Universidad Autónoma de Querétaro
Facultad de Ingeniería
Ingeniería en Automatización

Evaluación del desempeño de perfiles de velocidad con base en la ejecución del controlador aplicado a un motor de corriente directa

TESIS

Que como parte de los requisitos para obtener el grado de:

Ingeniero en Automatización

Presenta:

Emmanuel Rodríguez Núñez

Dirigido por:

Dr. Miguel Ángel Martínez Prado

Centro Universitario
Querétaro, Qro.
7 de febrero de 2023

Universidad Autónoma de Querétaro
Facultad de Ingeniería
Ingeniería en Automatización

Evaluación del desempeño de perfiles de velocidad con base en la ejecución del controlador aplicado a un motor de corriente directa

TESIS

Que como parte de los requisitos para obtener el grado de Ingeniero en Automatización

Presenta:

Emmanuel Rodríguez Núñez

Dirigido por:

Dr. Miguel Ángel Martínez Prado

Dr. Miguel Ángel Martínez Prado Presidente	Firma
Dr. Juvenal Rodríguez Reséndiz Secretario	Firma
M.C. José Luis Avendaño Juarez Vocal	Firma
M.C. Adyr Andrés Estévez de Bén Suplente	Firma

Dr. Manuel Toledano Ayala
Director de la Facultad

Centro Universitario
Querétaro, Qro.
7 de febrero de 2023

A mi familia y amigos.

Agradecimientos

Agradezco a Dios por permitirme transitar el camino para cumplir este objetivo.

A mi familia y amigos por su amor incondicional.

A mis profesores por compartirme con vocación su conocimiento.

Y mi Universidad por educarme en la verdad y en el honor.

Resumen

En este trabajo se desarrolla la evaluación del desempeño de cuatro perfiles de velocidad en un motor de corriente directa. Desde el punto de vista del desempeño del controlador de posición, se observa la influencia de su sobrepaso en el comportamiento de los perfiles Triangular, Trapezoidal, Parabólico y Polinomial. Por medio de tecnologías de bajo costo, basado en un Microcontrolador STM32F411CEU y aplicando tareas en tiempo real se realiza la estimación de la Raíz de Desviación Cuadrática Media de posición angular, el sobreimpulso y consumo energético de cada trazo de movimiento en una trayectoria definida, con la finalidad de establecer criterios adecuados al momento de selección de un perfil de velocidad sobre otro e incrementar eficiencia energética y posición en trayectorias punto a punto.

Palabras clave: Motor de corriente directa, perfil de movimiento, perfil de velocidad, consumo energético, eficiencia energética, control de posición

Abstract

This work develops an evaluation of performance about four velocity profiles applied on a direct current motor. This comparision come through out to the point of view of the development to the position controller. Watching the influnce of the overshoot and rise time from the response on the behaviour of the Triangular, Trapezoidal, Parabolic and Polinomial profiles. Throug low cost technologies, based on a Microcontroller STM32F411CEU and applying realtime tasks it is made the estimation of the Deviation Root Square Mean of the angular position, jerk and the energy consumption on a defined path, whose aim is to stablish right criteria at the time of select a velocity profile instead another one and to increase the position and energy efficiency on point to point trajectories.

Keywords: Direct Current Motor, motion profile, velocity profile, energy consumption, energetic efficiency, position control.

Nomenclatura

DC	Corriente directa
MCU	Microcontrolador
PtP	Punto a punto
CNC	Control Numérico Computarizado
v_i	Voltaje de entrada
i	Intensidad de corriente
v_e	Voltaje generado
τ_g	Torque generado
J_m	Inercia del motor
ω	Velocidad angular
θ	Posición angular
k_b	Fricción viscosa
k_e	Constante contraelectromotriz
k_t	Constante de torque
ω_{nid}	Frecuencia natural de identificación de la planta
ζ_{id}	Factor de amortiguamiento de identificación de la planta
M_{pid}	Sobrepaso de identificación de la planta
T_{pid}	Tiempo pico de identificación de la planta
k_{pid}	Ganancia proporcional para de identificación de la planta
T_s	Tiempo de muestreo
RDCM	Raíz de Desviación Cuadrática Media
α	Aceleración angular
θ_f	Posición angular final
Jerk	Sobreimpulso
UART	Unidad Asíncrona Transmisora Receptora

Índice

Índice de figuras

Índice de tablas

1. Introducción

El consumo de recursos naturales en la actualidad es una de las grandes preocupaciones en un escenario global, el crecimiento de la población demanda la producción de bienes y servicios para mejorar la calidad de vida. En este panorama, los motores de Corriente directa (DC) tienen un rol fundamental ya que son ampliamente utilizados en sectores de manufactura, automotriz, médico, recreativo, aeroportuario y en muchos otros campos alrededor del mundo. De acuerdo a [2] un promedio aproximado de la demanda energética global es de un 46 % requerida por motores, este estimado de un estudio de 55 países esta dado por:

- Industrial: 29 %;
- Comercial: 9 %;
- Agricultura:1 %;
- Residencial: 6 %;
- Transporte: 1 %;

Si bien esta información no hace diferenciación acerca de los tipos de motores y sus usos específicos, logra aportar una valiosa perspectiva en cómo estos sistemas representan una variable importante en el uso de la energía eléctrica.

La implementación de perfiles de velocidad para motores de DC por medio de algoritmos de control, representa una vía que permite observar parámetros cuantificables que forman parte de una lista de variables físicas de importancia tales, como voltaje, corriente eléctrica y posición en la ejecución de tareas específicas. El desarrollo de este documento plantea aplicar un sistema de libre acceso utilizando herramientas de bajo costo, que permitan realizar la experimentación en una plataforma de pruebas para validar el nivel de gasto en energía, el Sobreimpulso (*Jerk*) asociado y la Raíz de Desviación Cuadrática Media (RDCM) de posición en una trayectoria propuesta.

Los análisis serán llevados a cabo en los perfiles de movimiento mayormente usados en la industria tales como triangular, trapezoidal, parabólico y polinomial cuyos casos se realizarán en un entorno de pruebas con un riel de guía lineal de doble eje, un motor de DC con escobillas y la fase de control por medio de un

Microcontrolador (MCU) STM32F411CEU6 basado en un sistema multitarea implementado en prioridades de ejecución.

1.1. Descripción del problema

La gestión de perfiles de velocidad en máquinas industriales presenta al motor como una herramienta que permite altos niveles de producción. De acuerdo a [9], este hecho motiva a indagar cómo los motores se encuentran ligados al consumo de energía y la eficiencia en controladores de sistemas electromecánicos.

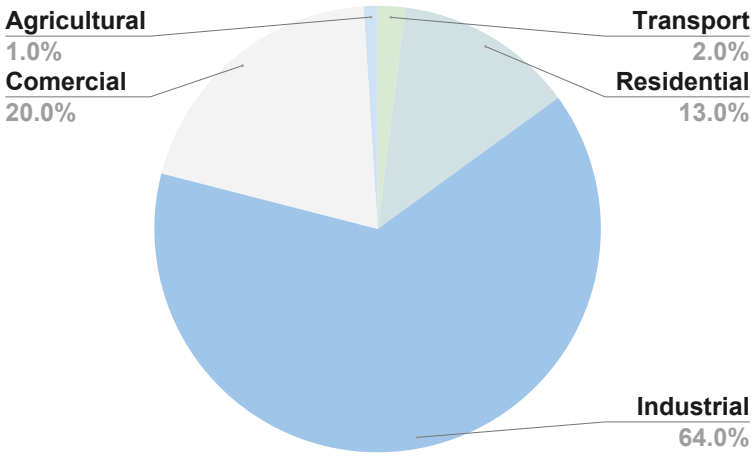

Figura 1: Tomada de [9], Energía utilizada por motores en cada sector.

El amplio uso de motores en la industria como se muestra en la Figura 1 nos invita a realizar más investigación acerca de la mejora en las tareas realizadas con perfiles de velocidad y los ahorros de energía que conllevan. Por ello es relevante hacerlo manifiesto a través de bien conocidos métodos de control de movimiento en combinación a tecnologías contemporáneas que se encuentran de fácil acceso.

La implementación de este sistema permite conocer el impacto que tienen las características de la respuesta del control de posición en los perfiles de velocidad y el monto de energía consumida por motores de DC. Un atributo positivo de esta propuesta es que garantiza al usuario, debido a los resultados cuantitativos, realizar una decisión imparcial en la selección de un perfil de velocidad en máquinas en donde estos sistemas electromecánicos se encuentran disponibles.

1.2. Justificación

La búsqueda de eficiencia de trayectorias, ahorro de energía y recursos que prolonguen su vida útil en sistemas que utilizan motores eléctricos, permite mantener el interés y promueve la investigación en torno al desempeño de perfiles de velocidad.

A nivel industrial, es frecuente utilizar mejoras que involucran cambios en hardware, lo cual eleva costos y durante el tiempo de implementación disminuye la productividad. Para cambiar esta situación, una de las soluciones más efectivas es buscar la mejora directa por software en campo, es aquí donde un perfil de movimiento tiene una incidencia importante. Como sugiere [3], un perfil de velocidad presenta una opción viable para investigar y buscar la optimización, lo cual ha sido llevado a cabo aproximadamente los últimos 100 años.

De acuerdo a [18], en países donde los estándares y políticas de mejoras en controladores de motores son implementados, por ejemplo, Canadá y Estados Unidos de América, el marcador de eficiencia en motores es del 70 % mientras que en países Europeos donde las políticas y los programas no han sido adoptados es frecuentemente por debajo del 15 %. El consumo energético a nivel global realizado por motores eléctricos es un escenario que tiene que ser observado para realizar esfuerzos en la búsqueda de contribuir en números positivos.

Es de amplio interés el uso de soluciones que puedan implementarse cuando el costo no puede ser muy alto o los recursos se encuentran limitados en cuanto a tecnología se refiere, propiciar la investigación y aumento de interés en el desarrollo de nuevas trayectorias en el uso de perfiles de velocidad, y como menciona [8], la importancia del control de posición para alcanzar un desempeño adecuado en los perfiles de movimiento.

Enfrentar estos retos en motores de DC, es una oportunidad para exponer la importancia de la Ingeniería en Automatización en nuestros días, además de promover el uso de tecnologías de bajo costo con el objetivo de proveer cambios positivos en la huella de carbono.

1.3. Objetivos e Hipótesis

1.3.1. Objetivo general

Evaluar el desempeño entre diferentes perfiles de velocidad dependiendo de nivel de sobrepaso de la respuesta del controlador de posición aplicado en un motor de DC, a través de algoritmos de control basados en un MCU y realizando la misma tarea de movimiento en un dispositivo punto a punto (PtP). Adicionalmente comparar resultados para conocer el mejor rendimiento cuando una secuencia repetitiva es aplicada sobre un motor.

La generación de estas comparativas es a través del monitoreo del rendimiento de los perfiles tradicionales y/o polinomiales, que permitan establecer documentación que proporcione información de utilidad al usuario que desea su implementación desde el controlador de posición.

En este documento se proporciona la evaluación del desempeño y la información de energía, Jerk y la Raíz de Desviación Cuadrática Media (RDCM) de posición encontrada en un trayecto definido. Se busca la correlación existente entre la ejecución del perfil y el desempeño presentado por el controlador utilizado.

1.3.2. Objetivos particulares

- Implementar una serie de pruebas con el objetivo de obtener un número suficiente de datos para comparar, tasar el consumo energético y su relación con el desempeño del controlador de posición.
- Implementar una estación de experimentación para llevar a cabo las pruebas de consumo.
- Promover el uso de sistemas de bajo costo que permiten mejoras. Con la implementación de tecnologías que las industrias pueden encontrar atractivas en procesos donde motores de DC están involucrados.

1.4. Hipótesis

Determinar que un nivel de sobrepaso mayor en la respuesta de control de posición, influye en al menos 2 % de RDCM de posición angular y un 10 % en consumo

energético en la implementación de perfiles de velocidad al ejecutar una tarea PtP bajo las mismas condiciones aplicadas en un motor de DC.

1.5. Antecedentes

Desde 1873 que se empleó el término de servomecanismo según [4], siendo usados aproximadamente desde 1851 se ha buscado la manera de controlar eficientemente la energía utilizada por un motor. Sin embargo, el motor eléctrico y el uso de microcontroladores en conjunto surge hasta después de 1950 presentando una vertiente más efectiva en el uso de control de retroalimentación [21]. La electrónica ha evolucionado desde entonces en capacidad de memoria y velocidad de procesamiento de los MCU lo cual permite realizar algoritmos y tareas de manera más satisfactoria.

Un método altamente usado en cuanto optimización en el movimiento y de energía en motores de DC son los perfiles de velocidad o también llamados perfiles de movimiento [20]. Los que mayormente son utilizados en la industria y campo médico-farmacéutico dependiendo de la tarea que realizarán son: Trapezoidal, parabólico, S-curve por mencionar algunos[7].

Dentro del área de investigación, se han utilizado diferentes plataformas para observar el comportamiento y el consumo energético. En [12] se buscan soluciones mediante la implementación de algoritmos en un Field Programable Gate Array (por sus siglas en inglés FPGA) con un perfil de movimiento trapezoidal. También se ha abordado desde la óptica de su desempeño en el jerk, comparando perfiles S-curve y trapezoidal como presenta [5]. Análogamente [13], con la implementación de algoritmos recursivos en trazos de movimiento polinomiales y trigonométricos para la minimización del jerk.

Por su parte, [1] presenta una comparación de la eficiencia de los perfiles trapezoidal y sinusoidal. Y donde históricamente la industria de manufactura presenta los más altos niveles de emisiones de carbono, [17] investiga el potencial en ahorro de energía mediante la optimización en perfiles de aceleración en Robots Industriales.

Es claro que la eficiencia en el funcionamiento de motores ha sido una búsqueda constante de mejoras y que aún concierne un análisis en diferentes campos de especialización tal como la Robótica y la Automatización.

2. Motivación

La implementación de perfiles de movimiento es ampliamente investigada, sin embargo, no son del todo claro ciertos criterios del controlador a utilizar para su construcción, y si el comportamiento del mismo tiene relación con el desempeño y eficiencia del perfil utilizado.

Para hablar de la importancia de mejoras en eficiencia, saber que dos terceras partes de la demanda eléctrica en la industria es por el uso de motores, y que, el potencial que se tiene para mejoras en estos sistemas puede ser entre el 20 % y el 25 % para el año 2030 de acuerdo a los datos de la Organización por la Economía y Cooperación y Desarrollo (OECD) [2], quien sugiere una importante implementación de políticas para motivar una mejora de estos sistemas. En este escenario es de gran interés el ahorro que pueda llevarse acabo no solo por los costos económicos sino por el impacto ambiental que esto representa.

Es de interés en este trabajo la implementación de una propuesta de análisis, que permita tener en cuenta el desarrollo de un perfil de velocidad tomando en consideración los factores previos de control que tienen injerencia en su desempeño. Con ello, observar parámetros como el jerk, eficiencia y el monto de energía consumida. A fin de encontrar pautas adecuadas que puedan ser implementadas en la optimización de trayectorias PtP en tareas repetitivas usadas en máquinas que utilizan motores de DC.

3. Fundamentos teóricos

3.1. Motor de corriente directa

En máquinas herramientas industriales, Control Numérico Computarizado (CNC) y aplicaciones de uso médico es común el uso de motores de DC. Acorde a [6], las ventajas como precisión, rapidez y par de arranque elevado, los hacen una opción robusta en la ejecución de tareas aplicadas es los campos mencionados.

Se presenta a continuación una descripción que permita observar en la generalidad los conceptos básicos de un motor de DC.

Un motor de DC presenta una gran variedad de componentes, algunos represen-

tativos son:

- Rotor

  Parte central donde los embobinados permiten la circulación de corriente, misma que interactúa con el campo magnético circundante induciendo así así su giro.

- Estator

  Elemento fijo que presenta por lo general, imanes permanentes que crean el campo magnético circundante.

- Carcasa

  Es el contenedor del conjunto en general de los demás componentes.

- Armadura

  Es una pieza de metal en la que los embobinados son enrollados y permite la interacción de estos con el campo magnético que lo rodea.

- Conmutador

  Pieza que permite a la corriente eléctrica fluya en el sentido correcto impulsando al rotor en la dirección adecuada.

- Eje de giro

  Transmite la rotación generada por medio de acoplamientos al exterior.

- Escobillas o carbones

  Permiten el flujo de electricidad hacia los embobinados presentes en el rotor.

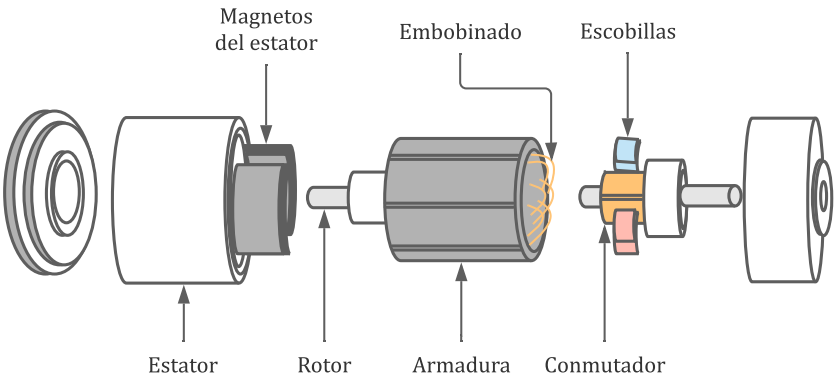

Figura 2: Estructura de motor de DC.

3.1.1. Principio de funcionamiento

Al circular corriente a través del embobinado por medio de las escobillas, el movimiento es generado cuando se repelen los polos magnéticos de un imán permanente al interactuar con los polos del electroimán formado por la bobina y el rotor montados sobre el eje. Se genera un par de fuerza que promueve el rompimiento de la inercia provocando el giro de su eje. Este proceso de convertir la energía eléctrica en mecánica puede ser modelado dinámicamente para permitir acciones de control en el sistema.

3.1.2. Modelado de un motor de DC

Los componentes eléctricos básicos de un motor de DC pueden ser representados como se muestra a continuación.

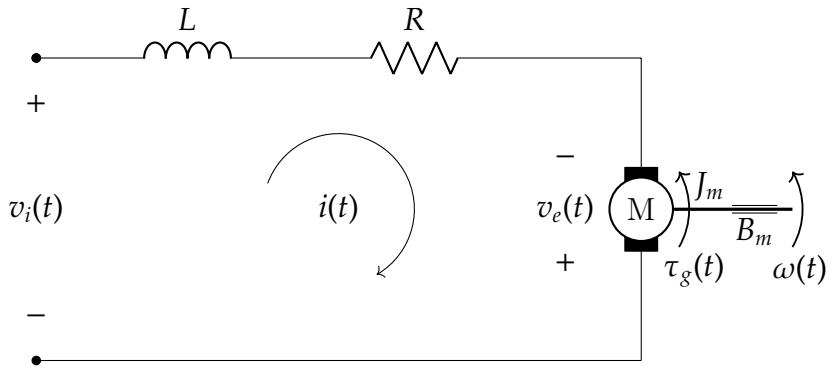

Figura 3: Circuito motor DC.

En la Figura 3 la representación eléctrica y mecánica de un motor DC nos muestra las variables que, por medio de la ley de voltajes de Kirchhoff y la segunda ley de Newton para la rotación se obtienen las ecuaciones para realizar el modelo dinámico.

La ecuación eléctrica del motor de la forma

$$v_i(t) = Ri(t) + L\frac{di(t)}{dt} + v_e(t) \tag{3.1}$$

donde,

- $v_i(t)$ es el voltaje de entrada.

- R es la resistencia.
- L es la inductancia.
- $i(t)$ es la corriente.
- $v_e(t)$ es el voltaje generado.

Ecuación mecánica de la forma

$$\tau_g = J_m(t)\frac{d\omega(t)}{dt} + k_b\omega(t) \tag{3.2}$$

donde,

- τ_g es el torque generado.
- J_m es la inercia del motor.
- ω es la velocidad angular.
- k_b es la fricción viscosa.

De la ecuación 3.1 es conocido que, el voltaje generado es directamente proporcional al producto de la velocidad angular $\omega(t)$ y la contra-electromotriz k_e, esto es:

$$v_e = k_e\omega(t) \tag{3.3}$$

De forma equivalente de la ecuación 3.2 se sabe que el torque generado cuenta con una proporción directa al producto de la Constante de torque (k_t) ejercida por el motor y la corriente $i(t)$ que circula por cada uno de los devanados del mismo, esto es:

$$\tau_g = k_t i(t) \tag{3.4}$$

En las ecuaciones 3.3 y 3.4 se presentan dos constantes que son intrínsecas del modelo electromecánico de un motor DC, cuando se usa el mismo sistema de unidades se sabe que:

$$k_e = k_b \tag{3.5}$$

Las ecuaciones eléctrica y mecánica al ser llevadas al dominio de la frecuencia

$$vI(s) = RI(s) + LsI(s) + K_e\omega(s) \tag{3.6}$$

$$K_tI(s) = J_ms\omega(s) + k_b\omega(s) \tag{3.7}$$

De las ecuaciones presentadas y aplicando las relaciones matemáticas correspondientes se obtienen funciones de transferencia que permiten asociar posición angular $\theta(t)$, velocidad angular $\omega(t)$, voltaje aplicado $v_i(t)$ e intensidad de corriente $i(t)$.

Despejando $I(s)$ de (3.6)

$$I(s) = \frac{vI(s) - K_e\omega(s)}{Ls + R} \tag{3.8}$$

Sustituyendo (3.8) en (3.6)

$$\frac{\omega(s)}{V_i(s)} = \frac{K_t}{(Ls + R)(J_mS + k_b) + k_tk_e} \tag{3.9}$$

Este modelo de velocidad permite aplicar una integral temporal para obtener el modelo de posición siguiente.

$$\frac{\theta(s)}{V_i(s)} = \frac{K_t}{s\left[(Ls + R)(J_mS + k_b) + k_tk_e\right]} \tag{3.10}$$

Así como muestra [19] en su diagrama de motor controlado por inducido, de forma similar la Fig. 4 presenta función de transferencia de la velocidad angular y el voltaje de entrada así como las constantes involucradas.

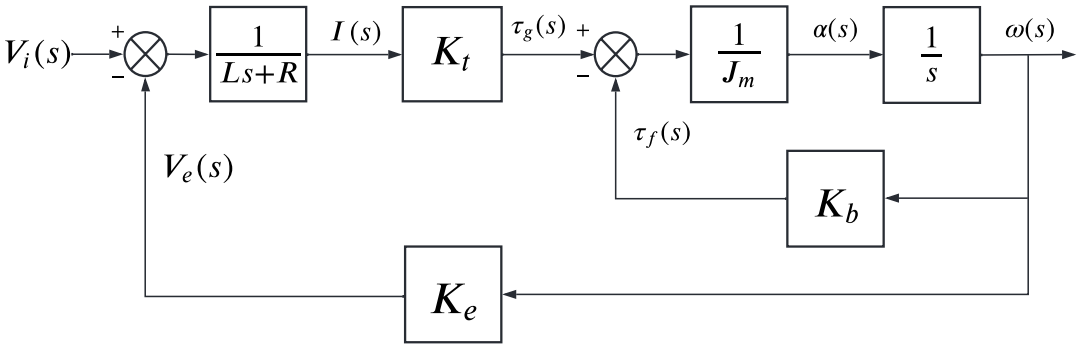

Figura 4: Modelo dinámico de un servomotor.

En el análisis realizado por [19], indica que es común en algunos modelos de motores DC el despreciar la constante de tiempo del inducido, lo cual sugiere que posible y útil el uso de modelos simplificados presentando resultados confiables. Para el caso de este documento se presenta el caso del uso de un transductor lo cual reduce la parte eléctrica a una constate unitaria K_a, por lo que se tiene.

$$\frac{\theta(s)}{V(s)} = \frac{K_a K_t}{s(J_m s + K_b)} \tag{3.11}$$

donde,

- k_a es la ganancia del servoamplificador.
- $V(s)$ es la corriente suministrada siendo esta una referencia de voltaje suministrada a través de un transductor.

La comprensión de este sistema permite conceptualizar por medio de una función de transferencia en la propuesta de control de posición de dicha máquina.

3.2. Antecedentes matemáticos

3.2.1. Perfil de velocidad

Un perfil de velocidad proporciona información del movimiento de un motor, gráficamente muestra cómo debe comportarse en términos de posición, velocidad y

aceleración. Su uso es amplio en máquinas de control numérico, robots manipuladores y máquinas herramientas realizan tareas que deben cumplir cierta trayectoria en un tiempo determinado, lo cual trae consigo una idea del uso de la cinemática básica.

Es conocido que una trayectoria es el desplazamiento de una partícula en un conjunto de posiciones, primitivamente de un punto A hacia un punto B dentro de un marco de referencia. Como sugiere [7], aplicando el factor tiempo requerido entre el desplazamiento es posible aplicar la primera y segunda derivada de posición $s(t)$ con el objetivo de obtener los principios cinemáticos usados en perfiles de movimiento, esto es:

Velocidad

$$v(t) = \frac{dS}{dt} \tag{3.12}$$

Aceleración

$$a(t) = \frac{dv}{dt} \tag{3.13}$$

De forma análoga e inversa, si es dada la aceleración $a(t)$ en lugar de posición es razonable el uso de la integración de la siguiente forma:

Velocidad:

$$v(t) = \int_{t_0}^{t_f} a(t)dt$$

Posición:

$$s(t) = \int_{t_0}^{t_f} v(t)dt$$

Las relaciones cinemáticas se muestran en la Figura 5.

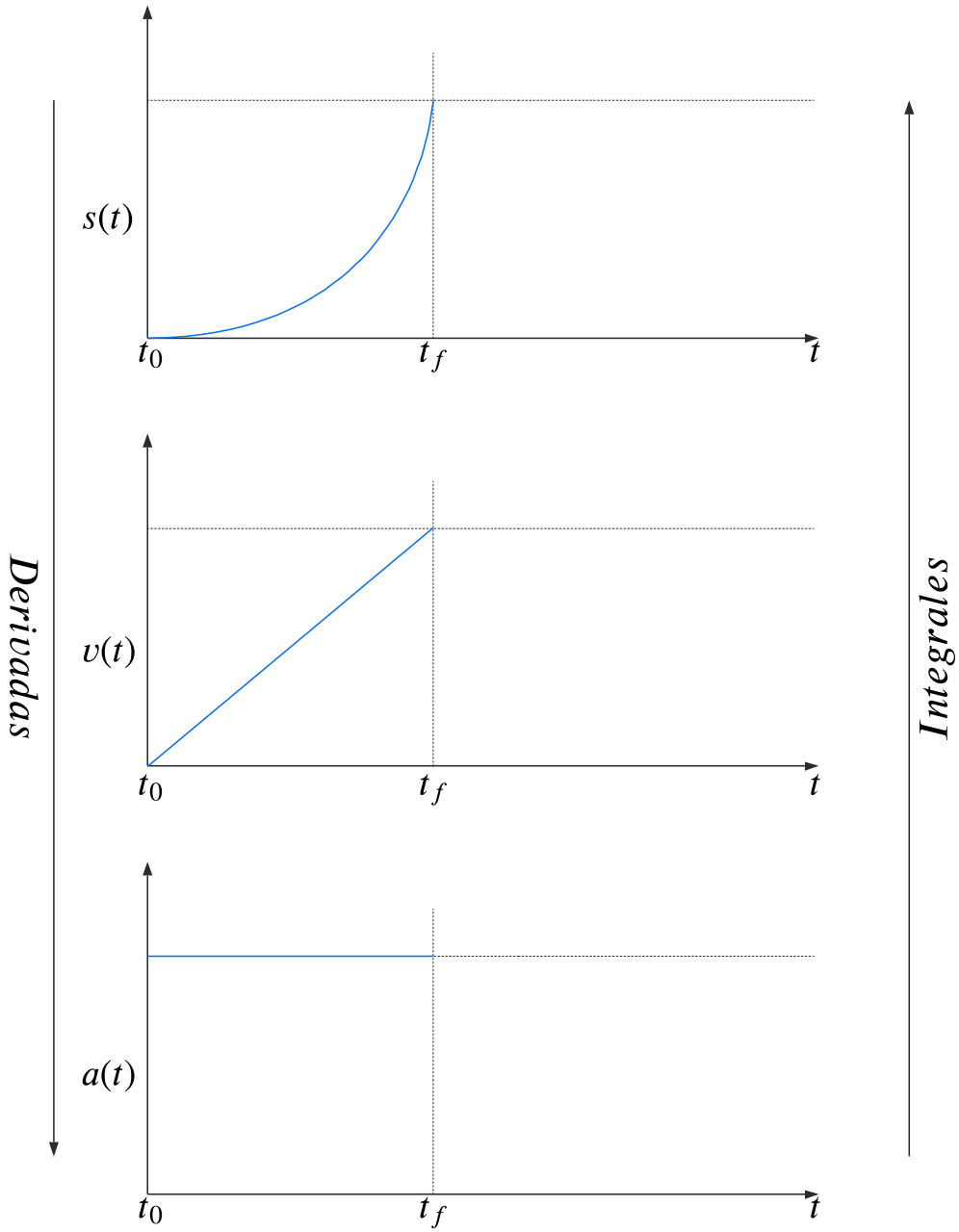

Figura 5: Relaciones cinemáticas.

