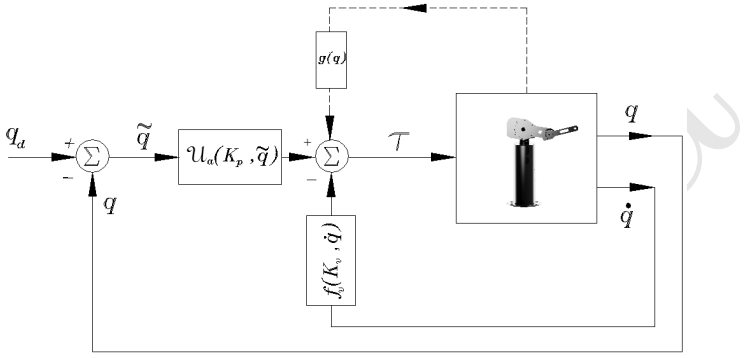
Capítulo 10

Control de robots manipuladores



Material Web

Simulación del regulador PD 2

Herramienta de MATLAB pidtool 35

Simulación con servomecanismos 39

10.1 Simulación del regulador PD

El algoritmo de control proporcional-derivativo más compensación de gravedad es uno de los esquemas más conocido en la literatura de control automático, en su versión escalar está formado por la siguiente ecuación:

$$\tau = k_p\tilde{q} - k_v\dot{q} + g(q)$$

donde $k_p, k_v \in \mathbb{R}_+$ son las ganancias proporcional y derivativa, respectivamente, el error de posición $\tilde{q}$ se define como la diferencia entre la posición deseada q_d y la posición q, es decir: $\tilde{q} = q_d - q$. La velocidad de movimiento es $\dot{q}$, la compensación del par gravitacional es $g(q)$ y τ es la energía aplicada al servomotor para producir movimiento en el robot.

El control proporcional derivativo más compensación de gravedad consiste de un término proporcional al error de posición $k_p\tilde{q}$ más un término de inyección de amortiguamiento a través de la velocidad de movimiento denominado acción de control derivativo $-k_v\dot{q}$ el cual tiene el efecto de un amortiguador o freno mecánico, es decir, genera efecto de amortiguamiento a través de la inyección de velocidad articular $\dot{q}$, absorbiendo los sobreimpulsos que tenga la respuesta transitoria, eliminando las oscilaciones y picos. En la etapa estacionaria la posición del extremo final es constante y por tal motivo la velocidad de movimiento es cero, desapareciendo el efecto de amortiguamiento. El signo menos que aparece en la acción derivativa disminuye la magnitud del control proporcional $k_p\tilde{q}$. Para sistemas mecánicos que se mueven en un plano vertical o en general en el espacio tridimensional existe el efecto gravitacional por lo que se requiere realizar la compensación de este fenómeno $g(q)$.

10.1.1 Péndulo

Considere como planta de estudio a un péndulo que se mueve en un plano vertical $x - y$ bajo la acción de la gravedad como se muestra en la figura 10.1. El péndulo consiste de un servomotor y una barra metálica (por ejemplo de aluminio) acoplada mecánicamente al rotor del servomotor. El movimiento del péndulo se realiza sobre un plano vertical xy, el eje z es perpendicular al plano de la hoja y la posición de casa $q_1 = 0$ es sobre el eje y_-; movimiento positivos corresponden al sentido en contra de las manecillas del reloj y para movimientos negativos se determinan en el sentido de las manecillas del reloj.

El péndulo es una planta de estudio vigente debido a su dinámica no lineal y sus aplicaciones prácticas

en ingeniería mecatrónica, lo que lo convierte en un sistema clave para propósitos de investigación científica. Particularmente en docencia representa una herramienta pedagógica muy útil para la enseñanza de algoritmos de control. El modelo dinámico del péndulo está dado por la siguiente ecuación:

$$\tau_1 \quad = \quad I_{p1}\ddot{q}_1 + m_1 g l_{c1} \operatorname{sen}(q_1) + b_1 \dot{q}_1$$

donde $I_{p1} \in \mathbb{R}_+$ es el momento de inercia del péndulo, $m_1 \in \mathbb{R}_+$ es la masa (del servomotor más la barra metálica), g es la constante de aceleración gravitacional, l_{c1} es el centro de masa, b_1 es el coeficiente de fricción viscosa, $q_1, \dot{q}_1, \ddot{q}_1 \in \mathbb{R}$ son la posición, velocidad y aceleración articular del péndulo, respectivamente y τ_1 representa la entrada o energía aplicada al servomotor. En el modelo dinámico del péndulo sólo fue considerado el fenómeno de fricción viscosa, mientras que la fricción de Coulomb y estática se omitieron.

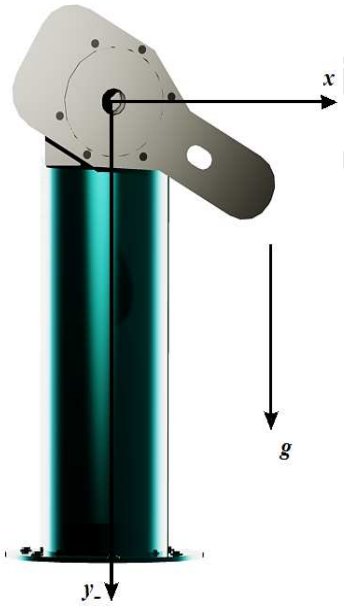


Figura 10.1 Péndulo.

El problema de regulación o control de posición para un péndulo consiste en colocar el el extremo final en una posición deseada q_{d1} (constante en el tiempo) para cualquier condición inicial. Es decir, matemáticamente el problema se describe como encontrar un regulador τ tal que la velocidad de movimiento $\dot{q}(t)$ y el error de posición $\tilde{q}$ convergen asintóticamente a cero $\forall t \geq 0$ sin importar las condiciones iniciales $\tilde{q}(0)$ y $\dot{q}(0)$, es decir:

$$\lim_{t\to\infty} \begin{bmatrix} \tilde{q}(t) \\ \dot{q}(t) \end{bmatrix} \to \begin{bmatrix} 0 \\ 0 \end{bmatrix} \in \mathbb{R}^2 \quad \forall\, t \geq 0.$$

Para finalidades prácticas, considere la simulación de un péndulo construido con un servomotor de transmisión directa con torque máximo de 15 Nm (±15 Nm para el primer y tercer cuadrante, respectivamente). La longitud de la barra l_1 es de 0.45m, los valores numéricos de los parámetros que forman el modelo dinámico como el momento de inercia I_{p1}, centro de gravedad l_{c1}, coeficientes de fricción viscosa b_1 están concentrados en la tabla 10.1. El problema de control consiste en posicionar el extremo final en 90 grados, considere condiciones iniciales posición $q_1(0) = 0$ grados y para la velocidad $\dot{q}_1$ =0 grados/seg; la posición de casa es colocada sobre el eje y_-.

Tabla 10.1 Parámetros del péndulo robot.

Masa	Fricción viscosa
m_1=5 kg	b_1 =0.17 Nm-seg/rad
Centro de masa	Longitud
l_{c1}=0.081 m	l_1=0.45 m
Mom. de inercia del rotor	Cap. del servomotor
I_{r1}=0.16 Nm-seg^2/rad	±15 Nm

El momento de inercia del péndulo I_{p1} se obtiene del momento de inercia del rotor I_{r1} y del centro de masa de la barra de aluminio l_{c2}, es decir $I_{p1} = I_{r1} + m_1 l_{c1}^2$.

Por lo tanto, la ecuación en lazo cerrado que define el problema de regulación o control de posición de un péndulo por medio del algoritmo proporcional derivativo más compensación de gravedad tiene la siguiente forma:

$$\frac{d}{dt}\begin{bmatrix}\tilde{q}_1\\ \dot{q}_1\end{bmatrix} = \begin{bmatrix}-\dot{q}_1\\ \underbrace{\frac{1}{[\,I_{r1}+m_1 l_{c1}^2\,]}}_{I_{p1}}[\,k_{p1}\tilde{q}_1 - k_{v1}\dot{q}_1 - b_1\dot{q}_1\,]\end{bmatrix}$$

El modelo dinámico del péndulo se encuentra implementado en el programa `pendulo.m` descrito en el cuadro 10.1, en la línea 12 se recibe la energía mecánica que proporciona el algoritmo proporcional derivativo para mover al péndulo a su posición deseada q_{d1}.

El código **MATLAB** del algoritmo de control proporcional derivativo PD más compensación de gravedad se encuentra en el archivo `pdpendulo.m` ubicado en el cuadro 10.2. Observe en la línea 15 la regla de sintonía

para la ganancia proporcional $k_{p1} \leq \frac{\tau_{max}}{q_{d1}}$, la cual garantiza no saturar al servoamplificador del péndulo, es decir trabaja en la parte lineal por lo que se evitan fenómenos como ruido mecánico, vibraciones, procesos térmicos y dinámica no modelada. Esto facilita que la respuesta del péndulo sea suave sin sobre impulsos, no oscilaciones. En la línea 20 se sintoniza la ganancia derivativa k_{v1} al 50 % del valor de la ganancia proporcional k_{p1}. Para realizar la simulación observe el cuadro 10.3 con el código fuente **MATLAB** del programa principal `pdpendulosimu.m`; se emplea el método de integración numérica Runge-Kutta a través de la función `ode45(...)`. La línea 13 registra la forma de la señal del par aplicado τ del control PD y en la figura 10.2 se ilustran los resultados de simulación.

Código Fuente 10.1 pendulo

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo pendulo.m Versión de **MATLAB** 2012a

```
1  function xp=pendulo(t,x)
2      %variables de estados (entradas) % q1=x(1); %posición articular
3      qp1=x(2); %velocidad articular
4      %parámetros del péndulo
5      m1=5; %masa
6      lc1=0.01; %centro de gravedad
7      g=9.81; %constante de aceleración gravitacional
8      b1=0.17; %coeficiente de fricción viscosa
9      Ir1=0.16; %momento de inercia del rotor
10     Ip1=Ir1+m1*lc1*lc1; %momento de inercia del péndulo: Ip = Ir1 + m1*lc1^2
11     %en este punto se inserta el algoritmo de control el cual proporciona la energía para mover al robot
12     [tau1, ~ ]= pdpendulo(q1,qp1); %par aplicado (controlador)
13     %aceleración articular del péndulo
14     qpp1=(tau1-b1*qp1-m1*g*lc1*sin(q1))/Ip1;
15     %vector de salida
16     xp=[qp1; %xp(1)=x(2) velocidad articular
17         qpp1]; %xp(2)=qpp aceleración articular
18 end
```

Código Fuente 10.2 pdpendulo

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo pdpendulo.m Versión de **MATLAB** 2012a

```
1  function [tau1, qt1] = pdpendulo(q1,qp1)
2      %variables de estados (entradas) q1=q_1(t) es la posición articular
3      % qp1=q̇_1(t) representa la velocidad articular
4      %parámetros de la compensación de gravedad del péndulo
5      m1=5;%masa
6      lc1=0.01;%centro de gravedad
7      g=9.81;%constante de aceleración gravitacional
8      %vector de pares de gravitacionales
9      par_grav = m1*g*lc1*sin(q1);% m_1 g l_c1 sen(q_1)
10     qd1=90;% posición deseada q_d1 en grados
11     tau_max=15;% par máximo del servomotor de transmisión directa
12     %Reglas de sintonía para las ganancias del controlador: k_p1 ≤ τ_max/q_d1 se evitan sobre impulsos y
13     %oscilaciones, la respuesta del péndulo se mejora notablemente ya que se trabaja en la zona lineal
14     %del servoamplificador, lejos de los límites de saturación. La sintonía de k_v1 es muy simple.
15     kp1= 0.8*(tau_max/qd1);% regla de sintonía para la ganancia proporcional k_p1 ≤ 0.8 τ_max/q_d1
16     %la sintonía de la ganancia derivativa k_v1 es un porcentaje de la ganancia proporcional k_p1
17     %debido a que la sintonía de k_p1 garantiza trabajar en la zona lineal del servoamplificador
18     %la sintonía para k_v1 es muy simple, se recomienda un porcentaje del 50 % al 90 % de k_p1
19     %lo anterior produce respuestas transitorias sin sobre impulsos, libres de oscilaciones y el estado
       estacionario sin rizo o ruido mecánico, mejorando el desempeño de control
20     kv1=0.5*kp1;% regla de sintonía para la ganancia derivativa: k_v1 = 0.5k_p1
21     qt1=pi*qd1/180-q1;%error de posición q̃(t) = q_d1 − q_1(t)
22     %control proporcional derivativo más compensación de gravedad
23     tau1=kp1*qt1-kv1*qp1+par_grav;%control PD: τ = k_p1 q̃_1 − k_v1 q̇_1 + m_1 g l_c1 sen(q_1)
24 end
```

En la línea 13 del cuadro 10.3 se registra la señal del par aplicado al servomotor τ y señal de error de posición $\tilde{q}_1$. Una vez que se realizó el proceso de integración numérica (ver línea cf:simu:ode45), el resultado se deposita en la variable de estado $\boldsymbol{x}(t) \in \mathbb{R}^2$ de la dinámica de la ecuación en lazo cerrado (modelo dinámico del péndulo con el algoritmo de control PD); dicha variable de estado $\boldsymbol{x}$ contiene en su primera componente $x_1(t) = q_1(t)$ (variable en **MATLAB** `q1`) el movimiento articular del péndulo y la segunda componente $x_2(t) = \dot{q}_1$ la velocidad del péndulo (variable **MATLAB** `qp1`). Con esta información se utilizan como argumentos de entrada en la función `pdpendulo(q1(k),qp1(k))` para obtener el par aplicado τ (variable **MATLAB** `tau1`) como se indica en la línea 14.

Código Fuente 10.3 pdpendulosimu

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo pdpendulosimu.m Versión de **MATLAB** 2012a

```
1  clc; clear all; close all;
2  format short
3  %parámetros de simulación
4  ti=0;%tiempo inicial (segundos)
5  tf=10;%tiempo de final (segundos)
6  h=0.001;%incremento de tiempo (segundos)
7  ts=ti:h:tf;%vector tiempo
8  ci=[0;0];%condiciones iniciales
9  opciones=odeset('RelTol',1e-3,'InitialStep',1e-3,'MaxStep',1e-3);
10 %solución numérica de la ecuación en lazo cerrado: modelo dinámico del péndulo y control PD
11 [t, x]=ode45('pendulo',ts,ci,opciones);
12 q1=x(:,1); qp1=x(:,2); [n,m]=size(t); tau1=zeros(n,1); qt1=zeros(n,1);
13 for  k=1:n
14 |   [tau1(k), qt1(k)] =pdpendulo(q1(k),qp1(k));% registra torque y error de posición del control PD
15 end
16 subplot(3,1,1); plot(t,180*qt1/pi)% grafica el error de posición q̃1 en grados vs t
17 subplot(3,1,2); plot( t, 180*q1/pi)% grafica la posición q1 en grados vs t
18 subplot(3,1,3); plot(t,tau1)% grafica el par o torque aplicado τ1vs t
```

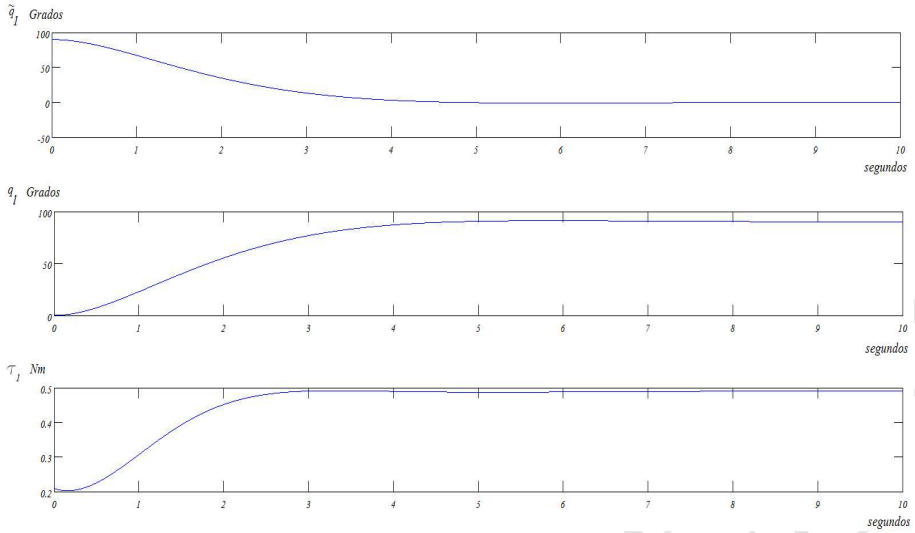


Figura 10.2 Respuesta del péndulo al control PD: $k_{p1} \leq \frac{15}{q_{d1}}$, $k_{v1} = 0.5k_{p1}$.

Para realizar la simulación del péndulo y control PD en MATLAB

Para llevar a cabo la simulación del péndulo con el algoritmo de control proporcional derivativo se requieren los siguientes archivos: el modelo dinámico del péndulo en `pendulo.m`, el algoritmo de control PD en `pdpendulo.m` y el programa principal `pdpendulosimu.m` los cuales pueden ser cargados desde su directorio de trabajo al editor de **MATLAB**.

