FACULTAD DE INGENIERIA
Universidad de Buenos Aires

(TA135) Taller de Automatización y Control
1er Cuatrimestre de 2024

Actividad 6

Facundo Arballo	– Padrón 105096 –	farballo@fi.uba.ar
Francisco Spaltro	– Padrón 102098 –	fspaltro@fi.uba.ar

Resumen

Implementar un observador de estados para el ángulo y velocidad angular del péndulo, midiendo únicamente el ángulo, y un observador para el sesgo de velocidad angular

1. Observador midiendo el ángulo

Sea

$$x = \begin{pmatrix} \theta & \omega \end{pmatrix}^T \qquad\qquad y = \theta$$

Luego:

$$\hat{x}_{k+1} = A_d \hat{x}_k + L(\theta_k - C_d \hat{x}_k)$$

Con lo cual

$$\begin{pmatrix} \hat{\theta}_{k+1} \\ \hat{\omega}_{k+1} \end{pmatrix} = A_d \begin{pmatrix} \hat{\theta}_k \\ \hat{\omega}_k \end{pmatrix} + L(\theta_k - C_d \begin{pmatrix} \hat{\theta}_k \\ \hat{\omega}_k \end{pmatrix})$$

Planteando las matrices genéricas

$$A_d = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

$$L = \begin{pmatrix} l_1 \\ l_2 \end{pmatrix}$$

y reemplazando, se obtiene:

$$\begin{pmatrix} \hat{\theta}_{k+1} \\ \hat{\omega}_{k+1} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} \hat{\theta}_k \\ \hat{\omega}_k \end{pmatrix} + \begin{pmatrix} l_1 \\ l_2 \end{pmatrix} (\theta_k - \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} \hat{\theta}_{k+1} \\ \hat{\omega}_{k+1} \end{pmatrix}$$

donde

$$C_d = \begin{pmatrix} 1 & 0 \end{pmatrix}$$

Se obtiene, entonces:

$$\hat{\theta}_{k+1} = a_{11}\hat{\theta}_k + a_{12}\hat{\omega}_k + l_1(\theta_k - \hat{\theta}_k) \tag{1.1}$$

A partir de Matlab

```
l = 0.16;
r2 = 2.625;
g = 9.8;
T = 0.01;

Ad = eye(2) + [0 1; -g/l -r2] * T;
Cd = [1 0];
L = place(Ad', Cd', [0.5 0.6]);
```

