



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

29 de Diciembre de 2019

Índice

Índice	I
Índice de figuras	I
1. Enunciado	1
1.1. Resumen enunciado	1
1.2. Resolución	1
2. Implementación de los algoritmos	2
2.1. Sobre los archivos de MATLAB y Octave	4
3. Aproximación de la solución del sistema	5
4. Primer caso	6
5. Segundo caso	8
6. Otros casos	10
7. Observaciones y conclusiones	13
8. Bibliografía	14
Apéndices	15
A. Código fuente	15
A.1. Consideraciones para el código	15
A.2. Archivos fuente de MATLAB	16
A.2.1. tp2.m	16
A.2.2. pendulum.m	26
A.2.3. rk4.m	29
A.2.4. plot_solution.m	31
A.2.5. romberg.m	34
A.2.6. trapezcomp.m	35
B. Captura de la salida	36
B.1. Consideraciones para el código	36
B.2. Archivo de captura de la salida	37
B.2.1. salida.txt	37

Índice de figuras

4.1. Gráfico de la solución para el primer caso.	7
5.1. Gráfico de la solución para el primer caso.	9
6.1. Gráfico de la solución para el primer caso con mayor cantidad de puntos.	11
6.2. Gráfico de la solución para un sistema sobre-amortiguado.	12

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:

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

Planteando:

$$\begin{aligned} \theta &= x_1 \\ \dot{\theta} &= x_2 \end{aligned}$$

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

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\frac{b}{m} \cdot x_2 - \frac{g}{l} \cdot \sin(x_1) \end{cases} \quad (1.2)$$

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 **L^AT_EX**, para los archivos “**CSV**” se usó el paquete de **L^AT_EXpgfplotstable**.

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 (ángulo) y su derivada (velocidad angular) y se calcula la integral del módulo de la posición (ángulo) 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.m**” (apéndice [\[A.2.6\]](#)): Función que implementa el método de integración de trapecios compuesto (usado en Romberg).

“**romberg.m**” (apéndice [\[A.2.5\]](#)): Función que implementa el método de integración 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.

Algo importante a aclarar, es que decidimos implementar el método de integración de Romberg en su forma “standard”, en esta forma se requiere la evaluación de la función a integrar en puntos arbitrarios, para poder luego integrar la función que representa el módulo de la posición del péndulo, se interpoló la función entre los puntos obtenidos con Runge-Kutta, usando un spline. Esta no era la única forma de realizar la integración, pero era la que mas se adaptaba a usar el algoritmo de Romberg sin modificaciones, además era muy simple dado que **MATLAB** y **Octave** proveen la función **interp1** que puede realizar distintos tipos de

interpolación sobre un array de puntos, entre ellas, spline. El método de Romberg se implementó, así como está descrito en la bibliografía, usando interpolación de Richardson y el método de trapecios compuesto, para lo cual se implementó el mismo, también en su forma “standard”.

El error del método de integración de Romberg ($R_{n,n}$), asumiendo que la función es suficientemente diferenciable, está en $O(h^{(2 \cdot n + 2)})$, donde h es el paso, que en el caso de este método, vale en cada iteración $h = \frac{1}{2^n} \cdot (a - b)$, siendo a y b los límites de integración y n el número de iteración.

El hecho de que usemos interpolación en la función que integramos, sin embargo, hace que este error no sea del todo correcto, una determinación más detallada, haría necesario analizar cómo afecta el spline este resultado. Dado que usamos un nivel de Romberg de 6 y el intervalo tiene en todos los casos una longitud de 20s, ignorando la cuestión de la interpolación, tendríamos un error del orden de 0,3.

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	0.000000000000000
0.2000000000	0.428315087292475	-0.924314351982748
0.4000000000	0.175709914049286	-1.524073049304326
0.6000000000	-0.142463407885895	-1.556916160332241
0.8000000000	-0.407010149910302	-1.008781534885019
19.2000000000	-0.415713351925902	-0.884154047180177
19.4000000000	-0.505243416520664	0.012928905395467
19.6000000000	-0.410594725597481	0.905002954749895
19.8000000000	-0.164778335500693	1.477269716303494
20.0000000000	0.142581276925356	1.498993631310300

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:

$$I = 6,5419117322670095$$

Como se dijo anteriormente, el error, no es fácilmente determinable dado que se tiene el error en la solución del sistema y el uso del spline, sin embargo, solo basándose en el error del método de Romberg, solo un decimal sería significativo.

