Preliminares matemáticos

Facultad de Ciencias de la Electrónica

Benemérita Universidad Autónoma de Puebla

Licenciatura en Ingeniería Mecatrónica

Dr. Fernando Reyes Cortés

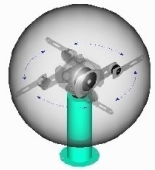
Robótica

ftp://ece.buap.mx/pub/profesor/FernandoReyes/Robotica/

Primavera 2020

Parte V

Sistemas Dinámicos lineales de segundo orden



Robótica FCE

BUAP

Contenido

Sistemas Dinámicos lineales de segundo orden

El modelo dinámico de un sistema lineal de segundo orden está dado por la siguiente ecuación:

$$\ddot{y} + a_1\dot{y} + a_0 y = b_0 u \tag{1}$$

la relación que existe con la función de transferencia del sistema lineal se encuentra establecida de la siguiente manera:

$$\ddot{y}(t) + a_1\dot{y}(t) + a_0 y(t) = b_0 u(t) \Rightarrow \left[s^2 + a_1 s + a_0\right] y(s) = b_0 u(s) \iff G(s) = \frac{y(s)}{u(s)} = \frac{b_0}{s^2 + a_1 s + a_0}$$

En un sistema lineal invariante en el tiempo, el modelo dinámico (1) y la función de transferencia $G(s)$ proporcionan información específica sobre el comportamiento del sistema. Este tipo de modelado puede ser representado en función de los parámetros $(\rho,\ w_n)$:

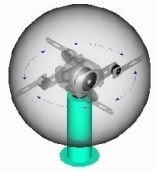
- El parámetro de amortiguamiento ρ, físicamente representa la cantidad de fricción, inyecta a la respuesta del sistema amortiguamiento o freno mecánico. La fricción es un fenómeno disipativo que convierte la energía mecánica en energía térmica.
- La frecuencia natural w_n, es la frecuencia de resonancia u oscilación natural del sistema.

Sea $a_0 = b_0 = w_n^2$ y $a_1 = 2\rho w_n$, el modelo dinámico lineal de segundo orden y la función de transferencia quedan:

$$\ddot{y}(t) + a_1\dot{y}(t) + a_0 y(t) = b_0 u(t) \Rightarrow \ddot{y}(t) + 2\rho w_n\, \dot{y}(t) + w_n^2\, y(t) = w_n^2 u(t) \tag{2}$$

$$G(s) = \frac{y(s)}{u(s)} = \frac{b_0}{s^2 + a_1 s + a_0} = \frac{w_n^2}{s^2 + 2\rho w_n s + w_n^2} \tag{3}$$

La ubicación de los polos en el plano s, queda en términos de ρ y w_n definiendo el comportamiento del sistema.



$s = \frac{d}{dt}; \dot{x} = sx, \ddot{x} = s^2x, y = x_1$ (condiciones iniciales cero):

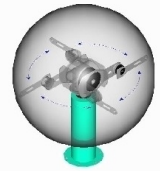
$$\begin{aligned} \frac{y(s)}{u(s)} &= \frac{b_0}{s^2 + a_1 s + a_0} = \frac{w_n^2}{s^2 + 2\rho w_n s + w_n^2} \\ w_n^2 u &= \underbrace{\ddot{y}}_{\dot{x}_2} + 2\rho w_n \underbrace{\dot{y}}_{x_2} + w_n^2 \underbrace{y}_{x_1} \end{aligned}$$

$s = \frac{d}{dt}; \dot{y} = sy, \ddot{y} = s^2y, y = x_1$ (condiciones iniciales cero), w_n es la frecuencia natural de resonancia y ρ es el factor de amortiguamiento:

Variables de estado: variable física $y = y(t)$, $x = x(t)$ variable fase.

Modelo dinámico en variables fase

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -w_n^2 x_1 - 2\rho w_n x_2 + w_n^2 u \\ \underbrace{\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{\dot{\boldsymbol{x}}} &= \underbrace{\begin{bmatrix} 0 & 1 \\ -w_n^2 & -2\rho w_n \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{\boldsymbol{x}} + \underbrace{\begin{bmatrix} 0 \\ w_n^2 \end{bmatrix}}_{B} u \\ \dot{\boldsymbol{x}} &= A\boldsymbol{x} + Bu; \qquad y = [1\ 0] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \boldsymbol{c}^T \boldsymbol{x}. \end{aligned}$$



Sistema lineal (modelo dinámico)

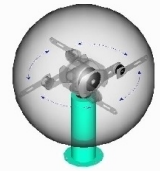
$$\begin{aligned} \dot{\boldsymbol{x}} &= A\boldsymbol{x} + Bu \\ y &= \boldsymbol{c}^T\boldsymbol{x} \end{aligned}$$

$s = \frac{d}{dt}$ (condiciones iniciales cero):

Función de transferencia:

$$\begin{aligned} \dot{\boldsymbol{x}} &= A\boldsymbol{x} + Bu \Rightarrow (sI - A)\boldsymbol{x} = Bu \\ \boldsymbol{x} &= [sI - A]^{-1}Bu \\ y &= \boldsymbol{c}^T\boldsymbol{x} = \boldsymbol{c}^T[sI - A]^{-1}Bu \\ \frac{y}{u} &= \boldsymbol{c}^T[sI - A]^{-1}B \end{aligned}$$

En el dominio de la frecuencia $s = jw$.

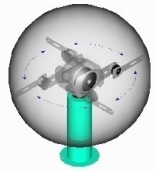


Robótica FCE

BUAP

Partiendo del modelo dinámico obtener la función de transferencia: Función de transferencia:

$$
\begin{aligned}
\dot{\boldsymbol{x}} &= A\boldsymbol{x} + Bu = \begin{bmatrix} 0 & 1 \\ -w_n^2 & -2\rho w_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ w_n^2 \end{bmatrix} u \\
y &= \begin{bmatrix} 1\ 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\
\boldsymbol{x}(s) &= [sI - A]^{-1} Bu = \left[s \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & 1 \\ -w_n^2 & -2\rho w_n \end{pmatrix} \right]^{-1} \begin{bmatrix} 0 \\ w_n^2 \end{bmatrix} u \\
&= \begin{bmatrix} s & -1 \\ w_n^2 & s + 2\rho w_n \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ w_n^2 \end{bmatrix} u = \frac{1}{s^2 + 2\rho w_n s + w_n^2} \begin{bmatrix} s + 2\rho w_n & 1 \\ -w_n^2 & s \end{bmatrix} \begin{bmatrix} 0 \\ w_n^2 \end{bmatrix} u = \frac{1}{s^2 + 2\rho w_n s + w_n^2} \begin{bmatrix} w_n^2 \\ s w_n^2 \end{bmatrix} u \\
y(s) &= \boldsymbol{c}^T \boldsymbol{x} = \boldsymbol{c}^T [sI - A]^{-1} Bu = \begin{bmatrix} 1 & 0 \end{bmatrix} \frac{1}{s^2 + 2\rho w_n s + w_n^2} \begin{bmatrix} w_n^2 \\ s w_n^2 \end{bmatrix} u = \frac{w_n^2}{s^2 + 2\rho w_n s + w_n^2} u \\
\frac{y(s)}{u(s)} &= \frac{w_n^2}{s^2 + 2\rho w_n s + w_n^2}
\end{aligned}
$$



El comportamiento de la respuesta del sistema de segundo orden para una entrada escalón, considerando los casos de subamortiguamiento, amortiguamiento crítico y sobreamortiguamiento.

