|  |  |
| --- | --- |
| **horizontal-naranja** | **PROGRAMA DE INGENIERÌA MECATRÓNICA**  **UNIVERSIDAD AUTONOMA DE BUCARAMANGA - UNAB** |
| **DISEÑO DE UN CONTROLADOR PID & LQR PARA UN MANIPULADOR DE UN GRADO DE LIBERTAD**  ***M.Sc. Hernando González Acevedo*** |

|  |  |
| --- | --- |
| **INTEGRANTES** | **CÓDIGO** |
| **Neil Sebastián Castro Caicedo** | **U00131867** |
| **Hubert Armando Delgado Maestre** | **U00091396** |
| **Paula Andrea Portilla Corredor** | **U00139413** |

***TRABAJO PREVIO***

1. Investigar que estrategias de control se pueden aplicar a motorreductores para minimizar los efectos de la zona muerta, no linealidades y backslash del sistema. Consultar tres referencias y describir la técnica aplicada. Completar la siguiente tabla.

|  |  |
| --- | --- |
| **Cita bibliográfica** | **Descripción de la técnica de control** |
| Control System Engineering, Norman S. Nise, Editorial: Wiley, California State Polytechnic University, Pomona, 2019 | **Compensación de feedforward:** Es una estrategia fundamental en el control de sistemas, especialmente en el contexto de motorreductores donde la precisión y la estabilidad son críticas. Al anticipar y contrarrestar las perturbaciones conocidas o predecibles antes de que influyan en la salida del sistema, esta técnica ofrece una solución proactiva que va más allá de las respuestas reactivas típicas de la retroalimentación. |
| Adaptive Backlash Compensation Method Based on Touch State Observation for a Solid Ducted Rocket | **Compensador backslash:** Técnica utilizada para minimizar o eliminar los efectos negativos de la holgura, generalmente mediante la implementación de técnicas de control específicas. Esto puede incluir la aplicación de algoritmos de control que anticipen y compensen la holgura, la utilización de sistemas de retroalimentación adecuados para detectar y corregir la desviación causada por la holgura, o incluso el diseño de mecanismos físicos en el sistema para reducir la holgura en primer lugar. |
| Nonlinear Control Hassan K. Khalil Department of Electrical and Computer Engineering Michigan State University | **Control No Lineal y Adaptativo:** Representa una estrategia avanzada y versátil utilizada en sistemas que exhiben comportamientos no lineales o que experimentan cambios en el tiempo. Al implementar esta técnica, se emplean modelos matemáticos más complejos que capturan de manera precisa la dinámica del sistema, lo que permite una respuesta más precisa y adaptable a las variaciones en las condiciones operativas. Una de las características fundamentales del Control No Lineal y Adaptativo es su capacidad para adaptarse de forma continua según evolucionan las condiciones del sistema, lo que garantiza un rendimiento óptimo ante cambios inesperados o fluctuaciones en el entorno operativo. |

***MODELO MATEMÁTICO***

1. Determinar el modelo matemático de un manipulador de un grado libertad acoplado a un motorreductor DC (figura 1). La variable de entrada al modelo es la señal de modulación de ancho de pulso (PWM) que modifica el voltaje de alimentación del motor () y las variables de salida son la posición angular de la barra y la velocidad mecánica. Linealizar el modelo, asumiendo como punto de equilibrio , y representar la dinámica del sistema en espacio de estados.

|  |  |
| --- | --- |
|  |  |

*Fig. 1 Manipulador de un grado de libertad*

|  |  |
| --- | --- |
| **Ecuaciones diferenciales (Modelo no lineal)** | Ecuaciones diferenciales observadas:  Como tenemos cuatro derivadas, tenemos cuatro estados, de los cuales pueden ser 𝜃𝐿, 𝜃𝑚, 𝑊𝑚, 𝑊𝐿 y 𝑖𝑎(t) pero, dado que las W y la theta guardan proporcional entre sí, la cual está definida por la relación de engranajes n.    Entonces tomamos como estados 𝑖𝑎(𝑡), 𝜃𝐿 𝑦 𝑊𝐿, quedando:  𝒙𝟏 = 𝒊𝒂  𝒙𝟐 = 𝜽𝑳  𝒙𝟑 = 𝑾𝑳  La entrada del sistema será el voltaje 𝑉𝑖𝑛(𝑡), procedemos con la derivación de los estados, obteniendo: |
| **Vector de estados** | **Vectores de Estados:**  **Vectores**  **Entradas:**  **Salidas:** |
| **Nombre y valor de los parámetros del modelo matemático** | Vcc : [V] Voltaje  Ro : [Ohms] Resistencia motor  Lo : [H] Inductancia motor  Kvo : [V/rad s ] Cte contraelectromotriz motor  Kio : [Nm/A] Cte par del motor  no : Relación caja de engranajes  Jmo : [Kg.m2] Inercia motor  Jlo : [Kg.m2] Inercia barra  Bmo : [Kg/s] Fricción viscosa motor  Blo : [Kg/s] Fricción viscosa barra  mo : [kg] masa de la barra  do : [m] Distancia del eje al centro de gravedad  go : [m/s^2] Gravedad |
| **Punto de equilibrio** | **Asumiendo valor de x20:** |
| **Representación en espacio de estados del modelo lineal (Expresión simbólica en función de los parámetros)** | **.** |
| **Representación en espacio de estados del modelo lineal (Expresión numérica)** |  |
| **Función de transferencia**  **(Expresión numérica, representada en polos, ceros y ganancia)** |  |
| **Función de transferencia**  **(Expresión numérica, representada en polos, ceros y ganancia)** |  |
| **Función de transferencia**  **(Expresión numérica, representada en polos, ceros y ganancia)** |  |
| **Respuesta transitoria del modelo lineal. Determinar el sobrepaso, el tiempo de levantamiento y el tiempo de establecimiento** |  |

***CONTROLADOR PID DE UN GRADO DE LIBERTAD***

1. Diseñar un sistema para controlar la posición angular de una barra, utilizando un regulador PID en continua de un grado de libertad. Implementar en Simulink el diagrama de bloques del sistema en lazo cerrado y comparar la respuesta transitoria del modelo lineal con el modelo no lineal, dada una señal de referencia variable.

|  |  |
| --- | --- |
| **Función de transferencia del controlador** | Controlador PID pseudo |
| **Respuesta transitoria de la variable del modelo lineal y del modelo no lineal en lazo cerrado** | Azul: Setpoint [ -70 , 70 ]º  Celeste: Respuesta transitoria posición NO LINEAL  Rojo: Respuesta transitoria posición LINEAL  Eje X: Tiempo [s]  Eje Y: Grados [rads] |
| **Respuesta transitoria de la acción de control del modelo lineal y del modelo no lineal en lazo cerrado** | Setpoint [ -70 , 70 ]º Azul: Respuesta transitoria acción control LINEAL  Rojo: Respuesta transitoria acción control NO LINEAL  Eje X: Tiempo [s]  Eje Y: PWM [%] |
| **Análisis de los resultados** | Se ha diseñado un controlador con un tiempo de respuesta cercano al de la planta en lazo abierto. Sin embargo, durante la simulación, se observa que el controlador alcanza un ángulo máximo de aproximadamente 70 grados. Más allá de este punto, la discrepancia entre la simulación lineal y no lineal se hace evidente, lo que sugiere que el controlador podría estar operando cerca de su límite.  Con el objetivo de evitar la zona muerta en el rango de [-25, 25]% de PWM, se ha buscado un diseño de controlador con acciones de control más altas. Esto implica garantizar que el controlador pueda proporcionar una respuesta efectiva incluso en condiciones extremas, donde la señal de control podría no producir cambios perceptibles en el sistema. |