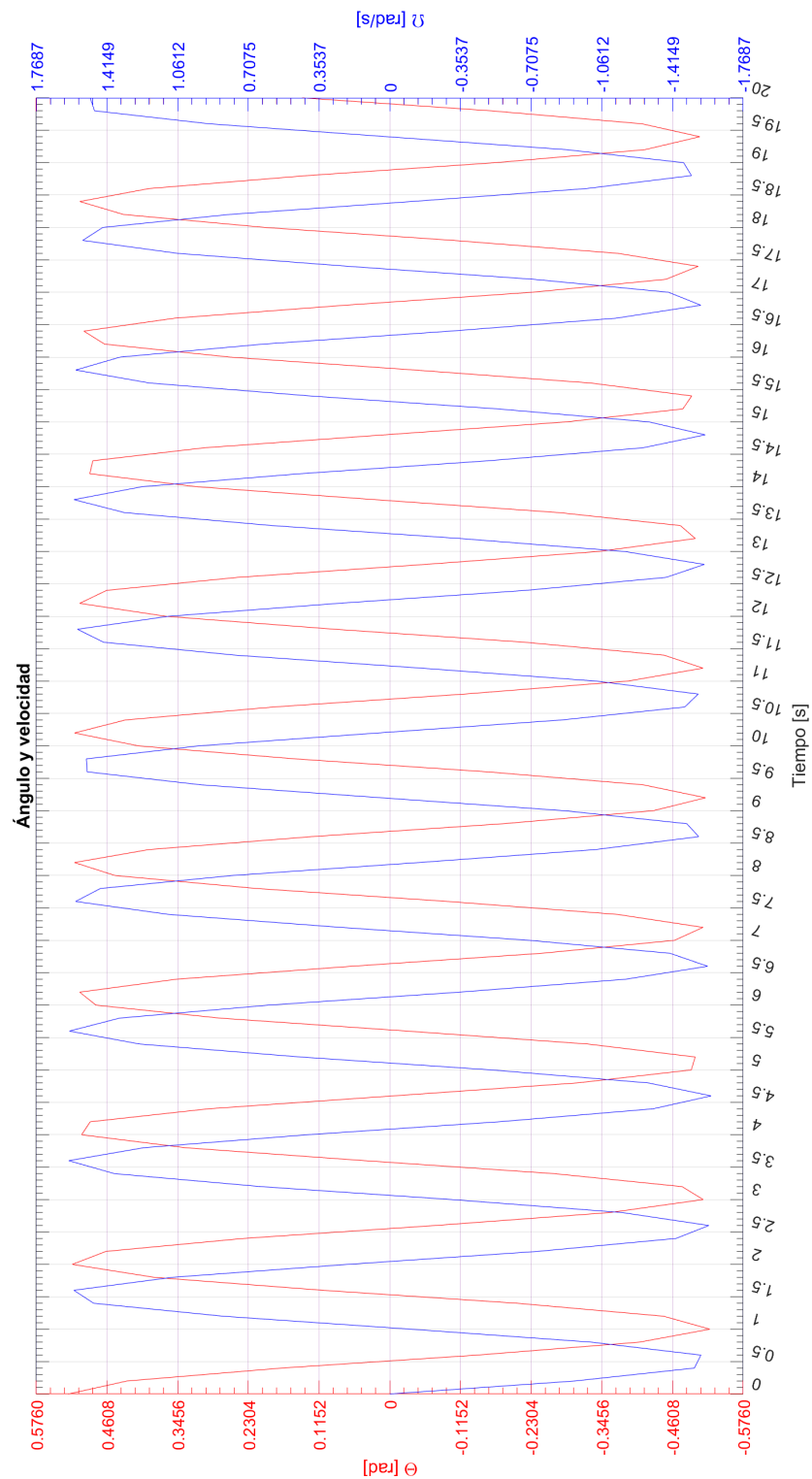


Figura 4.1: Gráfico de la solución para el primer caso.

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	2.617993877991494
0.2000000000	0.903505438553949	1.115462552160061
0.4000000000	0.961574033561247	-0.518529636080321
0.6000000000	0.714414334007562	-1.877398965081030
0.8000000000	0.255331869758280	-2.573580770695137
19.2000000000	-0.002044202806820	-0.025232569429452
19.4000000000	-0.006158130915492	-0.014792804051463
19.6000000000	-0.007660317428277	-0.000016881970728
19.8000000000	-0.006258371494630	0.013356312010605
20.0000000000	-0.002734467478418	0.020640635026120

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:

$$I = 2,7320644496497102$$

Vale lo mismo dicho anteriormente para el error.