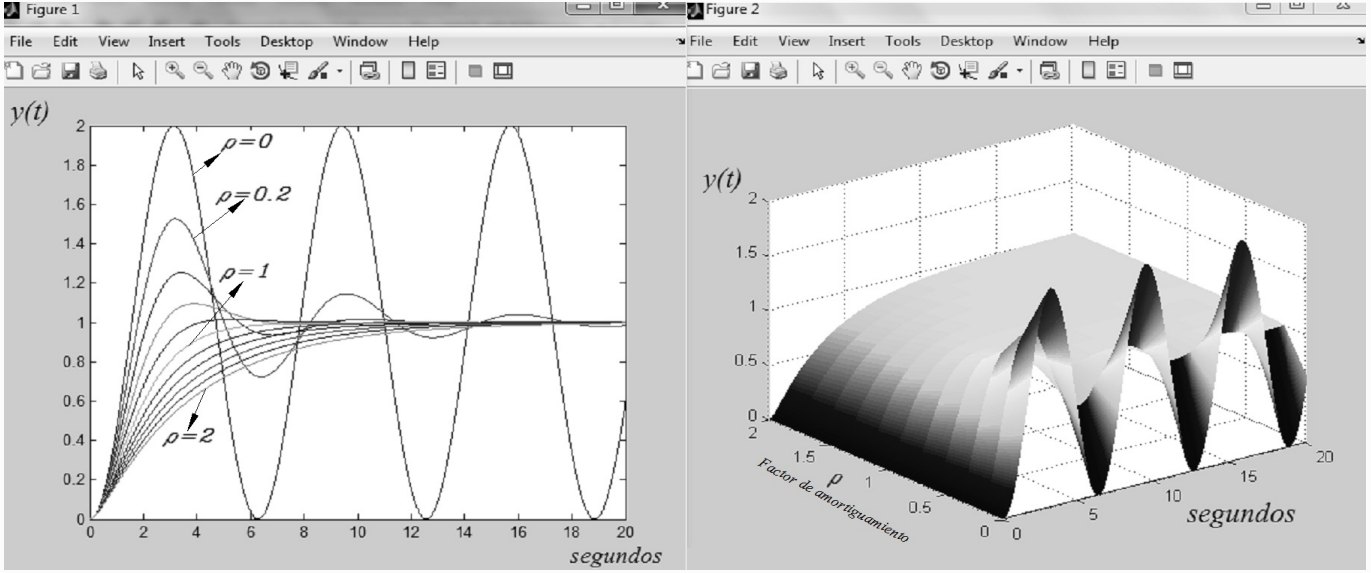


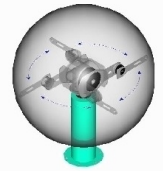
Figura 1: Respuesta a un escalón del sistema $\frac{w_n^2}{s^2+2\rho w_n s+w_n^2}$ variando $\rho \in [0,2]$ y $w_n = 1$.

Código MATLAB 1: sistema2orden

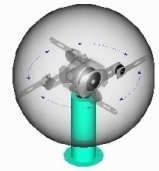
Robótica, Período de primavera (16 de enero de 2020). Licenciatura en Ingeniería Mecatrónica.
Dr. Fernando Reyes Cortés. Facultad de Ciencias de la Electrónica, BUAP.

sistema2orden.m — MATLAB 2019a

```
1  clc; clear all; close all;
2  format short
3  % tiempo de simulación (segundos) tini=0; h=0.001; tfinal = 20;
4  t=tini:h:tfinal; % vector tiempo
5  wn=1;
6  rho=0.2;
7  % G(s) = wn^2/(s^2+2*rho*wn*s+wn^2)
8  num=[wn*wn];
9  den=[1, 2*rho*wn, wn*wn];
10 step(num,den,t)
```



Respuesta a un escalón



La conversión del modelo dinámico (2) al modelo de variables de estado $\dot{\boldsymbol{x}} = A\boldsymbol{x} + Bu$, $y = \boldsymbol{c}^T\boldsymbol{x}$ se encuentra estableciendo un adecuado cambio de variables de la siguiente forma. Sea $x_1 = y$, entonces se obtiene:

$$\underbrace{\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{\dot{\boldsymbol{x}}} = \underbrace{\begin{bmatrix} 0 & 1 \\ -w_n^2 & -2\rho w_n \end{bmatrix}}_{A}\underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{\boldsymbol{x}} + \underbrace{\begin{bmatrix} 0 \\ w_n^2 \end{bmatrix}}_{B} u \tag{4}$$

$$y = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{\boldsymbol{c}^T}\underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{\boldsymbol{x}} \tag{5}$$

Robótica FCE BUAP

Especificaciones de la respuesta temporal

Considere la función de transferencia

$$G(s) = \frac{w_n^2}{s^2 + 2\rho w_n s + w_n^2} \quad \Rightarrow \quad \dot{x} = \begin{bmatrix} 0 & 1 \\ -w_n^2 & -2\rho w_n \end{bmatrix} x + \begin{bmatrix} 0 \\ w_n^2 \end{bmatrix} u \qquad y = \begin{bmatrix} 1 & 0 \end{bmatrix} x$$

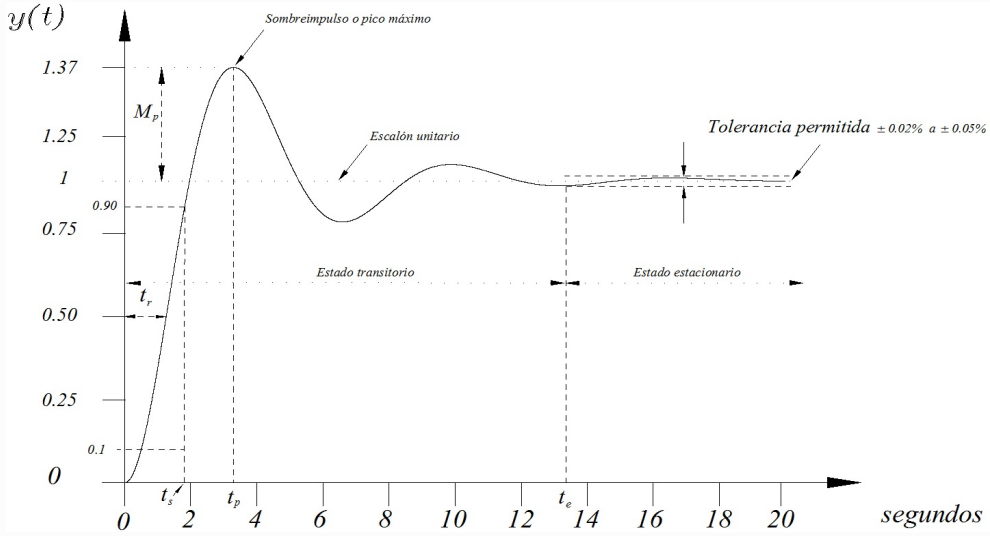
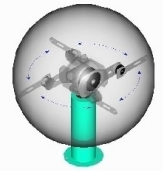
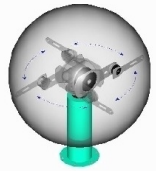


Figura 2: Especificaciones del régimen transitorio y estacionario de la respuesta: $\frac{w_n^2}{s^2+2\rho w_n s+w_n^2}$.



Especificaciones de la respuesta temporal

Las características de respuesta temporal a una entrada escalón para un sistema de segundo orden se especifican a través de un conjunto de parámetros en el dominio del tiempo que definen la etapa transitoria y estacionaria. Estos parámetros son muy útiles para realizar estudios comparativos del desempeño entre dos o más sistemas.
A continuación se presenta la descripción de los parámetros en el dominio del tiempo y la interpretación gráfica se ilustra en la figura 2.



