FACULTAD DE INGENIERIA

Universidad de Buenos Aires

Análisis numérico I - 75.12/95.04

Trabajo práctico N° 2

Sistemas de ecuaciones diferenciales ordinarias

Alumnos:

José Higuera — Padrón N° 100251 — jhiguera@fi.uba.ar

Diego Luna — Padrón N° 75451 — diegorluna@gmail.com

Juan Segundo Marquez — Padrón N° 100556 — segundoprez777@gmail.com

Juan Manuel Maschio — Padrón N° 100801 — juanmmaschio@gmail.com

Docentes:

Mag. Ing. Miryam Sassano

Ing. Ignacio Bello

Ing. Matías Payva

Lic. Andrés Porta

Ing. Ezequiel García

Ing. Ignacio Santiago Cerruti

6 de Diciembre de 2019

Índice

Índice de figuras

Índice de cuadros

1. Enunciado

1.1. Resumen enunciado

Este TP consiste en realizar una implementación del algoritmo Runge-Kutta de orden 4, para cuyo testeo se propuso resolver el sistema de ecuaciones diferenciales que se obtiene a partir de la conocida ecuación diferencial de segundo orden que se obtiene al plantear la oscilación de un péndulo:

$$\ddot{\theta} + \frac{b}{m} \cdot \dot{\theta} + \frac{g}{l} \cdot sin\,(\theta) = 0 \tag{1.1}$$

Planteando:

$$\theta = x_1$$
$$\dot{\theta} = x_2$$

Y reemplazando en la ecuación (1.1), se obtiene el sistema:

$$\begin{cases} \dot{x_1} = x_1 \\ \dot{x_2} = -\frac{b}{m} \cdot x_2 - \frac{g}{l} \cdot sin\,(x_1) \end{cases} \tag{1.2}$$

Además de lo anterior, se pide que se implemente el método de integración de Romberg, y se aplique al cálculo del área encerrada bajo la curva del módulo del desplazamiento del péndulo.

1.2. Resolución

Para la resolución de la parte de programación del trabajo práctico e implementar los algoritmos pedidos, decidimos usar **MATLAB**, mayormente por conocerlo previamente y la sencillez con la que se pueden escribir scripts que implementen los algoritmos. A pesar de que la resolución se realizó en **MATLAB**, se prestó atención a la compatibilidad con **Octave**, ya que la compatibilidad en los paquetes básicos es alta y con un poco de cuidado y algo de programación condicional se puede lograr que los scripts funcionen en ambos entornos. Todos los resultados numéricos se guardaron por código desde **MATLAB** en formato **"CSV"** y las imágenes se guardaron también por código en formato **"PNG"** y luego se incorporaron desde LaTeX, para los archivos **"CSV"** se usó el paquete de LaTeX **pgfplotstable**.

2. Implementación de los algoritmos

El script resuelve en principio los casos pedidos en el enunciado, y luego toma interactivamente con diálogos nuevos parámetros al usuario, los cuales valida y utiliza para resolver el nuevo sistema, en cada caso se grafica las funciones de posición (angulo) y su derivada (velocidad angular) y se calcula la integral del módulo de la posición (angulo) en el intervalo.

A continuación se listan los archivos de **MATLAB** y su función:

"tp2.m" (apéndice [A.2.1]): Script principal que se debe ejecutar para realizar todos los cálculos y generar los archivos de resultados.

"pendulum.m" (apéndice [A.2.2]): Función que invoca nuestra implementación de Runge-Kutta de orden 4 para el sistema del péndulo.

"rk4.m" (apéndice [A.2.3]): Función con nuestra implementación del algoritmo Runge-Kutta de orden 4 en forma genérica matricial.

"plot_solution.m" (apéndice [A.2.4]): Función que grafica el ángulo y la velocidad obtenidas.

"trapezcomp_rk4.m" (apéndice [A.2.6]): Función que implementa el método de integración de trapecios compuesto adaptado a usar Runge-Kutta 4 (usado en Romberg).

"romberg_rk4.m" (apéndice [A.2.5]): Función que implementa el método de integración de Romberg adaptado a usar Runge-Kutta de orden 4.

"f_rk4_num_sol.m" (apéndice [A.2.7]): Función auxiliar que se usa para generar un handle a función que lleva implicito el sistema de ecuaciones diferenciales a resolver por Runge Kutta de orden 4, se usa en el método adaptado de Romberg.

En el apéndice correspondiente (apéndice [A.2]) se incluye el código completo de cada archivo.

"salida.txt" (apéndice [B.2.1]): La salida del script principal, se captura automáticamente en **MATLAB** al ejecutar el script principal.

El método de Romberg se implementó de manera de llamar a Runge-Kutta de orden 4, se optimizó de modo que solo se hace una llamada, para lograr esto se aprovechó el hecho de que el método de trapecios compuesto que se llama en cada nivel de Romberg, tiene un paso que es la mitad que en el nivel anterior, por lo tanto solo se resuelve el sistema para el paso mas pequeño, que corresponde al último nivel, luego en cada nivel se llama a una versión adaptada de trapecios que saltea los puntos necesarios para el paso necesario (se saltean $2^{(p-k)}$ donde p es el nivel de Romberg, y k corresponde al nivel en que se está calculando).

El método de Romberg se implementó,así como está descripto en la bibliografía, usando interpolación de Richardson y el método de trapecios compuesto, con las modificaciones antes mencionadas.

El error del método de integración de Romberg ($R_{n,n}$), asumiendo que la función es suficientemente diferenciable, está en $O\left(h^{(2 \cdot p)}\right)$, donde h es el paso, que en el caso de esté método, vale en cada nivel $h = \frac{1}{2^n} \cdot (b - a)$, siendo a y b los límites de integración y n el nivel, con lo que se tiene un error que está en $O\left(\frac{1}{2^p} \cdot (b - a)\right)$ para el método de nivel p.

El nivel de Romberg usado fue de 15, así que se espera un error del orden de $\frac{1}{2^p} \cdot (b - a) = \frac{20-0}{2^{30}} = 2 \times 10^{-8}$.

2.1. Sobre los archivos de MATLAB y Octave

Hay algunas cosas a comentar sobre las diferencias entre **MATLAB** y **Octave**, como se comentó anteriormente, se logró la compatibilidad de ejecución entre los entornos, sin embargo las salidas no son completamente equivalentes, debido a limitaciones en **Octave**, las salidas gráficas no son completamente equivalentes, en particular **MATLAB** permite la generación de **DataTips**, cosa que **Octave** aún no soporta, otra cuestión quizás mas importante es la eficiencia en ejecución, en algunos casos los tiempo de ejecución en **Octave** se hacen demasiado largos si se usan arrays muy extensos, lo cual se mitigó usando compilación condicional. Otra cosa a mencionar que no hace a la funcionalidad directamente, pero si a la presentación, es que debido a limitaciones en ambos entornos respecto a la codificación de los archivos y soporte incompleto o inadecuado de **UNICODE** en la línea de comando de **Windows**, se producen problemas en las salidas con símbolos que no sean parte de Latin-1 (ISO 8859-1), en particular las palabras con tilde, esto se ve aún mas complicado porque **Windows** usa una variante (CP1252) que no es completamente compatible con Latin-1 y el hecho de que los entornos de **MATLAB** y **Octave** no se comportan consistentemente en **Windows** y sistemas tipo Unix como **Linux**, en Unix es prácticamente universal la codificación de **UNICODE**, **UTF-8**. **MATLAB** sigue la codificación del sistema operativo, mientras que **Octave** intenta usar **UTF-8** siempre, pero en **Windows** no es completo el soporte. El tema es complicado y no hace al trabajo práctico el lidiar con el mismo, dado que trabajamos mayormente con **MATLAB**, tanto en **Windows** como en **Linux**, y la mayoría usa **Windows**, se optó por dejar los archivos en **CP1252**, siendo esta la codificación usual. Se incluyen simplemente por comodidad dos scripts de **Python**, **"utf8.py"** y **"cp1252.py"**, que convierten la codificación de todos los archivos **".m"** a las respectivas codificaciones, de esa manera, según el sistema en que se ejecuten los scripts, se puede lograr una salida con codificación correcta.