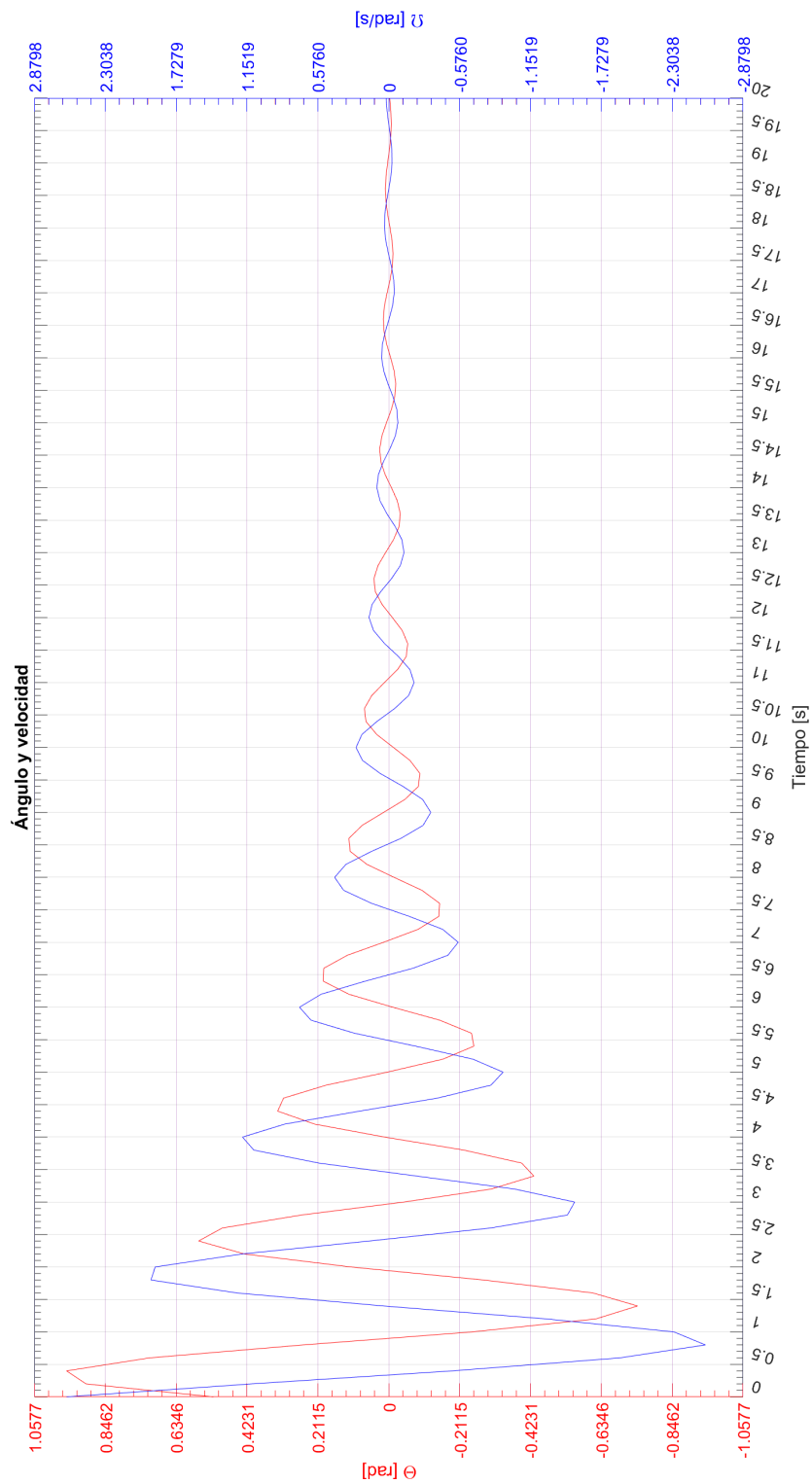


Figura 5.1: Gráfico de la solución para el primer caso.

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 $h = 0,001s$, 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:

$$m = 1\text{kg}$$

$$l = 1\text{m}$$

$$b = 5\text{N s/m}$$

$$h = 0,2s$$

$$\Theta_0 = 30^\circ$$

$$\dot{\Theta}_0 = 0^\circ/s$$

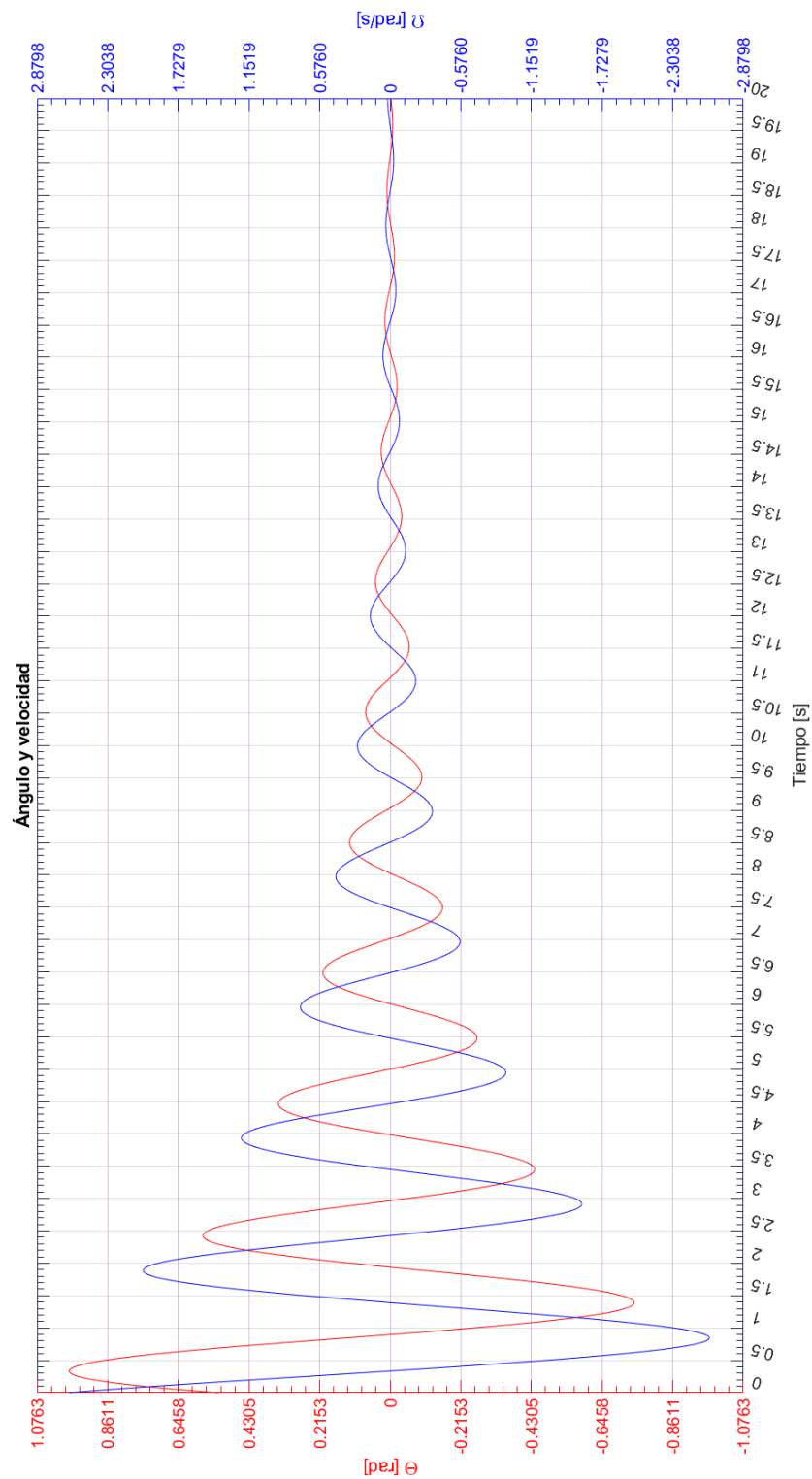


Figura 6.1: Gráfico de la solución para el primer caso con mayor cantidad de puntos.

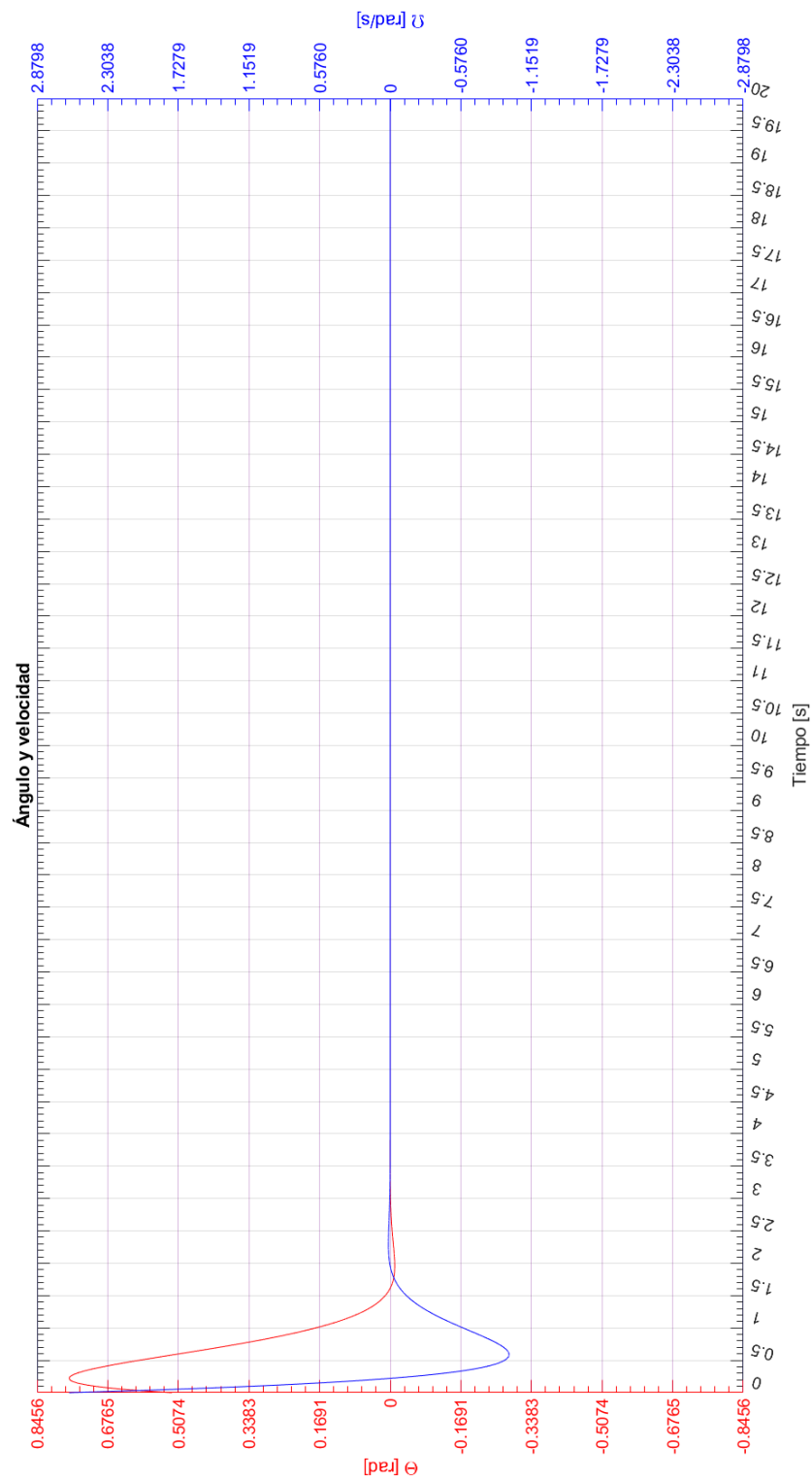


Figura 6.2: Gráfico de la solución para un sistema sobre-amortiguado.