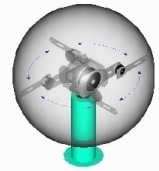
- **Tiempo de subida** t_s es el lapso de tiempo que requiere la respuesta $y(t)$ en ir desde cero o también en pasar del 5 % hasta alcanzar el 95 % del valor final de la magnitud de la señal escalón.
- **Tiempo de retardo** t_r es el tiempo requerido por la respuesta del sistema $y(t)$ en alcanzar por primera vez el 50 % de la magnitud de la entrada escalón.
- M_p es el **máximo pico o sobreimpulso** que la respuesta alcanza; su medición se realiza en términos porcentuales con referencia al valor final o magnitud de la entrada escalón:

$$M_p = \frac{y(t_p) - y(t_\infty)}{y(t_\infty)} \times 100\,\% \tag{6}$$

- **Tiempo de pico** t_p es el tiempo que la respuesta alcanza el primer sobreimpulso o máximo pico.
- La evolución del tiempo t hacia infinito es t_∞, por lo que $y(t_\infty)$ es el valor de la respuesta en estado estacionario.
- **Tiempo de estado estacionario** t_e es el tiempo en que la respuesta del sistema se mantiene en la banda de tolerancia del valor final; generalmente se establece como banda de tolerancia a la región donde la variación de la respuesta $y(t)$ debido al rizo o ruido se encuentra comprendida dentro del $\pm 2\,\%$ al $\pm 5\,\%$ de la magnitud de la entrada escalón. El tiempo t_e define el momento en que la respuesta del sistema entra en estado estacionario, es decir: $t \geq t_e$ y la etapa o fase transitoria queda determinada para $t < t_e$.

Robots manipuladores son sistemas mecánicos muy complejos cuya descripción analítica requiere de ecuaciones diferenciales.

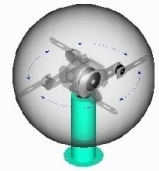
- La naturaleza no lineal, multivariable y fuerte acoplamiento en su comportamiento dinámico ofrece un amplio espectro en la formulación de problemas de control teóricos y prácticos.
- El modelo dinámico del robot manipulador permite explicar todos los fenómenos físicos que se encuentran en su estructura mecánica, tales como efectos inerciales, fuerzas centrípetas y de Coriolis, par gravitacional y fricción, los cuales son fenómenos físicos propios de la naturaleza dinámica del robot.
- Hay varios métodos de modelado de la física como el de Newton o el de Hamilton.
- Sin embargo, la mejor opción como metodología de modelado la representa las ecuaciones de movimiento de Euler-Lagrange debido a las propiedades matemáticas que se deducen de manera natural, ya que facilitan el análisis y diseño de algoritmos de control.



El modelo dinámico de robots manipuladores es fundamental en aplicaciones de simulación, diseño y construcción del sistema mecánico, así como en análisis y diseño de algoritmos de control.

Concepto de simulación

El área de ***simulación*** requiere del modelo dinámico debido a que este modelo puede reproducir todos los fenómenos físicos del robot sin la necesidad de usar un robot real (realidad virtual), y esta característica resulta clave para evaluar algoritmos de control, técnicas de planeación de trayectorias, programación de aplicaciones industriales, etc.



Robótica FCE

BUAP

Simulación

La simulación es el empleo del modelo dinámico para analizar y describir su comportamiento dinámico en una computadora o sistema mínimo digital y de ahí inferir aplicaciones.

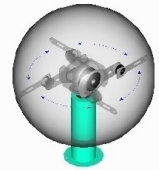
Es importante no confundir simulación con animación, ya que son procesos completamente diferentes:

- La animación no requiere incorporar efectos dinámicos en el movimiento del robot, generalmente son ecuaciones estáticas como la cinemática directa.
- Simulación emplea ecuaciones diferenciales para reproducir fielmente los fenómenos físicos del robot.

Aplicaciones del modelo dinámico

El modelo dinámico es muy útil en:

- La construcción de un robot se fundamenta en el modelo dinámico; los esquemas y planos de ingeniería de los eslabones se deducen directamente del modelo dinámico y se trasladan a un programa CAD para su maquinado y construcción mecánica. De esta forma, un robot industrial puede ser estudiado y se pueden hacer las adecuaciones pertinentes antes de llegar a la etapa de construcción física.
- Simulación: estudio y análisis de la dinámica de un robot manipulador.
- Diseño de algoritmos de control: posición y trayectoria.



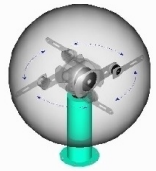
Sistema dinámico

Un sistema dinámico autónomo de primer orden está dado por:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \tag{7}$$

donde $\boldsymbol{x}$ es la variable de estado, la cual proporciona información interna sobre la evolución de los estados internos del sistema.

- La naturaleza autónoma significa que el tiempo t se encuentra implícito, ya que no aparece de manera explícita en la ecuación diferencial ordinaria de primer orden.
- $\boldsymbol{x}$ es una función continua del tiempo $\boldsymbol{x} = \boldsymbol{x}(t)$, así como $\dot{\boldsymbol{x}} = \dot{\boldsymbol{x}}(t)$.
- $\dot{\boldsymbol{x}} = \frac{d}{dt}\boldsymbol{x}(t) \in \mathbb{R}^n$
- La función $\boldsymbol{f}(\boldsymbol{x}(t))$ es continua Lipschitz sobre $\mathbb{R}^n$, continua para toda condición inicial $\boldsymbol{x}(0) \in \mathbb{R}^n$ y continuamente diferenciable en t.
- Existe la solución $\boldsymbol{x} \in \mathbb{R}^n$ de la ecuación diferencial ordinaria de primer orden (7), es única, continua en t y diferenciable con respecto al tiempo.
- $\boldsymbol{f}$ es un mapa vectorial $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$.
- La ecuación diferencial ordinaria de primer orden (7) representa sistemas dinámicos lineales y no lineales.
- La linealidad y no linealidad es con respecto a la variable de estado $\boldsymbol{x}$.



Robótica FCE BUAP

Muestreo de señales continuas

Muestreo de una señal $y(t)$ en tiempo continuo t significa remplazar la señal $y(t)$ por un conjunto de valores discretos $y(t_k)$ en función del tiempo discreto t_k.

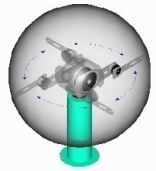
Notación y conceptos

Sea $t \in \mathbb{R}_+$ la representación para la evolución del tiempo continuo y t_k representa un subconjunto del tiempo continuo, es decir: $t_k \subset t$, de tal forma que significa muestras instantáneas, la versión muestreada de la señal $y(t)$ está denota por $y(t_k)$.

El muestreo periódico se define como una operación lineal, donde el tiempo discreto t_k evoluciona como múltiplos del periodo de muestreo de la siguiente forma:

$$t_k = kh$$

donde $h \in \mathbb{R}_+$ representa el periodo de muestreo y $k = 0, 1, 2, 3, \cdots, n$ es un número entero positivo: $k \in 0 \cup N$; la frecuencia de muestreo es $f_s = \frac{1}{h}$ [Hz].

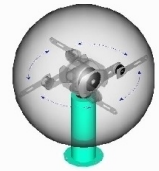


Muestreo

- Es el proceso de tomar un conjunto de muestras o pequeñas partes del sistema o planta de estudio para su examen, procesamiento o análisis.

- En la práctica, muestreo significa que una señal continua en el tiempo se remplaza por una secuencia de números, los cuales representan valores de la señal en ciertos instantes; la señal continua representa la variación en el tiempo de las variables del proceso se convierte en una secuencia de números, los cuales se procesan en la computadora digital.

- El procesamiento da una nueva secuencia de números, que se convierten en una señal continua y se aplican al proceso. El proceso de convertir una secuencia de números en una señal continua se denomina reconstrucción de la señal.