Siguiendo estos principios cinemáticos se desarrollan los perfiles de movimiento más usados en la industria.

3.2.2. Desarrollo de perfil de velocidad Triangular

Como es mostrado en la Figura 6, su nombre alude a la forma en que su movimiento se comporta. Una característica notable es, que muestra dos curvas lineales que representan la aceleración y desaceleración del eje del motor. Durante el tiempo de ejecución de una tarea se obtienen linealmente incrementos y decrementos en la velocidad.

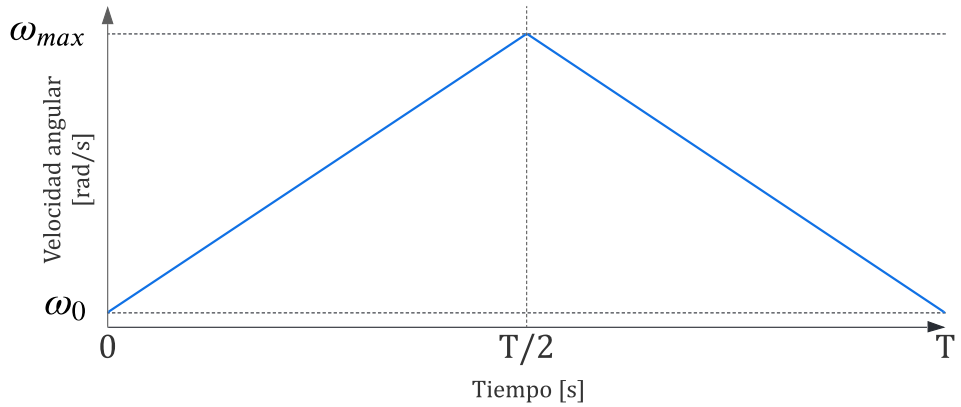

Figura 6: Función de velocidad.

Para una función lineal existe una constante como derivada, siendo en el caso de la primera parte de la función triangular una constante a positiva mientras que en la parte de desaceleración sera una constante negativa $-a$ de la misma magnitud como se muestra en Fig. 7.

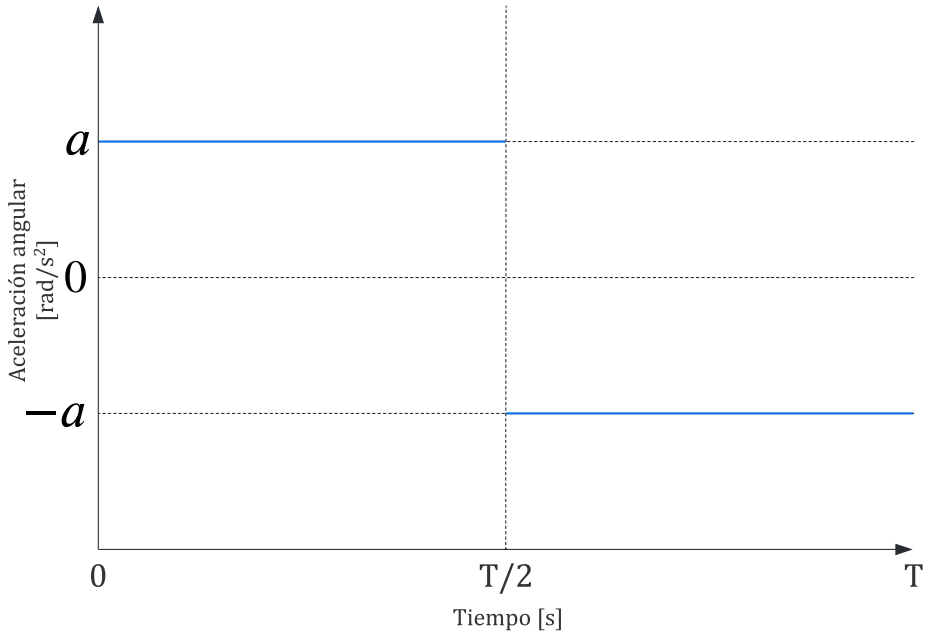

Figura 7: Función de aceleración.

La aceleración $\alpha\colon [0,T] \to \mathbb{R}$ está dada por

$$\alpha(t) = \begin{cases} +a & 0 < t \leq \frac{T}{2} \\ -a & \frac{T}{2} < t \leq T \end{cases} \tag{3.14}$$

Donde T es el tiempo del movimiento. Como se comentó en la sección anterior por medio de una integración respecto al tiempo de la ecuación (3.14) se obtiene la velocidad angular $\omega(t)$. Si se asume una condición inicial estática donde $\omega(0) = 0$, para $0 \leq t \leq T/2$, se obtiene

$$\omega(t) = \int_0^t d\omega(s) + \omega(0) = \int_0^t d\alpha(s)ds + 0 = a\int_0^t ds = at \tag{3.15}$$

En $t = T/2$ se obtiene la máxima velocidad angular

$$\omega_{max} = \omega\left(\frac{T}{2}\right) = \frac{aT}{2} \tag{3.16}$$

Esto es para $T/2 < t \leq T$, se tiene

$$\omega(t) = \int_{T/2}^{t} d\omega(s) + \omega\left(\frac{T}{2}\right) = \int_{T/2}^{t} \alpha(s)ds + \frac{aT}{2} = -at + aT \tag{3.17}$$

Por ello,

$$\omega(t) = \begin{cases} +at, & 0 < t \leq \frac{T}{2} \\ -at + aT, & \frac{T}{2} < t \leq T \end{cases} \tag{3.18}$$

Análogamente para la posición angular θ se tiene la función.

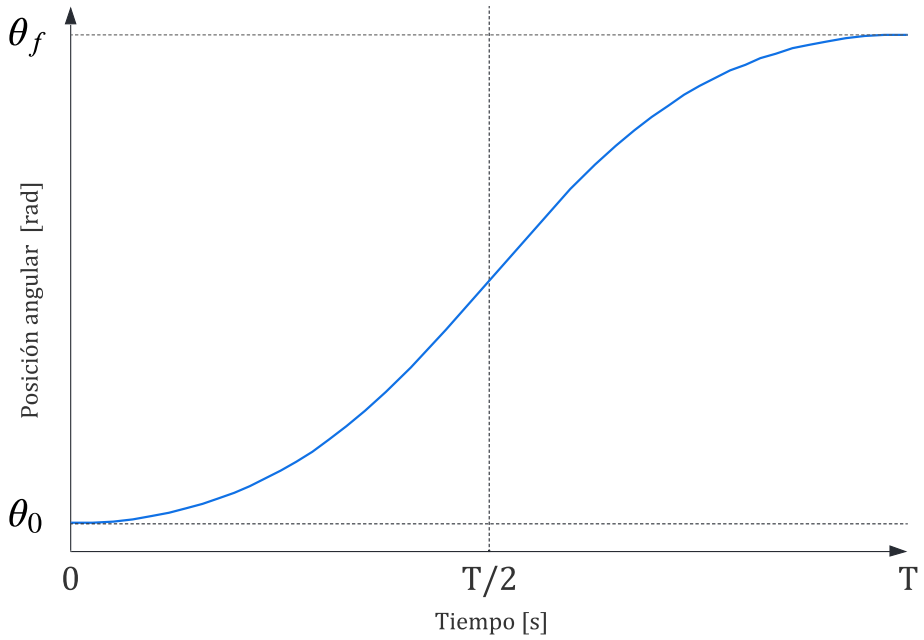

Figura 8: Función de posición.

Es calculada como sigue.

Para $0 \leq t \leq T/2$,

$$\theta(t) = \int_0^t d\theta(s) + \theta(0) = a \int_0^t s ds + \theta(0) = \frac{at^2}{2} + \theta(0) \tag{3.19}$$

Asumiendo $\theta(0) = 0$ se obtiene

$$\theta(t) = \frac{1}{2} a t^2 \tag{3.20}$$

En mitad de ciclo

$$\theta\left(\frac{T}{2}\right) = \frac{1}{8} a T^2 \tag{3.21}$$

Para $T/2 < t \leq T$ se desarrolla

$$\begin{aligned} \theta(t) &= \int_{T/2}^t d\theta(s) + \theta\left(\frac{T}{2}\right) \\ &= -a \int_{T/2}^t s ds + aT \int_{T/2}^t ds + \frac{aT^2}{8} \\ &= -\frac{at^2}{2} + aTt - \frac{aT^2}{4} \end{aligned}$$

Obteniendo

$$\theta(t) = -\frac{1}{2} a t^2 + aTt - \frac{1}{4} a T^2 \tag{3.22}$$

Por lo tanto,

$$\omega(t) = \begin{cases} \frac{1}{2}at^2, & 0 < t \leq \frac{T}{2} \\ -\frac{1}{2}at^2 + aTt, & \frac{T}{2} < t \leq T \end{cases} \tag{3.23}$$

Esto permite calcular la posición angular final θ_f que sucede en el tiempo $t = T$ como

$$\theta_f = \theta(T) = -\frac{aT^2}{2} + aT^2 - \frac{aT^2}{4} = \frac{1}{4}aT^2 \tag{3.24}$$

Una relación ente la posición angular final θ_f, la máxima velocidad angular ω_{max} y el tiempo requerido para realizar el movimiento en T es

$$\theta_f = \frac{1}{2}\omega_{max}T \tag{3.25}$$

De esta ecuación, el tiempo T es obtenido como

$$T = \frac{2\theta_f}{\omega_{max}} \tag{3.26}$$

Ahora es posible resolver para la aceleración a en la ecuación (3.16)

$$a = \frac{2\omega_{max}}{T} \tag{3.27}$$

En términos de posición y velocidad angular

$$a = \frac{\omega_{max}^2}{\theta_f} \tag{3.28}$$

Si bien el perfil de velocidad triangular presenta una geometría sencilla en su trayecto, es relativamente ineficiente debido a que la velocidad angular $\omega(t)$ comienza a decrementar en el preciso momento de alcanzar su máximo valor ω_{max} hasta convertirse a cero, esto implica que la energía usada en alcanzar la velocidad máxima no es aprovechada por el motor al comenzar la desaceleración al llegar a ese punto.

3.2.3. Desarrollo de perfil de velocidad Trapezoidal

Se presenta una alternativa que permite un mejor aprovechamiento de la energía por un periodo, esto es cuando se alcanza la velocidad angular máxima ω_{max}, manteniendo su máximo valor durante un intervalo de tiempo. Esto puede observarse en la Figura 9 del perfil de velocidad Trapezoidal.

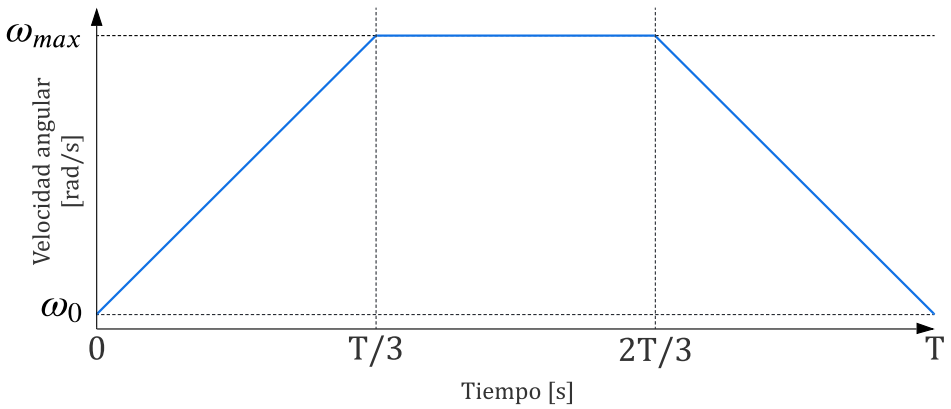

Figura 9: Función de velocidad.

Se define la aceleración para el presente perfil

$$\alpha(t) = \begin{cases} +a, & 0 \leq t \leq \frac{T}{3} \\ 0, & \frac{T}{3} < t \leq \frac{2T}{3} \\ -a, & \frac{2T}{3} < t \leq T \end{cases} \tag{3.29}$$

Mostrando su función en Fig.10

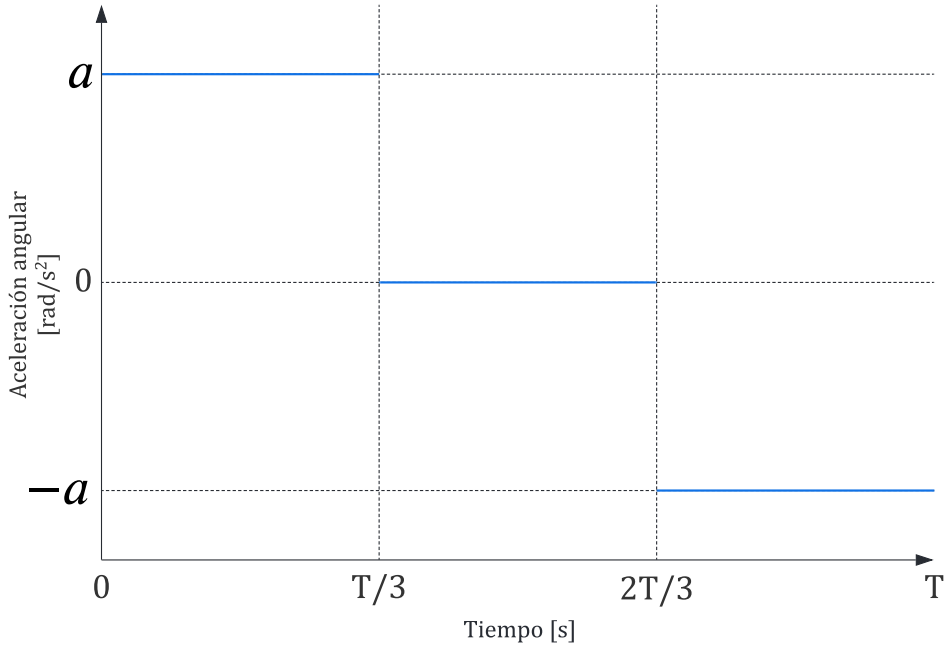

Figura 10: Función de aceleración.

Siendo $a > 0$ una constante de aceleración y T la duración del movimiento. Es posible el cálculo de $\omega(t)$ por medio de la integración de $\alpha(t)$.

Como se realizó anteriormente asumimos que $\omega(0) = 0$, para $0 \leq t \leq T/3$, esto es

$$\omega(t) = \int_0^t d\omega(s) + \omega(0) = a \int_0^t ds + 0 = at \tag{3.30}$$

La máxima velocidad es alcanzada en $t = T/3$, esto es

$$\omega_{max} = \left(\frac{T}{3}\right) = \frac{1}{3}aT \tag{3.31}$$

Para el intervalo $T/3 < t \leq 2T/3$ la velocidad permanece constante

$$\omega(t) = \int_{T/3}^{t} d\omega(s) + \omega\left(\frac{T}{3}\right) = \int_{T/3}^{t} 0ds + \frac{aT}{3} = \frac{1}{3}aT \tag{3.32}$$

Finalmente, para $2T/3 < t \leq T$ se desarrolla

$$\begin{aligned}\omega(t) &= \int_{2T/3}^{t} d\omega(s) + \omega\left(\frac{2T}{3}\right) \\ &= -a\int_{2T/3}^{t} ds + \omega\left(\frac{T}{3}\right) \\ &= -a + \frac{2aT}{3} + \frac{aT}{3}\end{aligned}$$

Obteniendo

$$\omega(t) = -at + aT \tag{3.33}$$

Siendo para los tres intervalos,

$$\omega(t) = \begin{cases} +at, & 0 \leq t \leq \frac{T}{3} \\ \frac{1}{3}aT, & \frac{T}{3} < t \leq \frac{2T}{3} \\ -at + aT, & \frac{2T}{3} < t \leq T \end{cases} \tag{3.34}$$

Para calcular la posición angular, se realiza el procedimiento para cada intervalo.

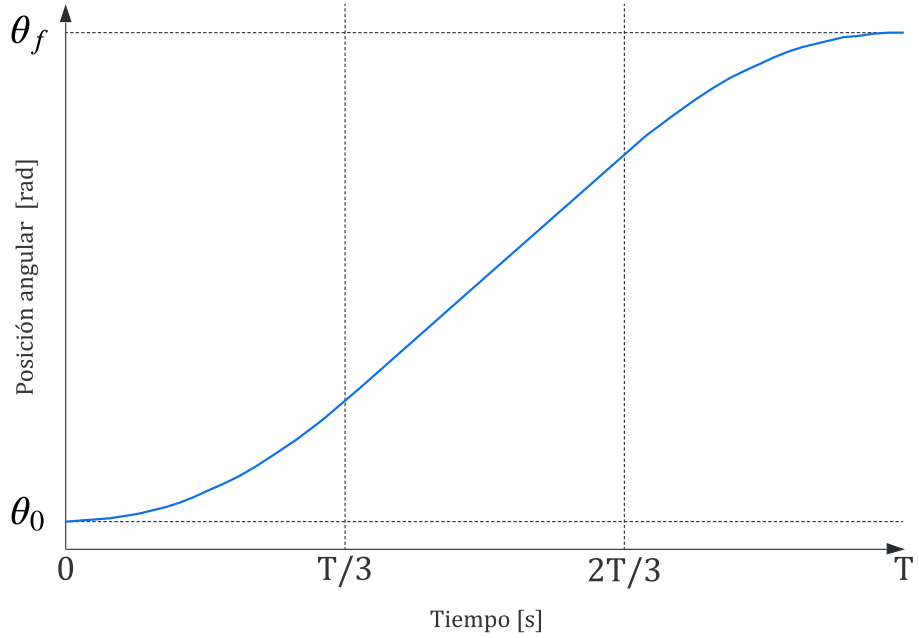

Figura 11: Función de posición.

Para $0 \leq t \leq T/3$, se asume $\theta(0) = 0$,

$$\theta(t) = \int_0^t d\theta(s) + \theta(0) = a \int_0^t sds + = \frac{1}{2}at^2 \tag{3.35}$$

Para $T/3 < t \leq 2T/3$,

$$\begin{aligned}\theta(t) &= \int_{T/3}^{t} d\theta(s) + \theta\left(\frac{T}{3}\right) \\ &= \frac{aT}{3}\int_{T/3}^{t} ds + \frac{aT^2}{18} \\ &= \frac{aT}{3}t - \frac{aT^2}{9} + \frac{aT^2}{18}\end{aligned}$$

Se obtiene

$$\theta(t) = \frac{1}{3}aTt - \frac{1}{18}aT^2 \tag{3.36}$$

Finalmente, para $2t/3 < t \leq T$,

$$\begin{aligned}\theta(t) &= \int_{2T/3}^{t} d\theta(s) + \theta\left(\frac{2T}{3}\right) \\ &= -a\int_{2T/3}^{t} sds + aT\int_{2T/3}^{t} ds + \frac{aT^2}{6} \\ &= -\frac{a}{2}t^2 + \frac{4aT^2}{18} + aTt - \frac{2aT^2}{9} + \frac{aT^2}{6}\end{aligned}$$

Se obtiene

$$\theta(t) = -\frac{1}{2}at^2 + aTt - \frac{1}{18}5aT^2 \tag{3.37}$$

Después, la posición final θ_f. Esto es, la posición cuando $t = T$, es dado por

$$\theta_f = \theta(T) = \frac{2}{9}aT^2 \tag{3.38}$$

En donde se sustituye la ecuación (3.31),

$$\theta_f = \frac{2T}{3} \cdot \frac{aT}{3} = \frac{2}{3} T\omega_{max} \tag{3.39}$$

Así,

$$T = \frac{3\theta_f}{2\omega_{max}} \tag{3.40}$$

De la misma forma, de la ecuación (3.31), implica

$$a = \frac{3\omega_{max}}{T} \tag{3.41}$$

En términos de posición y velocidad angular

$$a = 2\frac{\omega_{max}^2}{\theta_f} \tag{3.42}$$

En términos de aceleración, la función utilizada para el perfil trapezoidal presenta discontinuidades, lo cual significa que en consumo energético solicitará impulsos grandes al comienzo y en los cambios de velocidad para llevar el motor a paro.

3.2.4. Desarrollo de perfil de velocidad Parabólico

La segunda derivada de la curva de posición que no presenta discontinuidades, es un característica de una función parabólica. En un perfil de velocidad representa una curva más suave en comparación con los presentados con anterioridad, lo que podría suponer un menor consumo de energía.

El perfil parabólico de velocidad se desarrolla como sigue

$$\alpha(t) = mt + \alpha(0) \tag{3.43}$$

Donde

$$m = \frac{-a - a(+a)}{T - 0} = \frac{-2a}{T} \tag{3.44}$$

Esto es,

$$\alpha(t) = -\frac{2a}{T}t + a \tag{3.45}$$

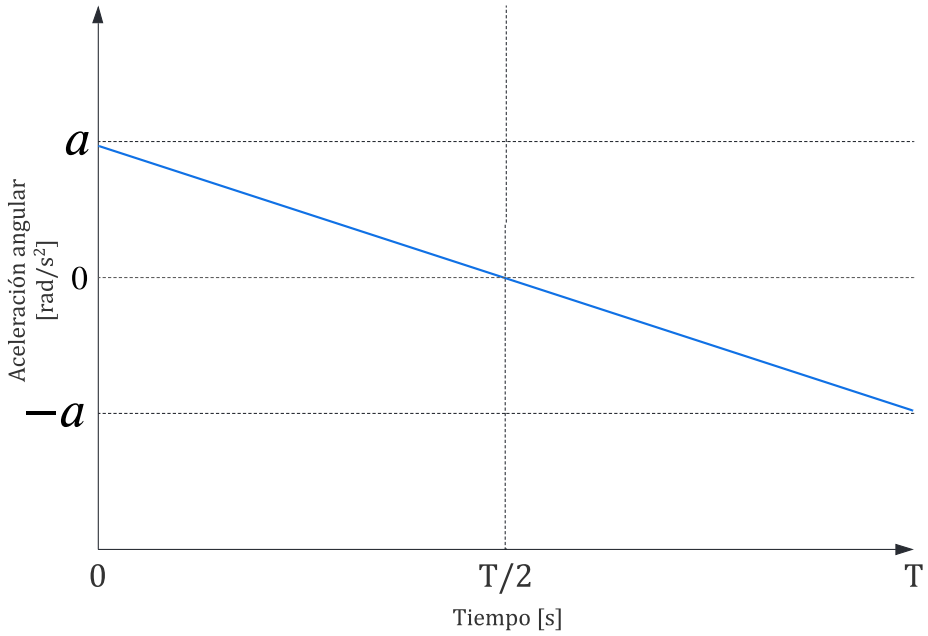

Figura 12: Función de aceleración.

Se asume, que $\omega(0) = 0$, la velocidad angular se obtiene como

$$\omega(t) = \int_0^t d\omega(s) + \omega(0) = -\frac{2a}{T}\int_0^t sds + a\int_0^t ds = -\frac{a}{T}t^2 + at \tag{3.46}$$

La velocidad máxima es obtenida en $t = T/2$, desde $\alpha(T/2) = 0$ y $m < 0$,

$$\omega_{max} = \omega\left(\frac{T}{2}\right) = \frac{1}{4}aT \tag{3.47}$$

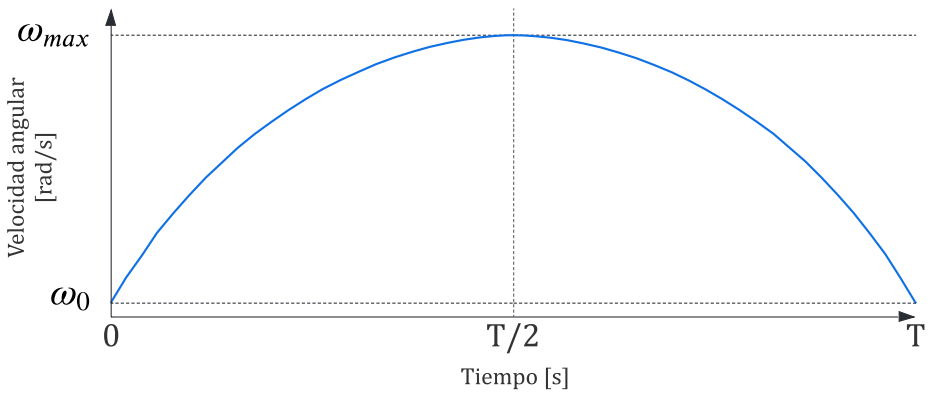

Figura 13: Función de velocidad.

De forma similar, para calcular la posición angular mostrada en Fig. 14

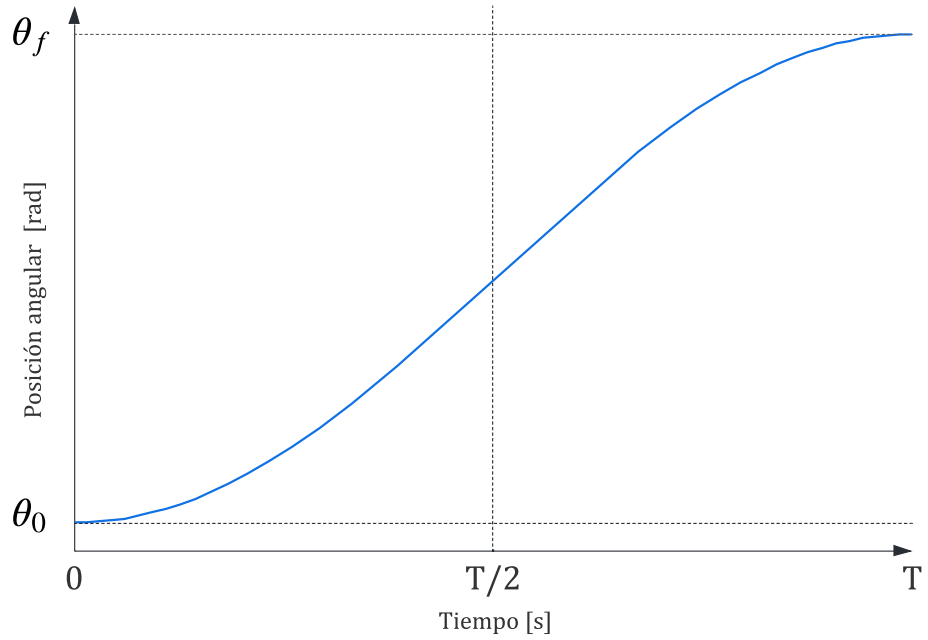

Figura 14: Función de posición.

$$\theta(t) = \int_0^t d\theta(t) + \theta(0) = -\frac{a}{T}\int_0^t s^2 ds + a\int_0^t s ds = -\frac{at^3}{3T} + \frac{at^2}{2} \tag{3.48}$$

Entonces, la posición final es

$$\theta_f = \theta(T) = \frac{1}{6}at^2 \tag{3.49}$$

Usando la ecuación (3.47),

$$\theta_f = \frac{2aT^2}{12} = \frac{2T}{3}\frac{aT}{4} = \frac{2}{3}T\omega_{max} \tag{3.50}$$

Esto es, el total de la duración del movimiento

$$T = \frac{3\theta_f}{2\omega_{max}} \tag{3.51}$$

Así mismo, resolviendo para la aceleración con la ecuación (3.47) obtenemos

$$a = \frac{4\omega_{max}}{T} \tag{3.52}$$

En términos de posición y velocidad angular

$$a = \frac{8}{3}\frac{\omega_{max}^2}{\theta_f} \tag{3.53}$$

3.2.5. Desarrollo de perfil de velocidad Polinomial

La curva de aceleración de los modelos previos no es suave del todo, las derivadas tienen discontinuidades que pueden introducir vibraciones no deseadas en los motores. Dichas vibraciones no representan un problema para el perfil en sí, sino en la vida útil del sistema físico.

Una mejora significativa sobre los modelos anteriores es vista en el modelo poli-

nomial presentado en la curva de Fig. 15.

El esquema de movimiento de este perfil, presenta en su ascenso y descenso un comportamiento parabólico , de forma puntual en los intervalos $0 < t \leq T/3$ y en $2T/3 < t \leq T$. Esto hace que la llegada al valor máximo de velocidad angular sea menos abrupta. Ahora bien, el perfil parabólico al alcanzar su máxima velocidad, inmediatamente comienza a disminuirla hasta llegar a cero, lo cual impide aprovechar la energía utilizada para alcanzar este punto, mientras que el perfil polinomial (trapezoidal modificado) la aprovecha durante $T/3 < t \leq 2T/3$. esto mostrado a continuación en Fig.15.

