# Modelado y Simulación de Sistemas Dinámicos
# Práctica 3 - Métodos de Integración Numérica

Ernesto Kofman

**Problema 1. Absorción de un Fármaco**

La dinámica de la concentración de un fármaco que se absorve en el sistema digestivo puede modelarse mediante la ecuación diferencial

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

1. Obtener la solución analítica de la ecuación diferencial anterior a partir de la condición inicial $x(0) = x_0$.
2. Escribir una función en Octave `solfarmaco(a,x0,t)` que permita calcular la solución analítica de esta ecuación en los instantes de tiempo definidos por el arreglo `t`, a partir de la condición inicial `x0` y con el parámetro `a`.
3. Graficar la solución analítica de la ecuación para $a = 1$ y $x_0 = 1$.

**Problema 2. Solución analítica de sistemas LTI**

Escribir una función en Octave `ltiSolve(A,B,u,x0,t)` que permita obtener la solución analítica de un sistema LTI de la forma

$$\dot{\mathbf{x}}(t) = A\ \mathbf{x}(\mathbf{t}) + B\ \mathbf{u} \tag{2a}$$

para los instantes de tiempo definidos por el arreglo `t`, a partir de la condición inicial `x0` considerando la entrada $\mathbf{u}$ constante.

Tener en cuenta que la solución analítica de la Ec.(2a) está dada por

$$\mathbf{x}(t) = e^{A\ t}\ \mathbf{x}(0) + A^{-1}(e^{A\ t} - I)\ B\ \mathbf{u} \tag{2b}$$

Probar el funcionamiento de la función para los siguientes parámetros:

$$A = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix},\ B = \begin{bmatrix} 0 \\ 1 \end{bmatrix},\ \mathbf{x}(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix},\ \mathbf{u} = 1$$

**Problema 3. Sistema Masa–Resorte**

La dinámica de un sistema tipo *masa–resorte–amortiguador* se puede modelar mediante las siguiente ecuaciones de estado

$$\begin{aligned} \dot{x}_1(t) &= x_2 \\ \dot{x}_2(t) &= -\frac{k}{m}\ x_1(t) - bm\ x_2(t) + \frac{F(t)}{m} \end{aligned} \tag{3a}$$

donde $x_1(t)$ representa la posición de la masa y $x_2(t)$ la velocidad.

1. Utilizando la función `ltiSolve` del problema anterior, obtener y graficar la solución analítica para $m = k = b = 1$ y $F(t) = 1$.
2. Repetir el punto anterior con $b = 0$ y luego con $b = 10$.

**Problema 4. Método de Forward Euler**

El método de Forward Euler aproxima la solución de una Ecuación Diferencial Ordinaria (EDO)

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t) \tag{4a}$$

con la fórmula

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + h\ \mathbf{f}(\mathbf{x}(t_k), t_k) \tag{4b}$$

y puede implementarse mediante la siguiente función de Octave:

```
function [t,x]=feuler(f,x0,t0,tf,h)
  t=[t0:h:tf];
  x=zeros(length(x0),length(t));
  x(:,1)=x0;
  for k=1:length(t)-1
    x(:,k+1)=x(:,k)+h*f(x(:,k),t(k));
  endfor
endfunction
```

Implementar el método y probar su funcionamiento con el modelo del Problema 1 con parámetro $a = 1$ usando un paso $h = 0.1$ y simulando hasta un tiempo final que considere adecuado. Repetir para el sistema masa resorte del Problema 3 con $b = m = k = F = 1$ con un paso $h = 0.1$.

Para simular cada modelo deberá crearse una función en Octave tal que dados $x$ y $t$ devuelva la derivada $\dot{x}(t)$.

**Problema 5. Precisión del Método de Forward Euler**

Simular el sistema masa resorte Problema 3 con $m = b = k = F = 1$ utilizando el método de Forward Euler con paso $h = 0.1$ y con paso $h = 0.01$. Luego:

1. Evaluar para cada valor de $h$ el error cometido en el primer paso, comparando la solución numérica con la analítica. Observar como cambia el error con el tamaño del paso de integración $h$. Puede utilizarse el comando[1] `norm(x(:,2)-xa(:,2),1)` para evaluar el error en el primer paso.