1. Seleccionar un periodo de muestreo y discretizar el controlador, ajustar la ganancia antiwindup, en caso de que el controlador presente la componente integral. Implementar en Simulink el diagrama de bloques del sistema en lazo cerrado y comparar la respuesta transitoria del modelo lineal con el modelo no lineal, dada una señal de referencia variable.

|  |  |
| --- | --- |
| **Periodo de muestreo** |  |
| **Función de transferencia del controlador** | Controlador PID pseudo discretizado |
| **Código del controlador en el bloque de Matlab function** | function [us, Con\_out] = fcn(Sp, Pv, Con\_ini)    en\_1 = Con\_ini(1);  I\_error\_1 = Con\_ini(2);  difact = Con\_ini(3);  Ad\_1 = Con\_ini(4);    umin = -100;  umax = 100;    Kp = 12.194801852041582;  Ki = 1.836264365827931;  Kd = 58.215749090323079;  N = 0.135321750384943;  kaw=0.1;      en = Sp - Pv;  Ap = Kp\*en;  Ad = Kd\*(en-en\_1) + N\*Ad\_1;  I\_error = en + I\_error\_1 + kaw\*difact;  Ai = Ki\*I\_error;  un = Ap + Ai + Ad;  us = un;    if un < umin  us = umin;  end  if un > umax  us = umax;  end    en\_1 = en;  I\_error\_1 = I\_error;  difact = us - un;  Ad\_1 = Ad;    Con\_out = [en\_1; I\_error\_1; difact; Ad\_1]; |
| **Respuesta transitoria de la variable del modelo lineal y del modelo no lineal en lazo cerrado** | Azul: Setpoint [ -70 , 70 ]º  Naranja: Respuesta transitoria posición LINEAL discreta  Verde: Respuesta transitoria posición NO LINEAL discreta  Eje X: Tiempo [s]  Eje Y: Grados [rads] |
| **Respuesta transitoria de la acción de control del modelo lineal y del modelo no lineal en lazo cerrado** | Setpoint [ -70, 70 ]º Rojo: Respuesta transitoria acción control LINEAL discreta  Azul: Respuesta transitoria acción control NO LINEAL discreta  Eje X: Tiempo [s]  Eje Y: PWM [%] |
| **Análisis de los resultados** | Después de discretizar el controlador, se observa que el ángulo crítico sigue siendo aproximadamente 70 grados. Sin embargo, debido al período de muestreo, se produce un ligero sobrepaso al acercarse al ángulo crítico. A pesar de esto, se sostiene que el proceso de discretización del controlador se ha realizado de manera satisfactoria. |

1. Programar en el sistema embebido la estrategia de control. Evaluar la respuesta transitoria de la variable del proceso y la señal de control para diferentes valores de la señal de referencia. Comparar los datos experimentales con los datos simulados, dada la misma señal de referencia.

|  |  |
| --- | --- |
| **Señal de referencia y respuesta transitoria de la variable (Datos experimentales vs Datos simulados)** | Marrón: Setpoint [ 90 , -60, 35, -70, 45, -35, 50 ]º  Rojo: Respuesta transitoria posición SIMULADO lineal discreta  Verde: Respuesta transitoria posición SIMULADO no lineal discreta  Azul: Respuesta transitoria posición EXPERIMENTAL  Eje X: Tiempo [s]  Eje Y: Grados [rads] |
| **Respuesta transitoria de la variable (Datos experimentales vs Datos simulados)** | Setpoint [ 90 , -60, 35, -70, 45, -35, 50 ]º  Rojo: Respuesta transitoria acción control SIMULADO lineal discreta  Azul: Respuesta transitoria acción control SIMULADO no lineal discreta  Violeta: Respuesta transitoria acción control EXPERIMENTAL  Eje X: Tiempo [s]  Eje Y: PWM [%] |
| **Análisis de los resultados** | Al implementar el controlador en Arduino, se aplicaron dos técnicas de compensación, detalladas en el último punto del laboratorio: la anti-zona muerta y una técnica empírica para prevenir velocidades bajas al aproximarse al Setpoint. Sin estas dos compensaciones, resultaba imposible estabilizar la planta en un tiempo razonable.  Teóricamente, el controlador para el sistema no lineal no puede controlar puntos de consigna mayores a aproximadamente 70 grados absolutos. Sin embargo, gracias a los compensadores, se logra estabilizar la planta y, además, superar este límite teórico. La gráfica verde, que representa el modelo teórico no lineal, oscila en torno al punto de consigna de 90 grados, pero no se vuelve completamente inestable.  El compensador de zona muerta juega un papel crucial al evitar que el valor del ancho de pulso modulado (PWM) permanezca demasiado tiempo en esta zona, lo que podría detener el motor. Por este motivo, se observan escalones en la acción de control al entrar en la zona muerta y al activarse el compensador.    Estos escalones a bajos SP cercanos a la zona muerta < abs(30) pueden generar oscilaciones propias del compensador actuando para evitando la zona muerta, en valores altos de SP es poco probable observar este efecto.    Pero esto trae un inconveniente, se genera una leve zona muerta al acercarse al SP (backslash), este problema se corregirá para el último punto del laboratorio.    En conclusión, el controlador logra estabilizarse en todos los SP que fueron programados y su acción de control se mantiene estable evitando oscilaciones al rededor del valor de SP. |

***CONTROLADOR PID DE DOS GRADOS DE LIBERTAD***

1. Diseñar un sistema para controlar la posición angular de una barra, utilizando un regulador PID en continua de dos grados de libertad. Implementar en Simulink el diagrama de bloques del sistema en lazo cerrado y comparar la respuesta transitoria del modelo lineal con el modelo no lineal, dada una señal de referencia variable.

