



**Universidad de Valladolid**



**ESCUELA DE INGENIERÍAS  
INDUSTRIALES**

**UNIVERSIDAD DE VALLADOLID**

**ESCUELA DE INGENIERIAS INDUSTRIALES**

**Grado en Ingeniería en Tecnologías Industriales**

**Estrategias de resolución de problemas  
de ingeniería formulados en términos de  
EDOs de 2º orden.**

**Autor:**

**Puente Taboada, Raquel**

**Tutor:**

**Álvarez López, Jorge  
Departamento Matemática Aplicada**

**Valladolid, julio 2017.**



## AGRADECIMIENTOS

Es aquí donde quiero expresar mi más sincero agradecimiento a todos los que de un modo u otro han formado parte del presente proyecto.

En primer lugar, a mi tutor, Jorge Álvarez López, sin el cual no podría haber realizado este trabajo, por su dedicación, su esfuerzo, su amabilidad, por toda la atención que me ha dedicado y por siempre ayudarme a seguir avanzando.

A mis padres, Aurora y Samuel, sin los que no habría sido posible llegar hasta aquí, y que siempre han estado a mi lado para aconsejarme durante estos años.

A mi novio, Álvaro, por toda la ayuda, la paciencia, el tiempo y el apoyo que me ha brindado siempre.

A mis compañeras, pero sobre todo amigas, Inés y Paula, por estos cuatro años de carrera juntas, porque, aunque haya habido momentos difíciles, ha habido otros muchos buenos juntos, y, compartirlos con vosotras ha hecho que sean más especiales.

Al resto de mis compañeros y amigos que me ha dado la Universidad y la Ingeniería, en especial Andrea y Raúl.

Sin olvidar al resto de personas, las de siempre, vosotros ya sabéis quiénes sois, de un modo u otro también formáis parte de esto.

Han sido años de mucho esfuerzo, pero con vuestra ayuda todo ha sido mucho más fácil.

Gracias a todos.



## RESUMEN

En muchas áreas de las Ciencias, Ingeniería e Industria, surgen problemas que involucran la resolución numérica de Ecuaciones Diferenciales Ordinarias de Segundo Orden. Continúa siendo éste un activo campo de investigación y se siguen proponiendo métodos para aproximar las soluciones de Problemas de Valores Iniciales y en la Frontera gobernados por E.D.O.s de este tipo. En este proyecto, describiremos brevemente algunos métodos "clásicos" e introduciremos nuevos métodos numéricos para este tipo de problemas, basados en aproximar las soluciones mediante Splines y Pseudo-Splines de clase  $C^2$ , utilizando técnicas de colocación. Para resolver los P.V.F. combinamos técnicas basadas en el Método del Disparo, con otras basadas en resolver sistemas lineales tridiagonales mediante el Algoritmo de Thomas. Finalmente, un buen número de experimentos numéricos, la mayoría del ámbito de la Ingeniería, permiten analizar la precisión y orden de los métodos.

## PALABRAS CLAVE

Ecuaciones Diferenciales Ordinarias, Splines, Método de Colocación, Método del Disparo, Integración Numérica.

## ABSTRACT

In many areas of Science, Engineering and Industry, problems that involve the numerical resolution of Second-Order Ordinary Differential Equations arise. The research in this area is still very active and lots of methods are being proposed to approximate the solutions of Initial and Boundary Value Problems governed by Second-Order O.D.E.s. In this project, we will describe some "classical" methods and introduce other new numerical methods for these type of problems, based on the approximation of the solutions by using  $C^2$  Splines and Pseudo-Splines, with the help of collocations techniques. To solve the B.V.P. we combine techniques based on the Shooting Method, with others based on solving tridiagonal linear systems using the Thomas Algorithm. Finally, a good number of numerical experiments, most of them in the Engineering field, allow us to analyse the precision and order of the different methods.

## KEY WORDS

Ordinary Differential Equations, Splines, Collocation Method, Shooting Method, Numerical Integration.





# ÍNDICE





## ÍNDICE

ÍNDICE.....	VII
ÍNDICE DE FIGURAS .....	XIII
ÍNDICE DE TABLAS .....	XIX
ÍNDICE DE ECUACIONES .....	XXIII
<b>INTRODUCCIÓN Y OBJETIVOS.....</b>	<b>1</b>
1. JUSTIFICACIÓN Y OBJETIVOS .....	3
1.1. <i>Justificación</i> .....	3
1.2. <i>Objetivos</i> .....	6
1.3. <i>Estructura</i> .....	7
<b>ECUACIONES DIFERENCIALES DE SEGUNDO ORDEN .....</b>	<b>9</b>
2. ECUACIONES DIFERENCIALES DE SEGUNDO ORDEN.....	11
2.1. <i>Introducción a las Ecuaciones Diferenciales de Segundo Orden</i> .....	11
2.2. <i>Introducción a los métodos numéricos</i> .....	13
2.2.1. <i>Condiciones de aplicabilidad de métodos numéricos</i> .....	13
2.2.2. <i>Nomenclatura en métodos numéricos</i> .....	14
2.3. <i>Problemas de Valores Iniciales</i> .....	15
2.3.1. <i>Método de Euler Explícito</i> .....	17
2.3.2. <i>Métodos de Taylor</i> .....	19
2.3.3. <i>Método trapezoidal o método de Heun</i> .....	21
2.3.4. <i>Métodos de Runge-Kutta</i> .....	22
2.3.5. <i>Métodos Adams-Bashfort-Moulton</i> .....	27
2.4. <i>Problema con Valores de Frontera</i> .....	30
<b>MÉTODOS DE RESOLUCIÓN BASADOS EN SPLINES .....</b>	<b>33</b>
3. MÉTODOS DE RESOLUCIÓN BASADOS EN SPLINES .....	35
3.1. <i>Introducción a la utilización de Splines</i> .....	35
3.2. <i>Resolución de Problemas Lineales de Valores Iniciales</i> .....	36
3.2.1. <i>Planteamiento del Método de Colocación basado en Splines Cúbicos</i> ...	36
3.2.2. <i>Obtención del método de colocación, basado en Splines Cúbicos, para</i> <i>aproximar la solución de P.V.I. lineales</i> .....	39
3.2.3. <i>Método de resolución "inverso" basado en Splines Cúbicos</i> .....	42
3.2.4. <i>Método de colocación basado en Pseudo- Splines Cuárticos</i> .....	46
3.3. <i>Resolución de Problemas Lineales con Valores en la Frontera</i> .....	52
3.3.1. <i>Método de colocación basado en Splines Cúbicos para P.V.F.s</i> .....	52
3.3.2. <i>Método de colocación basado en Pseudo-Splines Cuárticos para P.V.F.</i> 58	
3.4. <i>Resolución de Problemas de Valores Iniciales No Lineales</i> .....	66
3.4.1. <i>Método de resolución basado en Splines Cúbicos</i> .....	66
3.4.2. <i>Método de resolución basado en Pseudo - Splines Cuárticos</i> .....	69

<b>MÉTODO DEL DISPARO .....</b>	<b>73</b>
4. MÉTODO DEL DISPARO .....	75
4.1. <i>Introducción al Método del Disparo</i> .....	75
4.2. <i>Método del Disparo Lineal</i> .....	77
4.2.1. Problemas con condiciones de contorno simples .....	77
4.2.2. Problemas con condiciones de contorno generales .....	82
4.3. <i>Método del Disparo No Lineal</i> .....	83
<b>EXPERIMENTOS NUMÉRICOS .....</b>	<b>91</b>
5. EXPERIMENTOS NUMÉRICOS .....	93
5.1. <i>Introducción</i> .....	93
5.2. <i>Ejemplo 1</i> .....	98
5.2.1. Planteamiento del problema .....	98
5.2.2. Resolución .....	98
5.2.3. Resultados .....	98
5.3. <i>Ejemplo 2</i> .....	106
5.3.1. Planteamiento del problema .....	106
5.3.2. Resolución .....	106
5.3.3. Resultados .....	107
5.4. <i>Ejemplo 3</i> .....	111
5.4.1. Planteamiento del problema .....	111
5.4.2. Resolución .....	112
5.4.3. Resultados .....	113
5.5. <i>Ejemplo 4</i> .....	118
5.5.1. Planteamiento del problema .....	118
5.5.2. Resolución .....	119
5.5.3. Resultados .....	119
5.6. <i>Ejemplo 5</i> .....	126
5.6.1. Planteamiento del problema .....	126
5.6.2. Resolución .....	127
5.6.3. Resultados .....	127
5.7. <i>Ejemplo 6</i> .....	138
5.7.1. Planteamiento del problema .....	138
5.7.2. Resolución .....	139
5.7.3. Resultados .....	139
5.8. <i>Ejemplo 7</i> .....	150
5.8.1. Planteamiento del problema .....	150
5.8.2. Resolución .....	150
5.8.3. Resultados .....	151
5.9. <i>Ejemplo 8</i> .....	156
5.9.1. Planteamiento del problema .....	156
5.9.2. Resolución .....	156
5.9.3. Resultados .....	157
5.10. <i>Ejemplo 9</i> .....	160
5.10.1. Planteamiento del problema .....	160
5.10.2. Resolución .....	160
5.10.3. Resultados .....	160

<b>CONCLUSIONES .....</b>	<b>165</b>	
6. CONCLUSIONES .....	167	
6.1. <i>Resultados obtenidos</i> .....	167	
6.2. <i>Conclusiones</i> .....	168	
6.3. <i>Líneas de continuación</i> .....	169	
<b>BIBLIOGRAFÍA.....</b>	<b>171</b>	
<b>ANEXOS.....</b>	<b>179</b>	
7. ANEXOS: CÓDIGO DE MATLAB .....	181	
7.1. <i>Métodos de resolución basados en Splines</i> .....	181	
7.1.1. Funciones para resolver Problemas de Valores Iniciales Lineales.....	181	
7.1.2. Funciones para resolver Problemas con Valores en la Frontera Lineales	186	
7.1.3. Funciones para resolver Problemas de Valores Iniciales No Lineales ...	196	
7.2. <i>Métodos de resolución basados en el Método del Disparo</i> .....	201	
7.2.1. Funciones para resolver Problemas con Valores de Frontera Lineales con	condiciones generales .....	201
7.2.2. Funciones para resolver Problemas con Valores de Frontera No Lineales	con condiciones de contorno simples .....	206
7.3. <i>Programas utilizados en los Experimentos Numéricos</i> .....	225	
7.3.1. Ejemplo 1 .....	225	
7.3.2. Ejemplo 2 .....	228	
7.3.3. Ejemplo 3 .....	231	
7.3.4. Ejemplo 4 .....	235	
7.3.5. Ejemplo 5 .....	238	
7.3.6. Ejemplo 6 .....	245	
7.3.7. Ejemplo 7 .....	252	
7.3.8. Ejemplo 8 .....	255	
7.3.9. Ejemplo 9 .....	257	





# ÍNDICE DE FIGURAS



## ÍNDICE DE FIGURAS

Figura 2.1. Interpretación geométrica del método de Euler.....	19
Figura 2.2. Ejemplo comparativo de la precisión de los métodos Euler, Heun y RK4 .....	27
Figura 4.1. Método del Disparo para resolución de P.V.F lineales .....	79
Figura 4.2. Método del Disparo para resolución de P.V.F no lineales .....	84
Figura 4.3. Selección de elevaciones $tk$ para alcanzar al valor final $z_2$ .....	85
Figura 5.1. Gráfica de las soluciones numéricas y exactas ( $h = 0.4$ ).....	100
Figura 5.2. Detalle de la gráfica anterior: Zoom de las soluciones numéricas y exactas ( $h = 0.4$ ) .....	101
Figura 5.3. Errores en las aproximaciones obtenidas con Splines Cúbicos ( $h = 0.4$ ) .....	101
Figura 5.4. Gráfica en doble escala logarítmica de los errores para Splines Cúbicos en el punto $x = 2.8$ , para los distintos pasos.....	104
Figura 5.5. Gráfica en doble escala logarítmica de los errores para Pseudo-Splines Cuárticos en el punto $x = 2.8$ , para los distintos pasos.....	105
Figura 5.6. Representación gráfica de las soluciones ( $h = 0.6$ ) .....	108
Figura 5.7. Detalle de la gráfica anterior: zoom de las soluciones numéricas y exactas ( $h = 0.6$ ) .....	108
Figura 5.8. Gráficas de las soluciones numéricas $h = 0.0375$ .....	111
Figura 5.9. Gráficas de las soluciones numéricas y asintótica tomando 6 puntos ( $\varepsilon = 10^{-4}$ ).....	114
Figura 5.10. Detalle de la gráfica anterior: zoom de las soluciones numéricas y exactas con 6 puntos ( $\varepsilon = 10^{-4}$ ).....	114
Figura 5.11. Gráfica de los errores en la aproximación a la solución tomando 8 puntos ( $\varepsilon = 10^{-4}$ ).....	115
Figura 5.12. Gráficas de las soluciones numéricas y asintótica tomando 12 subintervalos ( $\varepsilon = 10^{-8}$ ) .....	117

Figura 5.13. Detalle de la gráfica anterior: zoom de las soluciones numéricas y asintótica con 12 subintervalos ( $\varepsilon = 10^{-8}$ ) .....	117
Figura 5.14. Gráfica de las soluciones numéricas y exactas ( $h = 0.2$ ) .....	120
Figura 5.15. Detalle de la gráfica anterior: zoom de las soluciones numéricas y exactas ( $h = 0.2$ ) .....	121
Figura 5.16. Errores en las aproximaciones obtenidas mediante Splines Cúbicos ( $h = 0.2$ ) .....	122
Figura 5.17. Errores en las aproximaciones obtenidas con el Pseudo Splines Cuárticos ( $h = 0.2$ ).....	122
Figura 5.18. Gráfica en doble escala logarítmica de los errores para Splines Cúbicos en el punto $x = 0.2$ , para los distintos pasos .....	123
Figura 5.19. Gráfica en doble escala logarítmica de los errores para Splines Cúbicos en el punto $x = 0.2$ , para los distintos pasos .....	124
Figura 5.20. Gráfica de las soluciones numéricas ( $h = 0.4$ ) .....	128
Figura 5.21. Detalle de la gráfica anterior: zoom de las soluciones numéricas ( $h = 0.4$ ) .....	129
Figura 5.22. Gráfica de las soluciones numéricas ( $h = 0.125$ ) .....	131
Figura 5.23. Gráfica de las soluciones numéricas ( $h = 0.2$ ).....	132
Figura 5.24. Detalle de la gráfica anterior: zoom de las soluciones numéricas ( $h = 0.2$ ) .....	133
Figura 5.25. Gráfica de las soluciones numéricas ( $h = 0.0625$ ) .....	134
Figura 5.26. Gráfica de las soluciones numéricas $h = 0.1$ .....	135
Figura 5.27. Detalle de la gráfica anterior: zoom de las soluciones numéricas ( $h = 0.1$ ) .....	136
Figura 5.28. Gráfica de las soluciones numéricas $h = 0.003125$ .....	137
Figura 5.29. Gráfica de las soluciones numéricas para $\mu = 0.2$ $h = 0.4$ .....	139
Figura 5.30. Detalle de la gráfica anterior: zoom de las soluciones numéricas, $\mu = 0.2$ ( $h = 0.4$ ).....	140



Figura 5.31. Gráfica de las soluciones numéricas para $\mu = 0.2$ $h = 0.0125$ .....	142
Figura 5.32. Gráfica de las soluciones numéricas para $\mu = 1$ $h = 0.04$ .....	143
Figura 5.33. Detalle de la gráfica anterior: zoom de las soluciones numéricas, $\mu = 1$ ( $h = 0.4$ ).....	144
Figura 5.34. Gráfica de las soluciones numéricas para $\mu = 1$ $h = 0.0125$ .	145
Figura 5.35. Gráfica de las soluciones numéricas para $\mu = 5$ $h = 0.4$ .....	146
Figura 5.36. Solución numérica obtenida mediante Pseudo-Splines Cuárticos para $\mu = 5$ $h = 0.4$ .....	147
Figura 5.37. Detalle de la gráfica 5.36: zoom de las soluciones numéricas, $\mu = 5$ ( $h = 0.4$ ) .....	148
Figura 5.38. Gráfica de las soluciones numéricas para $\mu = 5$ $h = 0.0125$ .	149
Figura 5.39. Gráfica de las soluciones numéricas y exactas $h = 0.2$ .....	151
Figura 5.40. Detalle de la gráfica anterior: zoom de las soluciones numéricas y exacta ( $h = 0.2$ ) .....	152
Figura 5.41. Gráfica de los errores en la aproximación a la solución ( $h = 0.003125$ ).....	154
Figura 5.42. Gráfica en doble escala logarítmica del error para Splines Cúbicos en el punto $x = 0.5$ .....	155
Figura 5.43. Gráfica en doble escala logarítmica del error para Pseudo-Splines Cuárticos en $x = 0.5$ .....	155
Figura 5.44. Gráfica de las soluciones numéricas ( $h = 0.125$ ) .....	157
Figura 5.45. Detalle de la gráfica anterior: zoom de las soluciones numéricas ( $h = 0.125$ ) .....	158
Figura 5.46. Gráfica de las soluciones numéricas ( $h = 0.00390625$ ) .....	159
Figura 5.47. Gráfica de las soluciones numéricas ( $h = 0.125$ ) .....	161
Figura 5.48. Detalle de la gráfica anterior: zoom de las soluciones numéricas ( $h = 0.125$ ).....	162
Figura 5.49. Gráfica de las soluciones numéricas ( $h = 0.00390625$ ) .....	163





# ÍNDICE DE TABLAS



## ÍNDICE DE TABLAS

Tabla 5.1. Tabla resumen de los métodos desarrollados, sus características y aplicaciones .....	94
Tabla 5.2. Valores de las soluciones numéricas ( $h = 0.4$ ) .....	99
Tabla 5.3. Errores en las soluciones numéricas de los distintos métodos $h = 0.42i$ , $i = 0, 1, \dots, 7$ .....	102
Tabla 5.4. Valores de las soluciones numéricas ( $h = 0.6$ ) .....	107
Tabla 5.5. Soluciones numéricas de los distintos métodos $h = 0.62i$ , $i = 0, 1, \dots, 7$ .....	109
Tabla 5.6. Valores de las soluciones numéricas y asintótica con $n = 5$ y $\varepsilon = 10 - 4$ .....	113
Tabla 5.7. Valores de las soluciones numéricas y asintótica con $n = 12$ y $\varepsilon = 10 - 8$ .....	116
Tabla 5.8. Valores de las soluciones numéricas y exacta ( $h = 0.2$ ) .....	119
Tabla 5.9. Errores en las soluciones numéricas ( $h = 0.22i$ , $i = 0, 1, \dots, 7$ ) .....	125
Tabla 5.10. Valores de las soluciones numéricas en $x = 99, 99.8$ ( $h = 0.4$ ) .....	129
Tabla 5.11. Valores de las soluciones numéricas en $x = 99, 99.8$ $h = 0.42i$ , $i = 0, 1, \dots, 4$ .....	130
Tabla 5.12. Valores de las soluciones numéricas en $x = 199, 199.8$ ( $h = 0.2$ ): .....	133
Tabla 5.13. Valores de las soluciones numéricas en $x = 199, 199.8$ $h = 0.22i$ , $i = 0, 1, \dots, 4$ .....	134
Tabla 5.14. Valores de las soluciones numéricas en $x = 99, 99.8$ ( $h = 0.1$ ) .....	135
Tabla 5.15. Valores de las soluciones numéricas en $x = 99, 99.8$ ( $h = 0.42i$ , $i = 0, 1, \dots, 4$ ) .....	137
Tabla 5.16. Valores de las soluciones numéricas en $x = 99, 99.8$ para $\mu = 0.2$ ( $h = 0.4$ ) .....	141



Tabla 5.17. Valores de las soluciones numéricas en  $x = 99, 99.8$  para  $\mu = 0.2$   
 $h = 0.42i, i = 0, 1, \dots, 4$  .....141

Tabla 5.18. Valores de las soluciones numéricas en  $x = 99, 99.8$  para  $\mu = 1$   
 $(h = 0.4)$  .....143

Tabla 5.19. Valores de las soluciones numéricas en  $x = 99, 99.8$  para  $\mu = 1$   
 $h = 0.42i, i = 0, 1, \dots, 4$  .....144

Tabla 5.20. Valores de las soluciones numéricas en  $x = 99, 99.8$  para  $\mu = 5$   
 $h = 0.42i, i = 0, 1, \dots, 4$  .....148

Tabla 5.21. Valores de las soluciones numéricas en  $x = 0.5, 1$  ( $h = 0.2$ ) .152

Tabla 5.22. Errores en las soluciones numéricas de los métodos en  $x = 0.5, 1$   
 $h = 0.22i, i = 0, 1, \dots, 6$  .....153

Tabla 5.23. Valores de las soluciones numéricas  $h = 0.125$  .....157

Tabla 5.24. Valores de las soluciones numéricas  $h = 0.1252i, i = 0, 1, \dots, 5$   
.....158

Tabla 5.25. Valores de las soluciones numéricas ( $h = 0.125$ ) .....161

Tabla 5.26. Valores de las soluciones numéricas ( $h = 0.1252i, i = 0, 1, \dots, 5$ )  
.....162



# ÍNDICE DE ECUACIONES







## ÍNDICE DE ECUACIONES

Ecuación ( 2.1 )	12
Ecuación ( 2.2 )	12
Ecuación ( 2.3 )	13
Ecuación ( 2.4 )	13
Ecuación ( 2.5 )	15
Ecuación ( 2.6 )	16
Ecuación ( 2.7 )	16
Ecuación ( 2.8 )	16
Ecuación ( 2.9 )	17
Ecuación ( 2.10 )	17
Ecuación ( 2.11 )	17
Ecuación ( 2.12 )	18
Ecuación ( 2.13 )	18
Ecuación ( 2.14 )	18
Ecuación ( 2.15 )	18
Ecuación ( 2.16 )	20
Ecuación ( 2.17 )	20
Ecuación ( 2.18 )	20
Ecuación ( 2.19 )	20
Ecuación ( 2.20 )	20
Ecuación ( 2.21 )	21
Ecuación ( 2.22 )	21
Ecuación ( 2.23 )	21
Ecuación ( 2.24 )	21
Ecuación ( 2.25 )	22
Ecuación ( 2.26 )	22
Ecuación ( 2.27 )	23
Ecuación ( 2.28 )	23
Ecuación ( 2.29 )	23
Ecuación ( 2.30 )	23
Ecuación ( 2.31 )	23
Ecuación ( 2.32 )	23
Ecuación ( 2.33 )	24
Ecuación ( 2.34 )	24
Ecuación ( 2.35 )	24
Ecuación ( 2.36 )	25
Ecuación ( 2.37 )	25
Ecuación ( 2.38 )	25
Ecuación ( 2.39 )	26
Ecuación ( 2.40 )	26
Ecuación ( 2.41 )	27
Ecuación ( 2.42 )	28
Ecuación ( 2.43 )	28



Ecuación ( 2.44 ).....	28
Ecuación ( 2.45 ).....	29
Ecuación ( 2.46 ).....	29
Ecuación ( 2.47 ).....	29
Ecuación ( 2.48 ).....	29
Ecuación ( 2.49 ).....	29
Ecuación ( 2.50 ).....	29
Ecuación ( 2.51 ).....	30
Ecuación ( 2.52 ).....	30
Ecuación ( 2.53 ).....	30
Ecuación ( 2.54 ).....	30
Ecuación ( 2.55 ).....	31
Ecuación ( 3.1 ).....	36
Ecuación ( 3.2 ).....	37
Ecuación ( 3.3 ).....	37
Ecuación ( 3.4 ).....	37
Ecuación ( 3.5 ).....	37
Ecuación ( 3.6 ).....	37
Ecuación ( 3.7 ).....	38
Ecuación ( 3.8 ).....	38
Ecuación ( 3.9 ).....	38
Ecuación ( 3.10 ).....	38
Ecuación ( 3.11 ).....	38
Ecuación ( 3.12 ).....	38
Ecuación ( 3.13 ).....	39
Ecuación ( 3.14 ).....	39
Ecuación ( 3.15 ).....	40
Ecuación ( 3.16 ).....	40
Ecuación ( 3.17 ).....	40
Ecuación ( 3.18 ).....	40
Ecuación ( 3.19 ).....	40
Ecuación ( 3.20 ).....	40
Ecuación ( 3.21 ).....	41
Ecuación ( 3.22 ).....	41
Ecuación ( 3.23 ).....	41
Ecuación ( 3.24 ).....	41
Ecuación ( 3.25 ).....	42
Ecuación ( 3.26 ).....	42
Ecuación ( 3.27 ).....	42
Ecuación ( 3.28 ).....	42
Ecuación ( 3.29 ).....	43
Ecuación ( 3.30 ).....	43
Ecuación ( 3.31 ).....	43
Ecuación ( 3.32 ).....	44



Ecuación ( 3.33 ).....	44
Ecuación ( 3.34 ).....	44
Ecuación ( 3.35 ).....	45
Ecuación ( 3.36 ).....	45
Ecuación ( 3.37 ).....	45
Ecuación ( 3.38 ).....	45
Ecuación ( 3.39 ).....	45
Ecuación ( 3.40 ).....	45
Ecuación ( 3.41 ).....	46
Ecuación ( 3.42 ).....	46
Ecuación ( 3.43 ).....	46
Ecuación ( 3.44 ).....	46
Ecuación ( 3.45 ).....	47
Ecuación ( 3.46 ).....	47
Ecuación ( 3.47 ).....	47
Ecuación ( 3.48 ).....	47
Ecuación ( 3.49 ).....	47
Ecuación ( 3.50 ).....	48
Ecuación ( 3.51 ).....	48
Ecuación ( 3.52 ).....	49
Ecuación ( 3.53 ).....	49
Ecuación ( 3.54 ).....	49
Ecuación ( 3.55 ).....	49
Ecuación ( 3.56 ).....	50
Ecuación ( 3.57 ).....	50
Ecuación ( 3.58 ).....	50
Ecuación ( 3.59 ).....	50
Ecuación ( 3.60 ).....	51
Ecuación ( 3.61 ).....	52
Ecuación ( 3.62 ).....	52
Ecuación ( 3.63 ).....	52
Ecuación ( 3.64 ).....	53
Ecuación ( 3.65 ).....	53
Ecuación ( 3.66 ).....	53
Ecuación ( 3.67 ).....	53
Ecuación ( 3.68 ).....	53
Ecuación ( 3.69 ).....	53
Ecuación ( 3.70 ).....	53
Ecuación ( 3.71 ).....	54
Ecuación ( 3.72 ).....	54
Ecuación ( 3.73 ).....	54
Ecuación ( 3.74 ).....	54
Ecuación ( 3.75 ).....	55
Ecuación ( 3.76 ).....	55



Ecuación ( 3.77 ).....	56
Ecuación ( 3.78 ).....	57
Ecuación ( 3.79 ).....	57
Ecuación ( 3.80 ).....	57
Ecuación ( 3.81 ).....	58
Ecuación ( 3.82 ).....	58
Ecuación ( 3.83 ).....	58
Ecuación ( 3.84 ).....	59
Ecuación ( 3.85 ).....	59
Ecuación ( 3.86 ).....	59
Ecuación ( 3.87 ).....	59
Ecuación ( 3.88 ).....	59
Ecuación ( 3.89 ).....	60
Ecuación ( 3.90 ).....	60
Ecuación ( 3.91 ).....	60
Ecuación ( 3.92 ).....	61
Ecuación ( 3.93 ).....	62
Ecuación ( 3.94 ).....	62
Ecuación ( 3.95 ).....	62
Ecuación ( 3.96 ).....	62
Ecuación ( 3.97 ).....	63
Ecuación ( 3.98 ).....	63
Ecuación ( 3.99 ).....	63
Ecuación ( 3.100 ).....	63
Ecuación ( 3.101 ).....	63
Ecuación ( 3.102 ).....	64
Ecuación ( 3.103 ).....	64
Ecuación ( 3.104 ).....	65
Ecuación ( 3.105 ).....	65
Ecuación ( 3.106 ).....	65
Ecuación ( 3.107 ).....	65
Ecuación ( 3.108 ).....	65
Ecuación ( 3.109 ).....	66
Ecuación ( 3.110 ).....	66
Ecuación ( 3.111 ).....	67
Ecuación ( 3.112 ).....	67
Ecuación ( 3.113 ).....	67
Ecuación ( 3.114 ).....	67
Ecuación ( 3.115 ).....	67
Ecuación ( 3.116 ).....	67
Ecuación ( 3.117 ).....	68
Ecuación ( 3.118 ).....	68
Ecuación ( 3.119 ).....	68
Ecuación ( 3.120 ).....	68



Ecuación ( 3.121 ) .....	68
Ecuación ( 3.122 ) .....	68
Ecuación ( 3.123 ) .....	68
Ecuación ( 3.124 ) .....	69
Ecuación ( 3.125 ) .....	69
Ecuación ( 3.126 ) .....	69
Ecuación ( 3.127 ) .....	69
Ecuación ( 3.128 ) .....	69
Ecuación ( 3.129 ) .....	69
Ecuación ( 3.130 ) .....	70
Ecuación ( 3.131 ) .....	70
Ecuación ( 3.132 ) .....	70
Ecuación ( 3.133 ) .....	70
Ecuación ( 3.134 ) .....	70
Ecuación ( 3.135 ) .....	71
Ecuación ( 3.136 ) .....	71
Ecuación ( 3.137 ) .....	71
Ecuación ( 3.138 ) .....	72
Ecuación ( 3.139 ) .....	72
Ecuación ( 4.1 ) .....	75
Ecuación ( 4.2 ) .....	75
Ecuación ( 4.3 ) .....	77
Ecuación ( 4.4 ) .....	77
Ecuación ( 4.5 ) .....	77
Ecuación ( 4.6 ) .....	77
Ecuación ( 4.7 ) .....	78
Ecuación ( 4.8 ) .....	78
Ecuación ( 4.9 ) .....	79
Ecuación ( 4.10 ) .....	79
Ecuación ( 4.11 ) .....	79
Ecuación ( 4.12 ) .....	79
Ecuación ( 4.13 ) .....	81
Ecuación ( 4.14 ) .....	81
Ecuación ( 4.15 ) .....	81
Ecuación ( 4.16 ) .....	81
Ecuación ( 4.17 ) .....	82
Ecuación ( 4.18 ) .....	82
Ecuación ( 4.19 ) .....	82
Ecuación ( 4.20 ) .....	82
Ecuación ( 4.21 ) .....	82
Ecuación ( 4.22 ) .....	83
Ecuación ( 4.23 ) .....	83
Ecuación ( 4.24 ) .....	83
Ecuación ( 4.25 ) .....	83



Ecuación ( 4.26 ).....	84
Ecuación ( 4.27 ).....	84
Ecuación ( 4.28 ).....	85
Ecuación ( 4.29 ).....	85
Ecuación ( 4.30 ).....	85
Ecuación ( 4.31 ).....	86
Ecuación ( 4.32 ).....	86
Ecuación ( 4.33 ).....	86
Ecuación ( 4.34 ).....	87
Ecuación ( 4.35 ).....	87
Ecuación ( 4.36 ).....	87
Ecuación ( 4.37 ).....	87
Ecuación ( 4.38 ).....	87
Ecuación ( 4.39 ).....	87
Ecuación ( 4.40 ).....	88
Ecuación ( 4.41 ).....	88
Ecuación ( 4.42 ).....	88
Ecuación ( 5.1 ).....	98
Ecuación ( 5.2 ).....	98
Ecuación ( 5.3 ).....	106
Ecuación ( 5.4 ).....	111
Ecuación ( 5.5 ).....	112
Ecuación ( 5.6 ).....	118
Ecuación ( 5.7 ).....	127
Ecuación ( 5.8 ).....	127
Ecuación ( 5.9 ).....	131
Ecuación ( 5.10 ).....	138
Ecuación ( 5.11 ).....	139
Ecuación ( 5.12 ).....	150
Ecuación ( 5.13 ).....	150
Ecuación ( 5.14 ).....	156
Ecuación ( 5.15 ).....	160









# INTRODUCCIÓN Y OBJETIVOS

## Capítulo 1



## 1. JUSTIFICACIÓN Y OBJETIVOS

### 1.1. Justificación <sup>[2]</sup> <sup>[9]</sup> <sup>[11]</sup>

Las Ecuaciones Diferenciales juegan un papel fundamental en las matemáticas, y muy especialmente en sus aplicaciones. Muchos de los problemas que surgen en el ámbito de la ciencia, la ingeniería, la industria y de la tecnología pueden formularse en términos de ecuaciones de este tipo, por lo que saber resolverlas cobra una importancia capital. Desde los cálculos que requiere la construcción de maquinaria eléctrica o de dispositivos electrónicos, hasta el cálculo de trayectorias de proyectiles, la investigación de la estabilidad de maquinaria o de aeronaves en vuelo, el curso de una reacción química, el crecimiento de poblaciones e incluso el avance de ciertas enfermedades, por mencionar solo algunos ejemplos, precisan del cálculo o aproximación de soluciones de Ecuaciones Diferenciales.

Es más, con frecuencia las leyes físicas que gobiernan fenómenos se escriben en forma de ecuaciones diferenciales, por lo que éstas, en sí, constituyen una expresión cuantitativa de dichas leyes. Por poner solo algunos ejemplos: las ecuaciones del movimiento de cuerpos (Segunda Ley de Newton), la ecuación que describe los sistemas oscilantes, la propagación de ondas, la transmisión de calor, la difusión, la conservación de masa o el movimiento de partículas subatómicas, se formulan en términos de Ecuaciones Diferenciales Ordinarias de Segundo Orden.

La teoría de las Ecuaciones Diferenciales comenzó a desarrollarse a finales del siglo XVII, casi simultáneamente con la aparición del cálculo diferencial e integral. En la actualidad, las Ecuaciones Diferenciales se han convertido en una herramienta poderosa para la investigación. En Mecánica, Astronomía, Física, Tecnología etc., han propiciado un enorme progreso. A modo de ejemplo, del estudio de las ecuaciones diferenciales del movimiento de los cuerpos celestes, Newton dedujo las leyes del movimiento planetario descubiertas empíricamente por Kepler. También, en 1846, Le Verrier predijo la existencia del planeta Neptuno y determinó su posición en el cielo, basándose en el análisis numérico de esas mismas ecuaciones.

Como vemos, no es exagerado afirmar que la naturaleza se describe por medio de ecuaciones diferenciales, de modo que un conocimiento de esta materia nos ayudará a entender mejor los fenómenos naturales. Debido a lo anterior, los esfuerzos de los científicos se han dirigido desde un principio, a la búsqueda de métodos de resolución y expresión de las soluciones de estas ecuaciones en forma adecuada.

De este modo, los primeros métodos de resolución fueron analíticos. Se buscaba la expresión analítica de la solución exacta, expresada en términos de funciones conocidas denominadas funciones elementales. Para ello se recurría al cálculo de primitivas, a los cambios de variable, a manipulaciones algebraicas, etc. Pronto se comprobó que la mayoría de los problemas de este tipo no admitían soluciones en términos de funciones elementales. Ante la imposibilidad de resolver muchas de estas ecuaciones de manera exacta, surgieron los métodos numéricos orientados a aproximar dichas soluciones de manera eficiente y precisa.

Resumiendo, los métodos de resolución analíticos, permiten expresar la solución en forma exacta, como una función de la variable independiente, en tanto que los numéricos tienen como objetivo calcular valores que toma la solución en una serie de puntos u obtener una función que, sin ser la solución exacta, la aproxime adecuadamente con la precisión deseada.

La necesidad de recurrir a métodos alternativos a los analíticos obedece, como hemos adelantado, a que, con la excepción de unos cuantos casos más o menos simples, la inmensa mayoría de las Ecuaciones Diferenciales no puede ser resuelta satisfactoriamente en forma exacta. Así, en un sistema tan simple como un péndulo, la amplitud de la oscilación ha de ser pequeña y el rozamiento ha de ser despreciable, para obtener una solución sencilla que describa aproximadamente su movimiento periódico. Sin embargo, su comportamiento puede ser estudiado, sin necesidad de estas simplificaciones, aplicando métodos numéricos.

Es por ello, que, en este Trabajo de Fin de Grado, vamos a profundizar en el análisis y desarrollo de diversos métodos numéricos de resolución de Ecuaciones Diferenciales, centrándonos, concretamente, en las de segundo orden, cuya resolución continúa hoy en día estudiándose y es objeto de muchos trabajos de investigación. No ha resultado difícil encontrar diversas publicaciones científicas recientes, de no más de diez años atrás, que demuestran lo anterior. Por ejemplo, en relación con varios de los métodos numéricos que desarrollaremos más adelante, basados en Splines, pueden leerse varias publicaciones recogidas en la bibliografía, en el apartado de Artículos y Publicaciones Científicas: [18]-[30], donde además puede observarse como varias de ellas han sido presentadas en recientes congresos científicos.

Tras un breve repaso de algunos de los métodos numéricos clásicos de los que se dispone a la hora de resolver Ecuaciones Diferenciales, estudiaremos la resolución de dichas ecuaciones con nuevos métodos que proponemos.

Utilizaremos, para su implementación, el software matemático Matlab, puesto que la aparición de Matlab ha constituido un hito fundamental en los softwares destinados al Análisis Numérico.

Matlab comenzó como un paquete interactivo de cálculos matriciales, de ahí su nombre: MATrix LABoratory, desarrollado inicialmente por C. Moler con fines docentes; pero rápidamente cobró notoriedad entre los científicos e ingenieros como una ayuda inestimable en el cálculo científico. Por ello, fue rediseñado a mediados de los años 80, convirtiéndose desde ese momento en un software comercial que ofrece un entorno de desarrollo integrado (I.D.E), con un lenguaje de programación de alto nivel propio para el cálculo científico (lenguaje M). Incorpora capacidades de visualización, gran variedad de bibliotecas, así como otras funcionalidades: manejo de matrices dispersas, conexión directa con hardware de control, o el módulo Simulink, destinado al diseño y simulación de sistemas de control.

Este software da acceso a algoritmos muy eficientes, con pocas líneas de código y permite a los especialistas centrarse directamente en la esencia matemática del problema, sin preocuparse por aspectos puramente computacionales tales como la declaración de variables o manejo de memoria, lo que le ha permitido convertirse en todo un estándar en la industria. De hecho, se afirma que una buena cantidad de los cálculos para el diseño de los transbordadores espaciales de la NASA es realizada con Matlab.

En nuestro caso, Matlab nos permitirá programar los distintos métodos desarrollados para la resolución de las E.D.O de Segundo Orden, y comprobar, de modo más rápido, sencillo y visual, cómo se comporta cada uno de ellos, dependiendo del tipo de problemas, sus parámetros o sus condiciones.

Los nuevos métodos que proponemos, permiten obtener aproximaciones adecuadas a las soluciones de Problemas de Valores Iniciales (P.V.I) y Problemas de Valores en la Frontera (P.V.F) gobernados por Ecuaciones Diferenciales de Segundo Orden. Se basan en la aproximación de las soluciones de E.D.O.s, lineales y no lineales, de 2º orden mediante Splines y Pseudo-Splines de clase  $C^2$  (la función y sus dos primeras derivadas son continuas), utilizando técnicas de colocación. Para resolver los P.V.F. se combinan técnicas basadas en el Método del Disparo, con otras basadas en resolver sistemas lineales tridiagonales mediante el Algoritmo de Thomas.

Finalmente, se incluye un buen número de experimentos numéricos, implementados en Matlab, la mayoría del ámbito de la Ingeniería, para analizar la precisión y exactitud de los nuevos métodos propuestos, así como su orden de convergencia.

## 1.2. Objetivos

En este trabajo de fin de grado se proponen los siguientes objetivos:

- Analizar en profundidad la importancia de las Ecuaciones Diferenciales de Segundo Orden en el modelado de fenómenos comunes, mostrando diversos ejemplos relacionados con la ingeniería.
- Estudiar diferentes métodos de resolución de Problemas de Valores Iniciales y Problemas de Valores en la Frontera, formulados en términos de Ecuaciones Diferenciales Ordinarias de Segundo Orden, que aparecen en los modelos mencionados.
- Proponer nuevos métodos numéricos para la resolución de Ecuaciones Diferenciales de Segundo Orden, acompañadas de condiciones iniciales o de contorno.
- Implementar adecuadamente cada uno de los métodos anteriores en código de Matlab, de modo que facilite su utilización para poder realizar simulaciones con un grado de precisión adecuado mediante dicho software.
- Aplicar, recurriendo a Matlab, estos métodos de resolución a diversos modelos correspondientes a problemas de interés en ingeniería, y analizar los resultados obtenidos.
- Observar e interpretar el comportamiento de las soluciones ante modificaciones de los parámetros del modelo.
- Elaborar conclusiones sobre la aplicación de cada método y sus variantes, a partir de las soluciones obtenidas con Matlab.
- En base a las conclusiones anteriores, describir qué ventajas podrían aportar las nuevas variantes que han sido propuestas respecto a los métodos tradicionales que se utilizan hoy en día.

Todo ello, sin olvidar la principal intencionalidad de todo trabajo de fin de grado, que es su finalidad educativa y formativa.

### 1.3. Estructura

El presente trabajo tiene dos partes bien diferenciadas. La primera es teórica y trata de explicar las ecuaciones que van a manejarse, los tipos de problemas a los que pueden dar lugar, y los diferentes métodos numéricos para resolverlos. Posteriormente, en la segunda parte, que es completamente experimental, se aplican dichos métodos a diferentes problemas, mediante el software Matlab, para comprobar su validez.

Capítulo 1: Justificación y objetivos. Expone brevemente el fundamento de este proyecto y los objetivos que se persiguen con su desarrollo.

Capítulo 2: Ecuaciones Diferenciales Ordinarias de Segundo Orden. Consideramos necesaria una breve introducción teórica sobre qué son las Ecuaciones Diferenciales de Segundo Orden, su importancia, los diferentes tipos de problema asociados a ellas, y los métodos que se conocen para su resolución.

Capítulo 3: Métodos de resolución de E.D.O.s basados en Splines. Se introducen de manera teórica los Splines como métodos de resolución de E.D.O.s. Se estudia su utilización para distintos problemas, y se proponen nuevas variantes y nuevos métodos basados en Pseudo-Splines Cuárticos.

Capítulo 4: Método del Disparo. En este capítulo se realiza una explicación teórica del Método del Disparo Lineal y No Lineal, aplicado a Problemas de Valores en la Frontera.

Capítulo 5: Experimentos numéricos. Para evaluar los métodos numéricos propuestos, en este capítulo se muestran una serie de ejemplos de problemas, presentando los resultados obtenidos al resolverlos con dichos métodos, así como los resultados derivados de su análisis.


Capítulo 6: Conclusiones. Se presentan las conclusiones finales del trabajo, obtenidas a partir de los resultados explicados en el capítulo anterior. También se presentan posibles líneas de continuación de este proyecto.

Bibliografía. Por último, se muestra un listado de todos los libros, conferencias, artículos, direcciones de Internet, programas, etc., consultados para la elaboración de este proyecto.

Anexo: Código de Matlab utilizado en las simulaciones. Incluyen los códigos de las diferentes funciones de Matlab en los que se han implementado los métodos numéricos propuestos, así como los programas donde se recogen los diferentes ejemplos numéricos que se han utilizado.







# ECUACIONES DIFERENCIALES DE SEGUNDO ORDEN

## Capítulo 2



## 2. ECUACIONES DIFERENCIALES DE SEGUNDO ORDEN

### 2.1. Introducción a las Ecuaciones Diferenciales de Segundo Orden <sup>[9]</sup> <sup>[13]</sup>

Como ya hemos mencionado en el capítulo anterior, en las ciencias y en muchas de sus aplicaciones, incluyendo la ingeniería, aparecen numerosos problemas que se resuelven a partir de leyes y relaciones físicas que requieren del desarrollo de modelos matemáticos para su comprensión. Con frecuencia, estos modelos se describen en términos de ecuaciones que contienen derivadas de distintos órdenes de una función incógnita, esto es, mediante Ecuaciones Diferenciales.

Las Ecuaciones Diferenciales involucran relaciones, válidas en cierto intervalo, entre una variable (la variable dependiente o función incógnita) y sus derivadas sucesivas. Proporcionan, por tanto, descripciones matemáticas de cómo las variables y sus derivadas se relacionan entre sí de manera dinámica. Su resolución permite estudiar las características de los sistemas que modelan y una misma ecuación puede describir procesos correspondientes a diversas disciplinas.

Las ecuaciones diferenciales se pueden clasificar, básicamente, atendiendo a dos criterios:

- a) Tipo: Si la función incógnita viene dada en términos de una única variable independiente, entonces la ecuación se denomina ecuación diferencial ordinaria, abreviadamente E.D.O. En otro caso, cuando la función incógnita depende de dos o más variables independientes, la ecuación se dice que es una ecuación (diferencial) en derivadas parciales, abreviadamente E.D.P.
- b) Orden: Es el orden de la derivada de orden más alto que aparece en la ecuación diferencial.

En nuestro caso, pretendemos abordar el estudio de las Ecuaciones Diferenciales Ordinarias de Segundo Orden, pues éstas aparecen con mayor frecuencia en múltiples aplicaciones del ámbito de la ciencia y la ingeniería.

Concretamente, empezaremos estudiando en detalle la Ecuación Diferencial Ordinaria Lineal de Segundo Orden:

$$y'' = p(x) \cdot y' + q(x) \cdot y + r(x) \quad x \in [a, b] \quad (2.1)$$

donde  $x$  es la variable independiente que pertenece al intervalo  $[a, b]$ , e  $y = y(x)$  es la variable dependiente (función incógnita).

De aquí en adelante utilizaremos la notación de Lagrange para las derivadas, denotando la primera y segunda derivadas de la variable dependiente  $y$ , respectivamente:

$$y' = \frac{dy}{dx} \quad , \quad y'' = \frac{d^2y}{dx^2} \quad (2.2)$$

Por otra parte, supondremos que  $p(x)$ ,  $q(x)$ ,  $r(x)$  son funciones que están definidas y son continuas en un cierto intervalo. En particular, podrán ser constantes en algunos casos, y entonces hablaremos de Ecuaciones Diferenciales Lineales con coeficientes constantes, y variables en otros, tratándose entonces de Ecuaciones Diferenciales Lineales de coeficientes variables.

Los problemas en los que resultan ser parámetros constantes, son fáciles de estudiar, pues se pueden resolver de manera exacta mediante procedimientos sencillos.

El caso variable es más difícil de estudiar y resolver; pero tiene múltiples aplicaciones de interés que, en algunos casos, permiten estudiar modelos más fieles a la realidad. De hecho, aunque algunas ecuaciones de este tipo pueden resolverse analíticamente de manera exacta, la solución viene dada con frecuencia en términos de cuadraturas que no permiten evaluar dicha solución de manera sencilla sin recurrir a métodos de integración numérica.

Cuando  $r(x) = 0$  para todo  $x$ , se dice que la Ecuación Diferencial es homogénea. En caso contrario, la ecuación se dice que es no homogénea.

## 2.2. Introducción a los métodos numéricos [4] [5] [9] [6]

### 2.2.1. Condiciones de aplicabilidad de métodos numéricos

Para poder aplicar métodos numéricos a la resolución de Ecuaciones Diferenciales Ordinarias, debemos trabajar con problemas bien puestos, (bien planteados), esto es, problemas que tienen una única solución que depende continuamente de los datos. Aunque nuestro estudio se centra en E.D.O.s de 2º orden, es bien conocido que este tipo de ecuaciones pueden reescribirse en términos de un sistema de Ecuaciones Diferenciales de primer orden, a costa, eso sí, de aumentar la dimensión del sistema de ecuaciones resultante.

Es por ello que empezaremos considerando el Problema de Valores Iniciales dado por:

$$\begin{cases} y' = f(x, y) \\ y(a) = z_1 \end{cases}, \quad x \in [a, b] \quad (2.3)$$

Se considera que un problema del tipo anterior está bien puesto si satisface el siguiente teorema:

**Teorema 2.1.-** Sea  $f(x, y)$  una función definida y continua en  $D_{ab}$ :

$$D_{ab} = \{(x, \vec{y}): a \leq x \leq b, y_i \in \mathbb{R}, i = 1, \dots, m\} \text{ (a, b finitos)}$$

para la que existe una constante  $L$  tal que:

$$\|\vec{f}(x, \vec{y}) - \vec{f}(x, \vec{y}^*)\| \leq L \|\vec{y} - \vec{y}^*\|, \quad \forall (x, \vec{y}), (x, \vec{y}^*) \in D_{ab} \quad (2.4)$$

esto es,  $f$  satisface una condición Lipschitz en  $D_{ab}$ , con constante de Lipschitz  $L$ . Entonces el problema está bien puesto, es decir:

- Existe una solución única del problema.
- El problema es totalmente estable.

La condición (2.4) es conocida como condición de Lipschitz y la constante  $L$  como constante de Lipschitz. Si se tiene que  $\vec{f}(x, \vec{y})$  es diferenciable en  $D_{ab}$  entonces  $\vec{f}(x, \vec{y})$  satisface la condición de Lipschitz en  $D_{ab}$ , lo que implica que  $\vec{f}(x, \vec{y})$  es continua en  $D_{ab}$ . Por tanto, la condición que se requiere es un poco más que la continuidad y un poco menos que la diferenciabilidad.

Si el problema no fuese totalmente estable, ningún método numérico tendría posibilidad de producir una solución aceptable, ya que cualquier método numérico introduce errores debidos a la discretización y al redondeo, lo que podría interpretarse como un problema perturbado.

### 2.2.2. Nomenclatura en métodos numéricos

Los métodos numéricos que vamos a estudiar para resolver nuestros problemas de Ecuaciones Diferenciales Ordinarias, se basan en la idea de la discretización, es decir, el intervalo continuo  $[a, b]$  de  $t$  es reemplazado por el conjunto de puntos  $a = x_0 < x_1 < x_2 < \dots < x_N = b$ . En el caso equiespaciado, se tiene que:  $x_n = a + n \cdot h, n = 0, 1, \dots, N$ , con  $h = \frac{b-a}{N}$ , y a  $h$  se le denomina tamaño de paso.

Denotaremos por  $\vec{y}_n$  a una aproximación a la solución  $\vec{y}(x_n)$  del problema en el punto  $x_n$ , es decir:  $\vec{y}_n \approx \vec{y}(x_n)$ .

Nuestro objetivo es obtener secuencias de valores  $\{\vec{y}_n : n = 0, 1, 2, \dots, N\}$  que aproximen la solución del problema considerado, en el conjunto discreto de puntos  $\{x_n : n = 0, 1, 2, \dots, N\}$ ; una secuencia de este tipo constituye una solución numérica.

Un método numérico es una ecuación en diferencias que contiene un número consecutivo de aproximaciones  $\vec{y}_{n+j}, j = 0, 1, \dots, k$  de la que es posible calcular secuencialmente la sucesión  $\{\vec{y}_n : n = 0, 1, 2, \dots, N\}$ ; naturalmente la ecuación en diferencias incluirá la función  $\vec{f}$  que define el problema.

Al entero  $k$  se le denomina número de pasos del método. Si  $k = 1$ , se dice que el método es de un paso, y a partir de las condiciones iniciales que proporcione el problema se puede obtener secuencialmente la sucesión  $\{y_n\}$  calculando las aproximaciones para  $n = 0, 1, 2, \dots, N$  a partir de la ecuación en diferencias.

Por otra parte, si  $k > 1$ , el método se denomina multipaso o de  $k$ -pasos. Si por ejemplo  $k = 2$ , el método es de dos pasos y es necesario proporcionar un valor de arranque adicional  $y_1$ , antes de empezar a calcular la sucesión  $\{y_n\}$ . Si fuera  $k = 3$ , se trataría de un método de tres pasos y sería necesario proporcionar dos valores adicionales de arranque  $y_1$  e  $y_2$ .

Si el método es tal que, dados  $\vec{y}_{n+j}$ ,  $j = 0, 1, \dots, k - 1$ , la ecuación en diferencias proporciona  $y_{n+k}$  explícitamente, se dice que el método es explícito. Si el valor  $y_{n+k}$  no se puede calcular sin resolver un sistema de ecuaciones implícitas, es decir, de ecuaciones no lineales, se dice que el método es implícito. En estos casos, la solución de un sistema no lineal en cada paso hace que los métodos sean más costosos de implementar que los explícitos.

Un algoritmo es un código de ordenador que implementa el método numérico. Además de calcular la sucesión  $\{\vec{y}_n : n = 0, 1, 2, \dots, N\}$  puede desarrollar otras tareas, como, por ejemplo: la estimación del error en la aproximación, cambiar el valor de  $h$  o decidir qué familia de métodos emplear en cada etapa de la solución.

### 2.3. Problemas de Valores Iniciales [3] [9] [17]

Para poder resolver la ecuación (2.1) se requieren ciertas condiciones iniciales. Al tratarse de una Ecuación Diferencial Ordinaria de Segundo Orden, serán necesarias exactamente dos condiciones para determinar la solución de manera única.

Es habitual encontrar problemas en los cuales las condiciones iniciales vienen dadas en términos de los valores que toma la solución (variable dependiente) y su primera derivada en el punto inicial  $x_0 = a$  del intervalo al que pertenece la variable independiente  $x$ , es decir, son del tipo:

$$y(x_0) = z_1, \quad y'(x_0) = z_2 \quad (2.5)$$

El problema consistente en resolver la ecuación (2.1) a partir de estas dos condiciones iniciales, (2.5), se denomina Problema de Valores Iniciales (abreviadamente P.V.I, o en inglés I.V.P).

Es bien conocido que la resolución de los P.V.I lineales del tipo considerado, en el caso homogéneo ( $r(x) = 0$ ), se apoya en dos resultados básicos: la combinación lineal de dos soluciones es otra solución, y toda solución es combinación lineal de dos soluciones independientes.

En base a lo anterior, la solución general de la ecuación (2.1), la cual vamos a reescribir de la siguiente forma:

$$y'' + p(x) \cdot y + q(x) \cdot y = r(x) \quad (2.6)$$

es sabido, que puede obtenerse como:

$$y(x) = y_h(x) + y_p(x) \quad (2.7)$$

Es decir, puede escribirse como la suma de la solución general,  $y_h$ , de la ecuación homogénea asociada a dicha ecuación:

$$y'' + p(x) \cdot y + q(x) \cdot y = 0 \quad (2.8)$$

con una solución particular  $y_p(x)$ . Esta última solución se calculará, cuando sea posible, por alguno de los muchos procedimientos disponibles. Puede obtenerse más información en la bibliografía, especialmente en las referencias [9], en su capítulo 2; en [3], capítulo 5.

En muchas de las situaciones de la vida real, la Ecuación Diferencial que modela el problema resulta imposible o demasiado complicada de resolver con exactitud, por lo que se hace necesario recurrir a procedimientos alternativos que permitan aproximar su solución.

Cuando la Ecuación Diferencial es no lineal, una posibilidad consiste en simplificar la ecuación diferencial, por ejemplo, linealizándola, de modo que podamos resolverla exactamente y utilizar después la solución de la ecuación simplificada para aproximar la solución de la ecuación original. Otra posibilidad, que examinaremos en este apartado, consiste en utilizar métodos numéricos para aproximar la solución del problema original. Este procedimiento es el que se emplea por lo regular, pues los métodos numéricos de aproximación dan resultados más exactos y en muchos casos información realista sobre el error cometido.

La mayoría de los métodos de tipo numérico que se emplean en la práctica, no producen una aproximación continua a la solución del Problema de Valor Inicial, sino que obtienen aproximaciones en algunos puntos específicos denominados puntos de red, que a menudo se seleccionan de modo que se encuentren igualmente espaciados. Si se requiere de valores intermedios, se utiliza entonces algún método de interpolación o se refina la red de puntos considerada.



A la hora de resolver numéricamente un P.V.I. cuya ecuación involucre derivadas de orden mayor que uno, como, por ejemplo:

$$\begin{cases} y'' = f(x, y, y') \\ y(x_0) = z_1 \\ y'(x_0) = z_2 \end{cases} \quad (2.9)$$

una estrategia muy utilizada en la práctica consiste en reescribir el problema en términos de un sistema de Ecuaciones Diferenciales de primer orden, pues para la resolución de este tipo de problemas se dispone de multitud de métodos numéricos, muy estudiados y, en general, muy efectivos. De hecho, posteriormente vamos a describir algunos de los métodos disponibles para resolver problemas formulados en términos de sistemas de Ecuaciones Diferenciales de primer orden.

En el caso que nos ocupa, podemos reescribir el P.V.I. de segundo orden (2.9) en forma de P.V.I. de primer orden, sin más que definir  $u(x) = y'(x)$ , obteniendo así:

$$\begin{cases} y' = u \\ u' = f(x, y, u) \\ y(x_0) = z_1 \\ u(x_0) = z_2 \end{cases} \quad (2.10)$$

A continuación, describiremos algunos de los métodos numéricos más comúnmente utilizados para resolver este tipo de problemas, puesto que después, nos serán de gran utilidad para desarrollar otros métodos numéricos en los que vamos a centrar este trabajo. Si se busca una explicación más exhaustiva sobre estos métodos se recomienda acudir a la bibliografía, destacamos [17], en su capítulo 6, o [3], capítulo 5.

### 2.3.1. Método de Euler Explícito

El método de Euler o de las Tangentes, es un método explícito que constituye el primer y más sencillo ejemplo de método numérico para la resolución de un P.V.I de primer orden del tipo:

$$\begin{cases} y'(x) = f(x, y(x)) \\ y(x_0) = y_0 = z_1 \end{cases} \quad (2.11)$$

Hay muy diversas formas de llegar a la expresión que define dicho método: a partir de la fórmula de derivación numérica de dos puntos, de un desarrollo de Taylor truncado, etc.

Nosotros lo haremos aquí interpretando la Ecuación Diferencial Ordinaria (E.D.O) como un campo de direcciones en el plano  $(x - y)$  y la condición inicial  $y(x_0) = y_0$  como punto  $(x_0, y_0)$  de dicho plano, con lo que podemos, en principio, aproximar la función solución  $y(x)$  por medio de la recta tangente a la misma que pasa por ese punto, esto es:

$$y(x) \cong y_0 + f(x_0, y_0)(x - x_0) \quad (2.12)$$

siendo la pendiente de dicha tangente:  $m = y'(x_0) = f(x_0, y_0)$ . Calculamos así, de manera aproximada, el valor de la solución  $y$  en el punto de abscisa  $x_1$  como:

$$y(x_1) \cong y_1 = y_0 + f(x_0, y_0)(x_1 - x_0) \quad (2.13)$$

A partir de este punto  $y_1$  calculado, podemos repetir el procedimiento anterior para obtener así otro punto aproximado  $(x_2, y_2)$ :

$$y(x_2) \cong y_2 = y_1 + f(x_1, y_1)(x_2 - x_1) \quad (2.14)$$

y así, iterando el proceso, podríamos ir calculando las aproximaciones que buscamos a la solución del problema en la sucesión de abscisas considerada.

Cuando no hay razones que lo desaconsejen, es habitual implementar este método tomando  $N + 1$  puntos de red equiespaciados (incluyendo el  $x_0$ ), es decir, calculando aproximaciones a la solución del problema en los puntos:  $x_{n+1} = x_n + h = x_0 + n \cdot h$ , siendo  $h$  el paso del método, que se calcula como:  $h = (a - b) / N$ . De esta forma se obtienen las fórmulas que nos proporcionan la solución aproximada a partir de la ecuación en diferencias:

$$\begin{cases} y_0 = z_1 \\ y_{n+1} = y_n + f(x_n, y_n) \cdot h \end{cases} \quad (2.15)$$

en los puntos de red considerados.

Lógicamente, la calidad de la aproximación así obtenida dependerá del problema concreto y del tamaño de paso utilizado (mejor cuanto más pequeño). De hecho, desde el punto de vista geométrico, el método de Euler aproxima a la función solución en el intervalo considerado, por medio de una línea poligonal y la aproximación obtenida será, en principio, tanto mejor cuanto mayor sea el número de pasos, esto es, cuanto más pequeño sea el tamaño de paso utilizado.

Lo anterior puede verse reflejado en la siguiente figura:

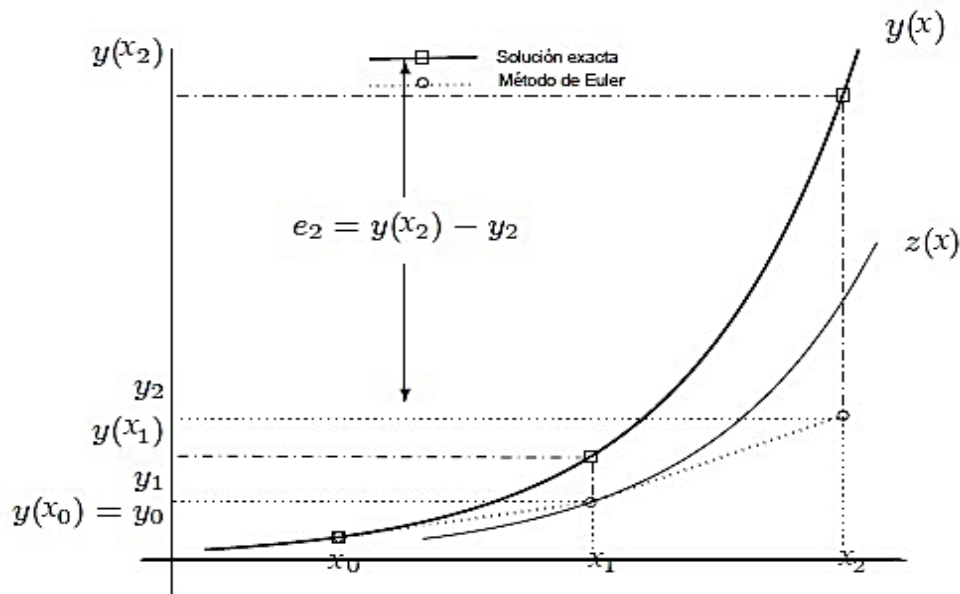


Figura 2.1. Interpretación geométrica del método de Euler

Por otro lado, en línea con lo anterior, en general el error será tanto mayor cuanto más grande sea el tamaño del paso  $h$  utilizado. De hecho, precisando más, podemos decir que el error local de truncación (por paso  $h$ ) del método de Euler es proporcional al cuadrado del paso  $h^2$  y el error global es proporcional al paso  $h$ , por lo que diremos que el método de Euler es de orden de convergencia uno.

En la práctica, el método de Euler (explícito) no se suele utilizar demasiado a menudo, pues se dispone de métodos mucho más eficientes y precisos. Además, el método no proporciona aproximaciones apropiadas a algunos problemas tales como los de tipo stiff (rígidos).

### 2.3.2. Métodos de Taylor

El Método de Euler que acabamos de describir no es más que un caso particular, de orden uno, de los métodos de Taylor, que consisten, de manera general, en buscar aproximaciones a la solución a partir de su polinomio de Taylor de un orden determinado.

Partiendo por tanto del P.V.I. (2.11), que presenta solución única  $y(x)$  en un entorno de  $x_0$  (solución que suponemos además derivable  $m$  veces en dicho entorno), aproximaremos dicha función por su polinomio de Taylor de orden  $m$ :

$$y(x) \cong y(x_0) + y'(x_0)(x - x_0) + \frac{1}{2} \cdot y''(x_0)(x - x_0)^2 + \frac{1}{3!} \cdot y'''(x_0)(x - x_0)^3 + \dots + \frac{y^{(m)}}{m!} \cdot (x - x_0)^m \quad (2.16)$$

y el error de aproximación viene determinado por el resto de orden  $m + 1$ , de manera que el error local de truncación es proporcional a  $(x - x_0)^{m+1}$ .

Si fijamos una sucesión de puntos equiespaciados:  $x_0, x_1, x_2, \dots$  con  $x_{n+1} = x_n + h$ , y denominamos  $y_0, y_1, \dots$  a los valores aproximados de  $y(x)$  en dichos puntos, tendremos por una parte que:

$$y(x_{n+1}) \cong y(x_n) + y'(x_n)h + \frac{1}{2} y''(x_n)h^2 + \dots + \frac{1}{m!} y^{(m)}(x_n)h^m \quad (2.17)$$

Para poder aplicar el método, necesitamos conocer las sucesivas derivadas de la solución (que a priori es desconocida) hasta orden  $m$ ; pero considerando la propia ecuación diferencial:  $y'(x) = f(x, y(x))$  es fácil deducir que se verifica:  $y^{(k)}(x) = f^{(k-1)}(x, y(x))$  para cada  $k$ .

Las derivadas de  $f$  se calculan utilizando la regla de la cadena y la propia ecuación diferencial que satisface la solución. Por ejemplo, para  $k = 2$ :

$$\begin{aligned} y''(x) = f'(x, y(x)) &= \frac{\partial f}{\partial x}(x, y(x)) + \frac{\partial f}{\partial y}(x, y(x)) \cdot y'(x) = \\ &= \frac{\partial f}{\partial x}(x, y(x)) + \frac{\partial f}{\partial y}(x, y(x)) \cdot f(x, y(x)) \end{aligned} \quad (2.18)$$

y así sucesivamente. Se tiene por tanto que:

$$y(x_{n+1}) \cong y(x_n) + f(x_n, y(x_n))h + \frac{1}{2} f'(x_n, y(x_n))h^2 + \dots + \frac{1}{m!} f^{(m-1)}(x_n, y(x_n))h^m \quad (2.19)$$

expresión a partir de la cual se define el método de Taylor de orden  $m$  por:

$$y_{n+1} = y_n + f(x_n, y_n)h + \frac{1}{2} f'(x_n, y_n)h^2 + \dots + \frac{1}{m!} f^{(m-1)}(x_n, y_n)h^m \quad (2.20)$$

partiendo del valor inicial  $y_0$  que obtenemos de la condición inicial.

El error local del método será en cada paso proporcional a  $h^{m+1}$ .

Es evidente, a la vista de la expresión que define el método de Taylor de orden  $m$ , que el caso  $m = 1$  proporciona el método de Euler explícito. También es claro que el método puede aplicarse con facilidad para paso variable.

La principal dificultad que presenta este método y que hace desaconsejable su uso en general, salvo en problemas concretos, es su alto coste computacional asociado al cálculo de las sucesivas derivadas involucradas.

### 2.3.3. Método trapezoidal o método de Heun

Otro método de resolución de problemas con ecuaciones diferenciales del tipo (2.11) es el método de Heun, también conocido como método de Euler modificado o mejorado, basado en integrar ambos lados de la ecuación:

$$\begin{aligned} y'(x) = f(x, y(x)) \rightarrow y(x)|_{x_n}^{x_{n+1}} &= y(x_{n+1}) - y(x_n) = \int_{x_n}^{x_{n+1}} f(x, y(x)) dx \\ \rightarrow y(x_{n+1}) &= y(x_n) + \int_{x_n}^{x_{n+1}} f(x, y(x)) dx \quad \text{con } y(x_0) = z_1 \end{aligned} \quad (2.21)$$

Si asumimos que el valor de la función  $f(x, y)$  es constante e igual a  $f(x_n, y(x_n))$  dentro de un paso de tiempo  $[x_n, x_{n+1}]$ , la ecuación anterior se convierte en la ecuación (2.15), equivalente al método de Euler.

Si en lugar de lo anterior recurrimos a la regla trapezoidal para llevar a cabo la integración, obtenemos:

$$y_{n+1} = y_n + \frac{h}{2} \cdot [f(x_n, y_n) + f(x_{n+1}, y_{n+1})] \quad (2.22)$$

Pero, el lado derecho de esta ecuación contiene  $y_{n+1}$ , el cual es desconocido en  $x_n$ . Para resolver este problema, sustituimos en el segundo miembro  $y_{n+1}$  por la siguiente aproximación:

$$y_{n+1} \cong y_n + h \cdot f(x_n, y_n) \quad (2.23)$$

obteniendo así la expresión final del método de Heun:

$$y_{n+1} = y_n + \frac{h}{2} \cdot [f(x_n, y_n) + f(x_{n+1}, y_n + h \cdot f(x_n, y_n))] \quad (2.24)$$

Este método de Heun es una especie de método predictor-corrector, puesto que predice el valor de  $y_{n+1}$  por la ecuación (2.23) en  $x_n$  y luego corrige el valor predicho mediante la ecuación (2.24) en  $x_{n+1}$ .

El error local de truncación del método de Heun es proporcional a  $h^3$ , luego es menor que el del método de Euler que como vimos era proporcional a  $h^2$ . De hecho, el método de Heun es de orden (de convergencia) dos.

### 2.3.4. Métodos de Runge-Kutta

Aunque el método de Heun es mejor que el de Euler, no es todavía lo suficientemente preciso para la mayoría de aplicaciones reales. Sin embargo, formalizando y generalizando las ideas utilizadas para mejorar el método de Euler, surgen los denominados métodos Runge-Kutta.

Para introducir dichos métodos, consideremos el intervalo  $[a, b]$ , y un tamaño de paso  $h$ . Buscamos calcular aproximaciones a la solución exacta en los puntos  $x_n = a + n \cdot h, n = 0, \dots, N$ , siendo  $N = \frac{b-a}{h}$ , que denotaremos por  $y_n, n = 0, \dots, N$ .

Dado un  $y_0$  (que obtenemos de la condición inicial), un método de Runge-Kutta de  $s$  etapas, está definido en el paso  $n + 1$  por:

$$\left\{ \begin{array}{l} k_1 = f(x_n + c_1 h, y_n + h \sum_{j=1}^s a_{1j} k_j \\ \vdots \\ k_i = f(x_n + c_i h, y_n + h \sum_{j=1}^s a_{ij} k_j \\ \vdots \\ k_s = f(x_n + c_s h, y_n + h \sum_{j=1}^s a_{sj} k_j \\ y_{n+1} = y_n + h \sum_{j=1}^s b_j k_j \end{array} \right. \quad (2.25)$$

donde con frecuencia se imponen además las llamadas condiciones de simplificación:  $c_i = \sum_{j=1}^s a_{ij}, i = 1, 2, \dots, s$ .

Este método se puede representar por su tablero de Butcher:

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array} \quad (2.26)$$

En base a lo anterior, el método puede ser de tres tipos:

- Explícito: si  $a_{ik} = 0$  para  $k \geq i, k = 1, 2, \dots, s$ .
- Semi-implícito: si  $a_{ik} = 0$  para  $k > i, k = 1, 2, \dots, s$ .
- Implícito: en cualquier otro caso.

La idea general de los Métodos de Runge-Kutta es sustituir el Problema de Valor Inicial de la ecuación (2.11), por la ecuación integral equivalente:

$$\int_{y_0}^{y(x)} dy = \int_{x_0}^x f(x, y(x)) dx \Rightarrow y = y_0 + \int_{x_0}^x f(x, y(x)) dx \quad (2.27)$$

para proceder a aproximar esta última integral mediante un método numérico adecuado (recordemos que  $y(x)$  es desconocida).

Si nuevamente planteamos el problema paso a paso tendremos:

$$y(x_{n+1}) = y(x_n) + \int_{x_n}^{x_{n+1}} f(x, y(x)) dx \quad (2.28)$$

#### A) Ejemplo de método Runge-Kutta explícito de segundo orden:

Una primera opción que podemos aplicar es integrar mediante el método de los trapecios, es decir tomando:

$$\int_{x_n}^{x_{n+1}} f(x, y(x)) dx \simeq \frac{1}{2} \cdot (f(x_n, y_n) + f(x_{n+1}, y_{n+1})) \quad (2.29)$$

Al ser desconocida  $y_{n+1}$  en la expresión anterior, lo aproximaremos por  $\bar{y}_{n+1}$ , donde  $\bar{y}_{n+1}$  es la estimación de  $y_{n+1}$  que resultaría aplicando el método de Euler. Tendremos así:

$$\int_{x_n}^{x_{n+1}} f(x, y(x)) dx \simeq \frac{1}{2} h \cdot (f(x_n, y_n) + f(x_{n+1}, \bar{y}_{n+1})) \quad (2.30)$$

donde:

$$\bar{y}_{n+1} = y_n + h \cdot f(x_n, y_n) \quad (2.31)$$

y llegaremos a la expresión del método:

$$y_{n+1} = y_n + \frac{h}{2} \cdot (f(x_n, y_n) + f(x_{n+1}, \bar{y}_{n+1})) \quad (2.32)$$

que no es otro que el método de Heun al que hemos hecho referencia anteriormente.

Podemos describir el método anterior con las expresiones siguientes:

$$\begin{cases} k_1 = f(x_n, y_n) \\ k_2 = f(x_n + h, y_n + h k_1) \\ y_{n+1} = y_n + \frac{1}{2} h (k_1 + k_2) \end{cases} \quad (2.33)$$

que ponen de manifiesto que el método de Heun es un método de tipo Runge-Kutta explícito.

Comparando para este método los desarrollos de Taylor de la solución exacta y de la proporcionada por la aproximación considerada, es posible demostrar que el error local es proporcional a  $h^3$  y, por tanto, el error global lo es a  $h^2$ , esto es, el método es de orden dos. De hecho, éste es uno de los muchos (hay infinitos) métodos Runge-Kutta explícitos de dos etapas y orden dos existentes.

Nótese que, aunque estamos describiendo el método en términos de un paso fijo  $h$ , no habría ningún problema en ir variando el tamaño de dicho paso, esto es, no es necesario considerar una red de puntos equiespaciados a la hora de buscar aproximaciones con el método.

#### B) Ejemplo de método Runge-Kutta explícito de tercer orden:

Se trata de la misma idea, pero integrando por el Método de Simpson:

$$\int_{x_n}^{x_{n+1}} f(x, y(x)) dx \approx \frac{h}{6} \left( f(x_n, y_n) + 4f\left(x_{n+\frac{1}{2}}, \bar{y}_{n+\frac{1}{2}}\right) + f(x_{n+1}, \bar{y}_{n+1}) \right) \quad (2.34)$$

donde  $\bar{y}_{n+1}$  e  $\bar{y}_{n+\frac{1}{2}}$  son estimaciones, ya que  $y_{n+1}$  e  $y_{n+\frac{1}{2}}$  no son conocidos.

La estimación de  $\bar{y}_{n+\frac{1}{2}}$  se hace por el método de Euler:

$$\bar{y}_{n+\frac{1}{2}} = y_n + \frac{h}{2} \cdot f(x_n, y_n) \quad (2.35)$$

mientras que para la estimación de  $\bar{y}_{n+1}$  se pueden considerar varias opciones.



Una posibilidad es recurrir al método de Euler de nuevo, como en la ecuación (2.24), y otra, por ejemplo, podría ser tomar:

$$\bar{y}_{n+1} = y_n + h \cdot f\left(x_{n+\frac{1}{2}}, \bar{y}_{n+\frac{1}{2}}\right) \quad (2.36)$$

que consiste en variar el Método de Euler tomando la aproximación a la pendiente de la recta tangente en el punto medio, en vez de la tangente en el punto considerado.

No obstante, lo más usual es tomar una combinación de las dos opciones, concretamente:

$$\bar{y}_{n+1} = y_n + h \cdot \left( 2 f\left(x_{n+\frac{1}{2}}, \bar{y}_{n+\frac{1}{2}}\right) - f(x_n, y_n) \right) \quad (2.37)$$

Podemos entonces describir el método de Runge-Kutta de tercer orden resultante así:

$$\begin{cases} k_1 = f(x_n, y_n) \\ k_2 = f\left(x_n + \frac{h}{2}, y_n + \frac{1}{2}h k_1\right) \\ k_3 = f\left(x_n + h, y_n + h(-k_1 + 2k_2)\right) \\ y_{n+1} = y_n + \frac{1}{6}h(k_1 + 4k_2 + k_3) \end{cases} \quad (2.38)$$

Cabe añadir que el error local de truncación en este método de tercer orden es proporcional a  $h^4$  y en consecuencia el error global lo es a  $h^3$ .

Conviene comentar en este punto que éste es solo uno de los infinitos métodos Runge-Kutta explícitos de tres etapas y orden tres existentes y que podría implementarse con paso variable sin dificultad.

### C) Ejemplo de método Runge-Kutta explícito de cuarto orden:

El método Runge-Kutta de cuarto orden, que habitualmente se denomina abreviadamente RK4 y que es uno de los infinitos métodos Runge-Kutta explícitos de cuatro etapas y orden cuatro, se puede deducir de manera similar a la expuesta en la sección anterior para el caso de tercer orden, eso sí, introduciendo previamente un nuevo paso intermedio en la aproximación de las derivadas involucradas.

La expresión más habitual de este método está determinada por las formulas siguientes:

$$\begin{cases} k_1 = f(x_n, y_n) \\ k_2 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2} k_1\right) \\ k_3 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2} k_2\right) \\ k_4 = f(x_n + h, y_n + h k_3) \\ y_{n+1} = y_n + \frac{1}{6} h (k_1 + 2k_2 + 2k_3 + k_4) \end{cases} \quad (2.39)$$

que, al igual que el método de tercer orden, puede considerarse basada en la regla de integración de Simpson:

$$\int_{x_n}^{x_{n+1}} f(x, y(x)) dx \approx \frac{h}{6} \left( f(x_n, y_n) + 4f\left(x_{n+\frac{1}{2}}, \bar{y}_{n+\frac{1}{2}}\right) + f(x_{n+1}, \bar{y}_{n+1}) \right) \quad (2.40)$$

Una vez más, se presentan varias opciones en la evaluación y es posible ajustar el método de modo que se garantice que el error local sea proporcional a  $h^5$ , es decir, garantizando exactitud hasta el cuarto orden del desarrollo de Taylor, lo cual lleva a un error global proporcional a  $h^4$ .

Lo anterior hace preferible en general este método frente a los otros descritos hasta ahora para resolver Ecuaciones Diferenciales. En cualquier caso, conviene mencionar que éste es solo uno de los infinitos métodos Runge-Kutta explícitos de cuatro etapas y orden cuatro disponibles y que, como ocurre con todos los métodos de este tipo, es fácil de implementar con paso variable.

Se muestra en la Figura 2.2 de la página siguiente, un ejemplo de comparación de los resultados obtenidos con los métodos de Euler, Heun y Runge-Kutta de cuarto orden. Se observa que el método RK4 es mejor que el método de Heun, mientras que el método de Euler es el peor en términos de precisión con el mismo tamaño de paso.

Sin embargo, en términos de coste computacional por paso, el orden se invierte, ya que el método de Euler, el método de Heun y el método RK4 necesitan 1, 2 y 4 evaluaciones de función por iteración, respectivamente. Obsérvese que una llamada de función requiere en general de mucho más tiempo de computación que una multiplicación y que por lo tanto el número de llamadas de función debe ser un criterio a tener en cuenta a la hora de estimar y comparar el tiempo computacional requerido.

