Trabajo Práctico 1 – Simulación de Sistemas Continuos

Dinámica de los Sistemas Físicos

Año 2023

En este Trabajo Práctico trataremos distintos problemas vinculados a la Simulación de Sistemas Continuos. Más específicamente, analizaremos diferentes propiedades de los métodos de integración numérica para ecuaciones diferenciales ordinarias y trabajaremos sobre los algoritmos de causalización de sistemas de ecuaciones diferenciales algebraicas.

Antes de concurrir al laboratorio, como trabajo previo deberán realizarse completos el Problema 1, el Problema 7 y deberá resolverse el ítem 1 del Problema 2. Se recomienda también tener programadas las funciones del ítem 2 del Problema 3, del ítem 1 del Problema 6 y del ítem 2 del Problema 8

1. Convertidor Buck

El esquema de la Figura 1 representa un convertidor de continua en continua llamado circuito *buck* o *reductor*. El circuito cuenta con una llave que conmuta en alta frecuencia para producir una tensión de salida sobre la resistencia de carga R cuyo valor medio es una fracción de la tensión de entrada U. La relación entre las tensiones de entrada y salida la determina el *ciclo de trabajo*, es decir, la fracción de tiempo que la llave permanece cerrada.

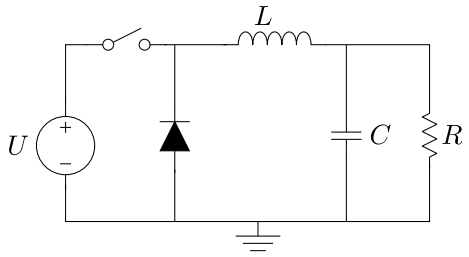

Figura 1: Circuito *Buck* o Reductor.

Trabajaremos modelando y simulando este circuito bajo diferentes hipótesis.

2. Análisis con un Modelo Simplificado

Si consideramos que la llave y el diodo son ideales y que el diodo no conduce cuando la llave está cerrada y es un conductor ideal con la llave está abierta, podemos plantear las siguientes ecuaciones para el modelo:

$$\begin{aligned} L\,\frac{\mathrm{d}i_L(t)}{\mathrm{d}t} - u_L(t) &= 0 \\ C\,\frac{\mathrm{d}u_C(t)}{\mathrm{d}t} - i_C(t) &= 0 \\ R\,i_R(t) - u_R(t) &= 0 \\ u_{SD}(t) - U\;s(t) &= 0 \\ i_{SD}(t) - i_L(t) &= 0 \\ u_{SD}(t) - u_L(t) - u_C(t) &= 0 \\ i_L(t) - i_C(t) - i_R(t) &= 0 \\ u_C(t) - u_R(t) &= 0 \end{aligned} \tag{1}$$

donde $u_{SD}(t)$ e $i_{SD}(t)$ representan la tensión y corriente a la izquierda de la inductancia y $s(t)$ es una señal de entrada conocida que vale 1 si la llave está cerrada y 0 si la llave está abierta.

En los distintos problemas consideraremos los siguientes valores para los parámetros: $U = 12\text{V}$, $L = 10^{-4}\text{Hy}$, $C = 10^{-4}\text{F}$ y $R = 10\Omega$.

Problema 1. Obtención de las Ecuaciones de Estado

A partir del sistema de DAEs de la Ec.(1),

1. Identificar las variables de estado y las variables algebraicas del modelo.
2. Eliminar las ecuaciones triviales y las variables algebraicas correspondientes.
3. Aplicar el algoritmo de ordenamiento y obtener un sistema explícito de Ecuaciones de Estado.

Problema 2. Solución Analítica de las Ecuaciones de Estado

1. Escribir las Ecuaciones de Estado obtenidas en el ítem 3 del Problema 1 en forma matricial:

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

2. Implementar en Octave o Matlab la función dada por el Código 1 que calcula la solución analítica de la Ec.(2) para distintos valores del tiempo t, a partir de una condición inicial $\mathbf{x}_0$ dada y considerando una entrada constante $u(t)$.