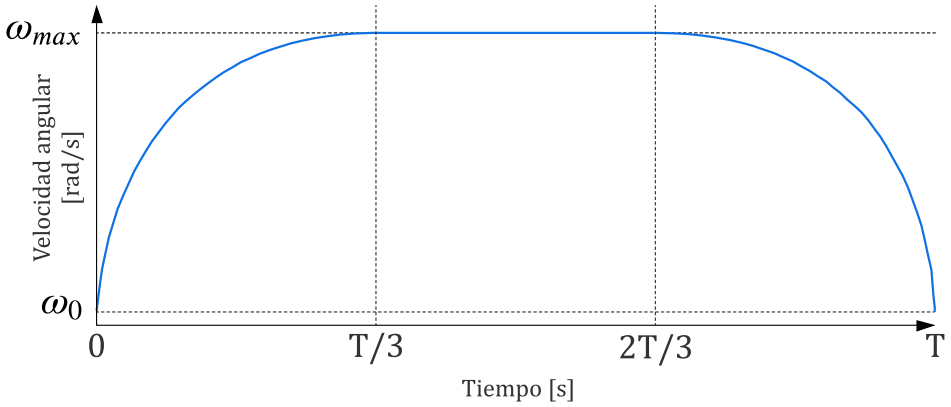

Figura 15: Función de velocidad.

La curva de aceleración Fig. 16 esta dada por

$$\alpha(t) = \begin{cases} \frac{-3a}{T}t + a, & 0 \leq t \leq \frac{T}{3} \\ 0, & \frac{T}{3} < t \leq \frac{2T}{3} \\ \frac{-3a}{T}t + 2a, & \frac{2T}{3} < t \leq T \end{cases} \tag{3.54}$$

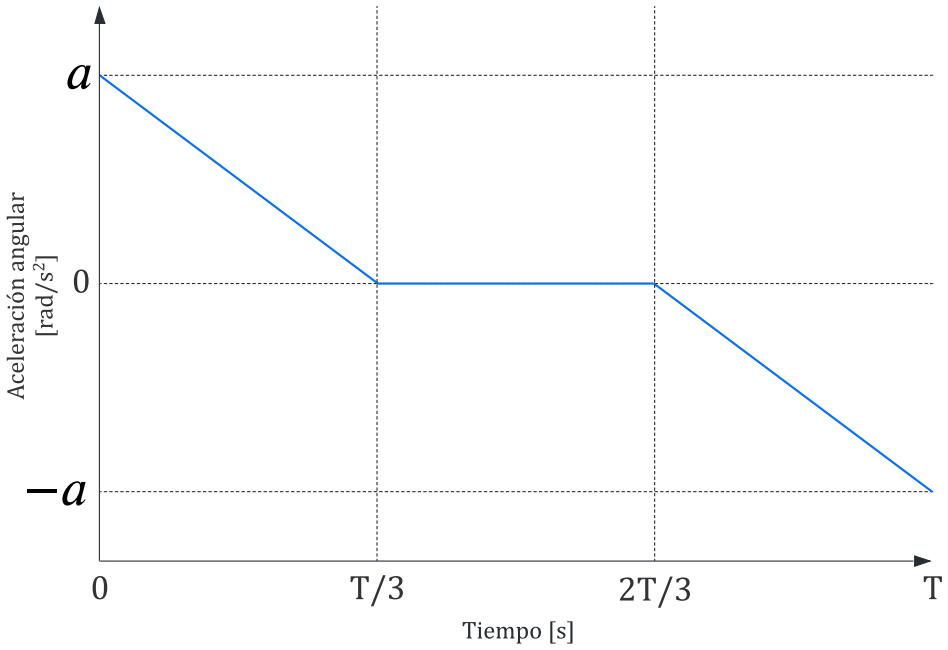

Figura 16: Función de aceleración.

Asumiendo que $\omega(0) = 0$, para $0 \leq t \leq T/3$, esto es

$$\omega(t) = \int_0^t d\omega(s) + \omega(0) = \frac{-3a}{T} \int_0^t s + ads + 0 = \frac{-3a}{2T} t^2 + at \tag{3.55}$$

La máxima velocidad es alcanzada en $t = T/3$, esto es

$$\omega_{max} = \left(\frac{T}{3}\right) = \frac{1}{6} aT \tag{3.56}$$

Para el intervalo $T/3 < t \leq 2T/3$ la velocidad permanece constante

$$\omega(t) = \int_{T/3}^{t} d\omega(s) + \omega\left(\frac{T}{3}\right) = \int_{T/3}^{t} 0ds + \frac{aT}{6} = \frac{1}{6}aT \tag{3.57}$$

Finalmente, para $2T/3 < t \leq T$ se desarrolla

$$\begin{aligned}\omega(t) &= \int_{2T/3}^{t} d\omega(s) + \omega\left(\frac{2T}{3}\right) \\ &= \frac{-3a}{T}\int_{2T/3}^{t} sds + 2a\int_{2T/3}^{t} ds + \omega\left(\frac{T}{3}\right)\end{aligned}$$

Obteniendo

$$\omega(t) = \frac{-3a}{2T}t^2 + 2at - \frac{1}{2}aT \tag{3.58}$$

Siendo para los tres intervalos,

$$\omega(t) = \begin{cases} \frac{-3a}{2T}t^2 + at, & 0 \leq t \leq \frac{T}{3} \\ \frac{1}{6}aT, & \frac{T}{3} < t \leq \frac{2T}{3} \\ \frac{-3a}{2T}t^2 + 2at - \frac{1}{2}aT, & \frac{2T}{3} < t \leq T \end{cases} \tag{3.59}$$

Para calcular la posición angular Fig. 17, se realiza el procedimiento para cada intervalo.

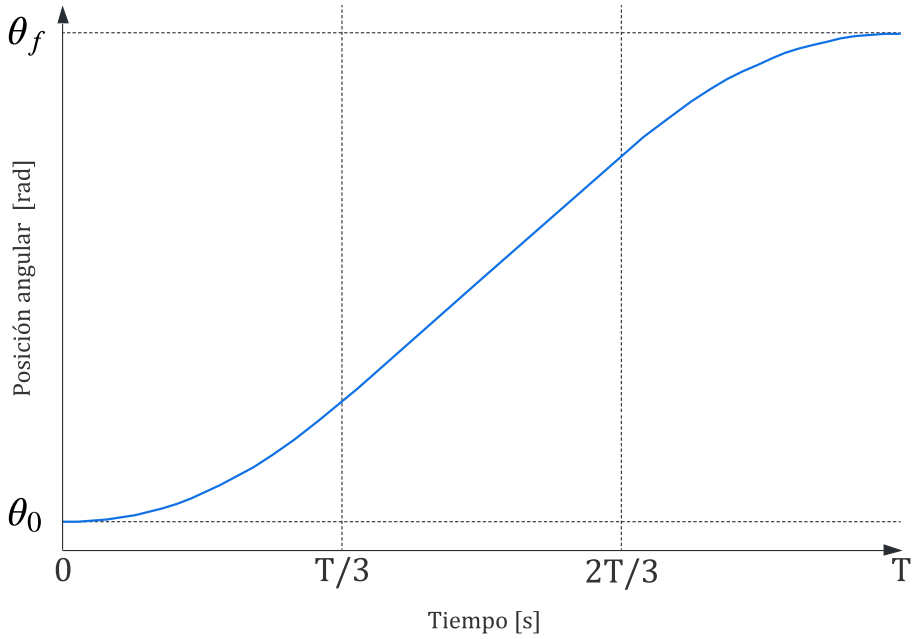

Figura 17: Función de posición.

Para $0 \leq t \leq T/3$, se asume $\theta(0) = 0$,

$$\theta(t) = \int_0^t d\theta(s) + \theta(0) = \frac{-3a}{2T} \int_0^t s^2 + asds + 0 = -\frac{a}{2T}t^3 + \frac{a}{2}t^2 \tag{3.60}$$

Para $t = T/3$,

$$\theta(t) = \frac{2}{54}aT^2 \tag{3.61}$$

Para $T/3 < t \leq 2T/3$,

$$\begin{aligned}\theta(t) &= \int_{T/3}^{t} d\theta(s) + \theta\left(\frac{T}{3}\right) \\ &= \frac{1}{6}aT\int_{T/3}^{t} ds + \frac{2}{54}aT^2\end{aligned}$$

Se obtiene

$$\theta(t) = \frac{1}{6}aTt - \frac{1}{54}aT^2 \tag{3.62}$$

Finalmente, para $2t/3 < t \leq T$,

$$\begin{aligned}\theta(t) &= \int_{2T/3}^{t} d\theta(s) + \theta\left(\frac{2T}{3}\right) \\ &= \frac{-3a}{2T}\int_{2T/3}^{t} s^2 ds + 2a\int_{2T/3}^{t} sds - \frac{1}{2}aT\int_{2T/3}^{t} sds + \frac{5}{54}aT^2\end{aligned}$$

Se obtiene

$$\theta(t) = -\frac{a}{2T}t^3 + at^2 - \frac{1}{2}aTt + \frac{7}{54}aT^2 \tag{3.63}$$

Después, la posición final θ_f. Esto es, la posición cuando $t = T$, es dado por

$$\theta_f = \theta(T) = \frac{7}{54}aT^2 \tag{3.64}$$

En donde se sustituye la ecuación (3.56),

$$\theta_f = \frac{7T}{9} \cdot \frac{aT}{6} = \frac{7}{9} T\omega_{max} \tag{3.65}$$

Así,

$$T = \frac{9\theta_f}{7\omega_{max}} \tag{3.66}$$

De la misma forma, de la ecuación (3.56), implica

$$a = \frac{6\omega_{max}}{T} \tag{3.67}$$

En términos de posición y velocidad angular

$$a = \frac{14}{3} \frac{\omega_{max}^2}{\theta_f} \tag{3.68}$$

En el siguiente apartado se muestra un sumario de cada unos de los perfiles de velocidad que son usados en este documento, Triangular, Trapezoidal, Parabólico y Polinomial.

3.3. Sumario de las ecuaciones de los perfiles de velocidad

Con el objetivo de mostrar las ecuaciones que son relevantes se realiza una tabla para cada perfil de velocidad.

Ecuaciones de perfil Triangular

Tabla 1: Velocidad angular de perfil triangular.

Intervalo	$\alpha(t)$	**Función**	$\omega(t)$
$0 < t \leq \frac{T}{2}$	$+a$	$\omega(t) = \int_0^t d\alpha(s)ds + \omega(0)$	$\omega(t) = at, \omega(0) = 0$
$\frac{T}{2} < t \leq T$	$-a$	$\omega(t) = \int_{T/2}^t \alpha(s)ds + \frac{aT}{2}$	$\omega(t) = -at + aT$

Tabla 2: Posición angular de perfil triangular.

Intervalo	$\omega(t)$	**Función**	$\theta(t)$
$0 < t \leq \frac{T}{2}$	at	$\theta(t) = a\int_0^t sds + \theta(0)$	$\theta(0) = 0, \theta(t) = \frac{at^2}{2}$
$\frac{T}{2} < t \leq T$	$-at + aT$	$\theta(t) = -a\int_{T/2}^t sds + aT\int_{T/2}^t ds + \frac{aT^2}{8}$	$\theta(t) = -\frac{at^2}{2} + aTt - \frac{aT^2}{4}$

Ecuaciones del perfil de velocidad Trapezoidal

Tabla 3: Velocidad angular del perfil trapezoidal.

Intervalo	$\alpha(t)$	**Función**	$\omega(t)$
$0 < t \leq \frac{T}{3}$	$+a$	$\omega(t) = \int_0^t d\omega(s) + \omega(0)$	$\omega(0) = 0, \omega(t) = at$
$\frac{T}{3} < t \leq \frac{2T}{3}$	0	$\omega(t) = \int_{T/3}^t d\omega(s) + \omega\left(\frac{T}{3}\right)$	$\omega(t) = \frac{aT}{3}$
$\frac{2T}{3} < t \leq T$	$-a$	$\omega(t) = \int_{2T/3}^t d\omega(s) + \omega\left(\frac{2T}{3}\right)$	$\omega(t) = -at + aT$

Tabla 4: Posición angular del perfil trapezoidal.

Intervalo	$\omega(t)$	**Función**	$\theta(t)$
$0 < t \leq \frac{T}{3}$	$+at$	$\theta(t) = \int_0^t d\theta(s) + \theta(0)$	$\theta(0) = 0, \theta(t) = \frac{at^2}{2}$
$\frac{T}{3} < t \leq \frac{2T}{3}$	$\frac{aT}{3}$	$\theta(t) = \int_{T/3}^t d\theta(s) + \theta\left(\frac{T}{3}\right)$	$\theta(t) = \frac{aT}{3}t - \frac{aT^2}{18}$
$\frac{2T}{3} < t \leq T$	$-at + aT$	$\theta(t) = \int_{2T/3}^t d\theta(s) + \theta\left(\frac{2T}{3}\right)$	$\theta(t) = -\frac{a}{2}t^2 + aTt - \frac{5aT^2}{18}$

Ecuaciones del perfil de velocidad Parabólico

Tabla 5: Velocidad angular del perfil Parabólico.

Intervalo	$\alpha(t)$	**Función**	$\omega(t)$
$0 < t \leq T$	$-\frac{2a}{T} + a$	$\omega(t) = \int_0^t d\omega(s) + \omega(0)$	$\omega(0) = 0, \omega(t) = -\frac{a}{T}t^2 + at$

Tabla 6: Posición angular del perfil Parabólico.

Intervalo	$\omega(t)$	**Función**	$\theta(t)$
$0 < t \leq T$	$-\frac{a}{T}t^2 + at$	$\theta(t) = \int_0^t d\theta(t) + \theta(0)$	$\theta(0) = 0, \theta(t) = -\frac{at^3}{3T} + \frac{at^2}{2}$

Ecuaciones de perfil de velocidad polinomial

Tabla 7: Velocidad angular del perfil polinomial.

Intervalo	$\alpha(t)$	**Función**	$\omega(t)$
$0 < t \leq \frac{T}{3}$	$\frac{-3a}{T}t + a$	$\omega(t) = \int_0^t d\omega(s) + \omega(0)$	$\omega(0) = 0, \omega(t) = \frac{-3a}{2T}t^2 + at$
$\frac{T}{3} < t \leq \frac{2T}{3}$	0	$\omega(t) = \int_{T/3}^t d\omega(s) + \omega\left(\frac{T}{3}\right)$	$\omega(t) = \frac{1}{6}aT$
$\frac{2T}{3} < t \leq T$	$\frac{-3a}{T}t + 2a$	$\omega(t) = \int_{2T/3}^t d\omega(s) + \omega\left(\frac{2T}{3}\right)$	$\omega(t) = \frac{-3a}{2T}t^2 + 2at - \frac{1}{2}aT$

Tabla 8: Posición angular del perfil polinomial.

Intervalo	$\omega(t)$	**Función**	$\theta(t)$
$0 < t \leq \frac{T}{3}$	$\frac{-3a}{2T}t^2 + at$	$\theta(t) = \int_0^t d\theta(s) + \theta(0)$	$\theta(0) = 0, \theta(t) = -\frac{a}{2T}t^3 + \frac{a}{2}t^2$
$\frac{T}{3} < t \leq \frac{2T}{3}$	$\frac{1}{6}aT$	$\theta(t) = \int_{T/3}^t d\theta(s) + \theta\left(\frac{T}{3}\right)$	$\theta(t) = \frac{1}{6}aTt - \frac{1}{54}aT^2$
$\frac{2T}{3} < t \leq T$	$\frac{-3a}{2T}t^2 + 2at - \frac{1}{2}aT$	$\theta(t) = \int_{2T/3}^t d\theta(s) + \theta\left(\frac{2T}{3}\right)$	$\theta(t) = -\frac{a}{2T}t^3 + at^2 - \frac{1}{2}aTt + \frac{7}{54}aT^2$

3.4. Energía

En un entorno industrial donde el uso de motores es más que frecuente, la optimización de la energía es indispensable. Realizar tareas que utilicen el movimiento generado por motores, desde la perspectiva de el ahorro de energía, debe ser contemplada la búsqueda de soluciones que aporten positivamente al impacto ambiental.

Una solución de acuerdo a [16], es el uso de metodologías para la reprogramación de trayectorias,que representa una optimización de consumo energético en Robots ABB. A su vez, [20] menciona acerca de eficiencia en el uso de energía debido a la suavidad de los perfiles de movimiento así como el *Jerk* asociado al desempeño. Adicionalmente [20], presenta modificaciones a un perfil de velocidad trapezoidal observando un menor consumo en un primer acercamiento.

Es sabido que dado un trayecto definido, la velocidad y aceleración es lo que influye directamente en el consumo energético en motores DC. El concepto de potencia es relacionado directamente a la energía usada por un sistema, siendo la potencia es la energía utilizada en un instante especifico de tiempo para efectuar un trabajo, mientras que, la energía será la sumatoria de cada unidad de potencia en un intervalo de tiempo.

Partiendo de un sistema electromecánico y sabiendo que la energía consumida esta ligada al comportamiento de la aceleración, se tiene dos leyes fundamentales para modelar las ecuaciones de energía, estas son:

Ley de Ohm,

$$I = \frac{V}{R} \tag{3.69}$$

Segunda ley de Newton aplicada a la rotación [11],

$$\tau = J_m \alpha \tag{3.70}$$

Se tiene la ecuación de potencia eléctrica instantánea como

$$W(t) = I^2(t)R \tag{3.71}$$

Y sustituyendo (3.69) en (3.71) se observa que

$$W(t) = I^2(t)R \tag{3.72}$$

Sabiendo que la energía es la integral de la potencia se tiene que

$$E(t) = \int_0^t W(\tau)d\tau = \int_0^t I^2(\tau)Rd\tau \tag{3.73}$$

Por otro lado, al desarrollar la ecuación (3.70) el par generado en el sistema en términos de velocidad se tiene que

$$\tau_g = J_m \frac{d\omega(t)}{dt} + \tau_b \tag{3.74}$$

De la ecuación 3.4 del modelo de un motor DC tenemos la corriente

$$I(t) = \frac{\tau_g}{K_t} \tag{3.75}$$

Sustituyendo (3.75) en (3.73) se obtiene

$$E(t) = \frac{R}{K_t^2} \int_0^T \tau^2(\tau) d\tau \tag{3.76}$$

Sustituyendo (3.74) en (3.76) se establece la ecuación de energía

$$E(t) = \frac{R}{K_t^2} \int_0^T \left[J_m \frac{d\omega(\tau)}{d\tau} + \tau_b \right]^2 d\tau \tag{3.77}$$

La ecuación (3.77) se desarrolla de la siguiente forma

$$E(t) = \frac{RJ_m^2}{K_t^2} \int_0^T \left[\frac{d\omega(\tau)}{d\tau} \right]^2 d\tau + \frac{2RJ_m\tau_b}{K_t^2} \int_0^T \left[\frac{d\omega(\tau)}{d\tau} \right] d\tau + \frac{R\tau_b^2}{K_t^2} \int_0^T d\tau \tag{3.78}$$

La ecuación (3.78) sugiere que son tres componentes de energía.

$$E_T = E_1 + E_2 + E_3 \tag{3.79}$$

Siendo

$$E_1 = \frac{RJ_m^2}{K_t^2} \int_0^t \left[\frac{d\omega(\tau)}{d\tau} \right]^2 d\tau \tag{3.80}$$

$$E_2 = \frac{2RJ_m\tau_b}{K_t^2} \int_0^t \left[\frac{d\omega(\tau)}{d\tau} \right] d\tau \tag{3.81}$$

$$E_3 = \frac{R\tau_b^2}{K_t^2} \int_0^t d\tau \tag{3.82}$$

Al analizar las ecuaciones de energía presentadas se observa que existe sólo una forma en que el usuario puede influir en el consumo de energía sin hacer cambios en los componentes del sistema (motor). Las razones son que, debido a que la ecuación (3.81) de E_2 depende de la aceleración en el tiempo 0 y en T lo cual implica que su valor es cero. Mientras que la ecuación (3.82) de E_3 indica que su valor es constante al ser valores no modificables al tratarse de parámetros intrínsecos de la construcción del motor.

Por lo tanto, la ecuación que proporciona una evaluación del modelo de energía de los perfiles de velocidad es la ecuación (3.80) de E_1. Esto es porque depende de la función de aceleración, parámetro que tendrá variaciones en cada perfil.En posteriores cálculos de energía, E_1 será referida únicamente como E.

3.4.1. Energía de perfil triangular

Partiendo de la ecuación de E (3.80) sabemos que la energía depende de la integración de la derivada de la velocidad, esto es, si sabemos el valor de los datos de la velocidad en un intervalo de tiempo dado, podremos establecer parámetros que nos proporcionen la información deseada. Sin embargo, estos valores tendrán que ser obtenidos por medio de la relación de las ecuaciones de la función de aceleración, velocidad y posición de este perfil.

De condiciones nulas y las ecuaciones (3.16), (3.25) y (3.26) se opta por obtener los intervalos de la velocidad angular en términos de posición y el periodo utilizado, de forma que

La aceleración a en términos de posición angular

$$a = 4\frac{\theta_f}{T^2} \tag{3.83}$$

Así los intervalos de la velocidad angular son

$$\omega(t) = \begin{cases} \frac{4\theta_f}{T^2}t & 0 < t \leq \frac{T}{2} \\ \frac{4\theta_f}{T^2}(T-t) & \frac{T}{2} < t \leq T \end{cases} \tag{3.84}$$

Intervalos de su derivada

$$\frac{d\omega(t)}{dt} = \begin{cases} \frac{4\theta_f}{T^2} & 0 < t \leq \frac{T}{2} \\ \frac{4\theta_f}{T^2} & \frac{T}{2} < t \leq T \end{cases} \tag{3.85}$$

Elevando al cuadrado la ecuación (3.85)

$$\left[\frac{d\omega(t)}{dt}\right]^2 = \begin{cases} \frac{16\theta_f^2}{T^4} & 0 < t \leq \frac{T}{2} \\ \frac{16\theta_f^2}{T^4} & \frac{T}{2} < t \leq T \end{cases} \tag{3.86}$$

Finalmente sustituyendo la (3.86) en (3.80)

$$E = \frac{RJ_m^2}{K_t^2}\frac{16\theta_f^2}{T^4}\left(\int_0^{T/2} dt + \int_{T/2}^{T} dt\right) \tag{3.87}$$

Resolviendo se obtiene la energía consumida por el perfil triangular

$$E = 16\frac{RJ_m^2\theta_f^2}{K_t^2T^3} \tag{3.88}$$

Recordando que del motor

- J_m es la inercia.
- R es la resistencia.
- k_t es la constante de par.

Los cuales son parámetros intrínsecos del sistema, mientras que la posición final θ_f y el periodo T son datos propuestos.

3.4.2. Energía de perfil trapezoidal

En similitud con el perfil anterior, se usa la relación de las ecuaciones de la función de aceleración, velocidad y posición, teniendo en cuenta las ecuaciones (3.31), (3.38) y (3.40), se plantea

La aceleración a en términos de posición angular

$$a = \frac{9}{2}\frac{\theta_f}{T^2} \tag{3.89}$$

Los intervalos de la velocidad angular en términos de la posición angular son

$$\omega(t) = \begin{cases} +\frac{9\theta_f}{2T^2}t, & 0 \leq t \leq \frac{T}{3} \\ \frac{3\theta_f}{2T}, & \frac{T}{3} < t \leq \frac{2T}{3} \\ -\frac{9\theta_f}{2T^2}t + \frac{9\theta_f}{2T^2}T, & \frac{2T}{3} < t \leq T \end{cases} \tag{3.90}$$

Intervalos de su derivada

$$\frac{d\omega(t)}{dt} = \begin{cases} +\frac{9\theta_f}{2T^2}, & 0 \leq t \leq \frac{T}{3} \\ 0, & \frac{T}{3} < t \leq \frac{2T}{3} \\ -\frac{9\theta_f}{2T^2}, & \frac{2T}{3} < t \leq T \end{cases} \tag{3.91}$$

Elevando al cuadrado la ecuación (3.91)

$$\left[\frac{d\omega(t)}{dt}\right]^2 = \begin{cases} +\frac{81\theta_f^2}{4T^4}, & 0 \leq t \leq \frac{T}{3} \\ 0, & \frac{T}{3} < t \leq \frac{2T}{3} \\ +\frac{81\theta_f^2}{4T^4}, & \frac{2T}{3} < t \leq T \end{cases} \tag{3.92}$$

Finalmente sustituyendo la (3.92) en (3.80)

$$E = \frac{RJ_m^2}{K_t^2}\frac{81\theta_f^2}{4T^4}\left(\int_0^{T/3} dt + \int_{2T/3}^{T} dt\right) \tag{3.93}$$

Resolviendo se obtiene la energía consumida por el perfil trapezoidal

$$E = \left(\frac{27}{2}\right)\frac{RJ_m^2\theta_f^2}{K_t^2T^3} \tag{3.94}$$

3.4.3. Energía de perfil parabólico

Teniendo en cuenta las ecuaciones (3.47), (3.50) y (3.51), se plantea

La aceleración a en términos de posición angular

$$a = 6\frac{\theta_f}{T^2} \tag{3.95}$$

Velocidad angular en términos de la posición angular son

$$\omega(t) = -\frac{6\theta_f}{T^3}t^2 + \frac{6\theta_f}{T^2}t \tag{3.96}$$

Su derivada

$$\frac{d\omega(t)}{dt} = -\frac{12\theta_f}{T^3}t + \frac{6\theta_f}{T^2} \tag{3.97}$$

Elevando al cuadrado la ecuación (3.97)

$$\left[\frac{d\omega(t)}{dt}\right]^2 = \frac{144\theta_f^2}{T^6}t^2 - \frac{144\theta_f^2}{T^5}t + \frac{36\theta_f^2}{T^4} \tag{3.98}$$

Finalmente sustituyendo la (3.98) en (3.80)

$$E = \frac{RJ_m^2}{K_t^2}\left(\frac{144\theta_f^2}{T^6}\int_0^T t^2dt - \frac{144\theta_f^2}{T^5}\int_0^T tdt + \frac{36\theta_f^2}{T^4}\int_0^T dt\right) \tag{3.99}$$

Resolviendo se obtiene la energía consumida por el perfil parabólico

$$E = 12\frac{RJ_m^2\theta_f^2}{K_t^2T^3} \tag{3.100}$$

3.4.4. Energía de perfil polinomial

Teniendo en cuenta las ecuaciones (3.55), (3.60) y (3.64), se plantea

La aceleración a en términos de posición angular

$$a = \frac{54}{7}\frac{\theta_f}{T^2} \tag{3.101}$$

Los intervalos de la velocidad angular en términos de la posición angular son

$$\omega(t) = \begin{cases} -\frac{3}{2T}\left(\frac{54}{7}\frac{\theta_f}{T^2}\right)t^2 + \left(\frac{54}{7}\frac{\theta_f}{T^2}\right)t, & 0 \le t \le \frac{T}{3} \\ \frac{1}{6}\left(\frac{54}{7}\frac{\theta_f}{T^2}\right)T, & \frac{T}{3} < t \le \frac{2T}{3} \\ -\frac{3}{2T}\left(\frac{54}{7}\frac{\theta_f}{T^2}\right)t^2 + 2\left(\frac{54}{7}\frac{\theta_f}{T^2}\right)t - \frac{1}{2}\left(\frac{54}{7}\frac{\theta_f}{T^2}\right)T, & \frac{2T}{3} < t \le T \end{cases} \tag{3.102}$$

Intervalos de su derivada

$$\frac{d\omega(t)}{dt} = \begin{cases} -\frac{162\theta_f}{7T^3}t + \frac{54}{7}\frac{\theta_f}{T^2}, & 0 \le t \le \frac{T}{3} \\ 0, & \frac{T}{3} < t \le \frac{2T}{3} \\ -\frac{162\theta_f}{7T^3}t + \frac{108}{7}\frac{\theta_f}{T^2}, & \frac{2T}{3} < t \le T \end{cases} \tag{3.103}$$

Elevando al cuadrado la ecuación (3.103)

$$\left[\frac{d\omega(t)}{dt}\right]^2 = \begin{cases} \frac{26244}{49}\frac{\theta_f^2}{T^6}t^2 - \frac{17496}{49}\frac{\theta_f^2}{T^5}t + \frac{2916}{49}\frac{\theta_f^2}{T^4}, & 0 \leq t \leq \frac{T}{3} \\ 0, & \frac{T}{3} < t \leq \frac{2T}{3} \\ \frac{26244}{49}\frac{\theta_f^2}{T^6}t^2 - \frac{34992}{49}\frac{\theta_f^2}{T^5}t + \frac{11664}{49}\frac{\theta_f^2}{T^4}\frac{2T}{3} < t \leq T \end{cases} \tag{3.104}$$

Finalmente sustituyendo la (3.104) en (3.80)

$$E = \frac{RJ_m^2}{K_t^2}\left(\int_0^{T/3}\left[\frac{d\omega(t)}{dt}\right]^2 dt + \int_{2T/3}^{T}\left[\frac{d\omega(t)}{dt}\right]^2 dt\right) \tag{3.105}$$

Resolviendo se obtiene la energía consumida por el perfil trapezoidal

$$E = \left(\frac{648}{49}\right)\frac{RJ_m^2\theta_f^2}{K_t^2T^3} \tag{3.106}$$

3.4.5. Comparación de la energía entre perfiles de velocidad

De acuerdo a los datos del motor proporcionados por el fabricante

- $J_m = 6.99x10^{-}6\ Kgm^2$
- $R = 1.11\ \Omega$
- $k_t = 36.4\ mNm/A$

y proponiendo los siguientes datos para la experimentación

- $\theta_f = 104.72rad \quad (40000 \quad cuentas \quad de \quad encoder)$
- $T = 2s$

Y sustituyendo los valores numéricos en las ecuaciones resultantes de Energía consumida por cada perfil (3.88),(3.94), (3.100) y (3.106) se tiene

Consumo energético teórico por el perfil triangular

$$E = 16\frac{RJ_m^2\theta_f^2}{K_t^2T^3} = 8.98e-10 \quad Joules \tag{3.107}$$

Consumo energético teórico por el perfil trapezoidal

$$E = \left(\frac{27}{2}\right)\frac{RJ_m^2\theta_f^2}{K_t^2T^3} = 7.57e-10 \quad Joules \tag{3.108}$$

Consumo energético teórico por el perfil parabólico

$$E = 12\frac{RJ_m^2\theta_f^2}{K_t^2T^3} = 6.73e-10 \quad Joules \tag{3.109}$$

Consumo energético teórico por el perfil polinomial

$$E = \left(\frac{648}{49}\right)\frac{RJ_m^2\theta_f^2}{K_t^2T^3} = 7.42e-10 \quad Joules \tag{3.110}$$

Los datos obtenidos nos indican que el perfil que presenta un menor nivel de consumo energético es el parabólico, partiendo de este dato los siguientes perfiles muestran un consumo mayor en el cálculo presentado, siendo el triangular 25 %, trapezoidal 11.11 % y el polinomial un 9.26 % mayor en consumo. Esto es, independientemente de la posición final deseada y del tiempo necesario para alcanzar la misma, los resultados de consumo permanecerán en esa proporción.

