  $n$  números que se obtienen secuencialmente en el tiempo:

$$p_n = \frac{1}{n} \sum_{k=1}^n x_k = \frac{1}{n} \left( x_n + \sum_{k=1}^{n-1} x_k \right)$$

$$p_{n-1} = \frac{1}{n-1} \sum_{k=1}^{n-1} x_k$$

Consideremos el promediado de  $n$  números que se obtienen secuencialmente en el tiempo:

$$p_n = \frac{1}{n} \sum_{k=1}^n x_k = \frac{1}{n} \left( x_n + \sum_{k=1}^{n-1} x_k \right)$$

$$p_{n-1} = \frac{1}{n-1} \sum_{k=1}^{n-1} x_k \quad \Rightarrow \quad \sum_{k=1}^{n-1} x_k = (n-1) p_{n-1}$$

Consideremos el promediado de  $n$  números que se obtienen secuencialmente en el tiempo:

$$p_n = \frac{1}{n} \sum_{k=1}^n x_k = \frac{1}{n} \left( x_n + \sum_{k=1}^{n-1} x_k \right)$$
$$p_{n-1} = \frac{1}{n-1} \sum_{k=1}^{n-1} x_k \quad \Rightarrow \quad \sum_{k=1}^{n-1} x_k = (n-1) p_{n-1}$$
$$p_n = p_{n-1} + \frac{1}{n} (x_n - p_{n-1})$$

En el tiempo  $n$  no es necesario tener almacenada toda la sucesión de muestras, porque el promedio previo  $p_{n-1}$  contiene resumida la historia del sistema.

# Codificando el promediador

## promediador.m

```
classdef promediador < handle
    properties
        n
        p
    end
    methods
        function obj = promediador
            obj.n = 0;
            obj.p = 0;
        end
        function p = agregar(obj,x)
            obj.n = obj.n + 1;
            obj.p = obj.p + 1/obj.n*(x - obj.p);
            p = obj.p;
        end
    end
end
```



**Ejercicio:** Compro “un kilo” de tortillas, pero no confío en que la cantidad entregada sea la correcta, así que efectúo una serie de pesajes con mi báscula casera. Tomo una nueva medición cada minuto, registrando los siguientes valores:

$n$ (min)	0	1	2	3	4	5
$z$ (kg)	1.000	0.980	0.972	0.973	0.970	0.967

¿Cuál es la mejor estimación del peso considerando el promedio acumulado después de 5 minutos?

# Una aproximación al filtro de Kalman

Por la ecuación del promediador, suponiendo que  $\{z_n\}$  son las medidas y  $\{\hat{x}_n\}$  son las mejores estimaciones del peso:

$$\hat{x}_n = \hat{x}_{n-1} + \frac{1}{n} (z_n - \hat{x}_{n-1})$$

Estimaciones sucesivas:

$$\hat{x}_0 = 1.000$$

$$\hat{x}_1 = \hat{x}_0 + \frac{1}{1} (z_1 - \hat{x}_0) = 0.990$$

$$\hat{x}_2 = \hat{x}_1 + \frac{1}{2} (z_2 - \hat{x}_1) = 0.984$$

$$\hat{x}_3 = \hat{x}_2 + \frac{1}{3} (z_3 - \hat{x}_2) = 0.981$$

$$\hat{x}_4 = \hat{x}_3 + \frac{1}{4} (z_4 - \hat{x}_3) = 0.979$$

$$\hat{x}_5 = \hat{x}_4 + \frac{1}{5} (z_5 - \hat{x}_4) = 0.977$$

# Una aproximación al filtro de Kalman

Notación “estadística” simplificada:

$$\hat{x}_{n|n} = \hat{x}_{n|n-1} + \frac{1}{n} (z_n - \hat{x}_{n|n-1})$$

$$\boxed{\text{Estimación del estado actual}} = \boxed{\text{Predicción del estado actual}} + \boxed{\text{Factor}} \times \left( \boxed{\text{Medición}} - \boxed{\text{Predicción del estado actual}} \right)$$

👉 En el ejercicio anterior, se dio la misma importancia al valor declarado por el vendedor que a la medición obtenida por el usuario. ¿Es posible asignar niveles de confianza diferentes? **No todos los vendedores son igual de honestos y no todas las balanzas son igual de exactas.**

# Una aproximación al filtro de Kalman

Notación “estadística” simplificada:

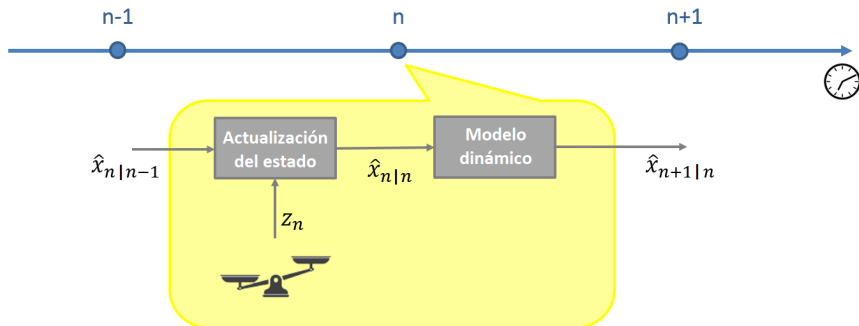
$$\hat{x}_{n|n} = \hat{x}_{n|n-1} + \frac{1}{n} (z_n - \hat{x}_{n|n-1})$$

$$\boxed{\text{Estimación del estado actual}} = \boxed{\text{Predicción del estado actual}} + \boxed{\text{Factor}} \times \left( \boxed{\text{Medición}} - \boxed{\text{Predicción del estado actual}} \right)$$

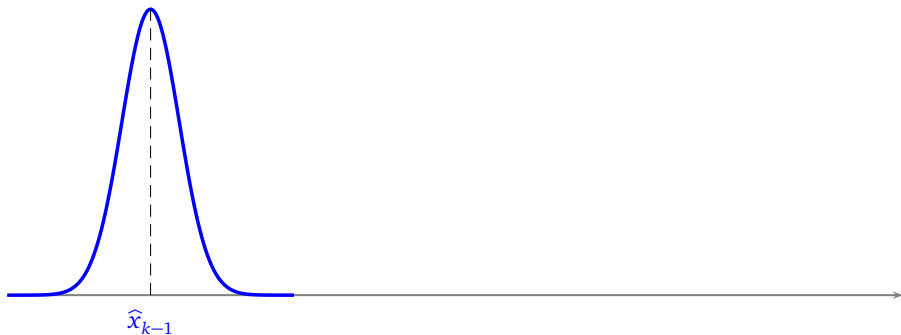
- 👉 En el ejercicio anterior, se dio la misma importancia al valor declarado por el vendedor que a la medición obtenida por el usuario. ¿Es posible asignar niveles de confianza diferentes? **No todos los vendedores son igual de honestos y no todas las balanzas son igual de exactas.**
- 👉 Las mediciones sucesivas parecen disminuir con el tiempo, posiblemente porque las tortillas pierden humedad y, en consecuencia, peso. ¿Es posible incluir la dinámica de la pérdida de humedad para obtener predicciones más confiables?