3. Aproximación de la solución del sistema

Para aproximar el valor de las soluciones del sistema del péndulo se escribió una función específica (apéndice [A.2.2]) que declara la función del sistema, los parámetros, las condiciones iniciales, llama a nuestra implementación del algoritmo de Runge-Kutta de orden 4, y devuelve la aproximación de la solución, luego el script principal se encarga de llamar a las funciones que grafican lo pedido. La implementación del método de Runge-Kutta de orden 4 (apéndice [A.2.3]), se realizó en forma matricial, lo cual permitió un código muy simple y compacto, y conformando a los mismos parámetros que toman las funciones incluidas en **MATLAB** u **Ocatve**, como ser **ode23b**, o **ode23s**, lo cual nos permitió utilizar las mismas inicialmente para ver los resultados esperados de nuestra implementación, luego de implementada, fue solo cuestión de reemplazar las llamadas a las funciones del entorno por la nuestra.

4. Primer caso

En la tabla [4.1] se muestra los primeros 5 y los últimos 5 valores de las soluciones del sistema del péndulo, la gráfica de las soluciones se puede ver en la figura [4.1], para este primer caso pedido, los parámetros del sistema son:

$$
\begin{aligned}
m &= 1\text{kg} \\
l &= 1\text{m} \\
b &= 0\text{N s/m} \\
h &= 0{,}2\text{s} \\
\Theta_0 &= 30^\circ \\
\dot{\Theta}_0 &= 0^\circ/\text{s}
\end{aligned}
$$

Tiempo [s]	Θ [rad]	$\dot{\Theta}$ [rad/s]
0.0000000000	0.523598775598299	1.745329251994330
0.2000000000	0.745620216773293	0.440431939459606
0.4000000000	0.699895909006134	−0.864522486714337
0.6000000000	0.425024100861386	−1.794496851871270
0.8000000000	0.028800409838075	−2.042088184793265
19.2000000000	−0.003752922923670	−0.015796294529635
19.4000000000	−0.005874756070728	−0.004944537185192
19.6000000000	−0.005676915095603	0.006654642522778
19.8000000000	−0.003451874556377	0.014749462534838
20.0000000000	−0.000194859858457	0.016756510620638

Cuadro 4.1: Tabla de salida para el algoritmo de Runge-Kutta 4 para el primer caso.

Para este caso el valor obtenido para la integral del módulo de la posición por el método de Romberg es:

$$I = 6{,}65414962 \pm 2 \times 10^{-8}$$

Asumiendo que el error es el que corresponde al orden. Como se dijo anteriormente, el orden del error se determina como $\frac{1}{2^p} \cdot (b - a)$ siendo a y b los límites de integración y p el nivel de Romberg.

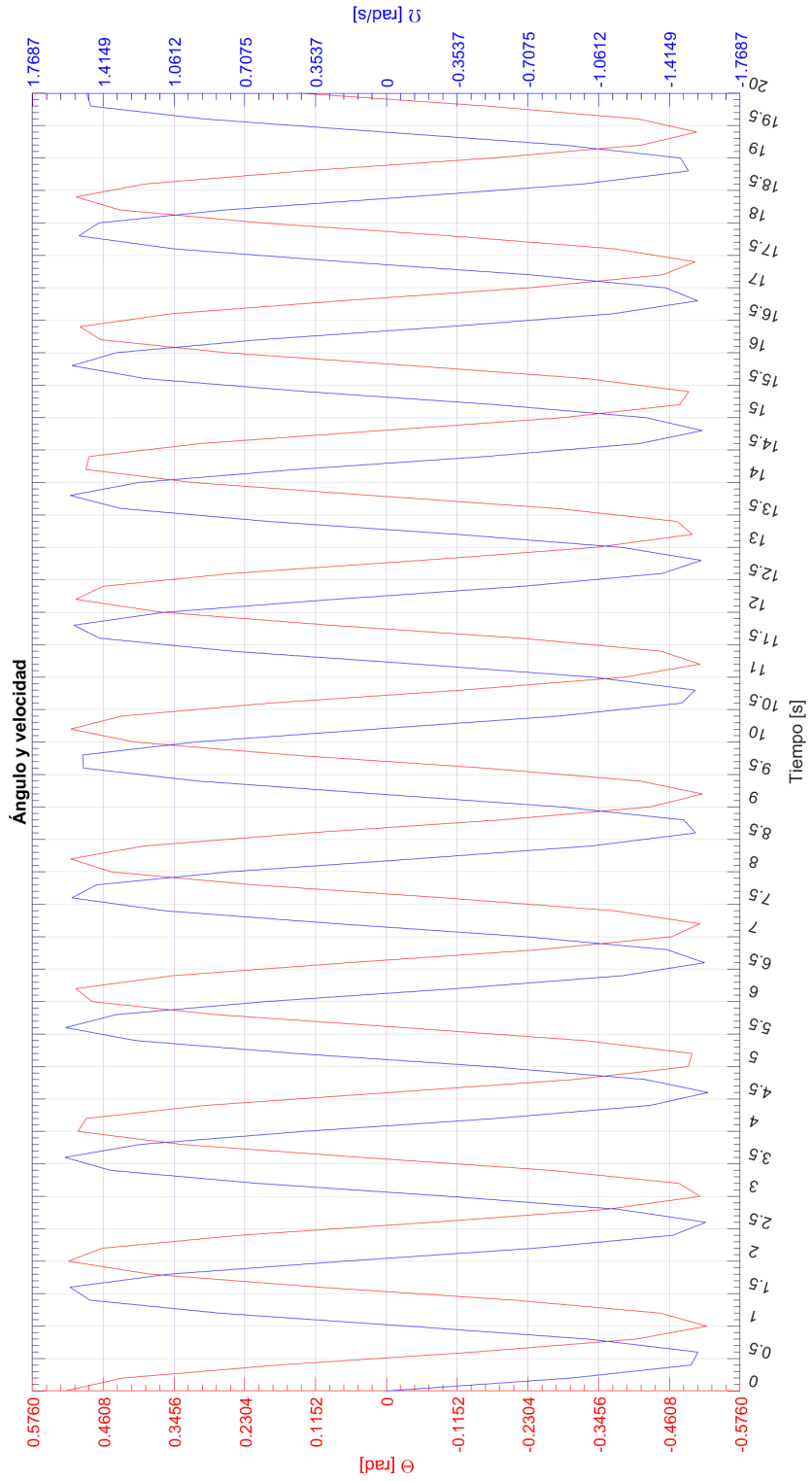

Figura 4.1: Gráfico de la solución para el primer caso.

5. Segundo caso

En la tabla [5.1] se muestra los primeros 5 y los últimos 5 valores de las soluciones del sistema del péndulo, la gráfica de las soluciones se puede ver en la figura [5.1], para este segundo caso pedido, los parámetros del sistema son:

$$
\begin{aligned}
m &= 1\text{kg} \\
l &= 1\text{m} \\
b &= 0{,}5\text{N s/m} \\
h &= 0{,}2\text{s} \\
\Theta_0 &= 30^\circ \\
\dot{\Theta}_0 &= 100^\circ/\text{s}
\end{aligned}
$$

Tiempo [s]	Θ [rad]	$\dot{\Theta}$ [rad/s]
0.0000000000	0.523598775598299	1.745329251994330
0.2000000000	0.745620216773293	0.440431939459606
0.4000000000	0.699895909006134	−0.864522486714337
0.6000000000	0.425024100861386	−1.794496851871270
0.8000000000	0.028800409838075	−2.042088184793265
19.2000000000	−0.003752922923670	−0.015796294529635
19.4000000000	−0.005874756070728	−0.004944537185192
19.6000000000	−0.005676915095603	0.006654642522778
19.8000000000	−0.003451874556377	0.014749462534838
20.0000000000	−0.000194859858457	0.016756510620638

Cuadro 5.1: Tabla de salida para el algoritmo de Runge-Kutta 4 para el primer caso.

