Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Eléctrica

Resolución numérica de problemas formulados en términos de EDOs

Autor:

Alonso de la Torre, Óscar

Tutor(es):

Álvarez López, Jorge

Departamento de Matemática Aplicada

Valladolid, Julio 2020.

AGRADECIMIENTOS

Con este trabajo se pone fin a una etapa académica llena de experiencias. No es un camino que haya recorrido solo y ha sido posible gracias al esfuerzo continuado de muchas personas, a las que pido perdón si omito algún nombre.

En primer lugar, al impulsor de este proyecto, Jorge Álvarez López, que tiene una gran intuición a la hora de ver por dónde se pueden presentar los problemas durante el desarrollo de este TFG. Gracias porque ha sido un apoyo inestimable en los tiempos más duros de la pandemia y siempre ha puesto todo su entusiasmo y conocimiento en las matemáticas.

A mi familia, en especial a mi madre, Cati, porque sin ella no hubiera podido hacer nada de esto.

A Mariné, que desde hace dos años me anima con los estudios y en especial este último 2020.

A mis profesores de matemáticas, José Antonio Abia y Marisa Fernando, que me enseñaron además de los fundamentos matemáticos de la carrera, que es imprescindible poner el mayor esfuerzo y trabajar duro para aprender. No hay caminos fáciles ni atajos.

A Jesús, que viéndole trabajar cada día, ha sido una fuente inagotable de motivación.

A Todos, GRACIAS!!

Añadir un sentido pésame a todas esas familias que recientemente han perdido un familiar por el COVID-19, y todas las que lo siguen perdiendo. Son miles los casos que hay en España y no debemos permitir que esto se olvide nunca. El encabezado va dedicado a toda la gente que ha fallecido de una manera tan sin sentido, a sus familiares y amigos, y a toda la sociedad que ahora necesita más apoyos que nunca.

RESUMEN

Resolver una ecuación diferencial es una operación artística dentro de las matemáticas. No existe ningún modo que satisfaga los criterios de rapidez, precisión, simplicidad, exactitud y adaptabilidad, para cualquier tipo de ecuación.

Este trabajo se basa en investigar métodos de colocación sobre una partición de un intervalo en la que deseamos conocer la solución de una ecuación diferencial y comprobar cómo se puede modificar para que satisfaga de un modo más preciso, alguno de los criterios en los que estamos interesados. Por ejemplo, se puede hacer una estimación más rápida, a costa de ser menos precisa. O se puede hacer una estimación más precisa en zonas más complicadas y más suave en zonas menos complicadas.

Se analizará el campo de aplicación de este método y su comparación con otros existentes.

PALABRAS CLAVE

Spline, Método de Colocación, Ecuaciones diferenciales ordinarias, solver, campo de pendientes

ABSTRACT

Solving a differential equation is an artistic operation within mathematics. There is no mode that meets the criteria of speed, precision, simplicity, accuracy and adaptability, for any type of equation.

This work is based on investigating collocation methods on a partition of an interval in which we want to know the solution of a differential equation and to see how it can be modified to meet in a more precise way, some of the criteria we are interested in. For example, you can make a faster estimate, at the cost of being less accurate. Or you can make a more accurate estimate in more complicated areas and softer in less complicated areas.

The scope of this method and its comparison with other existing methods will be analyzed.

KEY WORDS

Spline, Collocation method, Ordinary Differential Equations, Solver, Slope Field

ÍNDICE

ÍNDICE DE FIGURAS

Nota: Las figuras que no indiquen expresamente la fuente, han sido realizadas por el autor de este trabajo, bien de su propia mano, o bien mediante el empleo del software utilizado.

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

1. INTRODUCCIÓN Y OBJETIVOS

1.1. JUSTIFICACIÓN

El inicio del estudio teórico de las Ecuaciones Diferenciales se remonta a finales del siglo XVII, poco después de la aparición del cálculo diferencial e integral. Pronto se convirtieron en una poderosa herramienta para la investigación científica y técnica, propiciando un enorme progreso.

Son muchos los problemas que surgen en el ámbito de la ingeniería, la industria, la física y la ciencia en general, que pueden ser formulados en términos de ecuaciones diferenciales de primer orden.

En muchos aspectos, la naturaleza puede describirse por medio de ecuaciones diferenciales, por lo que su estudio y resolución nos proporcionará un mejor entendimiento de los fenómenos naturales.

Es por ello que este tipo de ecuaciones reciben una atención especial en el campo de la matemática aplicada.

Desafortunadamente, la inmensa mayoría de los problemas formulados en términos de este tipo de ecuaciones, no admite una resolución analítica, por lo que cobra especial interés su resolución numérica con vistas a disponer de aproximaciones a la solución exacta suficiente precisas.

La mayoría de los métodos numéricos orientados a la aproximación de soluciones de ecuaciones diferenciales se basan en la idea de calcular valores aproximados de la solución exacta del problema en una serie de puntos del intervalo de interés.

En este Trabajo de Fin de Grado, nos centraremos en aproximar, no sólo unos puntos concretos, sino que, además, obtendremos una aproximación directa a la solución particular de la EDO. Luego compararemos estos métodos con los implementados en el software de uso.

Para su implementación, utilizaremos el software matemático Matlab, por ser éste un software destinados al Análisis Numérico con cuyo uso estamos familiarizados.

Matlab comenzó siendo un paquete interactivo orientado a cálculos matriciales, de ahí su nombre: MATrix LABoratory, desarrollado por C. Moler con fines docentes; pero en poco tiempo se extendió su uso entre los ingenieros y científicos como una práctica herramienta para el cálculo numérico. En los años 80 se rediseñó y pasó a convertirse en un software comercial que incorpora un lenguaje de programación de alto nivel (lenguaje M). También proporciona multitud de funcionalidades de visualización, diseño y simulación, así como bibliotecas orientadas a distintos propósitos.

Por todo ello, utilizaremos Matlab a la hora de implementar y comparar los distintos métodos desarrollados en esta memoria para la resolución aproximada de Ecuaciones Diferenciales Ordinarias de Primer Orden.

Los métodos que proponemos se basan en la aproximación de las soluciones de E.D.O.s, lineales y no lineales, de primer orden, mediante Splines y Pseudo-Splines con distintos grados de regularidad, utilizando para ello técnicas de colocación.

Incluimos una serie de experimentos numéricos, implementados en Matlab, con vistas a analizar y comparar los distintos métodos considerados, así como su orden de convergencia.

1.2. OBJETIVOS

En la realización de este trabajo se fijan los siguientes objetivos:

- Analizar la importancia de las Ecuaciones Diferenciales de Primer Orden en el modelado de muchos fenómenos comunes, mostrando algunos ejemplos relacionados con la ingeniería.
- Estudiar diferentes métodos que permiten aproximar las soluciones de Problemas de Valores Iniciales dados en términos de Ecuaciones Diferenciales Ordinarias de Primer Orden.
- Proponer y comparar distintas estrategias relativas al grado de regularidad y a los puntos de colocación utilizados a la hora de diseñar los métodos numéricos propuestos.
- Programar en código de Matlab los métodos comentados anteriormente y realizar simulaciones con vistas a determinar su orden y comportamiento al ser aplicados a distintos problemas.
- Aplicar los códigos de los métodos de resolución a varios problemas, algunos de ellos de interés en ingeniería, analizando y comparando los resultados obtenidos.
- Analizando las soluciones obtenidas con Matlab para los diferentes métodos, obtener conclusiones que nos permitan describir las ventajas que pueden aportar las distintas variantes que han sido propuestas.
- Por último, pero no por ello menos importante: no conviene olvidar la principal motivación de todo trabajo de fin de grado, esto es, su finalidad formativa y educativa.

1.3. ESTRUCTURA

Vamos a estructurar este TFG en varias partes:

En la **primera parte** se da un breve pero necesario repaso a las ecuaciones diferenciales ordinarias de primer orden y se analiza la importancia que tienen desde el punto de vista de la ingeniería. También Se observan los Métodos tradicionales de resolución de este tipo de ecuaciones, y la manera en que un ordenador trabaja con ellos.

En la **segunda parte** se introduce el método de resolución basado en una sucesión de puntos de colocación y con la estructura basada en Splines. Se analiza con profundidad el caso básico, qué nos va a servir de base para todos los demás. También veremos algunas mejoras para este método cómo es el imponer más puntos de colocación o más regularidad en el Spline.

En la **tercera parte** nos enfrentamos al problema de adaptarnos al entorno en el que se encuentra la solución y ser más o menos exigentes con el tamaño de paso, incidiendo en un paso más pequeño cuando la solución sea más difícil de aproximar y pudiendo ser más grande cuando la solución sea más fácilmente aproximable.

En la **cuarta** se analizan los resultados de estos métodos Mediante la realización de distintos experimentos comparativos sobre funciones de prueba conocidas, justificando los cambios que se pueden ir introduciendo en los métodos. Se analiza cómo influye el cálculo del error estimado y sus modificaciones, y cómo afectan éstas a la aproximación.

2. ECUACIONES DIFERENCIALES ORDINARIAS Y MODELOS

El modelo matemático asociado a un fenómeno físico consiste en la extracción de las propiedades más relevantes que tienen importancia en el desarrollo de ese fenómeno, para establecer una representación abstracta basada en reglas matemáticas.

Los modelos físicos parten de una hipótesis. Tratamos de predecir qué va a pasar mediante la observación de lo que ya ha acontecido. Y al principio todos esos modelos parten de la intuición y la creatividad de una persona de ciencia. Es lo que realmente hace avanzar la tecnología. Luego, la matemática, se encarga de domar esa creatividad y de darle el rigor que hace que la ciencia sea ciencia.

Todos los fenómenos que conocemos se pueden describir de este modo usando los modelos matemáticos ligados a las distintas leyes físicas, que nos van a proporcionar ese soporte riguroso y formal que nos permite analizar con un muy alto grado de exactitud, cualquier fenómeno.

De entre todos los tipos de ecuaciones existentes en el universo matemático, en este trabajo de fin de grado, nos vamos a ocupar únicamente del estudio de las ecuaciones diferenciales, ya son la forma natural de describir problemas que involucren una variación o un cambio de algún tipo de magnitud.

Si atendemos a los distintos tipos de modelos, los más simples son los que pueden ser descritos mediante una ecuación diferencial de primer orden. Estos modelos son adecuados para representar multitud de fenómenos.

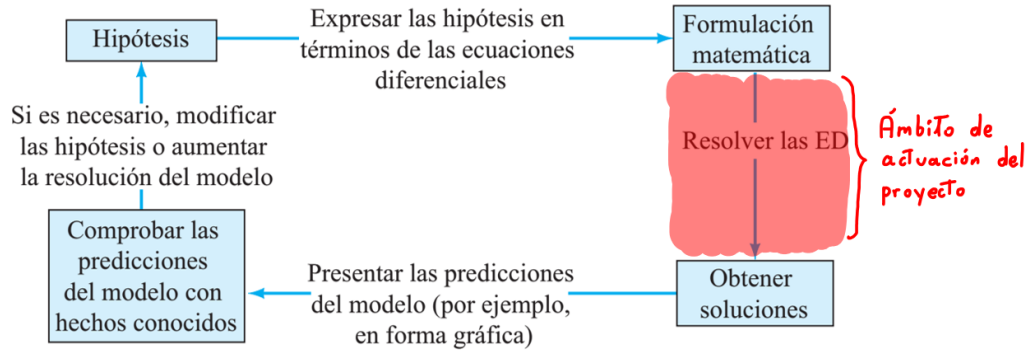


Figura 2-1: Forma de trabajar y ámbito de actuación

Fuente: Circuitos Eléctricos. Fraile mora.

(Mora, 2012)

El modelado de un fenómeno se puede ver como la unión de dos tareas:

- La representación de dicho fenómeno mediante una formulación matemática.
- El análisis numérico de dicho modelo.

La formulación matemática consiste, como se ha dicho antes, en representar de forma abstracta las relaciones entre las magnitudes más relevantes que intervienen en el fenómeno objeto del estudio, a menudo en forma de ecuaciones diferenciales. Esto se consigue usando distintas suposiciones basadas en la observación y la experiencia, acerca de cómo funciona el fenómeno.

El análisis numérico de dicho modelo consiste en ensayar el modelo obtenido a fin de comprobar la exactitud de sus resultados con las observaciones realizadas y estimar los parámetros principales del proceso.

(Bernis, 2013)

2.1. INTRODUCCIÓN A LAS ECUACIONES DIFERENCIALES ORDINARIAS

Una ecuación es la expresión matemática de una condición de igualdad. Lo habitual es que, en dicha expresión, aparezcan términos que se conozcan y otros que se desconozcan. En el caso de una ecuación diferencial, el término que se desconoce es una función. Además, en dicha ecuación debe aparecer la expresión de la derivada de dicha función respecto de la variable independiente.

Ya solo nos queda decir que si es ordinaria significa que la función solución contiene una única variable independiente.

En ingeniería, lo más habitual es que la variable independiente sea el tiempo, ya que se estudian fenómenos dinámicos.

Un ejemplo clásico de este tipo de ecuaciones es la que describe la evolución con respecto al tiempo del ángulo α que forma un péndulo respecto de su vertical, si lo dejamos libre con un cierto ángulo inicial. La ecuación que describe esto es:

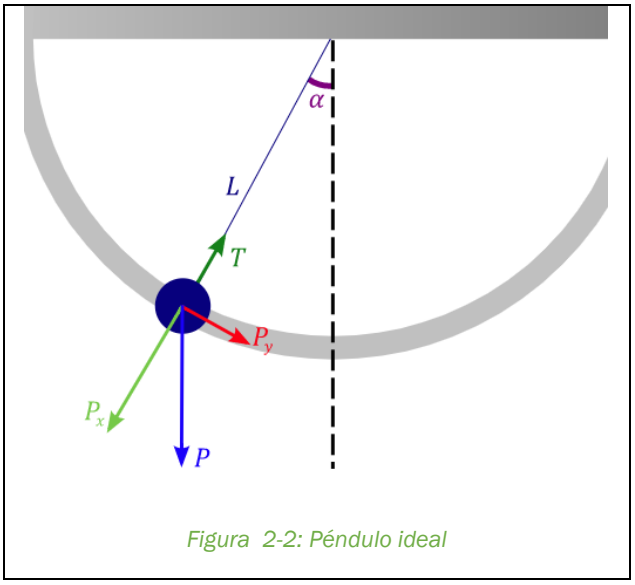


Figura 2-2: Péndulo ideal

Fuente: https://ca.m.wikipedia.org/wiki/Fitxer:Pendulo.svg

$\frac{d^2\,\alpha(t)}{dt} = -\frac{g}{l} \cdot sen(\alpha)$	*Ecuación 2-1: EDO del péndulo*

Donde g es la aceleración gravitatoria estándar en la superficie de la tierra y l es la longitud de la cuerda del péndulo.

(Ross, 1981)

2.2. INTRODUCCIÓN A LAS ECUACIONES DIFERENCIALES DE PRIMER ORDEN

En este TFG nos vamos a ocupar de las ecuaciones diferenciales ordinarias de primer orden, esto es las que únicamente contienen la expresión de la primera derivada de la función solución y que contienen una única variable independiente.

Estas ecuaciones pueden ser lineales o no lineales, dependiendo de si la función solución o su derivada, aparecen de una manera lineal en la ecuación. A la hora de desarrollar el método Nos vale para los dos casos, ya que se diferencian únicamente en la resolución del sistema de ecuaciones.

Antes de empezar, Hay que destacar que Vamos a estudiar la ecuación Únicamente en un intervalo para la variable independiente, $[a, b]$ con $a < b$. También hay que destacar en la función $y(x)$ es continua y derivable en el intervalo $[a, b]$.

Vamos a comenzar con la expresión general de una ecuación de este tipo:

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

Donde $q(x)$ y $r(x)$ son funciones continuas en el intervalo $[a, b]$

Como caso particular, si $r(x) = 0$ para cualquier valor de la variable independiente, la ecuación diferencial es homogénea. En cualquier otro caso, la ecuación resultante es no homogénea.

Un ejemplo de una ecuación de este tipo es la que rige el comportamiento frente al tiempo de la tensión en bornes de un condensador que se está cargando.

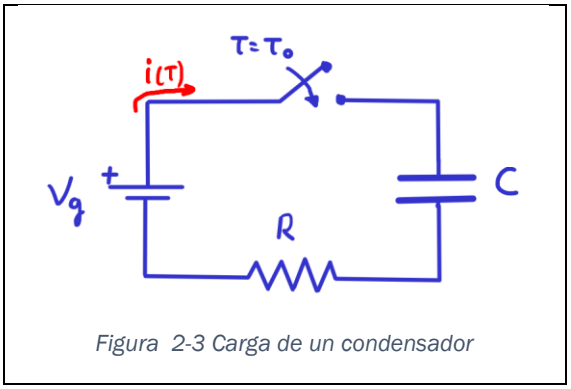


Figura 2-3 Carga de un condensador

$$y' = -\frac{1}{C \cdot R} \cdot y + \frac{V_g}{R \cdot C}$$

Ecuación 2-2 EDO carga condensador

Siendo

$y = V(t)$, tensión entre los bornes del condensador

R, parámetro de definición de la resistencia.

C, capacidad del condensador.

V_g, Tensión de la fuente

Esta ecuación tiene una resolución general analítica que corresponde con la expresión:

$$y = V_g \cdot \left(1 - e^{-\left(\frac{t}{RC}\right)}\right)$$

Ecuación 2-3 Solución carga de condensador

Si en el instante $t = 0$ el condensador se encuentra descargado, es decir, sin tensión alguna entre sus bornes. Con este dato de $V(0) = 0$, obtenemos la solución particular del problema.

Más adelante se abordará este problema en concreto. En la fase de experimentación.

2.3. FORMA DE ABORDAR EL PROBLEMA

En la gran mayoría de los caos no podemos obtener una solución analítica del problema, pero podemos aproximar la curva que se obtiene con la solución exacta para un PVI discretizando la variable independiente en el intervalo de interés e imponiendo distintas condiciones en cada punto de ese intervalo.

En los siguientes párrafos, vamos a definir los distintos conceptos que emplearemos, debido al método particular de solución que vamos a emplear.

Los métodos numéricos son un arte, ya que en función de que estrategia que usemos, vamos a obtener una solución muy cercana al objetivo o más alejada. Por las propiedades de convergencia de los métodos es razonable pensar que cuando más pequeño demos los pasos, más nos vamos a acercar al objetivo. Pero hay que buscar siempre un balance entre el número de pasos, el coste computacional, las exigencias de nuestro problema… No podemos dar pasos infinitamente pequeños y si les damos muy grandes obtenemos aproximaciones que no nos van a servir. El tamaño de paso no lo podemos conocer a priori, por eso, se puede calificar estos métodos como arte, en los que la experiencia del programador es fundamental.

Resulta muy adecuado, porque se refleja muy bien la forma de trabajar de los métodos numéricos, asemejar el problema al del movimiento de un cuerpo situado en la posición inicial y sometido a un campo de fuerzas o pendientes en base a la ecuación diferencial. El problema se reduce a buscar los puntos por los que pasa ese cuerpo a lo largo del intervalo de interés.

Tomaremos, por ejemplo, la función

$$y = e^{-2 \cdot x}$$

Y vamos a dibujar su campo de pendientes asociado para el intervalo $[0, 5]$

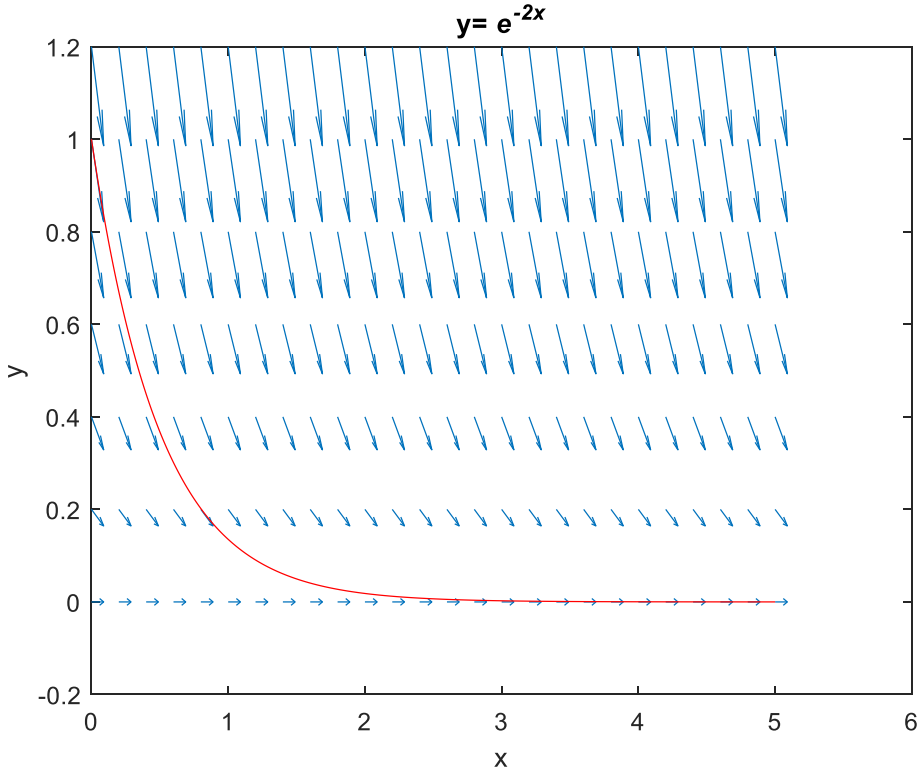


Figura 2-4 Campo de pendientes poco denso

Vemos que, a partir de la condición inicial en $x = 0$, siguiendo el campo de pendientes, se puede aproximar la solución particular de esa función.

Ahora nos preguntamos si existe alguna forma de mejorar esto. Si aumentamos el número de puntos en el que conocemos la pendiente, tenemos:

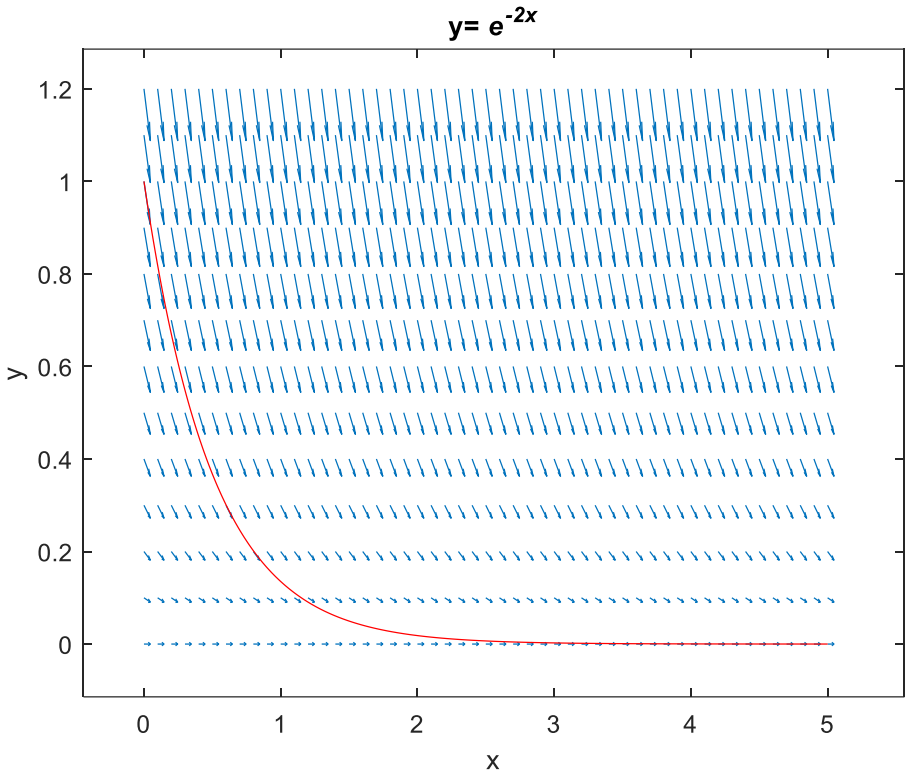


Figura 2-5 Campo de pendientes más denso

Parece, de este modo que, aumentando el número de puntos, tenemos más información y mejor puede ser el método.