Código 1: Función de Matlab/Octave que calcula la solución analítica de la Ec.(2)

```
function x=solvess(A,B,u,x0,t)
  n=length(x0);
  N=length(t);
  I=eye(n,n);
  x=zeros(n,N);
  for k=1:N
    x(:,k)=expm(A*t(k))*x0 + inv(A)*(expm(A*t(k)) - I)*B*u;
  end
end
```

3. Considerando la entrada $s(t) = 1$ (llave cerrada) y condiciones iniciales nulas, obtener y graficar la solución analítica del problema para el tiempo entre 0 y 0.01 con un intervalo de 10^{-5} segundos. **Ayuda:** Programar un *script* donde se definan los parámetros, se calculen las matrices, la condición inicial x_0, el vector de tiempo `t=[0:1e-5:0.01]` y se invoque `xa=solvess(A,B,u,x0,t)`.

Problema 3. Método de Forward Euler

Se propone ahora simular usando Octave o Matlab el sistema utilizando el método de Forward Euler, para lo cual se pide lo siguiente:

1. Implementar el método de Forward Euler dado por el Código 2.

Código 2: Función de Matlab/Octave que implementa método de Forward Euler

```
function [t,x]=feuler(f,x0,h,t0,tf)
  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));
  end
end
```

2. Programar en Octave o Matlab una función que permita evaluar la derivada del vector de estados en función del estado y del tiempo. Considerar por el momento en dicha función que la llave está cerrada todo el tiempo. **Ayuda:** Puede usar como modelo la función dada por el Código 3 que evalua la derivada del vector de estados para un sistema masa-resorte.

Código 3: Función de Matlab/Octave con el modelo de un sistema Masa-Resorte

```
function dx=masares(x,t)
  m=1; b=1; k=1;  %parametros
  F=1;            %entrada
  der_x1=x(2);
  der_x2=-k/m*x(1)-b/m*x(2)+F;
  dx=[der_x1; der_x2];  %vector de derivadas
end
```

3. Simular el sistema hasta un tiempo final $tf = 0.01$ usando un paso de integración $h = 10^{-5}$. Puede usarse para esto el comando `[t,x]=feuler(@buck,x0,1e-5,0,0.01)`.

   a) Graficar la solución obtenida y compararla con la solución analítica del ítem 3.

   b) Calcular el error cometido tras el primer paso usando `norm(x(:,2)-xa(:,2))`.

4. Repetir usando un paso de integración $h = 10^{-6}$ y analizar que ocurre en términos de la estabilidad y del error tras un paso.

Problema 4. Método de Heun

Repetir el Problema 3 utilizando ahora el método de Heun, que puede implementarse mediante el Código 4. Utilizar en este caso pasos de integración $h = 10^{-4}$, $h = 2\ 10^{-5}$ y $h = 10^{-5}$. Analizar la estabilidad y cómo cambia el error tras un paso al variar h.

Código 4: Función de Matlab/Octave que implementa método de Heun

```
function [t,x]=heun(f,x0,h,t0,tf)
  t=[t0:h:tf];
  x=zeros(length(x0),length(t));
  x(:,1)=x0;
  for k=1:length(t)-1
    k1=f(x(:,k),t(k));
    k2=f(x(:,k)+h*k1,t(k+1));
    x(:,k+1)=x(:,k)+h/2*(k1+k2);
  end
end
```

Problema 5. Método de Runge Kutta con Control de Paso

Simular el modelo utilizando ahora un método de RK23 de paso variable, que puede implementarse mediante el Código 5.

Código 5: Función de Matlab/Octave que implementa método de RK23

```
function [t,x]=rk23(f,x0,t0,tf,rtol,atol)
  t=zeros(1,10000);
  x=zeros(length(x0),length(t));
  t(1)=t0;
  x(:,1)=x0;
  h=tf/1e6;  %paso inicial
  k=1;
  while t(k)<tf
    if t(k)+h>tf
      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/4*k1+h/4*k2,t(k)+h/2);
    xa=x(:,k)+h/2*k1+h/2*k2;
    xb=x(:,k)+h/6*k1+h/6*k2+2/3*h*k3;
```

```
    if norm(xb-xa)<max(rtol*norm(xb),atol)
      x(:,k+1)=xb;
      t(k+1)=t(k)+h;
      k=k+1;
    end
    er=norm(xb-xa);
    e0=max(rtol*norm(xb),atol);
    h=0.8*h*(e0/er)^(1/3);
  end
  x=x(:,1:k);
  t=t(1:k);
end
```

1. Utilizar tolerancias relativa y absoluta $r_{tol} = 10^{-3}$, $a_{tol} = 10^{-6}$. Graficar los resultados y calcular número de pasos que realiza el método (`length(t)`). Graficar también el tamaño del paso h (`plot(diff(t))`).

