



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

- Resumen sobre EKF
- Estimación de estados con MATLAB
- Estimación de estados con Simulink
- Estimación de parámetros

Documentos y código fuente:

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

# Linealización de modelos no lineales

El filtro de Kalman está diseñado para modelos dinámicos **lineales**.

Es posible linealizar un modelo dinámico no lineal en un punto de operación  $\mathbf{x} = \mathbf{x}_0$ ,  $\mathbf{u} = \mathbf{u}_0$ :

$$\mathbf{x}' = \mathbf{f}(\mathbf{x}, \mathbf{u}) \quad \Rightarrow \quad \tilde{\mathbf{x}}' = \mathbf{A}\tilde{\mathbf{x}} + \mathbf{B}\tilde{\mathbf{u}}$$

donde  $\tilde{\mathbf{x}} = \mathbf{x} - \mathbf{x}_0$ ,  $\tilde{\mathbf{u}} = \mathbf{u} - \mathbf{u}_0$ . Por lo general, se omite la tilde  $\sim$  en la notación del modelo linealizado.

Las matrices  $\mathbf{A}$  y  $\mathbf{B}$  son las jacobianas de  $\mathbf{f}$ :

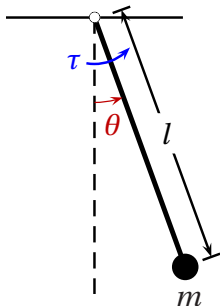
$$\mathbf{A} = \left. \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right|_{\mathbf{x}=\mathbf{x}_0, \mathbf{u}=\mathbf{u}_0} \quad \mathbf{B} = \left. \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \right|_{\mathbf{x}=\mathbf{x}_0, \mathbf{u}=\mathbf{u}_0}$$

## Ejercicio

Hallar un modelo lineal del péndulo, tomando como punto de operación la condición de reposo donde el péndulo cuelga libremente sin torque.

# El péndulo

Modelo determinista, sin fricción



$$\tau - mgl \sin(\theta) = ml^2 \ddot{\theta}$$

$$\ddot{\theta} = 1/(ml^2) \tau - (g/l) \sin(\theta)$$

Tomando  $\omega = \dot{\theta}$ , se reescribe como:

$$\dot{\theta} = \omega$$

$$\dot{\omega} = 1/(ml^2) \tau - (g/l) \sin(\theta)$$

Con  $\mathbf{x} = [\theta, \omega]^\top$  y  $u = \tau$ , se obtiene  $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, u)$ :

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ 1/(ml^2) u - (g/l) \sin(x_1) \end{bmatrix}$$

# El péndulo

## Cálculo de las matrices jacobianas

### MATLAB

```
% Definición de variables y parámetros
syms theta omega tau m l g
x = [theta;omega];
u = [tau];
% Dinámica del sistema no lineal
xdot = [omega;1/(m*l^2)*tau-g/l*sin(theta)];
% Cálculo de las matrices jacobianas
A = sym(zeros(numel(x),numel(x)));
B = sym(zeros(numel(x),numel(u)));
for i = 1:numel(x)
    for j = 1:numel(x)
        A(i,j) = diff(xdot(i),x(j));
    end
end
end
```

# El péndulo

## Cálculo de las matrices jacobianas

🚀 MATLAB

(continuación)

```
for i = 1:numel(x)
    for j = 1:numel(u)
        B(i,j) = diff(xdot(i),u(j));
    end
end
% Evaluación de las jacobianas en el punto de operación
x0 = [0;0];
u0 = [0];
A0 = subs(A,[x;u],[x0;u0])
B0 = subs(B,[x;u],[x0;u0])
```

- 🚀 El código para calcular simbólicamente las matrices jacobianas está disponible como subrutina en el repositorio de MathWorks:

<https://mathworks.com/matlabcentral/fileexchange/71075-jacobians>

Tiempo continuo

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t))$$

$$\mathbf{y}(t) = g(\mathbf{x}(t), \mathbf{u}(t))$$

# Discretización de modelos no lineales

## Tiempo continuo

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t))$$

$$\mathbf{y}(t) = g(\mathbf{x}(t), \mathbf{u}(t))$$

## Tiempo discreto

$$\mathbf{x}_{k+1} = \phi(\mathbf{x}_k, \mathbf{u}_k)$$

$$\mathbf{y}_k = g(\mathbf{x}_k, \mathbf{u}_k)$$



# Discretización de modelos no lineales

## Tiempo continuo

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t))$$

$$\mathbf{y}(t) = g(\mathbf{x}(t), \mathbf{u}(t))$$

## Tiempo discreto

$$\mathbf{x}_{k+1} = \phi(\mathbf{x}_k, \mathbf{u}_k)$$

$$\mathbf{y}_k = g(\mathbf{x}_k, \mathbf{u}_k)$$

$$\dot{\mathbf{x}}(t) \approx \frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{T_s}$$



$$\mathbf{x}_{k+1} = \mathbf{x}_k + T_s \dot{\mathbf{x}}(t)$$

# Discretización de modelos no lineales

## Tiempo continuo

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t))$$

$$\mathbf{y}(t) = g(\mathbf{x}(t), \mathbf{u}(t))$$

## Tiempo discreto

$$\mathbf{x}_{k+1} = \phi(\mathbf{x}_k, \mathbf{u}_k)$$

$$\mathbf{y}_k = g(\mathbf{x}_k, \mathbf{u}_k)$$

$$\dot{\mathbf{x}}(t) \approx \frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{T_s} \quad \Rightarrow \quad \mathbf{x}_{k+1} = \mathbf{x}_k + T_s \dot{\mathbf{x}}(t)$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + T_s f(\mathbf{x}_k, \mathbf{u}_k)$$

Forward Euler

$$\mathbf{x}_{k+1} = \mathbf{x}_k + T_s f(\mathbf{x}_{k+1}, \mathbf{u}_{k+1})$$

Backward Euler

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{T_s}{2} (f(\mathbf{x}_k, \mathbf{u}_k) + f(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}))$$

Trapezoidal

# Filtro de Kalman extendido (EKF)

El EKF una extensión del Filtro de Kalman para sistemas no lineales.

**Linealiza el sistema en cada nueva estimación del estado** usando las matrices jacobianas. Si no se tiene una expresión analítica de las jacobianas, se pueden calcular numéricamente.

- 👉 Para sistemas altamente no lineales, se recomienda usar el **Unscented Kalman Filter** (UKF). Si la distribución del ruido no es gaussiana, se recomienda usar un **filtro de partículas**.

Requisitos del EKF en tiempo discreto:

- Función de transición de estados:  $\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k)$
- Función de medición:  $\mathbf{y}_k = \mathbf{g}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k)$
- Covarianza del ruido del proceso:  $\mathbf{Q}$
- Covarianza del ruido de medición:  $\mathbf{R}$
- Estimación inicial:  $\hat{\mathbf{x}}_0^-, \mathbf{P}_0^-$

# EKF en MATLAB

## Estimación de estados del péndulo

El Control Systems Toolbox de MATLAB incluye la clase `extendedKalmanFilter` con los métodos `correct()` y `predict()`.

### MATLAB

```
% Parámetros del sistema
m = 1; l = 0.5; g = 9.8; Ts = 0.01;
% Función de transición en tiempo continuo
f = @(x,u) [x(2);1/(m*l^2)*u-g/l*sin(x(1))];
% Función de transición en tiempo discreto (forward Euler)
funTran = @(x,u) x+Ts*f(x,u);
% Función de medición
funMeas = @(x,u) x(1);
% Configuración del filtro de Kalman extendido
ekf = extendedKalmanFilter(funTran,funMeas);
ekf.ProcessNoise = diag([0,1e-4]);
ekf.MeasurementNoise = 1e-2;
ekf.State = [0;0];
```

