Universidad Nacional de Rosario

Facultad de Ciencias Exactas, Ingeniería y Agrimensura

Licenciatura en Ciencias de la Computación

Entrega 3

Modelado y Simulación de Sistemas Dinámicos

Arroyo Joaquín
Bolzan Francisco

Contents

1 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

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

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 $solfarmaco(a,\ x_0,\ t)$ 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 $x_0 = 1$

La ecuación diferencial es la siguiente

$$\dot{x}(t) = -a \cdot e^{-a\ t} \cdot x$$

La función de Octave es la siguiente

```
function x = solfarmaco(a, x0, t)
  x(1) = x0;
  for k = 1:t-1
    x(k+1) = x(k) + -a * e^(-a * k) * x0;
  endfor
end
```

y el resultado de la simulación

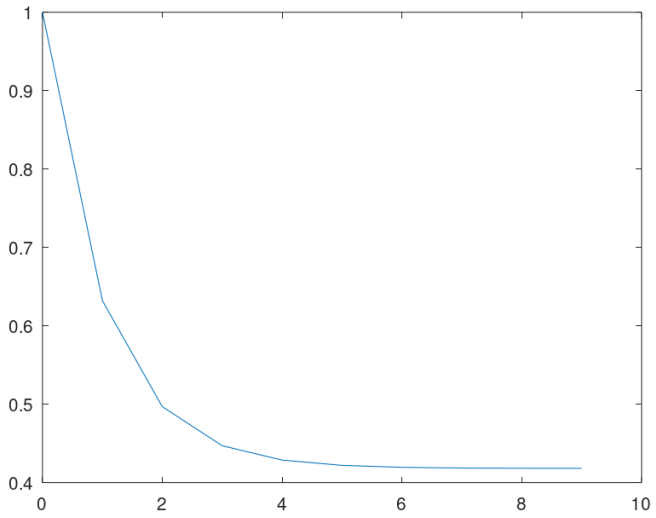

Figure 1: Simulación del modelo con $a = 1$, $x_0 = 1$ y $t = 100$.

2 Problema 2

Solución Analítica de Sistemas LTI. Escribir en Octave $ltiSolve(A, B, u, x_0, t)$ que permita obtener la solución analítica de un sistema LTI de la forma

$$\dot{\mathbf{x}}(t) = A \cdot \mathbf{x}(\mathbf{t}) + B \cdot \mathbf{u}$$

para los instantes de tiempo definidos por el arreglo $\mathbf{t}$, a partir de la condición inicial x_0 considerando la entrada $\mathbf{u}$ constante.

Tener en cuenta que la solución analítica de la ecuación está dada por

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

Probar el funcionamiento de la función para los siguientes parámetros:

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

El código de Octave es el siguiente

```
function x = ltiSolve(A, B, u, x0, t)
  x(:,1) = x0;
  for k = 1:t-1
    x(:,k+1) = expm(A * k) * x0 + inv(A) * (expm(A * k) - eye(2)) * B * u;
  endfor
end
```

y el resultado de la simulación

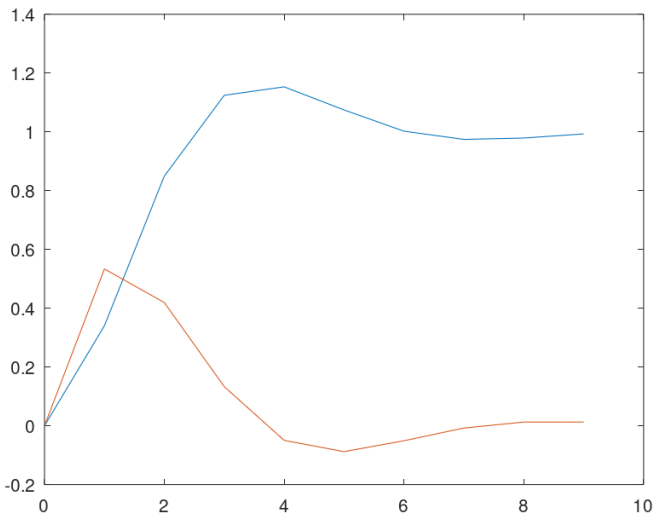

Figure 2: Simulación del modelo con los parámetros mencionados.