Retenedor y muestreador de orden cero

Un retenedor y muestreador de orden cero (*zero-order sampling and hold*) es un circuito electrónico muy común en los convertidores analógico/digital que se utilizan en control de procesos por computadora. Consiste en un interruptor electrónico que abre y cierra para tomar muestras de la señal continua $y(t)$ cada determinado periodo de muestreo h, la señal discreta $y(t_k)$ es transferida hacia un elemento capacitor a través de un *buffer* o seguidor de voltaje. El capacitor mantiene constante cada valor de conversión (retenedor de orden cero) de la señal muestrea $y(t_k)$ hasta el siguiente periodo de muestreo, la característica principal del retenedor de orden cero es generar una señal discreta $y(t_k)$ con escalones, cada escalón representa un el valor de la señal $y(t_k)$ constante durante un intervalo de muestreo $t_k = kh$ hasta la siguiente muestra $t_{k+1} = (k+1)h$; la figura 3 muestra la configuración típica de un retenedor y muestreador de orden cero.

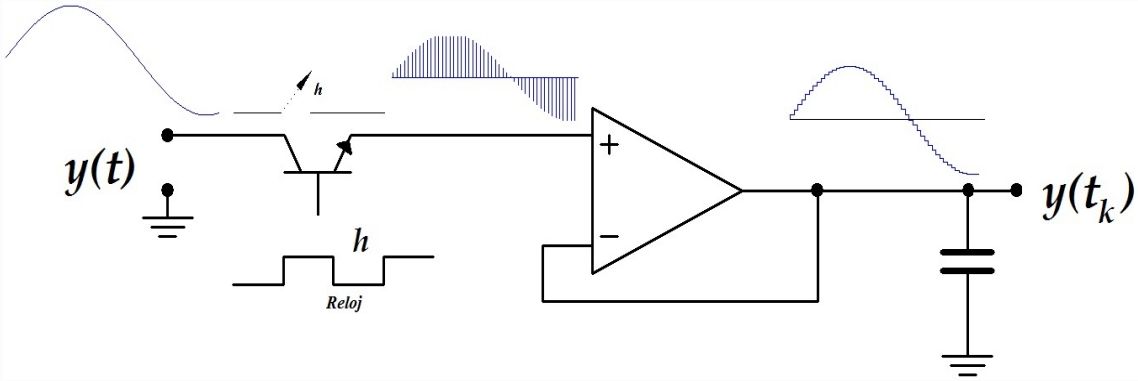


Figura 3: Circuito electrónico retenedor y muestreador de orden cero.

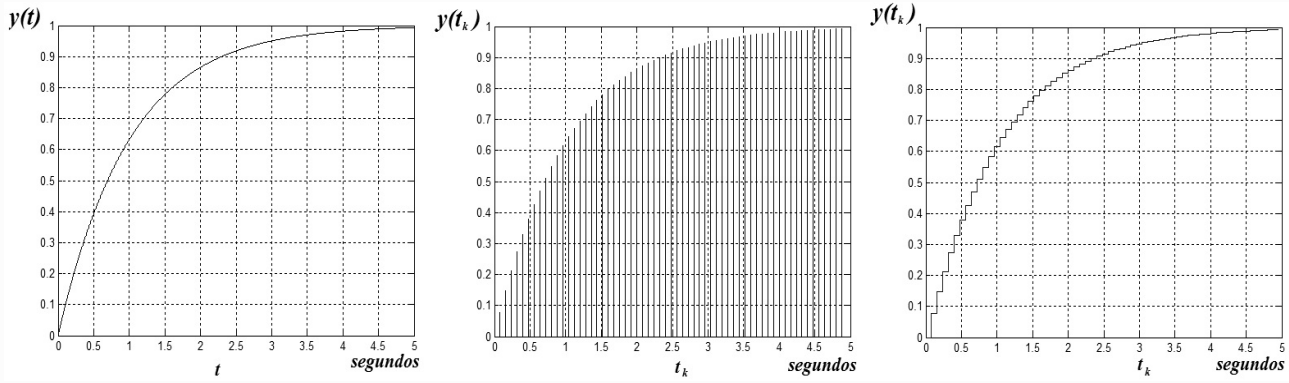


Figura 4: Proceso de discretización de una señal continua $y(t)$.

- Una señal discreta $y(t_k)$ representa una aproximación de la señal continua $y(t)$, las señales $y(t)$ y $y(t_k)$ son iguales $y(t) = y(t_k)$ cuando el tiempo continuo t es idéntico al tiempo discreto t_k.
- El grado de calidad de aproximación depende del número de punto o muestras que se tenga de la señal continua, la figura 5 muestra una señal continua donde se han tomado 5 muestras en forma aperiódica, esta información es insuficiente para propósitos de análisis de la señal y por lo tanto, no puede ser utilizada para aproximar a la señal continua.
- Observe que los puntos muestreados no corresponden a un retenedor de orden cero.

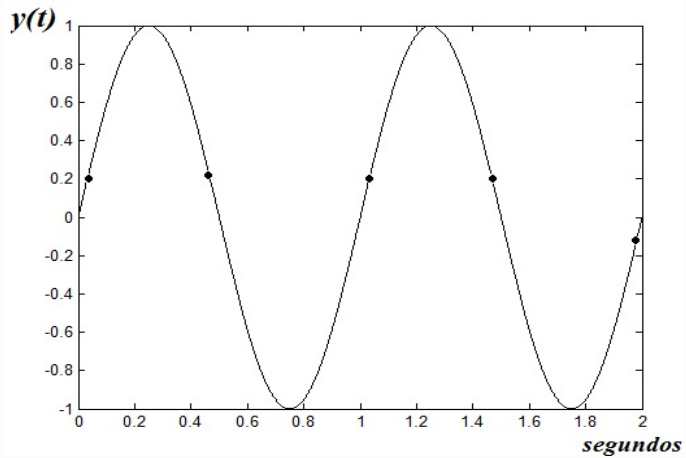
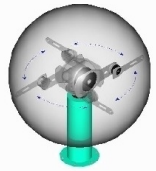


Figura 5: Señal continua y 5 puntos discretos.

- Para un sistema de primer orden el periodo de muestreo es razonable satisfacer: $0 < h \leq 0.1T$, siendo T la constante de tiempo.
- Para un sistema de segundo orden h es: $0 < h \leq 0.1t_s$; donde t_s es el tiempo de subida.



Ejemplo 4.1:

Considere las siguientes señales en tiempo continuo:

$$\begin{aligned} y(t) &= \text{sen}(t) \\ y(t) &= t \\ y(t) &= 1 - e^{-0.3t} \\ y(t) &= \cosh(t) \end{aligned}$$

las correspondientes señales muestreadas $y(t_k)$ utilizando un retenedor de orden cero se muestran en la figura 6.