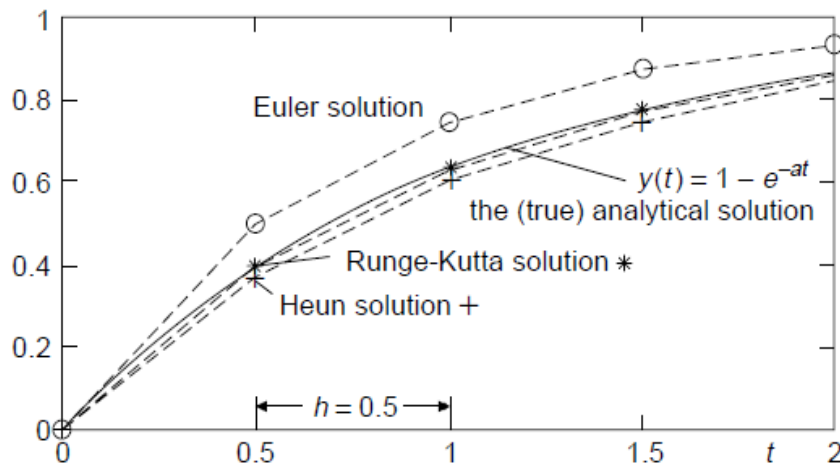


Figura 2.2. Ejemplo comparativo de la precisión de los métodos Euler, Heun y RK4

Matlab incorpora multitud de implementaciones de métodos Runge-Kutta y de otros tipos para la resolución aproximada de E.D.O.s. Por ejemplo, las funciones "ode23" y "ode45" implementan pares encajados (de órdenes 2-3 y 4-5 respectivamente) de métodos de tipo Runge-Kutta explícitos, con un ajuste de tamaño de paso adaptable, que utiliza un tamaño de paso grande o pequeño dependiendo de cómo se comporte la función  $f(x, y(x))$  que define la ecuación diferencial.

### 2.3.5. Métodos Adams-Bashfort-Moulton

Se trata de una familia de métodos predictor-corrector, que combinan un método explícito lineal multipaso (predictor), en el que la ecuación en diferencias sólo incluye combinaciones lineales de términos de los tipos:  $y_{n+j} = y(x_{n+j})$ ,  $f_{n+j} = f(x_{n+j}, y_{n+j})$ ,  $j = 0, 1, \dots, k$ , con un método implícito (corrector), dando lugar a un método explícito.

A modo de ejemplo, podemos considerar el método Adams-Bashforth-Moulton, que suele abreviarse como ABM, que consta de dos pasos.

El primero consiste en aproximar  $f(x, y(x))$  por el polinomio interpolador de Lagrange  $l_3(x)$  de grado menor o igual que tres que interpola los cuatro puntos:  $\{(x_{n-3}, f_{n-3}), (x_{n-2}, f_{n-2}), (x_{n-1}, f_{n-1}), (x_n, f_n)\}$  y sustituir dicho polinomio en la forma integral de la ecuación (2.21) de la ecuación diferencial para obtener una estimación de  $y_{n+1}$ :

$$p_{n+1} = y_n + \int_0^h l_3(x) dx = y_n + \frac{h}{24} (-9f_{n-3} + 37f_{n-2} - 59f_{n-1} + 55f_n) \quad (2.41)$$

El segundo paso es similar al anterior y consiste en aproximar  $f(x, y(x))$  por el polinomio interpolador de Lagrange  $l'_3(x)$  de grado menor o igual que tres que interpola los puntos:  $\{(x_{n-2}, f_{n-2}), (x_{n-1}, f_{n-1}), (x_n, f_n), (x_{n+1}, f_{n+1})\}$  para obtener una estimación corregida de  $y_{n+1}$ :

$$c_{n+1} = y_n + \int_0^h l'_3(x) dx = y_n + \frac{h}{24} (f_{n-2} - 5f_{n-1} + 19f_n + 9f_{n+1}) \quad (2.42)$$

Los coeficientes de las ecuaciones (2.41) y (2.42) se pueden obtener, por ejemplo, utilizando las rutinas MATLAB "*lagranp*" y "*polyint*", cada una de las cuales genera polinomios de Lagrange (sus coeficientes) e integra un polinomio, respectivamente.

Alternativamente, si escribimos los desarrollos en serie de Taylor de  $y_{n+1}$  centrado en  $x_n$  y de  $y_n$  centrado en  $x_{n+1}$  como:

$$\begin{aligned} y_{n+1} &= y_n + hf_n + \frac{h^2}{2} f'_n + \frac{h^3}{3!} f_n^{(2)} + \frac{h^4}{4!} f_n^{(3)} + \frac{h^5}{5!} f_n^{(4)} + \dots \Rightarrow \\ y_n &= y_{n+1} - hf_{n+1} + \frac{h^2}{2} f'_{n+1} - \frac{h^3}{3!} f_{n+1}^{(2)} + \frac{h^4}{4!} f_{n+1}^{(3)} - \frac{h^5}{5!} f_{n+1}^{(4)} + \dots \Rightarrow \\ y_{n+1} &= y_n + hf_{n+1} - \frac{h^2}{2} f'_{n+1} + \frac{h^3}{3!} f_{n+1}^{(2)} - \frac{h^4}{4!} f_{n+1}^{(3)} + \frac{h^5}{5!} f_{n+1}^{(4)} - \dots \end{aligned} \quad (2.43)$$

y reemplazamos las derivadas primera, segunda y tercera por sus diferentes aproximaciones mediante fórmulas de derivación numérica adecuadas, tenemos:

$$\begin{aligned} y_{n+1} &= y_n + hf_n + \frac{h^2}{2} \left( \frac{-\frac{1}{3}f_{n-3} + \frac{3}{2}f_{n-2} - 3f_{n-1} + \frac{11}{6}f_n}{h} + \frac{1}{4}h^3 f_n^{(4)} + \dots \right) \\ &\quad + \frac{h^3}{3!} \left( \frac{-f_{n-3} + 4f_{n-2} - 5f_{n-1} + 2f_n}{h^2} + \frac{11}{12}h^2 f_n^{(4)} + \dots \right) + \\ &\quad + \frac{h^4}{4!} \left( \frac{-f_{n-3} + 3f_{n-2} - 3f_{n-1} + f_n}{h^3} + \frac{3}{2}hf_n^{(4)} + \dots \right) + \frac{h^5}{120} f_n^{(4)} + \dots \Rightarrow \\ y_{n+1} &= y_n + \frac{h}{24} (-9f_{n-3} + 37f_{n-2} - 59f_{n-1} + 55f_n) + \frac{251}{720} h^5 f_n^{(4)} + \dots \Rightarrow \\ &\stackrel{(2.41)}{\implies} y_{n+1} \approx p_{n+1} + \frac{251}{720} h^5 f_n^{(4)} \end{aligned} \quad (2.44)$$

$$\begin{aligned}
 y_{n+1} &= y_n + hf_{n+1} - \\
 &\quad - \frac{h^2}{2} \left( \frac{-\frac{1}{3}f_{n-2} + \frac{3}{2}f_{n-1} - 3f_n + \frac{11}{6}f_{n+1}}{h} + \frac{1}{4}h^3f_{n+1}^{(4)} + \dots \right) + \\
 &\quad + \frac{h^3}{3!} \left( \frac{-f_{n-2} + 4f_{n-1} - 5f_n + 2f_{n+1}}{h^2} + \frac{11}{12}h^2f_{n+1}^{(4)} + \dots \right) - \\
 &\quad - \frac{h^4}{4!} \left( \frac{-f_{n-2} + 3f_{n-1} - 3f_n + f_{n+1}}{h^3} + \frac{3}{2}hf_{n+1}^{(4)} + \dots \right) + \frac{h^5}{120}f_{n+1}^{(4)} + \dots \Rightarrow \\
 y_{n+1} &= y_n + \frac{h}{24}(f_{n-2} - 5f_{n-1} + 19f_n + 9f_{n+1}) - \frac{19}{720}h^5f_{n+1}^{(4)} + \dots \Rightarrow \\
 &\stackrel{(2.42)}{\implies} y_{n+1} \approx c_{n+1} - \frac{19}{720}h^5f_{n+1}^{(4)} \tag{2.45}
 \end{aligned}$$

A partir de estas ecuaciones y bajo el supuesto de que  $f_{n+1}^{(4)} \cong f_n^{(4)} \cong K$  podemos escribir los errores del método predictor y el corrector:

$$E_{P,n+1} = y_{n+1} - p_{n+1} \approx \frac{251}{720}h^5f_n^{(4)} \cong \frac{251}{720}Kh^5 \tag{2.46}$$

$$E_{C,n+1} = y_{n+1} - c_{n+1} \approx \frac{-19}{720}h^5f_{n+1}^{(4)} \cong \frac{-19}{720}Kh^5 \tag{2.47}$$

Todavía no podemos usar estas fórmulas para estimar los errores de predicción y corrección, ya que  $K$  es desconocido. Pero, a partir de la diferencia entre estas dos fórmulas:

$$E_{P,n+1} - E_{C,n+1} = c_{n+1} - p_{n+1} \cong \frac{270}{720}Kh^5 \equiv \frac{270}{251}E_{P,n+1} \equiv \frac{-270}{19}E_{C,n+1} \tag{2.48}$$

podemos obtener las fórmulas prácticas para estimar los errores como:

$$E_{P,n+1} = y_{n+1} - p_{n+1} \cong \frac{251}{270} \cdot (c_{n+1} - p_{n+1}) \tag{2.49}$$

$$E_{C,n+1} = y_{n+1} - c_{n+1} \approx \frac{-19}{270} \cdot (c_{n+1} - p_{n+1}) \tag{2.50}$$

Estas fórmulas nos dan estimaciones aproximadas de cuán cerca están los valores predichos y corregidos al valor verdadero y por lo tanto se pueden usar para mejorarlos, así como para ajustar el tamaño del paso:

$$p_{n+1} \rightarrow p_{n+1} + \frac{251}{270}(c_n - p_n) \Rightarrow m_{n+1} \quad (2.51)$$

$$c_{n+1} \rightarrow c_{n+1} - \frac{19}{270}(c_{n+1} - p_{n+1}) \Rightarrow y_{n+1} \quad (2.52)$$

El método de Adams-Bashforth-Moulton (ABM) con las fórmulas de modificación anteriores queda finalmente resumido a continuación:

$$\left\{ \begin{array}{l} \text{Predictor (2.38): } p_{n+1} = y_n + \frac{h}{24}(-9 f_{n-3} + 37 f_{n-2} - 59 f_{n-1} + 55 f_n) \\ \text{Modificador (2.48): } m_{n+1} = p_{n+1} + \frac{251}{270}(c_n - p_n) \\ \text{Corrector (2.39): } c_{n+1} = y_n + \frac{h}{24}(f_{n-2} - 5 f_{n-1} + 19 f_{n-1} + 9 f(x_{n+1}, m_{n+1})) \\ y_{n+1} = c_{n+1} - \frac{19}{270}(c_{n+1} - p_{n+1}) \end{array} \right. \quad (2.53)$$

Este esquema necesita sólo dos evaluaciones (nuevas) de función por iteración, mientras que tiene un error de truncación proporcional a  $h^5$  (por lo que es de orden cuatro) y por lo tanto se espera que funcione mejor que los métodos discutidos hasta ahora. Este método, se implementa mediante la rutina incorporada en Matlab "ode113".

## 2.4. Problema con Valores de Frontera [3] [10] [12] [16] [20] [

Los Problemas de Valores Iniciales han sido ampliamente estudiados a lo largo del avance del cálculo numérico, como hemos podido observar en el apartado anterior. Debido a esto, en este trabajo dedicaremos especial atención a los casos en los que a la Ecuación Diferencial Ordinaria de Segundo Orden tipo, ecuación (2.1), viene acompañada de otro tipo de condiciones, concretamente las denominadas condiciones de contorno, que se caracterizan porque los valores  $y(x_0)$ ,  $y(x_f)$  vienen dados, en los puntos inicial y final del intervalo al que pertenece la variable independiente, es decir:  $x_0 = a$ ,  $x_f = b$ , respectivamente. Concretamente:

$$y(a) = y(x_0) = z_1, \quad y(b) = y(x_n) = z_2 \quad (2.54)$$

La aplicación de estas dos condiciones, (2.54), con el fin de encontrar una solución numérica para la ecuación, da lugar a lo que se denomina Problema con Valor en la Frontera, P.V.F, o más conocido por su nombre inglés: Boundary Value Problems, B.V.P, que queda expresado como:

$$\begin{cases} y'' = f(x, y, y') \\ y(a) = z_1 \\ y(b) = z_2 \end{cases} \quad (2.55)$$

Es importante resaltar que si el P.V.F. anterior se expresa mediante una ecuación del tipo la (2.1) que habíamos considerado al inicio de este capítulo:  $y'' = p(x) \cdot y' + q(x) \cdot y + r(x)$ ,  $x \in [a, b]$ , entonces se dice que el P.V.F. es de tipo lineal. En caso contrario, diremos que el P.V.F. es del tipo no lineal.

Algunos de los principales métodos de resolución empleados para este tipo de problemas, se resumen a continuación:

A) MÉTODO DEL DISPARO: Hay que distinguir entre:

- Método del Disparo Lineal: para ecuaciones lineales, se basa en la sustitución del Problema lineal con Valor de Frontera (P.V.F.) por dos Problemas de Valores Iniciales (P.V.I.), que se pueden resolver posteriormente, por ejemplo, empleando alguno de los métodos de resolución explicados en el apartado anterior.

- Método del Disparo para Problemas No Lineales: la solución a este problema no puede expresarse en general como una combinación lineal finita de soluciones de problemas de valores iniciales. Se basa en la construcción de una sucesión de soluciones a Problemas de Valores Iniciales que converja a la solución del P.V.F.

B) MÉTODO DE DIFERENCIAS FINITAS: De nuevo, diferenciamos entre:

- Método de Diferencias Finitas para Problemas Lineales: se basa en reemplazar las derivadas que intervienen en la ecuación diferencial que define el problema, mediante fórmulas de derivación numérica (fórmulas en diferencias) adecuadas. Lo anterior transforma la ecuación diferencial en una ecuación en diferencias y la aproximación a la solución en los puntos considerados se obtiene resolviendo el sistema lineal de ecuaciones (algebraicas) resultante.

- Método de Diferencias Finitas para Problemas No Lineales: se parece al método lineal; pero en este caso el sistema de ecuaciones (algebraicas) que se obtiene a partir de la ecuación en diferencias no será lineal y, por lo tanto, se requiere de algún método (iterativo, de la secante, de Newton, etc) que permita resolverlo. Obviamente el coste computacional asociado es en general muy superior al del caso lineal.

### C) MÉTODO DE RAYLEIGH-RITZ:

En este método el problema se aborda de una forma distinta, ya que se reformula el Problema con Valores en la Frontera como un problema que consiste en seleccionar, del conjunto de todas las funciones suficientemente derivables que satisfacen las condiciones de frontera, aquélla que reduzca al mínimo una determinada integral.

Algunos de estos métodos vamos a explicarlos más detenidamente en capítulos posteriores. Por ello, no es objetivo de este capítulo incorporar una descripción teórica detallada de los mismos, sino simplemente mencionarlos. Si se busca una descripción más detallada recomendamos acudir a la referencia [3] en su Capítulo 11.





# MÉTODOS DE RESOLUCIÓN BASADOS EN SPLINES

## Capítulo 3



## 3. MÉTODOS DE RESOLUCIÓN BASADOS EN SPLINES

### 3.1. Introducción a la utilización de Splines [18] a [30]

La teoría de funciones Spline es relativamente reciente. Fue desarrollada en la década de 1940 por el matemático estadounidense de origen rumano Isaac Jacob Schoenberg (1903-1990). Inicialmente el término Spline hacía referencia a una clase de funciones definidas a trozos por polinomios y que verificaban ciertas condiciones de regularidad en los puntos de contacto de los polinomios que definían el Spline; pero hoy en día se recogen bajo dicha denominación generalizaciones de la idea anterior a funciones definidas a trozos de muy diversos tipos.

Una de las primeras y principales aplicaciones de los Splines la encontramos en la interpolación segmentaria de funciones a partir de una tabla de datos de la misma, esto es, a partir de valores de la función en ciertos puntos que se suelen denominar nodos de interpolación.

Los Splines clásicos basados en funciones polinomiales a trozos son fáciles y baratos, computacionalmente hablando, de evaluar y las condiciones de regularidad que verifican permiten obtener interpolaciones de funciones mediante curvas suaves. De hecho, es habitual que el grado de los polinomios utilizados al construir los Splines sea bajo, lo que evita problemas habituales en la interpolación polinomial no segmentaria tales como la aparición de oscilaciones.

A pesar de las siete décadas que han transcurrido desde su introducción y del rápido desarrollo de la teoría de Splines, debido sobre todo a las múltiples e importantes aplicaciones que tienen en campos muy diferentes, hoy en día se sigue investigando activamente en este campo y se siguen encontrando nuevas aplicaciones y generalizaciones de interés.

Actualmente, la teoría de Splines está teniendo una influencia esencial en grandes áreas del análisis numérico moderno, tales como:

- Interpolación, aproximación y ajuste de datos.
- Derivación e integración numérica.
- Tratamiento numérico de ecuaciones integrales, diferenciales y en derivadas parciales.
- Aproximación óptima.

- Cálculo de autovalores y autofunciones de operadores.
- Teorías de control.
- Geometría computacional y diseño geométrico asistido por ordenador.
- Métodos numéricos de la teoría de probabilidades y estadística.
- Ondas y fractales.

En este trabajo nos centraremos en su aplicación a ciertas Ecuaciones Diferenciales Ordinarias de Segundo Orden. En concreto, desarrollaremos varias estrategias basadas en Splines y orientadas a resolver problemas que vienen formulados en términos de E.D.O.s de Segundo Orden. Problemas de este tipo surgen con frecuencia en el ámbito de la ingeniería, aunque no se circunscriben únicamente a este campo.

Consideraremos tanto Problemas de Valores Iniciales (P.V.I) como Problemas con Valores en la Frontera (P.V.F). Aunque comenzaremos estudiando problemas lineales a coeficientes no necesariamente constantes, también trataremos el caso de problemas no lineales.

Recogemos, sin ánimo de ser exhaustivos, algunas aportaciones recientes en este campo de autores en revistas internacionales y congresos científicos en el apartado de la bibliografía correspondiente a artículos científicos, concretamente de las referencias [18] a [30].

## 3.2. Resolución de Problemas Lineales de Valores Iniciales

[18] a [30]

### 3.2.1. Planteamiento del Método de Colocación basado en Splines Cúbicos

Consideramos la ecuación diferencial de segundo orden tipo, planteada ya en el capítulo anterior, ecuación (2.1):

$$y''(x) = p(x) \cdot y'(x) + q(x) \cdot y(x) + r(x) \quad x \in [a, b] \quad (3.1)$$

acompañada de un par de condiciones iniciales y/o de contorno que permitan garantizar que el problema tenga una única solución  $y$  en el intervalo  $[a, b]$ .

Pretendemos obtener una aproximación a la única solución de la ecuación anterior, (3.1), de la forma  $y(x) \cong S(x)$ , donde  $S$  es un Spline cúbico basado en los nodos  $\{x_i\}_{i=1}^{n+1}$  :

$$a = x_1 < x_2 < \dots < x_n < x_{n+1} = b \quad (3.2)$$

Tenemos así que:

$$S(x) = \begin{cases} S_1(x) = a_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3 & x \in [x_1, x_2] \\ S_2(x) = a_2 + b_2(x - x_2) + c_2(x - x_2)^2 + d_2(x - x_2)^3 & x \in [x_2, x_3] \\ \dots & \dots \\ S_k(x) = a_k + b_k(x - x_k) + c_k(x - x_k)^2 + d_k(x - x_k)^3 & x \in [x_k, x_{k+1}] \\ \dots & \dots \\ S_n(x) = a_n + b_n(x - x_n) + c_n(x - x_n)^2 + d_n(x - x_n)^3 & x \in [x_n, x_{n+1}] \end{cases} \quad (3.3)$$

esto es,  $S$  viene dado en cada intervalo  $[x_k, x_{k+1}]$ , por el polinomio  $S_k$  de grado  $\leq 3$  siguiente:

$$S_k(x) = a_k + b_k \cdot (x - x_k) + c_k \cdot (x - x_k)^2 + d_k \cdot (x - x_k)^3 \quad (3.4)$$

para cada  $k = 1, 2, \dots, n$ .

Disponemos por tanto de  $4n$  parámetros que son las incógnitas a determinar:  $a_k, b_k, c_k, d_k$ ,  $k = 1, 2, \dots, n$ .

Para que  $S$  sea un spline cúbico en  $[a, b]$ , ha de satisfacer además que  $S \in C^2[a, b]$ , lo que implica que:  $S, S'$  y  $S''$  han de ser continuas en  $[a, b]$ . Esto impone  $3 \cdot (n - 1)$  condiciones que se traducen en ecuaciones asociadas a los  $n - 1$  nodos intermedios:  $x_2, x_3, \dots, x_n$ .

Concretamente, ha de verificar:

$$\left. \begin{aligned} S_k(x_{k+1}) &= S_{k+1}(x_{k+1}) \\ S'_k(x_{k+1}) &= S'_{k+1}(x_{k+1}) \\ S''_k(x_{k+1}) &= S''_{k+1}(x_{k+1}) \end{aligned} \right\} \forall k = 1, 2, \dots, n - 1 \quad (3.5)$$

lo que se traduce en:

$$\left. \begin{aligned} a_{k+1} &= a_k + h_k \cdot b_k + h_k^2 \cdot c_k + h_k^3 \cdot d_k \\ b_{k+1} &= b_k + 2 \cdot h_k \cdot c_k + 3 \cdot h_k^2 \cdot d_k \\ c_{k+1} &= c_k + 3 \cdot h_k \cdot d_k \end{aligned} \right\} \forall k = 1, 2, \dots, n - 1 \quad (3.6)$$

donde  $h_k = x_{k+1} - x_k$ ,  $k = 1, 2, \dots, n$ .

Si definimos:

$$\left. \begin{aligned} a_{n+1} &= S(x_{n+1}) = S_n(x_{n+1}) \\ b_{n+1} &= S'(x_{n+1}) = S'_n(x_{n+1}) \\ c_{n+1} &= \frac{1}{2} S''(x_{n+1}) = \frac{1}{2} S''_n(x_{n+1}) \end{aligned} \right\} \quad (3.7)$$

las relaciones (3.6) también se satisfacen para  $k = n$ .

Imponemos además  $n + 1$  condiciones de “colocación”, consistentes en exigir que el spline  $S$  satisfaga la ecuación diferencial (3.1) en los  $n + 1$  nodos considerados:  $x_1, x_1, \dots, x_{n+1}$ . Obviamente, en los nodos laterales  $x_1$  y  $x_{n+1}$  las derivadas involucradas serán derivadas laterales por la derecha o por la izquierda respectivamente.

Queremos pues que se satisfaga:

$$S''(x_k) = p(x_k) \cdot S'(x_k) + q(x_k) \cdot S(x_k) + r(x_k) \quad (3.8)$$

para cada  $x_k$  con  $k = 1, \dots, n + 1$ .

O lo que es lo mismo:

$$S''(x_k) = p_k \cdot S'(x_k) + q_k \cdot S(x_k) + r_k \quad (3.8')$$

si utilizamos las abreviaturas:  $p_k = p(x_k)$ ,  $q_k = q(x_k)$ ,  $r_k = r(x_k)$ .

Tenemos así, en primer lugar, la condición:

$$S''_k(x_k) = p_k \cdot S'_k(x_k) + q_k \cdot S_k(x_k) + r_k \quad (3.9)$$

para cada  $k = 1, 2, \dots, n$ , que se traduce en:

$$2 \cdot c_k = p_k \cdot b_k + q_k \cdot a_k + r_k \quad (3.10)$$

Y adicionalmente:

$$S''_n(x_{n+1}) = p_{n+1} \cdot S'_n(x_{n+1}) + q_{n+1} \cdot S_n(x_{n+1}) + r_{n+1} \quad (3.11)$$

que da lugar a la igualdad:

$$\begin{aligned} 2 \cdot c_n + 6 \cdot h_n \cdot d_n &= p_{n+1} \cdot (b_n + 2 \cdot h_n \cdot c_n + 3 \cdot h_n^2 \cdot d_n) + \\ &+ q_{n+1} \cdot (a_n + h_n \cdot b_n + h_n^2 \cdot c_n + h_n^3 \cdot d_n) + r_{n+1} \quad (= 2 \cdot c_{n+1}) \end{aligned} \quad (3.12)$$

o mejor, expresada recurriendo a las definiciones hechas en el sistema de ecuaciones (3.7):

$$2 \cdot c_{n+1} = p_{n+1} \cdot b_{n+1} + q_{n+1} \cdot a_{n+1} + r_{n+1} \quad (3.12')$$

Resumiendo, ya tenemos las condiciones:

$$\left. \begin{aligned} a_{k+1} &= a_k + h_k \cdot b_k + h_k^2 \cdot c_k + h_k^3 \cdot d_k \\ b_{k+1} &= b_k + 2 \cdot h_k \cdot c_k + 3 \cdot h_k^2 \cdot d_k \\ c_{k+1} &= c_k + 3 \cdot h_k \cdot d_k \end{aligned} \right\} k = 1, 2, \dots, n \quad (3.6)$$

$$2 \cdot c_k = p_k \cdot b_k + q_k \cdot a_k + r_k, \text{ para } k = 1, 2, \dots, n + 1 \quad (3.10)$$

Aunque tres condiciones son redundantes por satisfacerse en virtud de la definición hecha en (3.7). Solo faltaría imponer las dos condiciones iniciales y/o de contorno que acompañan a la E.D.O. de 2º orden, para que el problema tenga solución única y resolver el sistema lineal de  $4n$  ecuaciones con  $4n$  incógnitas para obtener la aproximación a la solución buscada.

### 3.2.2. Obtención del método de colocación, basado en Splines Cúbicos, para aproximar la solución de P.V.I. lineales

Consideramos un problema (lineal) de valores iniciales tipo:

$$\left\{ \begin{aligned} y''(x) &= p(x) \cdot y'(x) + q(x) \cdot y(x) + r(x) \\ y(a) &= z_1 \\ y'(a) &= z_2 \end{aligned} \right. \quad (3.13)$$

que admita solución única.

Con objeto de encontrar una aproximación a su solución, consideramos el spline cúbico descrito en el punto 3.2.1. En cada intervalo  $I_k = [x_k, x_{k+1}]$  tenemos:

$$\left\{ \begin{aligned} S_k(x) &= a_k + b_k \cdot (x - x_k) + c_k \cdot (x - x_k)^2 + d_k \cdot (x - x_k)^3 \\ S'_k(x) &= b_k + 2 \cdot c_k \cdot (x - x_k) + 3 \cdot d_k \cdot (x - x_k)^2 \\ S''_k(x) &= 2 \cdot c_k + 6 \cdot d_k \cdot (x - x_k) \end{aligned} \right. \quad (3.14)$$

Así, en el intervalo inicial:

$$\begin{cases} S_1(x) = a_1 + b_1 \cdot (x - a) + c_1 \cdot (x - a)^2 + d_1 \cdot (x - a)^3 \\ S_1'(x) = b_1 + 2 \cdot c_1 \cdot (x - a) + 3 \cdot d_1 \cdot (x - a)^2 \\ S_1''(x) = 2 \cdot c_1 + 6 \cdot d_1 \cdot (x - a) \end{cases} \quad (3.15)$$

Utilizando las condiciones iniciales, es fácil comprobar que:

$$\begin{cases} S_1(a) = a_1 = z_1 \\ S_1'(a) = b_1 = z_2 \\ S_1''(a) = 2 \cdot c_1 \end{cases} \quad (3.16)$$

Recordemos que nuestro objetivo es aproximar  $Y(x) \cong S(x)$  imponiendo que  $S$  satisfaga la ecuación diferencial en los nodos, esto es, las llamadas condiciones de colocación.

La condición de colocación en  $x_1 = a$  viene a imponer:

$$S_1''(a) = p(a) \cdot S_1'(a) + q(a) \cdot S_1(a) + r(a) \quad (3.17)$$

La ecuación anterior, (3.17) junto con los resultados obtenidos en (3.16), dan lugar a la ecuación:

$$2 \cdot c_1 = p_1 \cdot z_2 + q_1 \cdot z_1 + r_1 \quad (3.18)$$

que permite obtener el coeficiente  $c_1$ :

$$c_1 = \frac{1}{2} \cdot (p_1 \cdot z_2 + q_1 \cdot z_1 + r_1) \quad (3.19)$$

Quedan así determinados los coeficientes buscados  $a_1, b_1, c_1$  en el primer intervalo  $[x_1, x_2] = [a, x_2]$ .

Análogamente, podemos obtener el valor del spline y de sus dos primeras derivadas en el punto final del intervalo, esto es en  $x_2 = a + h$ , sin más que sustituir dicho punto en las expresiones (3.15):

$$\begin{cases} S_1(a + h) = a_1 + b_1 \cdot h + c_1 \cdot h^2 + d_1 \cdot h^3 \\ S_1'(a + h) = b_1 + 2 \cdot c_1 \cdot h + 3 \cdot d_1 \cdot h^2 \\ S_1''(a + h) = 2 \cdot c_1 + 6 \cdot d_1 \cdot h \end{cases} \quad (3.20)$$



La condición de colocación en el punto  $x_2$  toma así la forma:

$$S_1''(a + h_1) = p_2 \cdot S_1'(a + h_1) + q_2 \cdot S_1(a + h_1) + r_2 \quad (3.21)$$

que, utilizando las relaciones anteriores, se traduce en imponer:

$$\begin{aligned} 2 \cdot c_1 + 6 \cdot d_1 \cdot h_1 &= p_2 \cdot (b_1 + 2 \cdot c_1 \cdot h_1 + 3 \cdot d_1 \cdot h_1^2) + \\ &+ q_2 \cdot (a_1 + b_1 \cdot h_1 + c_1 \cdot h_1^2 + d_1 \cdot h_1^3) + r_2 \end{aligned} \quad (3.22)$$

De esta ecuación, (3.22), podemos obtener el valor de  $d_1$ , que era la única incógnita que nos faltaba por determinar en el primer intervalo, sin más que despejar:

$$\begin{aligned} h_1(6 - 3 p_2 h_1 - q_2 h_1^2) d_1 &= \\ &= p_2 (b_1 + 2c_1 h_1) + q_2(a_1 + b_1 h_1 + c_1 h_1^2) + r_2 - 2c_1 \Rightarrow \\ \Rightarrow h_1(6 - 3 p_2 h_1 - q_2 h_1^2) d_1 &= \\ &= q_2 a_1 + (p_2 + q_2 h_1) b_1 + (2 p_2 h_1 + q_2 h_1^2 - 2) c_1 + r_2 \Rightarrow \\ \Rightarrow d_1 &= \frac{q_2 a_1 + (p_2 + q_2 h_1) b_1 + (2 p_2 h_1 + q_2 h_1^2 - 2) c_1 + r_2}{h_1 (6 - 3 p_2 h_1 - q_2 h_1^2)} \end{aligned} \quad (3.23)$$

Finalmente, sustituyendo los valores de los coeficientes  $a_1, b_1, c_1$  obtenidos en las ecuaciones (3.16) y (3.19), a partir de las condiciones iniciales y la condición de colocación del punto  $x_1 = a$ , obtenemos  $d_1$ :

$$d_1 = \frac{q_2 z_1 + (p_2 + q_2 h_1) z_2 + (2 p_2 h_1 + q_2 h_1^2 - 2) \frac{1}{2} (p_1 z_2 + q_1 z_1 + r_1) + r_2}{h_1 (6 - 3 p_2 h_1 - q_2 h_1^2)} \quad (3.24)$$

y queda completamente determinado  $S_1$  en  $I_1 = [x_1, x_2]$ .

Ahora describiremos inductivamente cómo determinar los restantes  $S_k$  en los intervalos  $I_k = [x_k, x_{k+1}]$  para  $k = 2, 3, \dots, n$ .

Para ello, suponemos conocidos los coeficientes:  $a_{k-1}, b_{k-1}, c_{k-1}, d_{k-1}$  que definen  $S_{k-1}$  en  $I_{k-1} = [x_{k-1}, x_k]$  y determinamos a partir de ellos los coeficientes:  $a_k, b_k, c_k$  y  $d_k$ .

A partir de las condiciones de continuidad en el nodo  $x_k$  sabemos que se verifica:

$$\left. \begin{aligned} a_k &= a_{k-1} + h_{k-1} b_{k-1} + h_{k-1}^2 c_{k-1} + h_{k-1}^3 d_{k-1} \\ b_k &= b_{k-1} + 2 h_{k-1} c_{k-1} + 3 h_{k-1}^2 d_{k-1} \\ c_k &= c_{k-1} + 3 h_{k-1} d_{k-1} \end{aligned} \right\} \quad (3.25)$$

que permiten determinar  $a_k$ ,  $b_k$  y  $c_k$ .

Para determinar  $d_k$ , imponemos la condición de colocación en el nodo  $x_{k+1}$ , esto es:

$$S_k''(x_{k+1}) = p_{k+1} S_k'(x_{k+1}) + q_{k+1} S_k(x_{k+1}) + r_{k+1} \quad (3.26)$$

que se traduce en la ecuación:

$$\begin{aligned} 2 c_k + 6 h_k d_k &= p_{k+1}(b_k + 2 h_k c_k + 3 h_k^2 d_k) + \\ &+ q_{k+1}(a_k + h_k b_k + h_k^2 c_k + h_k^3 d_k) + r_{k+1} \end{aligned} \quad (3.27)$$

y que permite despejar  $d_k$  en términos de los coeficientes ya conocidos  $a_k$ ,  $b_k$  y  $c_k$ :

$$d_k = \frac{q_{k+1} a_k + (p_{k+1} + q_{k+1} h_k) b_k + (2 p_{k+1} h_k + q_{k+1} h_k^2 - 2) c_k + r_{k+1}}{h_k (6 - 3 p_{k+1} h_k - q_{k+1} h_k^2)} \quad (3.28)$$

Con lo que tenemos completamente determinado  $S_k$  en  $I_k$ . Podemos así construir  $S$  en  $[a, b]$  a partir de  $S_1$  en  $I_1$  utilizando el proceso inductivo descrito para  $k = 2, 3, \dots, n$ .

### 3.2.3. Método de resolución "inverso" basado en Splines Cúbicos

Planteamos ahora una variante del método explicado en los apartados anteriores, pero basada en los mismos principios.

En esta nueva variante en lugar de obtener los coeficientes  $a_k$ ,  $b_k$ ,  $c_k$  y  $d_k$  de  $S_k$  en  $I_k$  a partir de los de  $S_{k-1}$  en  $I_{k-1}$ , lo hacemos a la inversa, esto es, a partir de los coeficientes  $a_{k+1}$ ,  $b_{k+1}$ ,  $c_{k+1}$  y  $d_{k+1}$  de  $S_{k+1}$  en  $I_{k+1}$ , tratamos de obtener  $a_k$ ,  $b_k$ ,  $c_k$  y  $d_k$ .

Lo anterior será de utilidad para construir soluciones a problemas de ecuaciones diferenciales cuyas condiciones sean "finales", es decir, vengan dadas en la forma  $y(b) = z_1$ ,  $y'(b) = z_2$ , en vez de en la forma descrita en el problema de valores iniciales (esto es:  $y(a) = z_1$ ,  $y'(a) = z_2$ ).

Como vimos en (3.6), las condiciones de continuidad en el nodo  $x_{k+1}$  toman la forma:

$$\left. \begin{aligned} a_{k+1} &= a_k + h_k \cdot b_k + h_k^2 \cdot c_k + h_k^3 \cdot d_k \\ b_{k+1} &= b_k + 2 \cdot h_k \cdot c_k + 3 \cdot h_k^2 \cdot d_k \\ c_{k+1} &= c_k + 3 \cdot h_k \cdot d_k \end{aligned} \right\} \forall k = 1, 2, \dots, n-1 \quad (3.29)$$

La condición de colocación en el nodo  $x_k$  proporciona la ecuación adicional:

$$2 \cdot c_k = p_k \cdot b_k + q_k \cdot a_k + r_k \quad (3.30)$$

Tenemos así el sistema lineal de ecuaciones:

$$\begin{cases} a_k + h_k \cdot b_k + h_k^2 \cdot c_k + h_k^3 \cdot d_k = a_{k+1} \\ b_k + 2 \cdot h_k \cdot c_k + 3 \cdot h_k^2 \cdot d_k = b_{k+1} \\ c_k + 3 \cdot h_k \cdot d_k = c_{k+1} \\ q_k \cdot a_k + p_k \cdot b_k - 2 \cdot c_k = -r_k \end{cases} \quad (3.31)$$

en términos de las incógnitas  $a_k$ ,  $b_k$ ,  $c_k$  y  $d_k$ .

Para resolver este sistema, lo planteamos en forma matricial y aplicamos eliminación gaussiana:

$$\begin{bmatrix} 1 & h & h^2 & h^3 & a_{k+1} \\ 0 & 1 & 2h & 3h^2 & b_{k+1} \\ 0 & 0 & 1 & 3h & c_{k+1} \\ q_k & p_k & -2 & 0 & -r_k \end{bmatrix} \rightarrow$$

$$\begin{bmatrix} 1 & h & h^2 & h^3 & a_{k+1} \\ 0 & 1 & 2 \cdot h & 3 \cdot h^2 & b_{k+1} \\ 0 & 0 & 1 & 3 \cdot h & c_{k+1} \\ 0 & p_k - q_k h & -2 - q_k h^2 & -q_k h^3 & -r_k - q_k a_{k+1} \end{bmatrix} \rightarrow$$

$$\begin{bmatrix} 1 & h & h^2 & h^3 & a_{k+1} \\ 0 & 1 & 2 \cdot h & 3 \cdot h^2 & b_{k+1} \\ 0 & 0 & 1 & 3 \cdot h & c_{k+1} \\ 0 & 0 & -2 - q_k h^2 - 2h(p_k - q_k h) & -q_k h^3 - 3h^2(p_k - q_k h) & -r_k - q_k a_{k+1} - (p_k - q_k h)b_{k+1} \end{bmatrix}$$

simplificando:

$$\begin{bmatrix} 1 & h & h^2 & h^3 & a_{k+1} \\ 0 & 1 & 2h & 3h^2 & b_{k+1} \\ 0 & 0 & 1 & 3h & c_{k+1} \\ 0 & 0 & -2 - 2p_k h + q_k h^2 & 2q_k h^3 - 3p_k h^2 & -r_k - q_k a_{k+1} - (p_k - q_k h)b_{k+1} \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} 1 & h & h^2 & h^3 & a_{k+1} \\ 0 & 1 & 2h & 3h^2 & b_{k+1} \\ 0 & 0 & 1 & 3h & c_{k+1} \\ 0 & 0 & 0 & (*) & (**) \end{bmatrix} \quad (3.32)$$

donde los términos denotados con asteriscos vienen dados por:

$$(*) \Rightarrow -q_k h^3 + 3p_k h^2 + 6h$$

$$(**) \Rightarrow -r_k - q_k a_{k+1} - (p_k - q_k h)b_{k+1} - (-2 - 2p_k h + q_k h^2)c_{k+1}$$

Despejando  $d_k$  de la última ecuación del sistema resultante asociado a la matriz (3.32), se obtiene:

$$d_k = \frac{-r_k - q_k a_{k+1} - (p_k - q_k h_k) b_{k+1} - (-2 - 2p_k h_k + q_k h_k^2) c_{k+1}}{h_k (6 + 3p_k h_k - q_k h_k^2)} \quad (3.33)$$

Las expresiones del resto de coeficientes se deducen recursivamente de la misma matriz (3.32), e implicarían utilizar el valor obtenido para  $d_k$ :

$$c_k = c_{k+1} - 3h_k d_k$$

$$b_k = b_{k+1} - 2h_k c_k - 3h_k^2 d_k \quad (3.34)$$

$$a_k = a_{k+1} - h_k b_k - h_k^2 c_k - h_k^3 d_k$$

De este modo quedaría completamente determinado  $S_k$  a partir de  $S_{k+1}$  para  $k = n - 1, n - 2, \dots, 1$ .

Solo falta ver cómo obtener los coeficientes  $a_n, b_n, c_n$  y  $d_n$  que definen  $S_n$  en  $I_n = [x_n, x_{n+1}]$ .

De las condiciones "finales" deducimos:

$$\begin{cases} y(b) = z_1 \Rightarrow S(b) = S_n(x_{n+1}) = a_n + h_n b_n + h_n^2 c_n + h_n^3 d_n = a_{n+1} = z_1 \\ y'(b) = z_2 \Rightarrow S'(b) = S'_n(x_{n+1}) = b_n + 2h_n c_n + 3h_n^2 d_n = b_{n+1} = z_2 \end{cases} \quad (3.35)$$

La condición de colocación en el nodo  $x_{n+1}$  da lugar a:

$$\begin{aligned} 2 c_{n+1} &= p_{n+1} b_{n+1} + q_{n+1} a_{n+1} + r_{n+1} \Rightarrow \\ &\Rightarrow c_{n+1} = \frac{1}{2} (p_{n+1} z_2 + q_{n+1} z_1 + r_{n+1}) \end{aligned} \quad (3.36)$$

Finalmente, la condición de colocación en  $x_n$  proporciona:

$$2 c_n = p_n b_n + q_n a_n + r_n \quad (3.37)$$

Recordando que  $c_{n+1} = c_n + 3 h_n$ , se obtiene el sistema lineal de ecuaciones a resolver, que es:

$$\left. \begin{aligned} a_n + h_n b_n + h_n^2 c_n + h_n^3 d_n &= z_1 \\ b_n + 2 h_n c_n + 3 h_n^2 d_n &= z_2 \\ c_n + 3 h_n d_n &= \frac{1}{2} (p_{n+1} z_2 + q_{n+1} z_1 + r_{n+1}) \\ q_n a_n + p_n b_n - 2 c_n &= -r_n \end{aligned} \right\} \quad (3.38)$$

y que es muy similar al anteriormente resuelto.

Resolviéndolo del mismo modo que antes, obtenemos:

$$\begin{aligned} d_n &= \\ &= \frac{-r_n - q_n z_1 - (p_n - q_n h_n) z_2 - (-2 - 2p_n h_n + q_n h_n^2) \frac{1}{2} (p_{n+1} z_2 + q_{n+1} z_1 + r_{n+1})}{h_n (6 + 3 p_n h_n - q_n h_n^2)} \end{aligned} \quad (3.39)$$

y los demás coeficientes se obtienen recursivamente a partir de  $d_n$  como sigue:

$$\left. \begin{aligned} c_n &= \frac{1}{2} (p_{n+1} z_2 + q_{n+1} z_1 + r_{n+1}) - 3 h_n d_n \\ b_n &= z_2 - 2 h_n c_n - 3 h_n^2 d_n \\ a_n &= z_1 - h_n b_n - h_n^2 c_n - h_n^3 d_n \end{aligned} \right\} \quad (3.40)$$

Tenemos así completamente definido el spline que aproxima la solución de nuestro problema.

### 3.2.4. Método de colocación basado en Pseudo- Splines Cuárticos

Nos proponemos ahora describir y construir nuevos métodos, basados en las ideas anteriores (colocación con Splines), que mejoran a los considerados en apartados anteriores.

Concretamente, buscamos aproximaciones a la solución del problema que venimos considerando, del tipo  $y(x) \cong S(x)$ ; pero donde ahora  $S(x)$  denotará un Pseudo-Spline Cuártico (de grado cuatro). Lo denominaremos Pseudo-Spline porque, aunque es de cuarto orden, solo impondremos que  $S \in C^2 [a, b]$  (en vez de  $S \in C^3 [a, b]$ ). Seguiremos tomando los nodos del intervalo:  $a = x_1 < x_2 < x_3 < \dots < x_n < x_{n+1} = b$ .

Como hemos adelantado, el problema considerado será el mismo que el que analizamos en el apartado 3.2.2, definido en (3.13). Utilizando las abreviaturas:  $y = y(x)$ ,  $p = p(x)$ ,  $q = q(x)$ ,  $r = r(x)$ , el problema viene dado por:

$$\begin{cases} y'' = p \cdot y' + q \cdot y + r \\ y(a) = z_1 \\ y'(a) = z_2 \end{cases} \quad (3.41)$$

Ahora pretendemos que nuestro spline  $S$  satisfaga la ecuación diferencial del problema (3.41), en los  $n + 1$  nodos del intervalo:  $x_k$ ,  $k = 1, 2, \dots, n + 1$ ; pero también en los  $n$  puntos intermedios:  $\frac{x_k + x_{k+1}}{2}$ ,  $k = 1, 2, \dots, n$ . Tenemos así  $2n + 1$  puntos de colocación.

Definimos  $h_k = x_{k+1} - x_k \quad \forall k = 1, 2, \dots, n$ .

En cada intervalo  $[x_k, x_{k+1}]$  ( $k = 1, 2, \dots, n$ ) describimos el Pseudo-Spline Cuártico así:

$$S_k(x) = a_k + b_k(x - x_k) + c_k(x - x_k)^2 + d_k(x - x_k)^3 + e_k(x - x_k)^4 \quad (3.42)$$

con lo que se tendrá, al derivar dicho spline:

$$S'_k(x) = b_k + 2 c_k (x - x_k) + 3 d_k (x - x_k)^2 + 4 e_k (x - x_k)^3 \quad (3.43)$$

$$S''_k(x) = 2 \cdot c_k + 6 \cdot d_k \cdot (x - x_k) + 12 \cdot e_k \cdot (x - x_k)^2 \quad (3.44)$$

La propiedad de continuidad  $S \in C^2 [a, b]$ , se traduce en las condiciones:

$$\begin{cases} a_{k+1} = a_k + h_k b_k + h_k^2 c_k + h_k^3 d_k + h_k^4 e_k \\ b_{k+1} = b_k + 2 h_k c_k + 3 h_k^2 d_k + 4 h_k^3 e_k \\ c_{k+1} = c_k + 3 h_k d_k + 6 h_k^2 e_k \end{cases} \quad (3.45)$$

que en principio son válidas para  $k = 1, 2, \dots, n - 1$ ; pero si definimos:

$$\begin{cases} a_{n+1} = S(x_{n+1}) = S_n(x_{n+1}) = a_n + h_n b_n + h_n^2 c_n + h_n^3 d_n + h_n^4 e_n \\ b_{n+1} = S'(x_{n+1}) = S'_n(x_{n+1}) = b_n + 2 h_n c_n + 3 h_n^2 d_n + 4 h_n^3 e_n \\ c_{n+1} = \frac{1}{2} \cdot S''(x_{n+1}) = \frac{1}{2} \cdot S''_n(x_{n+1}) = c_n + 3 h_n d_n + 6 h_n^2 e_n \end{cases} \quad (3.46)$$

las relaciones de (3.45) pasan a ser válidas para  $k = 1, 2, \dots, n$ .

Introducimos también las notaciones siguientes:

$$x_{k+\frac{1}{2}} = \frac{x_k + x_{k+1}}{2} = x_k + \frac{h_k}{2}, \quad k = 1, 2, \dots, n$$

$$p_k = p(x_k), \quad q_k = q(x_k), \quad r_k = r(x_k), \quad k = 1, 2, \dots, n + 1 \quad (3.47)$$

$$p_{k+\frac{1}{2}} = p\left(x_{k+\frac{1}{2}}\right), \quad q_{k+\frac{1}{2}} = q\left(x_{k+\frac{1}{2}}\right), \quad r_{k+\frac{1}{2}} = r\left(x_{k+\frac{1}{2}}\right), \quad k = 1, 2, \dots, n$$

Que el spline  $S$  satisfaga la ecuación diferencial del problema (3.41) en los puntos  $x_k$  y  $x_{k+\frac{1}{2}}$  se traduce en las condiciones de colocación:

$$\begin{cases} 2 c_k = p_k b_k + q_k a_k + r_k & k = 1, 2, \dots, n + 1 \\ 2 c_{k+\frac{1}{2}} = p_{k+\frac{1}{2}} b_{k+\frac{1}{2}} + q_{k+\frac{1}{2}} a_{k+\frac{1}{2}} + r_{k+\frac{1}{2}} & k = 1, 2, \dots, n \end{cases} \quad (3.48)$$

donde:

$$\begin{cases} a_{k+\frac{1}{2}} = S\left(x_{k+\frac{1}{2}}\right) \\ b_{k+\frac{1}{2}} = S'\left(x_{k+\frac{1}{2}}\right) \\ c_{k+\frac{1}{2}} = \frac{1}{2} \cdot S''\left(x_{k+\frac{1}{2}}\right) \end{cases} \quad (3.49)$$

A partir de nuestras notaciones, es fácil comprobar que se satisfacen relaciones similares a las descritas en (3.45), esto es:

$$\begin{cases} a_{k+\frac{1}{2}} = a_k + \frac{h_k}{2} b_k + \left(\frac{h_k}{2}\right)^2 c_k + \left(\frac{h_k}{2}\right)^3 d_k + \left(\frac{h_k}{2}\right)^4 e_k \\ b_{k+\frac{1}{2}} = b_k + 2 \left(\frac{h_k}{2}\right) c_k + 3 \left(\frac{h_k}{2}\right)^2 d_k + 4 \left(\frac{h_k}{2}\right)^3 e_k \\ c_{k+\frac{1}{2}} = c_k + 3 \left(\frac{h_k}{2}\right) d_k + 6 \left(\frac{h_k}{2}\right)^2 e_k \end{cases} \quad (3.50)$$

Las dos condiciones adicionales para determinar unívocamente el spline se obtienen imponiendo las dos condiciones iniciales que acompañan a la ecuación diferencial ordinaria del problema (3.41). De hecho:

$$\begin{cases} y(a) = z_1 \Rightarrow S(a) = S_1(a) = a_1 = z_1 \\ y'(a) = z_2 \Rightarrow S'(a) = S'_1(a) = b_1 = z_2 \end{cases} \quad (3.51)$$

Finalmente, tras todo el proceso, el sistema lineal de ecuaciones que vamos a resolver para obtener los coeficientes del spline, se obtiene añadiendo a las dos condiciones iniciales las siguientes:

$$\begin{cases} a_{k+1} = a_k + h_k b_k + h_k^2 c_k + h_k^3 d_k + h_k^4 e_k \\ b_{k+1} = b_k + 2 h_k c_k + 3 h_k^2 d_k + 4 h_k^3 e_k \\ c_{k+1} = c_k + 3 h_k d_k + 6 h_k^2 e_k \end{cases}, \quad k = 1, 2, \dots, n \quad (3.45)$$

$$\begin{cases} 2 c_{k+1} = p_{k+1} b_{k+1} + q_{k+1} a_{k+1} + r_{k+1}, & k = 0, 1, \dots, n \\ 2 c_{k+\frac{1}{2}} = p_{k+\frac{1}{2}} b_{k+\frac{1}{2}} + q_{k+\frac{1}{2}} a_{k+\frac{1}{2}} + r_{k+\frac{1}{2}}, & k = 1, 2, \dots, n \end{cases} \quad (3.48)$$

donde:

$$\begin{cases} c_{k+\frac{1}{2}} = c_k + \frac{3}{2} h_k d_k + \frac{3}{2} h_k^2 e_k \\ b_{k+\frac{1}{2}} = b_k + h_k c_k + \frac{3}{4} h_k^2 d_k + \frac{1}{2} h_k^3 e_k \\ a_{k+\frac{1}{2}} = a_k + \frac{1}{2} h_k b_k + \frac{1}{4} h_k^2 c_k + \frac{1}{8} h_k^3 d_k + \frac{1}{16} h_k^4 e_k \end{cases}, \quad k = 1, 2, \dots, n \quad (3.50)$$

Nótese que, en principio, tenemos  $5n + 3$  condiciones; pero tres de ellas no imponen ninguna restricción por ser fruto de la definición de  $a_{n+1}$ ,  $b_{n+1}$  y  $c_{n+1}$  (que no son parámetros involucrados en la definición del spline) hecha en (3.46) y que nos permite describir las restantes condiciones en una forma más compacta y uniforme.



Empezamos describiendo como podemos despejar  $d_k$ ,  $e_k$  en función de  $a_k$ ,  $b_k$ ,  $c_k$ , en cada intervalo  $I_k = [x_k, x_{k+1}]$ .

La condición de colocación en el nodo  $x_{k+1}$  es:

$$2 c_{k+1} = p_{k+1} b_{k+1} + q_{k+1} a_{k+1} + r_{k+1} \quad (3.52)$$

que en virtud de (3.45) puede reescribirse como sigue:

$$\begin{aligned} 2(c_k + 3 h_k d_k + 6 h_k^2 e_k) &= p_{k+1} (b_k + 2 h_k c_k + 3 h_k^2 d_k + 4 h_k^3 e_k) + \\ &+ q_{k+1} \cdot (a_k + h_k b_k + h_k^2 c_k + h_k^3 d_k + h_k^4 e_k) + r_{k+1} \Rightarrow \\ \Rightarrow h_k (6 - 3 p_{k+1} h_k - q_{k+1} h_k^2) d_k &+ h_k^2 (12 - 4 p_{k+1} h_k - q_{k+1} h_k^2) e_k = \\ &= r_{k+1} + q_{k+1} a_k + (p_{k+1} + q_{k+1} h_k) \cdot b_k + (-2 + 2 p_{k+1} h_k + q_{k+1} h_k^2) \cdot c_k \end{aligned} \quad (3.53)$$

Análogamente, imponiendo la condición de colocación en el nodo  $x_{k+\frac{1}{2}}$  se deduce:

$$\begin{aligned} \frac{h_k}{2} \left(6 - \frac{3}{2} p_{k+\frac{1}{2}} h_k - \frac{1}{4} q_{k+\frac{1}{2}} h_k^2\right) d_k &+ \frac{h_k^2}{4} \left(12 - 2 p_{k+\frac{1}{2}} h_k - \frac{1}{4} q_{k+\frac{1}{2}} h_k^2\right) e_k = \\ = r_{k+\frac{1}{2}} + q_{k+\frac{1}{2}} a_k + \left(p_{k+\frac{1}{2}} + \frac{1}{2} q_{k+\frac{1}{2}} h_k\right) b_k &+ \left(-2 + p_{k+\frac{1}{2}} h_k + \frac{1}{4} q_{k+\frac{1}{2}} h_k^2\right) c_k \end{aligned} \quad (3.54)$$

Resolvemos el sistema 2x2 en las incógnitas  $d_k$  y  $e_k$ . Podemos plantear las ecuaciones resultantes en la forma matricial:

$$[A] \cdot [X] = [B] \rightarrow \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \cdot \begin{bmatrix} d_k \\ e_k \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

y abreviando  $h_k = h$  resulta el sistema:

$$\begin{aligned} &\begin{bmatrix} h(6 - 3 p_{k+1} h - q_{k+1} h^2) & h^2(12 - 4 p_{k+1} h - q_{k+1} h^2) \\ h(3 - 0,75 p_{k+\frac{1}{2}} h - 0,125 q_{k+\frac{1}{2}} h^2) & h^2(3 - 0,5 p_{k+\frac{1}{2}} h - 0,0625 q_{k+\frac{1}{2}} h^2) \end{bmatrix} \cdot \begin{bmatrix} d_k \\ e_k \end{bmatrix} \\ = &\begin{bmatrix} r_{k+1} + q_{k+1} a_k + (p_{k+1} + q_{k+1} h) b_k + (-2 + 2 p_{k+1} h + q_{k+1} h^2) c_k \\ r_{k+\frac{1}{2}} + q_{k+\frac{1}{2}} a_k + \left(p_{k+\frac{1}{2}} + 0,5 q_{k+\frac{1}{2}} h\right) b_k + \left(-2 + p_{k+\frac{1}{2}} h + 0,25 q_{k+\frac{1}{2}} h^2\right) c_k \end{bmatrix} \end{aligned} \quad (3.55)$$

Un sistema lineal del tipo:  $\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$ , se puede resolver mediante el siguiente procedimiento:

$$\begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix} \cdot \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} a_{11} \cdot a_{22} - a_{12} \cdot a_{21} & 0 \\ 0 & a_{11} \cdot a_{22} - a_{12} \cdot a_{21} \end{bmatrix}$$

y la solución resulta ser:

$$x_1 = \frac{a_{22} \cdot b_1 - a_{12} \cdot b_2}{a_{11} \cdot a_{22} - a_{12} \cdot a_{21}}, \quad x_2 = \frac{-a_{21} \cdot b_1 + a_{11} \cdot b_2}{a_{11} \cdot a_{22} - a_{12} \cdot a_{21}} \quad (3.56)$$

Aplicando lo anterior al sistema (3.56) es fácil deducir que:

$$d_k = \frac{a_{22} \cdot b_1 - a_{12} \cdot b_2}{a_{11} \cdot a_{22} - a_{12} \cdot a_{21}}, \quad e_k = \frac{-a_{21} \cdot b_1 + a_{11} \cdot b_2}{a_{11} \cdot a_{22} - a_{12} \cdot a_{21}} \quad (3.57)$$

y solo faltaría sustituir en los segundos miembros de las expresiones dadas en (3.57), los elementos de las matrices  $[A]$  y  $[B]$  que se deducen del sistema (3.55) (eso sí, con  $h_k$  en vez de  $h$ ).

Ahora completaremos la descripción de cómo obtener el Pseudo-Spline Cuártico por un procedimiento inductivo. Para ello, observemos que en el primer intervalo  $I_1 = [x_1, x_2]$  se deduce, de las dos condiciones iniciales (véase (3.51)):

$$\begin{cases} y(a) = z_1 \Rightarrow S(a) = S_1(a) = a_1 = z_1 \\ y'(a) = z_2 \Rightarrow S'(a) = S_1'(a) = b_1 = z_2 \end{cases} \quad (3.58)$$

y de la condición de colocación en el primer nodo  $x_1 = a$ :

$$S_1''(a) = p_1 S_1'(a) + q_1 S_1(a) + r_1 \Rightarrow c_1 = \frac{1}{2} (p_1 z_2 + q_1 z_1 + r_1) \quad (3.59)$$

A partir de estos valores de  $a_1$ ,  $b_1$  y  $c_1$  se determinan sin dificultad  $d_1$  y  $e_1$  resolviendo el sistema de dos ecuaciones lineales del modo anteriormente descrito, con lo que queda totalmente determinado  $S$  en  $I_1$ .

Tras esto, pasamos a completar nuestro procedimiento inductivo, viendo cómo determinar  $S$  en los restantes  $I_k = [x_k, x_{k+1}]$ , para  $k = 2, 3, \dots, n$ .

Supuestos conocidos  $a_{k-1}$ ,  $b_{k-1}$ ,  $c_{k-1}$ ,  $d_{k-1}$  y  $e_{k-1}$ , esto es, los coeficientes que definen  $S_{k-1}$  en  $I_{k-1}$ , se obtienen directamente de las condiciones de continuidad en el nodo  $x_k$  las relaciones (véase (3.45)):

$$\begin{cases} a_k = a_{k-1} + b_{k-1} h_{k-1} + c_{k-1} h_{k-1}^2 + d_{k-1} h_{k-1}^3 + e_{k-1} h_{k-1}^4 \\ b_k = b_{k-1} + 2 c_{k-1} h_{k-1} + 3 d_{k-1} h_{k-1}^2 + 4 e_{k-1} h_{k-1}^3 \\ c_k = c_{k-1} + 3 d_{k-1} h_{k-1} + 6 e_{k-1} h_{k-1}^2 \end{cases} \quad (3.60)$$

que determinan los valores de  $a_k$ ,  $b_k$  y  $c_k$ .

Las dos condiciones de colocación en los puntos  $x_{k+\frac{1}{2}}$  y  $x_{k+1}$  dan lugar a un sistema de dos ecuaciones lineales en las dos incógnitas  $d_k$  y  $e_k$ , que resolveríamos del modo anteriormente descrito, con lo que  $S$  quedaría completamente descrito en el intervalo  $I_k$ . Esto completa nuestro razonamiento inductivo que determina unívocamente  $S$  en todo el intervalo.

En el programa de Matlab que hemos desarrollado y posteriormente utilizaremos, en realidad se resuelve el sistema anterior tomando como incógnitas  $h d_k$  y  $h^2 e_k$ , pues es posible sacar factor común y hacer más sencillos los cálculos.

Además, como puede observarse en el código anexo, se introducen los siguientes cambios en la nomenclatura:

$$\begin{aligned} e_k &\rightarrow c_{1k}, & d_k &\rightarrow c_{2k}, & c_k &\rightarrow c_{3k}, & b_k &\rightarrow c_{4k}, & a_k &\rightarrow c_{5k} \\ p_{k+\frac{1}{2}} &\rightarrow P(2k), & q_{k+\frac{1}{2}} &\rightarrow Q(2k), & r_{k+\frac{1}{2}} &\rightarrow R(2k), & k &= 1 \dots n \\ p_k &\rightarrow P(2k-1), & q_k &\rightarrow Q(2k-1), & r_k &\rightarrow R(2k-1), & k &= 1 \dots n+1 \end{aligned}$$

con objeto de facilitar y simplificar su programación, así como de reducir el coste computacional asociado (evitando repetición de evaluaciones).

Este programa de Matlab y los experimentos numéricos que realizaremos, nos permitirán más adelante verificar la utilidad de este nuevo método, permitiéndonos observar su mejora en orden y exactitud respecto del anteriormente considerado.

### 3.3. Resolución de Problemas Lineales con Valores en la Frontera <sup>[18] a [30]</sup>

#### 3.3.1. Método de colocación basado en Splines Cúbicos para P.V.F.s

Vamos a explicar ahora el planteamiento de un nuevo método de resolución basado en Splines Cúbicos, aplicado en este caso a Problemas con Valores en la Frontera del tipo:

$$\begin{cases} y'' = p y' + q y + r & , \quad x \in [a, b] \\ z_{11} y(a) + z_{12} y'(a) = z_{13} \\ z_{21} y(b) + z_{22} y'(b) = z_{23} \end{cases} \quad (3.61)$$

que admitan solución única.

Hemos expresado las condiciones de contorno de este problema (3.61) de manera genérica, para que pueda aplicarse a una gran variedad de casos de interés. Además, como se puede observar, hemos utilizado la notación abreviada:  $y = y(x)$ ,  $p = p(x)$ ,  $q = q(x)$ ,  $r = r(x)$ .

Buscamos, del mismo modo que hacíamos para Problemas de Valores Iniciales, aproximar  $y(x) \cong S(x)$ , siendo  $S$  un Spline Cúbico definido en los nodos:  $a = x_1 < x_2 < x_3 < \dots < x_n < x_{n+1} = b$ .

Como viene siendo habitual, definimos:

$$\left. \begin{aligned} I_k &= [x_k, x_{k+1}] \\ h_k &= x_{k+1} - x_k \end{aligned} \right\} \quad k = 1, 2, \dots, n \quad (3.62)$$

A fin de simplificar las ecuaciones y el proceso de resolución, describiremos en cada intervalo  $I_k$ , el spline  $S$ , que viene dado en dicho intervalo por un polinomio  $S_k$  de grado  $\leq 3$ , en una base adecuada.

Recogemos a continuación las expresiones que definen  $S_k$  y sus primeras derivadas en el intervalo  $I_k$ :

$$\begin{cases} S_k(x) = \frac{1}{h_k} [M_{k+1}(x - x_k)^3 + M_k(x_{k+1} - x)^3 + N_{k+1}(x - x_k) + N_k(x_{k+1} - x)] \\ S'_k(x) = \frac{1}{h_k} [3 M_{k+1}(x - x_k)^2 - 3 M_k(x_{k+1} - x)^2 + N_{k+1} - N_k] \\ S''_k(x) = \frac{1}{h_k} [6 M_{k+1}(x - x_k) + 6 M_k(x_{k+1} - x)] \end{cases} \quad (3.63)$$

Es inmediato comprobar que  $S''$  es continua en  $[a, b]$  por construcción, pues en cada nodo intermedio  $x_2, x_3, \dots, x_n$  se tiene:

$$S''_k(x_{k+1}) = 6 \cdot M_{k+1} = S''_{k+1}(x_{k+1}), \quad k = 1, 2, \dots, n-1 \quad (3.64)$$

Por otra parte, la continuidad de  $S$  en  $[a, b]$  impone las condiciones:

$$S_k(x_{k+1}) = S_{k+1}(x_{k+1}), \quad k = 1, 2, \dots, n-1$$

esto es:

$$h_k^2 \cdot M_{k+1} + N_{k+1} = h_{k+1}^2 \cdot M_{k+1} + N_{k+1} \quad (3.65)$$

La ecuación anterior se satisface automáticamente si tomamos un tamaño de paso uniforme de modo que:

$$h_k = h_{k+1} = h = \frac{b-a}{n} \quad \forall k = 1, 2, \dots, n \quad (3.66)$$

Para que  $S'$  también sea continua se habrá de verificar que:

$$S'_k(x_{k+1}) = S'_{k+1}(x_{k+1}), \quad k = 1, 2, \dots, n-1$$

que se traduce en:

$$\begin{aligned} 3h M_{k+1} + \frac{1}{h}(N_{k+1} - N_k) &= -3h M_{k+1} + \frac{1}{h}(N_{k+2} - N_{k+1}) \Rightarrow \\ \Rightarrow 6h^2 M_{k+1} &= N_{k+2} - 2N_{k+1} + N_k, \quad k = 1, 2, \dots, n-1 \end{aligned} \quad (3.67)$$

Imponemos ahora  $n+1$  condiciones de colocación consistentes en que  $S$  satisfaga la ecuación diferencial de nuestro problema dada en (3.53), en los  $n+1$  nodos:  $x_1, x_2, \dots, x_{n+1}$ , esto es:

$$S''(x_k) = p_k \cdot S'(x_k) + q_k \cdot S(x_k) + r_k \quad (3.68)$$

donde recordamos que:  $p_k = p(x_k)$ ,  $q_k = q(x_k)$ ,  $r_k = r(x_k)$ .

Para  $k = 1, 2, \dots, n$  la ecuación anterior (3.68) se traduce en:

$$6M_k = \frac{p_k}{h}(-3h^2 M_k + N_{k+1} - N_k) + \frac{q_k}{h}(h^3 M_k + h N_k) + r_k \quad (3.69)$$

Mientras que para  $k = n+1$ , da lugar a:

$$6M_{n+1} = \frac{p_{n+1}}{h}(3h^2 M_{n+1} + N_{n+1} - N_n) + \frac{q_{n+1}}{h}(h^3 M_{n+1} + h N_{n+1}) + r_{n+1} \quad (3.70)$$

Solo falta añadir a lo anterior las ecuaciones que proporcionan las dos condiciones de contorno que acompañan a la E.D.O del problema (3.61) y que garantizan la existencia de una solución única del mismo.

Para ello comenzamos escribiendo los valores que toman el Spline y sus derivadas en el primer punto  $x_1$  del intervalo  $I_1$ , esto es:

$$\begin{cases} S_1(x_1) = h^2 M_1 + N_1 \\ S'_1(x_1) = -3 h M_1 + \frac{N_2 - N_1}{h} \\ S''_1(x_1) = 6 M_1 \end{cases} \quad (3.71)$$

y tras ello, hacemos lo mismo, pero para el punto final del último intervalo  $I_n$ , obteniendo:

$$\begin{cases} S_n(x_{n+1}) = h^2 M_{n+1} + N_{n+1} \\ S'_n(x_{n+1}) = 3 h M_{n+1} + \frac{N_{n+1} - N_n}{h} \\ S''_n(x_{n+1}) = 6 M_{n+1} \end{cases} \quad (3.72)$$

Buscamos ahora obtener un sistema lineal de ecuaciones en términos solo de los  $N_k$ , a partir de las relaciones obtenidas. Para los puntos intermedios del intervalo  $x_2, x_3, \dots, x_n$ , la ecuación (3.69) se traduce en:

$$6 M_{k+1} = \frac{p_{k+1}}{h} (-3 h^2 M_{k+1} + N_{k+2} - N_{k+1}) + q_{k+1} (h^2 M_{k+1} + N_{k+1}) + r_{k+1} \quad (3.73)$$

para  $k = 1, 2, \dots, n - 1$ .

Despejando  $M_{k+1}$  de (3.67) y sustituyendo en (3.73) se tiene:

$$\begin{aligned} \frac{N_{k+2} - 2 N_{k+1} + N_k}{h^2} &= \frac{p_{k+1}}{2 h} \cdot (-N_{k+2} + 2 N_{k+1} - N_k + 2 N_{k+2} - 2 N_{k+1}) + \\ &+ \frac{q_{k+1}}{6} \cdot (N_{k+2} - 2 N_{k+1} + N_k + 6 N_{k+1}) + r_{k+1} \Rightarrow \\ 6 (N_{k+2} - 2 N_{k+1} + N_k) &= 3 h p_{k+1} \cdot (N_{k+2} - N_k) + \\ &+ h^2 q_{k+1} \cdot (N_{k+2} + 4 N_{k+1} + N_k) + 6 h^2 r_{k+1} \Rightarrow \\ (6 + 3 h p_{k+1} - h^2 q_{k+1}) N_k &+ (-12 - 4 h^2 q_{k+1}) \cdot N_{k+1} + \\ &+ (6 - 3 h p_{k+1} - h^2 q_{k+1}) N_{k+2} = 6 h^2 r_{k+1} \end{aligned} \quad (3.74)$$

Para los puntos inicial y final  $x_1$  y  $x_{n+1}$ , las relaciones (3.69) y (3.70) dan lugar a las ecuaciones:

$$\begin{cases} 6 M_1 = \frac{p_1}{h} (-3 h^2 M_1 + N_2 - N_1) + q_1 (h^2 M_1 + N_1) + r_1 \\ 6 M_{n+1} = \frac{p_{n+1}}{h} (3 h^2 M_{n+1} + N_{n+1} - N_n) + q_{n+1} (h^2 M_{n+1} + N_{n+1}) + r_{n+1} \end{cases} \quad (3.75)$$

Al igual que antes, buscamos eliminar  $M_1$  y  $M_{n+1}$  en función de los  $N_k$  en las ecuaciones anteriores, despejando de las condiciones de contorno dadas en el problema (3.61) a partir de las expresiones (3.71) y (3.72). Esto nos permite obtener  $M_1 = M_1(N_1, N_2)$  y  $M_{n+1} = M_{n+1}(N_n, N_{n+1})$ . Concretamente:

$$\begin{cases} z_{11} (h^2 M_1 + N_1) + z_{12} \left( -3 h M_1 + \frac{N_2 - N_1}{h} \right) = z_{13} \\ z_{21} (h^2 M_{n+1} + N_{n+1}) + z_{22} \left( 3 h M_{n+1} + \frac{N_{n+1} - N_n}{h} \right) = z_{23} \end{cases} \Rightarrow$$

$$\begin{cases} M_1 = \frac{z_{13} + \left( -z_{11} + \frac{z_{12}}{h} \right) N_1 - \frac{z_{12}}{h} N_2}{h \cdot (-3 z_{12} + z_{11} h)} = \frac{-z_{12} N_2 + (z_{12} - z_{11} h) N_1 + z_{13} h}{h^2 \cdot (-3 z_{12} + z_{11} h)} \\ M_{n+1} = \frac{z_{23} + \left( -z_{21} - \frac{z_{22}}{h} \right) N_{n+1} + \frac{z_{22}}{h} N_n}{h \cdot (3 z_{22} + z_{21} h)} = \frac{z_{22} N_n + (-z_{22} - z_{21} h) N_{n+1} + z_{23}}{h^2 \cdot (3 z_{22} + z_{21} h)} \end{cases} \quad (3.76)$$

Al llevar estas expresiones (3.76) a las relaciones recogidas en (3.71) se obtienen dos ecuaciones lineales que completan el sistema lineal tridiagonal que hemos de resolver para obtener el spline buscado  $S$ .

Concretamente, para la primera ecuación tenemos:

$$\begin{aligned} & \frac{-6 z_{12} N_2 + 6 \cdot (z_{12} - z_{11} h) N_1 + 6 z_{13} h}{-3 z_{12} + z_{11} h} = \\ & = p_1 h \left( \frac{3 z_{12} N_2 - 3 (z_{12} - z_{11} h) N_1 - 3 z_{13} h}{-3 z_{12} + z_{11} h} + N_2 - N_1 \right) + \\ & + q_1 h^2 \left( \frac{-z_{12} N_2 + (z_{12} - z_{11} h) N_1 + z_{13} h}{-3 z_{12} + z_{11} h} + N_1 \right) + r_1 h^2 \Rightarrow \end{aligned}$$

y operando para simplificar esta expresión:

$$\begin{aligned} & -6 z_{12} N_2 + 6 (z_{12} - z_{11} h) N_1 + 6 z_{13} h = \\ & = p_1 h [3 z_{12} N_2 - 3 (z_{12} - z_{11} h) N_1 - 3 z_{13} h + (-3 z_{12} + z_{11} h) (N_2 - N_1)] + \\ & + q_1 h^2 [-z_{12} N_2 + (z_{12} - z_{11} h) N_1 + z_{13} h + (-3 z_{12} + z_{11} h) N_1] + \\ & + r_1 (-3 z_{12} + z_{11} h) h^2 \Rightarrow \end{aligned}$$

$$\begin{aligned}
 & N_1 [6 (z_{12} - z_{11} h) + h p_1 [3 (z_{12} - z_{11} h) + (-3 z_{12} + z_{11} h)] - \\
 & \quad - q_1 h^2 [(z_{12} - z_{11} h) + (-3 z_{12} + z_{11} h)]] + \\
 & + N_2 [-6 z_{12} - p_1 h [3 z_{12} + (-3 z_{12} + z_{11} h)] - q_1 h^2 (-z_{12})] = \\
 & = (-3 z_{12} + z_{11} h) h^2 r_1 - 6 z_{13} h - 3 z_{13} h^2 p_1 + z_{13} h^3 q_1 \Rightarrow \\
 & [6 (z_{12} - z_{11} h) - 2 z_{11} h^2 p_1 + 2 z_{12} h^2 q_1] N_1 + \\
 & \quad + [-6 z_{12} - z_{11} h^2 p_1 + z_{12} h^2 q_1] N_2 = \\
 & = (-3 z_{12} + z_{11} h) h^2 r_1 - 6 z_{13} h - 3 z_{13} h^2 p_1 + z_{13} h^3 q_1
 \end{aligned}$$

esto es:

$$\begin{aligned}
 & [2 (3 + q_1 h^2) z_{12} - 2 (3 + p_1 h) h z_{11}] N_1 - [(6 - q_1 h^2) z_{12} + p_1 h^2 z_{11}] N_2 = \\
 & = -(6 + 3 p_1 h - q_1 h^2) h z_{13} + r_1 h^2 (-3 z_{12} + h z_{11}) \quad (3.77)
 \end{aligned}$$

Análogamente, para la otra ecuación:

$$\begin{aligned}
 & \frac{6 z_{22} N_n + 6 (-z_{22} - z_{21} h) N_{n+1} + 6 z_{23} h}{3 z_{22} + z_{21} h} = \\
 & = p_{n+1} h \left( \frac{3 z_{22} N_n + 3 (-z_{22} - z_{21} h) N_{n+1} + 3 z_{23} h}{3 z_{22} + z_{21} h} + N_{n+1} - N_n \right) + \\
 & + q_{n+1} h^2 \left( \frac{z_{22} N_n + (-z_{22} - z_{21} h) N_{n+1} + z_{23} h}{3 z_{22} + z_{21} h} + N_{n+1} \right) + r_{n+1} h^2 \Rightarrow
 \end{aligned}$$

y operando:

$$\begin{aligned}
 & 6 z_{22} N_n - 6 (z_{22} + z_{21} h) N_{n+1} + 6 z_{23} h = \\
 & = p_{n+1} h [3 z_{22} N_n - 3 (z_{22} + z_{21} h) N_{n+1} + 3 z_{23} h + (3 z_{22} + z_{21} h) (N_{n+1} - N_n)] \\
 & + q_{n+1} h^2 [z_{22} N_n - (z_{22} + z_{21} h) N_{n+1} + z_{23} h + (3 z_{22} + z_{21} h) N_{n+1}] + \\
 & \quad + r_{n+1} (3 z_{22} + z_{21} h) h^2 \Rightarrow
 \end{aligned}$$

$$\begin{aligned}
 & [6 z_{22} - 3 p_{n+1} h z_{22} + (3 z_{22} + z_{21} h) h p_{n+1} - q_{n+1} h^2 z_{22}] N_n + \\
 & + [-6 (z_{22} + z_{21} h) + 3 p_{n+1} (z_{22} + z_{21} h) h - p_{n+1} (3 z_{22} + z_{21} h) h + \\
 & \quad + q_{n+1} (z_{22} + z_{21} h) h^2 - q_{n+1} (3 z_{22} + z_{21} h) h^2] N_{n+1} = \\
 & = (3 z_{22} + z_{21} h) h^2 r_{n+1} - 6 z_{23} h + 3 z_{23} h^2 p_{n+1} + z_{23} h^3 q_{n+1} \Rightarrow
 \end{aligned}$$



$$\begin{aligned}
 & [6 z_{22} + z_{21} h^2 p_{n+1} - z_{22} h^2 q_{n+1}] N_n + \\
 & + [-6 (z_{22} + z_{21} h) + 2 z_{21} h^2 p_{n+1} - 2 z_{22} h^2 q_{n+1}] N_{n+1} = \\
 & = (3 z_{22} + z_{21} h) h^2 r_{n+1} - 6 z_{23} h + 3 z_{23} h^2 p_{n+1} + z_{23} h^3 q_{n+1}
 \end{aligned}$$

esto es:

$$\begin{aligned}
 & [(6 - q_{n+1} h^2) z_{22} + p_{n+1} h^2 z_{21}] N_n + \\
 & + [-2 (3 + q_{n+1} h^2) z_{22} + 2 (-3 + p_{n+1} h) h z_{21}] N_{n+1} = \quad (3.78) \\
 & = (-6 + 3 p_{n+1} h + q_{n+1} h^2) h z_{23} + r_{n+1} h^2 (3 z_{22} + h z_{21})
 \end{aligned}$$

Finalmente se resuelve el sistema tridiagonal de  $n + 1$  ecuaciones lineales en las incógnitas  $N_k$ ,  $k = 1, 2, \dots, n + 1$ .

Conocidos los  $N_k$ , los  $M_k$  se calculan a partir de (3.67) y (3.76) y el spline cúbico queda completamente determinado.

Aunque no sería necesario, pues podemos evaluar nuestro Spline en las bases consideradas en su obtención (reduciendo de hecho el coste computacional asociado), pretendemos reescribir el Spline obtenido en las bases "canónicas" (en cada intervalo, centradas en el punto inicial de dicho intervalo) consideradas al obtener el Spline Cúbico de colocación para el Problema de Valores Iniciales, esto es:

$$\begin{aligned}
 S_k(x) &= \frac{1}{h} [M_{k+1} (x - x_k)^3 + M_k (x_{k+1} - x)^3 + N_{k+1} (x - x_k) + \\
 & + N_k (x_{k+1} - x)] = c_{k1} (x - x_k)^3 + c_{k2} (x - x_k)^2 + c_{k3} (x - x_k) + c_{k4} \quad (3.79)
 \end{aligned}$$

Hacemos esto con vistas a utilizar en su construcción y evaluación, mediante el programa Matlab, las funciones "mkpp" y "ppval" respectivamente, que implementa dicho programa y que utilizan esas bases al construir y evaluar funciones polinomiales a trozos.