3.5. Control de posición

El control de posición es una importante herramienta que se utiliza en la generación de perfiles de velocidad. La razón de ello, es la construcción por etapas de la curva del perfil. Cada etapa, se ejecuta mediante el proceso de alcance de las posiciones deseadas a través del controlador. Esto es mostrado en la Fig. 18

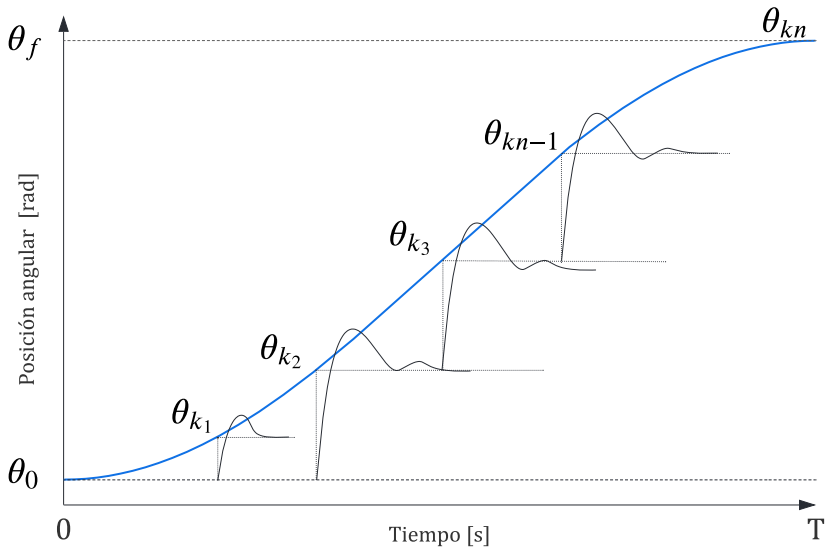

Figura 18: Etapas de curva de perfil de movimiento.

Se sugiere que, para obtener una curva de perfil de movimiento óptima, es necesario observar adecuadamente el comportamiento de la respuesta del controlador de posición. Esto es porque se controla la forma en que se comporta el movimiento del motor hasta llegar a la posición deseada, por lo tanto, la implementación y análisis de la respuesta de ambos es adecuado para alcanzar el objetivo de este documento.

De teoría de control moderno [15], se menciona que para obtener resultados aceptables, es necesario encontrar el modelo matemático de la planta, posteriormente, utilizar un método para el análisis del sistema. En este documento se implementa el método de identificación por medio de la respuesta transitoria, basado en un sistema de segundo orden subamortiguado.

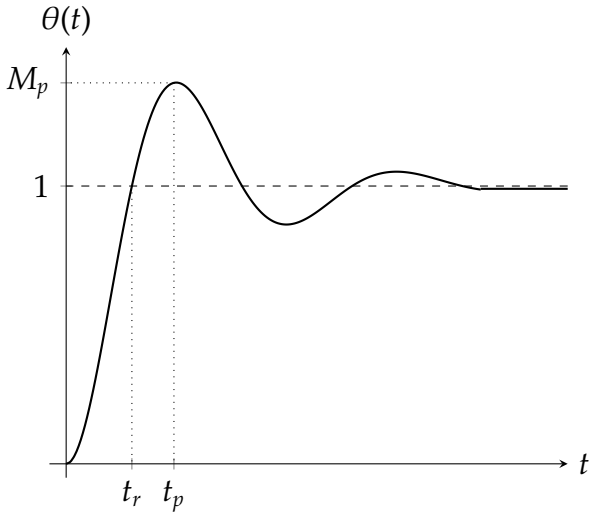

Figura 19: Respuesta de sistema de segundo orden subamortiguado.

Un sistema subamortiguado de segundo orden muestra características de interés que permiten modelar un sistema físico, donde parámetros obtenidos de la respuesta transitoria se relacionan con los valores de entrada deseados.

En Fig. 19 se muestran los parámetros de interés para un sistema de este tipo, donde

- t_r es el tiempo de respuesta en alcanzar el valor deseado.
- t_p es el tiempo de pico máximo de la respuesta al escalón unitario.
- M_p es el sobreimpulso de la respuesta al escalón unitario.

Es sabido que estos parámetros de respuesta transitoria tienen una relación directa en términos de amortiguamiento ζ y la frecuencia natural ω_n del sistema.

Sabiendo esto, se puede aplicar un controlador proporcional para obtener dichos valores de los parámetros en la planta.

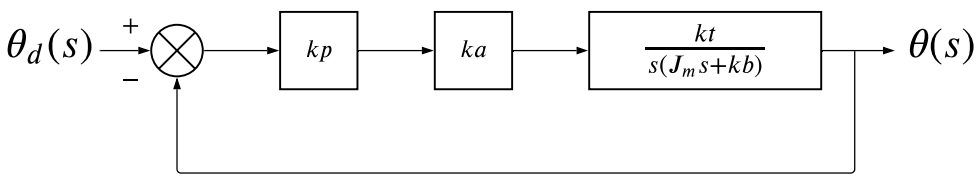

Figura 20: Sistema en lazo cerrado con controlador proporcional.

4. Metodología

4.1. Modelado

De forma general se cuenta con el sistema Fig.21

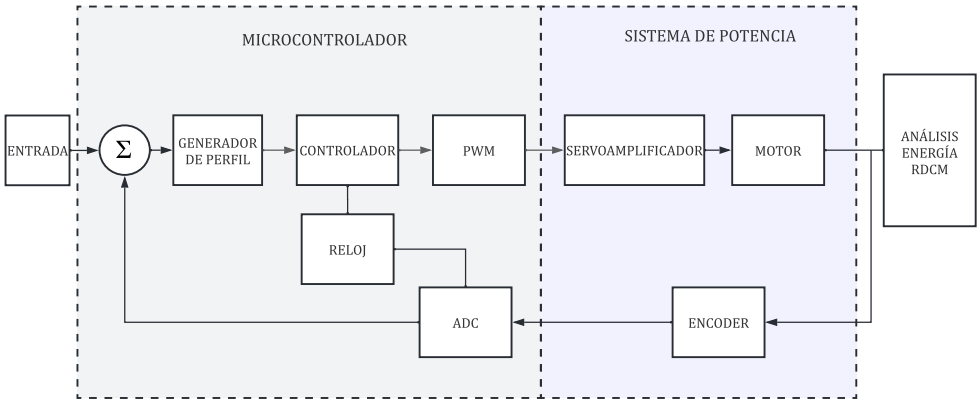

Figura 21: Bloques del sistema.

Un modelo de control exacto de un sistema puede ser sustituido por una aproximación con un modelo simplificado, permitiendo una implementación más rápida, con resultados confiables y la reducción de trabajo de computo [10].

Como se planteó, la identificación de la planta se realiza encontrando las relaciones del comportamiento en el transitorio del sistema aplicando un controlador proporcional mostrado en Fig. 26

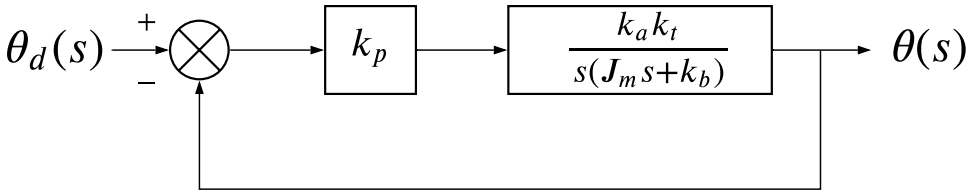

Figura 22: Lazo cerrado de controlador proporcional.

Y utilizando las ecuaciones que propone [14].

$$\zeta = \frac{|\ln(M_p)|}{\sqrt{\pi^2 + \ln^2(M_p)}} \tag{4.1}$$

$$\omega_n = \frac{\pi}{t_p \sqrt{1-\zeta^2}} \tag{4.2}$$

Y las correspondientes para obtener el tiempo de respuesta óptimo para el sistema

$$\omega_d = \omega_n \sqrt{1-\zeta^2} \tag{4.3}$$

$$\sigma = \omega_n \zeta \tag{4.4}$$

$$\beta = tan^{-}1\frac{\omega_d}{\sigma} \tag{4.5}$$

$$Tr = \frac{\pi - \beta}{\omega_d} \tag{4.6}$$

En estas ecuaciones se muestra la relación directa entre los valores que se obtienen de la respuesta al impulso, es decir M_p y T_p.

Al desarrollar el sistema presentado en Fig.26 se obtiene la siguiente función de transferencia

$$\frac{\theta(s)}{\theta_d(s)} = \frac{\frac{k_p K_a k_t}{J_m}}{s^2 + \frac{k_b}{J_m}s + \frac{k_p k_a k_t}{J_m}} \tag{4.7}$$

Es notorio que la (4.7) presenta un modelo general de un sistema de segundo orden, de forma que

$$\frac{\frac{k_p K_a k_t}{J_m}}{s^2 + \frac{k_b}{J_m}s + \frac{k_p k_a k_t}{J_m}} = \frac{k\omega_n}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{4.8}$$

De los numeradores en (4.8) se obtienen los valores de las constantes, esto es

$$2\zeta\omega_n = \frac{k_b}{J_m} \tag{4.9}$$

Y

$$\omega_n^2 = \frac{k_p k_a k_t}{J_m} \tag{4.10}$$

Se observa que tanto en (4.9) y en (4.10) mantienen una relación proporcional con $\frac{1}{J_m}$, por lo que, si le asignamos un valor unitario a J_m, entonces

$$2\zeta\omega_n = k_b \tag{4.11}$$

A su vez

$$\omega_n^2 = k_p k_a k_t \tag{4.12}$$

Se ingresa una posición deseada $\theta_d(s)$ agregando solo la ganancia proporcional. Arrojando el sistema retroalimentado una respuesta $\theta(s)$ que será evaluada.

Se utiliza el siguiente entorno de pruebas.

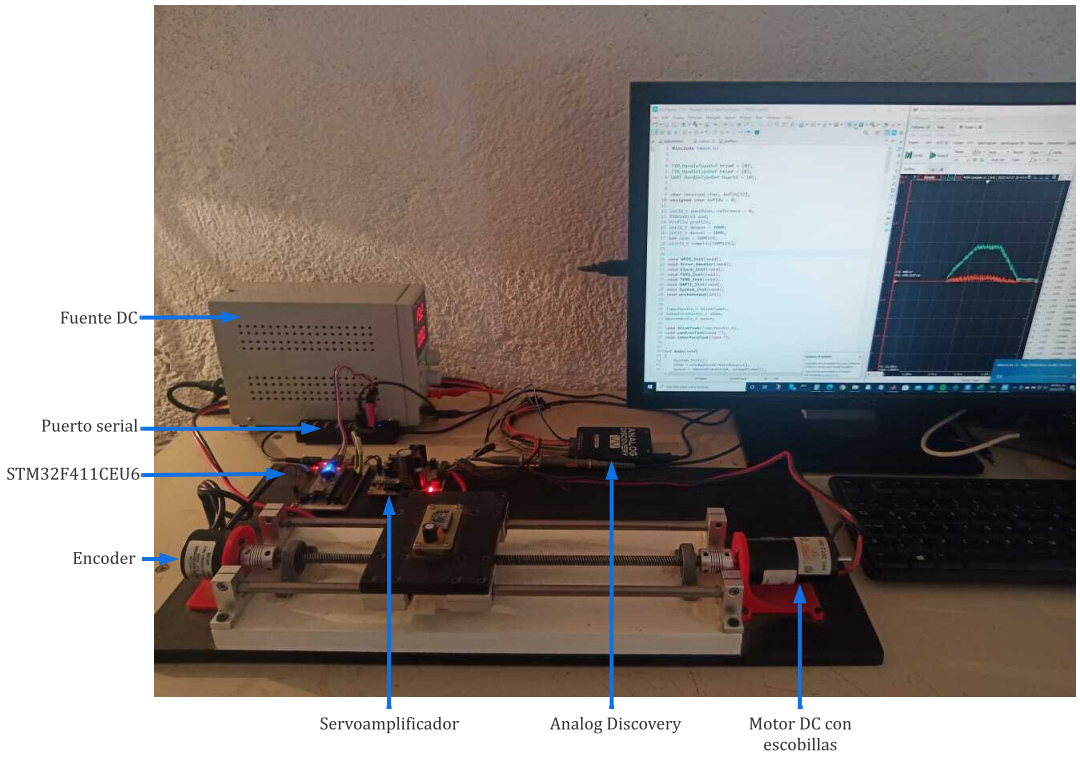

Figura 23: Plataforma de experimentación.

Para calcular parámetros de la planta se obtienen los datos por medio de la experimentación en la plataforma, los datos de posición son obtenidos mediante la lectura del puerto analógico digital, el uso de la comunicación *UART* del microcontrolador y el software de desarrollo STM32CubeIDE.

4.2. Diseño

Partiendo del diagrama a bloques del sistema de Fig. 24

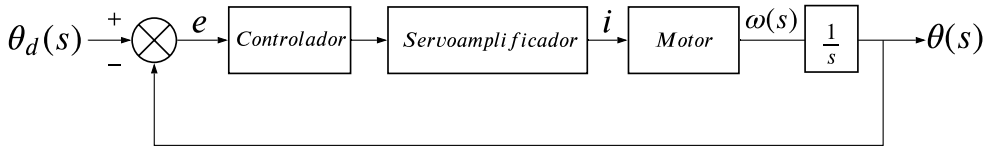

Figura 24: Sistema en lazo cerrado.

En este modelo, por medio de un servoamplificador se transduce un voltaje e en corriente i en forma lineal, donde, por medio de una tarjeta de circuito integrado se obtiene el control del flujo de corriente , siendo este componente una constante K_a de valor unitario (3.11).

La identificación antes propuesta es llevada a cabo, con una $\theta_d(s) = 1000$ cuentas de encoder, y una $k_p = 5.0$ proporcionando la siguiente información

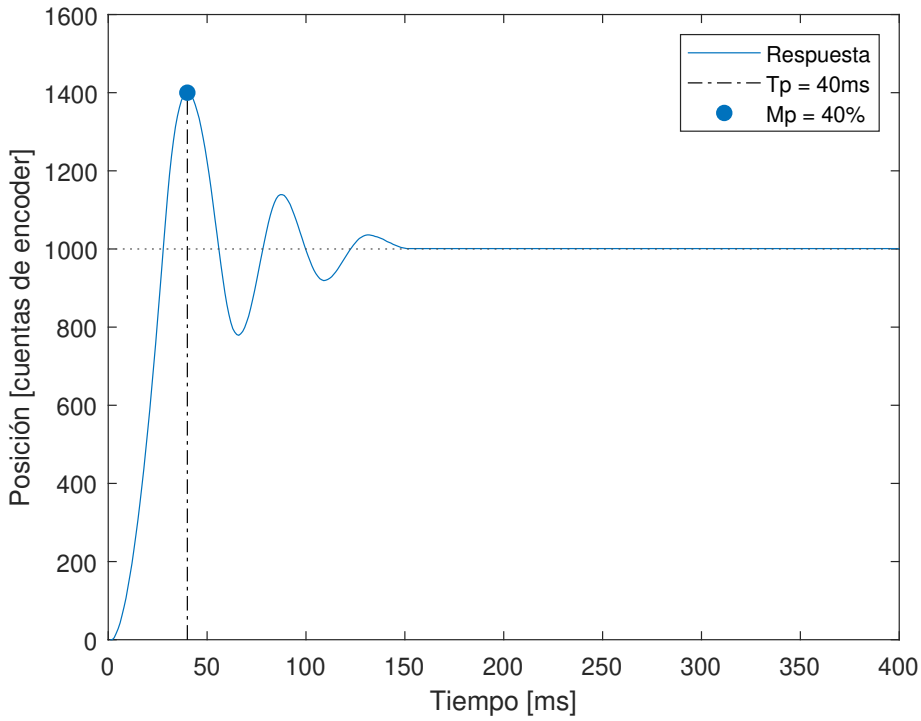

Figura 25: Respuesta al escalón de controlador proporcional.

Como se observa la respuesta de la Fig. 25 el sobrepaso $M_{pid} = 40\,\%$ mientras que $T_{pid} = 40$ ms Observando que el sistema se estabiliza aproximadamente en 150 ms.

Al utilizar las ecuaciones (4.1) y (4.2), se obtiene que $\zeta_{id} = 0.2880$ y $\omega_{nid} = 81.8123$ El amortiguamiento es relativamente bajo, por lo que será necesario aplicar un controlador diferente.

Es importante aclarar que M_{pid}, T_{pid}, ζ_{id}, ω_{nid} y k_{pid} son los parámetros de identificación de la planta para obtener las constantes siguientes

- Jm=1.0;
- $Ka = 1.0$;
- $k_b = 2\zeta_{id}\omega_{nid}$
- $k_t = \frac{\omega_{nid}{}^2}{k_{pid}}$

Con estos datos, se obtendrán los parámetros de la planta que se utilizarán el el controlador propuesto a continuación.

Controlador tipo proporcional derivativo

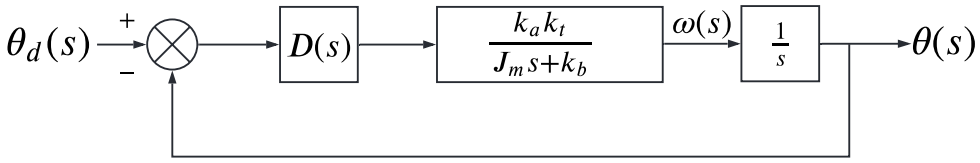

Figura 26: Lazo cerrado de controlador proporcional derivativo.

Donde,

$$D(s) = k_p\,(1 + Td(s)) \tag{4.13}$$

Considerando los criterios de diseño para lazo cerrado:

- Error cero en estado estable

  En este sistema el integrador aplicado a la velocidad angular proporciona un polo en el origen, lo cual permite el error en estado estable. Lo cual implica que las propias características del modelo incluyen esta ventaja. Siendo innecesario agregar una ganancia proporcional.

- Sobrepaso

  Se propone $M_p = 20\,\%$

- Tiempo de respuesta óptimo para el sistema de acuerdo a la ecuación (4.6)

  Se obtiene $T_r = 0.0236s$ y se propone $T_r = 0.015s$.

Para encontrar los polos de alzo cerrado se tiene

- Tiempo de muestreo $T_s = 0.001\text{s}$
- Factor de amortiguamiento

$$\zeta = \frac{|\ln(M_p)|}{\sqrt{\pi^2 + \ln^2(M_p)}} \tag{4.14}$$

- Frecuencia natural del sistema

$$\omega_n = \frac{\pi - arctan\frac{1-\zeta^2}{\zeta}}{T_r\sqrt{1-\zeta^2}} \tag{4.15}$$

Por medio del método del lugar de las raíces [15], evaluando la condición de fase se obtiene

$$\phi_1 = 180 - arctan\frac{\sqrt{(1-\zeta^2)}}{\zeta} \tag{4.16}$$

$$\phi_2 = arctan\frac{\omega_n\sqrt{(1-\zeta^2)}}{\frac{k_b}{J_m} - \zeta * \omega_n} \tag{4.17}$$

$$\phi_3 = 180 + \phi_1 + \phi_2 \tag{4.18}$$

Se tiene la ecuación de Td como

$$T_d = \frac{tan\phi_3}{\omega_n\sqrt{1-\zeta^2} + \zeta\omega tan\phi_3} \tag{4.19}$$

Ahora cumpliendo la condición de magnitud se obtiene la ganancia proporcional

$$k_p = \frac{\sqrt{k_b^2 - 2k_bJ_m\zeta\omega_n + J_m^2\omega_n^2}}{k_ak_t\sqrt{1 - 2\zeta\omega_nT_d + T_d^2\omega_n^2}} \tag{4.20}$$

Y la ganancia derivativa

$$k_d = \frac{T_dk_p}{T_s} \tag{4.21}$$

4.2.1. Validación de modelo

Por medio del software Matlab y los resultados obtenidos de la identificación, se obtienen las respuestas en simulación y experimental del control PD de posición.

El lugar geométrico de las raíces mostrado en Fig. 27 , muestra que el compor-

tamiento del sistema tiende a la estabilidad con el control propuesto.

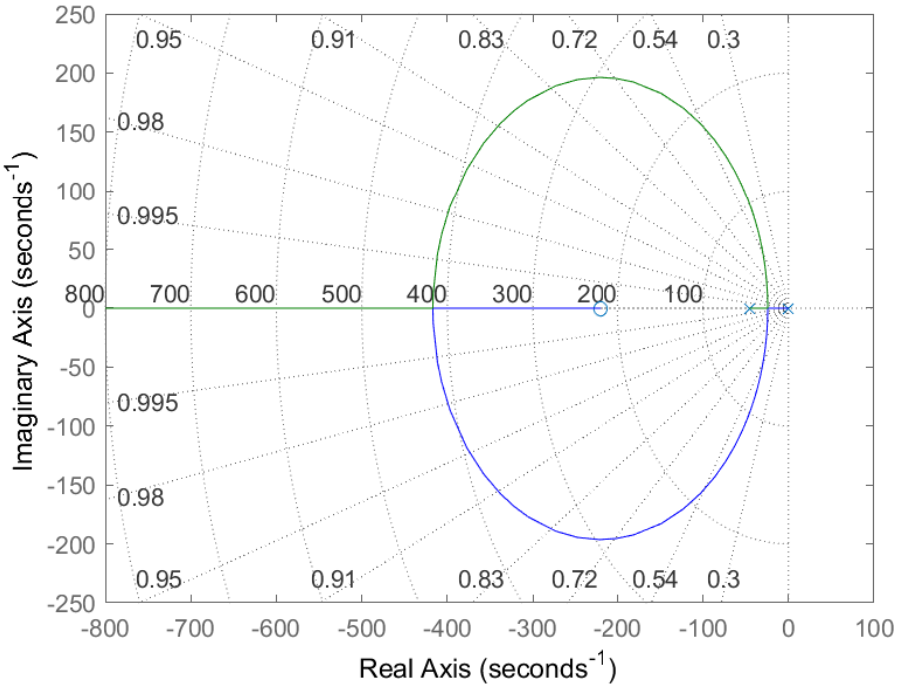

Figura 27: Lugar geométrico de las raíces.

A continuación, en Fig. 28 se muestra la forma de la respuesta obtenida por el simulador.

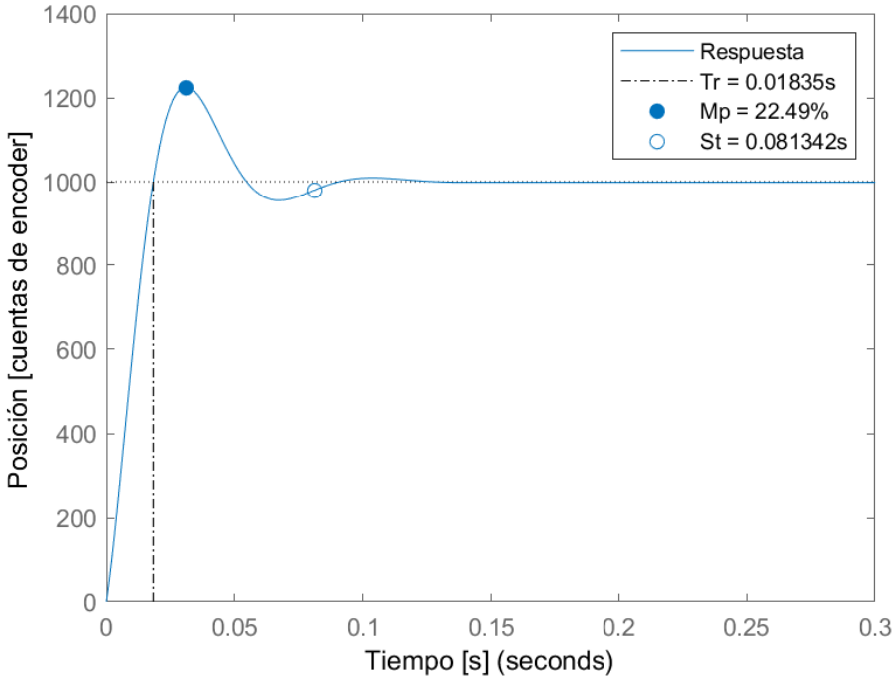

Figura 28: Simulación con control PD del sistema en lazo cerrado, con $kp = 7.07$ y $kd = 32.04$.

Tabla 9: Información de validación del modelo del controlador teórico.

Tr deseado	**Tr obtenido**	**Mp deseado**	**Mp obtenido**
$0.0236s$	$0.0184s$	20 %	22.49 %

La respuesta obtenida de forma experimental se muestra a continuación en la Fig. 29

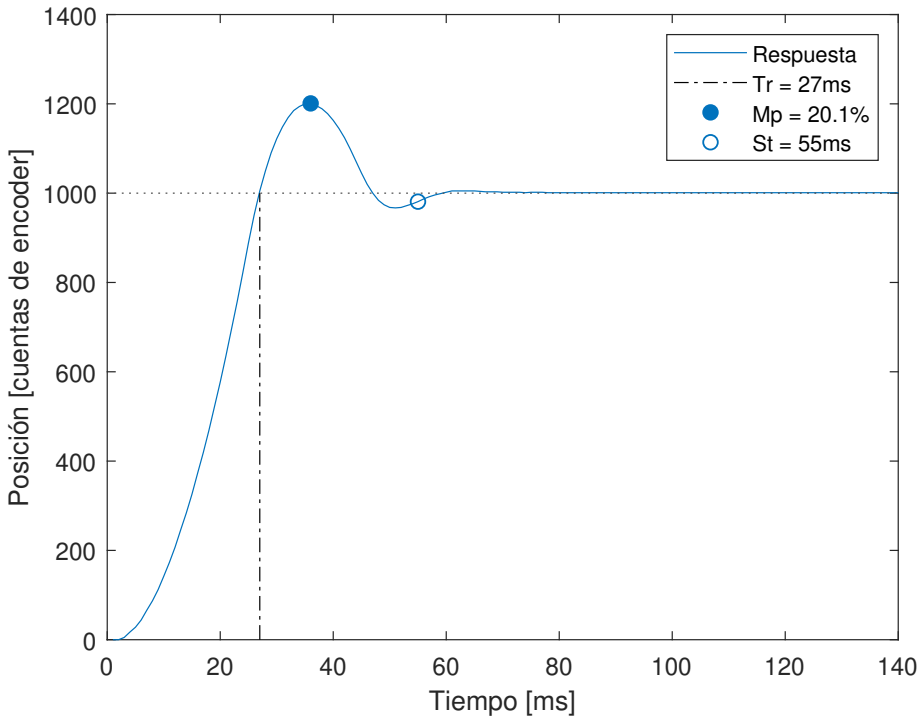

Figura 29: Respuesta experimental del control PD de posición, con $kp = 7.07$ y $kd = 32.04$.

Tabla 10: Información de validación del modelo del controlador.

Tr deseado	**Tr obtenido**	**Mp deseado**	**Mp obtenido**
$0.0236s$	$0.027s$	20 %	20.1 %

Realizada la validación el modelo de acuerdo a la información proporcionada por la identificación del sistema, se comienza la implementación de los perfiles de movimiento de forma experimental. Se realiza el procedimiento para cada nivel de sobrepaso que se utiliza en la implementación de cada perfil. Es decir Mp de 20 %, 15 %, 10 % y 5 %.

4.3. Implementación

4.3.1. Entorno

Un sistema en tiempo real permite un aprovechamiento de los recursos del microcontrolador, presentando velocidades de adquisición de datos relativamente altas, en el caso de esta experimentación, la velocidad de muestreo alcanzada por un MCU STM32F411CEU es de 1 ms a través de su convertidor analógico digital.