# EKF en MATLAB

## Estimación de estados del péndulo

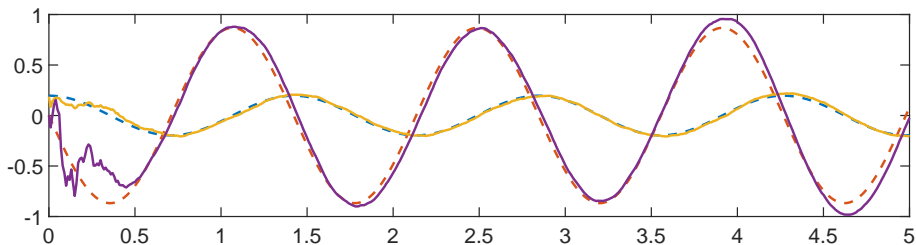
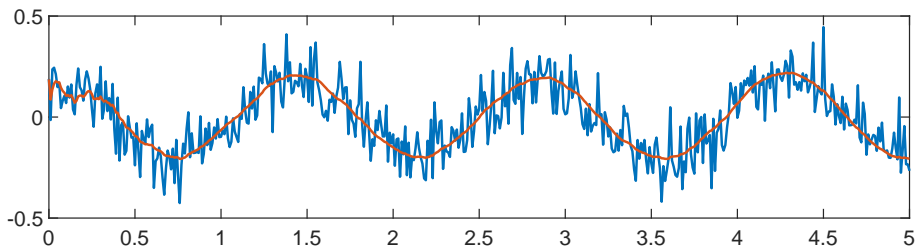
### 🚩 MATLAB

(continuación)

```
% Simulación del péndulo sin torque de entrada
[t,x] = ode45(@(t,x) f(x,0),0:Ts:5,[pi/16;0]);
y = x(:,1) + 0.1*randn(size(x(:,1)));
u = zeros(size(t));
% Estimación de estados
xhat = zeros(2,length(t));
for k = 1:length(t)
    ekf.correct(y(k),u(k));
    xhat(:,k) = ekf.State;
    ekf.predict(u(k));
end
subplot(211); plot(t,y,t,xhat(1,:), 'LineWidth',1)
subplot(212); plot(t,x, '--',t,xhat, 'LineWidth',1);
```

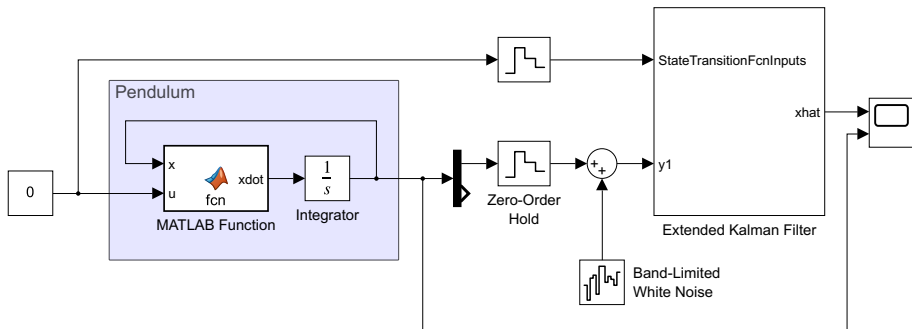
# EKF en MATLAB

## Estimación de estados del péndulo



# EKF en Simulink

## Estimación de estados del péndulo



### funTran.m

```
function xnext = funTran(x,u)
m = 1; l = 0.5; g = 9.8; Ts = 0.01;
xdot = [x(2); 1/(m*l^2)*u - g/l*sin(x(1))];
xnext = x + Ts*xdot;
```

### funMeas.m

```
function y = funMeas(x)
y = x(1);
```

**¿Qué hacer si no conocemos algún parámetro del sistema?**

Por ejemplo, supongamos desconocida la aceleración de la gravedad en el modelo del péndulo.



## ¿Qué hacer si no conocemos algún parámetro del sistema?

Por ejemplo, supongamos desconocida la aceleración de la gravedad en el modelo del péndulo.