La simulación puede ser realizada desde el editor integrado del entorno de programación de **MATLAB** seleccionando el archivo `pdpendulosimu.m` y oprimiendo el icono *play* comando (`run`).

También la simulación la puede realizar directamente en la ventana de comandos de **MATLAB** por teclear el nombre del programa principal (previo indicar la trayectoria donde se encuentran todos los archivos), por ejemplo:

```
fx >> cd('C:/robot/control') ↩
fx >> pdpendulosimu ↩
```

Es recomendable que en el momento de realizar la descarga de archivos desde el sitio Web del libro todos los programas fuente **MATLAB** los almacene en la misma carpeta o directorio de su preferencia.

Control tangente hiperbólico

Un algoritmo de control con mejor desempeño al proporcional derivativo es el tangente hiperbólico tanh(...) el cual tiene la siguiente estructura matemática:

$$\tau = k_{p1} \tanh(\tilde{q}_1) - k_{v1} \tanh(\dot{q}_1) + g(q_1)$$

donde $k_{p1}, k_{v1} \in \mathbb{R}_+$ son las ganancias proporcional y derivativa, respectivamente, el error de posición $\tilde{q}$ es argumento de entrada a la función tangente hiperbólica $k_p \tanh(\tilde{q}_1)$, la acción derivativa está dada por $-k_{v1} \tanh(\dot{q}_1)$ más la compensación del par gravitacional es $g(q_1)$.

La figura 10.3 muestra el perfil de la función tangente hiperbólica $\tanh(\tilde{q}_1)$en función del error, observe sus propiedades naturales de saturación en $pm1$ y la forma de su gráfica corresponde a la respuesta de un servoamplificador (zona lineal y de saturación). La gráfica de la tangente hiperbólica para la acción derivativa $\tanh(\dot{q}_1)$ resulta de forma similar. Bajo esta idea, las ganancias proporcional k_{p1} y derivativa k_{v1} corresponden a los límetes de saturación de la función tanh(...) para la parte proporcional y freno mecánico, respectivamente.

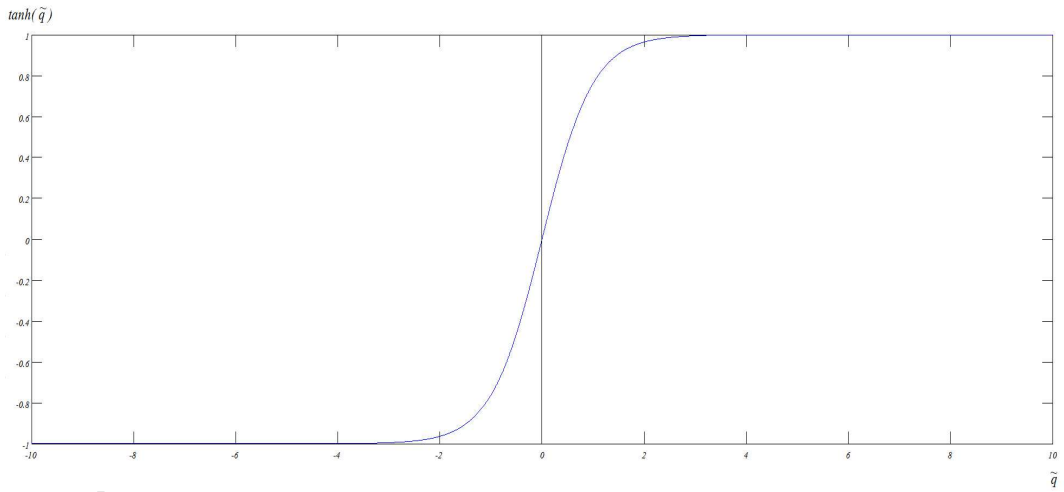


Figura 10.3 Función tanh($\tilde{q}$).

Por lo tanto, a diferencia con la sintonía de ganancias para el algoritmo de control proporcional o derivativo, resulta mucha más simple para el control tangente hiperbólica, ya que no depende de la posición deseada q_{d1}. En este caso, sólo debe satisfacer la ganancia proporcional ser menor al par o torque máximo del servomotor, se propone: $k_{p1} \leq 0.8\tau_{max}$ y mantener una relación con la ganancia derivativa como $k_{v1} = k_{p1}$.

Considere el cuadro 10.4 con el código fuente **MATLAB** del modelo dinámico del péndulo cuyo código corresponde al cuadro 10.1 del archivo `pendulo.m`. La única diferencia se encuentra en la línea 12 está en comentarios el algoritmo de control PD y en la línea 13 se activa la energía que entrega el algoritmo de control tangente hiperbólica, el cual se describe en el archivo `tanhpendulo.m` del cuadro 10.5. Note que en la línea 15 se programa la regla de sintonía de la ganancia proporcional $k_{p1} \leq 0.8\tau_{max}$.

Por lo tanto, para realizar correctamente la simulación del péndulo con el control tanh(...) tome en cuenta el siguiente procedimiento:

Para realizar la simulación del péndulo y control tanh(...) en MATLAB

- Para llevar a cabo la simulación del péndulo con el algoritmo de control tangente hiperbólico se requieren los siguientes archivos: el modelo dinámico del péndulo en `pendulo.m`, el algoritmo de control tanh(...) en `tanhpendulo.m` y el programa principal `tanhpendulosimu.m` los cuales pueden ser cargados desde su directorio de trabajo hacia el editor de **MATLAB**.
- **IMPORTANTE:** el archivo `pendulo.m` es el mismo que ya se utilizó con el control PD, para que se pueda utilizar con el control tanh(...) refiérase al cuadro 10.4 en la línea 12 colocar al algoritmo PD como comentario e insertar la función del control tanh(...) como se indica en la línea 13, salvar nuevamente todos los archivos.
- La simulación puede ser realizada desde el editor integrado del entorno de programación de **MATLAB** seleccionando el archivo `tanhpendulosimu.m` y oprimiendo el icono *play* comando (`run`).
- También la simulación puede realizarla directamente en la ventana de comandos de **MATLAB** por teclear el nombre del programa principal (previo indicar la trayectoria donde se encuentran todos los archivos), por ejemplo:

  $fx >>$ `cd('C:/robot/control')` $\hookleftarrow$

  $fx >>$ `tanhpendulosimu` $\hookleftarrow$
- Es recomendable que en el momento de realizar la descarga de archivos desde el sitio Web del libro todos los programas fuente **MATLAB** los almacene en la misma carpeta o directorio de su preferencia.

La figura 10.4 muestra el comportamiento que tiene el error de posición $\tilde{q}_1$. Note que la convergencia

asintótica a cero, lo que significa que la respuesta $q_1(t)$ del péndulo bajo el control tanh(...) alcanza el valor deseado de 90 grados (gráfica intermedia). Observe que el desplazamiento articular q_1 tiene un perfil muy suave en estado transitorio y el estado estacionario permanece constante. El par aplicado al servomotor (gráfica inferior) tiene un máximo pico de 11.006 Nm, disminuyendo rápidamente en fracciones de segundo hasta alcanzar el valor de 0.4095 Nm el cual corresponde a la compensación del par gravitacional. La ventaja del algoritmo de control tanh(...) en comparación al PD es que la sintonía de las ganancias proporcional y derivativa no dependen de la posición deseada q_{d1}, únicamente del par máximo del servomotor. En contraste, el algoritmo proporcional derivativo requiere una nueva sintonía en sus ganancias cada vez que se modifique la posición deseada q_{d1}.

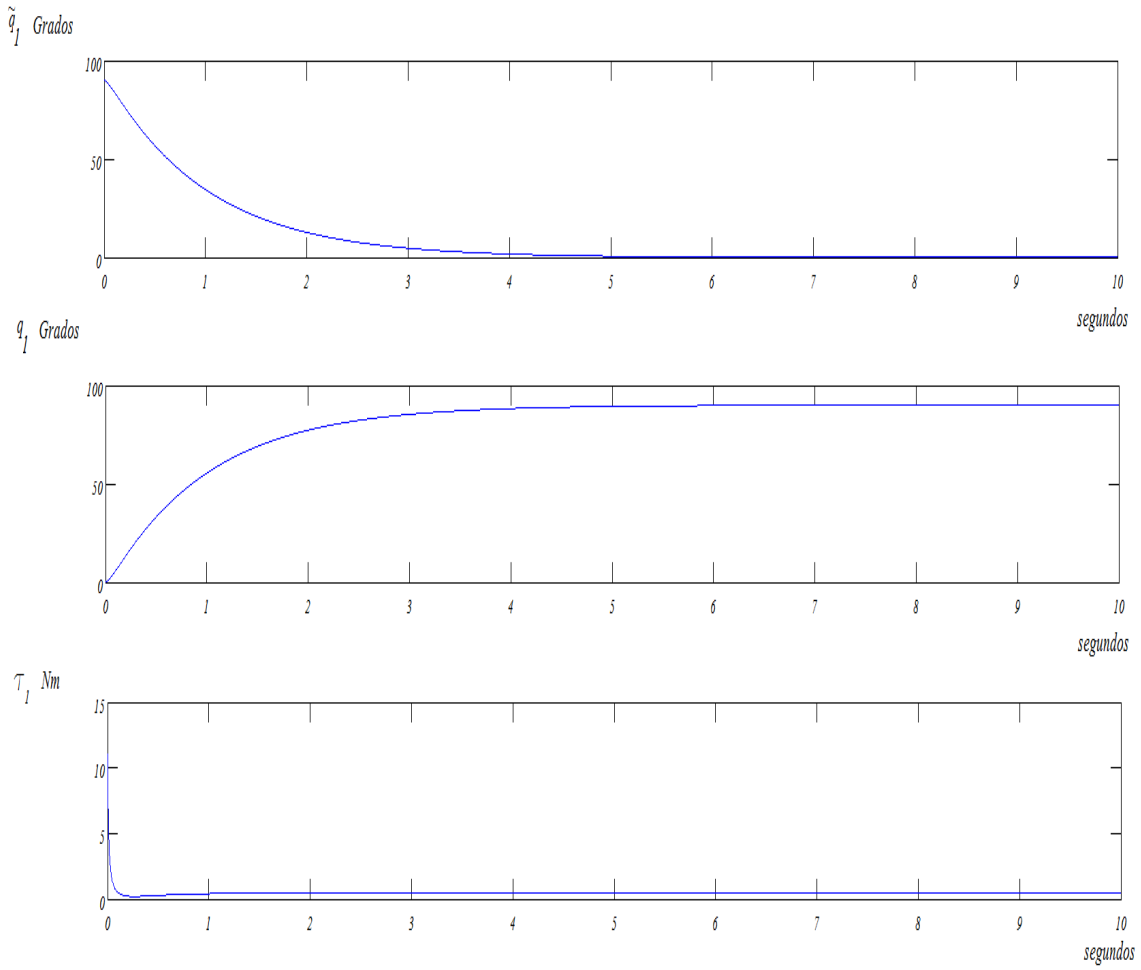


Figura 10.4 Respuesta del péndulo utilizando control tanh(...).

El modelo dinámico del péndulo descrito en el archivo `pendulo.m` del cuadro 10.1 se vuelve a presentar en el cuadro 10.4 con la finalidad de advertir al lector que en la línea 12 el algoritmo de control PD que inicialmente fue programado se desactiva por medio de un % comentario y en la línea 13 se inserta el código [`tau1, ∼ ]= tanhpendulo(q1,qp1);` correspondiente al control tanh(...).

La idea principal consiste para nuevos diseños de control o futuros algoritmos se puedan insertar de forma parecida utilizando el modelo dinámico del péndulo.

Código Fuente 10.4 pendulo

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo pendulo.m Versión de **MATLAB** 2012a

```
1  function xp=pendulo(t,x)
2      q1=x(1);%posición articular
3      qp1=x(2);%velocidad articular
4      %parámetros del péndulo
5      m1=5;%masa
6      lc1=0.01;%centro de gravedad
7      g=9.81;%constante de aceleración gravitacional
8      b1=0.17;%coeficiente de fricción viscosa
9      Ir1=0.16;%momento de inercia del rotor
10     Ip1=Ir1+m1*lc1*lc1;%momento de inercia del péndulo: Ip = Ir1 + m1*lc1^2
11     %en este punto se inserta el algoritmo de control el cual proporciona la energía para mover al robot
12     %[tau1, ∼ ]= pdpendulo(q1,qp1);% se desactiva el control PD como comentario
13     [tau1, ∼ ]= tanhpendulo(q1,qp1);%en esta línea se inserta el control tanh(...)
14     %aceleración articular del péndulo
15     qpp1=(tau1-b1*qp1-m1*g*lc1*sin(q1))/Ip1;
16     %vector de salida
17     xp=[qp1;%xp(1)=x(2) velocidad articular
18         qpp1];%xp(2)=qpp aceleración articular
19 end
```

Código Fuente 10.5 tanhpendulo

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo tanhpendulo.m Versión de MATLAB 2012a

```
1  function [tau1, qt1] = tanhpendulo(q1,qp1)
2      %variables de estados (entradas) q1=q_1(t) es la posición articular
3      % qp1=q̇_1(t) representa la velocidad articular
4      %parámetros de la compensación de gravedad del péndulo
5      m1=5;%masa
6      lc1=0.01;%centro de gravedad
7      g=9.81;%constante de aceleración gravitacional
8      %vector de pares de gravitacionales
9      par_grav = m1*g*lc1*sin(q1);% m_1 g l_c1 sen(q_1)
10     qd1=90;% posición deseada q_d1 en grados
11     qt1=pi*qd1/180-q1;%error de posición en radianes
12     tau_max=15;% par máximo 15 Nm del servomotor de transmisión directa
13     %corresponde al servomotor modelo 1015B de Parker-Compumotor
14     % regla de sintonía para las ganancias del algoritmo de control tanh(...)
15     kp1= 0.8*tau_max;%regla de sintonía para la ganancia proporcional k_p1 ≤ 0.8τ_max
16     %con el 80% del par máximo como regla de sintonía para k_p1 se garantiza que
17     %el servoamplificador trabajará en la zon lineal, lejos de la zona de saturación
18     kv1=kp1;%regla de sintonía para la ganancia derivativa k_v1 = k_p1
19     %control tanhgente hiperbólica: τ_1 = k_p1 tanh(q̃_1) − k_v1 tanh(q̇_1) + m_1 g l_c1 sen(q_1)
20     tau1=kp1*tanh(qt1)-kv1*tanh(qp1)+par_grav;
21 end
```

Código Fuente 10.6 tanhpendulosimu

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo tanhpendulosimu.m Versión de **MATLAB** 2012a

```
clc;
clear all;
close all;
format short
%parámetros de simulación
ti=0;%tiempo inicial (segundos)
tf=10;%tiempo de final (segundos)
h=0.001;%incremento de tiempo (segundos)
ts=ti:h:tf;%vector tiempo
ci=[0;0];%condiciones iniciales q1(0) = 0 y q1p(0) = 0
opciones=odeset('RelTol',1e-3,'InitialStep',1e-3,'MaxStep',1e-3);
%solución numérica de la ecuación en lazo cerrado: modelo dinámico del péndulo y control tanh(...)
[t, x]=ode45('pendulo',ts,ci,opciones);
q1=x(:,1); qp1=x(:,2); [n,m]=size(t); tau1=zeros(n,1); qt1=zeros(n,1);
for  k=1:n
    [tau1(k), qt1(k)] =tanhpendulo(q1(k),qp1(k));%registra torque y error de posición de tanh(...)
end
%se obtienen las gráficas del error de posición q1t, posición q1 y torque aplicado tau1
subplot(3,1,1); plot(t,180*qt1/pi)% grafica el error de posición q1t en grados vs t
subplot(3,1,2); plot( t, 180*q1/pi)% grafica la posición q1 en grados vs t
subplot(3,1,3); plot(t,tau1)% grafica el par o torque aplicado tau1vs t
```

10.1.2 Control de posición de un robot manipulador de 2 gdl

Los robots manipuladores de 2 gdl tienen amplias aplicaciones en la industria realizando actividades de pintado y traslado de objetos, ensamble, estibado, corte de figuras, etc. Considere el robot manipulador de 2 gdl que se mueve en el plano $x-y$ como se indica en la figura 10.5.

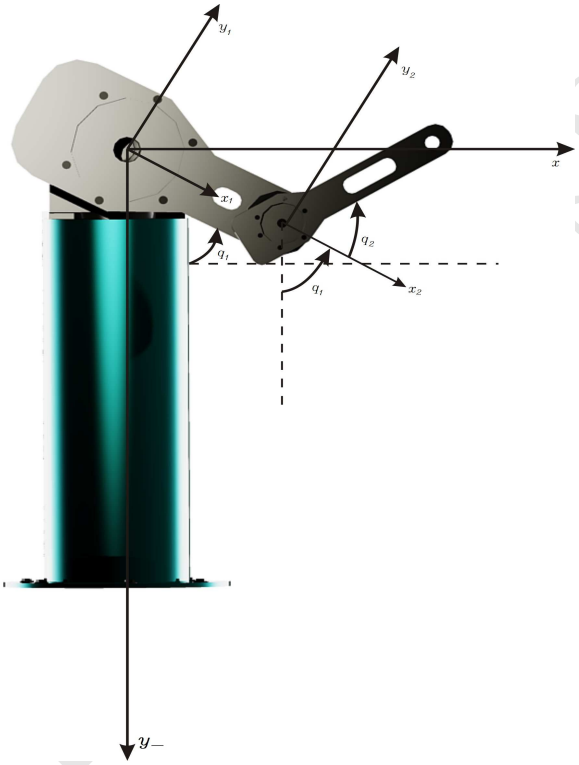


Figura 10.5 Robot manipulador planar de 2 gdl.