Para este caso el valor obtenido para la integral del módulo de la posición por el método de Romberg es:

$$I = 2{,}10109933 \pm 2 \times 10^{-8}$$

Vale lo mismo dicho anteriormente para el error.

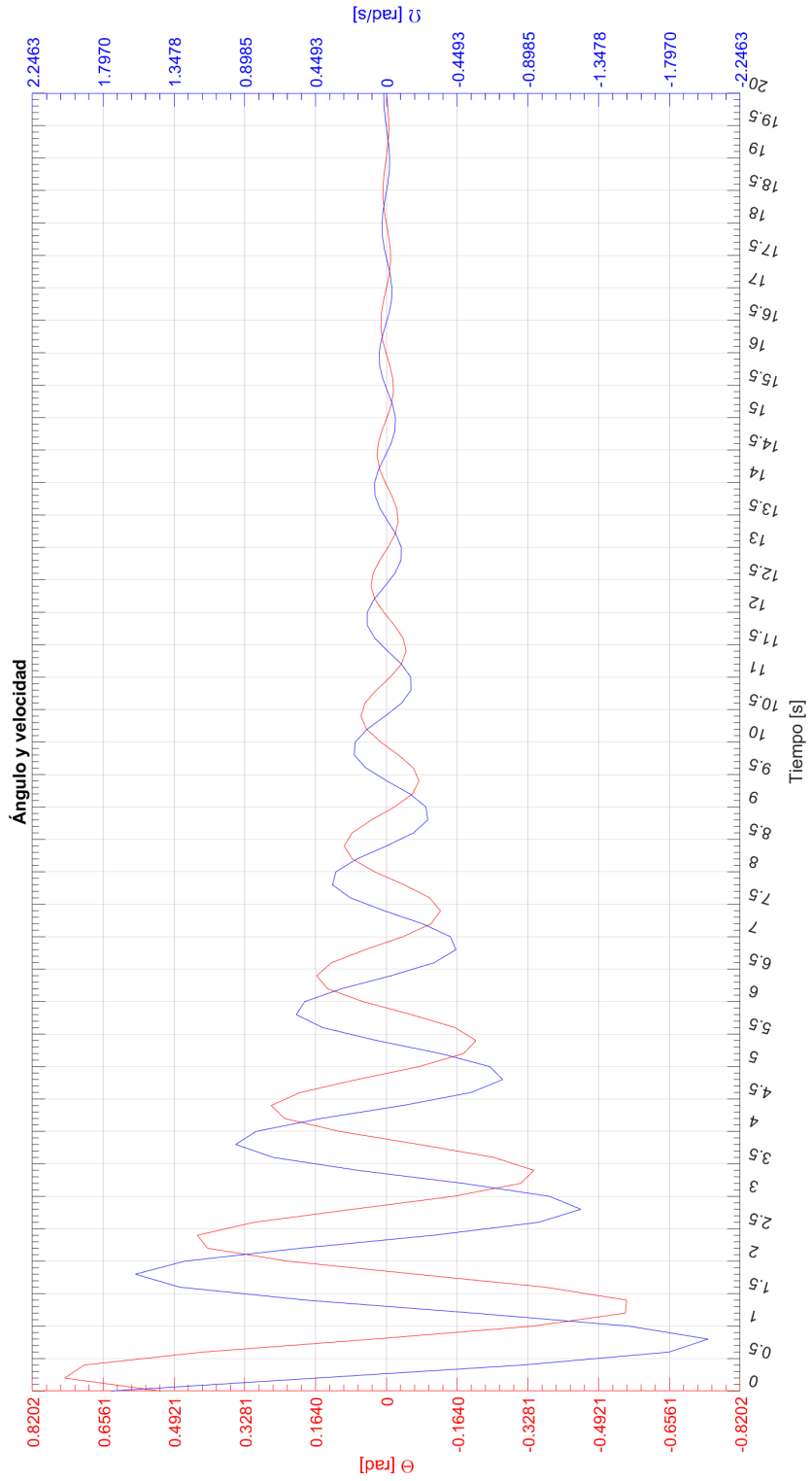

Figura 5.1: Gráfico de la solución para el primer caso.

6. Otros casos

En la figura [6.1] se muestra el gráfico de las soluciones del sistema del péndulo con los mismos parámetros que el segundo caso pedido, pero reduciendo el paso a $h = 0{,}001\text{s}$, se puede ver como se puede discernir mucho mejor la formas de las soluciones con esta mayor cantidad de puntos.
Finalmente en la figura [6.2] se puede ver las soluciones para el sistema para un caso en que no llegan casi a producirse oscilaciones, caso sobre-amortiguado, los parámetros para este caso son:

$$
\begin{aligned}
m &= 1\text{kg} \\
l &= 1\text{m} \\
b &= 5\text{N}\,\text{s/m} \\
h &= 0{,}001\text{s} \\
\Theta_0 &= 30^\circ \\
\dot{\Theta}_0 &= 0^\circ/\text{s}
\end{aligned}
$$

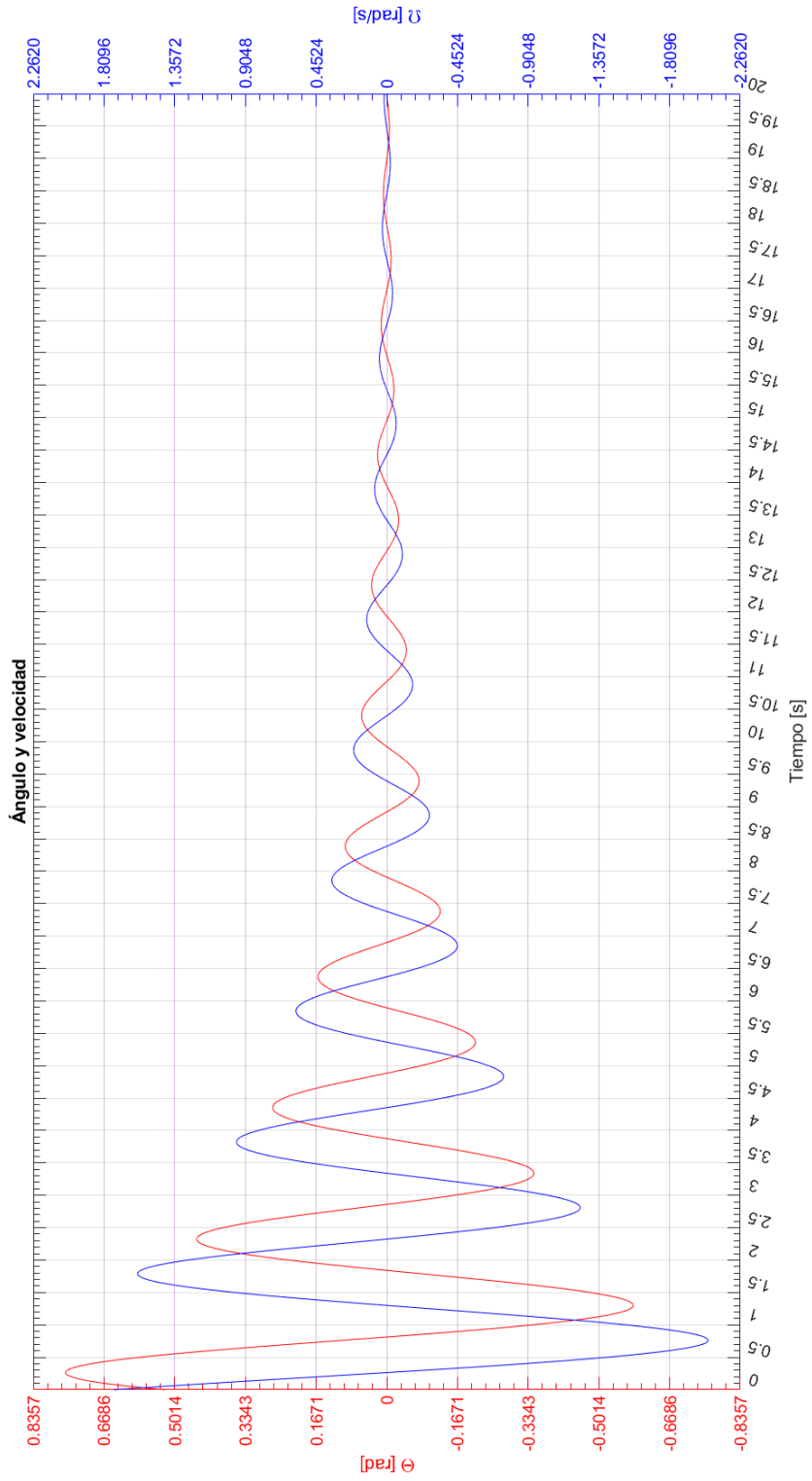

Figura 6.1: Gráfico de la solución para el primer caso con mayor cantidad de puntos.

