Licenciatura en Fisica
Instituto de Fisica
Pontificia Universidad Católica de Valparaiso

1^{er} semestre 2016
Mackarena Garrido, Nicolás Sepúlveda

Equilibrista con PID

Resumen : *En el presente documento se revisarán los conceptos detrás de un controlador PID. Este consiste en mantener un valor aproximadamente constante por medio de un algoritmo que entrega un valor de salida necesario para ajustar las desviaciones, y que en principio depende de la diferencia entre el valor medido y el deseado. Este algoritmo es la suma de controladores de tipo proporcional, integral y derivativo. Se uso este modelo para construir una estructura que pueda regular su posición y mantenerse en equilibrio sobre su eje de rotación, cuyo objetivo no se cumplió debido a la estructura de éste. Utilizamos un motor SERVO, un acelerometro "Triple Axis Accelerometer V2" y un arduinoUNO.*

Palabras claves *: PID, controlador, arduino, regulador, ángulo*

Profesor guía :
German Varas
german.varas@pucv.cl / *tél. (+56) 32 227 4889*
Instituto de Fisica
Av. Universidad 330 (Curauma), Valparaiso http://fis.ucv.cl/guaras

29 de diciembre de 2018

Índice general

Capítulo 1

Introducción

Contents

1.1. Introducción

Los sistemas cuyas variables no son estacionarias o no se encuentran en un equilibrio estable, necesitian de una retroalimentacion dependiente de las variaciones con respecto al estado deseado. Un ejemplo puntual, es la regulación de la temperatura del agua en la ducha. Si esta es muy baja, se necesita realizar un aumento de agua caliente que probablemente va a exceder el valor a alcanzar, resultando en una temperatura más alta que necesitará ser reajustada: el proceso reiterado hace tender la variable a un valor fijo. Hay que notar que los cambios en los flujos de agua dependen de diferentes parámetros, como qué tan desajustada está la temperatura, o la función de respuesta de las llaves.

Ejemplos como el anterior llevan a la idea de hacer automática la acción de autoregulación de una variable, por lo que se introducen los controladores PID. En general, el proceso trata de recibir un valor asociado a alguna magnitud que se deberá mantaner constante, mediante un sensor de entrada. Luego, se obtiene la diferencia entre el dato medido y valor meta, definido como *SetPoint*, para tener una variable denominada *error*. Este valor se ocupará en 3 procesos matematicos, llamados control *propocional*, *integral* y *derivativo*, para así calcular una valor de respuesta que retroalimentará al sistema para reajustarse. Ocurrido esto, se volverá a tomar una medicion desde el sistema y se hará lo mismo iterativamente [1]. En este documento se tratarán los fundamentos teóricos básicos y se expondrá un experimento en base a un controlador PID que fue realizado con códigos que midieron la aceleración de la estructura para así obtener el ángulo de declinación, y generar respuestas en el torque una vara y entregar la reacción necesaria para el equilibrio. Por último se hablará de por qué creemos que esto no resultó de la manera esperada.

1.2. Teotría

Para equilibrar una variable específica de un sistema, se necesita medir el *error*; cantidad que indica la diferencia entre el valor en el instante de medición, el *input*, y el esperado, el *setpoint*. Dado el error, el controlador realiza en función de este un cambio en otra variable, llamado *output*, con el propósito de disminuirlo. La forma en que estos valores son medidos y modificados depende del sistema.

El algoritmo mediante el cual funciona el controlador PID consiste en una ecuación que considera la suma de tres términos: proporcional, integral y derivativo. Cada uno es resultado de un valor asociado al error, calculado de distintas maneras; que está multiplicado por una constante característica tanto del término, como del sistema. Con las variables definidas ya identificadas, la implementación del controlador en el sistema se resume a la búsqueda de las constantes específicas.

La ecuación para obtener el *output* es:

$$Output = P + I + D. \tag{1.1}$$

En las secciones siguientes se tratarán las funciones de los tres términos términos [2].

1.2.1. Proporcional

El primer término establece una relación proporcional con el error. Este es modulado por la constante k_p, que es una cuantificación de la reacción del sistema a un error específico. Mientras

mayor sea el error, más fuerte será la reacción, con el fin de minimizar esta diferencia. Esta respuesta se caracteriza por llevar al sistema a un valor *debajo* del setpoint, si k_p es bajo, o por generar una reacción excesiva si k_p es alto.
La ecuación para el término proporcional es:

$$P = k_p e, \tag{1.2}$$

con e el error.
Notar que la respuesta cuando $k_p = 1$ es igual al error.