El modelo dinámico de un robot manipulador antropomórfico de 2 gdl sin tomar en cuenta la fricción de Coulomb y estática está dado por la siguiente expresión:

$$\boldsymbol{\tau} = M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{f}_f(\dot{\boldsymbol{q}})$$

$$\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix} = \underbrace{\begin{bmatrix} m_1 l_{c1}^2 + m_2 l_1^2 + m_2 l_{c2}^2 + 2m_2 l_1 l_{c2}\cos(q_2) + I_1 + I_2 & m_2 l_{c2}^2 + m_2 l_1 l_{c2}\cos(q_2) + I_2 \\ m_2 l_{c2}^2 + m_2 l_1 l_{c2}\cos(q_2) + I_2 & m_2 l_{c2}^2 + I_2 \end{bmatrix}}_{M(\boldsymbol{q})} \begin{bmatrix}\ddot{q}_1\\ \ddot{q}_2\end{bmatrix}$$

$$+\underbrace{\begin{bmatrix} -2m_2l_1l_{c2}\ \text{sen}(q_2)\dot{q}_2 & -m_2l_1l_{c2}\ \text{sen}(q_2)\dot{q}_2 \\ m_2l_1l_{c2}\ \text{sen}(q_2)\dot{q}_1 & 0 \end{bmatrix}}_{C(\boldsymbol{q},\dot{\boldsymbol{q}})}\begin{bmatrix}\dot{q}_1\\ \dot{q}_2\end{bmatrix}+$$

$$\underbrace{g\begin{bmatrix} l_{c1}m_1\ \text{sen}(q_1)+m_2l_1\ \text{sen}(q_1)+m_2l_{c2}\ \text{sen}(q_1+q_2) \\ m_2l_{c2}\ \text{sen}(q_1+q_2)\end{bmatrix}}_{\boldsymbol{g}(\boldsymbol{q})}+\underbrace{\begin{bmatrix} b_1 & 0\\ 0 & b_2\end{bmatrix}\begin{bmatrix}\dot{q}_1\\ \dot{q}_2\end{bmatrix}}_{\boldsymbol{f}_f(\dot{\boldsymbol{q}})}$$

La terminología y significado de los parámetros del robot de 2 gdl se encuentran en la tabla 10.2.

Tabla 10.2 Parámetros del robot planar de 2 gdl

Eslabón	Significado	Notación	Valor
Hombro	Masa del eslabón 1	m_1	23.902 kg
	Longitud del eslabón 1	l_1	0.45 m
	Inercia del eslabón 1	I_1	1.266 Nm seg^2/rad
	Centro de masa del eslabón 1	l_{c1}	0.091 m
	Coeficiente de fricción viscosa	b_1	2.288 Nm seg/rad
Codo	Masa del eslabón 2	m_2	3.88 kg
	Longitud del eslabón 2	l_2	0.45 m
	Inercia del eslabón 2	I_2	0.093 Nm seg^2/rad
	Centro de masa del eslabón 2	l_{c2}	0.048
	Coeficiente de fricción viscosa	b_2	0.175 Nm seg/rad
	Aceleración debida a la gravedad	g	9.81 m/seg^2

Para propósitos de simulación y desarrollo de aplicaciones se requiere el modelo dinámico numérico del robot manipulador, a continuación se presenta el modelo numérico de un robot prototipo de 2 gdl:

$$\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix} = \underbrace{\begin{bmatrix} 2.351+0.1676\ \cos(q_2) & 0.102+0.0838\ \cos(q_2)\\ 0.102+0.0838\ \cos(q_2) & 0.102\end{bmatrix}}_{M(\boldsymbol{q})}\begin{bmatrix}\ddot{q}_1\\ \ddot{q}_2\end{bmatrix}+$$

$$\underbrace{\begin{bmatrix} -0.1676\ \text{sen}(q_2)\dot{q}_2 & -0.0838\ \text{sen}(q_2)\dot{q}_2 \\ 0.084\ \text{sen}(q_2)\dot{q}_1 & 0.0 \end{bmatrix}}_{C(\boldsymbol{q},\dot{\boldsymbol{q}})} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} +$$

$$\underbrace{9.81 \begin{bmatrix} 3.9211\ \text{sen}(q_1) + 0.1862\ \text{sen}(q_1+q_2) \\ 0.1862\ \text{sen}(q_1+q_2) \end{bmatrix}}_{\boldsymbol{g}(\boldsymbol{q})} + \underbrace{\begin{bmatrix} 2.288 & 0 \\ 0 & 0.175 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}}_{\boldsymbol{f}_f(\dot{\boldsymbol{q}})}$$

Control de posición de un robot manipulador de 2 gdl

El problema de control de posición o regulación de un robot manipulador de 2 gdl consiste en llevar su extremo final a un punto deseado constante en el tiempo $\boldsymbol{q}_d$ sin importar las condiciones iniciales $[\,\boldsymbol{q}(0) \quad \dot{\boldsymbol{q}}(0)\,]^T$.

Desde el punto de vista matemático la descripción del problema de posición significa encontrar una ley de control $\boldsymbol{\tau}$ tal que el error de posición $\tilde{\boldsymbol{q}}(t)$ y la velocidad de movimiento $\dot{\boldsymbol{q}}(t)$ asintóticamente tiendan al punto de equilibrio, es decir:

$$\lim_{t\to\infty} \begin{bmatrix} \tilde{\boldsymbol{q}}(t) \\ \dot{\boldsymbol{q}}(t) \end{bmatrix} \to \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{2n} \quad \forall\ t \geq 0.$$

Simulación de control de posición de un robot manipulador de 2 gdl

La simulación consiste en llevar a las articulaciones del hombro y codo de un robot manipulador planar de 2 gdl a las siguientes posiciones deseadas: $[\,q_{d1} \quad q_{d2}\,] = [\,45 \quad 90\,]$ grados.

Considere el siguiente algoritmo de control proporcional derivativo más compensación de gravedad:

$$\begin{aligned} \boldsymbol{\tau} &= K_p\tilde{\boldsymbol{q}} - K_v\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) \\ \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} &= \begin{bmatrix} k_{p1} & 0 \\ 0 & k_{p2} \end{bmatrix} \begin{bmatrix} \tilde{q}_1 \\ \tilde{q}_2 \end{bmatrix} - \begin{bmatrix} k_{v1} & 0 \\ 0 & k_{v2} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} + 9.81 \begin{bmatrix} 3.9211\ \text{sen}(q_1) + 0.1862\ \text{sen}(q_1+q_2) \\ 0.1862\ \text{sen}(q_1+q_2) \end{bmatrix} \end{aligned}$$

suponga que los límites físicos de los servomotores de transmisión directa del robot manipulador son los que se indican en la tabla 10.3.

La **regla de sintonía** tiene la finalidad de trabajar a los servoamplificadores en la parte lineal, es decir no saturar sus límites físicos $\tau_{1_{max}}$ y $\tau_{2_{max}}$. La sintonía de las ganancias proporcional queda en función del

torque máximo τ_{max_i} y de la posición deseada q_{di}, para $i = 1, 2$; es decir: $k_{pi} \leq \frac{\tau_{max_i}}{q_{d1}}$. La ganancia derivativa se calcula como el 50 % de la ganancia proporcional $k_{vi} = 50\,\%k_{pi}$, con $i = 1, 2$.

Tabla 10.3 Límites físicos de los servomotores del robot de 2 gdl.

Articulación	Par máximo τ_{max}
Hombro	$\tau_{max_1} = 150$ Nm
Codo	$\tau_{max_2} = 15$ Nm

Regla de sintonía de las ganancias proporcional y derivativa

- $k_{pi} \leq 0.8\tau_{i_{max}}$, para $i = 1, 2$.
- $k_{vi} = 0.5k_{pi}$, para $i = 1, 2$.

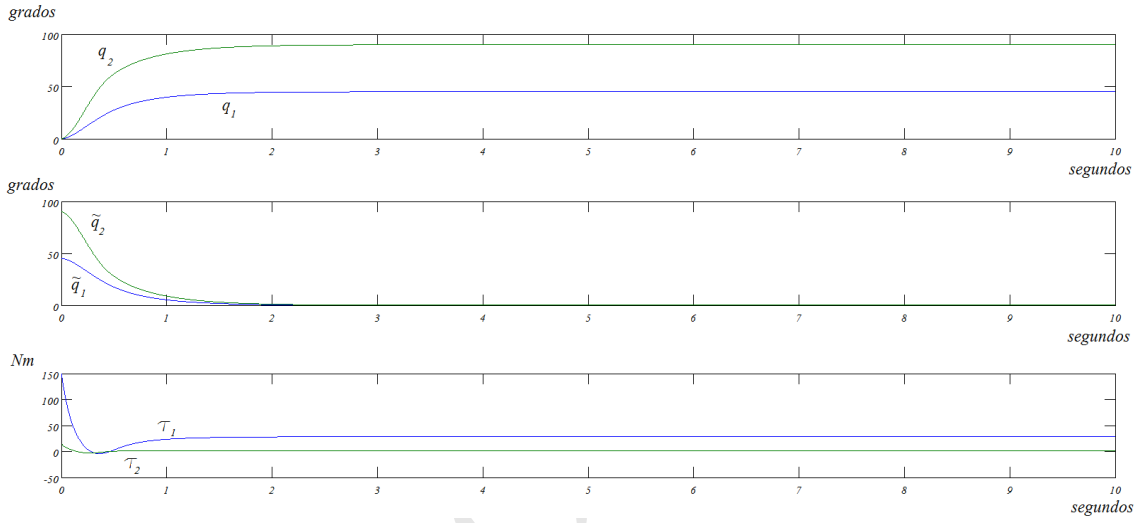
El cuadro 10.7 describe el código fuente en MATLAB para implementar el modelo dinámico de un robot manipulador de 2 gdl (ver programa `robot2gdl.m`); note que el vector de estados $\boldsymbol{x} = [\, q_1 \quad q_2 \quad \dot{q}_1 \quad \dot{q}_2 \,]^T \in \mathbb{R}^4$ se encuentra indicado de la línea 4 a la línea 9. La matriz de inercia $M(\boldsymbol{q}) \in \mathbb{R}^{2\times 2}$ se encuentra implementada en la línea 10, el par gravitacional y fenómeno de fricción se encuentran en las líneas 13 y 18, respectivamente.

La energía aplicada a los servomotores del robot manipulador para mover los eslabones hacia el vector de posición deseada ($\boldsymbol{q}_d = [\, 45 \quad 90 \,]^T$ grados) se obtiene por medio del algoritmo de control proporcional derivativo (PD) más compensación de gravedad el cual se ubica en la línea 23. Observe que la función PD retorna el par aplicado $\boldsymbol{\tau}$ y la variable error $\tilde{\boldsymbol{q}}$; sin embargo, la variable de estado $\tilde{\boldsymbol{q}}$ no se utiliza en la función del modelo dinámico `robot2gdl.m`, por lo que se deshabilita por medio del comando $\sim$ (ver línea 23). La aceleración del robot o ecuación en lazo cerrado está en la línea 24 y la deriva del vector de estado $\dot{\boldsymbol{x}}$ se encuentra en la línea 25.

El algoritmo de control proporcional derivativo PD más compensación de gravedad se encuentra en el programa `PD.m` descrito en el cuadro 10.8. La sintonía de las ganancias está implementada en las líneas 14-21; ambas ganancias se incorporar a la programación del control PD como se ilustra en la línea 24.

El programa principal se denomina `robot2gdlPDsimu.m` cuyo código se presenta en el cuadro 10.9; el tiempo de simulación es de 0 a 10 segundos con incrementos fijos de un milisegundo como se encuentra programado

en la línea 9, las condiciones iniciales son $\boldsymbol{x}(0) = [\, q_1(0) \quad q_2(0) \quad \dot{q}_1(0) \quad \dot{q}_2(0) \,]^T = [\, 0 \quad 0 \quad 0 \quad 0 \,]^T$ (ver línea 10). Por medio de integración numérica Runge-Kutta `ode45(...)` en la línea 13 se realiza la solución de la ecuación en lazo cerrado (modelo dinámico del robot y control PD). El registro del par aplicado $\boldsymbol{\tau}$ y error de posición $\tilde{\boldsymbol{q}}$ se lleva a cabo en la línea 13 y las gráficas de la simulación se presentan a partir de la línea 20, los resultados indican perfiles de la posición con transitorio suave y convergencia asintótica del error de posición a cero; el par aplicado de cada articulación se encuentra dentro de los límites físicos de los servoamplificadores del robot como se ilustra en la figura 10.6.



Simulación en MATLAB del robot de 2 gdl y control PD

Para llevar a cabo la simulación del robot de 2 gdl con el algoritmo de control PD se requieren los siguientes archivos: el modelo dinámico del robot de 2 gdl en `robot2gdl.m`, el algoritmo de control PD en `PD.m` y el programa principal `robot2gdlPDsimu.m` (**todos los archivos deben estar almacenados en la misma carpeta o directorio del usuario**).

La simulación puede ser realizada desde el editor integrado del entorno de programación de **MATLAB** seleccionando el archivo `robot2gdlPDsimu.m` y oprimiendo el icono *play* comando (`run`).

Otra forma para realizar la simulación es tecleando el nombre del programa principal en la ventana de comandos de **MATLAB** (previo indicar la trayectoria donde se encuentran los 3 archivos), por ejemplo:

```
fx >>  cd('C:/robot/control')  ↩
fx >>  robot2gdlPDsimu  ↩
```

Código Fuente 10.7 robot2gdl

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo robot2gdl.m Versión de **MATLAB** 2012a

```
1  %Modelo dinámico numérico del robot manipulador planar de 2 gdl
2  function xp =robot2gdl(t,x)
3      %vector de estados x = [ q1   q2   q̇1   q̇2 ]^T ∈ ℝ^4
4      q1=x(1);%posición de la articulación del hombro q1
5      q2=x(2);%posición de la articulación del codo q2
6      q = [q1; q2];%vector de posición articular q̇ = [ q1   q2 ]^T ℝ^2
7      qp1=x(3);%velocidad articular del hombro q̇1
8      qp2=x(4);% velocidad articular del codo q̇2
9      qp = [qp1; qp2];%vector de velocidad articular q̇ = [ q̇1   q̇2 ]^T ∈ ℝ^2
10     %matriz de inercias M(q) ∈ ℝ^2×2
11     M=[2.351+0.1676*cos(q2), 0.102+0.0838*cos(q2);
12         0.102+0.0838*cos(q2), 0.102];
13     %matriz de fuerzas centrípetas y de Coriolis C(q,q̇) ∈ ℝ^2×2
14     C=[-0.1676*sin(q2)*qp2, -0.0838*sin(q2)*qp2;
15         0.084*sin(q2)*qp1, 0.0];
16     %par gravitacional g(q) ∈ ℝ^2
17     gq=9.81*[3.9211*sin(q1)+0.1862*sin(q1+q2);
18         0.1862*sin(q1+q2)];
19     %fricción viscosa f_f(q̇) = Bq̇ ∈ ℝ^2
20     fr=[2.288*qp1;
21         0.175*qp2];
22     %algoritmo de control proporcional derivativo
23     [tau, ~ ]=PD(q,qp);%energía proporcionada por el control PD para mover al robot hacia q_d
24     qpp = M^(-1)*(tau-C*qp-gq-fr);%dinámica del robot y estructura del algoritmo de control
25     xp = [qp1; qp2; qpp(1); qpp(2)];%vector de salida ẋ = [ q̇1   q̇2   q̈1   q̈2 ]^T ∈ ℝ^4
26 end
```

Código Fuente 10.8 PD

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo PD.m Versión de **MATLAB** 2012a

```
%Algoritmo de control proporcional derivativo PD para un robot de 2 gdl
function [tau, qtilde] = PD(q,qp)
    q1=q(1);%posición articular del hombro q_1
    q2=q(2);%posición articular del codo q_2
    qp1=qp(1); %velocidad articular del hombro q̇_1
    qp2=qp(2);%velocidad articular del hombro q̇_2
    %compensación del par gravitacional
    par_grav=(9.81*[3.9211*sin(q1)+0.1862*sin(q1+q2);
            0.1862*sin(q1+q2)]);
    %vector de posiciones deseadas q_d = [ q_d1   q_d2 ]^T
    qd=(pi/180)*[45 ;% posición deseada del hombro
             90];% posición deseada del codo
    %regla de sintonía para la ganancia proporcional
    kp1=150/qd(1);%regla de sintonía para la ganancia proporcional del hombro k_p1 ≤ τ_1max / q_d1
    kp2=15/qd(2);%regla de sintonía para la ganancia proporcional del codo k_p2 ≤ τ_2max / q_d2
    Kp=[kp1, 0;
        0, kp2];%matriz de ganancia proporcional K_p ∈ IR^{2×2}
    %regla de sintonía para la ganancia derivativa
    kv1=0.5*kp1; kv2=0.5*kp2;%sintonía para las ganancias derivativas: k_vi ≤ 0.5k_pi para i = 1, 2
    Kv=[kv1, 0;
        0, kv2];%matriz de ganancia derivativa K_v ∈ IR^{2×2}
    qtilde=qd-q;% error de posición q̃ = q_d − q
    %algoritmo de control PD: τ = K_p q̃ − K_v q̇ + g(q)
    tau=Kp*qtilde-Kv*qp+par_grav ;
end
```