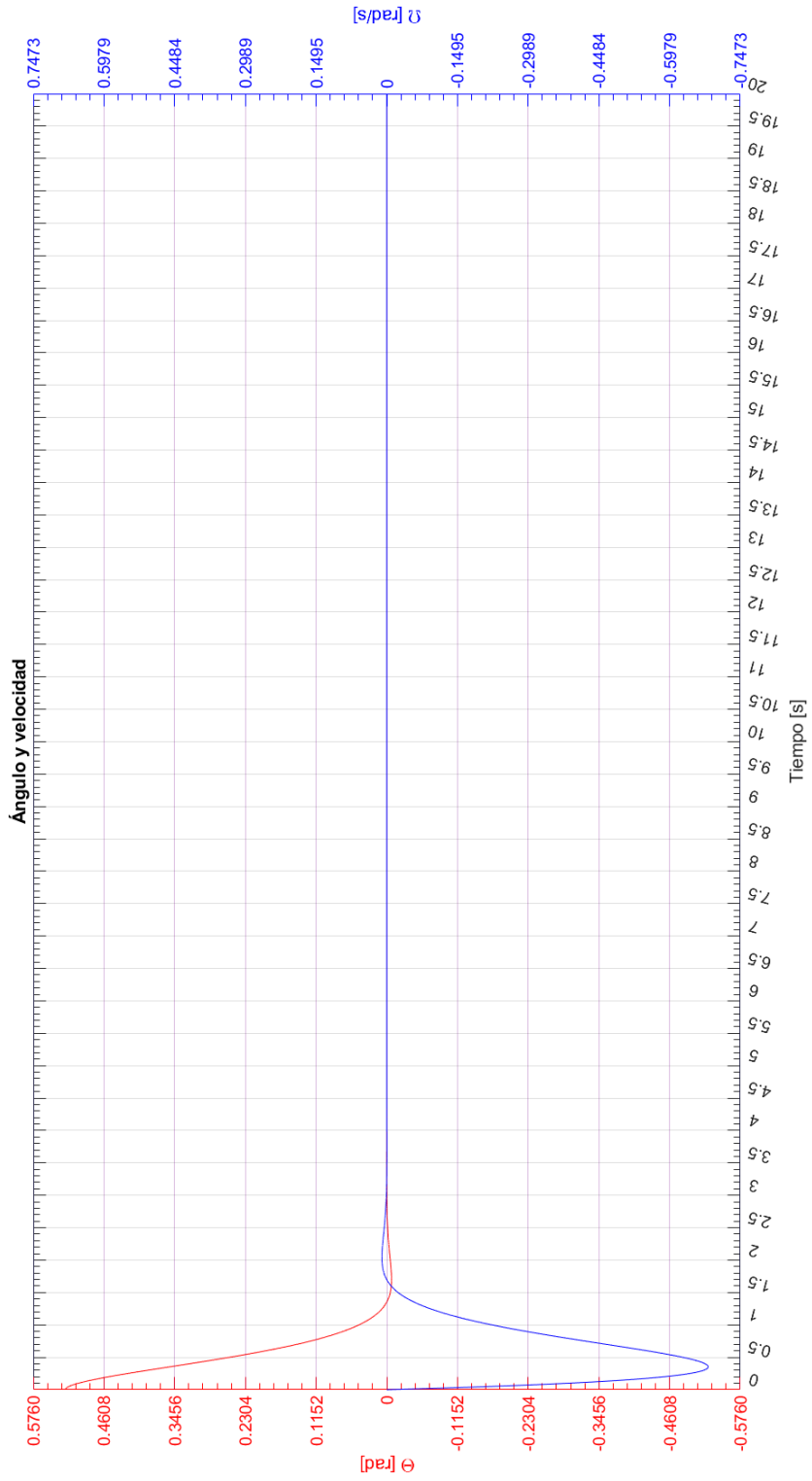

Figura 6.2: Gráfico de la solución para un sistema sobre-amortiguado.

7. Observaciones y conclusiones

El método numérico implementado en el caso propuesto se comporta en general como es esperado, sirvió la comparación con los métodos proveídos por **MATLAB** u **Octave** y también tener un problema planteado para el cual la solución se conoce de antemano. El algoritmo de Runge-Kutta fue implementado en forma matricial, lo cual a pesar de ser mas complicado de pensar inicialmente, permite lograr un código mas simple y entendible.

8. Bibliografía

Referencias

[1] *Álgebra lineal y ecuaciones diferenciales con MATLAB (1er Edición)*
Author: M. Golubitsky
Author: Dellnitz, M.
Publisher: INTERNATIONAL THOMSON EDITORES Edición (2001)
Copyright: © 2001 INTERNATIONAL THOMSON EDITORES.
ISBN 13: 978-9-706-86040-8
Website: https://www.marcialpons.es/libros

[2] *The PgfplotsTable Package*
Author: Dr. Christian Feuersänger
Copyright: © 2018, Christian Feuersänger.
Website: https://sourceforge.net/projects/pgfplots/

[3] *The Listings Package*
Author: Carsten Heinz
Author: Brooks Moses
Copyright: © 1996–2004, Carsten Heinz; © 2006–2007, Brooks Moses.
Website: http://www.ctan.org/pkg/listings

[4] *The Listingsutf8 Package*
Author: Heiko Oberdiek
Copyright: © 2007, Heiko Oberdiek.
Website: http://www.ctan.org/pkg/listingsutf8

Apéndices

A. Código fuente

A.1. Consideraciones para el código

El código está escrito en **MATLAB**, se trato de hacerlo ordenado y con todos los comentarios necesarios, así como también se hizo un uso consistente del indentado con tabulaciones a 4 espacios. La presentación del código en el informe se hizo directamente desde los fuentes hacia LaTeX usando el package **"listingsutf8"** [4], que es una extensión con soporte para **UTF8** del paquete **"listings"** [3], este paquete produce una salida formateada y con coloreado del código y también permite el agregado de números de líneas, el código fuente en **MATLAB** es soportado directamente, la salida que se produce es muy buena, pero no es perfecta, ya que cuestiones como el tabulado o el ancho total de las líneas pueden producir problemas, en caso de que alguno de estos problemas hagan confuso o incomprensible el código por favor remitirse a los fuentes originales.