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 (1^{er} 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

```
1 % Implementa el TP2, declarando la variables necesarias y llamando a las
2 % respectivas funciones. Los gráficos son salvados en formato PNG en el
3 % correspondiente directorio del informe, también los resultados numéricos
4 % son salvados en el correspondiente directorio del informe, donde el
5 % código de Latex los levanta automáticamente para generar el archivo
6 % compilado final del informe.
7 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
8
9 % Limpio todas las variables globales.
10 clear all;
11
12 % Cierro todos los gráficos.
13 close all;
14
15 % Determino si estoy trabajando en MATLAB u Octave.
16 Is_Octave = (5 == exist('OCTAVE_VERSION', 'builtin'));
17
18 % Determino el OS en el que estoy trabajando.
19 if (ismac)
20     OS = 'Mac';
21     % Mac plaform.
22     % En general al igual que en Linux y otros Unix, se usa UTF-8, pero
23     % no es necesariamente así.
24 elseif (isunix)
25     OS = 'Linux';
26     % Linux plaform.
27     % En general se usa UTF-8, con lo que bastaría con codificar los
28     % scripts en UTF-8 para que la codificación sea correcta.
29 elseif (ispc)
30     OS = 'Windows';
31     % Windows platform.
32     % Esto es necesario mayormente para Octave en Windows, en MATLAB
33     % CP1252 es el default. Los scripts deberían estar codificados
34     % en CP1252.
35
36     if (Is_Octave)
37         major = int8(str2double(substr(OCTAVE_VERSION, 1, ...
38             index(OCTAVE_VERSION, ".") - 1)));
39
40         if (major >= 5)
41             [~, ~] = system ('chcp 65001');
42         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('Iniciando 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 filas para Romberg.
101 romberg_levels = 10;
102
103 % En el caso de Octave reduzco los niveles porque es menos eficiente y
104 % tarda demasiado.
105 if (Is_Octave)
106     romberg_levels = 6;
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 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
159 %%
160 % Cálculo para el primer juego de parámetros.
161
162 fprintf(strjoin({'Calculando la estimación de '...
163     'la solución del sistema con m = 1Kg, l = 1m, ' ...
164     'b = 0Ns/m, h = 0.2s, Theta0 = pi/6, Omega0 = 0...'}, ''));
165
166 % Resuelvo la ecuación del péndulo para los parámetros dados.
167 % tend, h, b, l, m, Theta_0, Omega_0
168 [t, x, ~] = pendulum(tend, 0.2, 0, 1, 1, pi/6, 0);
169
170 % Generación completa.
171 fprintf('Listo\n\n');
172
173 % Guardo los resultados en un archivo.
174 fprintf('Salvando los resultados en un archivo "CSV".....');
175
176 % Armo la tabla de resultados finales con:
177 % 1 - Tiempo.
178 % 2 - Theta.
179 % 3 - d(Theta)/dt.
180 results = [t', x(:,1), x(:,2)];
```

```
181
182 % Guardo la tabla de las iteraciones.
183 dlmwrite(fullfile(results_directory, ...
184     strjoin({solutions_prefix, respuesta_1, '.csv'}, '')), ...
185     results, 'precision', '%.15f');
186
187 % Guardado completo.
188 fprintf('Listo\n\n');
189
190 % Genero el gráfico de la solución.
191 fprintf('Generando un gráfico de la solución...');
192
193 graphic_handle1 = ...
194     plot_solution(t, x(:,1)', x(:,2)', 'Ángulo y velocidad', 100);
195
196 % Generación completa.
197 fprintf('Listo\n\n');
198
199 %%%
200 solution_complete_name = ...
201     fullfile(images_directory, ...
202     strjoin({grafico_respuesta_prefix, grafico_1, ...
203     '.png'}, ''));
204
205 fprintf('Salvando el gráfico en un archivo "PNG".....');
206
207 % Salvo el gráfico en un archivo.
208 saveas(graphic_handle1, solution_complete_name);
209
210 % Salvado completo.
211 fprintf('Listo\n\n');
212 %%%
213
214 fprintf('Calculando la integral del módulo de la posición.....');
215
216 % Genero la función del módulo interpolada usando spline.
217 fint = @(y) interp1(t, abs(x(:,1)'), y, 'spline');
218
219 % Calculo la integral usando Romberg.
220 I_romb = romberg(fint, 0, tend, romberg_levels);
221
222 % Listo.
223 fprintf('Listo\n\n');
224
225 fprintf(strjoin({'El valor de la integral aproximada con Romberg del ' ...
226     'módulo de la posición es: %.16f.\n\n'}, ''), ...
```

```
227     I_romb(size(I_romb,1), size(I_romb,2));
228
229 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
230 %%
231 % Cálculo para el segundo juego de parámetros.
232
233 fprintf(strjoin({'Calculando la estimación de '...
234     'la solución del sistema con m = 1Kg, l = 1m, ' ...
235     'b = 0.5Ns/m, h = 0.2s, Theta0 = pi/6, Omega0 = 5*pi/6...'}, ''));
236
237 % Resuelvo la ecuación del péndulo para los parámetros dados.
238 % tend, h, b, l, m, Theta_0, Omega_0
239 [t, x, s] = pendulum(tend, 0.2, 0.5, 1, 1, pi/6, 5*pi/6);
240
241 % Generación completa.
242 fprintf('Listo\n\n');
243
244 % Guardo los resultados en un archivo.
245 fprintf('Salvando los resultados en un archivo "CSV".....');
246
247 % Armo la tabla de resultados finales con:
248 % 1 - Tiempo.
249 % 2 - Theta.
250 % 3 - d(Theta)/dt.
251 results = [t', x(:,1), x(:,2)];
252
253 % Guardo la tabla de las iteraciones.
254 dlmwrite(fullfile(results_directory, ...
255     strjoin({solutions_prefix, respuesta_2, '.csv'}, '')), ...
256     results, 'precision', '%.15f');
257
258 % Guardado completo.
259 fprintf('Listo\n\n');
260
261 % Genero el gráfico de la solución.
262 fprintf('Generando un gráfico de la solución...');
263
264 graphic_handle2 = ...
265     plot_solution(t, x(:,1)', x(:,2)', 'Ángulo y velocidad', 100);
266
267 % Generación completa.
268 fprintf('Listo\n\n');
269
270 %%%
271 solution_complete_name = ...
272     fullfile(images_directory, ...
```

```
273     strjoin({grafico_respuesta_prefix, grafico_2, ...
274     '.png'}, ''));
275
276 fprintf('Salvando el gráfico en un archivo "PNG".....');
277
278 % Salvo el gráfico en un archivo.
279 saveas(graphic_handle2, solution_complete_name);
280
281 % Salvado completo.
282 fprintf('Listo\n\n');
283 %%%
284
285 fprintf('Calculando la integral del módulo de la posición.....');
286
287 % Genero la función del módulo interpolada usando spline.
288 fint = @(y) interp1(t, abs(x(:,1)'), y, 'spline');
289
290 % Calculo la integral usando Romberg.
291 I_romb = romberg(fint, 0, tend, romberg_levels);
292
293 % Listo.
294 fprintf('Listo\n\n');
295
296 fprintf(strjoin({'El valor de la integral aproximada con Romberg del ' ...
297     'módulo de la posición es: %.16f.\n\n'}, ''), ...
298     I_romb(size(I_romb,1), size(I_romb,2)));
299
300 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
301 %%
302 % Cálculo para parámetros dados por el usuario.
303
304 while (1)
305
306     % Ingreso los valores iniciales.
307
308     % Masa.
309     m = 1;
310
311     % Longitud del hilo.
312     l = 1;
313
314     % Rozamiento.
315     b = 0.5;
316
317     % Paso.
318     h = 0.001;
```

```
319
320     % Ángulo inicial.
321     Theta_0 = pi/6;
322
323     % Velocidad angular inicial.
324     Omega_0 = 5*pi/6;
325
326     % En el caso de Octave, un valor muy chico causa problemas
327     if (Is_Octave)
328         h = 0.01;
329     end
330
331     answer = questdlg(strjoin({'¿Desea resolver el sistema', ...
332         ' para otros parámetros?'}, ''), ...
333         'Pregunta', 'Si', 'No', 'No');
334
335     if (~strcmp('Si', answer))
336         break;
337     end %if
338
339     % Pido los datos para resolver el problema.
340     answer = inputdlg({'Masa del péndulo [Kg]', 'Largo del hilo (m)', ...
341         'Rozamiento [Kg/s]', 'Paso [s]', 'Ángulo inicial [rad]', ...
342         'Velocidad angular inicial [rad/s]'}, ...
343         'Parámetros del sistema', ...
344         [1 50; 1 50; 1 50; 1 50; 1 50; 1 50], ...
345         {num2str(m, 16) num2str(l, 16) num2str(b, 16) num2str(h, 16) ...
346         num2str(Theta_0, 16) num2str(Omega_0, 16)});
347
348     % Valido los datos.
349     if (isempty(answer))
350         continue;
351
352     end % if
353
354     m = str2double(answer{1});
355
356     l = str2double(answer{2});
357
358     b = str2double(answer{3});
359
360     h = str2double(answer{4});
361
362     Theta_0 = str2double(answer{5});
363
364     Omega_0 = str2double(answer{6});
```

```
365
366     % Valido los datos ingresados.
367     if isnan(h) || isnan(b) || isnan(l) || isnan(m) || isnan(Theta_0) || ...
368         isnan(Omega_0) || (b < 0) || (l <= 0) || (m <= 0) || ...
369         ((Theta_0 == 0) && (Omega_0 == 0))
370
371         dlg = errordlg('Parámetros inválidos', 'Error', 'modal');
372
373         uiwait(dlg);
374
375         continue;
376
377     end %if
378
379     fprintf(strjoin({'Calculando la estimación de '...
380         'la solución del sistema...'}, ''));
381
382     % Resuelvo la ecuación del péndulo para los parámetros dados.
383     [t, x, s] = pendulum(tend, h, b, l, m, Theta_0, Omega_0);
384
385     if (~s)
386         continue;
387     end % fi
388
389     % Listo.
390     fprintf('Listo\n\n');
391
392     if exist('graphic_handle', 'var') == 1
393         % Cierro el gráfico previo.
394         close(graphic_handle);
395     end
396
397     % Genero el gráfico de la solución.
398     fprintf('Generando un gráfico de la solución...');
399
400     % Genero el gráfico de las soluciones.
401     graphic_handle = plot_solution(t, x(:,1)', x(:,2)', ...
402         'Ángulo y velocidad', 100);
403
404     % Listo.
405     fprintf('Listo\n\n');
406
407
408
409     fprintf('Calculando la integral del módulo de la posición.....');
410
```

```
411 % Genero la función del módulo interpolada usando spline.
412 fint = @(y) interp1(t, abs(x(:,1)'), y, 'spline');
413
414 % Calculo la integral usando Romberg.
415 I_romb = romberg(fint, 0, tend, romberg_levels);
416
417 % Listo.
418 fprintf('Listo\n\n');
419
420 fprintf(strjoin({'El valor de la integral aproximada con Romberg del ' ...
421     'módulo de la posición es: %.16f.\n\n'}, ''), ...
422     I_romb(size(I_romb,1), size(I_romb,2)));
423
424 end
425
426 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
427 %%
428 % Guardo la salida del programa.
429
430 % El script se ejecutó correctamente..
431 fprintf('\n\nEjecución del TP2 terminada.\n\n');
432
433 % Detengo la captura de la salida del script.
434 diary off;
435
436 % Copio el archivo de salida.
437 fprintf('Copiando el archivo de salida.....');
438
439 [status, ~] = copyfile('diary', output_file);
440
441 % Chequeo que la copia se realizó correctamente.
442 if (~status)
443     fprintf(strjoin({'\nFalló la copia', ...
444         'del archivo de salida.\n\n'}));
445
446     return;
447 else %if
448     fprintf('Listo\n\n');
449 end %if
450
451 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

A.2.2. pendulum.m