3 Problema 3

Sistema Masa-Resorte. La dinámica de un sistema tipo *masa-resorte-amortiguador* se puede modelar mediante las siguientes ecuaciones de estado

$$\dot{x}_1(t) = x_2$$
$$\dot{x}_2(t) = -\frac{k}{m}x_1(t) - \frac{b}{m}x_2(t) + \frac{F(t)}{m}$$

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

1. Utilizando la función *ltiSolve* del problema 2, 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 $b = 10$.

y los resultados de las simulaciones

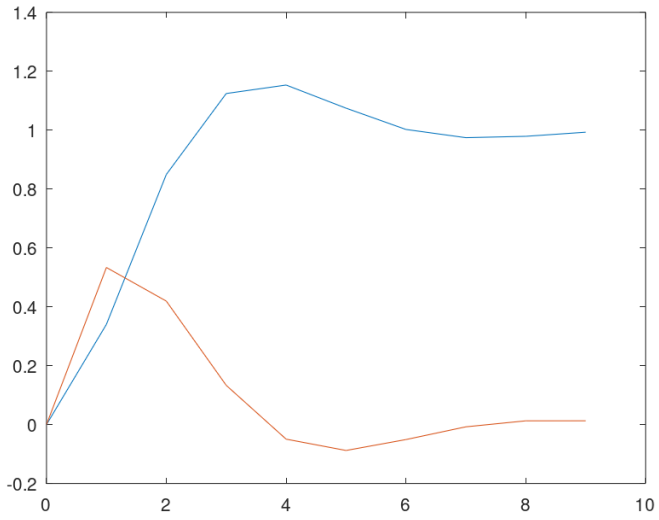

Figure 3: Simulación del modelo con $m = k = b = 1$ y $F(t) = 1$.

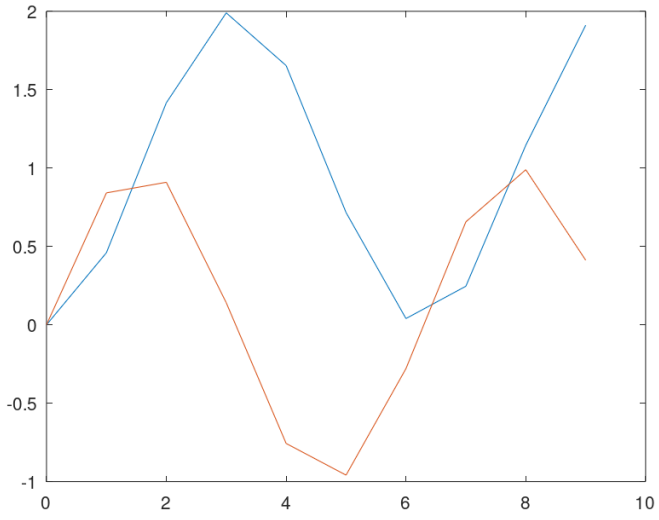

Figure 4: Simulación del modelo cambiando $b = 0$.

