

Universidad de Costa Rica
Escuela de Ingeniería Eléctrica
IE0117: Programación bajo plataformas abiertas

Controlador PID para Motor Síncrono

Informe Técnico Final

Daniel Díaz Molina	B22245
Luis Fernando Mora Mora	B24449
María Moraga Vargas	B14467

8 de mayo del 2016

1. Introducción

El presente trabajo es una simulación tanto de un motor síncrono como de un controlador PID, el cual tiene como fin controlar óptimamente algunas variables del motor, como lo es la velocidad. De esta manera se busca brindar mediante el uso del código abierto, una herramienta tecnológica en el uso del motor síncrono, lo que permite al usuario tener un producto acorde a sus necesidades y con una mayor factibilidad económica, además de integrar algunas herramientas importantes de la programación, como lo es el uso de Python y sus librerías Numpy y Sympy, las cuales permiten realizar cálculos complejos.

Es importante destacar que con la implementación de estas herramientas, el proyecto también comprende la trascendencia que ha tenido la automatización del trabajo a través del tiempo, ya que el uso de la programación es un insumo de gran valor para llevar a cabo tareas con una mayor productividad, no solo en términos de trabajo físico, sino que también en la parte intelectual.

Otro aspecto importante de este proyecto, es que el trabajo en equipo ayuda a fomentar las diferentes perspectivas que se tienen al abordar y generar posibles soluciones, permitiendo incorporar los conocimientos técnicos que se han puesto en práctica durante el curso de programación en plataformas abiertas, al mismo tiempo que ha contribuido como base para el desarrollo e implementación de un pensamiento más crítico, que busca siempre alternativas de mejora.

2. Nota Histórica

Este proyecto comprende, entre muchos temas, dos aspectos de gran trascendencia para el desarrollo del mismo. El primero de ellos es el uso del lenguaje de programación Python, el cual es un lenguaje, entre muchos aspectos, interpretado y multiparadigma. Este es administrado por la Python Software Foundation, bajo una licencia de código abierto, denominada Python Software Foundation License. Python fue creado a finales de los ochetas, por Guido Van Rossum en el CWI, en los Países Bajos.

En 1991, ya estaban presentes clases con herencia, manejo de excepciones, funciones y los algunos tipos modulares, posteriormente en 1994 se presentaron algunas herramientas de programación funcional. Luego en 1995 se presentaron varias versiones del software. Van Rossum buscaba, cada vez más, hacer la programación accesible, con un nivel de 'alfabetización' básico en lenguajes de programación, similar a la alfabetización básica en inglés y con habilidades matemáticas necesarias, en donde Python tuviera un papel crucial, debido a su orientación hacia una sintaxis limpia. La opción de usar Python con software disponible bajo GNU GPL se volvió más deseada, razón por la cual, para el 2000 se fueron dando arreglos de bugs, y de una nueva licencia compatible con GPL. De ahí en adelante se han dado nuevas implementaciones y arreglos, por ejemplo, una implementación del *scoping* y adiciones a la biblioteca estándar de Python, entre otras.

El segundo aspecto importante en este proyecto es el uso de los motores síncronos. Este es un tipo de motor de corriente alterna, en donde la rotación del eje está sincronizada con la frecuencia de la corriente de alimentación, es decir su velocidad de giro es constante. Los motores tuvieron su inicio en 1866 cuando Werner von Siemens patentó la dinamo, con lo cual no solo contribuyó al inicio de los motores eléctricos sino que también concepto de Ingeniería eléctrica.

Posteriormente, a mediados de la década de 1880, gracias a Nikola Tesla y a Werner von Siemens, la ingeniería eléctrica se fue desarrollando como disciplina, lo que le dió paso a la fascinación de los motores eléctricos y su trascendencia para provocar una nueva era de revolución. Pacinotti inventó la idea del motor eléctrico de corriente continua. Seguidamente los primeros motores eléctricos técnicamente utilizables fueron creados por el ingeniero Moritz von Jacobi, quien los presentó en 1834.

3. Marco Teórico

3.1. Principio Básico de los Motores síncronos

Los motores síncronos, al igual que todos los motores de corriente alterna, se componen de dos partes principales: estator y rotor. El rotor es alimentado con corriente directa por medio de escobillas, mientras que el estator es alimentado con tres fases de corriente alterna, que oscila a una frecuencia f . La interacción entre los campos magnéticos y las respectivas corrientes en cada parte del motor causan que el rotor gire Kosow (1993).