Sin embargo, la mayoría de puntos que se encuentran representados no nos sirven para nada. Lo que le importa al ordenador es obtener la información del campo de pendientes en el entorno de los sitios por los que pasa. Imaginemos que pudiéramos representar el mismo número de puntos, pero más cerca de la función. En ese caso, vamos a ser capaces de aproximar mejor la solución. Veamos las siguientes ampliaciones de una pequeña zona de la curva.

También se puede ver que, para incrementos de paso lo suficientemente pequeños, aproximar localmente esa función por un Spline con un gran componente lineal, no es una locura, ya que la función parece una recta, como se puede ver en la figura 2 6. Aunque no hay que perder de vista, que realmente no lo es, tal como se ve en el ejemplo descrito en la figura 2 5.

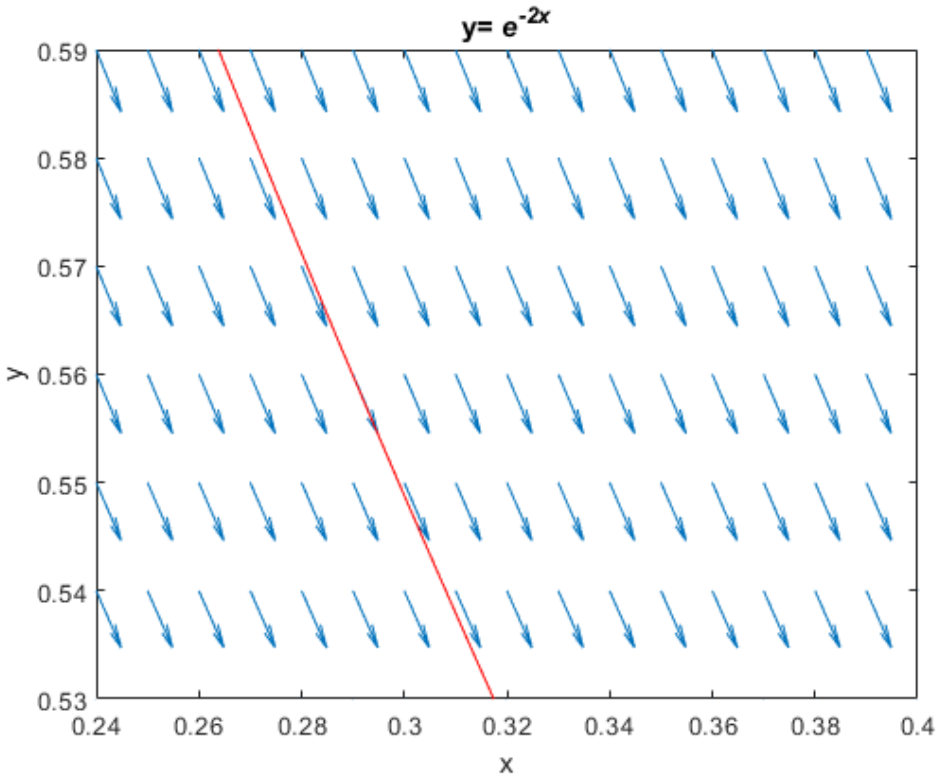


Figura 2-6 Ampliación del campo con espaciado de 0,01

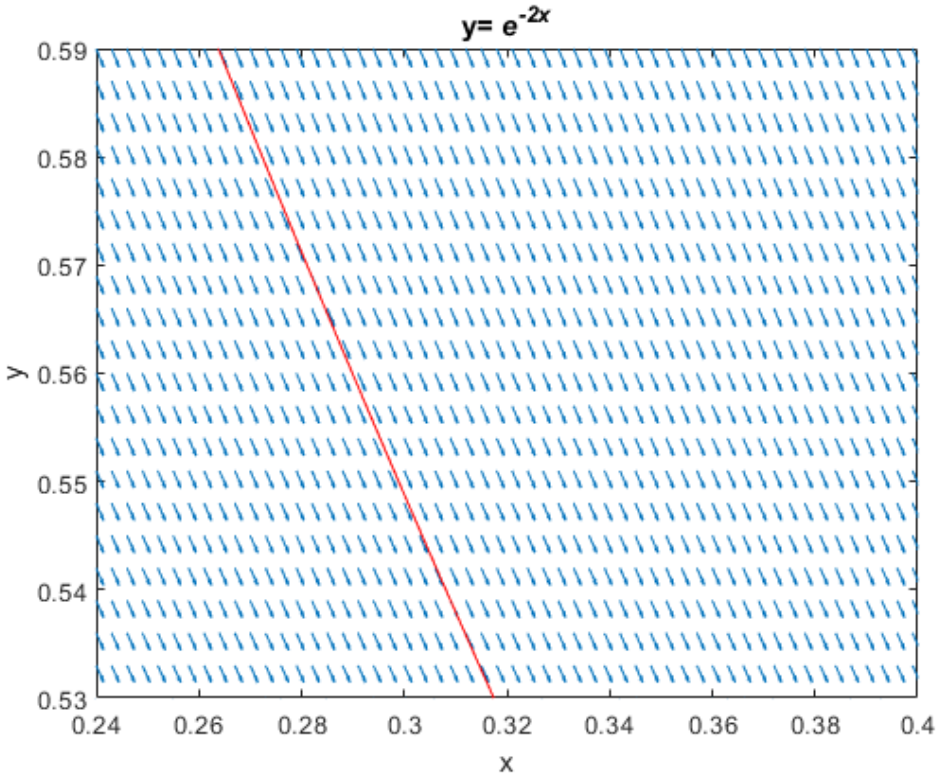


Figura 2-7 Ampliación del campo con espaciado de 0,003

Si somos capaces de aproximar localmente el campo por donde se va a mover nuestra aproximación, seremos capaces de obtener unas muy buenas aproximaciones. Más o menos, la forma de obtener información del campo y de recorrerlo, es la forma de trabajar en la que difieren los distintos métodos de resolución de ecuaciones.

Otro punto que es obligado destacar es que el campo de pendientes lo recorremos a lo largo del eje de las variables independientes, por lo que, cuando tengamos una pendiente elevada, es obligado dar pasos más pequeños. Ya que la componente tangente a la curva que se desvía de nuestra dirección de avance es elevada. En cambio, en los tramos en que el campo de pendientes sea más horizontal, se puede dar un tamaño de paso más grande.

Más información sobre esto se puede encontrar en (O'Connor, 2017)

2.4. MÉTODO DE EULER

El método de Euler es el método básico para comprender cómo funcionan los métodos numéricos.

En la práctica tiene poco uso, pero su interés radica en que sirve de base para otros métodos y se puede ver de manera muy sencilla el modo de trabajar.

De manera general, los métodos numéricos, comparten una característica que se van a introducir mediante este primero.

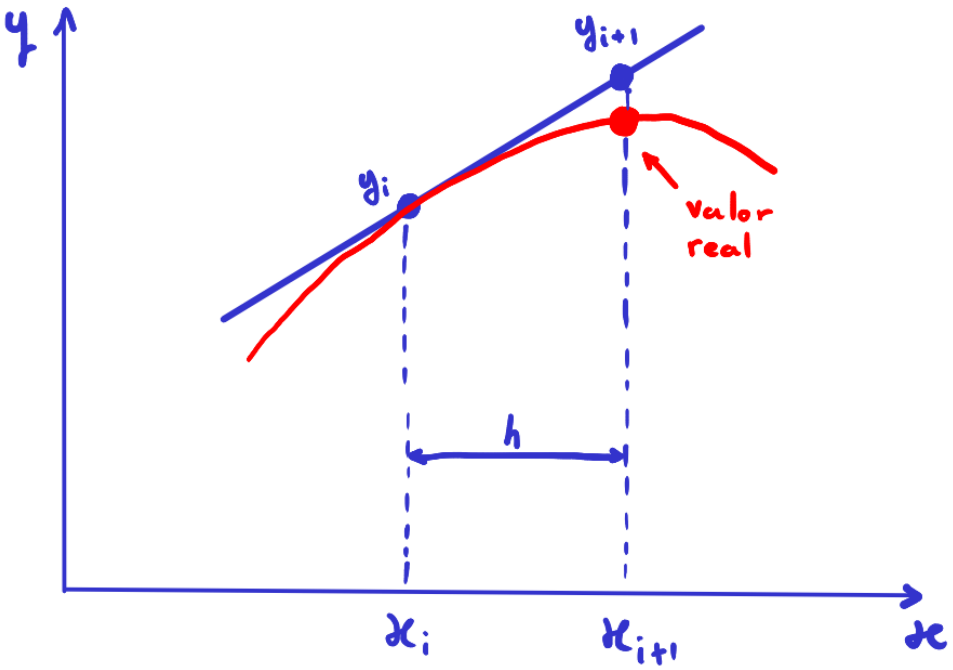


Figura 2-8 Método de Euler

Una vez formulado el problema y obtenidas las expresiones correspondientes a $q(x)$, $r(x)$, y conocidos los extremos del intervalo de interés $[a, b]$ y la condición inicial $y(a)$ se procede a lo siguiente:

Particionamos el intervalo de interés en un número n de subintervalos de tamaño de paso h.

$$h = \frac{b-a}{n}$$

$$x_i = a + i \cdot h, \qquad i = 0, 1, 2, \cdots, n$$

Obtenemos de este modo una partición

$$[a, b] = [a, \quad x_1, \quad x_2, \quad \cdots, \quad x_{n-1}, \quad b]$$

De este modo logramos transformar el problema a resolver en n problemas más pequeños y con un ámbito más local. Esto es común a la filosofía de trabajo del resto de métodos numéricos.

La unión de todos los subintervalos $[x_i, x_{i+1}]$ conforma el intervalo $[a, b]$ y, además, cada intervalo tiene sólo los nodos en común con sus intervalos adyacentes.

Aquí lo que hace el método de Euler es suponer, que en un intervalo lo suficientemente pequeño, la función se comporta de una manera lineal. Esto es una suposición muy habitual cuando trabajamos en torno a algún punto de equilibrio.

Dado que conocemos para el primer subintervalo el punto inicial y su derivada en ese punto, se puede aproximar la función por su desarrollo de Taylor de orden 1. Con esta aproximación, podemos estimar cuánto vale la función en el nodo x_1.

$$y_1 = y_0 + h \cdot [\, q(x_0) \cdot y_0 + r(x_0) \,]$$

Aplicando de una manera recursiva esta ecuación obtenemos una sucesión de puntos que constituye la solución aproximada de la ecuación.

Debido a que estamos aproximando una curva mediante una recta, la estimación de cada punto no va a ser la solución exacta de la ecuación. A este error lo llamaremos error de truncamiento. Se puede disminuir el error dando pasos más pequeños, dándose el caso de que $error \to 0$ cuando $h \to 0$. Sin embargo, no podemos dar los pasos infinitamente pequeños, ya que los

ordenadores tienen un límite para trabajar y se van produciendo errores de redondeo, cuando trabajamos con números pequeños, que hacen del todo erróneos los cálculos.

2.5. MÉTODOS DE RUNGE KUTTA

Como hemos visto, el método de Euler presenta unas buenas propiedades en cuanto a estabilidad y convergencia del método. No obstante, tenemos la limitación de que hay que dar pasos muy pequeños si queremos conseguir buenos resultados.

Para conseguir una mayor rapidez se necesita aproximar la función por algo más preciso que por sus términos lineales de expansión en serie de Taylor, pero cada término adicional que podamos añadir, requiere el cálculo de las derivadas sucesivas de la función.

Los métodos de Runge-Kutta son capaces de solventar en gran parte este problema.

Un método Runge-Kutta como el planteado en los anexos forma su estimación del siguiente punto de la función, en base a 4 cálculos del campo de pendientes., partiendo de un punto (x_i, y_i) y llegando a otro punto (x_{i+1}, y_{i+1}) del siguiente modo:

$$y_{i+1} = y_i + \frac{h}{6} \cdot (k_1 + 2 \cdot k_2 + 2 \cdot k_3 + k_4)$$

Siendo los distintos coeficientes de k, la pendiente en los siguientes puntos:

k_1: Punto de la última aproximación conocida.

k_2: Punto al que nos lleva la pendiente de k_1, de recorrer la mitad del intervalo, partiendo de la última aproximación.

k_3: Punto al que nos lleva la pendiente de k_2, de recorrer la mitad del intervalo, partiendo de la última aproximación

k_4: Punto al que nos lleva la pendiente de k_3 de recorrer todo el intervalo, partiendo de la última aproximación

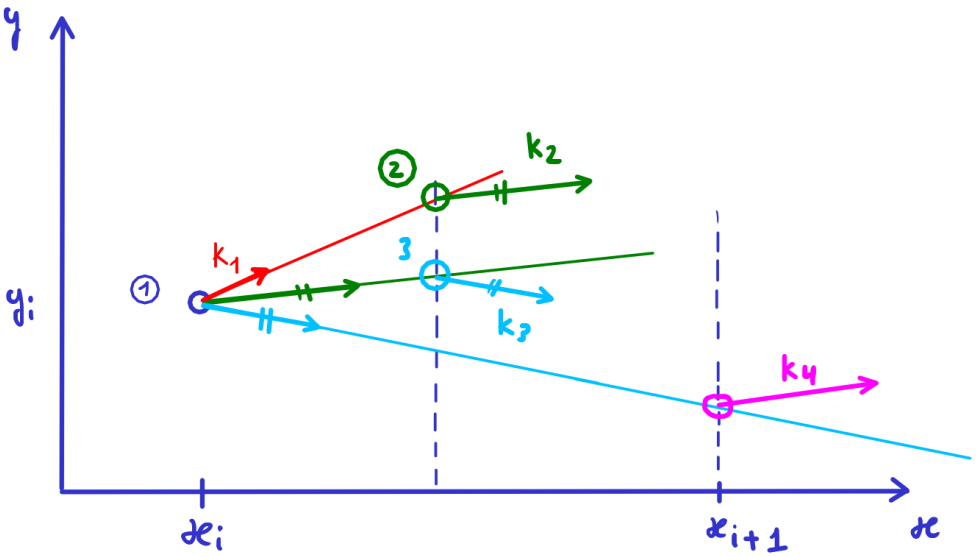


Figura 2-9 Cálculo de los k de un Runge-Kutta

Formalmente, se escribiría como:

$$k_1 = f(t_i . y_i)$$

$$k_2 = f\left(t_i + \frac{1}{2} \cdot h, y_i + \frac{1}{2} \cdot h \cdot k_1\right)$$

$$k_3 = f\left(t_i + \frac{1}{2} \cdot h, y_i + \frac{1}{2} \cdot h \cdot k_2\right)$$

$$k_4 = f(t_i + h, y_i + k_3 \cdot h)$$

Ecuación 2-4: Coeficientes k de Runge-Kutta

Al final obtenemos una media ponderada de la pendiente en el intervalo en que avanzamos. Nótese que los coeficientes k_2 y k_3, correspondientes a la evaluación de la pendiente en el punto intermedio del intervalo tienen más importancia que los coeficientes k_1 y k_4.

3. MÉTODOS DE RESOLUCIÓN BASADOS EN SPLINES

Los métodos basados en los Splines nos proporcionan una aproximación que es continua y de clase C1 al menos en el dominio.

A través de la aproximación local en cada subintervalo, y con las condiciones de regularidad que imponen las funciones Spline, vamos a tener una aproximación global de la función objetivo.

3.1. INTRODUCCIÓN A LOS SPLINES

Las funciones Spline son relativamente nuevas en matemáticas. Aunque llevan muchos años con nosotros, su tratamiento numérico se ha hecho posible gracias a los ordenadores y a sus enormes posibilidades de cálculo.

Ya se han usado en ingeniería con gran fruición durante la primera y segunda guerra mundial, en la construcción de aviones y barcos, si bien se utilizaban de una manera mecánica y no numérica.

La filosofía del método es sencilla. Si tenemos una serie de n puntos existe un único polinomio, de grado $n-1$ que pasa por todos ellos. Esto está muy bien si queremos aproximar funciones a partir de unos puntos conocidos de ellas, pero en funciones tupo runge no funciona muy bien, ya que el polinomio interpolador oscila mucho respecto de la función a aproximar.

La solución a esto es bien fácil. Si con polinomios de grado alto la función oscila, vamos a usar polinomios de grado bajo en los que sea estable, y en vez de aproximar todo el intervalo de interés, vamos a ir aproximando cada vez una pequeña parte de él. Luego, en los puntos de unión de uno con otro, impondremos diversas condiciones de regularidad tales como continuidad de la función, de la primera derivada, ... etc. Lo más habitual es imponer sólo clase C1.

Y se puede ver que funcionan muy bien y tienen unas grandes propiedades de adaptabilidad a la función y a sus cambios en distintas zonas.

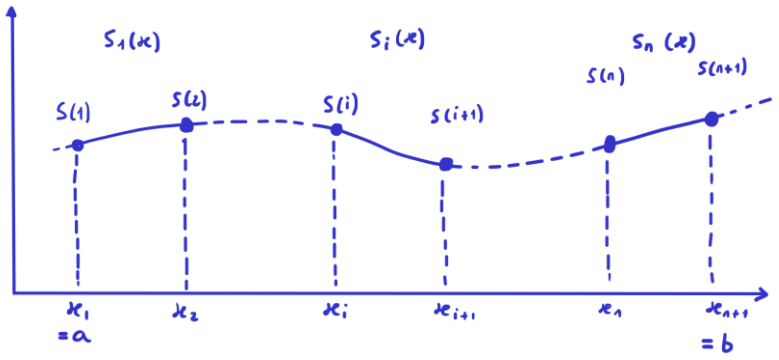


Figura 3-1 Esquema Splines

3.2. INTRODUCCIÓN AL MÉTODO.

Consideremos la ecuación diferencial ordinaria de primer orden:

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

Esta ecuación viene acompañada de una condición que garantiza que el problema tiene una única solución en el intervalo $[a, b]$.

$$y(c) = z_1 \quad c \in [a, b]$$

Condición inicial

Es el caso de que la condición la conocemos cuando $c = a$, es decir, al inicio el intervalo. Este tipo de problemas en los que a la EDO le acompaña esta condición se conoce como problemas de Cauchy y son los más habituales de encontrar.

Condición final

En este caso $c = b$. Se les conoce como problemas de valor final o problemas de valor inicial, dependiendo de la literatura a consultar.

Condición intermedia

En este caso $c \in (a, b)$. El valor conocido es un punto interior del intervalo.

Pretendemos obtener una función polinomial a trozos que aproxime de una manera suficientemente exacta a la función solución de la ecuación diferencial, es decir, $y(x) \cong S(x)$, donde $S(x)$ es un Spline basado en los nodos $\{x_i\}_1^{n+1}$.

Para obtener dicha aproximación, dividiremos el intervalo $[a, b]$ en n subintervalos.

Si expandimos la expresión de la función Spline, podemos ver de una forma fácil que se necesita determinar un número de parámetros igual al producto del número de subintervalos por el (grado+1) de los polinomios S_i. Como nuestro caso básico lo construiremos con polinomios de grado 2, necesitamos determinar $3 \cdot n$ incógnitas.

$$S(x) = \begin{cases} S_1(x) = a_1 + b_1(x - x_1) + c_1(x - x_1)^2 & x \in [x_1, x_2) \\ S_2(x) = a_2 + b_2(x - x_2) + c_2(x - x_2)^2 & x \in [x_2, x_3) \\ \dots & \\ S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 & x \in [x_i, x_{i+1}) \\ \dots & \\ S_n(x) = a_n + b_n(x - x_n) + c_n(x - x_n)^2 & x \in [x_n, x_{n+1}] \end{cases}$$

Ecuación 3-1 Definición de la función Spline

Los distintos métodos que vamos a exponer a continuación se basan en que, en los $n + 1$ nodos que sirven para delimitar cada subintervalo, se satisfaga la ecuación diferencial.

Además de ello, en los $n - 2$ nodos interiores del intervalo, se deben cumplir unas condiciones de regularidad entre los distintos polinomios.

Además de ello, se debe de satisfacer la condición inicial.

Todo esto nos da un sistema de ecuaciones de la siguiente forma:

$n - 1$	Condiciones de continuidad
$n - 1$	Condiciones de continuidad de la derivada
$n + 1$	Condiciones de colocación
1	Condición inicial

$3 \cdot n$	Total de ecuaciones

Por tanto, tenemos un sistema de ecuaciones compatible determinado, que nos garantiza que el problema va a tener solución y, además, esta va a ser única.

Adicionalmente a estas condiciones podremos imponer otras condiciones adicionales, las cuales pueden mejorar la estimación de la función, al imponer condiciones adicionales:

- Que se satisfaga la ecuación diferencial en los puntos medios de cada subintervalo. Esto nos proporciona n ecuaciones más y debemos aumentar un grado los polinomios del Spline. Esta modificación mejora de forma notable la exactitud de la aproximación en cada subintervalo.
- Que la derivada segunda de los distintos polinomios sea una función regular.

Si queremos modificar el método, necesitamos añadir n condiciones más cada vez que aumentamos de grado el polinomio. Esto se tratará cuando impongamos más condiciones de regularidad o más puntos de colocación.

La solución que obtendremos del programa va a estar dada en forma de dos elementos:

- Un vector de tamaño $N+1$, que almacena los nodos, distintos y en orden creciente y cuyo primer y último elemento corresponden con los extremos del intervalo a, y b.

$$[x_1 = a, \quad x_2, \quad x_3, \quad \cdots, \quad x_n, \quad x_{n+1} = b]$$

- Una matriz de tamaño $N \times (G+1)$, almacenando los coeficientes de los polinomios en cada subintervalo.

$$\begin{bmatrix} a_1 & b_1 & c_1 & \cdots \\ a_2 & b_2 & c_2 & \cdots \\ a_3 & b_3 & c_3 & \cdots \\ \vdots & \vdots & \vdots & \cdots \\ a_N & b_N & c_N & \ddots \end{bmatrix}$$

N es el número de subintervalos en los que se divide el intervalo de cálculo. La unión de todos ellos va a conformar el domino de nuestra solución aproximada.

G es el grado máximo de los polinomios que consideramos.