1.2.2. Integral

El segundo término, a diferencia del proporcional, es una función del tiempo. Este realiza una integración del error en un intervalo definido, lo que implica la toma de valores anteriores al que se está procesando. La suma en el tiempo entonces reacciona a qué tan grande es el error y a su duración, teniendo una respuesta más grande cuando las diferencias son prolongadas, y por el contrario, pequeña si también lo son los tiempos o bien la variable está cerca del setpoint. Asímismo, mientras el error no sea cero, el término integral crecerá en el tiempo, por lo que demás cumple la función de estabilizar los errores añadidos por la correción del término proporcional. La ecuación para cada iteración es:

$$S_e = (S_e + e) * dt \tag{1.3}$$

en donde S_e es la suma total de los errores en el tiempo, e el error actual y dt el intervalo de tiempo. La transformación para contínuos de esta expresión es una integral. Finalmente, el término integral se define como $I = k_i * S_e$, y en este caso la constante k_i modula la rapidez de las respuestas.

1.2.3. Derivativa

El último término, el derivativo, es función del ratio de cambio del error. Su función es la estabilización del output cuando el error está cambiando muy rápido, situaciones en que la respuesta del sistema tiende a sobrereacciones. Cumple entonces un rol de amortiguador, aplicando correcciones fuertes antes de que el error crezca demasiado.

El factor asociado al error para el término derivativo es

$$D_e = (E_t - e_{t-1})/dt, \tag{1.4}$$

y en la ecuación del algoritmo:

$$D = k_d D_e. \tag{1.5}$$

Capítulo 2

Proyecto de equilibrio

Contents

2.1. Proyecto

Implementaremos el controlador PID armando un artefacto que se mantenga de pie gracias a la ayuda de una vara que rotará para regular la posición por efectos del controlador PID. Este tendrá liberta de rotar solo sobre un eje paralelo a la superficie. Para armar nuestro equilibrista con PID, utilizamos un arduinoUNO, un motor servo, un acelerómetro "Triple Axis Accelerometer V2" marca DFROBOT, varios cables, una batería de 9V y por su puesto el pc para programar el código que utilizaremos. El modelo fue armado en una estructura de cartón piedra que consiste en 2 partes, la parte superior de 11.5 cm de largo y de 8 cm de ancho, donde está ubicado el Arduino en una cara (ver Fig.2.1) y el motor Servo (este traspasa la superficie de cartón), por el reverso colocamos el acelerómetro pegado horizontalmente a la orilla superior de la caja(ver Fig.2.2). La parte inferior de la estructura se une a la superior por las orillas, terminando en una especie de punta que será el punto de apoyo y el eje de rotación, (esta parte mide 11.5 cm de largo también), pero la rotación solo será en dos dimensiones por lo tanto la punta tiene grosor en el eje z (ver Fig.2.3) Ya que no vamos a interactuar con el eje z, solo necesitamos el ángulo de rotación del eje y con respecto al eje z, para eso el acelerómetro mide las aceleraciones de estos dos ejes y después de los cálculos matemáticos mencionados anteriormente obtenemos el ángulo. Este esta conectado al arduinoUNO en la entrada GND y la entrada de 5V. Una vez tenemos el dato correspondiente al ángulo, se ingresa al código de PID(que veremos más adelante) y el PID devuelve el valor que será entregado al motor SERVO, este ultimo conectado al Arduino en tres lugares, al puerto 9, a la otra entrada GND y a la entrada VIN, al motor se le acopla una varilla horizontal que es la encargada de mantener el equilibrio del Equilibrista al rotar por acción del motor, cambiando el movimiento de toda la estructura. Y porúltimo el Arduino se conecta por entrada USB al PC.

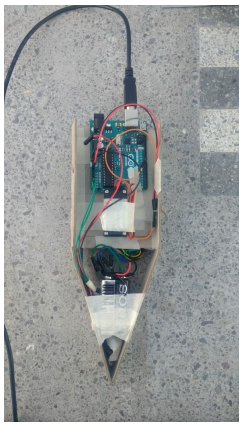

Figura 2.1: Foto de la estructura, donde se puede ver el arduinoUNO, el reverso del motor SERVO y la bateria

Figura 2.2: Foto de la estructura, donde se puede ver el acelerometro y el motor SERVO donde se colocara la barra

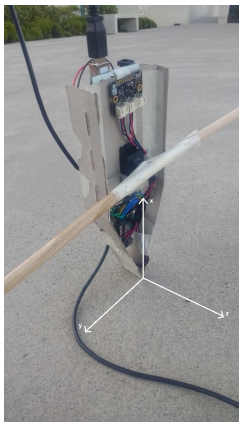

Figura 2.3: Foto de la estructura sobre el eje de rotación.