2. Evaluar para cada valor de $h$ el error máximo a lo largo de toda la simulación observando como cambia este error con el tamaño del paso $h$. Puede usar para esto el comando `norm(x-xa,1)`.

**Problema 6. Estabilidad del Método de Forward Euler**

Considerar nuevamente el sistema masa-resorte del Problema 3:

1. Usando parámetros $m = k = b = F = 1$, obtener por prueba y error el máximo valor del paso de integración $h$ para el cual el método de Forward Euler preserva la estabilidad numérica.

2. Repetir el punto anterior tomando $b = 10$ y $b = 0$.

3. Analizar ahora de manera teórica el resultado obtenido en cada caso. Tener en cuenta que al usar Forward Euler en un sistema LTI, la ecuación diferencial original (2a) se transforma en una ecuación en diferencias

$$\mathbf{x}(k+1) = A_d\ \mathbf{x}(k) + B_d\ \mathbf{u}(k) \tag{6a}$$

   donde $A_d = I + h\ A$, con autovalores $\lambda_d = 1 + h\ \lambda$ (siendo $\lambda$ los autovalores de $A$). Para evaluar los autovalores de una matriz Octave cuenta con el comando `eig`.

**Problema 7. Método de Backward Euler** El método de Backward Euler aproxima la solución de la EDO (4a) con la fórmula

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + h\ \mathbf{f}(\mathbf{x}(t_{k+1}), t_{k+1}) \tag{7a}$$

y puede implementarse de manera simple (aunque ineficiente) mediante la siguiente función de Octave:

```
function [t,x]=beuler(f,x0,t0,tf,h)
  t=[t0:h:tf];
  x=zeros(length(x0),length(t));
  x(:,1)=x0;
 for k=1:length(t)-1
     F=@(xk1) xk1-x(:,k)-h*f(xk1,t(k)+h); %function que debe ser cero
     x(:,k+1)=fsolve(F,x(:,k));
  endfor
endfunction
```

[1] La norma 1 de una matriz da como resultado la suma de los valores absolutos de los elementos de la columna donde esta suma es máxima. En un vector columna, la norma 1 es simplemente la suma de los valores absolutos de los elementos.

Programar el método y repetir los problemas 5 y 6 usando Backward Euler en lugar de Forward Euler. Para el análisis teórico de estabilidad, usar el hecho que $A_d = (I - h\ A)^{-1}$.

**Problema 8. Método de Heun**

El método de Heun aproxima la solución de la EDO (4a) con la fórmula

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \frac{h}{2}\ (\mathbf{k}_1 + \mathbf{k}_2) \tag{8a}$$

donde

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{f}(\mathbf{x}(t_k), t_k) \\ \mathbf{k}_2 &= \mathbf{f}(\mathbf{x}(t_k) + h\ \mathbf{k}_1, t_k + h) \end{aligned} \tag{8b}$$

Implementar este método en Octave y y repetir los problemas 5 y 6 usando el método de Heun. Para el análisis teórico de estabilidad, usar el hecho que $A_d = I + h\ A + 0.5\ (h\ A)^2$.

**Problema 9. Regla Trapezoidal** La Regla Trapezoidal aproxima la solución de la EDO (4a) con la fórmula

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + 0.5\ h\ [\mathbf{f}(\mathbf{x}(t_k), t_k) + \mathbf{f}(\mathbf{x}(t_{k+1}), t_{k+1})] \tag{9a}$$

y puede implementarse de manera simple (aunque ineficiente) mediante la siguiente función de Octave:

```
function [t,x]=traprule(f,x0,t0,tf,h)
  t=[t0:h:tf];
  x=zeros(length(x0),length(t));
  x(:,1)=x0;
 for k=1:length(t)-1
    F=@(xk1) xk1-x(:,k)-0.5*h*(f(xk1,t(k)+h)+f(x(:,k),t(k)));
    x(:,k+1)=fsolve(F,x(:,k));
  endfor
endfunction
```

Programar el método y repetir los problemas 5 y 6 con la Regla Trapezoidal. Para el análisis teórico de estabilidad, usar el hecho que $A_d = (I - 0.5\ h\ A)^{-1}\ (I + 0.5\ h\ A)$.

