# Trabajo Práctico Introductorio
# Matlab, Simulink y Métodos de Integración Numérica

Control I – Dinámica de los Sistemas Físicos

## 1. Introducción

Los sitemas continuos habitualmente se representan mediante ecuaciones diferenciales ordinarias (EDOs). Por lo tanto, si se quiere predecir el comportamiento de dichos sistemas ante distintas entradas y/o condiciones iniciales, se deberán resolver las ecuaciones diferenciales en cuestión.

Desafortunadamente, a excepción de los casos lineales y algunos casos no lineales triviales, las EDOs carecen de solución analítica. Inclusive en los pocos casos en los que se puede encontrar una solución, el procedimiento suele ser bastante engorroso y el resultado suele ser una expresión por demás de compleja.

Por este motivo, se recurre casi siempre al uso de distintos algoritmos numéricos que permiten obtener una solución aproximada de la EDO en distintos instantes de tiempo. Estos algoritmos, denominados *métodos de integración de ecuaciones diferenciales* se suelen implementar en distintas herramientas de software, algunas de propósito general (como Matlab, Scilab, etc.) o bien para dominios específicos (el PSpice por ejemplo para simular circuitos).

En este Trabajo Práctico utilizaremos algunos métodos de integración numérica muy conocidos y exploraremos algunas de sus características. Como herramienta utilizaremos primero Matlab y luego el entorno gráfico de simulación que provee dicho software, llamado Simulink.

El objetivo del trabajo es familiarizarse tanto con los algoritmos de simulación como con las herramientas Matlab/Simulink.

## 2. Simulación de Sistemas Dinámicos con Matlab

Comenzaremos este trabajo práctico realizando algunas simulaciones de sistemas sencillos utilizando Matlab como herramienta de programación, de análisis y de graficación de los resultados.

### 2.1. Sistema Elemental de Primer Orden – Euler

La siguiente *Ecuación Diferencial Ordinaria* puede representar, por ejemplo, la dinámica de un circuito RL serie donde $R = 1\Omega$, $L = 1Hy$ con una fuente de tensión de $1V$:

$$\dot{x}(t) = 1 - x(t) \tag{1}$$

Si consideramos una condición inicial nula $x(0) = 0$, podemos resolver muy fácilmente la ecuación, obteniendo la solución analítica:

$$x_a(t) = 1 - e^{-t} \tag{2}$$

Si bien en este caso es muy sencillo obtener la solución, esto en general no es posible y se debe recurrir a los *Métodos de Integración Numérica*.

El más simple de estos algoritmos es el *Método de Euler*. Dado un sistema de la forma:

$$\dot{x}(t) = f(t, x) \tag{3}$$

con condición inicial conocida $x(t_0) = x_0$, el método de Euler brindará una solución numérica según la fórmula:

$$x(t_{k+1}) = x(t_k) + h \cdot f(t_k, x(t_k))$$

donde $h \triangleq t_{k+1} - t_k$ se denomina *paso de integración*. Normalmente se suelen obviar los argumentos $t_k$ y se reescribe:

$$x_{k+1} = x_k + h \cdot f(t_k, x_k) \tag{4}$$

Para el sistema de la Ec.(1), teniendo en cuenta que $f(t_k, x_k) = 1 - x_k$,el método de Euler calculará la secuencia de valores según:

$$x_{k+1} = x_k + h \cdot (1 - x_k)$$

Los cálculos normalmente se realizan utilizando un programa en una computadora. En Matlab, la función que calcula esta solución puede programarse como sigue:

```
function [t,x]=ejemplo1(h)
%resuelve dx/dt=1-x
%entre t=0 y t=10 para x0=0
%utilizando el método de Euler
    x=0;
    t=0;
    N=10/h;
    for k=1:N
        t(k+1)=t(k)+h;
        x(k+1)=x(k)+h*(1-x(k));
    end
end
```

Luego, invocando desde la línea de comando por ejemplo

```
[t,x]=ejemplo1(0.1);
```

podemos obtener la solución brindada por el método de Euler, de forma que la variable $t$ contenga el tiempo y $x$ los valores de la solución.

Luego, ejecutando

```
plot(t,x)
```

podremos ver graficada la solución.

Una vez obtenida la solución numérica, podemos también calcular la solución analítica para los mismos instantes de tiempo ejecutando:

```
xa=1-exp(-t);
```

Se pide entonces:

**Ejercicio** 2.1.1 Programar la rutina que resuelve con Euler el sistema de la Ec.(1).

**Ejercicio** 2.1.2 Obtener la solución para los pasos de integración $h = 0.01$, $h = 0.1$ y $h = 1$ y comparar cada solución con la solución analítica graficando el error.