Con lo cual

$$A_d = \begin{pmatrix} 1 & 0{,}01 \\ -0{,}6125 & 0{,}9738 \end{pmatrix}$$

$$L = \begin{pmatrix} 0{,}8738 \\ 17{,}0939 \end{pmatrix}$$

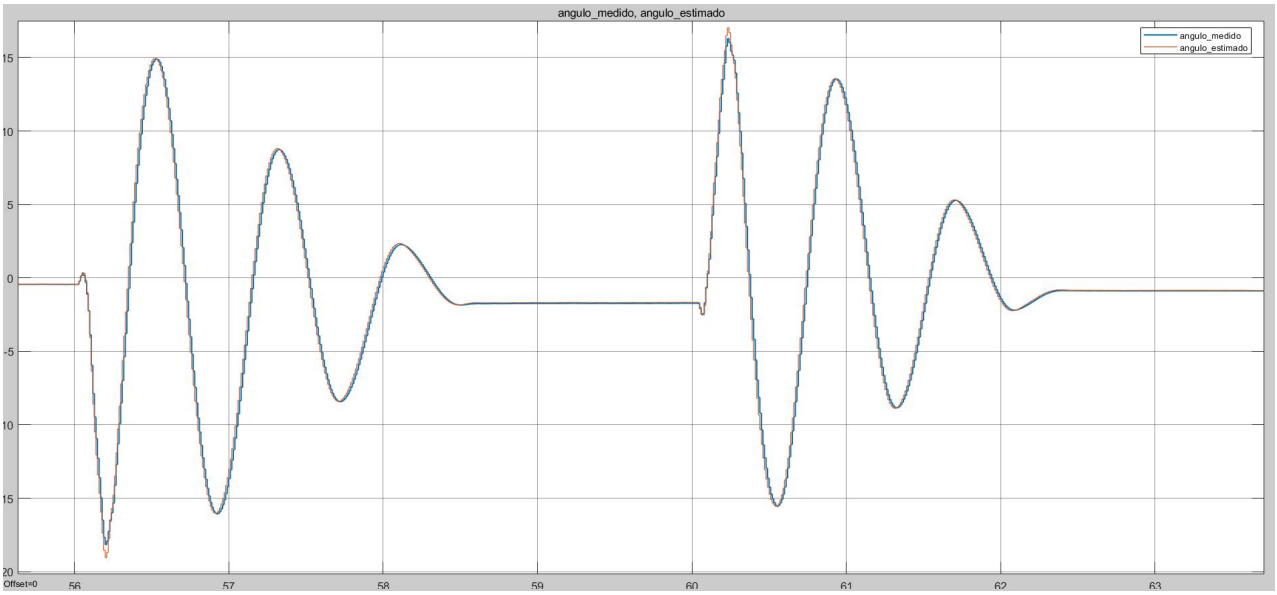

Figura 1.1: Ángulo medido y estimado

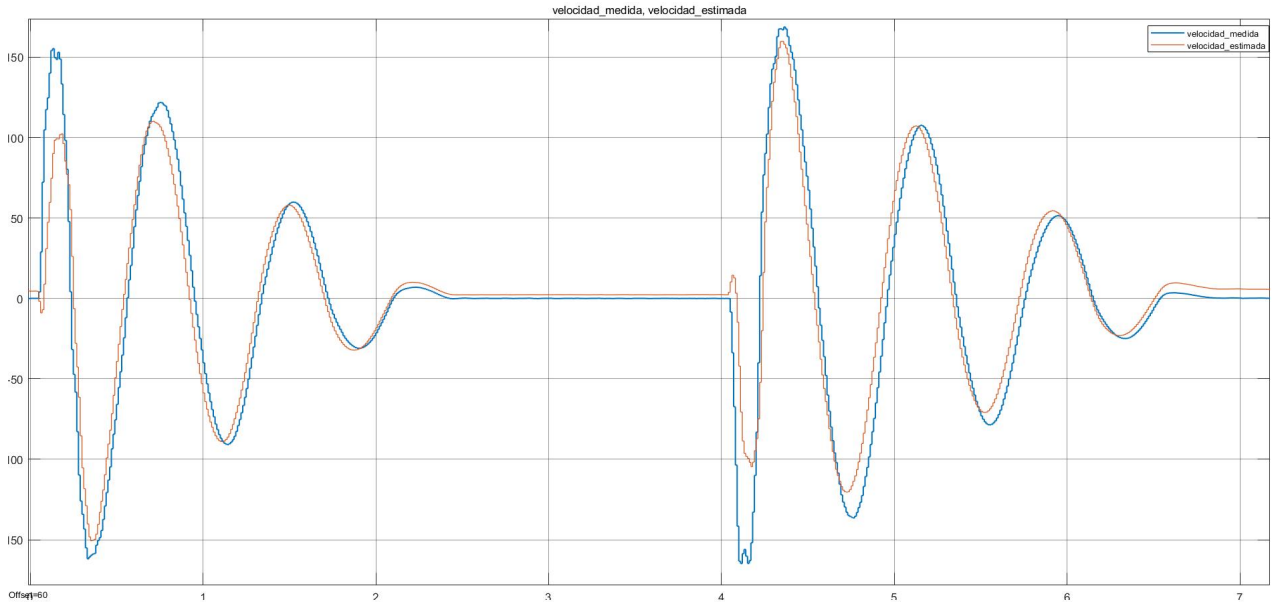

Figura 1.2: Velocidad medida y estimada

2. Sesgo

Para el sesgo, se agrego arbitrariamente una componente continua de $10\frac{\text{grados}}{s}$. Sean los vectores y matrices

$$x_k = \begin{pmatrix} \theta_k \\ \omega_k \\ b_k \end{pmatrix}$$

$$A_d = \begin{pmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$C_d = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

Con lo cual la matriz L tiene que ser de 3x2, de la forma

$$l_{11}l_{12}l_{21}l_{22}l_{31}l_{32}$$

Con lo cual:

$$\begin{pmatrix} \hat{\theta}_{k+1} \\ \hat{\omega}_{k+1} \\ \hat{b}_{k+1} \end{pmatrix} = \begin{pmatrix} a_{11}\hat{\theta}_k + a_{12}\hat{\omega}_k \\ a_{21}\hat{\theta}_k + a_{22}\hat{\omega}_k \\ \hat{b}_k \end{pmatrix} + \begin{pmatrix} l_{11} & l_{12} \\ l_{21} & l_{22} \\ l_{31} & l_{32} \end{pmatrix} \begin{pmatrix} \theta_k - \hat{\theta}_k \\ \omega_k - \hat{\omega}_k - \hat{b}_l \end{pmatrix}$$