Código Fuente 10.9 robot2gdlPDsimu

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo robot2gdlPDsimu.m Versión de MATLAB 2012a

```
%Programa principal para la simulación del robot manipulador de 2 gdl y control PD
clc;
clear all;
close all;
format short g
ti=0;%tiempo inicial de simulación
h=0.001;%incremento de un milisegundo el tiempo t = t + h
tf = 10;%tiempo final de simulación
t=ti:h:tf;%vector tiempo
ci=[0; 0; 0; 0];%condiciones iniciales [q1(0)  q2(0)  q̇1(0)  q̇2(0)]^T = [0  0  0  0]^T
opciones=odeset('RelTol',h*1e-3,'InitialStep', h,'MaxStep',h);
%solución numérica del modelo dinámico del robot manipulador de 2 gdl y control PD
[t, x]=ode45('robot2gdl',t,ci,opciones);
q1=x(:,1); q2=x(:,2);%se asignan posiciones articulares
qp1=x(:,3); qp2=x(:,4);% velocidades articulares
[n, m]=size(t);%obtiene el número de renglones n y columnas m para las variables auxiliares q̃ y τ
tau1=zeros(n,1); tau2=zeros(n,1);%variables auxiliares para el registro del par aplicado τ
qtilde1=zeros(n,1); qtilde2=zeros(n,1);%variables auxiliares para el registro del error de posición q̃
for k=1:n
    [tau, qtilde] = PD([q1(k);q2(k)], [qp1(k); qp2(k)]);%registra error de posición q̃ y par τ
    tau1(k)=tau(1); tau2(k)=tau(2);%registra par o torque τ
    qtilde1(k)=qtilde(1); qtilde2(k)=qtilde(2);% registra error de posición q̃
end
subplot(3,1,1); plot( t, 180*q1/pi, t,180*q2/pi)%grafica posiciones (q1, q2) en grados
subplot(3,1,2); plot(t, 180*qtilde1/pi, t, 180*qtilde2/pi)%grafica errores de posición en grados (q̃1, q̃2)
subplot(3,1,3); plot( t, tau1, t,tau2)% grafica los torques o pares aplicados (τ1, τ2)
```

Control saturado de un robot de 2 gdl

Otra estrategia de control para un robot de 2 gdl es la estructura con acciones acotadas de control como la que se muestra a continuación:

$$\boldsymbol{\tau} = K_p \frac{\operatorname{senh}(\tilde{\boldsymbol{q}})}{1+\cosh(qt)} - K_v \frac{\operatorname{senh}(\dot{\boldsymbol{q}})}{1+\cosh(qp)} + \boldsymbol{g}(\boldsymbol{q})$$

$$\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix} = \begin{bmatrix}k_{p1} & 0\\ 0 & k_{p2}\end{bmatrix} \begin{bmatrix}\frac{\operatorname{senh}(\tilde{q}_1)}{1+\cosh(\tilde{q}_1)}\\ \\ \frac{\operatorname{senh}(\tilde{q}_2)}{1+\cosh(\tilde{q}_2)}\end{bmatrix} - \begin{bmatrix}k_{v1} & 0\\ 0 & k_{v2}\end{bmatrix} \begin{bmatrix}\frac{\operatorname{senh}(\dot{q}_1)}{1+\cosh(\dot{q}_1)}\\ \\ \frac{\operatorname{senh}(\dot{q}_2)}{1+\cosh(\dot{q}_2)}\end{bmatrix} + 9.81 \begin{bmatrix}3.9211\ \operatorname{sen}(q_1) + 0.1862\ \operatorname{sen}(q_1+q_2)\\ 0.1862\ \operatorname{sen}(q_1+q_2)\end{bmatrix}$$

los límites de saturación de este control lo determinan las ganancias proporcional k_{pi} y derivativa k_{vi}, satisfaciendo $k_{pi} \leq 0.8\tau_{max_i}$, con $i = 1, 2$.

El cuadro 10.10 contiene el código fuente con la implementación del modelo dinámico del robot manipulador de 2 gdl y el control saturado (ver programa `robot2gdlsaturado.m`); observe que en la línea 23 se recibe la energía mecánica para mover al robot a la posición deseada $\boldsymbol{q}_d = [\, q_{d1} \quad q_{d2} \,]^T$.

En el archivo `saturado.m` se encuentra la implementación del esquema de control saturado descrito en el cuadro 10.11 y el programa principal para realizar la simulación se describe en el cuadro 10.12 (programa `robot2gdlsaturadosimu.m`); en la línea 20 se realiza el registro del par aplicado a las articulaciones del robot.

Los resultados de simulación se ilustran en la figura 10.7, el perfil de la posición del hombro $q_1(t)$ y del codo $q_2(t)$ presentan un transitorio muy suave, libres de oscilaciones y sobreimpulsos entrando al régimen estacionario sin rizo, ni ruido mecánico. Esto es verificado en la gráfica intermedia correspondiente a los errores de posición de ambas articulaciones $\tilde{q}_1(t)$ y $\tilde{q}_2(t)$. Note como los pares o pares aplicados se encuentran por debajo de los límites de saturación $\tau_1 < \tau_{max_1}$ y $\tau_2 < \tau_{max_2}$. Particularmente el perfil del par τ_1 presenta una pequeña curva descendente en el intervalo $t \in [0.2, 0.5]$ segundos, esto se debe a la acción derivativa que inyecta amortiguamiento para frenar el movimiento del robot, es decir absorve los sobreimpulsos y oscilaciones que pudiera presentar la respuesta del robot; observe que para el intervalo $t \in [0.5, 1]$ segundos la pendiente de dicha curva cambia de signo, ahora en forma ascendente indicando la entrada suave al estado estacionario.

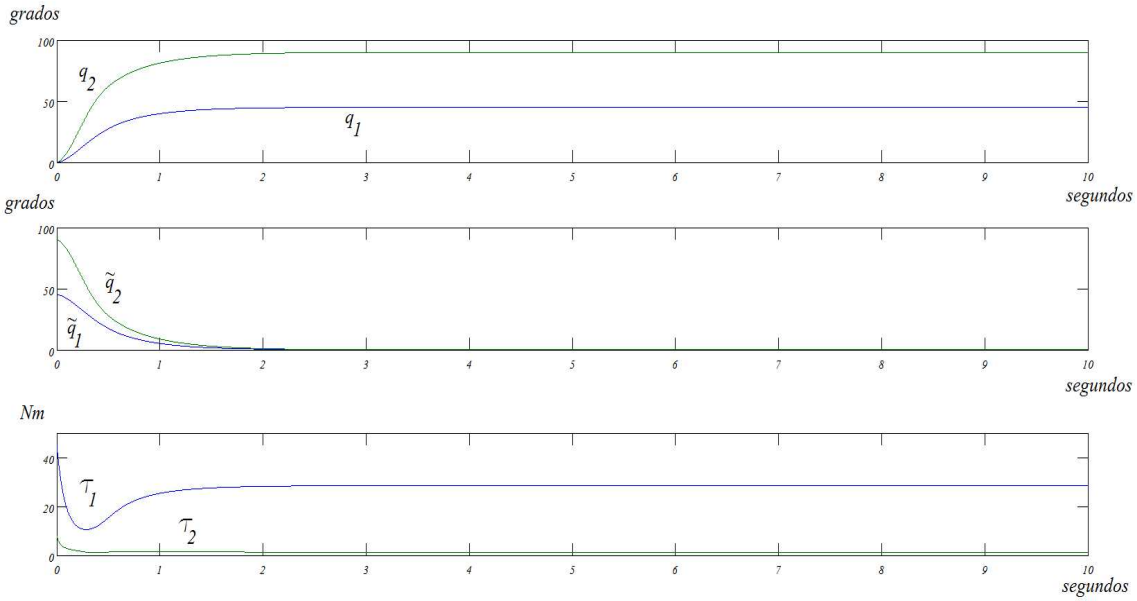


Figura 10.7 Respuesta del robot de 2 gdl (control saturado).

Simulación en MATLAB del robot de 2 gdl y control saturado

Para llevar a cabo la simulación del robot de 2 gdl con el algoritmo de control saturado se requieren 3 archivos: el modelo dinámico del robot de 2 gdl en `robot2gdlsaturado.m`, el algoritmo de control saturado en `saturado.m` y el programa principal `robot2gdlsaturadosimu.m` (**todos los archivos deben estar almacenados en la misma carpeta o directorio del usuario**).

La forma más simple y recomendada para realizar la simulación es a través del editor integrado al entorno de programación de **MATLAB** seleccionando el archivo `robot2gdlsaturadosimu.m` y oprimiendo el icono *play* comando (`run`).

Un forma alternativa para realizar la simulación es escribiendo el nombre del programa principal directamente en la ventana de comandos de **MATLAB** (previo indicar la trayectoria donde se encuentran los 3 archivos), por ejemplo:

```
fx >> cd('C:/robot/control') ↩
fx >> robot2gdlsaturadosimu ↩
```

Código Fuente 10.10 robot2gdlsaturado

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo robot2gdlsaturado.m Versión de MATLAB 2012a

```
%Modelo dinámico numérico del robot manipulador planar de 2 gdl
function xp =robot2gdlsaturado(t,x)
    %vector de estados x = [ q1   q2   q̇1   q̇2 ]^T ∈ IR^4
    q1=x(1);%posición de la articulación del hombro q1
    q2=x(2);%posición de la articulación del codo q2
    q = [q1; q2];%vector de posición articular q̇ = [ q1   q2 ]^T IR^2
    qp1=x(3);%velocidad articular del hombro q̇1
    qp2=x(4);% velocidad articular del codo q̇2
    qp = [qp1; qp2];%vector de velocidad articular q̇ = [ q̇1   q̇2 ]^T ∈ IR^2
    %matriz de inercias M(q) ∈ IR^2×2
    M=[2.351+0.1676*cos(q2), 0.102+0.0838*cos(q2);
        0.102+0.0838*cos(q2), 0.102];
    %matriz de fuerzas centrípetas y de Coriolis C(q,q̇) ∈ IR^2×2
    C=[-0.1676*sin(q2)*qp2, -0.0838*sin(q2)*qp2;
        0.084*sin(q2)*qp1, 0.0];
    %par gravitacional g(q) ∈ IR^2
    gq=9.81*[3.9211*sin(q1)+0.1862*sin(q1+q2);
        0.1862*sin(q1+q2)];
    %fricción viscosa f_f(q̇) = Bq̇ ∈ IR^2
    fr=[2.288*qp1;
        0.175*qp2];
    %algoritmo de control proporcional derivativo
    [tau, ~ ]=saturado(q,qp);%energía proporcionada por el control saturado para mover a q_d
    qpp = M^(-1)*(tau-C*qp-gq-fr);%dinámica del robot y estructura del algoritmo de control
    xp = [qp1; qp2; qpp(1); qpp(2)];%vector de salida ẋ = [ q̇1   q̇2   q̈1   q̈2 ]^T ∈ IR^4
end
```

Código Fuente 10.11 saturado

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo saturado.m Versión de **MATLAB** 2012a

```
1  %Algoritmo de control saturado para un robot de 2 gdl
2  function [tau, qtilde] = saturado(q,qp)
3      q1=q(1);%posición articular del hombro q_1(t)
4      q2=q(2);%posición articular del codo q_2(t)
5      qp1=qp(1);%velocidad articular del hombro q̇_1(t)
6      qp2=qp(2);%velocidad articular del codo q̇_2(t)
7      %par gravitacional articular
8      par_grav=(9.81*[3.9211*sin(q1)+0.1862*sin(q1+q2);
9           0.1862*sin(q1+q2)]);
10     qd=(pi/180)*[45 ;
11          90];%vector de velocidades deseadas q_d = [ q_d1   q_d2 ]^T
12     %límites físicos de los servoamplificadores
13     tau_max1=150;  tau_max2=15;%τ_max1 = 150 Nm y τ_max2 = 15 Nm
14     %regla de sintonía de la ganancia proporcional: k_pi ≤ 0.8τ_max_i, con i = 1, 2; K_p ∈ IR^2×2
15     kp1=0.8*tau_max1 kp2=0.8*tau_max2;
16     Kp=[kp1, 0;
17          0, kp2];
18     %regla de sintonía de la ganancia derivativa: k_vi = 0.5k_pi, con i = 1, 2; K_v ∈ IR^2×2
19     kv1=0.5*kp1; kv2=0.5*kp2;
20     Kv=[kv1, 0;
21          0, kv2];
22     qtilde=qd-q;%error de posición q̃ = q_d − q
23     %ley de control: τ = K_p senh(q̃)/(1+cosh(q̃)) − K_v senh(q̇)/(1+cosh(q̇)) + g(q)
24     tau=Kp*sinh(qtilde)./(1+cosh(qtilde))-Kv*sinh(qp)./(1+cosh(qp))+par_grav;
25 end
```

Código Fuente 10.12 robot2gdlsaturadosimu

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo robot2gdlsaturadosimu.m Versión de MATLAB 2012a

```
%Programa principal para la simulación del robot manipulador de 2 gdl y control saturado
clc;
clear all;
close all;
format short g
ti=0;%tiempo inicial de simulación
h=0.001;%incremento de un milisegundo el tiempo t = t + h
tf = 10;%tiempo final de simulación
t=ti:h:tf;%vector tiempo
ci=[0; 0; 0; 0];%condiciones iniciales [q1(0) q2(0) q̇1(0) q̇2(0)]^T = [0 0 0 0]^T
opciones=odeset('RelTol',h*1e-3,'InitialStep', h,'MaxStep',h);
%solución numérica del modelo dinámico del robot manipulador de 2 gdl y control PD
[t, x]=ode45('robot2gdlsaturado',t,ci,opciones);
q1=x(:,1); q2=x(:,2);%se asignan posiciones articulares
qp1=x(:,3); qp2=x(:,4);% velocidades articulares
[n, m]=size(t);%obtiene el número de renglones n y columnas m para las variables auxiliares q̃ y τ
tau1=zeros(n,1); tau2=zeros(n,1);%variables auxiliares para el registro del par aplicado τ
qtilde1=zeros(n,1); qtilde2=zeros(n,1);%variables auxiliares para el registro del error de posición q̃
for k=1:n
    [tau, qtilde] = saturado([q1(k);q2(k)], [qp1(k); qp2(k)]);%registra error de posición q̃ y par τ
    tau1(k)=tau(1); tau2(k)=tau(2);%registra par o torque τ
    qtilde1(k)=qtilde(1); qtilde2(k)=qtilde(2);% registra error de posición q̃
end
subplot(3,1,1); plot( t, 180*q1/pi, t,180*q2/pi)%grafica posiciones (q1, q2) en grados
subplot(3,1,2); plot(t, 180*qtilde1/pi, t, 180*qtilde2/pi)%grafica errores de posición en grados (q̃1, q̃2)
subplot(3,1,3); plot( t, tau1, t,tau2)% grafica los torques o pares aplicados (τ1, τ2)
```

10.1.3 Control de un robot de 3 gdl

Otra estrategia de control para un robot de 3 gdl es la de un algoritmo de control saturado con estructura de arcotangente como se muestra a continuación:

$$\boldsymbol{\tau} = K_p \operatorname{atan}(\tilde{\boldsymbol{q}}) - K_v \operatorname{atan}(\dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q})$$

$$\begin{bmatrix}\tau_1\\ \tau_2\\ \tau_3\end{bmatrix} = \begin{bmatrix}k_{p1} & 0 & 0\\ 0 & k_{p2} & 0\\ 0 & 0 & k_{p3}\end{bmatrix}\begin{bmatrix}\operatorname{atan}(\tilde{q}_1)\\ \operatorname{atan}(\tilde{q}_2)\\ \operatorname{atan}(\tilde{q}_3)\end{bmatrix} - \begin{bmatrix}k_{v1} & 0 & 0\\ 0 & k_{v2} & 0\\ 0 & 0 & k_{v3}\end{bmatrix}\begin{bmatrix}\operatorname{atan}(\dot{q}_1)\\ \operatorname{atan}(\dot{q}_2)\\ \operatorname{atan}(\dot{q}_3)\end{bmatrix} + 9.81\begin{bmatrix}0\\ 3.9211\operatorname{sen}(q_1) + 0.1862\operatorname{sen}(q_1+q_2)\\ 0.1862\operatorname{sen}(q_1+q_2)\end{bmatrix}$$

Figura 10.8 Robot manipulador FANUC.