2.1.1. Codigo PID utilizado

```
# include < PID_v1.h >
double Setpoint, Input, Output;
//la siguiente linea activa el PID y sus variables
//en la siguiente linea tenemos los valores asignados a las variables de PID
//siendo Kp=4.3, Ki=0 y Kd=0.1 PID myPID(Input, Output, Setpoint,4.3,0.4,0.1, DIRECT);
#include¡math.h¿
#include¡stdio.h¿
#include ¡Servo.h¿
#define AX 5 //relaciona el dato de la aceleracion en el eje x con el puerto 5
#define AY 4 //relaciona el dato de la aceleracion en el eje x con el puerto 4
#define AZ 3 //relaciona el dato de la aceleracion en el eje x con el puerto 3
int valx,valy,valz; //definimos las variables que vamos a usar double b;
double angulo; //aqui se guarda el valor del angulo que usaremos
Servo myservo; // crea un objeto servo para controlar el servo
int val;

void setup() {
myservo.attach(9); // attaches the servo on pin 9 to the servo object
//recibimos la señal de las respectivas entradas
pinMode(AX,INPUT);
```

```
pinMode(AY,INPUT);
pinMode(AZ,INPUT);
Serial.begin(9600);
Input = angulo; //asignamos a input el valor de langulo para que lo reciba PID
Setpoint = 0; // este es el angulo que permanecera constante
myPID.SetMode(AUTOMATIC); //enciende el PID
}

void loop() {
float a;
for (int i=0;i¡10;i++) {
valx+=analogRead(AX);delay(1); //a valx le asigna lo que lee en AX y espera un segundo para
leer el siguiente dato, ya que se necesita una variacion de tiempo para obtener una aceleracion.
valy+=analogRead(AY);delay(1);
valz+=analogRead(AZ);delay(1);
}

valx=valx/10;
valy=valy/10;
valz=valz/10;
delay(195); //espera 195 milisegundos para volver a tomar datos
b=(double) ((valx-320))/((valy-320)); //se obtiene el cuociente entre las dos aceleraciones
a=atan(b); // este es el valor del angulo, una vez qe se saca el arcotangente
double angulo; //se define la variable para el angulo
angulo=a/3.14*180; //convierte el valor a angulo
Serial.print(.ª: ");
Serial.println(angulo); // imprime los valores del angulo para ir verificando
//se vuelven a cero todos los valores
valx=0;
valy=0;
valz=0;
Input = angulo;
myPID.Compute(); //se inicia el PID
Serial.print(.ºutput pid: ");
Serial.print(Output); //imprime para ver los valores que salen del PID
Output=map(Output,-90,90,69,111);
myservo.write(Output); //ingresamos al servo el valor que sale del PID para mover el motor
}
```

2.2. Resultados

Una vez armada la estructura del Equilibrista, con una barra de madera de un diámetro de 1 cm y de 1 metro de largo, hicimos una variación de las constantes Kp, Ki y Kd y del tiempo en el que el acelerómetro toma los datos. Notamos que para valores muy grandes de Kp el movimiento era muy brusco ya que barre ángulos muy grandes en cada corrección, para Kp pequeños oscilaba bastante rápido. Luego al variar Kd, notamos que mientras más alto este valor la oscilación era menos fluida. Modificamos la parte inferior de la estructura, pero al hacerla más larga el equilibrista perdía más rápido el equilibrio lo cual le daba menos tiempo para regular su posición. De esta manera nos quedamos con el largo original. No se logró que el equilibrista se mantuviera por si solo perpendicular al eje de rotación por más de unos segundos. Esto fue

debido a que la barra que escogimos fue demasiado pesada y provocó que el motor no la lograse mover correctamente. Se probó una varilla de plástico hueca y resultó que era notablemente mejor la movilidad de esta gracias al motor, siendo más preciso para ángulos pequeños.

Capítulo 3

Conclusiones y referencias

Contents

3.1. Conclusiones

Ya que solo nos centramos en el largo de la barra pero no así en su peso, se nos dificultó encontrar las variables apropiadas para mantener el equilibrio del sistema, pues la estructura de este no era óptima y por esto el equilibrista entraba en estados de oscilación que, o no contrarestaban la aceleración de gravedad, o eran demasiado abruptos para mantener la estabilidad provocando su caída.

Además, la poca facilidad para reconocer el tipo de movimiento que era generado por la constante proporcional y la derivativa, nos dificultaron observar el correspondiente a la constante integrativa, por lo que nuestra experiencia con la constante derivativa.

Bibliografía

[1] http://www.ni.com/white-paper/3782/en/.

[2] Anderson R. and Cervo D.:Pro Arduinno, 2013.