Sin embargo, los motores síncronos deben ser arrancados como motores de inducción, y para ello cuentan con una estructura de jaula de ardilla que les permite acelerarse hasta velocidad síncrona. Una vez que llegan a velocidad sincrónica, se mantienen girando a esta velocidad, ya que el movimiento es guiado por un campo magnético giratorio que gira a

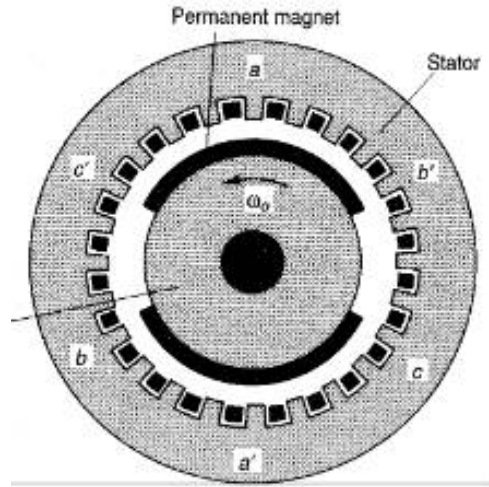


Figura 1: Motor de Imán Permanente

una velocidad proporcional a la frecuencia de la corriente que lo alimenta. Dicha velocidad síncrona puede encontrarse mediante la siguiente ecuación:

$$\omega = \frac{120f}{p} \quad (1)$$

Donde "p" es el número de pares de polos que tiene la máquina.

Ahora, entre los tipos de motores sincrónicos que hay, el más relevante para este proyecto es el motor síncrono de imán permanente. Este tipo de motor no necesita alimentación en corriente directa, ya que el campo magnético que se requiere en el rotor lo provee el material magnetizado del que está hecho. La figura 1 muestra el corte transversal de un motor de este tipo. El campo magnético proporcionado por el rotor magnetizado es constante, por lo tanto el flujo magnético es también constante. Esto último es relevante a la hora de analizar las ecuaciones que describen la tensión entregada e inducida en las bobinas y el torque producido por el rotor.

3.1.1. Ecuaciones del Motor

Por efecto de inducción electromagnética, el campo magnético producido por las bobinas del estator genera una tensión inducida en el rotor (Chapman, 2009) descrita por la siguiente ecuación :

$$E_a = \sqrt{2}\pi N_c \Phi f \quad (2)$$

Donde " Φ " es el flujo magnético del imán permanente, " f " es la frecuencia de oscilación de la corriente y N_c es el número de vueltas en las bobinas (cada una debe tener el mismo número de vueltas). Al saber que es constante, se puede simplificar la ecuación a la siguiente expresión:

$$E_a = K f \quad (3)$$

Se observa que la tensión inducida entonces sería constante para una red eléctrica cuya frecuencia no cambia en el tiempo.

Por otro lado, esta tensión inducida se puede expresar de forma *fasorial*, al igual que la tensión de fase en las bobinas (tensión de alimentación), denotada V_Φ . Debido a las reactancias inductivas y entrehierro, estas tensiones difieren en fase y magnitud. Al ángulo de desfase se le llama *ángulo de par* de la máquina, y se denota " δ ". Este, junto con las tensiones, describen el par generado por el motor según la siguiente ecuación:

$$\tau = \frac{3V_\Phi E_a \sin(\delta)}{\omega_m \chi_s} \quad (4)$$

Donde τ es el par generado por el motor, y χ_s es la reactancia síncrona de la máquina.

Por otro lado, una vez que el motor está girando, la fricción produce un torque en sentido opuesto al giro, que se puede modelar como una cantidad proporcional a la velocidad de giro:

$$\tau_f = K_f \omega \quad (5)$$

Las ecuaciones (4) y (5), junto con la segunda ley de Newton para cuerpos girando termina de describir el conjunto rotor-carga. Esta carga.^{es} lo que se le acopla al eje del rotor, que generalmente es una caja reductora de engranes para transformar la alta velocidad de giro del motor a una más baja y útil. Las cargas acopladas se pueden modelar simplemente como un momento de inercia extra, por lo que la ecuación de movimiento es de la forma:

$$\alpha = \frac{\tau - \tau_f}{I_r + I_L} \quad (6)$$

Donde α es la aceleración angular del conjunto, " I_r ."^{es} el momento de inercia del rotor e " I_L ."^{es} el momento de inercia de la carga acoplada al eje.

Conociendo la aceleración angular del conjunto, se puede encontrar su velocidad angular y su posición angular por integración directa:

$$\omega = \int_{t_0}^t \alpha dt \quad (7)$$

$$\theta = \int_{t_0}^t \omega dt \quad (8)$$

Las integrales anteriores se pueden aproximar en forma discreta, utilizando la regla del punto medio (Neuhauser, 2004), de la forma:

$$\omega = \int_{t_0}^t \alpha dt \sim \frac{\alpha_0 + \alpha_t}{2} \Delta t \quad (9)$$

$$\theta = \int_{t_0}^t \omega dt \sim \frac{\omega_0 + \omega_t}{2} \Delta t \quad (10)$$

Donde Δt es un intervalo de tiempo constante.