Con este fin, basta observar que se verifica:

$$\begin{aligned}
 M_k(x_{k+1} - x)^3 &= -M_k(x - x_{k+1})^3 = \\
 & -M_k(x - x_k + x_k - x_{k+1})^3 = -M_k[(x - x_k) - h]^3 = \\
 & = -M_k[(x - x_k)^3 - 3h (x - x_k)^2 + 3h^2 (x - x_k) - h^3] \quad (3.80)
 \end{aligned}$$

y:

$$N_k (x_{k+1} - x) = -N_k (x - x_{k+1}) = -N_k [(x - x_k) - h] \quad (3.81)$$

para concluir que:

$$\begin{cases} c_{k1} = \frac{1}{h} [M_{k+1} - M_k] \\ c_{k2} = 3 M_k \\ c_{k3} = -3 h M_k + (N_{k+1} - N_k) \frac{1}{h} \\ c_{k4} = h^2 M_k + N_k \end{cases} \quad (3.82)$$

### 3.3.2. Método de colocación basado en Pseudo-Splines Cuárticos para P.V.F.

Del mismo modo que hicimos para problemas de valores iniciales, vamos a desarrollar un novedoso método basado en Pseudo-Splines Cuárticos, para resolver Problemas con Valores en la Frontera que vienen definidos por la ecuación diferencial lineal que venimos considerando de manera ya habitual:  $y''(x) = p(x) y'(x) + q(x) y(x) + r(x)$ , junto a un par de condiciones de frontera, donde  $x \in [a, b]$ .

Recordamos que denotábamos el tamaño de paso como:  $h = \frac{b-a}{n}$ , y los diferentes intervalos en que queda dividido el intervalo  $[a, b]$  por los  $n + 1$  nodos  $a = x_1 < x_2 < x_3 < \dots < x_n < x_{n+1} = b$ , mediante:  $I_k = [x_k, x_{k+1}]$  siendo  $x_k = a + (k - 1) \cdot h$  con  $k = 1, 2, \dots, n + 1$ .

En el intervalo  $I_k = [x_k, x_{k+1}]$ , se define el Spline que aproxima la solución de nuestro P.V.F. por:

$$\begin{aligned} S_k(x) = & \frac{1}{h^4} [A_{k+1} (x - x_k)^4 + A_k (x_{k+1} - x)^4] + \\ & + \frac{1}{h^3} [B_{k+1} (x - x_k)^3 + B_k (x_{k+1} - x)^3] + \\ & + \frac{1}{h^2} W_k (x - x_k) (x_{k+1} - x) \left[ 1 + \frac{1}{h^2} (x - x_k) (x_{k+1} - x) \right] \end{aligned} \quad (3.83)$$

Para sus primeras derivadas se tiene:

$$\begin{aligned}
 S'_k(x) = & \frac{4}{h^4} [A_{k+1} (x - x_k)^3 - A_k (x_{k+1} - x)^3] + \\
 & + \frac{3}{h^3} [B_{k+1} (x - x_k)^2 - B_k (x_{k+1} - x)^2] - \\
 & - \frac{2}{h^2} W_k \left( x - x_{k+\frac{1}{2}} \right) \left[ 1 + \frac{2}{h^2} (x - x_k) (x_{k+1} - x) \right]
 \end{aligned} \tag{3.84}$$

$$\begin{aligned}
 S''_k(x) = & \frac{12}{h^4} [A_{k+1} (x - x_k)^2 + A_k (x_{k+1} - x)^2] + \\
 & + \frac{6}{h^3} [B_{k+1} (x - x_k) + B_k (x_{k+1} - x)] - \\
 & - \frac{2}{h^2} W_k \left[ 1 + \frac{2}{h^2} (x - x_k) (x_{k+1} - x) - \frac{4}{h^2} \left( x - x_{k+\frac{1}{2}} \right)^2 \right]
 \end{aligned} \tag{3.85}$$

donde  $x_{k+\frac{1}{2}} = \frac{x_k + x_{k+1}}{2} = x_k + \frac{h}{2}$ ,  $k = 1, 2, \dots, n$ .

A partir de las expresiones anteriores, es fácil comprobar que:

$$\left. \begin{aligned}
 S_k(x_k) &= A_k + B_k \\
 S'_k(x_k) &= \frac{1}{h} (W_k - 4 A_k - 3 B_k) \\
 S''_k(x_k) &= \frac{6}{h^2} (2 A_k + B_k)
 \end{aligned} \right\} k = 1, 2, \dots, n \tag{3.86}$$

$$\left. \begin{aligned}
 S_k(x_{k+1}) &= A_{k+1} + B_{k+1} \\
 S'_k(x_{k+1}) &= \frac{1}{h} (4 A_{k+1} + 3 B_{k+1} - W_k) \\
 S''_k(x_{k+1}) &= \frac{6}{h^2} (2 A_{k+1} + B_{k+1})
 \end{aligned} \right\} k = 1, 2, \dots, n \tag{3.87}$$

$$\left. \begin{aligned}
 S_k\left(x_{k+\frac{1}{2}}\right) &= \frac{1}{16} (A_k + A_{k+1} + 2 B_k + 2 B_{k+1} + 5 W_k) \\
 S'_k\left(x_{k+\frac{1}{2}}\right) &= \frac{1}{4 h} (-2 A_k + 2 A_{k+1} - 3 B_k + 3 B_{k+1}) \\
 S''_k\left(x_{k+\frac{1}{2}}\right) &= \frac{3}{h^2} (A_k + A_{k+1} + B_k + B_{k+1} - W_k)
 \end{aligned} \right\} k = 1, 2, \dots, n \tag{3.88}$$

Se tiene, por tanto, que  $S$  y  $S''$  son continuas en  $[a, b]$  por construcción. La continuidad de  $S'$  en  $[a, b]$  proporciona:

$$\begin{aligned} S'_{k-1}(x_k) &= S'_k(x_k) \Rightarrow \\ \frac{1}{h} (4 A_k + 3 B_k - W_{k-1}) &= \frac{1}{h} (W_k - 4 A_k - 3 B_k) \Rightarrow \\ 2 (4 A_k + 3 B_k) &= W_{k-1} + W_k, \quad \forall k = 2, 3, \dots, n \end{aligned} \quad (3.89)$$

Las condiciones de colocación en los nodos dan lugar a las siguientes ecuaciones.

$$\begin{aligned} \text{En } x_k \Rightarrow S''_k(x_k) &= p_k S'_k(x_k) + q_k S_k(x_k) + r_k \Rightarrow \\ \frac{6}{h^2} (2 A_k + B_k) &= \frac{1}{h} p_k (W_k - 4 A_k - 3 B_k) + q_k (A_k + B_k) + r_k \Rightarrow \\ 6 (2 A_k + B_k) &= h p_k (W_k - 4 A_k - 3 B_k) + h^2 q_k (A_k + B_k) + h^2 r_k \end{aligned} \quad (3.90)$$

para  $k = 1, 2, \dots, n$ .

$$\begin{aligned} \text{En } x_{n+1} \Rightarrow S''_n(x_{n+1}) &= p_{n+1} S'_n(x_{n+1}) + q_{n+1} S_n(x_{n+1}) + r_{n+1} \Rightarrow \\ \frac{6}{h^2} (2 A_{n+1} + B_{n+1}) &= \\ = \frac{1}{h} p_{n+1} (4 A_{n+1} + 3 B_{n+1} - W_n) &+ q_{n+1} (A_{n+1} + B_{n+1}) + r_{n+1} \Rightarrow \\ 6 (2 A_{n+1} + B_{n+1}) &= \\ = h p_{n+1} (4 A_{n+1} + 3 B_{n+1} - W_n) &+ h^2 q_{n+1} (A_{n+1} + B_{n+1}) + h^2 r_{n+1} \end{aligned} \quad (3.91)$$

En las expresiones anteriores hemos utilizado, como ya viene siendo habitual, las notaciones:  $p_k = p(x_k)$ ,  $q_k = q(x_k)$ ,  $r_k = r(x_k)$ , para los valores de  $k = 1, 2, \dots, n + 1$ .

Las condiciones de colocación en los puntos medios de los intervalos  $x_{k+\frac{1}{2}} \forall k = 1, 2, \dots, n$  proporcionan:

$$S''_k \left( x_{k+\frac{1}{2}} \right) = p_{k+\frac{1}{2}} S'_k \left( x_{k+\frac{1}{2}} \right) + q_{k+\frac{1}{2}} S_k \left( x_{k+\frac{1}{2}} \right) + r_{k+\frac{1}{2}} \Rightarrow$$

$$\begin{aligned}
 \Rightarrow \frac{3}{h^2} (A_k + A_{k+1} + B_k + B_{k+1} - W_k) &= \\
 &= \frac{1}{4h} p_{k+\frac{1}{2}} (-2A_k + 2A_{k+1} - 3B_k + 3B_{k+1}) + \\
 &+ \frac{1}{16} q_{k+\frac{1}{2}} (A_k + A_{k+1} + 2B_k + 2B_{k+1} + 5W_k) + r_{k+\frac{1}{2}} \Rightarrow \\
 \Rightarrow 48 (A_k + A_{k+1} + B_k + B_{k+1} - W_k) &= \\
 &= 4h p_{k+\frac{1}{2}} (-2A_k + 2A_{k+1} - 3B_k + 3B_{k+1}) + \\
 &+ h^2 q_{k+\frac{1}{2}} (A_k + A_{k+1} + 2B_k + 2B_{k+1} + 5W_k) + 16h^2 r_{k+\frac{1}{2}} \quad (3.92)
 \end{aligned}$$

para  $k = 1, 2, \dots, n$ , siendo:  $p_{k+\frac{1}{2}} = p(x_{k+\frac{1}{2}})$ ,  $q_{k+\frac{1}{2}} = q(x_{k+\frac{1}{2}})$  y  $r_{k+\frac{1}{2}} = r(x_{k+\frac{1}{2}})$ .

Y sólo faltaría añadir las condiciones iniciales y/o de contorno del problema que garanticen la existencia de una única solución.

Resumiendo, para obtener la expresión del Pseudo-Spline Cuártico que aproxima la solución de nuestro problema, habrá que añadir a las dos ecuaciones lineales que surgen de imponer las condiciones iniciales y/o de contorno, las siguientes  $3n$  ecuaciones:

$$8A_k + 6B_k = W_{k-1} + W_k, \quad \forall k = 2, 3, \dots, n \quad (3.89)$$

$$\begin{aligned}
 (12 + 4hp_k - h^2q_k)A_k + (6 + 3hp_k - h^2q_k)B_k &= hp_kW_k + h^2r_k \\
 \forall k = 1, 2, \dots, n \quad (3.90)
 \end{aligned}$$

$$\begin{aligned}
 (12 - 4hp_{n+1} - h^2q_{n+1})A_{n+1} + (6 - 3hp_{n+1} - h^2q_{n+1})B_{n+1} &= \\
 = -hp_{n+1}W_n + h^2r_{n+1} \quad (3.91)
 \end{aligned}$$

$$\begin{aligned}
 \left(48 + 8hp_{k+\frac{1}{2}} - h^2q_{k+\frac{1}{2}}\right)A_k + \left(48 - 8hp_{k+\frac{1}{2}} - h^2q_{k+\frac{1}{2}}\right)A_{k+1} &+ \\
 + \left(48 + 12hp_{k+\frac{1}{2}} - 2h^2q_{k+\frac{1}{2}}\right)B_k + \left(48 - 12hp_{k+\frac{1}{2}} - 2h^2q_{k+\frac{1}{2}}\right)B_{k+1} &= \\
 = \left(48 + 5h^2q_{k+\frac{1}{2}}\right)W_k + 16h^2r_{k+\frac{1}{2}}, \quad \forall k = 1, 2, \dots, n \quad (3.92)
 \end{aligned}$$

Estas ecuaciones permiten obtener los valores de los parámetros buscados:  $A_k$  y  $B_k$ , para  $k = 1, 2, \dots, n + 1$ , y  $W_k$  para  $k = 1, 2, \dots, n$  resolviendo el sistema de  $3n + 2$  ecuaciones lineales.

### Resolución

Aunque nuestro propósito principal es aplicar este método de colocación basado en Pseudo-Splines Cuárticos a la resolución aproximada de P.V.F. (de ahí el tipo de funciones base utilizadas al describir  $S$  en cada intervalo), también podría ser aplicado a P.V.I. (de hecho, ya lo hemos hecho en apartados anteriores expresando  $S$  en otra base). Vamos a explicar previamente la obtención de  $S$  (en las nuevas bases) en el caso del P.V.I., para que sea más sencillo entender cómo se aplica después a la resolución de P.V.F. con condiciones de contorno (lineales) generales.

#### A) APLICACIÓN A PROBLEMAS DE VALORES INICIALES:

En este caso consideramos la resolución del problema:

$$\left. \begin{aligned} y''(x) &= p(x) \cdot y'(x) + q(x) \cdot y(x) + r(x) \\ y(a) &= z_1 \\ y'(a) &= z_2 \end{aligned} \right\} \quad (3.93)$$

Comenzamos a resolverlo para el intervalo inicial  $I_1 = [x_1, x_2]$ , en el que podemos determinar los parámetros  $A_1, B_1$  y  $W_1$  que definen el spline a partir de las dos condiciones iniciales y la condición de colocación en el nodo  $x_1$ , como se muestra a continuación.

Planteamos las ecuaciones que se obtienen de las condiciones iniciales:

$$S(a) = y(a) = z_1 \Rightarrow S_1(a) = A_1 + B_1 = z_1 \quad (3.94)$$

$$\begin{aligned} S'(a) = y'(a) = z_2 \Rightarrow S'_1(a) &= \frac{1}{h} (W_1 - 4A_1 - 3B_1) = z_2 \\ \Rightarrow W_1 &= 4A_1 + 3B_1 + h z_2 \end{aligned} \quad (3.95)$$

y de la condición de colocación en  $x_1$ :

$$\begin{aligned} S''(a) = p_1 S'(a) + q_1 S(a) + r_1 &\Rightarrow \frac{6}{h^2} (2A_1 + B_1) = p_1 z_2 + q_1 z_1 + r_1 \Rightarrow \\ \Rightarrow 2A_1 + B_1 &= \frac{h^2}{6} (p_1 z_2 + q_1 z_1 + r_1) \end{aligned} \quad (3.96)$$

Ahora resolvemos el sistema formado por las tres ecuaciones anteriores para obtener las tres incógnitas:  $A_1, B_1$  y  $W_1$ , y obtenemos sus expresiones en función de valores conocidos:

$$\text{De (3.96) - (3.94)} \Rightarrow \quad A_1 = -z_1 + \frac{h^2}{6} (p_1 z_2 + q_1 z_1 + r_1) \quad (3.97)$$

$$\text{De (3.94)} \Rightarrow \quad B_1 = 2 z_1 - \frac{h^2}{6} (p_1 z_2 + q_1 z_1 + r_1) \quad (3.98)$$

$$\text{De (3.95)} \Rightarrow \quad W_1 = 2 z_1 + h z_2 + \frac{h^2}{6} (p_1 z_2 + q_1 z_1 + r_1) \quad (3.99)$$

Una vez definido parcialmente el Spline en el intervalo inicial (solo faltaría determinar  $A_2$  y  $B_2$ ), vamos a explicar cómo obtener todos los parámetros que definen el Spline  $S$  en un intervalo genérico:  $I_k = [x_k, x_{k+1}]$ , para valores de  $k = 1, 2, \dots, n$ . Recurriremos a un argumento inductivo que describe como obtener a partir de  $A_k, B_k$  y  $W_{k-1}$ , los parámetros  $W_k, A_{k+1}$  y  $B_{k+1}$ .

Para el caso  $k = 1$ ,  $W_1$  se calcula como acabamos de ver.

Para  $k \geq 2$ ,  $W_k$  se obtiene de la condición de continuidad de  $S'$  en  $x_k$  como:

$$W_k = -W_{k-1} + 2 (4 A_k + 3 B_k), \quad k = 2, 3, \dots, n \quad (3.100)$$

Los parámetros  $A_{k+1}$  y  $B_{k+1}$  se obtienen a partir de las condiciones de colocación en los puntos  $x_{k+\frac{1}{2}}$  y  $x_{k+1}$ , esto es, para  $k = 1, 2, \dots, n$ :

$$\left\{ \begin{array}{l} \left( 48 - 8h p_{k+\frac{1}{2}} - h^2 q_{k+\frac{1}{2}} \right) A_{k+1} + \left( 48 - 12h p_{k+\frac{1}{2}} - 2h^2 q_{k+\frac{1}{2}} \right) B_{k+1} = \\ = - \left( 48 + 8h p_{k+\frac{1}{2}} - h^2 q_{k+\frac{1}{2}} \right) A_k - \left( 48 + 12h p_{k+\frac{1}{2}} - 2h^2 q_{k+\frac{1}{2}} \right) B_k + \\ + \left( 48 + 5h^2 q_{k+\frac{1}{2}} \right) W_k + 16h^2 r_{k+\frac{1}{2}} \\ \\ \left( 12 - 4h p_{k+1} - h^2 q_{k+1} \right) A_{k+1} + \left( 6 - 3h p_{k+1} - h^2 q_{k+1} \right) B_{k+1} = \\ = -h p_{k+1} W_k + h^2 r_{k+1} \end{array} \right. \quad (3.101)$$

Resolviendo el sistema lineal  $2 \times 2$  en las incógnitas  $A_{k+1}$  y  $B_{k+1}$ , dado que todas las demás cantidades involucradas son conocidas a partir de las etapas previas, quedaría completamente definido el spline.

## B) APLICACIÓN A PROBLEMAS DE VALORES EN LA FRONTERA CON CONDICIONES DE CONTORNO GENERALES

Procedemos ahora a explicar la resolución de P.V.F con condiciones generales, es decir problemas del tipo:

$$\begin{cases} y''(x) = p(x) y'(x) + q(x) y(x) + r(x) \\ z_{11} y(a) + z_{12} y'(a) = z_{13} \\ z_{21} y(b) + z_{22} y'(b) = z_{23} \end{cases} \quad (3.102)$$

A partir de las condiciones iniciales se deduce:

$$\begin{aligned} & \begin{cases} z_{11} S(a) + z_{12} S'(a) = z_{13} \\ z_{21} S(b) + z_{22} S'(b) = z_{23} \end{cases} \Rightarrow \\ \Rightarrow & \begin{cases} z_{11} (A_1 + B_1) + z_{12} \frac{1}{h} (W_1 - 4A_1 - 3B_1) = z_{13} \\ z_{21} (A_{n+1} + B_{n+1}) + z_{22} \frac{1}{h} (4A_{n+1} + 3B_{n+1} - W_n) = z_{23} \end{cases} \Rightarrow \\ \Rightarrow & \begin{cases} (h z_{11} - 4 z_{12}) A_1 + (h z_{11} - 3 z_{12}) B_1 = h z_{13} - z_{12} W_1 \\ (h z_{21} + 4 z_{22}) A_{n+1} + (h z_{21} + 3 z_{22}) B_{n+1} = h z_{23} + z_{22} W_n \end{cases} \quad (3.103) \end{aligned}$$

Hay que añadir a estas dos condiciones, las restantes  $3n$  que ya hemos descrito, esto es:

- Las  $n - 1$  que surgen al imponer que  $S'$  sea continua en  $[a, b]$ .
- Las  $n + 1$  condiciones de colocación en los  $n + 1$  nodos.
- Las  $n$  condiciones de colocación en los  $n$  puntos medios de los intervalos.

Para resolver el sistema lineal de  $3n + 2$  ecuaciones en las  $3n + 2$  incógnitas, vamos a describir cómo podemos reducirlo a un problema más sencillo. Concretamente, lo transformaremos en un sistema lineal tridiagonal de dimensión  $n$  en las incógnitas  $W_k$ ,  $\forall k = 1, 2, \dots, n$ , que podemos resolver, por ejemplo, mediante el algoritmo de Thomas, con un coste computacional asociado muy bajo.

El proceso es un tanto largo y tedioso, por lo que no lo recogeremos aquí en detalle y nos conformaremos con describir el procedimiento seguido. No obstante, se pueden consultar las expresiones resultantes en el Anexo, en su apartado 7.1.2.B del programa que incluimos en este trabajo.



Descripción del procedimiento de resolución:

1. El primer paso es despejar, a partir de la primera condición frontera,  $B_1$  en términos de  $A_1$  y  $W_1$ , pues el resto de parámetros que aparecen en dicha ecuación son conocidos a partir de los datos del problema. Obtenemos así una expresión del tipo:

$$B_1 = B_1(A_1, W_1) \quad (3.104)$$

2. Análogamente, a partir de la condición frontera en el punto final despejamos  $B_{n+1}$  en términos de  $A_{n+1}$  y  $W_n$ , obteniendo:

$$B_{n+1} = B_{n+1}(A_{n+1}, W_n) \quad (3.105)$$

3. De las condiciones de continuidad de  $S'$  en los nodos intermedios  $x_k$ , despejamos para cada  $k$ ,  $B_k$  en términos de  $A_k$ ,  $W_{k-1}$  y  $W_k$ , obteniendo así para cada  $k = 2, 3, \dots, n$ :

$$B_k = B_k(A_k, W_{k-1}, W_k) \quad (3.106)$$

4. Sustituimos las expresiones anteriores en las  $n + 1$  condiciones de colocación correspondientes a los  $n + 1$  nodos  $x_k$  ( $k = 1, 2, \dots, n + 1$ ) y despejamos a continuación los  $A_k$  en términos de los  $W_k$ .

Es fácil comprobar que se obtienen así expresiones del tipo:

$$\left. \begin{aligned} A_1 &= A_1(W_1) \\ A_k &= A_k(W_{k-1}, W_k), \quad k = 2, 3, \dots, n \\ A_{n+1} &= A_{n+1}(W_n) \end{aligned} \right\} \quad (3.107)$$

5. Por tanto, sustituyendo estas expresiones obtenidas para los  $A_k$  para valores de  $k = 1, 2, \dots, n + 1$ , en las de los  $B_k$ , se tiene:

$$\left. \begin{aligned} B_1 &= B_1(A_1(W_1), W_1) \\ B_k &= B_k(A_k(W_{k-1}, W_k), W_{k-1}, W_k), \quad k = 2, 3, \dots, n \\ B_{n+1} &= B_{n+1}(A_{n+1}(W_n), W_n) \end{aligned} \right\} \quad (3.108)$$

6. Finalmente, llevando estas expresiones para los parámetros  $A_k$  y  $B_k$  ( $k = 1, 2, \dots, n + 1$ ), a las  $n$  restantes condiciones de colocación en los  $n$  puntos medios  $x_{k+\frac{1}{2}}$  ( $k = 1, 2, \dots, n$ ), se obtiene un sistema lineal tridiagonal de dimensión  $n$  en las  $n$  incógnitas  $W_k$  con  $k = 1, 2, \dots, n$ .

Resolviendo dicho sistema (nosotros lo hemos hecho con el Algoritmo de Thomas) se obtienen los  $W_k$  y a partir de ellos los  $A_k$  y  $B_k$ , con lo que el spline queda completamente determinado.

### 3.4. Resolución de Problemas de Valores Iniciales No Lineales [18] a [30]

En este apartado vamos a estudiar la resolución de problemas de valores iniciales definidos por ecuaciones diferenciales de segundo orden, pero que, a diferencia de apartados anteriores, se caracterizan por ser no lineales. Es decir, vamos a tratar de resolver el problema de valores iniciales definido por:

$$\left. \begin{aligned} y'' &= f(x, y, y') \\ y(a) &= z_1 \\ y'(a) &= z_2 \end{aligned} \right\} \quad (3.109)$$

donde  $f(x, y, y')$  es una función no lineal.

Del modo habitual, buscaremos encontrar una aproximación a su única solución de la forma  $y(x) \cong S(x)$  en el intervalo  $x \in [a, b]$ , siendo  $S$  un Spline basado en los nodos:  $a = x_1 < x_2 < \dots < x_n < x_{n+1} = b$ .

Como venimos haciendo, denotaremos los intervalos definidos por cada par de nodos consecutivos por  $I_k = [x_k, x_{k+1}]$  y el tamaño de paso entre estos nodos por  $h_k = x_{k+1} - x_k$ , para los valores de  $k = 1, 2, \dots, n$ .

#### 3.4.1. Método de resolución basado en Splines Cúbicos

Vamos a comenzar explicando el caso en que el Spline que aproxima la solución del problema anterior (3.109) es cúbico. En  $I_k = [x_k, x_{k+1}]$ , este Spline Cúbico  $S$  viene definido por  $S_k$  del siguiente modo:

$$S_k(x) = a_k + b_k(x - x_k) + c_k(x - x_k)^2 + d_k(x - x_k)^3 \quad (3.110)$$

Expresión de la que pueden obtenerse sus derivadas:

$$S'_k(x) = b_k + 2 c_k (x - x_k) + 3 d_k (x - x_k)^2 \quad (3.111)$$

$$S''_k(x) = 2 c_k + 6 d_k (x - x_k) \quad (3.112)$$

A partir de lo anterior se deduce que:

$$\left. \begin{aligned} S_k(x_k) &= a_k \\ S'_k(x_k) &= b_k \\ S''_k(x_k) &= 2 c_k \end{aligned} \right\} \quad (3.113)$$

$$\left. \begin{aligned} S_k(x_{k+1}) &= a_k + b_k h_k + c_k h_k^2 + d_k h_k^3 \\ S'_k(x_{k+1}) &= b_k + 2 c_k h_k + 3 d_k h_k^2 \\ S''_k(x_{k+1}) &= 2 c_k + 6 d_k h_k \end{aligned} \right\} \quad (3.114)$$

Y para que se cumpla la continuidad de  $S \in C^2$  en  $[a, b]$ , deben cumplirse las siguientes condiciones:

$$\left. \begin{aligned} S_k(x_{k+1}) &= S_{k+1}(x_{k+1}) \\ S'_k(x_{k+1}) &= S'_{k+1}(x_{k+1}) \\ S''_k(x_{k+1}) &= S''_{k+1}(x_{k+1}) \end{aligned} \right\} \quad k = 1, 2, \dots, n-1 \Rightarrow$$

$$\Rightarrow \left. \begin{aligned} a_{k+1} &= a_k + b_k h_k + c_k h_k^2 + d_k h_k^3 \\ b_{k+1} &= b_k + 2 c_k h_k + 3 d_k h_k^2 \\ c_{k+1} &= c_k + 3 d_k h_k \end{aligned} \right\} \quad (3.115)$$

Por último, las condiciones de colocación en los nodos aportan:

$$\left. \begin{aligned} S''_k(x_k) &= f(x_k, S_k(x_k), S'_k(x_k)), \quad k = 1, 2, \dots, n \\ S''_n(x_{n+1}) &= f(x_{n+1}, S_n(x_{n+1}), S'_n(x_{n+1})), \quad k = n+1 \end{aligned} \right\} \quad (3.116)$$

### Resolución

Si incluimos las dos condiciones iniciales, ya disponemos de todas las ecuaciones necesarias y podemos proceder a resolver el sistema (en general no lineal) de ecuaciones resultante. Comenzamos resolviéndolo en el primer intervalo:  $I_1 = [x_1, x_2]$ .

De las condiciones iniciales, puede deducirse con facilidad que:

$$S(a) = z_1 \Rightarrow a_1 = z_1 \quad (3.117)$$

$$S'(a) = z_2 \Rightarrow b_1 = z_2 \quad (3.118)$$

De las condiciones de colocación (3.111) para el punto  $x_1 = a$ , se deduce:

$$S''(a) = f(a, S(a), S'(a)) \Rightarrow c_1 = \frac{1}{2} f(a, z_1, z_2) \quad (3.119)$$

La condición de colocación en el punto  $x_2 = a + h_1$ , permite despejar  $d_1$  de la ecuación resultante, resolviendo si es necesario la ecuación no lineal:

$$\begin{aligned} S''(a + h_1) = f(a + h_1, S(a + h_1), S'(a + h_1)) \Rightarrow \\ 2c_1 + 6 d_1 h_1 = f(a + h_1, a_1 + h_1 b_1 + h_1^2 c_1 + h_1^3 d_1, b_1 + 2 h_1 c_1 + 3 h_1^2 d_1) \end{aligned} \quad (3.120)$$

Nótese que, en dicha ecuación no lineal, la única incógnita es  $d_1$ .

A partir de lo anterior, para el resto de intervalos  $I_k = [x_k, x_{k+1}]$ , siendo  $k = 2, \dots, n$ , procedemos inductivamente como se indica a continuación.

Conocidos  $a_{k-1}$ ,  $b_{k-1}$ ,  $c_{k-1}$  y  $d_{k-1}$ , es decir los coeficientes de  $S_{k-1}$  en  $I_{k-1}$ , se obtienen:

$$\begin{cases} a_k = a_{k-1} + b_{k-1} h_{k-1} + c_{k-1} h_{k-1}^2 + d_{k-1} h_{k-1}^3 \\ b_k = b_{k-1} + 2 c_{k-1} h_{k-1} + 3 d_{k-1} h_{k-1}^2 \\ c_k = c_{k-1} + 3 d_{k-1} h_{k-1} \end{cases} \quad (3.121)$$

a partir de las condiciones de continuidad (3.115), asociadas al nodo  $x_k$ .

Y de las condiciones de colocación de (3.116), se tiene para el nodo final  $x_{k+1}$  la condición:

$$S''_k(x_{k+1}) = f(x_{k+1}, S_k(x_{k+1}), S'_k(x_{k+1})) \quad (3.122)$$

y basta despejar  $d_k$  de la ecuación resultante:

$$2c_k + 6 d_k h_k = f(x_{k+1}, a_k + h_k b_k + h_k^2 c_k + h_k^3 d_k, b_k + 2 h_k c_k + 3 h_k^2 d_k) \quad (3.123)$$

si es necesario, resolviendo la ecuación no lineal anterior respecto de la única incógnita  $d_k$ . Con esto, quedaría completamente determinado el Spline que aproxima la solución de nuestro problema.

### 3.4.2. Método de resolución basado en Pseudo - Splines Cuárticos

Procedemos ahora a introducir y construir un nuevo método de resolución del P.V.I definido en (3.109), basado en este caso en la utilización de Pseudo-Splines  $S$  Cuárticos, al igual que ya hicimos en apartados anteriores para P.V.I. y para P.V.F. lineales.

En  $I_k = [x_k, x_{k+1}]$  el Spline  $S$  viene definido por  $S_k$  del siguiente modo:

$$S_k(x) = a_k + b_k(x - x_k) + c_k(x - x_k)^2 + d_k(x - x_k)^3 + e_k(x - x_k)^4 \quad (3.124)$$

y sus dos primeras derivadas por:

$$S'_k(x) = b_k + 2 c_k (x - x_k) + 3 d_k (x - x_k)^2 + 4 e_k (x - x_k)^3 \quad (3.125)$$

$$S''_k(x) = 2 c_k + 6 d_k (x - x_k) + 12 e_k (x - x_k)^2 \quad (3.126)$$

Por tanto, se tiene en el punto  $x_k$ :

$$\left. \begin{aligned} S_k(x_k) &= a_k \\ S'_k(x_k) &= b_k \\ S''_k(x_k) &= 2c_k \end{aligned} \right\} \quad (3.127)$$

y en el punto  $x_{k+1}$ :

$$\left. \begin{aligned} S_k(x_{k+1}) &= a_k + b_k h_k + c_k h_k^2 + d_k h_k^3 + e_k h_k^4 \\ S'_k(x_{k+1}) &= b_k + 2 c_k h_k + 3 d_k h_k^2 + 4 e_k h_k^3 \\ S''_k(x_{k+1}) &= 2 c_k + 6 d_k h_k + 12 e_k h_k^2 \end{aligned} \right\} \quad (3.128)$$

Y definiendo, como venimos haciendo:  $x_{k+\frac{1}{2}} = \frac{x_k+x_{k+1}}{2} = x_k + \frac{1}{2} h_k$ , para  $k = 1, 2, \dots, n$  se tiene también:

$$\left. \begin{aligned} S_k\left(x_{k+\frac{1}{2}}\right) &= a_k + \frac{1}{2} b_k h_k + \frac{1}{4} c_k h_k^2 + \frac{1}{8} d_k h_k^3 + \frac{1}{16} e_k h_k^4 \\ S'_k\left(x_{k+\frac{1}{2}}\right) &= b_k + c_k h_k + \frac{3}{4} d_k h_k^2 + \frac{1}{2} e_k h_k^3 \\ S''_k\left(x_{k+\frac{1}{2}}\right) &= 2 c_k + 3 d_k h_k + 3 e_k h_k^2 \end{aligned} \right\} \quad (3.129)$$

Por otra parte, para que se cumplan las condiciones de continuidad correspondientes a que  $S \in C^2[a, b]$ , han de verificarse:

$$\left. \begin{aligned} S_k(x_{k+1}) &= S_{k+1}(x_{k+1}) \\ S'_k(x_{k+1}) &= S'_{k+1}(x_{k+1}) \\ S''_k(x_{k+1}) &= S''_{k+1}(x_{k+1}) \end{aligned} \right\}, \quad k = 1, 2, \dots, n-1 \Rightarrow$$

$$\Rightarrow \left. \begin{aligned} a_{k+1} &= a_k + b_k h_k + c_k h_k^2 + d_k h_k^3 + e_k h_k^4 \\ b_{k+1} &= b_k + 2 c_k h_k + 3 d_k h_k^2 + 4 e_k h_k^3 \\ c_{k+1} &= c_k + 3 d_k h_k + 6 e_k h_k^2 \end{aligned} \right\} \quad (3.130)$$

Las condiciones de colocación las imponemos en los  $n+1$  nodos  $x_k$ , para  $k = 1, 2, \dots, n+1$  y en los  $n$  puntos medios de los intervalos, esto es:  $x_{k+\frac{1}{2}}$ , con  $k = 1, 2, \dots, n$ . Se tienen así las condiciones:

$$\left. \begin{aligned} S''_k(x_k) &= f(x_k, S_k(x_k), S'_k(x_k)), & k = 1, 2, \dots, n \\ S''_n(x_{n+1}) &= f(x_{n+1}, S_n(x_{n+1}), S'_n(x_{n+1})), & k = n+1 \\ S''_k\left(x_{k+\frac{1}{2}}\right) &= f\left(x_{k+\frac{1}{2}}, S_k\left(x_{k+\frac{1}{2}}\right), S'_k\left(x_{k+\frac{1}{2}}\right)\right), & k = 1, 2, \dots, n \end{aligned} \right\} \quad (3.131)$$

### Resolución:

Al igual que en el apartado anterior, vamos a comenzar resolviendo el problema en el intervalo inicial  $I_1 = [x_1, x_2]$ .

De las condiciones iniciales se deduce:

$$S(a) = z_1 \Rightarrow a_1 = z_1 \quad (3.132)$$

$$S'(a) = z_2 \Rightarrow b_1 = z_2 \quad (3.133)$$

La condición de colocación en  $x_1 = a$ , se tiene:

$$S''(a) = f(a, S(a), S'(a)) \Rightarrow c_1 = \frac{1}{2} f(a, z_1, z_2) \quad (3.134)$$

Los dos restantes condiciones de colocación en los puntos medio y final de  $I_1$  dan lugar a un sistema de dos ecuaciones no lineales (en general) en las dos incógnitas correspondientes a los parámetros  $d_1$  y  $e_1$ .

Concretamente, para la condición de colocación en  $x_{\frac{3}{2}}$ :

$$\begin{aligned}
 S_1''\left(x_{\frac{3}{2}}\right) &= f\left(x_{\frac{3}{2}}, S\left(x_{\frac{3}{2}}\right), S'\left(x_{\frac{3}{2}}\right)\right) \Rightarrow \\
 \Rightarrow 2 c_1 + 3 d_1 h_1 + 3 e_1 h_1^2 &= \\
 = f\left(x_{\frac{3}{2}}, a_1 + \frac{1}{2} b_1 h_1 + \frac{1}{4} c_1 h_1^2 + \frac{1}{8} d_1 h_1^3 + \frac{1}{16} e_1 h_1^4, b_1 + c_1 h_1 + \frac{3}{4} d_1 h_1^2 + \frac{1}{2} e_1 h_1^3\right) & \quad (3.135)
 \end{aligned}$$

y para la condición de colocación en  $x_2$ :

$$\begin{aligned}
 S_1''(x_2) &= f(x_2, S_1(x_2), S_1'(x_2)) \Rightarrow \\
 \Rightarrow 2 c_1 + 6 d_1 h_1 + 12 e_1 h_1^2 &= \\
 = f(x_2, a_1 + b_1 h_1 + c_1 h_1^2 + d_1 h_1^3 + e_1 h_1^4, b_1 + 2c_1 h_1 + 3d_1 h_1^2 + 4e_1 h_1^3) & \quad (3.136)
 \end{aligned}$$

Nótese que, en este punto, todas las cantidades involucradas en ambas ecuaciones se conocen a excepción de  $d_1$  y  $e_1$ . Basta pues resolver el sistema con cualquiera de los métodos disponibles para obtener  $d_1$  y  $e_1$ , de modo que  $S$  queda completamente determinado en  $I_1$ .

Tras esto, pasamos a describir, por un procedimiento inductivo, cómo determinar  $S$  en los restantes  $I_k = [x_k, x_{k+1}]$ , para  $k = 2, 3, \dots, n$ .

Conocidos  $a_{k-1}$ ,  $b_{k-1}$ ,  $c_{k-1}$ ,  $d_{k-1}$  y  $e_{k-1}$ , esto es, los coeficientes que definen  $S_{k-1}$  en  $I_{k-1}$ , se obtienen directamente de las condiciones de continuidad en el nodo  $x_k$  las relaciones:

$$\begin{cases}
 a_k = a_{k-1} + b_{k-1} h_{k-1} + c_{k-1} h_{k-1}^2 + d_{k-1} h_{k-1}^3 + e_{k-1} h_{k-1}^4 \\
 b_k = b_{k-1} + 2 c_{k-1} h_{k-1} + 3 d_{k-1} h_{k-1}^2 + 4 e_{k-1} h_{k-1}^3 \\
 c_k = c_{k-1} + 3 d_{k-1} h_{k-1} + 6 e_{k-1} h_{k-1}^2
 \end{cases} \quad (3.137)$$

que determinan los valores de  $a_k$ ,  $b_k$  y  $c_k$ .

Las dos condiciones de colocación en los puntos  $x_{k+\frac{1}{2}}$  y  $x_{k+1}$  dan lugar a un sistema de dos ecuaciones no lineales (en general) en las dos incógnitas  $d_k$  y  $e_k$ , que son los únicos parámetros que nos resta por determinar para tener  $S_k$  completamente descrito en el intervalo  $I_k$ .

Concretamente, la condición de colocación en el punto  $x_{k+\frac{1}{2}}$  toma la forma:

$$\begin{aligned} S_k''\left(x_{k+\frac{1}{2}}\right) &= f\left(x_{k+\frac{1}{2}}, S\left(x_{k+\frac{1}{2}}\right), S'\left(x_{k+\frac{1}{2}}\right)\right) \Rightarrow \\ \Rightarrow 2 c_k + 3 d_k h_k + 3 e_k h_k^2 &= \\ &= f\left(x_{k+\frac{1}{2}}, a_k + \frac{1}{2} b_k h_k + \frac{1}{4} c_k h_k^2 + \frac{1}{8} d_k h_k^3 + \frac{1}{16} e_k h_k^4, b_k \right. \quad (3.138) \\ &\quad \left. + c_k h_k + \frac{3}{4} d_k h_k^2 + \frac{1}{2} e_k h_k^3\right) \end{aligned}$$

mientras que la condición de colocación en  $x_2$  viene dada por:

$$\begin{aligned} S_k''(x_{k+1}) &= f(x_{k+1}, S(x_{k+1}), S'(x_{k+1})) \Rightarrow \\ \Rightarrow 2 c_k + 6 d_k h_k + 12 e_k h_k^2 &= \\ &= f(x_{k+1}, a_k + b_k h_k + c_k h_k^2 + d_k h_k^3 + e_k h_k^4, b_k + 2c_k h_k \quad (3.139) \\ &\quad + 3d_k h_k^2 + 4e_k h_k^3) \end{aligned}$$

Y basta resolver el sistema de dos ecuaciones no lineales (en general), para determinar  $d_k$  y  $e_k$ , con lo que  $S_k$  queda completamente determinado en  $I_k$ , para todo  $k = 1, 2, \dots, n$ , mediante el procedimiento inductivo descrito.





# MÉTODO DEL DISPARO

## Capítulo 4



## 4. MÉTODO DEL DISPARO

### 4.1. Introducción al Método del Disparo [3] [9] [18] [20] [27]

El método del Disparo es uno de los métodos numéricos clásicos más estudiados y utilizados para aproximar la solución de un Problema de Valor en la Frontera de segundo orden.

Este método, que se explica con amplio detalle en las referencias [9] [18] [20] y [27], se basa en resolver varios problemas de valores iniciales (P.V.I), para así encontrar, a partir de las aproximaciones obtenidas, la solución al problema planteado. Según el tipo de Problema de Valor en la Frontera considerado, se distingue entre:

- *Método del Disparo Lineal:*

En su formulación más sencilla, resuelve Ecuaciones Diferenciales Lineales de Segundo Orden, con condiciones en la frontera, del tipo:

$$\begin{cases} y'' = p(x) y' + q(x) y + r(x), & x \in [a, b] \\ y(a) = z_1 \\ y(b) = z_2 \end{cases} \quad (4.1)$$

El método está basado en la obtención de la solución del problema (4.1) a partir de una combinación lineal de dos soluciones de la misma E.D.O., pero con valores iniciales. Cuando no es posible o es muy costoso obtener la solución exacta de este par de Problemas de Valores Iniciales, se recurre al empleo de un método de resolución numérica que permita aproximar dichas soluciones. En este caso, la combinación lineal de las soluciones así obtenidas, permitirá aproximar la solución del problema original.

- *Método del Disparo No Lineal:*

En su versión más sencilla, resuelve, en general de forma aproximada, Ecuaciones Diferenciales de Segundo Orden No Lineales, con condiciones de frontera, del tipo:

$$\begin{cases} y'' = f(x, y, y'), & x \in [a, b] \\ y(a) = z_1 \\ y(b) = z_2 \end{cases} \quad (4.2)$$

Aun pareciéndose al Método del Disparo Lineal, la solución del problema, en este caso, no se expresa como una combinación lineal de las soluciones a dos Problemas de Valores Iniciales. En su lugar, se construye una sucesión de Problemas de Valores Iniciales que comparten la misma E.D.O. del problema original, cuyas soluciones converjan a la solución buscada. La sucesión puede obtenerse de muy diversas formas, siendo el método de Newton uno de los más utilizados en su implementación, en combinación con un método de integración numérica que permita aproximar las soluciones de los Problemas de Valores Iniciales asociados.

En los apartados siguientes vamos a explicar con más detalle los fundamentos teóricos de estos dos tipos de Métodos del Disparo.

Previamente, es necesario introducir el siguiente teorema que establece las condiciones generales que garantizan que exista solución a un P.V.F de segundo orden y, además, que dicha solución sea única.

**Teorema 4.1.-** Suponemos que la función  $f$  en el problema con valores en la frontera del tipo (4.2), es continua en el conjunto:

$$D = \{(x, y, y') \mid a \leq x \leq b, \quad -\infty \leq y \leq \infty, \quad -\infty \leq y' \leq \infty\}$$

y que  $\frac{\partial f}{\partial y}$ , y  $\frac{\partial f}{\partial y'}$  también son continuas en  $D$ . Si, además:

- i.  $\frac{\partial f}{\partial y}(x, y, y') > 0$  para toda  $(x, y, y') \in D$ , y
- ii. existe una constante  $M$ , con  $\left| \frac{\partial f}{\partial y'}(x, y, y') \right| \leq M$ , para toda  $(x, y, y') \in D$

entonces el Problema con Valores en la Frontera tiene una solución única.

Como consecuencia del teorema anterior, se puede deducir con facilidad el siguiente resultado:

**Corolario 4.1.-** Si el Problema lineal con Valores en la Frontera:

$$y'' = p(x) y' + q(x) y + r(x), \quad x \in [a, b], \quad y(a) = z_1, \quad y(b) = z_2$$

satisface:

- i.  $p(x)$ ,  $q(x)$  y  $r(x)$  son continuas en  $[a, b]$ , y
- ii.  $q(x) > 0$  en  $[a, b]$ .

entonces el problema con valores en la frontera tiene solución única.

## 4.2. Método del Disparo Lineal [3] [9] [18] [20] [27]

Este planteamiento, como ya hemos indicado, corresponde al caso en que la Ecuación Diferencial Ordinaria de Segundo Orden que modela el Problema de Valores de Frontera es lineal:  $y'' = p(x) y' + q(x) y + r(x)$ .

### 4.2.1. Problemas con condiciones de contorno simples

Para aproximar la solución única del problema lineal del tipo (4.1), que está garantizada por el cumplimiento de las hipótesis del Corolario 4.1, primero hay que considerar los siguientes Problemas de Valores Iniciales:

$$\begin{cases} y_1'' = p(x) \cdot y_1' + q(x) \cdot y_1 + r(x), & x \in [a, b] \\ y_1(a) = z_1 \\ y_1'(a) = 0 \end{cases} \quad (4.3)$$

$$\begin{cases} y_2'' = p(x) \cdot y_2' + q(x) \cdot y_2, & x \in [a, b] \\ y_2(a) = 0 \\ y_2'(a) = 1 \end{cases} \quad (4.4)$$

Dado que, tanto (4.3) como (4.4) cumplen las hipótesis del Corolario 4.1, la solución del P.V.F (4.1) puede obtenerse tomando:

$$y(x) = y_1(x) + \left( \frac{z_2 - y_1(b)}{y_2(b)} \right) \cdot y_2(x) \quad (4.5)$$

siendo  $y_1(x)$ ,  $y_2(x)$  las soluciones de los problemas (4.3) y (4.4) respectivamente.

A partir de la expresión (4.5), se pueden obtener las derivadas primera y segunda de la solución de nuestro problema, que vienen dadas por:

$$\begin{cases} y'(x) = y_1'(x) + \left( \frac{z_2 - y_1(b)}{y_2(b)} \right) y_2'(x) \\ y''(x) = y_1''(x) + \left( \frac{z_2 - y_1(b)}{y_2(b)} \right) y_2''(x) \end{cases} \quad (4.6)$$

Se tiene así que:

$$y''(x) = p(x)y'(x) + q(x)y(x) + r(x) \Rightarrow$$

que al aplicar las relaciones dadas en (4.6) resulta:

$$\begin{aligned} \Rightarrow y''(x) &= p(x) \left( y_1'(x) + \left( \frac{z_2 - y_1(b)}{y_2(b)} \right) y_2'(x) \right) + \\ &+ q(x) \left( y_1(x) + \left( \frac{z_2 - y_1(b)}{y_2(b)} \right) y_2(x) \right) + r(x) \Rightarrow \\ \Rightarrow y''(x) &= p(x) y_1'(x) + q(x) y_1(x) + r(x) + \\ &+ \left( \frac{z_2 - y_1(b)}{y_2(b)} \right) (p(x) y_2'(x) + q(x) y_2(x)) \end{aligned} \quad (4.7)$$

y es trivial ver que la solución propuesta satisface la E.D.O.

Además, la solución de (4.5) satisface las condiciones de frontera, puesto que se cumple:

$$\begin{cases} y(a) = y_1(a) + \left( \frac{z_2 - y_1(b)}{y_2(b)} \right) \cdot y_2(a) = z_1 + \left( \frac{z_2 - y_1(b)}{y_2(b)} \right) \cdot 0 = z_1 \\ y(b) = y_1(b) + \left( \frac{z_2 - y_1(b)}{y_2(b)} \right) \cdot y_2(b) = y_1(b) + z_2 - y_1(b) = z_2 \end{cases} \quad (4.8)$$

Por tanto,  $y(x)$  es la solución única de nuestro P.V.F (4.1), sujeta a la condición de que  $y_2(b) \neq 0$ . Veremos enseguida que, en el caso de que sea  $y_2(b) = 0$ , se tiene que  $y_2(x) = 0$ ,  $x \in [a, b]$ , con lo que bastaría tomar  $y(x) = y_1(x)$ .

Como se puede comprobar, el Método del Disparo para ecuaciones lineales está basado en la sustitución del P.V.F. por dos P.V.I. Para aproximar las soluciones  $y_1(x)$ ,  $y_2(x)$  de los dos P.V.I formulados en términos de Ecuaciones Diferenciales Lineales, se puede recurrir, por ejemplo, a cualquiera de los métodos que se presentaron en el Capítulo 2. Una vez que se cuenta con estas aproximaciones, la solución del P.V.F. se aproxima recurriendo a la relación (4.5).

Desde el punto de vista gráfico, el método tiene el aspecto que se observa en la figura 4.1.

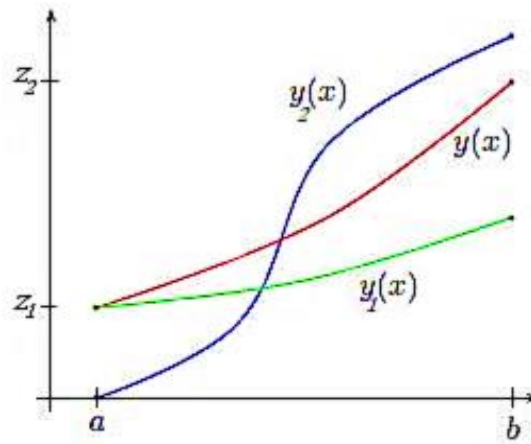


Figura 4.1. Método del Disparo para resolución de P.V.F lineales

Como hemos mencionado antes, solo quedaría estudiar el caso particular en que  $y_2(b) = 0$ , para el que la solución (4.5), no está definida. En este caso, podemos definir una tercera solución de la ecuación homogénea a partir de otro par de condiciones iniciales:

$$\begin{cases} y_3'' = p(x) y_3' + q(x) y_3 \\ y_3(a) = 1 \\ y_3'(a) = 0 \end{cases} \quad (4.9)$$

Por la teoría de Ecuaciones Diferenciales Ordinarias, la solución existe, es única y se puede escribir como la suma de una solución particular de la ecuación no homogénea y una combinación lineal de dos soluciones (independientes) de la homogénea, es decir:

$$y(x) = y_1(x) + c_2 y_2(x) + c_3 y_3(x) \quad (4.10)$$

Por otra parte, de las condiciones de contorno tenemos:

$$y(a) = z_1 = y_1(a) + c_2 y_2(a) + c_3 y_3(a) = z_1 + c_3 \quad (4.11)$$

por lo que  $c_3 = 0$ .

Además, de la condición de contorno en  $b$  y de la suposición  $y_2(b) = 0$ , se deduce:

$$y(b) = z_2 = y_1(b) + c_2 y_2(b) + c_3 y_3(b) = y_1(b) \quad (4.12)$$

por lo que se cumple:  $y_1(a) = z_1$ ,  $y_1(b) = z_2$ .

Como  $y_1(x)$  es una solución que satisface las condiciones de contorno, y además es única,  $y_2(x)$  debe ser idénticamente nula. Por lo tanto, en este caso, la solución del problema es  $y_1(x)$ .

Se presenta a continuación un ejemplo de algoritmo del método del disparo lineal, para el caso de nodos equiespaciados.

### *Algoritmo del método del Disparo para problemas lineales*

ENTRADA: extremos  $a, b$ ; condiciones de frontera  $z_1, z_2$ ; número de subintervalos  $N$ .

SALIDA: aproximaciones  $w_{1,i}$  a  $y(x_i)$ ;  $w_{2,i}$  a  $y'(x_i)$ ; para cada  $i = 0, \dots, N$ .

1. Tomar  $h = \frac{b-a}{N}$ ;  $y_{1,0} = z_1$ ;  $y_{1,2,0} = 0$ ;  $y_{2,1,0} = 0$ ;  $y_{2,2,0} = 1$ .
2. Para  $i = 1, \dots, N - 1$ , hacer los pasos 3 y 4.
3. Tomar  $x = a + ih$ .
4. Aplicar Runge - Kutta para obtener  $y_{1,1,i+1}$ ,  $y_{1,2,i+1}$ ,  $y_{2,1,i+1}$ ,  $y_{2,2,i+1}$ .
5. Tomar  $w_{1,0} = z_1$ ;  $w_{2,0} = \frac{z_2 - u_{1,N}}{v_{1,N}}$ . Mostrar  $(a, w_{1,0}, w_{2,0})$ .
6. Para  $i = 1, \dots, N$ , tomar

$$w_{1,i} = u_{1,i} + w_{2,0} \cdot v_{1,i}, \quad w_{2,i} = u_{2,i} + w_{2,0} \cdot v_{2,i}, \quad x = a + ih$$

7. Mostrar  $x, w_{1,i}, w_{2,i}$ .
8. Parar.

En el paso 4 del algoritmo anterior se recurre al método Runge-Kutta de cuarto orden, para obtener las aproximaciones a las soluciones de los P.V.I  $y_1(x)$  y  $y_2(x)$ , pero este método puede sustituirse por cualquiera de los que ya se han presentado en los capítulos 2 y 3.

Este algoritmo permite obtener aproximaciones a la solución del P.V.F. y a su primera derivada. Además, su uso no se limita a problemas que verifiquen el Corolario 4.1, sino que también proporciona resultados satisfactorios en muchos problemas que no cumplen esas hipótesis.



### Posibles errores por redondeo

A pesar de que este método es bastante exacto, en ciertos casos pueden producirse errores de redondeo que perjudican la precisión de las soluciones obtenidas.

Por ejemplo, cuando la aproximación a la solución  $y_1(x)$  crece rápidamente cuando  $x$  recorre el intervalo  $[a, b]$ , entonces el valor de  $y_1$  es grande; si además,  $z_2$  es pequeño en comparación con  $y_{11,N}$ , entonces:

$$y_{12,0} = \frac{z_2 - y_{11,N}}{y_{21,N}} \cong -\frac{y_{11,N}}{y_{21,N}} \quad (4.13)$$

de modo que:

$$\left. \begin{aligned} w_{1,i} &= y_{11,i} + w_{2,0} \cdot y_{21,i} \cong y_{11,i} - \frac{y_{11,N}}{y_{21,N}} y_{21,i} \\ w_{2,i} &= y_{12,i} + w_{2,0} \cdot y_{22,i} \cong y_{12,i} - \frac{y_{12,N}}{y_{22,N}} y_{22,i} \end{aligned} \right\} \quad (4.14)$$

Como consecuencia de lo anterior puede producirse una pérdida de dígitos significativos por cancelación. Pero al ser  $y_{11,i}$  una aproximación a  $y_1(x_i)$ , es posible vigilar el comportamiento de  $y_1(x)$  para tratar de prevenir situaciones de este tipo.

Además, si  $y_{11,i}$  aumenta rápidamente al pasar de  $a$  a  $b$ , puede aplicarse el método del disparo inverso, es decir, hacia atrás, resolviendo en este caso los siguientes P.V.I:

$$\begin{cases} y_1'' = p(x) \cdot y_1' + q(x) \cdot y_1 + r(x), & x \in [a, b] \\ y_1(b) = z_1 \\ y_1'(b) = 0 \end{cases} \quad (4.15)$$

y

$$\begin{cases} y_2'' = p(x) \cdot y_2' + q(x) \cdot y_2, & x \in [a, b] \\ y_2(b) = 0 \\ y_2'(b) = 1 \end{cases} \quad (4.16)$$

Si este Método del Disparo Inverso todavía presenta eliminación de dígitos significativos y si aún con el aumento de precisión no se logra mayor exactitud, será necesario recurrir a otros métodos numéricos como los que se presentaron en el capítulo anterior basados en la utilización de Splines.

#### 4.2.2. Problemas con condiciones de contorno generales

El método que acabamos de explicar en el apartado anterior puede generalizarse para P.V.F. cuyas condiciones de contorno son más generales:

$$\begin{cases} y'' = p(x) \cdot y' + q(x) \cdot y + r(x), & x \in [a, b] \\ z_{11} y(a) + z_{12} y'(a) = z_{13} \\ z_{21} y(b) + z_{22} y'(b) = z_{23} \end{cases} \quad (4.17)$$

En este caso, escribimos la solución del problema (4.17) anterior como la siguiente combinación lineal:

$$y(x) = y_1(x) + d_2 y_2(x) + d_3 y_3(x) \quad (4.18)$$

donde  $y_1(x)$ ,  $y_2(x)$  e  $y_3(x)$  son las aproximaciones a las soluciones de los problemas de valores iniciales intermedios, que definimos a continuación.

Los P.V.I mencionados que nos permiten obtener la solución a nuestro P.V.F. de condiciones generales son:

$$\begin{cases} y_1'' = p(x) y_1' + q(x) y_1 + r(x) \\ y_1(a) = 0 \\ y_1'(a) = 0 \end{cases} \quad (4.19)$$

$$\begin{cases} y_2'' = p(x) y_2' + q(x) y_2 \\ y_2(a) = 0 \\ y_2'(a) = 1 \end{cases} \quad (4.20)$$

$$\begin{cases} y_3'' = p(x) y_3' + q(x) y_3 \\ y_3(a) = 1 \\ y_3'(a) = 0 \end{cases} \quad (4.21)$$

Al sustituir en las ecuaciones de las condiciones de contorno generales, resulta el sistema de ecuaciones:

$$\left. \begin{aligned} z_{11} d_3 + z_{12} d_2 &= z_{13} \\ z_{21}[y_1(b) + d_2 y_2(b) + d_3 y_3(b)] + z_{22}[y_1'(b) + d_2 y_2'(b) + d_3 y_3'(b)] &= z_{23} \end{aligned} \right\}$$

que se puede reescribir como:

$$\left. \begin{aligned} z_{12} d_2 + z_{11} d_3 &= z_{13} \\ [z_{21}y_2(b) + z_{22}y_2'(b)]d_2 + [z_{21}y_3(b) + z_{22}y_3'(b)]d_3 &= \\ &= z_{23} - [z_{21}y_1(b) + z_{22}y_1'(b)] \end{aligned} \right\} \quad (4.22)$$

Resolviendo el sistema anterior (4.22) para  $d_2$  y  $d_3$  a partir de los valores de las tres soluciones de los Problemas de Valores Iniciales y sus derivadas primeras en  $b$ :

$$\begin{aligned} d_2 &= \frac{z_{13} [z_{21} y_3(b) + z_{22} y_3'(b)] - [z_{23} - (z_{21} y_1(b) + z_{22} y_1'(b))] z_{11}}{z_{12} [z_{21} y_3(b) + z_{22} y_3'(b)] - z_{11} [z_{21} y_2(b) + z_{22} y_2'(b)]} \end{aligned} \quad (4.23)$$

$$\begin{aligned} d_3 &= \frac{-z_{13} [z_{21} y_2(b) + z_{22} y_2'(b)] + [z_{23} - (z_{21} y_1(b) + z_{22} y_1'(b))] z_{12}}{z_{12} [z_{21} y_3(b) + z_{22} y_3'(b)] - z_{11} [z_{21} y_2(b) + z_{22} y_2'(b)]} \end{aligned} \quad (4.24)$$

### 4.3. Método del Disparo No Lineal [3] [9] [18] [20] [27]

Los problemas que aparecen en Ciencias, Ingeniería e Industria, son con frecuencia no lineales. En este apartado vamos a ver como implementamos el método del disparo para problemas de ecuaciones diferenciales de segundo orden no lineales con condiciones de contorno sencillas. Es decir, problemas del tipo:

$$\left. \begin{aligned} y'' &= f(x, y, y') \\ y(a) &= z_1 \\ y(b) &= z_2 \end{aligned} \right\}, \quad x \in [a, b] \quad (4.25)$$

Como ya hemos comentado, la solución en el caso no lineal no puede expresarse como combinación lineal de las dos soluciones de dos Problemas de Valores Iniciales.

Buscaremos la solución construyendo una sucesión de soluciones a problemas de valores iniciales que involucran un parámetro  $t$ , que converja a la solución buscada.

Concretamente, resolveremos, de manera exacta o aproximada, problemas del tipo:

$$\left. \begin{aligned} y'' &= f(x, y, y') \\ y(a) &= z_1 \\ y'(a) &= t \end{aligned} \right\} \quad (4.26)$$

en el intervalo  $[a, b]$ , para una sucesión de valores de  $t$  que denotaremos por  $(t_k)_{k=0}^{\alpha}$  y de tal modo que:

$$\lim_{k \rightarrow \infty} y(b, t_k) = y(b) = z_2 \quad (4.27)$$

donde con  $y(x, t_k)$  denotaremos la solución al problema (4.26) con  $t = t_k$  y con  $y(x)$  la solución del problema original.

Esta técnica se conoce como “Método del Disparo No Lineal” porque podría interpretarse como una sucesión de disparos desde el punto  $(a, z_1)$  con ángulos de disparo dados en términos de los  $t_k$  y buscando dar en el blanco situado en el punto  $(b, z_2)$ .

Lo explicado en el párrafo anterior se representa en la siguiente imagen:

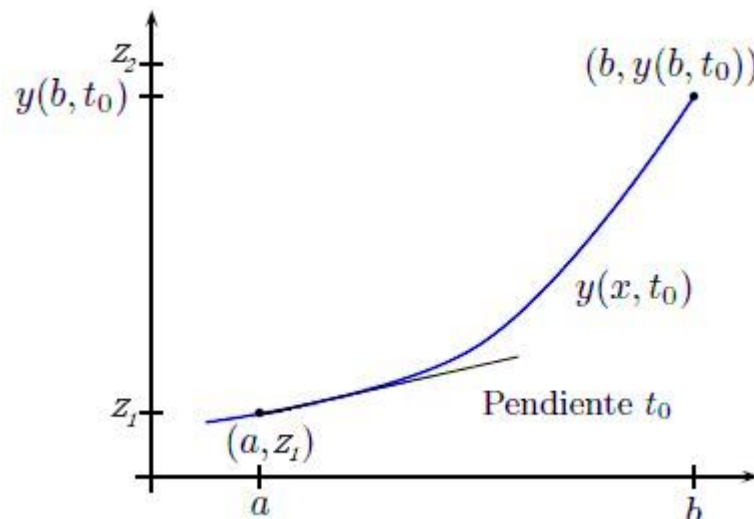


Figura 4.2. Método del Disparo para resolución de P.V.F no lineales

En la situación anterior, empezaríamos realizando un primer disparo con un ángulo dado en términos de  $t_0$ , cuya trayectoria vendría dada por la solución  $y(x, t_0)$  al problema:

$$\left. \begin{aligned} y'' &= f(x, y, y') \\ y(a) &= z_1 \\ y'(a) &= t_0 \end{aligned} \right\} \quad (4.28)$$

en el intervalo  $[a, b]$ .

Si  $y(b, t_0)$  no está lo suficientemente cerca de  $z_2$ , realizaríamos un segundo disparo con ángulo de disparo  $t_1$  y repetiríamos el proceso construyendo así una sucesión  $(t_k)_k$  con soluciones asociadas  $y(x, t_k)$  hasta obtener la solución buscada o una aproximación suficientemente buena.

Obviamente hay que construir la sucesión de valores  $(t_k)_k$  de manera adecuada para aspirar a que:

$$\lim_{k \rightarrow \infty} y(x, t_k) = y(x) \quad (4.29)$$

o equivalentemente:

$$\lim_{k \rightarrow \infty} y(b, t_k) = z_2 \quad (4.30)$$

Gráficamente, el procedimiento anterior:

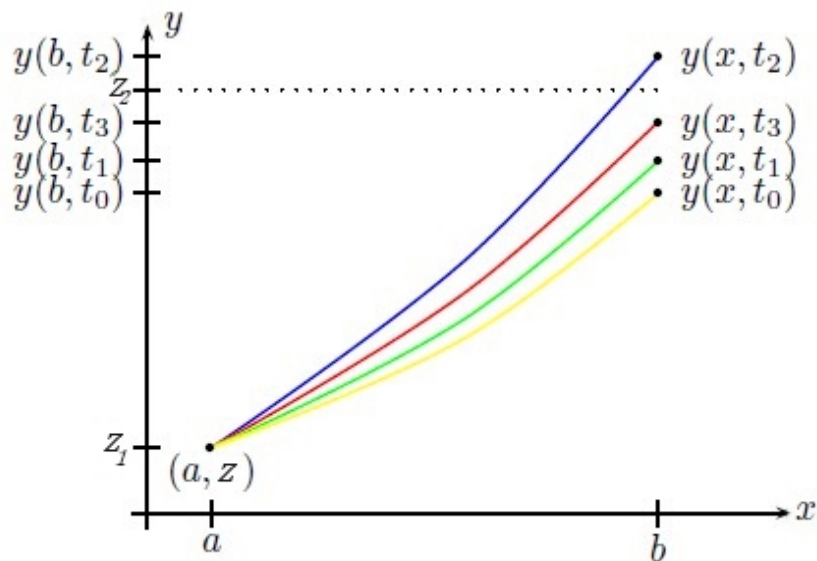


Figura 4.3. Selección de elevaciones  $t_k$  para alcanzar al valor final  $z_2$

Observamos que en realidad estamos buscando la solución del problema no lineal (en general):

$$y(b, t) - z_2 = 0 \quad (4.31)$$

Esta ecuación no lineal en  $t$  puede resolverse con cualquiera de los métodos de resolución conocidos: método de bisección, método de la secante, método de Newton, etc.

Si aplicamos el método de la secante para resolver el problema, necesitamos partir de dos valores iniciales de  $t$ :  $t_0$  y  $t_1$ . Los restantes valores de la sucesión  $(t_k)_k$  se obtienen mediante:

$$t_k = t_{k-1} - \frac{(t_{k-1} - t_{k-2})(y(b, t_{k-1}) - z_2)}{y(b, t_{k-1}) - y(b, t_{k-2})}, \quad k = 2, 3, \dots \quad (4.32)$$

Nótese que la sucesión así construida podría no converger o hacerlo a un valor no deseado de  $t$  para el que  $y(x, t) \neq y(x)$ . De hecho, la iteración anterior puede fracasar si para algún  $k$  se anula el denominador involucrado en el segundo miembro de la expresión anterior.

Es bien conocido que, en general, cuando converge, el método de la secante lo hace con orden  $\frac{1+\sqrt{5}}{2} \cong 1,62$ .

Si aplicamos el método de Newton para resolver el problema, bastará tomar un único valor inicial de  $t$ :  $t_0$  y generar, a partir de él, una sucesión  $(t_k)_k$  definida por:

$$t_k = t_{k-1} - \frac{y(b, t_{k-1}) - z_2}{\frac{dy}{dt}(b, t_{k-1})}, \quad k = 1, 2, \dots \quad (4.33)$$

De nuevo, la sucesión generada de este modo podría no converger o hacerlo a un valor distinto del deseado. De hecho, la sucesión podría dejar de estar definida para algún valor de  $k$  si se anula el denominador involucrado en la expresión que define el método.

Pese a estas dificultades que pueden surgir, son frecuentes los casos en los que el método converge, al menos localmente, partiendo de un iterante inicial  $t_0$  adecuado. De hecho, en muchos de estos casos la convergencia del método es cuadrática (orden 2).

La aplicación del método de Newton presenta la siguiente dificultad: al no disponer de una representación explícita de  $y(b, t)$ , no es directa la obtención de  $\frac{dy}{dt}(b, t_{k-1})$ .

Para salvar esta dificultad, podemos observar que, para cada  $t$ , se tiene que  $y(x, t)$  es solución del problema:

$$\begin{cases} y''(x, t) = f(x, y(x, t), y'(x, t)) \\ y(a, t) = z_1 \\ y'(a, t) = t \end{cases}, \quad x \in [a, b] \quad (4.34)$$

Derivando respecto de  $t$ , se tiene:

$$\frac{\partial y''}{\partial t}(x, t) = \frac{\partial f}{\partial t}(x, y(x, t), y'(x, t)) \quad (4.35)$$

Dado que  $x$  y  $t$  son independientes, se tiene que  $\frac{\partial x}{\partial t} = 0$  y aplicando la regla de la cadena se tiene para  $x \in [a, b]$ , que:

$$\begin{aligned} \frac{\partial f}{\partial t}(x, y(x, t), y'(x, t)) &= \\ &= \frac{\partial f}{\partial y}(x, y(x, t), y'(x, t)) \frac{\partial y}{\partial t}(x, t) + \frac{\partial f}{\partial y'}(x, y(x, t), y'(x, t)) \frac{\partial y'}{\partial t}(x, t) \end{aligned} \quad (4.36)$$

De las condiciones iniciales se deduce:

$$\frac{\partial y}{\partial t}(a, t) = 0 \quad (4.37)$$

$$\frac{\partial y'}{\partial t}(a, t) = 1 \quad (4.38)$$

Suponiendo que el orden de derivación de  $x$  y  $t$  puede invertirse, e introduciendo la notación  $v(x, t) = \frac{\partial y}{\partial t}(x, t)$ , el Problema de Valores Iniciales, para todo  $x \in [a, b]$ , toma la forma:

$$\begin{cases} v''(x, t) = \frac{\partial f}{\partial y}(x, y(x, t), y'(x, t)) v(x, t) + \frac{\partial f}{\partial y'}(x, y(x, t), y'(x, t)) v'(x, t) \\ v(a, t) = 0 \\ v'(a, t) = 1 \end{cases} \quad (4.39)$$

Por tanto, cada iteración con el método de Newton requerirá de la resolución, exacta o aproximada, de dos P.V.I. dados por los sistemas (4.34) y (4.39) con  $t = t_{k-1}$  (para el  $k$ -ésimo iterante que proporcionará  $t_k$ ). A partir de las dos soluciones obtenidas se tendrá:

$$t = t_{k-1} - \frac{y(b, t_{k-1}) - z_2}{v(b, t_{k-1})}, \quad k = 1, 2, \dots \quad (4.40)$$

En nuestro caso, buscaremos aproximar las soluciones de (4.34) y (4.39) mediante Splines, ya sean Cúbicos, lo que se implementa como *spline3pcnl* o Cuárticos, dando lugar a la función *spline4pcnl*.

A mayores, cabe destacar que, ligeros cambios en el procedimiento anterior permiten resolver Problemas de Valores de Frontera del tipo del (4.25) pero en los cuales alguna de sus condiciones de frontera se encuentra expresada mediante una derivada de la función. Es decir, problemas como:

$$\left. \begin{array}{l} y'' = f(x, y, y') \\ y(a) = z_1 \\ y'(b) = z_2 \end{array} \right\}, \quad x \in [a, b] \quad (4.41)$$

o bien:

$$\left. \begin{array}{l} y'' = f(x, y, y') \\ y'(a) = z_1 \\ y(b) = z_2 \end{array} \right\}, \quad x \in [a, b] \quad (4.42)$$

Dado que los cambios son sencillos, no vamos a entrar en más detalle en este capítulo. No obstante, pueden apreciarse los procedimientos que deben llevarse a cabo en estos casos en los listados de los programas de Matlab que corresponden a las funciones en los cuales se han implementado:

- Para problemas del tipo del (4.41) con derivada en el extremo final del intervalo, encontramos los métodos *spline3pcnl2* y *spline4pcnl2*, en los apartados 7.2.2.C y 7.2.2.D del Anexo respectivamente.
- Para problemas del tipo del (4.42) con derivada en el extremo inicial, se presentan los métodos *spline3pcnl3* y *spline4pcnl3*, en los mismos apartados que hemos mencionado.



Las principales diferencias que encontramos entre estos métodos y los métodos que se han desarrollado en este apartado (*spline3pcnl* y *spline4pcnl*) y que resuelven problemas cuyas condiciones no están dadas en términos de derivadas, sino en evaluaciones de la propia función, son:

- Entre *spline3pcnl* y *spline3pcnl2*, y análogamente entre *spline4pcnl* y *spline4pcnl2*:

El método *spline3pcnl2* evalúa la derivada primera del spline en  $x = b$  en vez de evaluar el Spline (sin derivar) en este mismo punto  $x = b$ , esto es, el método de Newton se aplica a la derivada en vez de a la propia función.

- Entre *spline3pcnl* y *spline3pcnl2*, y análogamente entre *spline4pcnl* y *spline4pcnl2*:

Se intercambian los dos últimos argumentos de las llamadas a las funciones *spline3p* y *spline3pnl* como puede apreciarse mejor en los códigos de los Anexos.





# EXPERIMENTOS NUMÉRICOS

## Capítulo 5



## 5. EXPERIMENTOS NUMÉRICOS

### 5.1. Introducción <sup>[15]</sup>

Como ya se ha dicho, los problemas que involucran Ecuaciones Diferenciales de Segundo Orden son habituales en todas las ramas de las Ciencias y de la Ingeniería.

Por ejemplo, en Biología aparecen en el modelado de dinámica de poblaciones; en Física, las leyes de Newton y muchas otras se expresan como problemas asociados a este tipo de ecuaciones; en Ingeniería Química surgen en la evaluación de las concentraciones de diversos reactivos durante una reacción; en Ingeniería Civil, aparecen a menudo en fenómenos de deflexión de barras y vigas; etc.

En la mayoría de los casos, las ecuaciones diferenciales que surgen de estos modelos no pueden resolverse analíticamente.

Debido a lo anterior, resulta necesario usar métodos numéricos para dar aproximaciones, suficientemente buenas, a la solución de dichos modelos. En ese sentido, se han introducido, en los Capítulos 3 y 4 del presente proyecto, varios métodos numéricos, algunos novedosos, que permiten obtener buenas aproximaciones a partir de la Teoría de Splines, utilizando el Método de Colocación y el Método del Disparo.

Someteremos a continuación estos métodos a un buen número de pruebas numéricas, con el fin de poner de manifiesto su precisión, orden de convergencia y adecuación a la resolución de problemas realistas que surgen con frecuencia en las aplicaciones.

Los problemas están extraídos de la literatura científica, con especial énfasis en aplicaciones a la Ingeniería, e incluyen algunos ejemplos difíciles de resolver con otros métodos numéricos de los que se aplican habitualmente en estas situaciones.

Vamos a recordar a continuación los distintos métodos introducidos en capítulos precedentes, atendiendo al tipo de problemas que resuelven. Para ello se presenta, en primer lugar, una tabla resumen (véase Tabla 5.1), y posteriormente una explicación breve del contenido de la misma para facilitar su entendimiento.

Se resume lo anterior en la siguiente tabla:

Tabla 5.1. Tabla resumen de los métodos desarrollados, sus características y aplicaciones

Tipo de PROBLEMA	Tipo de CONDICIONES	FUNDAMENTO en que se basa el método	Apartado de su DESARROLLO TEÓRICO	FUNCIÓN que lo implementa	Apartado de su CÓDIGO DE MATLAB
P.V.I. Lineal	Simples	Splines Cúbicos	3.2.1 y 3.2.2	spline3p	7.1.1.A
		Pseudo-Splines Cuárticos	3.2.4	spline4p	7.1.1.B
P.V.I. No Lineal	Simples	Splines Cúbicos	3.4.1	spline3pnl	7.1.3.A
		Pseudo-Splines Cuárticos	3.4.2	spline4pnl	7.1.3.B
P.V.F. Lineal	Generales	Splines Cúbicos y Algoritmo de Thomas	3.3.1	spline3pc2	7.1.2.A
		Pseudo-Splines Cuárticos y Algoritmo de Thomas	3.3.2	spline4pc2	7.1.2.B
		Splines Cúbicos y Método del Disparo Lineal	4.2.2	spline3pc	7.2.1.A
		Pseudo-Splines Cuárticos y Método del Disparo Lineal	4.2.2	spline4pc	7.2.1.B
P.V.F. No Lineal	Generales	Splines Cúbicos y Método del Disparo No Lineal	4.3	spline3pcnl	7.2.2.A
		Pseudo-Splines Cuárticos y Método del Disparo No Lineal	4.3	spline4pcnl	7.2.2.B
	Especiales	Splines Cúbicos y Método del Disparo No Lineal	4.3	spline3pcnl2 spline3pcnl3	7.2.2.C
		Pseudo-Splines Cuárticos y Método del Disparo No Lineal	4.3	spline4pcnl2 spline4pcnl3	7.2.2.D

- Para Problemas de Valores Iniciales Lineales:

Se han introducido dos métodos principales para este tipo de P.V.I. lineales con condiciones iniciales:

- A. En primer lugar, un método de colocación basado en Splines Cúbicos, que, aun no siendo nuevo, ha sido implementado de forma original y eficiente desde el punto de vista computacional. Fue introducido y explicado de manera teórica en 3.2.1 y 3.2.2.
- B. Basado también en el método de colocación, se presenta un método nuevo, construido a partir de Pseudo-Splines Cuárticos y desarrollado en el apartado 3.2.4.

Estos son los dos métodos que se van a utilizar en los experimentos numéricos para problemas de valores iniciales lineales. El código de las funciones, que se han implementado utilizando el software Matlab lo incluimos en el apartado 7.1.1 de los Anexos.

Por otra parte, a lo largo del Capítulo 3 se han introducido otros dos posibles métodos de resolución de este tipo de problemas: un Método de Resolución Inverso basado en Splines cúbicos y explicado en el apartado 3.2.3, que puede aplicarse en el caso de que las condiciones iniciales estén dadas en el punto final  $b$  del intervalo; y otro método, basado en Pseudo-Splines Cuárticos, que permite resolver P.V.I. lineales utilizando la misma base introducida para P.V.F. (véase el apartado 3.3.2 A).

- Para Problemas de Valores Iniciales No Lineales:

Siguiendo el mismo esquema que en los problemas del caso anterior, se han presentado para P.V.I. No Lineales con condiciones iniciales dos nuevos métodos:

- A. Un método de colocación que recurre al uso de Splines Cúbicos y cuya explicación teórica se puede encontrar en el apartado 3.4.1.
- B. Otro método de colocación basado en Pseudo-Splines Cuárticos, introducido y descrito en el apartado 3.4.2 de esta memoria.

Puede encontrarse en el apartado 7.1.3 de los Anexos el código de las funciones que implementan estos métodos.

- Para Problemas de Valores de Frontera Lineales:

Para el caso de P.V.F. lineales con Condiciones de contorno generales, se han explicado los siguientes métodos:

- A. Un primer método de colocación basado en Splines Cúbicos, que se apoya en el Algoritmo de Thomas para la resolución del sistema lineal tridiagonal de ecuaciones resultante, que permite obtener el Spline. La teoría de este método está recogida en el apartado 3.3.1.
- B. Un segundo método de colocación que recurre, como el método anterior, al Algoritmo de Thomas, pero aplicando Pseudo-Splines Cuárticos. Su descripción y construcción se puede encontrar en el apartado 3.3.2.

El código de las funciones de estos dos métodos se incluye en el apartado 7.1.2 de los Anexos.

- C. Un tercer método de colocación, cuyo fundamento es el Método del Disparo Lineal, y que se apoya en la construcción de varios Splines Cúbicos para los P.V.I. asociados. La explicación teórica se recoge en el apartado 4.2.2.
- D. Un último método de colocación, que también se basa en el Método del Disparo Lineal, pero que recurre a métodos de Pseudo-Splines Cuárticos para resolver los P.V.I. intermedios. Este método ha sido desarrollado en el apartado 4.2.2.

Finalmente, la implementación de estos dos métodos alternativos en Matlab se presenta en las funciones del apartado 7.2.1 de los Anexos.

- Para Problemas de Valores de Frontera No Lineales:

Por último, los métodos numéricos para P.V.F. No lineales con condiciones simples, están todos ellos fundamentados en el Método del Disparo No Lineal, puesto que para este tipo de problemas resulta muy complejo y costoso resolverlos directamente recurriendo exclusivamente a Splines.