# Una aproximación al filtro de Kalman

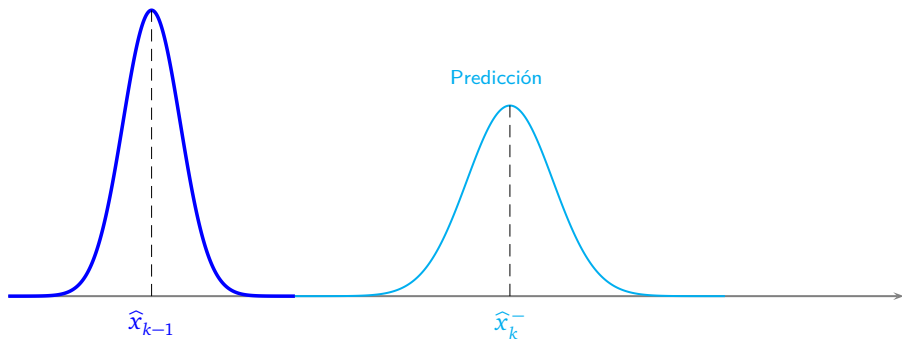
$$\hat{x}_{n|n} = \hat{x}_{n|n-1} + \underbrace{K_n}_{\text{Ganancia}} \underbrace{(z_n - \hat{x}_{n|n-1})}_{\text{Innovación}}$$



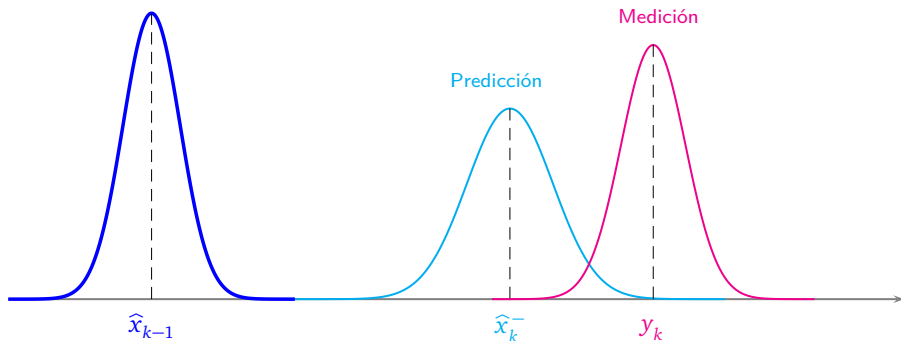
## Propagación de distribuciones gaussianas



## Propagación de distribuciones gaussianas

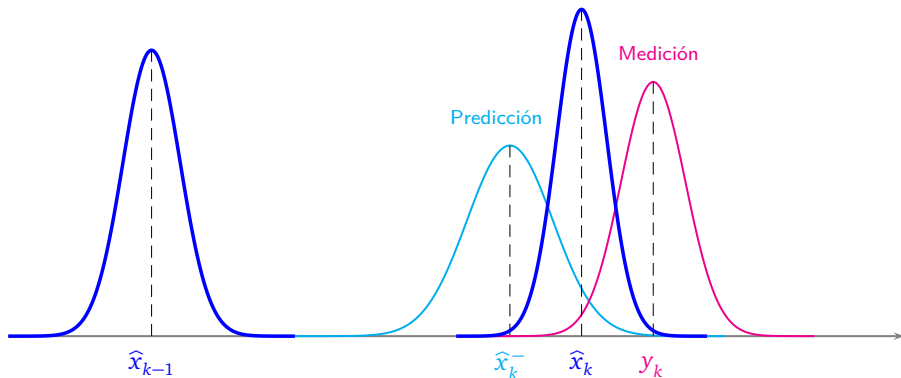


## Propagación de distribuciones gaussianas

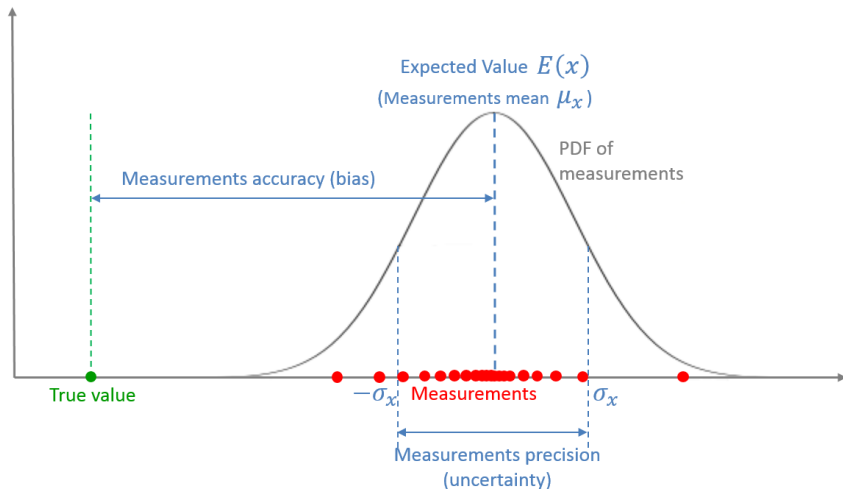




## Propagación de distribuciones gaussianas



# Estadística de las mediciones



# Discretización de modelos lineales

En tiempo continuo:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)$$



En tiempo discreto:

$$\mathbf{x}_{k+1} = \mathbf{A}_d \mathbf{x}_k + \mathbf{B}_d \mathbf{u}_k$$

$$\mathbf{y}_k = \mathbf{C} \mathbf{x}_k + \mathbf{D} \mathbf{u}_k$$

# Discretización de modelos lineales

En tiempo continuo:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)$$



En tiempo discreto:

$$\mathbf{x}_{k+1} = \mathbf{A}_d \mathbf{x}_k + \mathbf{B}_d \mathbf{u}_k$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{D}\mathbf{u}_k$$

$$\mathbf{A}_d = \exp(\mathbf{A}T_s) = \mathbf{I} + \sum_{i=1}^{\infty} \frac{1}{i!} (\mathbf{A}T_s)^i$$

$$\mathbf{B}_d = \left( \int_0^{T_s} \exp(\mathbf{A}\tau) d\tau \right) \mathbf{B} = \mathbf{A}^{-1} (\exp(\mathbf{A}T_s) - \mathbf{I}) \mathbf{B}$$