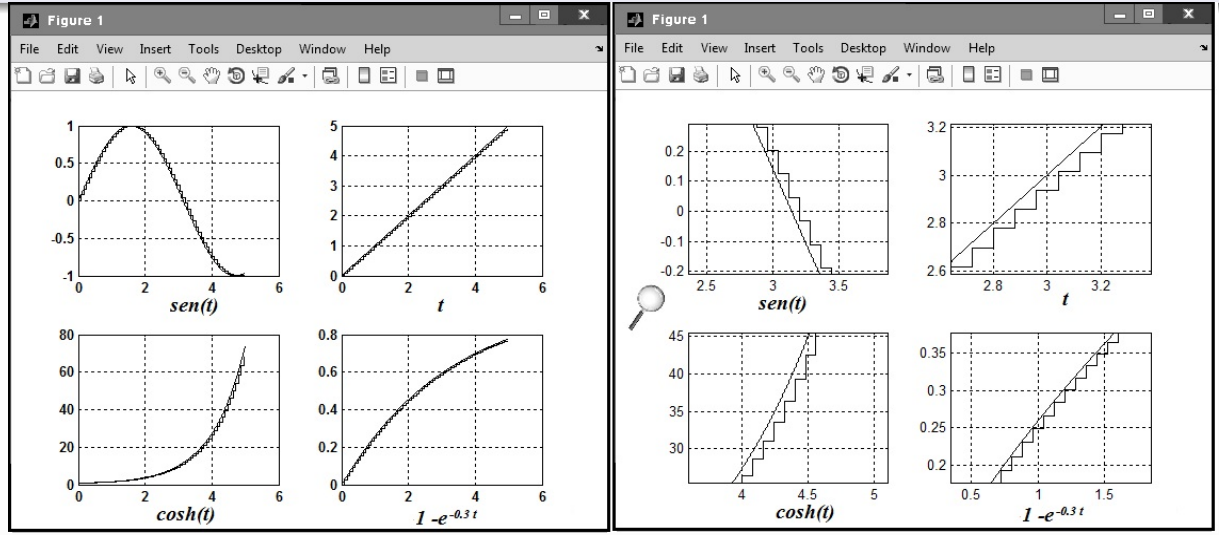
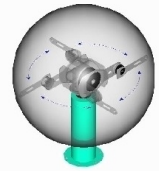


Figura 6: Señales discretizadas con retenedor de orden cero y periodo de muestreo h=0.08 segundos.

Método de integración de Runge-Kutta

Uno de los métodos más importante para integración numérica de ecuaciones diferenciales de primer orden $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ son los métodos de Runge-Kutta. Dichos métodos se basan en aproximar la solución a través de series de Taylor.

El método más simple, el denominado método de primer orden, usa una serie de expansión de Taylor de primer orden, el método de segundo orden usa la serie de expansión de Taylor de segundo orden, y así sucesivamente (el método de Euler es equivalente al de primer orden de Runge-Kutta). **MATLAB** tiene funciones del método de Runge-Kutta para los órdenes segundo, tercero, cuarto y quinto.



La serie de Taylor para evaluarla en el t_k-ésimo tiempo a la función $x(t_k)$ está dada por la siguiente expresión:

$$\begin{aligned} x(t_k) &= x(t_{k-1}) + (t_k - t_{k-1})\dot{x}(t_{k-1}) + \frac{(t_k - t_{k-1})^2}{2!}\ddot{x}(t_{k-1}) + \cdots + \\ &\quad \frac{(t_k - t_{k-1})^n}{n!} \overbrace{\dddot{x}\cdots}^{\text{n derivadas}}_{(t_{k-1})} + \cdots \end{aligned} \tag{8}$$

El término $t_k - t_{k-1}$ representa un pequeño intervalo el cual será representado por $h = t_k - t_{k-1} = kh - (k-1)h$, de tal forma que la serie de Taylor queda expresada de la siguiente forma:

$$x(t_k) = x(t_{k-1}) + h\dot{x}(t_{k-1}) + \frac{h^2}{2!}\ddot{x}(t_{k-1}) + \cdots + \frac{h^n}{n!} \overbrace{\dddot{x}\cdots}^{\text{n derivadas}}_{(t_{k-1})} + \cdots \tag{9}$$

Método de Runge-Kutta de primer orden

La integración numérica de la ecuación $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ por el método de Runge-Kutta de primer orden está dado de la siguiente manera:

$$x(t_k) \quad = \quad x(t_{k-1}) + h\dot{x}(t_{k-1}) = x(t_{k-1}) + hf(t_{k-1}) \tag{10}$$

La interpretación geométrica de la ecuación (10) significa que el valor de estimación de $x(t_k)$ es igual a la línea tangente que la une con el valor $x(t_{k-1})$ como se muestra en la figura 7. Este es un proceso iterativo para todos los puntos $x(t_k)$, con $k = 1, 2, \cdots, n$.

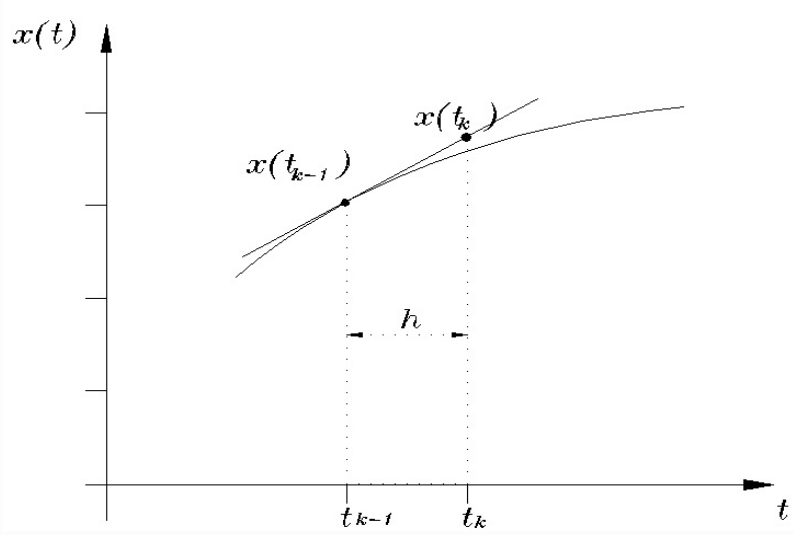
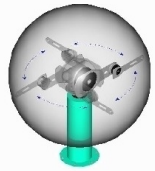


Figura 7: Cálculo de $x(t_k)$ usando el primer el método de Runge-Kutta primer orden.

Método de Runge-Kutta de orden mayor

Los métodos de Runge-Kutta de orden mayor (dos, tercero, cuarto, y quinto) se usan para aproximar funciones desconocidas; la aproximación se realiza a través de varias líneas tangentes, por lo tanto la exactitud es mejorada. Por ejemplo, en el método de integración de cuarto orden usa la serie de Taylor con las primeras cuatro derivadas, es decir la estimación de la función $x(t_k)$ es a través de 4 líneas tangentes.



Funciones ode

MATLAB tiene las funciones para integrar la solución numérica de ecuaciones diferenciales ordinarias de primer orden llamadas `ode`.

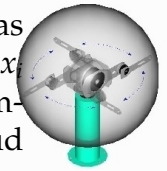
- ode15s
- ode113
- ode23, ode23b, ode 23s, ode23t
- **ode45**

```
opciones=odeset('RelTol',1e-6,'AbsTol',1e-6,'InitialStep',1.0e-3,'MaxStep',1.0e-3)
```

Error de integración: en cada i-ésimo paso de integración se estima un error local de la i-ésima componente de la solución; este error debe ser menor o igual que el error aceptable, el cual es una función de la tolerancia relativa `RelTol` y de la tolerancia absoluta `AbsTol`. La expresión para $e(i)$ satisface la siguiente condición: $|e(i)| = \max(\text{RelTol}|x(i), \text{AbsTol}(i)|)$.

`RelTol` `RelTol` representa el error relativo de tolerancia que se aplica a todas las componentes x_i del vector solución $\boldsymbol{x} \in \mathbb{R}^n$.Este parámetro también controla el número correcto de dígitos en todas las componentes x_i. El valor típico `RelTol`= 1e-3, el cual corresponde al 0.1 % de exactitud.

`AbsTol` Error absoluto de tolerancia que se aplica a todas las componentes individuales x_i del vector solución $\boldsymbol{x}$. También determina la exactitud cuando la solución se aproxima a cero.
Si `AbsTol` es un vector, entonces su dimensión debe ser la misma del vector $\boldsymbol{x}$. El valor de cada componente de `AbsTol` se aplica a la correspondiente componente x_i.
Si `AbsTol` es un escalar, ese valor se aplica a todas las componentes x_i del vector solución $\boldsymbol{x}$.
Generalmente, el valor típico que se le asigna a `AbsTol` es: `AbsTol`= 1e-6.