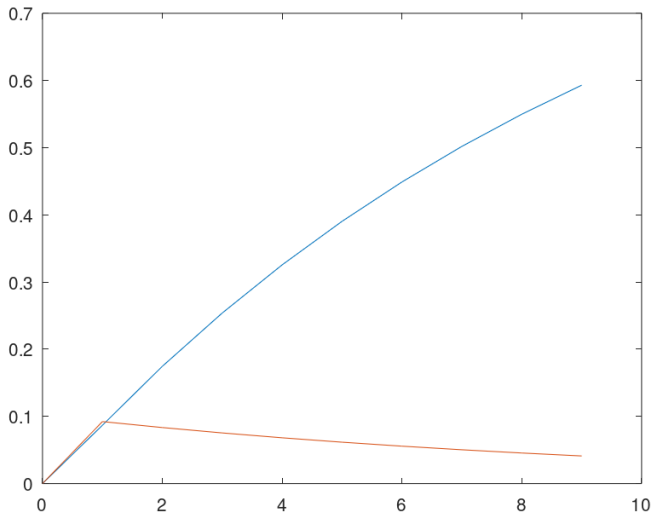

Figure 5: Simulación del modelo cambiando $b = 10$.

4 Problema 4

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

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

con la fórmula

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

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
end
```

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 $b = m = k = F = 1$ y 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 $\dot{x}(t)$.

La función que calcula las derivadas de las ecuaciones del problema 1 y 3 son:

```
function dx = solfarmaco_feuler(x, t)
  a = 1;
  dx = [-a * e^(-a * t) * t];
end
```

```
function dx = masares_feuler(x, t)
  b = 10;
  m = k = F = 1;
  dx = [x(2); 1/m*(-k*x(1) - b*x(2) +F)];
end
```

y los resultados de las simulaciones con este método y estas funciones son

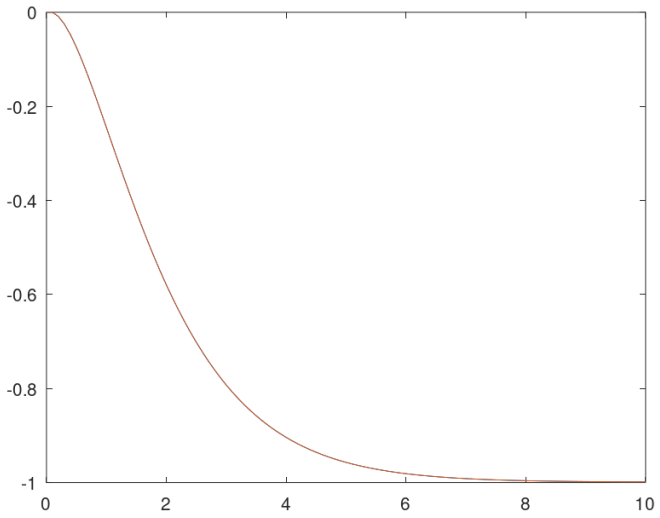

Figure 6: Simulación del modelo del problema 1 con FE.

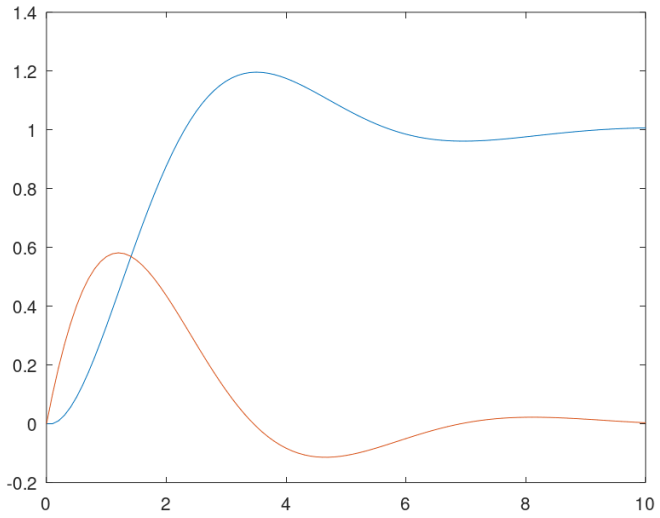

Figure 7: Simulación del modelo del problema 3 con FE.

5 Problema 5

Precisión del Método de Fordward 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 $norm(x(:,2) - xa(:,2), 1)$ para evaluar el error en el primer paso.

   Con $h = 0.01$ tenemos *error en primer paso* $= 3.9644e\text{-}3$ y con $h = 0.1$ tenemos *error en primer paso* $= 0.0.043$.

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 $norm(x - xa, 1)$.

   Con $h = 5.7726e\text{-}3$ tenemos *error máximo en toda la simulación* $= 0.2$ y con $h = 0.1$ tenemos *error máximo en toda la simulación* $= 0.06$.

Estos valores fueron obtenidos con el siguiente código