A.2. Archivos fuente de MATLAB

A.2.1. tp2.m

```
% Implementa el TP2, declarando la variables necesarias y llamando a las
% respectivas funciones. Los gráficos son salvados en formato PNG en el
% correspodiente directorio del informe, también los resultados numéricos
% son salvados en el correspondiente directorio del informe, donde el
% código de Latex los levanta automáticamente para generar el archivo
% compilado final del informe.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Limpio todas las variables globales.
clear all;

% Cierro todos los gráficos.
close all;

% Determino si estoy trabajando en MATLAB u Octave.
Is_Octave = (5 == exist('OCTAVE_VERSION', 'builtin'));

% Determino el OS en el que estoy trabajando.
if (ismac)
    OS = 'Mac';
    % Mac plaform.
    % En general al igual que en Linux y otros Unix, se usa UTF-8, pero
    % no es necesariamente así.
elseif (isunix)
    OS = 'Linux';
    % Linux plaform.
    % En general se usa UTF-8, con lo que bastaría con codificar los
    % scripts en UTF-8 para que la codificación sea correcta.
elseif (ispc)
    OS = 'Windows';
    % Windows platform.
    % Esto es necesario mayormente para Octave en Windows, en MATLAB
    % CP1252 es el default. Los scripts deberían estar codificados
    % en CP1252.

    if (Is_Octave)
        major =  int8(str2double(substr(OCTAVE_VERSION, 1, ...
            index(OCTAVE_VERSION, ".") - 1)));

        if (major >= 5)
            [~, ~] = system ('chcp 65001');
        else
```

```
43              [~, ~] = system('chcp 1252');
44          end
45
46      else
47          [~, ~] = system('chcp 1252');
48      end
49
50  else
51      OS = 'Sistema desconocido';
52  end %if
53
54  % Limpio la línea de comando, para poder capturar la salida limpia.
55  clc;
56
57  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
58
59  %%
60
61  % Defino el archivo para la captura de la salida del script.
62  output_file = './salida.txt';
63
64  % Detengo la captura de la salida del script.
65  diary off;
66
67  % Borro el archivo si existía previamente.
68  if (exist(output_file, 'file'))
69      delete(output_file);
70  end
71
72  % Borro el archivo si existía previamente.
73  % if (exist('diary', 'file'))
74  %     delete('diary');
75  % end
76
77  % Inicio la captura de la salida.
78  diary on;
79
80  % Inicio la ejecución.
81  fprintf('Inicializando las variables globales para el TP2...');
82
83  % Declaro el directorio para las imágenes.
84  images_directory = fullfile('..', 'informe', 'img');
85
86  % Declaro el directorio para los archivos ".csv".
87  results_directory = fullfile('..', 'informe', 'results');
88
```

```
89 % Declaro los nombres de los archivos a guardar.
90 grafico_respuesta_prefix = 'grafico_respuesta';
91 grafico_1 = '_1';
92 grafico_2 = '_2.png';
93 solutions_prefix = 'valores_respuesta';
94 respuesta_1 = '_1';
95 respuesta_2 = '_2';
96
97 % Tiempo final.
98 tend = 20;
99
100 % Cantidad de niveles para Romberg.
101 romberg_rk4_levels = 15;
102
103 % En el caso de Ocatve reduzco los nivels porque es menos eficiente y
104 % tarda demasiado.
105 if (Is_Octave)
106     romberg_rk4_levels = 9;
107 end % if
108
109 fprintf('Listo\n\n');
110
111 if (~Is_Octave) % Si estoy trabajando en MATLAB.
112     env = 'MATLAB';
113 else            % Si estoy trabajando en Octave.
114     env = 'Octave';
115 end %if
116
117 fprintf('Trabajando en: %s %s sobre %s.\n\n\n\n', env, version, OS);
118
119
120 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
121 %%
122
123 fprintf('Creando el directorio para las imágenes...');
124
125 % Creo el directorio para las imágenes y verifico que exista.
126 [~, ~, ~] = mkdir(images_directory);
127 success = (7 == exist(images_directory, 'dir'));
128
129 % Chequeo que el directorio para las imágenes exista.
130 if (~success)
131     fprintf(strjoin({'\nNo se pudo crear ', ...
132         'el directorio para las imágenes.\n\n'}, ''));
133
134     exit;
```

```
135 else
136     fprintf('Listo\n\n');
137 end
138
139 fprintf('Creando el directorio para los resultados numéricos...');
140
141 % Creo el directorio para los archivos numéricos y verifico que exista.
142 [~, ~, ~] = mkdir(results_directory);
143 success = (7 == exist(results_directory, 'dir'));
144
145 % Chequeo que el directorio para los archivos numéricos exista.
146 if (~success)
147     fprintf(strjoin({'\nNo se pudo crear' , ...
148         'el directorio para los resultados.\n\n'}, ''));
149
150     exit;
151 else
152     fprintf('Listo\n\n');
153 end
154
155 % Seteo el formato de números por pantalla.
156 format long;
157
158 fprintf('\n\nCálculos para los parámetros pedidos:\n\n');
159
160 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
161 %%
162 % Cálculo para los parámetros pedidos.
163
164 % Masa.
165 m_given = [1 1];
166
167 % Longitud del hilo.
168 l_given = [1 1];
169
170 % Rozamiento.
171 b_given = [0 0.5];
172
173 % Paso.
174 h_given = [0.2 0.2];
175
176 % Ángulo inicial.
177 Theta_0_given = [30 30];
178
179 % Velocidad angular inicial.
180 Omega_0_given = [0 100];
```

```
181
182 % Nombre de los gráficos.
183 grafico = {grafico_1 grafico_2};
184
185 for i=1:2
186
187     fprintf(strjoin({'Calculando la estimación de '...
188         'la solución del sistema con m = %.3fKg, l = %.3fm, ' ...
189         'b = %.3fNs/m,\nh = %.3fs, Theta0 = %.3fº,', ...
190         ' Omega0 = %.3fº/s...'}, ''), ...
191         m_given(i), l_given(i), b_given(i), h_given(i), ...
192         Theta_0_given(i), Omega_0_given(i));
193
194     % Resuelvo la ecuación del péndulo para los parámetros dados.
195     % tend, h, b, l, m, Theta_0, Omega_0
196     [t, x, f_sol, ~] = pendulum(tend, h_given(i), b_given(i), ...
197         l_given(i), m_given(i), ...
198         Theta_0_given(i)*pi/180, Omega_0_given(i)*pi/180);
199
200     % Generación completa.
201     fprintf('Listo\n\n');
202
203     % Guardo los resultados en un archivo.
204     fprintf('Salvando los resultados en un archivo "CSV"......');
205
206     % Armo la tabla de resultados finales con:
207     % 1 - Tiempo.
208     % 2 - Theta.
209     % 3 - d(Theta)/dt.
210     results = [t', x(:,1), x(:,2)];
211
212     % Guardo la tabla de las iteraciones.
213     dlmwrite(fullfile(results_directory, ...
214         strjoin({solutions_prefix, respuesta_1, '.csv'}, '')), ...
215         results, 'precision', '%.15f');
216
217     % Guardado completo.
218     fprintf('Listo\n\n');
219
220     % Genero el gráfico de la  solución.
221     fprintf('Generando un gráfico de la solución...');
222
223     graphic_handle1 = ...
224         plot_solution(t, x(:,1)', x(:,2)', 'Ángulo y velocidad', 100);
225
226     % Generación completa.
```

```
227     fprintf('Listo\n\n');
228
229     %%%
230     solution_complete_name = ...
231         fullfile(images_directory, ...
232         strjoin({grafico_respuesta_prefix, grafico{i}, ...
233         '.png'}, ''));
234
235     fprintf('Salvando el gráfico en un archivo "PNG"......');
236
237     % Salvo el gráfico en un archivo.
238     saveas(graphic_handle1, solution_complete_name);
239
240     % Salvado completo.
241     fprintf('Listo\n\n');
242     %%%
243
244     fprintf('Calculando la integral del módulo de la posición......');
245
246     % Calculo la integral usando Romberg.
247     I_romb = romberg_rk4(f_sol, 0, tend, romberg_rk4_levels);
248
249     % Listo.
250     fprintf('Listo\n\n');
251
252     fprintf(strjoin({'El valor de la integral aproximada con Romberg ' ...
253         'de nivel %u del módulo de la posición es: %.16f.\n\n'}, ''), ...
254         romberg_rk4_levels, I_romb(size(I_romb,1), size(I_romb,2)));
255
256     fprintf('\n');
257
258 end
259
260 fprintf('\n');
261
262 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
263 %%
264 % Cálculo para parámetros dados por el usuario.
265
266 fprintf('Cálculos para parámetros ingresados por el usuario:\n\n');
267
268 while (1)
269
270     % Ingreso los valores iniciales.
271
272     % Masa.
```

```
273     m = 1;
274
275     % Longitud del hilo.
276     l = 1;
277
278     % Rozamiento.
279     b = 0.5;
280
281     % Paso.
282     h = 0.001;
283
284     % Ángulo inicial.
285     Theta_0 = 30;
286
287     % Velocidad angular inicial.
288     Omega_0 = 100;
289
290     % En el caso de Octave, un valor muy chico causa problemas
291     if (Is_Octave)
292         h = 0.01;
293     end
294
295     answer = questdlg(strjoin({'¿Desea resolver el sistema', ...
296         ' para otros parámetros?'}, ''), ...
297         'Pregunta', 'Si', 'No', 'No');
298
299     if ( ~strcmp('Si', answer))
300         break;
301     end %if
302
303     % Pido los datos para resolver el problema.
304     answer = inputdlg({'Masa del péndulo [Kg]','Largo del hilo (m)', ...
305         'Rozamiento [Kg/s]', 'Paso [s]', 'Ángulo inicial [º]', ...
306         'Velocidad angular inicial [º/s]'},...
307         'Parámetros del sistema', ...
308         [1 50; 1 50; 1 50; 1 50; 1 50; 1 50], ...
309         {num2str(m, 16) num2str(l, 16) num2str(b, 16) num2str(h, 16) ...
310         num2str(Theta_0, 16) num2str(Omega_0, 16)});
311
312     % Valido los datos.
313     if (isempty(answer))
314         continue;
315
316     end % if
317
318     m = str2double(answer{1});
```

```
319
320     l = str2double(answer{2});
321
322     b = str2double(answer{3});
323
324     h = str2double(answer{4});
325
326     Theta_0 = str2double(answer{5});
327
328     Omega_0 = str2double(answer{6});
329
330     % Valido los datos ingresados.
331     if isnan(h) || isnan(b) || isnan(l) || isnan(m) || isnan(Theta_0) || ...
332             isnan(Omega_0) || (b < 0) || (l <= 0) || (m <= 0) || ...
333             ((Theta_0 == 0) && (Omega_0 == 0))
334
335         dlg = errordlg('Parámteros inválidos', 'Error', 'modal');
336
337         uiwait(dlg);
338
339         continue;
340
341     end %if
342
343     fprintf(strjoin({'Calculando la estimación de '...
344         'la solución del sistema con m = %.3fKg, l = %.3fm, ' ...
345         'b = %.3fNs/m,\nh = %.3fs, Theta0 = %.3fº,', ...
346         ' Omega0 = %.3fº/s...'}, ''), ...
347         m, l, b, h, Theta_0, Omega_0);
348
349     % Resuelvo la ecuación del péndulo para los parámetros dados.
350     [t, x, f_sol, s] = pendulum(tend, h, b, l, m, ...
351         Theta_0*pi/180, Omega_0*pi/180);
352
353     if (~s)
354         continue;
355     end % fi
356
357     % Listo.
358     fprintf('Listo\n\n');
359
360     if exist('graphic_handle', 'var') == 1
361         % Cierro el gráfico previo.
362         close(graphic_handle);
363     end
364
```

```
365     % Genero el gráfico de la  solución.
366     fprintf('Generando un gráfico de la solución...');
367
368     % Genero el gráfico de las soluciones.
369     graphic_handle = plot_solution(t, x(:,1)', x(:,2)', ...
370         'Ángulo y velocidad', 100);
371
372     % Listo.
373     fprintf('Listo\n\n');
374
375     fprintf('Calculando la integral del módulo de la posición......');
376
377     % Calculo la integral usando Romberg.
378     I_romb = romberg_rk4(f_sol, 0, tend, romberg_rk4_levels);
379
380     % Listo.
381     fprintf('Listo\n\n');
382
383     fprintf(strjoin({'El valor de la integral aproximada con Romberg ' ...
384         'de nivel %u del módulo de la posición es: %.16f.\n\n'}, ''), ...
385         romberg_rk4_levels, I_romb(size(I_romb,1), size(I_romb,2)));
386
387     fprintf('\n');
388
389 end
390
391 fprintf('\n');
392
393 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
394 %%
395 % Guardo la salida del programa.
396
397 % El script se ejecutó correctamente..
398 fprintf('\n\nEjecución del TP2 terminada.\n\n');
399
400 % Detengo la captura de la salida del script.
401 diary off;
402
403 % Copio el archivo de salida.
404 fprintf('Copiando el archivo de salida......');
405
406 [status, ~] = copyfile('diary', output_file);
407
408 % Chequeo que la copia se realizó correctamente.
409 if (~status)
410     fprintf(strjoin({'\nFalló la copia', ...
```

```
411         'del archivo de salida.\n\n'}));
412
413     return;
414 else  %if
415     fprintf('Listo\n\n');
416 end  %if
```