```
1 % Resuelve el sistema del péndulo en el intervalo [0, tend] y con los
2 % parámetros dados, o unos tomados por default.
3 %
4 % Parámetros:
5 % -----
6 % tend:      Final del intervalo en el cual aproximar las funciones
7 %            solución del sistema (Se asume el inicio en 0).
8 % h:         Paso deseado para el cálculo de las aproximaciones, el valor
9 %            finalmente usado puede que sea menor para acomodarse al
10 %           intervalo de aproximación.
11 % b:         Coeficiente de rozamiento viscoso.
12 %            [Este parámetro es opcional, el default es 0 Kg/s].
13 % l:         Longitud del hilo.
14 %            [Este parámetro es opcional, el default es 1 m].
15 % m:         Masa del péndulo.
16 %            [Este parámetro es opcional, el default es 1 Kg].
17 % Theta_0:   Ángulo inicial.
18 %            [Este parámetro es opcional, el default es Pi/6 rad].
19 % Omega_0:   Velocidad angular inicial.
20 %            [Este parámetro es opcional, el default es 0 rad/s].
21 %
22 % Salidas:
23 % -----
24 % t:         Vector con los valores de la variable independiente donde se
25 %            aproximó las m funciones solución del sistema en el intervalo.
26 % x:         Matriz de dimensión Nx2, donde N es la cantidad de puntos
27 %            de aproximación de las funciones dentro del intervalo los
28 %            valores de las filas corresponden a las aproximaciones de las
29 %            funciones solución en los valores de la variable independiente
30 %            correspondientes al mismo índice en el array t.
31 %
32 function [t, x, s] = pendulum(tend, h, b, l, m, Theta_0, Omega_0)
33
34 if (nargin < 2)
35     fprintf(strjoin({'\nError: ', ...
36         'Insuficientes ', ...
37         'parámetros.\n'}, ''));
38     t = [];
39     x = [];
40     s = 0;
41     return;
42 end %if
43
44 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 curato orden.  
106             [Theta_0, Omega_0], h);  
107  
108 s = 1;  
109  
110 end % function
```

A.2.3. rk4.m