```
b = 1; % 0 y 10
m = k = F = 1;
A = [0, 1; -k/m, -b/m];
u = 1;
B = [0; F/m];
x0 = [0.1; 0.1];
t = 10;
xa = ltiSolve(A, B, u, x0, t);
h = 0.1;
[t,x] = feuler(@masares_feuler, [0;0], 0, 10, h);
x = x(:, 1:10:end - 1);
n = norm(x(:, 2) - xa(:, 2), 1); % Error en primer paso
nt = norm(x - xa, 1); % Error maximo en toda la simulacion
```

6 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 se transforma en una ecuación en diferencias

$$\mathbf{x}(k+1) = A_d \cdot \mathbf{x}(k) + B_d \cdot \mathbf{u}(k)$$

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

Por prueba y error obtuvimos los siguientes resultados:

1. Para $b = 1$, el valor máximo para el cuál FE preserva la estabilidad numérica es $h < 1$.

2. Para $b = 0$ no encontramos valor para el cuál se preserve la estabilidad.

3. Y para $b = 10$, el valor máximo es $h \leq 0.2$.

Para el análisis teórico tenemos:

- Con $b = 1$

  $A = \begin{bmatrix} 0 & 0 \\ -1 & -1 \end{bmatrix}$

  cuyos autovalores son:

  $\lambda_1 = -\frac{1}{2} + i\frac{\sqrt{3}}{2} \quad \lambda_2 = -\frac{1}{2} - i\frac{\sqrt{3}}{2}$

  Luego, necesitamos que para λ_i con $i = 1, 2$ que $|1 + h \cdot \lambda_i| < 1$.

  Tomando a λ_1 tenemos:

  $|1 + h \cdot \lambda_1| < 1$

  $|1 + h \cdot (-\frac{1}{2} + i\frac{\sqrt{3}}{2})| < 1$

  $|1 + -h\frac{1}{2} + hi\frac{\sqrt{3}}{2})| < 1$

  $\sqrt{(1 - h\frac{1}{2})^2 + (hi\frac{\sqrt{3}}{2})^2} < 1$

  $\sqrt{1 - h + \frac{1}{4}h^2 - \frac{3}{4}h^2} < 1$

  $-h - \frac{1}{2}h^2 < 0$

  $-h \cdot (1 + \frac{1}{2}h) < 0$

  $1 + \frac{1}{2}h > 0$

  $h > -2$

  Tomando a λ_2 tenemos:

  $|1 + h \cdot \lambda_2| < 1$

  $|1 + h \cdot (-\frac{1}{2} - i\frac{\sqrt{3}}{2})| < 1$

  $|1 + -h\frac{1}{2} - hi\frac{\sqrt{3}}{2})| < 1$

  $\sqrt{(1 - h\frac{1}{2})^2 - (hi\frac{\sqrt{3}}{2})^2} < 1$

  $\sqrt{1 - h + \frac{1}{4}h^2 + \frac{3}{4}h^2} < 1$

  $-h + h^2 < 0$

  $h \cdot (-1 + h) < 0$

  $h < 1$

  Podemos concluir que $0 < h < 1$.

- Con $b = 0$

  $A = \begin{bmatrix} 0 & 0 \\ -1 & 0 \end{bmatrix}$

  cuyos autovalores son:

  $\lambda_1 = i \quad \lambda_2 = -i$

  Tomando a λ_2 tenemos:

  $|1 + h \cdot \lambda_2| < 1$

  $|1 - hi| < 1$

  $\sqrt{1 + h^2} < 1$

  $1 + h^2 < 1$

  $h < 0$

  Por lo que podemos concluir que no existe $h > 0$ que cumpla esto.

- Con $b = 10$

  $A = \begin{bmatrix} 0 & 0 \\ -1 & -10 \end{bmatrix}$

  cuyos autovalores son:

  $\lambda_1 = -5 + 2\sqrt{6}$ $\lambda_2 = -5 - 2\sqrt{6}$

  Análogo al primer punto.

7 Problema 7

Método de Backward Euler. El método de Backward Euler aproxima la solución de la EDO del **Problema 4** co la fórmula

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

y puede implementarse de manera simple (aunque ineficiente) mediante la siguiente función en 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); %funcion que debe ser cero
        x(:,k+1)=fsolve(F,x(:,k));
    end
end
```