3.2. Lazo de Control Realimentado y Algoritmo PID

En la teoría de control automático, el tipo de control más utilizado es el *control realimentado*. En el mismo, se miden directamente las variables que se quieren controlar y se realimentan a un dispositivo controlador, el cual toma una decisión basándose en la variable medida. El algoritmo que utiliza el dispositivo controlador se llama *modo de control*, y se representa, usualmente con una función de transferencia con variable compleja s .

El modo de control está encargado de recibir una señal de referencia para compararla con el valor de la señal realimentada, de manera que se restan y se genera una *señal de error*. La señal de salida del controlador, que modificará una variable del sistema a controlar (usualmente llamado "Proceso"), será la multiplicación en frecuencia de la señal de error y la función del modo de control. La figura 2 muestra esta dinámica.

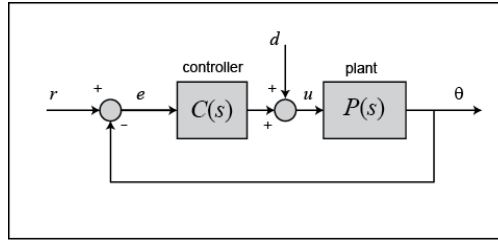


Figura 2: Sistema de Control Realimentado

Un algoritmo PID, que por sus siglas significa *Proporcional-Integral-Derivativo*, es un modo de control cuya función de transferencia es de la forma:

$$C(s) = K_p \left\{ 1 + \frac{1}{T_i s} + T_d s \right\} \quad (11)$$

Donde " K_p " es la constante del modo proporcional, T_i es el "tiempo integral" del modo integral, y T_d es el "tiempo derivativo" del modo derivativo.

Ambos tiempos, integral y derivativo, representan el tiempo que dura el respectivo modo en equiparar la respuesta del modo proporcional. Este algoritmo se puede implementar de forma discreta utilizando la misma regla del punto medio, mencionada anteriormente, y utilizando una aproximación para la derivada de una función que depende del tiempo, tal que:

$$\frac{df(t)}{dt} \sim \frac{f(t) - f(t + \Delta t)}{\Delta t} \quad (12)$$

Donde Δt es el mismo intervalo constante al usado en la integración.

4. Funcionamiento del Software Implementado

4.1. Clase PMSM (Motor Síncrono de Imán Permanente)

Esta clase modela las ecuaciones (2) hasta (10). El constructor recibe los parámetros propios del sistema. La clase provee las siguientes funciones:

- `begin(omega_inicial)`: Establece la velocidad inicial del motor (en radianes por segundo).
- `setInput(delta)`: Establece la entrada del motor, el ángulo δ entre V_Φ y E_a .
- `timestep(delta_t)`: Avanza el estado del motor un intervalo de tiempo Δt , esta función retorna una tupla que contiene la aceleración angular, la velocidad angular y el ángulo del eje, en ese orden, después del Δt .

4.2. Clase PID

Esta clase modela el controlador PID, en particular las ecuaciones (11) y (12). El constructor recibe los 3 parámetros del controlador, y además límites para el máximo y mínimo del esfuerzo del controlador. Provee las siguientes funciones:

- `begin(input)`: Establece el valor inicial de la variable controlada, normalmente esto sería en el punto de operación.
- `setRef(ref)`: Establece el valor deseado de la variable controlada.
- `setInput(input)`: Establece el valor sensado de la variable controlada.
- `timestep(delta_t)`: Avanza el estado del controlador un Δt , y retorna el esfuerzo del controlador.

4.3. Módulos conectados por YARP

YARP, que en inglés significa "*Yet Another Robot Platform*".^{es} una colección de programas, destinados a control automático en robótica, que permite conectar módulos de programas entre sí en tiempo real. Para poder conectar un objeto tipo PMSM y otro tipo PID se desarrollaron dos módulos en yarp para python.

El primer módulo de yarp crea un objetivo tipo motor utilizando la clase PMSM. Por medio de yarp se crean diferentes puertos por los que se puede conectar posteriormente. El segundo módulo crea un objeto tipo PID utilizando la clase PID, y se definen tres puertos de entrada: `/pid/in/inr` para cambiar el valor de la velocidad deseada en el motor y `/pid/in/inx` para conectar la salida de velocidad del motor como señal realimentada.