A.2.2. pendulum.m

```
% Resuelve el sistema del péndulo en el intervalo [0, tend] y con los
% parámetros dados, o unos tomados por default.
%
% Parámetros:
% -----------
% tend:     Final del intervalo en el cual aproximar las funciones
%           solución del sistema (Se asume el inicio en 0).
% h:        Paso deseado para el cálculo de las aproximaciones, el valor
%           finalmente usado puede que sea menor para acomodarse al
%           intervalo de aproximación.
% b:        Coeficiente de rozamiento viscoso.
%           [Este parámetro es opcional, el default es 0 Kg/s].
% l:        Longitud del hilo.
%           [Este parámetro es opcional, el default es 1 m].
% m:        Masa del péndulo.
%           [Este parámetro es opcional, el default es 1 Kg].
% Theta_0:  Ángulo inicial.
%           [Este parámetro es opcional, el default es Pi/6 rad].
% Omega_0:  Velocidad angular inicial.
%           [Este parámetro es opcional, el default es 0 rad/s].
%
% Salidas:
% --------
% t:        Vector con los valores de la variable independiente donde se
%           aproximó las m funciones solución del sistema en el intervalo.
% x:        Matriz de dimensión Nx2, donde N es la cantidad de puntos
%           de aproximación de las funciones dentro del intervalo los
%           valores de las filas corresponden a las aproximaciones de las
%           funciones solución en los valores de la variable independiente
%           correspondientes al mismo índice en el array t.
%
function [t, x, f_sol, s] = pendulum(tend, h, b, l, m, Theta_0, Omega_0)

if (nargin < 2)
    fprintf(strjoin({'\nError: ',...
        'Insuficientes ', ...
        'parámetros.\n'}, ''));
    t = [];
    x = [];
    s = 0;
    return;
end %if

if (tend < 0)
```

```
45      fprintf(strjoin({'\nError: ',...
46          'Tiempo final ', ...
47          'inválido.\n'}, ''));
48      t = [];
49      x = [];
50      s = 0;
51      return;
52  end %if
53
54  if (h >= tend/10) || (h <= 0)
55      fprintf(strjoin({'\nError: ',...
56          'Paso ',...
57          'inválido.\n'}, ''));
58      t = [];
59      x = [];
60      s = 0;
61      return;
62  end %if
63
64  if (nargin < 3)                  % Aseguro de tener el valor si no se dió.
65      b = 0;                       % Valor por omisión.
66  end %if
67
68  if (nargin < 4)                  % Aseguro de tener el valor si no se dió.
69      l = 1;                       % Valor por omisión.
70  end %if
71
72  if (nargin < 5)                  % Aseguro de tener el valor si no se dió.
73      m = 1;                       % Valor por omisión.
74  end %if
75
76  if (nargin < 6)                  % Aseguro de tener el valor si no se dió.
77      Theta_0 = Pi/6;              % Valor por omisión.
78  end %if
79
80  if (nargin < 7)                  % Aseguro de tener el valor si no se dió.
81      Omega_0 = 0;                 % Valor por omisión.
82  end %if
83
84
85  if isnan(h) || isnan(b) || isnan(l) || isnan(m) || isnan(Theta_0) || ...
86          isnan(Omega_0) || (b < 0) || (l <= 0) || (m <= 0) || ...
87          ((Theta_0 == 0) && (Omega_0 == 0))
88      fprintf(strjoin({'\nError: ',...
89          'Parámetros del péndulo ',...
90          'inválidos.\n'}, ''));
```

```
91      t = [];
92      x = [];
93      s = 0;
94      return;
95  end  %if
96
97
98  g = 9.81;                           % Aceleración de la gravedad.
99
100
101 f = @(t,x) [x(2); ...                % Sistema de dos ecuaciones del péndulo.
102     -(b/m)*x(2)- ...
103     (g/l)*sin(x(1))];
104
105 [t, x] = rk4(f, [0 tend + h],... % Llamo a Runge-Kutta de cuarto orden.
106     [Theta_0, Omega_0], h);
107
108
109 % Genero el handle a función que lleva implícito el sistema a resolver
110 % por Runge-Kutta de orden 4, y el índice a la solución que se quiere
111 % obtener, se usa para el cálculo de la integral.
112 f_sol = @(a, b, step) ...
113     abs(f_rk4_num_sol(f, [a, b], [Theta_0, Omega_0], step, 1));
114
115
116 s = 1;
117
118 end  % function
```