Para un subintervalo cualquiera, i, la solución viene proporcionada por la evaluación del polinomio

$$a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 + \cdots + n_i \cdot (x - x_i)^{G-1}$$

Para valores de $x \in [x_i, \ x_{i+1}]$

3.3. PLANTEAMIENTO MÉTODO DE COLOCACIÓN BÁSICO (SPLINE GRADO 2)

Vamos a considerar un problema de valores iniciales en el intervalo $[a, b]$.

$$\begin{cases} y'(x) = q(x) \cdot y(x) + r(x) \\ y(a) = z_1 \end{cases}$$

Ecuación 3-2 Problema de Cauchy

Con el objeto de encontrar una aproximación a la función $y(x)$, hacemos una partición del intervalo $[a, b]$ en n subintervalos, generando entonces los nodos x_i en los que impondremos que se satisfaga la ecuación diferencial:

$$a = x_1 < x_2 < \cdots < x_n < x_{n+1} = b$$

Consideramos, entonces, el siguiente Spline cuadrático, basado en los nodos $\{x_i\}_{i=1}^{n+1}$ definido del siguiente modo:

$$S(x) = \begin{cases} S_1(x) = a_1 + b_1(x - x_1) + c_1(x - x_1)^2 & x \in [x_1, x_2) \\ S_2(x) = a_2 + b_2(x - x_2) + c_2(x - x_2)^2 & x \in [x_2, x_3) \\ \cdots \\ S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 & x \in [x_i, x_{i+1}) \\ \cdots \\ S_n(x) = a_n + b_n(x - x_n) + c_n(x - x_n)^2 & x \in [x_n, x_{n+1}] \end{cases}$$

Ecuación 3-3 Spline de grado 2

El procedimiento para el cálculo de los coeficientes parte de las condiciones que imponemos en cada subintervalo.

En primer lugar, como se puede ver en la figura 3 2, tenemos unas condiciones que se deben cumplir en el primer subintervalo. Este es el intervalo de arranque del método. Una vez satisfechas estas condiciones y generado el primer Spline, el cálculo de los coeficientes para los subintervalos posteriores se hace de manera recursiva, imponiendo las condiciones de continuidad y de colocación, de la manera que se detalla.

Tabla 3-1 Condiciones a imponer en el Spline

Condiciones que imponemos en cada intervalo	
Intervalo inicial	Intervalo $i,\ i = 2, 3, \dots, n$
• Condición inicial • Condición de colocación en el nodo x_1 • Condición de colocación en el nodo x_2	• Condición de continuidad de la función. • Condición de continuidad de la derivada primera en el nodo x_i • Condición de colocación en el nodo x_{i+1}

En la siguiente figura se puede ver de forma gráfica las condiciones que se deben cumplir en cada subintervalo y las diferencias entre el cálculo de los coeficientes del polinomio asociado al primer subintervalo y al resto.

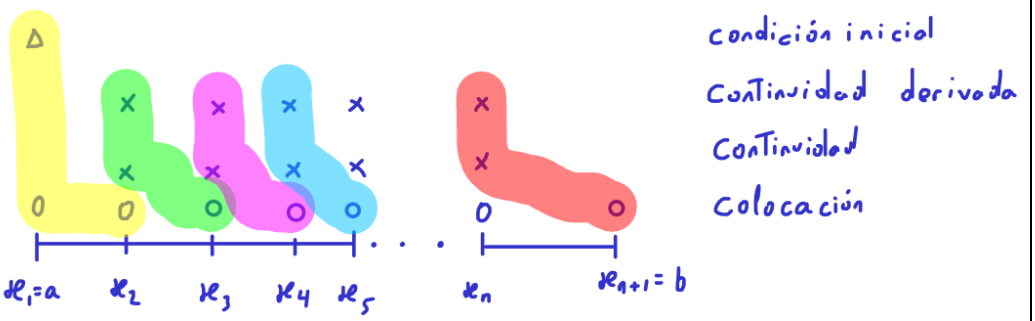


Figura 3-2 Condiciones de los Spline caso básico

En cada intervalo $[x_i, x_{i+1}]$ con $k = (1,2, \dots, n)$ definimos el Spline que aproxima a la solución como:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2$$

Derivando este polinomio respecto de su variable independiente tenemos:

$$S_i'(x) = b_i + 2\ c_i(x - x_i)$$

CONDICIONES A CUMPLIR EN EL PRIMER INTERVALO

Para el intervalo inicial, la aproximación a la solución que calcularemos tiene la forma:

$$\begin{cases} S_1(x) = a_1 + b_1(x - x_1) + c_1(x - x_1)^2 \\ S_1'(x) = b_1 + 2\,c_1(x - x_1) \\ S_1(a) = z_1 \end{cases}$$

Con base a esta expresión, vamos imponiendo las condiciones fijadas por el problema.

Condición inicial

El valor de la función en el nodo inicial es z_1

$$y(x_1) = z_1$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo inicial.

$$a_1 + b_1(x_1 - x_1) + c_1(x_1 - x_1)^2 = z_1$$

$$\boldsymbol{a_1 = z_1}$$

Condición de colocación en el nodo inicial del intervalo.

Si la función $y(x)$ es solución de la ecuación diferencial, en el nodo x_1, se cumple:

$$y'(x_1) = q(x_1) \cdot y(x_1) + r(x_1)$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo inicial.

$$b_1 + 2\,c_1(x_1 - x_1) = q(x_1) \cdot [a_1 + b_1(x_1 - x_1) + c_1(x_1 - x_1)^2] + r(x_1)$$

$$\boldsymbol{b_1 = q(x_1) \cdot a_1 + r(x_1)}$$

Condición de colocación en el nodo final del subintervalo (x_2)

Si la función $y(x)$ es solución de la ecuación diferencial, en el nodo x_1, se cumple:

$$y'(x_2) = q(x_2) \cdot y(x_2) + r(x_2)$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo inicial.

$$b_1 + 2\,c_1(x_2 - x_1) = q(x_2) \cdot [a_1 + b_1(x_2 - x_1) + c_1(x_2 - x_1)^2] + r(x_2)$$

De esta expresión todos los valores son conocidos a excepción del coeficiente c_1. Se puede calcular en base al resto de valores.

$$\boldsymbol{c_1 = \frac{q(x_2) \cdot [a_1 + b_1(x_2 - x_1)] + r(x_2) - b_1}{2\,(x_2 - x_1) - q(x_2)(x_2 - x_1)^2}}$$

Con el cálculo de estos coeficientes ya tenemos definido el polinomio en el primer subintervalo $[x_1, x_2]$.

$$S_1(x) = a_1 + b_1(x - x_1) + c_1(x - x_1)^2 \qquad x \in [x_0, x_1]$$

CONDICIONES A CUMPLIR EN EL RESTO DE INTERVALOS

Una vez que tenemos calculados los coeficientes del polinomio en el primer subintervalo, las condiciones para el resto de subintervalos se obtienen a partir de las exigencias de continuidad con el polinomio anterior en el nodo x_i y de la condición de colocación en el nodo x_{i+1}.

Condición de continuidad de la función.

Se tiene que cumplir que $S_{i-1}(x_i) = S_i(x_i)$, es decir, tanto el polinomio S_{i-1}, como el polinomio S_i, evaluados en su nodo de unión deben tomar el mismo valor.

Escribiendo los polinomios en base a sus dominios de definición:

$$a_{i-1} + b_{i-1}(x_i - x_{i-1}) + c_{i-1}(x_i - x_{i-1})^2 = a_i + b_i(x_i - x_i) + c_i(x_i - x_i)^2$$

Simplificando, obtenemos de forma directa el coeficiente a_i

$$\boldsymbol{a_i = a_{i-1} + b_{i-1}(x_i - x_{i-1}) + c_{i-1}(x_i - x_{i-1})^2}$$

Condición de continuidad de derivada primera.

Se tiene que cumplir que $S'_{i-1}(x_i) = S'_i(x_i)$, es decir, tanto la derivada del polinomio S_{i-1}, como la del polinomio S_i, evaluadas en su nodo de unión deben tomar el mismo valor.

Escribiendo las derivadas de los polinomios en base a sus dominios de definición:

$$b_{i-1} + 2\,c_{i-1}(x_i - x_{i-1}) = b_i + 2\,c_i(x_i - x_i)$$

Simplificando obtenemos de forma directa el coeficiente b_i

$$\boldsymbol{b_i = b_{i-1} + 2\,c_{i-1}(x_i - x_{i-1})}$$

Condición de colocación en el nodo final del subintervalo (x_{i+1})

Si la función $y(x)$ es solución de la ecuación diferencial, en el nodo x_{i+1}, se cumple:

$$y'(x_{i+1}) = q(x_{i+1}) \cdot y(x_{i+1}) + r(x_{i+1})$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo $(i+1)$.

$$b_i + 2\,c_i(x_{i+1} - x_i) = q(x_{i+1}) \cdot [a_i + b_i(x_{i+1} - x_i) + c_i(x_{i+1} - x_i)^2] + r(x_{i+1})$$

De esta expresión todos los valores son conocidos a excepción del coeficiente c_i. Se puede calcular en base al resto de valores.

$$\boldsymbol{c_i = \frac{q(x_{i+1}) \cdot [a_i + b_i(x_{i+1} - x_i)] + r(x_{i+1}) - b_i}{2\,(x_{i+1} - x_i) - q(x_{i+1})(x_{i+1} - x_i)^2}}$$

Con el cálculo de estos coeficientes, se tiene perfectamente definido el polinomio en el subintervalo $k = [x_k, x_{k+1}],\ k = 2, \dots, n.$

$$S_k(x) = a_k + b_k(x - x_k) + c_k(x - x_k)^2 \qquad x \in [x_k, x_{k+1}]$$

Usando de forma recursiva estas expresiones para los $n-1$ subintervalos que van desde el intervalo 2 hasta el intervalo n, obtenemos la función Spline que aproxima a la función solución.

La implementación de este proceso se puede ver en el anexo 2, apartado 3.

3.4. PLANTEAMIENTO DEL MÉTODO INVERSO

Ahora, vamos a tener un caso en el que el valor conocido se encuentra al final del intervalo de interés. Estos problemas se abordan desde un punto de vista matemático, pero desde el punto de vista de la ingeniería son menos interesantes. No obstante, como puede darse el caso de tener un problema con esta tipología se incluye en este texto.

Destacar, además que mediante la unión de este método y el del apartado 3.2, se puede partir de una condición inicial conocida y recorrer la variable independiente en ambos sentidos.

Tabla 3-2 Resumen condiciones Spline grado 2 inverso

Condiciones que imponemos en cada intervalo	
Intervalo inicial	Intervalo $i,\ i = 1, 3, \dots, n$
• Condición inicial • Condición de colocación en el nodo x_n • Condición de colocación en el nodo x_{n+1}	• Condición de continuidad de la función en x_{i+1}. • Condición de continuidad de la derivada primera en el nodo x_{i+1} • Condición de colocación en el nodo x_i

En la siguiente figura se puede ver de forma gráfica las condiciones que se deben cumplir en cada subintervalo y las diferencias entre el cálculo de los coeficientes del polinomio asociado al primer subintervalo y al resto.

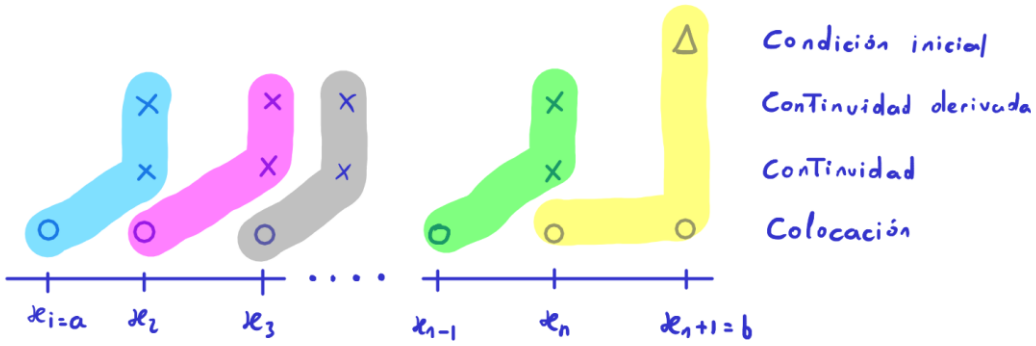


Figura 3-3 Planteamiento inverso del método de colocación

En cada intervalo $[x_i, x_{i+1}]$ con $i = (1,2, \dots, n)$ definimos el Spline que aproxima a la solución como:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2$$

Derivando este polinomio respecto de su variable independiente tenemos:

$$S_i'(x) = b_i + 2\, c_i(x - x_i)$$

CONDICIONES A CUMPLIR EN EL PRIMER INTERVALO

Para el intervalo inicial, la aproximación a la solución que calcularemos tiene la forma:

$$\begin{cases} S_n(x) = a_n + b_n(x - x_n) + c_n(x - x_n)^2 \\ S_n'(x) = b_n + 2\, c_n(x - x_n) \\ S_n(b) = z_1 \end{cases}$$

Con base a esta expresión, vamos imponiendo las condiciones fijadas por el problema.

Condición inicial

El valor de la función en el nodo inicial es z_1

$$y(x_{n+1}) = z_1$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo inicial.

$$a_n + b_n(x_{n+1} - x_n) + c_n(x_{n+1} - x_n)^2 = z_1$$

$$a_n + b_n(x_{n+1} - x_n) + c_n(x_{n+1} - x_n)^2 = z_1$$

Ecuación 3-4

Condición de colocación en el nodo inicial del intervalo.

Si la función $y(x)$ es solución de la ecuación diferencial, en el nodo x_1, se cumple:

$$y'(x_n) = q(x_n) \cdot y(x_n) + r(x_n)$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo inicial.

$$b_n + 2\, c_n(x_n - x_n) = q(x_n) \cdot [a_n + b_n(x_n - x_n) + c_n(x_n - x_n)^2] + r(x_n)$$

$$\boldsymbol{b_n = q(x_n) \cdot a_n + r(x_n)}$$

Ecuación 3-5

Condición de colocación en el nodo final del subintervalo

Si la función $y(x)$ es solución de la ecuación diferencial, en el nodo x_1, se cumple:

$$y'(x_{n+1}) = q(x_{n+1}) \cdot y(x_{n+1}) + r(x_{n+1})$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo final.

$$b_n + 2\,c_n(x_{n+1} - x_n) = q(x_{n+1}) \cdot [a_n + b_n(x_{n+1} - x_n) + c_n(x_{n+1} - x_n)^2] + r(x_n)$$

De esta expresión todos los valores son conocidos a excepción del coeficiente c_1. Se puede calcular en base al resto de valores.

$$b_n + 2\,c_n(x_{n+1} - x_n) = q(x_{n+1}) \cdot [a_n + b_n(x_{n+1} - x_n) + c_n(x_{n+1} - x_n)^2] + r(x_n)$$

Ecuación 3-6

Las ecuaciones [3 4] , [3 5] y [3 6] forman un sistema cuya resolución nos da el cálculo de los coeficientes del Spline.

Con el cálculo de estos coeficientes ya tenemos definido el polinomio en el primer subintervalo $[x_n, x_{n+1}]$.

$$S_n(x) = a_n + b_n(x - x_n) + c_n(x - x_n)^2 \qquad x \in [x_n, x_{n+1}]$$

CONDICIONES A CUMPLIR EN EL RESTO DE INTERVALOS

Una vez que tenemos calculados los coeficientes del polinomio en el primer subintervalo, las condiciones para el resto de subintervalos se obtienen a partir de las exigencias de continuidad con el polinomio posterior en el nodo x_{i+1} y de la condición de colocación en el nodo x_i.

$$\begin{cases} S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 \\ S_i'(x) = b_i + 2\,c_i(x - x_i) \\ S_i(x_{i+1}) = a_{i+1} \\ S'_i(x_{i+1}) = b_{i+1} \end{cases}$$

Ecuación 3-7 Condiciones de continuidad y colocación del problema inverso

Condición de continuidad de la función.

Se tiene que cumplir que $S_i(x_{i+1}) = S_{i+1}(x_{i+1})$, es decir, tanto el polinomio S_i, como el polinomio S_{i+1}, evaluados en su nodo de unión deben tomar el mismo valor.

Escribiendo los polinomios en base a sus dominios de definición:

$$a_i + b_i(x_{i+1} - x_i) + c_{i-1}(x_{i+1} - x_i)^2 = a_{i+1} + b_{i+1}(x_{i+1} - x_{i+1}) +$$
$$\dots c_{i+1}(x_{i+1} - x_{i+1})^2$$

Simplificando, obtenemos de la primera ecuación.

$$a_i + b_i(x_{i+1} - x_i) + c_{i-1}(x_{i+1} - x_i)^2 = a_{i+1}$$

Ecuación 3-8

Condición de continuidad de derivada primera.

Se tiene que cumplir que $S'_i(x_{i+1}) = S'_{i+1}(x_{i+1})$, es decir, tanto la derivada del polinomio S_{i+1}, como la del polinomio S_i, evaluadas en su nodo de unión deben tomar el mismo valor.

Escribiendo las derivadas de los polinomios en base a sus dominios de definición:

$$b_i + 2\,c_i(x_{i+1} - x_i) = b_{i+1} + 2\,c_{i+1}(x_{i+1} - x_{i+1})$$

Obteniendo de este modo la segunda ecuación

$$b_i + 2\,c_i(x_{i+1} - x_i) = b_{i+1}$$

Ecuación 3-9

Condición de colocación en el nodo inicial del subintervalo (x_i)

Si la función $y(x)$ es solución de la ecuación diferencial, en el nodo x_i, se cumple:

$$y'(x_i) = q(x_i) \cdot y(x_i) + r(x_i)$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo (i).

$$b_i + 2\,c_i(x_i - x_i) = q(x_i) \cdot [a_i + b_i(x_i - x_i) + c_i(x_i - x_i)^2] + r(x_i)$$

Quedando, por tanto:

$$b_i = q(x_i) \cdot a_i + r(x_i)$$

Ecuación 3-10

Resolviendo el sistema formado por las ecuaciones 3 8, 3 9 y 3 10, obtenemos el cálculo de los coeficientes del Spline, el cual ya queda perfectamente definido en el subintervalo $i = [x_i, x_{i+1}],\ k = i, \dots, n-1$.

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 \qquad x \in [x_i, x_{i+1}]$$

Usando de forma recursiva estas expresiones para los $n-1$ subintervalos que van desde el intervalo 1 hasta el intervalo $n-1$, obtenemos la expresión de la función Spline que aproxima a la solución.

$$S(x) = \begin{cases} S_1(x) = a_1 + b_1(x - x_1) + c_1(x - x_1)^2 & x \in [x_1, x_2) \\ S_2(x) = a_2 + b_2(x - x_2) + c_2(x - x_2)^2 & x \in [x_2, x_3) \\ \dots \\ S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 & x \in [x_i, x_{i+1}) \\ \dots \\ S_n(x) = a_n + b_n(x - x_n) + c_n(x - x_n)^2 & x \in [x_n, x_{n+1}] \end{cases}$$

Ecuación 3-11 Spline grado 2

3.5. MÉTODO CON PSEUDO-SPLINES DE ORDEN 3

Esta es una mejora que viene de un planteamiento lógico. En el primer caso, el Spline Satisface la ecuación diferencial en los extremos del intervalo, pero no tenemos más información de cómo se comporta dentro del intervalo. Con esta mejora le vamos a imponer que satisfaga también la ecuación diferencial en el nodo intermedio del intervalo.

Esto nos va a proporcionar una solución más exacta, ya que impone más condiciones de colocación.

De igual manera que en el apartado anterior tenemos unas condiciones que se deben cumplir en el primer subintervalo, y una vez satisfechas, las condiciones del resto de intervalos se obtienen de una manera recursiva.

$$S(x) = \begin{cases} S_1(x) = a_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3 & x \in [x_1, x_2) \\ S_2(x) = a_2 + b_2(x - x_2) + c_2(x - x_2)^2 + d_2(x - x_2)^3 & x \in [x_2, x_3) \\ \dots \\ S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 & x \in [x_i, x_{i+1}) \\ \dots \\ S_n(x) = a_n + b_n(x - x_n) + c_n(x - x_n)^2 + d_n(x - x_n)^3 & x \in [x_n, x_{n+1}] \end{cases}$$

Ecuación 3-12 Definición de la función Spline de grado 3 y clase C1

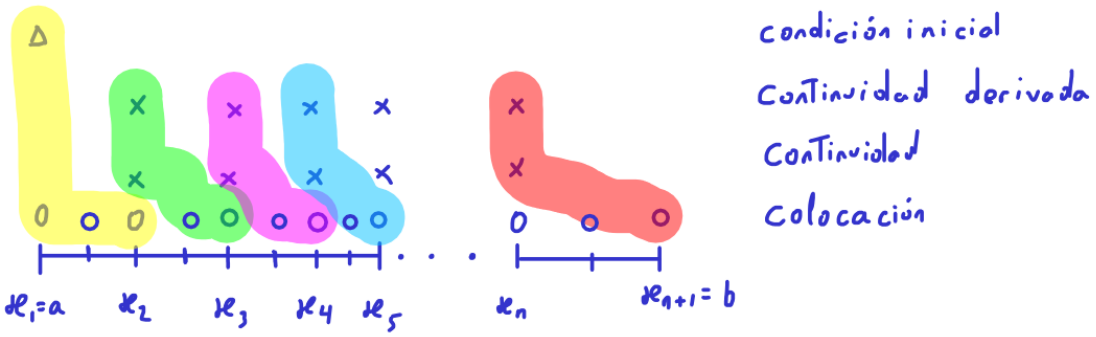


Figura 3-4 Condiciones colocación Pseudo-Splines

Tabla 3-3 Condiciones de colocación Pseudo-Splines

Condiciones que imponemos en cada intervalo	
Intervalo inicial	Intervalo $i,\ i = 2, 3, \dots, n$
• Condición inicial • Condición de colocación en el nodo x_1 • Condición de colocación en el nodo $\frac{x_1+x_2}{2}$ • Condición de colocación en el nodo x_2	• Condición de continuidad de la función. • Condición de continuidad de la derivada primera en el nodo x_i • Condición de colocación en el nodo $\frac{x_i+x_{i+1}}{2}$ • Condición de colocación en el nodo x_{i+1}

En cada intervalo $[x_i, x_{i+1}]$ con $k = (1,2, \dots, n)$ definimos el Pseudo-Spline que aproxima a la solución como:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

Derivando este polinomio respecto de su variable independiente tenemos:

$$S_i'(x) = b_i + 2\ c_i(x - x_i) + 3\ d_i(x - x_i)^2$$

CONDICIONES A CUMPLIR EN EL PRIMER INTERVALO

Para el intervalo inicial tenemos:

$$\begin{cases} S_1(x) = a_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3 \\ S_1'(x) = b_1 + 2\ c_1(x - x_1) + 3\ d_1(x - x_1)^2 \\ S_1(a) = z_1 \end{cases}$$

De donde tenemos que:

Condición inicial

El valor de la función en el nodo inicial es z_1

$$y(x_1) = z_1$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo inicial.

$$a_1 + b_1(x_1 - x_1) + c_1(x_1 - x_1)^2 + d_1(x_1 - x_1)^3 = z_1$$

$$\boldsymbol{a_1 = z_1}$$

Ecuación 3-13

Condición de colocación en el nodo inicial del intervalo.