**Ejercicio** 2.1.3 Analizar que pasa cuando se hace $h = 2$ y $h = 3$.

## 2.2. Método de Heun

El método de Euler realiza una aproximación de primer orden. Una alternativa más precisa es el método de Heun, cuya fórmula es:

$$\begin{aligned} k_1 &= f(t_k, x_k) \\ k_2 &= f(t_k, x_k + h \cdot k_1) \\ x_{k+1} &= x_k + \frac{h}{2} \cdot (k_1 + k_2) \end{aligned} \tag{5}$$

Para el sistema de la Ec.(1), la rutina que calcula la solución con este método puede ser la que sigue:

```
function [t,x]=ejemplo2(h)
%resuelve dx/dt=1-x
%entre t=0 y t=10 para x0=0
%utilizando el método de Heun
    x=0;
    t=0;
```

```
    N=10/h;
    for k=1:N
        t(k+1)=t(k)+h;
        k1=1-x(k);
        k2=1-(x(k)+h*k1);
        x(k+1)=x(k)+h/2*(k1+k2);
    end
end
```

Se pide entonces:

**Ejercicio** 2.2.1 Programar la rutina anterior y repetir el Ejercicio 2.1.2 usando Heun.

**Ejercicio** 2.2.2 Comparar el error obtenido con Heun y con Euler y explicar la diferencia.

**Ejercicio** 2.2.3 Repetir el ejercicio 2.1.3 y comparar.

## 2.3. Método de Backward Euler

Los métodos de Euler y Heun son explícitos. Por cuestiones de estabilidad se suelen usar en ciertos casos métodos implícitos como el de Backward Euler, cuya fórmula es:

$$x_{k+1} = x_k + h \cdot f(t_{k+1}, x_{k+1}) \tag{6}$$

Si bien este método requiere iteraciones para resolver la ecuación implícita, en el caso del sistema de la Ec.(1) dicha ecuación se puede resolver.

En este caso, la rutina que calcula la solución del sistema con Backward Euler puede ser la que sigue:

```
function [t,x]=ejemplo3(h)
%resuelve dx/dt=1-x
%entre t=0 y t=10 para x0=0
%utilizando el método de Backward Euler
    x=0;
    t=0;
    N=10/h;
    for k=1:N
        t(k+1)=t(k)+h;
        x(k+1)=(x(k)+h)/(1+h);
    end
end
```

Se pide entonces:

**Ejercicio** 2.3.1 Programar la rutina anterior y repetir el Ejercicio 2.1.2 usando Backward Euler.

**Ejercicio** 2.3.2 Comparar los errores obtenidos con Heun y con Euler y explicar la diferencia.

**Ejercicio** 2.3.3 Repetir con Backward Euler el Ejercicio 2.1.3 y comparar.

## 2.4. Métodos de Integración para EDOs Generales

En los ejemplos anteriores, programamos las rutinas para simular un ejemplo particular. Esto, salvo para aplicaciones específicas, no es muy práctico.

Usualmente, las rutinas que implementan los distintos métodos numéricos se programan de manera genérica, y los sistemas se programan en rutinas separadas.

Por ejemplo, la siguiente rutina de Matlab implementa el método de Euler de manera genérica:

```
function [t,x]=euler1(f,x0,tf,h)
%resuelve dx/dt=f(t,x)
%entre t=0 y t=tf para x(0)=x0
%utilizando el método de Euler
    x=x0;
    t=0;
```

```
    N=tf/h;
    for k=1:N
        t(k+1)=t(k)+h;
        x(:,k+1)=x(:,k)+h*f(t(k),x(:,k));
    end
end
```

Para utilizar esta rutina, debemos invocarla diciendo cuál es el sistema que queremos simular. Si quisiéramos simular la ecuación[1]:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\sin(x_1) - 0.1 \cdot x_2 \end{aligned} \tag{7}$$

deberemos programarla como sigue:

```
function dx=pendulo(t,x)
  dx=[x(2);-sin(x(1))-0.1*x(2)];
end
```

Luego, para simular esta ecuación desde la condición inicial $x_0 = [1\ 0]^T$ podemos invocar desde Matlab:

```
 [t,x]=euler1(@pendulo,[1;0],50,0.1);
 plot(t,x)
```

Podemos hacer lo mismo para el método de Heun:

```
function [t,x]=heun2(f,x0,tf,h)
%resuelve dx/dt=f(t,x)
%entre t=0 y t=tf para x(0)=x0
%utilizando el método de Heun
    x=x0;
    t=0;
    N=tf/h;
    for k=1:N
        t(k+1)=t(k)+h;
        k1=f(t(k),x(:,k));
        k2=f(t(k),x(:,k)+h*k1);
        x(:,k+1)=x(:,k)+h/2*(k1+k2);
    end
end
```

o para cualquier otro método (aunque la programación de métodos implícitos es bastante más complicada).