|  |  |
| --- | --- |
| **Diagrama de bloques de la estrategia de control** |  |
| **Función de transferencia del controlador** | Controlador PID pseudo |
| **Función de transferencia del controlador** | Controlador PD pseudo |
| **Respuesta transitoria de la variable del modelo lineal y del modelo no lineal en lazo cerrado** | Azul: Setpoint [ -65 , 65 ]º  Rojo: Respuesta transitoria posición LINEAL  Celeste: Respuesta transitoria posición NO LINEAL  Eje X: Tiempo [s]  Eje Y: Grados [rads] |
| **Respuesta transitoria de la acción de control del modelo lineal y del modelo no lineal en lazo cerrado** | Setpoint [ -65 , 65 ]º Azul: Respuesta transitoria acción control LINEAL  Rojo: Respuesta transitoria acción control NO LINEAL  Eje X: Tiempo [s]  Eje Y: PWM [%] |
| **Análisis de los resultados** | De igual manera que 1 grado de libertad (1DOF), se diseñó el controlador con un tiempo de respuesta similar al de la planta en lazo abierto. Sin embargo, inicialmente resultó difícil alcanzar acciones de control altas. Por lo tanto, se optó por establecer un ancho de pulso modulado (PWM) inicial que rompiera la inercia inicial del sistema.  Al realizar simulaciones, se observó un resultado contrario a la expectativa inicial: teóricamente, el sistema de 2 grados de libertad (2DOF) mostraba una respuesta más desfavorable. Generaba mayores oscilaciones en el modelo no lineal para valores de punto de consigna (SP) mayores a aproximadamente 65 grados absolutos, aunque ligeramente menos pronunciadas que en el caso de 1DOF. A pesar de intentar varios diseños de controladores y diferentes combinaciones, no se observaron mejoras significativas en el rendimiento del sistema. |

1. Seleccionar un periodo de muestreo y discretizar el controlador, ajustar la ganancia antiwindup, en caso de que el controlador presente la componente integral. Implementar en Simulink el diagrama de bloques del sistema en lazo cerrado y comparar la respuesta transitoria del modelo lineal con el modelo no lineal, dada una señal de referencia variable.

|  |  |
| --- | --- |
| **Periodo de muestreo** |  |
| **Función de transferencia del controlador** |  |
| **Función de transferencia del controlador** |  |
| **Código del controlador en el bloque de Matlab function** | function [us, Con\_out] = fcn(Sp, Pv, Con\_ini)    en\_1 = Con\_ini(1);  yn\_1 = Con\_ini(2);  I\_error\_1 = Con\_ini(3);  difact = Con\_ini(4);  Ad\_1 = Con\_ini(5);  Ad1\_ = Con\_ini(6);    umin = -100;  umax = 100;    Kp = -3.357670445385118;  Ki = 3.822951239710286;  Kd = 32.464560395668343;  N = 0.782109908780621;  Kpd = 0.285776886757601;  Kdp = 3.408050560944484;  N1 = 0.997168715666942;  Ka= 0.1;    en = Sp - Pv;  Ap = Kp\*en;  Ad = Kd\*(en-en\_1) + N\*Ad1\_;  I\_error = en + I\_error\_1 + Ka\*difact;  Ai = Ki\*I\_error;  U1 = Ap + Ai + Ad;    Ap1 = Kpd \* Pv;  Ad1 = Kdp \* (Pv - yn\_1) + N1\*Ad\_1;  U2 = Ap1 + Ad1;    Uc = U1 - U2;  us = Uc;      if Uc < umin  us = umin;  end  if Uc > umax  us = umax;  end    en\_1 = en;  yn\_1 = Pv;  I\_error\_1 = I\_error;  difact = us - Uc;  Ad\_1 = Ad1;  Ad1\_ = Ad;    Con\_out = [en\_1; yn\_1; I\_error\_1; difact; Ad\_1; Ad1\_]; |
| **Respuesta transitoria de la variable del modelo lineal y del modelo no lineal en lazo cerrado** | Rojo: Setpoint [ -65 , 65 ]º  Naranja: Respuesta transitoria posición LINEAL discreta  Verde: Respuesta transitoria posición NO LINEAL discreta  Eje X: Tiempo [s]  Eje Y: Grados [rads] |
| **Respuesta transitoria de la acción de control del modelo lineal y del modelo no lineal en lazo cerrado** | Setpoint [ -65, 65 ]º Naranja: Respuesta transitoria acción control LINEAL discreta  Azul: Respuesta transitoria acción control NO LINEAL discreta  Eje X: Tiempo [s]  Eje Y: PWM [%] |
| **Análisis de los resultados** | Al discretizar, se observa que las oscilaciones en continua se intensifican ligeramente en el modelo no lineal, pero aún se mantienen dentro de los límites aceptables. Por lo tanto, se decide proceder con la implementación del programa en Arduino para observar la respuesta transitoria experimental y compararla con los datos teóricos. Este enfoque experimental proporcionará una valiosa retroalimentación sobre el rendimiento del sistema en condiciones reales y permitirá una evaluación más completa de la efectividad del controlador diseñado. |

1. Programar en el sistema embebido la estrategia de control. Evaluar la respuesta transitoria de la variable del proceso y la señal de control para diferentes valores de la señal de referencia. Comparar los datos experimentales con los datos simulados, dada la misma señal de referencia.

|  |  |
| --- | --- |
| **Señal de referencia y respuesta transitoria de la variable (Datos experimentales vs Datos simulados)** | Marrón: Setpoint [ 90 , -60, 35, -70, 45, -35, 50 ]º  Naranja: Respuesta transitoria posición SIMULADO lineal discreta  Verde: Respuesta transitoria posición SIMULADO no lineal discreta  Azul: Respuesta transitoria posición EXPERIMENTAL  Eje X: Tiempo [s]  Eje Y: Grados [rads] |
| **Respuesta transitoria de la variable (Datos experimentales vs Datos simulados)** | Setpoint [ 90 , -60, 35, -70, 45, -35, 50 ]º  Rojo: Respuesta transitoria acción control SIMULADO lineal discreta  Azul: Respuesta transitoria acción control SIMULADO no lineal discreta  Violeta: Respuesta transitoria acción control EXPERIMENTAL  Eje X: Tiempo [s]  Eje Y: PWM [%] |
| **Análisis de los resultados** | Al programar el controlador en Arduino, se implementaron dos técnicas de compensación detalladas en el último punto del laboratorio: la anti-zona muerta y una técnica empírica para prevenir velocidades bajas al acercarse al setpoint. Sin la aplicación de estos dos compensadores, resultaba imposible lograr la estabilización de la planta en un tiempo razonable.  De manera experimental, se observó una respuesta aún mejor de lo predicho teóricamente. De hecho, el modelo no lineal con los puntos de consigna utilizados en el sistema embebido llevó a una completa inestabilidad de la planta, algo que no se había experimentado en el caso de 1 grado de libertad (1DOF). Sin embargo, entre 1DOF y 2DOF, se obtuvo una respuesta transitoria mejorada en este último caso, con una ligera reducción en los picos y una señal más suave.  Al igual que en 1DOF, surgieron nuevos problemas relacionados con zonas muertas (backlash) al aproximarse al punto de consigna. No obstante, los picos generados por el compensador de zona muerta se redujeron considerablemente, y el sistema respondió de manera más efectiva ante perturbaciones, como se pudo observar en el video.  En conclusión, el controlador logró estabilizarse en todos los setpoints programados, manteniendo una acción de control estable y evitando oscilaciones alrededor del valor de SP. Este éxito experimental subraya la efectividad del diseño del controlador y la importancia de las técnicas de compensación empleadas para mejorar el rendimiento del sistema. |

***CONTROLADOR EN CASCADA***

1. Diseñar un sistema para controlador en cascada, el lazo interno corresponde al control de velocidad y el lazo externo corresponde al control de la posición angular de la barra. Implementar en Simulink el diagrama de bloques del sistema en lazo cerrado y comparar la respuesta transitoria del modelo lineal con el modelo no lineal, dada una señal de referencia variable.