Por medio de lenguaje C en el entorno de desarrollo STM32CubeIDE se realiza la programación que permite la gestión de los perfiles que se comparan en este documento.

La adquisición de datos de energía se realiza por medio de la plataforma externa Analog Discovery y su software *WaveForms*.

La implementación es llevada a cabo en un Motor MAXON 137576-36MMDIA-24VDC-70W con un encoder rotativo LPD3806-600BM-G5-24C.

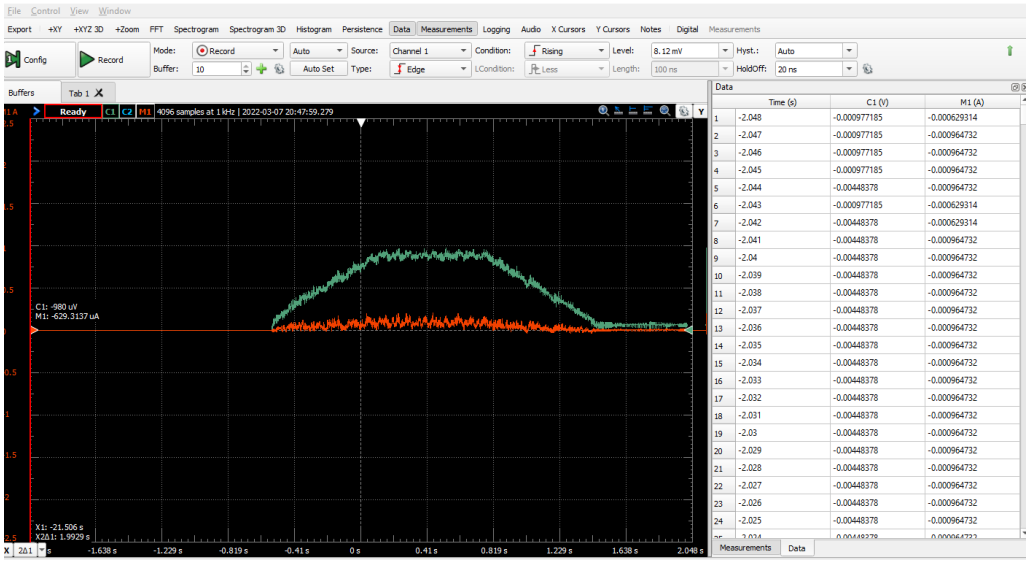

Figura 30: Adquisición de datos en WaveForms.

4.3.2. Perfiles numéricos

Establecidas las ecuaciones y la implementación del control de posición, se realiza la transición a un modelo numérico para cada perfil de velocidad. En el entorno de Matlab se realiza la validación de los perfiles propuestos, esto es, por medio de un

modelo discreto de cada perfil de velocidad, permitiendo la implementación en el MCU.

Sabiendo que el sistema es asintóticamente estable, se considera

Tiempo de muestreo

$$T_s = \frac{T_r}{10} \tag{4.22}$$

Por lo sugerido en (4.22), un tiempo de muestreo que el MCU puede gestionar correctamente es de $T_s = 0.001$ s.

En el caso de este documento, el sistema es causal y presenta el tiempo inicial = 0, siendo así, se muestra la forma general de una señal de valor continuo

$$x(t) = a \tag{4.23}$$

Sustituyendo t por kTs y aplicando un tren de impulsos, el muestreo de la señal es

$$x_s(t) = \sum_{k=0}^{\infty} x(kTs)\delta(t - kTs) \tag{4.24}$$

Donde $\delta(t - kts)$ es la función Delta de Dirac aplicado un retardo.

- ***Perfil triangular numérico***

  La ecuación (3.28) está en términos de posición y velocidad angular deseadas, valores que el usuario determina. Así es como de la integración de la curva de aceleración en Fig.31 se obtiene la velocidad angular.

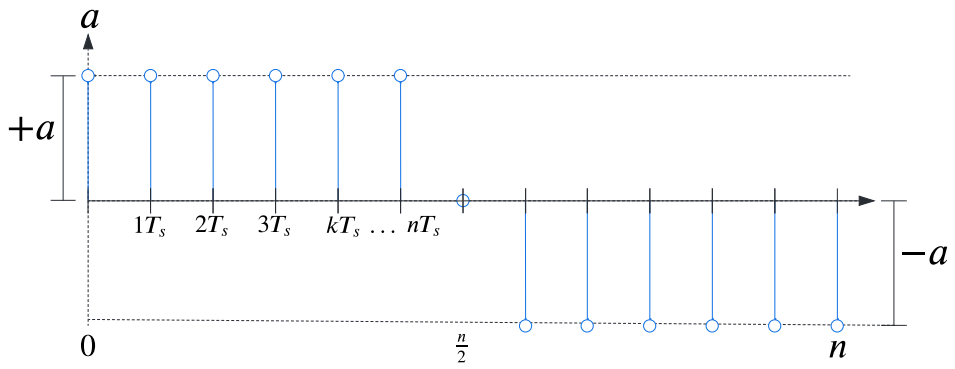

Figura 31: Muestreo de función de aceleración de perfil triangular.

$$\omega(kT_s) = \sum_{k=0}^{\frac{n}{2}} \alpha(kTs) + \sum_{k=\frac{n}{2}}^{n} \alpha(kTs) \tag{4.25}$$

Sustituyendo $a(t)$ por $a(kT_s)$ en la ecuación de velocidad angular (3.23), y n, que es la cantidad total de muestras dependiente de la relación de T/Ts.

$$\omega(kT_s) = \begin{cases} +akT_s & 0 < k \leq \frac{n}{2} \\ -akT_s + aT & \frac{n}{2} < k \leq n \end{cases} \tag{4.26}$$

De la misma forma para obtener la posición angular y establecer los datos que se envían al control de posición del microcontrolador.

$$\theta(kT_s) = \begin{cases} \frac{1}{2}akT_s^2 & 0 < k \leq \frac{n}{2} \\ -\frac{1}{2}akT_s^2 + akT_sT - \frac{1}{4}aT^2 & \frac{n}{2} < k \leq n \end{cases} \tag{4.27}$$

- ***Perfil trapezoidal numérico***

  De forma similar al perfil anterior se obtiene la función de $\theta(kT_s)$ de la ecuaciónes (3.35), (3.36) y (3.37) .

$$\theta(kT_s) = \begin{cases} \frac{1}{2}akT_s^2 & 0 < k \leq \frac{n}{3} \\ \frac{1}{3}akT_sT - \frac{1}{18}aT^2 & \frac{n}{2} < k \leq \frac{2n}{3} \\ \frac{1}{2}akT_s^2 + akT_sT - \frac{5}{18}aT^2 & \frac{2n}{3} < k \leq n \end{cases} \tag{4.28}$$

- ***Perfil parabólico numérico***

  Análogamente a los perfiles anteriores de la ecuación 3.45 se obtiene

$$\theta(kT_s) = -\frac{1}{3}\frac{kT_s^3}{T} + \frac{1}{2}akT_s^2 \quad para \quad 0 < k \leq n \tag{4.29}$$

- ***Perfil polinomial numérico***

  En forma similar de la ecuaciones (3.60), (3.62) y (3.63) se obtiene que

$$\theta(kT_s) = \begin{cases} -\frac{1}{2T}akT_s^3 + \frac{1}{2}akT_s^2 & 0 < k \leq \frac{n}{3} \\ \frac{1}{6}akT_sT - \frac{1}{54}aT^2 & \frac{n}{2} < k \leq \frac{2n}{3} \\ -\frac{1}{2}akT_s^3 + akT_s^2 - \frac{1}{2}akT_s + \frac{7}{54}aT^2 & \frac{2n}{3} < k \leq n \end{cases} \tag{4.30}$$

Cada una de las ecuaciones de posición correspondientes a cada perfil, son ingresadas al MCU. A través del software Matlab se obtiene la diferencia entre el valor ideal de cada perfil y su correspondiente de forma experimental.

5. Resultados

5.0.1. Respuesta de control derivativo de posición

Se muestra en las siguientes figuras y tablas las respuestas experimentales del control derivativo de posición con sus datos de interés.

Sobrepaso 20 %

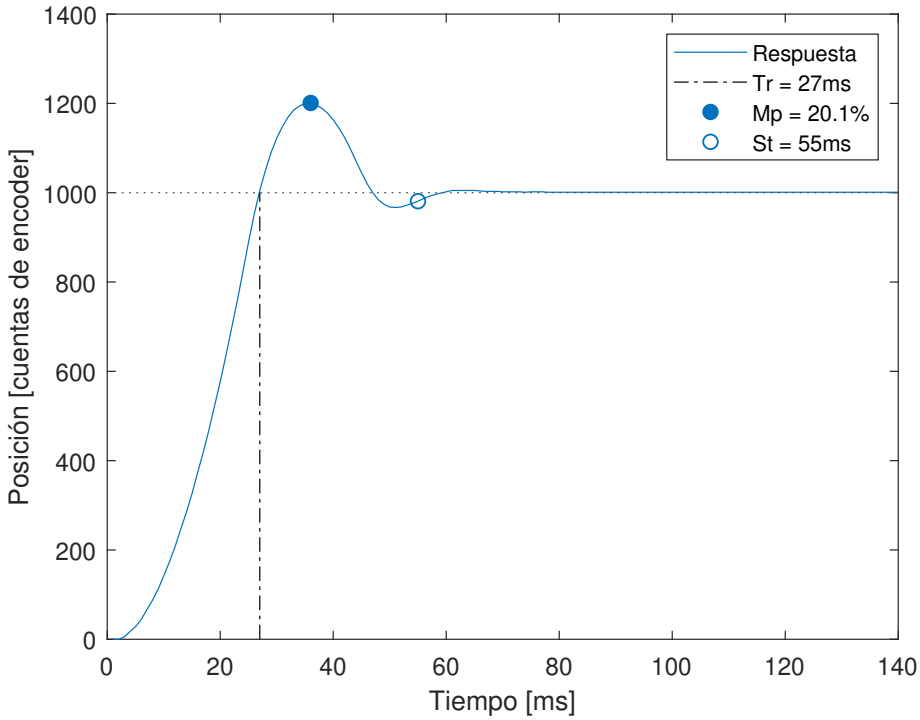

Figura 32: Respuesta con 20 % de sobrepaso.

Tabla 11: Datos de respuesta al escalón con $MP = 20\,\%$.

Sobrepaso	**Tr**	**St**	**Kp**	**Kd**
20.1 %	27*ms*	55*ms*	14.283	39.505

Sobrepaso 15 %

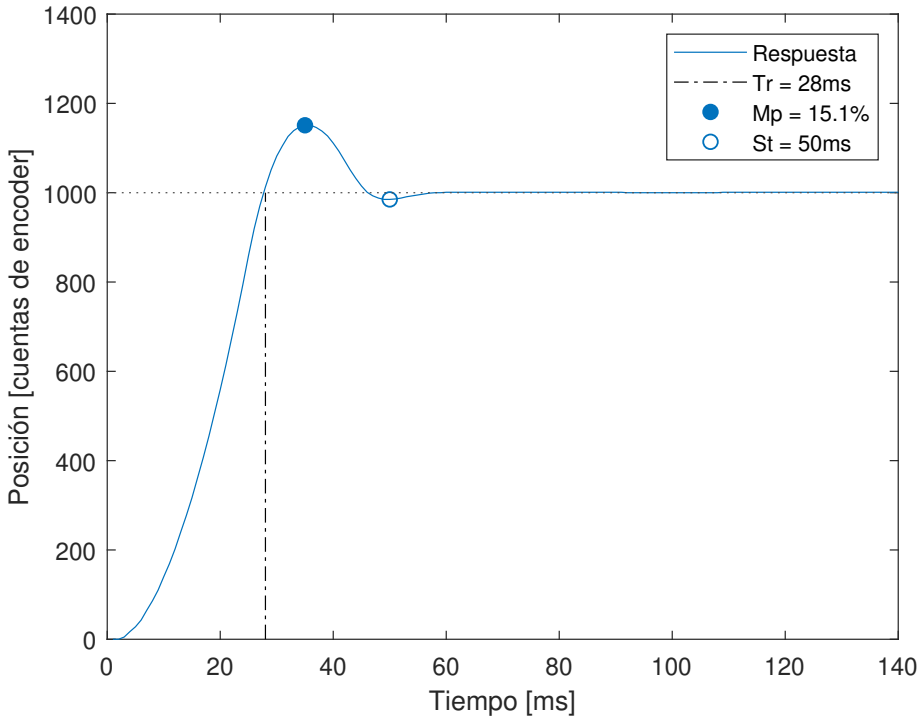

Figura 33: Respuesta con 15 % de sobrepaso.

Tabla 12: Datos de respuesta al escalón con $MP = 15\,\%$

Sobrepaso	**Tr**	**St**	**Kp**	**Kd**
15.1 %	28*ms*	50*ms*	15.651	52.08

Sobrepaso 10 %

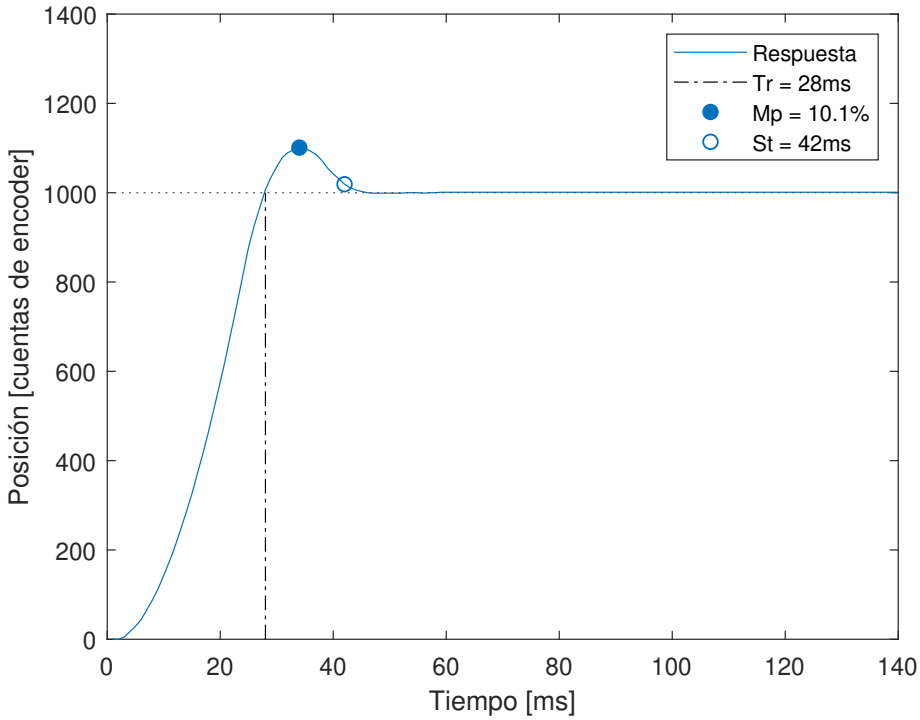

Figura 34: Respuesta con 10 % de sobrepaso.

Tabla 13: Datos de respuesta al escalón con $MP = 10\,\%$

Sobrepaso	**Tr**	**St**	**Kp**	**Kd**
10.1 %	28*ms*	42*ms*	17.741	72.017

Sobrepaso 5 %

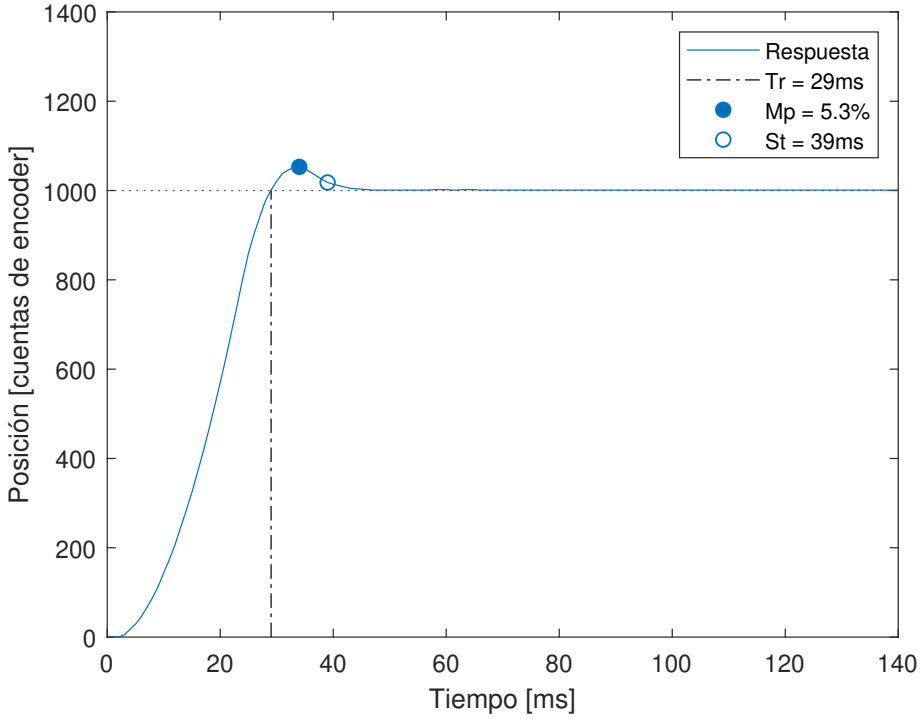

Figura 35: Respuesta con 5 % de sobrepaso.

Tabla 14: Datos de respuesta al esc5ón con $MP = 5\,\%$

Sobrepaso	**Tr**	**St**	**Kp**	**Kd**
5.3 %	29*ms*	39*ms*	20.249	92.999

Con cada una de las respuestas se realizan las series de pruebas de una trayectoria completa utilizando los perfiles propuestos.

5.1. Respuestas de perfil correspondientes al nivel de sobrepaso

A continuación las respuestas del perfil de movimiento de acuerdo nivel de sobrepaso obtenido por el control de posición. Se muestran las curvas esperadas de posición, velocidad, aceleración en contraste a las obtenidas.

5.1.1. Resultados de perfil triangular

Información gráfica en Fig. 36 del perfil triangular.

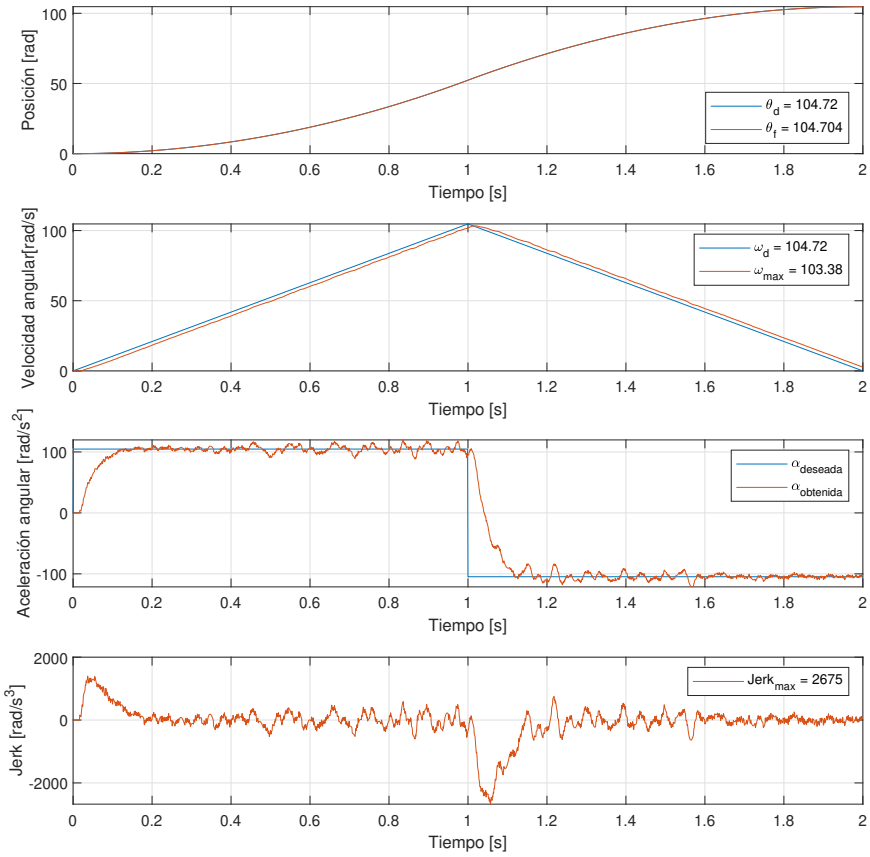

Figura 36: Perfil triangular usando respuesta con $M_p = 20\,\%$.

Se presenta la información gráfica en Fig. 37 de voltaje, corriente, potencia y energía consumida.

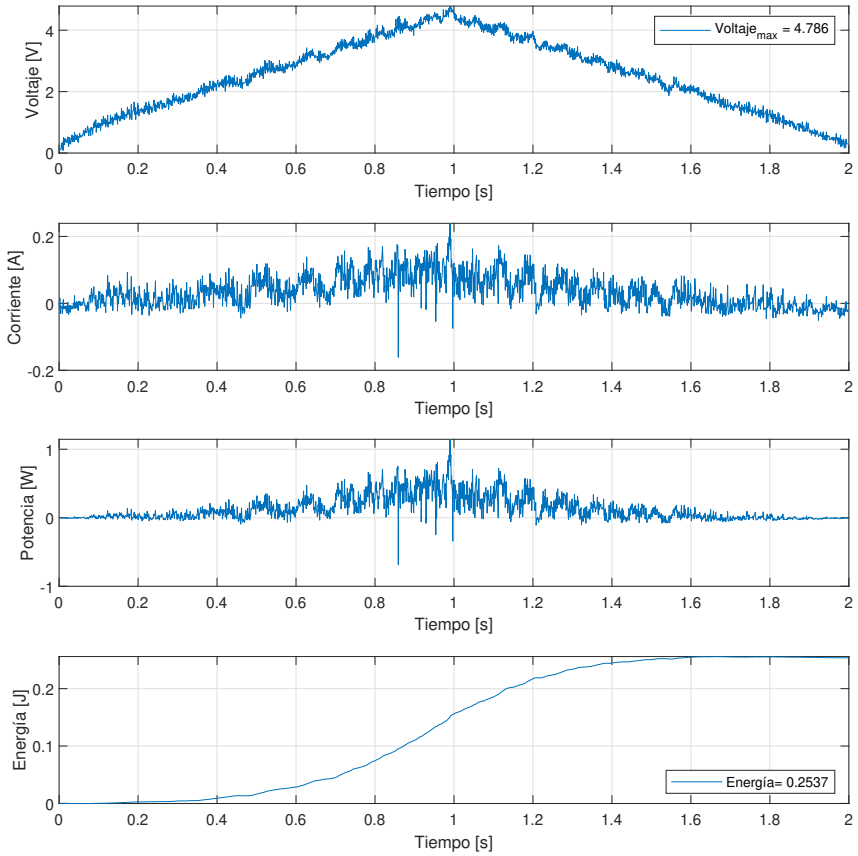

Figura 37: Energía de perfil triangular usando respuesta con $M_p = 20\,\%$.

Se realizan los experimentos correspondientes a cada nivel de sobrepaso obteniendo los datos mostrados en la tabla 15.

Tabla 15: Datos de perfil triangular acuerdo a nivel de sobrepaso.

Mp	$\omega_{max}[rad/s]$	$\alpha_{max}[rad/s^2]$	$Jerk_{max}[rad/s^3]$	$Energia[J]$	**RDCM**$[rad]$
20 %	103.38	119.75	2675.03	0.2537	0.7040
15 %	103.36	118.80	2768.75	0.3135	0.7067
10 %	103.14	117.01	2559.21	0.2849	0.7067
5 %	103.11	115.61	2526.44	0.3157	0.7093

5.1.2. Resultados de perfil trapezoidal

Información gráfica en Fig. 36 del perfil trapezoidal.

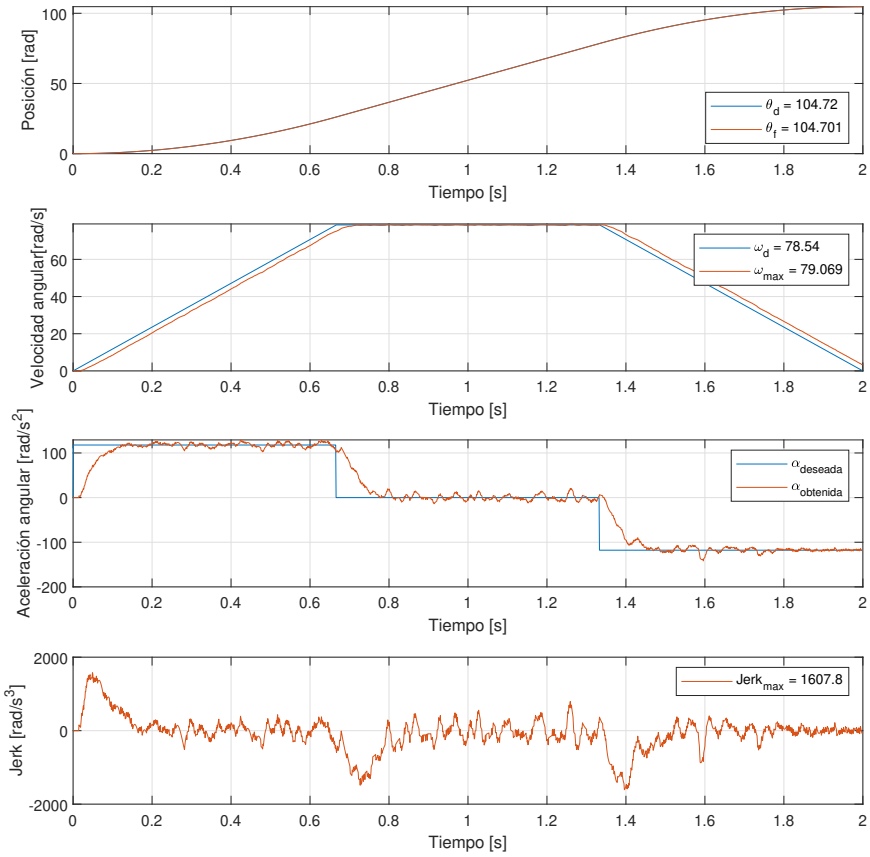

Figura 38: Perfil trapezoidal usando respuesta con $M_p = 20\,\%$.

Se presenta la información gráfica en Fig. 39 de voltaje, corriente, potencia y energía consumida.

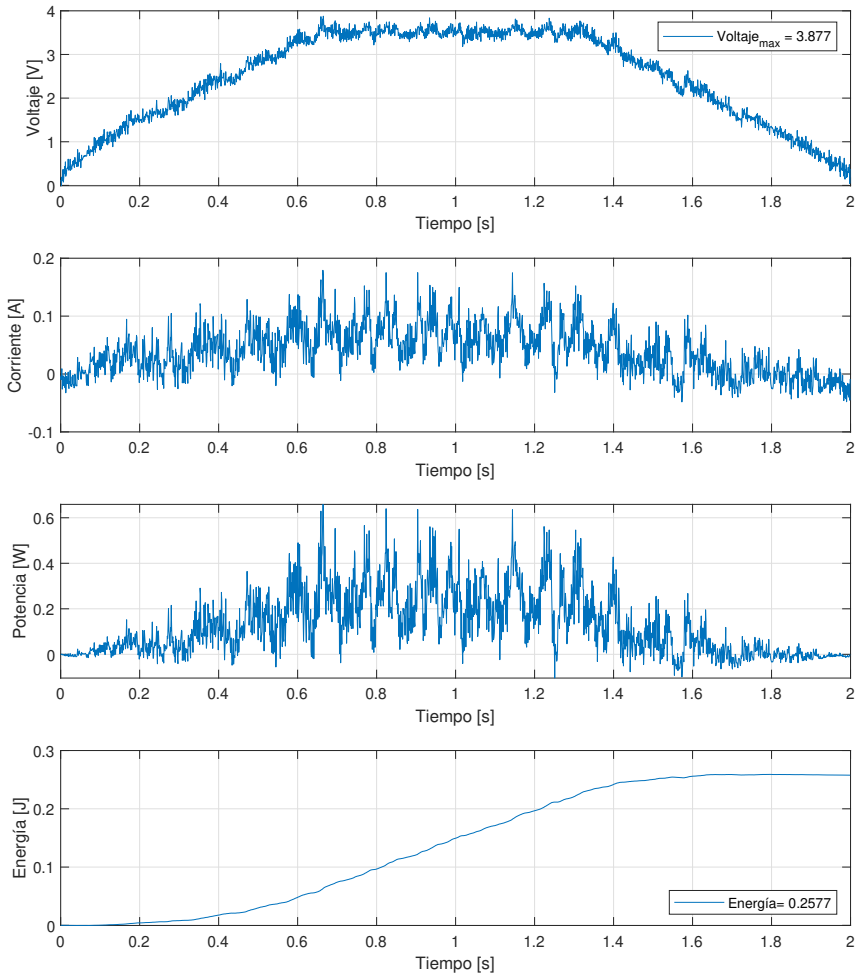

Figura 39: Energía de perfil trapezoidal usando respuesta con $M_p = 20\,\%$.