Robótica FCE BUAP

`NormControl` significa la norma del error relativo. Esta opción solicita a la función `ode45` que el error de integración en cada paso cumpla con la siguiente condición: $\|e\| \leq \max(\text{RelTol}\|x\|, \text{AbsTol})$. Se habilita/deshabilita como: `on (off)`. Para funciones suaves, la función `ode45` entrega un error equivalente a la exactitud solicitada. La exactitud es menor para problemas donde el intervalo de integración es grande y para problemas moderadamente inestables.

`NormControl`

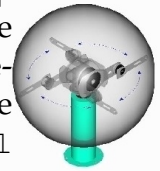
Paso de integración

Especifica el valor inicial del paso de integración. Representa una cota superior en la magnitud del primer paso. En caso de no especificarlo, entonces el tamaño del paso se obtiene como la pendiente de la solución en el tiempo inicial. Si la pendiente de todas las componentes de la solución es cero, entonces el procedimiento puede generar un tamaño de paso muy grande. Por lo tanto, es recomendable iniciar con un valor adecuado esta opción.

`InitialStep`

Representa una cota superior del tamaño del paso, es un valor escalar positivo que se obtiene de la siguiente forma: $0.1|t_{\text{inicial}} - t_{\text{final}}|$. La ecuación diferencial tiene soluciones o coeficientes periódicos, es recomendable inicializar `MaxStep`=$\frac{1}{4}$ `del periodo`.

`MaxStep`



Robótica FCE

BUAP

Función ode15s

Para el caso en que el proceso de integración de la función `ode45` sea muy lento debido a la rigidez del sistema dinámico, entonces es recomendable usar la función `ode15s`. Esta función es de orden variable, multi pasos basada en diferenciación numérica (*backward* o método Gear).

```
[t,x]=ode15s('nombre_funcion',ts,cond_iniciales,opciones)
```

Función ode113

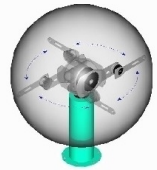
Esta función se recomienda para resolver sistemas dinámicos complicados; mantiene un error de integración riguroso. La función `ode113` es un método de integración de orden variable propuesto por Adams-Bashforth-Moulton.

```
[t,x]=ode113('nombre_funcion',ts,cond_iniciales,opciones)
```

Función ode23

La función `ode23` utiliza el segundo y tercer método de integración. Tiene menor exactitud comparada con la función `ode45`. Sin embargo, puede realizar más rápido el proceso de integración numérica. La sintaxis está dada por:

```
[t,x]=ode23('nombre_funcion',ts,cond_iniciales,opciones)
```



Función ode23b

La función `ode23tb` es una combinación de métodos de Runge-Kutta y diferenciación numérica (*backward*) de segundo orden. Para sistemas dinámicos con alta rigidez del resorte y errores de integración muy pequeños, la exactitud de integración es baja.

```
[t,x]=ode23tb('nombre_funcion',ts,cond_iniciales,opciones)
```

Función ode23s

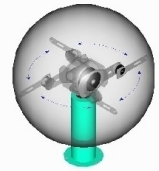
Es muy útil para sistemas dinámicos masa-resorte con alta rigidez en que el resorte la matriz de masas es constante. La función `ode23s` se basa en el método modificado de segundo orden de Rosenbrock. Debido a que el proceso de integración es por pequeños pasos, puede ser más eficiente que la función `ode15s` para tolerancias pequeñas en el error de integración. Además es mucho más efectivo en sistemas dinámicos con rigidez donde `ode15s` no lo es.

```
[t,x]=ode23s('nombre_funcion',ts,cond_iniciales,opciones)
```

Función ode23t

La función `ode23t` emplea el método trapezoidal usando una interpolación libre. Es recomendable para resolver sistemas dinámicos moderados en la rigidez del resorte, sin utilizar amortiguamiento. La exactitud en la integración es baja.

```
[t,x]=ode23t('nombre_funcion',ts,cond_iniciales,opciones)
```



Robótica FCE

BUAP

Simulación de sistemas dinámicos en MATLAB.

La simulación de sistemas dinámicos en **MATLAB** se realiza por medio de las funciones `ode`

Es importante resaltar que la estructura matemática del modelo dinámico debe ser de la forma tradicional en variables fase como una ecuación diferencial ordinaria de primer orden:

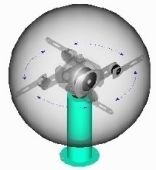
$$\dot{x} = f(x)$$

- Si el modelo original del sistema es una ecuación diferencial de orden superior, entonces mediante un adecuado cambio de variables de estados siempre es posible transformarlo a la estructura requerida.

La función en **MATLAB** que utilizaremos para realizar simulaciones es la técnica de integración de Runge-Kutta 4/5 adaptable:

`ode45(...)`

Documentación de esta función en la ventana de comandos: f_x >> `doc ode45` ↩



Robótica FCE BUAP

Función ode45

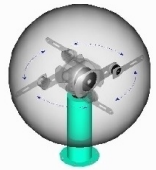
La sintaxis de la función `ode45` es la siguiente:

```
[t,x]=ode45('nombre_funcion',ts,cond_iniciales,opciones)
```

La función `ode45` utiliza los métodos de Runge-Kutta de cuarto y quinto orden. `nombre_funcion` representa una función `M-File` donde está implementado el sistema dinámico en la estructura matemática $\dot{x} = f(x)$.

- La función `ode45` retorna la solución del sistema $\dot{x} = f(x)$, es decir x, así como el vector de tiempo t.
- El tiempo de simulación o el intervalo de integración se encuentra especificado por $ts = [t_{\text{inicial}}, t_{\text{final}}]$, por ejemplo se puede especificar como un intervalo $[0, 10]$.
- Es mucho mejor expresarlo como: $ts = 0 : h : 10$, el cual incluye incrementos de tiempo, de acuerdo al paso de integración h, por ejemplo un milisegundo, $h = 0.001$.
- Las condiciones iniciales se encuentran determinadas por `cond_iniciales`, su forma depende del orden del sistema.
- Por ejemplo, para un sistema escalar `cond_inciales=0`; para el caso vectorial $x(0) \in \mathbb{R}^3$, tenemos `cond_inciales=[0;0;0]`.
- El cuarto parámetro de `opciones` es muy importante debido a que contiene las propiedades de integración numérica, y de eso depende la simulación del sistema. Para tal efecto se emplea la función `odeset` de la siguiente forma:

```
opciones=odeset('RelTol',1e-6,'AbsTol',1e-6,'InitialStep',h,'MaxStep',h)
```



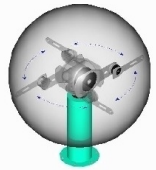
Simulación de sistemas dinámicos

Función de transferencia es la relación de entrada a salida de un sistema lineal con condiciones iniciales cero.

$$\begin{aligned} \frac{y}{u} &= \frac{b}{s+a} \\ \dot{x} &= -ax + bu \\ y &= cx \end{aligned}$$

La solución analítica del modelo dinámico $\dot{x}(t) = -ax(t) + bu(t)$ está dada por la siguiente ecuación:

$$x(t) = e^{-a(t-t_0)}x(t_0) + \int_{t_0}^{t} e^{-a(t-\sigma)}bu(\sigma)d\sigma$$



Simulación de sistemas dinámicos

Para el caso en que $u(t) = 0$ y $t_0 = 0$ la solución analítica de $\dot{x} = -ax + bu(t)$ se reduce a la siguiente expresión:

$$x(t) = x(0)e^{-at}$$

Ejemplo 6.1:

Realizar un programa en **MATLAB** para llevar a cabo la simulación del sistema $\dot{x} = -ax$; considere $a = 2$ y $x(0) = 1$. ¿Qué sucede cuando $x(0) = 0$?