# Discretización de modelos lineales

En tiempo continuo:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)$$



En tiempo discreto:

$$\mathbf{x}_{k+1} = \mathbf{A}_d \mathbf{x}_k + \mathbf{B}_d \mathbf{u}_k$$


$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{D}\mathbf{u}_k$$

$$\mathbf{A}_d = \exp(\mathbf{A}T_s) = \mathbf{I} + \sum_{i=1}^{\infty} \frac{1}{i!} (\mathbf{A}T_s)^i$$

$$\mathbf{B}_d = \left( \int_0^{T_s} \exp(\mathbf{A}\tau) d\tau \right) \mathbf{B} = \mathbf{A}^{-1} (\exp(\mathbf{A}T_s) - \mathbf{I}) \mathbf{B}$$

Atajo numérico:

$$\exp \left( \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} T_s \right) = \begin{bmatrix} \mathbf{A}_d & \mathbf{B}_d \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$$

 `c2d()`

- 👉 El filtro de Kalman es **óptimo** en el sentido de que proporciona la estimación del estado con la **mínima varianza del error**, dadas las mediciones y el modelo dinámico del sistema.

- 👉 El filtro de Kalman es **óptimo** en el sentido de que proporciona la estimación del estado con la **mínima varianza del error**, dadas las mediciones y el modelo dinámico del sistema.
- 👉 Se asume que existe incertidumbre en la dinámica del sistema (**ruido de proceso**: estímulos inciertos, dinámicas desconocidas o difíciles de modelar). También se considera incertidumbre en la medición (**ruido de medición**: interferencia y mala calibración en los sensores, imprecisión por baja resolución). Los ruidos se asumen blancos, de media cero y no correlacionados en el tiempo (e.g. gaussianos).

- 👉 El filtro de Kalman es **óptimo** en el sentido de que proporciona la estimación del estado con la **mínima varianza del error**, dadas las mediciones y el modelo dinámico del sistema.
- 👉 Se asume que existe incertidumbre en la dinámica del sistema (**ruido de proceso**: estímulos inciertos, dinámicas desconocidas o difíciles de modelar). También se considera incertidumbre en la medición (**ruido de medición**: interferencia y mala calibración en los sensores, imprecisión por baja resolución). Los ruidos se asumen blancos, de media cero y no correlacionados en el tiempo (e.g. gaussianos).
- 👉 El filtro de Kalman se puede implementar en tiempo real (en línea) con bajo costo computacional.



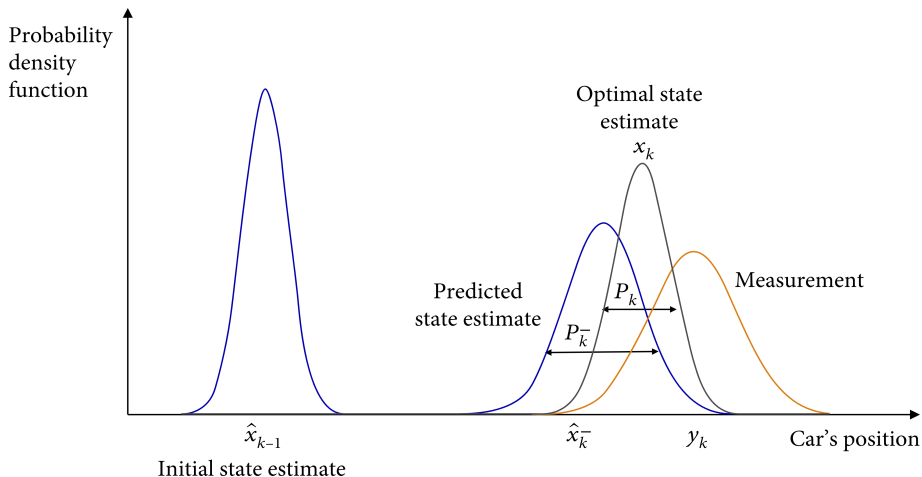
El filtro de Kalman es un estimador insesgado:

$$\mathbb{E}[\hat{\mathbf{x}}] = \mathbb{E}[\mathbf{x}]$$

Más precisamente, es el estimador insesgado de mínima varianza (MVUE, **minimum variance unbiased estimator**):

$$\underset{\hat{\mathbf{x}}}{\operatorname{argmín}} \mathbb{E}[(\mathbf{x} - \hat{\mathbf{x}})(\mathbf{x} - \hat{\mathbf{x}})^{\top}]$$

# Idea general del proceso de filtrado



# Filtro de Kalman en tiempo discreto

## Señales y sistemas

Dinámica lineal estocástica en tiempo discreto:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{w}_k$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{v}_k$$

# Filtro de Kalman en tiempo discreto

## Señales y sistemas

Dinámica lineal estocástica en tiempo discreto:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{w}_k$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{v}_k$$

Modelo de ruido blanco gaussiano:

$$\mathbb{E}[\mathbf{w}_k] = \mathbf{0} \qquad \mathbb{E}[\mathbf{v}_k] = \mathbf{0}$$

# Filtro de Kalman en tiempo discreto

## Señales y sistemas

Dinámica lineal estocástica en tiempo discreto:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{w}_k$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{v}_k$$

Modelo de ruido blanco gaussiano:

$$\mathbb{E}[\mathbf{w}_k] = \mathbf{0} \qquad \mathbb{E}[\mathbf{v}_k] = \mathbf{0}$$

$$\mathbb{E}[\mathbf{w}_k \mathbf{w}_i^T] = \begin{cases} \mathbf{Q} & \text{para } i = k \\ \mathbf{0} & \text{para } i \neq k \end{cases}$$

$$\mathbb{E}[\mathbf{v}_k \mathbf{v}_i^T] = \begin{cases} \mathbf{R} & \text{para } i = k \\ \mathbf{0} & \text{para } i \neq k \end{cases}$$

$$\mathbb{E}[\mathbf{w}_k \mathbf{v}_i^T] = \mathbf{0} \quad \text{para toda } k, i$$