En este punto, se pide:

**Ejercicio** 2.4.1 Programar las rutinas de Euler y Heun; y la función que calcula la ecuación del péndulo.

**Ejercicio** 2.4.2 Simular con pasos de integración $h = 0.01$, $h = 0.1$ y $h = 1$ con ambos métodos y comparar los resultados.

# 3. Simulación con Simulink

Expresar modelos mediante ecuaciones suele ser una tarea bastante complicada, particularmente cuando los modelos son complejos. Por este motivo, la mayor parte de las herramientas de simulación cuentan con interfaces que permiten representar los modelos en forma gráfica.

Uno de los lenguajes gráficos más utilizados para el modelado es el de los *Diagramas de Bloques*; y Matlab tiene una aplicación, denominada *Simulink*, que permite editar los modelos como Diagramas de Bloques y simularlos utilizando los distintos métodos numéricos ya programados en Matlab.

Además, Simulink posee una interfaz bastante sencilla que permite seleccionar los distintos métodos y sus parámetros para cada simulación.

En esta segunda parte continuaremos trabajando sobre los distintos métodos de integración numérica, pero utilizando ahora Simulink como herramienta.

[1] Esta ecuación podría representar, entre otras cosas, la dinámica de un péndulo con fricción

## 3.1. Métodos de Paso Fijo

El sistema de la Ec.(7) puede representarse en Simulink mediante el DB de la Figura 1. En dicho DB pueden verse además los bloques *To Workspace* que nos permitirán graficar los resultados de simulación en Matlab. Alternativamente pueden utilizarse bloques Scope para visualizar los resultados directamente en Simulink.

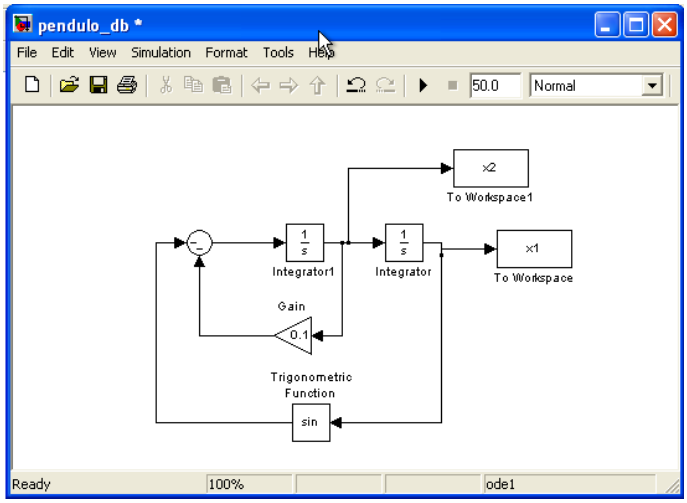


Figura 1: Diagrama de Bloques en Simulink del Péndulo

Para elegir los parámetros de simulación en tanto, se debe seleccionar la opción correspondiente, lo que abre la ventana de la Figura 2. En dicha figura, los parámetros fueron elegidos para utilizar el método de Euler, simulando con un paso fijo de 0.1 hasta un tiempo final de 50.