|  |  |
| --- | --- |
| **Función de transferencia del controlador del lazo interno** | Controlador PI |
| **Función de transferencia del controlador del lazo externo** | Controlador P |
| **Respuesta transitoria de la variable del modelo lineal y del modelo no lineal en lazo cerrado (señal de velocidad)** |  |
| **Respuesta transitoria de la variable del modelo lineal y del modelo no lineal en lazo cerrado (señal posición de la barra)** |  |
| **Respuesta transitoria de la acción de control del modelo lineal y del modelo no lineal en lazo cerrado** |  |
| **Análisis de los resultados** | El lazo interno (velocidad) se diseñó más rápido que el externo y se pusieron topes de saturación de [-40,40] rads, el externo se trató de diseñar con tiempo cercano al de la planta el lazo abierto, pero no se logró, el menor tiempo posible fue 10 segundos aprox. para el lazo externo. Se observa en simulink el comportamiento estable para el modelo líneal y no lineal, pero no se lograron valores de Sp altos, la máxima variación posible antes de inestabilizar el modelo no lineal fue de [-30,30]º.    Por otra parte, se intentó pasar de bloques de PID en simulink a funciones LTI mediante las ecuaciones de PID C = Kp + Ki/s, pero no fue posible, la planta se inestabilizaba al usar los bloques LTI de las funciones de transferencia, por esta razón no fue posible discretizar los controladores y observar la respuesta transitoria en discreta.    Al no tener la planta discretizada no se logró programar el sistema embebido y se procede a avanzar al siguiente punto del laboratorio. |

1. Seleccionar un periodo de muestreo y discretizar el controlador, ajustar la ganancia antiwindup, en caso de que el controlador presente la componente integral. Implementar en Simulink el diagrama de bloques del sistema en lazo cerrado y comparar la respuesta transitoria del modelo lineal con el modelo no lineal, dada una señal de referencia variable.

|  |  |
| --- | --- |
| **Periodo de muestreo** |  |
| **Función de transferencia del controlador del lazo interno** |  |
| **Función de transferencia del controlador del lazo externo** |  |
| **Código del controlador en el bloque de Matlab function** |  |
| **Respuesta transitoria de la variable del modelo lineal y del modelo no lineal en lazo cerrado (señal de velocidad)** |  |
| **Respuesta transitoria de la variable del modelo lineal y del modelo no lineal en lazo cerrado (señal posición de la barra)** |  |
| **Respuesta transitoria de la acción de control del modelo lineal y del modelo no lineal en lazo cerrado** |  |
| **Análisis de los resultados** |  |

1. Programar en el sistema embebido la estrategia de control. Evaluar la respuesta transitoria de la variable del proceso y la señal de control para diferentes valores de la señal de referencia. Comparar los datos experimentales con los datos simulados, dada la misma señal de referencia.

|  |  |
| --- | --- |
| **Señal de referencia y respuesta transitoria de la variable (Datos experimentales vs Datos simulados – velocidad angular)** |  |
| **Señal de referencia y respuesta transitoria de la variable (Datos experimentales vs Datos simulados – posición de la barra)** |  |
| **Respuesta transitoria de la variable (Datos experimentales vs Datos simulados)** |  |
| **Análisis de los resultados** |  |

***CONTROLADOR LQR***

1. Eliminar el estado asociado a la corriente de armadura del modelo matemático y obtener la nueva representación en espacio de estados, definiendo como estados la posición y velocidad mecánica de la barra.

|  |  |
| --- | --- |
| **Ecuaciones diferenciales (Modelo no lineal)** | Partimos de las ecuaciones diferenciales observadas:    Como tenemos cuatro derivadas, tenemos cuatro estados, de los cuales pueden ser 𝜃𝐿,𝜃𝑚,𝑊𝑚 ,𝑊𝐿 y 𝑖𝑎(𝑡) pero, se definirá la derivada de 𝑖𝑎(𝑡) como cero, se elimina un estado, además dado que las W y los theta guardan proporcional entre sí, la cual está definida por la relación de engranajes n.    Entonces tomamos como estados 𝜃𝐿 𝑦 𝑊𝐿, quedando:    La entrada del sistema será el voltaje 𝑉𝑖𝑛(𝑡).  Procedemos con la derivación de los estados, obteniendo:    Despejamos el resultado de la función tres para obtener una expresión en términos de estados y entradas.  Se obtiene que 𝑇2=𝑇1𝑛, entonces    Entonces 𝑇2    Despejamos 𝑇2 𝑒𝑛 𝑥3̇    Obteniendo de este modo: |
| **Vector de estados** |  |
| **Nombre y valor de los parámetros del modelo matemático** | Vcc = 7 V;  Ro = 5.15610211779038; %ESTIMADA  Lo = 0.667669785204658; %ESTIMADA  Kio = 0.00119530882984502; %ESTIMADA  Kvo = 0.0035694528924319; %ESTIMADA  Jmo = 2.08549279330578e-07;%ESTIMADA  Bmo = 2.02156082362446e-07 %ESTIMADA;  no = 120; % Relación de engranajes  Jlo = 0.00100697016255642; %ESTIMADA  Blo = 8.18575752786119e-06 %ESTIMADA  mo = 0.17244; % Masa de la barra [kg]  do = 0.06862-0.021; % Distancia del eje al centro de gravedad [m]  go = 9.81; % gravedad [m/s^2] |
| **Puntos de equilibrio** | % PUNTOS DE EQUILIBRIO  X10 = 0;  X20 = 0;  U10 = 0;  Vin0 = Ro\*X10 |
| **Representación en espacio de estados del modelo lineal (Expresión simbólica en función de los parámetros)** |  |
| **Representación en espacio de estados del modelo lineal (Expresión numérica)** |  |
| **Comparación de la respuesta transitoria del modelo lineal de 3 estados con el de 2 estados. Determinar el sobrepaso, el tiempo de levantamiento y el tiempo de establecimiento de cada modelo** |  |

1. Diseñar un sistema para controlar la posición angular de una barra, utilizando un regulador LQR en continua. Implementar en Simulink el diagrama de bloques del sistema en lazo cerrado y comparar la respuesta transitoria del modelo lineal con el modelo no lineal, dada una señal de referencia variable.

|  |  |
| --- | --- |
| **Matriz de penalidad** |  |
| **Matriz de penalidad** |  |
| **Matriz de ganancia que multiplica los estados** |  |
| **Ganancia integral** |  |
| **Respuesta transitoria de la variable del modelo lineal y del modelo no lineal en lazo cerrado** | Azul: Setpoint [ -120 , 120 ]º  Rojo: Respuesta transitoria posición LINEAL  Celeste: Respuesta transitoria posición NO LINEAL  Eje X: Tiempo [s]  Eje Y: Grados [rads] |
| **Respuesta transitoria de la acción de control del modelo lineal y del modelo no lineal en lazo cerrado** | Setpoint [ -120 , 120 ]º Azul: Respuesta transitoria acción control LINEAL  Rojo: Respuesta transitoria acción control NO LINEAL  Eje X: Tiempo [s]  Eje Y: PWM [%] |
| **Análisis de los resultados** | El valor máximo que el modelo no lineal puede soportar con el controlador LQR es de aproximadamente 120 grados, un valor superior al alcanzado con los controladores PID previamente analizados. Aunque no fue posible diseñar un controlador LQR con acciones de control que comenzaran en valores altos, se logró obtener un comportamiento de control más suave en comparación con los PID. |