2. Repetir el punto anterior para tolerancias relativa y absoluta $r_{tol} = a_{tol} = 10^{-6}$.

Problema 6. Simulación del Modelo Conmutado

Consideraremos ahora que la llave conmuta con un período de $T = 10^{-4}$seg. y que el *ciclo de trabajo* es del 50 %. Para simular el modelo bajo esta suposición, se pide lo siguiente:

1. Modificar la función desarrollada en el ítem 2 del Problema 3 de manera que ahora la señal $s(t)$ cambia de acuerdo a las conmutaciones de la llave. **Ayuda:** Para generar la señal dentro de la función que calcula la derivada, puede usar el siguiente fragmento de código:

Código 6: Parte de una función de Matlab/Octave que calcula la conmutación de la llave.

```
function dx=buck1(x,t)
  T=1e-4;dc=0.5; %periodo y ciclo de trabajo
  r=rem(t,T); %resto de la division entera entre el tiempo y el periodo
  if r<dc*T
    s=1;
  else
    s=0;
  end
  ....  %continua
```

2. Simular este nuevo sistema usando el método de Heun con los siguientes pasos de integración: $h = 2\ 10^{-5}$, $h = 10^{-5}$ y $h = 10^{-6}$. Explicar lo que observa.

3. Simular utilizando RK23 con tolerancias relativa y absoluta $r_{tol} = 10^{-3}$ y $a_{tol} = 10^{-6}$. Graficar los resultados y la evolución del paso de integración.

4. Comparar los resultados con lo que se obtiene a partir del modelo de Modelica del Código 7 simulando con OpenModelica.

Código 7: Modelo de la Ec.(1) en Modelica

```
model buck
parameter Real T=1e-4,dc=0.5,L=1e-4,C=1e-4,R=10,U=12;
Real iL,uL,uC,iC,uR,iR,uSD,iSD,s;
equation
s=if rem(time,T)<dc*T then 1 else 0;
   L* der(iL)-uL=0;
   C* der(uC)-iC=0;
   R* iR-uR=0;
   uSD-U* s=0;
   iSD-iL=0;
```

```
    uSD-uL-uC=0;
    iL-iC-iR=0;
    uC-uR=0;
end buck;
```

5. Observar la corriente $i_{SD}(t)$, o su *alias* $i_L(t)$, y analizar que ocurre con la corriente por el diodo mientras la llave se encuentra abierta para tiempos entre $t = 3\ 10^{-4}$ y $t = 6\ 10^{-4}$. ¿Hay algún problema con las hipótesis simplificatorias que se hicieron para llegar a la Ec.(1)?

3. Análisis con un Modelo más Realista

Para considerar la posibilidad que tanto la llave como el diodo estén abiertos simultáneamente, es necesario modelar más adecuadamente el diodo. Podríamos en principio considerar ideales la llave y el diodo de manera que la corriente por los mismos sea nula al estar abiertos.

El problema con esta idea es que cuando ambos elementos estén abiertos, la corriente por la inductancia será nula y tendremos una *singularidad estructural*, ya que la corriente dejará de ser una variable de estado. Esto implicaría un cambio de estructura (y de orden) en el modelo, lo cuál complica tanto el análisis como las simulaciones.

Una opción más simple es considerar que tanto la llave como el diodo tienen asociada una pequeña resistencia de conducción R_{on} cuando están cerrados y una gran resistencia de corte R_{off} al estar abiertos. Con esta idea, podemos modificar el sistema de la Ec.(1) como sigue:

$$\begin{aligned}
L\,\frac{\mathrm{d}i_L(t)}{\mathrm{d}t} - u_L(t) &= 0\\
C\,\frac{\mathrm{d}u_C(t)}{\mathrm{d}t} - i_C(t) &= 0\\
R\ i_R(t) - u_R(t) &= 0\\
u_D(t) - R_D(i_D(t))\ i_D(t) &= 0\\
u_S(t) - R_S(s(t))\ i_S(t) &= 0\\
u_F(t) - U &= 0\\
i_D(t) + i_S(t) - i_L(t) &= 0\\
u_D(t) + u_L(t) + u_C(t) &= 0\\
i_S(t) - i_F(t) &= 0\\
u_F(t) - u_S(t) + u_D(t) &= 0\\
i_L(t) - i_C(t) - i_R(t) &= 0\\
u_C(t) - u_R(t) &= 0
\end{aligned} \tag{3}$$

donde

$$R_D(i_D) = \begin{cases} R_{\text{on}} & \text{si } i_D > 0 \\ R_{\text{off}} & \text{en otro caso} \end{cases} \qquad R_S(s) = \begin{cases} R_{\text{on}} & \text{si } s = 1 \\ R_{\text{off}} & \text{en otro caso} \end{cases} \tag{4}$$

Problema 7. Obtención de las Ecuaciones de Estado

1. Repetir el Problema 1 usando el modelo de la Ec.(3) y recurriendo al procedimiento de rasgado si apareciera algún lazo algebraico. Suponer que la resistencia del diodo es R_D y la de la llave R_S (por el momento ignorar posibles cambios de valor).