El modelo dinámico de un robot manipulador de 3 gdl se presenta a continuación:

$$\boldsymbol{\tau} = M(\boldsymbol{q}) + C(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) + B\dot{\boldsymbol{q}}$$

$$\begin{bmatrix}\tau_1\\ \tau_2\\ \tau_3\end{bmatrix} = \begin{bmatrix}m_{11} & m_{12} & m_{13}\\ m_{21} & m_{22} & m_{23}\\ m_{31} & m_{32} & m_{33}\end{bmatrix}\begin{bmatrix}\ddot{q}_1\\ \ddot{q}_2\\ \ddot{q}_3\end{bmatrix} + \begin{bmatrix}c_{11} & c_{12} & c_{13}\\ c_{21} & c_{22} & c_{23}\\ c_{31} & c_{32} & c_{33}\end{bmatrix}\begin{bmatrix}\dot{q}_1\\ \dot{q}_2\\ \dot{q}_3\end{bmatrix} + \begin{bmatrix}b_1 & 0 & 0\\ 0 & b_2 & 0\\ 060 & b_3 & \end{bmatrix}\begin{bmatrix}\dot{q}_1\\ \dot{q}_2\\ \dot{q}_3\end{bmatrix}$$

$$+g\begin{bmatrix}0\\ l_{c2}m_2\operatorname{sen}(q_2) + m_3l_3\operatorname{sen}(q_2) + m_3l_{c3}\operatorname{sen}(q_2+q_3)\\ m_3l_{c3}\operatorname{sen}(q_2+q_3)\end{bmatrix}$$

La utilidad del modelo dinámico resulta importante cuando se dispone de los valores numéricos de todos los parámetros del robot; bajo este enfoque la tabla 10.4 muestra la terminología, significado y valores de un robot prototipo de 3 gdl con motores de transmisión directa.

Tabla 10.4 Parámetros del robot antropomórfico de 3 gdl

Eslabón	Significado	Notación	Valor
Base	Masa del eslabón 1	m_1	26.9 kg
	Longitud del eslabón 1	l_1	0.45 m
	Inercia del eslabón 1	I_{z1}	1.266 Nm-seg^2/rad
	Inercia del eslabón 1	I_{y1}	0.089 Nm-seg^2/rad
	Inercia del eslabón 1	I_{x1}	0.03 Nm-seg^2/rad
	Centro de masa del eslabón 1	l_{c1}	0.091 m
Hombro	Coeficiente de fricción viscosa	b_1	2.288 Nm-seg/rad
	Masa del eslabón 2	m_2	30 kg
	Longitud del eslabón 2	l_2	0.45 m
	Inercia del eslabón 2	I_{z2}	0.084 Nm-seg^2/rad
	Inercia del eslabón 2	I_{y2}	0.003 Nm-seg^2/rad
	Inercia del eslabón 2	I_{x2}	0.05 Nm-seg^2/rad
	Centro de masa del eslabón 2	l_{c2}	0.038 m
	Coeficiente de fricción viscosa	b_2	0.2 Nm-seg/rad
Codo	Masa del eslabón 3	m_3	3.88
	Longitud del eslabón 3	l_3	0.45 m
	Inercia del eslabón 3	I_{z3}	0.056 Nm-seg^2/rad
	Inercia del eslabón 3	I_{y3}	0.0012 Nm-seg^2/rad
	Inercia del eslabón 3	I_{x3}	0.009 Nm-seg^2/rad
	Centro de masa del eslabón 3	l_{c3}	0.048
	Coeficiente de fricción viscosa	b_3	0.175 Nm-seg/rad
	Aceleración debida a la gravedad	g	9.81 m/seg^2

Para propósitos prácticos considere un un robot manipulador de 3 gdl de transmisión directa cuyos límites físicos de cada articulación se muestran en la tabla 10.5.

Tabla 10.5 Límites físicos de los servomotores del robot de 3 gdl.

Articulación	Par máximo τ_{max}
Base	$\tau_{max_1} = 50$ Nm
Hombro	$\tau_{max_2} = 150$ Nm
Codo	$\tau_{max_3} = 15$ Nm

Los cuadros de código fuente **MATLAB** 10.14, 10.14, 10.15 muestran los programas del modelo dinámico del robot de 3 gdl `saturado3gdl.m`, control arcotangente `saturado3gdl.m` y programa principal `robot3gdlsimu`, respectivamente. La figura 10.9 muestra los resultados de simulación.

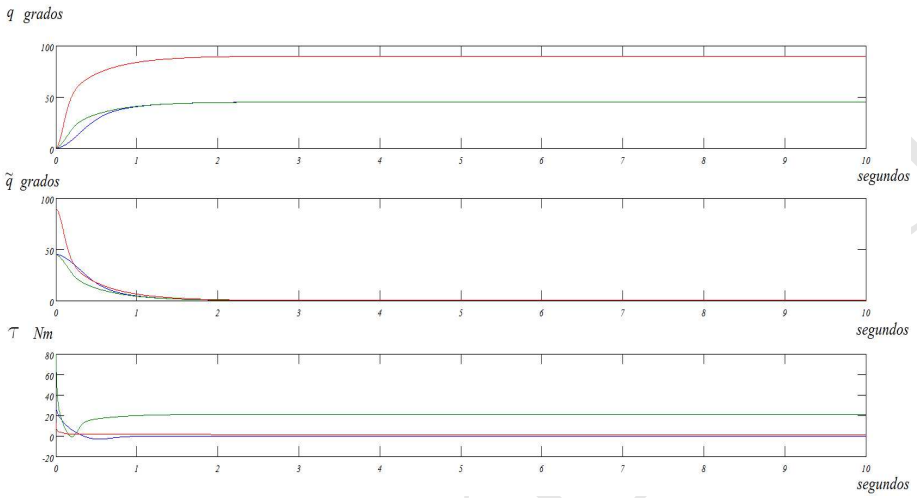


Figura 10.9 Respuesta del robot 3 gdl (control saturado).

Simulación en MATLAB del robot de 3 gdl y control saturado

Para llevar a cabo la simulación del robot de 3 gdl con el algoritmo de control saturado se requieren 3 archivos: el modelo dinámico del robot de 3 gdl en `robot3gdl.m`, el algoritmo de control saturado en `saturado3gdl.m` y el programa principal `robot3gdlsimu.m` (**todos los archivos deben estar almacenados en la misma carpeta o directorio del usuario**).

La forma más simple y recomendada para realizar la simulación es a través del editor integrado al entorno de programación de **MATLAB** seleccionando el archivo `robot3gdlsimu.m` y oprimiendo el icono *play* comando (`run`).

Un forma alternativa para realizar la simulación es escribiendo el nombre del programa principal directamente en la ventana de comandos de **MATLAB** (previo indicar la trayectoria (`path`) de la unidad del disco duro donde el usuario tiene los 3 archivos), por ejemplo:

```
fx >> cd('C:/robot/control')%la trayectoria es definida por el usuario ↩
fx >> robot3gdlsimu ↩
```

$$
\begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} =
\underbrace{\begin{bmatrix}
I_{y2}\,\mathrm{sen}^2(q_2) + I_{y3}\,\mathrm{sen}^2(q_2+q_3) + I_{z1} + I_{z2}\cos^2(q_2) + I_{z3}\cos^2(q_2+q_3) + m_2 l_{c2}^2\cos^2(q_2) + m_3\left[l_2\cos(q_2) + l_{c3}\cos(q_2+q_3)\right]^2 & 0 & 0 \\
0 & I_{x2} + I_{x3} + m_3 l_2^2 + m_2 l_{c2}^2 + m_3 l_{c3}^2 + 2 m_3 l_2 l_{c3}\cos(q_3) & I_{x3} + m_3 l_{c3}^2 + m_3 l_2 l_{c3}\cos(q_3) \\
0 & I_{x3} + m_3 l_{c3}^2 + m_3 l_2 l_{c3}\cos(q_3) & I_{x3} + m_3 l_{c3}^2
\end{bmatrix}}_{M(\boldsymbol{q})}
\begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \end{bmatrix}
$$

$$
+ \underbrace{\begin{bmatrix}
\left[I_{y2} - I_{z2} - m_2 l_{c2}^2\right]\cos(q_2)\,\mathrm{sen}(q_2)\dot{q}_1 - \left[I_{y3} - I_{z3}\right]\cos(q_2+q_3)\,\mathrm{sen}(q_2+q_3)\dot{q}_1 - m_3\left[l_2\cos(q_2) + l_{c3}\cos(q_2+q_3)\right]l_{c3}\,\mathrm{sen}(q_2+q_3)\dot{q}_2 - m_3 l_{c3}\,\mathrm{sen}(q_2+q_3)\dot{q}_3 & \left[I_{y2} - I_{z2} - m_2 l_{c2}^2\right]\cos(q_2)\,\mathrm{sen}(q_2)\dot{q}_2 - \left[I_{y3} - I_{z3}\right]\cos(q_2+q_3)\,\mathrm{sen}(q_2+q_3)\dot{q}_2 - m_3\left[l_2\cos(q_2) + l_{c3}\cos(q_2+q_3)\right]l_{c3}\,\mathrm{sen}(q_2+q_3)\dot{q}_3 - m_3 l_2 l_{c3}\,\mathrm{sen}(q_3)\dot{q}_3 & \left[I_{y3} - I_{z3}\right]\cos(q_2+q_3)\,\mathrm{sen}(q_2+q_3)\dot{q}_1 - m_3 l_2 l_{c3}\,\mathrm{sen}(q_3)\dot{q}_2 - m_3 l_2 l_{c3}\,\mathrm{sen}(q_3)\dot{q}_3 \\
\left[I_{y2} - I_{z2} - m_2 l_{c2}^2\right]\cos(q_2)\,\mathrm{sen}(q_2)\dot{q}_1 + \left[I_{y3} - I_{z3}\right]\cos(q_2+q_3)\,\mathrm{sen}(q_2+q_3)\dot{q}_1 + m_3\left[l_2\cos(q_2) + l_{c3}\cos(q_2+q_3)\right]l_{c3}\,\mathrm{sen}(q_2+q_3)\dot{q}_2 + \left[I_{y3} - I_{z3}\right]\cos(q_2+q_3)\,\mathrm{sen}(q_2+q_3)\dot{q}_3 + m_3 l_{c3}\,\mathrm{sen}(q_2+q_3)l_2\,\mathrm{sen}(q_2)\dot{q}_3 - m_3 l_{c3}\,\mathrm{sen}(q_2+q_3)l_{c3}\,\mathrm{sen}(q_2+q_3)\dot{q}_3 & 0 & 0 \\
\left[I_{z2} - I_{y2} + m_2 l_{c1}^2\right]\cos(q_2)\,\mathrm{sen}(q_2)\dot{q}_1 + \left[I_{z3} - I_{y3}\right]\cos(q_2+q_3)\,\mathrm{sen}(q_2+q_3)\dot{q}_1 + m_3\left[l_1\cos(q_2) + l_{c2}\cos(q_2+q_3)\right]l_1\,\mathrm{sen}(q_2)\dot{q}_1 + m_3\left[l_1\cos(q_2) + l_{c2}\cos(q_2+q_3)\right]l_{c2}\,\mathrm{sen}(q_2+q_3)\dot{q}_1 & 0 & 0 \\
\left[I_{z3} - I_{y3}\right]\cos(q_2+q_3)\,\mathrm{sen}(q_2+q_3)\dot{q}_1 + m_3 l_{c2}\,\mathrm{sen}(q_2+q_3)\left[l_1\cos(q_2) + l_{c2}\cos(q_2+q_3)\right]\dot{q}_1 & 0 & 0
\end{bmatrix}}_{C(\boldsymbol{q},\dot{\boldsymbol{q}})}
\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix}
$$

$$
+ \underbrace{\begin{bmatrix}
\left[I_{y2} - I_{z2} - m_2 l_{c2}^2\right]\cos(q_2)\,\mathrm{sen}(q_2)\dot{q}_2 - \left[I_{y3} - I_{z3}\right]\cos(q_2+q_3)\,\mathrm{sen}(q_2+q_3)\dot{q}_2 - m_3\left[l_2\cos(q_2) + l_{c3}\cos(q_2+q_3)\right]l_{c3}\,\mathrm{sen}(q_2+q_3)\dot{q}_2 - m_3 l_{c3}\,\mathrm{sen}(q_2+q_3)l_2\,\mathrm{sen}(q_2)\dot{q}_3 - m_3 l_{c3}\,\mathrm{sen}(q_2+q_3)l_{c3}\,\mathrm{sen}(q_2+q_3)\dot{q}_3 & \left[I_{y3} - I_{z3}\right]\cos(q_2+q_3)\,\mathrm{sen}(q_2+q_3)\dot{q}_1 - m_3 l_2 l_{c3}\,\mathrm{sen}(q_3)\dot{q}_2 - m_3 l_2 l_{c3}\,\mathrm{sen}(q_3)\dot{q}_3 & -m_3 l_{c2}\,\mathrm{sen}(q_2+q_3)\,l_{c1}\cos(q_2)\dot{q}_1 - m_3 l_{c2}^2\,\mathrm{sen}(q_2+q_3)\cos(q_2+q_3)\dot{q}_1 \\
\left[I_{y2} - I_{z2} - m_2 l_{c2}^2\right]\cos(q_2)\,\mathrm{sen}(q_2)\dot{q}_1 + \left[I_{y3} - I_{z3}\right]\cos(q_2+q_3)\,\mathrm{sen}(q_2+q_3)\dot{q}_1 + m_3\left[l_2\cos(q_2) + l_{c3}\cos(q_2+q_3)\right]l_2\,\mathrm{sen}(q_2)\dot{q}_1 - m_3\left[l_2\cos(q_2) + l_{c3}\cos(q_2+q_3)\right]l_{c2}\,\mathrm{sen}(q_2+q_3)\dot{q}_1 & -l_1 m_3 l_{c2}\,\mathrm{sen}(q_3)\dot{q}_3 & -l_1 m_3 l_{c2}\,\mathrm{sen}(q_3)(\dot{q}_2+\dot{q}_3) \\
l_1 m_3 l_{c2}\,\mathrm{sen}(q_3)\dot{q}_2 & 0 & 0
\end{bmatrix}}_{C(\boldsymbol{q},\dot{\boldsymbol{q}})}
\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix}
$$

$$
+ \underbrace{\begin{bmatrix} 0 \\ m_2 l_{c2} g\,\mathrm{sen}(q_2) + m_3 l_3 g\,\mathrm{sen}(q_2) + m_3 l_{c3} g\,\mathrm{sen}(q_2+q_3) \\ m_3 l_{c3} g\,\mathrm{sen}(q_2+q_3) \end{bmatrix}}_{\boldsymbol{g}(\boldsymbol{q})}
+ \underbrace{\begin{bmatrix} b_1\dot{q}_1 \\ b_2\dot{q}_2 \\ b_3\dot{q}_3 \end{bmatrix}}_{\boldsymbol{f}_f(\dot{\boldsymbol{q}},\boldsymbol{f}_e)}
$$