1. Seleccionar un periodo de muestreo y discretizar el controlador, ajustar la ganancia antiwindup, en caso de que el controlador presente la componente integral. Implementar en Simulink el diagrama de bloques del sistema en lazo cerrado y comparar la respuesta transitoria del modelo lineal con el modelo no lineal, dada una señal de referencia variable.

|  |  |
| --- | --- |
| **Periodo de muestreo** |  |
| **Matriz de ganancia que multiplica los estados** |  |
| **Ganancia integral** |  |
| **Código del controlador en el bloque de Matlab function** | function [us,Con\_out] = fcn(Sp,X\_est,Con\_ini)    umax = 100;  umin = -100;  kaw = zeros(1);  CL = [1, 0]  Ki = [13.038404810405284];  Kes = [8.39272964861829,6.19627625077861];    Vk\_1 = Con\_ini(1);  dif\_act = Con\_ini(2);    Tm = 0.03;  Kid = Ki\*Tm;    Pv = CL\*X\_est;  en = Sp - Pv;  Vk = en + Vk\_1 + kaw\*dif\_act;  un = Kid\*Vk - Kes\*X\_est;  us = un;    Vk\_1 = Vk;    if un>umax  us = umax;  end  if un<umin  us = umin;  end    dif\_act = us - un;  Con\_out = [Vk\_1; dif\_act]; |
| **Respuesta transitoria de la variable del modelo lineal y del modelo no lineal en lazo cerrado** | Marrón: Setpoint [ -120 , 120 ]º  Naranja: Respuesta transitoria posición LINEAL discreta  Verde: Respuesta transitoria posición NO LINEAL discreta  Eje X: Tiempo [s]  Eje Y: Grados [rads] |
| **Respuesta transitoria de la acción de control del modelo lineal y del modelo no lineal en lazo cerrado** | Setpoint [ -120, 120 ]º Rojo: Respuesta transitoria acción control LINEAL discreta  Azul: Respuesta transitoria acción control NO LINEAL discreta  Eje X: Tiempo [s]  Eje Y: PWM [%] |
| **Análisis de los resultados** | Con la información del análisis del punto anterior, se procede a programar el controlador LQR en el sistema embebido. Este paso es crucial para validar experimentalmente el rendimiento del controlador y observar cómo se comporta el sistema en condiciones prácticas. A pesar de las limitaciones en las acciones de control iniciales, se espera que el enfoque LQR proporcione una mejor estabilidad y respuesta del sistema en comparación con los controladores PID previos. |

1. Programar en el sistema embebido la estrategia de control. Evaluar la respuesta transitoria de la variable del proceso y la señal de control para diferentes valores de la señal de referencia. Comparar los datos experimentales con los datos simulados, dada la misma señal de referencia.

|  |  |
| --- | --- |
| **Señal de referencia y respuesta transitoria de la variable (Datos experimentales vs Datos simulados)** | Marrón: Setpoint [ 90 , -60, 35, -70, 45, -35, 50 ]º  Rojo: Respuesta transitoria posición SIMULADO lineal discreta  Verde: Respuesta transitoria posición SIMULADO no lineal discreta  Azul: Respuesta transitoria posición EXPERIMENTAL  Eje X: Tiempo [s]  Eje Y: Grados [rads] |
| **Respuesta transitoria de la variable (Datos experimentales vs Datos simulados)** | Setpoint [ 90 , -60, 35, -70, 45, -35, 50 ]º  Rojo: Respuesta transitoria acción control SIMULADO lineal discreta  Azul: Respuesta transitoria acción control SIMULADO no lineal discreta  Violeta: Respuesta transitoria acción control EXPERIMENTAL  Eje X: Tiempo [s]  Eje Y: PWM [%] |
| **Análisis de los resultados** | Al programar el controlador en Arduino se aplicaron dos técnicas de compensación (estas se detallan en el último punto del laboratorio): Anti-zona muerta y una técnica empírica para evitar velocidades bajas al acercarse al SP. Sin estos dos compensadores resultaba imposible estabilizar la planta en un tiempo prudente.  En este método, a diferencia de los PID mostrados anteriormente, no se generan mayores oscilaciones, pero se intensifica el efecto del backslash. Como se observa en estos 3 SP donde es mayormente visible:    A pesar que teóricamente se puede llegar hasta SP abs(120) en el modelo no lineal , experimentalmente no fue así dado al efecto de backslash el cual genera un acción de control fuerte al romper la inercial de la holgura del tren de engranajes y en Sp altos esto inestabiliza el sistema.  Otra ventaja es que reacciona mejor y de manera rápida ante perturbaciones, esto se observa en el video. En conclusión, el controlador logra estabilizarse en todos los SP que fueron programados y su acción de control se mantiene estable evitando oscilaciones al rededor del valor de SP. |

***COMPENSACIÓN DE NO LINEALIDADES DEL SISTEMA***

1. Del trabajo previo seleccionar una técnica que minimice los efectos de la zona muerta y backslash, programarla en el sistema embebido para la estrategia de control que presente mejor respuesta transitoria. Evaluar la respuesta transitoria de la variable del proceso y la señal de control para diferentes valores de la señal de referencia.