A.2.3. rk4.m

```
% Implementa el método de Runge-Kutta de orden 4.
%
% Parámetros:
% -----------
% f:        Puntero a la función que toma un valor escalar y devuelve un
%           array de m valores correspondientes a las m funciones
%           del sistema a aproximar.
% tspan:    Array de dos valores con el valor inicial y final del
%           intervalo en el cual aproximar las funciones.
% y0:       Array de m valores iniciales para las funciones.
% step:     Paso deseado para el cálculo de las aproximaciones, el valor
%           finalmente usado puede que sea menor para acomodarse al
%           intervalo de aproximación.
%
% Salidas:
% --------
% t:        Vector con los valores de la variable independiente donde se
%           aproximó las m funciones solución del sistema en el intervalo.
% y:        Matriz de dimensión Nxm, donde N es la cantidad de puntos
%           de aproximación de las funciones dentro del intervalo y m la
%           cantidad de funciones del sistema, los valores de las filas
%           corresponden a las aproximaciones de las funciones solución
%           en los valores de la variable independiente correspondientes
%           al mismo índice en el array t.
%
function [t, y] = rk4(f, tspan, y0, step)

N = ceil((tspan(2) - tspan(1))/step); % Determino la cantidad de pasos.

h = (tspan(2) - tspan(1))/N;          % Adapto el paso al intervalo.

u = zeros(1, N);                      % Inicializo la variable
                                      % que contendrá los pasos
                                      % de la variable independiente.

u(1) = tspan(1);                      % Guardo el primer valor de
                                      % la variable independiente, que
                                      % corresponde al inicio del
                                      % intervalo.

m = length(y0);                       % Obtengo la cantidad de
                                      % funciones a estimar.

z = zeros(N, m);                      % Inicializo la matriz que
```

```
45                                      % contendrá las estimaciones de
46                                      % las m funciones solución.
47
48 z(1, : ) = y0;                       % Guardo los valores iniciales
49                                      % en la matriz.
50
51 k = zeros(m, 4);                     % Inicializo la matriz que
52                                      % contendrá los k de cada
53                                      % iteración.
54
55 for i = 1:N-1
56
57     k(:, 1) = f(u(i), z(i, :));      % Calulo de la primera columna
58                                      % de la matriz k, corresponde
59                                      % a los k1 de las m funciones.
60
61     k(:, 2) = f(u(i) + (1/2)*h, ...  % Calulo de la segunda columna
62         z(i, :) + (1/2)*h*k(:, 1)'); % de la matriz k, corresponde
63                                      % a los k2 de las m funciones.
64
65
66     k(:, 3) = f((u(i) + (1/2)*h), ... % Calulo de la tercera columna
67         (z(i, :) + (1/2)*h*k(:, 2)')); % de la matriz k, corresponde
68                                      % a los k3 de las m funciones.
69
70     k(:, 4) = f((u(i) + h),...       % Calulo de la cuarta columna
71         (z(i, :) + k(:, 3)'*h));     % de la matriz k, corresponde
72                                      % a los k4 de las m funciones.
73
74     z(i+1, :) = z(i, :) +  ...       % Calculo el próximo valor
75         (1/6)*(k(:, 1)' + ...        % de la variable independiente.
76         2*k(:, 2)' + 2*k(:, 3)' + ...
77         k(:, 4)')*h;
78
79     u(i + 1) = u(1) + i*h;           % Calculo el próximo valor de la
80                                      % variable independiente.
81 end % for
82
83 t = u;                               % Devuelvo los valores de la
84                                      % variable independiente.
85
86 y = z;                               % Devuelvo la matriz de
87                                      % estimaciones de las m
88                                      % funciones solución del sistema.
89
90 end %function
```

A.2.4. plot_solution.m

```
% Grafica el ángulo y la velocidad angular.
%
% Parámetros:
% -----------
% t:         Vector con los valores de la variable independiente donde se
%            aproximó las  funciones solución del sistema en el intervalo.
% Theta:     Array de los valores de ángulo correspodientes a la variable
%            independiente,t, a graficar.
% Omega:     Array de los valores de la velocidad angular
%            correspodientes a la variable independiente,t, a graficar.
% titulo:    Título para el gráfico.
% sz_perc:   Tamaño del gráfico en porcentaje de la pantalla.
%
% Salidas:
% --------
% graphic_handle: Puntero al gráfico.
%
function [graphic_handle] = plot_solution(t, Theta, Omega, titulo, sz_perc)

if (nargin < 5)                  % Aseguro de tener un tamaño.
    sz_perc = 50;                % Valor por omisión.
end %if

if (sz_perc < 10.0)              % Ajusto el tamaño de salida.
    sz_perc = 10.0;
elseif (sz_perc > 100.0)
    sz_perc = 100.0;
end % if

% Determino si estoy trabajando en MATLAB u Octave.
Is_Octave = (5 == exist('OCTAVE_VERSION', 'builtin'));

% Calculo el tamaño y la posición de la imagen.
pict_size = sz_perc/100;
pict_pos = (1 - pict_size)/2;

% Genero la figura, a un % del tamaño de la panatalla y centrada.
% No parece funcionar en Octave, pero no genera errores tampoco.
figure1 = figure('units', 'normalized', 'outerposition', ...
    [pict_pos pict_pos pict_size pict_size]);

% Seteo el color de fondo para el gráfico.
set(figure1, 'Color', [1 1 1]);

```

```
45 % Creo los ejes.
46 axes1 = axes('Parent', figure1);
47
48 % Activo eje izquierdo.
49 %yyaxis(axes1, 'left');
50
51 % Creo el gráfico de Theta.
52 % plot(axes1, t, Theta, 'Color', [1 0 0]);
53
54 % Grafico los datos.
55 [yyax, h1, h2] = plotyy(axes1, t, Theta, t, Omega);
56
57 % Seteo el color de los ejes de ordenadas.
58 set(yyax, {'ycolor'}, {[1 0 0]; [0 0 1]})
59
60 % Seteo el color de los gráficos.
61 set(h1, 'color', [1 0 0]);
62 set(h2, 'color', [0 0 1]);
63
64 % Seteo el label para y.
65 ylabel(yyax(1), '\Theta [rad]');
66
67 % Seteo el label para x.
68 xlabel(axes1, 'Tiempo [s]');
69
70 % Seteo el título.
71 title(axes1, titulo);
72
73 % Octave no soporta labels inclinados.
74 if (~Is_Octave)
75     % Labels inclinados.
76     xtickangle(axes1, 75);
77 end % if
78
79 % Límites para las abcisas.
80 xlim(axes1, [t(1) t(length(t))]);
81
82 % Determino los límites.
83 maxy = max([abs(min(Theta)) abs(max(Theta))]);
84
85 % Ticks para el eje y izquierdo.
86 yticks1 = (-1.1*maxy : 1.1*maxy/5 : 1.1*maxy);
87
88 % Límites para el eje y izquierdo.
89 ylim(yyax(1), [yticks1(1) yticks1(length(yticks1))]);
90
```

```
91 % Muestro la "caja" que contiene al gráfico.
92 box(axes1,'on');
93
94 % Seteo las propiedades del eje de abcisas.
95 set(axes1, 'FontSize',14, 'XGrid','on','XMinorTick','on','XTick',...
96     (t(1): 0.5: t(length(t))),...
97     'TickLabelInterpreter','tex');
98
99 % Seteo las propiedades del eje de ordenadas 1.
100 set(yyax(1), 'FontSize',14, 'XGrid','on','XMinorTick','on','XTick',...
101     (t(1): 0.5: t(length(t))),...
102     'YGrid','on','YMinorTick','on','YTick',...
103     yticks1, 'TickLabelInterpreter','tex');
104
105 % Seteo el label para y.
106 ylabel(yyax(2), '\Omega [rad/s]');
107
108 % Determino los límites.
109 maxy = max([abs(min(Omega)) abs(max(Omega))]);
110
111 % Ticks para el eje y izquierdo.
112 yticks2 = (-1.1*maxy : 1.1*maxy/5 : 1.1*maxy);
113
114 % Límites para el eje y izquierdo.
115 ylim(yyax(2), [yticks2(1) yticks2(length(yticks2))]);
116
117 % Seteo las propiedades del eje de ordenadas 2.
118 set(yyax(2), 'FontSize',14,...
119     'YGrid','on','YMinorTick','on','YTick',...
120     yticks2, 'TickLabelInterpreter','tex');
121
122 % Fuerzo a que se muestre al gráfico de inmediato.
123 drawnow;
124
125 % Asigno el valor del handle del gráfico que devuelvo.
126 graphic_handle = figure1;
```