Además, en todos los casos, se ha recurrido al método de Newton para realizar las iteraciones que se necesitan para resolver las ecuaciones no lineales resultantes de la aplicación del método.



- A. Un primer método que, aparte de lo explicado en el párrafo anterior, recurre a los métodos de colocación basados en Splines Cúbicos para resolver los problemas intermedios.
- B. Otro método que sigue los mismos principios, pero que resuelve los P.V.I. intermedios aplicando los métodos desarrollados mediante Pseudo- Splines Cuárticos.

Estos dos tipos de problemas siguen el mismo desarrollo teórico, el cual puede encontrarse en el apartado 4.3.1.

Por otra parte, se presentan otros dos métodos para el caso de que el P.V.F. no lineal esté acompañado de condiciones de contorno especiales, en las cuales una de las condiciones de contorno (en el punto inicial o final del intervalo) viene dada en términos de una derivada. Estos métodos se exponen de manera teórica en el apartado 4.3.2 y son:

- C. Un método que recurre a Splines Cúbicos para todas las resoluciones previas de P.V.I. no lineales, basado como ya se ha dicho, en el Método del Disparo No Lineal.
- D. Otro método que, con los mismos fundamentos, se ayuda en este caso de los métodos basados en Pseudo- Splines Cuárticos.

Las funciones que permiten implementar estos cuatro últimos métodos en Matlab se incorporan en el apartado 7.2.2 de los Anexos.

Como hemos adelantado, en este capítulo se hará un breve análisis sobre la precisión, la eficiencia, el orden de convergencia y las propiedades de cada uno de los métodos anteriormente mencionados. Para llevar a cabo este análisis, se aplicarán los códigos implementados en Matlab de dichos métodos sobre una serie de ejemplos de problemas ligados al ámbito de la Ingeniería.

Se incluirán, para los distintos problemas estudiados, tablas y gráficas relativas al error cometido, así como representaciones numéricas de las soluciones que permitan visualizar su comportamiento.

## 5.2. Ejemplo 1 <sup>[10]</sup>

### 5.2.1. Planteamiento del problema

Si  $u$  representa el potencial electrostático entre dos esferas metálicas concéntricas, de radios  $R_1$  y  $R_2$ , con  $0 < R_1 < R_2$ , de modo que el potencial de la esfera interior se mantenga constante en  $V_1$  voltios y el de la exterior sea 0 voltios, entonces el potencial de la región situada entre ambas esferas viene dado por la solución de la ecuación de Laplace:

$$\begin{cases} u''(r) + \frac{2}{r} u'(r) = 0 \\ u(R_1) = V_1 \\ u(R_2) = 0 \end{cases}, \quad r \in [R_1, R_2] \quad (5.1)$$

Dicha solución se conoce y toma la forma:

$$u(r) = \frac{V_1 R_1}{r} \cdot \left( \frac{R_2 - r}{R_2 - R_1} \right) \quad (5.2)$$

### 5.2.2. Resolución

Como podemos observar, nuestro primer ejemplo (5.1) toma la forma de un P.V.F. Lineal, con condiciones de contorno sencillas en los extremos del intervalo.

Es por ello que, para su resolución vamos a poder recurrir a los métodos implementados en las funciones: *spline3pc*, *spline3pc2*, *spline4pc* y *spline4pc2*. Puede verse el código de Matlab utilizado para resolver este Ejemplo 1 en el apartado 7.3.1. del Anexo.

En nuestra simulación hemos supuesto los siguientes valores para los parámetros del problema:  $V_1 = 110 \text{ V}$ ,  $R_1 = 2 \text{ m}$ ,  $R_2 = 4 \text{ m}$ .

### 5.2.3. Resultados

En este apartado vamos a presentar las tablas de resultados y las gráficas obtenidas mediante Matlab.

En primer lugar, mostramos en la siguiente tabla, para diferentes valores de  $x = r$  del intervalo en que está definido el problema, los valores de la solución exacta:  $y(x)$ , frente a los de las aproximaciones numéricas obtenidas mediante los métodos *spline3pc*:  $y_{31}(x)$ , *spline3pc2*:  $y_{32}(x)$ , *spline4pc*:  $y_{41}(x)$  y *spline4pc2*:  $y_{42}(x)$ , respectivamente. Para la integración del problema se ha utilizado un tamaño de paso fijo  $h = 0.4$ .

Tabla 5.2. Valores de las soluciones numéricas ( $h = 0.4$ )

$x$	$y_{31}(x)$	$y_{32}(x)$	$y_{41}(x)$	$y_{42}(x)$	$y(x)$
2.0	1.100000e+02	1.100000e+02	1.100000e+02	1.100000e+02	1.100000e+02
2.4	7.292875e+01	7.292875e+01	7.333731e+01	7.333731e+01	7.333333e+01
2.8	4.672321e+01	4.672321e+01	4.714662e+01	4.714662e+01	4.714286e+01
3.2	2.719333e+01	2.719333e+01	2.750258e+01	2.750258e+01	2.750000e+01
3.6	1.206656e+01	1.206656e+00	1.222347e+01	1.222347e+01	1.222222e+01
4.0	-1.065814e-14	3.552714e-15	1.776357e-15	1.776357e-15	0.000000e+00

En dicha tabla observamos que, prescindiendo de pequeños errores de redondeo, los resultados obtenidos con las funciones *spline3pc* y *spline3pc2* son virtualmente idénticos. Lo mismo ocurre con los resultados obtenidos al aplicar las funciones *spline4pc* y *spline4pc2*. Esto no debería sorprendernos, pues, en ausencia de errores de redondeo, implementan exactamente el mismo Spline (o Pseudo-Spline), aunque implementado en distintas bases y construido de manera muy diferente. En general, cuando no haya inconveniente en utilizar nodos equiespaciados, será preferible la implementación de las versiones que hacen uso del Algoritmo de Thomas, por su menor coste computacional. Sin embargo, en algunos problemas concretos será deseable hacer uso de nodos no equiespaciados, en aras de una mayor eficiencia y precisión.

También observamos en la tabla, al comparar con la solución exacta de la que en este caso se dispone, que en tanto que el error absoluto máximo en los puntos considerados viene a ser de 0.4 en el caso del Spline Cúbico, está por debajo de 0.004 en el caso del Pseudo-Spline Cuártico. Los errores relativos para los métodos basados en el Pseudo-Spline Cuártico, están por debajo de 0.000055.

A continuación, se muestra una gráfica en la que puede observarse el comportamiento de dichas soluciones. Nótese la diferencia de escala entre el eje de abscisas y el de ordenadas (las soluciones se mueven de 110 a 0 en un intervalo de amplitud 2).

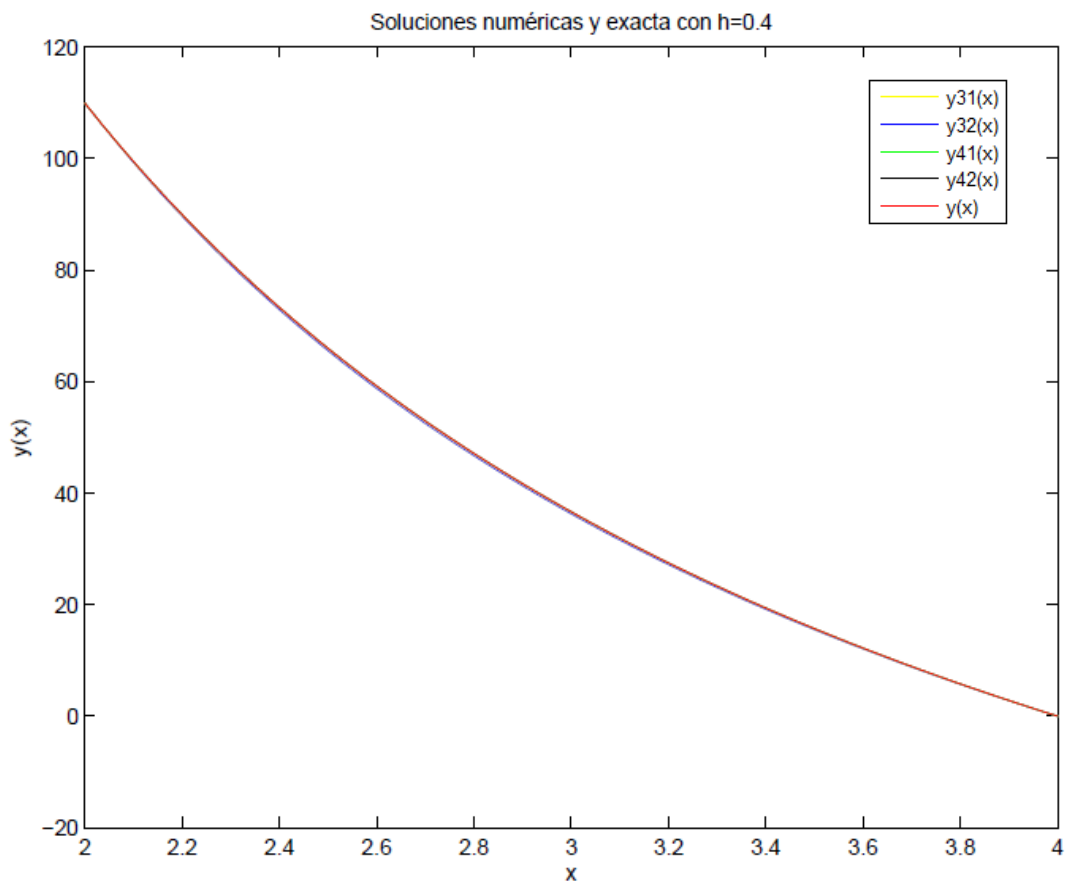
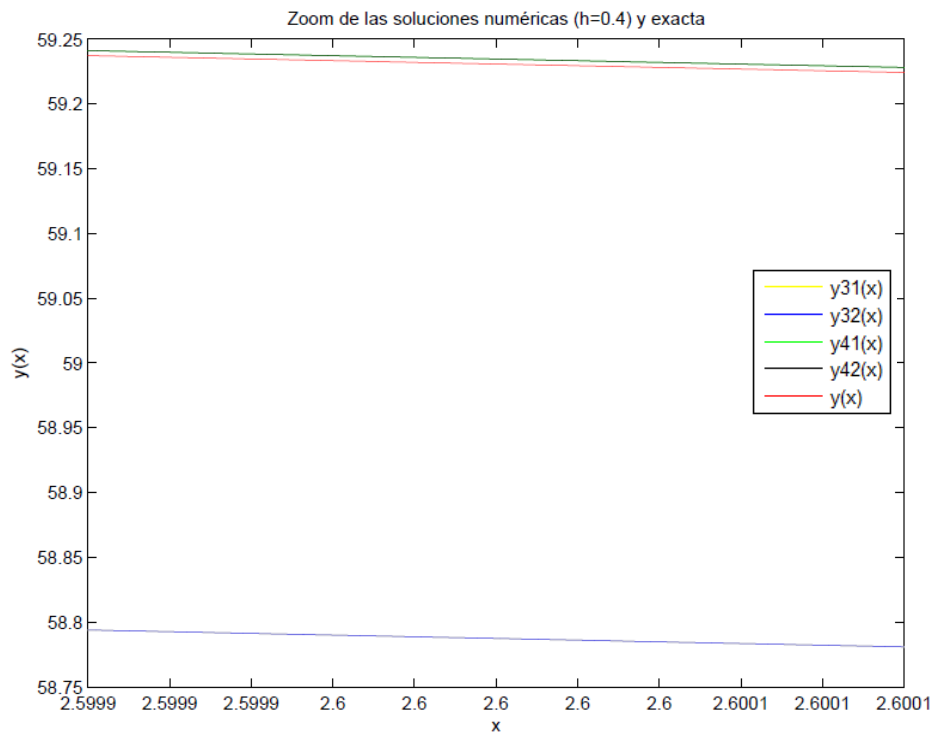


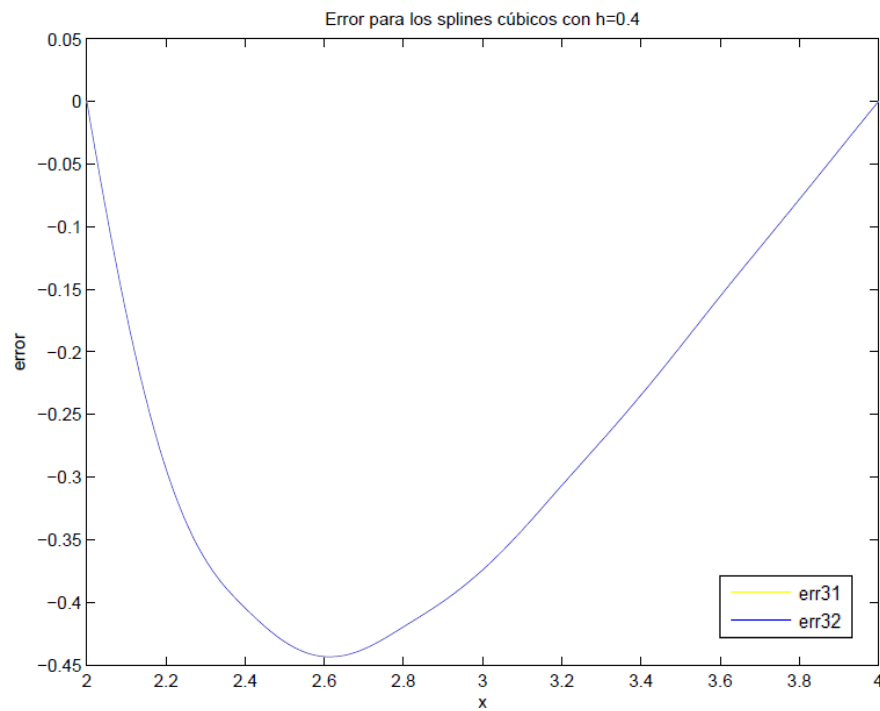
Figura 5.1. Gráfica de las soluciones numéricas y exactas ( $h = 0.4$ )

En esta primera gráfica, las soluciones son prácticamente indistinguibles, pues los errores relativos son, en todos los casos (incluso en el caso del Spline Cúbico), difícilmente apreciables.

Por esta razón, incluimos una segunda gráfica haciendo zoom en un pequeño intervalo, que permite apreciar las diferencias de manera más evidente.

Figura 5.2. Detalle de la gráfica anterior: Zoom de las soluciones numéricas y exactas ( $h = 0.4$ )

Repetimos ahora las gráficas anteriores representando el error cometido en cada punto del intervalo.

Figura 5.3. Errores en las aproximaciones obtenidas con Splines Cúbicos ( $h = 0.4$ )

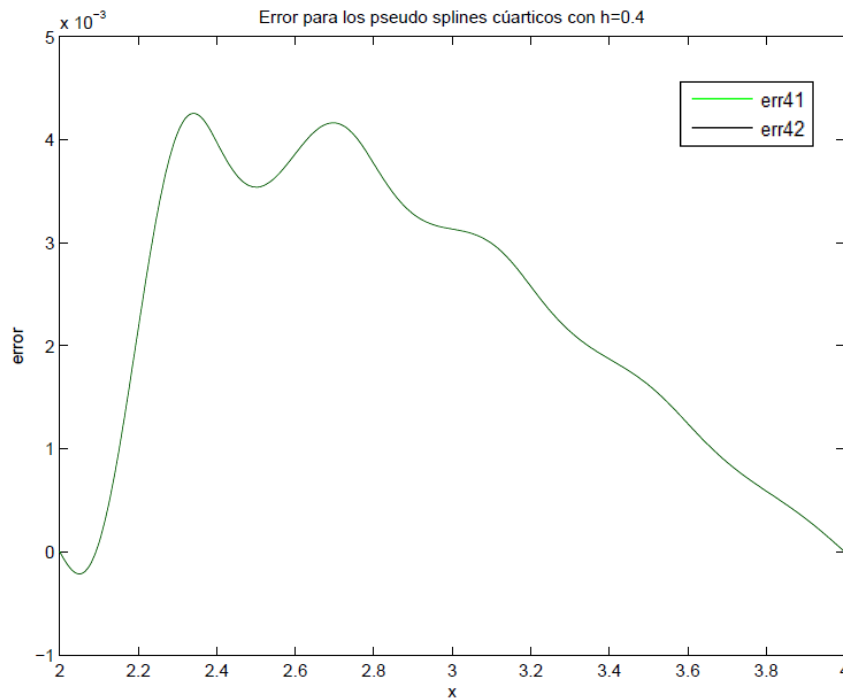


Figura 5.4. Errores en las aproximaciones obtenidas con Pseudo-Splines Cuárticos ( $h = 0.4$ )

Con el fin de poder apreciar la evolución de los errores al ir disminuyendo el tamaño del paso en la integración numérica del problema, se muestran a continuación unas tablas que recogen dichos errores para los distintos métodos, en varios de los puntos de integración. Los tamaños de paso considerados, se han ido variando a partir del paso inicial  $h = 0.4$ , dividiéndolo por sucesivas potencias de 2, según la siguiente expresión:  $h = \frac{0.4}{2^i}$ , para valores de  $i = 1, 2, \dots, 7$ .

Tabla 5.3. Errores en las soluciones numéricas de los distintos métodos ( $h = \frac{0.4}{2^i}$ ,  $i = 0, 1, \dots, 7$ )

ERRORES CON EL MÉTODO 'spline3pc'								
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$	$i = 7$
2.0	0.00e+00	0.00e+00	1.42e-14	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
2.4	-4.05e-01	-9.65e-02	-2.39e-02	-5.95e-03	-1.49e-03	-3.71e-04	-9.28e-05	-2.32e-05
2.8	-4.20e-01	-1.01e-01	-2.49e-02	-6.22e-03	-1.55e-03	-3.88e-04	-9.71e-05	-2.43e-05
3.2	-3.07e-01	-7.39e-02	-1.83e-02	-4.57e-03	-1.14e-03	-2.85e-04	-7.13e-05	-1.78e-05
3.6	-1.56e-01	-3.76e-02	-9.33e-03	-2.33e-03	-5.82e-04	-1.45e-04	-3.63e-05	-9.09e-06
4.0	-1.07e-14	7.99e-15	9.77e-15	1.44e-14	1.11e-16	3.89e-15	2.78e-16	7.81e-15

ERRORES CON EL MÉTODO 'spline3pc2'								
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$	$i = 7$
2.0	0.00e+00	1.42e-14	1.42e-14	-1.42e-14	0.00e+00	0.00e+00	0.00e+00	-1.42e-14
2.4	-4.05e-01	-9.65e-02	-2.39e-02	-5.95e-03	-1.49e-03	-3.71e-04	-9.28e-05	-2.32e-05
2.8	-4.20e-01	-1.01e-01	-2.49e-02	-6.22e-03	-1.55e-03	-3.88e-04	-9.71e-05	-2.43e-05
3.2	-3.07e-01	-7.39e-02	-1.83e-02	-4.57e-03	-1.14e-03	-2.85e-04	-7.13e-05	-1.78e-05
3.6	-1.56e-01	-3.76e-02	-9.33e-03	-2.33e-03	-5.82e-04	-1.45e-04	-3.63e-05	-9.09e-06
4.0	-1.07e-14	-5.33e-15	-2.22e-15	4.88e-15	2.44e-15	-4.88e-15	-2.44e-15	4.88e-15
ERRORES CON EL MÉTODO 'spline4pc'								
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$	$i = 7$
2.0	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
2.4	3.98e-03	2.51e-04	1.58e-05	9.85e-07	6.16e-08	3.85e-09	2.41e-10	1.50e-11
2.8	3.77e-03	2.38e-04	1.49e-05	9.33e-07	5.83e-08	3.64e-09	2.28e-10	1.42e-11
3.2	2.58e-03	1.63e-04	1.02e-05	6.37e-07	3.98e-08	2.49e-09	1.56e-10	9.70e-12
3.6	1.24e-03	7.85e-05	4.92e-06	3.08e-07	1.92e-08	1.20e-09	7.51e-11	4.67e-12
4.0	1.78e-15	-6.22e-15	-1.24e-14	2.22e-16	1.11e-16	4.33e-15	-6.19e-15	-1.50e-14
ERRORES CON EL MÉTODO 'spline4pc2'								
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$	$i = 7$
2.0	0.00e+00	0.00e+00	1.42e-14	-1.42e-14	0.00e+00	0.00e+00	0.00e+00	0.00e+00
2.4	3.98e-03	2.51e-04	1.58e-05	9.85e-07	6.16e-08	3.85e-09	2.38e-10	-3.25e-12
2.8	3.77e-03	2.38e-04	1.49e-05	9.33e-07	5.83e-08	3.64e-09	2.25e-10	-1.99e-11
3.2	2.58e-03	1.63e-04	1.02e-05	6.37e-07	3.98e-08	2.49e-09	1.54e-10	-1.83e-11
3.6	1.24e-03	7.85e-05	4.92e-06	3.08e-07	1.92e-08	1.20e-09	7.56e-11	-7.02e-12
4.0	1.78e-15	-4.44e-15	-2.66e-15	4.22e-15	2.55e-15	-5.05e-15	-2.47e-15	4.90e-15

En las tablas observamos de nuevo que, prescindiendo de pequeños errores de redondeo, los resultados obtenidos con las funciones *spline3pc* y *spline3pc2* son virtualmente idénticos. Lo mismo ocurre con los resultados obtenidos al aplicar las funciones *spline4pc* y *spline4pc2*. En este último caso, para los tamaños de paso de integración más pequeños considerados, las diferencias son más apreciables (como cabía esperar).

El orden de convergencia efectivo de cada método aplicado a este problema, puede deducirse de las tablas anteriores, observando cómo disminuye el error al reducir el tamaño del paso considerado. También puede determinarse a partir de las gráficas de las aproximaciones a la solución, representando en doble escala logarítmica el error frente al paso  $h$ , y midiendo la pendiente de las gráficas resultantes.

Para determinar, del modo descrito, el orden efectivo de los métodos, hemos considerado uno de los puntos más cercanos a aquellos en los que se hace máximo el error de los métodos basados en Splines Cúbicos y en Pseudo-Splines Cuárticos, concretamente  $x = 2.8$ .

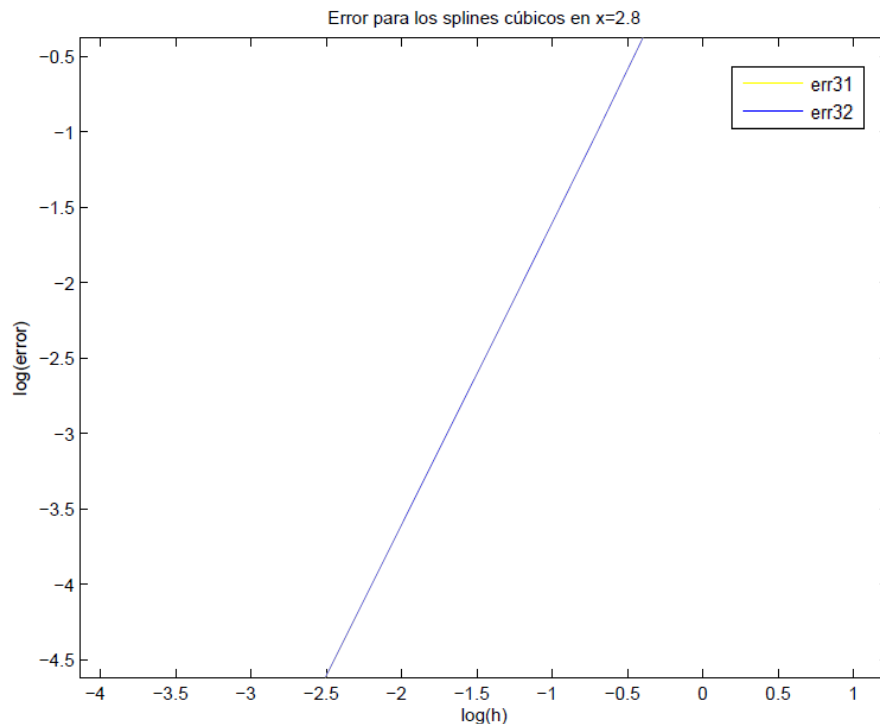


Figura 5.4. Gráfica en doble escala logarítmica de los errores para Splines Cúbicos en el punto  $x = 2.8$ , para los distintos pasos



Como puede observarse, la curva representada en la gráfica es casi una recta de pendiente:  $m \cong \frac{-0.4 - (-4.6)}{-0.4 - (-2.5)} = 2$ , por lo que los métodos basados en Splines Cúbicos convergen a la solución de este problema con orden 2.

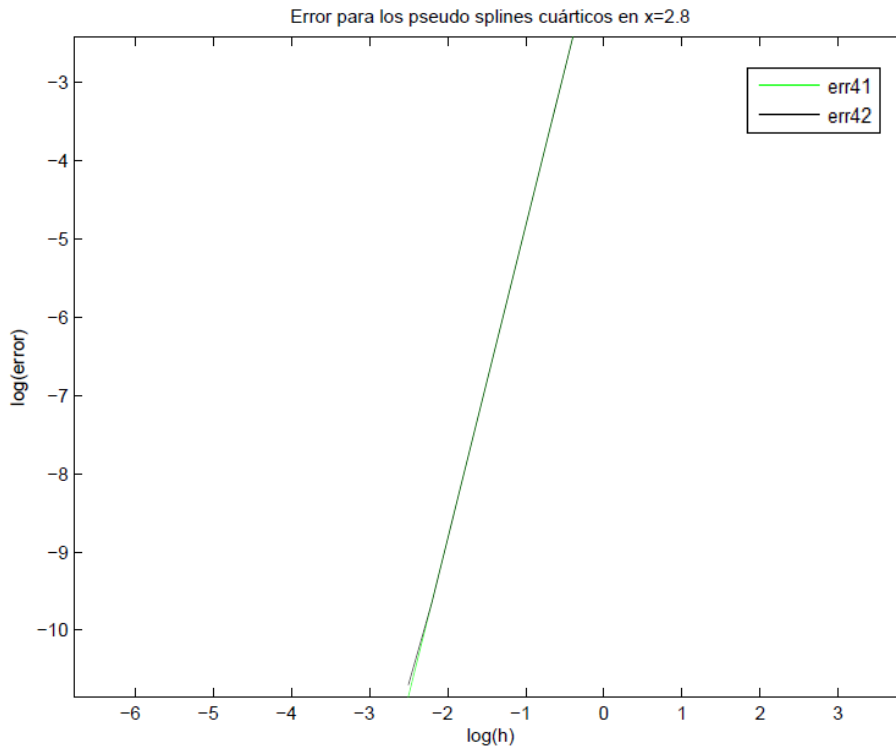


Figura 5.5. Gráfica en doble escala logarítmica de los errores para Pseudo-Splines Cuárticos en el punto  $x = 2.8$ , para los distintos pasos

Para los Pseudo-Splines Cuárticos, la pendiente de la gráfica es mayor:  $m \cong \frac{-2.4 - (-10.8)}{-0.4 - (-2.5)} = 4$ , de modo que este tipo de métodos, aplicados a este problema, muestran un orden efectivo de convergencia 4.

A la luz de los resultados anteriores, se observa claramente que los métodos basados en Pseudo-Splines Cuárticos son mucho mejores que los basados en Splines Cúbicos, puesto que con menos del doble puntos de colocación (evaluación del funcional lineal que define el problema) consigue errores mucho menores. Por ejemplo, de las gráficas anteriores se deduce que los segundos, con el tamaño de paso más pequeño considerado en las gráficas hacen uso de una evaluación (del funcional lineal que define la ecuación) más que los primeros con tamaño de paso doble, en tanto que obtienen en torno cinco dígitos menos de precisión en la solución.

### 5.3. Ejemplo 2 [28] [29]

#### 5.3.1. Planteamiento del problema

Consideremos el siguiente P.V.F., que corresponde a un caso real:

$$\begin{cases} u''(S) + (3 \cot g(S) + 2 \operatorname{tg}(S)) u'(S) + 0,7 u(S) = 0 \\ u(30) = 0 \\ u(60) = 5 \end{cases}, S \in [30, 60] \quad (5.3)$$

El problema anterior surge al considerar la distribución de tensiones en una membrana esférica sometida a cargas normales y tangenciales.

El autor del artículo, que corresponde a las publicaciones de la referencia [28] y [29] de la Bibliografía, del cual hemos extraído este problema, menciona que para su aplicación física se requiere una buena aproximación de  $u'(S)$ . Asimismo, menciona que algunos intentos de resolverlo mediante un Método del Disparo han resultado poco satisfactorios.

Aunque el problema admite una solución única, ésta presenta rápidas variaciones y no se dispone de una solución exacta del mismo, lo que hace este problema especialmente interesante para probar la validez de los métodos que se han presentado en este proyecto.

Otro aspecto que hay que tener en cuenta es que, al evaluar las funciones trigonométricas, la variable independiente  $S$  viene dada en grados sexagesimales.

#### 5.3.2. Resolución

Como podemos ver este problema, al igual que el del ejemplo 1, es un P.V.F. con condiciones de contorno simples en los extremos del intervalo, por lo que de nuevo vamos a recurrir a su resolución mediante los métodos implementados en las siguientes funciones *spline3pc*, *spline3pc2*, *spline4pc* y *spline4pc2*.

Aunque se trate del mismo tipo de problema, su análisis nos va a permitir extraer conclusiones diferentes. En el apartado 7.3.2 del Anexo, puede encontrarse el código del programa mediante el cual se obtienen los resultados que se presentan a continuación.

### 5.3.3. Resultados

Comenzamos presentando la tabla que recoge las aproximaciones a la solución que han sido obtenidas mediante los métodos numéricos: *spline3pc*:  $y_{31}(x)$ , *spline3pc2*:  $y_{32}(x)$ , *spline4pc*:  $y_{41}(x)$ , y *spline4pc2*:  $y_{42}(x)$ , evaluadas en diferentes puntos de  $x = S$ , dentro del intervalo  $[30, 60]$  para el que está definido el problema.

Para estas aproximaciones se ha fijado el tamaño de paso fijo en la integración  $h = 0.6$ . Por razones idénticas a las mencionadas al estudiar el problema anterior, los resultados (salvo errores de redondeo) de los dos métodos basados en Splines Cúbicos coinciden, y lo mismo ocurre para los dos basados en Pseudo-Splines Cuárticos.

Tabla 5.4. Valores de las soluciones numéricas ( $h = 0.6$ )

$x$	$y_{31}(x)$	$y_{32}(x)$	$y_{41}(x)$	$y_{42}(x)$
35.01	1.713850e+02	1.713850e+02	1.714374e+02	1.714374e+02
40.01	8.894574e+01	8.894574e+01	8.894925e+01	8.894925e+01
45.01	4.405628e+01	4.405628e+01	4.405783e+01	4.405783e+01
50.01	2.123613e+01	2.123613e+01	2.123670e+01	2.123670e+01
55.01	1.018986e+01	1.018986e+01	1.019001e+01	1.019001e+01

Aunque no se dispone de la solución exacta de este problema, hemos comparado nuestros resultados con los que recoge el autor de uno de los artículos que lo mencionan, y que incluye una tabla similar (evaluando sus aproximaciones en los mismos puntos).

Los seis primeros dígitos (que son los únicos que muestra su tabla) de cada aproximación en cada punto, coinciden con los que proporcionan nuestros métodos basados en Pseudo-Splines Cuárticos, por lo que podemos darlos por buenos. Eso sí, la solución de referencia que utiliza el autor se obtiene mediante un método en diferencias centradas basada en una red de 3000 puntos, en tanto que nuestros métodos sólo utilizan 50 intervalos (el tamaño de paso fijo que hemos utilizado es  $h = 0.6$ ).

Pasamos a representar gráficamente estas soluciones:

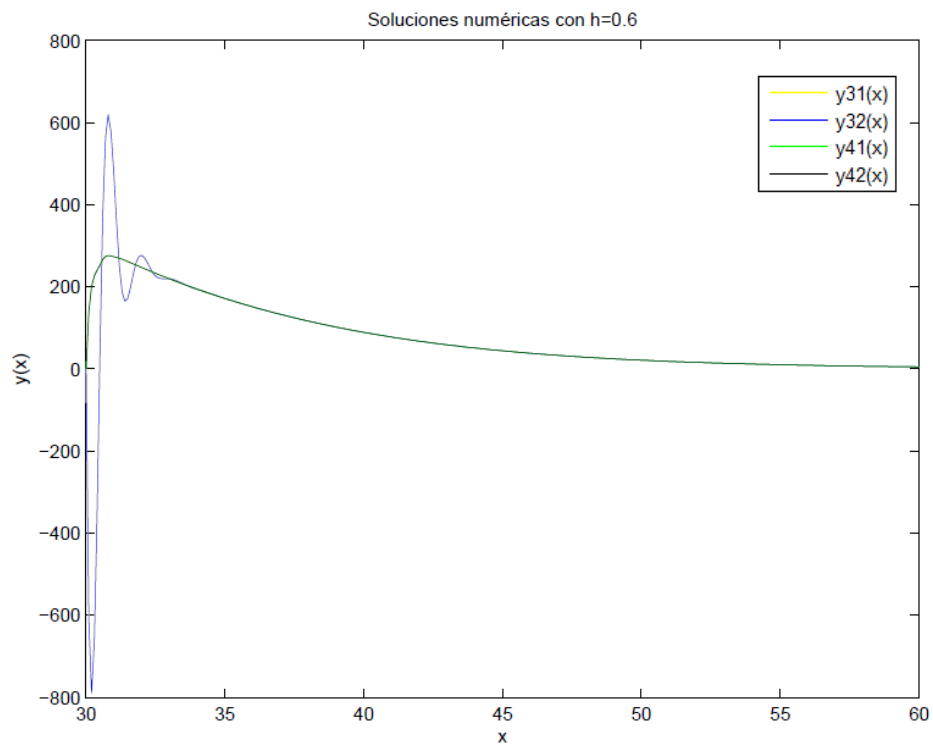


Figura 5.6. Representación gráfica de las soluciones ( $h = 0.6$ )

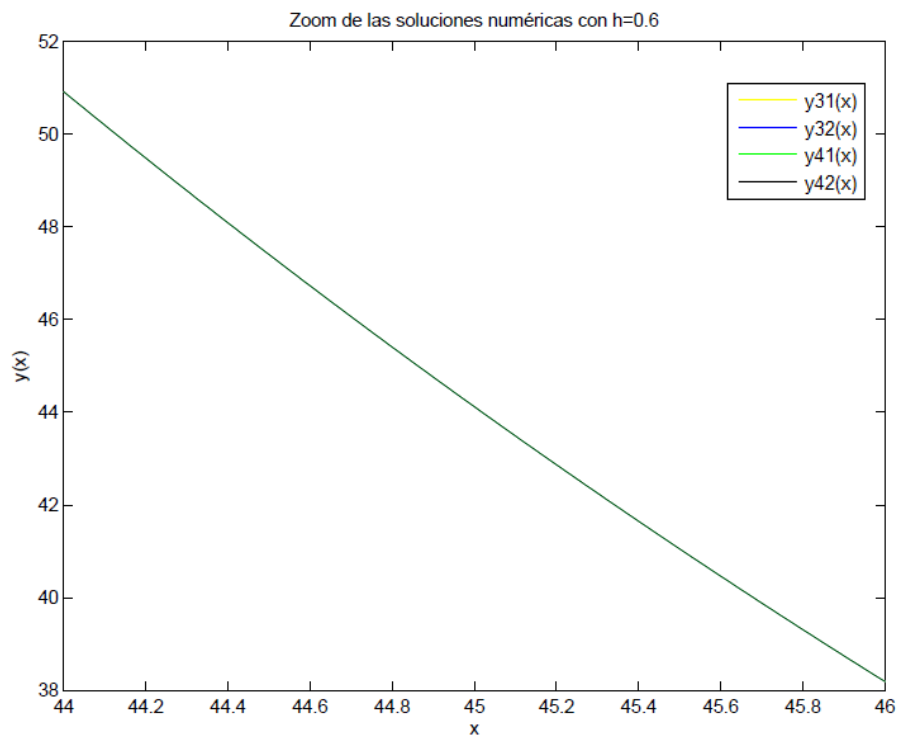


Figura 5.7. Detalle de la gráfica anterior: zoom de las soluciones numéricas y exactas ( $h = 0.6$ )

En las dos gráficas anteriores, podemos observar que los dos métodos basados en Pseudo-Splines Cuárticos reproducen fielmente la solución, en tanto que los basados en Splines Cúbicos, para este tamaño de paso, sólo lo hacen tras un período inicial de fuertes fluctuaciones. Veremos más adelante que estas fluctuaciones iniciales desaparecen completamente al considerar tamaños de paso más pequeños.

Estamos interesados en estudiar el comportamiento del error frente al tamaño de paso utilizado, con vistas a deducir el orden de convergencia efectivo que presentan nuestros métodos al ser aplicados al problema que nos ocupa. Para ello, vamos a ir variando el número de subintervalos  $n$  de acuerdo a la expresión:  $n = 50 \cdot 2^i$ ,  $i = 0, 1, 2, 3, 4$ , o lo que es lo mismo, iremos tomando tamaños de paso fijo  $h = 0.6, 0.3, 0.15, 0.075$  y  $0.0375$ . Tras los experimentos numéricos, obtenemos las siguientes tablas:

Tabla 5.5. Soluciones numéricas de los distintos métodos ( $h = \frac{0.6}{2^i}$ ,  $i = 0, 1, \dots, 7$ )

SOLUCIONES NUMÉRICAS CON EL MÉTODO 'spline3pc'					
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
30	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
35	1.716006e+02	1.716509e+02	1.716522e+02	1.716526e+02	1.716527e+02
40	8.906718e+01	8.906981e+01	8.907047e+01	8.907064e+01	8.907068e+01
45	4.411973e+01	4.412089e+01	4.412119e+01	4.412126e+01	4.412128e+01
50	2.126742e+01	2.126784e+01	2.126795e+01	2.126798e+01	2.126798e+01
55	1.020471e+01	1.020482e+01	1.020485e+01	1.020486e+01	1.020486e+01
60	5.000000e+00	5.000000e+00	5.000000e+00	5.000000e+00	5.000000e+00

SOLUCIONES NUMÉRICAS CON EL MÉTODO 'spline3pc2'					
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
30	4.547474e-13	0.000000e+00	7.105427e-15	0.000000e+00	4.440892e-16
35	1.716006e+02	1.716509e+02	1.716522e+02	1.716526e+02	1.716527e+02
40	8.906718e+01	8.906981e+01	8.907047e+01	8.907064e+01	8.907068e+01
45	4.411973e+01	4.412089e+01	4.412119e+01	4.412126e+01	4.412128e+01
50	2.126742e+01	2.126784e+01	2.126795e+01	2.126798e+01	2.126798e+01
55	1.020471e+01	1.020482e+01	1.020485e+01	1.020486e+01	1.020486e+01
60	5.000000e+00	5.000000e+00	5.000000e+00	5.000000e+00	5.000000e+00

SOLUCIONES NUMÉRICAS CON EL MÉTODO 'spline4pc'					
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
30	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
35	1.716527e+02	1.716527e+02	1.716527e+02	1.716527e+02	1.716527e+02
40	8.907069e+01	8.907069e+01	8.907069e+01	8.907069e+01	8.907069e+01
45	4.412129e+01	4.412128e+01	4.412128e+01	4.412128e+01	4.412128e+01
50	2.126799e+01	2.126798e+01	2.126798e+01	2.126798e+01	2.126798e+01
55	1.020486e+01	1.020486e+01	1.020486e+01	1.020486e+01	1.020486e+01
60	5.000000e+00	5.000000e+00	5.000000e+00	5.000000e+00	5.000000e+00

SOLUCIONES NUMÉRICAS CON EL MÉTODO 'spline4pc2'					
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
30	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
35	1.716527e+02	1.716527e+02	1.716527e+02	1.716527e+02	1.716527e+02
40	8.907069e+01	8.907069e+01	8.907069e+01	8.907069e+01	8.907069e+01
45	4.412129e+01	4.412128e+01	4.412128e+01	4.412128e+01	4.412128e+01
50	2.126799e+01	2.126798e+01	2.126798e+01	2.126798e+01	2.126798e+01
55	1.020486e+01	1.020486e+01	1.020486e+01	1.020486e+01	1.020486e+01
60	5.000000e+00	5.000000e+00	5.000000e+00	5.000000e+00	5.000000e+00

A medida que disminuimos el tamaño de paso, las soluciones numéricas obtenidas con los distintos métodos difieren cada vez menos entre sí.

Es más, a partir de tamaños de paso menores que  $h = 0.6$ , los métodos basados en Pseudo-Splines Cuárticos proporcionan los mismos resultados (para el número de cifras mostrado en la tabla, pues en realidad seguirían mejorando en precisión al disminuir el paso).

De hecho, si representamos las distintas soluciones para el tamaño de paso más pequeño considerado ( $h = 0.0375$ ), vemos que la gráfica que se obtiene es prácticamente idéntica para los cuatro métodos.

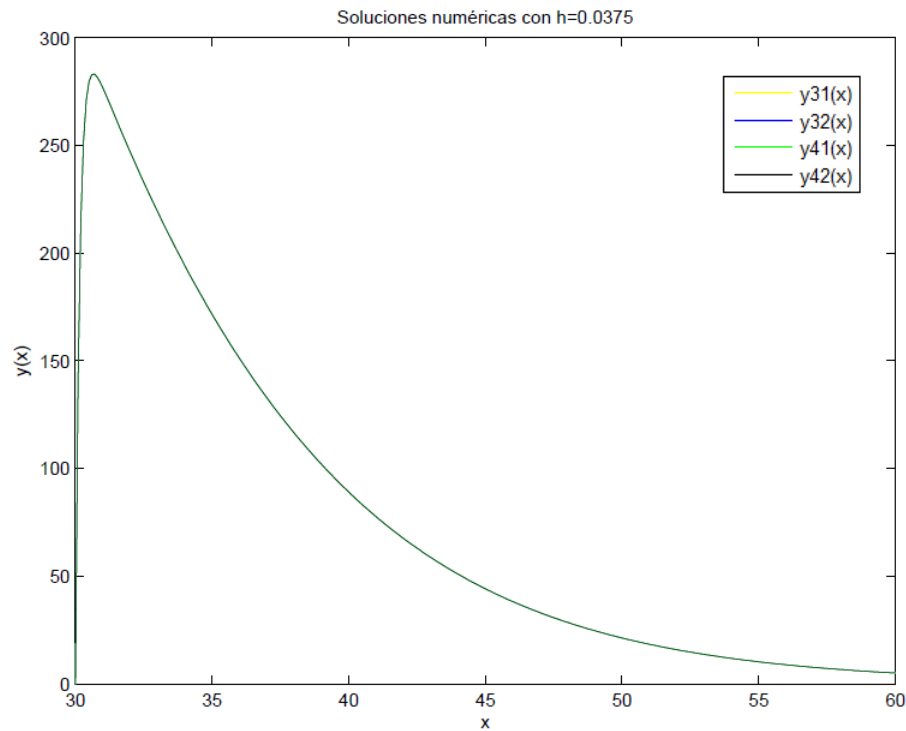


Figura 5.8. Gráficas de las soluciones numéricas ( $h = 0.0375$ )

## 5.4. Ejemplo 3 [28] [29]

### 5.4.1. Planteamiento del problema

En este ejemplo consideremos el P.V.F. lineal:

$$\begin{cases} \varepsilon u''(S) - (2 - S^2) u(S) = -1 \\ u(-1) = u(1) = 0 \end{cases}, \quad 0 < \varepsilon \ll 1 \quad (5.4)$$

Este problema de perturbación viene dado, a modo de ilustración, en un artículo de 'Geophysical Dynamics', es decir, de Dinámica Geofísica, concretamente en la referencia [28] de la Bibliografía.

La Geodinámica es el subcampo de la Geofísica que se ocupa de la dinámica de la Tierra, estudia aspectos como la tectónica de placas, fenómenos geológicos como la propagación del fondo marino, la construcción de montañas, volcanes, terremotos, fallas, etc. También trata de sondear la

actividad interna midiendo los campos magnéticos, la gravedad y las ondas sísmicas, así como la mineralogía de las rocas y su composición isotópica. Métodos de geodinámica también se aplican a la exploración de otros planetas. Este campo es de gran relevancia tanto en Ingeniería Geofísica, como en Ingeniería Civil y Aeronáutica.

El problema (5.4) considerado, admite una única solución que es simétrica respecto de 0 y tiene una ‘capa límite’ en 1 de anchura  $O(\sqrt{\varepsilon})$ . Esta simetría nos permite reescribir el problema, como hace uno de los autores de los artículos mencionados, restringiendo su estudio al intervalo  $[0, 1]$  y considerando las nuevas condiciones frontera:  $u'(0) = u(1) = 0$ . Así lo haremos en lo que sigue.

En el otro artículo mencionado se establece que una solución asintótica al problema viene dada por:

$$u(s) \sim \frac{1}{2 - S^2} - \exp\left(-\frac{1 + S}{\sqrt{\varepsilon}}\right) - \exp\left(-\frac{1 - S}{\sqrt{\varepsilon}}\right) \quad (5.5)$$

#### 5.4.2. Resolución

Para resolver el problema anterior, vamos a tomar inicialmente el valor de  $\varepsilon = 10^{-4}$  (Ejemplo 3.a), y después consideraremos el caso de  $\varepsilon = 10^{-8}$  (Ejemplo 3.b), ambos incluidos en los Anexos en su apartado 7.3.3.

Como puede verse, se trata de un Problema de Valores de Frontera, con condiciones de contorno generales. No obstante, este caso no puede resolverse mediante los métodos *spline3pc2* y *spline4pc2*, utilizando un número pequeño de subintervalos equiespaciados, debido a que las rápidas variaciones en el comportamiento de la solución, especialmente cuando nos aproximamos a la ‘capa límite’, no permiten obtener soluciones numéricas suficientemente precisas.

Eso sí, conviene comentar en este punto que, tomando tamaños de paso suficientemente pequeños (cuanto menor sea  $\varepsilon$  más pequeño ha de tomarse  $h$ ), con nodos equiespaciados, los métodos que hemos descartado funcionarían correctamente. Debido a lo anterior, vamos a resolverlo a partir de los métodos *spline3pc* y *spline4pc* basados en el Método del Disparo y que permiten utilizar una red de nodos no equiespaciados.



Es en este punto precisamente donde reside la principal dificultad de este problema, y la razón por la que se ha elegido estudiarlo, pues estamos buscando soluciones no demasiado precisas; pero que se obtengan con un coste computacional asociado muy bajo.

### 5.4.3. Resultados

#### a) Ejemplo 3.a

Tomamos para este caso, como ya hemos mencionado, un valor del parámetro  $\varepsilon = 10^{-4}$ . Si seleccionamos un número de subintervalos  $n = 5$  y seleccionamos los nodos:  $x = 0, 0.4, 0.85, 0.96, 0.99, 1$ , en línea con lo que el autor hace en su artículo, las soluciones obtenidas mediante los métodos *spline3pc*:  $y_{31}(x)$  y *spline4pc*:  $y_{41}(x)$ , y la solución asintótica  $y(x)$  se recogen en la siguiente tabla:

Tabla 5.6. Valores de las soluciones numéricas y asintótica con  $n = 5$  y  $\varepsilon = 10^{-4}$

$x$	$y_{31}(x)$	$y_{41}(x)$	$y(x)$
0.0	4.999395e-01	5.000157e-01	5.000000e-01
0.2	4.933164e-01	5.102377e-01	5.102041e-01
0.4	5.436989e-01	5.434788e-01	5.434783e-01
0.6	7.012256e-01	6.077200e-01	6.097561e-01
0.8	7.950783e-01	7.464503e-01	7.352941e-01
1.0	-2.045031e-13	8.667511e-13	0.000000e+00

A continuación, pasaremos a representar gráficamente las soluciones numéricas obtenidas.

En la gráfica que se muestra en la Figura 5.9 de la siguiente página, podemos comprobar que el método *spline3pc*, basado en Splines Cúbicos, presenta oscilaciones amplias entorno a la solución asintótica, especialmente a medida que se acerca a la ‘capa límite’. Un fenómeno similar se observa para alguno de los métodos considerados en uno de los artículos que hemos mencionado.

Sin embargo, el método *spline4pc*, que recurre a Pseudo-Splines Cuárticos, sigue con mayor exactitud dicha solución a lo largo de todo el intervalo.

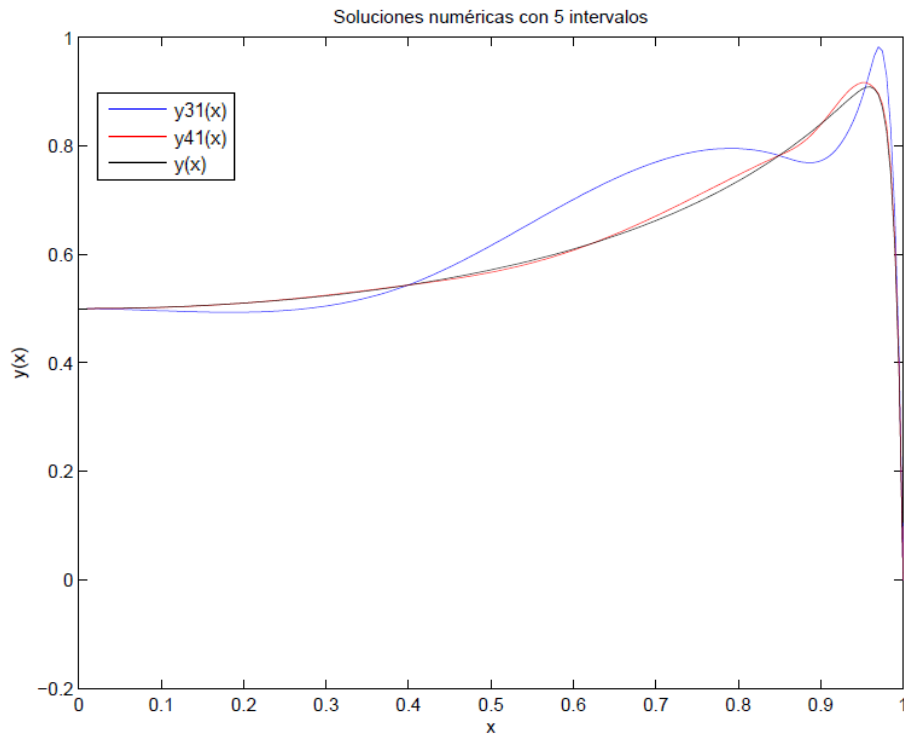


Figura 5.9. Gráficas de las soluciones numéricas y asintótica tomando 6 puntos ( $\varepsilon = 10^{-4}$ )

Ampliando la gráfica anterior en su tramo final obtenemos:

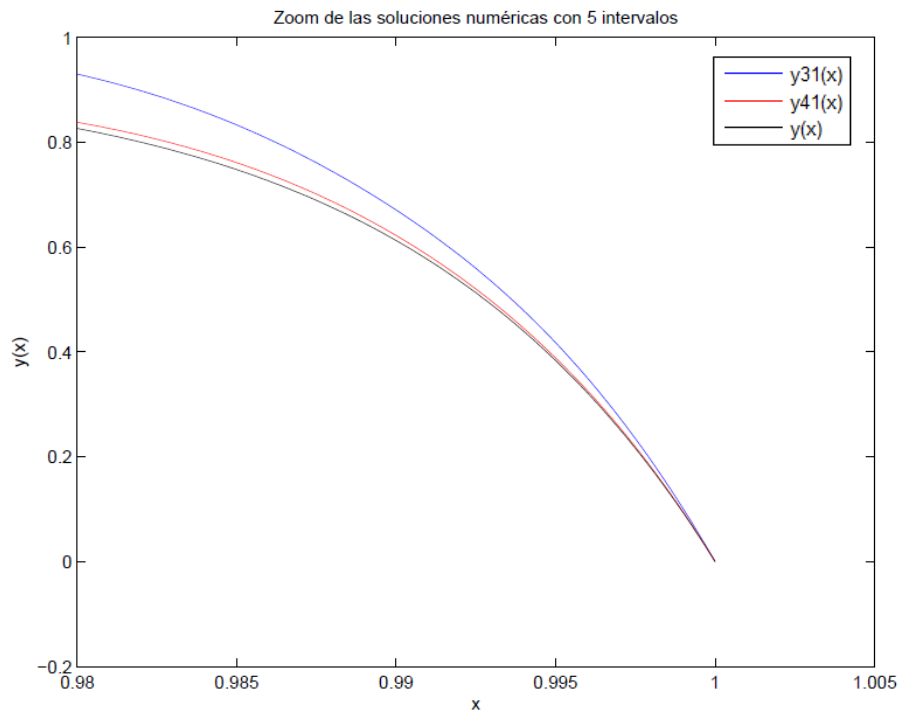


Figura 5.10. Detalle de la gráfica anterior: zoom de las soluciones numéricas y exactas con 6 puntos ( $\varepsilon = 10^{-4}$ )

En la Figura anterior podemos observar que el error es mayor en el método basado en Splines Cúbicos como comentamos. Además, podemos ver como las tres soluciones tienden rápidamente a 0 en el extremo del intervalo en el que está definido el problema.

Si ahora representamos gráficamente el comportamiento de los errores cometidos con cada método en la aproximación a la solución, tomando 8 puntos (concretamente los mismos que el autor de uno de los artículos menciona):  $x = 0, 0.3, 0.6, 0.85, 0.95, 0.97, 0.99, 1$  y 7 subintervalos, resulta la siguiente figura:

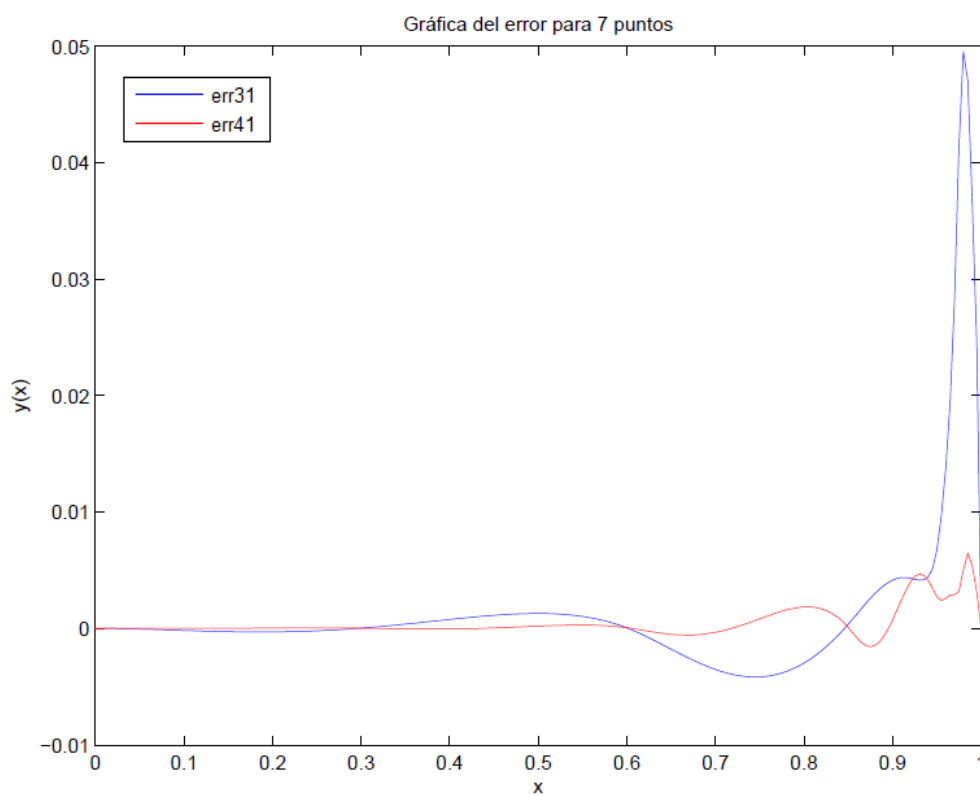


Figura 5.11. Gráfica de los errores en la aproximación a la solución tomando 8 puntos ( $\varepsilon = 10^{-4}$ )

Es sencillo deducir de la anterior gráfica, que los errores cometidos por el método *spline3pc* que recurre a Splines Cúbicos son mayores que los del método *spline4pc* que hace uso de los Pseudo-Splines Cuárticos.

También se aprecia con facilidad, lo que ya hemos comentado, que cuanto más se aproximan las soluciones a la ‘capa límite’ que se encuentra al final del intervalo, mayores son los errores.

## b) Ejemplo 3.b

Ahora vamos a estudiar el comportamiento de las soluciones para el valor del parámetro  $\varepsilon = 10^{-8}$  y  $n = 12$  subintervalos. Los nodos considerados son:  $x = 0, 0.4, 0.8, 0.95, 0.98, 0.992, 0.997, 0.999, 0.9995, 0.9999, 0.99993, 0.99996, 1$ .

La tabla de las soluciones obtenidas, aplicando los métodos *spline3pc*:  $y_{31}(x)$  y *spline4pc*:  $y_{41}(x)$ , y con la solución asintótica  $y(x)$ , en este caso es:

Tabla 5.7. Valores de las soluciones numéricas y asintótica con  $n = 12$  y  $\varepsilon = 10^{-8}$ 

$x$	$y_{31}(x)$	$y_{41}(x)$	$y(x)$
0.0	5.000000e-01	5.000000e-01	5.000000e-01
0.2	5.293047e-01	5.102041e-01	5.102041e-01
0.4	5.434782e-01	5.434783e-01	5.434783e-01
0.6	5.133230e-01	6.097561e-01	6.097561e-01
0.8	7.352942e-01	7.352942e-01	7.352941e-01
1.0	7.385650e-09	1.625149e-05	0.000000e+00

Observamos cómo, en los puntos considerados, la solución numérica que proporciona *spline4pc* es prácticamente la misma que proporciona nuestra solución asintótica de referencia. No ocurre así con la solución obtenida a partir de *spline3pc* (por las fluctuaciones que presenta y que ya hemos comentado con anterioridad).

Las gráficas que resultan al representar gráficamente el comportamiento de las soluciones numéricas frente a la asintótica ponen esto aún más de manifiesto.

Como venimos haciendo en los ejemplos anteriores mostramos en la siguiente página la gráfica de las soluciones, Figura 5.12, y una segunda que amplía su tramo final, Figura 5.13.

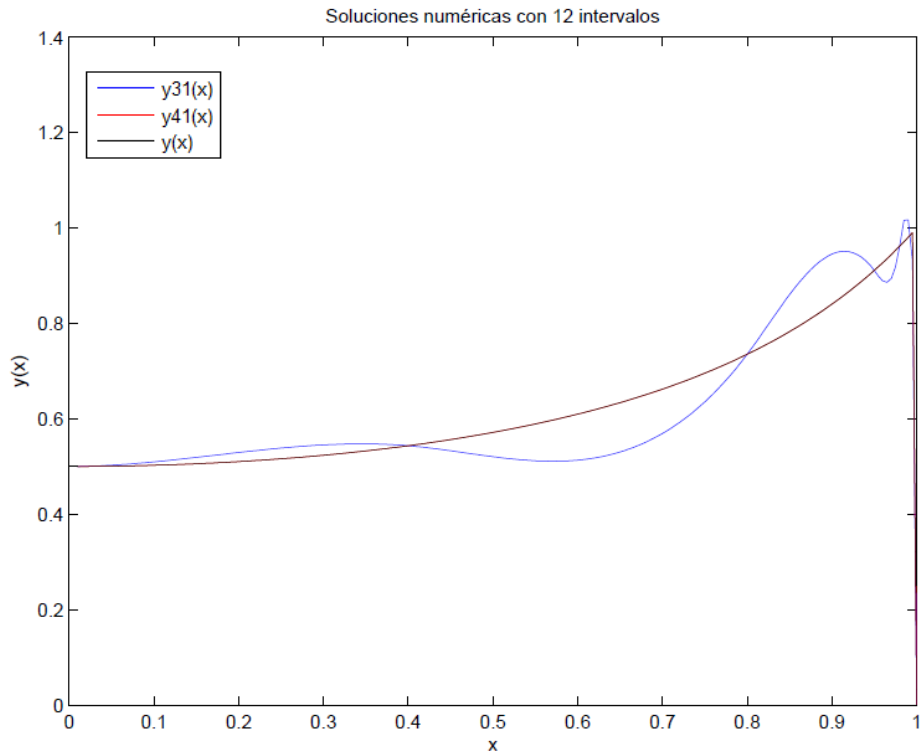


Figura 5.12. Gráficas de las soluciones numéricas y asintótica tomando 12 subintervalos ( $\varepsilon = 10^{-8}$ )

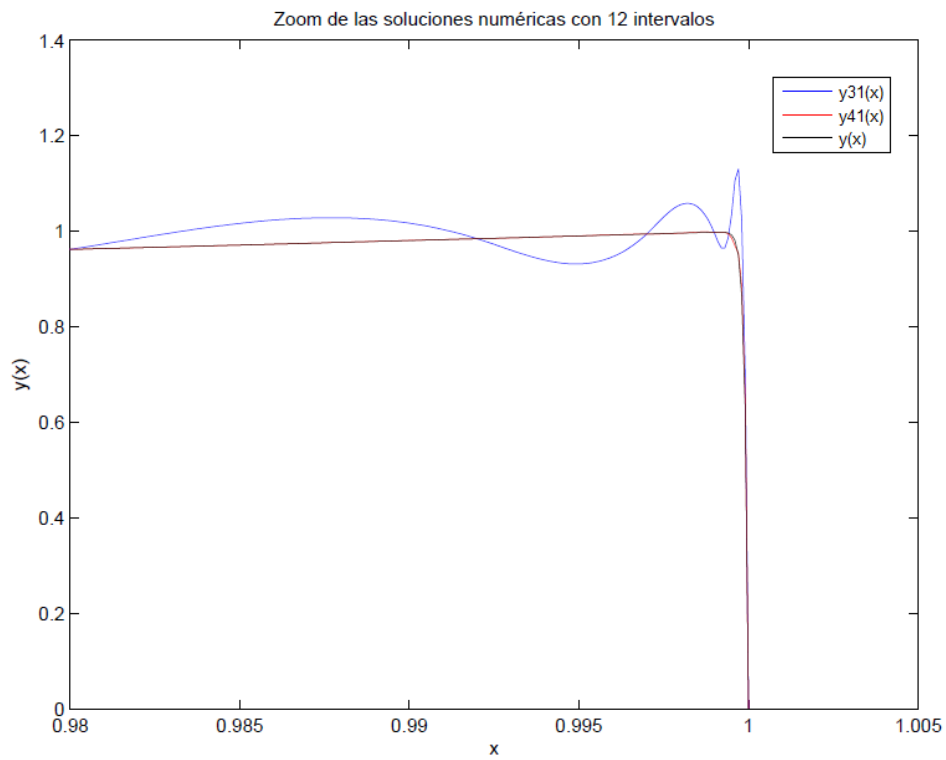


Figura 5.13. Detalle de la gráfica anterior: zoom de las soluciones numéricas y asintótica con 12 subintervalos ( $\varepsilon = 10^{-8}$ )

Como resulta lógico, variando el valor del parámetro también se modifican las aproximaciones a la solución obtenidas mediante los métodos.

Al disminuir el valor de  $\varepsilon$ , puede apreciarse que aumentan las dificultades de integración y hay que tener especial cuidado con el crecimiento de los errores de redondeo. Esto se debe a que, en este problema tan difícil, aparecen coeficientes de tamaños muy dispares y es bien sabido que la suma de magnitudes de tamaño muy diferente da lugar a pérdidas de cifras significativas.

Esto se suma además a las dificultades inherentes ya de por sí al Método del Disparo, que presenta problemas de redondeo en situaciones como ésta en la que la solución varía bruscamente. Procede por tanto recurrir a implementaciones de los métodos que tengan esto en cuenta; pero no lo haremos en esta memoria.

El método *spline4pc* fundamentado en el Método del Disparo y en los Pseudo-Splines Cuárticos, proporciona una aproximación razonable a la solución con un coste computacional asociado muy bajo. Es más, este método que hemos utilizado, obtiene mejores aproximaciones en general que los que se proponen en los artículos de las referencias.

## 5.5. Ejemplo 4 <sup>[29]</sup>

### 5.5.1. Planteamiento del problema

Consideremos el P.V.F. lineal:

$$\begin{cases} u''(x) + 2cxu'(x) + 2cu(x) = 0 \\ u(0) = 1 \\ u(1) = e^{-a} \end{cases} \quad (5.6)$$

Cuya solución exacta es conocida y viene dada por:  $u(x) = e^{-cx^2}$ .

### 5.5.2. Resolución

Para resolver este Problema de Valores de Frontera Lineal, recurrimos a los métodos que venimos considerando en ejemplos anteriores, es decir los que se corresponden con las funciones implementadas en Matlab como: *spline3pc*, *spline3pc2*, *spline4pc* y *spline4pc2*.

El script que hemos utilizado para resolver este problema se puede encontrar en los Anexos en su apartado 7.3.4. Es necesario previamente definir el valor del parámetro  $c$  que vamos a tomar como  $c = 20$  para nuestros experimentos.

### 5.5.3. Resultados

Si consideramos nuestro problema en el intervalo  $x \in [0,1]$  y lo dividimos en 5 subintervalos para su resolución, seleccionando un tamaño de paso fijo  $h = 0.2$ , los resultados de las aproximaciones numéricas obtenidos con los cuatro métodos considerados, así como la solución exacta evaluada en los diferentes puntos del intervalo proporcionan la siguiente tabla:

Tabla 5.8. Valores de las soluciones numéricas y exacta ( $h = 0.2$ )

$x$	$y_{31}(x)$	$y_{32}(x)$	$y_{41}(x)$	$y_{42}(x)$	$y(x)$
0.0	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
0.2	4.972331e-01	4.972331e-01	4.591046e-01	4.591046e-01	4.493290e-01
0.4	2.397672e-02	2.397672e-02	3.859763e-02	3.859763e-02	4.076220e-02
0.6	-1.059661e-04	-1.059661e-04	1.027771e-03	1.027771e-03	7.465858e-04
0.8	-7.386089e-03	-7.386089e-03	5.380593e-05	5.380593e-05	2.760773e-06
1.0	2.061154e-09	2.061154e-09	2.061154e-09	2.061154e-09	2.061154e-09

O bien, si representamos gráficamente las soluciones anteriores:

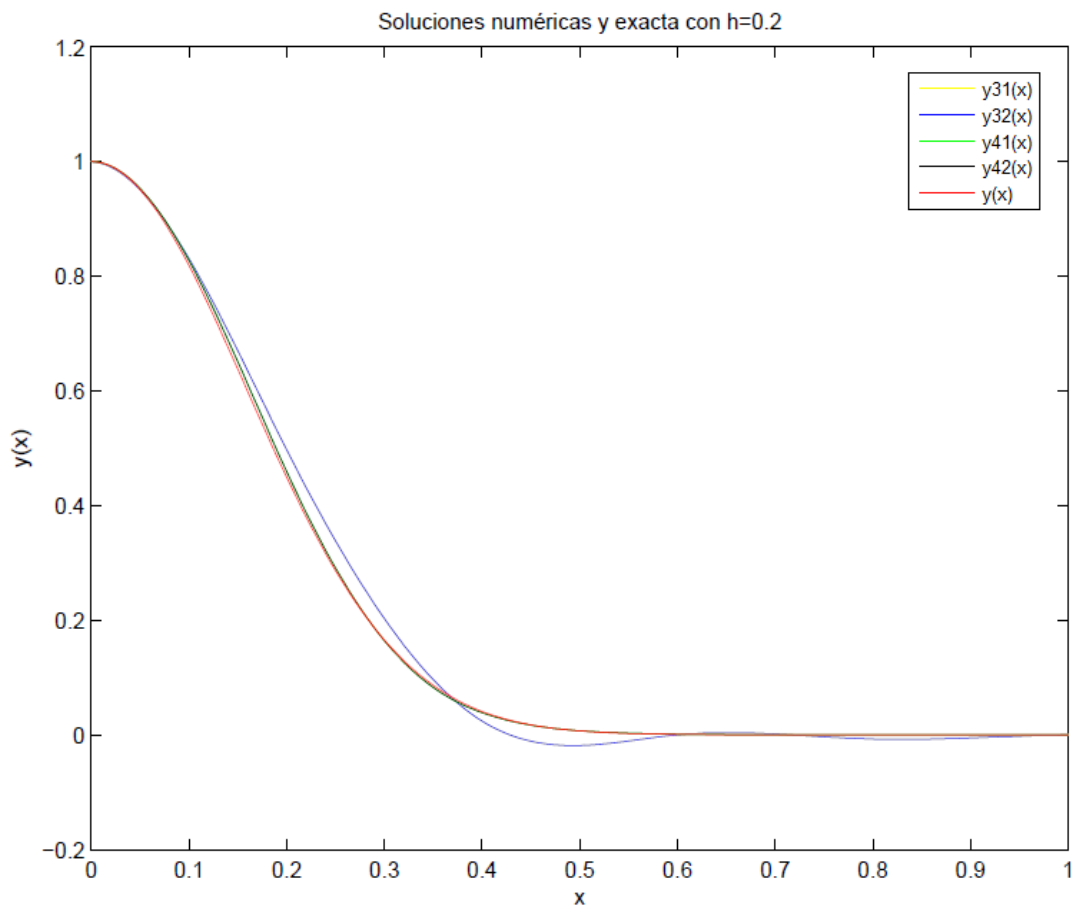


Figura 5.14. Gráfica de las soluciones numéricas y exactas ( $h = 0.2$ )

Podemos observar que las soluciones proporcionadas por los métodos basados en Splines Cúbicos dan lugar a aproximaciones indistinguibles entre sí, pero claramente distinguibles (para el tamaño de paso considerado) de la solución exacta, en tanto que los métodos basados en Pseudo-Splines Cuárticos son indistinguibles de la solución exacta ya para este tamaño de paso.

Para apreciar esto mejor, reproducimos a continuación otra gráfica centrada en un pequeño intervalo final. Nótese que los errores aparecen escalados con un factor  $10^{-4}$ .



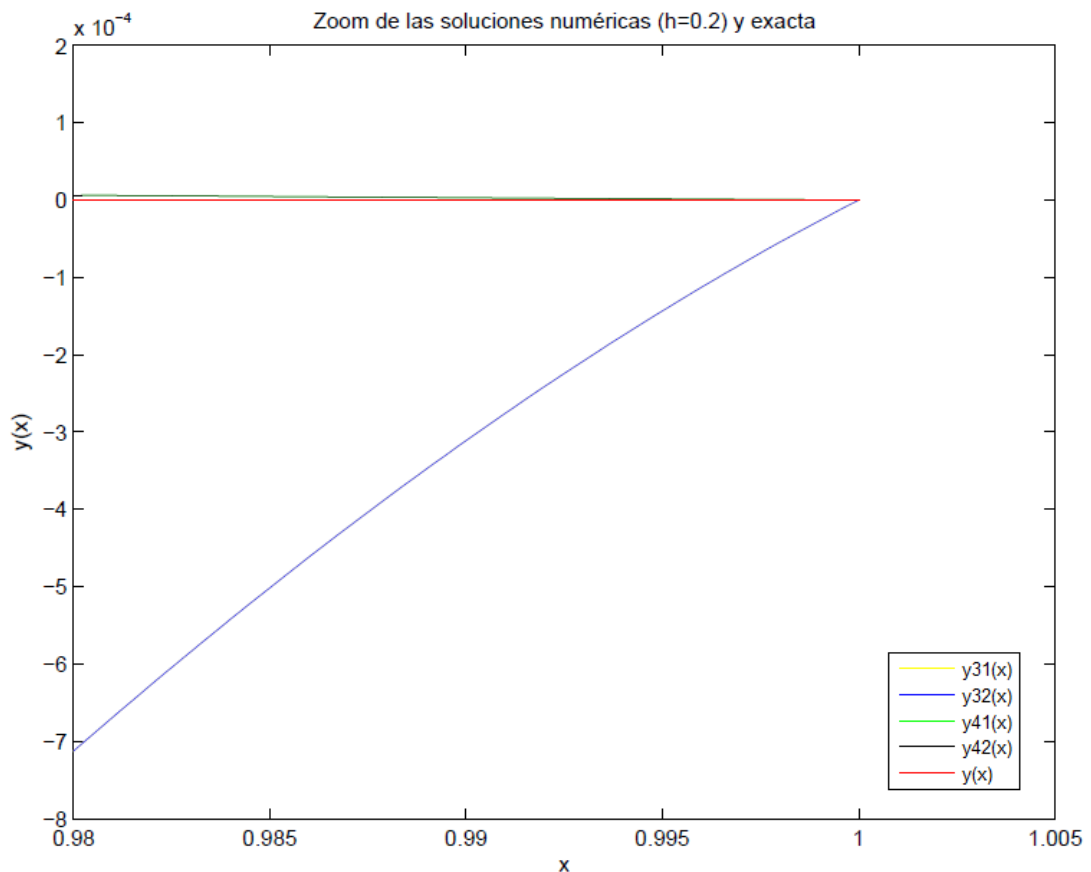
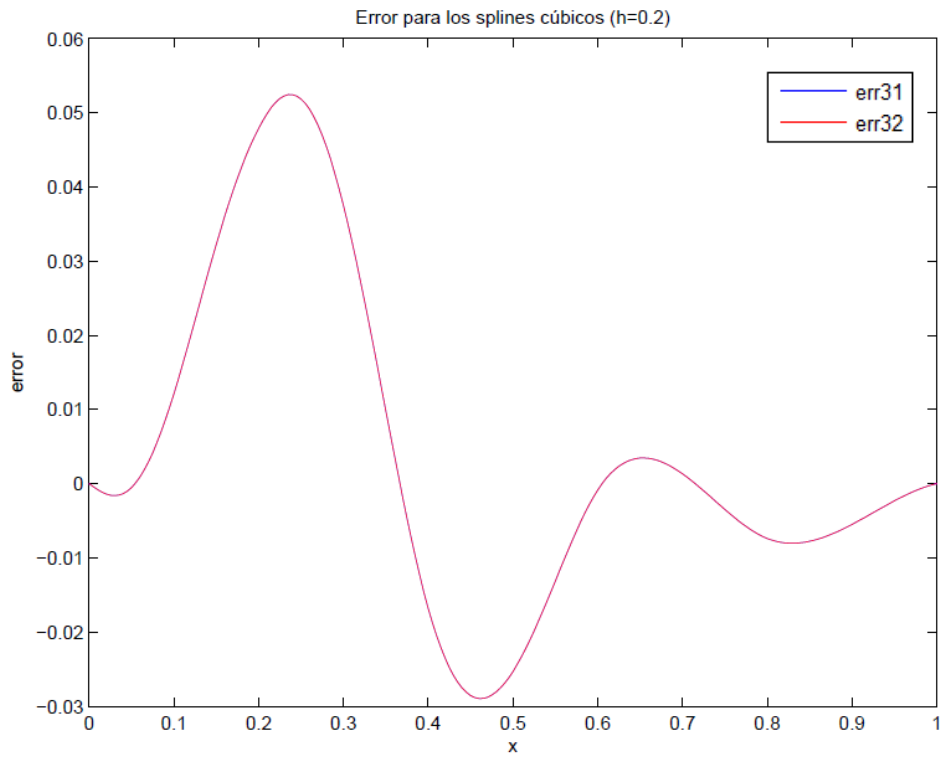
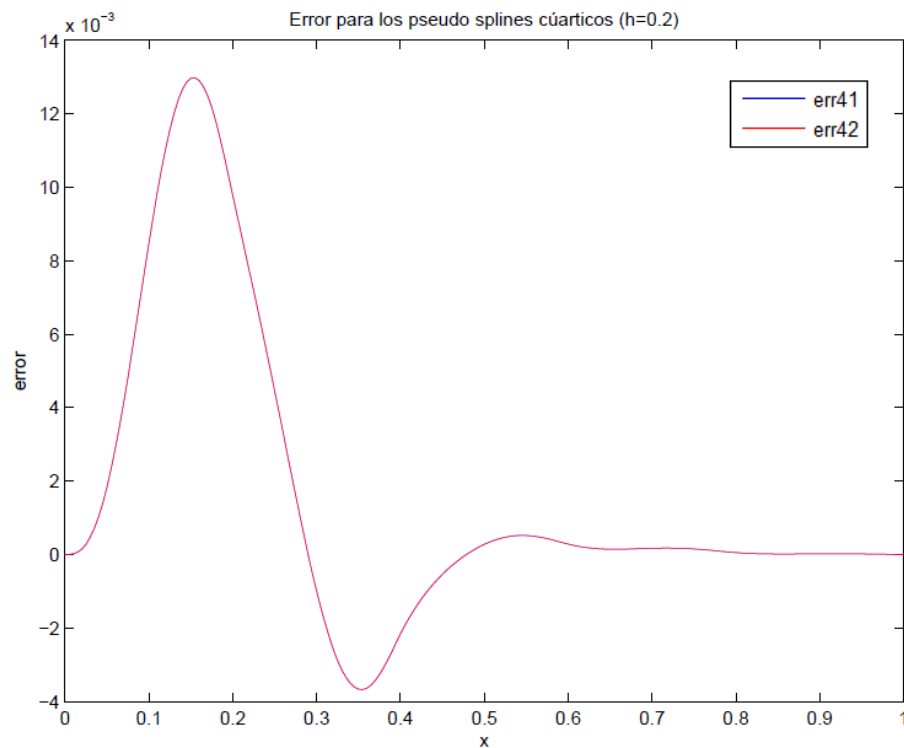


Figura 5.15. Detalle de la gráfica anterior: zoom de las soluciones numéricas y exactas ( $h = 0.2$ )

Con vistas a percibir el error cometido por los métodos en cada punto del intervalo de integración, recogemos a continuación dos gráficas para los errores absolutos (hemos respetado el signo) de los métodos basados en Splines Cúbicos y en Pseudo-Splines Cuárticos respectivamente.

Obsérvese que los errores aparecen escalados con un factor  $10^{-3}$  en la segunda de las gráficas.

Asimismo, es claro que los mayores errores se producen en el tramo inicial del intervalo de integración (que es cuando la solución aún no ha decrecido a casi cero).

Figura 5.16. Errores en las aproximaciones obtenidas mediante Splines Cúbicos ( $h = 0.2$ )Figura 5.17. Errores en las aproximaciones obtenidas con el Pseudo Splines Cúbicos ( $h = 0.2$ )

Se aprecia claramente sin más que observar las escalas, que el error cometido utilizando Pseudo-Splines Cuárticos es notablemente menor que el correspondiente a los Splines Cúbicos.

Además, puede distinguirse que sea cual sea el método empleado, con todos se produce un pico del error cerca del punto  $x = 0.2$ . Por ello, vamos a determinar experimentalmente, del mismo modo que hicimos en el Ejemplo 1, el orden de los métodos en este punto.