|  |  |
| --- | --- |
| **Justificación de la estrategia de control seleccionada (PID 1 DOF, PID 2 DOF, control en cascada y control LQR)** | Durante el desarrollo del laboratorio, se aplicaron estrategias de control (1DOF, 2DOF y LQR) junto con los compensadores de zona muerta y un método empírico específico (detallado más adelante), ya que el sistema no podía estabilizarse sin su ayuda. Se observaba que el sistema continuaba buscando el punto de consigna, oscilando ligeramente alrededor de él. Después de un análisis exhaustivo, se decidió aplicar estas técnicas de compensación en conjunto para el controlador LQR.  Se estableció como objetivo alcanzar un valor de punto de consigna de 120 grados, ya que teóricamente el controlador LQR debería ser capaz de estabilizar el sistema hasta este límite sin inestabilizarlo. Esto contrasta con los valores máximos obtenidos anteriormente (90 grados) utilizando las técnicas de control y compensación mencionadas previamente. Este último objetivo representa un desafío importante y demuestra la capacidad del enfoque LQR para controlar sistemas con rangos más amplios de operación. |
| **Diseño de la estrategia de control para minimizar los efectos de la zona muerta, no linealidades y backslash del sistema** | Estrategia anti-backslash:  El motor antes de frenarse por completo debido a la fricción viscosa genera un leve movimiento debido a la holgura del tren de engranajes, esto nosotros lo tomaremos como el “intento de movimiento” para esto nos basamos en la lectura actual de la variable “en” y “en\_1”, de esta manera podemos saber el momento en el cual el motor se detuvo y aplicar un compensador (ya sea positivo o negativo) para romper la inercia lo más rápido posible sin inestabilizar el sistema.      Estrategia anti-zona muerta:  Del laboratorio anterior conocemos que la zona muerta de la planta se encuentra entre [–25,25]% del PWM, conforme a esto se decide eliminar esos valores de PWM, para esto es necesario saber la dirección de giro y el valor del error para saber cuándo aplicar un PWM positivo o negativo. Se redujo la zona muerta a [-18,18]% dado que ofrecía mejores resultados.      Estrategia empírica contra velocidad baja que permita entrar a la zona de backslash:  Sabemos que la estrategia de control empieza a disminuir su valor de PWM por ende la velocidad al ir reduciendo su valor del error a medida que se acerca al SP, esto conlleva a que se pueda ingresar a la zona de backslash, para evitar eso se tiene dos opciones, aumentar el valor de la ganancia integral o acelerar la velocidad cuando el detecta que está disminuyendo demasiado. Se opto por la segunda dado que la primera es muy sensible y al cambiar levemente la ganancia el sistema se inestabiliza en la mayoría de las ocasiones. De esta manera la compensación anti backslash se aplica ANTES/DESPUES de entrar a la zona y esta busca EVITAR entrar a la zona de backslash. |
| **Señal de referencia y respuesta transitoria de la variable** | Marrón: Setpoint [ 150 , -60, 35, -70, 45, -130, 50 ]º  Naranja: Respuesta transitoria posición SIMULADO lineal discreta  Verde: Respuesta transitoria posición SIMULADO no lineal discreta  Azul: Respuesta transitoria posición EXPERIMENTAL  Eje X: Tiempo [s]  Eje Y: Grados [rads] |
| **Respuesta transitoria de la variable** | Setpoint [ 150 , -60, 35, -70, 45, -130, 50 ]º  Rojo: Respuesta transitoria acción control SIMULADO lineal discreta  Azul: Respuesta transitoria acción control SIMULADO no lineal discreta  Violeta: Respuesta transitoria acción control EXPERIMENTAL  Eje X: Tiempo [s]  Eje Y: PWM [%] |
| **Análisis de los resultados** | Se aplicaron las tres técnicas de compensación mencionadas anteriormente en el controlador LQR dado que este solo presentaba problemas de backslash y teóricamente tenía mejor rango de operación.  Se observa que el backslash se redujo casi en su totalidad, el tiempo de establecimiento teórico y simulado es cercano y sumado a esto se garantiza un rango de operación de SP [-150,150]º, mayor al que se planteó en un inicio que era 120 basándonos en el modelo no lineal teórico.  El siguiente sobrepaso al existir un cambio de SP se debe a la programación del backslash la cual requiere un ajuste fino para evitar esos leves retrasos en la respuesta.    En conclusión, se logró eliminar la mayoría de problemas existentes por la no linealidad de la planta, el modelo no lineal teórico y los datos experimentales concuerdan en su mayoría, incluyendo el tiempo de establecimiento, se logra mantener estable el SP sin existir oscilaciones y por último se logró estabilidad en abs(150) grados de un máximo de abs(180). |

***CONCLUSIONES***

1. Indicar las conclusiones más importantes del desarrollo de la práctica de laboratorio (máximo 200 palabras).

|  |
| --- |
| **Conclusiones** |
| * Los controladores sin compensadores no logran estabilizarse en un punto de consigna (SP), ya que el error integral (Ki) aumenta gradualmente la acción de control, generando movimientos no deseados en un intento por llevar la variable medida (output) a cero. Esta dificultad radica en que los parámetros del motor y la barra, utilizados para calcular los parámetros del controlador, no concuerdan plenamente con la realidad. La falta de una toma de datos eficaz para garantizar que los parámetros sean precisos contribuye a que pequeñas variaciones en las estimaciones puedan afectar significativamente la respuesta del sistema. Esto se evidencia claramente en la toma de datos experimental, donde el controlador lucha por estabilizarse en un SP sin la ayuda de compensadores. * Como se anticipaba, el controlador LQR exhibe la mejor respuesta transitoria y una mayor robustez ante perturbaciones, como se confirmó experimentalmente. Sin embargo, esto se logró gracias a la implementación de tres compensadores adicionales que apoyaron al controlador para obtener resultados óptimos. A diferencia de los controladores PID que presentan problemas de oscilaciones y backlash con solo dos compensadores, el controlador LQR solo presenta backlash. Una vez se aplica el tercer compensador, la señal de control se vuelve más estable y continua, permitiendo alcanzar valores de SP cercanos a 180 grados. * Se intentó programar un compensador en Arduino que ajustara ligeramente las ganancias en momentos específicos del transitorio, como en la zona muerta o para grandes y pequeñas variaciones. Sin embargo, la implementación de este compensador resultó en una acción de control con un alto nivel de rizado, tanto visualmente como en la monitorización del sistema. Por esta razón, se descartó su uso. En su lugar, se optó por una técnica empírica que varía el valor de la señal de control a diferentes velocidades para evitar la zona de backlash. Esta última técnica ofreció mejores resultados al reducir el rizado y proporcionar un movimiento más continuo y suave del péndulo, lo cual se observó tanto visualmente como en el monitor del sistema. |

|  |
| --- |
| **Bibliografía** |
| [1] *Nise, N. S. (s. f.-b). Control Systems Engineering, 7th Edition (7.a ed.). Wiley Global Education. https://gnindia.dronacharya.info/EEE/5thSem/Downloads/ControlSystem/Books/CONTROL-SYSTEM-REFERENCE-BOOK-2.pdf*  [2] *Khalil, H. K. (s. f.). Nonlinear Control. Michigan Publishing Services. https://www.eolss.net/sample-chapters/C18/E6-43-21.pdf*  [3] *Wang, A., Zeng, Q., Ma, L., & Wang, H. (2020). Adaptive Backlash Compensation Method Based on Touch State Observation for a Solid Ducted Rocket. International Journal Of Aerospace Engineering, 2020, 1-12. https://doi.org/10.1155/2020/6698158* |