Si la función $y(x)$ es solución de la ecuación diferencial, en el nodo x_1, se cumple:

$$y'(x_1) = q(x_1) \cdot y(x_1) + r(x_1)$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo inicial.

$$b_1 + 2\,c_1(x_1 - x_1) + 3\,d_1(x_1 - x_1)^2 = q(x_1) \cdot [a_1 + b_1(x_1 - x_1) + c_1(x_1 - x_1)^2 + d_1(x_1 - x_1)^3] + r(x_1)$$

Simplificando:

$$\boldsymbol{b_1 = q(x_1) \cdot a_1 + r(x_1)}$$

Ecuación 3-14

Condición de colocación en el nodo intermedio del intervalo

Si la función $y(x)$ es solución de la ecuación diferencial, en el punto $\frac{x_1+x_2}{2}$, se cumple:

$$y'\left(\frac{x_1 + x_2}{2}\right) = q\left(\frac{x_1 + x_2}{2}\right) \cdot y\left(\frac{x_1 + x_2}{2}\right) + r\left(\frac{x_1 + x_2}{2}\right)$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el punto intermedio.

$$b_1 + 2\,c_1 \cdot \left(\frac{h_1}{2}\right) + 3\,d_1\left(\frac{h_1}{2}\right)^2 = q_{1+\frac{1}{2}} \cdot \left[a_1 + b_1 \cdot \left(\frac{h_1}{2}\right) + c_1 \cdot \left(\frac{h_1}{2}\right)^2 + d_1 \cdot \left(\frac{h_1}{2}\right)^3\right] + r_{1+\frac{1}{2}}$$

De aquí se obtiene:

$$\boldsymbol{\alpha_1 \cdot c_1 + \alpha_2 \cdot d_1 = \beta_1}$$

Ecuación 3-15

Donde

$$\alpha_1 = 2\left(\frac{h_1}{2}\right) - q_{1+\frac{1}{2}} \cdot \left(\frac{h_1}{2}\right)^2$$

$$\alpha_2 = 3 \cdot \left(\frac{h_1}{2}\right)^2 - q_{1+\frac{1}{2}} \cdot \left(\frac{h_1}{2}\right)^3$$

$$\beta_1 = q_{1+\frac{1}{2}} \cdot \left[a_1 + b_1 \cdot \left(\frac{h_1}{2}\right)\right] + r_{1+\frac{1}{2}} - b_1$$

Condición de colocación en el nodo final del intervalo.

Si la función $y(x)$ es solución de la ecuación diferencial, en el nodo x_2, se cumple:

$$y'(x_2) = q(x_2) \cdot y(x_2) + r(x_2)$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo final.

$$b_1 + 2\,c_1(h_1) + 3\,d_1(h_1)^2 = q(x_2) \cdot [a_1 + b_1(h_1) + c_1(h_1)^2 + d_1(h_1)^3] + r(x_2)$$

De aquí se obtiene:

$$\boldsymbol{\alpha_3 \cdot c_1 + \alpha_4 \cdot d_1 = \beta_1}$$

Ecuación 3-16

Donde

$\alpha_3 = 2(h_1) - q(x_2) \cdot (h_1)^2$

$\alpha_4 = 3 \cdot (h_1)^2 - q(x_2) \cdot (h_1)^3$

$\beta_2 = q(x_2) \cdot [a_1 + b_1(h_1)] + r(x_2) - b_1$

Tanto la ecuación 3 13 como la ecuación 3 14 tienen una resolución directa. Sin embargo, las ecuaciones 3 15 y 3 16 forman un sistema de ecuaciones cuya resolución nos proporciona los valores de los coeficientes c_1 y d_1

$$\begin{bmatrix} \alpha_1 & \alpha_2 \\ \alpha_3 & \alpha_4 \end{bmatrix} \cdot \begin{bmatrix} c_i \\ d_i \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}$$

Una vez resuelto, ya tendremos la definición del Spline en el primer intervalo.

CONDICIONES A CUMPLIR EN EL RESTO DE INTERVALOS

La expresión del Spline en el subintervalo i y de su derivada es de la forma:

$$\begin{cases} S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \\ S_i'(x) = b_i + 2\,c_i(x - x_i) + 3\,d_i(x - x_i)^2 \end{cases}$$

Condición de continuidad de la función

Se tiene que cumplir que $S_{i-1}(x_i) = S_i(x_i)$, es decir, tanto el polinomio S_{i-1}, como el polinomio S_i, evaluados en su nodo de unión deben tomar el mismo valor.

Escribiendo los polinomios en base a sus dominios de definición:

$$\begin{aligned} a_{i-1} + b_{i-1}(x_i - x_{i-1}) + c_{i-1}(x_i - x_{i-1})^2 + d_{i-1}(x_i - x_{i-1})^3 \\ = a_i + b_i(x_i - x_i) + c_i(x_i - x_i)^2 + d_i(x_i - x_i)^3 \end{aligned}$$

Simplificando, obtenemos de forma directa el coeficiente a_i

$$\boldsymbol{a_i = a_{i-1} + b_{i-1}(h_{i-1}) + c_{i-1}(h_{i-1})^2 + d_{i-1}(h_{i-1})^3}$$

Ecuación 3-17

Condición de continuidad de derivada primera

Se tiene que cumplir que $S'_{i-1}(x_i) = S'_i(x_i)$, es decir, tanto la derivada del polinomio S_{i-1}, como la del polinomio S_i, evaluadas en su nodo de unión deben tomar el mismo valor.

Escribiendo las derivadas de los polinomios en base a sus dominios de definición:

$$\begin{aligned} b_{i-1} + 2\,c_{i-1}(x_i - x_{i-1}) + 3\,d_{i-1}(x_i - x_{i-1})^2 \\ = b_i + 2\,c_i(x_i - x_i) + 3\,d_i(x_i - x_i)^2 \end{aligned}$$

De donde

$$b_i = b_{i-1} + 2\,c_{i-1}(h_{i-1}) + 3\,d_{i-1}(h_{i-1})^2$$

Ecuación 3-18

Condición de colocación en el punto intermedio

Si la función $y(x)$ es solución de la ecuación diferencial, en el punto $x_{i+\frac{1}{2}}$, se cumple:

$$y'_{i+\frac{1}{2}} = q_{i+\frac{1}{2}} \cdot y_{i+\frac{1}{2}} + r_{i+\frac{1}{2}}$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el punto intermedio.

$$b_i + 2\,c_i\left(\frac{h_i}{2}\right) + 3\,d_i\left(\frac{h_i}{2}\right)^2 = q_{i+\frac{1}{2}} \cdot \left[a_i + b_i\left(\frac{h_i}{2}\right) + c_i\left(\frac{h_i}{2}\right)^2 + d_i\left(\frac{h_i}{2}\right)^3\right] + r_{i+\frac{1}{2}}$$

De aquí se obtiene la siguiente ecuación

$$\alpha_{i1} \cdot c_i + \alpha_{i2} \cdot d_i = \beta_{i1}$$

Ecuación 3-19

Donde

$$\alpha_{i1} = 2 \cdot \left(\frac{h_i}{2}\right) - q_{i+\frac{1}{2}} \cdot \left(\frac{h_i}{2}\right)^2$$

$$\alpha_{i2} = 3 \cdot \left(\frac{h_i}{2}\right)^2 - q_{i+\frac{1}{2}} \cdot \left(\frac{h_i}{2}\right)^3$$

$$\beta_{i1} = q_{i+\frac{1}{2}} \cdot \left[a_i + b_i\left(\frac{h_i}{2}\right)\right] + r_{i+\frac{1}{2}} - b_i$$

Condición de colocación en el punto final

Si la función $y(x)$ es solución de la ecuación diferencial, en el nodo x_{i+1}, se cumple:

$$y'_{i+1} = q_{i+1} \cdot y_{i+1} + r_{i+1}$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el punto intermedio.

$$b_i + 2\,c_i h_i + 3\,d_i {h_i}^2 = q_{i+1} \cdot \left[a_i + b_i h_i + c_i {h_i}^2 + d_i {h_i}^3\right] + r_{i+1}$$

De aquí se obtiene la siguiente ecuación

$$\boldsymbol{\alpha_{i3} \cdot c_i + \alpha_{i4} \cdot d_i = \beta_{i2}}$$

Ecuación 3-20

Donde

$\alpha_{i3} = 2 \cdot h_i - q_{i+1} \cdot {h_i}^2$

$\alpha_{i4} = 3 \cdot {h_i}^2 - q_{i+1} \cdot {h_i}^3$

$\beta_{i2} = q_{i+1} \cdot [a_i + b_i \cdot h_i] + r_{i+1} - b_i$

Estas dos últimas igualdades forman un sistema de ecuaciones cuya resolución nos proporciona los valores de los coeficientes c_i y d_i.

3.6. MÉTODO CON SPLINES DE GRADO 3

Este método también usa Splines de grado 3. Sin embargo, a diferencia del anterior, ese grado extra lo vamos a usar para imponer más regularidad en la función, de tal manera que sea continua y derivable dos veces.

$$S(x) = \begin{cases} S_1(x) = a_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3 & x \in [x_1, x_2) \\ S_2(x) = a_2 + b_2(x - x_2) + c_2(x - x_2)^2 + d_2(x - x_2)^3 & x \in [x_2, x_3) \\ \dots \\ S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 & x \in [x_i, x_{i+1}) \\ \dots \\ S_n(x) = a_n + b_n(x - x_n) + c_n(x - x_n)^2 + d_n(x - x_n)^3 & x \in [x_n, x_{n+1}] \end{cases}$$

Ecuación 3-21 Spline de grado 3

Tabla 3-4 Resumen condiciones Pseudo-Spline de grado 3

Condiciones que imponemos en cada intervalo	
Intervalo inicial	Intervalo $i,\ i = 2, 3, \dots, n$
• Condición inicial • Condición de colocación en el nodo x_1 • Condición de colocación en el nodo $\frac{x_1+x_2}{2}$ • Condición de colocación en el nodo x_2	• Condición de continuidad de la función. • Condición de continuidad de la derivada primera en el nodo x_i • Condición de continuidad de la derivada segunda en el nodo x_i • Condición de colocación en el nodo x_{i+1}

En cada intervalo $[x_i, x_{i+1}]$ con $i = (1,2, \dots, n)$ definimos el Pseudo-Spline que aproxima a la solución como:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

Derivando este polinomio respecto de su variable independiente tenemos:

$$S_i'(x) = b_i + 2\, c_i(x - x_i) + 3\, d_i(x - x_i)^2$$

CONDICIONES A CUMPLIR EN EL PRIMER INTERVALO

En este primer intervalo necesitamos el cálculo de 4 coeficientes. Para obtener el cuarto se ha optado por la estrategia de imponer un nodo más en el punto medio del intervalo, en el que se debe satisfacer la condición de colocación.

Para el intervalo inicial tenemos:

$$\begin{cases} S_1(x) = a_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3 \\ S_1'(x) = b_1 + 2\,c_1(x - x_1) + 3\,d_1(x - x_1)^2 \\ S_1(a) = z_1 \end{cases}$$

De donde tenemos que:

Condición inicial

El valor de la función en el nodo inicial es z_1

$$y(x_1) = z_1$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo inicial.

$$a_1 + b_1(x_1 - x_1) + c_1(x_1 - x_1)^2 + d_1(x_1 - x_1)^3 = z_1$$

$$\boldsymbol{a_1 = z_1}$$

Ecuación 3-22

Condición de colocación en el nodo inicial del intervalo

Si la función $y(x)$ es solución de la ecuación diferencial, en el nodo x_1, se cumple:

$$y'(x_1) = q(x_1) \cdot y(x_1) + r(x_1)$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo inicial.

$$\begin{aligned} b_1 + 2\,c_1(x_1 - x_1) + 3\,d_1(x_1 - x_1)^2 \\ = q(x_1) \cdot [a_1 + b_1(x_1 - x_1) + c_1(x_1 - x_1)^2 + d_1(x_1 - x_1)^3] \\ + r(x_1) \end{aligned}$$

$$\boldsymbol{b_1 = q(x_1) \cdot a_1 + r(x_1)}$$

Ecuación 3-23

Condición de colocación en el nodo intermedio del intervalo

Si la función $y(x)$ es solución de la ecuación diferencial, en el punto $\frac{x_1+x_2}{2}$, se cumple:

$$y'\left(\frac{x_1+x_2}{2}\right) = q\left(\frac{x_1+x_2}{2}\right)\cdot y\left(\frac{x_1+x_2}{2}\right) + r\left(\frac{x_1+x_2}{2}\right)$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el punto intermedio.

$$b_1 + 2\,c_1\cdot\left(\frac{h_1}{2}\right) + 3\,d_1\left(\frac{h_1}{2}\right)^2 = q_{1+\frac{1}{2}}\cdot\left[a_1 + b_1\cdot\left(\frac{h_1}{2}\right) + c_1\cdot\left(\frac{h_1}{2}\right)^2 + d_1\cdot\left(\frac{h_1}{2}\right)^3\right] + r_{1+\frac{1}{2}}$$

De aquí se obtiene:

$$\alpha_1\cdot c_1 + \alpha_2\cdot d_1 = \beta_1$$

Ecuación 3-24

Donde

$$\alpha_1 = 2\left(\frac{h_1}{2}\right) - q_{1+\frac{1}{2}}\cdot\left(\frac{h_1}{2}\right)^2$$

$$\alpha_2 = 3\cdot\left(\frac{h_1}{2}\right)^2 - q_{1+\frac{1}{2}}\cdot\left(\frac{h_1}{2}\right)^3$$

$$\beta_1 = q_{1+\frac{1}{2}}\cdot\left[a_1 + b_1\cdot\left(\frac{h_1}{2}\right)\right] + r_{1+\frac{1}{2}} - b_1$$

Condición de colocación en el nodo final del intervalo

Si la función $y(x)$ es solución de la ecuación diferencial, en el nodo x_2, se cumple:

$$y'(x_2) = q(x_2)\cdot y(x_2) + r(x_2)$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo final.

$$b_1 + 2\,c_1(h_1) + 3\,d_1(h_1)^2 = q(x_2)\cdot[a_1 + b_1(h_1) + c_1(h_1)^2 + d_1(h_1)^3] + r(x_2)$$

De aquí se obtiene:

$$\boldsymbol{\alpha_3 \cdot c_1 + \alpha_4 \cdot d_1 = \beta_1}$$

Ecuación 3-25

Donde

$\alpha_3 = 2(h_1) - q(x_2) \cdot (h_1)^2$

$\alpha_4 = 3 \cdot (h_1)^2 - q(x_2) \cdot (h_1)^3$

$\beta_2 = q(x_2) \cdot [a_1 + b_1(h_1)] + r(x_2) - b_1$

Estas dos últimas igualdades, Ecuación 3 24 y Ecuación 3 25, forman un sistema de ecuaciones cuya resolución nos proporciona los valores de los coeficientes c_1 y d_1

$$\begin{bmatrix} \alpha_1 & \alpha_2 \\ \alpha_3 & \alpha_4 \end{bmatrix} \cdot \begin{bmatrix} c_i \\ d_i \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}$$

Ecuación 3-26

CONDICIONES A CUMPLIR EN EL RESTO DE INTERVALOS

Condición de continuidad de la función

Se tiene que cumplir que $S_{i-1}(x_i) = S_i(x_i)$, es decir, tanto el polinomio S_{i-1}, como el polinomio S_i, evaluados en su nodo de unión deben tomar el mismo valor.

Escribiendo los polinomios en base a sus dominios de definición:

$$a_{i-1} + b_{i-1}(h_{i-1}) + c_{i-1}(h_{i-1})^2 + d_{i-1}(h_{i-1})^3 = a_i + b_i(x_i - x_i) + c_i(x_i - x_i)^2 + d_i(x_i - x_i)^3$$

Simplificando, obtenemos de forma directa el coeficiente a_i

$$\boldsymbol{a_i = a_{i-1} + b_{i-1}(h_{i-1}) + c_{i-1}(h_{i-1})^2 + d_{i-1}(h_{i-1})^3}$$

Ecuación 3-27

Condición de continuidad de derivada primera.

Se tiene que cumplir que $S'_{i-1}(x_i) = S'_i(x_i)$, es decir, tanto la derivada del polinomio S_{i-1}, como la del polinomio S_i, evaluadas en su nodo de unión deben tomar el mismo valor.

Escribiendo las derivadas de los polinomios en base a sus dominios de definición:

$$b_{i-1} + 2\,c_{i-1}(h_{i-1}) + 3\,d_{i-1}(h_{i-1})^2 = b_i + 2\,c_i(x_i - x_i) + 3\,d_i(x_i - x_i)^2$$

De donde

$$\boldsymbol{b_i = b_{i-1} + 2\,c_{i-1}(h_{i-1}) + 3\,d_{i-1}(h_{i-1})^2}$$

Ecuación 3-28

Condición de continuidad de derivada segunda

Se tiene que cumplir que $S''_{i-1}(x_i) = S''_i(x_i)$, es decir, tanto la derivada segunda del polinomio S_{i-1}, como la del polinomio S_i, evaluadas en su nodo de unión deben tomar el mismo valor.

Escribiendo las derivadas segundas de los polinomios en base a sus dominios de definición:

$$2\,c_{i-1} + 6\,d_{i-1}(h_{i-1}) = 2\,c_i + 6\,d_i(x_i - x_i)^2$$

De donde

$$\boldsymbol{c_i = \frac{2\,c_{i-1} + 6\,d_{i-1}(h_{i-1})}{2}}$$

Ecuación 3-29

Condición de colocación en el nodo final del intervalo

Si la función $y(x)$ es solución de la ecuación diferencial, en el nodo x_{i+1}, se cumple:

$$y'_{i+1} = q_{i+1} \cdot y_{i+1} + r_{i+1}$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo final.

$$b_i + 2\,c_i \cdot h_i + 3\,d_i h_i^2 = q_{i+1} \cdot [a_i + b_i \cdot h_i + c_i h_i^2 + d_i h_i^3] + r_{i+1}$$

Despejando el coeficiente d_i tenemos

$$d_i = \frac{q_{i+1} \cdot \left(a_i + b_i \cdot h_i + c_i \cdot h_i^2\right) + r_{i+1} - b - 2 \cdot c \cdot h_i}{3 \cdot h_i^2 - q_{i+1} \cdot h_i^3}$$

Ecuación 3-30

Estas ecuaciones tienen una resolución directa y no precisan de un sistema para su resolución.

3.7. PLANTEAMIENTO MÉTODO DE COLOCACIÓN BÁSICO (SPLINE GRADO 2) NO LINEAL

Vamos a considerar un problema de valores iniciales en el intervalo $[a, b]$.

$$\begin{cases} y'(x) = q(x) \cdot y(x) + r(x) \\ y(a) = z_1 \end{cases}$$

Ecuación 3-31 Problema de Cauchy

Con el objeto de encontrar una aproximación a la función $y(x)$, hacemos una partición del intervalo $[a, b]$ en n subintervalos, generando entonces los nodos x_i en los que impondremos que se satisfaga la ecuación diferencial:

$$a = x_1 < x_2 < \cdots < x_n < x_{n+1} = b$$

Consideramos, entonces, el siguiente Spline cuadrático, basado en los nodos $\{x_i\}_{i=1}^{n+1}$ definido del siguiente modo:

$$S(x) = \begin{cases} S_1(x) = a_1 + b_1(x - x_1) + c_1(x - x_1)^2 & x \in [x_1, x_2) \\ S_2(x) = a_2 + b_2(x - x_2) + c_2(x - x_2)^2 & x \in [x_2, x_3) \\ \cdots \\ S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 & x \in [x_i, x_{i+1}) \\ \cdots \\ S_n(x) = a_n + b_n(x - x_n) + c_n(x - x_n)^2 & x \in [x_n, x_{n+1}] \end{cases}$$

Ecuación 3-32 Spline de grado 2

El procedimiento para el cálculo de los coeficientes parte de las condiciones que imponemos en cada subintervalo.

En primer lugar, como se puede ver en la figura 3 2, tenemos unas condiciones que se deben cumplir en el primer subintervalo. Este es el intervalo de arranque del método. Una vez satisfechas estas condiciones y generado el primer Spline, el cálculo de los coeficientes para los subintervalos posteriores se hace de manera recursiva, imponiendo las condiciones de continuidad y de colocación, de la manera que se detalla.

Tabla 3-5 Condiciones a imponer en el Spline

Condiciones que imponemos en cada intervalo	
Intervalo inicial	Intervalo $i,\ i = 2, 3, \dots, n$
• Condición inicial • Condición de colocación en el nodo x_1 • Condición de colocación en el nodo x_2	• Condición de continuidad de la función. • Condición de continuidad de la derivada primera en el nodo x_i • Condición de colocación en el nodo x_{i+1}

En la siguiente figura se puede ver de forma gráfica las condiciones que se deben cumplir en cada subintervalo y las diferencias entre el cálculo de los coeficientes del polinomio asociado al primer subintervalo y al resto.

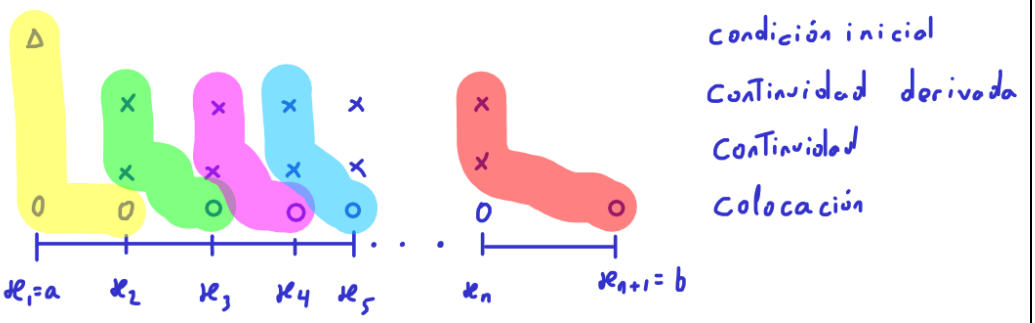


Figura 3-5 Condiciones de los Spline caso básico

En cada intervalo $[x_i, x_{i+1}]$ con $k = (1,2, \dots, n)$ definimos el Spline que aproxima a la solución como:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2$$

Derivando este polinomio respecto de su variable independiente tenemos:

$$S_i'(x) = b_i + 2\ c_i(x - x_i)$$

CONDICIONES A CUMPLIR EN EL PRIMER INTERVALO

Para el intervalo inicial, la aproximación a la solución que calcularemos tiene la forma:

$$\begin{cases} S_1(x) = a_1 + b_1(x - x_1) + c_1(x - x_1)^2 \\ S_1'(x) = b_1 + 2\,c_1(x - x_1) \\ S_1(a) = z_1 \end{cases}$$

Con base a esta expresión, vamos imponiendo las condiciones fijadas por el problema.

Condición inicial

El valor de la función en el nodo inicial es z_1

$$y(x_1) = z_1$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo inicial.

$$a_1 + b_1(x_1 - x_1) + c_1(x_1 - x_1)^2 = z_1$$

$$\boldsymbol{a_1 = z_1}$$

Ecuación 3-33: a

Condición de colocación en el nodo inicial del intervalo.