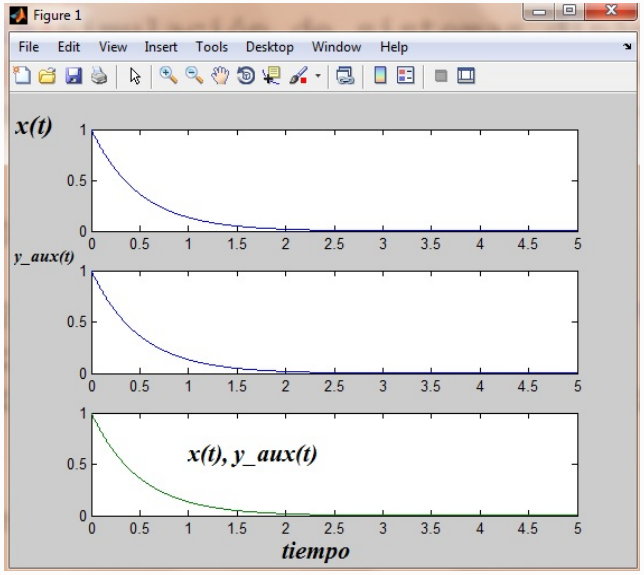
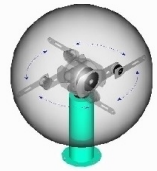


Figura 8: Simulación del sistema lineal $\dot{x}(t) = -ax$; $x(t) = x(0)e^{-at}$.

Simulación en MATLAB del sistema dinámico $\dot{x} = -ax$

La función `ejemplo9.m` que se muestra en el código fuente 2 contiene el código en **MATLAB** para implementar el modelo dinámico descrito por la ecuación diferencial ordinaria (ode): $\dot{x}(t) = -ax(t)$. La simulación se realiza con el programa principal `ejemplo10.m` descrito en el código fuente 3.
El resultado de la simulación se muestra en la figura 8.



Código MATLAB 2: ejemplo9

Robótica, Período de primavera (16 de enero de 2020). Licenciatura en Ingeniería Mecatrónica.
Dr. Fernando Reyes Cortés. Facultad de Ciencias de la Electrónica, BUAP.

ejemplo9.m — **MATLAB** 2019a

```
1 function xp =ejemplo9(t,x)
2     %Sistema dinámico ẋ = −ax
3     a=2;
4     xp=-a*x;
5 end
```

Robótica FCE

BUAP

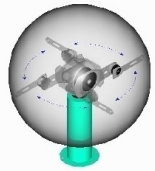
Simulación en MATLAB del sistema dinámico $\dot{x} = -ax$

Código MATLAB 3: ejemplo10

Robótica, Período de primavera (16 de enero de 2020). Licenciatura en Ingeniería Mecatrónica.
Dr. Fernando Reyes Cortés. Facultad de Ciencias de la Electrónica, BUAP.

ejemplo10.m — MATLAB 2019a

```
1  clc;
2  clear all;
3  close all;
4  format short
5   %tiempo de simulación (segundos)
6  ti=0; h=0.001;  tfinal = 5;  ts=ti:h:tfinal; %vector tiempo
7  opciones=odeset('RelTol',1e-06,'InitialStep',h,'MaxStep',h);
8  x0=1; %condición inicial
9  [t, x]=ode45('ejemplo9',ts,x0,opciones); %Integración por el método de Runge-Kutta (4/5)
10  % salida del sistema y=c*x
11 c=1; y=c*x;
12 y_aux=x0*exp(-2*t); %solución analítica y_aux = cx(t) = cy_aux(0)e^{-at} del sistema ẋ = -ax; y_aux(t) = cx(t)
13 subplot(3,1,1); plot(t,x) % gráfica de la solución numérica x(t)
14 subplot(3,1,2); plot(t,y_aux) %gráfica de la solución analítica y(t)
15 subplot(3,1,3); plot(t,x,t,y_aux) % comparación entre la solución numérica x(t) y analítica y(t)
```



Simulación de sistemas dinámicos

Ejemplo 6.2:

Considere el caso del sistema dinámico de primer orden:

$$\begin{aligned} \dot{x}(t) &= -ax(t) + bu(t) \\ y(t) &= cx(t) \end{aligned}$$

Diseñe un programa en **MATLAB** para realizar la simulación del sistema cunado la entrada corresponde a un escalón unitario $u(t) = 1$. Considere los parámetros $a = 2$ y $b = 2$.

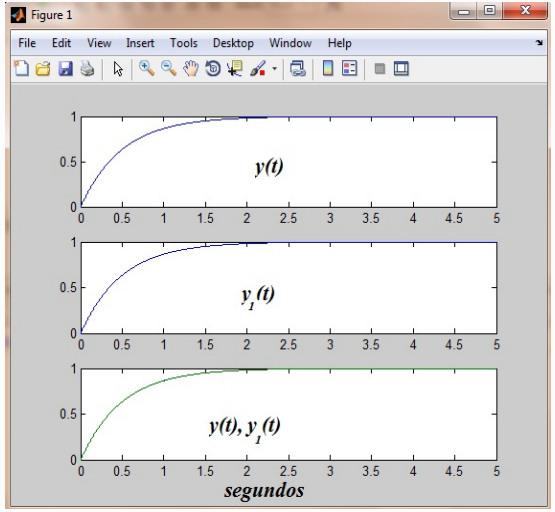


Figura 9: Simulación del sistema lineal $\dot{x}(t) = -ax(t) + bu(t)$.

Simulación en MATLAB del sistema dinámico $\dot{x}(t) = -ax + bu(t)$ para una entrada escalón

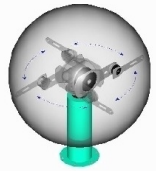
El modelo dinámico $\dot{x} = -ax + bu$ se encuentra implementado em lenguaje **MATLAB** en a función **ejemplo12.m** tal y como se describe en el código fuente 4. El programa principal **ejemplo13.m** permite realizar la simulación, se encuentra descrito en el código fuente 5. El resultado de la simulación se muestra en la figura 9.

Código MATLAB 4: ejemplo12

Robótica, Período de primavera (16 de enero de 2020). Licenciatura en Ingeniería Mecatrónica.
Dr. Fernando Reyes Cortés. Facultad de Ciencias de la Electrónica, BUAP.

ejemplo12.m — MATLAB 2019a

```
1  function xp =ejemplo12(t,x)
2      %Sistema dinámico ẋ = −ax + bu
3      a=2;
4      b=2;
5      %entrada escalón
6      if  t==0
7          u=0;
8          else
9          u=1;
10     end
11     xp=-a*x+b*u;
12 end
```



Robótica FCE

BUAP

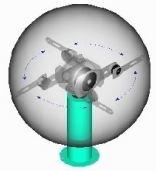
Simulación en MATLAB del sistema dinámico $\dot{x} = -ax + bu$

Código MATLAB 5: ejemplo13

Robótica, Período de primavera (16 de enero de 2020). Licenciatura en Ingeniería Mecatrónica.
Dr. Fernando Reyes Cortés. Facultad de Ciencias de la Electrónica, BUAP.