```
1 % Implementa el método de Runge-Kutta de orden 4.
2 %
3 % Parámetros:
4 % -----
5 % f:          Puntero a la función que toma un valor escalar y devuelve un
6 %             array de m valores correspondientes a las m funciones solución
7 %             del sistema a aproximar.
8 % tspan:      Array de dos valores con el valor inicial y final del
9 %             intervalo en el cual aproximar las funciones.
10 % y0:         Array de m valores iniciales para las funciones.
11 % step:       Paso deseado para el cálculo de las aproximaciones, el valor
12 %             finalmente usado puede que sea menor para acomodarse al
13 %             intervalo de aproximación.
14 %
15 % Salidas:
16 % -----
17 % t:          Vector con los valores de la variable independiente donde se
18 %             aproximó las m funciones solución del sistema en el intervalo.
19 % y:          Matriz de dimensión Nxm, donde N es la cantidad de puntos
20 %             de aproximación de las funciones dentro del intervalo y m la
21 %             cantidad de funciones del sistema, los valores de las filas
22 %             corresponden a las aproximaciones de las funciones solución
23 %             en los valores de la variable independiente correspondientes
24 %             al mismo índice en el array t.
25 %
26 function [t, y] = rk4(f, tspan, y0, step)
27
28 N = ceil((tspan(2) - tspan(1))/step); % Determino la cantidad de pasos.
29
30 h = (tspan(2) - tspan(1))/N;          % Adapto el paso al intervalo.
31
32 u = zeros(1, N);                      % Inicializo la variable
33                                       % que contendrá los pasos
34                                       % de la variable independiente.
35
36 u(1) = tspan(1);                      % Guardo el primer valor de
37                                       % la variable independiente, que
38                                       % corresponde al inicio del
39                                       % intervalo.
40
41 m = length(y0);                      % Obtengo la cantidad de
42                                       % funciones a estimar.
43
44 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

```
1 % Grafica el ángulo y la velocidad angular.
2 %
3 % Parámetros:
4 % -----
5 % t:          Vector con los valores de la variable independiente donde se
6 %             aproximó las funciones solución del sistema en el intervalo.
7 % Theta:      Array de los valores de ángulo correspondientes a la variable
8 %             independiente, t, a graficar.
9 % Omega:      Array de los valores de la velocidad angular
10 %            correspondientes a la variable independiente, t, a graficar.
11 % titulo:     Título para el gráfico.
12 % sz_perc:    Tamaño del gráfico en porcentaje de la pantalla.
13 %
14 % Salidas:
15 % -----
16 % graphic_handle: Puntero al gráfico.
17 %
18 function [graphic_handle] = plot_solution(t, Theta, Omega, titulo, sz_perc)
19
20 if (nargin < 5)                                % Aseguro de tener un tamaño.
21     sz_perc = 50;                               % Valor por omisión.
22 end %if
23
24 if (sz_perc < 10.0)                             % Ajusto el tamaño de salida.
25     sz_perc = 10.0;
26 elseif (sz_perc > 100.0)
27     sz_perc = 100.0;
28 end % if
29
30 % Determino si estoy trabajando en MATLAB u Octave.
31 Is_Octave = (5 == exist('OCTAVE_VERSION', 'builtin'));
32
33 % Calculo el tamaño y la posición de la imagen.
34 pict_size = sz_perc/100;
35 pict_pos = (1 - pict_size)/2;
36
37 % Genero la figura, a un % del tamaño de la pantalla y centrada.
38 % No parece funcionar en Octave, pero no genera errores tampoco.
39 figure1 = figure('units', 'normalized', 'outerposition', ...
40     [pict_pos pict_pos pict_size pict_size]);
41
42 % Seteo el color de fondo para el gráfico.
43 set.figure1, 'Color', [1 1 1]);
44
```

```
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.m