Código Fuente 10.13 robot3gdl

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo robot3gdl.m Versión de MATLAB 2012a

```
%Modelo dinámico numérico del robot manipulador de 3 gdl
function xp = robot3gdl(t,x)
    q1=x(1); q2=x(2); q3=x(3); q = [q1; q2; q3];%vector de posición articular q
    qp1=x(4); qp2=x(5); qp3=x(6); qp=[qp1; qp2; qp3];%vector de velocidad articular q̇
    %parámetros del robot antropomórfico de 3 gdl
    Iz1=1.26; Iz2=0.084; Iz3=0.056; Iy1=0.089; Iy2=0.003; Iy3=0.0012; Ix1=0.03; Ix2=0.05; Ix3=0.009; g=9.81;
    m1=26.902; l1=0.45; b1=2.288; m2=30; l2=0.45; lc2=0.038; b2=0.2; m3=3.880; l3=0.45; lc3=0.048; b3=0.175;
    m11=Iy2*sin(q2)*sin(q2)+Iy3*sin(q2+q3)*sin(q2+q3)+Iz1+Iz2*cos(q2)*cos(q2)+ Iz3*cos(q2+q3)*cos(q2+q3)+...
    m2*lc2*lc2*cos(q2)*cos(q2)+m3*(l2*cos(q2)+lc3*cos(q2+q3))*(l2*cos(q2)+lc3*cos(q2+q3));
    m12=0; m13=0; m21=0; m22=Ix2+Ix3+m3*l2*l2+m2*lc2*lc2+m3*lc3*lc3+ 2*m3*l2*lc3*cos(q3);
    m23=Ix3+m3*lc3*lc3+m3*l2*lc3*cos(q3); m31=0; m32=Ix3+m3*lc3*lc3+m3*l2*lc3*cos(q3); m33=Ix3+m3*lc3*lc3;
    M=[m11, m12, m13; m21, m22, m23; m31, m32, m33];%matriz de inercia M(q)
    gamma112=(Iy2-Ix2-m2*lc2*lc2)*cos(q2)*sin(q2)+(Iy3-Iz3)*cos(q2+q3)*sin(q2+q3)...
                    -m3*(l2*cos(q2)+lc3*cos(q2+q3))*(l2*sin(q2)+lc3*sin(q2+q3));
    gamma113=(Iy3-Iz3)*cos(q2+q3)*sin(q2+q3)-m3*lc3*sin(q2+q3)*(l2*cos(q2)+lc3*cos(q2+q3));
    gamma121=(Iy2-Iz2-m2*lc2*lc2)*cos(q2)*sin(q2)+(Iy3-Iz3)*cos(q2+q3)*sin(q2+q3)...
                    -m3*(l2*cos(q2)+lc3*cos(q2+q3))*(l2*sin(q2)+lc3*sin(q2+q3));
    gamma131=(Iy3-Iz3)*cos(q2+q3)*sin(q2+q3)-m3*lc3*sin(q2+q3)*(l2*cos(q2)+lc3*cos(q2+q3));
    gamma211=(Ix2-Iy2+m2*lc2*lc2)*cos(q2)*sin(q2)+
    (Iz3-Iy3)*cos(q2+q3)*sin(q2+q3)+m3*(l2*cos(q2)+lc3*cos(q2+q3))* (l2*sin(q2)+lc3*sin(q2+q3));
    gamma223=-l2*m3*lc3*sin(q3); gamma232=-l2*m3*lc3*sin(q3); gamma233=-l2*m3*lc3*sin(q3);
    gamma311=(Iz3-Iy3)*cos(q2+q3)*sin(q2+q3)+m3*lc3*sin(q2+q3)*(l2*cos(q2)+lc3*cos(q2+q3));
    gamma322=l2*m3*lc3*sin(q3);
    c11=gamma112*qp2+gamma113*qp3; c12=gamma121*qp1; c13=gamma131*qp1;
    c21=gamma211*qp1; c22=gamma223*qp3; c23=gamma232*qp2+gamma233*qp3;
    c31=gamma311*qp1; c32=gamma322*qp2;  c33=0;
    C=[c11, c12, c13; c21, c22, c23; c31, c32, c33];%matriz de Coriolis C(q, q̇)
    gq11=0; gq21=(lc2*m2++m3*l3)*sin(q2)+m3*lc3*sin(q2+q3); gq31=m3*lc3*sin(q2+q3);
    gq=g*[gq11; gq21; gq31];%par gravitacional
    fr=[b1*qp1; b2*qp2; b3*qp3];%par de fricción
    [tau, ~ ]=saturado3gdl(q,qp);
    qpp = inv(M)*(tau-C*qp-gq-fr);%aceleración articular
    xp = [qp1; qp2; qp3; qpp(1); qpp(2); qpp(3)];%vector de salida
end
```

Código Fuente 10.14 saturado3gdl

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo saturado3gdl.m Versión de MATLAB 2012a

```
1  %Control saturado del robot manipulador de 3 gdl
2  function [tau, qt] = saturado3gdl(q,qp)
3      q1=q(1);%posición articular de la base q1(t)
4      q2=q(2);%posición articular del hombro q2(t)
5      q3=q(3);%posición articular del codo q3(t)
6      qp1=qp(1);%velocidad articular de la base q̇1(t)
7      qp2=qp(2);%velocidad articular de la base q̇2(t)
8      qp3=qp(3);%velocidad articular de la base q̇3(t)
9      %parámetros del vector de par gravitacional
10     m1=26.902; l1=0.45; m2=30; l2=0.45; lc2=0.038; m3=3.880; l3=0.45; lc3=0.048; g=9.81;
11     %par gravitacional articular
12     par_grav=g*[0; (lc2*m2++m3*l3)*sin(q2)+m3*lc3*sin(q2+q3); m3*lc3*sin(q2+q3)];
13     %vector de posición deseada
14     qd=[45*pi/180; 45*pi/180; 90*pi/180];% q_d = [q_d1  q_d2  q_d3]^T
15     %vector error de posición
16     qt=[qd(1)-q1;qd(2)-q2; qd(3)-q3];% q̃ = q_d - q
17     %límites físicos de los servoamplificadores τ_max1 = 50 Nm, τ_max2 = 150 Nm, τ_max3 = 15 Nm
18     tau_max1=50; tau_max2=150; tau_max3=15;
19     %sintonía de las ganancia proporcional k_pi ≤ 0.8τ_max_i con i = 1, 2, 3
20     kp1=0.8*tau_max1; kp2=0.8*tau_max2; kp3=0.8*tau_max3;
21     Kp=[kp1, 0,0;0, kp2,0; 0,0,kp3];% K_p ∈ IR^3×3
22     %sintonía de las ganancia derivativa k_vi = 0.5k_pi para i = 1, 2, 3
23     kv1=0.5*kp1; kv2=0.5*kp2; kv3=0.5*kp3;
24     Kv=[kv1, 0,0;0, kv2,0;0,0,kv3];% K_v ∈ IR^3×3
25     tau=Kp*atan(qt)-Kv*atan(qp)+par_grav;%τ = K_p atan(q̃) - K_v atan(q̇) + g(q)
26 end
```

Código Fuente 10.15 robot3gdlsimu

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo robot3gdlsimu.m Versión de MATLAB 2012a

```
%Programa principal para la simulación del robot manipulador de 3 gdl y control saturado
clc;
clear;
close all;
format short g
%parámetros de simulación:
ti=0; h=0.0025; tf = 10;%tiempo de simulación 0 a 10 segundos
ts=ti:h:tf;%vector tiempo
%configuración de la función ode45(...)
opciones=odeset('RelTol',1e-06,'InitialStep', h, 'MaxStep', h);
ci=[0; 0; 0; 0; 0; 0];%condiciones iniciales [q1(0) q20, q3(0), q̇1(0), q̇2(0), q̇3(0)]^T = [0, 0, 0, 0, 0, 0]^T
%solución numérica del sistema, con condición iniciales cero
[t, x]=ode45('robot3gdl',ts,ci, opciones);
q1=x(:,1); q2=x(:,2); q3=x(:,3); qp1=x(:,4); qp2=x(:,5); qp3=x(:,6);
[n,m]=size(t);
tau1=zeros(n,1); tau2=zeros(n,1); tau3=zeros(n,1);
qtilde1=zeros(n,1); qtilde2=zeros(n,1); qtilde3=zeros(n,1);
for  k=1:n
    [tau, qtilde] = saturado3gdl([q1(k);q2(k); q3(k)], [qp1(k); qp2(k);qp3(k)]);
    tau1(k)=tau(1); tau2(k)=tau(2); tau3(k)=tau(3);
    qtilde1(k)=qtilde(1); qtilde2(k)=qtilde(2); qtilde3(k)=qtilde(3);
end
%gráficas de posiciones q, errores de posición q̃ y torques τ
subplot(3,1,1); plot( t, 180*q1/pi, t,180*q2/pi, t,180*q3/pi);%posiciones articulares q
subplot(3,1,2); plot(t, 180*qtilde1/pi, t, 180*qtilde2/pi, t, 180*qtilde3/pi)%velocidades articulares q̃
subplot(3,1,3); plot( t, tau1, t,tau2, t, tau3) %pares aplicados τ
```

10.2 Herramienta de MATLAB pidtool

El entorno de programación **MATLAB** tiene una herramienta gráfica denominada `pidtool` con ambiente propio de programación interactiva para realizar simulación de sistemas lineales con el algoritmo de control proporcional integral derivativo PID bajo la siguiente sintaxis:.

```
pidtool(sys,'tipo')
```

Realiza la simulación del sistema lineal `sys` con el control PID, `sys` puede ser sistema representado por un modelo en espacio de estados $\dot{\boldsymbol{x}} = A\boldsymbol{x} + Bu$; $y = \boldsymbol{c}^T\boldsymbol{x}$ o una función de transferencia $G(s) = \frac{\text{num}(s)}{\text{den}(s)}$.

La función `pidtool` toma en cuenta la configuración que se muestra en el diagrama a bloques de la figura 10.10.

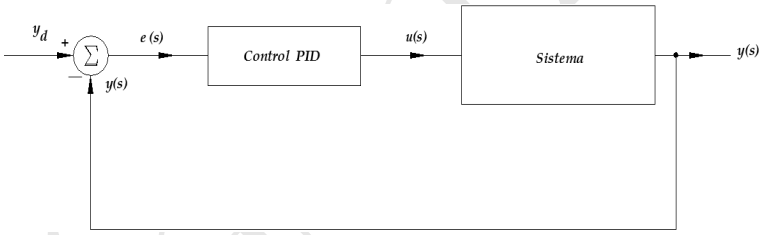


Figura 10.10 Diagrama a bloques del control PID en la función `pidtool`.

El argumento de entrada '`tipo`' es una cadena de caracteres que determina la forma de configurar a la función `pidtool(...)`, es decir indica las opciones para trabajar con el algoritmo de control PID en sus diferentes variantes (ver tabla 10.6) como puede ser: proporcional P, proporcional derivativo PD, proporcional integral PI, proporcional integral derivativo PID; inclusive trabajar como derivador lo cual se consigue anulando la ganancias proporcional e integral o como integrador ($K_p = 0$ y K_v=0).

Una forma de sintonizar las ganancias proporcional K_p, integral K_i y derivativa K_v es por medio de características del transitorio (por ejemplo, tiempo de subida t_s) y aspectos deseables del comportamiento en estado estacionario (estabilidad), estos cirterios son tomados en cuenta por la función `pidtune(...)` con la siguiente sintaxis:

```
[pid]= pidtune(sys,'tipo')
```

Tabla 10.6 Opciones de la función `pidtool`.

Configuración	Características de control	Control continuo	Control discreto
'p'	Control proporcional	K_p	K_p
'i'	Acción de control integral	$\frac{K_i}{s}$	$K_i\frac{h}{z-1}$
'pi'	Control proporcional-integral	$K_p + \frac{K_i}{s}$	$K_p + K_i\frac{h}{z-1}$
'pd'	Control proporcional-derivativo	$K_p + K_v s$	$K_p + K_v\frac{z-1}{h}$
'pd'	Control proporcional-integral-derivativo	$K_p + \frac{K_i}{s} + K_v s$	$K_p + K_v\frac{h}{z-1} + K_d\frac{z-1}{h}$

Con al finalidad de obtener alto desempeño en la respuesta del sistema con el esquema de control PID, la sintonía se realiza en base a las características del sistema `sys` de tal forma que el régimen transitorio sea rápido, con un mínimo de sobreimpulsos y oscilaciones y tener fase estacionaria libre de rizo u oscilaciones. El argumento de entrada '`tipo`' es una cadena de caracteres que especifica las características del PID como se muestra en la tabla 10.6. La función `pidtune(...)` retorna la sintonía de las ganancias en la variable `pid`.

Cabe mencionar que la función `pidtune(...)` ya está autocontenida en la función `pidtool(...)`, por lo tanto, la utilidad de `pidtune(...)` consiste en sintonizar las ganancias del control PID para ser utilizadas con otras funciones dentro de un programa con código fuente **Matlab** por ejemplo, con la función `feedback(...)` y función `step(...)` tal y como se ilustra en el siguiente ejercicio 10.1.

♣ Ejemplo 10.1

Considere la planta de segundo orden:

$$G_s(s) = \frac{w_n^2}{s^2 + 2\rho w_n s + w_n^2}$$

donde $w_n = 1$, ρ=1. Controlar la salida de la planta $y(t)$ para una entrada escalón unitario usando el algoritmo PID. Realice un programa en **Matlab** para simular el proceso por medio de la funciones `pidtune(...)` y `pidtool(...)`.

Solución

La forma de realizar la simulación de la planta y el algoritmo de control se encuentra en el programa `cap10_pidtool.m` descrito en el cuadro 10.16.

En la línea 5 se define la función de transferencia de la planta o sistema $G_s(s) = \frac{w_n^2}{s^2+2\rho w_n s+w_n^2}$, la sintonía de las ganancias proporcional K_p, integral K_i y derivativa K_v se realiza con la función `pidtune` (ver línea 6), cuyo resultado se muestra en la ventana de comandos:

```
pid=
```

$$K_p + K_i\frac{1}{s} + K_d s$$

$$K_p = 1.92, \quad K_i = 1.3, \quad K_d = 0.258$$

Continuous-time PID controller in parallel form.

La función `pidtune(...)` retorna la estructura de control PID con el valor numérico de las ganancias. La función de transferencia que relaciona el lazo de control desde la entrada de referencia deseada $y_d(s)$ hasta la salida $y(s)$ es: $G_1(s) = \frac{y(s)}{y_d(s)} = \frac{G_{PID}(s)G_s(s)}{1+G_{PID}(s)G_s(s)}$, la cual se calcula en la línea 7 con la función `G1= feedback(pid*G,1);` la simulación en tiempo continuo de la planta y algoritmo PID se realiza en la línea 8 generando la figura 10.11. En la línea 9 se realiza el mismo proceso de simulación con la función `pidtool(G1,'pid')` obteniendo la figura 10.12. Observe que la interface gráfica de `pidtool(...)` describe la sintonía de las ganancias del controlador, así como información importante sobre la fase transitoria de la respuesta de la planta bajo el control PID y características de la etapa estacionaria.

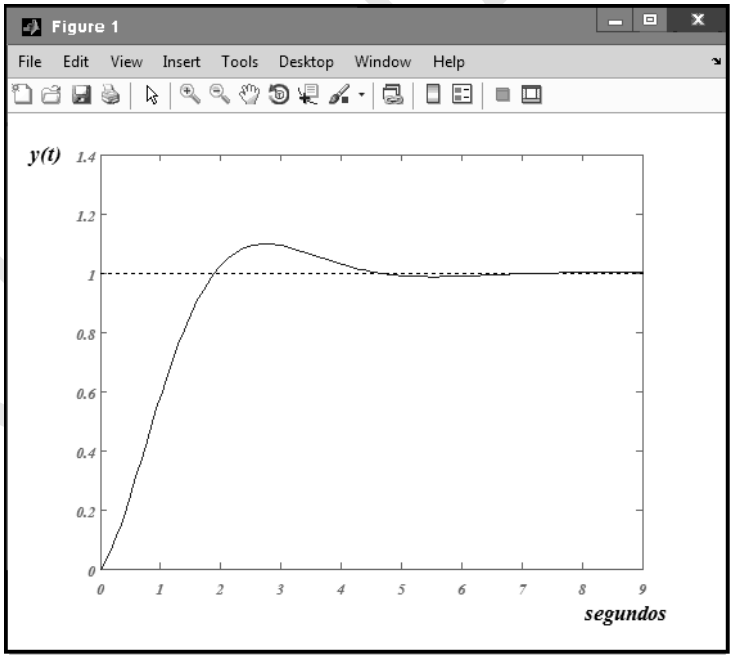


Figura 10.11 Respuesta de la planta de segundo orden con el esquema PID.

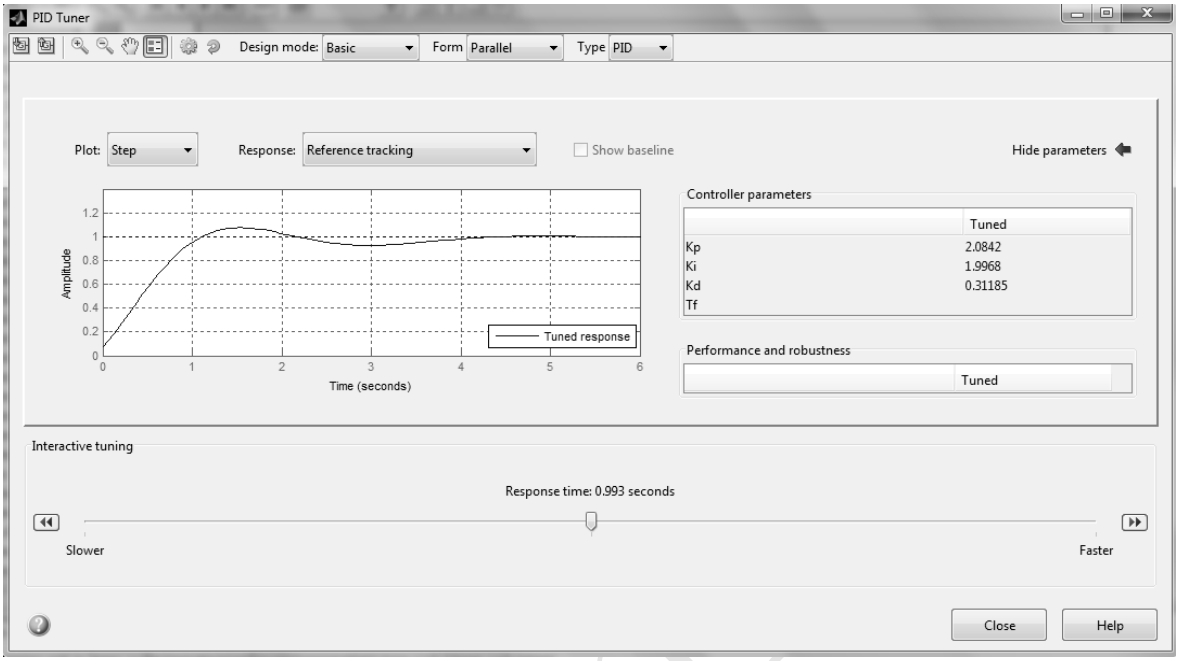


Figura 10.12 Interface gráfica de la función `pidtool`.