Podemos considerar al parámetro desconocido como una variable de estado adicional  $x_3 := g$  y obtener un **modelo aumentado** agregando una ecuación diferencial con la derivada del parámetro igual a cero:

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = u / (ml^2) - (x_3/l) \sin(x_1)$$

$$\dot{x}_3 = 0$$

El parámetro se obtendrá de la estimación de  $x_3$ . Se debe asignar a  $x_3$  un valor inicial “razonable” distinto de cero. Por el transitorio en la respuesta inicial del estimador, el valor del parámetro se obtendrá promediando sólo los valores finales de  $x_3$ .

# Estimación de parámetros con EKF

## Cálculo de la aceleración de la gravedad

### MATLAB

```
% Parámetros del sistema
m = 1; l = 0.5; g = 9.8; Ts = 0.01;
% Función de transición en tiempo continuo
f = @(x,u) [x(2);1/(m*l^2)*u-g/l*sin(x(1))];
% Función de transición aumentada
f_aum = @(x,u) [x(2);1/(m*l^2)*u-x(3)/l*sin(x(1));0];
% Función de transición en tiempo discreto (forward Euler)
funTran = @(x,u) x+Ts*f_aum(x,u);
% Función de medición
funMeas = @(x,u) x(1);
% Configuración del filtro de Kalman extendido
ekf = extendedKalmanFilter(funTran,funMeas);
ekf.ProcessNoise = diag([0,1e-5,1e-5]);
ekf.MeasurementNoise = 1e-4;
ekf.State = [0;0;8];
```

# Estimación de parámetros con EKF

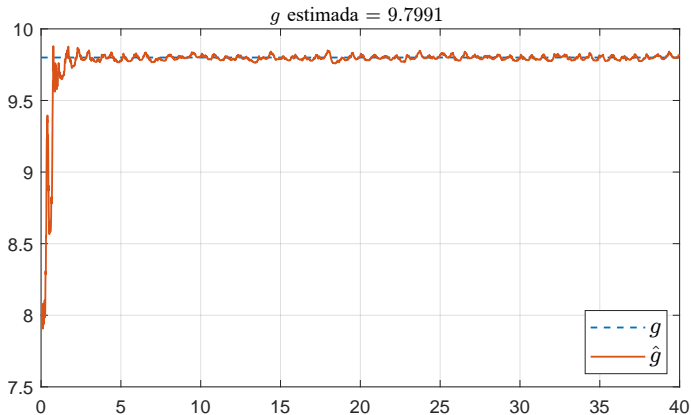
## Cálculo de la aceleración de la gravedad

### MATLAB

```
% Simulación del péndulo sin torque de entrada
t = 0:Ts:40;
[t,x] = ode45(@(t,x) f(x,0),t,[pi/16;0]);
y = x(:,1) + 0.01*randn(size(x(:,1)));
u = zeros(size(t));
% Estimación de estados y parámetro
xhat = zeros(3,length(t));
for k = 1:length(t)
    ekf.correct(y(k),u(k));
    xhat(:,k) = ekf.State;
    ekf.predict(u(k));
end
plot(t,g*ones(size(t)),'--',t,xhat(3,:))
gest = mean(xhat(3,2000:end));
title("g estimada = " + gest)
```

# Estimación de parámetros con EKF

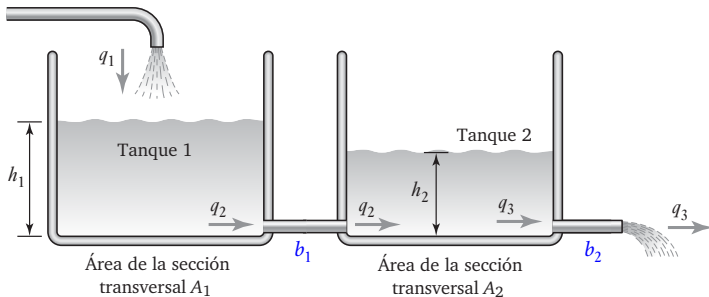
## Cálculo de la aceleración de la gravedad