# Filtro de Kalman en tiempo discreto

Actualización con la medición: `correct()`

Sea  $\hat{\mathbf{x}}_k^-$  una estimación previa (a priori) de  $\mathbf{x}_k$ :

$$\hat{\mathbf{e}}_k^- = \mathbf{x}_k - \hat{\mathbf{x}}_k^-$$

$$\hat{\mathbf{P}}_k^- = \mathbb{E}[\hat{\mathbf{e}}_k^- \hat{\mathbf{e}}_k^{-\top}] = \mathbb{E}[(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)^{\top}]$$

Cuando están disponibles las mediciones ( $\mathbf{y}$ ) se actualiza la estimación:

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{y}_k - \mathbf{C}\hat{\mathbf{x}}_k^-)$$

donde

$\hat{\mathbf{x}}_k$  : Estimación actualizada (a posteriori) de  $\mathbf{x}_k$

$\mathbf{K}_k$  : Factor de ajuste (Ganancia de Kalman)

# Filtro de Kalman en tiempo discreto

## Ganancia de Kalman

$$\begin{aligned}\mathbf{P}_k &= \mathbb{E}[\mathbf{e}_k \mathbf{e}_k^\top] = \mathbb{E}[(\mathbf{x}_k - \hat{\mathbf{x}}_k) (\mathbf{x}_k - \hat{\mathbf{x}}_k)^\top] \\ &= (\mathbf{I} - \mathbf{K}_k \mathbf{C}) \mathbf{P}_k^- (\mathbf{I} - \mathbf{K}_k \mathbf{C})^\top + \mathbf{K}_k \mathbf{R} \mathbf{K}_k^\top\end{aligned}$$

# Filtro de Kalman en tiempo discreto

## Ganancia de Kalman

$$\begin{aligned}\mathbf{P}_k &= \mathbb{E}[\mathbf{e}_k \mathbf{e}_k^\top] = \mathbb{E}[(\mathbf{x}_k - \hat{\mathbf{x}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k)^\top] \\ &= (\mathbf{I} - \mathbf{K}_k \mathbf{C}) \mathbf{P}_k^- (\mathbf{I} - \mathbf{K}_k \mathbf{C})^\top + \mathbf{K}_k \mathbf{R} \mathbf{K}_k^\top\end{aligned}$$

Cálculo de la ganancia óptima:

$$\frac{d(\text{tr}(\mathbf{P}_k))}{d\mathbf{K}_k} = \mathbf{0}$$



# Filtro de Kalman en tiempo discreto

## Ganancia de Kalman

$$\begin{aligned}\mathbf{P}_k &= \mathbb{E}[\mathbf{e}_k \mathbf{e}_k^\top] = \mathbb{E}[(\mathbf{x}_k - \hat{\mathbf{x}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k)^\top] \\ &= (\mathbf{I} - \mathbf{K}_k \mathbf{C}) \mathbf{P}_k^- (\mathbf{I} - \mathbf{K}_k \mathbf{C})^\top + \mathbf{K}_k \mathbf{R} \mathbf{K}_k^\top\end{aligned}$$

Cálculo de la ganancia óptima:

$$\frac{d(\text{tr}(\mathbf{P}_k))}{d\mathbf{K}_k} = \mathbf{0} \quad \Rightarrow \quad \mathbf{K}_k = \mathbf{P}_k^- \mathbf{C}^\top (\mathbf{C} \mathbf{P}_k^- \mathbf{C}^\top + \mathbf{R})^{-1}$$

# Filtro de Kalman en tiempo discreto

## Ganancia de Kalman

$$\begin{aligned}\mathbf{P}_k &= \mathbb{E}[\mathbf{e}_k \mathbf{e}_k^\top] = \mathbb{E}[(\mathbf{x}_k - \hat{\mathbf{x}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k)^\top] \\ &= (\mathbf{I} - \mathbf{K}_k \mathbf{C}) \mathbf{P}_k^- (\mathbf{I} - \mathbf{K}_k \mathbf{C})^\top + \mathbf{K}_k \mathbf{R} \mathbf{K}_k^\top\end{aligned}$$

Cálculo de la ganancia óptima:

$$\frac{d(\text{tr}(\mathbf{P}_k))}{d\mathbf{K}_k} = \mathbf{0} \quad \Rightarrow \quad \mathbf{K}_k = \mathbf{P}_k^- \mathbf{C}^\top (\mathbf{C} \mathbf{P}_k^- \mathbf{C}^\top + \mathbf{R})^{-1}$$

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

# Filtro de Kalman en tiempo discreto

Actualización en el tiempo: `predict()`

$$\hat{\mathbf{x}}_{k+1}^- = \mathbf{A}\hat{\mathbf{x}}_k + \mathbf{B}\mathbf{u}_k$$

# Filtro de Kalman en tiempo discreto

Actualización en el tiempo: **predict()**

$$\hat{\mathbf{x}}_{k+1}^- = \mathbf{A}\hat{\mathbf{x}}_k + \mathbf{B}\mathbf{u}_k$$

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

# Filtro de Kalman en tiempo discreto

Actualización en el tiempo: **predict()**

$$\hat{\mathbf{x}}_{k+1}^- = \mathbf{A}\hat{\mathbf{x}}_k + \mathbf{B}\mathbf{u}_k$$

$$\mathbf{e}_{k+1}^- = \mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1}^- \quad \Rightarrow \quad \mathbf{e}_{k+1}^- = \mathbf{A}\mathbf{e}_k + \mathbf{w}_k$$

# Filtro de Kalman en tiempo discreto

Actualización en el tiempo: **predict()**

$$\hat{\mathbf{x}}_{k+1}^- = \mathbf{A}\hat{\mathbf{x}}_k + \mathbf{B}\mathbf{u}_k$$

$$\mathbf{e}_{k+1}^- = \mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1}^- \quad \implies \quad \mathbf{e}_{k+1}^- = \mathbf{A}\mathbf{e}_k + \mathbf{w}_k$$

$$\mathbf{P}_{k+1}^- = \mathbb{E}[\mathbf{e}_{k+1}^- \mathbf{e}_{k+1}^{-\top}]$$

# Filtro de Kalman en tiempo discreto

Actualización en el tiempo: **predict()**

$$\hat{\mathbf{x}}_{k+1}^- = \mathbf{A}\hat{\mathbf{x}}_k + \mathbf{B}\mathbf{u}_k$$

$$\mathbf{e}_{k+1}^- = \mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1}^- \quad \implies \quad \mathbf{e}_{k+1}^- = \mathbf{A}\mathbf{e}_k + \mathbf{w}_k$$

$$\mathbf{P}_{k+1}^- = \mathbb{E}[\mathbf{e}_{k+1}^- \mathbf{e}_{k+1}^{-\top}] \quad \implies \quad \mathbf{P}_{k+1}^- = \mathbf{A}\mathbf{P}_k\mathbf{A}^\top + \mathbf{Q}$$