A partir de Matlab:

```
g=9.8;
l=0.18;
k=0.004;
m=0.065;
```

```
5   T=0.02;
6
7   Ad = eye(2) + [0  1; -g/l -k/(m*l^2)] * T;
8   Ad2 = eye(3);
9   Ad2(1:2, 1:2) = Ad;
10  Cd2 = [1 0 0; 0 1 1];
11  L = place(Ad2', Cd2', [0.5  0.6  0.95])';
```

se obtiene

$$L = \begin{pmatrix} 0{,}4630 & 0{,}0340 \\ -0{,}8630 & 0{,}0082 \\ -0{,}1746 & 0{,}4409 \end{pmatrix}$$

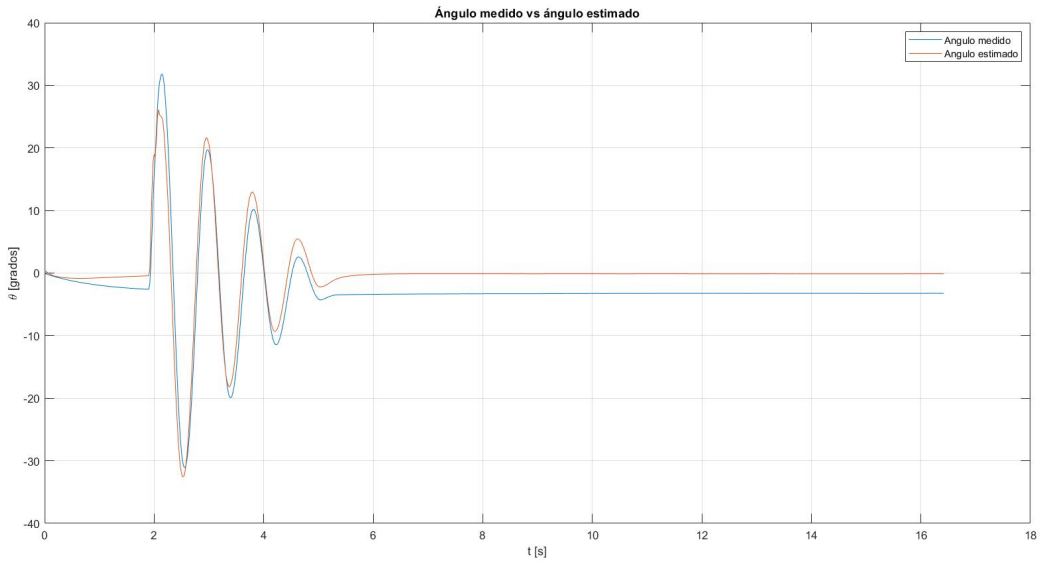

Figura 2.1: Ángulo medido y estimado

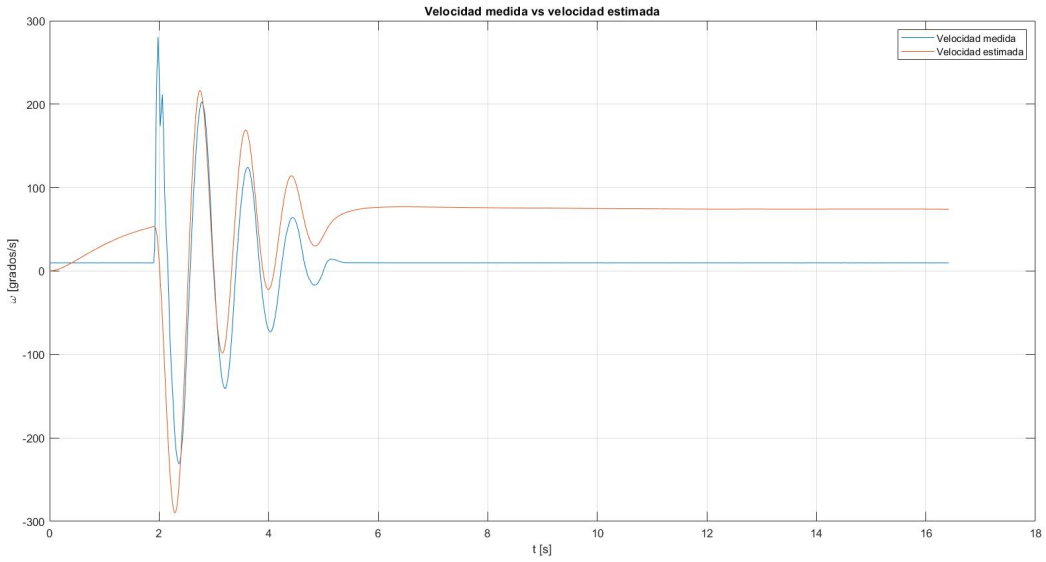

Figura 2.2: Velocidad medida y estimada

La estimación no dio como se esperaba, a pesar de los reiterados intentos por corregir el código. Creemos que puede deberse a la diferencia entre nuestra planta real con la que describe las matrices utilizadas. Este error entre las mediciones y las estimaciones puede verse en la estimación del sesgo, muy distinta a lo que debería haber dado

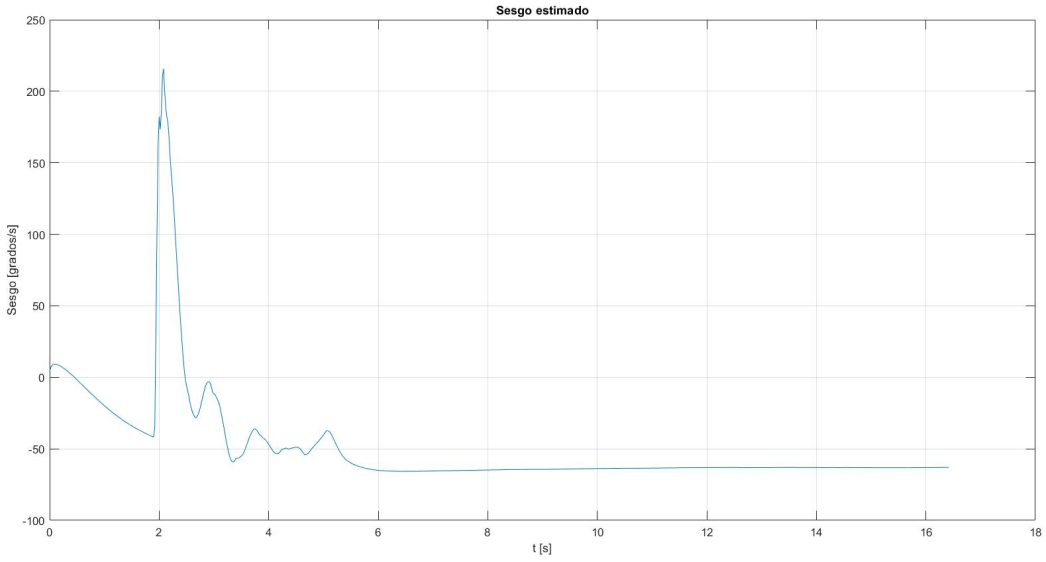

Figura 2.3: Sesgo estimado

3. Código

```
1 // La stdint.h define los ancho de datos como el uint16_t
```

```