# Estimación de parámetros con EKF

## Cálculo de parámetros hidráulicos

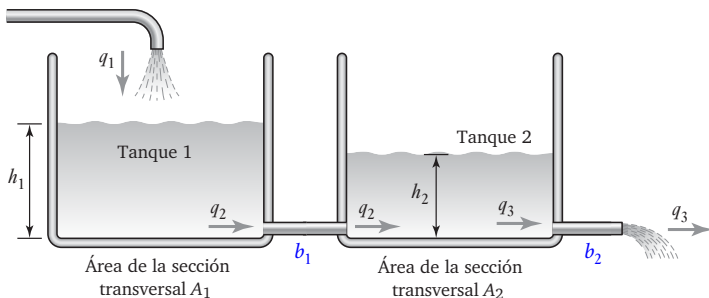
**Propósito:** Estimar los coeficientes  $b_1$  y  $b_2$  a partir de mediciones de las alturas  $h_1$  y  $h_2$  y del caudal  $q_2$  que fluye entre los dos tanques.



# Estimación de parámetros con EKF

## Cálculo de parámetros hidráulicos

**Propósito:** Estimar los coeficientes  $b_1$  y  $b_2$  a partir de mediciones de las alturas  $h_1$  y  $h_2$  y del caudal  $q_2$  que fluye entre los dos tanques.



$$q(b, h_1, h_2) = \cancel{b\sqrt{h_1 - h_2}} = b \operatorname{sign}(h_1 - h_2) \sqrt{|h_1 - h_2|}$$

# Estimación de parámetros con EKF

## Cálculo de parámetros hidráulicos

Ecuaciones del balance de masa:

$$A_1 \dot{h}_1 = q_1 - q_2$$

$$A_2 \dot{h}_2 = q_2 - q_3$$

# Estimación de parámetros con EKF

## Cálculo de parámetros hidráulicos

Ecuaciones del balance de masa:

$$A_1 \dot{h}_1 = q_1 - q_2$$

$$A_2 \dot{h}_2 = q_2 - q_3$$

Definiendo  $x_1 := h_1$ ,  $x_2 := h_2$ ,  $u := q_1$ , y despejando las derivadas, se obtiene el modelo dinámico no lineal:

$$\dot{x}_1 = (u - q(b_1, x_1, x_2)) / A_1$$

**Estado 1**

$$\dot{x}_2 = (q(b_1, x_1, x_2) - q(b_2, x_2, 0)) / A_2$$

**Estado 2**

$$y_1 = x_1$$

**Medición 1**

$$y_2 = x_2$$

**Medición 2**

$$y_3 = q(b_1, x_1, x_2)$$

**Medición 3**



# Estimación de parámetros con EKF

## Cálculo de parámetros hidráulicos

Ecuaciones del balance de masa:

$$A_1 \dot{h}_1 = q_1 - q_2$$

$$A_2 \dot{h}_2 = q_2 - q_3$$

Definiendo  $x_1 := h_1$ ,  $x_2 := h_2$ ,  $u := q_1$ , y despejando las derivadas, se obtiene el modelo dinámico no lineal:

$$\dot{x}_1 = (u - q(b_1, x_1, x_2)) / A_1 \quad \text{Estado 1}$$

$$\dot{x}_2 = (q(b_1, x_1, x_2) - q(b_2, x_2, 0)) / A_2 \quad \text{Estado 2}$$

$$y_1 = x_1 \quad \text{Medición 1}$$

$$y_2 = x_2 \quad \text{Medición 2}$$

$$y_3 = q(b_1, x_1, x_2) \quad \text{Medición 3}$$

¡Tanto  $\dot{x}$  como  $y$  son funciones no lineales del estado! 😊 😞

# Estimación de parámetros con EKF

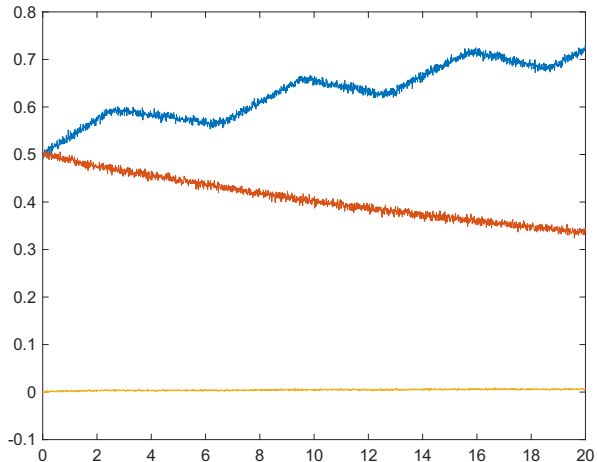
## Cálculo de parámetros hidráulicos

### MATLAB

```
% Parámetros del sistema
A1 = 0.5; A2 = 1; b1 = 0.01; b2 = 0.02; Ts = 0.01;
% Relación de las alturas de presión con los caudales (Bernoulli)
q = @(b,h1,h2) b*sign(h1-h2).*sqrt(abs(h1-h2));
% Suponemos un patrón de flujo de entrada
u = @(t) 1e-2*(1+square(t));
% Obtenemos las "mediciones" con la dinámica determinista...
xprima = @(t,x) [1/A1*u(t)-1/A1*q(b1,x(1),x(2));
    1/A2*q(b1,x(1),x(2))-1/A2*q(b2,x(2),0)];
[t,x] = ode45(xprima,0:Ts:20,[0.5;0.5]);
y = [x,q(b1,x(:,1),x(:,2))];
% ...pero agregamos ruido en los sensores
y(:,1) = y(:,1) + 0.005*randn(size(y(:,1)));
y(:,2) = y(:,2) + 0.005*randn(size(y(:,2)));
y(:,3) = y(:,3) + 0.002*randn(size(y(:,3)));
figure(1); plot(t,y)
```

# Estimación de parámetros con EKF

## Cálculo de parámetros hidráulicos



# Estimación de parámetros con EKF

## Cálculo de parámetros hidráulicos

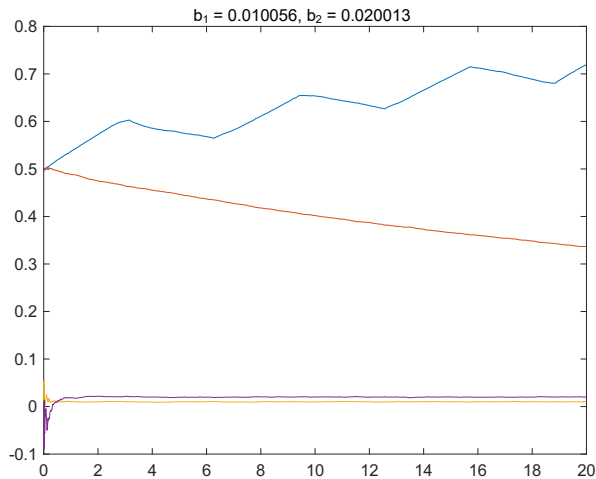
### 🔥 MATLAB

(continuación)

```
% Dinámica del modelo aumentado con  $x_3 \leftarrow b_1$ ,  $x_4 \leftarrow b_2$ 
f = @(x,u) [1/A1*u-1/A1*q(x(3),x(1),x(2));
            1/A2*q(x(3),x(1),x(2))-1/A2*q(x(4),x(2),0);0;0];
funTran = @(x,u) x + Ts*f(x,u); % Discretización forward Euler
funMeas = @(x,u) [x(1);x(2);q(x(3),x(1),x(2))];
ekf = extendedKalmanFilter(funTran,funMeas,[0.5;0.5;0.01;0.01]);
ekf.ProcessNoise = 1e-8;
ekf.MeasurementNoise = diag([0.005,0.005,0.002].^2);
uu = u(t);
xhat = zeros(4,numel(t));
for k = 1:numel(t)
    ekf.correct(y(k,:)',uu(k));
    xhat(:,k) = ekf.State;
    ekf.predict(uu(k));
end
figure(2); plot(t,xhat)
blest = mean(xhat(3,1000:end)); b2est = mean(xhat(4,1000:end));
title("b1 = " + blest + ", b2 = " + b2est)
```

# Estimación de parámetros con EKF

## Cálculo de parámetros hidráulicos





<https://www.facebook.com/GTurix/>