Las gráficas en doble escala logarítmica del error resultantes son:

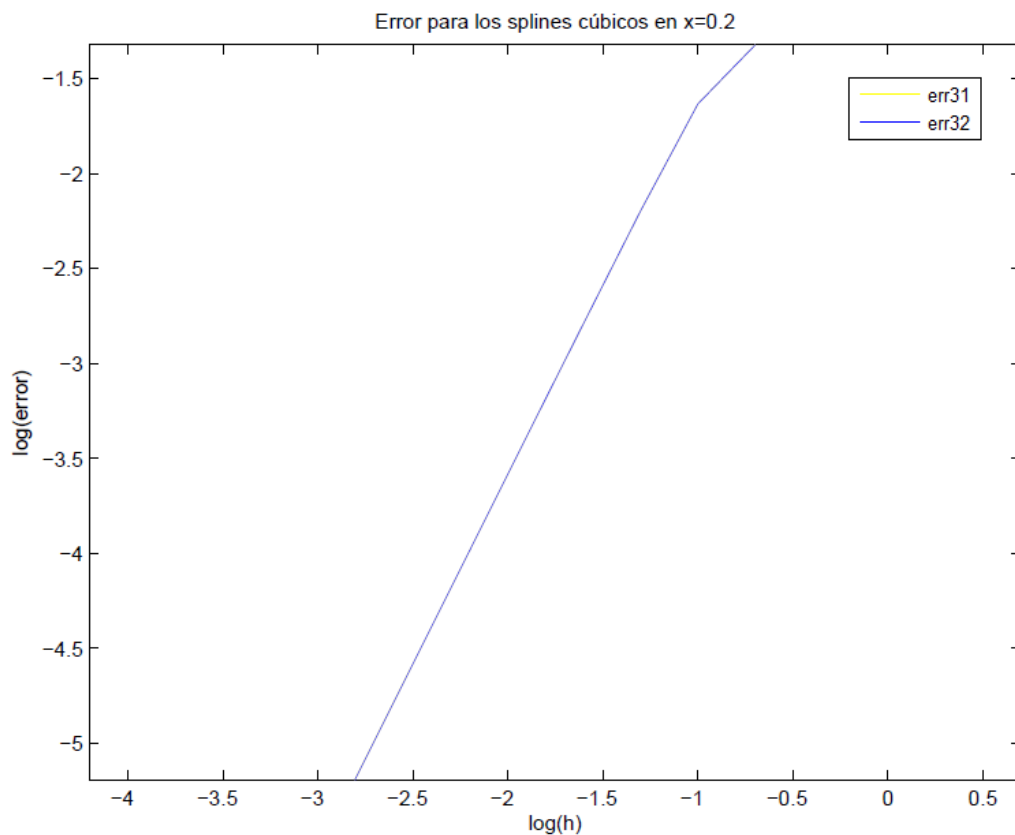


Figura 5.18. Gráfica en doble escala logarítmica de los errores para Splines Cúbicos en el punto  $x = 0.2$ , para los distintos pasos

La pendiente de esta gráfica marca el orden de los métodos *spline3pc* y *spline3pc2*, que hacen uso de Splines Cúbicos, de modo que dichos métodos convergerán con orden 2 (para este problema), pues  $m \cong \frac{-1.6 - (-5.2)}{-1 - (-2.8)} = 2$ .

Conviene observar que, por razones obvias, hemos descartado el dato correspondiente al tamaño de paso  $h = 0.2$  en nuestros cálculos.

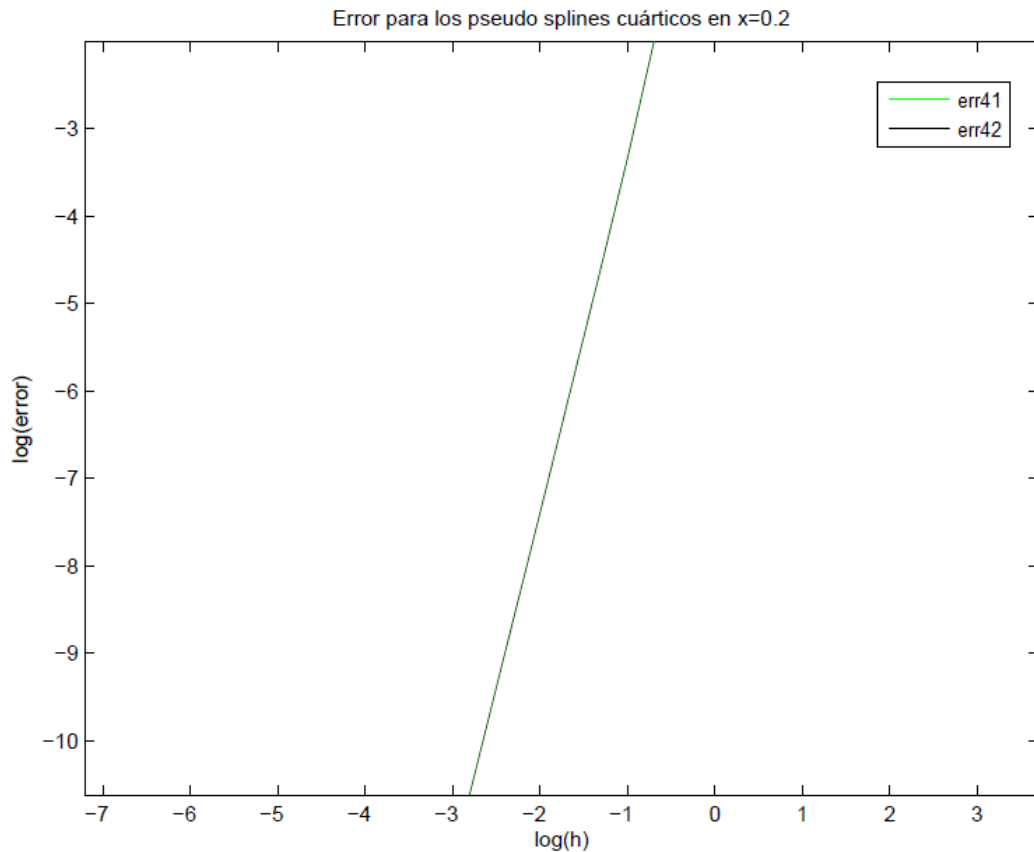


Figura 5.19. Gráfica en doble escala logarítmica de los errores para Splines Cúbicos en el punto  $x = 0.2$ , para los distintos pasos

La pendiente en este caso es:  $m \cong \frac{-2 - (-10.6)}{-0.7 - (-2.8)} = 4.1$ , luego los métodos *spline4pc* y *spline4pc2* basados en Pseudo-Splines Cuárticos muestran un orden de convergencia efectivo 4 (para este problema).

Ambos resultados coinciden con los que ya obtuvimos en el Ejemplo 1.

Finalmente, presentamos en la tabla 5.9 de la siguiente página los errores que se producen en las aproximaciones a la solución exacta, obtenidos con los diferentes métodos en función del tamaño de paso utilizado.

El paso se ha ido variando atendiendo a la expresión:  $h = \frac{0.2}{2^i}$  para valores de  $i = 0, 1, \dots, 7$ .

Tabla 5.9. Errores en las soluciones numéricas ( $h = \frac{0.2}{2^i}, i = 0, 1, \dots, 7$ )

ERRORES CON EL MÉTODO 'spline3pc'								
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$	$i = 7$
0.0	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
0.2	4.79e-02	2.34e-02	6.41e-03	1.64e-03	4.11e-04	1.03e-04	2.57e-05	6.44e-06
0.4	-1.68e-02	-9.54e-04	-2.25e-04	-5.49e-05	-1.36e-05	-3.40e-06	-8.49e-07	-2.12e-07
0.6	-8.53e-04	-6.43e-04	-1.82e-04	-4.67e-05	-1.17e-05	-2.94e-06	-7.35e-07	-1.84e-07
0.8	-7.39e-03	-2.48e-06	-2.05e-06	-6.64e-07	-1.76e-07	-4.48e-08	-1.12e-08	-2.81e-09
1.0	-1.39e-18	-9.55e-23	0.00e+00	-4.14e-25	4.14e-25	0.00e+00	0.00e+00	-4.14e-25
ERRORES CON EL MÉTODO 'spline3pc2'								
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$	$i = 7$
0.0	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
0.2	4.79e-02	2.34e-02	6.41e-03	1.64e-03	4.11e-04	1.03e-04	2.57e-05	6.44e-06
0.4	-1.68e-02	-9.54e-04	-2.25e-04	-5.49e-05	-1.36e-05	-3.40e-06	-8.49e-07	-2.12e-07
0.6	-8.53e-04	-6.43e-04	-1.82e-04	-4.67e-05	-1.17e-05	-2.94e-06	-7.35e-07	-1.84e-07
0.8	-7.39e-03	-2.48e-06	-2.05e-06	-6.64e-07	-1.76e-07	-4.48e-08	-1.12e-08	-2.81e-09
1.0	2.94e-18	-6.91e-23	0.00e+00	-1.24e-24	3.31e-24	2.07e-24	-3.72e-24	-1.65e-24
ERRORES CON EL MÉTODO 'spline4pc'								
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$	$i = 7$
0.0	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
0.2	9.78e-03	4.58e-04	2.63e-05	1.60e-06	9.97e-08	6.22e-09	3.89e-10	2.43e-11
0.4	-2.16e-03	-1.03e-04	-5.82e-06	-3.54e-07	-2.20e-08	-1.37e-09	-8.56e-11	-5.35e-12
0.6	2.81e-04	2.19e-05	1.32e-06	8.14e-08	5.07e-09	3.17e-10	1.98e-11	1.24e-12
0.8	5.10e-05	1.40e-06	6.32e-08	3.68e-09	2.26e-10	1.41e-11	8.78e-13	5.48e-14
1.0	8.95e-20	1.16e-22	4.14e-25	4.14e-25	0.00e+00	0.00e+00	4.14e-25	4.14e-25

ERRORES CON EL MÉTODO 'spline4pc2'								
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$	$i = 7$
0.0	-2.22e-16	-2.22e-16	0.00e+00	-2.22e-16	0.00e+00	-2.22e-16	0.00e+00	-2.22e-16
0.2	9.78e-03	4.58e-04	2.63e-05	1.60e-06	9.97e-08	6.22e-09	3.89e-10	2.43e-11
0.4	-2.16e-03	-1.03e-04	-5.82e-06	-3.54e-07	-2.20e-08	-1.37e-09	-8.56e-11	-5.38e-12
0.6	2.81e-04	2.19e-05	1.32e-06	8.14e-08	5.07e-09	3.17e-10	1.98e-11	1.24e-12
0.8	5.10e-05	1.40e-06	6.32e-08	3.68e-09	2.26e-10	1.41e-11	8.78e-13	5.48e-14
1.0	3.74e-19	3.68e-23	-1.12e-23	4.14e-25	2.07e-24	4.14e-25	4.14e-24	4.14e-25

De las tablas anteriores se pueden extraer los datos que nos han permitido deducir los órdenes de convergencia de nuestros métodos aplicados a este problema.

## 5.6. Ejemplo 5 [33] [34] [36] [37] [38]

### 5.6.1. Planteamiento del problema

En los siguientes dos ejemplos vamos a centrarnos en estudiar problemas que describen la evolución en tiempo de osciladores no lineales. Existen muchas situaciones en las que el movimiento oscilatorio está presente en nuestra vida cotidiana, y por ello este tipo de movimiento es ampliamente estudiado en Ciencias e Ingeniería. Desde una masa suspendida en un muelle elástico, al reloj de péndulo, un circuito eléctrico, o a las oscilaciones en las cuerdas de una guitarra. Las oscilaciones son más comunes de lo que intuitivamente estamos acostumbrados a pensar. Por ejemplo, en la naturaleza forman parte de algunos procesos biológicos tales como la respiración, las palpitations del corazón o las vibraciones de las moléculas.

En este Ejemplo 5, vamos a estudiar concretamente el oscilador de Duffing u oscilador de doble pozo, que es un ejemplo de oscilador no lineal referenciado con frecuencia en la literatura científica dedicada al estudio del caos, soluciones no periódicas, atractores extraños, etc.

Algunas de sus soluciones asemejan un sistema caótico, sensible a pequeñas perturbaciones externas, pese a estar definido por ecuaciones deterministas. Concretamente, el oscilador de doble pozo puede describirse en términos de la ecuación de Duffing:

$$y''(t) + g y'(t) - y(t) + y^3(t) = F \cos(\omega t) \quad (5.7)$$

Esta ecuación carece de solución exacta conocida. De hecho, su estudio ha planteado serias dificultades y continúa siendo ampliamente investigado. No obstante, en el trabajo [33] de la Bibliografía, se presentan una serie de gráficas donde se muestra su comportamiento típico. Usaremos estas gráficas como referencia para comparar nuestros resultados.

### 5.6.2. Resolución

Como puede observarse, la ecuación (5.7) es No Lineal, de modo que para resolver este problema emplearemos las funciones *spline3pnl* y *spline4pnl*, que aproximan soluciones de P.V.I. no lineales mediante Splines Cúbicos y Pseudo-Splines Cuárticos respectivamente. El código de Matlab que implementa la resolución de este problema puede consultarse en los Anexos, en su apartado 7.3.5.

Vamos a realizar tres experimentos numéricos diferentes con este problema, de modo que iremos variando el valor de sus parámetros, así como el intervalo en que consideramos el problema. No obstante, para todos ellos fijaremos el valor de la frecuencia como  $\omega = 1 \frac{\text{rad}}{\text{s}}$ .

### 5.6.3. Resultados

#### a) Ejemplo 5.a

Para el primer experimento se han seleccionado los siguientes valores de los parámetros:  $g = 0.25$  y  $F = 0.22$ . Además, se ha tomado:  $x \in [0, 100]$ , y las condiciones iniciales:

$$z_1 = 1, \quad z_2 = 0 \quad (5.8)$$

Dividiendo el problema en  $n = 250$  subintervalos, es decir tomando un paso fijo  $h = \frac{100}{250} = 0.4$ , las aproximaciones a la solución que obtenemos con los métodos *spline3pnl* y *spline4pnl*, al ser representadas gráficamente, muestran trayectorias que, tras una pequeña etapa transitoria, tienden a una gráfica periódica.

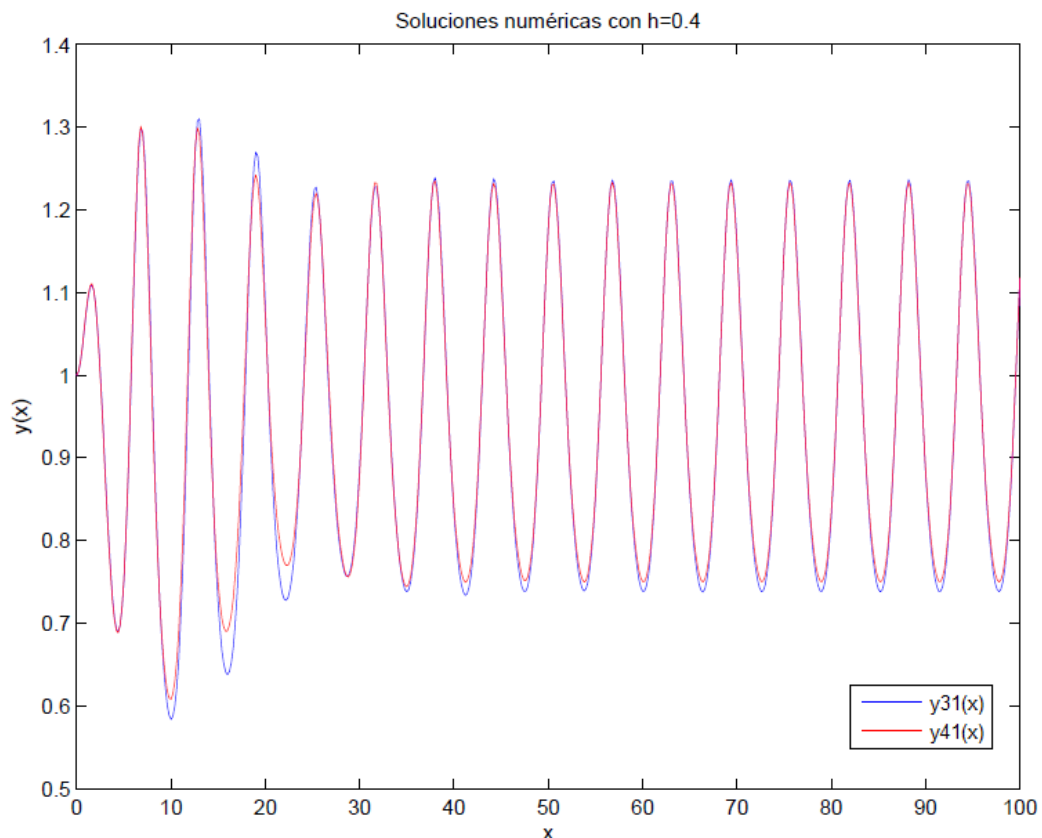


Figura 5.20. Gráfica de las soluciones numéricas ( $h = 0.4$ )

Esto es, inicialmente puede apreciarse un periodo transitorio en las soluciones, que a partir de valores de  $x \cong 30$ , comienza a desaparecer en favor de un periodo estacionario.

Otro aspecto a destacar, es que en la gráfica obtenida a partir del método *spline3pnl* pueden distinguirse pequeños sobrepicos respecto al método *spline4pnl*. Como cabe conjeturar a partir de los experimentos previos realizados, la solución proporcionada por el método *spline4pcnl* es mucho más precisa (como veremos posteriormente).

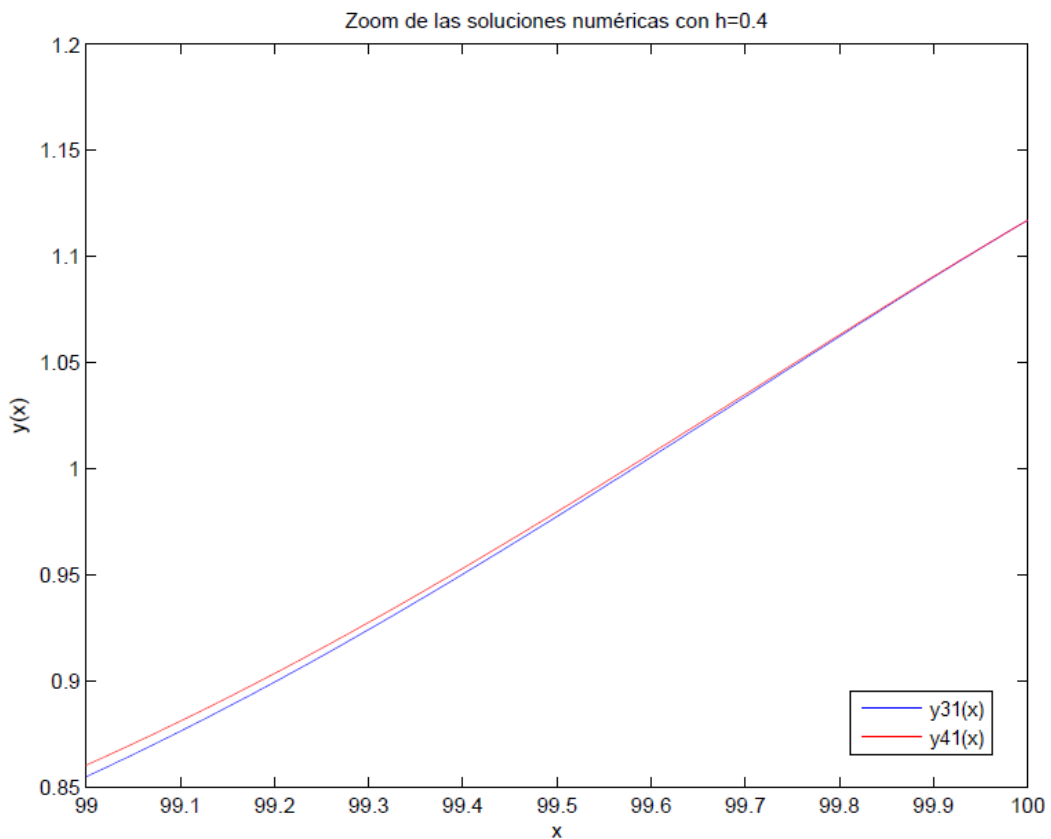
Presentamos a continuación los valores de las aproximaciones a la solución en su tramo final, esto es, se muestran solamente los valores que pertenecen al final del intervalo, es decir  $x = [99,99.8]$ . Nótese que los valores proporcionados por cada método son similares entre sí.



Tabla 5.10. Valores de las soluciones numéricas en  $x = [99, 99.8]$  ( $h = 0.4$ )

$x$	$y_{31}(x)$	$y_{41}(x)$
99.0	8.547070e-01	8.601515e-01
99.2	8.992977e-01	9.033694e-01
99.4	9.501786e-01	9.529193e-01
99.6	1.005343e+00	1.007030e+00
99.8	1.062210e+00	1.062961e+00

De hecho, si ampliamos la gráfica presentada en la Figura 5.21 para apreciar mejor las diferencias entre ambos métodos, observamos que dichas diferencias son relativamente pequeñas.


 Figura 5.21. Detalle de la gráfica anterior: zoom de las soluciones numéricas ( $h = 0.4$ )

Ahora vamos a analizar cómo evolucionan las soluciones obtenidas al ir dividiendo el paso por 2 sucesivamente:  $h = \frac{0.4}{2^i}$ ,  $i = 0, 1, \dots, 4$ . Con este fin, se presentan las siguientes tablas:

Tabla 5.11. Valores de las soluciones numéricas en  $x = [99, 99.8]$  ( $h = \frac{0.4}{2^i}, i = 0, 1, \dots, 4$ )

SOLUCIONES NUMÉRICAS CON EL MÉTODO 'spline3pnl'					
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
99.0	8.547070e-01	8.589277e-01	8.598599e-01	8.600854e-01	8.601413e-01
99.2	8.992977e-01	9.024801e-01	9.031529e-01	9.033138e-01	9.033536e-01
99.4	9.501786e-01	9.523315e-01	9.031529e-01	9.528753e-01	9.529008e-01
99.6	1.005343e+00	1.006689e+00	1.006937e+00	1.006994e+00	1.007008e+00
99.8	1.062210e+00	1.062816e+00	1.062918e+00	1.062940e+00	1.062945e+00
SOLUCIONES NUMÉRICAS CON EL MÉTODO 'spline4pnl'					
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
99.0	8.601515e-01	8.601593e-01	8.601598e-01	8.601599e-01	8.601599e-01
99.2	9.033694e-01	9.033670e-01	9.033669e-01	9.033669e-01	9.033669e-01
99.4	9.529193e-01	9.529099e-01	9.529093e-01	9.529092e-01	9.529092e-01
99.6	1.007030e+00	1.007014e+00	1.007013e+00	1.007013e+00	1.007013e+00
99.8	1.062961e+00	1.062948e+00	1.062947e+00	1.062947e+00	1.062947e+00

De hecho, si representamos la solución que se obtiene para el paso más pequeño considerado:  $h = 0.0125$ , las diferencias entre las soluciones obtenidas con cada método son prácticamente inapreciables.

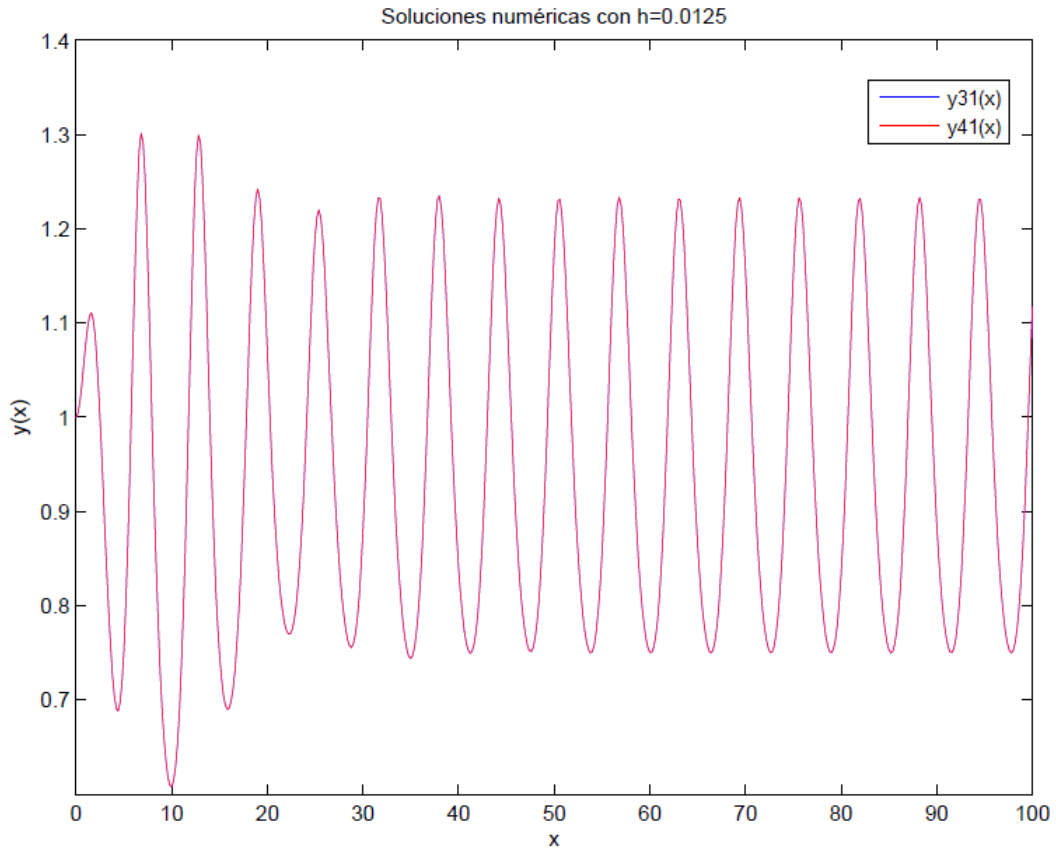


Figura 5.22. Gráfica de las soluciones numéricas ( $h = 0.125$ )

### b) Ejemplo 5.b

Si modificamos ahora el valor de los límites y parámetros elegidos, tomando:  $g = 0.25$  y  $F = 0.25$ , y ampliamos el intervalo considerado para el estudio a  $x \in [0, 200]$ , tomando como condiciones iniciales:

$$z_1 = 0.2, \quad z_2 = 0.1, \quad (5.9)$$

los valores de las soluciones obtenidas también se modifican, como es lógico.

Tomando  $n = 1000$ , es decir para un paso  $h = 0.2$ , la gráfica que determina el comportamiento de las soluciones toma la forma de la Figura 5.23 mostrada en la página siguiente.

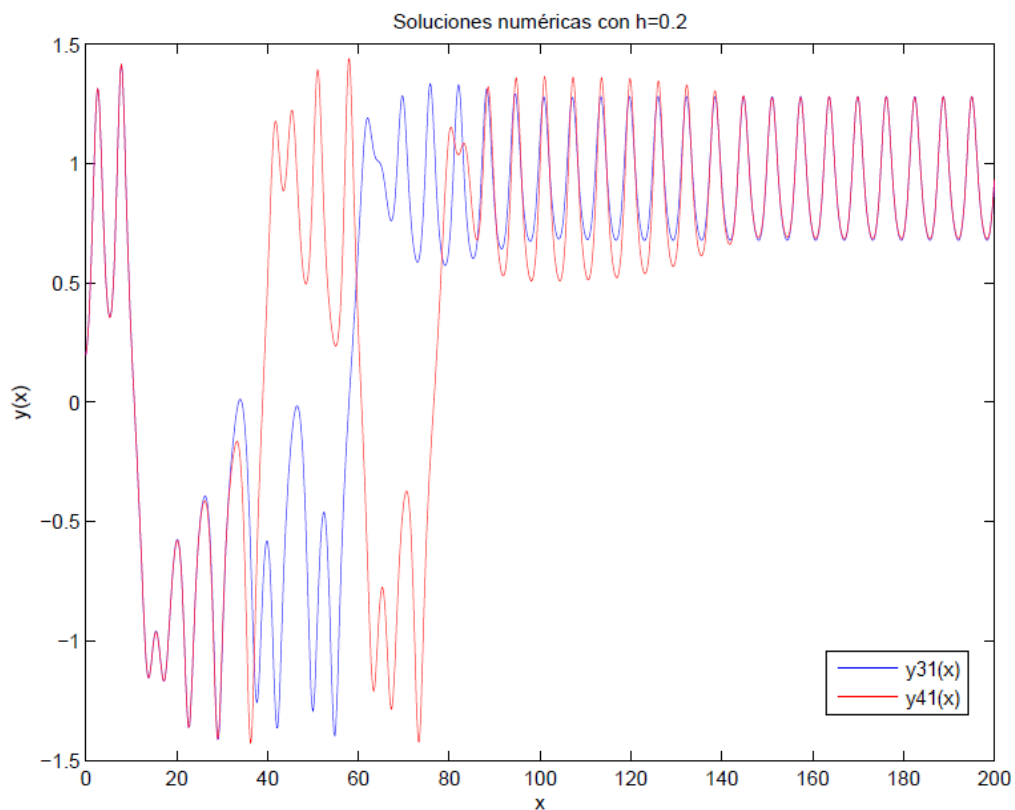


Figura 5.23. Gráfica de las soluciones numéricas ( $h = 0.2$ )

Destaca el hecho de que, inicialmente, la gráfica obtenida a partir de la función *spline3pnl*, es decir aquella correspondiente al método basado en Splines Cúbicos, se ‘desfasa’ o pierde respecto a la correspondiente a la del método *spline4pnl* (que es ya bastante precisa para este tamaño de paso), en un pequeño intervalo, aunque después recupera su comportamiento al acercarse a la fase periódica.

Además, se aprecia que el periodo transitorio en este caso, es mucho más largo y "complicado", llegando hasta valores de  $x \cong 140$ .

Tanto en este caso como en el anterior, las gráficas que hemos obtenido son similares a las proporcionadas en la referencia bibliográfica, para los mismos valores de los parámetros, lo que nos indica que los métodos reproducen bien las soluciones de este tipo de problemas.

En el tramo final del intervalo, ambas soluciones se comportan de modo muy similar, como puede apreciarse en la gráfica y en la tabla que se muestran a continuación.

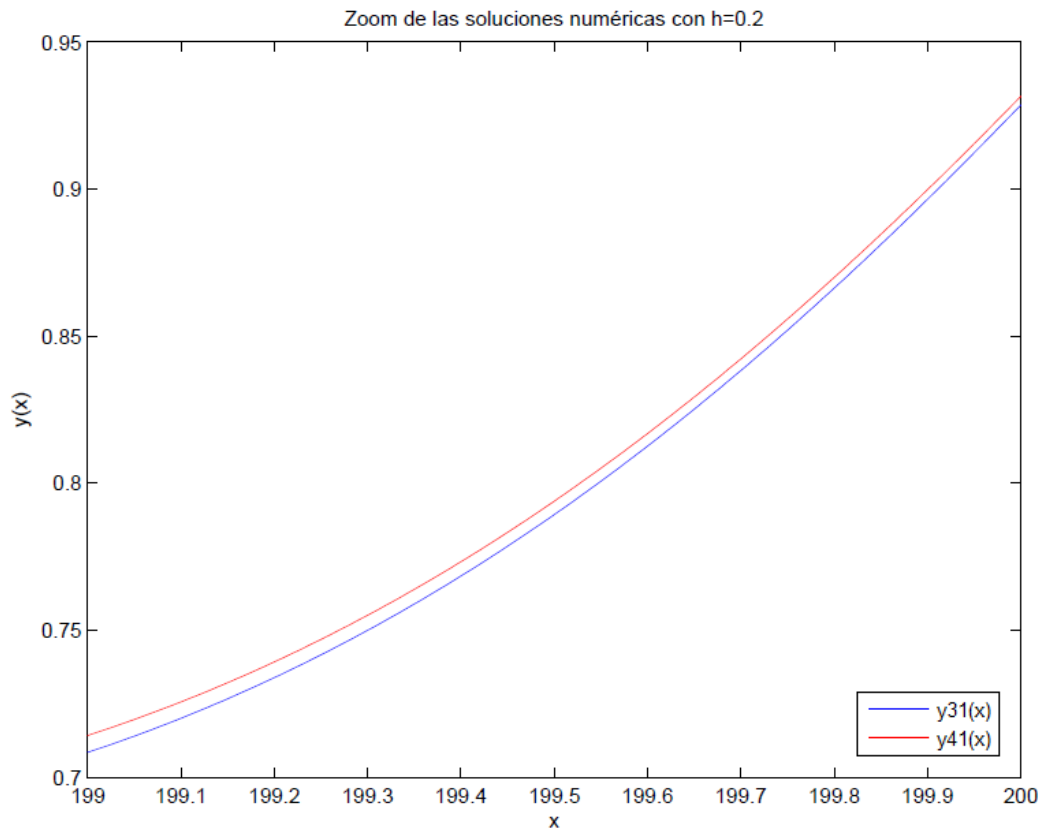


Figura 5.24. Detalle de la gráfica anterior: zoom de las soluciones numéricas ( $h = 0.2$ )

Tabla 5.12. Valores de las soluciones numéricas en  $x = [199,199.8]$  ( $h = 0.2$ ):

$x$	$y_{31}(x)$	$y_{41}(x)$
199.0	7.084074e-01	7.141733e-01
199.2	7.338354e-01	7.391891e-01
199.4	7.683941e-01	7.732497e-01
199.6	8.125706e-01	8.168505e-01
199.8	8.662801e-01	8.699178e-01

Si variamos el tamaño de paso en este caso, al igual que ocurría en el anterior, las soluciones que se obtienen con ambos métodos cada vez son más parecidas, como puede apreciarse en las siguientes tablas.

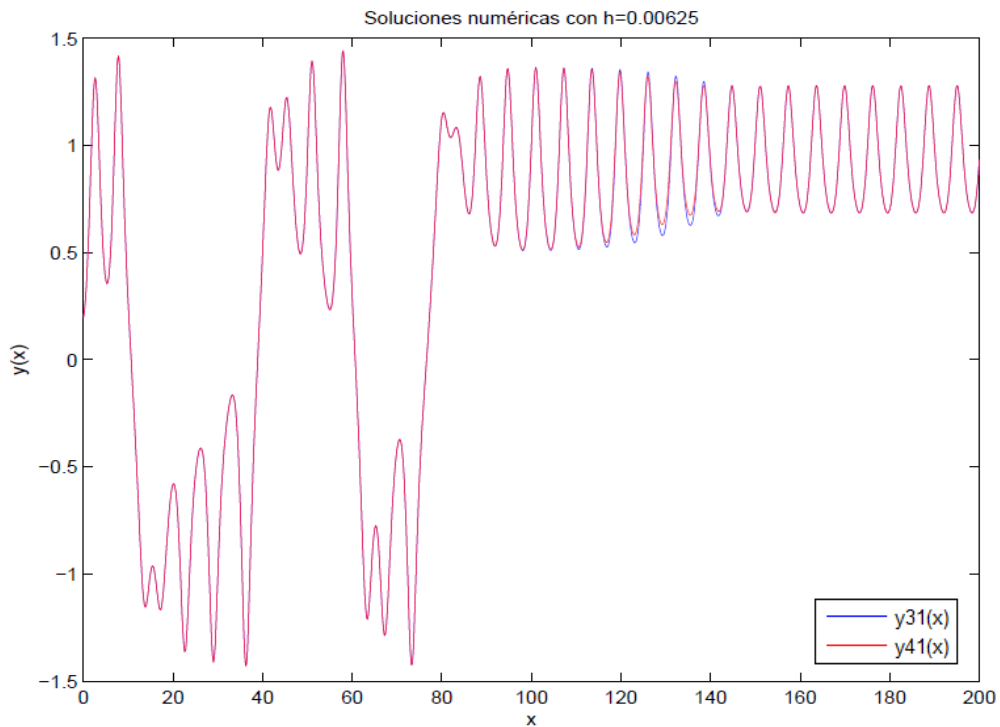
Tabla 5.13. Valores de las soluciones numéricas en  $x = [199, 199.8]$  ( $h = \frac{0.2}{2^i}$ ,  $i = 0, 1, \dots, 4$ )

SOLUCIONES NUMÉRICAS CON EL MÉTODO 'spline3pnl'					
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
199.0	7.084074e-01	-1.130471e+00	-7.873925e-01	7.141135e-01	7.141609e-01
199.2	7.338354e-01	-1.066647e+00	-9.367663e-01	7.391349e-01	7.391782e-01
199.4	7.683941e-01	-1.002896e+00	-1.085640e+00	7.732017e-01	7.732403e-01
199.6	8.125706e-01	-9.427963e-01	-1.220172e+00	8.168092e-01	8.168426e-01
199.8	8.662801e-01	-8.885507e-01	-1.323748e+00	8.698834e-01	8.699114e-01

SOLUCIONES NUMÉRICAS CON EL MÉTODO 'spline4pnl'					
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
199.0	7.141733e-01	7.141944e-01	7.141961e-01	7.141962e-01	7.141963e-01
199.2	7.391891e-01	7.392104e-01	7.392121e-01	7.392122e-01	7.392122e-01
199.4	7.732497e-01	7.732706e-01	7.732721e-01	7.732722e-01	7.732721e-01
199.6	8.168505e-01	8.168704e-01	8.168716e-01	8.168717e-01	8.168717e-01
199.8	8.699178e-01	8.699360e-01	8.699370e-01	8.699370e-01	8.699370e-01

De hecho, para un paso  $h = 0.0625$  las diferencias gráficas son casi inapreciables, salvo en algunos "picos":


 Figura 5.25. Gráfica de las soluciones numéricas ( $h = 0.0625$ )

### c) Ejemplo 5.c

Por último, vamos a considerar el caso en que aumenta aún más la fuerza:  $g = 0.25$ ,  $F = 0.4$ , manteniendo las restantes condiciones consideradas en un apartado anterior, esto es:  $z_1 = 0.2$ ,  $z_2 = 0.1$ , en el intervalo  $x \in [0, 100]$ .

En vista de los resultados que se han obtenido anteriormente, vamos a tomar un tamaño de paso inicial más pequeño que en los otros ejemplos, concretamente  $h = 0.1$ . Así obtenemos:

Tabla 5.14. Valores de las soluciones numéricas en  $x = [99, 99.8]$  ( $h = 0.1$ )

$x$	$y_{31}(x)$	$y_{41}(x)$
99.0	-2.919108e-01	-9.146368e-01
99.2	-4.002074e-01	-8.424019e-01
99.4	-5.125312e-01	-7.794289e-01
99.6	-6.273381e-01	-7.247614e-01
99.8	-7.418978e-01	-6.768416e-01

Vemos que, en este caso, las soluciones que proporciona cada método varían significativamente. Esto se aprecia mejor gráficamente:

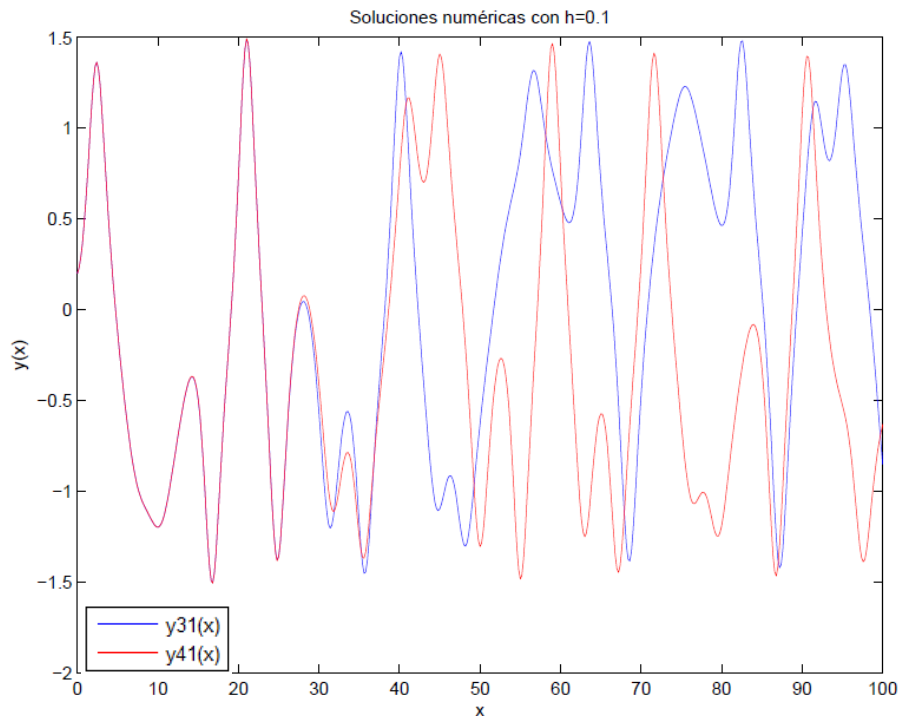


Figura 5.26. Gráfica de las soluciones numéricas ( $h = 0.1$ )

Observamos que en el primer tramo del intervalo ambos métodos proporcionan soluciones similares; pero, a partir de valores de  $x \cong 28$ , las soluciones se distancian progresivamente.

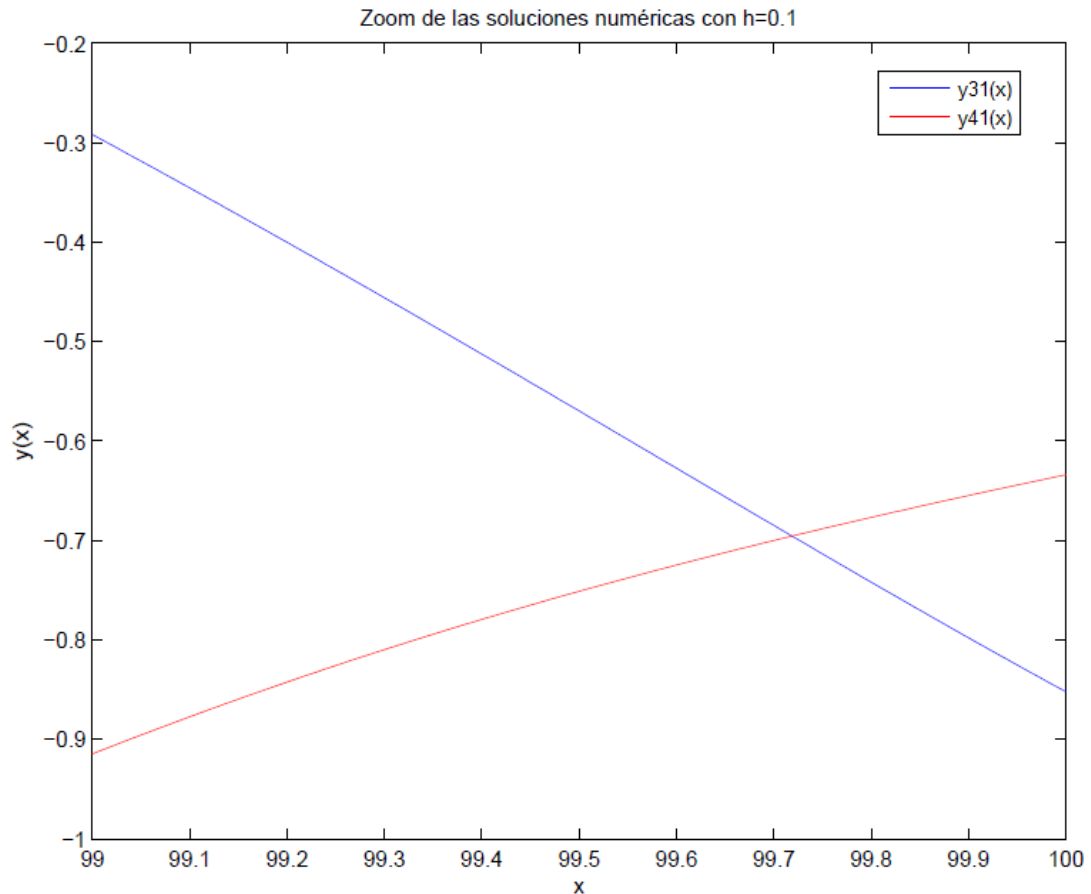


Figura 5.27. Detalle de la gráfica anterior: zoom de las soluciones numéricas ( $h = 0.1$ )

Incluso disminuyendo aún más el tamaño de paso hasta uno bastante más pequeño, para valores tan altos de la fuerza  $F$  las soluciones que se obtienen, aunque son casi coincidentes en un intervalo cada vez más amplio de valores, siguen diferenciándose a partir de cierto valor.

Por ejemplo, para un tamaño de paso  $h = 0.003125$ , las soluciones tienden a distanciarse a partir de  $x \cong 60$ , como se aprecia en la siguiente gráfica.



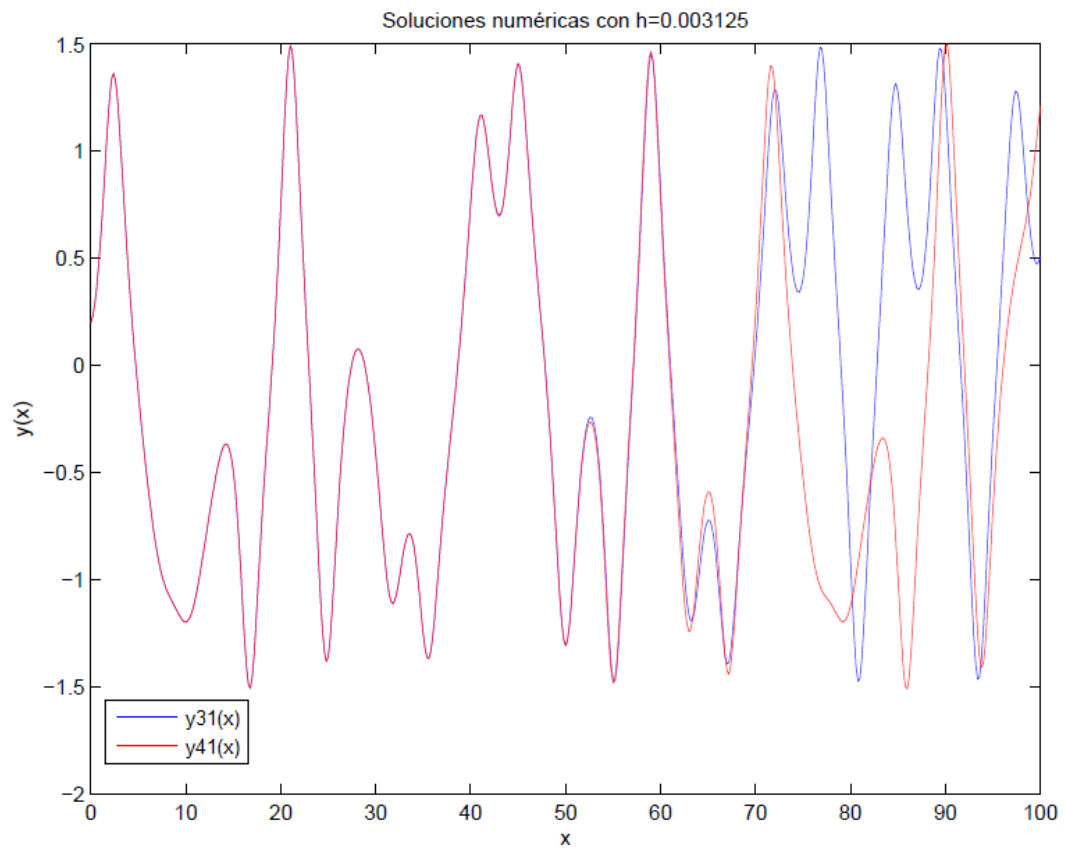


Figura 5.28. Gráfica de las soluciones numéricas ( $h = 0.003125$ )

Esto también es evidente a la vista de las siguientes tablas de las soluciones numéricas para ambos métodos:

Tabla 5.15. Valores de las soluciones numéricas en  $x = [99, 99.8]$  ( $h = \frac{0.4}{2^i}, i = 0, 1, \dots, 4$ )

SOLUCIONES NUMÉRICAS CON EL MÉTODO 'spline3pnl'					
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
99.0	-2.919108e-01	-2.106520e-01	-9.052675e-01	7.668309e-01	6.018780e-01
99.2	-4.002074e-01	-3.045591e-01	-1.086681e+00	7.189870e-01	5.369897e-01
99.4	-5.125312e-01	-4.009204e-01	-1.247330e+00	6.906817e-01	4.938603e-01
99.6	-6.273381e-01	-4.990097e-01	-1.366384e+00	6.844494e-01	4.740469e-01
99.8	-7.418978e-01	-5.975456e-01	-1.424748e+00	7.019797e-01	4.788611e-01

SOLUCIONES NUMÉRICAS CON EL MÉTODO 'spline4pnl'					
x	i = 0	i = 1	i = 2	i = 3	i = 4
99.0	-9.146368e-01	9.883682e-01	8.289623e-01	8.140529e-01	8.130979e-01
99.2	-8.424019e-01	1.036928e+00	8.984318e-01	8.840782e-01	8.831530e-01
99.4	-7.794289e-01	1.083733e+00	9.748365e-01	9.617844e-01	9.609360e-01
99.6	-7.247614e-01	1.127475e+00	1.055868e+00	1.044971e+00	1.044253e+00
99.8	-6.768416e-01	1.166437e+00	1.137341e+00	1.129461e+00	1.128931e+00

Las dos últimas columnas de la tabla de soluciones numéricas para el método *spline4pnl* muestran variaciones menores que una milésima en los puntos considerados, lo que nos permite conjeturar que la última aproximación obtenida a la solución es ya bastante buena. De hecho, nuevos experimentos con pasos aún más pequeños y comparaciones con resultados obtenidos con otros métodos, permiten confirmar nuestras sospechas.

## 5.7. Ejemplo 6 [33] [34] [36] [37] [38]

### 5.7.1. Planteamiento del problema

En este ejemplo vamos a considerar otro tipo de oscilador No Lineal, concretamente el Oscilador de Van del Pol, que describe un sistema dinámico que incluye retroalimentación positiva y un elemento resistivo no lineal.

En su aplicación original, el oscilador eléctrico con un elemento no lineal se utilizó como precursor de las primeras radios comerciales, pues un circuito de este tipo favorece las oscilaciones pequeñas y amortigua las grandes.

En el ámbito de los sistemas dinámicos, el oscilador de Van der Pol es un ejemplo de oscilador con amortiguamiento no lineal cuya evolución temporal viene dada en términos de la siguiente ecuación:

$$y''(t) - \mu (1 - y^2(t)) y'(t) + y(t) = 0 \quad (5.10)$$

De este tipo de oscilador tampoco se conoce su solución exacta, de modo que, de nuevo, tomaremos como referencia las gráficas del trabajo [33] incluido en la Bibliografía.

### 5.7.2. Resolución

De nuevo, este Problema de Valores Iniciales viene dado por un oscilador de tipo No Lineal, por lo que volveremos a hacer uso de los métodos implementados como *spline3pnl* y *spline4pnl*, basados en Splines Cúbicos y Pseudo-Splines Cuárticos respectivamente. El código correspondiente a este ejemplo puede encontrarse en el apartado 7.3.6 del Anexo.

Vamos a hacer experimentos variando el valor que toma el parámetro  $\mu$ . Para todos ellos consideraremos el intervalo en el que se define el problema como  $x \in [0, 100]$  y las condiciones iniciales dadas por:

$$z_1 = 0.1, \quad z_2 = 0 \quad (5.11)$$

### 5.7.3. Resultados

#### a) Ejemplo 6.a

Comenzamos eligiendo un valor pequeño para el parámetro  $\mu$ , concretamente  $\mu = 0.2$  y un tamaño de paso relativamente grande,  $h = 0.4$ . En este supuesto, la gráfica que identifica el comportamiento de las soluciones toma la siguiente forma:

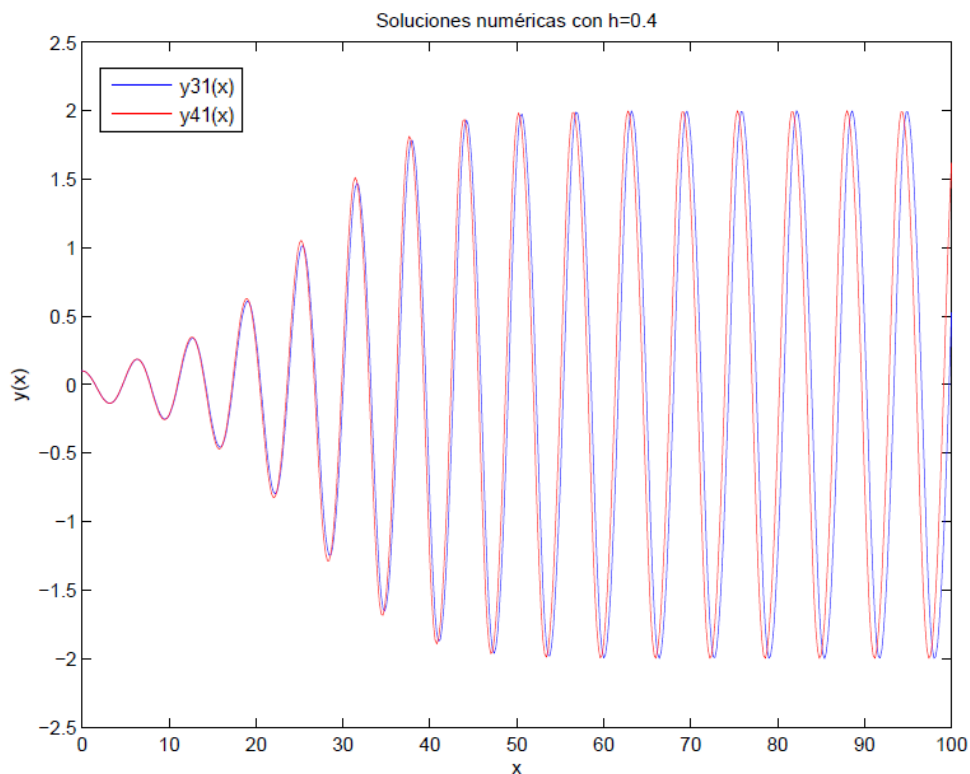


Figura 5.29. Gráfica de las soluciones numéricas para  $\mu = 0.2$  ( $h = 0.4$ )

De nuevo, se aprecia un pequeño transitorio inicial, que desaparece dando paso a un régimen estacionario periódico a partir de valores de  $x \cong 50$ .

También se detecta que progresivamente las soluciones que proporciona cada método se desacoplan, de modo que, aunque tienen la misma forma, la correspondiente a la aproximación mediante Splines Cúbicos parece retrasarse ligeramente.

Estas diferencias se aprecian mejor en la siguiente figura centrada en el último tramo del intervalo de integración:

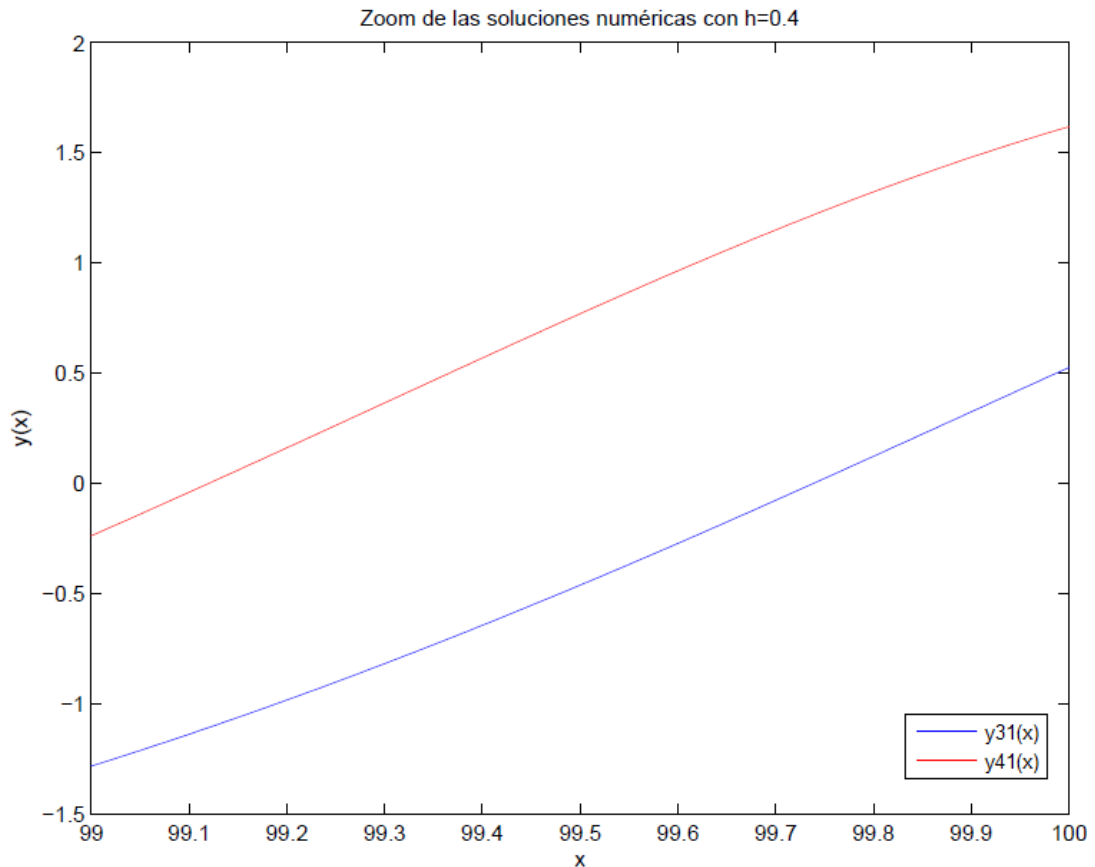


Figura 5.30. Detalle de la gráfica anterior: zoom de las soluciones numéricas,  $\mu = 0.2$  ( $h = 0.4$ )

El valor numérico de dichas soluciones en el tramo final del intervalo considerado se muestra a continuación en la tabla 5.16.

Tabla 5.16. Valores de las soluciones numéricas en  $x = [99, 99.8]$  para  $\mu = 0.2$  ( $h = 0.4$ )

$x$	$y_{31}(x)$	$y_{41}(x)$
99.0	-1.284664e+00	-2.388149e-01
99.2	-9.846712e-01	1.602786e-01
99.4	-6.453568e-01	5.681623e-01
99.6	-2.735474e-01	9.639289e-01
99.8	1.218064e-01	1.322204e+00

Si estudiamos la evolución de las soluciones al disminuir el tamaño de paso del modo que venimos haciendo habitualmente, en este mismo intervalo obtenemos la tabla que se expone a continuación:

 Tabla 5.17. Valores de las soluciones numéricas en  $x = [99, 99.8]$  para  $\mu = 0.2$  ( $h = \frac{0.4}{2^i}, i = 0, 1, \dots, 4$ )

SOLUCIONES NUMÉRICAS CON EL MÉTODO 'spline3pnl'					
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
99.0	-1.284664e+00	-5.340753e-01	-3.138233e-01	-2.570237e-01	-2.427222e-01
99.2	-9.846712e-01	-1.528526e-01	8.146217e-02	1.411652e-01	1.561493e-01
99.4	-6.453568e-01	2.488955e-01	4.888766e-01	5.490479e-01	5.640835e-01
99.6	-2.735474e-01	6.549336e-01	8.885838e-01	9.458873e-01	9.601204e-01
99.8	1.218064e-01	1.043329e+00	1.255926e+00	1.306509e+00	1.318970e+00
SOLUCIONES NUMÉRICAS CON EL MÉTODO 'spline4pnl'					
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
99.0	-2.388149e-01	-2.380042e-01	-2.379499e-01	-2.379464e-01	-2.379462e-01
99.2	1.602786e-01	1.610896e-01	1.611451e-01	1.611486e-01	1.611489e-01
99.4	5.681623e-01	5.690336e-01	5.690905e-01	5.690941e-01	5.690943e-01
99.6	9.639289e-01	9.647935e-01	9.648520e-01	9.648558e-01	9.648560e-01
99.8	1.322204e+00	1.323045e+00	1.323103e+00	1.323107e+00	1.323107e+00

Los resultados que proporciona cada método, a medida que disminuye el tamaño de paso, toman valores cada vez más parecidos.

Las dos últimas columnas de la tabla de soluciones numéricas para el método *spline4pnl* muestran variaciones menores que tres diezmilésimas en los puntos considerados, lo que nos permite aventurar que la última aproximación obtenida a la solución es ya muy precisa.

De hecho, para el tamaño de paso más pequeño considerado, esto es, para  $h = 0.0125$ , las gráficas de ambas soluciones apenas se diferencian entre sí:

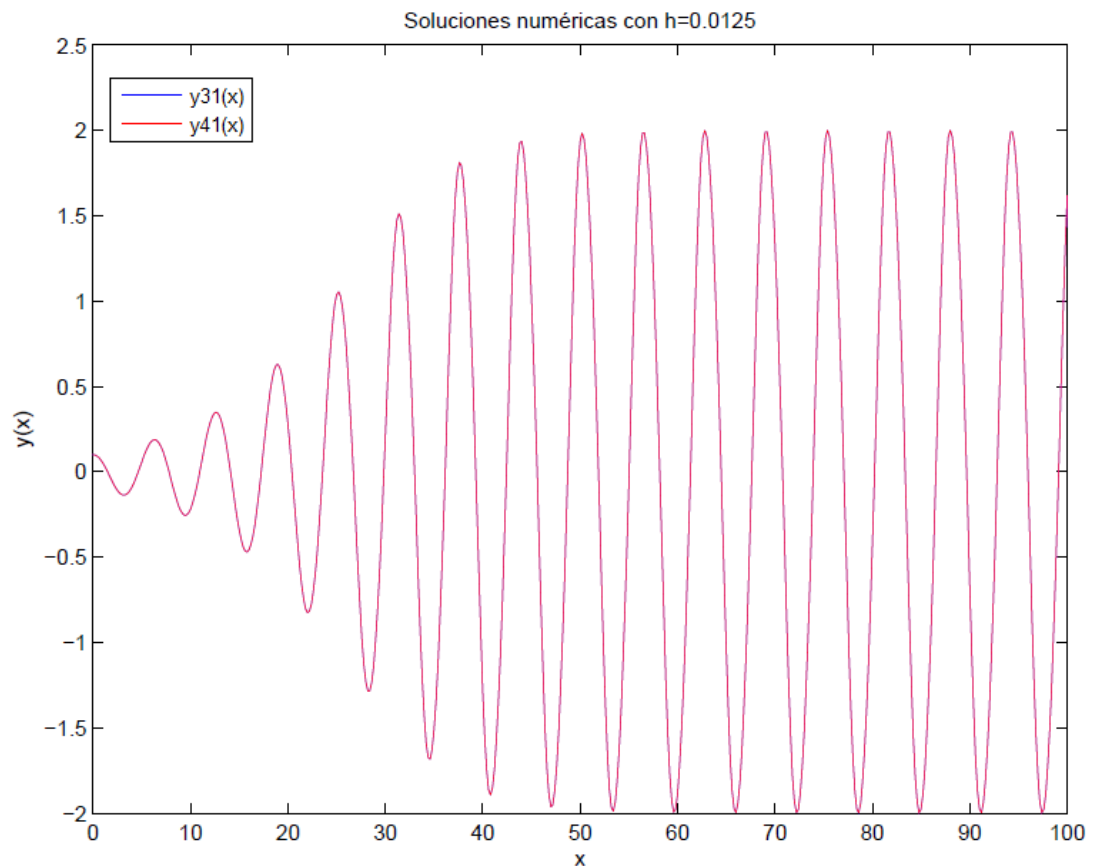


Figura 5.31. Gráfica de las soluciones numéricas para  $\mu = 0.2$  ( $h = 0.0125$ )

### b) Ejemplo 6.b

Vamos a analizar ahora el caso en el que el parámetro  $\mu$  es más elevado. Concretamente, tomaremos  $\mu = 1$ . El resto de los valores: las condiciones iniciales y el paso, los mantenemos.

Las soluciones proporcionadas por los métodos *spline3pnl* y *spline4pnl* muestran cómo ha afectado a las soluciones esta variación del parámetro.

En la siguiente gráfica representamos las soluciones numéricas obtenidas en el intervalo habitual:

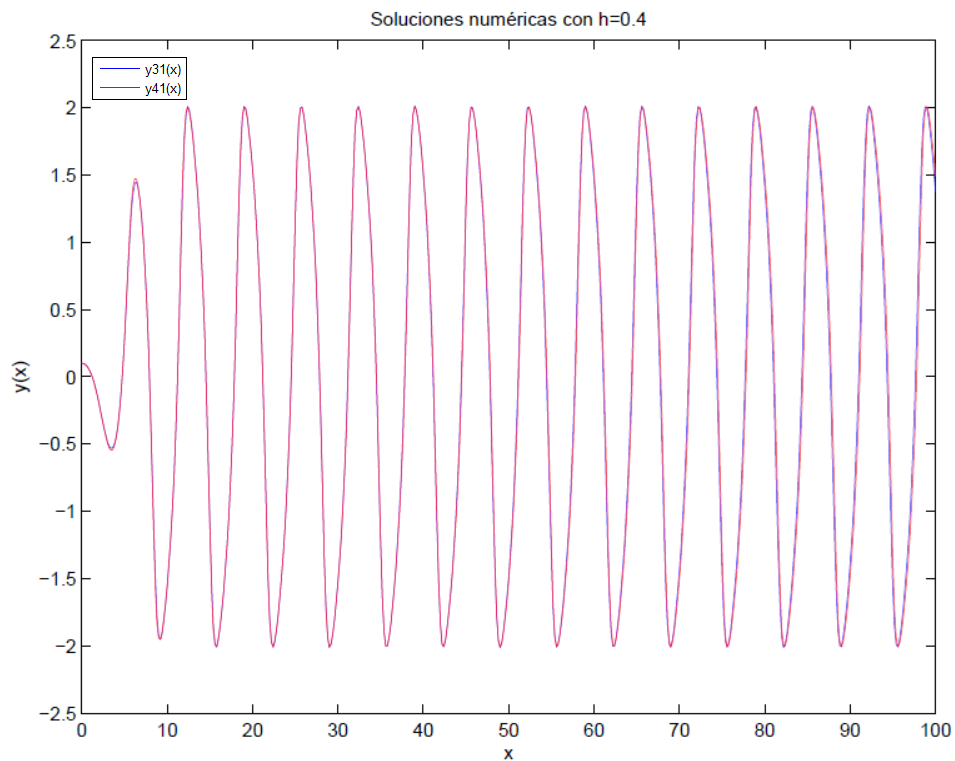


Figura 5.32. Gráfica de las soluciones numéricas para  $\mu = 1$  ( $h = 0.04$ )

El comportamiento de las soluciones varía ligeramente respecto al caso en que  $\mu = 0.2$ . Se observa ahora que el periodo transitorio es notablemente menor en este caso, y que las aproximaciones difieren menos entre sí, incluso al final del intervalo

Aun así, si hacemos zoom en el trozo final de nuestro intervalo de integración, sí se aprecian ligeras diferencias. Veámoslo en la siguiente tabla y en la siguiente gráfica:

Tabla 5.18. Valores de las soluciones numéricas en  $x = [99, 99.8]$  para  $\mu = 1$  ( $h = 0.4$ )

$x$	$y_{31}(x)$	$y_{41}(x)$
99.0	1.996112e+00	2.008445e+00
99.2	1.921785e+00	1.965199e+00
99.4	1.815690e+00	1.880564e+00
99.6	1.687959e+00	1.770236e+00
99.8	1.542025e+00	1.640281e+00

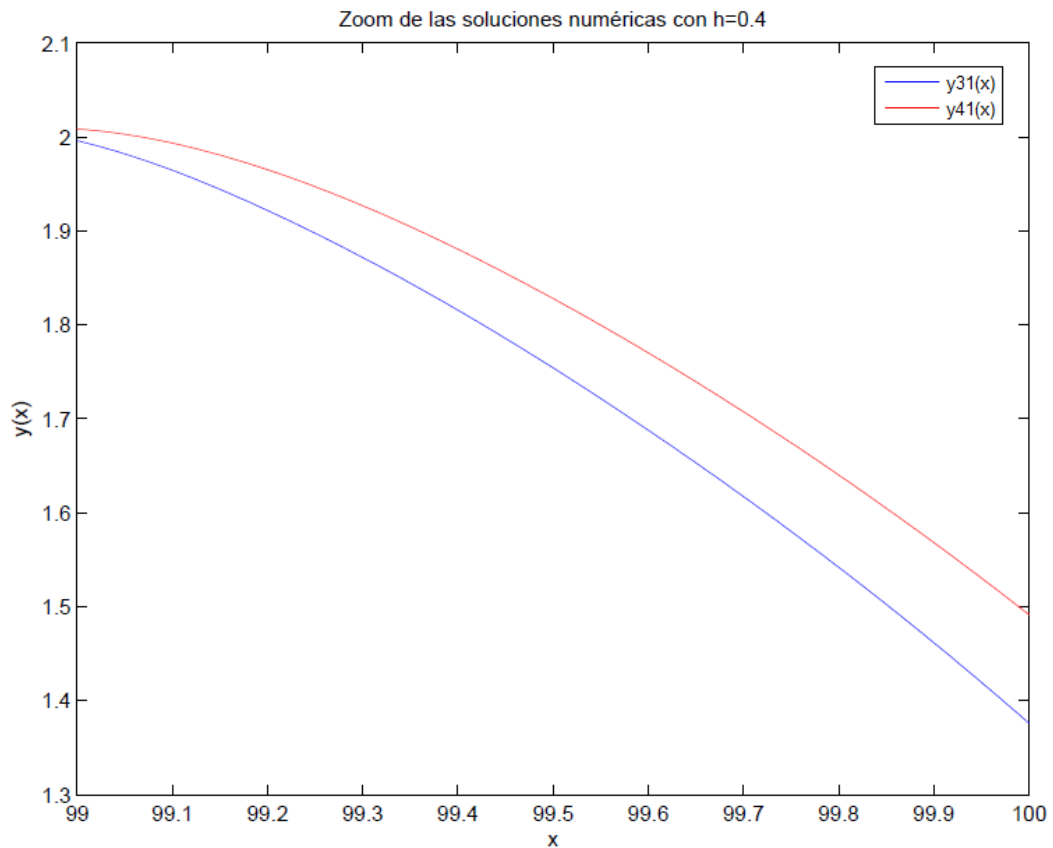


Figura 5.33. Detalle de la gráfica anterior: zoom de las soluciones numéricas,  $\mu = 1$  ( $h = 0.4$ )

Al disminuir el tamaño del paso para los diferentes métodos, de nuevo observamos que, como cabe esperar, mejoran las aproximaciones a la solución obtenidas.

La siguiente tabla muestra los resultados obtenidos en nuestros experimentos numéricos:

Tabla 5.19. Valores de las soluciones numéricas en  $x = [99, 99.8]$  para  $\mu = 1$  ( $h = \frac{0.4}{2^i}, i = 0, 1, \dots, 4$ )

SOLUCIONES NUMÉRICAS CON EL MÉTODO 'spline3pnl'					
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
99.0	1.996112e+00	2.010427e+00	2.009123e+00	2.008736e+00	2.008633e+00
99.2	1.921785e+00	1.976432e+00	1.975409e+00	1.974454e+00	1.974171e+00
99.4	1.815690e+00	1.895958e+00	1.896007e+00	1.894951e+00	1.894620e+00
99.6	1.687959e+00	1.787830e+00	1.789036e+00	1.788019e+00	1.787682e+00
99.8	1.542025e+00	1.659505e+00	1.661776e+00	1.660814e+00	1.660479e+00



SOLUCIONES NUMÉRICAS CON EL MÉTODO 'spline4pnl'					
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
99.0	2.008445e+00	2.008613e+00	2.008599e+00	2.008598e+00	2.008598e+00
99.2	1.965199e+00	1.973630e+00	1.974047e+00	1.974072e+00	1.974073e+00
99.4	1.880564e+00	1.893802e+00	1.894463e+00	1.894502e+00	1.894504e+00
99.6	1.770236e+00	1.786693e+00	1.787511e+00	1.787560e+00	1.787563e+00
99.8	1.640281e+00	1.659351e+00	1.660299e+00	1.660355e+00	1.660359e+00

Las dos últimas columnas de la tabla de soluciones numéricas para el método *spline4pnl*, ponen de manifiesto que las variaciones son menores que cuatro millonésimas en los puntos considerados, esto es, que la última aproximación obtenida a la solución es ya bastante precisa. Además, comparando los resultados de las últimas columnas obtenidas a partir de ambos métodos, observamos que las discrepancias se producen ya en las diezmilésimas.

Si representamos el comportamiento de las soluciones para el tamaño de paso  $h = 0.0125$ , también puede apreciarse cómo las diferencias entre las soluciones tienden a desaparecer. Veámoslo en la siguiente gráfica:

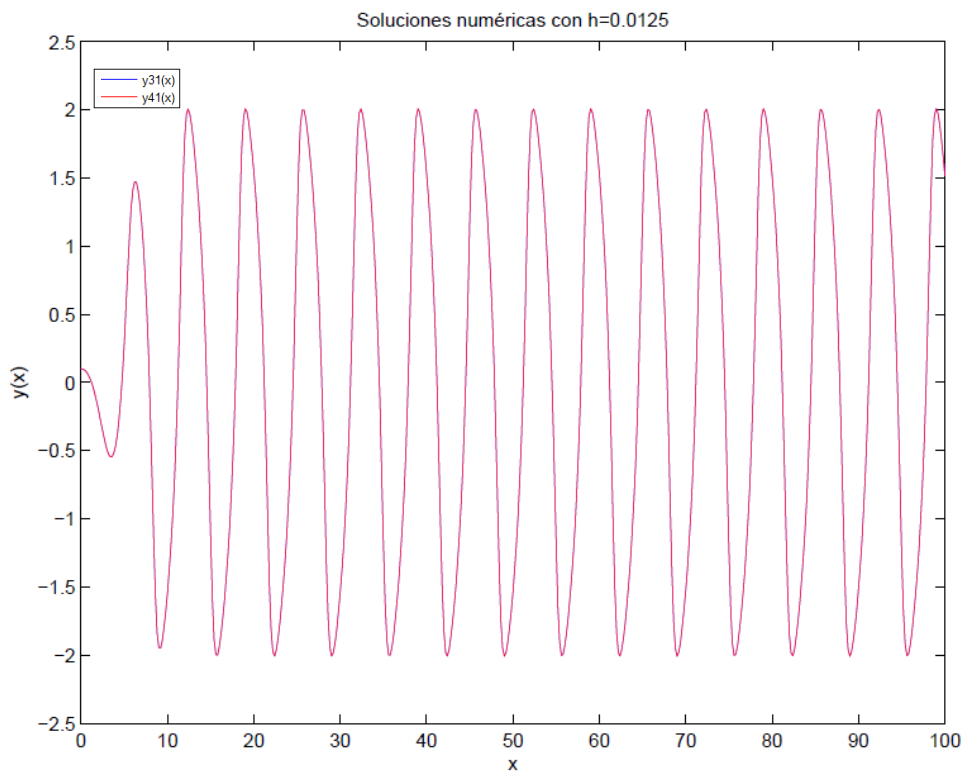


Figura 5.34. Gráfica de las soluciones numéricas para  $\mu = 1$  ( $h = 0.0125$ )

## c) Ejemplo 6.c

Por último, repetiremos nuestros experimentos aumentando de nuevo el valor del parámetro  $\mu$ , hasta un valor  $\mu = 5$ .

Estamos interesados en ver de qué modo afecta la variación de este parámetro tanto a la forma de las soluciones como a las dificultades de la integración del problema.

La representación de las soluciones numéricas obtenidas con cada método, para un tamaño de paso  $h = 0.4$ , se recogen a continuación en la siguiente gráfica:

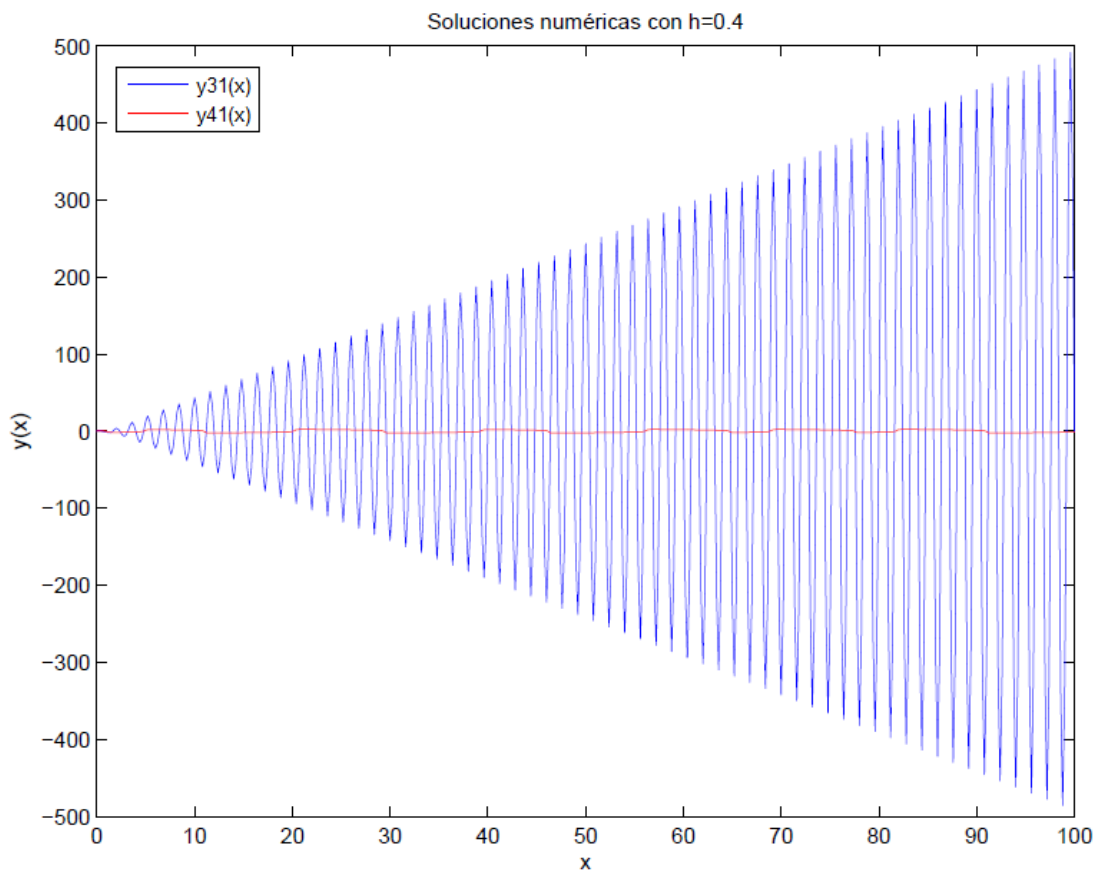


Figura 5.35. Gráfica de las soluciones numéricas para  $\mu = 5$  ( $h = 0.4$ )

En este caso se aprecian notables diferencias entre los resultados que se obtienen con los dos métodos.

La solución que proporciona la función *spline3pnl* se comporta de modo inestable, como se puede observar, para el tamaño de paso considerado.