```
1 % Implementa el método de Romberg.
2 %
3 % Parámetros:
4 % -----
5 % f:          Puntero a la función que cuya integral se desea aproximar.
6 % a:          Inicio del intervalo de integración.
7 % b:          Fin del intervalo de integración.
8 % p:          Número de filas.
9 % Salidas:
10 % -----
11 % I:          Valor de la integral aproximada.
12 function I = romberg(f, a, b, p)
13
14 I = zeros(p, p);
15 for k=1:p
16     % llamo al método de los trapecios para  $n = 2^k$ .
17     I(k,1) = trapezcomp(f, a, b, 2^(k-1));
18
19     % Fórmula recursiva de Romberg.
20     for j=1:k-1
21         I(k,j+1) = (4^j * I(k,j) - I(k-1,j)) / (4^j - 1);
22     end
23
24 end
25
26 end
```


A.2.6. trapezcomp.m

```
1 % Implementa el método de Trapecios.
2 %
3 % Parámetros:
4 % -----
5 % f:          Puntero a la función que cuya integral se desea aproximar.
6 % a:          Inicio del intervalo de integración.
7 % b:          Fin del intervalo de integración.
8 % n:          Número de paneles.
9 % Salidas:
10 % -----
11 % In:         Valor de la integral aproximada.
12 function In = trapezcomp(f, a, b, n)
13
14 % Incialización.
15 h = (b-a)/n;
16 x = a;
17
18 %Regla compuesta.
19 In =f(a);
20 for k=2:n
21     x = x + h;
22     In = In + 2. * f(x);
23 end
24 In = (In + f(b)) * h * 0.5;
25
26 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 L^AT_EX 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

Calculando la estimación de la solución del sistema con $m = 1\text{Kg}$, $l = 1\text{m}$, $b = 0\text{Ns/m}$, $h = 0.2\text{s}$, $\text{Theta0} = \pi/6$, Omega0

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 del módulo de la posición es: 6.5419117322670095.

Calculando la estimación de la solución del sistema con $m = 1\text{Kg}$, $l = 1\text{m}$, $b = 0.5\text{Ns/m}$, $h = 0.2\text{s}$, $\text{Theta0} = \pi/6$, Omega0

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 del módulo de la posición es: 2.7320644496497102.

Calculando la estimación de la solución del sistema...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 del módulo de la posición es: 2.7378140579622841.

Calculando la estimación de la solución del sistema...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 del módulo de la posición es: 0.5846264909781677.

Ejecución del TP2 terminada.