Si la función $y(x)$ es solución de la ecuación diferencial, en el nodo x_1, se cumple:

$$y'(x_1) = q(x_1) \cdot y(x_1) + r(x_1)$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo inicial.

$$b_1 + 2\,c_1(x_1 - x_1) = q(x_1) \cdot [a_1 + b_1(x_1 - x_1) + c_1(x_1 - x_1)^2] + r(x_1)$$

$$\boldsymbol{b_1 = q(x_1) \cdot a_1 + r(x_1)}$$

Ecuación 3-34: b

Condición de colocación en el nodo final del subintervalo (x_2)

Si la función $y(x)$ es solución de la ecuación diferencial, en el nodo x_1, se cumple:

$$y'(x_2) = q(x_2) \cdot y(x_2) + r(x_2)$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo inicial.

$$b_1 + 2\,c_1(x_2 - x_1) = q(x_2) \cdot [a_1 + b_1(x_2 - x_1) + c_1(x_2 - x_1)^2] + r(x_2)$$

De esta expresión todos los valores son conocidos a excepción del coeficiente c_1. Se puede calcular en base al resto de valores.

$$\boldsymbol{c_1 = \frac{q(x_2) \cdot [a_1 + b_1(x_2 - x_1)] + r(x_2) - b_1}{2\,(x_2 - x_1) - q(x_2)(x_2 - x_1)^2}}$$

Ecuación 3-35: c - no lineal

Esta ecuación se resuelve de manera no lineal mediante fsolve.

Con el cálculo de estos coeficientes ya tenemos definido el polinomio en el primer subintervalo $[x_1, x_2]$.

$$S_1(x) = a_1 + b_1(x - x_1) + c_1(x - x_1)^2 \qquad x \in [x_0, x_1]$$

CONDICIONES A CUMPLIR EN EL RESTO DE INTERVALOS

Una vez que tenemos calculados los coeficientes del polinomio en el primer subintervalo, las condiciones para el resto de subintervalos se obtienen a partir de las exigencias de continuidad con el polinomio anterior en el nodo x_i y de la condición de colocación en el nodo x_{i+1}.

Condición de continuidad de la función.

Se tiene que cumplir que $S_{i-1}(x_i) = S_i(x_i)$, es decir, tanto el polinomio S_{i-1}, como el polinomio S_i, evaluados en su nodo de unión deben tomar el mismo valor.

Escribiendo los polinomios en base a sus dominios de definición:

$$a_{i-1} + b_{i-1}(x_i - x_{i-1}) + c_{i-1}(x_i - x_{i-1})^2 = a_i + b_i(x_i - x_i) + c_i(x_i - x_i)^2$$

Simplificando, obtenemos de forma directa el coeficiente a_i

$$\boldsymbol{a_i = a_{i-1} + b_{i-1}(x_i - x_{i-1}) + c_{i-1}(x_i - x_{i-1})^2}$$

Ecuación 3-36: a

Condición de continuidad de derivada primera.

Se tiene que cumplir que $S'_{i-1}(x_i) = S'_i(x_i)$, es decir, tanto la derivada del polinomio S_{i-1}, como la del polinomio S_i, evaluadas en su nodo de unión deben tomar el mismo valor.

Escribiendo las derivadas de los polinomios en base a sus dominios de definición:

$$b_{i-1} + 2\,c_{i-1}(x_i - x_{i-1}) = b_i + 2\,c_i(x_i - x_i)$$

Simplificando obtenemos de forma directa el coeficiente b_i

$$\boldsymbol{b_i = b_{i-1} + 2\,c_{i-1}(x_i - x_{i-1})}$$

Ecuación 3-37: b

Condición de colocación en el nodo final del subintervalo (x_{i+1})

Si la función $y(x)$ es solución de la ecuación diferencial, en el nodo x_{i+1}, se cumple:

$$y'(x_{i+1}) = q(x_{i+1}) \cdot y(x_{i+1}) + r(x_{i+1})$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo $(i+1)$.

$$b_i + 2\,c_i(x_{i+1} - x_i) = q(x_{i+1}) \cdot [a_i + b_i(x_{i+1} - x_i) + c_i(x_{i+1} - x_i)^2] + r(x_{i+1})$$

De esta expresión todos los valores son conocidos a excepción del coeficiente c_i. Se puede calcular en base al resto de valores.

$$\boldsymbol{c_i = \frac{q(x_{i+1}) \cdot [a_i + b_i(x_{i+1} - x_i)] + r(x_{i+1}) - b_i}{2\,(x_{i+1} - x_i) - q(x_{i+1})(x_{i+1} - x_i)^2}}$$

Ecuación 3-38: c - no lineal

Esta ecuación se resuelve de manera no lineal mediante fsolve.

Con el cálculo de estos coeficientes, se tiene perfectamente definido el polinomio en el subintervalo $k = [x_k, x_{k+1}],\ k = 2, \dots, n$.

$$S_k(x) = a_k + b_k(x - x_k) + c_k(x - x_k)^2 \qquad x \in [x_k, x_{k+1}]$$

Usando de forma recursiva estas expresiones para los $n - 1$ subintervalos que van desde el intervalo 2 hasta el intervalo n, obtenemos la función Spline que aproxima a la función solución.

La implementación de este proceso se puede ver en el anexo 2, apartado 3.

3.8. MÉTODO CON PSEUDO-SPLINES DE ORDEN 3 (NO LINEAL)

Esta es una mejora que viene de un planteamiento lógico. En el primer caso, el Spline Satisface la ecuación diferencial en los extremos del intervalo, pero no tenemos más información de cómo se comporta dentro del intervalo. Con esta mejora le vamos a imponer que satisfaga también la ecuación diferencial en el nodo intermedio del intervalo.

Esto nos va a proporcionar una solución más exacta, ya que impone más condiciones de colocación.

De igual manera que en el apartado anterior tenemos unas condiciones que se deben cumplir en el primer subintervalo, y una vez satisfechas, las condiciones del resto de intervalos se obtienen de una manera recursiva.

$$S(x) = \begin{cases} S_1(x) = a_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3 & x \in [x_1, x_2) \\ S_2(x) = a_2 + b_2(x - x_2) + c_2(x - x_2)^2 + d_2(x - x_2)^3 & x \in [x_2, x_3) \\ \dots \\ S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 & x \in [x_i, x_{i+1}) \\ \dots \\ S_n(x) = a_n + b_n(x - x_n) + c_n(x - x_n)^2 + d_n(x - x_n)^3 & x \in [x_n, x_{n+1}] \end{cases}$$

Ecuación 3-39 Definición de la función Spline de grado 3 y clase C1

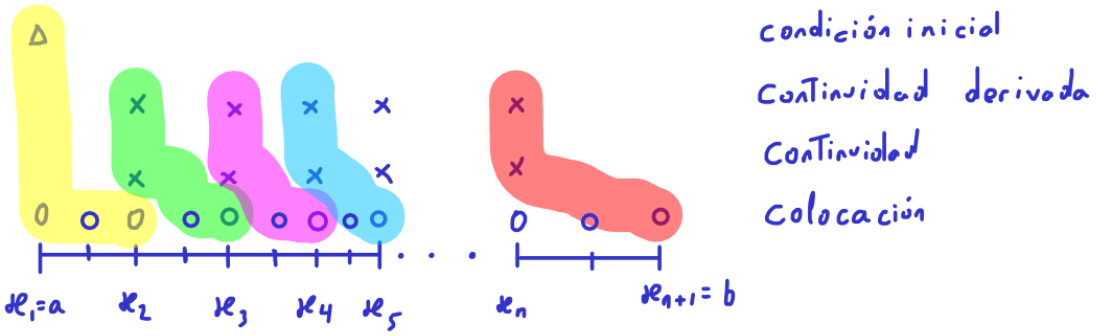


Figura 3-6 Condiciones colocación Pseudo-Splines

Tabla 3-6 Condiciones de colocación Pseudo-Splines

Condiciones que imponemos en cada intervalo	
Intervalo inicial	Intervalo $i,\ i = 2, 3, \dots, n$
• Condición inicial • Condición de colocación en el nodo x_1 • Condición de colocación en el nodo $\frac{x_1+x_2}{2}$ • Condición de colocación en el nodo x_2	• Condición de continuidad de la función. • Condición de continuidad de la derivada primera en el nodo x_i • Condición de colocación en el nodo $\frac{x_i+x_{i+1}}{2}$ • Condición de colocación en el nodo x_{i+1}

En cada intervalo $[x_i, x_{i+1}]$ con $k = (1,2, \dots, n)$ definimos el Pseudo-Spline que aproxima a la solución como:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

Derivando este polinomio respecto de su variable independiente tenemos:

$$S_i'(x) = b_i + 2\, c_i(x - x_i) + 3\, d_i(x - x_i)^2$$

CONDICIONES A CUMPLIR EN EL PRIMER INTERVALO

Para el intervalo inicial tenemos:

$$\begin{cases} S_1(x) = a_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3 \\ S_1'(x) = b_1 + 2\,c_1(x - x_1) + 3\,d_1(x - x_1)^2 \\ S_1(a) = z_1 \end{cases}$$

De donde tenemos que:

Condición inicial

El valor de la función en el nodo inicial es z_1

$$y(x_1) = z_1$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo inicial.

$$a_1 + b_1(x_1 - x_1) + c_1(x_1 - x_1)^2 + d_1(x_1 - x_1)^3 = z_1$$

$$\boldsymbol{a_1 = z_1}$$

Ecuación 3-40

Condición de colocación en el nodo inicial del intervalo.

Si la función $y(x)$ es solución de la ecuación diferencial, en el nodo x_1, se cumple:

$$y'(x_1) = q(x_1) \cdot y(x_1) + r(x_1)$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo inicial.

$$\begin{aligned} b_1 + 2\,c_1(x_1 - x_1) + 3\,d_1(x_1 - x_1)^2 \\ = q(x_1) \cdot [a_1 + b_1(x_1 - x_1) + c_1(x_1 - x_1)^2 + d_1(x_1 - x_1)^3] \\ + r(x_1) \end{aligned}$$

Simplificando:

$$\boldsymbol{b_1 = q(x_1) \cdot a_1 + r(x_1)}$$

Ecuación 3-41

Condición de colocación en el nodo intermedio del intervalo

Si la función $y(x)$ es solución de la ecuación diferencial, en el punto $\frac{x_1+x_2}{2}$, se cumple:

$$y'\left(\frac{x_1 + x_2}{2}\right) = q\left(\frac{x_1 + x_2}{2}\right) \cdot y\left(\frac{x_1 + x_2}{2}\right) + r\left(\frac{x_1 + x_2}{2}\right)$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el punto intermedio.

$$b_1 + 2\,c_1 \cdot \left(\frac{h_1}{2}\right) + 3\,d_1 \left(\frac{h_1}{2}\right)^2 = q_{1+\frac{1}{2}} \cdot \left[a_1 + b_1 \cdot \left(\frac{h_1}{2}\right) + c_1 \cdot \left(\frac{h_1}{2}\right)^2 + d_1 \cdot \left(\frac{h_1}{2}\right)^3\right] + r_{1+\frac{1}{2}}$$

De aquí se obtiene:

$$\boldsymbol{\alpha_1 \cdot c_1 + \alpha_2 \cdot d_1 = \beta_1}$$

Ecuación 3-42

Donde

$$\alpha_1 = 2\left(\frac{h_1}{2}\right) - q_{1+\frac{1}{2}} \cdot \left(\frac{h_1}{2}\right)^2$$

$$\alpha_2 = 3 \cdot \left(\frac{h_1}{2}\right)^2 - q_{1+\frac{1}{2}} \cdot \left(\frac{h_1}{2}\right)^3$$

$$\beta_1 = q_{1+\frac{1}{2}} \cdot \left[a_1 + b_1 \cdot \left(\frac{h_1}{2}\right)\right] + r_{1+\frac{1}{2}} - b_1$$

Condición de colocación en el nodo final del intervalo.

Si la función $y(x)$ es solución de la ecuación diferencial, en el nodo x_2, se cumple:

$$y'(x_2) = q(x_2) \cdot y(x_2) + r(x_2)$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo final.

$$b_1 + 2\,c_1(h_1) + 3\,d_1(h_1)^2 = q(x_2) \cdot [a_1 + b_1(h_1) + c_1(h_1)^2 + d_1(h_1)^3] + r(x_2)$$

De aquí se obtiene:

$$\boldsymbol{\alpha_3 \cdot c_1 + \alpha_4 \cdot d_1 = \beta_1}$$

Ecuación 3-43

Donde

$$\alpha_3 = 2(h_1) - q(x_2) \cdot (h_1)^2$$

$$\alpha_4 = 3 \cdot (h_1)^2 - q(x_2) \cdot (h_1)^3$$

$$\beta_2 = q(x_2) \cdot [a_1 + b_1(h_1)] + r(x_2) - b_1$$

Tanto la ecuación 3 13 como la ecuación 3 14 tienen una resolución directa. Sin embargo, las ecuaciones 3 15 y 3 16 forman un sistema de ecuaciones cuya resolución nos proporciona los valores de los coeficientes c_1 y d_1

$$\begin{bmatrix} \alpha_1 & \alpha_2 \\ \alpha_3 & \alpha_4 \end{bmatrix} \cdot \begin{bmatrix} c_i \\ d_i \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}$$

Ecuación 3-44: c y d - No lineal

Este sistema ser resuelve de manera no lineal mediante fsolve.

Una vez resuelto, ya tendremos la definición del Spline en el primer intervalo.

CONDICIONES A CUMPLIR EN EL RESTO DE INTERVALOS

La expresión del Spline en el subintervalo i y de su derivada es de la forma:

$$\begin{cases} S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \\ S_i'(x) = b_i + 2\, c_i(x - x_i) + 3\, d_i(x - x_i)^2 \end{cases}$$

Condición de continuidad de la función

Se tiene que cumplir que $S_{i-1}(x_i) = S_i(x_i)$, es decir, tanto el polinomio S_{i-1}, como el polinomio S_i, evaluados en su nodo de unión deben tomar el mismo valor.

Escribiendo los polinomios en base a sus dominios de definición:

$$a_{i-1} + b_{i-1}(x_i - x_{i-1}) + c_{i-1}(x_i - x_{i-1})^2 + d_{i-1}(x_i - x_{i-1})^3 = a_i + b_i(x_i - x_i) + c_i(x_i - x_i)^2 + d_i(x_i - x_i)^3$$

Simplificando, obtenemos de forma directa el coeficiente a_i

$$\boldsymbol{a_i = a_{i-1} + b_{i-1}(h_{i-1}) + c_{i-1}(h_{i-1})^2 + d_{i-1}(h_{i-1})^3}$$

Ecuación 3-45

Condición de continuidad de derivada primera

Se tiene que cumplir que $S'_{i-1}(x_i) = S'_i(x_i)$, es decir, tanto la derivada del polinomio S_{i-1}, como la del polinomio S_i, evaluadas en su nodo de unión deben tomar el mismo valor.

Escribiendo las derivadas de los polinomios en base a sus dominios de definición:

$$b_{i-1} + 2\,c_{i-1}(x_i - x_{i-1}) + 3\,d_{i-1}(x_i - x_{i-1})^2 = b_i + 2\,c_i(x_i - x_i) + 3\,d_i(x_i - x_i)^2$$

De donde

$$b_i = b_{i-1} + 2\,c_{i-1}(h_{i-1}) + 3\,d_{i-1}(h_{i-1})^2$$

Ecuación 3-46

Condición de colocación en el punto intermedio

Si la función $y(x)$ es solución de la ecuación diferencial, en el punto $x_{i+\frac{1}{2}}$, se cumple:

$$y'_{i+\frac{1}{2}} = q_{i+\frac{1}{2}} \cdot y_{i+\frac{1}{2}} + r_{i+\frac{1}{2}}$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el punto intermedio.

$$b_i + 2\,c_i\left(\frac{h_i}{2}\right) + 3\,d_i\left(\frac{h_i}{2}\right)^2 = q_{i+\frac{1}{2}} \cdot \left[a_i + b_i\left(\frac{h_i}{2}\right) + c_i\left(\frac{h_i}{2}\right)^2 + d_i\left(\frac{h_i}{2}\right)^3\right] + r_{i+\frac{1}{2}}$$

De aquí se obtiene la siguiente ecuación

$$\alpha_{i1} \cdot c_i + \alpha_{i2} \cdot d_i = \beta_{i1}$$

Ecuación 3-47

Donde

$$\alpha_{i1} = 2 \cdot \left(\frac{h_i}{2}\right) - q_{i+\frac{1}{2}} \cdot \left(\frac{h_i}{2}\right)^2$$

$$\alpha_{i2} = 3 \cdot \left(\frac{h_i}{2}\right)^2 - q_{i+\frac{1}{2}} \cdot \left(\frac{h_i}{2}\right)^3$$

$$\beta_{i1} = q_{i+\frac{1}{2}} \cdot \left[a_i + b_i \left(\frac{h_i}{2}\right)\right] + r_{i+\frac{1}{2}} - b_i$$

Condición de colocación en el punto final

Si la función $y(x)$ es solución de la ecuación diferencial, en el nodo x_{i+1}, se cumple:

$$y'_{i+1} = q_{i+1} \cdot y_{i+1} + r_{i+1}$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el punto intermedio.

$$b_i + 2\, c_i h_i + 3\, d_i {h_i}^2 = q_{i+1} \cdot \left[a_i + b_i h_i + c_i {h_i}^2 + d_i {h_i}^3\right] + r_{i+1}$$

De aquí se obtiene la siguiente ecuación

$$\boldsymbol{\alpha_{i3} \cdot c_i + \alpha_{i4} \cdot d_i = \beta_{i2}}$$

Ecuación 3-48

Donde

$$\alpha_{i3} = 2 \cdot h_i - q_{i+1} \cdot {h_i}^2$$

$$\alpha_{i4} = 3 \cdot {h_i}^2 - q_{i+1} \cdot {h_i}^3$$

$$\beta_{i2} = q_{i+1} \cdot [a_i + b_i \cdot h_i] + r_{i+1} - b_i$$

$$\begin{bmatrix} \alpha_{i1} & \alpha_{i2} \\ \alpha_{i3} & \alpha_{i4} \end{bmatrix} \cdot \begin{bmatrix} c_i \\ d_i \end{bmatrix} = \begin{bmatrix} \beta_{i1} \\ \beta_{i2} \end{bmatrix}$$

Ecuación 3-49: c y d - No lineal

Este sistema ser resuelve de manera no lineal mediante fsolve.

Estas dos últimas igualdades forman un sistema de ecuaciones cuya resolución nos proporciona los valores de los coeficientes c_i y d_i.

3.9. MÉTODO CON SPLINES DE GRADO 3 (NO LINEAL)

Este método también usa Splines de grado 3. Sin embargo, a diferencia del anterior, ese grado extra lo vamos a usar para imponer más regularidad en la función, de tal manera que sea continua y derivable dos veces.

$$S(x) = \begin{cases} S_1(x) = a_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3 & x \in [x_1, x_2) \\ S_2(x) = a_2 + b_2(x - x_2) + c_2(x - x_2)^2 + d_2(x - x_2)^3 & x \in [x_2, x_3) \\ \dots \\ S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 & x \in [x_i, x_{i+1}) \\ \dots \\ S_n(x) = a_n + b_n(x - x_n) + c_n(x - x_n)^2 + d_n(x - x_n)^3 & x \in [x_n, x_{n+1}] \end{cases}$$

Ecuación 3-50 Spline de grado 3

Tabla 3-7 Resumen condiciones Pseudo-Spline de grado 3

Condiciones que imponemos en cada intervalo	
Intervalo inicial	Intervalo $i,\ i = 2, 3, \dots, n$
• Condición inicial • Condición de colocación en el nodo x_1 • Condición de colocación en el nodo $\frac{x_1+x_2}{2}$ • Condición de colocación en el nodo x_2	• Condición de continuidad de la función. • Condición de continuidad de la derivada primera en el nodo x_i • Condición de continuidad de la derivada segunda en el nodo x_i • Condición de colocación en el nodo x_{i+1}

En cada intervalo $[x_i, x_{i+1}]$ con $i = (1,2, \dots, n)$ definimos el Pseudo-Spline que aproxima a la solución como:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

Derivando este polinomio respecto de su variable independiente tenemos:

$$S_i'(x) = b_i + 2\, c_i(x - x_i) + 3\, d_i(x - x_i)^2$$

CONDICIONES A CUMPLIR EN EL PRIMER INTERVALO

En este primer intervalo necesitamos el cálculo de 4 coeficientes. Para obtener el cuarto se ha optado por la estrategia de imponer un nodo más en el punto medio del intervalo, en el que se debe satisfacer la condición de colocación.

Para el intervalo inicial tenemos:

$$\begin{cases} S_1(x) = a_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3 \\ S_1'(x) = b_1 + 2\,c_1(x - x_1) + 3\,d_1(x - x_1)^2 \\ S_1(a) = z_1 \end{cases}$$

De donde tenemos que:

Condición inicial

El valor de la función en el nodo inicial es z_1

$$y(x_1) = z_1$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo inicial.

$$a_1 + b_1(x_1 - x_1) + c_1(x_1 - x_1)^2 + d_1(x_1 - x_1)^3 = z_1$$

$$\boldsymbol{a_1 = z_1}$$

Ecuación 3-51

Condición de colocación en el nodo inicial del intervalo

Si la función $y(x)$ es solución de la ecuación diferencial, en el nodo x_1, se cumple:

$$y'(x_1) = q(x_1) \cdot y(x_1) + r(x_1)$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo inicial.

$$\begin{aligned} b_1 + 2\,c_1(x_1 - x_1) + 3\,d_1(x_1 - x_1)^2 \\ = q(x_1) \cdot [a_1 + b_1(x_1 - x_1) + c_1(x_1 - x_1)^2 + d_1(x_1 - x_1)^3] \\ + r(x_1) \end{aligned}$$

$$\boldsymbol{b_1 = q(x_1) \cdot a_1 + r(x_1)}$$

Ecuación 3-52

Condición de colocación en el nodo intermedio del intervalo

Si la función $y(x)$ es solución de la ecuación diferencial, en el punto $\frac{x_1+x_2}{2}$, se cumple:

$$y'\left(\frac{x_1+x_2}{2}\right) = q\left(\frac{x_1+x_2}{2}\right)\cdot y\left(\frac{x_1+x_2}{2}\right) + r\left(\frac{x_1+x_2}{2}\right)$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el punto intermedio.

$$b_1 + 2\,c_1\cdot\left(\frac{h_1}{2}\right) + 3\,d_1\left(\frac{h_1}{2}\right)^2 = q_{1+\frac{1}{2}}\cdot\left[a_1 + b_1\cdot\left(\frac{h_1}{2}\right) + c_1\cdot\left(\frac{h_1}{2}\right)^2 + d_1\cdot\left(\frac{h_1}{2}\right)^3\right] + r_{1+\frac{1}{2}}$$

De aquí se obtiene:

$$\alpha_1\cdot c_1 + \alpha_2\cdot d_1 = \beta_1$$

Ecuación 3-53

Donde

$$\alpha_1 = 2\left(\frac{h_1}{2}\right) - q_{1+\frac{1}{2}}\cdot\left(\frac{h_1}{2}\right)^2$$

$$\alpha_2 = 3\cdot\left(\frac{h_1}{2}\right)^2 - q_{1+\frac{1}{2}}\cdot\left(\frac{h_1}{2}\right)^3$$

$$\beta_1 = q_{1+\frac{1}{2}}\cdot\left[a_1 + b_1\cdot\left(\frac{h_1}{2}\right)\right] + r_{1+\frac{1}{2}} - b_1$$

Condición de colocación en el nodo final del intervalo

Si la función $y(x)$ es solución de la ecuación diferencial, en el nodo x_2, se cumple:

$$y'(x_2) = q(x_2)\cdot y(x_2) + r(x_2)$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo final.

$$b_1 + 2\,c_1(h_1) + 3\,d_1(h_1)^2 = q(x_2)\cdot[a_1 + b_1(h_1) + c_1(h_1)^2 + d_1(h_1)^3] + r(x_2)$$

De aquí se obtiene:

$$\boldsymbol{\alpha_3 \cdot c_1 + \alpha_4 \cdot d_1 = \beta_1}$$

Ecuación 3-54

Donde

$\alpha_3 = 2(h_1) - q(x_2) \cdot (h_1)^2$

$\alpha_4 = 3 \cdot (h_1)^2 - q(x_2) \cdot (h_1)^3$

$\beta_2 = q(x_2) \cdot [a_1 + b_1(h_1)] + r(x_2) - b_1$

Estas dos últimas igualdades, Ecuación 3 24 y Ecuación 3 25, forman un sistema de ecuaciones cuya resolución nos proporciona los valores de los coeficientes c_1 y d_1

$$\begin{bmatrix} \alpha_1 & \alpha_2 \\ \alpha_3 & \alpha_4 \end{bmatrix} \cdot \begin{bmatrix} c_i \\ d_i \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}$$

Ecuación 3-55

Este sistema se resuelve de manera no lineal mediante fsolve.