El comportamiento de la solución correspondiente al método implementado a partir de la función *spline4pnl*, apenas se distingue en la gráfica anterior, por lo que la vamos a representar por separado para poder observarla más en detalle:

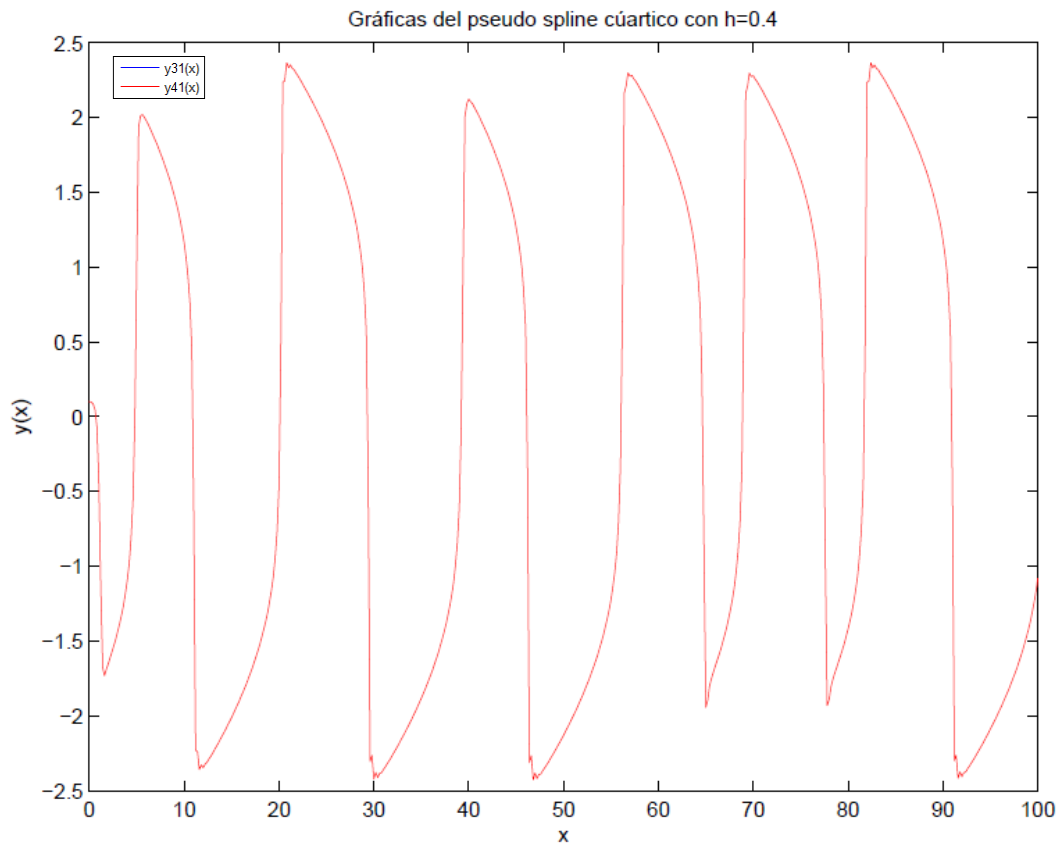


Figura 5.36. Solución numérica obtenida mediante Pseudo-Splines Cuárticos para  $\mu = 5$  ( $h = 0.4$ )

La forma que presenta la solución numérica anterior nos hace sospechar la causa de las inestabilidades que presenta el método basado en Splines Cúbicos. Observamos "picos" y rapidísimas variaciones (algunos trozos de la gráfica son casi verticales) de los que el método no puede dar cuenta con un paso tan grande.

El método basado en Pseudo-Splines Cuárticos es mucho más preciso (orden 4 frente a 2) y además utiliza casi el doble de puntos de colocación para un mismo tamaño de paso, por lo que se comporta mejor, si bien, como veremos posteriormente al reducir el tamaño de paso, no proporciona aún una solución numérica precisa.

Las diferencias entre ambos métodos para este valor del parámetro  $\mu$  son claras. De hecho, si se amplía el intervalo final de  $x$  de la Figura 5.36, las gráficas que se distinguen en la representación son totalmente dispares:

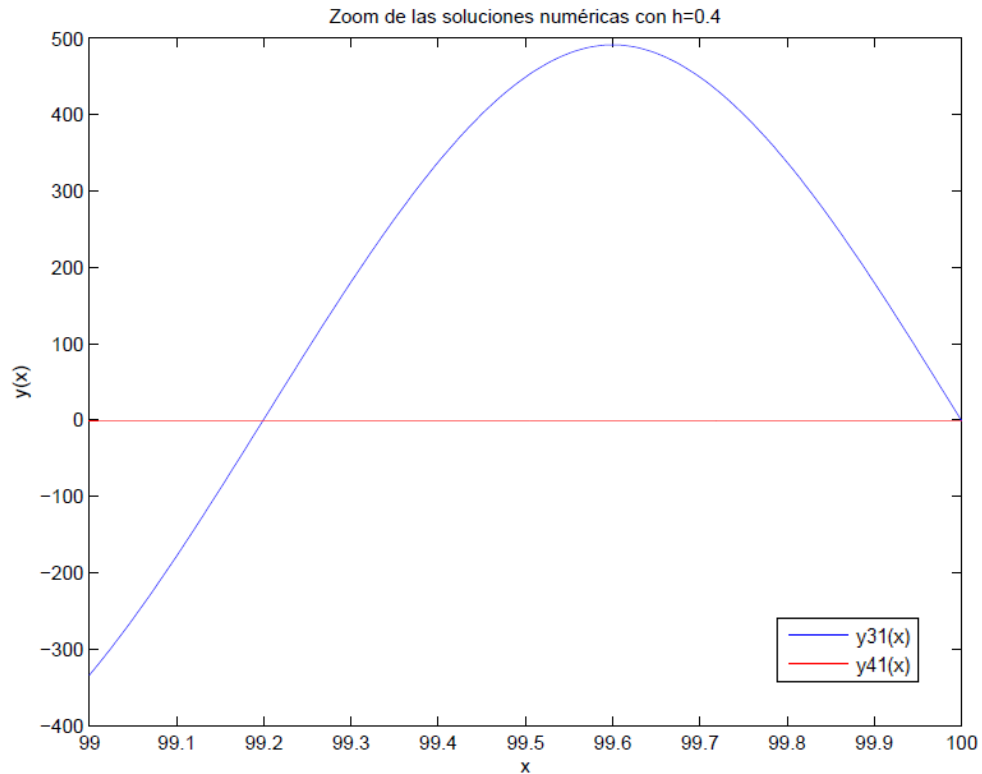


Figura 5.37. Detalle de la gráfica 5.36: zoom de las soluciones numéricas,  $\mu = 5$  ( $h = 0.4$ )

Al disminuir el tamaño de paso, se puede percibir una gran mejora en las soluciones numéricas proporcionadas por ambos métodos, pero muy especialmente en la del método *spline3pnl*. A medida que disminuye el paso, cada vez se asemejan más las soluciones de ambos métodos. Veámoslo:

Tabla 5.20. Valores de las soluciones numéricas en  $x = [99, 99.8]$  para  $\mu = 5$  ( $h = \frac{0.4}{2^i}, i = 0, 1, \dots, 4$ )

SOLUCIONES NUMÉRICAS CON EL MÉTODO ' <i>spline3pnl</i> '					
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
99.0	-3.340929e+02	1.147167e+00	1.876527e+00	1.919262e+00	1.924975e+00
99.2	9.964614e-01	1.047317e+00	1.846662e+00	1.890555e+00	1.896415e+00
99.4	3.377855e+02	9.133571e-01	1.815927e+00	1.861082e+00	1.867102e+00
99.6	4.913358e+02	7.043931e-01	1.784234e+00	1.830774e+00	1.836969e+00
99.8	3.375382e+02	2.668194e-01	1.751485e+00	1.799551e+00	1.805938e+00

SOLUCIONES NUMÉRICAS CON EL MÉTODO 'spline4pnl'					
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
99.0	-1.427466e+00	1.864198e+00	1.923882e+00	1.926342e+00	1.926452e+00
99.2	-1.374210e+00	1.833980e+00	1.895294e+00	1.897817e+00	1.897930e+00
99.4	-1.315473e+00	1.802858e+00	1.865950e+00	1.868542e+00	1.868658e+00
99.6	-1.249336e+00	1.770739e+00	1.835784e+00	1.838450e+00	1.838569e+00
99.8	-1.172760e+00	1.737516e+00	1.804717e+00	1.807465e+00	1.807588e+00

Ahora las dos últimas columnas de la tabla de soluciones numéricas para el método *spline4pnl*, difieren en poco más de una diezmilésima en los puntos considerados. Además, comparando los resultados de las últimas columnas obtenidas a partir de ambos métodos, observamos que las discrepancias son de menos de dos milésimas.

Para un paso  $h = 0.0125$ , la representación de las distintas soluciones ya es muy similar entre sí, y resultan muy difíciles de distinguir:

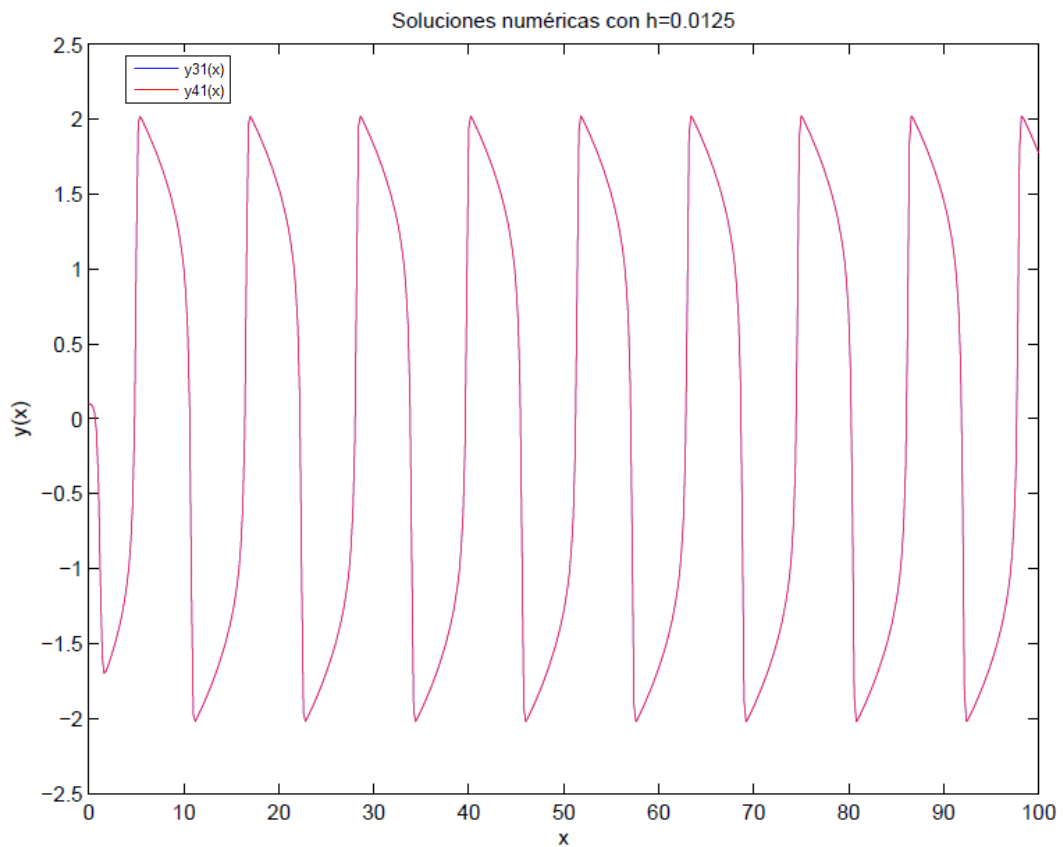


Figura 5.38. Gráfica de las soluciones numéricas para  $\mu = 5$  ( $h = 0.0125$ )

## 5.8. Ejemplo 7 <sup>[28]</sup>

### 5.8.1. Planteamiento del problema

Consideremos en este ejemplo un problema interesante tanto desde el punto de vista científico como desde el teórico.

El P.V.F. no lineal dado en términos de la ecuación del tipo Bratu:

$$\begin{cases} u''(S) = -e^{u(S)} \\ u(0) = u(1) = 0 \end{cases} \quad (5.12)$$

Esta ecuación es muy mencionada en la literatura científica, como ejemplo de ecuación No lineal. Además, la ecuación de Bratu es una ecuación diferencial ampliamente utilizada para el diseño de reactores químicos, de modo que es de gran interés en Ingeniería, especialmente en Ingeniería Química.

El problema (5.12) admite dos soluciones, ambas positivas y simétricas respecto del punto medio del intervalo 0.5, siendo una de ellas notablemente mayor que la otra.

Las soluciones, en este caso, son conocidas y vienen dadas por la expresión:

$$u(S) = -2 \ln \left( \frac{\cosh(\mu (S - 0.5))}{\cosh(\frac{\mu}{2})} \right) \quad (5.13)$$

donde ha de tomarse  $\mu$  como una de las dos soluciones positivas de la ecuación no lineal:  $2 \mu^2 = \cosh^2 \left( \frac{\mu}{2} \right)$ .

Concretamente, se obtienen los valores:  $\mu_1 = 0.758582299525377$  y  $\mu_2 = 5.469351386061054$ , respectivamente, para cada una de las dos posibles soluciones.

### 5.8.2. Resolución

Este Problema de Valores de Frontera es No lineal, de modo que para su resolución vamos a utilizar los métodos fundamentados en el Método del Disparo que recurren a Splines Cúbicos y Pseudo-Splines Cuárticos, es decir, los métodos *spline3pcnl* y *spline4pcnl*.

El problema está definido para valores de  $S = x \in [0,1]$ , y en nuestros experimentos vamos a aproximar la solución del problema que se obtiene al tomar  $\mu = \mu_1 = 0.758582299525377$ , esto es, la más pequeña de las dos que admite nuestro problema.

El código de Matlab que nos permite realizar los experimentos se encuentra recogido en los Anexos, en su apartado 7.3.7.

### 5.8.3. Resultados

Para nuestros experimentos numéricos, hemos comenzado tomando un número pequeño de subintervalos  $n = 5$ , es decir, hemos tomado un tamaño de paso fijo  $h = 0.2$ .

Si representamos gráficamente las soluciones numéricas que proporcionan los métodos, junto a la solución exacta evaluada en los diferentes puntos considerados, se obtiene la siguiente gráfica, que muestra soluciones numéricas difícilmente distinguibles de la solución exacta de nuestro problema y simétricas respecto del punto medio del intervalo:

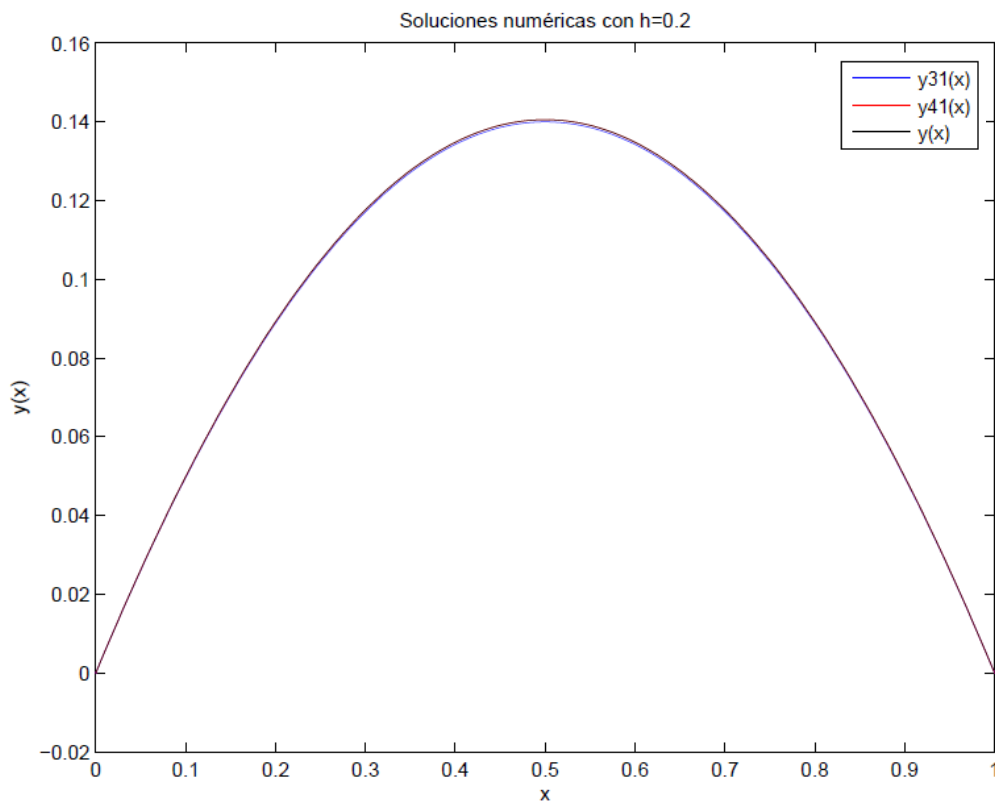
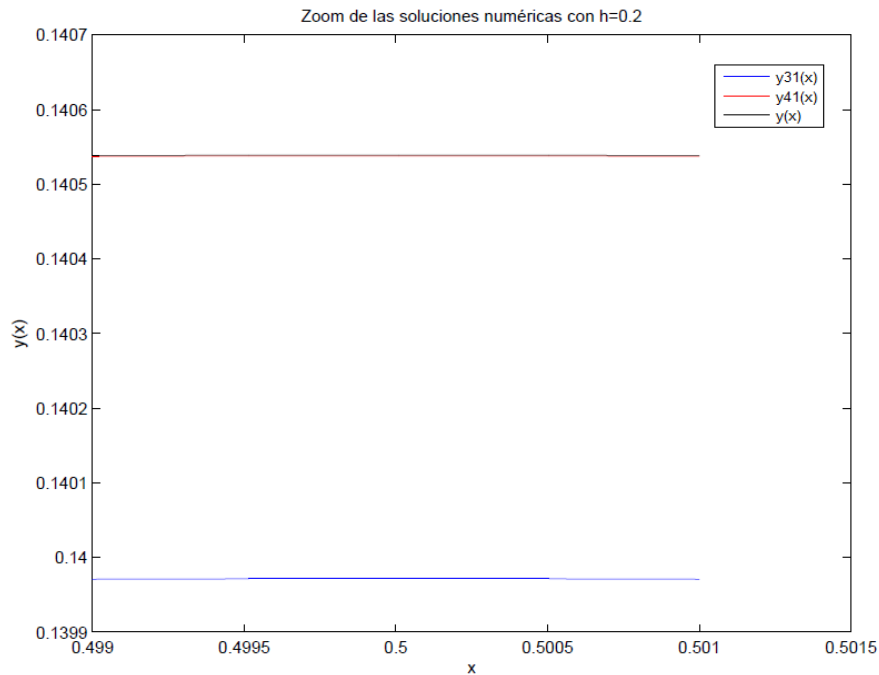


Figura 5.39. Gráfica de las soluciones numéricas y exactas ( $h = 0.2$ )


 Figura 5.40. Detalle de la gráfica anterior: zoom de las soluciones numéricas y exacta ( $h = 0.2$ )

En la última gráfica se aprecia que, pese al zoom realizado, la solución numérica que proporciona *spline4pcnl* es prácticamente indistinguible de la solución exacta, mientras que las diferencias entre éstas y la solución proporcionada por *spline3pcnl* son mucho mayores: No obstante, nótese que, en vista de la escala de la gráfica, los valores de dichas diferencias no son grandes como pudieran parecer.

Vamos a recoger a continuación, en una tabla, los valores de las soluciones numéricas y exactas en varios puntos del intervalo de integración. Los puntos se han tomado en la mitad final del intervalo, pues, como ya vimos en la Figura 5.40, estas soluciones son simétricas respecto al punto  $x = 0.5$ .

 Tabla 5.21. Valores de las soluciones numéricas en  $x = [0.5, 1]$  ( $h = 0.2$ )

$x$	$y_{31}(x)$	$y_{41}(x)$	$y(x)$
0.5	1.399715e-01	1.405381e-01	1.405392e-01
0.6	1.342531e-01	1.347891e-01	1.347903e-01
0.7	1.171402e-01	1.176082e-01	1.176091e-01
0.8	8.884431e-02	8.918923e-02	8.918993e-02
0.9	4.965443e-02	4.984644e-02	4.984679e-02
1.0	9.714451e-17	-1.526557e-16	-0.000000e+00



Observamos errores por debajo de una diezmilésima para el método *spline4pcnl* y menores de cuatro centésimas para el método *spline3pcnl*.

Si ahora analizamos la evolución de dichas soluciones numéricas cuando se modifica el tamaño de paso de acuerdo a la relación:  $h = \frac{0.2}{2^i}$ ,  $i = 0, 1, \dots, 6$ , obtenemos la siguiente tabla para los errores, al comparar con la solución exacta de la que se dispone en este caso:

Tabla 5.22. Errores en las soluciones numéricas de los métodos en  $x = [0.5, 1]$  ( $h = \frac{0.2}{2^i}$ ,  $i = 0, 1, \dots, 6$ )

SOLUCIONES NUMÉRICAS CON EL MÉTODO ' <i>spline3pcnl</i> '							
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$
0.5	-5.68e-04	-1.42e-04	-3.55e-05	-8.89e-06	-2.22e-06	-5.56e-07	-1.39e-07
0.6	-5.37e-04	-1.36e-04	-3.40e-05	-8.50e-06	-2.12e-06	-5.31e-07	-1.33e-07
0.7	-4.69e-04	-1.17e-04	-2.93e-05	-7.34e-06	-1.83e-06	-4.59e-07	-1.15e-07
0.8	-3.46e-04	-8.72e-05	-2.18e-05	-5.47e-06	-1.37e-06	-3.42e-07	-8.54e-08
0.9	-1.92e-04	-4.76e-05	-1.19e-05	-2.98e-06	-7.45e-07	-1.86e-07	-4.66e-08
1.0	9.71e-17	-6.94e-18	-3.47e-17	2.26e-17	6.85e-17	3.12e-17	9.54e-18
SOLUCIONES NUMÉRICAS CON EL MÉTODO ' <i>spline4pcnl</i> '							
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$	$i = 6$
0.5	-1.13e-06	-7.39e-08	-4.61e-09	-2.88e-10	-1.80e-11	-1.12e-12	-3.83e-14
0.6	-1.14e-06	-7.04e-08	-4.39e-09	-2.74e-10	-1.71e-11	-1.07e-12	-3.83e-14
0.7	-9.16e-07	-6.00e-08	-3.74e-09	-2.34e-10	-1.46e-11	-9.03e-13	-3.60e-14
0.8	-7.06e-07	-4.37e-08	-2.73e-09	-1.70e-10	-1.06e-11	-6.52e-13	-2.83e-14
0.9	-3.50e-07	-2.32e-08	-1.44e-09	-9.03e-11	-5.64e-12	-3.35e-13	-1.69e-14
1.0	-1.53e-16	0.00e+00	-4.16e-17	1.73e-17	9.80e-17	2.18e-14	-2.83e-15

Se observa claramente como para el primero de los métodos tiende a dividirse por cuatro el error cuando se divide por dos el paso, en tanto que, para el segundo de los métodos el error se viene a dividir por dieciséis. Dicho de otro modo, los métodos muestran un orden de convergencia dos y cuatro respectivamente, para este problema.

Si representamos los errores cometidos por los dos métodos que hemos considerado para un tamaño de paso pequeño como es  $h = 0.003125$ , se obtiene la gráfica:

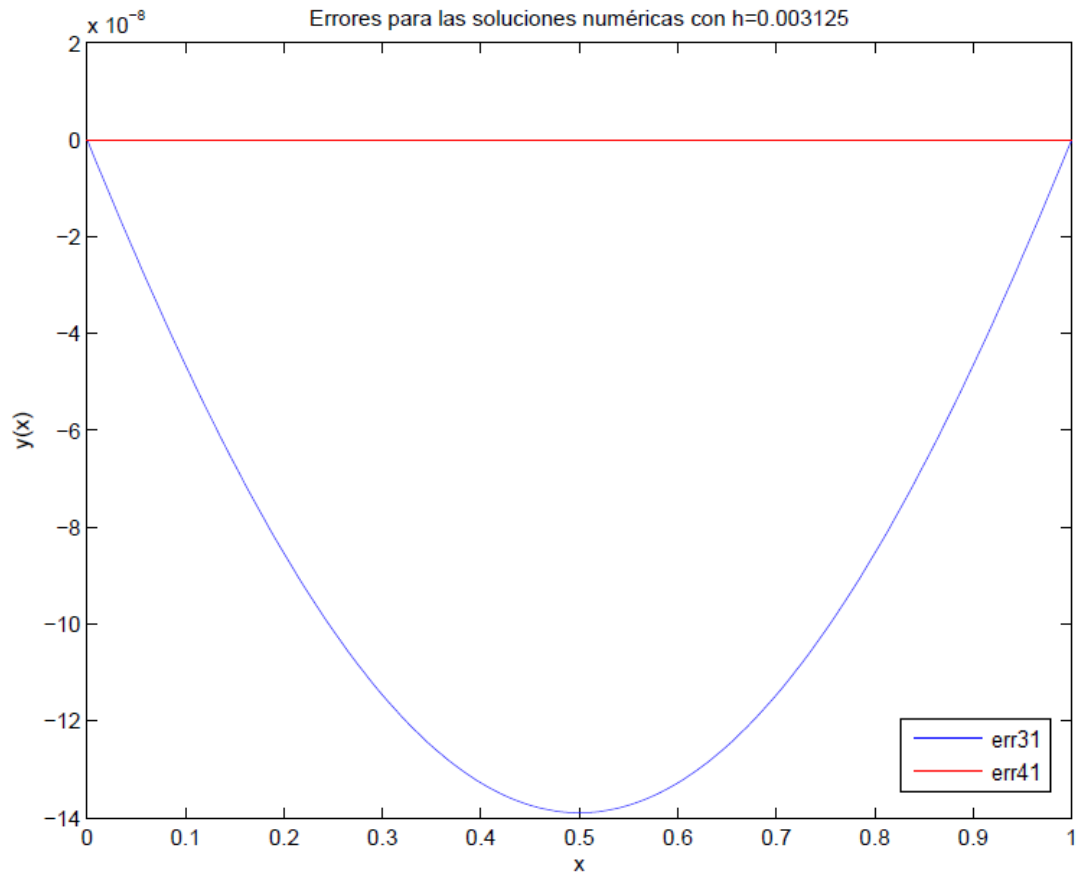


Figura 5.41. Gráfica de los errores en la aproximación a la solución ( $h = 0.003125$ )

Como vemos, el error al utilizar Pseudo-Splines Cuárticos es prácticamente nulo (roza la precisión máxima que se puede obtener en doble precisión), mientras que para el caso de Splines Cúbicos, es mayor y sigue un comportamiento parabólico, alcanzando su máximo en el punto medio del intervalo  $x = 0.5$ . No obstante, puede observarse en la escala de la gráfica anterior ( $\cdot 10^{-8}$ ), que sus valores son pequeños, en cualquier caso.

Si representamos en doble escala logarítmica los errores de cada método en dicho punto medio, frente al tamaño de paso, podemos determinar el orden de los métodos *spline3pcnl* y *spline4pcnl* experimentalmente.

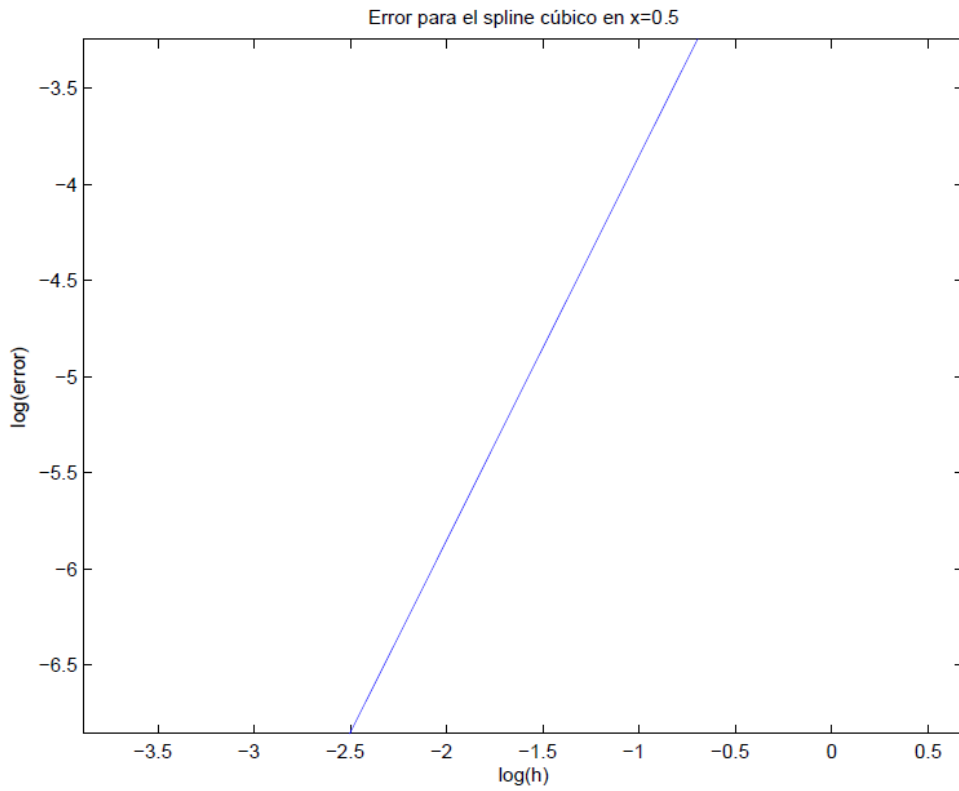


Figura 5.42. Gráfica en doble escala logarítmica del error para Splines Cúbicos en el punto  $x = 0.5$ .

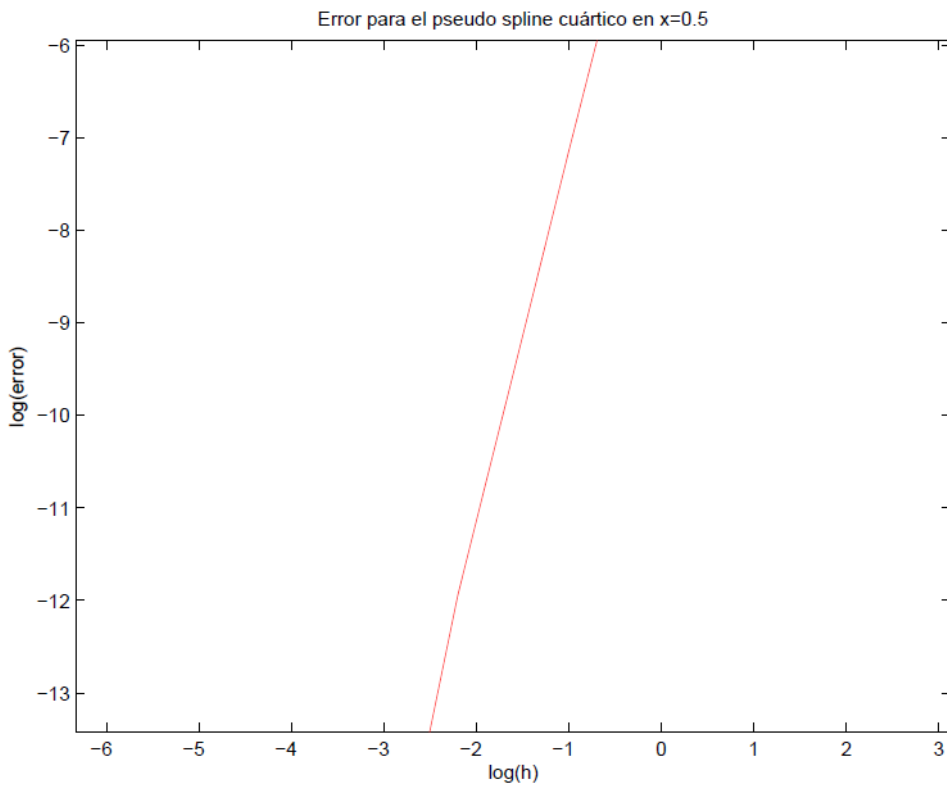


Figura 5.43. Gráfica en doble escala logarítmica del error para Pseudo-Splines Cuárticos en  $x = 0.5$ .

La pendiente de la gráfica para el método *spline3pcnl*, proporciona el orden efectivo para este problema, que como ya adelantamos a partir de la tabla de errores es  $m \cong \frac{-3.2 - (-6.8)}{-0.7 - (-2.5)} = 2$ .

Por otra parte, para el método *spline4pcnl*, se observa en la gráfica anterior, una pendiente dada por  $m \cong \frac{-6 - (-13.3)}{0.7 - (-2.5)} \cong 4$ , lo que determina una convergencia para este problema de orden 4.

## 5.9. Ejemplo 8 <sup>[30]</sup>

### 5.9.1. Planteamiento del problema

La distribución de temperaturas de una aleta de radiación de perfil trapezoidal, viene determinada por el siguiente P.V.F. no lineal:

$$\begin{cases} y''(r) = \left( -\frac{1}{r+p} + \frac{tg\alpha}{(1-r)tg\alpha + \theta} \right) y'(r) + \frac{\beta y^4(r)}{(1-r)tg\alpha + \theta} \\ y(0) = 1 \\ y'(1) = 0 \end{cases} \quad (5.14)$$

### 5.9.2. Resolución

En vista de las condiciones del Problema de Valores de Frontera (5.14), por estar la condición en el punto final dada en términos de una derivada, vamos a proceder a su resolución mediante los métodos *spline3pcnl2* y *spline4pcnl2*.

Para su resolución, hemos elegido los siguientes valores de los parámetros:  $\alpha = 6^\circ$ ,  $\rho = 0.5$ ,  $\theta = 0.05$ ,  $\beta = 0.1$ .

Consideraremos grados sexagesimales en las evaluaciones de las funciones trigonométricas.

El código que implementa este problema, aparece incluido en el apartado 7.3.8 de los Anexos.

### 5.9.3. Resultados

Seleccionando un tamaño de paso  $h = 0.125$ , hemos recogido en la siguiente tabla los valores de las aproximaciones numéricas a la solución en varios puntos del intervalo en el que está definido nuestro problema, esto es,  $x \in [0,1]$ . Se tiene:

Tabla 5.23. Valores de las soluciones numéricas ( $h = 0.125$ )

$x$	$y_{31}(x)$	$y_{41}(x)$
0.0	1.000000e+00	1.000000e+00
0.2	9.034501e-01	9.044018e-01
0.4	8.386996e-01	8.400033e-01
0.6	7.941344e-01	7.956722e-01
0.8	7.656932e-01	7.674569e-01
1.0	7.546521e-01	7.565531e-01

Nótese que son similares entre sí y que, de hecho, difieren en menos de dos centésimas. Esto se aprecia en la siguiente gráfica que muestra las aproximaciones obtenidas a la distribución de temperaturas:

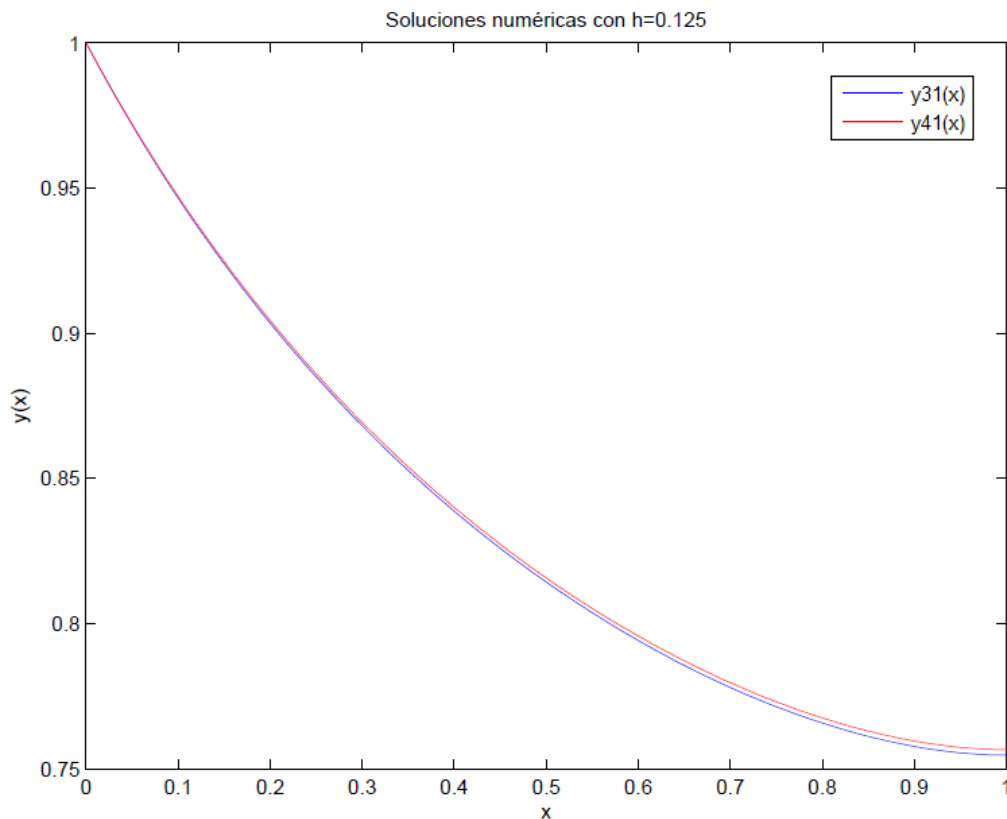


Figura 5.44. Gráfica de las soluciones numéricas ( $h = 0.125$ )

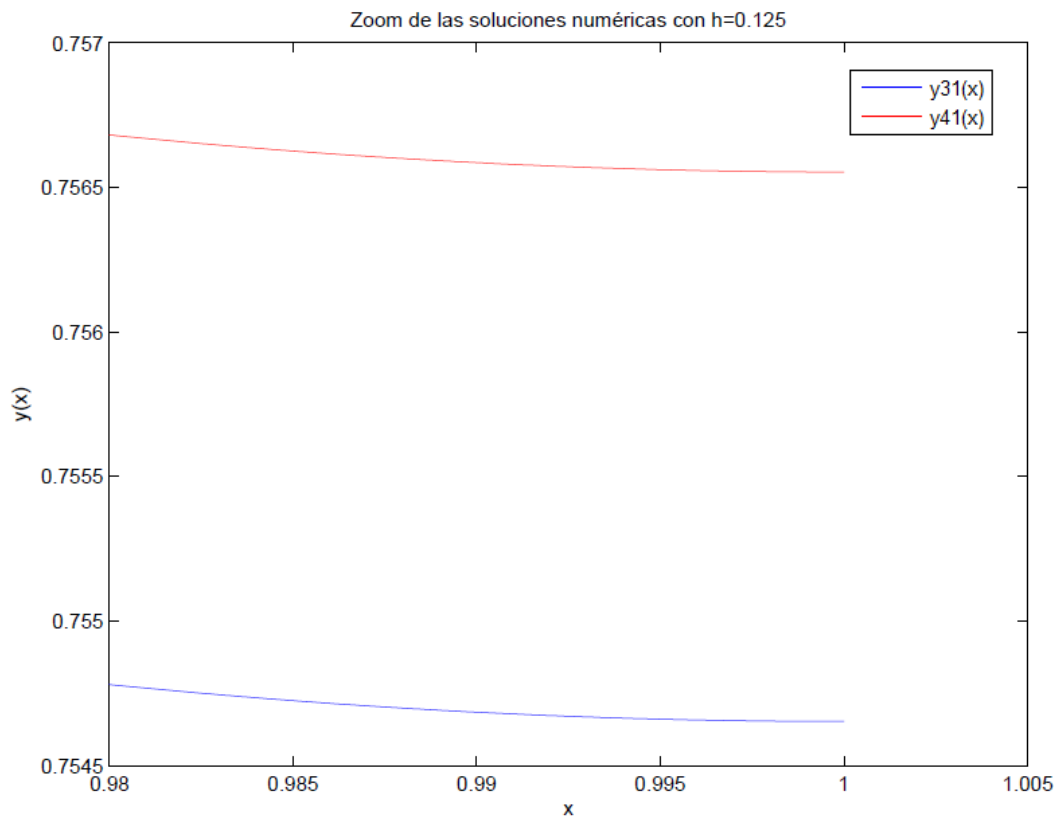


Figura 5.45. Detalle de la gráfica anterior: zoom de las soluciones numéricas ( $h = 0.125$ )

Al ir disminuyendo el tamaño del paso, como ocurría con otros métodos evaluados en ejemplos anteriores, las soluciones son cada vez más parecidas, como puede apreciarse en las siguientes tablas:

Tabla 5.24. Valores de las soluciones numéricas ( $h = \frac{0.125}{2^i}, i = 0, 1, \dots, 5$ )

SOLUCIONES NUMÉRICAS CON EL MÉTODO 'spline3pcnl2'						
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$
0.0	1.00000e+00	1.00000e+00	1.00000e+00	1.00000e+00	1.00000e+00	1.00000e+00
0.2	9.034501e-01	9.041667e-01	9.043400e-01	9.043830e-01	9.043938e-01	9.043964e-01
0.4	8.386996e-01	8.396788e-01	8.399195e-01	8.399793e-01	8.399943e-01	8.399980e-01
0.6	7.941344e-01	7.952889e-01	7.955743e-01	7.956454e-01	7.956632e-01	7.956676e-01
0.8	7.656932e-01	7.670176e-01	7.673461e-01	7.674281e-01	7.674486e-01	7.674538e-01
1.0	7.546521e-01	7.560789e-01	7.564369e-01	7.565265e-01	7.565489e-01	7.565545e-01

SOLUCIONES NUMÉRICAS CON EL MÉTODO 'spline4pcnl2'						
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$
0.0	1.00000e+00	1.00000e+00	1.00000e+00	1.00000e+00	1.00000e+00	1.00000e+00
0.2	9.044018e-01	9.043976e-01	9.043974e-01	9.043973e-01	9.043973e-01	9.043973e-01
0.4	8.400033e-01	8.399995e-01	8.399993e-01	8.399993e-01	8.399993e-01	8.399993e-01
0.6	7.956722e-01	7.956693e-01	7.956691e-01	7.956691e-01	7.956691e-01	7.956691e-01
0.8	7.674569e-01	7.674556e-01	7.674555e-01	7.674555e-01	7.674555e-01	7.674555e-01
1.0	7.565531e-01	7.565561e-01	7.565563e-01	7.565563e-01	7.565563e-01	7.565563e-01

Las tres últimas columnas de la tabla de soluciones numéricas para el método *spline4pcnl2* son idénticas en los puntos considerados (para el número de cifras mostrado). Además, comparando los resultados de las últimas columnas obtenidas a partir de ambos métodos, observamos que las discrepancias son de menos de dos millonésimas.

De hecho, si representamos las soluciones numéricas obtenidas con ambos métodos, para el tamaño de paso más pequeño considerado en nuestro experimento, observamos que son indistinguibles:

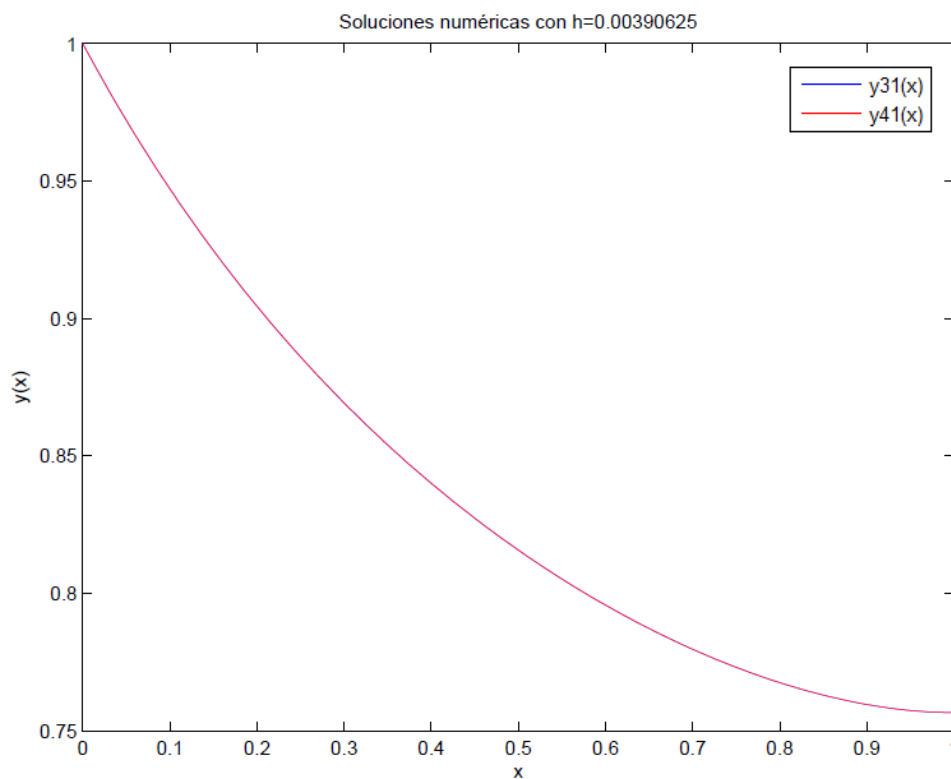


Figura 5.46. Gráfica de las soluciones numéricas ( $h = 0.00390625$ )

## 5.10. Ejemplo 9 <sup>[30]</sup>

### 5.10.1. Planteamiento del problema

La deflexión de una viga en voladizo sometida a una carga concentrada se puede obtener resolviendo el P.V.F. siguiente:

$$\begin{cases} y''(x) + \lambda x \cos(y(x)) = 0 \\ y'(0) = 0 \\ y(1) = 0 \end{cases} \quad (5.15)$$

Este problema surge a menudo en el Cálculo de Estructuras y la Resistencia de Materiales, campos muy destacados en varias ramas de la Ingeniería Industrial tales como: Ingeniería Civil, Ingeniería de Edificación e Ingeniería Mecánica.

### 5.10.2. Resolución

Este Problema de Valores de Frontera tiene condiciones especiales, ya que la condición de contorno en el punto inicial del intervalo  $x \in [0,1]$ , en que está definido el problema, viene dada en términos de una derivada. Es por ello que utilizaremos los métodos *spline3pcn/3* y *spline4pcn/3* en su estudio.

Para nuestros experimentos hemos elegido un valor del parámetro  $\lambda = 8$ .

### 5.10.3. Resultados

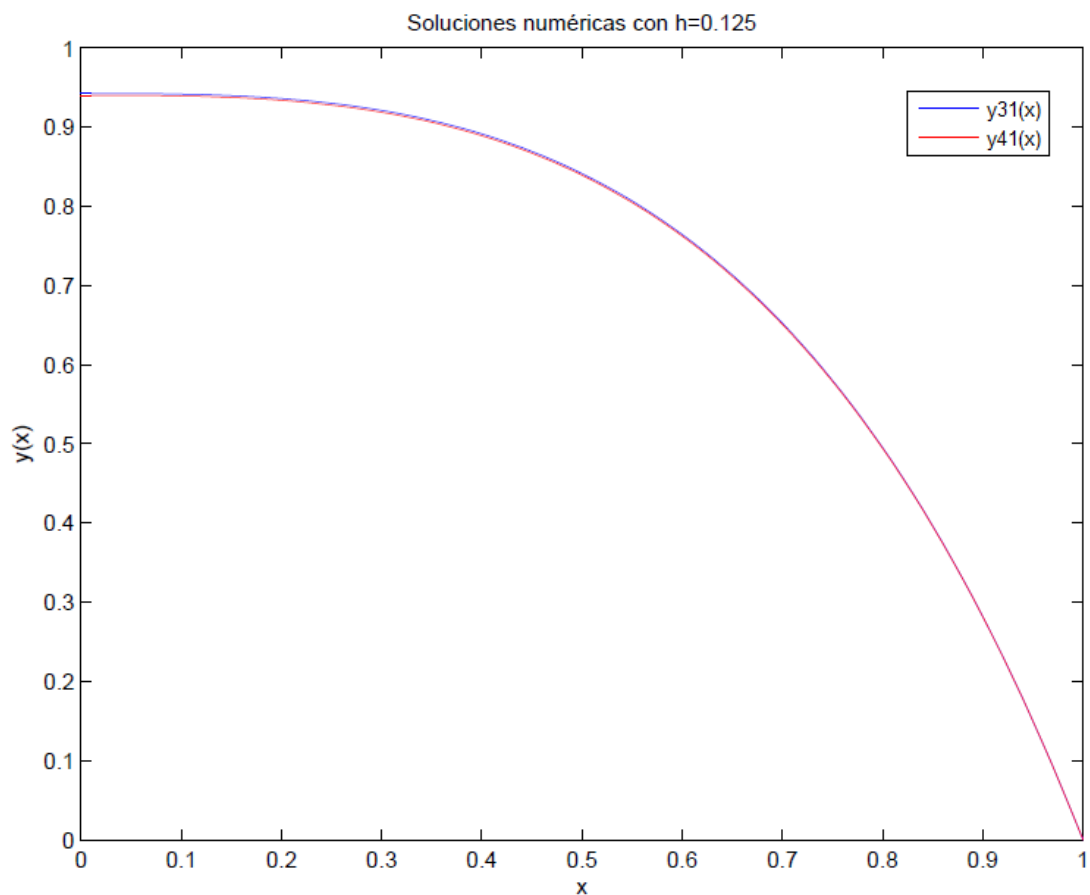
Tomando un número de subintervalos  $n = 8$ , que determina un tamaño de paso  $h = 0.125$ , se obtienen las aproximaciones numéricas a la solución del problema (5.15) que se presentan en la Tabla 5.25 de la siguiente página.



Tabla 5.25. Valores de las soluciones numéricas ( $h = 0.125$ )

$x$	$y_{31}(x)$	$y_{41}(x)$
0.0	9.424559e-01	9.401299e-01
0.2	9.361630e-01	9.338299e-01
0.4	8.914288e-01	8.891204e-01
0.6	7.644683e-01	7.624775e-01
0.8	4.955586e-01	4.945404e-01
1.0	2.220446e-15	1.887379e-15

La representación de estas soluciones proporciona las gráficas:

Figura 5.47. Gráfica de las soluciones numéricas ( $h = 0.125$ )

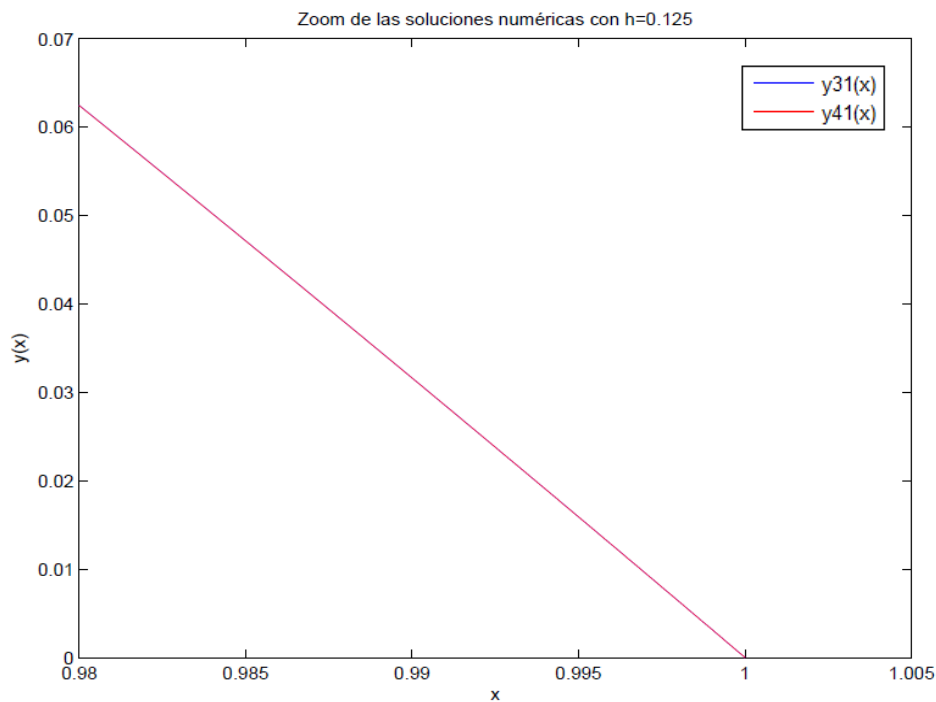


Figura 5.48. Detalle de la gráfica anterior: zoom de las soluciones numéricas ( $h = 0.125$ )

Como vemos, a pesar de haber ampliado la gráfica de las soluciones, ambos métodos proporcionan soluciones muy parecidas, lo que indica que estos métodos resuelven con precisión este tipo de problema.

El resultado anterior puede mejorarse aún más disminuyendo el tamaño de paso. Es por ello que repetimos nuestro experimento con sucesivas reducciones del tamaño del paso y recogemos los resultados en las siguientes tablas:

Tabla 5.26. Valores de las soluciones numéricas ( $h = \frac{0.125}{2^i}, i = 0, 1, \dots, 5$ )

SOLUCIONES NUMÉRICAS CON EL MÉTODO 'spline3pci3'						
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$
0.0	9.424559e-01	9.407096e-01	9.402688e-01	9.401584e-01	9.401307e-01	9.401238e-01
0.2	9.361630e-01	9.344110e-01	9.339686e-01	9.338577e-01	9.338299e-01	9.338230e-01
0.4	8.914288e-01	8.896952e-01	8.892561e-01	8.891460e-01	8.891184e-01	8.891115e-01
0.6	7.644683e-01	7.629729e-01	7.625900e-01	7.624938e-01	7.624697e-01	7.624637e-01
0.8	4.955586e-01	4.947930e-01	4.945913e-01	4.945402e-01	4.945274e-01	4.945242e-01
1.0	2.220446e-15	1.526557e-15	1.720846e-15	2.456368e-15	1.800643e-15	1.153591e-15

SOLUCIONES NUMÉRICAS CON EL MÉTODO 'spline4pcnl3'						
$x$	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$
0.0	9.401299e-01	9.401221e-01	9.401216e-01	9.401215e-01	9.401215e-01	9.401215e-01
0.2	9.338299e-01	9.338213e-01	9.338207e-01	9.338207e-01	9.338207e-01	9.338207e-01
0.4	8.891204e-01	8.891100e-01	8.891093e-01	8.891092e-01	8.891092e-01	8.891092e-01
0.6	7.624775e-01	7.624627e-01	7.624618e-01	7.624617e-01	7.624617e-01	7.624617e-01
0.8	4.945404e-01	4.945242e-01	4.945232e-01	4.945231e-01	4.945231e-01	4.945231e-01
1.0	1.887379e-15	1.332268e-15	3.941292e-15	1.318390e-15	4.510281e-16	1.283695e-15

Las tres últimas columnas de la tabla de soluciones numéricas para el método *spline4pcnl2* son idénticas en los puntos considerados (para el número de cifras mostrado), salvo por errores de redondeo en el nodo final. Además, comparando los resultados de las últimas columnas obtenidas a partir de ambos métodos, observamos que las discrepancias son de menos de tres millonésimas.

En la siguiente gráfica incluimos las soluciones numéricas que proporcionan ambos métodos para el tamaño de paso más pequeño considerado. Como se observa, son indistinguibles a esta escala y reproducen fielmente la solución exacta de nuestro problema.

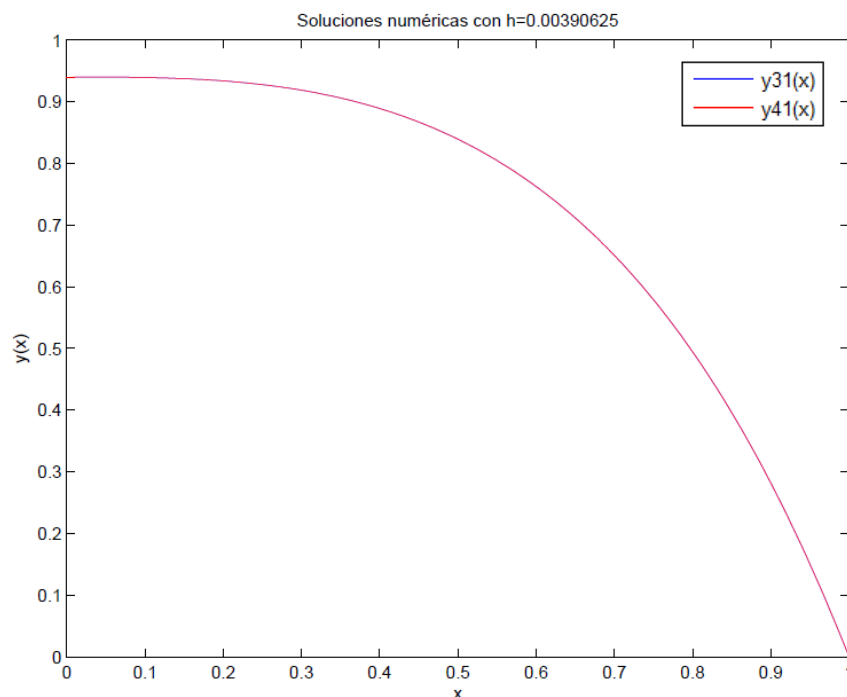


Figura 5.49. Gráfica de las soluciones numéricas ( $h = 0.00390625$ )





# CONCLUSIONES

## Capítulo 6



## 6. CONCLUSIONES

### 6.1. Resultados obtenidos

Tras el extenso análisis que se ha realizado en el capítulo anterior, podemos concluir que los objetivos planteados en un principio se han cumplido sobradamente.

Hemos comprobado que los métodos propuestos en este Trabajo de Fin de Grado permiten aproximar mediante funciones polinomiales a trozos de clase  $C^2$ , con precisión y coste computacional asociado bajo, las soluciones a problemas de valores iniciales y de contorno, tanto lineales como no lineales, que involucran Ecuaciones Diferenciales Ordinarias de Segundo Orden, como se pretendía inicialmente.

En general, todos los métodos considerados proporcionan soluciones numéricas precisas cuando se utiliza un tamaño de paso adecuado al problema. Eso sí, son especialmente destacables los resultados que se obtienen al aplicar nuestros métodos de colocación basados en Pseudo-Splines Cuárticos. Hasta donde hemos podido comprobar, consultando una extensa bibliografía del tema, no han sido implementados previamente en ningún artículo científico.

Los métodos de colocación basados en Splines Cúbicos, sin ser originales, han sido implementado de manera eficiente y original para varios problemas, reduciendo el coste computacional asociado a su aplicación.

Hemos mostrado experimentalmente que los métodos basados en Splines Cúbicos tienen un orden de convergencia cuadrática, en tanto que los nuevos métodos propuestos, basados en Pseudo-Splines Cuárticos, convergen con orden cuatro.

Es especialmente reseñable que las aproximaciones numéricas que proporcionan nuestros métodos vienen dadas en términos de funciones muy sencillas de evaluar (son polinomiales a trozos) y de derivar. Además, el ser de clase  $C^2$  las hace especialmente indicadas para aplicaciones a la Industria y la Ingeniería. Por ejemplo, si los movimientos de una máquina industrial vienen descritos por la solución de un problema del tipo considerado en esta memoria, es en general importante que tanto la posición, como la velocidad y aceleración se comporten de manera continua. Nuestras soluciones numéricas cumplirían con estos requerimientos.

## 6.2. Conclusiones

Los análisis realizados a través de numerosos experimentos numéricos, que como ya hemos mencionado muestran el cumplimiento de los objetivos perseguidos en este proyecto, nos permiten, a mayores, extraer las siguientes conclusiones:

- En este trabajo se ha analizado, con éxito, la gran importancia de problemas formulados en términos de Ecuaciones Diferenciales de Segundo Orden en muchos ámbitos, especialmente el de Ingeniería, mediante diversos ejemplos que se han proporcionado.
- Se han ampliado nuestros conocimientos relativos a la resolución de Problemas de Valores Iniciales y en la Frontera, dados en términos de Ecuaciones Diferenciales Ordinarias de Segundo Orden, introduciendo para ello, brevemente, los métodos clásicos utilizados para tal fin, y presentando nuevos métodos innovadores.
- Se ha logrado implementar con éxito en Matlab los nuevos métodos, resultando programas sencillos, precisos y de bajo coste computacional asociado.
- Los resultados que se obtienen al aplicar nuestros métodos implementados en Matlab a problemas reales del ámbito de la Ingeniería y de la Ciencia son incluso mejores de lo que cabía esperar inicialmente.
- Se ha cumplido con el carácter formativo inherente a este tipo de proyectos. De hecho, este trabajo nos ha permitido experimentar en primera persona el ámbito de la Investigación, relacionada en este caso con el Análisis Numérico.

Además, el análisis realizado nos permite identificar algunas propiedades y características destacables de los métodos propuestos, que en algunos aspectos mejoran a otros clásicos. Principalmente:

- Son muy precisos (especialmente los de orden 4).
- Son sencillos de implementar.
- El coste computacional asociado a su aplicación es bajo.
- Las soluciones que proporcionan son de clase  $C^2$ .



- Pueden aplicarse con éxito a un gran número de problemas de interés.
- Permiten resolver algunos problemas "difíciles", para los que muchos métodos clásicos no proporcionan una solución precisa a un coste computacional razonable.

### 6.3. Líneas de continuación

Se proponen las siguientes líneas futuras para continuar con este trabajo:

- Estudiar una posible generalización de los nuevos métodos propuestos, que sea aplicable a problemas que involucren otros tipos de ecuaciones diferenciales, por ejemplo, ecuaciones con órdenes diferentes al considerado: Ecuaciones Diferenciales de Primer Orden, Tercer Orden, etc.
- Buscar una extensión de nuestros métodos que los haga aplicables a la resolución de Sistemas de Ecuaciones Diferenciales Ordinarias.
- Estudiar la obtención de métodos basados en Splines y Pseudo-Splines, pero utilizando otro tipo de técnicas distintas a la de colocación, como, por ejemplo: la de mínimos cuadrados, la de Rayleigh-Ritz, etc.
- Implementar los métodos de modo que se vean afectados en menor medida por los errores de redondeo en problemas "difíciles" y especialmente sensibles a la acumulación de los mismos.





# BIBLIOGRAFÍA



## BIBLIOGRAFÍA

En esta sección de la memoria se muestra la bibliografía consultada. Se ordena por orden alfabético del primer apellido del primer autor.

### Libros y documentos:

- [1] Álvarez López, Jorge. *Métodos GRK para ecuaciones diferenciales ordinarias*. Tesis Doctoral, Universidad de Valladolid, 2002.
- [2] Braun, Martin. *Differential equations and their applications: An introduction to applied mathematics*. New York, USA; Springer, 1978.
- [3] Burden, Richard L.; Faires, J. Douglas. *Análisis numérico*. México, Grupo Editorial Iberoamérica, 1985.
- [4] Butcher, J.C. *Numerical methods for ordinary differential equations*. England, John Wiley & Sons, 2008.
- [5] Evans, Gwynne; Blackledge, Jonathan; Yardley, Peter. *Numerical methods for partial differential equations*. Leicester, Springer Science & Business Media, 2012.
- [6] Greenspan, Donald. *Numerical solution of ordinary differential equations*. Weinheim, Alemania; John Wiley & Sons, 2006.
- [7] Hairer, Ernst; Wanner, Gerhard; Nørsett, Syvert P. *Solving ordinary differential equations I: Nonstiff problems*. Suiza; Springer, 1987.
- [8] Hairer, Ernst; Wanner, Gerhard; Nørsett, Syvert P. *Solving ordinary differential equations II: Stiff and differential-algebraic problems*. Suiza, Springer, 2002.
- [9] Logan, J.David. *A first course in differential equations*. Suiza, Springer, 2015.
- [10] Merchán Ríos, Nuria. *Software de resolución de problemas con valor en frontera de ecuaciones diferenciales ordinarias*. Proyecto Fin de Carrera, Universidad Pontificia de Comillas, 2009.
- [11] Martínez, Andrei. *El análisis numérico en los últimos 25 años*. Paper, Universidad de Almería, 2003.

- [12] Nagle, R. Kent; Saff, Edward; Snider, Arthur David. *Ecuaciones diferenciales y problemas con valores en la frontera*. México, Pearson Educación, 2005
- [13] Novo, Sylvia; Obaya, Rafael; Rojo, Jesús. *Ecuaciones y sistemas diferenciales*. Madrid, Mc.Graw Hill, 1995.
- [14] Rutishauser, Heinz. *Lectures on numerical mathematics*. Basel, Berlin, Birkhäuser Boston, 1990.
- [15] Soetart, Karline; Cash, Jeff; Mazzia, Francesca. *Solving differential equations in R*. Dordrecht, London; Springer, 2012.
- [16] Thomas, J.R. Hughes. *The finite element method*. Mineola, New York; Dover Publications, 2000.
- [17] Young Yang, Won; Chung, Tae-Sang; Cao, Wenwu; Morris, John. *Applied numerical methods using Matlab*. New Jersey, USA; John Wiley & Sons, 2005.

#### Artículos y publicaciones científicas:

- [18] Altay, N; Demiralp, M. *Numerical Solution of Ordinary Differential Equations via Splines*. 1st WSEAS International Conference on Multivariate Analysis and its Application in Science and Engineering. (2008): 141-145.  
[http://www.wseas.us/e-library/conferences/2008/istanbul/mino-maase/mino-maase\\_21.pdf](http://www.wseas.us/e-library/conferences/2008/istanbul/mino-maase/mino-maase_21.pdf)
- [19] Caglar, N.; Caglar, H. *B-spline method for solving linear system of second-order boundary value problems*. Computers & Mathematics with Applications, Vol. 57, Issue 5, (2009): 757–762.  
<http://www.sciencedirect.com/science/article/pii/S0898122108005816>
- [20] Chang, J. *B-Spline Method for Solving Boundary Value Problems of Linear Ordinary Differential Equations*. International Conference on Information Computing and Applications, (2010): 326-333.  
[http://link.springer.com/chapter/10.1007/978-3-642-16339-5\\_43](http://link.springer.com/chapter/10.1007/978-3-642-16339-5_43)

- [21] Chang, J. *Comparison of B-spline Method and Finite Difference Method to Solve BVP of Linear ODEs*. Journal of Computers, V.6, N.10, (2011): 2149-2155.  
<http://www.jcomputers.us/vol6/jcp0610-20.pdf>
- [22] Liu, H.; Liu, L. *A difference scheme based on spline approximations to solve the singularly-perturbed Neumann problems*. International Conference on Computer Modeling and Simulation, (2009): 209-212.  
<http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4797384>
- [23] Lamnii, A. *Spline solution of some linear boundary value problems*. Applied Mathematics E-Notes, 8, (2008): 171-178.  
<http://eudml.org/doc/55295>
- [24] Munguia, M; Bhatta, D. *Use of Cubic B-Spline in Approximating Solutions of Boundary Value Problems*. Applications and Applied Mathematics, Vol. 10, Issue 2, (2015), 750–771.  
[https://www.pvamu.edu/mathematics/wp-content/uploads/sites/49/07\\_r773\\_bhatta\\_aam\\_r773\\_db\\_112914\\_jv\\_r.pdf](https://www.pvamu.edu/mathematics/wp-content/uploads/sites/49/07_r773_bhatta_aam_r773_db_112914_jv_r.pdf)
- [25] Pop, D.N. *Approximation methods for second order nonlinear polylocal problems*. Stud. Univ. Babes-Bolyai Math. 56, N°2, (2011): 515–526.  
<http://www.cs.ubbcluj.ro/~studia-m/2011-2/pop-trimbitas-final.pdf>
- [26] Rashidinia, J. *Cubic Spline Method for Two-Point Boundary Value Problems*. IUST International Journal of Engineering Science, V.19, N.5-2, (2008): 39-43.  
[http://www.sid.ir/en/VEWSSID/J\\_pdf/807200805-210.pdf](http://www.sid.ir/en/VEWSSID/J_pdf/807200805-210.pdf)
- [27] Rashidinia, J. *Spline Collocation Method for Solution of Higher Order Linear Boundary Value Problems*. TWMS J. Pure Appl. Math., V.6, N.1, (2015): 38-47.  
<http://static.bsu.az/w24/V6%20No1/pp38-47.pdf>

- [28] Robert D.R, Shampine L.F. A Collocation Method for Boundary Value Problems. Springer-Verlag, Numer. (1972) Math. 19, 1-28.  
<https://link.springer.com/article/10.1007/BF01459946>
- [29] Robert D.R. A Comparison of Collocation and Finite Differences for Two-Point Boundary Value Problems. SIAM Journal on Numerical Analysis, Vol. 14, No. 1 (Mar., 1977), pp. 19-39.  
<http://www.jstor.org/stable/2156619>
- [30] Siraj-ul-Islam, Imran A., Božidar Š. *The numerical solution of second-order boundary-value problems by collocation method with the Haar wavelets*. Elsevier Ltd. (2010).  
<http://www.sciencedirect.com/science/article/pii/S0895717710003006?via%3Dihub>
- [31] Srivastava, P. K. *Study of Differential Equations with their Polynomial and Nonpolynomial Spline Based Approximation*- Acta Tehnica Corviniensis - Bulletin of Engineering, Tomme VII, 3, (2014): 139-150.  
<http://acta.fih.upt.ro/pdf/2014-3/ACTA-2014-3-22.pdf>

**Páginas web:** Comprobadas a 13/07/2017

- [32] Bravo Yuste, Santos. *La ecuación de Duffing*. Universidad de Extremadura, Facultad de Ciencias, Departamento de Física Estadística. Disponible en:  
<http://www.eweb.unex.es/eweb/fisteor/santos/mfm/Alumnos/Eugenio/Html/La%20ecuacion%20de%20Duffing.htm>
- [33] Carreño Saavedra, Jhon Fredy. *Oscilaciones: del oscilador armónico al oscilador caótico*. Universidad Industrial de Santander, Escuela de Física. Disponible en:  
<http://www.ilia.miscomunidades.com/Computational%20Physics/Oscilaciones%20Caoticas.compressed.pdf>



- [34] Cuevas, Juan Carlos. *Dinámica no lineal y caos*. Universidad Autónoma de Madrid, Facultad de Ciencias, Departamento de Física Teórica Condensada. Disponible en: [https://www.uam.es/personal\\_pdi/ciencias/jcuevas/Teaching/Resumen-Capitulo2.pdf](https://www.uam.es/personal_pdi/ciencias/jcuevas/Teaching/Resumen-Capitulo2.pdf)
- [35] Merino Morlesin, Manuel. *Ejercicios de Ecuaciones Diferenciales con Matlab: Sistemas de ecuaciones diferenciales*. Universidad de Huelva, Facultad de Ciencias experimentales. Departamento de Matemáticas. Disponible en: [http://www.uhu.es/manuel.merino/files/Ecuaciones\\_Diferenciales/Boletines\\_problemas/Boletin2.pdf](http://www.uhu.es/manuel.merino/files/Ecuaciones_Diferenciales/Boletines_problemas/Boletin2.pdf)
- [36] Nebot, Miguel. *Oscilaciones y ondas: Oscilador armónico simple y amortiguado*. IFIC: Instituto de Física Corpuscular, Universidad de Valencia. Disponible en: [http://ific.uv.es/~nebot/Oscilaciones\\_y\\_Ondas/Tema\\_1.pdf](http://ific.uv.es/~nebot/Oscilaciones_y_Ondas/Tema_1.pdf)
- [37] Nebot, Miguel. *Oscilaciones y ondas: Oscilaciones forzadas*. IFIC: Instituto de Física Corpuscular, Universidad de Valencia. Disponible en: [http://ific.uv.es/~nebot/Oscilaciones\\_y\\_Ondas/Tema\\_2.pdf](http://ific.uv.es/~nebot/Oscilaciones_y_Ondas/Tema_2.pdf)
- [38] Nebot, Miguel. *Oscilaciones y ondas: Oscilaciones acopladas*. IFIC: Instituto de Física Corpuscular, Universidad de Valencia. Disponible en: [http://ific.uv.es/~nebot/Oscilaciones\\_y\\_Ondas/Tema\\_3.pdf](http://ific.uv.es/~nebot/Oscilaciones_y_Ondas/Tema_3.pdf)
- [39] Sandoval Mercado, Carlos. *Ejemplos y teoría de las vibraciones paramétricas*. Universidad de las Américas Puebla, Escuela de Ingeniería, Departamento de Ingeniería Mecánica. Disponible en: [http://catarina.udlap.mx/u\\_dl\\_a/tales/documentos/lim/sandoval\\_m\\_c/capitulo2.pdf](http://catarina.udlap.mx/u_dl_a/tales/documentos/lim/sandoval_m_c/capitulo2.pdf)





# ANEXOS

## Código de Matlab



## 7. ANEXOS: CÓDIGO DE MATLAB

En este anexo se presenta el código de los diferentes programas y funciones de Matlab en los que se han implementado los métodos numéricos de resolución que se han explicado a lo largo del proyecto.

También se incluyen los programas que corresponden a los ejemplos de experimentación numérica que se han utilizado para el análisis de dichos métodos.

### 7.1. Métodos de resolución basados en Splines

Comenzamos mostrando los códigos de Matlab de los métodos cuyo fundamento es la Teoría de Splines y que fueron explicados en el Capítulo 3.