# Filtro de Kalman en tiempo discreto

Actualización en el tiempo: **predict()**

$$\hat{\mathbf{x}}_{k+1}^- = \mathbf{A}\hat{\mathbf{x}}_k + \mathbf{B}\mathbf{u}_k$$

$$\mathbf{e}_{k+1}^- = \mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1}^- \quad \implies \quad \mathbf{e}_{k+1}^- = \mathbf{A}\mathbf{e}_k + \mathbf{w}_k$$

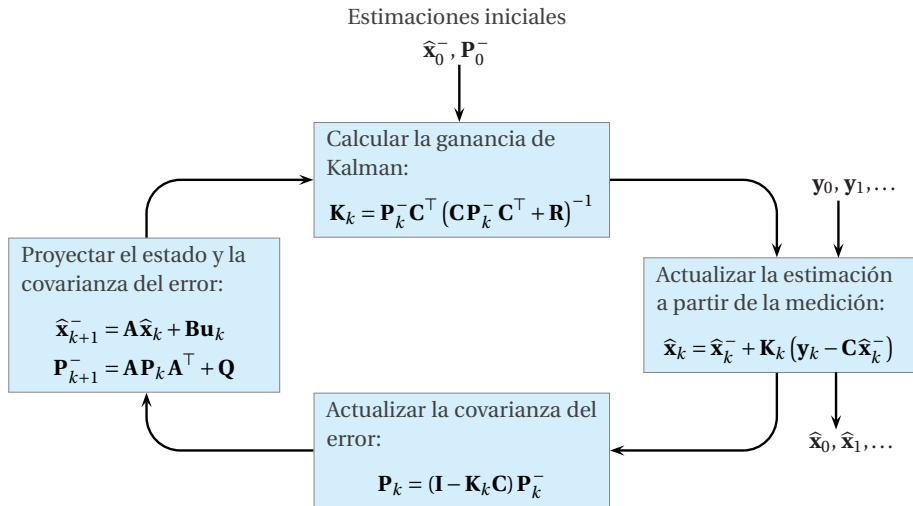
$$\mathbf{P}_{k+1}^- = \mathbb{E}[\mathbf{e}_{k+1}^- \mathbf{e}_{k+1}^{-\top}] \quad \implies \quad \mathbf{P}_{k+1}^- = \mathbf{A}\mathbf{P}_k\mathbf{A}^\top + \mathbf{Q}$$

Para cerrar el ciclo del FK:

$$k+1 \rightarrow k$$



# Filtro de Kalman: El algoritmo



## Requisitos de entrada

- Parámetros del sistema:  $\mathbf{A}$ ,  $\mathbf{B}$ ,  $\mathbf{C}$ ,  $\mathbf{Q}$ ,  $\mathbf{R}$ ,  $T_s$ .
- Estimaciones iniciales:  $\hat{\mathbf{x}}_0^-, \mathbf{P}_0^-$ .
- En cada paso de tiempo  $k$ , proporcionar:  $\mathbf{u}_k$ ,  $\mathbf{y}_k$ .

## Subrutinas o métodos

- `initialize( $\hat{\mathbf{x}}_0^-, \mathbf{P}_0^-$ )`: Inicializa el vector de estado y la matriz de covarianza del error.
- `[ $\hat{\mathbf{x}}_k, \mathbf{P}_k$ ] = correct( $\mathbf{y}_k, \hat{\mathbf{x}}_k^-, \mathbf{P}_k^-$ )`: Calcula la ganancia  $\mathbf{K}_k$ . Luego, usa esa ganancia para estimar el estado y la covarianza del error a partir de la medición actual y de la predicción obtenida de la dinámica del sistema.
- `[ $\hat{\mathbf{x}}_{k+1}^-, \mathbf{P}_{k+1}^-$ ] = predict( $\mathbf{u}_k, \hat{\mathbf{x}}_k$ )`: Predice el estado del sistema y la covarianza del error para el siguiente paso de tiempo.

# Código del FK orientado a objetos

📁 KalmanFilter.m

```
classdef KalmanFilter < handle

% KalmanFilter : State estimator for linear systems
% Please read the attached license file

    properties
        A    % State-transition matrix
        B    % Control matrix
        C    % Observation matrix
        Q    % Process noise covariance matrix
        R    % Measurement noise covariance matrix
        K    % Kalman gain
        x    % Estimated state
        P    % Error covariance matrix
    end
```

# Código del FK orientado a objetos

🚀 KalmanFilter.m

(continuación)

methods

```
function obj = KalmanFilter(A,B,C,Q,R,Ts)
```

```
    obj.A = A; obj.B = B; obj.C = C;
```

```
    obj.Q = Q; obj.R = R;
```

```
    if nargin > 5
```

```
        obj.A = expm(A*Ts);
```


```
        obj.B = (obj.A-eye(size(A)))/A*B;
```

```
    end
```

```
    obj.initialize
```

```
end
```

# Código del FK orientado a objetos

 KalmanFilter.m

(continuación)

```
function initialize(obj,x,P)
    if nargin < 2
        obj.x = zeros(size(obj.A,2),1);
    else
        obj.x = x;
    end
    if nargin < 3
        obj.P = eye(size(obj.A));
    else
        obj.P = P;
    end
end
function x = predict(obj,u)
    obj.x = obj.A*obj.x + obj.B*u;
    obj.P = obj.A*obj.P*obj.A' + obj.Q;
    x = obj.x;
end
```

# Código del FK orientado a objetos

🚩 KalmanFilter.m

(continuación)