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 $A_d = (I - h \cdot A)^{-1}$.

- Con $h = 0.1$ tuvimos *error en primer paso* $= 0.036$ y *error máximo simulación* $= 0.053$.
- Con $h = 0.01$ tuvimos *error en primer paso* $= 3.8950e - 3$ y *error máximo simulación* $= 5.6865e - 3$.

Luego, con $b = 1 = 0 = 10$ no encontramos valores máximos de h para el cuál se pierde la estabilidad.

Para el análisis teórico tenemos:

$A_d = (I - h \cdot A)^{-1} \implies \mu_i = 1/(1 - h \cdot \lambda_i)$

Siguiendo el razonamiento planteado en el ejercicio 6 y para los mismos autovalores λ_i de A tenemos que para los autovalores μ_i de A_d vemos que dichos autovalores decrecen a medida que h crece, por lo que no hay máximo de h.

8 Problema 8

Método de Heun. El método de Heun aproxima la solución de la EDO del **Problema 4** con la fórmula

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \tfrac{h}{2}\ (\mathbf{k}_1 + \mathbf{k}_2)$$

donde

$$\mathbf{k}_1 = \mathbf{f}(\mathbf{x}(t_k),\ t_k)$$
$$\mathbf{k}_2 = \mathbf{f}(\mathbf{x}(t_k) + h \cdot \mathbf{k}_1,\ t_k + h)$$

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

- Con $h = 0.1$ tuvimos *error en primer paso* $= 2.0339e-3$ y *error máximo simulación* $= 2.0690e$-3.
- Con $h = 0.01$ tuvimos *error en primer paso* $= 1.9937e$-5 y *error máximo simulación* $= 1.9937e$-5.

Luego,

- Con $b = 1$, el valor máximo para el cuál se preserva la estabilidad numérica es $h < 2$.
- Con $b = 0$ no encontramos valor de h.
- Y con $b = 10$ el valor máximo es $h \leq 0.2$

Para el análisis teórico tenemos:
$A_d = I + h \cdot A + 0.5\ (h \cdot A)^2 \implies \mu_i = 1 + (h \cdot \lambda_i) + (0,5 \cdot h^2 \cdot \lambda_i^2)$
Siguiendo el razonamiento planteado en el ejercicio 6, para los autovalores μ_i de A_d confirmamos los resultados obtenidos.
Para el análisis teórico tenemos:

- Con $b = 1$
  $A = \begin{bmatrix} 0 & 0 \\ -1 & -1 \end{bmatrix}$
  cuyos autovalores son:
  $\lambda_1 = -\frac{1}{2} + i\frac{\sqrt{3}}{2} \quad \lambda_2 = -\frac{1}{2} - i\frac{\sqrt{3}}{2}$
  Luego, necesitamos que para λ_i con $i = 1, 2$ que $|1 + (h \cdot \lambda_i) + (0.5 \cdot h^2 \cdot \lambda_i^2)| < 1$.