***ANEXOS***

|  |  |
| --- | --- |
| **Código común para todos los controladores** | #include "MeanFilterLib.h"  MeanFilter<float> pwmFil(3);  MeanFilter<float> pFil(3);  MeanFilter<float> rpmFil(3);    const byte Encoder\_C1 = 2; // Cable amarillo pin 3 digital  const byte Encoder\_C2 = 3; // Cable verde al pin 4 digital  byte Encoder\_C1Last;  int paso = 0;  boolean direccion, flagPWM, autoincrement;    int entrada3 = 13;  int entrada4 = 12;  int enableB = 11;    int maxPWM = 130; // VCC 7V    int userE, Sp;    const int tMuestro = 30;    volatile int n = 0;  volatile byte ant = 0;  volatile byte act = 0;    double P = 0;  double R = 3840;    float arduinoPWM;    float rpm = 0;  unsigned long timeold = 0;  unsigned long startTime = 0;  unsigned long elapsedTime = 0;      // 1DOF  double en\_2, en\_1, un\_2, un\_1, un, difact, us, I\_error\_1, errPer, I\_error, Ad\_1, en\_3;  // 2DOF  double Ad1\_, yn\_1;  // Estados  double Vk\_1, dif\_act;    void reiniciarVAR() {  autoincrement = false;  startTime = 0; // Reiniciar el tiempo de inicio si autoincrement es false  elapsedTime = 0;  timeold = 0;  rpm = 0;  P = 0;  Sp = 0;  flagPWM = false;  analogWrite(enableB, 0); // Equivalente a 50% del máximo (128 de 255)  }      void calculapulso() {    int Lstate = digitalRead(Encoder\_C1);  if ((Encoder\_C1Last == LOW) && Lstate == HIGH)  {  int val = digitalRead(Encoder\_C2);  if (val == LOW && direccion)  {  direccion = false; //Reverse  }  else if (val == HIGH && !direccion)  {  direccion = true; //Forward  }  }  Encoder\_C1Last = Lstate;    if (!direccion) paso++;  else paso--;      ant = act;    if (digitalRead(Encoder\_C1)) bitSet(act, 1); else bitClear(act, 1);  if (digitalRead(Encoder\_C2)) bitSet(act, 0); else bitClear(act, 0);        if (ant == 2 && act == 0) n++;  if (ant == 0 && act == 1) n++;  if (ant == 3 && act == 2) n++;  if (ant == 1 && act == 3) n++;    if (ant == 1 && act == 0) n--;  if (ant == 3 && act == 1) n--;  if (ant == 0 && act == 2) n--;  if (ant == 2 && act == 3) n--;    }      void direMotor() {  // Cambia la dirección del motor y establece la velocidad  if (flagPWM) {  digitalWrite(entrada3, LOW);  digitalWrite(entrada4, HIGH);    } else {    // IZQUIERDA POSITIVO  digitalWrite(entrada3, HIGH);  digitalWrite(entrada4, LOW);  }    analogWrite(enableB, map(abs(us), 0, 100, 0, maxPWM));  }    void setup()  {  Serial.begin(9600);  pinMode(entrada3, OUTPUT);  pinMode(entrada4, OUTPUT);  pinMode(enableB, OUTPUT);  pinMode(Encoder\_C2, INPUT);  flagPWM = false;  autoincrement = false;    attachInterrupt(digitalPinToInterrupt(Encoder\_C1), calculapulso, CHANGE);  attachInterrupt(digitalPinToInterrupt(Encoder\_C2), calculapulso, CHANGE);  Serial.println("LISTO");    }    void loop()  {      if (Serial.available()) { // Verificar si hay datos disponibles en el puerto serial  userE = Serial.parseInt(); // Leer el valor entero enviado por el puerto serial  // Verificar si el valor recibido está dentro del rango válido (0-100)  if (userE != 0 && userE != 404 ) {  flagPWM = (userE < 0) ? true : false;  autoincrement = true;  //Sp = userE; // Para enviar por teclado    }    if (userE == 404) {  reiniciarVAR();    }    }    if (autoincrement == true) {  if (startTime == 0) {  startTime = millis(); // Registrar el tiempo de inicio  }    elapsedTime = millis() - startTime; // Calcular el tiempo transcurrido      if (elapsedTime > 3000) {      if (millis() - timeold >= tMuestro) {  escalonPWM();  rpm = (paso \* (0.100 / (millis() - timeold)) \* (60000 / 360)) \* -1 ;  rpm = rpmFil.AddValue(rpm);  P = (n \* 360.0) / R;  P = pFil.AddValue(P);  controlEstados();  direMotor();  impresionSerial();  timeold = millis();  paso = 0;  }    if (elapsedTime >= 73000 ) {  reiniciarVAR();  }    }        }  }    void escalonPWM() {  if (elapsedTime > 3001) Sp = 150; // PWMX60x80    if (elapsedTime > 13000) Sp = -60; // PWMX60x80  if (elapsedTime > 23000) Sp = 35; // PWMX60x80  if (elapsedTime > 33000) Sp = -70; // PWMX60x80  if (elapsedTime > 43000) Sp = 45; // PWMX60x80  if (elapsedTime > 53000) Sp = -130; // PWMX60x80  if (elapsedTime > 63000) Sp = 50; // PWMX60x80    }          void impresionSerial() {  arduinoPWM = pwmFil.AddValue(arduinoPWM);    Serial.print(Sp ); Serial.print(", "); Serial.print(P ); Serial.print(", "); Serial.print(arduinoPWM); Serial.print("\n");    //Serial.print(elapsedTime ); Serial.print(", "); Serial.print(Sp ); Serial.print(", "); Serial.print(P ); Serial.print(", "); Serial.print(rpm ); Serial.print(", "); Serial.print(arduinoPWM); Serial.print("\n");  } |
| **Código función en el sistema embebido del controlador PID de un grado de libertad** | void control1DOF() {    float Kp = 12.194801852041582;  float Ki = 1.836264365827931;  float Kd = 58.215749090323079;  float N = 0.135321750384943;  float kaw = 0.1;    double en = (Sp \* 0.0174533) - (P \* 0.0174533);    if (abs(en) > abs(0.065 \* Sp \* 0.0174533) ) {  float Ap = Kp \* en;  float Ad = Kd \* (en - en\_1) + N \* Ad\_1;  float I\_error = en + I\_error\_1 + kaw \* difact;  float Ai = Ki \* I\_error;  un = Ap + Ai + Ad;  us = un;  float ud = un;  if (en<0 && abs(ud) < 18) us = -18;  if ( en>0 && abs(ud) < 18) us = 18;      if (un < -100) {  us = -100;  }  if (un > 100) {  us = 100;  }      en\_1 = en;  float en\_2 = en\_1;  en\_3 = en\_2;  I\_error\_1 = I\_error;  difact = us - un;  Ad\_1 = Ad;    arduinoPWM = us;  if(en\_3 == en && abs(rpm) <2) us = us\*1.6;  if (abs(rpm) >= 0 && abs(rpm) < 0.7) us = us \* 1.5;  if (abs(rpm) >= 0.7 && abs(rpm) < 1) us = us \* 1.4;  if (abs(rpm) >= 1 && abs(rpm) < 2) us = us \* 1.3;  flagPWM = (us < 0) ? true : false;    }      } |