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>
#include <Servo.h>
#include <stdint.h>

Adafruit_MPU6050 mpu;
Servo servo;

// Definimos los pines a utilizar
#define PIN_PWM 9 //OC1A
#define PIN_POTE A0
#define MINIMO_POTE 10
#define MAXIMO_POTE 712
#define MINIMO 1350
#define MAXIMO 5100
#define TOP 39999
#define FRECUENCIA_LECTURA 50

int LIMITE_ANGULO_SUPERIOR = 135;
int LIMITE_ANGULO_INFERIOR = 45;

uint16_t offset_angulo = 5;
uint16_t periodo_lectura;
uint16_t tiempo_inicial = 0;
uint16_t tiempo_final = 0;

float offset_giro_x = -0.04;
float offset_giro_y = 0.03;
float offset_giro_z = 0;

float offset_accel_x = 1.13;
float offset_accel_y = 0.39;
float offset_accel_z = -0.75;

float w_giro_x = 0;
float angulo_gir_x = 0;
float angulo_accel_x = 0;
float angulo_x = 0;
float velocidad = 0;

float alpha = 0.98;

// ACTIVIDAD 6
float theta_k = 0;
float theta_k_1 = 0;
float w_k = 0;
float w_k_1 = 0;
float b_k = 0;
float b_k_1 = 0;

float a11 = 1;
float a12 = 0.02;
float a21 = -1.0889;
float a22 = 0.9620;