  Tomando a λ_1 tenemos:

  $|1 + (h \cdot \lambda_1) + (0.5 \cdot h^2 \cdot \lambda_1^2)| < 1$

  $|1 + (h \cdot (-\frac{1}{2} + i\frac{\sqrt{3}}{2})) + (0.5 \cdot h^2 \cdot (-\frac{1}{2} + i\frac{\sqrt{3}}{2})^2)| < 1$
  Despejando h obtenemos lo siguiente:
  $\sqrt{\frac{h^4}{4} - \frac{h^3}{2} + \frac{h^2}{2} - h + 1} < 1$
  $\frac{h^4}{4} - \frac{h^3}{2} + \frac{h^2}{2} - h < 0$
  Lo que nos da como solución $0 < h < 2$

  Tomando a λ_2 tenemos:

  Análogamente al punto anterior despejando h obtenemos lo siguiente:
  $\frac{h^4}{4} - \frac{h^3}{2} + \frac{h^2}{2} - h < 0$
  Lo que nos da como solución $0 < h < 2$

  Concluímos que el valor supuesto es correcto.

- Con $b = 0$ análogo al punto 2 del ejercicio 6.
- Con $b = 10$ análogo al primer punto de este ejercicio.

9 Problema 9

Regla Trapezoidal. La Regla Trapezoidal aproxima la solución de la EDO con la fórmula

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

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 $A_d = (I - 0.5 \cdot h \cdot A)^{-1} \cdot (I + 0.5 \cdot h \cdot A)$.

- Con $h = 0.1$ tuvimos *error en primer paso* = 9.9377e-4 y *error máximo simulación* = 9.9377e-4.
- Con $h = 0.01$ tuvimos *error en primer paso* = 9.9433e-6 y *error máximo simulación* = 9.9433e-6.

Luego, con $b = 1 = 0 = 10$ no encontramos valores máximos de h para el cuál se pierde la estabilidad.

Para el análisis teórico, siguiendo la misma lógica que en el ejercicio 7 y osbervando que:
$A_d = (I - 0.5 \cdot h \cdot A)^{-1} \cdot (I + 0.5 \cdot h \cdot A) \implies \mu_i = (1 + (0,5 \cdot h \cdot \lambda_i))/(1 - (0,5 \cdot h \cdot \lambda_i))$
Podemos ver que de nuevo, el autovalor decrece a medida que h crece.

10 Problema 10

Método Explícito de Paso Variable. La siguiente función de Octave implementa un algoritmo que compara un paso de un método 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 $b = 100$.

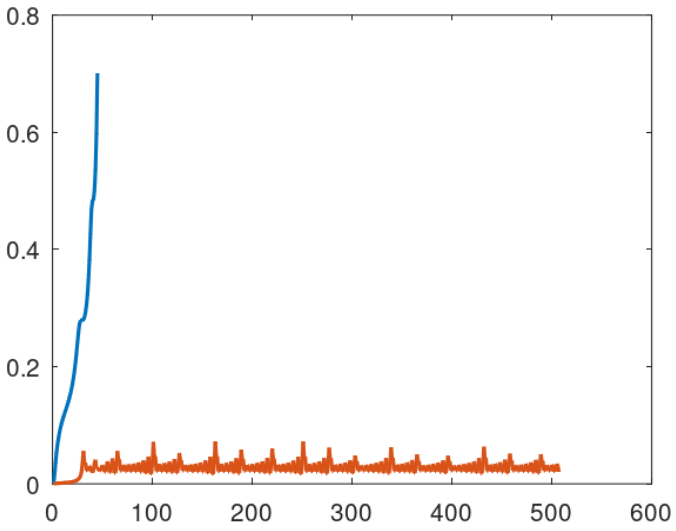

Figure 8: Evolución del valor de h para $b = 1$ y $b = 100$.

Podemos observar que para el caso $b = 100$ el h fluctúa constantemente, lo que nos indica que operó siempre sobre su límite, lo cual a su vez aumenta el número de pasos.