| **Código función en el sistema embebido del controlador PID de dos grados de libertad** | void control2DOF() {  double Kp = 0.01357670445385118;  double Ki = 1.822951239710286;  double Kd = 10.464560395668343;  double N = 0.782109908780621;  double Kpd = 1.285776886757601;  double Kdp = 13.408050560944484;  double N1 = 0.997168715666942;  double Ka = 0.1;    double en = (Sp \* 0.0174533) - (P \* 0.0174533);    if (abs(en) > abs(0.065 \* Sp \* 0.0174533) ) {    float Ap = Kp \* en;  float Ad = Kd \* (en - en\_1) + N \* Ad1\_;  float I\_error = en + I\_error\_1 + Ka \* difact;  float Ai = Ki \* I\_error;  float U1 = Ap + Ai + Ad;    float Ap1 = Kpd \* (P \* 0.0174533);  float Ad1 = Kdp \* ((P \* 0.0174533) - yn\_1) + N1 \* Ad\_1;  float U2 = Ap1 + Ad1;    float Uc = U1 - U2;  us = Uc;  float ud = Uc;  if (en < 0 && abs(ud) < 18) us = -18;  if ( en > 0 && abs(ud) < 18) us = 18;    if (Uc < -100) {  us = -100;  }  if (Uc > 100) {  us = 100;  }    en\_1 = en;  float en\_2 = en\_1;  en\_3 = en\_2;  yn\_1 = P \* 0.0174533;  I\_error\_1 = I\_error;  difact = us - Uc;  Ad\_1 = Ad1;  Ad1\_ = Ad;      arduinoPWM = us;  if (en\_3 == en && abs(rpm) < 2) us = us \* 1.4;  if (abs(rpm) >= 0 && abs(rpm) < 0.7) us = us \* 1.3;  //if (abs(rpm) >= 0.7 && abs(rpm) < 1) us = us \* 1.2;  //if (abs(rpm) >= 1 && abs(rpm) < 2) us = us \* 1.1;  flagPWM = (us < 0) ? true : false;    }  } |
| **Código función en el sistema embebido del controlador en cascada** |  |
| **Código en el sistema embebido del controlador LQR** | void controlEstados() {    double kaw = 0.1;  double CL[] = {1, 0};  double Ki[] = {13.038404810405284};  double Kes[] = {8.392729648618287, 6.196276250778606};  double X\_est[] = {(P \* 0.0174533), (rpm \* 0.104719755)}; // Corregir según valores de P y rpm  double backlashCompensation = 30;  double Kid = Ki[0] \* tMuestro / 1000;    double Pv = CL[0] \* X\_est[0] + CL[1] \* X\_est[1]; // Corregido el acceso a los elementos de los arreglos  double en = (Sp \* 0.0174533) - X\_est[0]; // Corregido el acceso al elemento del arreglo X\_est    if (abs(en) > abs(0.065 \* Sp \* 0.0174533)) {    double Vk = en + Vk\_1 + kaw \* dif\_act;  double un = Kid \* Vk - X\_est[1] \* Kes[0]; // Corregido el acceso a los elementos de los arreglos    us = un;  double ud = un;    if (en < 0 && abs(ud) < 13) us = -13;  if ( en > 0 && abs(ud) < 13) us = 13;    if (en < 0 && abs(ud) < 10 && abs(Sp) < 40) us = -10;  if ( en > 0 && abs(ud) < 10 && abs(Sp) < 40) us = 10;    if (un > 100) {  us = 100;  }  if (un < -100) {  us = -100;  }    dif\_act = us - un;  Vk\_1 = Vk;  arduinoPWM = us;    if (en\_3 == en&& abs(P) < 100) us = us \* 3.3;  if (en\_3 == en&& abs(P) > 100) us = us \* 3.1;      en\_1 = en;  float en\_2 = en\_1;  en\_3 = en\_2;        // mayor  if (abs(rpm) >= 0.7 && abs(rpm) < 3 && (abs(Sp) > abs(P)) && abs(P) < 100) us = us \* 3;  if (abs(rpm) >= 3 && abs(rpm) < 7 && (abs(Sp) > abs(P)) && abs(P) < 100) us = us \* 2.8;  if (abs(rpm) >= 7 && abs(rpm) < 10 && (abs(Sp) > abs(P)) && abs(P) < 100) us = us \* 2.6;  if (abs(rpm) >= 10 && abs(rpm) < 15 && (abs(Sp) > abs(P)) && abs(P) < 100) us = us \* 2.4;            flagPWM = (us < 0) ? true : false;    }      } |
| **Código función en el sistema embebido para minimizar los efectos de la zona muerta, no linealidades y backslash** | void controlEstadosCompe() {    double kaw = 0.1;  double CL[] = {1, 0};  double Ki[] = {13.038404810405284};  double Kes[] = {8.392729648618287, 6.196276250778606};  double X\_est[] = {(P \* 0.0174533), (rpm \* 0.104719755)}; // Corregir según valores de P y rpm  double backlashCompensation = 30;  double Kid = Ki[0] \* tMuestro / 1000;    double Pv = CL[0] \* X\_est[0] + CL[1] \* X\_est[1]; // Corregido el acceso a los elementos de los arreglos  double en = (Sp \* 0.0174533) - X\_est[0]; // Corregido el acceso al elemento del arreglo X\_est    if (abs(en) > abs(0.065 \* Sp \* 0.0174533)) {    double Vk = en + Vk\_1 + kaw \* dif\_act;  double un = Kid \* Vk - X\_est[1] \* Kes[0]; // Corregido el acceso a los elementos de los arreglos    // Implementación del compensador de backlash  if (en > 0 && en\_1 < 0) { // Intento de movimiento positivo  un += backlashCompensation; // Agregamos una compensación positiva  } else if (en < 0 && en\_1 > 0) { // Intento de movimiento negativo  un -= backlashCompensation; // Agregamos una compensación negativa  }    us = un;  double ud = un;    if (en < 0 && abs(ud) < 13) us = -13;  if ( en > 0 && abs(ud) < 13) us = 13;    if (en < 0 && abs(ud) < 10 && abs(Sp) < 40) us = -10;  if ( en > 0 && abs(ud) < 10 && abs(Sp) < 40) us = 10;    if (un > 100) {  us = 100;  }  if (un < -100) {  us = -100;  }    dif\_act = us - un;  Vk\_1 = Vk;  arduinoPWM = us;    if (en\_3 == en&& abs(P) < 100) us = us \* 3.3;  if (en\_3 == en&& abs(P) > 100) us = us \* 3.1;      en\_1 = en;  float en\_2 = en\_1;  en\_3 = en\_2;            // mayor  if (abs(rpm) >= 0.7 && abs(rpm) < 3 && (abs(Sp) > abs(P)) && abs(P) < 100) us = us \* 3;  if (abs(rpm) >= 3 && abs(rpm) < 7 && (abs(Sp) > abs(P)) && abs(P) < 100) us = us \* 2.8;  if (abs(rpm) >= 7 && abs(rpm) < 10 && (abs(Sp) > abs(P)) && abs(P) < 100) us = us \* 2.6;  if (abs(rpm) >= 10 && abs(rpm) < 15 && (abs(Sp) > abs(P)) && abs(P) < 100) us = us \* 2.4;            flagPWM = (us < 0) ? true : false;    }      } |

***VIDEO CON LOS RESULTADOS DE LA VALIDACIÓN EXPERIMENTAL***

|  |  |
| --- | --- |
| **Enlace web del controlador PID de un grado de libertad** | <https://youtu.be/c8wspE1FeqY> |
| **Enlace web del controlador PID de dos grados de libertad** | <https://youtu.be/heOr7KcdJ5o> |
| **Enlace web del controlador en cascada** |  |
| **Enlace web del controlador que minimiza efectos de la zona muerta, no linealidades y backslash** | <https://youtu.be/_6yDcYWksC4> |
| **Enlace web del controlador LQR** | <https://youtu.be/Ath1TLYX6pg> |