A.2.5. romberg_rk4.m

```
% Implementa el método de Romberg adaptado para usar Runge-Kutta.
%
% Parámetros:
% -----------
% f:        Puntero a la función que calcula Runge-Kutta de orden 4,
%           tomando como parámetros los límites de la solución y el paso,
%           el sistema que se resuelve está implicito en el puntero, es
%           transparente para este método.
% a:        Inicio del intervalo de integración.
% b:        Fin del intervalo de integración.
% p:        Número de niveles.
% Salidas:
% --------
% I:        Valor de la integral aproximada.
function I = romberg_rk4(f, a, b, p)

% Calculo el paso que se usa en el último nivel de Romberg.
h_final = (b - a)/2^(p - 1);

% Llamo a Runge-Kutta de cuarto orden. Uso el paso mas pequeño del método.
% Aprovecho la relación de dos que existe entre el tamaño de paso en
% dos llamadaas sucesivas al método de integración de trapecios compuesto,
% de esta manera, solo necesito una llamada a Runge-Kutta y en cada
% nivel, para trapecios simplemente salteo 2^(p - k) puntos, donde k es el
% número de nivel y p la cantidad de niveles de Romberg totales.

z = f(a, b, h_final);

I = zeros(p, p);

for k=1:p
    % llamo al método de los trapecios para 2^k pasos de integración.
    I(k,1) = trapezcomp_rk4(z, a, b, k, p);

    % Fórmula recursiva de Romberg.
    for j=1:k-1
        I(k,j+1) = (4^j * I(k,j) - I(k-1,j)) / (4^j - 1);
    end

end

end
```

A.2.6. trapezcomp_rk4.m

```
% Implementa el método de Trapecios adaptado para usar Runge-Kutta.
%
% Parámetros:
% -----------
% x:        Array con los puntos de la función a la cual calcular la
%           aproximación de la integral, contiene 2^(p - 1) + 1 puntos,
%           pero salteo de a 2^(p - k), usando efectivamente 2^(k - 1) + a
%           puntos en el cálculo.
% a:        Inicio del intervalo de integración.
% b:        Fin del intervalo de integración.
% n:        Número de paneles.
% Salidas:
% --------
% In:       Valor de la integral aproximada.
function In = trapezcomp_rk4(x, a, b, k, p)

% Incialización.
h = (b-a)/(2^(k-1));

loop_step = 2^(p - k);
loop_start = 1 + loop_step;
loop_finish = 2^(p-1);

%Regla compuesta.
In = x(1);

for j=loop_start:loop_step:loop_finish
    In = In + 2 * x(j);
end

In = (In + x(length(x))) * h * 0.5;

end
```

A.2.7. f_rk4_num_sol.m

```
% Esta función auxiliar solo se usa para generar un handle a función
% que lleva implicito un sistema de ecuaciones diferenciales a resolver
% por Runge-Kutta 4 y sus parámetros, excepto el intervalo y el paso.
%
% Parámetros:
% -----------
% f:         Puntero a la función que toma un valor escalar y devuelve un
%            array de m valores correspondientes a las m funciones
%            del sistema a aproximar.
% tspan:     Array de dos valores con el valor inicial y final del
%            intervalo en el cual aproximar las funciones.
% y0:        Array de m valores iniciales para las funciones.
% step:      Paso deseado para el cálculo de las aproximaciones, el valor
%            finalmente usado puede que sea menor para acomodarse al
%            intervalo de aproximación.
% numsol:    Índice de la solución del sistema a devolver.
%
% Salidas:
% --------
% sol:        atriz de dimensión Nxm, donde N es la cantidad de puntos
%            de aproximación de las funciones dentro del intervalo y m la
%            cantidad de funciones del sistema, los valores de las filas
%            corresponden a las aproximaciones de las funciones solución
%            en los valores de la variable independiente correspondientes
%            al mismo índice en el array t.
function sol=f_rk4_num_sol(f, tspan, y0, step, numsol)

[~, temp] = rk4(f, tspan, y0, step);

sol = temp(:, numsol)';

end
```

B. Captura de la salida

B.1. Consideraciones para el código

El texto corresponde a la captura que se hace desde el script de **MATLAB**, la codificación es automáticamente convertida por LaTeX usando el package **"listingsutf8"**.

B.2. Archivo de captura de la salida

B.2.1. salida.txt

```
Inicializando las variables globales para el TP2...Listo

Trabajando en: MATLAB 9.7.0.1190202 (R2019b) sobre Windows.



Creando el directorio para las imágenes...Listo

Creando el directorio para los resultados numéricos...Listo



Cálculos para los parámetros pedidos:

Calculando la estimación de la solución del sistema con m = 1.000Kg, l = 1.000m, b = 0.000Ns/m,
h = 0.200s, Theta0 = 30.000º, Omega0 = 0.000º/s...Listo

Salvando los resultados en un archivo "CSV"......Listo

Generando un gráfico de la solución...Listo

Salvando el gráfico en un archivo "PNG"......Listo

Calculando la integral del módulo de la posición......Listo

El valor de la integral aproximada con Romberg de nivel 15 del módulo de la posición es: 6.6541496163424361.


Calculando la estimación de la solución del sistema con m = 1.000Kg, l = 1.000m, b = 0.500Ns/m,
h = 0.200s, Theta0 = 30.000º, Omega0 = 100.000º/s...Listo

Salvando los resultados en un archivo "CSV"......Listo

Generando un gráfico de la solución...Listo

Salvando el gráfico en un archivo "PNG"......Listo

Calculando la integral del módulo de la posición......Listo

El valor de la integral aproximada con Romberg de nivel 15 del módulo de la posición es: 2.1010993325885252.
```

```
Cálculos para parámetros ingresados por el usuario:

Calculando la estimación de la solución del sistema con m = 1.000Kg, l = 1.000m, b = 0.500Ns/m,
h = 0.001s, Theta0 = 30.000º, Omega0 = 100.000º/s...Listo

Generando un gráfico de la solución...Listo

Calculando la integral del módulo de la posición......Listo

El valor de la integral aproximada con Romberg de nivel 15 del módulo de la posición es: 2.1010993325885252.


Calculando la estimación de la solución del sistema con m = 1.000Kg, l = 1.000m, b = 5.000Ns/m,
h = 0.001s, Theta0 = 30.000º, Omega0 = 0.000º/s...Listo

Generando un gráfico de la solución...Listo

Calculando la integral del módulo de la posición......Listo

El valor de la integral aproximada con Romberg de nivel 15 del módulo de la posición es: 0.2855416685189804.




Ejecución del TP2 terminada.
```