CONDICIONES A CUMPLIR EN EL RESTO DE INTERVALOS

Condición de continuidad de la función

Se tiene que cumplir que $S_{i-1}(x_i) = S_i(x_i)$, es decir, tanto el polinomio S_{i-1}, como el polinomio S_i, evaluados en su nodo de unión deben tomar el mismo valor.

Escribiendo los polinomios en base a sus dominios de definición:

$$a_{i-1} + b_{i-1}(h_{i-1}) + c_{i-1}(h_{i-1})^2 + d_{i-1}(h_{i-1})^3 = a_i + b_i(x_i - x_i) + c_i(x_i - x_i)^2 + d_i(x_i - x_i)^3$$

Simplificando, obtenemos de forma directa el coeficiente a_i

$$\boldsymbol{a_i = a_{i-1} + b_{i-1}(h_{i-1}) + c_{i-1}(h_{i-1})^2 + d_{i-1}(h_{i-1})^3}$$

Ecuación 3-56

Condición de continuidad de derivada primera.

Se tiene que cumplir que $S'_{i-1}(x_i) = S'_i(x_i)$, es decir, tanto la derivada del polinomio S_{i-1}, como la del polinomio S_i, evaluadas en su nodo de unión deben tomar el mismo valor.

Escribiendo las derivadas de los polinomios en base a sus dominios de definición:

$$b_{i-1} + 2\,c_{i-1}(h_{i-1}) + 3\,d_{i-1}(h_{i-1})^2 = b_i + 2\,c_i(x_i - x_i) + 3\,d_i(x_i - x_i)^2$$

De donde

$$\boldsymbol{b_i = b_{i-1} + 2\,c_{i-1}(h_{i-1}) + 3\,d_{i-1}(h_{i-1})^2}$$

Ecuación 3-57

Condición de continuidad de derivada segunda

Se tiene que cumplir que $S''_{i-1}(x_i) = S''_i(x_i)$, es decir, tanto la derivada segunda del polinomio S_{i-1}, como la del polinomio S_i, evaluadas en su nodo de unión deben tomar el mismo valor.

Escribiendo las derivadas segundas de los polinomios en base a sus dominios de definición:

$$2\,c_{i-1} + 6\,d_{i-1}(h_{i-1}) = 2\,c_i + 6\,d_i(x_i - x_i)^2$$

De donde

$$\boldsymbol{c_i = \frac{2\,c_{i-1} + 6\,d_{i-1}(h_{i-1})}{2}}$$

Ecuación 3-58

Condición de colocación en el nodo final del intervalo

Si la función $y(x)$ es solución de la ecuación diferencial, en el nodo x_{i+1}, se cumple:

$$y'_{i+1} = q_{i+1} \cdot y_{i+1} + r_{i+1}$$

Sustituyendo la función $y(x)$ por el polinomio que la aproxima en ese intervalo y particularizamos para el nodo final.

$$b_i + 2\,c_i \cdot h_i + 3\,d_i h_i^2 = q_{i+1} \cdot [a_i + b_i \cdot h_i + c_i h_i^2 + d_i h_i^3] + r_{i+1}$$

Despejando el coeficiente d_i tenemos

$$d_i = \frac{q_{i+1} \cdot \left(a_i + b_i \cdot h_i + c_i \cdot h_i^2\right) + r_{i+1} - b - 2 \cdot c \cdot h_i}{3 \cdot h_i^2 - q_{i+1} \cdot h_i^3}$$

Ecuación 3-59

Esta ecuación se resuelve de manera no lineal mediante fsolve.

Estas ecuaciones tienen una resolución directa y no precisan de un sistema para su resolución.

4. MÉTODOS DE CONTROL DEL ERROR Y DEL TAMAÑO DE PASO

En este capítulo vamos a describir distintas estrategias que se pueden usar en los métodos basados en Splines para tener un paso adaptativo.

Una explicación excelente para este apartado se puede encontrar en la siguiente referencia bibliográfica.

(Rojo, 2020)

4.1. NECESIDAD DE CONTROL DE PASO

Hay que tener siempre presente que toda solución proporcionada va a tener un cierto error respecto de la función exacta. Este error lo vamos a fijar en forma de una cierta tolerancia con respecto al valor real de la función.

En este punto ya tenemos el primer problema, ya que, como se ha dicho antes, normalmente no se conoce el valor real de la función, ya que, de conocerlo, se habría resuelto el problema de manera exacta. Además, con un método numérico es imposible conocer el error cometido.

El segundo problema es de funcionamiento del método. Hemos visto que el campo de pendientes no es constante y existen, por lo general, zonas de variación más rápida y zonas de variación más lenta. Trabajar bien en los dos tipos de zonas es incompatible con un escenario en el que los nodos están equiespaciados. Si funcionan bien en las zonas de variación lenta, nos van a funcionar peor en las zonas de variación rápida, pudiendo llegarse a errores inaceptables. Por otro lado, si tomamos un paso lo suficientemente pequeño para aproximar en las zonas de variación rápida, vamos a realizar muchos más cálculos de los necesarios en las zonas de variación más lenta.

Vamos a ver que, usando una combinación de métodos, podemos dar una solución a estos problemas. En su fundamento se trata de avanzar para los mismos nodos con dos estrategias de resolución distintas, una de las cuales debe ofrecer mejores resultados que la otra, y de la diferencia entre ambas estimaciones, nos podemos hacer una idea muy aproximada de cómo está funcionando nuestro método.

Si la estimación del error es aceptable, se acepta el paso y se continúa el método con un paso propuesto proporcional a la diferencia entre el error estimado y el error permitido.

Si la estimación no es aceptable, se rechaza el paso y se usa un paso más pequeño, proporcional a la diferencia entre el error estimado y el error permitido.

4.2. ORDEN DE LOS MÉTODOS A USAR.

Las estrategias de control de paso se basan en que tenemos dos técnicas diferentes para calcular una aproximación. Una de ellas debe ser más precisa que la otra, a fin de que podamos tener un elemento de comparación. Esto lo conseguimos usando un método que converja más rápido localmente a la solución (pares encajados), o mediante la comparación de avanzar con un método con un paso h y con un paso $\frac{h}{2}$. En ambos casos, tendremos una aproximación más exacta que la otra.

El orden de un método nos sirve para estimar la velocidad de convergencia del mismo. Vamos a suponer que trabajamos con tamaños de pasó lo suficientemente pequeños, y suponemos que para un tamaño de paso h_1 el error es E_1. Ahora avanzamos con un tamaño de paso y bajo las mismas condiciones de partida con $h_2 = \frac{1}{10} \cdot h_1$. El nuevo error cometido es, en orden de magnitud es

$$error_2 = \frac{1}{10^p} \cdot error_1$$

Siendo p el orden del método que usamos.

Para un método de orden 1, reducir el paso por 10, es decir, efectuar 10 veces más operaciones, nos reduce el error por 10. Sin embargo, para un método de orden 2, efectuar 10 veces más operaciones, reduce el error por 100. Si es de orden 3, lo hace por 1000, etc.... Vemos que, nos interesa trabajar con métodos de orden lo más alto posible.

Determinación empírica del orden de convergencia de un método numérico.

Vamos a avanzar en un determinado intervalo con un tamaño de paso h_1 y obtendremos un valor para la función Spline1 en el final del intervalo de $\hat{y}_1$. Luego repetimos en el mismo intervalo y con el mismo método, pero esta vez con un tamaño de paso $h_2 = \frac{h_1}{2}$, es decir, la mitad que el anterior. Obtendremos de este modo un valor en el final del intervalo al evaluar la función Spline2 de $\hat{y}_2$.

Si restamos ambos valores del valor real de la función en ese punto, y nos quedamos con el valor absoluto tenemos

$$E_{h_1} = K \cdot h_1^p$$

$$E_{h_2} = K \cdot h_2^p$$

Para despejar el orden p,

$$\frac{E_{h_1}}{E_{h_2}} = \frac{K \cdot h_1^p}{K \cdot h_2^p} = \frac{h_1^p}{\left(\frac{h_1}{2}\right)^p} = 2^p \longrightarrow p = \log_2\left(\frac{E_{h_1}}{E_{h_2}}\right)$$

Otra forma inteligente de ver el orden, y que es más visual, se basa en estudiar la evolución del error global a lo largo de in cierto intervalo y relacionarlo con el tamaño de paso. De este modo, se obtiene una relación lineal, después de un ajuste a una escala logarítmica, cuya pendiente nos indica el orden del método.

Este caso se puede ver desarrollado en los experimentos numéricos.

4.3. TOLERANCIA DEL MÉTODO

Vamos a establecer para nuestro intervalo una tolerancia en la que le indicamos al método que no se separe de la solución exacta una distancia mayor de esa tolerancia. Este método supone que los errores son acumulativos y crecientes según avanza el método.

Vamos a fijar un valor para la tolerancia

$$Atol$$

Con base a este valor y al intervalo de resolución que tenemos, vamos a calcular una tolerancia unitaria, es decir, en cada unidad que recorramos en el eje de la variable independiente, nos vamos a poder alejar de la función objetivo un valor máximo de:

$$Utol = \frac{Atol}{b - a}$$

Pero realmente para nuestro algoritmo lo que necesitamos es una tolerancia por paso, que corresponde al valor de multiplicar la tolerancia unitaria por el tamaño de paso.

$$Stol = Utol \cdot h$$

Siendo h la distancia entre los dos extremos del subintervalo.

La nomenclatura ha sido escogida para que siga el mismo criterio que otras funciones implementadas en Matlab, correspondiendo “tol” a tolerancia, y los Prefijos A, U, S, correspondiendo con Absoluta, Unitary, y Step.

4.4. MECANISMO DE ADAPTACIÓN DEL PASO

Vamos a dejar para los siguientes apartados la forma particular en que cada uno estima el error, pero antes de eso vamos a ver una serie de cuestiones comunes a todos ellos.

- Estos métodos ofrecen una estimación del error, no el error en sí. Esto es natural ya que, si tuviéramos un conocimiento del valor exacto de la función, no tendríamos que aproximar nada.
- El error que estiman es el error local de truncación, es decir, el error cometido en un único paso.

Vamos a explicar cómo se realiza la adaptación del paso para el método de pares encajados, siendo perfectamente adaptable al método de extrapolación de Richardson.

Cabe destacar aquí que no se pueden dar pasos infinitamente pequeños. La explicación a ello se puede ver en el anexo de definiciones, pero a grandes rasgos esto está motivado porque los ordenadores trabajan con unas cantidades finitas y al operar con cifras muy similares los errores producidos por las operaciones de redondeo anulan por completo los beneficios de la reducción del paso.

Si la tolerancia admitida en el paso corresponde con $Stol$, queda claro que el paso óptimo a dar, es aquel que nos permita conseguir esa tolerancia en dicho paso.

El error local cometido al dar un paso de tamaño h para un método de orden $p1$ es

$$E_{p1}(h_i) = K \cdot h_i^{(p1+1)}$$

Este error es la diferencia entre el valor real de la función y la aproximación en ese punto.

$$E_{p1}(h_i) = S_{p1}(x_{i+1}) - y(x_{i+1})$$

Del mismo modo, el error local cometido al dar el mismo paso, pero esta vez con un método de orden $p2 > p1$ es

$$E_{p2}(h_i) \cdot K \cdot h_i^{(p2+1)}$$

Siendo el error cometido

$$E_{p2}(h_i) = S_{p2}(x_{i+1}) - y(x_{i+1})$$

De este modo, un método de orden $p2$ proporciona una estimación mucho más precisa para el mismo tamaño de paso.

Vamos a estimar el error como la diferencia entre ambos

$$\hat{E}_i \approx |E_{p1} - E_{p2}| = |S_{p1}(x_{i+1}) - S_{p2}(x_{i+1})|$$

En cualquier caso, tanto el nuevo paso sugerido para el siguiente subintervalo en el caso de que el error sea menor que la tolerancia, como el modificado para iterar de nuevo, en el caso de ser mayor el error que la tolerancia, corresponde con la expresión:

$$h_{i+1} = \alpha \cdot h_i$$

Siendo

$$\alpha = C_s \cdot \left(\frac{Stol_i}{E_i}\right)^{\frac{1}{p_1}}$$

Ecuación 4-1 Coeficiente Alpha

Donde

C_s es un coeficiente de seguridad, con un valor empírico entre $0{,}84$ y $0{,}90$ que hace que sea menos probable rechazar un paso. Lo que hace este coeficiente es, en definitiva, decirle al método, que el error es un poco mayor que lo que en realidad hemos estimado, con lo cual trabajamos a favor de aumentar la seguridad de la estimación.

En caso de rechazar el paso

$E_i > Stol_i$, con lo que $\alpha < 1$

En caso de aceptar el paso

$E_i > Stol_i$, con lo que $\alpha > 1$

También, se debe tener en cuenta, tras la práctica y la experimentación de estos métodos, que no es conveniente tener unos cambios de paso muy bruscos, por lo que se fijará un valor máximo y un valor mínimo para el parámetro α de modulación de paso. De este modo, el rango del parámetro α queda como:

$$\alpha \in \left[\frac{1}{5}, \quad C_s \cdot \left(\frac{Stol_i}{E_i}\right)^{\frac{1}{p_1}}, \quad 3\right]$$

Ecuación 4-2 Rango de aplicación de Alpha

Siendo

$\alpha_{mín} = \frac{1}{5}$

$\alpha_{máx} = 3$

Estos valores de $\alpha_{mín}$ y $\alpha_{máx}$ no son simétricos. Es mucho más permisivo el que acorta el paso, ya que lo que se busca es mantener el error dentro de los límites establecidos. No obstante, el programa se desarrolla de manera que pueda seguir operando, pero mostrando que se ha salido de las condiciones idóneas de ese valor α.

A la hora del valor máximo, este sí se puede limitar. No debe preocuparnos dar un paso más pequeño que el que deberíamos dar, ya que el incremento del paso anterior por 3 es suficiente para la mayoría de los casos y, aunque puede llevar a unas pocas más de operaciones, se trabaja en favor de la precisión del método.

De todos modos, también se va a fijar un tamaño de paso máximo, a fin de que el método no dé pasos exageradamente elevados. Este paso máximo se va a escoger en función del intervalo de interés.

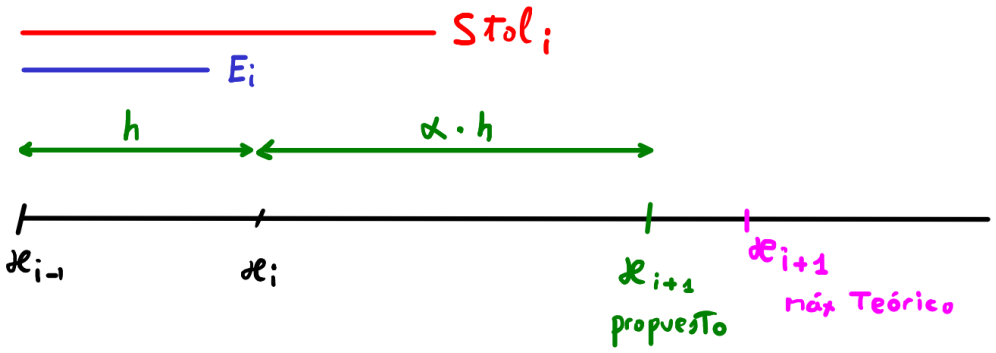


Figura 4-1 Aceptación de paso y adaptación del siguiente

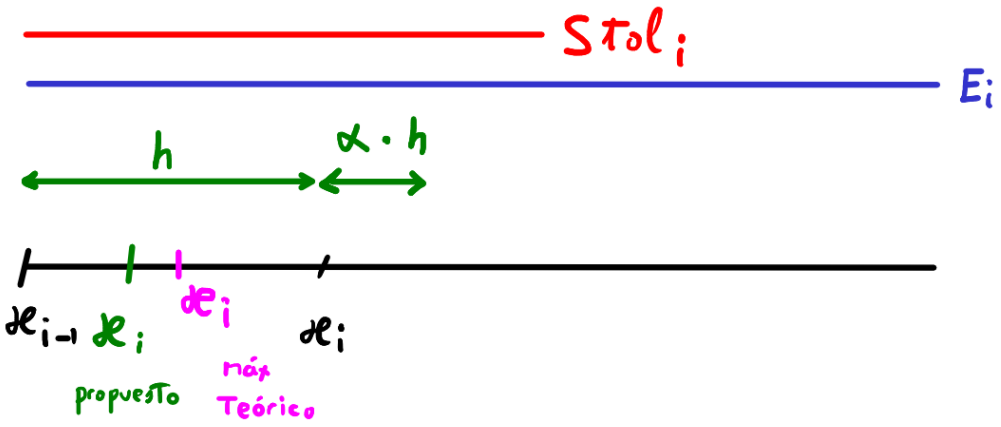


Figura 4-2 Rechazo del paso y cálculo del nuevo

(Rojo, 2020)

4.5. CONTROL MEDIANTE PARES ENCAJADOS

En este apartado nos vamos a centrar en las particularidades de cada método al estimar el error local en cada paso, ya que una vez conocido, la estimación del paso siguiente se realiza de la misma manera, según se ha descrito.

Este método se basa en el avance desde el mismo punto de referencia, el cual consideramos como parte de la solución exacta de acuerdo a la definición de error local en el anexo, con dos métodos de órdenes distintos. Esto es que para el intervalo i se parta del punto (x_i, y_i), siendo y_i el resultado de evaluar el Spline S_{i-1} en el punto x_i.

Al avanzar con dos métodos De la diferencia entre los puntos finales se obtiene un valor que estima de una manera bastante aproximada, el error local.

Aquí la notación que usamos para el error local en el paso i es E_i, habida cuenta que en todos estos métodos trabajamos únicamente con estimadores del error real.

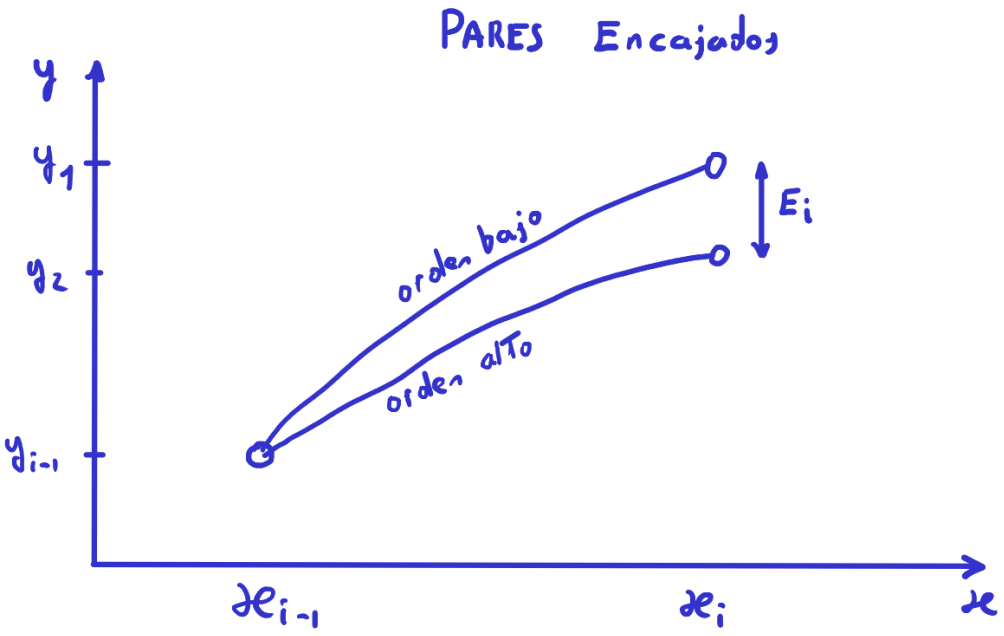


Figura 4-3 Pares encajados

Para ver más claro esto, vamos a tomar una función y a ver qué pasa con el error local en un paso.

Tomaremos la función

$$f(x) = 2 + 2 \cdot e^{-2 \cdot x^2}$$

Que es la solución de la Ecuación diferencial

$$\begin{cases} y'(x) = -4 \cdot x \cdot y(x) + 8 \cdot x \\ y(0) = 4 \end{cases}$$

Y vamos a calcular el error local para un paso $h = 1$ partiendo del punto $(0, 4)$

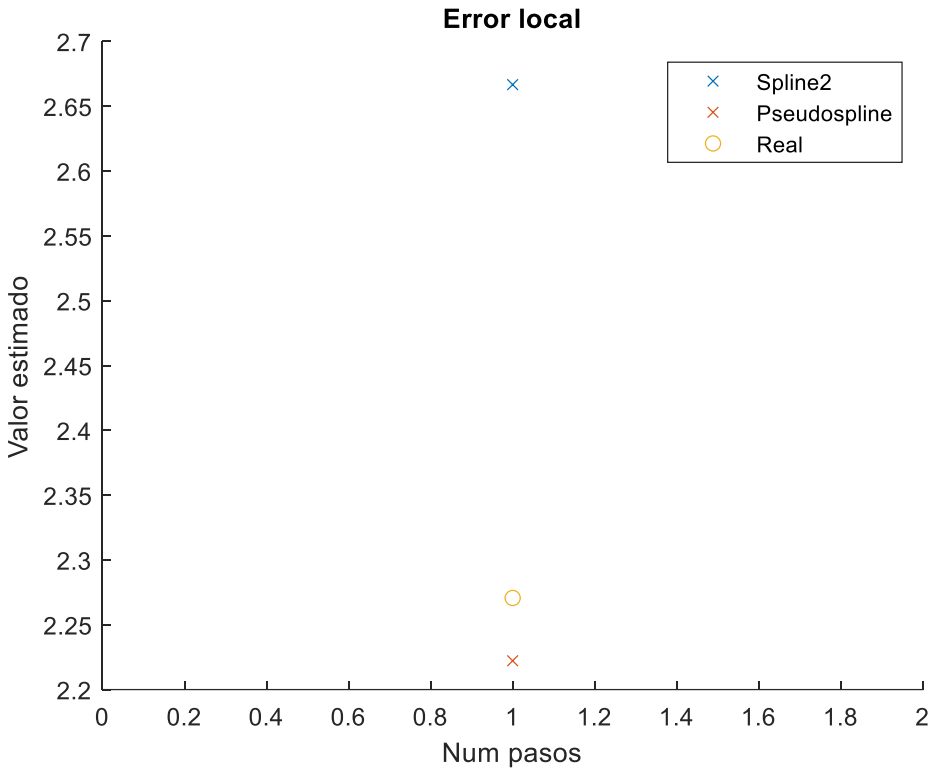


Figura 4-4 Error local Spline2 Vs PseudoSpline. h=1

Usando el estimador calculado para un paso i, tenemos un error local de

$$Y_1 - Y_2 = 0{,}44444444444$$

El valor real del error en ese paso sería

$$Y_1 - Y_{real} = 0{,}39599610019$$

Este es un ejemplo muy visual, pero al mismo tiempo, muy alejado del funcionamiento del programa.