Código Fuente 10.16 cap10_pidtool

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo cap10_pidtool.m Versión de MATLAB 2012a

```
1 clc; clear all;  close all; format short
2 wn=1; rho=1;% parámetros de la planta de segundo orden
3 num=wn*wn;% numerador
4 den=[1, 2*rho*wn, wn*wn];% denominador
5 G = tf(num,den);  % Función de transferencia: G(s) = num(s)/den(s) = wn^2/(s^2+2*rho*wn*s+wn^2)
6 pid = pidtune(G,'pid') % sintónia de las ganancias del control PID: Kp, Ki, Kv
7 G1= feedback(pid*G,1);% retorna G1(s) = G_PID(s)G_s(s)/(1+G_PID(s)G_s(s))
8 step(G1) % simulación de la planta y control PID en tiempo continuo
9 pidtool(G1,'pid') %ambiente de programación pidtool(...)
```

10.3 Simulación con servomecanismos

El tema de simulación de control de servomecanismos se ilustra en esta sección por medio de dos prototipos: sistema masa resorte y centrífuga. Asímismo, también se presenten varias esquemas de control.

10.3.1 Sistema masa resorte

Considere un sistema masa resorte amortiguador con movimiento horizontal como se ilustra en la figura 10.13. El movimiento del sistema se realiza sobre el eje x; no se considera movimiento sobre los ejes y, ni z; la fuerza aplicada f es el algoritmo de control proporcional derivativo en dirección positiva del eje x, por lo que el movimiento del bloque rígido m y la elongación del resorte de rigidez k se lleva a cabo exclusivamente sobre el eje x.

El origen del sistema de referencia cartesiano $\Sigma(x, y, z)$ se encuentra como se ilustra en la figura 10.13, por lo que $x = z = 0$ m y el sistema masa resorte se encuentra colocado una altura constante sobre el eje y denota por y^*.

El modelo dinámico del sistema masa resorte en configuración horizontal se encuentra descrito por la siguiente ecuación:

$$f = m\ddot{x} + kx + b\dot{x}$$

Control PD

Considere el algoritmo de control proporcional derivativo PD:

$$f = k_p\tilde{x} - k_v\dot{x}$$

donde $k_p, k_v \in \mathbb{R}_+$ son las ganancias proporcional y derivativa, el error de posición $\tilde{x}$ se define como la diferencia entre la posición deseada x_d y la posición actual del sistema masa resorte: $\tilde{x} = x_d - x$ y f es la fuerza aplicada. En combinación con el modelo dinámico del sistema masa resorte se obtiene la siguiente ecuación en lazo cerrado:

$$\frac{d}{dt}\begin{bmatrix}\tilde{x}\\ \dot{x}\end{bmatrix} = \begin{bmatrix}-\dot{x}\\ \frac{1}{m}\left[k_p\tilde{x} - k_v\dot{x} - b\dot{x} - kx\right]\end{bmatrix}$$

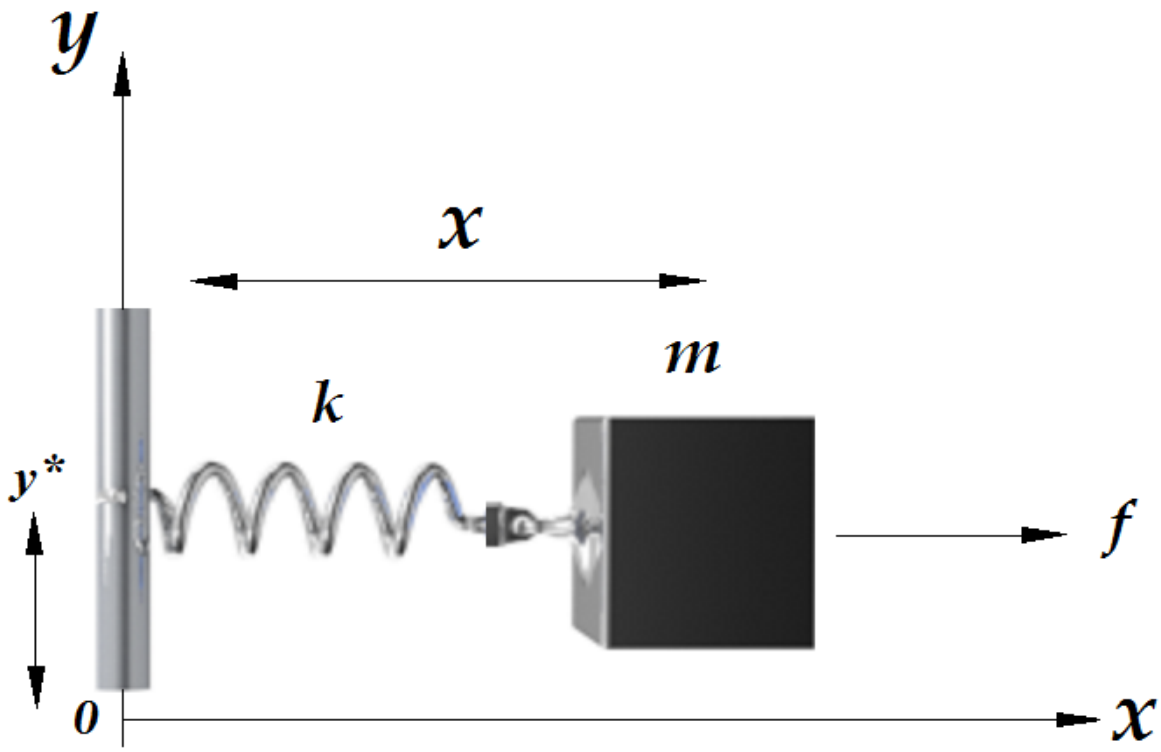


Figura 10.13 Sistema masa resorte horizontal.

Tabla 10.7 Parámetros del sistema masa resorte

Parámetro	Valor
Masa m	5 kg
Coeficiente de fricción viscosa interno b	0.007 N-seg/m
Constante de rigidez k	6 N/m
Constante de la aceleración de la gravedad g	9.81 m/seg^2

La simulación consiste en posición al sistema masa resorte a 10 cm con respecto a su posición de reposo.

Código Fuente 10.17 smrhpd

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo smrhpd.m Versión de MATLAB 2012a

```
function xp =smrhpd(t,x)
    %modelo dinámico del sistema masa resorte en configuración horizontal
    %la fuerza de movimiento lo proporciona el algoritmo proporcional derivativo
    %vector de estados: x = [x1; x2]
    %x1 representa la posición de desplazamiento del bloque rígido de masa m
    %x2 es la velocidad de movimiento
    x1=x(1); % posición articular x = x1
    x2=x(2); %velocidad articular ẋ = x2
    %parámetros del sistema masa resorte
    m=5; %masa
    k=6; %constante de rigidez del resorte
    b=0.007; %coeficiente de fricción viscosa propio o interno del sistema
    %algoritmo de control proporcional derivativo f
    f=pd(x1,x2); %f = kp*x̃ - kv*ẋ
    %ecuación en lazo cerrado
    x2p=(f-b*x2-k*x1)/m; %aceleración del sistema masa resorte
    %vector de salida formado por: velocidad y aceleración de movimiento
    xp=[x2; %velocidad ẋ
        x2p]; %aceleración ẍ
end
```

Código Fuente 10.18 pd

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo pd.m Versión de MATLAB 2012a

```
1  function f=pd(x1,x2)
2      %ganancia proporcional k_p
3      kp=250;
4      %ganancia derivativa k_v
5      kv=60;
6      %posición deseada x_d
7      xd=0.1;
8      %error de posición xtilde = x_d - x
9      xtilde=xd-x1;
10     %control proporcional derivativo PD
11     %f = k_p*xtilde - k_v*xdot
12     f=kp*xtilde-kv*x2;
13 end
```

Código Fuente 10.19 smrhpdsimu

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo smrhpdsimu.m Versión de MATLAB 2012a

```
1  clc;
2  clear all;
3  close all;
4  format short
5  %parámetros de simulación
6  ti=0;
7  h=0.001;
8  tf = 10;
9  ts=ti:h:tf;
10 % vector tiempo para simulación
11 ci=[0; 0];%condiciones iniciales x1(0)=0, x2(0)=0
12 %configuración de la función de integración numérica ode45(...)
13 opciones=odeset('RelTol',1e-06,'InitialStep', h,'MaxStep',h);
14 [t, x]=ode45('smrhpd',ts,ci,opciones);
15 plot(t,x(:,1))%grafica la posición x1(t)
```

Simulación en MATLAB del sistema masa resorte y control PD

Para llevar a cabo la simulación del sistema masa resorte con el algoritmo de control PD se requieren 3 archivos: el modelo dinámico del sistema masa resorte en `smrhpd.m`, el algoritmo de control PD en `pd.m` y el programa principal `smrhpdsimu.m` (**todos los archivos deben estar almacenados en la misma carpeta o directorio definido por el usuario**).

La forma más simple y recomendada para realizar la simulación es a través del editor integrado al entorno de programación de **MATLAB** seleccionando el archivo `smrhpdsimu.m`, oprimir el icono *play* (comando `run`).

Un forma alternativa para realizar la simulación es escribiendo el nombre del programa principal directamente en la ventana de comandos de **MATLAB** (previo indicar la trayectoria (`path`) de la unidad del disco duro donde el usuario tiene los 3 archivos), por ejemplo:

```
fx >> cd('C:/robot/control') %la trayectoria es definida por el usuario ↩
fx >> smrhpdsimu ↩
```

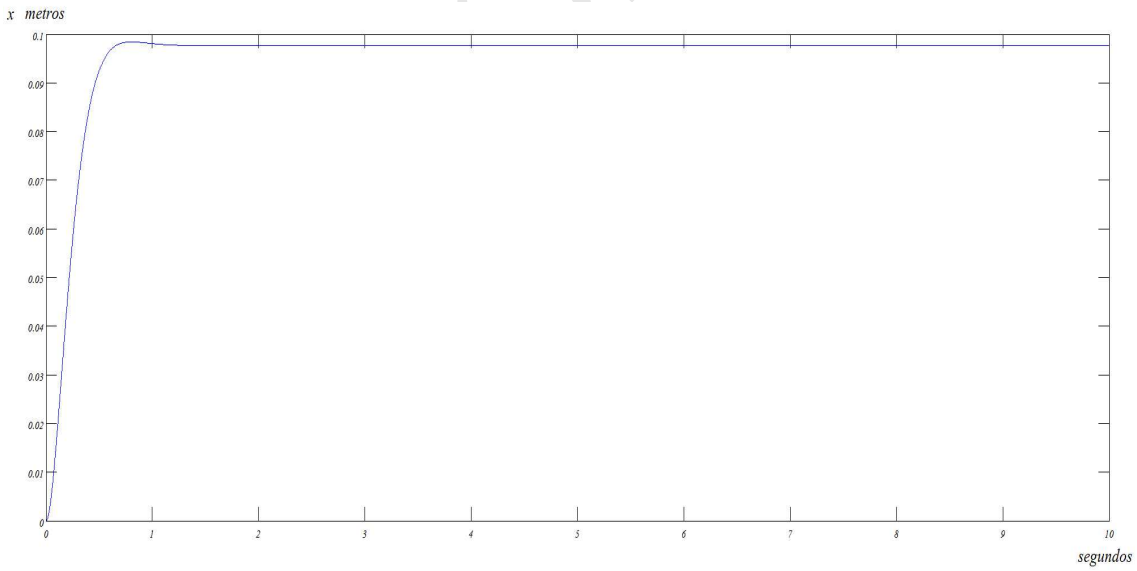


Figura 10.14 Respuesta del sistema masa resorte horizontal con control PD.

Control tangente hiperbólica

El algoritmo de control tangente hiperbólica está dado por la siguiente ecuación:

$$f = k_p \tanh(\tilde{x}) - k_v \tanh(\dot{x})$$

Por lo tanto, la ecuación en lazo cerrado que combina el modelo dinámico del sistema masa resorte y control tangente hiperbólico adquiere la siguiente expresión:

$$\frac{d}{dt}\begin{bmatrix}\tilde{x}\\ \dot{x}\end{bmatrix} = \begin{bmatrix}-\dot{x}\\ \frac{1}{m}\left[\, k_p \tanh(\tilde{x}) - k_v \tanh(\dot{x}) - b\dot{x} - kx \,\right]\end{bmatrix}$$

Para realizar la simulación del sistema masa resorte en configuración horizontal se requieren los arhchivos del modelo dinámico que describe los efectos físicos del sistema masa resorte `smrhtanh.m`, el esquema de control tangente hiperbólico `controltanh.m` y el programa principal que permite realizar la simulación `smrhtanhsimu.m` descritos respectivamente en los cuadro 10.20, 10.21 y 10.22. Los resultados de simulación se ilustran en la figura 10.15.

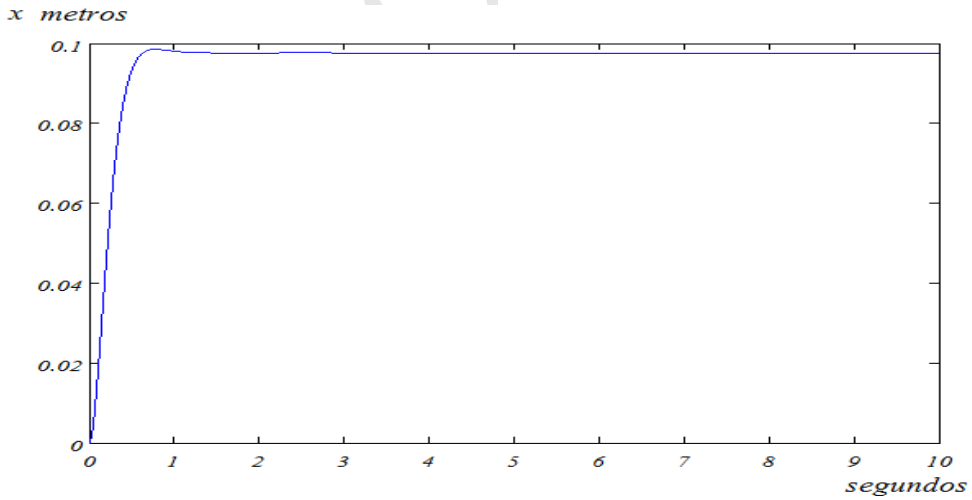


Figura 10.15 Respuesta del sistema masa resorte horizontal con control tangente hiperbólica.

Código Fuente 10.20 smrhtanh

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo smrhtanh.m Versión de **MATLAB** 2012a

```
1  function xp =smrhtanh(t,x)
2      %modelo dinámico del sistema masa resorte en configuración horizontal
3      %la fuerza de movimiento lo proporciona el algoritmo proporcional derivativo
4      %vector de estados: x = [x1; x2]
5      %x1 representa la posición de desplazamiento del bloque rígido de masa m
6      %x2 es la velocidad de movimiento
7      x1=x(1);% posición articular x = x1
8      x2=x(2);%velocidad articular ẋ = x2
9      %parámetros del sistema masa resorte
10     m=5;%masa
11     k=6;%constante de rigidez del resorte
12     b=0.007;%coeficiente de fricción viscosa propio o interno del sistema
13     %algoritmo de control proporcional derivativo f
14     f=controltanh(x1,x2);%f = kp*x̃ - kv*ẋ
15     %ecuación en lazo cerrado
16     x2p=(f-b*x2-k*x1)/m;%aceleración del sistema masa resorte
17     %vector de salida formado por: velocidad y aceleración de movimiento
18     xp=[x2;%velocidad ẋ
19         x2p];%aceleración ẍ
20 end
```

Código Fuente 10.21 controltanh

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo controltanh.m Versión de MATLAB 2012a

```
1  function f=controltanh(x1,x2)
2      %ganancia proporcional k_p
3      kp=250;
4      %ganancia derivativa k_v
5      kv=60;
6      %posición deseada x_d
7      xd=0.1;
8      %error de posición \tilde{x} = x_d - x
9      xtilde=xd-x1;
10     %control proporcional derivativo PD
11     %f = k_p tanh(\tilde{x}) - k_v tanh(\dot{x})
12     f=kp*tanh(xtilde)-kv*tanh(x2);
13 end
```

Código Fuente 10.22 smrhtanhsimu

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo smrhtanhsimu.m Versión de MATLAB 2012a

```
1  clc;
2  clear all;
3  close all;
4  format short
5  %parámetros de simulación
6  ti=0;
7  h=0.001;
8  tf = 10;
9  ts=ti:h:tf;
10 % vector tiempo para simulación
11 ci=[0; 0];%condiciones iniciales x1(0)=0, x2(0)=0
12 %configuración de la función de integración numérica ode45(...)
13 opciones=odeset('RelTol',1e-06,'InitialStep', h,'MaxStep',h);
14 [t, x]=ode45('smrhtanh',ts,ci,opciones);
15 plot(t,x(:,1)) %grafica la posición x1(t)
```

Simulación en MATLAB del sistema masa resorte y control tangente hiperbólica

Para llevar a cabo la simulación del sistema masa resorte con el algoritmo de control tangente hiperbólica se requieren 3 archivos: el modelo dinámico del sistema masa resorte en `smrhtanh.m`, el algoritmo de control tangente hiperbólica en `controltanh.m` y el programa principal `smrhtanhsimu.m` (**todos los archivos deben estar almacenados en la misma carpeta o directorio definido por el usuario**).