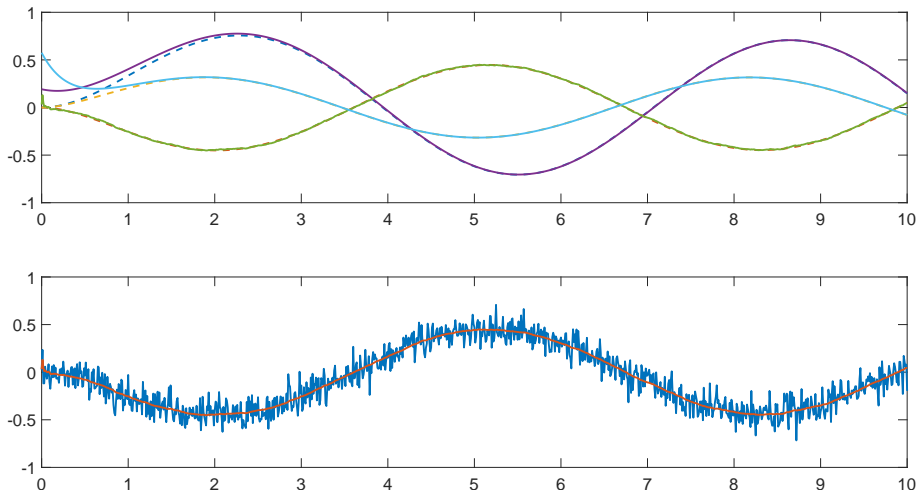
```
function x = correct(obj,y)
    obj.K = obj.P*obj.C'/(obj.C*obj.P*obj.C'+obj.R);
    obj.x = obj.x + obj.K*(y-obj.C*obj.x);
    obj.P = (eye(size(obj.K*obj.C))-obj.K*obj.C)*obj.P;
    x = obj.x;
end
function x = estimate(obj,u,y)
    n = size(obj.x,1);
    m = size(y,2);
    x = zeros(n,m);
    for k = 1:m
        x(:,k) = obj.correct(y(:,k));
        obj.predict(u(:,k));
    end
end
end
end
```

# Ejemplo de estimación de estados

## MATLAB

```
A = diag([-1,-2,-3]); B = [1;-1;1]; C = [0,1,0];
planta = ss(A,B,C,0);
Q = 1e-5*eye(3); R = 1e-2; Ts = 0.01;
%-
t = 0:Ts:10; u = sin(t);
[y,t,x] = lsim(planta,u,t,[0;0;0]);
y = y + sqrt(R)*randn(size(y));
%-
kf = KalmanFilter(A,B,C,Q,R,Ts);
kf.initialize(rand(3,1));
xhat = zeros(3,1001);
for k = 1:numel(t)
    xhat(:,k) = kf.estimate(u(k),y(k));
end
% xhat = kf.estimate(u,y');
subplot(211); plot(t,x,'--',t,xhat)
subplot(212); plot(t,y,t,C*xhat)
```

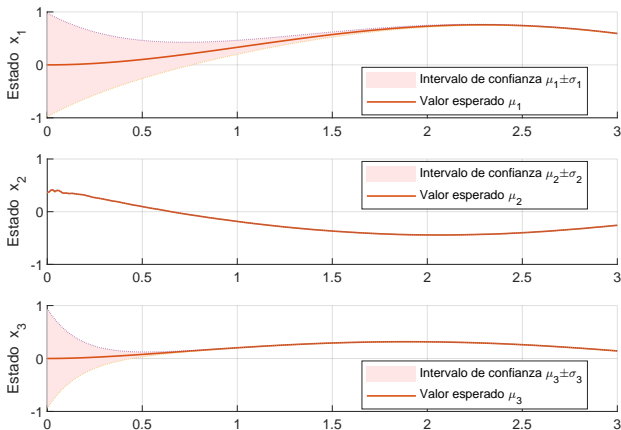
# Ejemplo de estimación de estados





# Confianza de las estimaciones

El valor esperado del estado está dado por  $\hat{\mathbf{x}}$ , y su incertidumbre por la diagonal de  $\mathbf{P}$ . Así, la  $i$ -ésima variable de estado es una variable aleatoria con media  $\mu_i = \hat{x}_i$  y varianza  $\sigma_i^2 = p_{ii}$ . En la siguiente figura se muestra cómo cambian en el tiempo los intervalos de confianza  $\mu \pm \sigma$  de las estimaciones del ejemplo anterior.



A la larga, conforme  $k \rightarrow \infty$ , la ganancia de Kalman ( $\mathbf{K}_k$ ) y la covarianza del error ( $\mathbf{P}_k$ ) tienden a alcanzar valores finales constantes:

$$\mathbf{K}_{\infty} = \lim_{k \rightarrow \infty} \mathbf{K}_k$$

$$\mathbf{P}_{\infty} = \lim_{k \rightarrow \infty} \mathbf{P}_k$$

✦ En MATLAB, los valores finales  $\mathbf{K}_{\infty}$  y  $\mathbf{P}_{\infty}$  se pueden calcular con las funciones `kalman()` y `dlqe()`.

# Filtro de Kalman en MATLAB

Modelo MATLAB para un proceso estocástico lineal:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{G}\mathbf{w}_k$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{D}\mathbf{u}_k + \mathbf{H}\mathbf{w}_k + \mathbf{v}_k$$

# Filtro de Kalman en MATLAB

Modelo MATLAB para un proceso estocástico lineal:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{G}\mathbf{w}_k \\ \mathbf{y}_k &= \mathbf{C}\mathbf{x}_k + \mathbf{D}\mathbf{u}_k + \mathbf{H}\mathbf{w}_k + \mathbf{v}_k \end{aligned} \quad \Rightarrow \quad \begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k + [\mathbf{B} \ \mathbf{G}] [\mathbf{u}_k \ \mathbf{w}_k]^T \\ \mathbf{y}_k &= \mathbf{C}\mathbf{x}_k + [\mathbf{D} \ \mathbf{H}] [\mathbf{u}_k \ \mathbf{w}_k]^T + \mathbf{v}_k \end{aligned}$$

# Filtro de Kalman en MATLAB

Modelo MATLAB para un proceso estocástico lineal:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{G}\mathbf{w}_k \\ \mathbf{y}_k &= \mathbf{C}\mathbf{x}_k + \mathbf{D}\mathbf{u}_k + \mathbf{H}\mathbf{w}_k + \mathbf{v}_k \end{aligned} \quad \Rightarrow \quad \begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k + [\mathbf{B} \ \mathbf{G}] [\mathbf{u}_k \ \mathbf{w}_k]^\top \\ \mathbf{y}_k &= \mathbf{C}\mathbf{x}_k + [\mathbf{D} \ \mathbf{H}] [\mathbf{u}_k \ \mathbf{w}_k]^\top + \mathbf{v}_k \end{aligned}$$

El modelo de estados se representa usando `ss()` con matrices aumentadas:  
 $\mathbf{A}_s = \mathbf{A}$ ,  $\mathbf{B}_s = [\mathbf{B}, \mathbf{G}]$ ,  $\mathbf{C}_s = \mathbf{C}$ ,  $\mathbf{D}_s = [\mathbf{D}, \mathbf{H}]$ .