Se realizan los experimentos correspondientes a cada nivel de sobrepaso obteniendo los datos mostrados en la tabla 16.

Tabla 16: Datos de perfil trapezoidal acuerdo a nivel de sobrepaso.

Mp	$\omega_{max}[rad/s]$	$\alpha_{max}[rad/s^2]$	$Jerk_{max}[rad/s^3]$	$Energia[J]$	**RDCM**[*rad*]
20 %	79.07	129.23	1607.83	0.2577	0.7014
15 %	78.93	128.67	1520.21	0.2927	0.7040
10 %	78.82	129.09	1552.15	0.2485	0.7067
5 %	78.79	128.45	1485.40	0.3185	0.7093

5.1.3. Resultados de perfil parabólico

Información gráfica en Fig. 40 del perfil parabólico

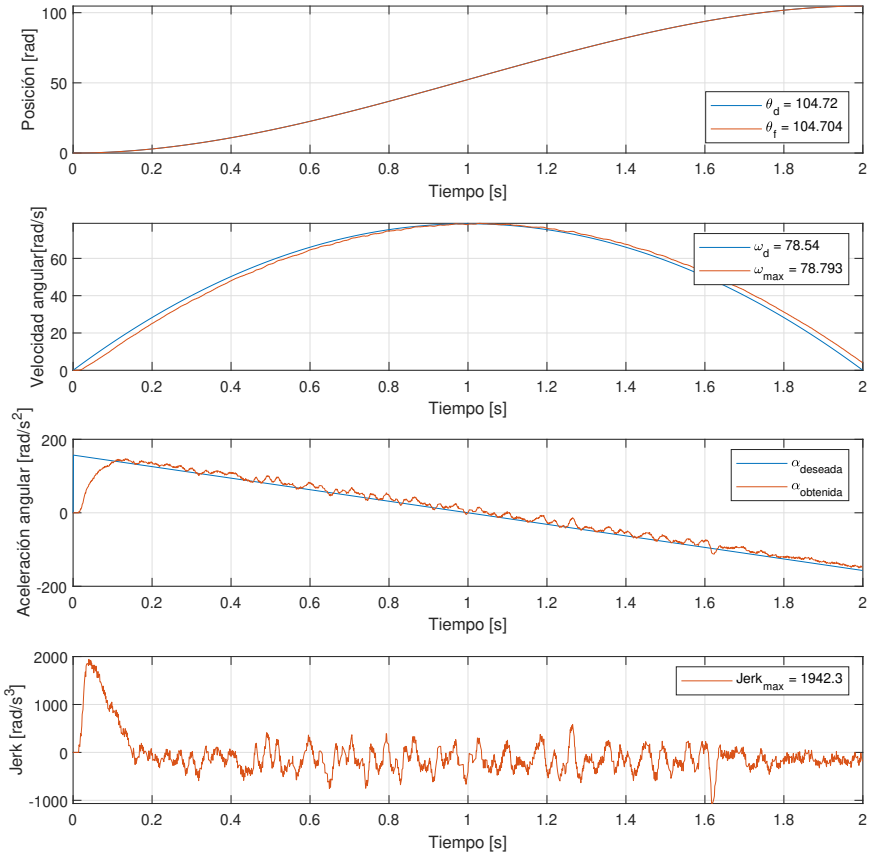

Figura 40: Perfil parabólico usando respuesta con $M_p = 20\,\%$.

Se presenta la información gráfica en Fig. 41 de voltaje, corriente, potencia y energía consumida.

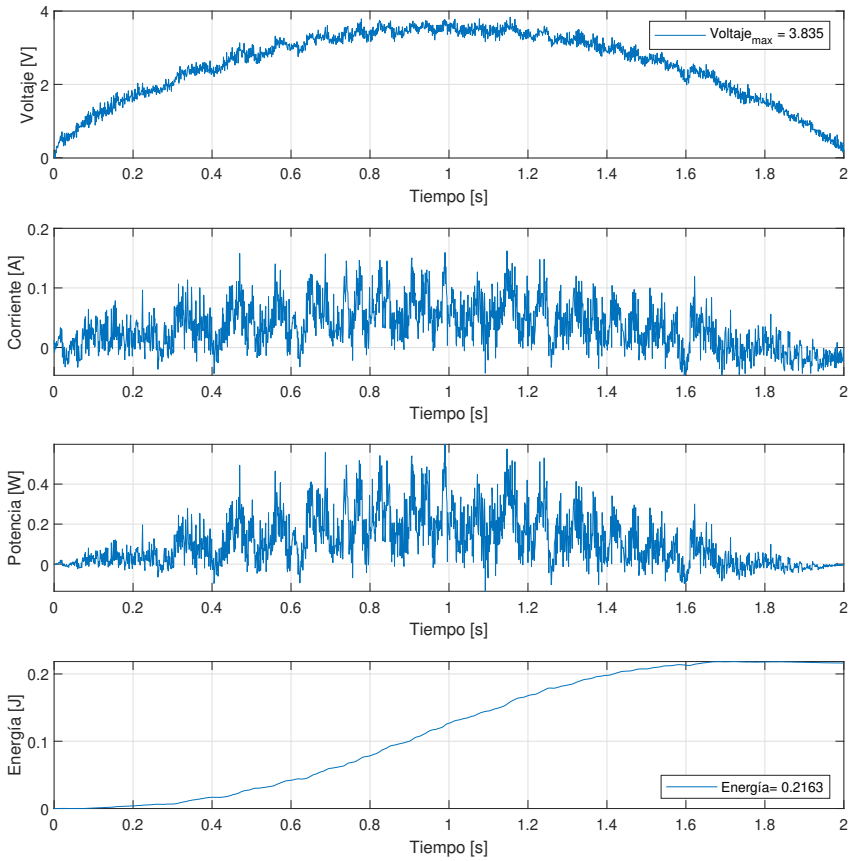

Figura 41: Energía de perfil parabólico usando respuesta con $M_p = 20\,\%$.

Se realizan los experimentos correspondientes a cada nivel de sobrepaso obteniendo los datos mostrados en la tabla 17.

Tabla 17: Datos de perfil parabólico acuerdo a nivel de sobrepaso.

Mp	$\omega_{max}[rad/s]$	$\alpha_{max}[rad/s^2]$	$Jerk_{max}[rad/s^3]$	$Energia[J]$	**RDCM**$[rad]$
20 %	78.79	146.96	1942.28	0.2163	0.7040
15 %	78.67	146.44	1853.73	0.2324	0.7040
10 %	78.78	145.25	1893.13	0.2302	0.7067
5 %	78.67	144.80	1833.71	0.2837	0.7093

5.1.4. Resultados de perfil polinomial

Información gráfica en Fig. 42 del perfil polinomial.

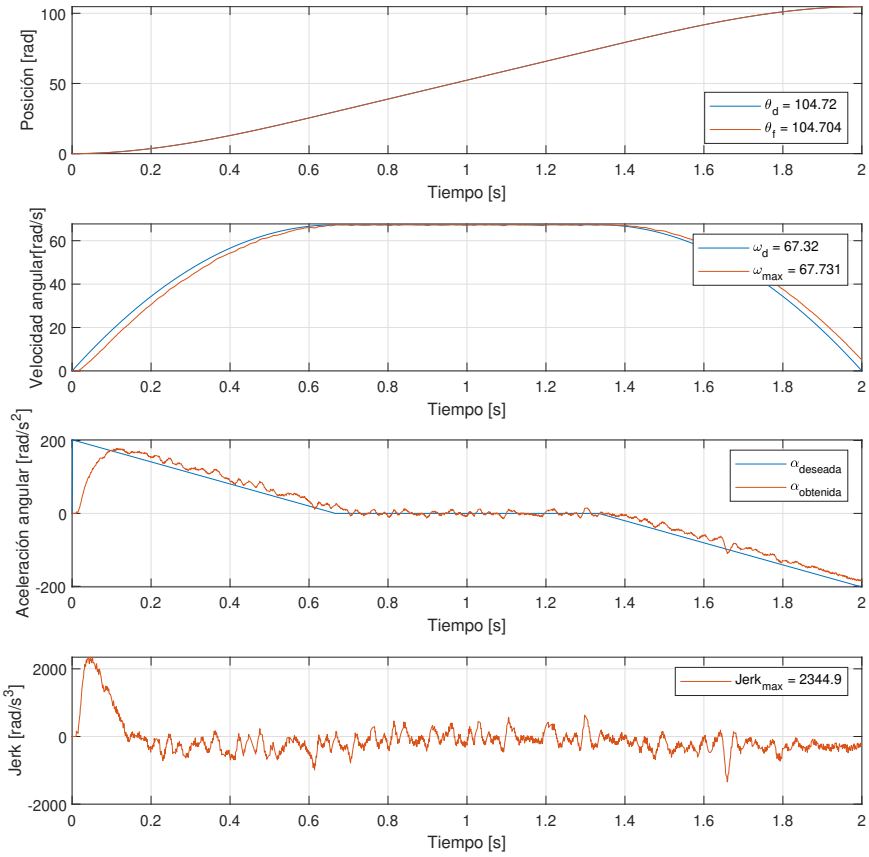

Figura 42: Perfil polinomial usando respuesta con $M_p = 20\,\%$.

Se presenta la información gráfica en Fig. 43 de voltaje, corriente, potencia y energía consumida.

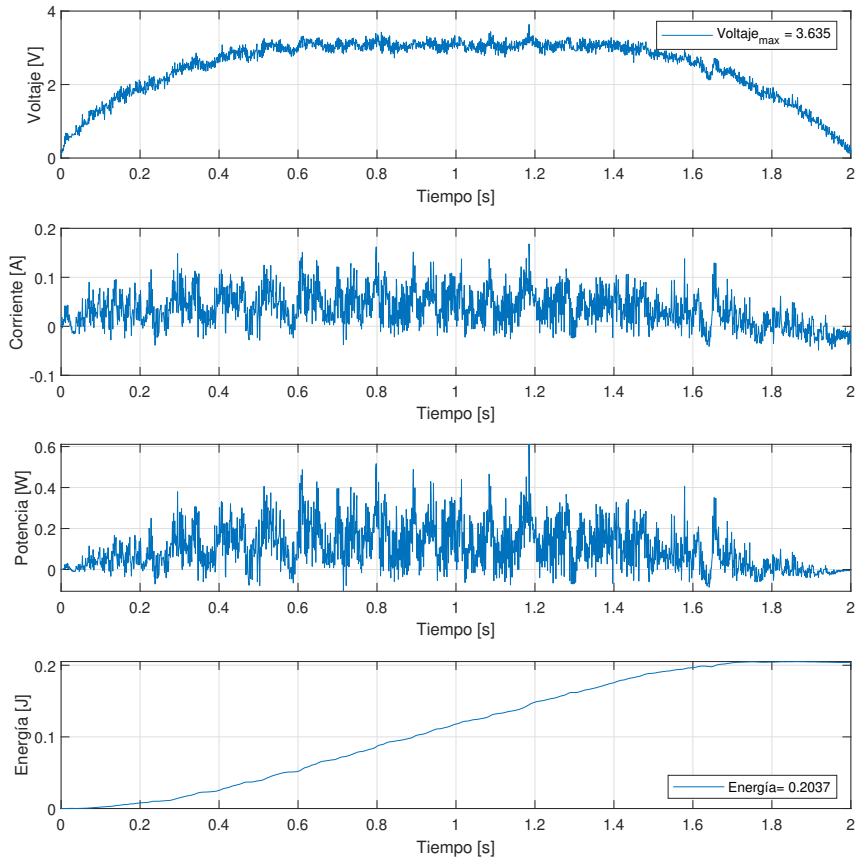

Figura 43: Energía de perfil polinomial usando respuesta con $M_p = 20\,\%$.

Se realizan los experimentos correspondientes a cada nivel de sobrepaso obteniendo los datos mostrados en la tabla 18.

Tabla 18: Datos de perfil polinomial acuerdo a nivel de sobrepaso.

Mp	$\omega_{max}[rad/s]$	$\alpha_{max}[rad/s^2]$	$Jerk_{max}[rad/s^3]$	$Energia[J]$	**RDCM**$[rad]$
20 %	67.73	178.19	2344.87	0.2037	0.7040
15 %	67.71	178.88	2371.07	0.2288	0.7040
10 %	67.62	177.49	2359.85	0.2171	0.7040
5 %	67.61	176.60	2314.87	0.2397	0.7067

En síntesis se muestra en la siguiente tabla los datos obtenidos para cada perfil en correspondencia al sobrepaso.

Tabla 19: Datos de perfiles obtenidos con $Mp = 20.1\,\%$

Perfil	*Jerk*[rad/s^3]	*Energia*[*J*]	**RDCM**[*rad*]
Triangular	2675.03	0.2537	0.7040
Trapezoidal	1607.83	0.2577	0.7014
Parabólico	1942.28	0.2163	0.7040
Polinomial	2344.87	0.2037	0.7040
$M_p = 20.1\,\%$ $T_r = 27ms$ $S_t = 55ms$			

Tabla 20: Datos de perfiles obtenidos con $Mp = 15.1\,\%$

Perfil	*Jerk*[rad/s^3]	*Energia*[*J*]	**RDCM**[*rad*]
Triangular	2768.75	0.3135	0.7067
Trapezoidal	1520.21	0.2927	0.7040
Parabólico	1853.73	0.2324	0.7040
Polinomial	2314.87	0.2288	0.7040
$M_p = 15.1\,\%$ $T_r = 28ms$ $S_t = 50ms$			

Tabla 21: Datos de perfiles obtenidos con $Mp = 10.1\,\%$

Perfil	*Jerk*[rad/s^3]	*Energia*[*J*]	**RDCM**[*rad*]
Triangular	2526.44	0.2849	0.7067
Trapezoidal	1552.15	0.2485	0.7067
Parabólico	1893.13	0.2302	0.7067
Polinomial	2359.85	0.2171	0.7040
$M_p = 10.1\,\%$ $T_r = 28ms$ $S_t = 42ms$			

Tabla 22: Datos de perfiles obtenidos con $Mp = 5.31\,\%$

Perfil	*Jerk*[rad/s^3]	*Energia*[*J*]	**RDCM**[*rad*]
Triangular	2526.44	0.3157	0.7093
Trapezoidal	1485.40	0.3185	0.7093
Parabólico	1833.71	0.2837	0.7093
Polinomial	2314.87	0.2397	0.7067
$M_p = 5.31\,\%$ $T_r = 29ms$ $S_t = 39ms$			

Contrastando los datos de RDCM y energía del perfil de las tablas 19 vs. 22 con 20.1 % y 5.31 % de sobrepaso respectivamente, las diferencias con un sobrepaso mayor son las siguientes:

- Triangular

  RDCM 0.75 % menor.

  Consumo energético 16.47 % menor.

  Jerk 8.75 % mayor.

- Trapezoidal

  RDCM 1.11 % menor.

Consumo energético 19.09 % menor.

Jerk 7.61 % mayor.

- Parabólico

  RDCM 0.75 % menor.

  Consumo energético 23.75 % menor.

  Jerk 5.59 % mayor.

- Polinomial

  RDCM 0.38 % menor.

  Consumo energético 12.69 % menor.

  Jerk 1.28 % mayor.

De la comparativa mostrada, se observa que en el caso de mayor sobrepaso, los valores resultantes presentan mayor eficiencia, esto es, para un menor RDCM existe un menor consumo de energía.

Por otro lado se contrasta el desempeño en el jerk, obteniendo valores más altos cuando existe un mayor sobrepaso en la respuesta del controlador.

6. Conclusiones

La implementación de un modelo aproximado permitió realizar una identificación más rápida, obteniendo de una forma mucho más sencilla las funciones de transferencia para la sintonización de las ganancias del controlador de posición utilizado, siendo esta identificación útil ya que no es necesario conocer a priori el modelo del motor ni sus características intrínsecas.

La expectativa de la hipótesis presentada se cumple en torno la influencia del nivel de sobrepaso en las respuestas del perfil del velocidad. Los resultados muestran un consumo energético promedio 18.0 % menor y en la RDCM de posición un 0.75 % menor en cada trazo, en ambos casos cuando el control de posición presenta en su respuesta un mayor sobrepaso, menor tiempo de respuesta y asentamiento. Esta información permite, dependiendo las necesidades de la tarea a realizar, qué criterios utilizar para la implementación de cada perfil, ya sea para reducir la aparición

de fallas tempranas provocadas por sobreimpulsos en el sistema mecánico o para mejorar la eficiencia energética en la ejecución de la aplicación requerida.

Al margen de los resultados de la trayectoria presentada como polinomial o trapezoidal modificado, su ejecución presentó datos no esperados, siendo muy positivos en cuanto consumo energético y de RDCM, lo cual abre una oportunidad de profundizar en la investigación de esta modificación e implementación de este modelo bajo distintas trayectorias.

Referencias

[1] Abdul Rahim Abdullah, Norhazilina Bahari, Mohd Abu Hassan, and Muhammad Sabri. Efficiency comparison of trapezoidal and sinusoidal method for brushless dc motor drive. *Applied Mechanics and Materials*, 785:248–252, 04 2015.

[2] International Energy Agency. *World Energy Outlook 2006*. Organisation for Economic Co-operation and Development (OECD), 2006.

[3] Jun Ma (Author); Xiaocong Li (Author); Kok Kiong Tan (Author). *Advanced Optimization for Motion Control Systems*. CRC Press, 1 edition, 2020.

[4] S. Bennett. *A History of Control Engineering 1800-1930*. Control Engineering Series 8. The Institution of Engineering and Technology, London, United Kingdom, 2008.

[5] José García-Martínez and Juvenal Rodriguez. Assessment of jerk performance s-curve and trapezoidal velocity profiles. *Engineering Congress (CONIIN), 2017 XIII International*, 05 2017.

[6] Jacek F. Gieras. *Electrical machines : fundamentals of electromechanical energy conversion*. CRC Press, 2017.

[7] Hakan Gürocak. *Industrial Motion Control: Motor Selection, Drives, Controller Tuning, Applications*. Wiley, 1 edition, 2015.

[8] Hong-Jun Heo, Yungdeug Son, and Jang-Mok Kim. A trapezoidal velocity profile generator for position control using a feedback strategy. *Energies*, 12, 03 2019.

[9] Shane Holt Hugh Falkner. *Walking the Torque: Proposed Work Plan for Energy-Efficiency Policy. Opportunities for Electric Motor-Driven Systems*. IEA Energy Papers 2011/08. OECD Publishing, 2011.

[10] Fuentes L. José. Solución óptima de modelos dinámicos detallados de robots industriales basados en la norma l1. `https://academica-e.unavarra.es/bitstream/handle/2454/31439/TFM_Jose%20Fuentes.pdf?sequence=1&isAllowed=y`, 2018. Universidad Pública de Navarra, Pamplona, España.

[11] Hiqmet Kamberaj. *Classical Mechanics (de Gruyter Textbook)*. De Gruyter, 2021.

[12] Victor Montalvo, Adyr Estévez-Bén, Juvenal Rodriguez, Gonzalo Macias-Bobadilla, Jorge Mendiola-Santíbañez, and Karla Camarillo-Gómez. Fpga-based architecture for sensing power consumption on parabolic and trapezoidal motion profiles. *Electronics*, 9, 08 2020.

[13] Kim-Doang Nguyen, I-Ming Chen, and Teck Ng. Planning algorithms for s-curve trajectories. *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, pages 1 – 6, 10 2007.

[14] Norman S. Nise. *Control System Engineering.* Wiley, 8 edition, 2019.

[15] Katsuhiko Ogata. *Modern Control Engineering.* Prentice Hal, 5 edition, 2010.

[16] Koen Paes, Wim Dewulf, Karel Vander Elst, Karel Kellens, and Peter Slaets. Energy efficient trajectories for an industrial abb robot. *Procedia CIRP*, 15:105–110, 2014. 21st CIRP Conference on Life Cycle Engineering.

[17] Georgios Pastras, Apostolos Fysikopoulos, and George Chryssolouris. A theoretical investigation on the potential energy savings by optimization of the robotic motion profiles. *Robotics and Computer-Integrated Manufacturing*, 58:55–68, 2019.

[18] Conrad U. Brunner Paul Waide. *Energy-Efficiency Policy Opportunities for Electric Motor-Driven Systems.* IEA Energy Papers 2011/07. OECD Publishing, 2011.

[19] Robert H. Bishop Richard C. Dorf. *Modern Control Systems (12th Edition).* 12 edition, 2010.

[20] P. Tokekar, N. Karnad, and V. Isler. Energy-optimal velocity profiles for car-like robots. In *2011 IEEE International Conference on Robotics and Automation*, pages 1457–1462, 2011.

[21] Ramón Silva-Ortigoza Victor Manuel Hernández-Guzmán. *Automatic Control with Experiments.* Advanced Textbooks in Control and Signal Processing. Springer International Publishing, 1st ed. edition, 2019.

Apéndice

A continuación se anexan los códigos utilizados en el microcontrolador y plataforma Matlab, con la finalidad de establecer una mejor comprensión de este documento.