**Problema 10. Método Explícito de Paso Variable** La siguiente función de Octave implementa un algoritmo que compara un paso de un método de Runge Kutta de orden 3 con un paso de Heun para estimar el error y ajustar automáticamente el paso de integración.

```
function [t,x]=rk23(f,x0,ti,tf,reltol,abstol)
 hmax=(tf-ti)/10;
 t=zeros(1,10000);
 x=zeros(length(x0),10000);
 t(1)=ti;
 x(:,1)=x0;
 k=1;
 h=1e-6*(tf-ti);
 while t(k)<tf
   if t(k)>tf-h
     h=tf-t(k);
   end
    k1=f(x(:,k),t(k));
    k2=f(x(:,k)+h*k1,t(k)+h);
    k3=f(x(:,k)+h*k1/4+h*k2/4,t(k)+h/2);
    xheun=x(:,k)+h*(k1/2+k2/2);
    xrk3=x(:,k)+h*(k1/6+k2/6+2*k3/3);
   err=norm(xrk3-xheun);
   maxerr=max(abstol, reltol*norm(xrk3));
```

```
    if err<maxerr
      x(:,k+1)=xrk3;
      t(:,k+1)=t(k)+h;
      k=k+1;
    end
    if err!=0
      h=0.8*h*(maxerr/err)^(1/3);
      if (h>hmax)
        h=hmax;
      end
    else
      h=hmax;
    end
  end
  t=t(1:k);
  x=x(:,1:k);
end
```

Programar el algoritmo brindado y simular con el mismo el sistema masa resorte del Problema 3 usando $b = 1$ y $b = 100$. Analizar en cada caso la evolución del tamaño del paso de integración $h$ y explicar lo que ocurre cuando $b = 100$.

**Problema 11. Método Implícito de Paso Variable** La siguiente función de Octave implementa un algoritmo que compara un paso de la regla trapezoidal con un paso de Heun para estimar el error y ajustar automáticamente el paso de integración.

```
function [t,x]=traprulevs(f,x0,ti,tf,reltol,abstol)
  hmax=(tf-ti)/10;
  t=zeros(1,10000);
  x=zeros(length(x0),10000);
  t(1)=ti;
  x(:,1)=x0;
  k=1;
  h=1e-6*(tf-ti);
  while t(k)<tf
    if t(k)>tf-h
      h=tf-t(k);
    end
     k1=f(x(:,k),t(k));
     k2=f(x(:,k)+h*k1,t(k)+h);
     xheun=x(:,k)+h*(k1/2+k2/2);
     F=@(xk1) xk1-x(:,k)-0.5*h*(f(xk1,t(k)+h)+f(x(:,k),t(k))); %function que debe ser cero
     xtrap=fsolve(F,x(:,k));
    err=norm(xtrap-xheun);
    maxerr=max(abstol, reltol*norm(xtrap));
    if err<maxerr
      x(:,k+1)=xtrap;
      t(:,k+1)=t(k)+h;
      k=k+1;
    end
    if err!=0
      h=0.8*h*(maxerr/err)^(1/3);
      if (h>hmax)
        h=hmax;
      end
    else
      h=hmax;
    end
```

```
  end
  t=t(1:k);
  x=x(:,1:k);
end
```

Programar el algoritmo brindado y simular con el mismo el sistema masa resorte del Problema 3 usando $b = 0$, $b = 1$ y $b = 100$. Analizar en cada caso la evolución del tamaño del paso de integración $h$ y comparar con lo obtenido en el problema anterior.