#### 7.1.1. Funciones para resolver Problemas de Valores Iniciales Lineales

##### A. Método de resolución basado en Splines Cúbicos para P.V.I. lineales

```
function [c] = spline3p(a,b,n,x,p,q,r,z1,z2)
%
% La función spline3p resuelve EDOs de 2ºorden lineales con
% condiciones iniciales mediante aproximaciones obtenidas con.
% splines cúbicos.
%
% spline3p es una función que calcula los coeficientes de un
% spline cúbico S en el intervalo [a,b], basado en los n+1 nodos
% distintos y ordenados en forma creciente, contenidos en el
% vector x=(x_1,x_2,...,x_{n+1}), con x_1=a y x_{n+1}=b
%
% En cada intervalo [x_k, x_{k+1}] el spline S viene dado por un
% polinomio S_k de grado <=3 definido por:
% S_k(x)=c(k,4)+c(k,3)*(x-x_k)+c(k,2)*(x-x_k)^2+c(k,1)*(x-x_k)^3
% para cada k=1, 2,..., n
%
% INPUT:
%
% a = extremo inicial del intervalo
% b = extremo final del intervalo
% n = número de subintervalos de la partición del intervalo [a,b]
% x = array con los n+1 nodos distintos, ordenados en forma
%     creciente y con x_1=a y x_{n+1}=b
```

```
% p,q,r = coeficientes (definidos como funciones de la variable
%      independiente) de la EDO de 2° orden: y''=p*y'+q*y+r
%      cuya solución tratamos de aproximar con el spline S
% z1,z2 = valores de las condiciones iniciales de la EDO de 2°
%      orden: y(a)=z1, y'(a)=z2
%
% OUTPUT:
%
% c = matriz de dimensión n*4 que contiene los 4n coeficientes que
%      determinan de manera única el spline cúbico. La fila k-ésima
%      de c contiene los coeficientes de S_k en el intervalo I_k,
%      en la base: [(x-x_k)^3, (x-x_k)^2, (x-x_k), 1] (orden
%      decreciente de potencias, esto es:
%      S_k(x)=c(k,1)*(x-x_k)^3+c(k,2)*(x-x_k)^2+c(k,3)*(x-x_k)
%      +c(k,4).
%
% Queremos que nos muestre bastantes decimales:
format long
%
% La fila k-ésima de c almacenará los coeficientes de S_k en orden
% decreciente de potencias

c=zeros(n,4);
%
% la componente k-ésima de h almacenará h_k=x_{k+1}-x_k
h=zeros(n,1);
for i=1:n
    h(i)=x(i+1)-x(i);
end
%
% Los vectores P, Q y R almacenarán respectivamente los valores de
% los coeficientes de la ecuación: y''=p*y'+q*y+r
% Los valores en los nodos x_k se almacenarán en las componentes
% k-ésimas para k=1, 2, ..., n+1
P=zeros(n+1,1);
Q=zeros(n+1,1);
R=zeros(n+1,1);
for i=1:n+1
    P(i)=p(x(i));
    Q(i)=q(x(i));
    R(i)=r(x(i));
end
%
% Calculamos los coeficientes de S en el primer intervalo a partir
% de las dos condiciones iniciales y de imponer que se satisfaga
% la EDO en los dos nodos x_1 y x_2 (condiciones de colocación):

c(1,4)=z1;
c(1,3)=z2;
c(1,2)=0.5*(P(1)*c(1,3)+Q(1)*c(1,4)+R(1));
```

```
c(1,1)=(R(2)+Q(2)*c(1,4)+(P(2)+Q(2)*h(1))*c(1,3)+(-2+h(1)...
*(2*P(2)+Q(2)*h(1))*c(1,2))/(h(1)*(6-h(1)*(3*P(2)+Q(2)*h(1)));
%c(1,1)=(Q(2)*c(1,3)+(2*P(2)+Q(2)*h(1))*c(1,2)+...
% ((P(2)-P(1))*c(1,3)+(Q(2)-Q(1))*c(1,4)+...
% (R(2)-R(1)))/(h(1))/(6-h(1)*(3*P(2)+Q(2)*h(1)))
%
% Calculamos los coeficientes de S en los restantes intervalos de
% forma recursiva:
for i=2:n
    c(i,4)=c(i-1,4)+h(i-1)*(c(i-1,3)+h(i-1)*(c(i-1,2)+ ...
        c(i-1,1)*h(i-1)));
    c(i,3)=c(i-1,3)+h(i-1)*(2*c(i-1,2)+3*c(i-1,1)*h(i-1));
    c(i,2)=c(i-1,2)+3*c(i-1,1)*h(i-1);
    c(i,1)=(R(i+1)+Q(i+1)*c(i,4)+(P(i+1)+Q(i+1)*h(i))*c(i,3)+ ...
        (-2+h(i)*(2*P(i+1)+Q(i+1)*h(i))*c(i,2))/(h(i)*(6-...
        h(i)*(3*P(i+1)+Q(i+1)*h(i))));
%c(i,1)=(Q(i+1)*c(i,3)+(2*P(i+1)+Q(i+1)*h(i))*c(i,2)...
% +((P(i+1)-P(i))*c(i,3)+(Q(i+1)-Q(i))*c(i,4)+(R(i+1)-R(i)))...
% /(h(i))/(6-h(i)*(3*P(i+1)+Q(i+1)*h(i)))
end
end
```

## B. Método de resolución basado en Pseudo-Splines Cuárticos para P.V.I. lineales

```
function [c] = spline4p(a,b,n,x,p,q,r,z1,z2)
%
% La función spline4p resuelve EDOs de 2ª orden lineales con
% condiciones iniciales mediante aproximaciones obtenidas con
% pseudo splines cúarticos.
%
% spline4p es una función que calcula los coeficientes de un
% pseudo spline cuártico S en el intervalo [a,b], basado en los
% n+1 nodos distintos y ordenados en forma creciente, contenidos
% en el vector: x=(x_1,x_2,...,x_{n+1}), con x_1=a y x_{n+1}=b
%
% En cada intervalo [x_k, x_{k+1}] el spline S viene dado por un
% polinomio S_k de grado <=4 definido por:
% S_k(x)=c(k,5)+c(k,4)*(x-x_k)+c(k,3)*(x-x_k)^2+c(k,2)*(x-x_k)^3
% +c(k,1)*(x-x_k)^4
% para cada k=1, 2,..., n
% INPUT:
```

```
%
% a = extremo inicial del intervalo
% b = extremo final del intervalo
% n = número de subintervalos de la partición del intervalo [a,b]
% x = array con los n+1 nodos distintos, ordenados en forma
%     creciente y con x_1=a y x_{n+1}=b
% p,q,r = coeficientes (definidos como funciones de la variable
%         independiente) de la EDO de 2º orden: y'=p*y'+q*y+r
%         cuya solución tratamos de aproximar con el spline S
% z1,z2 = valores de las condiciones iniciales de la EDO de 2º
%         orden: y(a)=z1, y'(a)=z2
%
% OUTPUT:
%
% c = matriz de dimensión n*5 que contiene los 5n coeficientes que
%     determinan de manera única el pseudo spline cúartico. La
%     fila k-ésima de c contiene los coeficientes de S_k en el
%     intervalo I_k, en la base:
%     [(x-x_k)^4, (x-x_k)^3, (x-x_k)^2, (x-x_k), 1]
%     (orden decreciente de potencias), esto es:
%     S_k(x)=c(k,1)*(x-x_k)^4+c(k,2)*(x-x_k)^3+c(k,3)*(x-x_k)^2
%           +c(k,4)*(x-x_k)+c(k,5).
%
%
% Queremos que nos muestre bastantes decimales:
format long;
%
% La fila k-ésima de c almacenará los coeficientes de S_k en orden
% decreciente de potencias
c=zeros(n,5);
%
% la componente k-ésima de h almacenará h_k=x_{k+1}-x_k
h=zeros(n,1);

for i=1:n
    h(i)=x(i+1)-x(i);
end
%
% Los vectores P, Q y R almacenarán respectivamente los valores de
% los coeficientes de la ecuación: y'=p*y'+q*y+r
%
% Los valores en los nodos x_k se almacenarán en las componentes
% 2*k-1 para k=1, 2, ..., n+1
%
% Los valores en los puntos medios de los intervalos:
% (x_k+x_{k+1})/2 se almacenarán en las componentes 2*k para
% k=1, 2, ..., n

P=zeros(2*n+1,1);
```



```
Q=zeros(2*n+1,1);
R=zeros(2*n+1,1);

for i=1:n+1
    P(2*i-1)=p(x(i));
    Q(2*i-1)=q(x(i));
    R(2*i-1)=r(x(i));
end

for i=1:n
    P(2*i)=p(0.5*(x(i)+x(i+1)));
    Q(2*i)=q(0.5*(x(i)+x(i+1)));
    R(2*i)=r(0.5*(x(i)+x(i+1)));
end

%
% Calculamos los coeficientes de S en el primer intervalo a partir
% de las dos condiciones iniciales y de imponer que se satisfaga
% la EDO en los dos nodos x_1, x_2 y en el punto medio:

c(1,5)=z1;
c(1,4)=z2;
c(1,3)=0.5*(P(1)*c(1,4)+Q(1)*c(1,5)+R(1));

a11=6-h(1)*(3*P(3)+Q(3)*h(1));
a12=12-h(1)*(4*P(3)+Q(3)*h(1));
a21=3-h(1)*(0.75*P(2)+0.125*Q(2)*h(1));
a22=3-h(1)*(0.5*P(2)+0.0625*Q(2)*h(1));
aa=a11*a22-a12*a21;

b1=R(3)+Q(3)*c(1,5)+(P(3)+Q(3)*h(1))*c(1,4)+(-2+h(1)*(2*P(3)+...
    +Q(3)*h(1)))*c(1,3);
b2=R(2)+Q(2)*c(1,5)+(P(2)+0.5*Q(2)*h(1))*c(1,4)+(-2+h(1)*(P(2)+...
    +0.25*Q(2)*h(1)))*c(1,3);

c(1,2)=(a22*b1-a12*b2)/(h(1)*aa);
c(1,1)=(-a21*b1+a11*b2)/(h(1)^2*aa);

%
% Calculamos los coeficientes de S en los restantes intervalos de
% forma recursiva:
for i=2:n
    c(i,5)=c(i-1,5)+h(i-1)*(c(i-1,4)+h(i-1)*(c(i-1,3)+...
        h(i-1)*(c(i-1,2)+c(i-1,1)*h(i-1))));
    c(i,4)=c(i-1,4)+h(i-1)*(2*c(i-1,3)+h(i-1)*(3*c(i-1,2)+...
        4*c(i-1,1)*h(i-1)));
    c(i,3)=c(i-1,3)+h(i-1)*(3*c(i-1,2)+6*c(i-1,1)*h(i-1));

    a11=6-h(i)*(3*P(2*i+1)+Q(2*i+1)*h(i));
    a12=12-h(i)*(4*P(2*i+1)+Q(2*i+1)*h(i));
    a21=3-h(i)*(0.75*P(2*i)+0.125*Q(2*i)*h(i));
    a22=3-h(i)*(0.5*P(2*i)+0.0625*Q(2*i)*h(i));
    aa=a11*a22-a12*a21;
    b1=R(2*i+1)+Q(2*i+1)*c(i,5)+(P(2*i+1)+Q(2*i+1)*h(i))*c(i,4)...
```

```
        +(-2+h(i)*(2*P(2*i+1)+Q(2*i+1)*h(i)))*c(i,3);
b2=R(2*i)+Q(2*i)*c(i,5)+(P(2*i)+0.5*Q(2*i)*h(i))*c(i,4)...
        +(-2+h(i)*(P(2*i)+0.25*Q(2*i)*h(i)))*c(i,3);

c(i,2)=(a22*b1-a12*b2)/(aa*h(i));
c(i,1)=(-a21*b1+a11*b2)/(aa*h(i)^2);

end

end
```

## 7.1.2. Funciones para resolver Problemas con Valores en la Frontera Lineales

### A. Método de resolución basado en Splines Cúbicos para P.V.F. lineales con condiciones de contorno generales

```
function [c] = spline3pc2(AlgThomas,a,b,n,p,q,r,z1,z2)
%
% La función spline3pc2 resuelve EDOs de 2ª orden lineales con
% condiciones de contorno bastante generales.
%
% spline3pc2 es una función que calcula los coeficientes de un
% spline cúbico S en el intervalo [a,b], basado en n+1 nodos,
% distintos equiespaciados y ordenados en forma creciente:
% x_k=a+(k-1)h, con h=(b-a)/n, para: k=1, 2,..., n+1, contenidos
% en el vector x=(x_1,x_2,...,x_{n+1}).
% Por tanto: x_1=a y x_{n+1}=b.
%
% En cada intervalo I_k=[x_k, x_{k+1}] el spline S viene dado por un
% polinomio S_k de grado <=3 definido por:
% S_k(x)=(1/h)*[M_{k+1}*(x-x_k)^3+M_k*(x_{k+1}-x)^3
%             +N_{k+1}*(x-x_k)+N_k*(x_{k+1}-x)]
% para cada k=1, 2,..., n.
%
% Es fácil comprobar que con la descripción anterior de S, tanto S
% como S'' son continuas en [a,b] independientemente de los de los
% valores parámetros M_k y N_k que tomemos (k=1, 2,..., n+1).
%
% Por tanto, para determinar S, bastará imponer la continuidad de
% S' en los n-1 nodos interiores, las n+1 condiciones de
%
% colocación consistente en imponer que S satisfaga la EDO de 2º
% orden en los n+1 nodos y las dos condiciones de contorno y/o
```

```
% iniciales que garantizan que la EDO de 2° orden posee una única
% solución.
%
% La continuidad de S' y las dos condiciones de contorno y/o
% iniciales permiten expresar los M_k en términos de los N_k.
% Basta pues resolver un sistema lineal tridiagonal de dimensión
% n+1 en las incógnitas N_k, lo que haremos con la ayuda del
% algoritmo de Thomas (implementado en la función AlgThomas) por
% su bajo coste computacional.
%
%
% INPUT:
%
% AlgThomas = función que implementa el algoritmo de Thomas
% a = extremo inicial del intervalo
% b = extremo final del intervalo
% n = número de subintervalos de la partición del intervalo [a,b]
% p,q,r = coeficientes (definidos como funciones de la variable
%         independiente) de la EDO de 2° orden: y''=p*y'+q*y+r
%         cuya solución tratamos de aproximar con el spline S
% z1,z2 = valores de las condiciones iniciales y/o de contorno de
%         la EDO de 2° orden descritas en forma de lista en la
%         forma:
%         En x=a, z1=[a1,b1,d1] -> denota: a1*y(a)+b1*y'(a)=d1
%         En x=b, z2=[a2,b2,d2] -> denota: a2*y(b)+b2*y'(b)=d2
%
%
% OUTPUT:
%
% c = matriz de dimensión n*4 que contiene los 4n coeficientes que
%     determinan de manera única el spline cúbico. La fila k-ésima
%     de c contiene los coeficientes de S_k en el intervalo I_k,
%     en la base: [(x-x_k)^3, (x-x_k)^2, (x-x_k), 1] (orden
%     decreciente de potencias), esto es:
%     S_k(x)=c(k,1)*(x-x_k)^3+c(k,2)*(x-x_k)^2+...
%             c(k,3)*(x-x_k)+c(k,4).
%
%
% Queremos que nos muestre bastantes decimales:
format long
%
% La fila k-ésima de c almacenará los coeficientes de S_k en la
% base: [(x-x_k)^3, (x-x_k)^2, (x-x_k), 1] (orden decreciente de
% potencias)
c=zeros(n,4);
%
```

```
% Los n+1 nodos equiespaciados se almacenarán en x
h=(b-a)/n;
x=zeros(n+1,1);
x(1)=a;
x(n+1)=b;
for i=2:n
    x(i)=a+(i-1)*h;
end
%
% Los vectores P, Q y R almacenarán respectivamente los valores de
% los coeficientes de la ecuación:  $y''=p*y'+q*y+r$ , en los nodos
%
% Los valores en los nodos  $x_k$  se almacenarán en las componentes
% k-ésimas para  $k=1, 2, \dots, n+1$ 
P=zeros(n+1,1);
Q=zeros(n+1,1);
R=zeros(n+1,1);
for i=1:n+1
    P(i)=p(x(i));
    Q(i)=q(x(i));
    R(i)=r(x(i));
end
%
% M y N contendrán los coeficientes de S en la base original.
% T es una matriz de dimensión  $4*(n+1)$  que contendrá los
% coeficientes no nulos del sistema lineal tridiagonal que hay que
% resolver para obtener los  $n+1$  coeficientes  $N_k$ . Concretamente:
% Las tres primeras filas de T contienen los elementos de las
% diagonales (de la más baja a la más alta) de la matriz
% tridiagonal de dimensión  $n+1$  que deseamos resolver:
%  $T(1,:)=[0, a_2, \dots, a_{n+1}]$ 
%  $T(2,:)=[b_1, \dots, b_{n+1}]$ 
%  $T(3,:)=[c_1, \dots, c_n, 0]$ 
%  $T(4,:)=[d_1, \dots, d_{n+1}]$  contiene el vector de términos
% independientes
M=zeros(n+1,1);
N=zeros(n+1,1);
T=zeros(4,n+1);
T(1,1)=0;
T(2,1)=2*(3+Q(1)*h^2)*z1(2)-2*(3+P(1)*h)*z1(1)*h;
T(3,1)=-((6-Q(1)*h^2)*z1(2)+P(1)*z1(1)*h^2);
T(4,1)=-((6+h*(3*P(1)-Q(1)*h))*z1(3)*h+R(1)*(-3*z1(2)+z1(1)*h)*h^2;
```

```
T(1,n+1)=(6-Q(n+1)*h^2)*z2(2)+P(n+1)*z2(1)*h^2;
T(2,n+1)=-2*(3+Q(n+1)*h^2)*z2(2)+2*(-3+P(n+1)*h)*z2(1)*h;
T(3,n+1)=0;
T(4,n+1)= ...
(-6+h*(3*P(n+1)+Q(n+1)*h))*z2(3)*h+R(n+1)*(3*z2(2)+z2(1)*h)*h^2;

for i=2:n
    T(1,i)=6+h*(3*P(i)-Q(i)*h);
    T(2,i)=-4*(3+Q(i)*h^2);
    T(3,i)=6-h*(3*P(i)+Q(i)*h);
    T(4,i)=6*R(i)*h^2;
end

%
% Resolvemos el sistema tridiagonal mediante el algoritmo de
% Thomas

N=AlgThomas(T(1,:),T(2,:),T(3,:),T(4,:));
%
% Obtenemos los M_k a partir de los N_k (utilizando las n-1
% condiciones obtenidas al imponer que S' es continua y las 2
% condiciones iniciales y/o de contorno contenidas en z1, z2

M(1)=(-z1(2)*N(2)+(z1(2)-z1(1)*h)*N(1)+z1(3)*h)/((-
3*z1(2)+z1(1)*h)*h^2);

M(n+1)=(z2(2)*N(n)-
(z2(2)+z2(1)*h)*N(n+1)+z2(3)*h)/((3*z2(2)+z2(1)*h)*h^2);

for i=2:n
    M(i)=(N(i+1)-2*N(i)+N(i-1))/(6*h^2);
end

%
% Calculamos los coeficientes de S_k en el intervalo I_k, en la
% base: [(x-x_k)^3, (x-x_k)^2, (x-x_k), 1] (orden decreciente de
% potencias), a partir de los coeficientes M_k y N_k de la base
% original

for i=1:n
    c(i,1)=(M(i+1)-M(i))/h;
    c(i,2)=3*M(i);
    c(i,3)=-3*M(i)*h+(N(i+1)-N(i))/h;
    c(i,4)=N(i)+M(i)*h^2;
end

end
```

## B. Método de resolución basado en Pseudo-Splines Cuárticos para P.V.F. lineales con condiciones de contorno generales

```
function [c] = spline4pc2(AlgThomas,a,b,n,p,q,r,z1,z2)
%
% La función spline4pc2 resuelve EDOs de 2ª orden lineales con
% condiciones de contorno bastante generales.
%
% spline4pc2 es una función que calcula los coeficientes de un
% pseudo spline cuártico S en el intervalo [a,b], basado en n+1
% nodos distintos, equiespaciados y ordenados en forma creciente:
% x_k=a+(k-1)h, con h=(b-a)/n, para: k=1, 2,..., n+1, contenidos
% en el vector x=(x_1,x_2,...,x_{n+1}).
% Por tanto: x_1=a y x_{n+1}=b.
%
% En cada intervalo I_k=[x_k, x_{k+1}] el spline S viene dado por un
% polinomio S_k de grado <=4 definido por:
% S_k(x)=(1/h^4)*[A_{k+1}*(x-x_k)^4+A_k*(x_{k+1}-x)^4]
%         +(1/h^3)*[B_{k+1}*(x-x_k)^3+B_k*(x_{k+1}-x)^3]
%         +(1/h^2)*W_k*(x-x_k)*(x_{k+1}-x)*[1+(1/h^2)*(x-x_k)*(x_{k+1}-x)]
% para cada k=1, 2,..., n.
%
% En la salida del programa, los coeficientes del pseudo spline
% cuártico se darán, sin embargo, en cada intervalo [x_k, x_{k+1}]
% en la base:
% S_k(x)=c(k,5)+c(k,4)*(x-x_k)+c(k,3)*(x-x_k)^2+c(k,2)*(x-x_k)^3
%         +c(k,1)*(x-x_k)^4
% para cada k=1, 2,..., n
%
% Es fácil comprobar que con la descripción original de S, tanto S
% como S'' son continuas en [a,b] independientemente de los
% valores de los parámetros A_k, B_k (k=1, 2,..., n+1) y
% W_k (k=1, 2,..., n) que tomemos.
%
% Por tanto, para determinar S, bastará imponer la continuidad de
% S' en los n-1 nodos interiores, las 2*n+1 condiciones de
% colocación consistentes en imponer que S satisfaga la EDO de 2º
% orden en los n+1 nodos y en los puntos medios de los intervalos
% y las dos condiciones de contorno y/o iniciales que garantizan
% que la EDO de 2º orden posee una única solución.
%
% A partir de las n-1 condiciones de continuidad de S', las 2
% condiciones de contorno y las n+1 condiciones de colocación en
% los n+1 nodos, conseguimos despejar los A_k y los B_k
% (k=1, 2,..., n+1) en términos de los W_k.
```

```
% Llevando las expresiones obtenidas a las n ecuaciones
% correspondientes a las condiciones de colocación impuestas en
% los puntos medios de los n intervalos, obtenemos finalmente un
% sistema lineal tridiagonal en las n incógnitas W_k(k=1,2,..., n)
% que resolveremos con la ayuda del algoritmo de Thomas
% (implementado en la función AlgThomas) por su bajo coste
% computacional.
%
%
% INPUT:
%
% AlgThomas = función que implementa el algoritmo de Thomas
% a = extremo inicial del intervalo
% b = extremo final del intervalo
% n = número de subintervalos de la partición del intervalo [a,b]
% p,q,r = coeficientes (definidos como funciones de la variable
%         independiente) de la EDO de 2° orden: y''=p*y'+q*y+r
%         cuya solución tratamos de aproximar con el spline S
% z1,z2 = valores de las condiciones iniciales y/o de contorno de
%         la EDO de 2°orden descritas en forma de lista en la forma:
%         En x=a, z1=[z11,z12,z13] -> denota: z11*y(a)+z12*y'(a)=z13
%         En x=b, z2=[z21,z22,z23] -> denota: z21*y(b)+z22*y'(b)=z23
%
% OUTPUT:
%
% c = matriz de dimensión n*5 que contiene los 5n coeficientes que
%     determinan de manera única el spline cúbico en la base
%     considerada para la salida.
%
% Queremos que nos muestre bastantes decimales:
format long;
%
% La fila k-ésima de c almacenará los coeficientes de S_k en la
% base: [(x-x_k)^4, (x-x_k)^3, (x-x_k)^2, (x-x_k), 1] (orden
% decreciente de potencias)
c=zeros(n,5);
%
% Los n+1 nodos equiespaciados se almacenarán en x
h=(b-a)/n;
x=zeros(n+1,1);
x(1)=a;
x(n+1)=b;
for i=2:n
    x(i)=a+(i-1)*h;
end
%
```

```
% Los vectores P, Q y R almacenarán respectivamente los valores de
% los coeficientes de la ecuación: y''=p*y'+q*y+r
%
% Los valores en los nodos x_k se almacenarán en las componentes
% 2*k-1 para k=1, 2, ..., n+1
%
% Los valores en los puntos medios de los intervalos:
% (x_k+x_{k+1})/2 se almacenarán en las componentes 2*k para
% k=1, 2, ..., n
%
% Nótese que en ambos casos hemos escalado convenientemente con
% potencias de h o h^2 los valores para evitar posteriores
% multiplicaciones.

P=zeros(2*n+1,1);
Q=zeros(2*n+1,1);
R=zeros(2*n+1,1);

for i=1:n+1
    P(2*i-1)=h*p(x(i));
    Q(2*i-1)=h^2*q(x(i));
    R(2*i-1)=h^2*r(x(i));
end

for i=1:n
    P(2*i)=h*p(0.5*(x(i)+x(i+1)));
    Q(2*i)=h^2*q(0.5*(x(i)+x(i+1)));
    R(2*i)=h^2*r(0.5*(x(i)+x(i+1)));
end

%
%
% A, B y W contendrán los coeficientes de S en la base original.
% T es una matriz de dimensión 4*n que contendrá los coeficientes
% no nulos del sistema lineal tridiagonal que hay que resolver
% para obtener los n coeficientes W_k. Concretamente:
%
% Las tres primeras filas de T contienen los elementos de las
% diagonales (de la más baja a la más alta) de la matriz
% tridiagonal de dimensión n que deseamos resolver:
% T(1,:)= [0, a_2, ..., a_n]
% T(2,:)= [b_1, ..., b_n]
% T(3,:)= [c_1, ..., c_{n-1}, 0]
% T(4,:)= [d_1, ..., d_n] contiene el vector de términos
independientes

A=zeros(n+1,1);
B=zeros(n+1,1);
W=zeros(n,1);
T=zeros(4,n);
s=zeros(3);
```



```
s(1) = (48 - 24 * P(2) - 5 * Q(2)) / (24 + 2 * Q(3));
s(2) = (144 + 36 * P(2) - 6 * Q(2)) / (z1(1) - 3 * z1(2) / h);
s(3) = (P(1) * z1(1) + (6 - Q(1)) * z1(2) / h) / ...
      ((6 + P(1)) * z1(1) - (12 + Q(1)) * z1(2) / h);

T(1,1) = 0;
T(2,1) = (6 + 3 * P(3) - Q(3)) * s(1) + (3 * (-4 * P(2) + Q(2)) + ...
      s(2) * z1(2) / h) * s(3) - (120 + 6 * P(2) + 16 * Q(2)) - s(2) * z1(2) / h;
T(3,1) = (6 - 3 * P(3) - Q(3)) * s(1) + (24 - 6 * P(2) - Q(2));
T(4,1) = 48 * R(2) - s(2) * z1(3) + 6 * s(1) * R(3) + ...
      (3 * (-4 * P(2) + Q(2)) + s(2) * z1(2) / h) * ...
      ((6 + 3 * P(1) - Q(1)) * z1(3) - (z1(1) - 3 * z1(2) / h) * R(1)) / ...
      ((6 + P(1)) * z1(1) - (12 + Q(1)) * z1(2) / h);

s(1) = (48 + 24 * P(2 * n) - 5 * Q(2 * n)) / (24 + 2 * Q(2 * n - 1));
s(2) = (144 - 36 * P(2 * n) - 6 * Q(2 * n)) / (z2(1) + 3 * z2(2) / h);
s(3) = (-P(2 * n + 1) * z2(1) - (6 - Q(2 * n + 1)) * z2(2) / h) / ...
      ((6 - P(2 * n + 1)) * z2(1) + (12 + Q(2 * n + 1)) * z2(2) / h);

T(1,n) = (6 + 3 * P(2 * n - 1) - Q(2 * n - 1)) * s(1) + (24 + 6 * P(2 * n) - Q(2 * n));
T(2,n) = (6 - 3 * P(2 * n - 1) - Q(2 * n - 1)) * s(1) + ...
      (3 * (4 * P(2 * n) + Q(2 * n)) - s(2) * z2(2) / h) * s(3) - ...
      (120 - 6 * P(2 * n) + 16 * Q(2 * n)) + s(2) * z2(2) / h;
T(3,n) = 0;
T(4,n) = 48 * R(2 * n) - s(2) * z2(3) + 6 * s(1) * R(2 * n - 1) - ...
      (3 * (4 * P(2 * n) + Q(2 * n)) - s(2) * z2(2) / h) * ...
      ((-6 + 3 * P(2 * n + 1) + Q(2 * n + 1)) * z2(3) + (z2(1) + 3 * z2(2) / h) * R(2 * n + 1)) / ...
      ((6 - P(2 * n + 1)) * z2(1) + (12 + Q(2 * n + 1)) * z2(2) / h);

for i = 2 : n - 1
    s(1) = (48 + 24 * P(2 * i) - 5 * Q(2 * i)) / (24 + 2 * Q(2 * i - 1));
    s(2) = (48 - 24 * P(2 * i) - 5 * Q(2 * i)) / (24 + 2 * Q(2 * i + 1));

    T(1,i) = (24 + 6 * P(2 * i) - Q(2 * i)) + (6 + 3 * P(2 * i - 1) - Q(2 * i - 1)) * s(1);
    T(2,i) = (-96 - 17 * Q(2 * i)) + (6 - 3 * P(2 * i - 1) - Q(2 * i - 1)) * s(1) + ...
            (6 + 3 * P(2 * i + 1) - Q(2 * i + 1)) * s(2);
    T(3,i) = (24 - 6 * P(2 * i) - Q(2 * i)) + (6 - 3 * P(2 * i + 1) - Q(2 * i + 1)) * s(2);
    T(4,i) = 48 * R(2 * i) + 6 * (R(2 * i - 1) * s(1) + R(2 * i + 1) * s(2));
end
%
% Resolvemos el sistema tridiagonal mediante el algoritmo de
% Thomas
W = AlgThomas(T(1,:), T(2,:), T(3,:), T(4,:));
%
```

```
% Obtenemos los A_k y B_k a partir de los W_k (utilizando las n-1
% condiciones de continuidad de S', las 2 condiciones de contorno
% y las n+1 condiciones de colocación en los n+1 nodos)
A(1) = ((z1(1) - 3*z1(2)/h) * (R(1) + P(1) * W(1)) + (6 + 3*P(1) - Q(1)) * ...
        (z1(2) * W(1) / h - z1(3))) / ((6 + P(1)) * z1(1) - (12 + Q(1)) * z1(2) / h);

B(1) = ((-z1(1) + 4*z1(2)/h) * A(1) - z1(2) * W(1) / h + z1(3)) / ...
        (z1(1) - 3*z1(2)/h);

A(n+1) = ((z2(1) + 3*z2(2)/h) * (R(2*n+1) - P(2*n+1) * W(n)) - ...
          (6 - 3*P(2*n+1) - Q(2*n+1)) * (z2(2) * W(n) / h + z2(3))) / ...
          ((6 - P(2*n+1)) * z2(1) + (12 + Q(2*n+1)) * z2(2) / h);

B(n+1) = ((-z2(1) - 4*z2(2)/h) * A(n+1) + z2(2) * W(n) / h + z2(3)) / ...
          (z2(1) + 3*z2(2)/h);

for i=2:n
    A(i) = (6*R(2*i-1) - (6 + 3*P(2*i-1) - Q(2*i-1)) * W(i-1) - ...
           (6 - 3*P(2*i-1) - Q(2*i-1)) * W(i)) / (24 + 2*Q(2*i-1));

    B(i) = (-8*A(i) + W(i-1) + W(i)) / 6;
end
%
% Calculamos los coeficientes de S_k en el intervalo I_k, en la
% base: [(x-x_k)^4, (x-x_k)^3, (x-x_k)^2, (x-x_k), 1] (orden
% decreciente de potencias), a partir de los A_k, B_k y W_k de la
% base original

for i=1:n
    c(i,1) = (A(i+1) + A(i) + W(i)) / h^4;
    c(i,2) = (-4*A(i) + B(i+1) - B(i) - 2*W(i)) / h^3;
    c(i,3) = (6*A(i) + 3*B(i)) / h^2;
    c(i,4) = (-4*A(i) - 3*B(i) + W(i)) / h;
    c(i,5) = A(i) + B(i);
end

end
```

### C. Algoritmo de Thomas

En los dos métodos que acabamos de presentar en este apartado, se recurre, como ya se ha indicado, al algoritmo de Thomas para resolver el sistema lineal tridiagonal resultante, cuya solución permite definir el spline.

Se ha elegido este algoritmo por su bajo coste computacional, por su sencillez y su fácil implementación.

Es por ello que presentamos a continuación la función 'AlgThomas' mediante la cual implementamos dicho algoritmo y a la que hacemos llamada en los métodos de resolución anteriores.

```
function x = AlgThomas(a,b,c,d)

% a, b, c contienen los elementos de las diagonales de la matriz
% tridiagonal que deseamos resolver:
% a=[0, a2,..., an], b=[b1,..., bn], c=[c1,...,c_n-1, 0]
% d=[d1,..., dn] contiene el vector de términos independientes
% N contiene la dimensión del sistema lineal a resolver
N = length(d);

% Supuesto que b1 no es 0:
c(1)=c(1)/b(1);
d(1)=d(1)/b(1);

for n=2:1:N
    temp=b(n)-a(n)*c(n-1);
    if (n<N)
        c(n)=c(n)/temp;
    end
    d(n)=(d(n)-a(n)*d(n-1))/temp;
end

% Now back substitute.
x(N)=d(N);

for n=(N - 1):-1:1
    x(n)=d(n)-c(n)*x(n+1);
end

end
```

### 7.1.3. Funciones para resolver Problemas de Valores Iniciales No Lineales

#### A. Método de resolución basado en Splines Cúbicos para P.V.I. No lineales

```
function [c] = spline3pnl(a,b,n,x,f,z1,z2)
%
% La función spline3pnl resuelve EDOs de 2ª orden no lineales con
% condiciones iniciales mediante aproximaciones obtenidas con
% splines cúbicos.
%
% spline3pnl es una función que calcula los coeficientes de un
% spline cúbico S en el intervalo [a,b], basado en los n+1 nodos y
% distintos ordenados en forma creciente, contenidos en el vector
% x=(x_1,x_2,...,x_{n+1}), con x_1=a y x_{n+1}=b
%
% En cada intervalo [x_k, x_{k+1}] el spline S viene dado por un
% polinomio S_k de grado <=3 definido por:
% S_k(x)=c(k,4)+c(k,3)*(x-x_k)+c(k,2)*(x-x_k)^2+c(k,1)*(x-x_k)^3
% para cada k=1, 2,..., n
%
%
% INPUT:
%
% a = extremo inicial del intervalo
% b = extremo final del intervalo
% n = número de subintervalos de la partición del intervalo [a,b]
% x = array con los n+1 nodos distintos, ordenados en forma
%     creciente y con x_1=a y x_{n+1}=b
% f = función que define el 2º miembro de la EDO de 2º orden:
%     y''=f(x,y,y') cuya solución tratamos de aproximar con el
%     spline S
% z1,z2 = valores de las condiciones iniciales de la EDO de 2º
%         orden: y(a)=z1, y'(a)=z2
%
% OUTPUT:
%
% c = matriz de dimensión n*4 que contiene los 4n coeficientes que
%     determinan de manera única el spline cúbico. La fila k-ésima
%     de c contiene los coeficientes de S_k en el intervalo I_k,
%     en la base: [(x-x_k)^3, (x-x_k)^2, (x-x_k), 1] (orden
%     decreciente de potencias), esto es:
%     S_k(x)=c(k,1)*(x-x_k)^3+c(k,2)*(x-x_k)^2+...
%     c(k,3)*(x-x_k)+c(k,4).
```

```
% Queremos que nos muestre bastantes decimales:
format long
%
% La fila k-ésima de c almacenará los coeficientes de S_k en orden
% decreciente de potencias
c=zeros(n,4);
%
% la componente k-ésima de h almacenará h_k=x_{k+1}-x_k
h=zeros(n,1);
for i=1:n
    h(i)=x(i+1)-x(i);
end
% Calculamos los coeficientes de S en el primer intervalo a partir
% de las dos condiciones iniciales y de imponer que se satisfaga
% la EDO en los dos nodos x_1 y x_2 (condiciones de colocación):
c(1,4)=z1;
c(1,3)=z2;
c(1,2)=0.5*f(x(1),z1,z2);
%
% Resolvemos la ecuación (posiblemente no lineal) respecto de la
% única incógnita c(1,1) utilizando el comando fsolve de Matlab
F=@(z) -f(x(2),c(1,4)+h(1)*(c(1,3)+h(1)*(c(1,2)+h(1)*z)),...
        c(1,3)+h(1)*(2*c(1,2)+3*h(1)*z))+2*c(1,2)+6*h(1)*z;
c(1,1)=fsolve(F,0,optimset('Display','off','TolFun',
        1e-15,'TolX',1e-15));
%
% Calculamos los coeficientes de S en los restantes intervalos de
% forma recursiva:
for i=2:n
    c(i,4)=c(i-1,4)+h(i-1)*(c(i-1,3)+h(i-1)*(c(i-1,2)+...
        c(i-1,1)*h(i-1)));
    c(i,3)=c(i-1,3)+h(i-1)*(2*c(i-1,2)+3*c(i-1,1)*h(i-1));
    c(i,2)=c(i-1,2)+3*c(i-1,1)*h(i-1);
    %
    % Resolvemos la ecuación (posiblemente no lineal) respecto de
    % la única incógnita c(i,1) utilizando el comando fsolve de
    % Matlab
    F=@(z) -f(x(i+1),c(i,4)+h(i)*(c(i,3)+h(i)*(c(i,2)+h(i)*z)),
            c(i,3)+h(i)*(2*c(i,2)+3*h(i)*z))+2*c(i,2)+6*h(i)*z;
    c(i,1)=fsolve(F,c(i-1,1),optimset('Display','off','TolFun',
            1e-15,'TolX',1e-15));
end
end
```

## B. Método de resolución basado en Pseudo-Splines Cuárticos para P.V.I. no lineales

```
function [c] = spline4pnl(a,b,n,x,f,z1,z2)
%
% La función spline4pnl resuelve EDOs de 2ª orden no lineales con
% condiciones iniciales mediante aproximaciones obtenidas con
% pseudo splines cúarticos.
%
% spline4pnl es una función que calcula los coeficientes de un
% pseudo spline cuártico S en el intervalo [a,b], basado en los
% n+1 nodos distintos y ordenados en forma creciente, contenidos
% en el vector: x=(x_1,x_2,...,x_{n+1}), con x_1=a y x_{n+1}=b
%
% En cada intervalo [x_k, x_{k+1}] el spline S viene dado por un
% polinomio S_k de grado <=4 definido por:
% S_k(x)=c(k,5)+c(k,4)*(x-x_k)+c(k,3)*(x-x_k)^2+c(k,2)*(x-x_k)^3
%       +c(k,1)*(x-x_k)^4
% para cada k=1, 2,..., n
%
%
% INPUT:
%
% a = extremo inicial del intervalo
% b = extremo final del intervalo
% n = número de subintervalos de la partición del intervalo [a,b]
% x = array con los n+1 nodos distintos, ordenados en forma
%     creciente y con x_1=a y x_{n+1}=b
% f = función que define el 2º miembro de la EDO de 2º orden:
%     y''=f(x,y,y') cuya solución tratamos de aproximar con el
%     spline S
% z1,z2 = valores de las condiciones iniciales de la EDO de 2º
%         orden: y(a)=z1, y'(a)=z2
%
% OUTPUT:
%
% c = matriz de dimensión n*5 que contiene los 5n coeficientes que
%     determinan de manera única el pseudo spline cúartico. La
%     fila k-ésima de c contiene los coeficientes de S_k en el
%     intervalo I_k, en la base:
%     [(x-x_k)^4, (x-x_k)^3, (x-x_k)^2, (x-x_k), 1] (orden
%     decreciente de potencias), esto es:
%     S_k(x)=c(k,1)*(x-x_k)^4+c(k,2)*(x-x_k)^3+c(k,3)*(x-x_k)^2
%           +c(k,4)*(x-x_k)+c(k,5).
%
%
```

```
% Queremos que nos muestre bastantes decimales:
format long;
%
% La fila k-ésima de c almacenará los coeficientes de S_k en orden
% decreciente de potencias
c=zeros(n,5);
%
% la componente k-ésima de h almacenará h_k=x_{k+1}-x_k
h=zeros(n,1);
for i=1:n
    h(i)=x(i+1)-x(i);
end
%
% Calculamos los coeficientes de S en el primer intervalo a partir
% de las dos condiciones iniciales y de imponer que se satisfaga
% la EDO en los dos nodos x_1, x_2 y en el punto medio:
c(1,5)=z1;
c(1,4)=z2;
c(1,3)=0.5*f(x(1),z1,z2);
%
% Resolvemos el sistema de dos ecuaciones (posiblemente no
% lineales) que surgen de imponer las condiciones de colocación en
% los puntos medio y final del intervalo, respecto de las
% incógnita c(1,1) y c(1,2) utilizando el comando fsolve de Matlab
sol=zeros(2,1);
X=zeros(2,1);
H=0.5*h(1);

F=@(X)[-f(x(1)+H,c(1,5)+H*(c(1,4)+H*(c(1,3)+H*(X(2)+X(1)*H))),...
    c(1,4)+H*(2*c(1,3)+H*(3*X(2)+4*X(1)*H)))+2*c(1,3)+...
    H*(6*X(2)+12*X(1)*H); -f(x(2),c(1,5)+h(1)*(c(1,4)+h(1)*...
    (c(1,3)+h(1)*(X(2)+X(1)*h(1))),c(1,4)+h(1)*(2*c(1,3)+h(1)*...
    (3*X(2)+4*X(1)*h(1)))+2*c(1,3)+h(1)*(6*X(2)+12*X(1)*h(1))];

sol=fsolve(F,[0;0],optimset('Display','off','TolFun',1e-15,
    'TolX',1e-15));

c(1,1)=sol(1);
c(1,2)=sol(2);
%
% Calculamos los coeficientes de S en los restantes intervalos de
% forma recursiva:
for i=2:n
    c(i,5)=c(i-1,5)+h(i-1)*(c(i-1,4)+h(i-1)*(c(i-1,3)+h(i-1)*...
        (c(i-1,2)+c(i-1,1)*h(i-1))));
```

```
c(i,4)=c(i-1,4)+h(i-1)*(2*c(i-1,3)+h(i-1)*(3*c(i-1,2)+...
    4*c(i-1,1)*h(i-1)));
c(i,3)=c(i-1,3)+h(i-1)*(3*c(i-1,2)+6*c(i-1,1)*h(i-1));
%
% Resolvemos el sistema de dos ecuaciones (posiblemente no
% lineales) que surgen de imponer las condiciones de colocación
% en los puntos medio y final del intervalo, respecto de las
% incógnita c(i,1) y c(i,2) utilizando el comando fsolve de
% Matlab

H=0.5*h(i);

F=@(X) [-f(x(i)+H, c(i,5)+H*(c(i,4)+H*(c(i,3)+H*(X(2)+...
    X(1)*H))), c(i,4)+H*(2*c(i,3)+H*(3*X(2)+4*X(1)*H)))+ ...
    2*c(i,3)+ H*(6*X(2)+12*X(1)*H);
-f(x(i+1), c(i,5)+h(i)*(c(i,4)+h(i)*(c(i,3)+ h(i)*...
    (X(2)+X(1)*h(i)))), c(i,4)+h(i)*(2*c(i,3)+h(i)*(3*X(2)+...
    4*X(1)*h(i))))+2*c(i,3)+h(i)*(6*X(2)+12*X(1)*h(i))];

sol=fsolve(F,[c(i-1,1); c(i-1,2)],optimset('Display','off',
    'TolFun',1e-15,'TolX',1e-15));

c(i,1)=sol(1);
c(i,2)=sol(2);

end

end
```



## 7.2. Métodos de resolución basados en el Método del Disparo

Para resolver el único caso de problemas que aún no hemos explicado, es decir, Problemas de Valores de Frontera No Lineales, debido a su complejidad, nos apoyaremos en el Método del Disparo no Lineal.

Pero antes, comenzaremos resolviendo Problemas de Valores de Frontera Lineales, por un procedimiento diferente al ya visto en los códigos anteriores, que se basa en la utilización del Método del Disparo (Lineal). Esto facilitará la posterior comprensión del caso No Lineal.

### 7.2.1. Funciones para resolver Problemas con Valores de Frontera Lineales con condiciones generales

En los dos ejemplos que se muestran en este apartado, para la resolución del P.V.F., se resuelven previamente 3 Problemas de Valores Iniciales, para lo cual recurrimos al uso de los métodos de resolución basados en Splines que hemos explicado anteriormente.

#### A. Método del Disparo para P.V.F. lineales, con condiciones de contorno generales, utilizando Splines Cúbicos

```
function [c] = spline3pc(spline3p,a,b,n,x,p,q,r,z1,z2)
%
% La función spline3pc resuelve EDOs de 2ª orden lineales con
% condiciones de contorno bastante generales mediante la técnica
% del disparo lineal basada en aproximaciones obtenidas mediante
% splines cúbicos.
%
% spline3pc es una función que calcula los coeficientes de un
% spline cúbico S en el intervalo [a,b], basado en los n+1 nodos
% distintos y ordenados en forma creciente, contenidos en el
% vector: x=(x_1,x_2,...,x_{n+1}), con x_1=a y x_{n+1}=b
%
% En cada intervalo [x_k, x_{k+1}] el spline S viene dado por un
% polinomio S_k de grado <=3 definido por:
% S_k(x)=c(k,4)+c(k,3)*(x-x_k)+c(k,2)*(x-x_k)^2+c(k,1)*(x-x_k)^3
% para cada k=1, 2,..., n
%
%
```

```
% INPUT:
%
% spline3p = función que calcula los coeficientes del spline
%           cúbico para problemas de valores iniciales
% a = extremo inicial del intervalo
% b = extremo final del intervalo
% n = número de subintervalos de la partición del intervalo [a,b]
% x = array con los n+1 nodos distintos, ordenados en forma
%     creciente y con x_1=a y x_{n+1}=b
% p,q,r = coeficientes (definidos como funciones de la variable
%           independiente) de la EDO de 2° orden: y''=p*y'+q*y+r
%           cuya solución tratamos de aproximar con el spline S
% z1,z2 = valores de las condiciones iniciales y/o de contorno de
%           la EDO de 2°orden descritas en forma de lista en la forma:
%           En x=a, z1=[z11,z12,z13] -> denota: z11*y(a)+z12*y'(a)=z13
%           En x=b, z2=[z21,z22,z23] -> denota: z21*y(b)+z22*y'(b)=z23
%
% OUTPUT:
%
% c = matriz de dimensión n*4 que contiene los 4n coeficientes que
%     determinan de manera única el spline cúbico. La fila k-ésima
%     de c contiene los coeficientes de S_k en el intervalo I_k,
%     en la base: [(x-x_k)^3, (x-x_k)^2, (x-x_k), 1] (orden
%     decreciente de potencias), esto es:
%     S_k(x)=c(k,1)*(x-x_k)^3+c(k,2)*(x-x_k)^2+...
%            c(k,3)*(x-x_k)+c(k,4).
%
%
% La función spline3pc resuelve EDOs de 2ª orden lineales con
% condiciones de contorno generales utilizando 3 llamadas a la
% función spline3p para resolver 3 problemas de valores iniciales
% con la misma EDO de 2° orden y utilizando los mismos nodos y la
% misma base. A partir de una combinación de estas tres soluciones
% se obtiene la solución del problema original.
%
%
% Queremos que nos muestre bastantes decimales:
format long;
%
% La fila k-ésima de c almacenará los coeficientes de S_k en orden
% decreciente de potencias
c=zeros(n,4);
%
% La fila k-ésima de ck almacenará los coeficientes de S_k, en
% orden decreciente de potencias, tras la k-ésima llamada a la
% función spline3p

c1=zeros(n,4);
```

```
c2=zeros(n,4);
c3=zeros(n,4);
%
% Resolvemos los tres problemas de valores iniciales y almacenamos
% las matrices ck que contienen sus coeficientes en la base
% considerada:
c1=spline3p(a,b,n,x,p,q,r,0,0);
c2=spline3p(a,b,n,x,p,q,@(x) 0,0,1);
c3=spline3p(a,b,n,x,p,q,@(x) 0,1,0);
%
% Calculamos los valores de los splines y de su 1ª derivada en el
% punto final b
% yk1 almacena el valor del k-ésimo spline en b
% yk2 almacena el valor de la 1ª derivada del k-ésimo spline en b
h=x(n+1)-x(n);
y11=c1(n,4)+h*(c1(n,3)+h*(c1(n,2)+h*c1(n,1)));
y12=c1(n,3)+h*(2*c1(n,2)+h*3*c1(n,1));
y21=c2(n,4)+h*(c2(n,3)+h*(c2(n,2)+h*c2(n,1)));
y22=c2(n,3)+h*(2*c2(n,2)+h*3*c2(n,1));
y31=c3(n,4)+h*(c3(n,3)+h*(c3(n,2)+h*c3(n,1)));
y32=c3(n,3)+h*(2*c3(n,2)+h*3*c3(n,1));
%
% Obtenemos la matriz de dimensión n*4 que contiene los 4n
% coeficientes que determinan de manera única el spline cúbico en
% la base considerada:
d=z1(2)*(z2(1)*y31+z2(2)*y32)-z1(1)*(z2(1)*y21+z2(2)*y22);
d2=(z1(3)*(z2(1)*y31+z2(2)*y32)-z1(1)*...
(z2(3)-(z2(1)*y11+z2(2)*y12)))/d;
d3=(-z1(3)*(z2(1)*y21+z2(2)*y22)+z1(2)*...
(z2(3)-(z2(1)*y11+z2(2)*y12)))/d;
c=c1+d2*c2+d3*c3;
end
```

## B. Método del Disparo para P.V.F. lineales, con condiciones de contorno generales, utilizando Pseudo-Splines Cuárticos

```
function [c] = spline4pc(spline4p,a,b,n,x,p,q,r,z1,z2)
%
% La función spline4pc resuelve EDOs de 2ª orden lineales con
% condiciones de contorno bastante generales mediante la técnica
% del disparo lineal basada en aproximaciones obtenidas mediante
% pseudo splines cúarticos.
%
% spline4pc es una función que calcula los coeficientes de un
```

```
% pseudo spline cuártico S en el intervalo [a,b], basado en los
% n+1 nodos distintos y ordenados en forma creciente, contenidos
% en el vector: x=(x_1,x_2,...,x_{n+1}), con x_1=a y x_{n+1}=b
%
% En cada intervalo [x_k, x_{k+1}] el spline S viene dado por un
% polinomio S_k de grado <=4 definido por:
% S_k(x)=c(k,5)+c(k,4)*(x-x_k)+c(k,3)*(x-x_k)^2+c(k,2)*(x-x_k)^3
%       +c(k,1)*(x-x_k)^4
% para cada k=1, 2,..., n
%
%
% INPUT:
%
% spline4p = función que calcula los coeficientes del pseudo
%           spline cuártico para problemas de valores iniciales
% a = extremo inicial del intervalo
% b = extremo final del intervalo
% n = número de subintervalos de la partición del intervalo [a,b]
% x = array con los n+1 nodos distintos, ordenados en forma
%     creciente y con x_1=a y x_{n+1}=b
% p,q,r = coeficientes (definidos como funciones de la variable
%         independiente) de la EDO de 2° orden: y''=p*y'+q*y+r
%         cuya solución tratamos de aproximar con el spline S
% z1,z2 = valores de las condiciones iniciales y/o de contorno de
%         la EDO de 2°orden descritas en forma de lista en la forma:
%         En x=a, z1=[z11,z12,z13] -> denota: z11*y(a)+z12*y'(a)=z13
%         En x=b, z2=[z21,z22,z23] -> denota: z21*y(b)+z22*y'(b)=z23
%
% OUTPUT:
%
% c = matriz de dimensión n*5 que contiene los 5n coeficientes que
%     determinan de manera única el pseudo spline cuártico. La
%     fila k-ésima de c contiene los coeficientes de S_k en el
%     intervalo I_k, en la base:
%     [(x-x_k)^4, (x-x_k)^3, (x-x_k)^2, (x-x_k), 1]
%     (orden decreciente de potencias), esto es:
%     S_k(x)=c(k,1)*(x-x_k)^4+c(k,2)*(x-x_k)^3+c(k,3)*(x-x_k)^2
%           +c(k,4)*(x-x_k)+c(k,5).
%
% La función spline4pc resuelve EDOs de 2ª orden lineales con
% de contorno generales utilizando 3 llamadas a la función
% condiciones spline4p para resolver 3 problemas de valores
% iniciales con la misma EDO de 2° orden y utilizando los mismos
% nodos y la misma base. A partir de una combinación de estas tres
% soluciones se obtiene la solución del problema original
%
% Queremos que nos muestre bastantes decimales:
format long;
```

```
%  
% La fila k-ésima de c almacenará los coeficientes de S_k en orden  
% decreciente de potencias  
c=zeros(n,5);  
%  
% La fila k-ésima de ck almacenará los coeficientes de S_k, en  
% orden decreciente de potencias, tras la k-ésima llamada a la  
% función spline4p  
c1=zeros(n,5);  
c2=zeros(n,5);  
c3=zeros(n,5);  
%  
% Resolvemos los tres problemas de valores iniciales y almacenamos  
% las matrices ck que contienen sus coeficientes en la base  
% considerada:  
c1=spline4p(a,b,n,x,p,q,r,0,0);  
c2=spline4p(a,b,n,x,p,q,@(x) 0,0,1);  
c3=spline4p(a,b,n,x,p,q,@(x) 0,1,0);  
%  
% Calculamos los valores de los splines y de su 1ª derivada en el  
% punto final b  
% yk1 almacena el valor del k-ésimo spline en b  
% yk2 almacena el valor de la 1ª derivada del k-ésimo spline en b  
h=x(n+1)-x(n);  
  
y11=c1(n,5)+h*(c1(n,4)+h*(c1(n,3)+h*(c1(n,2)+h*c1(n,1))));  
y12=c1(n,4)+h*(2*c1(n,3)+h*(3*c1(n,2)+h*4*c1(n,1)));  
y21=c2(n,5)+h*(c2(n,4)+h*(c2(n,3)+h*(c2(n,2)+h*c2(n,1))));  
y22=c2(n,4)+h*(2*c2(n,3)+h*(3*c2(n,2)+h*4*c2(n,1)));  
y31=c3(n,5)+h*(c3(n,4)+h*(c3(n,3)+h*(c3(n,2)+h*c3(n,1))));  
y32=c3(n,4)+h*(2*c3(n,3)+h*(3*c3(n,2)+h*4*c3(n,1)));  
%  
% Obtenemos la matriz de dimensión n*5 que contiene los 5n  
% coeficientes que determinan de manera única el spline cúbico en  
% la base considerada:  
d=z1(2)*(z2(1)*y31+z2(2)*y32)-z1(1)*(z2(1)*y21+z2(2)*y22);  
d2=(z1(3)*(z2(1)*y31+z2(2)*y32)-z1(1)*  
    (z2(3)-(z2(1)*y11+z2(2)*y12)))/d;  
d3=(-z1(3)*(z2(1)*y21+z2(2)*y22)+z1(2)*  
    (z2(3)-(z2(1)*y11+z2(2)*y12)))/d;  
c=c1+d2*c2+d3*c3;  
  
end
```

## 7.2.2. Funciones para resolver Problemas con Valores de Frontera No Lineales con condiciones de contorno simples

### A. Método del Disparo No Lineal para P.V.F. no lineales, con condiciones de contorno simples, utilizando Splines Cúbicos

```
function[c] =
    spline3pcnl(spline3p,spline3pnl,a,b,n,x,f,f1,f2,z1,z2,nmax)
%
% La función spline3pcnl resuelve EDOs de 2ª orden no lineales con
% condiciones de contorno sencillas mediante la técnica del
% disparo no lineal basada en aproximaciones obtenidas mediante
% splines cúbicos.
%
% spline3pcnl es una función que calcula los coeficientes de un
% spline cúbico S en el intervalo [a,b], basado en los n+1 nodos
% distintos y ordenados en forma creciente, contenidos en el
% vector: x=(x_1,x_2,...,x_{n+1}), con x_1=a y x_{n+1}=b
%
% En cada intervalo [x_k, x_{k+1}] el spline S viene dado por un
% polinomio S_k de grado <=3 definido por:
% S_k(x)=c(k,4)+c(k,3)*(x-x_k)+c(k,2)*(x-x_k)^2+c(k,1)*(x-x_k)^3
% para cada k=1, 2,..., n
%
%
% INPUT:
%
% spline3p = función que calcula los coeficientes del spline
%           cúbico para problemas de valores iniciales lineal
% spline3pnl = función que calcula los coeficientes del spline
%            cúbico para problemas de valores iniciales no
%            lineales
% a = extremo inicial del intervalo
% b = extremo final del intervalo
% n = número de subintervalos de la partición del intervalo [a,b]
% x = array con los n+1 nodos distintos, ordenados en forma
%     creciente y con x_1=a y x_{n+1}=b
% f = función que define el 2º miembro de la EDO de 2º orden:
%     y''=f(x,y,y') cuya solución tratamos de aproximar con el
%     spline S
% f1 = función que define la derivada parcial de f respecto de y:
%     df/dy. Concretamente: f1(x,y,y')= df/dy(x,y,y')
% f2 = función que define la derivada parcial de f respecto de y':
%     df/dy'. Concretamente: f2(x,y,y')= df/dy'(x,y,y')
% z1,z2 = valores de las condiciones de contorno sencillas de la
```

```
%          EDO de 2° orden:
%          En x=a, y(a)=z1
%          En x=b, y(b)=z2

% nmax = número máximo de iteraciones con el método de Newton
%
% OUTPUT:
%
% c = matriz de dimensión n*4 que contiene los 4n coeficientes que
%     determinan de manera única el spline cúbico. La fila k-ésima
%     de c contiene los coeficientes de S_k en el intervalo I_k,
%     en la base: [(x-x_k)^3, (x-x_k)^2, (x-x_k), 1] (orden
%     decreciente de potencias), esto es:
%     S_k(x)=c(k,1)*(x-x_k)^3+c(k,2)*(x-x_k)^2+
%             c(k,3)*(x-x_k)+c(k,4).
%
% La función spline3pcnl resuelve EDOs de 2ª orden no lineales con
% condiciones de contorno sencillas utilizando el método del
% disparo no lineal implementado a partir del método de Newton.
% Las aproximaciones a la solución en cada iteración se obtienen a
% partir de los métodos basados en splines cúbicos: spline3p y
% spline3pnl que se aplican para resolver los dos problemas de
% valores iniciales (lineal y no lineal respectivamente)
% utilizando los mismos nodos y la misma base.
%
%
% Queremos que nos muestre bastantes decimales:
format long;
%
% La fila k-ésima de c almacenará los coeficientes de S_k en orden
% decreciente de potencias
c=zeros(n,4);
%
% La fila k-ésima de cm almacenará los coeficientes de S_k, en
% orden decreciente de potencias, tras cada llamada a la función:
% spline3pnl en c1 y spline3p en c2
%
% La fila k-ésima de d1 almacenará los coeficientes de S'_k (el de
% c1), en orden decreciente de potencias, tras cada llamada a la
% función.
c1=zeros(n,4);
c2=zeros(n,4);
d1=zeros(n,3);
%
% Comienza la iteración por el método de Newton:
%
% Inicializamos el ángulo de disparo:
t=(z2-z1)/(b-a);
%
% En cada iteración del método de Newton resolvemos los dos
```

```
% problemas de valores iniciales asociados y almacenamos las
% matrices cm que contienen sus coeficientes en la base
% considerada:
%
for j=1:nmax
    %
    % Para el PVI no lineal aplicamos la función spline3pnl:
    c1=spline3pnl(a,b,n,x,f,z1,t);
    %
    % dl almacena los coeficientes de S'_k calculados a partir de
    % los de S_k (almacenados en c1)
    for i=1:3
        dl(:,i)=(4-i)*c1(:,i);
    end
    %
    % Obtenemos las funciones que proporcionan los coeficientes del
    % PVI lineal:
    pp1 = mkpp(x,c1);
    p1=@(z) ppval(pp1,z);
    pp2 = mkpp(x,dl);
    p2=@(z) ppval(pp2,z);
    F1=@(z) f1(z,p1(z),p2(z));
    F2=@(z) f2(z,p1(z),p2(z));
    %
    % Para el PVI lineal aplicamos la función spline3p:
    c2=spline3p(a,b,n,x,F2,F1,@(x) 0,0,1);
    %
    % Calculamos los valores de los splines en el punto final b.
    % yk almacena el valor del k-ésimo spline en b
    h=x(n+1)-x(n);
    y1=c1(n,4)+h*(c1(n,3)+h*(c1(n,2)+h*c1(n,1)));
    y2=c2(n,4)+h*(c2(n,3)+h*(c2(n,2)+h*c2(n,1)));
    %
    % Iteración del método de Newton para actualizar el ángulo de
    % disparo:
    t=t-(y1-z2)/y2;
end
%
% Obtenemos la matriz de dimensión n*4 que contiene los 4n
% coeficientes que determinan de manera única el spline cúbico en
% la base considerada:
c=c1;
end
```



## B. Método del Disparo No Lineal para P.V.F. no lineales con condiciones de contorno simples, utilizando Pseudo-Splines Cuárticos

```
function [c] =
    spline4pcnl(spline4p,spline4pnl,a,b,n,x,f,f1,f2,z1,z2,nmax)
%
%
% La función spline4pcnl resuelve EDOs de 2ª orden no lineales con
% condiciones de contorno sencillas mediante la técnica del
% disparo no lineal basada en aproximaciones obtenidas mediante
% pseudo splines cúarticos.
%
% spline4pcnl es una función que calcula los coeficientes de un
% pseudo spline cuártico S en el intervalo [a,b], basado en los
% n+1 nodos distintos y ordenados en forma creciente, contenidos
% en el vector: x=(x_1,x_2,...,x_{n+1}), con x_1=a y x_{n+1}=b
%
% En cada intervalo [x_k, x_{k+1}] el spline S viene dado por un
% polinomio S_k de grado <=4 definido por:
% S_k(x)=c(k,5)+c(k,4)*(x-x_k)+c(k,3)*(x-x_k)^2+c(k,2)*(x-x_k)^3
%         +c(k,1)*(x-x_k)^4
% para cada k=1, 2,..., n
%
%
% INPUT:
%
% spline4p = función que calcula los coeficientes del pseudo
%           spline cuártico para problemas de valores iniciales
%           lineal
% spline4pnl = función que calcula los coeficientes del pseudo
%             spline cuártico para problemas de valores iniciales
%             no lineales
% a = extremo inicial del intervalo
% b = extremo final del intervalo
% n = número de subintervalos de la partición del intervalo [a,b]
% x = array con los n+1 nodos distintos, ordenados en forma
%     creciente y con x_1=a y x_{n+1}=b
% f = función que define el 2º miembro de la EDO de 2º orden:
%     y''=f(x,y,y') cuya solución tratamos de aproximar con el
%     spline S
% f1 = función que define la derivada parcial de f respecto de y:
%     df/dy. Concretamente: f1(x,y,y')= df/dy(x,y,y')
% f2 = función que define la derivada parcial de f respecto de y':
%     df/dy'. Concretamente: f2(x,y,y')=df/dy' (x,y,y')
```

```
% z1,z2 = valores de las condiciones de contorno sencillas de la
% EDO de 2° orden:
% En x=a, y(a)=z1
% En x=b, y(b)=z2
% nmax = número máximo de iteraciones con el método de Newton
%
% OUTPUT:
%
% c = matriz de dimensión n*5 que contiene los 5n coeficientes que
% determinan de manera única el pseudo spline cúartico. La
% fila k-ésima de c contiene los coeficientes de S_k en el
% intervalo I_k, en la base:
% [(x-x_k)^4, (x-x_k)^3, (x-x_k)^2, (x-x_k), 1]
% (orden decreciente de potencias), esto es:
% S_k(x)=c(k,1)*(x-x_k)^4+c(k,2)*(x-x_k)^3+c(k,3)*(x-x_k)^2
% +c(k,4)*(x-x_k)+c(k,5).
%
% La función spline4pcnl resuelve EDOs de 2ª orden no lineales con
% condiciones de contorno sencillas utilizando el método del
% disparo no lineal implementado a partir del método de Newton.
% Las aproximaciones a la solución en cada iteración se obtienen a
% partir de los métodos basados en pseudo splines cúarticos:
% spline4p y spline4pnl que se aplican para resolver los dos
% problemas de valores iniciales (lineal y no lineal
% respectivamente) utilizando los mismos nodos y la misma base.
%
% Queremos que nos muestre bastantes decimales:
format long;
%
% La fila k-ésima de c almacenará los coeficientes de S_k en orden
% decreciente de potencias
c=zeros(n,5);
%
% La fila k-ésima de cm almacenará los coeficientes de S_k, en
% orden decreciente de potencias, tras cada llamada a la función:
% spline4pnl en c1 y spline4p en c2
%
% La fila k-ésima de d1 almacenará los coeficientes de S'_k (el de
% c1), en orden decreciente de potencias, tras cada llamada a la
% función.
c1=zeros(n,5);
c2=zeros(n,5);
d1=zeros(n,4);
%
% Comienza la iteración por el método de Newton:
%
% Inicializamos el ángulo de disparo:
t=(z2-z1)/(b-a);
%
```

```
% En cada iteración del método de Newton resolvemos los dos
% problemas de valores iniciales asociados y almacenamos las
% matrices cm que contienen sus coeficientes en la base
% considerada:
%
for j=1:nmax
    %
    % Para el PVI no lineal aplicamos la función spline4pnl:
    c1=spline4pnl(a,b,n,x,f,z1,t);
    %
    % d1 almacena los coeficientes de S'_k calculados a partir de
    % los de S_k (almacenados en c1)
    for i=1:4
        d1(:,i)=(5-i)*c1(:,i);
    end
    %
    % Obtenemos las funciones que proporcionan los coeficientes del
    % PVI lineal:
    pp1 = mkpp(x,c1);
    p1=@(z) ppval(pp1,z);
    pp2 = mkpp(x,d1);
    p2=@(z) ppval(pp2,z);
    F1=@(z) f1(z,p1(z),p2(z));
    F2=@(z) f2(z,p1(z),p2(z));
    %
    % Para el PVI lineal aplicamos la función spline4p:
    c2=spline4p(a,b,n,x,F2,F1,@(x) 0,0,1);
    %
    % Calculamos los valores de los pseudo-splines en el punto
    % final b y k almacena el valor del k-ésimo pseudo spline en b
    h=x(n+1)-x(n);
    y1=c1(n,5)+h*(c1(n,4)+h*(c1(n,3)+h*(c1(n,2)+h*c1(n,1))));
    y2=c2(n,5)+h*(c2(n,4)+h*(c2(n,3)+h*(c2(n,2)+h*c2(n,1))));
    %
    % Iteración del método de Newton para actualizar el ángulo de
    % disparo:
    t=t-(y1-z2)/y2;
end
%
% Obtenemos la matriz de dimensión n*5 que contiene los 4n
% coeficientes que determinan de manera única el pseudo spline
% cúartico en la base considerada:
c=c1;
end
```

### C. Método del Disparo No Lineal para P.V.F. no lineales, con condiciones de contorno especiales, utilizando Splines Cúbicos

Presentamos en este apartado dos variantes del método *spline3pcnl* que acabamos de presentar. Estas variantes consideran los casos en los que las condiciones de contorno, en lugar de venir dadas por el valor de la función en cada extremo del intervalo:  $y(a) = z_1$ ,  $y(b) = z_2$ , vienen definidas en uno de los dos extremos por su derivada.

El caso en el que ambos extremos estuvieran definidos a partir de derivadas de la función en los puntos extremos del intervalo, también sería sencillo de implementar modificando ligeramente cualquiera de los códigos que aquí presentamos, de modo que no vamos a incluirlo.

Comenzamos presentando el método *spline3pcnl2* que resuelve el caso en el cual la condición en el extremo final es la que está expresada mediante una derivada, es decir, el caso en que el problema tenga condiciones de contorno del tipo:  $y(a) = z_1$ ,  $y'(b) = z_2$ .

```
function [c] =
    spline3pcnl2(spline3p,spline3pnl,a,b,n,x,f,f1,f2,z1,z2,nmax)
%
% La función spline3pcnl2 resuelve EDOs de 2ª orden no lineales
% con condiciones de contorno sencillas mediante la técnica del
% disparo no lineal basada en aproximaciones obtenidas mediante
% splines cúbicos.
%
% spline3pcnl2 es una función que calcula los coeficientes de un
% spline cúbico S en el intervalo [a,b], basado en los n+1 nodos
% distintos y ordenados en forma creciente, contenidos en el
% vector: x=(x_1,x_2,...,x_{n+1}), con x_1=a y x_{n+1}=b
%
% En cada intervalo [x_k, x_{k+1}] el spline S viene dado por un
% polinomio S_k de grado <=3 definido por:
% S_k(x)=c(k,4)+c(k,3)*(x-x_k)+c(k,2)*(x-x_k)^2+c(k,1)*(x-x_k)^3
% para cada k=1, 2,..., n
%
% INPUT:
%
% spline3p = función que calcula los coeficientes del spline
%           cúbico para problemas de valores iniciales lineal
% spline3pnl = función que calcula los coeficientes del spline
%             problemas de valores iniciales no lineal
%
% a = extremo cúbico para inicial del intervalo
```

```
% b = extremo final del intervalo
% n = número de subintervalos de la partición del intervalo [a,b]
% x = array con los n+1 nodos distintos, ordenados en forma
%     creciente y con x_1=a y x_{n+1}=b
% f = función que define el 2º miembro de la EDO de 2º orden:
%     y''=f(x,y,y') cuya solución tratamos de aproximar con el
%     spline S
% f1 = función que define la derivada parcial de f respecto de y:
%     df/dy. Concretamente: f1(x,y,y')= df/dy(x,y,y')
% f2 = función que define la derivada parcial de f respecto de y':
%     df/dy'. Concretamente: f2(x,y,y')=df/dy'(x,y,y')
% z1,z2 = valores de las condiciones de contorno sencillas de la
%         EDO de 2º orden:
%         En x=a, y(a)=z1
%         En x=b, y'(b)=z2
% nmax = número máximo de iteraciones con el método de Newton
%
% OUTPUT:
%
% c = matriz de dimensión n*4 que contiene los 4n coeficientes que
%     determinan de manera única el spline cúbico. La fila k-ésima
%     de c contiene los coeficientes de S_k en el intervalo I_k,
%     en la base: [(x-x_k)^3, (x-x_k)^2, (x-x_k), 1] (orden
%     decreciente de potencias), esto es:
%     S_k(x)=c(k,1)*(x-x_k)^3+c(k,2)*(x-x_k)^2+
%             c(k,3)*(x-x_k)+c(k,4).
%
% La función spline3pcnl2 resuelve EDOs de 2ª orden no lineales
% con condiciones de contorno sencillas utilizando el método del
% disparo no lineal implementado a partir del método de Newton.
%
% Las aproximaciones a la solución en cada iteración se obtienen a
% partir de los métodos basados en splines cúbicos: spline3p y
% spline3pnl que se aplican para resolver los dos problemas de
% valores iniciales (lineal y no lineal respectivamente)
% utilizando los mismos nodos y la misma base.
%
%
% Queremos que nos muestre bastantes decimales:
format long;
%
% La fila k-ésima de c almacenará los coeficientes de S_k en orden
% decreciente de potencias
c=zeros(n,4);
%
```

```
% La fila k-ésima de cm almacenará los coeficientes de S_k, en
% orden decreciente de potencias, tras cada llamada a la función:
% spline3pnl en c1 y spline3p en c2

% La fila k-ésima de d1 almacenará los coeficientes de S'_k (el de
% c1), en orden decreciente de potencias, tras cada llamada a la
% función.
c1=zeros(n,4);
c2=zeros(n,4);
d1=zeros(n,3);
%
% Comienza la iteración por el método de Newton:
%
% Inicializamos el ángulo de disparo:
t=0;
%
% En cada iteración del método de Newton resolvemos los dos
% problemas de valores iniciales asociados y almacenamos las
% matrices cm que contienen sus coeficientes en la base
% considerada:
for j=1:nmax
    %
    % Para el PVI no lineal aplicamos la función spline3pnl:
    c1=spline3pnl(a,b,n,x,f,z1,t);
    %
    % d1 almacena los coeficientes de S'_k calculados a partir de
    % los de S_k (almacenados en c1)
    for i=1:3
        d1(:,i)=(4-i)*c1(:,i);
    end
    %
    % Obtenemos las funciones que proporcionan los coeficientes del
    % PVI lineal:
    pp1 = mkpp(x,c1);
    p1=@(z) ppval(pp1,z);
    pp2 = mkpp(x,d1);
    p2=@(z) ppval(pp2,z);
    F1=@(z) f1(z,p1(z),p2(z));
    F2=@(z) f2(z,p1(z),p2(z));
    %
    % Para el PVI lineal aplicamos la función spline3p:
    c2=spline3p(a,b,n,x,F2,F1,@(x) 0,0,1);
    %
    % Calculamos los valores de la derivada primera de los splines
    % en el punto final b.
    % yk almacena el valor de la derivada primera del k-ésimo
    % spline en b
    h=x(n+1)-x(n);
    y1=c1(n,3)+h*(2*c1(n,2)+h*3*c1(n,1));
```

```
y2=c2(n,3)+h*(2*c2(n,2)+h*3*c2(n,1));  
%  
% Iteración del método de Newton para actualizar el ángulo de  
% disparo:  
t=t-(y1-z2)/y2;  
end  
%  
% Obtenemos la matriz de dimensión n*4 que contiene los 4n  
% coeficientes que determinan de manera única el spline cúbico en  
% la base considerada:  
c=c1;  
end
```

Si en lugar de en el punto final, la condición que está dada mediante una derivada de la función es la correspondiente al punto inicial:  $y'(a) = z_1$ ,  $y(b) = z_2$ , recurriríamos al método: *spline3pcnl3*.

```
function [c] =  
spline3pcnl3(spline3p,spline3pnl,a,b,n,x,f,f1,f2,z1,z2,nmax)  
%  
% La función spline3pcnl3 resuelve EDOs de 2ª orden no lineales  
% con condiciones de contorno sencillas mediante la técnica del  
% disparo no lineal basada en aproximaciones obtenidas mediante  
% splines cúbicos.  
%  
% spline3pcnl3 es una función que calcula los coeficientes de un  
% spline cúbico S en el intervalo [a,b], basado en los n+1 nodos  
% distintos y ordenados en forma creciente, contenidos en el  
% vector: x=(x_1,x_2,...,x_{n+1}), con x_1=a y x_{n+1}=b  
%  
% En cada intervalo [x_k, x_{k+1}] el spline S viene dado por un  
% polinomio S_k de grado <=3 definido por:  
% S_k(x)=c(k,4)+c(k,3)*(x-x_k)+c(k,2)*(x-x_k)^2+c(k,1)*(x-x_k)^3  
% para cada k=1, 2,..., n  
%  
%  
% INPUT:  
%  
% spline3p = función que calcula los coeficientes del spline  
%           cúbico para problemas de valores iniciales lineal  
% spline3pnl = función que calcula los coeficientes del spline  
%             cúbico para problemas de valores iniciales no  
%             lineal
```

```
% a = extremo inicial del intervalo
% b = extremo final del intervalo
% n = número de subintervalos de la partición del intervalo [a,b]
% x = array con los n+1 nodos distintos, ordenados en forma
%     creciente y con x_1=a y x_{n+1}=b
% f = función que define el 2º miembro de la EDO de 2º orden:
%     y''=f(x,y,y') cuya solución tratamos de aproximar con el
%     spline S
% f1 = función que define la derivada parcial de f respecto de y:
%     df/dy. Concretamente: f1(x,y,y')= df/dy(x,y,y')
% f2 = función que define la derivada parcial de f respecto de y':
%     df/dy'. Concretamente: f2(x,y,y')=df/dy'(x,y,y')
% z1,z2 = valores de las condiciones de contorno sencillas de la
%         EDO de 2º orden:
%         En x=a, y'(a)=z1
%         En x=b, y'(b)=z2
% nmax = número máximo de iteraciones con el método de Newton
%
% OUTPUT:
%
% c = matriz de dimensión n*4 que contiene los 4n coeficientes que
%     determinan de manera única el spline cúbico. La fila k-ésima
%     de c contiene los coeficientes de S_k en el intervalo I_k,
%     en la base: [(x-x_k)^3, (x-x_k)^2, (x-x_k), 1] (orden
%     decreciente de potencias), esto es:
%     S_k(x)=c(k,1)*(x-x_k)^3+c(k,2)*(x-x_k)^2+
%     c(k,3)*(x-x_k)+c(k,4).
%
% La función spline3pcnl3 resuelve EDOs de 2ª orden no lineales
% con condiciones de contorno sencillas utilizando el método del
% disparo no lineal implementado a partir del método de Newton.
% Las aproximaciones a la solución en cada iteración se obtienen a
% partir de los métodos basados en splines cúbicos: spline3p y
% spline3pnl que se aplican para resolver los dos problemas de
% valores iniciales (lineal y no lineal respectivamente)
% utilizando los mismos nodos y la misma base.
%
%
% Queremos que nos muestre bastantes decimales:
format long;
%
% La fila k-ésima de c almacenará los coeficientes de S_k en orden
% decreciente de potencias
c=zeros(n,4);
%
% La fila k-ésima de cm almacenará los coeficientes de S_k, en
% orden decreciente de potencias, tras cada llamada a la función:
% spline3pnl en c1 y spline3p en c2
% La fila k-ésima de d1 almacenará los coeficientes de S'_k (el de
```



```
% c1), en orden decreciente de potencias, tras cada llamada a la
% función.
c1=zeros(n,4);
c2=zeros(n,4);
d1=zeros(n,3);
%
% Comienza la iteración por el método de Newton:
%
% Inicializamos el ángulo de disparo:
t=0;
%
% En cada iteración del método de Newton resolvemos los dos
% problemas de valores iniciales asociados y almacenamos las
% matrices cm que contienen sus coeficientes en la base
% considerada:
for j=1:nmax
    %
    % Para el PVI no lineal aplicamos la función spline3pnl:
    c1=spline3pnl(a,b,n,x,f,t,z1);
    %
    % d1 almacena los coeficientes de S'_k calculados a partir de
    % los de S_k (almacenados en c1)
    for i=1:3
        d1(:,i)=(4-i)*c1(:,i);
    end
    %
    % Obtenemos las funciones que proporcionan los coeficientes del
    % PVI lineal:
    pp1 = mkpp(x,c1);
    p1=@(z) ppval(pp1,z);
    pp2 = mkpp(x,d1);
    p2=@(z) ppval(pp2,z);
    F1=@(z) f1(z,p1(z),p2(z));
    F2=@(z) f2(z,p1(z),p2(z));
    %
    % Para el PVI lineal aplicamos la función spline3p:
    c2=spline3p(a,b,n,x,F2,F1,@(x) 0,1,0);
    %
    % Calculamos los valores de los splines en el punto final b.
    % yk almacena el valor del k-ésimo spline en b
    h=x(n+1)-x(n);
    y1=c1(n,4)+h*(c1(n,3)+h*(c1(n,2)+h*c1(n,1)));
    y2=c2(n,4)+h*(c2(n,3)+h*(c2(n,2)+h*c2(n,1)));
    %
    % Iteración del método de Newton para actualizar el ángulo de
    % disparo:
    t=t-(y1-z2)/y2;
end
%
```

```
% Obtenemos la matriz de dimensión n*4 que contiene los 4n
% coeficientes que determinan de manera única el spline cúbico en
% la base considerada:
c=c1;

end
```

#### D. Método del Disparo (No Lineal) para P.V.F. no lineales, con condiciones de contorno especiales, utilizando Pseudo-Splines Cuárticos

Del mismo modo ya explicado en el apartado anterior, introducimos las dos variantes del método *splin4pcnl* para abordar aquellos casos en que las condiciones de contorno vienen definidas en uno de los dos extremos por su derivada en vez de por el valor de la función. La diferencia con lo anterior es que, en este caso, el Método del Disparo No lineal se implementa recurriendo a Pseudo-Splines Cuárticos en lugar de a Splines Cúbicos.