# Filtro de Kalman en MATLAB

Modelo MATLAB para un proceso estocástico lineal:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{G}\mathbf{w}_k \\ \mathbf{y}_k &= \mathbf{C}\mathbf{x}_k + \mathbf{D}\mathbf{u}_k + \mathbf{H}\mathbf{w}_k + \mathbf{v}_k \end{aligned} \quad \Rightarrow \quad \begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k + [\mathbf{B} \ \mathbf{G}] [\mathbf{u}_k \ \mathbf{w}_k]^T \\ \mathbf{y}_k &= \mathbf{C}\mathbf{x}_k + [\mathbf{D} \ \mathbf{H}] [\mathbf{u}_k \ \mathbf{w}_k]^T + \mathbf{v}_k \end{aligned}$$

El modelo de estados se representa usando `ss()` con matrices aumentadas:  
 $\mathbf{A}_s = \mathbf{A}$ ,  $\mathbf{B}_s = [\mathbf{B}, \mathbf{G}]$ ,  $\mathbf{C}_s = \mathbf{C}$ ,  $\mathbf{D}_s = [\mathbf{D}, \mathbf{H}]$ .

## ➤ MATLAB

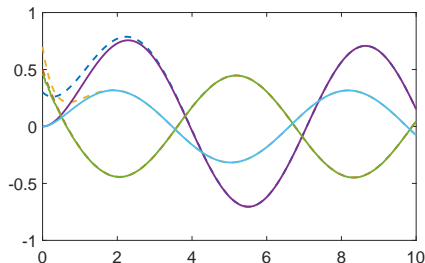
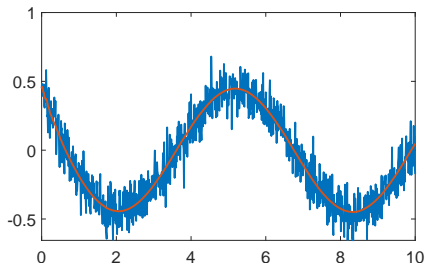
```
A = diag([-1,-2,-3]); B = [1;-1;1]; C = [0,1,0]; D = 0;
continuo = ss(A,B,C,D); Ts = 0.01;
discreto = c2d(continuo,Ts);
A = discreto.a; B = discreto.b; C = discreto.c;
G = eye(3);
H = zeros(1,3);
planta = ss(A,[B,G],C,[D,H],Ts);
Q = 1e-5*eye(3); R = 1e-2;
[estimador,K,P] = kalman(planta,Q,R)
```

# Filtro de Kalman en MATLAB

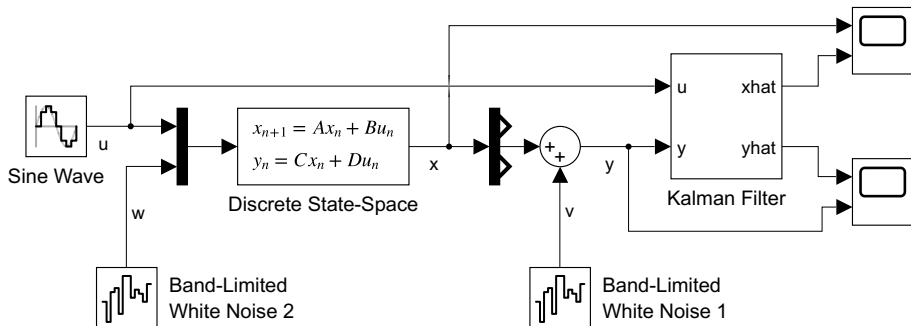
➤ MATLAB

(continuación)

```
t = 0:Ts:10; u = sin(t);  
[y,t,x] = lsim(discreto,u,t,[0.3;0.5;0.7]);  
y = y + 0.1*randn(size(y));  
[yest,t] = lsim(estimador,[u;y'],t);  
yhat = yest(:,1); xhat = yest(:,2:end);  
subplot(121); plot(t,y,t,yhat,'LineWidth',1)  
subplot(122); plot(t,x,'--',t,xhat,'LineWidth',1)
```



# Filtro de Kalman en Simulink



$$\mathbf{A} = \text{diag}([-1, -2, -3]), \quad \mathbf{B} = [1; -1; 1], \quad \mathbf{C} = [0, 1, 0], \quad T_s = 10\text{ms}$$

Sine Wave (Amplitude = 5, Frequency = 1)

$$\text{Noise power 1} = [1; 1; 1] \times 10^{-7}, \quad \text{Noise power 2} = [1 \times 10^{-4}]$$

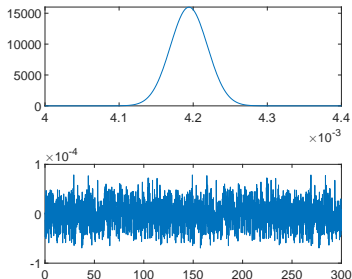
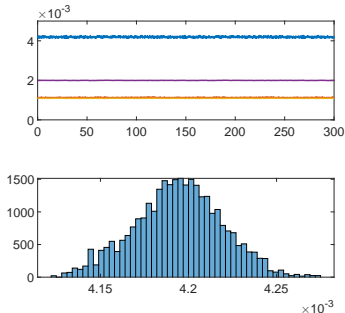
$$\text{Noise power} = \sigma^2 T_s$$



# Covarianza del ruido

## MATLAB

```
load leakFree % Get it from GitHub
subplot(221); plot(t,[FT1,FT2,FT3,FT4])
p = fitdist(FT1,'normal'); % [mean(FT1),var(FT1)]
subplot(222); fplot(@(x) p.pdf(x),[4e-3,4.4e-3])
subplot(223); histogram(FT1)
subplot(224); plot(t,FT1-mean(FT1))
cov([FT1,FT2,FT3,FT4])
```



¿Podemos filtrar una señal si no tenemos un modelo dinámico del sistema?

¿Podemos filtrar una señal si no tenemos un modelo dinámico del sistema?

Considerar el siguiente modelo dinámico:

$$x_{k+1} = x_k + w_k$$

$$y_k = x_k + v_k$$

En el filtro de Kalman, tomar  $\mathbf{A} = 1$ ,  $\mathbf{B} = 0$ ,  $\mathbf{C} = 1$ . Sintonizar empíricamente los valores de  $\mathbf{Q}$  y  $\mathbf{R}$ .