2. Escribir las Ecuaciones de Estado obtenidas en forma matricial para el caso $s = 0$ (llave abierta, $R_S = R_{\text{off}}$) considerando además negativa la corriente del diodo ($R_D = R_{\text{off}}$).

3. Calcular la posición de los autovalores de la matriz A suponiendo que los nuevos parámetros son $R_{\text{on}} = 10^{-5}\Omega$ y $R_{\text{off}} = 10^{5}\Omega$.

4. Determinar el máximo paso de integración que podría utilizarse para simular este sistema con el método de Forward Euler.

Problema 8. Simulación del Modelo Conmutado con Backward Euler

Consideraremos como en el Problema 6 que la llave conmuta con un período de $T = 10^{-4}$seg. y que el *ciclo de trabajo* es del 50 %. Para simular el sistema se pide lo siguiente:

1. Implementar el método de Backward Euler dado por el Código 8.

Código 8: Función de Matlab/Octave que implementa método de Borward Euler

```
function [t,x]=beuler(f,x0,h,t0,tf)
  t=[t0:h:tf];
  x=zeros(length(x0),length(t));
  x(:,1)=x0;
  for k=1:length(t)-1
    F=@(z)(z-x(:,k)-h*f(z,t(k)));  %funcion que debe ser 0
    x(:,k+1)=fsolve(F,x(:,k));
  end
end
```

2. Escribir una función que calcule la derivada de los estados a partir del modelo explícito obtenido en el Problema 7 pero teniendo en cuenta las conmutaciones de la llave y los cambios en el estado del diodo.**Ayuda:** Puede usar el siguiente fragmento de código:

Código 9: Parte de una función de Matlab/Octave que calcula la conmutación de la llave y el estado del diodo.

```
function dx=buck2(x,t)
  L=1e-4;C=1e-4;R=10;U=12;Ron=1e-5;Roff=1e5;   %parametros
  T=1e-4;dc=0.5;
  iL=x(1);
  uC=x(2);
 r=rem(t,T);  %resto de la division entera
  if r<dc*T
    s=1;
  else
    s=0;
  end
  if s>0.5
    RS=Ron;
  else
    RS=Roff;
  end
  RD=Ron;
   %suponemos inicialmente que el diodo conduce
  iD=(iL-U/RS)/(1+RD/RS);
   %ahora verificamos
  if iD<=0
    RD=Roff;
    %recalculamos la corriente con el valor correcto de RD
    iD=(iL-U/RS)/(1+RD/RS);
  end
   %....  sigue
```

3. Observar que ocurre al simular este modelo con el método de Forward Euler o con el método de Heun.

4. Observar que ocurre con RK23.

5. Simular el modelo utilizando Backward Euler con pasos de integración $h = 2\ 10^{-5}$, $h = 10^{-5}$ y $h = 10^{-6}$.

6. Comparar con el resultado que se obtiene usando el modelo de Modelica del Código 10 al simular con OpenModelica.

Código 10: Modelo de la Ec.(1) en Modelica

```
model buck2
parameter Real T=1e-4,dc=0.5,L=1e-4,C=1e-4,R=10,U=12;
parameter Real Ron=1e-5,Roff=1e5;
Real iL,uL,uC,iC,uR,iR,uD,iD,uS,iS,uF,iF,s;
equation
s=if rem(time,T)<dc*T then 1 else 0;
   L* der(iL)-uL=0;
   C* der(uC)-iC=0;
   R* iR-uR=0;
   uD=if iD>0 then Ron*iD else Roff*iD;
   uS=if s>0.5 then Ron*iS else Roff*iS;
   uF-U=0;
   iS+iD-iL=0;
   uD+uL+uC=0;
   iS-iF=0;
   uF-uS+uD=0;
   iL-iC-iR=0;
   uC-uR=0;
end buck2;
```