Si las condiciones son del tipo:  $y(a) = z_1$ ,  $y'(b) = z_2$ , con la condición en el punto final dada por la derivada de la función en dicho punto, vamos a utilizar el método *spline4pcnl2*:

```
function [c] =
    spline4pcnl2(spline4p, spline4pnl, a, b, n, x, f, f1, f2, z1, z2, nmax)
%
% La función spline4pcnl2 resuelve EDOs de 2ª orden no lineales
% con condiciones de contorno sencillas mediante la técnica del
% disparo no lineal basada en aproximaciones obtenidas mediante
% pseudo splines cuárticos.
%
% spline4pcnl2 es una función que calcula los coeficientes de un
% pseudo spline cuártico S en el intervalo [a,b], basado en los
% n+1 nodos distintos y ordenados en forma creciente, contenidos
% en el vector: x=(x_1,x_2,...,x_{n+1}), con x_1=a y x_{n+1}=b
%
% En cada intervalo [x_k, x_{k+1}] el spline S viene dado por un
% polinomio S_k de grado <=4 definido por:
% S_k(x)=c(k,5)+c(k,4)*(x-x_k)+c(k,3)*(x-x_k)^2+c(k,2)*(x-x_k)^3
%         +c(k,1)*(x-x_k)^4
% para cada k=1, 2,..., n
%
%
% INPUT:
%
```

```
% spline4p = función que calcula los coeficientes del pseudo
% spline cuártico para problemas de valores iniciales
% lineal
% spline4pnl = función que calcula los coeficientes del pseudo
% spline cuártico para problemas de valores iniciales
% no lineal
% a = extremo inicial del intervalo
% b = extremo final del intervalo
% n = número de subintervalos de la partición del intervalo [a,b]
% x = array con los n+1 nodos distintos, ordenados en forma
% creciente y con x_1=a y x_{n+1}=b
% f = función que define el 2º miembro de la EDO de 2º orden:
% y''=f(x,y,y') cuya solución tratamos de aproximar con el
% spline S
% f1 = función que define la derivada parcial de f respecto de y:
% df/dy. Concretamente: f1(x,y,y')= df/dy(x,y,y')
% f2 = función que define la derivada parcial de f respecto de y':
% df/dy'. Concretamente: f2(x,y,y')= df/dy'(x,y,y')
% z1,z2 = valores de las condiciones de contorno sencillas de la
% EDO de 2º orden:
% En x=a, y(a)=z1
% En x=b, y(b)=z2
% nmax = número máximo de iteraciones con el método de Newton
%
% OUTPUT:
%
% c = matriz de dimensión n*5 que contiene los 5n coeficientes que
% determinan de manera única el pseudo spline cuártico. La
% fila k-ésima de c contiene los coeficientes de S_k en el
% intervalo I_k, en la base:
% [(x-x_k)^4, (x-x_k)^3, (x-x_k)^2, (x-x_k), 1] (orden
% decreciente de potencias), esto es:
% S_k(x)=c(k,1)*(x-x_k)^4+c(k,2)*(x-x_k)^3+c(k,3)*(x-x_k)^2
% +c(k,4)*(x-x_k)+c(k,5).
%
% La función spline4pcnl2 resuelve EDOs de 2ª orden no lineales
% con condiciones de contorno sencillas utilizando el método del
% disparo no lineal implementado a partir del método de Newton.
%
% Las aproximaciones a la solución en cada iteración se obtienen a
% partir de los métodos basados en pseudo splines cuárticos:
% spline4p y spline4pnl que se aplican para resolver los dos
% problemas de valores iniciales (lineal y no lineal
% respectivamente) utilizando los mismos nodos y la misma base.
%
% Queremos que nos muestre bastantes decimales:
format long;
```

```
%
% La fila k-ésima de c almacenará los coeficientes de S_k en orden
% decreciente de potencias
c=zeros(n,5);
%
% La fila k-ésima de cm almacenará los coeficientes de S_k, en
% orden decreciente de potencias, tras cada llamada a la función:
% spline4pnl en c1 y spline4p en c2
% La fila k-ésima de d1 almacenará los coeficientes de S'_k (el de
% c1), en orden decreciente de potencias, tras cada llamada a la
% función.
c1=zeros(n,5);
c2=zeros(n,5);
d1=zeros(n,4);
%
% Comienza la iteración por el método de Newton:
%
% Inicializamos el ángulo de disparo:
t=0;
%
% En cada iteración del método de Newton resolvemos los dos
% problemas de valores iniciales asociados y almacenamos las
% matrices cm que contienen sus coeficientes en la base
% considerada:
%
for j=1:nmax
    %
    % Para el PVI no lineal aplicamos la función spline4pnl:
    c1=spline4pnl(a,b,n,x,f,z1,t);
    %
    % d1 almacena los coeficientes de S'_k calculados a partir de
    % los de S_k (almacenados en c1)
    for i=1:4
        d1(:,i)=(5-i)*c1(:,i);
    end
    %
    % Obtenemos las funciones que proporcionan los coeficientes del
    % PVI lineal:
    pp1 = mkpp(x,c1);
    p1=@(z) ppval(pp1,z);
    pp2 = mkpp(x,d1);
    p2=@(z) ppval(pp2,z);
    F1=@(z) f1(z,p1(z),p2(z));
    F2=@(z) f2(z,p1(z),p2(z));
    %
    % Para el PVI lineal aplicamos la función spline4p:
    c2=spline4p(a,b,n,x,F2,F1,@(x) 0,0,1);
    %
end
```

```
% Calculamos los valores de la derivada primera de los pseudo
% splines en el punto final b.
% yk almacena el valor de la derivada primera del k-ésimo
% pseudo spline en b
h=x(n+1)-x(n);
y1=c1(n,4)+h*(2*c1(n,3)+h*(3*c1(n,2)+h*4*c1(n,1)));
y2=c2(n,4)+h*(2*c2(n,3)+h*(3*c2(n,2)+h*4*c2(n,1)));
%
% Iteración del método de Newton para actualizar el ángulo de
% disparo:
t=t-(y1-z2)/y2;
end
%
% Obtenemos la matriz de dimensión n*5 que contiene los 4n
% coeficientes que determinan de manera única el pseudo spline
% cúartico en la base considerada:
c=c1;
end
```

Mientras que si la condición que aparece expresada a partir de una derivada es la que corresponde al punto inicial:  $y'(a) = z_1$ ,  $y(b) = z_2$ , utilizaremos el método: *spline4pcnl3*:

```
function [c] =
spline4pcnl3(spline4p,spline4pnl,a,b,n,x,f,f1,f2,z1,z2,nmax)
%
% La función spline4pcnl3 resuelve EDOs de 2ª orden no lineales
% con condiciones de contorno sencillas mediante la técnica del
% disparo no lineal basada en aproximaciones obtenidas mediante
% pseudo splines cuárticos.
%
% spline4pcnl3 es una función que calcula los coeficientes de un
% pseudo spline cuártico S en el intervalo [a,b], basado en los
% n+1 nodos distintos y ordenados en forma creciente, contenidos
% en el vector: x=(x_1,x_2,...,x_{n+1}), con x_1=a y x_{n+1}=b
%
% En cada intervalo [x_k, x_{k+1}] el spline S viene dado por un
% polinomio S_k de grado <=4 definido por:
% S_k(x)=c(k,5)+c(k,4)*(x-x_k)+c(k,3)*(x-x_k)^2+c(k,2)*(x-x_k)^3
%         +c(k,1)*(x-x_k)^4
% para cada k=1, 2,..., n
%
```

```
% INPUT:
%
% spline4p = función que calcula los coeficientes del pseudo
%           spline cuártico para problemas de valores iniciales
%           lineal
% spline4pnl = función que calcula los coeficientes del pseudo
%             spline cuártico para problemas de valores iniciales
%             no lineal
% a = extremo inicial del intervalo
% b = extremo final del intervalo
% n = número de subintervalos de la partición del intervalo [a,b]
% x = array con los n+1 nodos distintos, ordenados en forma
%     creciente y con x_1=a y x_{n+1}=b
% f = función que define el 2º miembro de la EDO de 2º orden:
%     y''=f(x,y,y') cuya solución tratamos de aproximar con el
%     spline S
% f1 = función que define la derivada parcial de f respecto de y:
%     df/dy. Concretamente: f1(x,y,y')=df/dy (x,y,y')
% f2 = función que define la derivada parcial de f respecto de y':
%     df/dy'. Concretamente: f2(x,y,y')=df/dy' (x,y,y')
% z1,z2 = valores de las condiciones de contorno sencillas de la
%         EDO de 2º orden:
%         En x=a, y'(a)=z1
%         En x=b, y'(b)=z2
% nmax = número máximo de iteraciones con el método de Newton
%
% OUTPUT:
%
% c = matriz de dimensión n*5 que contiene los 5n coeficientes que
%     determinan de manera única el pseudo spline cuártico. La
%     fila k-ésima de c contiene los coeficientes de S_k en el
%     intervalo I_k, en la base: [(x-x_k)^4, (x-x_k)^3, (x-x_k)^2,
%     (x-x_k), 1] (orden decreciente de potencias), esto es:
%     S_k(x)=c(k,1)*(x-x_k)^4+c(k,2)*(x-x_k)^3+c(k,3)*(x-x_k)^2
%           +c(k,4)*(x-x_k)+c(k,5).
%
% La función spline4pcnl3 resuelve EDOs de 2ª orden no lineales
% con condiciones de contorno sencillas utilizando el método del
% disparo no lineal implementado a partir del método de Newton.
%
% Las aproximaciones a la solución en cada iteración se obtienen a
% partir de los métodos basados en pseudo splines cuárticos:
% spline4p y spline4pnl que se aplican para resolver los dos
% problemas de valores iniciales (lineal y no lineal
% respectivamente) utilizando los mismos nodos y la misma base.
%
% Queremos que nos muestre bastantes decimales:
format long;
% La fila k-ésima de c almacenará los coeficientes de S_k en orden
```

```
% decreciente de potencias
c=zeros(n,5);
%
% La fila k-ésima de cm almacenará los coeficientes de S_k, en
% orden decreciente de potencias, tras cada llamada a la función:
% spline4pnl en c1 y spline4p en c2
% La fila k-ésima de d1 almacenará los coeficientes de S'_k (el de
% c1), en orden decreciente de potencias, tras cada llamada a la
% función.
c1=zeros(n,5);
c2=zeros(n,5);
d1=zeros(n,4);
%
% Comienza la iteración por el método de Newton:
%
% Inicializamos el ángulo de disparo:
t=0;
%
% En cada iteración del método de Newton resolvemos los dos
% problemas de valores iniciales asociados y almacenamos las
% matrices cm que contienen sus coeficientes en la base
% considerada:
%
for j=1:nmax
    %
    % Para el PVI no lineal aplicamos la función spline4pnl:
    c1=spline4pnl(a,b,n,x,f,t,z1);
    %
    % d1 almacena los coeficientes de S'_k calculados a partir de
    % los de S_k (almacenados en c1)
    for i=1:4
        d1(:,i)=(5-i)*c1(:,i);
    end
    %
    % Obtenemos las funciones que proporcionan los coeficientes del
    % PVI lineal:
    pp1 = mkpp(x,c1);
    p1=@(z) ppval(pp1,z);
    pp2 = mkpp(x,d1);
    p2=@(z) ppval(pp2,z);
    F1=@(z) f1(z,p1(z),p2(z));
    F2=@(z) f2(z,p1(z),p2(z));
    %
    % Para el PVI lineal aplicamos la función spline4p:
    c2=spline4p(a,b,n,x,F2,F1,@(x) 0,1,0);
    %
    % Calculamos los valores de los pseudo splines en el punto
    % final b
    % yk almacena el valor del k-ésimo pseudo spline en b
```

```
h=x (n+1) -x (n) ;
y1=c1 (n, 5) +h* (c1 (n, 4) +h* (c1 (n, 3) +h* (c1 (n, 2) +h*c1 (n, 1) ) ) ) ;
y2=c2 (n, 5) +h* (c2 (n, 4) +h* (c2 (n, 3) +h* (c2 (n, 2) +h*c2 (n, 1) ) ) ) ;
%
% Iteración del método de Newton para actualizar el ángulo de
disparo:
t=t- (y1-z2) /y2;

end

%
% Obtenemos la matriz de dimensión n*5 que contiene los 4n
% coeficientes que determinan de manera única el pseudo spline
% cúartico en la base considerada:
c=c1;

end
```



## 7.3. Programas utilizados en los Experimentos Numéricos

Presentamos en este apartado los Scripts de Matlab que corresponden a los experimentos numéricos expuestos en el Capítulo 5, que nos han permitido analizar los nuevos métodos de resolución propuestos en este proyecto.

### 7.3.1. Ejemplo 1

```
% EJEMPLO 1.
% Fijamos los parámetros del problema:
V1=110;
R1=2;
R2=4;
%
% Solución exacta:
sol=@(z) (V1*R1.*(R2-z))./(R2-R1).*z;
%
% Inputs para las llamadas de función:
a=R1;
b=R2;
n=5;
x=[a:0.4:b];
p=@(x) -2./x;
q=@(x) 0;
r=@(x) 0;
z1=[1,0,V1];
z2=[1,0,0];
%
% Llamadas de función:
c31=spline3pc(@spline3p,a,b,n,x,p,q,r,z1,z2);
c32=spline3pc2(@AlgThomas,a,b,n,p,q,r,z1,z2);
c41=spline4pc(@spline4p,a,b,n,x,p,q,r,z1,z2);
c42=spline4pc2(@AlgThomas,a,b,n,p,q,r,z1,z2);
%
% Construcción de los splines:
pp31 = @(z) ppval(mkpp(x,c31),z);
pp32 = @(z) ppval(mkpp(x,c32),z);
pp41 = @(z) ppval(mkpp(x,c41),z);
pp42 = @(z) ppval(mkpp(x,c42),z);
%
% Tabla de valores:
fprintf('Valores de las soluciones numéricas (h= 0.4):\n')
fprintf('Por columnas: x, y31(x), y32(x), y41(x), y42(x), y(x)\n')
fprintf('% 2.1f % 9.6e % 9.6e % 9.6e % 9.6e % 9.6e\n'
        [x;pp31(x);pp32(x);pp41(x);pp42(x);sol(x)])
%
```

```
% Gráfica de las soluciones:
x1 = a:0.005:b;
plot(x1,pp31(x1),'y-',x1,pp32(x1),'b-',x1,pp41(x1),'g-',
      x1,pp42(x1),'k-',x1,sol(x1),'r-')
xlabel('x');
ylabel('y(x)');
title('Soluciones numéricas y exacta con h=0.4');
print -dpdf 'p1.pdf';
%
% Detalle de la gráfica haciendo zoom:
x2 = 2.5999:0.000001:2.6001;
plot(x2,pp31(x2),'y-',x2,pp32(x2),'b-',x2,pp41(x2),'g-',
      x2,pp42(x2),'k-',x2,sol(x2),'r-')
xlabel('x');
ylabel('y(x)');
title('Zoom de las soluciones numéricas (h=0.4) y exacta');
print -dpdf 'p2.pdf';
%
% Gráfica del error splines cúbicos:
plot(x1,pp31(x1)-sol(x1),'y-',x1,pp32(x1)-sol(x1),'b-')
xlabel('x');
ylabel('error');
title('Error para los splines cúbicos con h=0.4');
print -dpdf 'p3.pdf';
%
% Gráfica del error pseudo splines cúarticos:
plot(x1,pp41(x1)-sol(x1),'g-',x1,pp42(x1)-sol(x1),'k-')
xlabel('x');
ylabel('error');
title('Error para los pseudo splines cúarticos con h=0.4');
print -dpdf 'p4.pdf';
%
% Evolución del error al ir disminuyendo el paso:
% Tomamos h=0.4, 0.2, 0.1, 0.05, 0.025, 0.0125, 0.00625 y 0.003125
% Utilizamos un bucle for:
err31=zeros(6,8);
err32=zeros(6,8);
err41=zeros(6,8);
err42=zeros(6,8);

for i=0:7
    n=5*2^i;
    w=[a:0.4/2^i:b];
    c31=spline3pc(@spline3p,a,b,n,w,p,q,r,z1,z2);
    c32=spline3pc2(@AlgThomas,a,b,n,p,q,r,z1,z2);
    c41=spline4pc(@spline4p,a,b,n,w,p,q,r,z1,z2);
    c42=spline4pc2(@AlgThomas,a,b,n,p,q,r,z1,z2);
    %
    % Construcción de las funciones de error para los splines:
```

```
p31 = @(z) ppval(mkpp(w,c31),z)-sol(z);
p32 = @(z) ppval(mkpp(w,c32),z)-sol(z);
p41 = @(z) ppval(mkpp(w,c41),z)-sol(z);
p42 = @(z) ppval(mkpp(w,c42),z)-sol(z);
%
% Almacenamos los errores en los puntos [a:0.2:b]
err31(:,i+1)=p31(x)';
err32(:,i+1)=p32(x)';
err41(:,i+1)=p41(x)';
err42(:,i+1)=p42(x)';

end
%
% Tablas de errores:
fprintf('Errores en las soluciones numéricas (h=0.4/2^i,
        i=0:7):\n')
fprintf(' Por columnas: x, err31(:,1), err31(:,2), err31(:,3),
        err31(:,4), err31(:,5), err31(:,6), err31(:,7),
        err31(:,8)\n')
fprintf('% 2.1f % 5.2e % 5.2e % 5.2e % 5.2e % 5.2e % 5.2e %
        5.2e % 5.2e\n', [x;err31(:,1)';err31(:,2)';
        err31(:,3)';err31(:,4)';err31(:,5)'; zerr31(:,6)';
        err31(:,7)';err31(:,8)'])
%
fprintf(' Por columnas: x, err32(:,1), err32(:,2), err32(:,3),
        err32(:,4), err32(:,5), err32(:,6), err32(:,7),
        err32(:,8)\n')
fprintf('% 2.1f % 5.2e % 5.2e % 5.2e % 5.2e % 5.2e % 5.2e %
        5.2e % 5.2e\n', [x;err32(:,1)';err32(:,2)';err32(:,3)';
        err32(:,4)';err32(:,5)';err32(:,6)';err32(:,7)';err32(:,8)'])
%
fprintf(' Por columnas: x, err41(:,1), err41(:,2), err41(:,3),
        err41(:,4), err41(:,5),err41(:,6), err41(:,7),err41(:,8)\n')
fprintf('% 2.1f % 5.2e % 5.2e % 5.2e % 5.2e % 5.2e % 5.2e %
        5.2e % 5.2e\n', [x;err41(:,1)';err41(:,2)';err41(:,3)';
        err41(:,4)';err41(:,5)';err41(:,6)';err41(:,7)';err41(:,8)'])
%
fprintf(' Por columnas: x, err42(:,1), err42(:,2), err42(:,3),
        err42(:,4),err42(:,5),err42(:,6), err42(:,7), err42(:,8)\n')
fprintf('% 2.1f % 5.2e % 5.2e % 5.2e % 5.2e % 5.2e % 5.2e %
        5.2e % 5.2e\n', [x;err42(:,1)';err42(:,2)';err42(:,3)';
        err42(:,4)';err42(:,5)';err42(:,6)';err42(:,7)';err42(:,8)'])
%
% Gráfica en doble escala logarítmica de los errores en el punto
% medio del intervalo para los distintos pasos:
N=[0:7];
H=0.4./2.^N;
%
% Splines cúbicos:
```

```
plot(log10(H), log10(abs([err31(3,:) ])), 'y-');
hold on;
plot(log10(H), log10(abs([err32(3,:) ])), 'b-');
xlabel('log(h)');
ylabel('log(error)');
title('Error para los splines cúbicos en x=2.8');
axis equal; % Igual medida para las unidades en los dos ejes
hold off;
print -dpdf 'p5.pdf';
% Muestran claramente pendiente 2 (por converger con orden 2)
%
%
% Pseudo splines cuárticos:
plot(log10(H), log10(abs([err41(3,:) ])), 'g-');
hold on;
plot(log10(H), log10(abs([err42(3,:) ])), 'k-');
xlabel('log(h)');
ylabel('log(error)');
title('Error para los pseudo splines cuárticos en x=2.8');
axis equal; % Igual medida para las unidades en los dos ejes
hold off;
print -dpdf 'p6.pdf';
% Muestran claramente pendiente 4 (por converger con orden 4)
```

### 7.3.2. Ejemplo 2

```
% EJEMPLO 2:
% Fijamos los parámetros del problema:
%
% Solución exacta:
%
% Inputs para las llamadas de función:
a=30;
b=60;
n=50;
x=[a:0.6:b];
p=@(x) -3.*cotd(x)-2.*tand(x);
q=@(x) -0.7;
r=@(x) 0;
z1=[1,0,0];
z2=[1,0,5];
%
% Llamadas de función:
c31=spline3pc(@spline3p,a,b,n,x,p,q,r,z1,z2);
```

```
c32=spline3pc2(@AlgThomas,a,b,n,p,q,r,z1,z2);
c41=spline4pc(@spline4p,a,b,n,x,p,q,r,z1,z2);
c42=spline4pc2(@AlgThomas,a,b,n,p,q,r,z1,z2);
%
% Construcción de los splines:
pp31 = @(z) ppval(mkpp(x,c31),z);
pp32 = @(z) ppval(mkpp(x,c32),z);
pp41 = @(z) ppval(mkpp(x,c41),z);
pp42 = @(z) ppval(mkpp(x,c42),z);
%
% Tabla de valores:
x0=[35.01:5:55.01];
fprintf('Valores de las soluciones numéricas (h= 0.6):\n')
fprintf('Por columnas: x, y31(x), y32(x), y41(x), y42(x), y(x)\n')
fprintf('% 4.2f % 9.6e % 9.6e % 9.6e % 9.6e\n',
        [x0;pp31(x0);pp32(x0);pp41(x0);pp42(x0)])
%
% Gráfica de las soluciones con h=0.6:
x1 = a:0.1:b;
plot(x1,pp31(x1),'y-',x1,pp32(x1),'b-',x1,pp41(x1),'g-',
      x1,pp42(x1),'k-')
xlabel('x');
ylabel('y(x)');
title('Soluciones numéricas con h=0.6');
print -dpdf 'p1.pdf';
%
% Detalle de la gráfica haciendo zoom:
x2 = 44.:0.01:46.;
plot(x2,pp31(x2),'y-',x2,pp32(x2),'b-',x2,pp41(x2),'g-',
      x2,pp42(x2),'k-')
xlabel('x');
ylabel('y(x)');
title('Zoom de las soluciones numéricas con h=0.6');
print -dpdf 'p2.pdf';
%
% Gráfica de los splines cúbicos:
plot(x1,pp31(x1),'y-',x1,pp32(x1),'b-')
xlabel('x');
ylabel('y(x)');
title('Gráficas de los splines cúbicos con h=0.6');
print -dpdf 'p3.pdf';
%
% Gráfica de los pseudo splines cúarticos:
plot(x1,pp41(x1),'g-',x1,pp42(x1),'k-')
xlabel('x');
ylabel('y(x)');
title('Gráficas de los pseudo splines cúarticos con h=0.6');
print -dpdf 'p4.pdf';
%
% Evolución de las soluciones al ir disminuyendo el paso:
% Tomamos h= 0.6, 0.3, 0.15, 0.075 y 0.0375
```

```
% Utilizamos un bucle for:
err31=zeros(7,5);
err32=zeros(7,5);
err41=zeros(7,5);
err42=zeros(7,5);

for i=0:4
    n=50*2^i;
    w=[a:0.6/2^i:b];
    c31=spline3pc(@spline3p,a,b,n,w,p,q,r,z1,z2);
    c32=spline3pc2(@AlgThomas,a,b,n,p,q,r,z1,z2);
    c41=spline4pc(@spline4p,a,b,n,w,p,q,r,z1,z2);
    c42=spline4pc2(@AlgThomas,a,b,n,p,q,r,z1,z2);
    %
    % Construcción de las funciones splines:
    p31 = @(z) ppval(mkpp(w,c31),z);
    p32 = @(z) ppval(mkpp(w,c32),z);
    p41 = @(z) ppval(mkpp(w,c41),z);
    p42 = @(z) ppval(mkpp(w,c42),z);
    %
    % Almacenamos los valores en los puntos [a:5:b]
    x2=[a:5:b];
    err31(:,i+1)=p31(x2)';
    err32(:,i+1)=p32(x2)';
    err41(:,i+1)=p41(x2)';
    err42(:,i+1)=p42(x2)';
end
%
% Gráfica de las soluciones h=0.0375:
x1 = a:0.1:b;
plot(x1,p31(x1),'y-',x1,p32(x1),'b-',x1,p41(x1),'g-',
     x1,p42(x1),'k-')
xlabel('x');
ylabel('y(x)');
title('Soluciones numéricas con h=0.0375');
print -dpdf 'p5.pdf';
%
%
% Tablas de valores:
fprintf('Valores de las soluciones numéricas (h=0.6/2^i,
        i=0:4):\n')
fprintf(' Por columnas: x, err31(:,1), err31(:,2), err31(:,3),
        err31(:,4), err31(:,5)\n')
fprintf('% 2.f % 9.6e % 9.6e % 9.6e % 9.6e % 9.6e\n',
        [x2;err31(:,1)';err31(:,2)';err31(:,3)';err31(:,4)';err31(:,5)'])
%
fprintf(' Por columnas: x, err32(:,1), err32(:,2), err32(:,3),
        err32(:,4), err32(:,5)\n')
fprintf('% 2.f % 9.6e % 9.6e % 9.6e % 9.6e % 9.6e\n',
```

```
[x2;err32(:,1)';err32(:,2)';err32(:,3)';err32(:,4)';err32(:,5)']
%
fprintf(' Por columnas: x, err41(:,1), err41(:,2), err41(:,3),
        err41(:,4), err41(:,5)\n')
fprintf('% 2.f % 9.6e % 9.6e % 9.6e % 9.6e % 9.6e\n',
        [x2;err41(:,1)';err41(:,2)';err41(:,3)';err41(:,4)';err41(:,5)'])
%
fprintf(' Por columnas: x, err42(:,1), err42(:,2), err42(:,3),
        err42(:,4), err42(:,5)\n')
fprintf('% 2.f % 9.6e % 9.6e % 9.6e % 9.6e % 9.6e\n',
        [x2;err42(:,1)';err42(:,2)';err42(:,3)';err42(:,4)';err42(:,5)'])
%
```

### 7.3.3. Ejemplo 3

#### a) Ejemplo 3.a

```
% EJEMPLO 3.a
% Fijamos los parámetros del problema:
format long;
ep=10^4;
%
% Solución exacta:
sol=@(z) 1./(2-z.^2)-exp(-sqrt(ep).*(1+z))-exp(-sqrt(ep).*(1-z));
%
% Inputs para las llamadas de función:
a=0;
b=1;
n=5;
x=[0 0.4 0.85 0.96 0.99 1];
p=@(x) 0;
q=@(x) ep.*(2-x.^2);
r=@(x) -ep;
z1=[0,1,0];
z2=[1,0,0];
%
% Llamadas de función:
c31=spline3pc(@spline3p,a,b,n,x,p,q,r,z1,z2);
%c32=spline3pc2(@AlgThomas,a,b,n,p,q,r,z1,z2);
c41=spline4pc(@spline4p,a,b,n,x,p,q,r,z1,z2);
%c42=spline4pc2(@AlgThomas,a,b,n,p,q,r,z1,z2);
%
% Construcción de los splines:
pp31 = @(z) ppval(mkpp(x,c31),z);
%pp32 = @(z) ppval(mkpp(x,c32),z);
```

```
pp41 = @(z) ppval(mkpp(x,c41),z);
%pp42 = @(z) ppval(mkpp(x,c42),z);
%
% Tabla de valores:
x0=[a:0.2:1];
fprintf('Valores de las soluciones numéricas con n=5:\n')
fprintf(' Por columnas: x, y31(x), y41(x), y(x)\n')
fprintf('% 3.1f % 9.6e % 9.6e % 9.6e\n',
        [x0;pp31(x0);pp41(x0);sol(x0)])
%
% Gráfica de las soluciones con 7 puntos:
x1 = a:0.005:b;
plot(x1,pp31(x1),'b-',x1,pp41(x1),'r-',x1,sol(x1),'k-')
xlabel('x');
ylabel('y(x)');
title('Soluciones numéricas con 5 intervalos');
print -dpdf 'p3a1.pdf';
%
% Detalle de la gráfica haciendo zoom:
x2 = 0.98:0.0001:1;
plot(x2,pp31(x2),'b-',x2,pp41(x2),'r-',x2,sol(x2),'k-')
xlabel('x');
ylabel('y(x)');
title('Zoom de las soluciones numéricas con 5 intervalos');
print -dpdf 'p3a2.pdf';
%
% Gráfica del spline cúbico:
plot(x1,pp31(x1),'b-',x1,sol(x1),'k-')
xlabel('x');
ylabel('y(x)');
title('Gráficas del spline cúbico con 5 intervalos');
print -dpdf 'p3a3.pdf';
%
% Gráfica de los pseudo splines cúarticos:
plot(x1,pp41(x1),'r-',x1,sol(x1),'k-')
xlabel('x');
ylabel('y(x)');
title('Gráficas del pseudo spline cúartico con 5 intervalos');
print -dpdf 'p3a4.pdf';
%
% Soluciones tomando 8 y 15 puntos:
%
err31=zeros(6,2);
err41=zeros(6,2);
n=5;
w=[0 0.4 0.85 0.96 0.99 1];
c31=spline3pc(@spline3p,a,b,n,w,p,q,r,z1,z2);
c41=spline4pc(@spline4p,a,b,n,w,p,q,r,z1,z2);
%
```



```
% Construcción de las funciones de error para los splines:
p31 = @(z) ppval(mkpp(w,c31),z)-sol(z);
p41 = @(z) ppval(mkpp(w,c41),z)-sol(z);
%
% Almacenamos los errores en los puntos:
x2=[0 0.4 0.85 0.96 0.99 1];
err31(:,1)=p31(x2)';
err41(:,1)=p41(x2)';
%
n=7;
w=[0 0.3 0.6 0.85 0.95 0.97 0.99 1];
c31=spline3pc(@spline3p,a,b,n,w,p,q,r,z1,z2);
c41=spline4pc(@spline4p,a,b,n,w,p,q,r,z1,z2);
%
% Construcción de las funciones de error para los splines:
p31 = @(z) ppval(mkpp(w,c31),z)-sol(z);
p41 = @(z) ppval(mkpp(w,c41),z)-sol(z);
%
% Almacenamos los valores en los puntos:
err31(:,2)=p31(x2)';
err41(:,2)=p41(x2)';
%
% Gráfica de los errores:
x1 = a:0.005:b;
plot(x1,p31(x1),'b-',x1,p41(x1),'r-')
xlabel('x');
ylabel('y(x)');
title('Gráfica del error para 7 puntos');
print -dpdf 'p3a5.pdf';
%
%
% Tablas de errores:
fprintf('Errores de las soluciones numéricas con n= 5 y 7:\n')
fprintf(' Por columnas: x, err31(:,1), err31(:,2)\n')
fprintf('% 7.5f % 9.6e % 9.6e\n',[x2;err31(:,1)';err31(:,2)'])
%
fprintf(' Por columnas: x, err41(:,1), err41(:,2)\n')
fprintf('% 7.5f % 9.6e % 9.6e\n',[x2;err41(:,1)';err41(:,2)'])
%
```

### b) Ejemplo 3.b

```
% Fijamos los parámetros del problema:
format long;
ep=10^8;
%
% Solución exacta:
sol=@(z) 1./(2-z.^2)-exp(-sqrt(ep).*(1+z))-exp(-sqrt(ep).*(1-z));
%
% Inputs para las llamadas de función:
a=0;
b=1;
n=12;
x=[0 0.4 0.8 0.95 0.98 0.992 0.997 0.999 0.9995 0.9999 0.99993
    0.99996 1];
p=@(x) 0;
q=@(x) ep.*(2-x.^2);
r=@(x) -ep;
z1=[0,1,0];
z2=[1,0,0];
%
% Llamadas de función:
c31=spline3pc(@spline3p,a,b,n,x,p,q,r,z1,z2);
%c32=spline3pc2(@AlgThomas,a,b,n,p,q,r,z1,z2);
c41=spline4pc(@spline4p,a,b,n,x,p,q,r,z1,z2);
%c42=spline4pc2(@AlgThomas,a,b,n,p,q,r,z1,z2);
%
% Construcción de los splines:
pp31 = @(z) ppval(mkpp(x,c31),z);
%pp32 = @(z) ppval(mkpp(x,c32),z);
pp41 = @(z) ppval(mkpp(x,c41),z);
%pp42 = @(z) ppval(mkpp(x,c42),z);
%
% Tabla de valores:
x0=[a:0.2:1];
fprintf('Valores de las soluciones numéricas con n=12:\n')
fprintf(' Por columnas: x, y31(x), y41(x), y(x)\n')
fprintf('% 3.1f % 9.6e % 9.6e % 9.6e\n',
        [x0;pp31(x0);pp41(x0);sol(x0)])
%
% Gráfica de las soluciones con 13 puntos:
x1 = a:0.005:b;
plot(x1,pp31(x1),'b-',x1,pp41(x1),'r-',x1,sol(x1),'k-')
xlabel('x');
ylabel('y(x)');
title('Soluciones numéricas con 12 intervalos');
print -dpdf 'p3b1.pdf';
%
```

```
% Detalle de la gráfica haciendo zoom:
x2 = 0.98:0.0001:1;
plot(x2,pp31(x2),'b-',x2,pp41(x2),'r-',x2,sol(x2),'k-')
xlabel('x');
ylabel('y(x)');
title('Zoom de las soluciones numéricas con 12 intervalos');
print -dpdf 'p3b2.pdf';
%
% Gráfica del spline cúbico:
plot(x1,pp31(x1),'b-',x1,sol(x1),'k-')
xlabel('x');
ylabel('y(x)');
title('Gráficas del spline cúbico con 12 intervalos');
print -dpdf 'p3b3.pdf';
%
% Gráfica de los pseudo splines cúarticos:
plot(x1,pp41(x1),'r-',x1,sol(x1),'k-')
xlabel('x');
ylabel('y(x)');
title('Gráficas del pseudo spline cúartico con 12 intervalos');
print -dpdf 'p3b4.pdf';
%
```

#### 7.3.4. Ejemplo 4

```
% EJEMPLO 4
% Fijamos los parámetros del problema:
c=20;
%
% Solución exacta:
sol=@(z) exp(-c.*z.^2);
%
% Inputs para las llamadas de función:
a=0;
b=1;
n=5;
x=[a:0.2:b];
p=@(x) -2*c.*x;
q=@(x) -2*c;
r=@(x) 0;
z1=[1,0,1];
z2=[1,0,exp(-c)];
%
```

```
% Llamadas de función:
c31=spline3pc(@spline3p,a,b,n,x,p,q,r,z1,z2);
c32=spline3pc2(@AlgThomas,a,b,n,p,q,r,z1,z2);
c41=spline4pc(@spline4p,a,b,n,x,p,q,r,z1,z2);
c42=spline4pc2(@AlgThomas,a,b,n,p,q,r,z1,z2);
%
% Construcción de los splines:
pp31 = @(z) ppval(mkpp(x,c31),z);
pp32 = @(z) ppval(mkpp(x,c32),z);
pp41 = @(z) ppval(mkpp(x,c41),z);
pp42 = @(z) ppval(mkpp(x,c42),z);
%
% Tabla de valores:
fprintf('Valores de las soluciones numéricas (h= 0.2):\n')
fprintf('Por columnas: x, y31(x), y32(x), y41(x), y42(x), y(x)\n')
fprintf('% 2.1f % 9.6e % 9.6e % 9.6e % 9.6e % 9.6e\n',
        [x;pp31(x);pp32(x);pp41(x);pp42(x);sol(x)])
%
% Gráfica de las soluciones:
x1 = a:0.005:b;
plot(x1,pp31(x1),'y',x1,pp32(x1),'b-',x1,pp41(x1),'g-',
     x1,pp42(x1),'k-',x1,sol(x1),'r-')
xlabel('x');
ylabel('y(x)');
title('Soluciones numéricas y exacta con h=0.2');
print -dpdf 'p41.pdf';
%
% Detalle de la gráfica haciendo zoom:
x2 = 0.98:0.0002:1;
plot(x2,pp31(x2),'y',x2,pp32(x2),'b-',x2,pp41(x2),'g-',
     x2,pp42(x2),'k-',x2,sol(x2),'r-')
xlabel('x');
ylabel('y(x)');
title('Zoom de las soluciones numéricas (h=0.2) y exacta');
print -dpdf 'p42.pdf';
%
% Gráfica del error splines cúbicos:
plot(x1,pp31(x1)-sol(x1),'b-',x1,pp32(x1)-sol(x1),'r-')
xlabel('x');
ylabel('error');
title('Error para los splines cúbicos (h=0.2)');
print -dpdf 'p43.pdf';
%
% Gráfica del error pseudo splines cúarticos:
plot(x1,pp41(x1)-sol(x1),'b-',x1,pp42(x1)-sol(x1),'r-')
xlabel('x');
ylabel('error');
title('Error para los pseudo splines cúarticos (h=0.2)');
print -dpdf 'p44.pdf';
%
% Evolución del error al ir disminuyendo el paso:
```

```
% Tomamos h= 0.4, 0.2, 0.1, 0.05, 0.025, 0.0125, 0.00625 y
0.003125
% Utilizamos un bucle for:
err31=zeros(6,8);
err32=zeros(6,8);
err41=zeros(6,8);
err42=zeros(6,8);
for i=0:7
    n=5*2^i;
    w=[a:0.2/2^i:b];
    c31=spline3pc(@spline3p,a,b,n,w,p,q,r,z1,z2);
    c32=spline3pc2(@AlgThomas,a,b,n,p,q,r,z1,z2);
    c41=spline4pc(@spline4p,a,b,n,w,p,q,r,z1,z2);
    c42=spline4pc2(@AlgThomas,a,b,n,p,q,r,z1,z2);
    %
    % Construcción de las funciones de error para los splines:
    p31 = @(z) ppval(mkpp(w,c31),z)-sol(z);
    p32 = @(z) ppval(mkpp(w,c32),z)-sol(z);
    p41 = @(z) ppval(mkpp(w,c41),z)-sol(z);
    p42 = @(z) ppval(mkpp(w,c42),z)-sol(z);
    %
    % Almacenamos los errores en los puntos [a:0.2:b]
    err31(:,i+1)=p31(x)';
    err32(:,i+1)=p32(x)';
    err41(:,i+1)=p41(x)';
    err42(:,i+1)=p42(x)';
end
%
% Tablas de errores:
fprintf('Errores en las soluciones numéricas (h=0.2/2^i,
        i=0:7):\n')
fprintf(' Por columnas: x, err31(:,1), err31(:,2), err31(:,3),
        err31(:,4), err31(:,5), err31(:,6), err31(:,7), err31(:,8)\n')
fprintf('% 2.1f % 5.2e % 5.2e % 5.2e % 5.2e % 5.2e % 5.2e % 5.2e %
        5.2e\n',[x;err31(:,1)';err31(:,2)';err31(:,3)';err31(:,4)';
        err31(:,5)';err31(:,6)';err31(:,7)';err31(:,8)'])
%
fprintf(' Por columnas: x, err32(:,1), err32(:,2), err32(:,3),
        err32(:,4), err32(:,5), err32(:,6), err32(:,7), err32(:,8)\n')
fprintf('% 2.1f % 5.2e % 5.2e % 5.2e % 5.2e % 5.2e % 5.2e % 5.2e %
        5.2e\n',[x;err32(:,1)';err32(:,2)';err32(:,3)';err32(:,4)';
        err32(:,5)';err32(:,6)';err32(:,7)';err32(:,8)'])
%
fprintf(' Por columnas: x, err41(:,1), err41(:,2), err41(:,3),
        err41(:,4), err41(:,5), err41(:,6), err41(:,7), err41(:,8)\n')
fprintf('% 2.1f % 5.2e % 5.2e % 5.2e % 5.2e % 5.2e % 5.2e % 5.2e %
        5.2e\n',[x;err41(:,1)';err41(:,2)';err41(:,3)';err41(:,4)';
        err41(:,5)';err41(:,6)';err41(:,7)';err41(:,8)'])
%
```

```
fprintf(' Por columnas: x, err42(:,1), err42(:,2), err42(:,3),  
      err42(:,4), err42(:,5), err42(:,6), err42(:,7), err42(:,8)\n')  
fprintf('% 2.1f % 5.2e % 5.2e % 5.2e % 5.2e % 5.2e % 5.2e % 5.2e %  
      5.2e\n', [x;err42(:,1)';err42(:,2)';err42(:,3)';err42(:,4)';  
      err42(:,5)';err42(:,6)';err42(:,7)';err42(:,8)'])  
  
%  
% Gráfica en doble escala logarítmica de los errores en el punto  
% x=0.2 para los distintos pasos:  
N=[0:7];  
H=0.2./2.^N;  
%  
% Splines cúbicos:  
plot(log10(H), log10(abs([err31(2,:)])), 'y-');  
hold on;  
plot(log10(H), log10(abs([err32(2,:)])), 'b-');  
xlabel('log(h)');  
ylabel('log(error)');  
title('Error para los splines cúbicos en x=0.2');  
axis equal; % Igual medida para las unidades en los dos ejes  
hold off;  
print -dpdf 'p45.pdf';  
% Muestran claramente pendiente 2 (por converger con orden 2)  
%  
% Pseudo splines cúarticos:  
plot(log10(H), log10(abs([err41(2,:)])), 'g-');  
hold on;  
plot(log10(H), log10(abs([err42(2,:)])), 'k-');  
xlabel('log(h)');  
ylabel('log(error)');  
title('Error para los pseudo splines cúarticos en x=0.2');  
axis equal; % Igual medida para las unidades en los dos ejes  
hold off;  
print -dpdf 'p46.pdf';  
% Muestran claramente pendiente 4 (por converger con orden 4)
```

### 7.3.5. Ejemplo 5

#### a) Ejemplo 5.a

```
% EJEMPLO 5.a  
% Fijamos los parámetros del problema:  
g=0.25;  
F=0.22;  
om=1;  
% Solución exacta:
```

```
%  
% Inputs para las llamadas de función:  
a=0;  
b=100;  
n=250;  
x=[a:0.4:b];  
f=@(x,y,z) -g.*z+y-y.^3+F.*cos(om.*x);  
%f1=@(x,y,z) -om.*F.*sin(om.*x);  
%f2=@(x,y,z) 1-3.*y.^2;  
%f3=@(x,y,z) -g;  
z1=1;  
z2=0;  
%  
% Llamadas de función:  
c31=spline3pnl(a,b,n,x,f,z1,z2);  
c41=spline4pnl(a,b,n,x,f,z1,z2);  
%  
% Construcción de los splines:  
pp31 = @(z) ppval(mkpp(x,c31),z);  
pp41 = @(z) ppval(mkpp(x,c41),z);  
%  
% Tabla de valores:  
fprintf('Valores de las soluciones numéricas en x=[99, 99.8]  
      (h= 0.4):\n');  
x0=[99:0.2:b-0.2];  
fprintf(' Por columnas: x, y31(x), y41(x)\n');  
fprintf('% 4.1f % 9.6e % 9.6e\n',[x0;pp31(x0);pp41(x0)]);  
%  
% Gráfica de las soluciones con h=0.4:  
x1 = a:0.2:b;  
plot(x1,pp31(x1),'b-',x1,pp41(x1),'r-')  
xlabel('x');  
ylabel('y(x)');  
title('Soluciones numéricas con h=0.4');  
print -dpdf 'p5a1.pdf';  
%  
% Detalle de la gráfica haciendo zoom:  
x2 = 99:0.005:b;  
plot(x2,pp31(x2),'b-',x2,pp41(x2),'r-')  
xlabel('x');  
ylabel('y(x)');  
title('Zoom de las soluciones numéricas con h=0.4');  
print -dpdf 'p5a2.pdf';  
%  
% Gráfica del spline cúbico:  
plot(x1,pp31(x1),'b-')  
xlabel('x');  
ylabel('y(x)');  
title('Gráficas del spline cúbico con h=0.4');  
print -dpdf 'p5a3.pdf';  
%
```

```
% Gráfica del pseudo spline cúartico:
plot(x1,pp41(x1),'r-')
xlabel('x');
ylabel('y(x)');
title('Gráficas del pseudo spline cúartico con h=0.4');
print -dpdf 'p5a4.pdf';
%
% Evolución de las soluciones al ir disminuyendo el paso:
% Tomamos h= 0.4/2^i para i=0:4, x=[0,100]
% Utilizamos un bucle for:
err31=zeros(5,5);
err41=zeros(5,5);
for i=0:4
    n=250*2^i;
    w=[a:0.4/2^i:b];
    c31=spline3pnl(a,b,n,w,f,z1,z2);
    c41=spline4pnl(a,b,n,w,f,z1,z2);
    %
    % Construcción de las funciones splines:
    p31 = @(z) ppval(mkpp(w,c31),z);
    p41 = @(z) ppval(mkpp(w,c41),z);
    %
    % Valores de las soluciones numéricas en los puntos
    % [99:0.2:99.8]:
    x2=[99:0.2:b-0.2];
    err31(:,i+1)=p31(x2)';
    err41(:,i+1)=p41(x2)';
end
%
% Gráfica de las soluciones h=0.0125:
x1 = a:0.2:b;
plot(x1,p31(x1),'b-',x1,p41(x1),'r-')
xlabel('x');
ylabel('y(x)');
title('Soluciones numéricas con h=0.0125');
print -dpdf 'p5a5.pdf';
%
%
% Tabla de valores de las soluciones numéricas en x=[99, 99.8]
%(h= 0.4/2^i, i=0:4):
fprintf('valores de las soluciones numéricas en x=[99, 99.8]
        (h= 0.4/2^i, i=0:4):\n')
fprintf(' Por columnas: x, err31(:,1), err31(:,2), err31(:,3),
        err31(:,4), err31(:,5)\n')
fprintf('% 4.1f % 9.6e % 9.6e % 9.6e % 9.6e % 9.6e\n',
        [x2;err31(:,1)';err31(:,2)';err31(:,3)';err31(:,4)';err31(:,5)'])
%
fprintf(' Por columnas: x, err41(:,1), err41(:,2), err41(:,3),
```



```
err41(:,4), err41(:,5)\n')  
fprintf('% 4.1f % 9.6e % 9.6e % 9.6e % 9.6e\n',  
[x2;err41(:,1)';err41(:,2)';err41(:,3)';err41(:,4)';err41(:,5)'])  
%
```

## b) Ejemplo 5.b

```
% EJEMPLO 5.b  
% Fijamos los parámetros del problema:  
g=0.25;  
F=0.25;  
om=1;  
%  
% Solución exacta:  
%  
% Inputs para las llamadas de función:  
a=0;  
b=200;  
n=1000;  
x=[a:0.2:b];  
f=@(x,y,z) -g.*z+y-y.^3+F.*cos(om.*x);  
%f1=@(x,y,z) -om.*F.*sin(om.*x);  
%f2=@(x,y,z) 1-3.*y.^2;  
%f3=@(x,y,z) -g;  
z1=0.2;  
z2=0.1;  
%  
% Llamadas de función:  
c31=spline3pnl(a,b,n,x,f,z1,z2);  
c41=spline4pnl(a,b,n,x,f,z1,z2);  
%  
% Construcción de los splines:  
pp31 = @(z) ppval(mkpp(x,c31),z);  
pp41 = @(z) ppval(mkpp(x,c41),z);  
%  
% Tabla de valores:  
fprintf('Valores de las soluciones numéricas en x=[199, 199.8]  
      (h= 0.2):\n')  
x0=[b-1:0.2:b-0.2];  
fprintf(' Por columnas: x, y31(x), y41(x)\n')  
fprintf('% 4.1f % 9.6e % 9.6e\n', [x0;pp31(x0);pp41(x0)])  
%  
% Gráfica de las soluciones con h=0.2:  
x1 = a:0.2:b;  
plot(x1,pp31(x1), 'b-', x1,pp41(x1), 'r-')  
xlabel('x');  
ylabel('y(x)');  
title('Soluciones numéricas con h=0.2');  
print -dpdf 'p5bb1.pdf';  
%
```

```
% Detalle de la gráfica haciendo zoom:
x2 = b-1:0.005:b;
plot(x2,pp31(x2),'b-',x2,pp41(x2),'r-')
xlabel('x');
ylabel('y(x)');
title('Zoom de las soluciones numéricas con h=0.2');
print -dpdf 'p5bb2.pdf';
%
% Gráfica del spline cúbico:
plot(x1,pp31(x1),'b-')
xlabel('x');
ylabel('y(x)');
title('Gráficas del spline cúbico con h=0.2');
print -dpdf 'p5bb3.pdf';
%
% Gráfica del pseudo spline cúartico:
plot(x1,pp41(x1),'r-')
xlabel('x');
ylabel('y(x)');
title('Gráficas del pseudo spline cúartico con h=0.2');
print -dpdf 'p5bb4.pdf';
%
% Evolución de las soluciones al ir disminuyendo el paso:
% Tomamos h= 0.2/2^i para i=0:4, x=[0,200]
% Utilizamos un bucle for:
err31=zeros(5,5);
err41=zeros(5,5);
for i=0:4
    n=1000*2^i;
    w=[a:0.2/2^i:b];
    c31=spline3pnl(a,b,n,w,f,z1,z2);
    c41=spline4pnl(a,b,n,w,f,z1,z2);
    %
    % Construcción de las funciones splines:
    p31 = @(z) ppval(mkpp(w,c31),z);
    p41 = @(z) ppval(mkpp(w,c41),z);
    %
    % Valores de las soluciones numéricas en los puntos
    % [199:0.2:199.8]:
    x2=[b-1:0.2:b-0.2];
    err31(:,i+1)=p31(x2)';
    err41(:,i+1)=p41(x2)';
end
%
% Gráfica de las soluciones h=0.00625:
x1 = a:0.2:b;
plot(x1,p31(x1),'b-',x1,p41(x1),'r-')
xlabel('x');
ylabel('y(x)');
title('Soluciones numéricas con h=0.00625');
print -dpdf 'p5bb5.pdf';
```

```
%  
%  
% Tabla de valores de las soluciones numéricas en x=[b-1, b-0.2]  
(h= 0.2/2^i, i=0:4):  
fprintf('Valores de las soluciones numéricas en x=[199, 199.8] (h=  
0.2/2^i, i=0:4):\n')  
fprintf(' Por columnas: x, err31(:,1), err31(:,2), err31(:,3),  
err31(:,4), err31(:,5)\n')  
fprintf('% 4.1f % 9.6e % 9.6e % 9.6e % 9.6e % 9.6e\n',  
[x2;err31(:,1)';err31(:,2)';err31(:,3)';err31(:,4)';err31(:,5)'])  
%  
fprintf(' Por columnas: x, err41(:,1), err41(:,2), err41(:,3),  
err41(:,4), err41(:,5)\n')  
fprintf('% 4.1f % 9.6e % 9.6e % 9.6e % 9.6e % 9.6e\n',  
[x2;err41(:,1)';err41(:,2)';err41(:,3)';err41(:,4)';err41(:,5)'])  
%
```

### c) Ejemplo 5.c

```
% EJEMPLO 5.c  
% Fijamos los parámetros del problema:  
g=0.25;  
F=0.4;  
om=1;  
%  
% Solución exacta:  
%  
% Inputs para las llamadas de función:  
a=0;  
b=100;  
n=1000;  
x=[a:0.1:b];  
f=@(x,y,z) -g.*z+y-y.^3+F.*cos(om.*x);  
%f1=@(x,y,z) -om.*F.*sin(om.*x);  
%f2=@(x,y,z) 1-3.*y.^2;  
%f3=@(x,y,z) -g;  
z1=0.2;  
z2=0.1;  
%  
% Llamadas de función:  
c31=spline3pnl(a,b,n,x,f,z1,z2);  
c41=spline4pnl(a,b,n,x,f,z1,z2);  
%  
% Construcción de los splines:  
pp31 = @(z) ppval(mkpp(x,c31),z);  
pp41 = @(z) ppval(mkpp(x,c41),z);  
%  
% Tabla de valores:  
fprintf('Valores de las soluciones numéricas en x=[99, 99.8]
```

```
(h= 0.1):\n')
x0=[99:0.2:b-0.2];
fprintf(' Por columnas: x, y31(x), y41(x)\n')
fprintf('% 4.1f % 9.6e % 9.6e\n', [x0;pp31(x0);pp41(x0)])
%
% Gráfica de las soluciones con h=0.1:
x1 = a:0.2:b;
plot(x1,pp31(x1), 'b-', x1,pp41(x1), 'r-')
xlabel('x');
ylabel('y(x)');
title('Soluciones numéricas con h=0.1');
print -dpdf 'p5c1.pdf';
%
% Detalle de la gráfica haciendo zoom:
x2 = 99:0.005:b;
plot(x2,pp31(x2), 'b-', x2,pp41(x2), 'r-')
xlabel('x');
ylabel('y(x)');
title('Zoom de las soluciones numéricas con h=0.1');
print -dpdf 'p5c2.pdf';
%
% Gráfica del spline cúbico:
plot(x1,pp31(x1), 'b-')
xlabel('x');
ylabel('y(x)');
title('Gráficas del spline cúbico con h=0.1');
print -dpdf 'p5c3.pdf';
%
% Gráfica del pseudo spline cúartico:
plot(x1,pp41(x1), 'r-')
xlabel('x');
ylabel('y(x)');
title('Gráficas del pseudo spline cúartico con h=0.1');
print -dpdf 'p5c4.pdf';
%
% Evolución de las soluciones al ir disminuyendo el paso:
% Tomamos h= 0.1/2^i para i=0:4, x=[0,100]
% Utilizamos un bucle for:
err31=zeros(5,5);
err41=zeros(5,5);
for i=0:4
    n=1000*2^i;
    w=[a:0.1/2^i:b];
    c31=spline3pnl(a,b,n,w,f,z1,z2);
    c41=spline4pnl(a,b,n,w,f,z1,z2);
    %
```

```
% Construcción de las funciones splines:
p31 = @(z) ppval(mkpp(w,c31),z);
p41 = @(z) ppval(mkpp(w,c41),z);
%
% Valores de las soluciones numéricas en los puntos
% [99:0.2:99.8]:
x2=[99:0.2:b-0.2];
err31(:,i+1)=p31(x2)';
err41(:,i+1)=p41(x2)';
end
%
% Gráfica de las soluciones h=0.003125:
x1 = a:0.2:b;
plot(x1,p31(x1), 'b-',x1,p41(x1), 'r-')
xlabel('x');
ylabel('y(x)');
title('Soluciones numéricas con h=0.003125');
print -dpdf 'p5c5.pdf';
%
%
% Tabla de valores de las soluciones numéricas en x=[99, 99.8]
% (h= 0.1/2^i, i=0:4):
fprintf('valores de las soluciones numéricas en x=[99, 99.8]
(h= 0.1/2^i, i=0:4):\n')
fprintf(' Por columnas: x, err31(:,1), err31(:,2), err31(:,3),
err31(:,4), err31(:,5)\n')
fprintf('% 4.1f % 9.6e % 9.6e % 9.6e % 9.6e % 9.6e\n',
[x2;err31(:,1)';err31(:,2)';err31(:,3)';err31(:,4)';err31(:,5)'])
%
fprintf(' Por columnas: x, err41(:,1), err41(:,2), err41(:,3),
err41(:,4), err41(:,5)\n')
fprintf('% 4.1f % 9.6e % 9.6e % 9.6e % 9.6e % 9.6e\n',
[x2;err41(:,1)';err41(:,2)';err41(:,3)';err41(:,4)';err41(:,5)'])
%
```

### 7.3.6. Ejemplo 6

#### a) Ejemplo 6.a

```
% EJEMPLO 6.a
% Fijamos los parámetros del problema:
mu=0.2;
%
% Solución exacta:
%
% Inputs para las llamadas de función:
a=0;
b=100;
```

```
n=250;
x=[a:0.4:b];
f=@(x,y,z) mu.*(1-y.^2).*z-y;
%f1=@(x,y,z) 0;
%f2=@(x,y,z) -2.*mu.*y.*z-1;
%f3=@(x,y,z) mu.*(1-y.^2);
z1=0.1;
z2=0;
%
% Llamadas de función:
c31=spline3pnl(a,b,n,x,f,z1,z2);
c41=spline4pnl(a,b,n,x,f,z1,z2);
%
% Construcción de los splines:
pp31 = @(z) ppval(mkpp(x,c31),z);
pp41 = @(z) ppval(mkpp(x,c41),z);
%
% Tabla de valores:
fprintf('Valores de las soluciones numéricas en x=[99, 99.8]
      (h= 0.4):\n')
x0=[b-1:0.2:b-0.2];
fprintf(' Por columnas: x, y31(x), y41(x)\n')
fprintf('% 4.1f % 9.6e % 9.6e\n',[x0;pp31(x0);pp41(x0)])
%
% Gráfica de las soluciones con h=0.4:
x1 = a:0.2:b;
plot(x1,pp31(x1),'b-',x1,pp41(x1),'r-')
xlabel('x');
ylabel('y(x)');
title('Soluciones numéricas con h=0.4');
print -dpdf 'p6a1.pdf';
%
% Detalle de la gráfica haciendo zoom:
x2 = b-1:0.005:b;
plot(x2,pp31(x2),'b-',x2,pp41(x2),'r-')
xlabel('x');
ylabel('y(x)');
title('Zoom de las soluciones numéricas con h=0.4');
print -dpdf 'p6a2.pdf';
%
% Gráfica del spline cúbico:
plot(x1,pp31(x1),'b-')
xlabel('x');
ylabel('y(x)');
title('Gráficas del spline cúbico con h=0.4');
print -dpdf 'p6a3.pdf';
%
% Gráfica del pseudo spline cúartico:
plot(x1,pp41(x1),'r-')
xlabel('x');
ylabel('y(x)');
```

```
title('Gráficas del pseudo spline cúartico con h=0.4');
print -dpdf 'p6a4.pdf';
%
% Evolución de las soluciones al ir disminuyendo el paso:
% Tomamos h= 0.4/2^i para i=0:4, x=[0,100]
% Utilizamos un bucle for:
err31=zeros(5,5);
err41=zeros(5,5);
for i=0:4
    n=250*2^i;
    w=[a:0.4/2^i:b];
    c31=spline3pnl(a,b,n,w,f,z1,z2);
    c41=spline4pnl(a,b,n,w,f,z1,z2);
    %
    % Construcción de las funciones splines:
    p31 = @(z) ppval(mkpp(w,c31),z);
    p41 = @(z) ppval(mkpp(w,c41),z);
    %
    % Valores de las soluciones numéricas en los puntos
    [99:0.2:99.8]:
    x2=[b-1:0.2:b-0.2];
    err31(:,i+1)=p31(x2)';
    err41(:,i+1)=p41(x2)';
end
%
% Gráfica de las soluciones h=0.0125:
x1 = a:0.2:b;
plot(x1,p31(x1), 'b-',x1,p41(x1), 'r-')
xlabel('x');
ylabel('y(x)');
title('Soluciones numéricas con h=0.0125');
print -dpdf 'p6a5.pdf';
%
%
% Tabla de valores de las soluciones numéricas en x=[99, 99.8] (h=
0.4/2^i, i=0:4):
fprintf('Valores de las soluciones numéricas en x=[99, 99.8]
(h= 0.4/2^i, i=0:4):\n')
fprintf(' Por columnas: x, err31(:,1), err31(:,2), err31(:,3),
err31(:,4), err31(:,5)\n')
fprintf('% 4.1f % 9.6e % 9.6e % 9.6e % 9.6e % 9.6e\n',
[x2;err31(:,1)';err31(:,2)';err31(:,3)';err31(:,4)';err31(:,5)'])
%
fprintf(' Por columnas: x, err41(:,1), err41(:,2), err41(:,3),
err41(:,4), err41(:,5)\n')
fprintf('% 4.1f % 9.6e % 9.6e % 9.6e % 9.6e % 9.6e\n',
[x2;err41(:,1)';err41(:,2)';err41(:,3)';err41(:,4)';err41(:,5)'])
%
```

## b) Ejemplo 6.b

```
% EJEMPLO 6.b
% Fijamos los parámetros del problema:
mu=1;
%
% Solución exacta:
%
% Inputs para las llamadas de función:
a=0;
b=100;
n=250;
x=[a:0.4:b];
f=@(x,y,z) mu.*(1-y.^2).*z-y;
%f1=@(x,y,z) 0;
%f2=@(x,y,z) -2.*mu.*y.*z-1;
%f3=@(x,y,z) mu.*(1-y.^2);
z1=0.1;
z2=0;
%
% Llamadas de función:
c31=spline3pnl(a,b,n,x,f,z1,z2);
c41=spline4pnl(a,b,n,x,f,z1,z2);
%
% Construcción de los splines:
pp31 = @(z) ppval(mkpp(x,c31),z);
pp41 = @(z) ppval(mkpp(x,c41),z);
%
% Tabla de valores:
fprintf('Valores de las soluciones numéricas en x=[99, 99.8]
      (h= 0.4):\n')
x0=[b-1:0.2:b-0.2];
fprintf(' Por columnas: x, y31(x), y41(x)\n')
fprintf('% 4.1f % 9.6e % 9.6e\n', [x0;pp31(x0);pp41(x0)])
%
% Gráfica de las soluciones con h=0.4:
x1 = a:0.2:b;
plot(x1,pp31(x1), 'b-',x1,pp41(x1), 'r-')
xlabel('x');
ylabel('y(x)');
title('Soluciones numéricas con h=0.4');
print -dpdf 'p6b1.pdf';
%
% Detalle de la gráfica haciendo zoom:
x2 = b-1:0.005:b;
plot(x2,pp31(x2), 'b-',x2,pp41(x2), 'r-')
xlabel('x');
ylabel('y(x)');
title('Zoom de las soluciones numéricas con h=0.4');
print -dpdf 'p6b2.pdf';
%
```



```
% Gráfica del spline cúbico:
plot(x1,pp31(x1),'b-')
xlabel('x');
ylabel('y(x)');
title('Gráficas del spline cúbico con h=0.4');
print -dpdf 'p6b3.pdf';
%
% Gráfica del pseudo spline cúartico:
plot(x1,pp41(x1),'r-')
xlabel('x');
ylabel('y(x)');
title('Gráficas del pseudo spline cúartico con h=0.4');
print -dpdf 'p6b4.pdf';
%
% Evolución de las soluciones al ir disminuyendo el paso:
% Tomamos h= 0.4/2^i para i=0:4, x=[0,100]
% Utilizamos un bucle for:
err31=zeros(5,5);
err41=zeros(5,5);
for i=0:4
    n=250*2^i;
    w=[a:0.4/2^i:b];
    c31=spline3pnl(a,b,n,w,f,z1,z2);
    c41=spline4pnl(a,b,n,w,f,z1,z2);
    %
    % Construcción de las funciones splines:
    p31 = @(z) ppval(mkpp(w,c31),z);
    p41 = @(z) ppval(mkpp(w,c41),z);
    %
    % Valores de las soluciones numéricas en los puntos
    [99:0.2:99.8]:
    x2=[b-1:0.2:b-0.2];
    err31(:,i+1)=p31(x2)';
    err41(:,i+1)=p41(x2)';
end
%
% Gráfica de las soluciones h=0.0125:
x1 = a:0.2:b;
plot(x1,p31(x1),'b-',x1,p41(x1),'r-')
xlabel('x');
ylabel('y(x)');
title('Soluciones numéricas con h=0.0125');
print -dpdf 'p6b5.pdf';
%
%
% Tabla de valores de las soluciones numéricas en x=[99, 99.8] (h=
0.4/2^i, i=0:4):
fprintf('Valores de las soluciones numéricas en x=[99, 99.8]
(h= 0.4/2^i, i=0:4):\n')
fprintf(' Por columnas: x, err31(:,1), err31(:,2), err31(:,3),
err31(:,4), err31(:,5)\n')
```

```
fprintf('% 4.1f % 9.6e % 9.6e % 9.6e % 9.6e % 9.6e\n',  
[x2;err31(:,1)';err31(:,2)';err31(:,3)';err31(:,4)';err31(:,5)'])  
%  
fprintf(' Por columnas: x, err41(:,1), err41(:,2), err41(:,3),  
err41(:,4), err41(:,5)\n')  
fprintf('% 4.1f % 9.6e % 9.6e % 9.6e % 9.6e %  
9.6e\n', [x2;err41(:,1)';err41(:,2)';err41(:,3)';err41(:,4)';err41(  
:,5)'])  
%
```

### c) Ejercicio 6.c

```
% EJERCICIO 6.c  
% Fijamos los parámetros del problema:  
mu=5;  
%  
% Solución exacta:  
%  
% Inputs para las llamadas de función:  
a=0;  
b=100;  
n=250;  
x=[a:0.4:b];  
f=@(x,y,z) mu.*(1-y.^2).*z-y;  
%f1=@(x,y,z) 0;  
%f2=@(x,y,z) -2.*mu.*y.*z-1;  
%f3=@(x,y,z) mu.*(1-y.^2);  
z1=0.1;  
z2=0;  
%  
% Llamadas de función:  
c31=spline3pnl(a,b,n,x,f,z1,z2);  
c41=spline4pnl(a,b,n,x,f,z1,z2);  
%  
% Construcción de los splines:  
pp31 = @(z) ppval(mkpp(x,c31),z);  
pp41 = @(z) ppval(mkpp(x,c41),z);  
%  
% Tabla de valores:  
fprintf('Valores de las soluciones numéricas en x=[99, 99.8]  
(h= 0.4):\n')  
x0=[b-1:0.2:b-0.2];  
fprintf(' Por columnas: x, y31(x), y41(x)\n')  
fprintf('% 4.1f % 9.6e % 9.6e\n', [x0;pp31(x0);pp41(x0)])  
%  
% Gráfica de las soluciones con h=0.4:  
x1 = a:0.2:b;  
plot(x1,pp31(x1), 'b-', x1,pp41(x1), 'r-')
```

```
xlabel('x');
ylabel('y(x)');
title('Soluciones numéricas con h=0.4');
print -dpdf 'p6c1.pdf';
%
% Detalle de la gráfica haciendo zoom:
x2 = b-1:0.005:b;
plot(x2,pp31(x2), 'b-', x2, pp41(x2), 'r-')
xlabel('x');
ylabel('y(x)');
title('Zoom de las soluciones numéricas con h=0.4');
print -dpdf 'p6c2.pdf';
%
% Gráfica del spline cúbico:
plot(x1,pp31(x1), 'b-')
xlabel('x');
ylabel('y(x)');
title('Gráficas del spline cúbico con h=0.4');
print -dpdf 'p6c3.pdf';
%
% Gráfica del pseudo spline cúartico:
plot(x1,pp41(x1), 'r-')
xlabel('x');
ylabel('y(x)');
title('Gráficas del pseudo spline cúartico con h=0.4');
print -dpdf 'p6c4.pdf';
%
% Evolución de las soluciones al ir disminuyendo el paso:
% Tomamos h= 0.4/2^i para i=0:4, x=[0,100]
% Utilizamos un bucle for:
err31=zeros(5,5);
err41=zeros(5,5);
for i=0:4
n=250*2^i;
w=[a:0.4/2^i:b];
c31=spline3pnl(a,b,n,w,f,z1,z2);
c41=spline4pnl(a,b,n,w,f,z1,z2);
%
% Construcción de las funciones splines:
p31 = @(z) ppval(mkpp(w,c31),z);
p41 = @(z) ppval(mkpp(w,c41),z);
%
% Valores de las soluciones numéricas en los puntos [99:0.2:99.8]:
x2=[b-1:0.2:b-0.2];
err31(:,i+1)=p31(x2)';
err41(:,i+1)=p41(x2)';
end
%
% Gráfica de las soluciones h=0.0125:
x1 = a:0.2:b;
plot(x1,p31(x1), 'b-', x1,p41(x1), 'r-')
```

```
xlabel('x');
ylabel('y(x)');
title('Soluciones numéricas con h=0.0125');
print -dpdf 'p6c5.pdf';
%
%
% Tabla de valores de las soluciones numéricas en x=[99, 99.8] (h=
0.4/2^i, i=0:4):
fprintf('Valores de las soluciones numéricas en x=[99, 99.8]
(h= 0.4/2^i, i=0:4):\n')
fprintf(' Por columnas: x, err31(:,1), err31(:,2), err31(:,3),
err31(:,4), err31(:,5)\n')
fprintf('% 4.1f % 9.6e % 9.6e % 9.6e % 9.6e % 9.6e\n',
[x2;err31(:,1)';err31(:,2)';err31(:,3)';err31(:,4)';err31(:,5)'])
%
fprintf(' Por columnas: x, err41(:,1), err41(:,2), err41(:,3),
err41(:,4), err41(:,5)\n')
fprintf('% 4.1f % 9.6e % 9.6e % 9.6e % 9.6e % 9.6e\n',
[x2;err41(:,1)';err41(:,2)';err41(:,3)';err41(:,4)';err41(:,5)'])
%
```

### 7.3.7. Ejemplo 7

```
% Fijamos los parámetros del problema:
mu=0.758582299525377;
%mu=5.469351386061054;
%
% Solución exacta:
sol=@(z) -2.*log((cosh(mu.*(z-0.5)))/(cosh(mu./2)));
%
% Inputs para las llamadas de función:
a=0;
b=1;
n=5;
x=[a:0.2:b];
f=@(x,y,z) -exp(y);
f1=@(x,y,z) -exp(y);
f2=@(x,y,z) 0;
z1=0;
z2=0;
nmax=6;
%
% Llamadas de función:
c31=spline3pcnl(@spline3p,@spline3pnl,a,b,n,x,f,f1,f2,z1,z2,nmax);
c41=spline4pcnl(@spline4p,@spline4pnl,a,b,n,x,f,f1,f2,z1,z2,nmax);
%
% Construcción de los splines:
```

```
pp31 = @(z) ppval(mkpp(x,c31),z);
pp41 = @(z) ppval(mkpp(x,c41),z);
%
% Tabla de valores:
fprintf('Valores de las soluciones numéricas y exacta en
      x=[0.5, 1] (h= 0.2):\n')
x0=[b-0.5:0.1:b];
fprintf(' Por columnas: x, y31(x), y41(x),sol(x)\n')
fprintf('% 4.1f % 9.6e % 9.6e % 9.6e\n',
      [x0;pp31(x0);pp41(x0);sol(x0)])
%
% Gráfica de las soluciones numéricas y exacta con h=0.2:
x1 = a:0.005:b;
plot(x1,pp31(x1),'b-',x1,pp41(x1),'r-',x1,sol(x1),'k-')
xlabel('x');
ylabel('y(x)');
title('Soluciones numéricas con h=0.2');
print -dpdf 'p7a1.pdf';
%
% Detalle de la gráfica haciendo zoom:
x2 = 0.499:0.000005:0.501;
plot(x2,pp31(x2),'b-',x2,pp41(x2),'r-',x2,sol(x2),'k-')
xlabel('x');
ylabel('y(x)');
title('Zoom de las soluciones numéricas con h=0.2');
print -dpdf 'p7a2.pdf';
%
% Gráfica del spline cúbico:
plot(x1,pp31(x1),'b-',x1,sol(x1),'k-')
xlabel('x');
ylabel('y(x)');
title('Gráficas del spline cúbico con h=0.2');
print -dpdf 'p7a3.pdf';
%
% Gráfica del pseudo spline cúartico:
plot(x1,pp41(x1),'r-',x1,sol(x1),'k-')
xlabel('x');
ylabel('y(x)');
title('Gráficas del pseudo spline cúartico con h=0.2');
print -dpdf 'p7a4.pdf';
%
% Evolución de las soluciones al ir disminuyendo el paso:
% Tomamos h= 0.2/2^i para i=0:6, x=[0,1]
% Utilizamos un bucle for:
err31=zeros(6,7);
err41=zeros(6,7);
for i=0:6
n=5*2^i;
w=[a:0.2/2^i:b];
```

```
c31=spline3pcnl(@spline3p,@spline3pnl,a,b,n,w,f,f1,f2,z1,z2,nmax);
c41=spline4pcnl(@spline4p,@spline4pnl,a,b,n,w,f,f1,f2,z1,z2,nmax);
%
% Construcción de las funciones de error para los splines:
p31 = @(z) ppval(mkpp(w,c31),z)-sol(z);
p41 = @(z) ppval(mkpp(w,c41),z)-sol(z);
%
% Errores de las soluciones numéricas en los puntos [0.5:0.1:1]:
x2=[b-0.5:0.1:b];
err31(:,i+1)=p31(x2)';
err41(:,i+1)=p41(x2)';
end
%
% Gráfica de los errores para las soluciones h=0.003125:
x1 = a:0.005:b;
plot(x1,p31(x1),'b-',x1,p41(x1),'r-')
xlabel('x');
ylabel('y(x)');
title('Errores para las soluciones numéricas con h=0.003125');
print -dpdf 'p7a5.pdf';
%
%
% Tabla de errores en x=[0.5, 1] (h= 0.2/2^i, i=0:6):
fprintf('Errores de las soluciones numéricas en x=[0.5, 1]
(h= 0.2/2^i, i=0:6):\n')
fprintf(' Por columnas: x, err31(:,1), err31(:,2), err31(:,3),
err31(:,4), err31(:,5), err31(:,6), err31(:,7)\n')
fprintf('% 4.1f % 5.2e % 5.2e % 5.2e % 5.2e % 5.2e % 5.2e %
5.2e\n',[x2;err31(:,1)';err31(:,2)';err31(:,3)';err31(:,4)';
err31(:,5)';err31(:,6)';err31(:,7)'])
%
fprintf(' Por columnas: x, err41(:,1), err41(:,2), err41(:,3),
err41(:,4), err41(:,5), err41(:,6), err41(:,7)\n')
fprintf('% 4.1f % 5.2e % 5.2e % 5.2e % 5.2e % 5.2e % 5.2e %
5.2e\n',[x2;err41(:,1)';err41(:,2)';err41(:,3)';err41(:,4)';
err41(:,5)';err41(:,6)';err41(:,7)'])
%
% Gráfica en doble escala logarítmica de los errores en el punto
% medio del intervalo para los distintos pasos:
N=[0:6];
H=0.2./2.^N;
%
% Spline cúbico:
plot(log10(H), log10(abs([err31(1,:)])), 'b-');
xlabel('log(h)');
ylabel('log(error)');
title('Error para el spline cúbico en x=0.5');
axis equal; % Igual medida para las unidades en los dos ejes
print -dpdf 'p7a6.pdf';
% Muestran claramente pendiente 2 (por converger con orden 2)
%
```

```
% Pseudo spline cúartico:
plot(log10(H), log10(abs([err41(1,:) ])), 'r-');
xlabel('log(h)');
ylabel('log(error)');
title('Error para el pseudo spline cuártico en x=0.5');
axis equal; % Igual medida para las unidades en los dos ejes
print -dpdf 'p7a7.pdf';
% Muestran claramente pendiente 4 (por converger con orden 4)
```

### 7.3.8. Ejemplo 8

```
% EJEMPLO 8
% Fijamos los parámetros del problema:
ap=6;
ro=0.5;
ti=0.05;
be=0.1;
%
% Solución exacta:
%
% Inputs para las llamadas de función:
a=0;
b=1;
n=8;
x=[a:0.125:b];
f=@(x,y,z) (-1./(x+ro)+tand(ap)./((1-x).*tand(ap)+ti)).*z+...
    (be.*y.^4)./((1-x).*tand(ap)+ti);
f1=@(x,y,z) (4*be.*y.^3)./((1-x).*tand(ap)+ti);
f2=@(x,y,z) -1./(x+ro)+tand(ap)./((1-x).*tand(ap)+ti);
z1=1;
z2=0;
nmax=6;
%
% Llamadas de función:
c31=spline3pcnl2(@spline3p,@spline3pn1,a,b,n,x,f,f1,f2,z1,z2,nmax)
;
c41=spline4pcnl2(@spline4p,@spline4pn1,a,b,n,x,f,f1,f2,z1,z2,nmax)
;
%
% Construcción de los splines:
pp31 = @(z) ppval(mkpp(x,c31),z);
pp41 = @(z) ppval(mkpp(x,c41),z);
%
% Tabla de valores:
fprintf('Valores de las soluciones numéricas (h= 0.125):\n')
x0=[a:0.2:b];
fprintf(' Por columnas: x, y31(x), y41(x)\n')
```

```
fprintf('% 3.1f % 9.6e % 9.6e\n', [x0;pp31(x0);pp41(x0)])
%
% Gráfica de las soluciones con h=0.125:
x1 = a:0.01:b;
plot(x1,pp31(x1), 'b-', x1,pp41(x1), 'r-')
xlabel('x');
ylabel('y(x)');
title('Soluciones numéricas con h=0.125');
print -dpdf 'p91.pdf';
%
% Detalle de la gráfica haciendo zoom:
x2 = 0.98:0.0001:1;
plot(x2,pp31(x2), 'b-', x2,pp41(x2), 'r-')
xlabel('x');
ylabel('y(x)');
title('Zoom de las soluciones numéricas con h=0.125');
print -dpdf 'p92.pdf';
%
% Gráfica del spline cúbico:
plot(x1,pp31(x1), 'b-')
xlabel('x');
ylabel('y(x)');
title('Gráficas del spline cúbico con h=0.125');
print -dpdf 'p93.pdf';
%
% Gráfica del pseudo spline cúartico:
plot(x1,pp41(x1), 'r-')
xlabel('x');
ylabel('y(x)');
title('Gráficas del pseudo spline cúartico con h=0.125');
print -dpdf 'p94.pdf';
%
% Evolución de las soluciones al ir disminuyendo el paso:
% Tomamos h= 0.125/2^i para i=0:5
% Utilizamos un bucle for:
err31=zeros(6,6);
err41=zeros(6,6);
for i=0:5
    n=8*2^i;
    w=[a:0.125/2^i:b];
    c31=spline3pcnl2(@spline3p,@spline3pnl,a,b,n,w,f,f1,f2,z1,z2,
        nmax);
    c41=spline4pcnl2(@spline4p,@spline4pnl,a,b,n,w,f,f1,f2,z1,z2,
        nmax);
%
% Construcción de las funciones splines:
p31 = @(z) ppval(mkpp(w,c31),z);
p41 = @(z) ppval(mkpp(w,c41),z);
%
% Valores de las soluciones numéricas en los puntos [a:0.2:b]:
x2=[a:0.2:b];
```



```
err31(:,i+1)=p31(x2)';
err41(:,i+1)=p41(x2)';
end
%
% Gráfica de las soluciones h=0.00390625:
x1 = a:0.005:b;
plot(x1,p31(x1), 'b-',x1,p41(x1), 'r-')
xlabel('x');
ylabel('y(x)');
title('Soluciones numéricas con h=0.00390625');
print -dpdf 'p95.pdf';
%
%
% Tabla de valores de las soluciones numéricas (h= 0.125/2^i,
i=0:5):
fprintf('valores de las soluciones numéricas (h= 0.125/2^i,
i=0:5):\n')
fprintf(' Por columnas: x, err31(:,1), err31(:,2), err31(:,3),
err31(:,4), err31(:,5), err31(:,6)\n')
fprintf('% 3.1f % 9.6e % 9.6e % 9.6e % 9.6e % 9.6e % 9.6e\n',
[x2;err31(:,1)';err31(:,2)';err31(:,3)';err31(:,4)';
err31(:,5)';err31(:,6)'])
%
fprintf(' Por columnas: x, err41(:,1), err41(:,2), err41(:,3),
err41(:,4), err41(:,5), err41(:,6)\n')
fprintf('% 3.1f % 9.6e % 9.6e % 9.6e % 9.6e % 9.6e % 9.6e\n',
[x2;err41(:,1)';err41(:,2)';err41(:,3)';err41(:,4)';
err41(:,5)';err41(:,6)'])
%
```

### 7.3.9. Ejemplo 9

```
% EJEMPLO 9
% Fijamos los parámetros del problema:
c=8;
%
% Solución exacta:
%
% Inputs para las llamadas de función:
a=0;
b=1;
n=8;
x=[a:0.125:b];
f=@(x,y,z) -c.*x.*cos(y);
f1=@(x,y,z) c.*x.*sin(y);
f2=@(x,y,z) 0;
z1=0;
z2=0;
```

```
nmax=6;
%
% Llamadas de función:
c31=spline3pcnl3(@spline3p,@spline3pn1,a,b,n,x,f,f1,f2,z1,z2,nmax)
;
c41=spline4pcnl3(@spline4p,@spline4pn1,a,b,n,x,f,f1,f2,z1,z2,nmax)
;
%
% Construcción de los splines:
pp31 = @(z) ppval(mkpp(x,c31),z);
pp41 = @(z) ppval(mkpp(x,c41),z);
%
% Tabla de valores:
fprintf('Valores de las soluciones numéricas (h= 0.125):\n')
x0=[a:0.2:b];
fprintf(' Por columnas: x, y31(x), y41(x)\n')
fprintf('% 3.1f % 9.6e % 9.6e\n',[x0;pp31(x0);pp41(x0)])
%
% Gráfica de las soluciones con h=0.125:
x1 = a:0.01:b;
plot(x1,pp31(x1),'b-',x1,pp41(x1),'r-')
xlabel('x');
ylabel('y(x)');
title('Soluciones numéricas con h=0.125');
print -dpdf 'p101.pdf';
%
% Detalle de la gráfica haciendo zoom:
x2 = 0.98:0.0001:1;
plot(x2,pp31(x2),'b-',x2,pp41(x2),'r-')
xlabel('x');
ylabel('y(x)');
title('Zoom de las soluciones numéricas con h=0.125');
print -dpdf 'p102.pdf';
%
% Gráfica del spline cúbico:
plot(x1,pp31(x1),'b-')
xlabel('x');
ylabel('y(x)');
title('Gráficas del spline cúbico con h=0.125');
print -dpdf 'p103.pdf';
%
% Gráfica del pseudo spline cúartico:
plot(x1,pp41(x1),'r-')
xlabel('x');
ylabel('y(x)');
title('Gráficas del pseudo spline cúartico con h=0.125');
print -dpdf 'p104.pdf';
%
% Evolución de las soluciones al ir disminuyendo el paso:
% Tomamos h= 0.125/2^i para i=0:5
% Utilizamos un bucle for:
```

```
err31=zeros(6,6);
err41=zeros(6,6);
for i=0:5
n=8*2^i;
w=[a:0.125/2^i:b];
c31=spline3pcnl3(@spline3p,@spline3pnl,a,b,n,w,f,f1,f2,z1,z2,nmax)
;
c41=spline4pcnl3(@spline4p,@spline4pnl,a,b,n,w,f,f1,f2,z1,z2,nmax)
;
%
% Construcción de las funciones splines:
p31 = @(z) ppval(mkpp(w,c31),z);
p41 = @(z) ppval(mkpp(w,c41),z);
%
% Valores de las soluciones numéricas en los puntos [a:0.2:b]:
x2=[a:0.2:b];
err31(:,i+1)=p31(x2)';
err41(:,i+1)=p41(x2)';
end
%
% Gráfica de las soluciones h=0.00390625:
x1 = a:0.005:b;
plot(x1,p31(x1),'b-',x1,p41(x1),'r-')
xlabel('x');
ylabel('y(x)');
title('Soluciones numéricas con h=0.00390625');
print -dpdf 'p105.pdf';
%
%
% Tabla de valores de las soluciones numéricas (h= 0.125/2^i,
i=0:5):
fprintf('valores de las soluciones numéricas (h= 0.125/2^i,
i=0:5):\n')
fprintf(' Por columnas: x, err31(:,1), err31(:,2), err31(:,3),
err31(:,4), err31(:,5), err31(:,6)\n')
fprintf('% 3.1f % 9.6e % 9.6e % 9.6e % 9.6e % 9.6e % 9.6e\n',
[x2;err31(:,1)';err31(:,2)';err31(:,3)';err31(:,4)';
err31(:,5)';err31(:,6)'])
%
fprintf(' Por columnas: x, err41(:,1), err41(:,2), err41(:,3),
err41(:,4), err41(:,5), err41(:,6)\n')
fprintf('% 3.1f % 9.6e % 9.6e % 9.6e % 9.6e % 9.6e % 9.6e\n',
[x2;err41(:,1)';err41(:,2)';err41(:,3)';err41(:,4)';
err41(:,5)';err41(:,6)'])
%
```