Lo razonable es trabajar con pasos más pequeños, ya que las desviaciones que se permiten en cada paso son muy pequeñas. Esto es necesario porque las soluciones que se buscan deben ser tan cercanas a la solución real como sea posible.

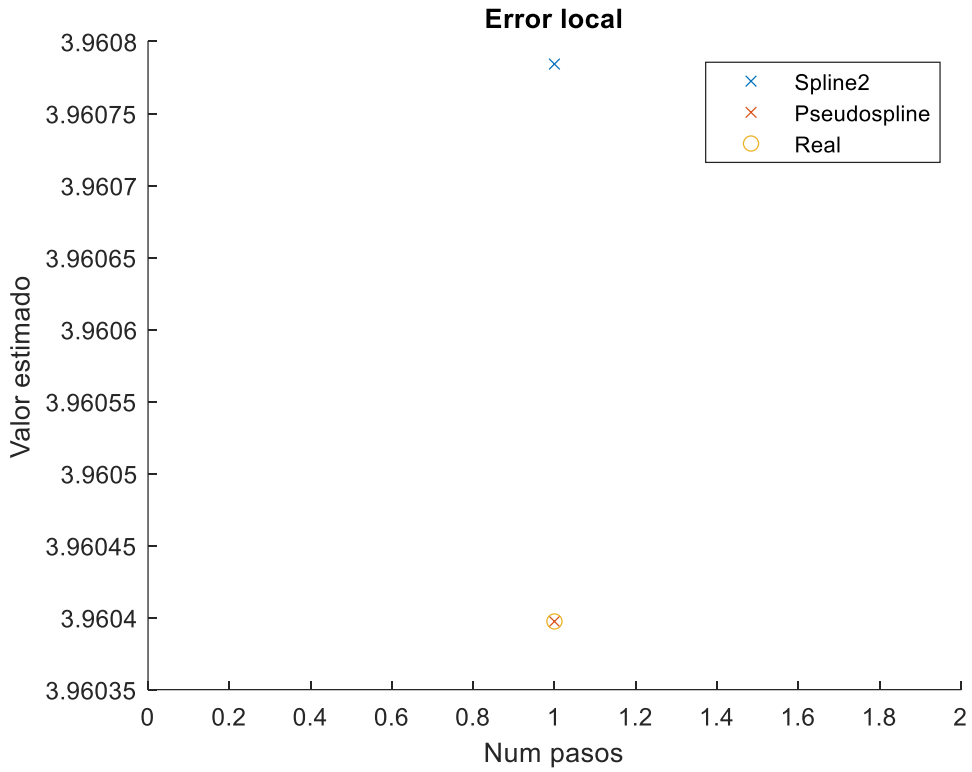


Figura 4-5 Error local Spline2 Vs PseudoSpline. h=0.1

Vamos a calcular el error local para un paso $h = 0{,}1$ partiendo del punto $(0, 4)$

Usando el estimador calculado para un paso i, tenemos un error local de

$$Y_1 - Y_2 = 3{,}869714453772488e \cdot 10^{-4}$$

El valor real del error en ese paso sería

$$Y_1 - Y_{real} = 3{,}869671119796081 \cdot 10^{-4}$$

Se puede comprobar que es un buen método de aproximación.

4.6. CONTROL MEDIANTE EXTRAPOLACIÓN DE RICHARSON

En este método de control, vamos a avanzar usando la misma estrategia de resolución para el método de avance y el método de control. La diferencia está en que en el método de control vamos a avanzar con un paso la mitad de pequeño que el del método de avance. Esto nos garantiza tener dos aproximaciones distintas. De la diferencia entre ellas vamos a calcular un error estimado en base a los valores que toman las funciones en el nodo final y

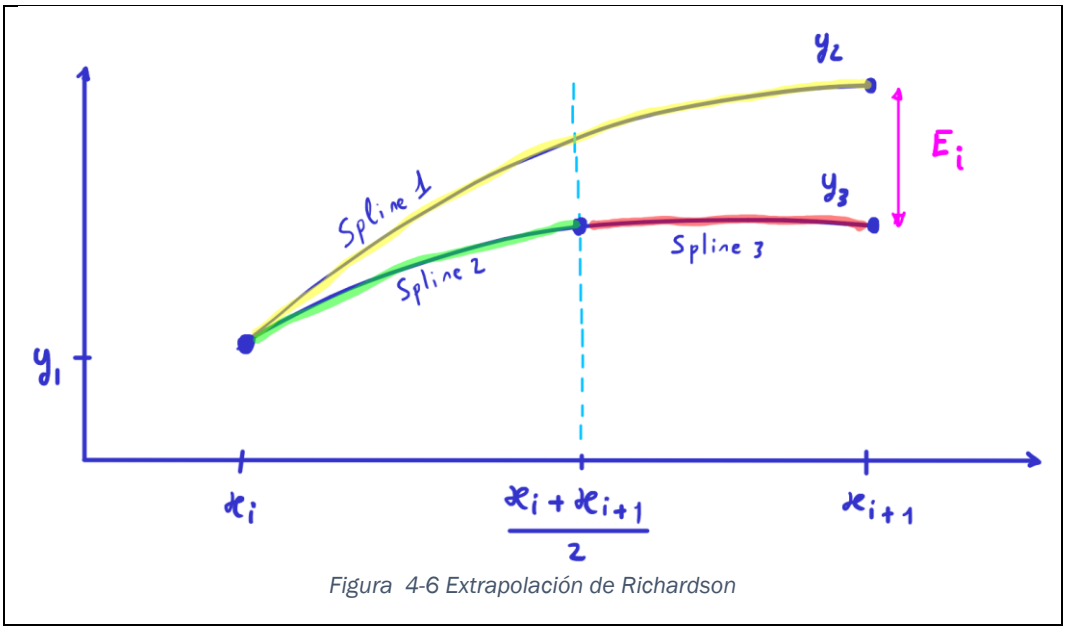


Figura 4-6 Extrapolación de Richardson

Una vez efectuada esta explicación, vamos a ver su funcionamiento real.

Tomaremos la misma función que en el apartado anterior

$$f(x) = 2 + 2 \cdot e^{-2 \cdot x^2}$$

Que es la solución de la Ecuación diferencial

$$\begin{cases} y'(x) = -4 \cdot x \cdot y(x) + 8 \cdot x \\ y(0) = 4 \end{cases}$$

Vamos a calcular el error local para un paso $h = 1$ partiendo del punto $(0, 4)$

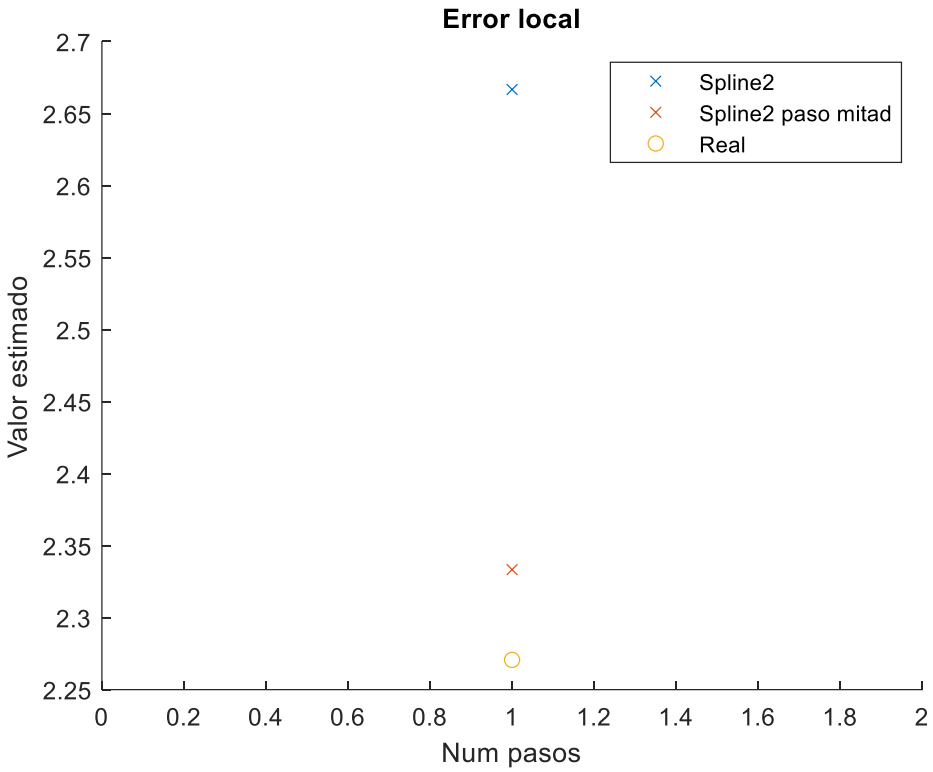


Figura 4-7 Error local Spline2 Vs Spline2(mitad). h=1

Usando el estimador calculado para un paso i, tenemos un error local de

$$Y_1 - Y_2 = 0{,}333333333333333$$

El valor real del error en ese paso sería

$$Y_1 - Y_{real} = 0{,}395996100193441$$

Con un ejemplo más cercano a la realidad tenemos:

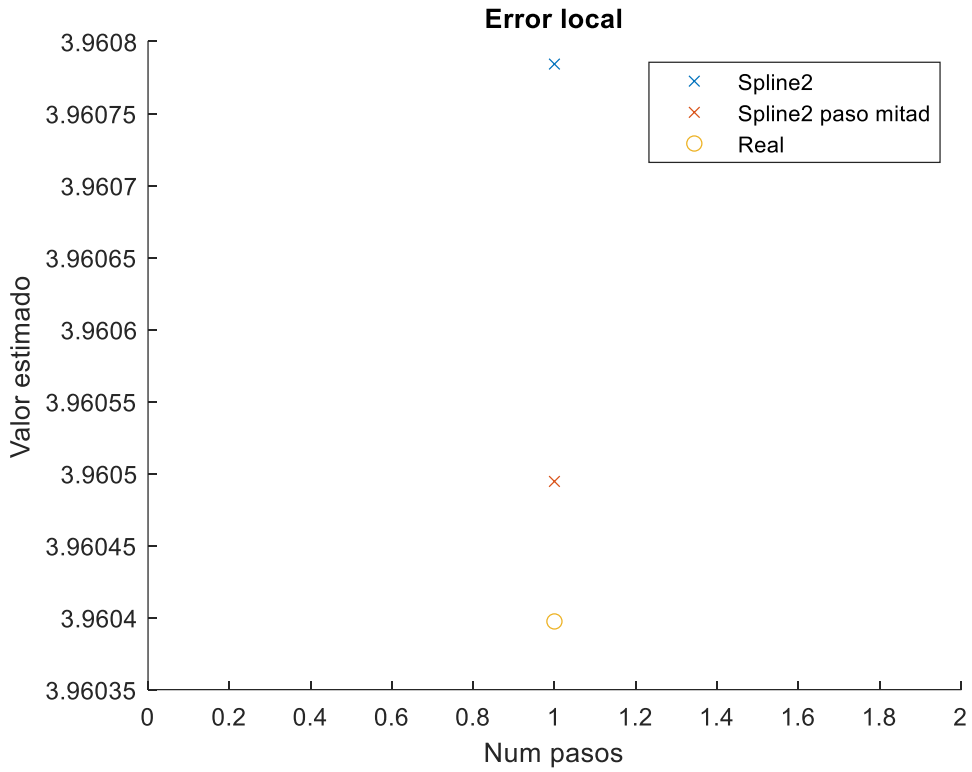


Figura 4-8 Error local Spline2 Vs Spline2(mitad). h=0.1

Usando el estimador calculado para un paso i, tenemos un error local de

$$Y_1 - Y_2 = 2{,}897568071120737 \cdot 10^{-4}$$

El valor real del error en ese paso sería

$$Y_1 - Y_{real} = 3{,}869671119796081 \cdot 10^{-4}$$

Se trata ya de una estimación más realista.

(Faires, 2017)

4.7. CONTROL MEDIANTE MODIFICACIÓN DE LA PENDIENTE

Tanto la extrapolación de Richardson, como el método de pares encajados, ofrecen una comparación de distancia sobre el eje de ordenadas. Esto quiere decir que va a funcionar muy bien en zonas con pendientes suaves, pero en zonas con pendientes muy elevadas, el método va a obligar a dar muchísimos pasos.

Modificando algunos parámetros, podemos conseguir unas estimaciones del error dependientes de esa distancia que nos proporcionan los métodos tradicionales. Va a ser una dependencia no lineal, ya que está relacionada con el coseno del ángulo que forma el campo de pendientes en el punto final de la aproximación del Spline con el eje horizontal.

Los métodos basados en Splines proporcionan polinomios y vamos a tratar de estimar el error como una distancia entre el polinomio en el intervalo i y la parte de función exacta en ese mismo intervalo.

Esto en los métodos Runge-Kutta no es posible, ya que únicamente tenemos una sucesión de puntos.

Como se verá más adelante, el proceso es menos costoso, pero tiene como contrapartida que también es menos preciso en esa zona.

Para llegar a una solución de compromiso, esta estimación del error entre funciones está afectada por un coeficiente de exigencia, que nos va a indicar cómo de estrictos vamos a ser con esa estimación. Dicho de otro modo, vamos a poder variar entre los dos casos extremos de distancia entre cotas o distancia entre funciones con un amplio rango de casos.

Vamos a añadir, además, que estos métodos proporcionan unas aproximaciones válidas y menos costosas, sacrificando la precisión. Si bien se ha comprobado que funcionan experimentalmente, no se puede decir que se comporten del mismo modo en todos los casos y que no presenten inexactitudes en algunas ocasiones.

De cualquier modo, queda a criterio del matemático, la aplicación de este método en un caso concreto, basándose en su experiencia.

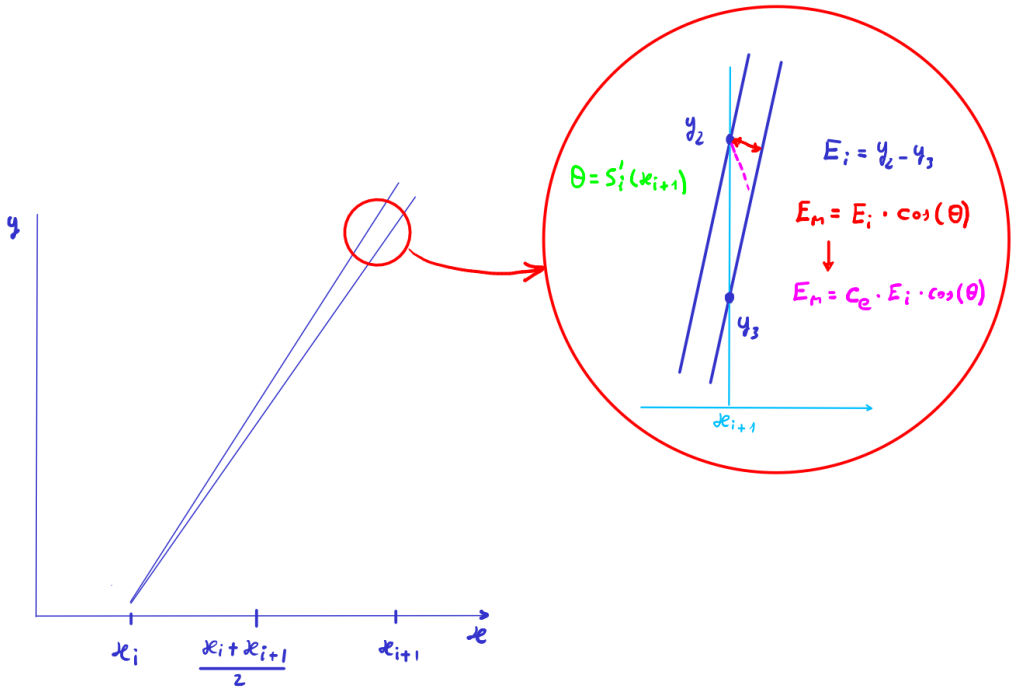


Figura 4-9 Extrapolación de Richardson modificada

$$E_i = y_2 - y_1$$

Sin embargo, esta distancia, cuando tenemos pendientes muy acusadas puede ser muy grande, obligando al método a dar pasos muy pequeños para satisfacer la tolerancia prefijada.

Esto se puede relajar un poco, dependiendo del tipo de ecuación, usando la proyección de ese error de cotas sobre la perpendicular al Spline en el extremo final del subintervalo.

Sabiendo que la derivada al final del intervalo i es, para el caso de Splines de grado dos,

$$S_i'(x_{i+1}) = b_i + 2 \cdot c_i(x_{i+1} + x_i)$$

Y

$$S_i'(x_{i+1}) = b_i + 2 \cdot c_i(x_{i+1} + x_i) + 3 \cdot d_i(x_{i+1} + x_i)^2$$

Podemos obtener el ángulo θ que forma la tangente a la gráfica en ese punto.

$$\theta = \operatorname{atan}\left(S_i'(x_{i+1})\right)$$

De este modo, si el Spline tiene poca pendiente, tenemos que

$$\theta \approx 0\, rad \implies \cos(\theta) \approx 1$$

Lo que encaja muy bien con la definición del error.

Para casos de pendiente más elevada

$$\theta \approx \frac{\pi}{2}\, rad \implies \cos(\theta) \approx 0$$

Por supuesto que no se va a llegar a ese caso, pero vemos como, mediante el coseno podemos Transformar un error E_i en otro más pequeño, y, por lo tanto, más susceptible de cumplir con la tolerancia.

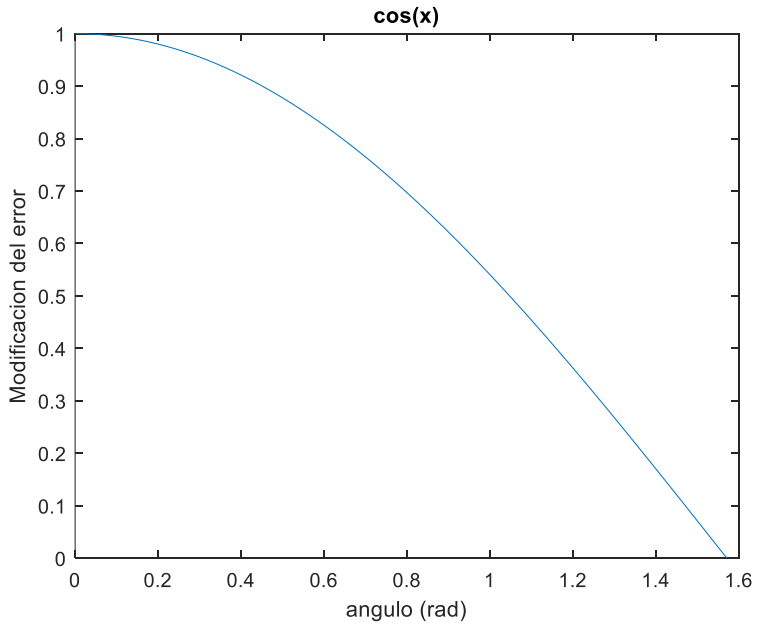
Naturalmente no es una buena idea dejar todo el peso de la decisión a ese criterio, por lo que usaremos un coeficiente de exigencia que va a modular el comportamiento de este criterio entre el caso de no tomar proyección alguna, o tomar proyección sobre la perpendicular.

La idea es tomar un Coeficiente, entre 0 y 100, que modifique de algún modo la importancia de este ángulo.

$Ce = 0$: La influencia del ángulo es total. Se calcula la proyección del error sobre la recta perpendicular al Spline en su punto final.

$Ce = 100$: La influencia del ángulo es nula, quedando el error sin modificar.

En el resto de casos, cuando el coeficiente es distinto de 100, el error nuevo se va a estimar menor que el vertical, siendo obvio que esto va a redundar en un número menor de pasos y una mayor inexactitud, pero, a cambio, una mayor rapidez.



Ecuación 4-3 Modificación del error

Esta no linealidad va a contribuir de manera desigual según la pendiente, siendo mayor la modificación en zonas de pendientes elevadas.

5. INTRODUCCIÓN A LOS SOLVER IMPLEMENTADOS EN MATLAB

5.1. MENCIÓN DE LOS MÉTODOS

Matlab posee varios métodos de resolución de ecuaciones diferenciales, llamado solvers, que, desde hace unos años, ayudan a científicos de todo el mundo a lidiar con las ecuaciones diferenciales y que van a proporcionar una sucesión de puntos que aproximan a la solución exacta.

Estos métodos se van corrigiendo y ampliando con el tiempo, y el que haya tantos quiere decir que cada uno tiene su zona de especialización, es decir, para una ecuación o una región dadas, presenta unas mejores características que otro método.

Vamos a describir de manera general un poco qué hace cada uno y, mediante la experimentación, observaremos su comportamiento.

Tabla 5-1: Resumen de solvers

Solver	Tipo de problema	Precisión
Ode45	Normal	Alto
Ode23	Normal	Bajo
Ode113		
Ode15s	Normal	Bajo
Ode23s	Staff	Medio
Ode23t	Stiff	Bajo
Ode23tb	Semi-stiff	Bajo
Ode15i	implícito	Bajo

Fuente: Adaptada de Bruce Mayer, (Métodos computacionales en ingeniería)

Vamos a describir de manera breve alguno de estos métodos.

- Ode45

  Se trata de un método que se adapta bien a casi todas las ecuaciones, menos a las que tienen un alto grado de rigidez. De los métodos integrados en Matlab es el que primero se suele probar y se basa en el método de Runge-Kutta. Es un método de un paso

- Ode23

Este método es similar al ode45, si bien es menos preciso, pero más rápido.

- Ode113

Se trata de un método multifase

- Ode15s

Resuelve ecuaciones complejas. Se trata de un método multipaso.

- Ode23s

Resuelve ecuaciones complejas. Es un método de un solo paso

- Ode23t

Resuelve ecuaciones de dificultad media

- Ode23tb

Este solver, corresponde a un método implícito Runge-Kutta. Tiene una precisión media y se emplea cuando los métodos generales ode34 y ode45 tarda mucho en encontrar una solución.

La guía correspondiente a esta tabla se puede consultar en la sección de apéndices.

Una manera muy visual de comparar parte del comportamiento de estos solvers es mediante el error que toleran.

Se ha usado un programa parecido al del cálculo del orden de los métodos que se han programado en este TFG, pero a diferencia del comportamiento de estos, en este caso salen gráficas horizontales.