A. Código implementado en STM32F411CEU6.

A.1. main.c

```
#include <main.h>


TIM_HandleTypeDef htim4 = {0};
TIM_HandleTypeDef htim2 = {0};
UART_HandleTypeDef huart1 = {0};


char received_char, bufIn[32];
unsigned char bufIdx = 0;

int32_t position, reference = 0;
PIDControl pid;
Profile profile;
int32_t despos = 1000;
int32_t desvel = 1000;
int sidx = SAMPLES;
uint32_t samples[SAMPLES];


void GPIO_Init(void);
void Error_Handler(void);
void Clock_Init(void);
void TIM2_Init(void);
void TIM4_Init(void);
void UART1_Init(void);
void System_Init(void);
void writeOutput(int);


TimerHandle_t blinkTimer;
SemaphoreHandle_t xSem;
QueueHandle_t queue;

void blinkTask(TimerHandle_t);
void controlTask(void *);
void interfaceTask(void *);


int main(void)
{
        System_Init();
        xSem = xSemaphoreCreateBinary();
        queue = xQueueCreate(64, sizeof(char));
        xTaskCreate(controlTask, "Control", configMINIMAL_STACK_SIZE * 2, NULL, 3, NULL);
        xTaskCreate(interfaceTask, "Interface", configMINIMAL_STACK_SIZE * 4, NULL, 1, NULL);
        blinkTimer = xTimerCreate("Blink", 250, pdTRUE, NULL, blinkTask);
        xTimerStart(blinkTimer, 0);
        vTaskStartScheduler();
        return 0;
}

void controlTask(void *pvParameters)
{
        TickType_t xLastWakeTime;
```

```
        const TickType_t xFrequency = pdMS_TO_TICKS(1);
        pid_init(&pid, &reference, &position);
        profile_init(&profile, &reference);
        xLastWakeTime = xTaskGetTickCount();
        while(1)
        {
                vTaskDelayUntil(&xLastWakeTime, xFrequency);
                position = __HAL_TIM_GET_COUNTER(&htim2);
                if (sidx < SAMPLES)
                {
                        samples[sidx++] = position;
                }
                profile_execute(&profile);
                pid_compute(&pid);
                writeOutput(pid.usat);
        }
}

void writeOutput(int duty)
{
        if (duty >= 0)
        {
                HAL_GPIO_WritePin(GPIOB, GPIO_PIN_7, GPIO_PIN_RESET);
                __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_1, duty);
        }
        else
        {
                HAL_GPIO_WritePin(GPIOB, GPIO_PIN_7, GPIO_PIN_SET);
                __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_1, -duty);
        }
}

void currentTask(void *pvParameters)
{

}

void interfaceTask(void *pvParameters)
{
        char buflen, bufOut[32];
        char cmd[8], args[32], *pch;
        unsigned int i, j;
        while(1)
        {
                xSemaphoreTake(xSem, portMAX_DELAY);
                pch = strtok (bufIn," ");
                strcpy(cmd, pch);
                pch = strtok(NULL, " ");
                strcpy(args, pch);
                /* Parsing user commands ******************************************/
                if (!strcmp(cmd, "RS"))
                {
                        reference = position = 0;
                        __HAL_TIM_SET_COUNTER(&htim2, 0);
                        pid_reset(&pid);
                        buflen = sprintf(bufOut, ": Ok\r\n");
                        for (i = 0; i < buflen; i++)
                                xQueueSend(queue, bufOut + i, portMAX_DELAY);
                }
                if (!strcmp(cmd, "TP"))
                {
                        buflen = sprintf(bufOut, ": %li\r\n", position);
                        for (i = 0; i < buflen; i++)
                                xQueueSend(queue, bufOut + i, portMAX_DELAY);
                }
                if (!strcmp(cmd, "TE"))
                {
                        buflen = sprintf(bufOut, ": %li\r\n", reference - position);
                        for (i = 0; i < buflen; i++)
                                xQueueSend(queue, bufOut + i, portMAX_DELAY);
                }
                if (!strcmp(cmd, "TT"))
                {
                        buflen = sprintf(bufOut, ": %i\r\n", pid.usat);
                        for (i = 0; i < buflen; i++)
                                xQueueSend(queue, bufOut + i, portMAX_DELAY);
                }
```

```
if (!strcmp(cmd, "BG"))
{
        sidx = 0;
        profile_start(&profile, despos, desvel);
        buflen = sprintf(bufOut, ": Ok\r\n");
        for (i = 0; i < buflen; i++)
                xQueueSend(queue, bufOut + i, portMAX_DELAY);
}
if (!strcmp(cmd, "TRI"))
{

        profile.type = TRIANGULAR;
        buflen = sprintf(bufOut, ": Ok\r\n");
        for (i = 0; i < buflen; i++)
                xQueueSend(queue, bufOut + i, portMAX_DELAY);
}
if (!strcmp(cmd, "TRAP"))
{

        profile.type = TRAPEZOIDAL;
        buflen = sprintf(bufOut, ": Ok\r\n");
        for (i = 0; i < buflen; i++)
                xQueueSend(queue, bufOut + i, portMAX_DELAY);
}
if (!strcmp(cmd, "PAR"))
{
        profile.type = PARABOLIC;
        buflen = sprintf(bufOut, ": Ok\r\n");
        for (i = 0; i < buflen; i++)
                xQueueSend(queue, bufOut + i, portMAX_DELAY);
}
if (!strcmp(cmd, "SIN"))
                {
                        profile.type = SINUSOIDAL;
                        buflen = sprintf(bufOut, ": Ok\r\n");
                        for (i = 0; i < buflen; i++)
                                xQueueSend(queue, bufOut + i, portMAX_DELAY);
                }
if (!strcmp(cmd, "RES"))
{
        for (j = 0; j < SAMPLES; j++)
        {
                buflen = sprintf(bufOut, "%li\r\n", samples[j]);
                for (i = 0; i < buflen; i++)
                        xQueueSend(queue, bufOut + i, portMAX_DELAY);
        }
}
if (!strcmp(cmd, "REF"))
{
        if (!strcmp(args, "?"))
        {
                buflen = sprintf(bufOut, ": %li\r\n", reference);
                for (i = 0; i < buflen; i++)
                        xQueueSend(queue, bufOut + i, portMAX_DELAY);
        }
        else
        {
                reference = atoi(args);
                sidx = 0;
                buflen = sprintf(bufOut, ": Ok\r\n");
                for (i = 0; i < buflen; i++)
                        xQueueSend(queue, bufOut + i, portMAX_DELAY);
        }
}
if (!strcmp(cmd, "KP"))
{
        if (!strcmp(args, "?"))
        {
                buflen = sprintf(bufOut, ": %.4f\r\n", pid.Kp);
                for (i = 0; i < buflen; i++)
                        xQueueSend(queue, bufOut + i, portMAX_DELAY);
        }
        else
        {
                pid.Kp = atof(args);
                buflen = sprintf(bufOut, ": Ok\r\n");
                for (i = 0; i < buflen; i++)
```

```
                        xQueueSend(queue, bufOut + i, portMAX_DELAY);
        }
}
if (!strcmp(cmd, "KI"))
{
        if (!strcmp(args, "?"))
        {
                buflen = sprintf(bufOut, ": %.4f\r\n", pid.Ki);
                for (i = 0; i < buflen; i++)
                        xQueueSend(queue, bufOut + i, portMAX_DELAY);
        }
        else
        {
                pid.Ki = atof(args);
                buflen = sprintf(bufOut, ": Ok\r\n");
                for (i = 0; i < buflen; i++)
                        xQueueSend(queue, bufOut + i, portMAX_DELAY);
        }
}
if (!strcmp(cmd, "KD"))
{
        if (!strcmp(args, "?"))
        {
                buflen = sprintf(bufOut, ": %.4f\r\n", pid.Kd);
                for (i = 0; i < buflen; i++)
                        xQueueSend(queue, bufOut + i, portMAX_DELAY);
        }
        else
        {
                pid.Kd = atof(args);
                buflen = sprintf(bufOut, ": Ok\r\n");
                for (i = 0; i < buflen; i++)
                        xQueueSend(queue, bufOut + i, portMAX_DELAY);
        }
}
if (!strcmp(cmd, "PR"))
{
        if (!strcmp(args, "?"))
        {
                buflen = sprintf(bufOut, ": %li\r\n", despos);
                for (i = 0; i < buflen; i++)
                        xQueueSend(queue, bufOut + i, portMAX_DELAY);
        }
        else
        {
                despos = atoi(args);
                sidx = 0;
                buflen = sprintf(bufOut, ": Ok\r\n");
                for (i = 0; i < buflen; i++)
                        xQueueSend(queue, bufOut + i, portMAX_DELAY);
        }
}
if (!strcmp(cmd, "PA"))
{
        if (!strcmp(args, "?"))
        {
                buflen = sprintf(bufOut, ": %li\r\n", despos);
                for (i = 0; i < buflen; i++)
                        xQueueSend(queue, bufOut + i, portMAX_DELAY);
        }
        else
        {
                despos = atoi(args) - reference;
                sidx = 0;
                buflen = sprintf(bufOut, ": Ok\r\n");
                for (i = 0; i < buflen; i++)
                        xQueueSend(queue, bufOut + i, portMAX_DELAY);
        }
}
if (!strcmp(cmd, "SP"))
{
        if (!strcmp(args, "?"))
        {
                buflen = sprintf(bufOut, ": %li\r\n", desvel);
                for (i = 0; i < buflen; i++)
                        xQueueSend(queue, bufOut + i, portMAX_DELAY);
        }
```

```
                        else
                        {
                                desvel = abs(atoi(args));
                                sidx = 0;
                                buflen = sprintf(bufOut, ": Ok\r\n");
                                for (i = 0; i < buflen; i++)
                                        xQueueSend(queue, bufOut + i, portMAX_DELAY);
                        }
                }
        }
}

void blinkTask(TimerHandle_t xTimer)
{
        HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13);
}

void vApplicationIdleHook(void)
{
        char byteSent;
        if (xQueueReceive(queue, &byteSent, 0) == pdTRUE)
                HAL_UART_Transmit(&huart1, (uint8_t *)&byteSent, 1, HAL_MAX_DELAY);
}

void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
        BaseType_t xHigherPriorityTaskWoken = pdFALSE;
        if (received_char == '\n')
        {
                bufIn[bufIdx++] = '\0';
                xSemaphoreGiveFromISR(xSem, &xHigherPriorityTaskWoken);
                bufIdx = 0;
        }
        else if (received_char != '\r')
        {
                bufIn[bufIdx++] = received_char;
        }

        HAL_UART_Receive_IT(&huart1, (uint8_t *)&received_char, 1);
        if (xHigherPriorityTaskWoken == pdTRUE)
                portYIELD_FROM_ISR(xHigherPriorityTaskWoken);
}

void System_Init(void)
{
        GPIO_Init();
        Clock_Init();
        TIM2_Init();
        TIM4_Init();
        UART1_Init();
}

void Clock_Init(void)
{
        RCC_OscInitTypeDef osc_config = {0};
        osc_config.OscillatorType = RCC_OSCILLATORTYPE_HSE;
        osc_config.HSEState = RCC_HSE_ON;
        osc_config.PLL.PLLState = RCC_PLL_ON;
        osc_config.PLL.PLLSource = RCC_PLLSOURCE_HSE;
        osc_config.PLL.PLLM = 25;
        osc_config.PLL.PLLN = 200;
        osc_config.PLL.PLLP = 2;
        osc_config.PLL.PLLQ = 4;
        if (HAL_RCC_OscConfig(&osc_config) != HAL_OK)
                Error_Handler();

        RCC_ClkInitTypeDef clock_config = {0};
        clock_config.ClockType = RCC_CLOCKTYPE_SYSCLK |
                        RCC_CLOCKTYPE_HCLK | RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2;
        clock_config.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
        clock_config.AHBCLKDivider = RCC_SYSCLK_DIV1;
        clock_config.APB1CLKDivider = RCC_HCLK_DIV2;
        clock_config.APB2CLKDivider = RCC_HCLK_DIV1;
        if (HAL_RCC_ClockConfig(&clock_config, FLASH_LATENCY_4) != HAL_OK)
                Error_Handler();
}
```

```
void UART1_Init(void)
{
        GPIO_InitTypeDef uart_pins = {0};
        uart_pins.Pin = GPIO_PIN_9 | GPIO_PIN_10;
        uart_pins.Mode = GPIO_MODE_AF_PP;
        uart_pins.Alternate = GPIO_AF7_USART1;
        __HAL_RCC_GPIOA_CLK_ENABLE();
        HAL_GPIO_Init(GPIOA, &uart_pins);

        huart1.Instance = USART1;
        huart1.Init.BaudRate = 115200;
        huart1.Init.Mode = UART_MODE_TX_RX;
        __HAL_RCC_USART1_CLK_ENABLE();
        if (HAL_UART_Init(&huart1) != HAL_OK)
                Error_Handler();

        HAL_NVIC_SetPriority(USART1_IRQn, 15, 0);
        HAL_NVIC_EnableIRQ(USART1_IRQn);

        HAL_UART_Receive_IT(&huart1, (uint8_t *)&received_char, 1);
}

void USART1_IRQHandler(void)
{
        HAL_UART_IRQHandler(&huart1);
}

void TIM2_Init(void)
{
        GPIO_InitTypeDef qei_pin = {0};
        qei_pin.Pin = GPIO_PIN_0 | GPIO_PIN_1;
        qei_pin.Mode = GPIO_MODE_AF_PP;
        qei_pin.Pull = GPIO_PULLUP;
        qei_pin.Alternate = GPIO_AF1_TIM2;
        __HAL_RCC_GPIOA_CLK_ENABLE();
        HAL_GPIO_Init(GPIOA, &qei_pin);


        TIM_Encoder_InitTypeDef encoderInit = {0};
        htim2.Instance = TIM2;
        htim2.Init.Prescaler = 0;
        htim2.Init.AutoReloadPreload = 0;
        htim2.Init.Period = 0xFFFFFFFF;
        encoderInit.EncoderMode = TIM_ENCODERMODE_TI12;
        encoderInit.IC1Polarity = TIM_ENCODERINPUTPOLARITY_RISING;
        encoderInit.IC2Polarity = TIM_ENCODERINPUTPOLARITY_RISING;
        encoderInit.IC1Selection = TIM_ICSELECTION_DIRECTTI;
        encoderInit.IC2Selection = TIM_ICSELECTION_DIRECTTI;
        __HAL_RCC_TIM2_CLK_ENABLE();
        if (HAL_TIM_Encoder_Init(&htim2, &encoderInit) != HAL_OK)
                Error_Handler();
        HAL_TIM_Encoder_Start(&htim2, TIM_CHANNEL_ALL);
}

void TIM4_Init(void)
{
        GPIO_InitTypeDef pwm_pin = {0};
        pwm_pin.Pin = GPIO_PIN_6;
        pwm_pin.Mode = GPIO_MODE_AF_PP;
        pwm_pin.Alternate = GPIO_AF2_TIM4;
        __HAL_RCC_GPIOB_CLK_ENABLE();
        HAL_GPIO_Init(GPIOB, &pwm_pin);

        TIM_OC_InitTypeDef pwmConfig = {0};
        htim4.Instance = TIM4;
        htim4.Init.Prescaler = 0;
        htim4.Init.Period = 999;
        __HAL_RCC_TIM4_CLK_ENABLE();

        if (HAL_TIM_PWM_Init(&htim4) != HAL_OK)
                Error_Handler();
        pwmConfig.OCMode = TIM_OCMODE_PWM1;
        pwmConfig.OCPolarity = TIM_OCPOLARITY_HIGH;
        pwmConfig.OCNPolarity = TIM_OCPOLARITY_LOW;
        pwmConfig.Pulse = 0;

        if (HAL_TIM_OC_ConfigChannel(&htim4, &pwmConfig, TIM_CHANNEL_1) != HAL_OK)
```

```
                Error_Handler();
        HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_1);
}

void GPIO_Init(void)
{
        GPIO_InitTypeDef led_pin = {0};
        led_pin.Pin = GPIO_PIN_13;
        led_pin.Mode = GPIO_MODE_OUTPUT_PP;
        __HAL_RCC_GPIOC_CLK_ENABLE();
        HAL_GPIO_Init(GPIOC, &led_pin);

        GPIO_InitTypeDef dir_pin = {0};
        dir_pin.Pin = GPIO_PIN_7;
        dir_pin.Mode = GPIO_MODE_OUTPUT_PP;
        __HAL_RCC_GPIOB_CLK_ENABLE();
        HAL_GPIO_Init(GPIOB, &dir_pin);

}



void Error_Handler(void)
{
        while(1)
        {
                HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13);
                HAL_Delay(100);
        }
}
```

A.2. pidcontrol.c

```
#include "pidcontrol.h"

void pid_init(PIDControl *pid, int32_t *ref, int32_t *pos)
{
    pid->reference = ref;
    pid->position = pos;
    pid_reset(pid);
}

void pid_reset(PIDControl *pid)
{
    pid->sum_error = 0;
    pid->pre_error = 0;


    pid->Ki = 0.0;

}

void pid_tune(PIDControl *pid, float kp, float ki, float kd)
{
    pid->Kp = kp;
    pid->Ki = ki;
    pid->Kd = kd;
}

void pid_compute(PIDControl *pid)
{
    int error;
    float up, ui, ud;
    error = *pid->reference - *pid->position;
    up = pid->Kp * error;

    pid->sum_error += error;
    if (pid->sum_error > EMAX) pid->sum_error = EMAX;
    if (pid->sum_error < -EMAX) pid->sum_error = -EMAX;
    ui = pid->Ki * pid->sum_error;

    ud = pid->Kd * (error - pid->pre_error);
    pid->pre_error = error;
```

```
    pid->uout = up + ui + ud;
    if (pid->uout > UMAX) pid->usat = UMAX;
    else if (pid->uout < -UMAX) pid->usat = -UMAX;
    else pid->usat = (int)pid->uout;
}
```

A.3. profile.c

```
#include "profile.h"
#include <math.h>

void profile_init(Profile *profile, int32_t *reference)
{
    profile_reset(profile);
    profile->setpoint = reference;
}

void profile_reset(Profile *profile)
{
    profile->type = TRIANGULAR;
    profile->state = STOPPED;
}

void profile_start(Profile *profile, int32_t despos, int32_t desvel)
{
    if (profile->type == TRIANGULAR)
            profile_start_triangular(profile, despos, desvel);
    if (profile->type == TRAPEZOIDAL)
            profile_start_trapezoidal(profile, despos, desvel);
    if (profile->type == PARABOLIC)
            profile_start_parabolic(profile, despos, desvel);
    if (profile->type == POLINOMIAL)
            profile_start_polinomial(profile, despos, desvel);
    profile->offset = *profile->setpoint;
}

void profile_execute(Profile *profile)
{
    if (profile->state == STOPPED)
            return;
    if (profile->type == TRIANGULAR) profile_exec_triangular(profile);
    if (profile->type == TRAPEZOIDAL) profile_exec_trapezoidal(profile);
    if (profile->type == PARABOLIC) profile_exec_parabolic(profile);
    if (profile->type == POLINOMIAL) profile_exec_polinomial(profile);
}

//======== START TRIANGULAR=============================================
void profile_start_triangular(Profile *ptr, int32_t pos, int32_t vel)
{
    double T, a;
    T = millround(fabs(2.0 * pos) / fabs(vel));
    a = sign(pos) * fabs(2.0 * vel) / T;
    ptr->period = (int)(T * 1000);
    ptr->q2 = a / 2.0;
    ptr->q1 = a * T;
    ptr->q0 = a * T * T / 4.0;
    ptr->k = 0;
    ptr->state = RUNNING;
}

void profile_exec_triangular(Profile *ptr)
{
    double t, pos;
    t = ptr->k * TS;
    if (ptr->k <= (ptr->period / 2))
            pos = ptr->q2 * t * t;
    else
            pos = -ptr->q2*t*t + ptr->q1*t - ptr->q0;
    *ptr->setpoint = (int)(pos + ptr->offset);
    ptr->k = ptr->k + 1;
    if (ptr->k > ptr->period)
            ptr->state = STOPPED;
```

```
}

//========== END TRIANGULAR=============================================


//========== START TRAPEZOIDAL=========================================

void profile_start_trapezoidal(Profile *ptr, int32_t pos, int32_t vel)
{
    double T, a;
    T = millround(fabs(3.0 * pos) / fabs(2.0 * vel));
    a = sign(pos) * fabs(3.0 * vel) / T;
    ptr->period = (int)(T * 1000);
    ptr->q2 = a / 2.0;
    ptr->q1 = a * T / 3.0;
    ptr->q0 = a * T * T / 18.0;
    ptr->k = 0;
    ptr->state = RUNNING;
}

void profile_exec_trapezoidal(Profile *ptr)
{
    double t, pos;
    t = ptr->k * TS;

    if (ptr->k <= (ptr->period / 3))
            pos = ptr->q2 * t * t;
    else if (ptr->k <= (2 * ptr->period / 3))
            pos = ptr->q1 * t - ptr->q0;
    else
            pos = -ptr->q2*t*t + 3.0*ptr->q1*t - 5.0*ptr->q0;

    *ptr->setpoint = (int)(pos + ptr->offset);
    ptr->k = ptr->k + 1;
    if (ptr->k > ptr->period)
            ptr->state = STOPPED;
}

//========== END TRAPEZOIDALL==========================================

//========== START PARABOLIC===========================================
void profile_start_parabolic(Profile *ptr, int32_t pos, int32_t vel)
{
    double T, a;
    T = millround(fabs(3.0 * pos) / fabs(2.0 * vel));
    a = sign(pos) * fabs(4.0 * vel) / T;
    ptr->period = (int)(T * 1000);

    ptr->q3 = -a/(3.0 * T);
    ptr->q2 = a / 2.0;
    ptr->q1 = 0.0;
    ptr->q0 = 0.0;

    ptr->k = 0;
    ptr->state = RUNNING;
}

void profile_exec_parabolic(Profile *ptr)
{
    double t, pos;
    t = ptr->k * TS;
    pos = ptr->q3 *t*t*t + ptr->q2 *t*t;
    *ptr->setpoint = (int)(pos + ptr->offset);
    ptr->k = ptr->k + 1;
    if (ptr->k > ptr->period)
            ptr->state = STOPPED;
}

//========== END PARABOLIC=============================================


//========== START POLINOMIAL==========================================

void profile_start_polinomial(Profile *ptr, int32_t pos, int32_t vel)
{
    double T, a;
```

```
    T = millround(fabs(9.0 * pos) / fabs(7.0*vel));
    a = sign(pos) * fabs(6.0 * vel) / T;
    ptr->w0 = (2.0* PI* 1.0)/ T;
    ptr->period = (int)(T * 1000);


    ptr->q3 = a/2.0;
    ptr->q2 = a*T*T;
    ptr->q1 = a*T;
    ptr->q0 = 1/T;



    ptr->k = 0;
    ptr->state = RUNNING;
}

void profile_exec_polinomial(Profile *ptr)
{
    double t, pos;
    t = ptr->k * TS;

    if (ptr->k <= (ptr->period / 3))
            pos = (-ptr->q3 *ptr->q0 )* t * t *t + ptr->q3*t*t ;
    else if (ptr->k <= (2 * ptr->period / 3))
            pos = (1.0/6.0)*ptr->q1 * t - (1.0/54.0)*ptr->q2;
    else
            pos = -(ptr->q3 *ptr->q0)*t*t*t + (ptr->q0*ptr->q1*t*t) - (1.0/2.0)*ptr->q1*t + (7.0/54.0)*ptr

    *ptr->setpoint = (int)(pos + ptr->offset);
    ptr->k = ptr->k + 1;
    if (ptr->k > ptr->period)
            ptr->state = STOPPED;
}

//============== END POLINOMIAL=================================

double millround(double value)
{
    return ((int)(value * 1000.0)) / 1000.0;
}

double sign(double value)
{
    if (value >= 0)
            return 1.0;
    else
            return -1.0;
}
```

A.4. main.h

```
#ifndef __MAIN_H
#define __MAIN_H

#include "stm32f4xx_hal.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <FreeRTOS.h>
#include <task.h>
#include <timers.h>
#include <semphr.h>
#include <queue.h>
#include "pidcontrol.h"
#include "profile.h"
#define SAMPLES     2000
#endif
```

A.5. pidcontrol.h

```
#ifndef INC_PIDCONTROL_H_
#define INC_PIDCONTROL_H_

#include <stdint.h>

#define TS          0.001
#define EMAX        1000
#define UMAX        1000

typedef struct
{
    int32_t *reference;
    int32_t *position;
    float Kp, Ki, Kd;
    int sum_error;
    int pre_error;
    float uout;
    int usat;
} PIDControl;

void pid_init(PIDControl *, int32_t *, int32_t *);
void pid_reset(PIDControl *);
void pid_tune(PIDControl *, float, float, float);
void pid_compute(PIDControl *);

#endif
```

A.6. profile.h

```
#ifndef INC_PROFILE_H_
#define INC_PROFILE_H_

#include <stdint.h>
#include <math.h>

#define TS  0.001
#define PI 3.1415926535

typedef enum ptype{TRIANGULAR, TRAPEZOIDAL, PARABOLIC,POLINOMIAL} ProfileType;

typedef enum pstate{STOPPED, RUNNING} ProfileState;

typedef struct
{
    ProfileType type;
    ProfileState state;
    int32_t *setpoint;
    int32_t offset;
    uint16_t k, period;
    double q0, q1, q2, q3, w0;
} Profile;

void profile_init(Profile *, int32_t *);
void profile_reset(Profile *);
void profile_start(Profile *, int32_t, int32_t);
void profile_execute(Profile *);

void profile_start_triangular(Profile *, int32_t, int32_t);
void profile_exec_triangular(Profile *);

void profile_start_trapezoidal(Profile *, int32_t, int32_t);
void profile_exec_trapezoidal(Profile *);

void profile_start_parabolic(Profile *, int32_t, int32_t);
void profile_exec_parabolic(Profile *);

void profile_start_polinomial(Profile *, int32_t, int32_t);
void profile_exec_polinomial(Profile *);

double millround(double);
double sign(double);
```

```
#endif
```

A.7. FreeRTOSConfig.h

```
#ifndef FREERTOS_CONFIG_H
#define FREERTOS_CONFIG_H

/*-----------------------------------------------------------
 * Application specific definitions.
 *
 * These definitions should be adjusted for your particular hardware and
 * application requirements.
 *
 * THESE PARAMETERS ARE DESCRIBED WITHIN THE 'CONFIGURATION' SECTION OF THE
 * FreeRTOS API DOCUMENTATION AVAILABLE ON THE FreeRTOS.org WEB SITE.
 *
 * See http://www.freertos.org/a00110.html
 *----------------------------------------------------------*/

/* Ensure stdint is only used by the compiler, and not the assembler. */
#ifdef __ICCARM__
    #include <stdint.h>
    extern uint32_t SystemCoreClock;
#endif
extern uint32_t SystemCoreClock;

#define configUSE_PREEMPTION                    1
#define configUSE_IDLE_HOOK                     1
#define configUSE_TICK_HOOK                     0
#define configCPU_CLOCK_HZ                      ( SystemCoreClock )
#define configTICK_RATE_HZ                      ( ( TickType_t ) 1000 )
#define configMAX_PRIORITIES                    ( 5 )
#define configMINIMAL_STACK_SIZE        ( ( unsigned short ) 130 )
#define configTOTAL_HEAP_SIZE                   ( ( size_t ) ( 75 * 1024 ) )
#define configMAX_TASK_NAME_LEN                 ( 10 )
#define configUSE_TRACE_FACILITY        1
#define configUSE_16_BIT_TICKS                  0
#define configIDLE_SHOULD_YIELD                 1
#define configUSE_MUTEXES                       1
#define configQUEUE_REGISTRY_SIZE       8
#define configCHECK_FOR_STACK_OVERFLOW  0
#define configUSE_RECURSIVE_MUTEXES     1
#define configUSE_MALLOC_FAILED_HOOK    0
#define configUSE_APPLICATION_TASK_TAG  0
#define configUSE_COUNTING_SEMAPHORES   1
#define configGENERATE_RUN_TIME_STATS   0

/* Co-routine definitions. */
#define configUSE_CO_ROUTINES           0
#define configMAX_CO_ROUTINE_PRIORITIES ( 2 )

/* Software timer definitions. */
#define configUSE_TIMERS                        1
#define configTIMER_TASK_PRIORITY       ( 2 )
#define configTIMER_QUEUE_LENGTH        10
#define configTIMER_TASK_STACK_DEPTH    ( configMINIMAL_STACK_SIZE * 2 )

/* Set the following definitions to 1 to include the API function, or zero
to exclude the API function. */
#define INCLUDE_vTaskPrioritySet        1
#define INCLUDE_uxTaskPriorityGet       1
#define INCLUDE_vTaskDelete                     1
#define INCLUDE_vTaskCleanUpResources   1
#define INCLUDE_vTaskSuspend                    1
#define INCLUDE_vTaskDelayUntil                 1
#define INCLUDE_vTaskDelay                      1

/* Cortex-M specific definitions. */
#ifdef __NVIC_PRIO_BITS
    /* __BVIC_PRIO_BITS will be specified when CMSIS is being used. */
    #define configPRIO_BITS                 __NVIC_PRIO_BITS
```

```
#else
    #define configPRIO_BITS                 4        /* 15 priority levels */
#endif

/* The lowest interrupt priority that can be used in a call to a "set priority"
function. */
#define configLIBRARY_LOWEST_INTERRUPT_PRIORITY                         0xf

/* The highest interrupt priority that can be used by any interrupt service
routine that makes calls to interrupt safe FreeRTOS API functions.  DO NOT CALL
INTERRUPT SAFE FREERTOS API FUNCTIONS FROM ANY INTERRUPT THAT HAS A HIGHER
PRIORITY THAN THIS! (higher priorities are lower numeric values. */
#define configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY        5

/* Interrupt priorities used by the kernel port layer itself.  These are generic
to all Cortex-M ports, and do not rely on any particular library functions. */
#define configKERNEL_INTERRUPT_PRIORITY
( configLIBRARY_LOWEST_INTERRUPT_PRIORITY << (8 - configPRIO_BITS) )
/* !!!! configMAX_SYSCALL_INTERRUPT_PRIORITY must not be set to zero !!!!
See http://www.FreeRTOS.org/RTOS-Cortex-M3-M4.html. */
#define configMAX_SYSCALL_INTERRUPT_PRIORITY
( configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY << (8 - configPRIO_BITS) )

/* Normal assert() semantics without relying on the provision of an assert.h
header file. */
#define configASSERT( x ) if( ( x ) == 0 ) { taskDISABLE_INTERRUPTS(); for( ;; ); }

/* Definitions that map the FreeRTOS port interrupt handlers to their CMSIS
standard names. */
#define vPortSVCHandler SVC_Handler
#define xPortPendSVHandler PendSV_Handler
#define xPortSysTickHandler SysTick_Handler

#endif /* FREERTOS_CONFIG_H */
```

B. Código de implementación en Matlab.

B.1. Identificación y Controlador PD

```
clc
clear all
close all

out=[];
kp_id=[];
tp=[];
mp=[];
%% Datos
[num1,txt,raw]=xlsread('Identification.xlsx','5.0');
[num2,txt,raw]=xlsread('Identification.xlsx','OUTKP5');

out=num2(:,8);%vector de salida
Ref=1000.0;    %Referencia
response=num1(:,1);%vector de respuesta de identificación
kp_id= [5.0]; % ganancia de identificación
td=0;

max_out=max(out);

%Picos entrada y salida
[pk1_in,loc1_in]= findpeaks(response);
[pk_out,loc_out]= findpeaks(out);

%Arreglos para entrada y salida
 peaks_In=[pk1_in loc1_in]
 peaks_Out=[pk_out loc_out];
 picos_in = peaks_In(:,1)
 picos_Out = peaks_Out(:,1)
 picos_OutT = peaks_Out(:,2);
```

```
%Pico máximo entrada
[m_in1 n_in1]= size(peaks_In);
[pmax,idx]= max(picos_in); %pmax= pico maximo ; idx= posicion del pico
tiempoPico = peaks_In(idx,2);

%Pico máximo salida
[m_out1 n_out1]= size(peaks_Out);
[pmaxOut,idx]= max(picos_Out); %pmax= pico maximo ; idx= posicion del pico
tiempoPicoO = peaks_Out(idx,2);

%%
%%% Identificacion propuesta de los parametros%%%
%%% de la respuesta transitoria de 2 orden %%%
MP=((pmax-Ref)/Ref)*100
mp(1,1)=MP;
TP(1,1)= tiempoPico*10^-3;
tp(1,1)=TP;

Data_table=table(Ref,kp_id(1,1),MP,TP)

Kp_id = kp_id(1,1);
Tp = tp(1,1);
Mp = mp(1,1)/100;
zeta_id = abs(log(Mp))/sqrt(pi^2 +(log(Mp))^2);
omega_id = pi/(Tp * sqrt(1- zeta_id^2));

%% Tr deseado

omega_d = omega_id*(sqrt(1-zeta_id^2));
sigma = zeta_id*omega_id;
beta = atan(omega_d/sigma);
Tr_d = (pi-beta)/(omega_d);
Tau=1/(zeta_id*omega_id); %%Constante de tiempo
Ts_5percent = 3/(zeta_id*omega_id); %% Tiempo de establecimiento con 5% de tolerancia
Ts_2percent = 4/(zeta_id*omega_id); %% de establecimiento con 2% de tolerancia

%% Paramtetros del motor con servoamplificador %
Jm=1.0;
Ka = 1.0;
Kb = 2 * zeta_id * omega_id;
Kt = omega_id^2/Kp_id;

%% Polos deseados de lazo cerrado %
Tr =0.015;
%Tr = Tr_d;
Mp_des = 0.2;
Ts=0.001;
zeta = abs(log(Mp_des))/sqrt(pi^2 + (log(Mp_des))^2);
omega = (pi - atan2(sqrt(1 - zeta^2), zeta))/(Tr * sqrt(1 - zeta ^2));

%%

%% Compute closed-loop poles location %
%sigma= -zeta* omega + omega * sqrt(1 - zeta^2) * 1i;
% Evaluate phase condition %
phi1 = 180 - atan2d(sqrt(1 - zeta^2), zeta );
phi2 = atan2d(omega * sqrt(1 - zeta^2), Kb/Jm - zeta*omega);
phi3 = 180 + phi1 + phi2;
Td = tand(phi3)/(omega*sqrt(1 - zeta^2) + zeta*omega*tand(phi3));
%%

%% Condición de magnitud %%%
num = omega*sqrt(Kb^2 - 2*Kb*Jm*zeta*omega + Jm^2*omega^2);
den = Ka*Kt*sqrt(1 - 2*Td*zeta*omega + Td^2*omega^2);
Kp = num/den;
Kd = (Td*Kp)/0.001;
% C = tf([Td 1],[1]);
% G = tf(Ka*Kt,[Jm Kb 0]);
C=tf([Kp*Td Kp], 1);   %Controlador PD
G=tf(Ka*Kt,[Jm Kb 0]); % Planta con Integrador de posición
H = [1];
%% Data table
Data=table(Td,Kp,Kd)
%%
%%
%%%Grafica de entrada experimental Kp para identificación%%%
figure('Name','Entrada experimental de identificación Kp')
```

```
%subplot(4,1,1)
plot(response,'color',[0 0.4470 0.7410]);
xlim([0 400])
ylim([0 1600])
xlabel('Tiempo [ms]');
ylabel('Posición [cuentas de encoder]');
hold on %TP
P8=[tiempoPico 0];P9=[tiempoPico pmax];
plot([P8(1) P9(1)],[P8(2) P9(2)],'-.k');
hold on %MP
plot(tiempoPico,pmax,'o','MarkerEdgeColor',[0 0.4470 0.7410],'MarkerFaceColor',
[0 0.4470 0.7410],'MarkerSize',6);
legend(['Respuesta'],['Tp = ',num2str(tiempoPico,5),'ms'],['Mp = ',num2str(MP,4),'%'],
'AutoUpdate','off', 'Location', 'Northeast');
hold on
yline(Ref,':k');
title('')
%grid on
hold on


%%
%%% Lugar de las raices%%%
figure('Name', 'Lugar de las raices')
%subplot(4,1,2)
rlocus(C*G);
title('')
grid on
%%
%%% Respuesta al escalon unitario%%%
figure('Name','Respuesta ideal del controlador PD de posición ')
CLTF = feedback(C*G,H);
%subplot(4,1,3)
opt = stepDataOptions('InputOffset',0,'StepAmplitude',1000);
step(CLTF,opt)
%xline(0.0184,':k');
xlim([0 0.3])
ylim([0 1400])
xlabel('Tiempo [s]');
ylabel('Posición [cuentas de encoder]');
S_CLTF = stepinfo(CLTF,'RiseTimeThreshold',[0.0 1.0]);
S_CLTF_MP=S_CLTF.Overshoot;
S_CLTF_ST=S_CLTF.SettlingTime;
S_CLTF_RT=S_CLTF.RiseTime;
S_CLTF_Peak=(S_CLTF.Peak)*1000;
S_CLTF_TPeak=S_CLTF.PeakTime;
S_CLTF_AmSetT=(S_CLTF.SettlingMin+(S_CLTF.SettlingMin*0.02))*1000;
S_CLTF_SteadyS=1000-(1000*0.02);
%legend(Leyenda);
hold on %TR
P1=[S_CLTF_RT 0];P2=[S_CLTF_RT 1000];
plot([P1(1) P2(1)],[P1(2) P2(2)],'-.k')
hold on %MP
plot(S_CLTF_TPeak,S_CLTF_Peak,'o','MarkerEdgeColor',[0 0.4470 0.7410],'MarkerFaceColor',
[0 0.4470 0.7410],'MarkerSize',6);
hold on %SteadyState
plot(S_CLTF_ST,S_CLTF_SteadyS,'o','MarkerEdgeColor',[0 0.4470 0.7410],'MarkerSize',6);
legend(['Respuesta'],
['Tr = ',num2str(S_CLTF_RT,5),'s'],['Mp = ',num2str(S_CLTF.Overshoot,4),'%'],
['St = ',num2str(S_CLTF.SettlingTime,5),'s'], 'Location', 'Northeast');

title('')
%grid on

%%% Salida experimental usando las ganancias obtenidas %%%
MP_Out = ((pmaxOut-Ref)/Ref)*100;
St_max = out(length(out));
deltaOut=abs(St_max - Ref);
maxError = 1000*0.02; % 2%
St_OutMin = Ref - maxError;
St_OutMax = Ref + maxError;
[row,col] = find( ((out>= St_OutMin)&(out< Ref)) | ((out<= St_OutMax)&(out> Ref)) );
rowSt = [];
St_array=[];
St_Am_array=[];
sizeRow = length(row);
numPeaks = length(picos_Out);
```

```
for count= 1:sizeRow
        rowSt(count,1) = row(count);
end

if(numPeaks == 3)
        for count2 = 1:sizeRow
            if((rowSt(count2,1)> picos_OutT(1,1)) & (rowSt(count2,1)< picos_OutT(2,1)) )
                St_array(count2,1) = rowSt(count2,1);
                St_Am_array(count2,1)=out(St_array(count2,1),1);
            end
        end
        for count3=1:length(St_Am_array)
            if(St_Am_array(count3,1)>979 & St_Am_array(count3,1)<985)
                Stout_Am = St_Am_array(count3,1);
                St_out = St_array(count3,1);
            end
        end
elseif(numPeaks == 2)
        for count2 = 1:sizeRow
            if((rowSt(count2,1)> picos_OutT(1,1)) & (rowSt(count2,1)< picos_OutT(2,1)) )
                St_array(count2,1) = rowSt(count2,1);
                St_Am_array(count2,1)=out(St_array(count2,1),1);
            end
        end
        for count3=1:length(St_Am_array)
            if(St_Am_array(count3,1)>979 & St_Am_array(count3,1)<986)
                Stout_Am = St_Am_array(count3,1);
                St_out= St_array(count3,1);
            end
        end
elseif(numPeaks == 1)
        for count2 = 1:sizeRow
            if((rowSt(count2,1)> picos_OutT(1,1))  )
                St_array(count2,1) = rowSt(count2,1);
                St_Am_array(count2,1)=out(St_array(count2,1),1);
            end
        end
        for count3=1:length(St_Am_array)
            if(St_Am_array(count3,1)>=1002 & St_Am_array(count3,1)<1003)
                Stout_Am = St_Am_array(count3,1);
                St_out= St_array(count3,1);
            end
        end

else
        for count2 = 1:sizeRow
            if(numPeaks==0 & rowSt(count2,1)< Ref)
                St_array(count2,1) = rowSt(count2,1);
                St_Am_array(count2,1)=out(St_array(count2,1),1);
            elseif(numPeaks>=4 & (rowSt(count2,1)> picos_OutT(1,1)) &
            (rowSt(count2,1)< picos_OutT(2,1)) )
                St_array(count2,1) = rowSt(count2,1);
                St_Am_array(count2,1)=out(St_array(count2,1),1);
            end
        end
        for count3=1:length(St_Am_array)
            if(St_Am_array(count3,1)>979 & St_Am_array(count3,1)<985)
                Stout_Am = St_Am_array(count3,1);
                St_out= St_array(count3,1);
                tiempoPicoO=0;
                pmaxOut=0;
                MP_Out=0;
            elseif(St_Am_array(count3,1)>1010 & St_Am_array(count3,1)<1020)
                Stout_Am = St_Am_array(count3,1);
                St_out= St_array(count3,1);
            end
        end
end

 St_AmOut = Stout_Am;
 [row,col] = find(out>=(Ref) |out>=(Ref-1));
 Tr_Out = row(1);

figure('Name','Respuesta experimental con PD de posición')
%subplot(4,1,4)
plot(out,'color',[0 0.4470 0.7410]);
xlim([0 140])
```

```
ylim([0 1400])
xlabel('Tiempo [ms]');
ylabel('Posición [cuentas de encoder]')
title('')
hold on %Tr
P6=[Tr_Out 0];P7=[Tr_Out 1000];
plot([P6(1) P7(1)],[P6(2) P7(2)],'-.k');
hold on %MP
if(tiempoPicoO==0)
    plot(tiempoPicoO,pmaxOut,'o','MarkerEdgeColor',[0 0.4470 0.7410],
    'MarkerFaceColor',[0 0.4470 0.7410],'MarkerSize',0.1);
else
    plot(tiempoPicoO,pmaxOut,'o','MarkerEdgeColor',[0 0.4470 0.7410],
    'MarkerFaceColor',[0 0.4470 0.7410],'MarkerSize',6);
end
%plot(tiempoPicoO,pmaxOut,'o','MarkerEdgeColor',[0 0.4470 0.7410],
'MarkerFaceColor',[0 0.4470 0.7410],'MarkerSize',6);
hold on %SteadyState
plot(St_out,St_AmOut,'o','MarkerEdgeColor',[0 0.4470 0.7410],'MarkerSize',6);
legend(['Respuesta'],['Tr = ',num2str(Tr_Out,5),'ms'],['Mp = ',num2str(MP_Out,4),'%'],
['St = ',num2str(St_out,5),'ms'],'AutoUpdate','off', 'Location', 'Northeast');
hold on
yline(Ref,':k');
```