ejemplo13.m — MATLAB 2019a

```
1  clc;  clear all; close all; format short
2  ti=0;  h=0.001;  tfinal = 5;  ts=ti:h:tfinal;  %tiempo de simulación (segundos) vector tiempo
3  opciones=odeset('RelTol',1e-06,'InitialStep',h,'MaxStep',h);
4  x0=0;%condición inicial
5  [t, x]=ode45('ejemplo12',ts,x0,opciones);%Runge-Kutta 4/5
6  c=1; y=c*x;% salida del sistema y=c x
7  %se genera la función de transferencia G(s) = num(s)/den(s) = 2/(s+2)
8  num=2;% numerador
9  den=[1 2];%denominador
10 G=tf(num,den);%forma comparativa para obtener la respuesta a un escalón del sistema por medio de su función
   de transferencia G(s) y la librería step(...).
11 [y1, t1]=step(G,t);%respuesta a un escalón
12 subplot(3,1,1); plot(t,y) %gráfica de y(t) por ode45(...)
13 subplot(3,1,2); plot(t,y1) %gráfica de y1(t) por la la función step(...)
14 subplot(3,1,3); plot(t,y,t,y1) % gráficas comparativas entre las señales y(t) y y1(t).
```



Robótica FCE

BUAP

Métodos numéricos derivadas e integrales $\dot{x} = -ax + bu$

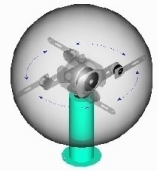
$$\begin{aligned} \frac{y}{u} &= c\frac{b}{s+a} \\ \dot{x} &= -ax + bu \\ y &= cx \end{aligned}$$

Suponga $t_0 = 0$ y como caso particular: $u(t) = 0$:

$$\begin{aligned} \dot{x} &= -ax \ \ \forall t \geq 0 \\ x(t) &= x(0)e^{-at} \qquad \Rightarrow \qquad |x(t)| < \gamma \quad \gamma \in \mathbb{R}_+ \\ x(t_k) &= e^{-ah}x(t_{k-1}) + \frac{b}{a}(1 - e^{-ah})u(t_{k-1}) \\ y(t_k) &= cx(t_k) \\ \dot{x}(t) &\simeq \dot{x}(t_k) = \frac{x(t_k) - x(t_{k-1})}{h} \qquad \text{Método Euler} \\ I_k &= I_{k-1} + h\dot{x}(t_k) \end{aligned}$$

$s = \frac{d}{dt}$; $\dot{x} = sx, y = x$:

$$\begin{aligned} \dot{x} &= -ax + bu \Rightarrow sx = -ax + bu \\ x(s+a) &= bu \\ y &= cx = c\frac{b}{s+a}u \qquad \Rightarrow \frac{y}{u} = c\frac{b}{s+a} \end{aligned}$$



Robótica FCE

BUAP

Ejemplo 6.3:

Diseñe un programa en **MATLAB** para:

- Estimar por el método de Euler la velocidad o derivada con respecto al tiempo de la señal $x(t)$ del sistema $\dot{x}(t) = -ax(t)$.
- Implementar el algoritmo de integración numérica trapezoidal aplicado a la velocidad del sistema $\dot{x}$ para verificar que el resultado concuerda con la posición x.

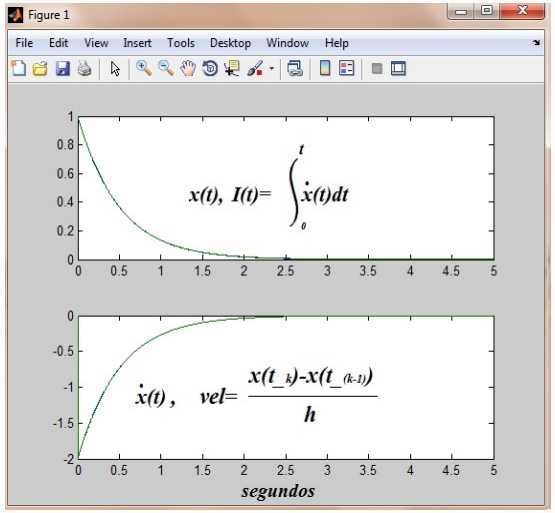


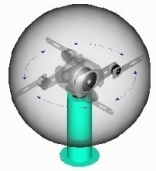
Figura 10: Diferenciación e integración numérica.

Código MATLAB 6: ejemplo11

Robótica, Período de primavera (16 de enero de 2020). Licenciatura en Ingeniería Mecatrónica.
Dr. Fernando Reyes Cortés. Facultad de Ciencias de la Electrónica, BUAP.

ejemplo11.m — MATLAB 2019a

```
1  clc;  clear all;  close all; format short
2   %tiempo de simulación (segundos)
3  ti=0;  h=0.001;  tfinal = 5;  ts=ti:h:tfinal;%vector tiempo
4  opciones=odeset('RelTol',1e-06,'InitialStep',h,'MaxStep',h);
5  x0=1;%condición inicial
6  [t, x]=ode45('ejemplo9',ts, x0, opciones);%Runge-Kutta 4/5
7  c=1; y=c*x;% salida del sistema y=c x
8  xp =ejemplo9(t,x);%señal de velocidad ideal
9  [n, m]=size(x); vel=zeros(n,m); I=zeros(n,m);
10 I(1)=x0;%condición inicial para realizar la integración numérica trapezoidal
11 for k=2:n
12     vel(k)=(x(k)-x(k-1))/(h);% Método de Euler
13     I(k)=I(k-1)+h*xp(k);%integración numérica trapezoidal o método Euler
14 end
15 subplot(2,1,1); plot(t,x,t,I) %comparación gráfica entre x y I = ∫_0^t ẋ(t)dt
16 subplot(2,1,2); plot(t,xp,t,vel) %comparación gráfica entre ẋ(t) y vel(t) = (x(t_k)−x(t_{k−1}))/h
```



Robótica FCE

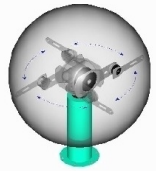
BUAP

Código MATLAB 7: simusso

Robótica, Período de primavera (16 de enero de 2020). Licenciatura en Ingeniería Mecatrónica.
Dr. Fernando Reyes Cortés. Facultad de Ciencias de la Electrónica, BUAP.

simusso.m — MATLAB 2019a

```
1  clc;
2  clear all;
3  close all;
4   % parámetros de simulación:
5  ti=0; tf = 10; h=0.001;
6   % intervalo de simulación
7  ts=ti:h:tf;
8  cond_iniciales=[0;0];
9  opciones=odeset('RelTol' , 1e-06, 'AbsTol' ,1e-06, 'InitialStep' ,h, 'MaxStep' ,h);
10 disp('Simulación de un sistema lineal de segundo orden' )
11 [t,x]=ode45('sso' ,ts,cond_iniciales,opciones);
12 figure
13 subplot(2,1,1); plot(t, x(:,1))
14 subplot(2,1,2); plot(t, x(:,2))
```



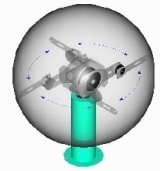
$$G(s) = \frac{w_n^2}{s^2 + 2\rho w_n s + w_n^2} \Rightarrow \dot{x} = \begin{bmatrix} 0 & 1 \\ -w_n^2 & -2\rho w_n \end{bmatrix} x + \begin{bmatrix} 0 \\ w_n^2 \end{bmatrix} u \quad y = \begin{bmatrix} 1 & 0 \end{bmatrix} x$$

Código MATLAB 8: sso

Robótica, Período de primavera (16 de enero de 2020). Licenciatura en Ingeniería Mecatrónica.
Dr. Fernando Reyes Cortés. Facultad de Ciencias de la Electrónica, BUAP.

sso.m — MATLAB 2019a

```
function xp =sso(t,x)
    %frecuencia natural de resonancia w_n
    w_n=1;
    % factor de amortiguamiento rho
    rho=0.35;
    %Matriz A
    A=[ 0, 1;
    -w_n^2, -2*rho*w_n];
    %Matriz B
    B=[0;
        w_n*w_n];
    %señal de entrada
    u=1;%sistema dinámico lineal
    xp=A*x+B*u;
end
```



Robótica FCE

BUAP