La forma más simple y recomendada para realizar la simulación es a través del editor integrado al entorno de programación de **MATLAB** seleccionando el archivo `smrhtanhsimu.m`, oprimir el icono *play* (comando `run`).

Un forma alternativa para realizar la simulación es escribiendo el nombre del programa principal directamente en la ventana de comandos de **MATLAB** (previo indicar la trayectoria (`path`) de la unidad del disco duro donde el usuario tiene los 3 archivos), por ejemplo:

```
fx >> cd('C:/robot/control')%la trayectoria es definida por el usuario ↩
fx >> smrhtanhsimu ↩
```

10.3.2 Centrífuga

Considere el sistema mecatrónico que se describe en la figura 10.16 denominado centrífuga se compone de un servomotor que gira alrededor del eje z, el ángulo de rotación está descrito por la variable q. El origen del sistema de referencia cartesiano $\Sigma(x, y, z)$ se ubica en la parte inferior de la base, la distancia del origen del sistema $\Sigma(x, y, z)$ al extremo final o punta terminal de la barra de la centrífuga está representada por $l_1 + l_2$ sen(δ), donde l_1 es la longitud del servomotor, l_2 es la longitud de la barra metálica que se encuentra acoplada al rotor, y δ es un ángulo constante que indica la inclinación de la barra l_2 con respecto a la horizontal, l_{c2} representa el centro de masa de la barra l_2; el centro de masa l_{c2} sirve como punto de referencia para finalidades del análisis dinámico.

El momento de inercia I de la centrífuga. Observe que cuando la centrífuga se encuentra en rotación describe un círculo sobre el plano $x - y$, el radio del círculo que genera el movimiento de la centrífuga se determina por $l_2 \cos(\delta)$, de esta forma el movimiento de la centrífuga tiene componentes en el eje x, y y altura sobre el eje z.

La acción de la gravedad g se encuentra en dirección vertical con sentido contrario al eje z.

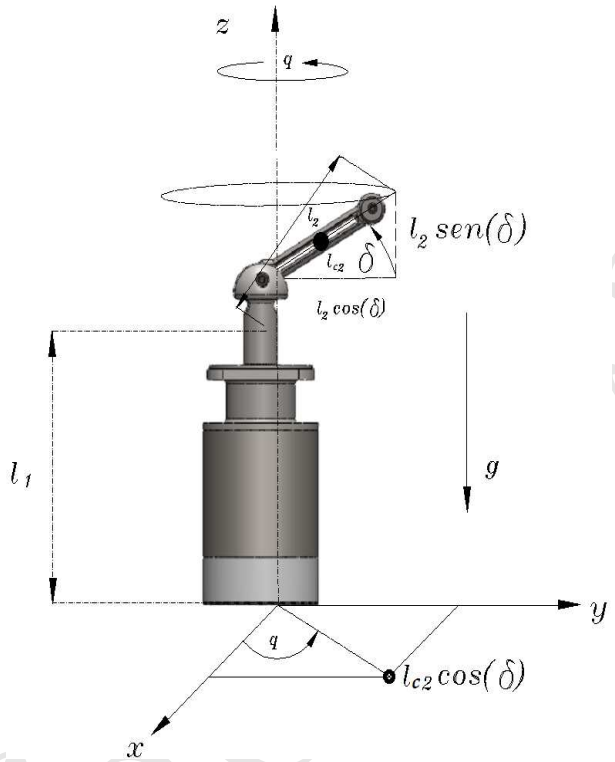


Figura 10.16 Centrífuga.

El modelo dinámico de la centrífuga está descrito por la siguiente ecuación: El modelo dinámico de una centrífuga incluyendo el fenómeno de fricción está dado por:

$$\tau = [\, ml_{c2}^2 \,\mathrm{sen}(\delta) + I \,]\, \ddot{q} + b\dot{q} \tag{10.1}$$

Considere la tabla 10.8 donde se muestran los valores numéricos de la centrífuga. Estos parámetros serán requeridos en procesos de simulación.

Para propósitos de simulación posicionar a la centrífuga en 90 grados por medio de un algoritmo de control con estructura arco tangente.

Tabla 10.8 Parámetros de la centrífuga.

Parámetro	Valor
m	5 kg
b	0.12 Nm-seg/rad
l_{c2}	0.01 m
I	0.08 Nm-seg^2/rad
δ	$45\frac{\pi}{180}$ rad

Control arcotangente

El esquema de control arco tangente se define de la siguiente forma:

$$\tau = k_p \text{atan}(\tilde{q}) - k_v \text{atan}(\dot{q})$$

donde $k_p, k_v \in \mathbb{R}_+$ son las ganancias proporcional y derivativa, respectivamente, el error de posición se define como la diferencia entre la posición deseada q_d y la posición actual del robot q, es decir: $\tilde{q} = q_d - q$, el par aplicado es τ.

La ecuación en lazo cerrado adquiere la siguiente estructura:

$$\frac{d}{dt}\begin{bmatrix}\tilde{q}\\ \dot{q}\end{bmatrix} = \begin{bmatrix}-\dot{q}\\ \frac{1}{ml_{c2}^2\ \text{sen}(\delta)+I}\left[k_p \text{atan}(\tilde{q}) - k_v \text{atan}(\dot{q}) - b\dot{q}\right]\end{bmatrix}$$

Simulación en MATLAB de la centrífuga y control arco tangente

Para llevar a cabo la simulación de la centrífuga con el algoritmo de control arco tangente se requieren 3 archivos: el modelo dinámico de la centrífuga en `centrifugaatan.m`, el algoritmo de control arco tangente en `controlatan.m` y el programa principal `centrifugaatansimu.m` (**todos los archivos deben estar almacenados en la misma carpeta o directorio definido por el usuario**).

La forma más simple y recomendada para realizar la simulación es a través del editor integrado dentro del entorno de programación de **MATLAB** seleccionando el archivo `centrifugaatansimu.m`, oprimir el icono *play* (comando `run`).

Un forma alternativa para realizar la simulación es escribiendo el nombre del programa principal directamente en la ventana de comandos de **MATLAB** (previo indicar la trayectoria (`path`) de la unidad del disco duro donde el usuario tiene los 3 archivos), por ejemplo:

```
fx >> cd('C:/robot/control')%la trayectoria es definida por el usuario ↩
fx >> centrifugaatansimu ↩
```

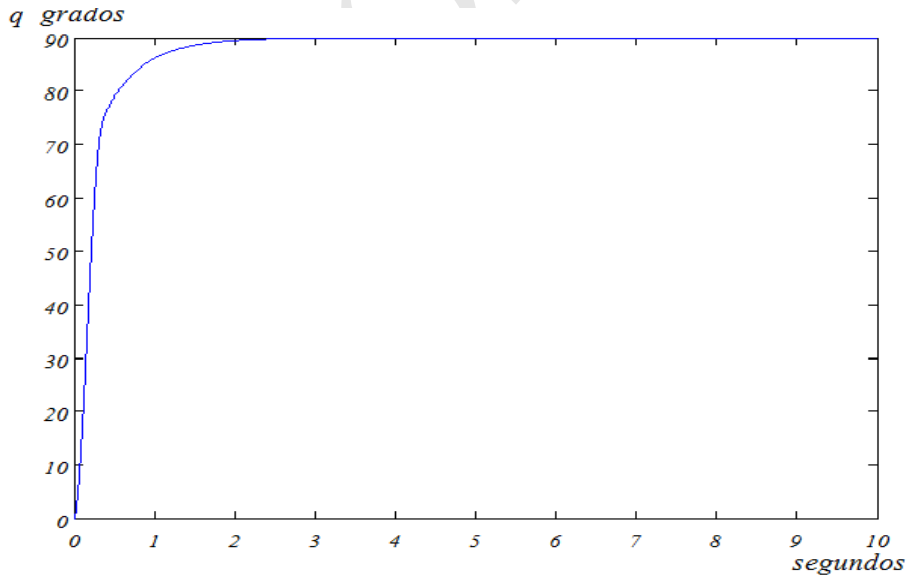


Figura 10.17 Respuesta de la centrífuga con el algoritmo de control arco tangente.

Código Fuente 10.23 centrifugaatan

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo centrifugaatan.m Versión de MATLAB 2012a

```
1  function xp =centrifugaatan(t,x)
2      %variables de estado
3      q=x(1); % posición articular
4      qp=x(2); % velocidad articular
5      %parámetros de la centrífuga
6      m=5; %masa de la centrífuga
7      lc2=0.01; %centro de masa de la barra inclinada
8      delta=45*pi/180; %ángulo de inclinación de la barra l_2
9      I=0.08; % momento de inercia del rotor del servomotor
10     b=0.12; %coeficiente de fricción viscosa
11     %aceleración rotacional de la centrífuga
12     tau=controlatan(q,qp);
13     qpp=(tau-b*qp)/(m*lc2*lc2*sin(delta)+I);
14     %vector de salida
15     xp=[qp;
16         qpp ];
17 end
```

Código Fuente 10.28 controlatan

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo controlatan.m Versión de **MATLAB** 2012a

```
1 function tau=controlatan(q,qp)
2     kp=10;%ganancia proporcional k_p
3     kv=5;%ganancia derivativa k_v
4     qd=pi*90/180;%posición deseada q_d
5     qtilde=qd-q;%error de posición q~
6     tau=kp*atan(qtilde)-kv*atan(qp);%τ = k_p atan(q~) − k_v atan(q̇)
7 end
```

Código Fuente 10.28 centrifugaatansimu

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo centrifugaatansimu.m Versión de **MATLAB** 2012a

```
1 clc; clear all; close all; format short
2 %parámetros de simulación
3 ti=0; h=0.001; tf = 10;
4 ts=ti:h:tf;%vector tiempo para simulación
5 ci=[0; 0];%condiciones iniciales x1(0)=0, x2(0)=0
6 %configuración de la función de integración numérica ode45(...)
7 opciones=odeset('RelTol',1e-06,'InitialStep', h,'MaxStep',h);
8 [t, x]=ode45('centrifugaatan',ts,ci,opciones);
9 plot(t,180*x(:,1)/pi)%grafica la posición x1(t)
```

Control tipo exponencial

Un esquema de control con estructura exponencial es el que se muestra a continuación:

$$\tau \quad = \quad k_p \left[1 - \alpha e^{-\alpha \tilde{q}^2} \right] \tilde{q} - k_v \left[1 - \beta e^{-\beta \dot{q}^2} \right] \dot{q}$$

donde $k_p, k_v \in \mathbb{R}_+$ son las ganancias proporcional y derivativa, respectivamente, $\alpha, \beta \in \mathbb{R}_+$; $\alpha < 1$ y $\beta < 1$ son parámetros de diseño, el error de posición se define como la diferencia entre la posición deseada q_d y la posición actual del robot q, es decir: $\tilde{q} = q_d - q$, el par aplicado es τ.

La ecuación en lazo cerrado adquiere la siguiente estructura:

$$\frac{d}{dt} \begin{bmatrix} \tilde{q} \\ \dot{q} \end{bmatrix} \quad = \quad \begin{bmatrix} -\dot{q} \\ \frac{1}{ml_{c2}^2 \ \mathrm{sen}(\delta) + I} \left[k_p \left[1 - \alpha e^{-\alpha \tilde{q}^2} \right] \tilde{q} - k_v \left[1 - \beta e^{-\beta \dot{q}^2} \right] \dot{q} - b\dot{q} \right] \end{bmatrix}$$

Simulación en MATLAB de la centrífuga y control exponencial

Para llevar a cabo la simulación de la centrífuga con el algoritmo de control tipo exponencial se requieren 3 archivos: el modelo dinámico de la centrífuga en `centrifugaexp.m`, el algoritmo de control exponencial en `controlexp.m` y el programa principal `centrifugaexpsimu.m` (**todos los archivos deben estar almacenados en la misma carpeta o directorio definido por el usuario**).

La forma más simple y recomendada para realizar la simulación es a través del editor integrado dentro del entorno de programación de **MATLAB** seleccionando el archivo `centrifugaexpsimu.m`, oprimir el icono *play* (comando `run`).

Un forma alternativa para realizar la simulación es escribiendo el nombre del programa principal directamente en la ventana de comandos de **MATLAB** (previo indicar la trayectoria (`path`) de la unidad del disco duro donde el usuario tiene los 3 archivos), por ejemplo:

```
fx >> cd('C:/robot/control')%la trayectoria es definida por el usuario ↩
fx >> centrifugaexpsimu ↩
```

Código Fuente 10.26 centrifugaexp

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo centrifugaexp.m Versión de MATLAB 2012a

```
1  function xp =centrifugaexp(t,x)
2      %variables de estado
3      q=x(1);% posición articular
4      qp=x(2);% velocidad articular
5      %parámetros de la centrífuga
6      m=5;%masa de la centrífuga
7      lc2=0.01;%centro de masa de la barra inclinada
8      delta=45*pi/180;%ángulo de inclinación de la barra l_2
9      I=0.08;% momento de inercia del rotor del servomotor
10     b=0.12;%coeficiente de fricción viscosa
11     %aceleración rotacional de la centrífuga
12     tau=controlexp(q,qp);
13     qpp=(tau-b*qp)/(m*lc2*lc2*sin(delta)+I);
14     %vector de salida
15     xp=[qp;
16         qpp ];
17 end
```

Código Fuente 10.28 controlexp

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo controlexp.m Versión de MATLAB 2012a

```
function tau=controlatan(q,qp)
    kp=10; kv=5;%ganancia proporcional k_p y ganancia derivativa k_v
    qd=pi*90/180;%posición deseada q_d
    qtilde=qd-q;%error de posición q~
    alpha=0.1; beta=0.1;%parámetros de diseño 0 < α < 1, 0 < β < 1
    τ = k_p[1 − αe^{−αq̃^2}]q̃ − k_v[1 − βe^{−βq̇^2}]q̇
    tau=kp*(1-alpha*exp(-alpha*qtilde*qtilde))*qtilde-kv*(1-beta*exp(-beta*qp*qp))*qpe;%
end
```

Código Fuente 10.28 centrifugaexpsimu

Mecatrónica. Control y Automatización.

Capítulo 10 Control de robots manipuladores.

Fernando Reyes Cortés, Jaime Cid Monjaraz y Emilio Vargas Soto.

Alfaomega Grupo Editor: "**Te acerca al conocimiento**".

Archivo centrifugaexpsimu.m Versión de MATLAB 2012a

```
clc; clear all; close all; format short
%parámetros de simulación
ti=0; h=0.001; tf = 10;
ts=ti:h:tf;%vector tiempo para simulación
ci=[0; 0];%condiciones iniciales x1(0)=0, x2(0)=0
%configuración de la función de integración numérica ode45(...)
opciones=odeset('RelTol',1e-06,'InitialStep', h,'MaxStep',h);
[t, x]=ode45('centrifugaexp',ts,ci,opciones);
plot(t,180*x(:,1)/pi)%grafica la posición x1(t)
```

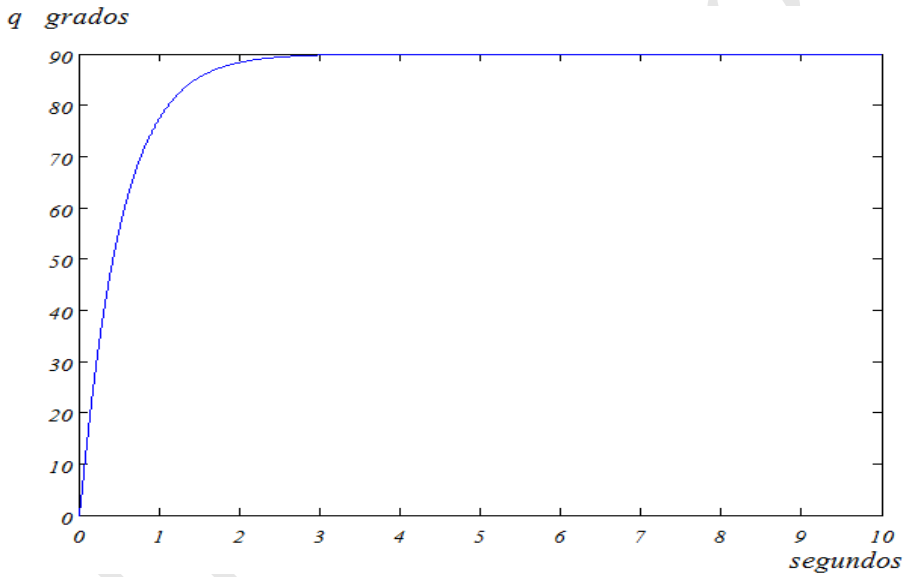


Figura 10.18 Respuesta de la centrífuga con el algoritmo de control exponencial.