```

```
58 float l11 = 0.4630;
59 float l12 = 0.0340;
60 float l21 = -0.8630;
61 float l22 = 0.0082;
62 float l31 = -0.1746;
63 float l32 = 0.4409;
64
65 void setup() {
66   Serial.begin(115200);
67
68   // Config IMU
69   // Try to initialize!
70
71   if (!mpu.begin()) {
72     Serial.println("Failed to find MPU6050 chip");
73     while (1) {
74       delay(10);
75     }
76   }
77   Serial.println("MPU6050 Found!");
78
79   mpu.setAccelerometerRange(MPU6050_RANGE_8_G); // set accelerometer range to
       +-8G
80   mpu.setGyroRange(MPU6050_RANGE_500_DEG);      // set gyro range to +- 500 deg
       /s
81   mpu.setFilterBandwidth(MPU6050_BAND_10_HZ);   // set filter bandwidth to
       5-10-21-44-94-184-260 Hz
82
83   pinMode(PIN_PWM, OUTPUT);
84   config_50_hz();
85   periodo_lectura = 1e6/FRECUENCIA_LECTURA;
86   OCR1A = 5100;
87   delay(1000);
88 }
89
90 void loop() {
91   // Utilizamos millis() en lugar de micros() porque esta  ltima  llega hasta
       65536 (2^16) y necesitar amos del orden de los 10^6 para 10 Hz
92   tiempo_inicial = micros();
93
94   obtener_angulo_giroscopo_x();
95   obtener_angulo_accel_x();
96
97   angulo_x = alpha*angulo_gir_x + (1-alpha)*angulo_accel_x;
98   velocidad = w_giro_x+10;
99
100  theta_k_1 = a11 * theta_k + a12 * w_k + l11 * (angulo_x - theta_k) + l12 * (
       velocidad - w_k - b_k);
101  w_k_1 = a21 * theta_k + a22 * w_k + l21 * (angulo_x - theta_k) + l22 * (
       velocidad - w_k - b_k);
102  b_k_1 = b_k + l31 * (angulo_x - theta_k) + l32 * (velocidad - w_k - b_k);
103
104  theta_k = theta_k_1;
105  w_k = w_k_1;
106  b_k = b_k_1;
107
```

```
108   matlab_send(angulo_x, theta_k_1, velocidad, w_k_1, b_k_1);
109
110   tiempo_final = micros();
111   delayMicroseconds(periodo_lectura - (tiempo_final - tiempo_inicial));
112 }
113
114
115 void config_50_hz(){
116   TCCR1A = (1 << COM1A1) | (1 << WGM11);
117   TCCR1B = (1 << WGM13) | (1 << WGM12) | (1 << CS11);
118   ICR1 = TOP;
119 }
120
121
122
123 void matlab_send(float dato1, float dato2, float dato3, float dato4, float
      dato5){
124   Serial.write("abcd");
125   byte * b = (byte *) &dato1;
126   Serial.write(b,4);
127   b = (byte *) &dato2;
128   Serial.write(b,4);
129   b = (byte *) &dato3;
130   Serial.write(b,4);
131   b = (byte *) &dato4;
132   Serial.write(b,4);
133   b = (byte *) &dato5;
134   Serial.write(b,4);
135 }
136
137 void obtener_angulo_giroscopo_x() {
138   sensors_event_t a, g, temp;
139   mpu.getEvent(&a, &g, &temp);
140
141   w_giro_x = (g.gyro.x-offset_giro_x) * 180 / M_PI;
142   angulo_gir_x = angulo_x + (g.gyro.x-offset_giro_x) * (periodo_lectura/1e6) *
      180 / M_PI;
143 }
144
145 void obtener_angulo_accel_x() {
146   sensors_event_t a, g, temp;
147   mpu.getEvent(&a, &g, &temp);
148
149   angulo_accel_x  = atan2(a.acceleration.y-offset_accel_y, a.acceleration.z-
      offset_accel_z) * 180 / M_PI;
150 }
```