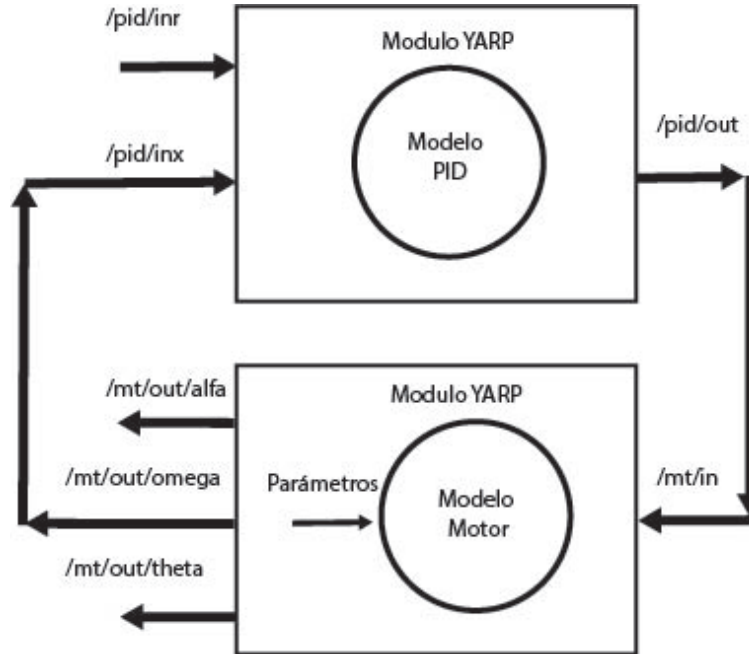


Figura 3: Esquema de Conexiones en YARP

5. Instrucciones de Uso y Tutorial

Primero se debe obtener el código fuente, este se encuentra en un repositorio público en GitHub. Para clonar el repositorio al directorio actual se requiere tener instalado git.

```
$ git clone https://github.com/danidim13/pid-motor-yarp.git
```

El repositorio cuenta con las clases del motor y el PID, además de los scripts para usarlos con YARP. Para correr estos últimos basta con ejecutarlos desde la línea de comandos. Si es necesario se debe dar permiso de ejecución a los archivos. También es necesario recordar que cada módulo de YARP se debe correr en una terminal diferente. Por ejemplo, para correr el controlador:

```
$ chmod u+x yarpPID.py      # dar permisos de ejecucion, esto
$ chmod u+x yarpMotor.py    # solo seria necesario la primera vez

$ yarpserver3 --write       # este comando y los 2 siguientes
$ ./yarpPID.py              # se deben correr cada uno en una
$ ./yarpMotor.py            # terminal diferente
```

Esto pone a correr el controlador y el modelo del motor. Luego para conectarlos se debe ejecutar lo siguiente en otra terminal:

```
$ yarp connect /pid/out /mt/in
$ yarp connect /mt/out/omega /pid/inx
```

Luego si se desea por ejemplo ver la velocidad del motor y cambiar la velocidad deseada en tiempo real:

```
$ yarp read /readw      # nuevamente estos dos se deben
$ yarp write /writew     # correr en terminales separadas

$ yarp connect /writew /pid/inr
$ yarp connect /mt/out/omega /readw
```

Como se puede suponer, los módulos se pueden conectar por separado a otros componentes por medio de YARP, de modo que el usuario podría darle otros usos sin ningún problema.

Referencias

(s.f.).

The Scipy Community. (2015, 18 de octubre). NumPy Reference Manual [Manual de software informático].

The Scipy Community. (2016, 20 de febrero). SciPy Reference Manual [Manual de software informático].

Chapman, S. (2009). *Electric Machinery Fundamentals* (3.^a ed.). Boston, MA: McGraw-Hill.

Kosow, I. (1993). *Máquinas eléctricas y transformadores*. Prentice Hall Hispanoamericana.

Descargado de <https://books.google.co.cr/books?id=5hJzpimPyXQC>

Neuhauser, C. (2004). *Matemáticas para ciencias*. Pearson-Prentice Hall. Descargado de <https://books.google.co.cr/books?id=APIw178ltvgC>

O'Dwyer, A. (2006). *Handbook of PI and PID Controller Tuning Rules* (2.^a ed.). Londres, Inglaterra: Imperial College Press.

Ogata, K. (2010). *Modern Control Engineering* (5.^a ed.). Nueva Jersey: Prentice Hall.

Worthing, B., Walker, B., Flores, G., Hilje, L., Mora, G., Carballo, M., ... others (1987). *The pesticide manual: a world compendium*. (n.º 668.65 P476p). British Crop Protection Council, Londres (RU).