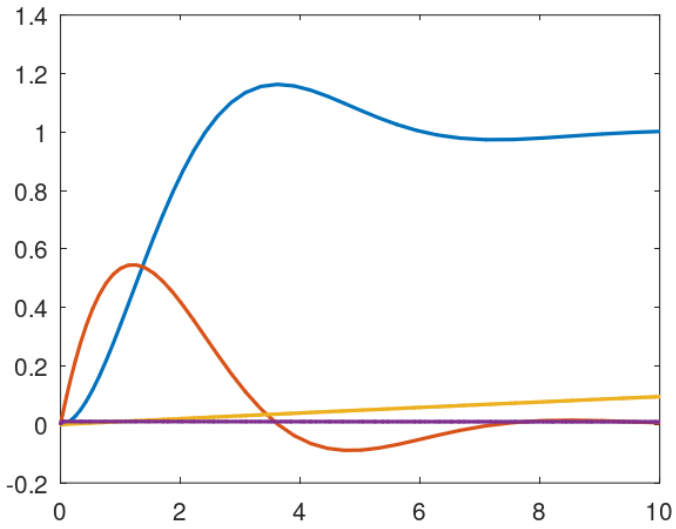

Figure 9: Valor de x para $b = 1$ y $b = 100$.

11 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.

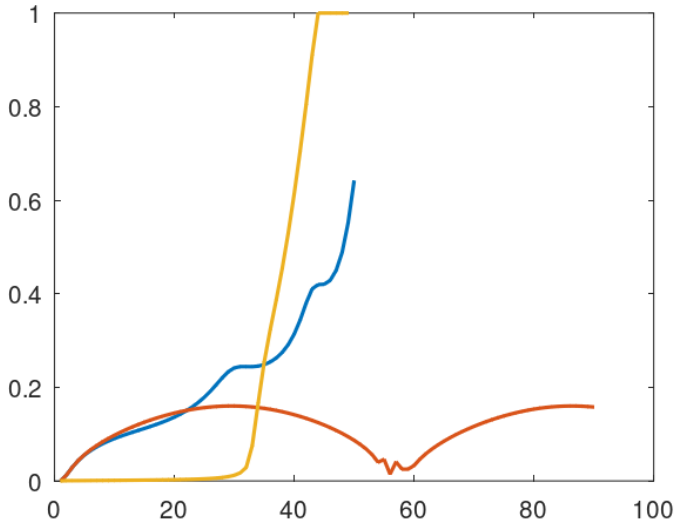

Figure 10: Evolución del valor de h para $b = 1$, $b = 0$ y $b = 100$.

Observamos que en este caso, ningún valor de h operó sobre el límite como en el ejercicio anterior, lo cual redujo significativamente la cantidad de pasos.

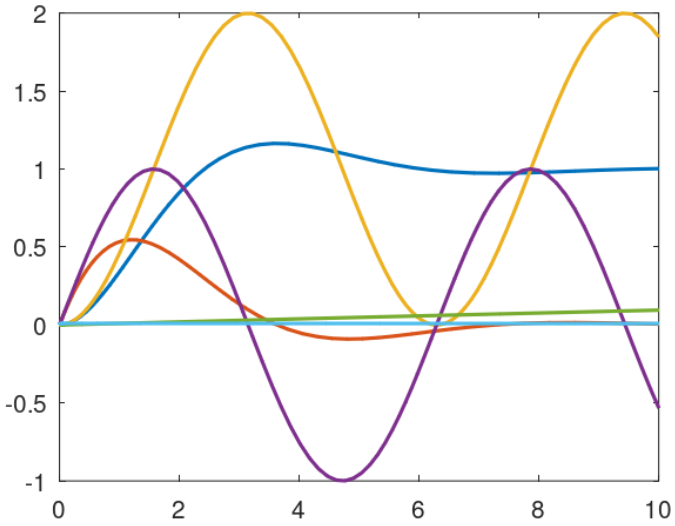

Figure 11: Valor de x para $b = 1$, $b = 0$ y $b = 100$.