# Filtrando señales sin modelo

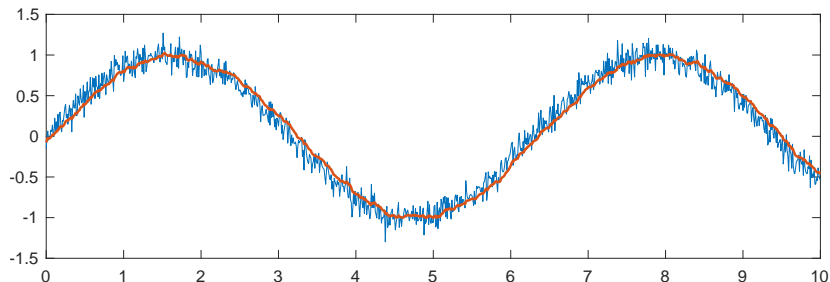
## MATLAB

```
t = 0:0.01:10;  
y = sin(t) + 0.1*randn(size(t));  
kf = KalmanFilter(1,0,1,0.01,1);  
yhat = kf.estimate(zeros(size(t)),y);  
p = plot(t,y,t,yhat);
```

# Filtrando señales sin modelo

## MATLAB

```
t = 0:0.01:10;  
y = sin(t) + 0.1*randn(size(t));  
kf = KalmanFilter(1,0,1,0.01,1);  
yhat = kf.estimate(zeros(size(t)),y);  
p = plot(t,y,t,yhat);
```



# Filtrado de señales sin modelo

## Algoritmo derivado del filtro de Kalman

1. Definir las varianzas:  $Q$  y  $R$
2. Ingresar estimaciones iniciales:  $x_0^-$  y  $P_0^-$
3. En cada paso de tiempo discreto  $k = 0, 1, 2, \dots$ 
  - 3.a. Obtener la medición  $y_k$
  - 3.b. Hacer  $\hat{x}_k = (R \hat{x}_k^- + P_k^- y_k) / (P_k^- + R)$
  - 3.c. Hacer  $P_k = P_k^- R / (P_k^- + R)$
  - 3.d. Hacer  $\hat{x}_{k+1}^- = \hat{x}_k$
  - 3.e. Hacer  $P_{k+1}^- = P_k + Q$

# Filtrado de señales sin modelo

## Implementación sin KalmanFilter

🚩 filtrar.m

```
function [xhat,Pnew] = filtrar(y,x,P,Q,R)
xhat = (R*x + P*y)/(R + P);
Pnew = P*R/(P + R) + Q;
```

🚩 MATLAB

```
t = 0:0.01:10;
y = sin(t) + 0.1*randn(size(t));
yhat = zeros(size(y));

x = 0; P = 0;
for k = 1:numel(t)
    [x,P] = filtrar(y(k),x,P,1e-4,1e-2);
    yhat(k) = x;
end

plot(t,y,t,yhat)
```

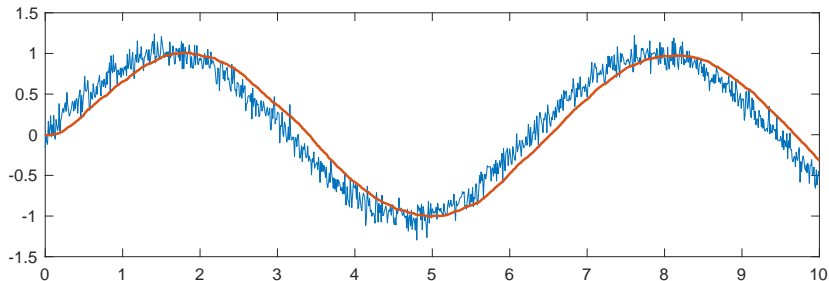
# No matar moscas a escopetazos

Filtro FIR de media móvil:

$$y_n = \frac{x_n + x_{n-1} + x_{n-2} + \cdots + x_{n-49}}{50}$$

## MATLAB

```
t = 0:0.01:10;  
y = sin(t) + 0.1*randn(size(t));  
yhat = filter(ones(50,1),50,y);  
plot(t,y,t,yhat)
```



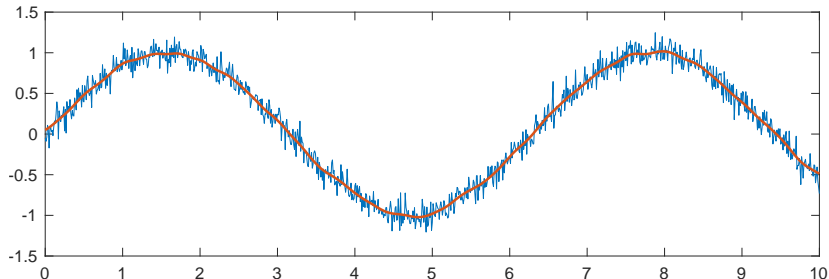


# No matar moscas a escopetazos

Suavizado gaussiano: Filtro de media móvil con ponderaciones gaussianas.

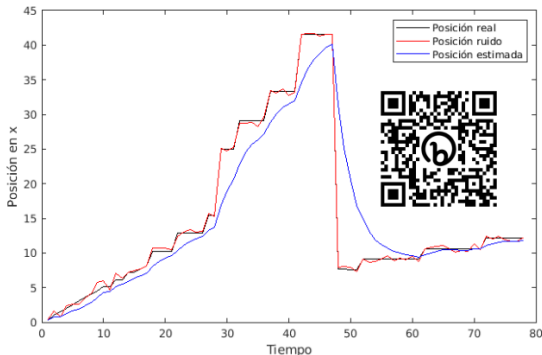
## MATLAB

```
t = 0:0.01:10;  
y = sin(t) + 0.1*randn(size(t));  
yhat = smoothdata(y, 'gaussian', 50);  
plot(t, y, t, yhat)
```



## Tarea 1: Estimar la posición del mouse

**Filtrado en línea:** Considere el ejemplo de medición en línea de posición en  $x$  del mouse sobre una figura en Matlab. A esta medición se la ha adicionado ruido gaussiano, el ejercicio consiste en determinar los valores apropiados de las matrices de covarianza tal que se logre la medición de la posición con un error mínimo:

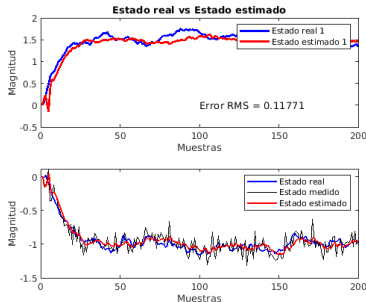


## Tarea 2: Estimar un estado

**Estimación de estados:** Considere un sistema lineal invariante en el tiempo continuo cuyas matrices de estado son:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

Estimar el estado no medido. Considerar ruido con varianza de 0.01 y media 0.



[illegible]