B.2. Comprobación Triangular

```
clc
clear all
close all
theta_res=[];
alpha_res=[];
voltage_res = [];
current_res = [];
power_res = [];
energy_res = [];
[num,txt,raw]=xlsread('perfiles.xlsx','Hoja15');


theta_res1=num(:,1);
theta_res=[];
voltage_res = num(:,2);
current_res = num(:,3);
power_res = num(:,5);
energy_res = num(:,7);


thetaF1  = 40000;%cuentas encoder
tiempo_deseado =2; %seg
omegaMax_deseado =  2.0 *thetaF1/(tiempo_deseado);
omegaMax1 = omegaMax_deseado;%cuentas de encoder por segundo


%%
% CONVERSION rad   rad/s  rad/s^2
encoder_counts= 2400; %cuentas por revolucion
rev = thetaF1/encoder_counts; % revoluciones solicitadas
rev_pm = omegaMax1/encoder_counts;% revoluciones por segundo solicitadas
Rpm= ((rev_pm)/1)*60; % Revoluciones por minuto
radF = rev*(2*pi); %posición max solicitada en radianes
omega_rad_s = Rpm*((2*pi)/60); %velocidad solicitada en radianes

% CONVERSION DATA
for i=1:2000
   theta_res(i,1)= (theta_res1(i,1)/encoder_counts)*(2*pi);
end
%%

thetaF = radF; %radianes
omegaMax = omega_rad_s; %radianes/s

T = 2*thetaF/omegaMax;
a= 2.0 * omegaMax/T;
a1 = (omegaMax^2 )/ thetaF;
```

```
c2 = a / 2.0;
c1 = a * T;
c0 = a * T^2 /4.0;
t1 = 1000 * T / 2;

Ts = 0.001;
OMEGA = 40.0;
q0 = 2.0 * OMEGA/(OMEGA * Ts +2.0);
q1 = -(OMEGA * Ts - 2.0)/(OMEGA * Ts +2.0);

samples = length(theta_res);
theta = zeros(samples,1);
omega = zeros (samples,1);
omega_res = zeros(samples,1);
alpha = zeros (samples,1);
alpha_res = zeros(samples,1);
jerk_res = zeros(samples,1);

time = zeros(samples ,1);

sum_sqr_error = 0;
for k = 2 : samples
    t = (k -1) * Ts;
    time(k) = t;
    omega_res(k) = q0 * (theta_res(k)- theta_res(k - 1)) + q1 *omega_res(k - 1);
    alpha_res(k) = q0 * (omega_res(k)-omega_res(k-1))+ q1 *alpha_res(k - 1);
    jerk_res(k) = q0 * (alpha_res(k)-alpha_res(k-1))+ q1 *jerk_res(k - 1);
    if(k <= t1)
       theta(k) = c2 * t^2;
       omega(k) = a * t;
       alpha(k) = a;
    else
        theta(k) = -c2*t^2 + a*t*T - c0;
        omega(k) = -a *t + a * T;
        alpha(k) = -a;

    end
end
sum_sqr_error = sum_sqr_error + (floor(theta(k)) - theta_res(k))^2;


format long

JerkAverage= abs(mean(jerk_res));
JerkMAxpos= abs(max(jerk_res));
JerkMAxneg= abs(min(jerk_res));

omegaMax_res= abs(max(omega_res))
alphaMax_res= abs(max(alpha_res))
if(JerkMAxpos> JerkMAxneg)
    JerkMAx = JerkMAxpos
else
    JerkMAx = JerkMAxneg
end

voltajeMax = abs(max(voltage_res));
total_energy= energy_res(2000)
ERMS = sqrt(sum_sqr_error)


figure('Name','Motion profile ata');
subplot(4,1,1);
plot(time,theta,time,theta_res);
legend(['\theta_d = ',num2str(theta(2000),6)],['\theta_f = ',num2str(theta_res(2000),6)],
'off', 'Location', 'Southeast');
xlabel('Tiempo [s]');
ylabel('Posición [rad]');
grid on
subplot(4,1,2);
plot(time , omega,time,omega_res);
legend(['\omega_d = ',num2str(omegaMax,5)],['\omega_{max} = ',num2str(omegaMax_res,5)],
'off', 'Location', 'Northeast');
xlabel('Tiempo [s]');
ylabel('Velocidad angular[rad/s]');
grid on
subplot(4,1,3);
```

```
plot(time , alpha,time,alpha_res);
legend('\alpha_{deseada}' ,'\alpha_{obtenida}','off', 'Location', 'Northeast');
xlabel('Tiempo [s]');
ylabel('Aceleración angular [rad/s^{2}]');
grid on
subplot(4,1,4);
plot(time ,jerk_res,'color',[0.8500 0.3250 0.0980]);
legend(['Jerk_{max} = ',num2str(JerkMAx,5)],'off', 'Location', 'Northeast');
xlabel('Tiempo [s]');
ylabel('Jerk [rad/s^{3}]');
grid on


figure('Name','Datos');
subplot(4,1,1);
plot(time,voltage_res);
legend(['Voltaje_{max} = ',num2str(voltajeMax,4)],'off', 'Location', 'Northeast');
xlabel('Tiempo [s]');
ylabel('Voltaje [V]');
grid on
subplot(4,1,2);
plot(time,current_res);
xlabel('Tiempo [s]');
ylabel('Corriente [A]');
grid on
subplot(4,1,3);
plot(time,power_res);
xlabel('Tiempo [s]');
ylabel('Potencia [W]');
grid on
subplot(4,1,4);
plot(time,energy_res);
legend(['Energía= ',num2str(energy_res(2000),4)],'off', 'Location', 'Southeast');
xlabel('Tiempo [s]');
ylabel('Energía [J]');
grid on
```

B.3. Comprobación Trapezoidal

```
clc
clear all
close all
theta_res=[];
alpha_res=[];
voltage_res = [];
current_res = [];
power_res = [];
energy_res = [];
[num,txt,raw]=xlsread('perfiles.xlsx','Hoja16');
theta_res1=num(:,1);
theta_res=[];
voltage_res = num(:,2);
current_res = num(:,3);
power_res = num(:,5);
energy_res = num(:,7);


thetaF1  = 40000;%cuentas encoder
tiempo_deseado =2; %seg
omegaMax_deseado =  3.0 *thetaF1/(2.0*tiempo_deseado);
omegaMax1 = omegaMax_deseado;%cuentas de encoder por segundo


%%
% CONVERSION rad   rad/s  rad/s^2
encoder_counts= 2400; %cuentas por revolucion
rev = thetaF1/encoder_counts; % revoluciones solicitadas
rev_pm = omegaMax1/encoder_counts;% revoluciones por segundo solicitadas
Rpm= ((rev_pm)/1)*60; % Revoluciones por minuto
radF = rev*(2*pi); %posición max solicitada en radianes
omega_rad_s = Rpm*((2*pi)/60); %velocidad solicitada en radianes

% CONVERSION DATA
```

```
for i=1:2000
   theta_res(i,1)= (theta_res1(i,1)/encoder_counts)*(2*pi);
end
%%

thetaF = radF; %radianes
omegaMax = omega_rad_s; %radianes/s

T = 3 * thetaF / (2* omegaMax);
a = 3 *omegaMax / T;
a1 = 2* (omegaMax^2 / thetaF);
c2 = a / 2.0;
c1 = a * T;
c0 = a * T * T /18.0;
t1 = 1000 * T / 3;
t2 = 2000 * T / 3;

%Filtro
Ts = 0.001;
%OMEGA = 40.0;
OMEGA = 40.0;
q0 = 2.0 * OMEGA/(OMEGA * Ts +2.0);
q1 = -(OMEGA * Ts - 2.0)/(OMEGA * Ts +2.0);

samples = length(theta_res);
theta = zeros(samples,1);
omega = zeros (samples,1);
alpha = zeros (samples,1);
omega_res = zeros(samples,1);
alpha_res = zeros(samples,1);
jerk_res = zeros(samples,1);
time = zeros(samples ,1);

sum_sqr_error = 0;

for k = 2 : samples
    t = (k -1) * Ts;
    time(k) = t;
    omega_res(k) = q0 * (theta_res(k)- theta_res(k - 1)) + q1 *omega_res(k - 1);
    alpha_res(k) = q0 * (omega_res(k)-omega_res(k-1))+ q1 *alpha_res(k - 1);
    jerk_res(k) = q0 * (alpha_res(k)-alpha_res(k-1))+ q1 *jerk_res(k - 1);
    if(k <= t1)
       theta(k) = c2 * t * t;
       omega(k) = a * t;
       alpha(k) = a;

    elseif(k<= t2)
        theta(k) = t * c1/3.0 - c0;
        omega(k) = a * T / 3.0;
        alpha(k) = 0;
    else
        theta(k) = -(c2*t*t) + (c1*t) - (5*c0);
        omega(k) = -(a *t)  + (a * T);
        alpha(k) = -a;
    end
end
format long
sum_sqr_error = sum_sqr_error + (floor(theta(k)) - theta_res(k))^2;

format long

omegaMax_res= abs(max(omega_res))
alphaMax_res= abs(max(alpha_res))
JerkAverage= abs(mean(jerk_res));
JerkMAxpos= abs(max(jerk_res));
JerkMAxneg= abs(min(jerk_res));
if(JerkMAxpos> JerkMAxneg)
    JerkMAx = JerkMAxpos
else
    JerkMAx = JerkMAxneg
end
voltajeMax = abs(max(voltage_res));
total_energy= energy_res(2000)
ERMS = sqrt(sum_sqr_error)

figure('Name','Motion profile data');
subplot(4,1,1);
```

```
plot(time,theta,time,theta_res);
legend(['\theta_d = ',num2str(theta(2000),6)],['\theta_f = ',num2str(theta_res(2000),6)],
'off', 'Location', 'Southeast');
xlabel('Tiempo [s]');
ylabel('Posición [rad]');
grid on
subplot(4,1,2);
plot(time , omega,time,omega_res);
legend(['\omega_d = ',num2str(omegaMax,5)],['\omega_{max} = ',num2str(omegaMax_res,5)],
'off', 'Location', 'Northeast');
xlabel('Tiempo [s]');
ylabel('Velocidad angular[rad/s]');
grid on
subplot(4,1,3);
plot(time , alpha,time,alpha_res);
legend('\alpha_{deseada}' ,'\alpha_{obtenida}','off', 'Location', 'Northeast');
xlabel('Tiempo [s]');
ylabel('Aceleración angular [rad/s^{2}]');
grid on
subplot(4,1,4);
plot(time ,jerk_res,'color',[0.8500 0.3250 0.0980]);
legend(['Jerk_{max} = ',num2str(JerkMAx,5)],'off', 'Location', 'Northeast');
xlabel('Tiempo [s]');
ylabel('Jerk [rad/s^{3}]');
grid on


figure('Name','Datos');
subplot(4,1,1);
plot(time,voltage_res);
legend(['Voltaje_{max} = ',num2str(voltajeMax,4)],'off', 'Location', 'Northeast');
xlabel('Tiempo [s]');
ylabel('Voltaje [V]');
grid on
subplot(4,1,2);
plot(time,current_res);
xlabel('Tiempo [s]');
ylabel('Corriente [A]');
grid on
subplot(4,1,3);
plot(time,power_res);
xlabel('Tiempo [s]');
ylabel('Potencia [W]');
grid on
subplot(4,1,4);
plot(time,energy_res);
legend(['Energía= ',num2str(energy_res(2000),4)],'off', 'Location', 'Southeast');
xlabel('Tiempo [s]');
ylabel('Energía [J]');
grid on
```

B.4. Comprobación Parabólico

```
clc
clear all
close all
theta_res=[];
alpha_res=[];
voltage_res = [];
current_res = [];
power_res = [];
energy_res = [];
[num,txt,raw]=xlsread('perfiles.xlsx','Hoja17');
theta_res1=num(:,1);
theta_res=[];
voltage_res = num(:,2);
current_res = num(:,3);
power_res = num(:,5);
energy_res = num(:,7);


thetaF1  = 40000;%cuentas encoder
tiempo_deseado =2; %seg
```

```
omegaMax_deseado =  3.0 *thetaF1/(2.0*tiempo_deseado);
omegaMax1 = omegaMax_deseado;%cuentas de encoder por segundo



%%
% CONVERSION rad   rad/s  rad/s^2
encoder_counts= 2400; %cuentas por revolucion
rev = thetaF1/encoder_counts; % revoluciones solicitadas
rev_pm = omegaMax1/encoder_counts;% revoluciones por segundo solicitadas
Rpm= ((rev_pm)/1)*60; % Revoluciones por minuto
radF = rev*(2*pi); %posición max solicitada en radianes
omega_rad_s = Rpm*((2*pi)/60); %velocidad solicitada en radianes

% CONVERSION DATA
for i=1:2000
   theta_res(i,1)= (theta_res1(i,1)/encoder_counts)*(2*pi);
end
%%

thetaF = radF; %radianes
omegaMax = omega_rad_s; %radianes/s

T= 3.0 *thetaF/(2.0*omegaMax);
a= 4.0 * omegaMax/T;
a1 = (8/3)*(omegaMax^2 / thetaF);

c2 = a / 2.0;
c1 = a /(3* T);
t1 = 1000 * T / 2;

Ts = 0.001;
OMEGA = 40.0;
q0 = 2.0 * OMEGA/(OMEGA * Ts +2.0);
q1 = -(OMEGA * Ts - 2.0)/(OMEGA * Ts +2.0);

samples = length(theta_res);
theta = zeros(samples,1);
omega = zeros (samples,1);
omega_res = zeros(samples,1);
alpha = zeros (samples,1);
alpha_res = zeros(samples,1);
jerk_res = zeros(samples,1);
time = zeros(samples ,1);

sum_sqr_error = 0;
for k = 2 : samples
    t = (k-1 ) * Ts;
    time(k) = t;
    omega_res(k) = q0 * (theta_res(k)- theta_res(k - 1)) + q1 *omega_res(k - 1);
    alpha_res(k) = q0 * (omega_res(k)-omega_res(k-1))+ q1 *alpha_res(k - 1);
    jerk_res(k) = q0 * (alpha_res(k)-alpha_res(k-1))+ q1 *jerk_res(k - 1);



    theta(k)= ((-a /(3* T))*t^3)+((a / 2.0)*t^2);
    omega(k) = ((-a/T)*t^2) + (a*t);
    alpha(k) = -(2*a*t)/T + a;
end

sum_sqr_error = sum_sqr_error + (floor(theta(k)) - theta_res(k))^2;

format long

omegaMax_res= abs(max(omega_res))
alphaMax_res= abs(max(alpha_res))
JerkAverage= abs(mean(jerk_res));
JerkMAxpos= abs(max(jerk_res));
JerkMAxneg= abs(min(jerk_res));
if(JerkMAxpos> JerkMAxneg)
    JerkMAx = JerkMAxpos
else
    JerkMAx = JerkMAxneg
end
voltajeMax = abs(max(voltage_res));
total_energy= energy_res(2000)
ERMS = sqrt(sum_sqr_error)
```

```
figure('Name','Motion profile ata');
subplot(4,1,1);
plot(time,theta,time,theta_res);
legend(['\theta_d = ',num2str(theta(2000),6)],['\theta_f = ',num2str(theta_res(2000),6)],
'off', 'Location', 'Southeast');
xlabel('Tiempo [s]');
ylabel('Posición [rad]');
grid on
subplot(4,1,2);
plot(time , omega,time,omega_res);
legend(['\omega_d = ',num2str(omegaMax,5)],['\omega_{max} = ',num2str(omegaMax_res,5)],
'off', 'Location', 'Northeast');
xlabel('Tiempo [s]');
ylabel('Velocidad angular[rad/s]');
grid on
subplot(4,1,3);
plot(time , alpha,time,alpha_res);
legend('\alpha_{deseada}' ,'\alpha_{obtenida}','off', 'Location', 'Northeast');
xlabel('Tiempo [s]');
ylabel('Aceleración angular [rad/s^{2}]');
grid on
subplot(4,1,4);
plot(time ,jerk_res,'color',[0.8500 0.3250 0.0980]);
legend(['Jerk_{max} = ',num2str(JerkMAx,5)],'off', 'Location', 'Northeast');
xlabel('Tiempo [s]');
ylabel('Jerk [rad/s^{3}]');
grid on


figure('Name','Datos');
subplot(4,1,1);
plot(time,voltage_res);
legend(['Voltaje_{max} = ',num2str(voltajeMax,4)],'off', 'Location', 'Northeast');
xlabel('Tiempo [s]');
ylabel('Voltaje [V]');
grid on
subplot(4,1,2);
plot(time,current_res);
xlabel('Tiempo [s]');
ylabel('Corriente [A]');
grid on
subplot(4,1,3);
plot(time,power_res);
xlabel('Tiempo [s]');
ylabel('Potencia [W]');
grid on
subplot(4,1,4);
plot(time,energy_res);
legend(['Energía= ',num2str(energy_res(2000),4)],'off', 'Location', 'Southeast');
xlabel('Tiempo [s]');
ylabel('Energía [J]');
grid on
```

B.5. Comprobación Polinomial

```
clc
clear all
close all
theta_res=[];
alpha_res=[];
voltage_res = [];
current_res = [];
power_res = [];
energy_res = [];
[num,txt,raw]=xlsread('perfiles.xlsx','Hoja18');
theta_res1=num(:,1);
theta_res=[];
voltage_res = num(:,2);
current_res = num(:,3);
power_res = num(:,5);
energy_res = num(:,7);
```

```
thetaF1  = 40000;%cuentas de encoder
tiempo_deseado =2; %seg
omegaMax_deseado = (9*thetaF1)/(tiempo_deseado*7);
omegaMax1 = omegaMax_deseado;%cuentas de encoder por segundo


%%
% CONVERSION rad   rad/s  rad/s^2
encoder_counts= 2400; %cuentas por revolucion
rev = thetaF1/encoder_counts; % revoluciones solicitadas
rev_pm = omegaMax1/encoder_counts;% revoluciones por segundo solicitadas
Rpm= ((rev_pm)/1)*60; % Revoluciones por minuto
radF = rev*(2*pi); %posición max solicitada en radianes
omega_rad_s = Rpm*((2*pi)/60); %velocidad solicitada en radianes

% CONVERSION DATA
for i=1:2000
   theta_res(i,1)= (theta_res1(i,1)/encoder_counts)*(2*pi);
end
%%

thetaF = radF; %radianes
omegaMax = omega_rad_s; %radianes/s

T= (9*thetaF)/(7*omegaMax);
a= 6.0 * omegaMax/T;

c2 = a / 2.0;
c1 = a * T;
c0 = a * T * T /18.0;
t1 = 1000 * T / 3;
t2 = 2000 * T / 3;

%Filtro
Ts = 0.001;
%OMEGA = 40.0;
OMEGA = 40.0;
q0 = 2.0 * OMEGA/(OMEGA * Ts +2.0);
q1 = -(OMEGA * Ts - 2.0)/(OMEGA * Ts +2.0);

samples = length(theta_res);
theta = zeros(samples,1);
omega = zeros (samples,1);
alpha = zeros (samples,1);
omega_res = zeros(samples,1);
alpha_res = zeros(samples,1);
jerk_res = zeros(samples,1);
time = zeros(samples ,1);

sum_sqr_error = 0;

for k = 2 : samples
    t = (k -1) * Ts;
    time(k) = t;
    omega_res(k) = q0 * (theta_res(k)- theta_res(k - 1)) + q1 *omega_res(k - 1);
    alpha_res(k) = q0 * (omega_res(k)-omega_res(k-1))+ q1 *alpha_res(k - 1);
    jerk_res(k) = q0 * (alpha_res(k)-alpha_res(k-1))+ q1 *jerk_res(k - 1);
    if(k <= t1)
       theta(k) = (-a/(2*T))*t^3 + (a/2)*t^2;
       omega(k) = (-(3*a)/(2*T))*t^2 + a*t ;
       alpha(k) = (-3*a/T)*t + a;

    elseif(k<= t2)
        theta(k) = (a*T/6)*t -((a*T^2)/54);
        omega(k) = (1/6)*a* T;
        alpha(k) = 0;
    else
       theta(k) = (-a/(2*T))*t^3 + a*t^2 - ((a*T)/2)*t + (7/54)*a*T^2;
       omega(k) = ((-3*a)/(2*T))*t^2 + 2*a*t -(1/2)*a*T;
       alpha(k) = (-3*a/T)*t + 2*a;
    end
end
format long
sum_sqr_error = sum_sqr_error + (floor(theta(k)) - theta_res(k))^2;
```

```
format long

JerkAverage= abs(mean(jerk_res));
JerkMAxpos= abs(max(jerk_res));
JerkMAxneg= abs(min(jerk_res));

omegaMax_res= abs(max(omega_res))
alphaMax_res= abs(max(alpha_res))
if(JerkMAxpos> JerkMAxneg)
    JerkMAx = JerkMAxpos
else
    JerkMAx = JerkMAxneg
end
total_energy= energy_res(2000)
ERMS = sqrt(sum_sqr_error)

voltajeMax = abs(max(voltage_res));


figure('Name','Motion profile ata');
subplot(4,1,1);
plot(time,theta,time,theta_res);
legend(['\theta_d = ',num2str(theta(2000),6)],['\theta_f = ',num2str(theta_res(2000),6)],
'off', 'Location', 'Southeast');
xlabel('Tiempo [s]');
ylabel('Posición [rad]');
grid on
subplot(4,1,2);
plot(time , omega,time,omega_res);
legend(['\omega_d = ',num2str(omegaMax,5)],['\omega_{max} = ',num2str(omegaMax_res,5)],
'off', 'Location', 'Northeast');
xlabel('Tiempo [s]');
ylabel('Velocidad angular[rad/s]');
grid on
subplot(4,1,3);
plot(time , alpha,time,alpha_res);
legend('\alpha_{deseada}' ,'\alpha_{obtenida}','off', 'Location', 'Northeast');
xlabel('Tiempo [s]');
ylabel('Aceleración angular [rad/s^{2}]');
grid on
subplot(4,1,4);
plot(time ,jerk_res,'color',[0.8500 0.3250 0.0980]);
legend(['Jerk_{max} = ',num2str(JerkMAx,5)],'off', 'Location', 'Northeast');
xlabel('Tiempo [s]');
ylabel('Jerk [rad/s^{3}]');
grid on


figure('Name','Datos');
subplot(4,1,1);
plot(time,voltage_res);
legend(['Voltaje_{max} = ',num2str(voltajeMax,4)],'off', 'Location', 'Northeast');
xlabel('Tiempo [s]');
ylabel('Voltaje [V]');
grid on
subplot(4,1,2);
plot(time,current_res);
xlabel('Tiempo [s]');
ylabel('Corriente [A]');
grid on
subplot(4,1,3);
plot(time,power_res);
xlabel('Tiempo [s]');
ylabel('Potencia [W]');
grid on
subplot(4,1,4);
plot(time,energy_res);
legend(['Energía= ',num2str(energy_res(2000),4)],'off', 'Location', 'Southeast');
xlabel('Tiempo [s]');
ylabel('Energía [J]');
grid on
```