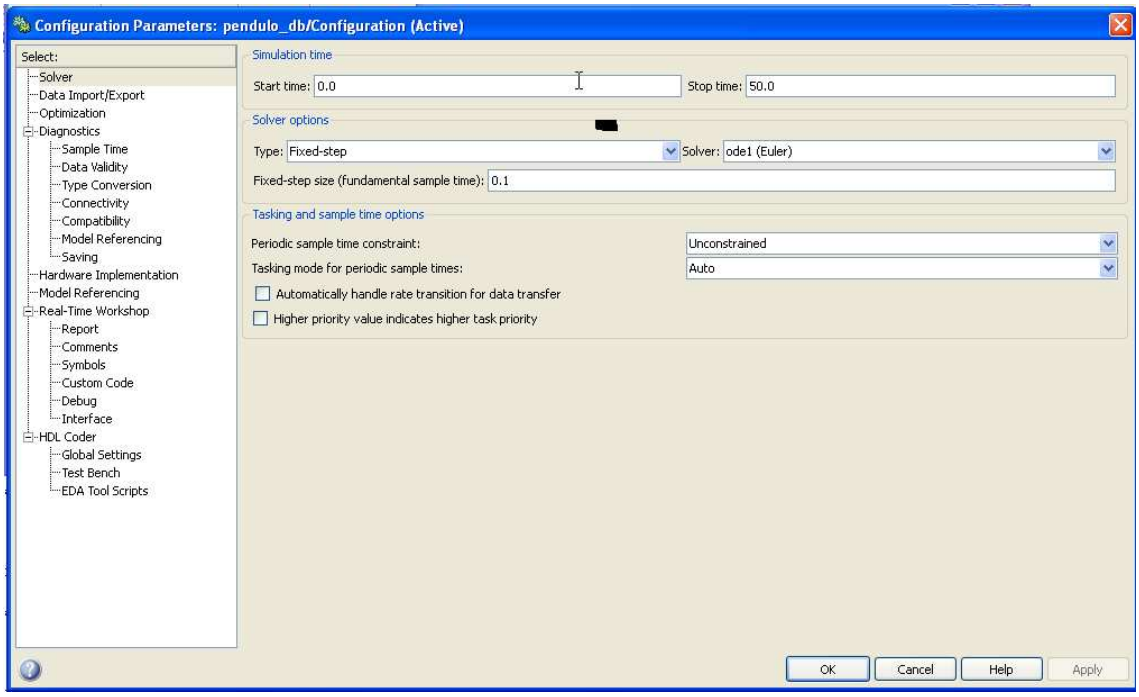


Figura 2: Ventana de Parámetros de Simulink.

Se pide entonces:

**Ejercicio** 3.1.1 Construir el DB del péndulo y simular con paso de integración $h = 0.1$ el sistema utilizando los métodos de Euler y Heun. Comparar los resultados con los del Ejercicio 2.4.2.

## 3.2. Métodos de Paso Variable

Uno de los problemas de los métodos numéricos es la elección adecuada del paso de integración $h$. Por este motivo, se han desarrollado *algoritmos de control de paso* que, en función de la tolerancia de error preestablecida, van ajustando el paso de integración a lo largo de la simulación.

En Simulink, los modelos pueden simularse utilizando diversos algoritmos de paso variable, eligiendo en cada caso distintos parámetros (tolerancia relativa y absoluta, paso inicial, entre otros).

Se pide entonces:

**Ejercicio** 3.2.1 Simular nuevamente el sistema del péndulo con el método 'ode45' (este algoritmo utiliza dos métodos de Runge–Kutta explícitos, uno de orden 4 y el otro de orden 5 para controlar el error). Utilizar tolerancias relativa y absoluta de 0.001.

**Ejercicio** 3.2.2 Observar cuantos pasos requirió la simulación. Para esto puede usarse el comando '$length(tout)$'.

**Ejercicio** 3.2.3 Graficar los resultados y en caso que hubiera pocos puntos, repetir la simulación con la opción de decimación. Esta opción permite interpolar puntos entre los puntos obtenidos.

**Ejercicio** 3.2.4 Repetir la simulación con el método 'ode15s' (este es un método implícito multipaso, basado en un algoritmo muy conocido llamado DASSL).

**Ejercicio** 3.2.5 Comparar lo obtenido con 'ode45' y 'ode15s'. ¿Que algoritmo conviene usar en este sistema?.

## 3.3. Sistemas Stiff

Los sistemas que tienen simultáneamente dinámica rápida y lenta se denominan stiff. Cuando son lineales, este hecho se refleja en la presencia de autovalores con sus partes reales muy distintas.

El siguiente sistema, que puede modelar un circuito serie RLC alimentado con una fuente de tensión constante en el cual[2] $L = C = 1$ y $R = 100$, es un ejemplo de sistema stiff:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -x_1 - 100 \cdot x_2 + 1 \end{aligned} \tag{8}$$

Se pide entonces:

**Ejercicio** 3.3.1 Calcular los autovalores[3] de la matriz de evolución del sistema de la Ec.(8) y determinar cuál es el máximo paso de integración que puede utilizar el método de Euler para obtener un resultado estable.

**Ejercicio** 3.3.2 Armar en Simulink un DB del sistema y simularlo hasta un tiempo final de 500 con el método 'ode45'. Observar el número de pasos, el resultado de la simulación y además graficar la evolución del paso $h$.

**Ejercicio** 3.3.3 Repetir el punto anterior utilizando ahora 'ode15s' y comparar con 'ode45'.

[2] En este caso estamos considerando parámetros admiensionales para simplificar el problema.

[3] El comando de Matlab '$eig(A)$' calcula los autovalores de la matriz $A$ en Matlab.