La razón de este comportamiento es que los Solvers que implementa Matlab tienen una estrategia de cambio de paso en la que mantiene constante un cierto error. Ese error se encuentra programado por defecto, y en base a las gráficas, nos podemos hacer una idea de cómo es de preciso el método.

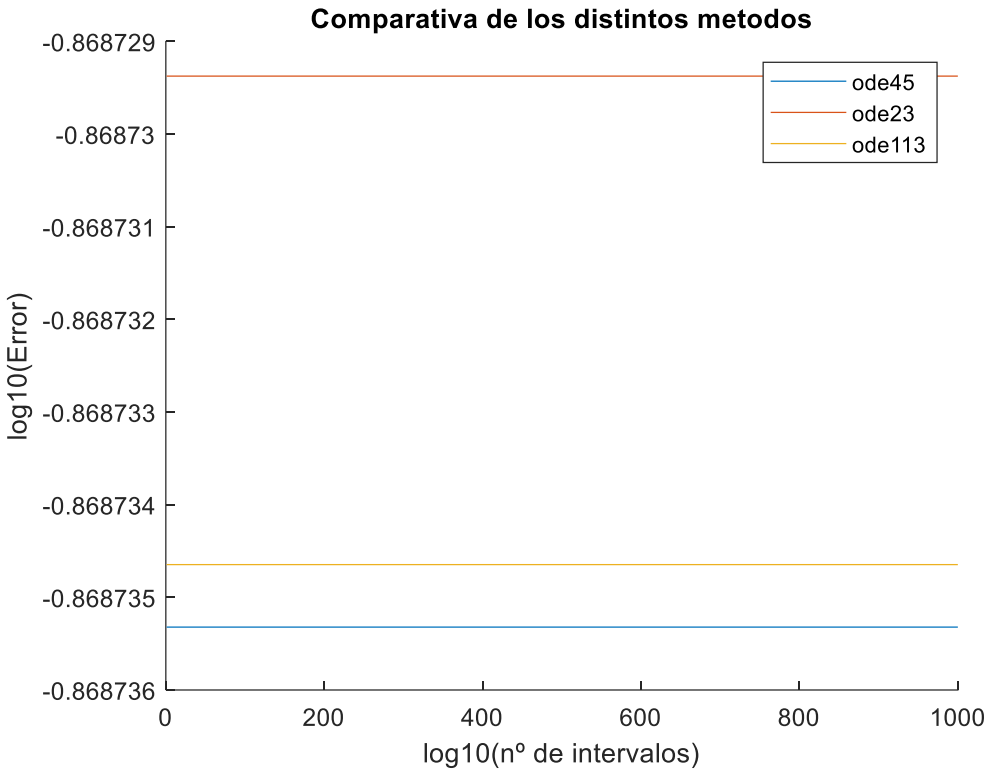


Figura 5-1 Comparativa errores en los solver (I)

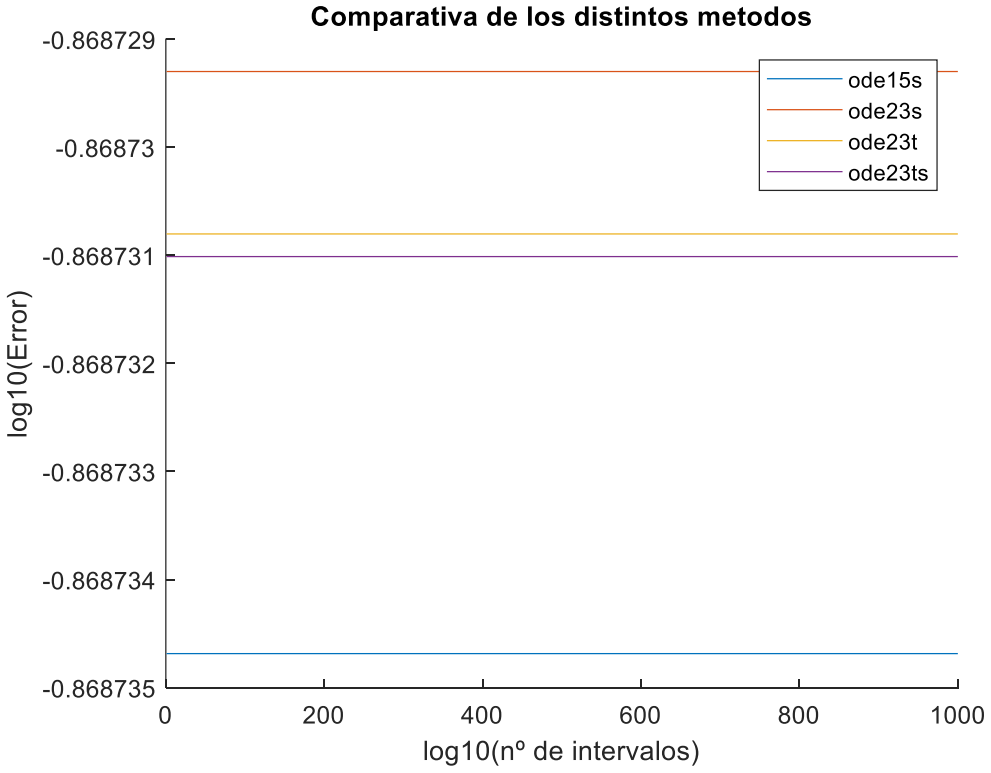


Figura 5-2 Comparativa errores en los solvers (II)

5.2. FUNCIONAMIENTO DE UN SOLVER IMPLEMENTADO EN MATLAB (ode45)

```
[x,y] = ode45(odefun,tspan,y0)
```

Es un solver pensado para resolver sistemas de ecuaciones diferenciales de la forma $y' = f(x, y)$.

Todas las sintaxis de los solvers tienen una estructura similar a fin de que se tenga una forma de trabajo estándar y sea fácil cambiar de uno a otro.

Típicamente esta función tiene tres parámetros de entrada.

`odefun`

En esta definición de función anónima integramos la ecuación diferencial que queremos resolver como $@(x, y)$, es decir, una función de dos variables, ya que trabajamos con ecuaciones diferenciales ordinarias

`tspan`

Es un vector que indica las posiciones inicial y final del intervalo. El método de los solver de Matlab no precisa de más información, ya que se determina el intervalo de manera automática para minimizar el error.

De manera ocasional, este vector puede contener información acerca de los puntos de interés, pero en la práctica lo que hace el método es calcular como siempre y dar la solución únicamente en los puntos de entrada.

`y0`

Es el valor inicial de la función en el primer punto del intervalo de interés.

La salida de este método es una sucesión de puntos $[x, y]$ que nos da los valores aproximados de la función en el intervalo de interés. Se trata, por lo general, de un método muy eficiente.

6. EXPERIMENTOS NUMÉRICOS

En estos experimentos, probamos los distintos métodos con varias funciones de las que conocemos su solución analítica y estableceremos una comparación entre unos y otros.

No se puede aventurar que uno sea mejor que los demás porque hay ocasiones en que un método es menos eficiente que otro y, sin embargo, con otras ecuaciones distintas, se invierte la idoneidad.

Simplemente, es para constatar los distintos tipos de funcionamiento que tienen estos métodos y compararlos con el que se ha desarrollado de Splines.

6.1. EXPERIMENTO 1: Orden de los métodos

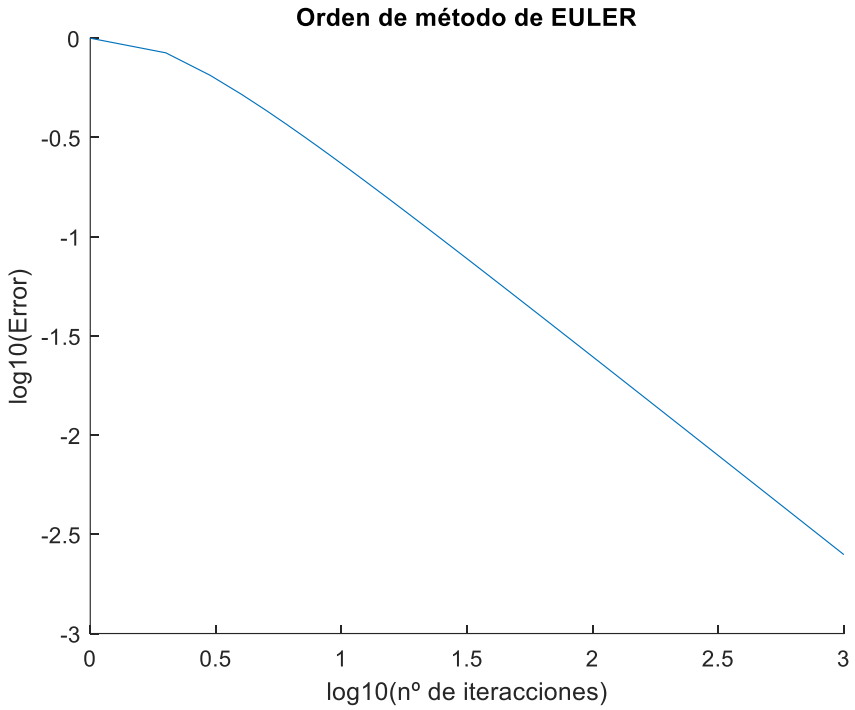


Figura 6-1 Orden método de Euler

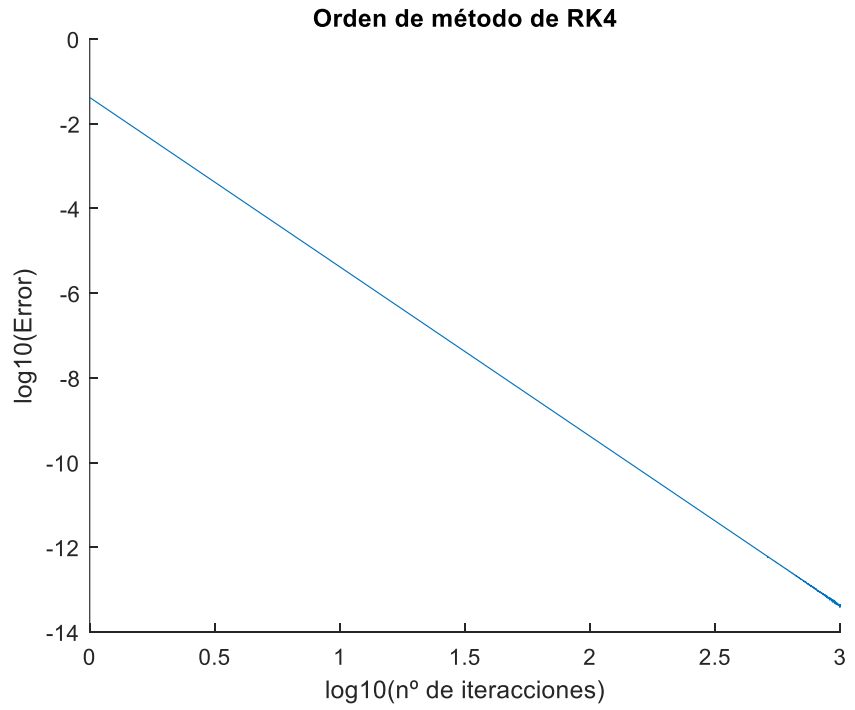


Figura 6-2 Orden Método de Runge Kutta

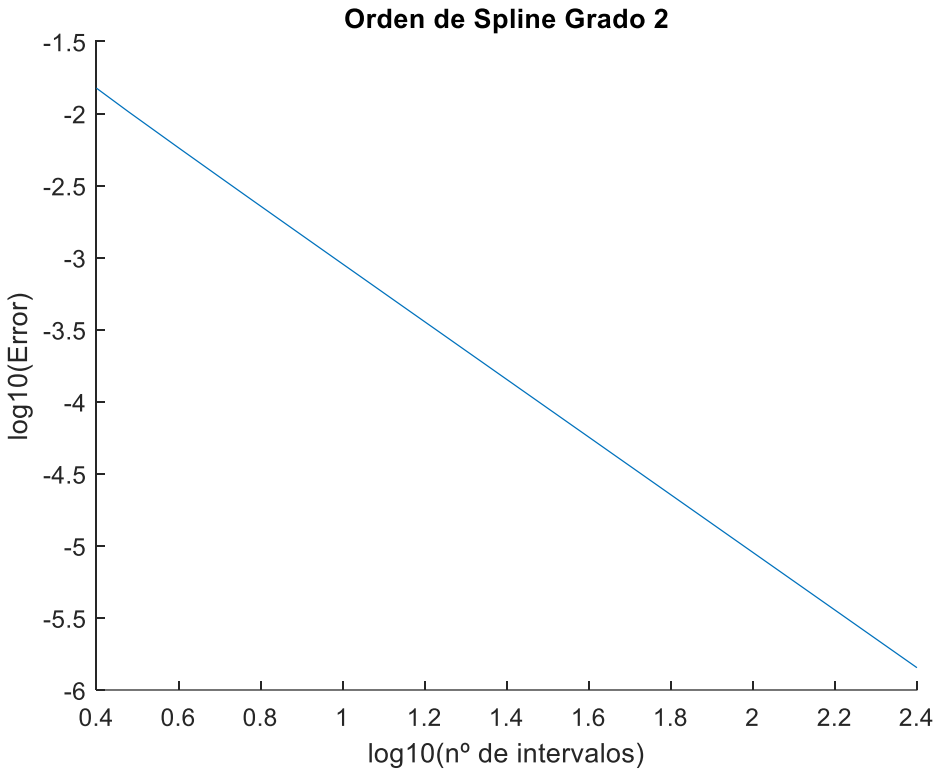


Figura 6-3 Orden Splines grado 2

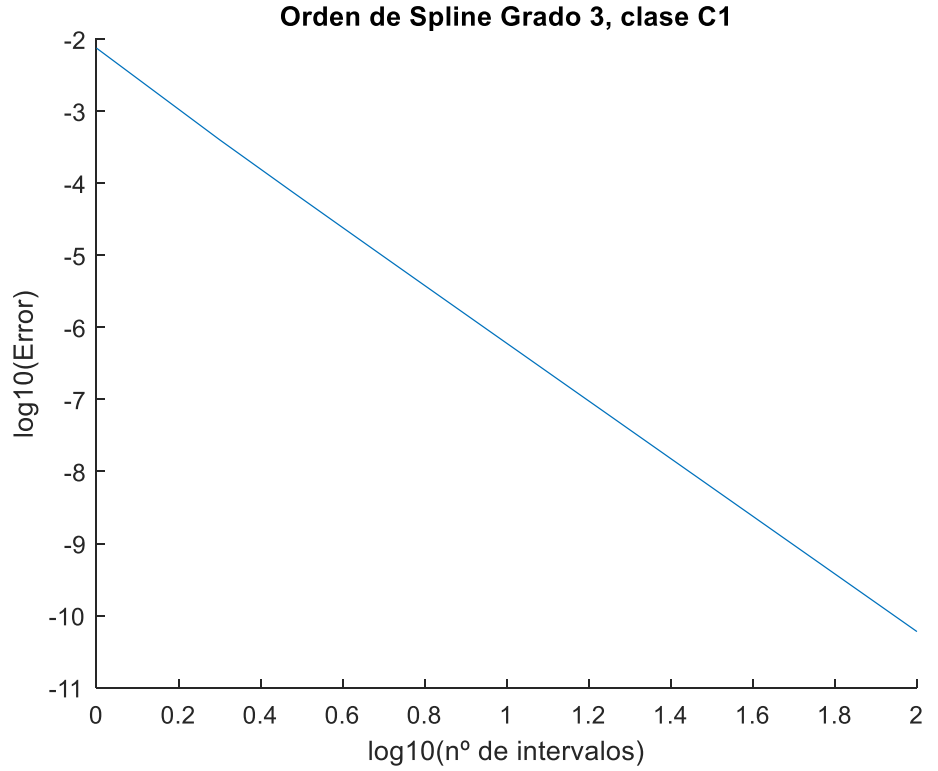


Figura 6-4 Orden Pseudo-Splines grado 3

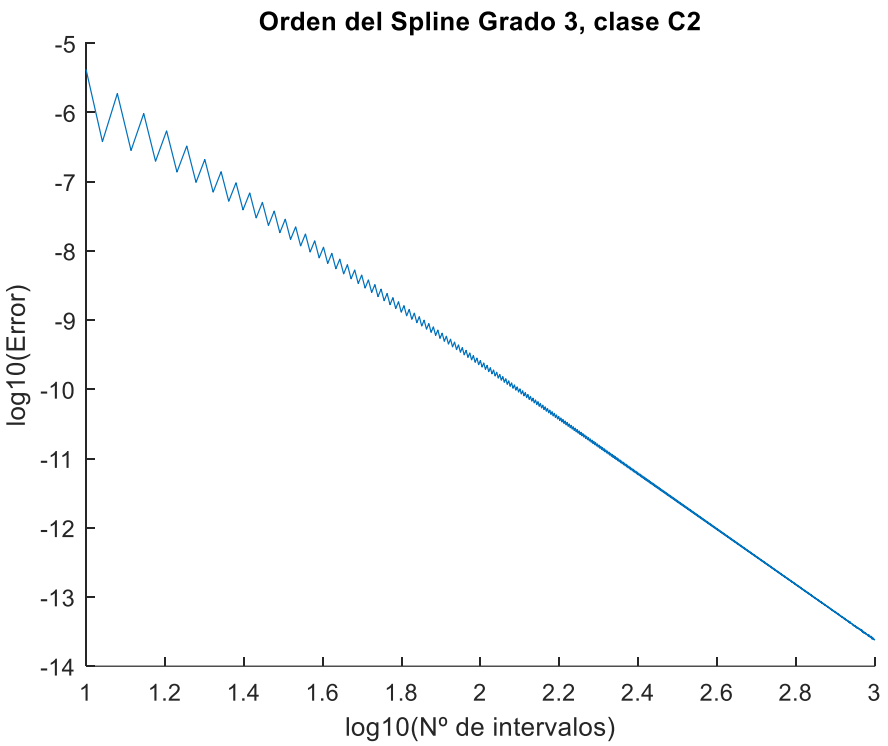


Figura 6-5 Orden Spline grado 3, clase C2

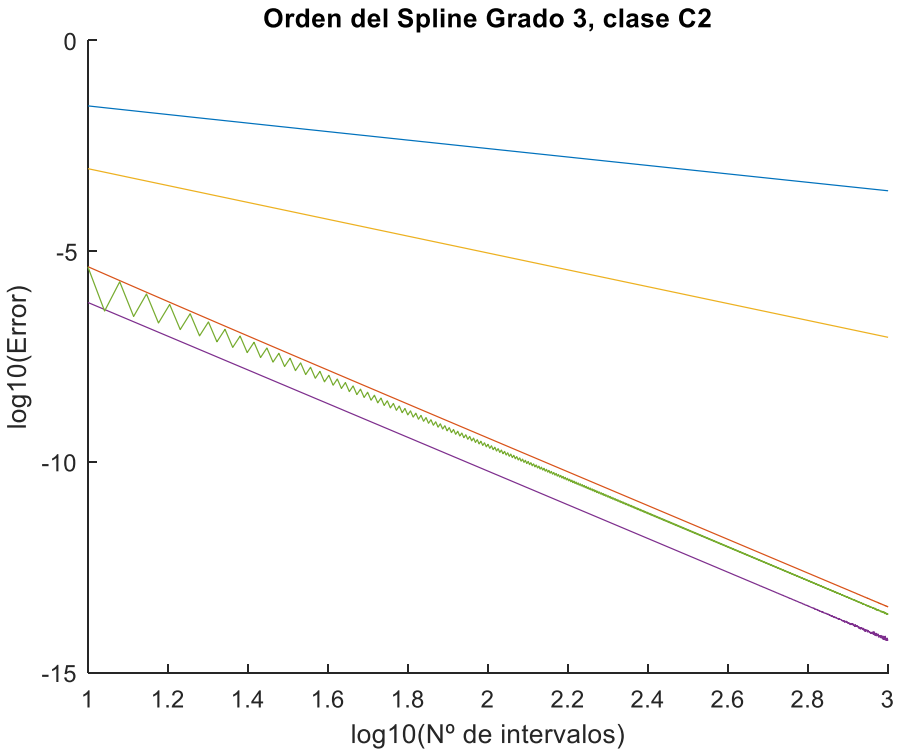


Figura 6-6 Comparativa ordenes

Obtenemos la pendiente, que representa el orden de los métodos como

$$p = -\frac{\Delta \log_{10} Error}{\Delta \log_{10} nº\ Iteracciones}$$

Ecuación 6-1 Cálculo del orden p

De la evaluación de la pendiente en las tablas tenemos:

Tabla 6-1 Comparación de órdenes de los distintos métodos

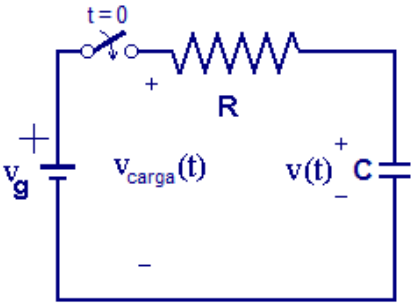
Método	Orden
Euler	1
RK4	4
Splines gado 2	2
PseudoSplines grado 3 (Clase C1)	4
Splines grado 3 (clase C2)	4

En la figura 6 1 se puede observar una parte distinta al inicio. Esto corresponde a una zona de pasos más grandes de lo que pide la función. Aunque no se ha tenido en cuenta para calcular el orden, es muy ilustrativo de que todos los métodos tienen sus limitaciones y su rango de aplicabilidad. Vemos que el método de Euler requiere unos tamaños de paso más pequeños que el método basado en PseudoSpline.

La figura 6 5 correspondiente a la evaluación del orden del método basado en Splines de grado 3 y de clase C2, presenta unos picos y no es tan lineal como las otras. Realmente estos pequeños picos no afectan al orden, ya que el método satisface las condiciones de orden 4. La explicación de estos picos tiene que ver con la imposición de la segunda derivada al principio de cada Spline. Esta imposición fuerza un poco al Spline a tener un comportamiento más oscilante en torno al punto medio. Si bien es cierto que, por ser de grado menor que cuatro sigue siendo estable, se observa este fenómeno oscilatorio.

6.2. EXPERIMENTO 2: Carga de un condensador. Comparativa con solvers

Retomando la explicación del capítulo 2



$$\frac{dv(t)}{dt} + \frac{1}{RC}v(t) = \frac{V_g}{RC}$$
$$v(0) = v_0$$

Figura 6-7: Proceso de carga de un condensador
Fuente: Aulamoisan

Tenemos un condensador el cual, en un instante dado, se conecta a una fuente de tensión a través de una resistencia. El condensador se encuentra inicialmente descargado.

Para calcular la larga debemos resolver la siguiente PVI:

$$\begin{cases} \frac{d\,v(t)}{dt} + \frac{1}{\mathrm{R}\cdot\mathrm{C}}\cdot \mathrm{v(t)} = \frac{V_g}{R\cdot C} \\ V(0) = 0 \end{cases}$$

Ecuación 6-2: EDO de carga de un condensador ideal

Los parámetros del sistema son:

$V_g = 10\,V$

$R = 2\,\Omega$

$C = 0.8\ F$

$intervalo\ de\ interés = [0, 10]\ segundos$

(Parra, 2020)

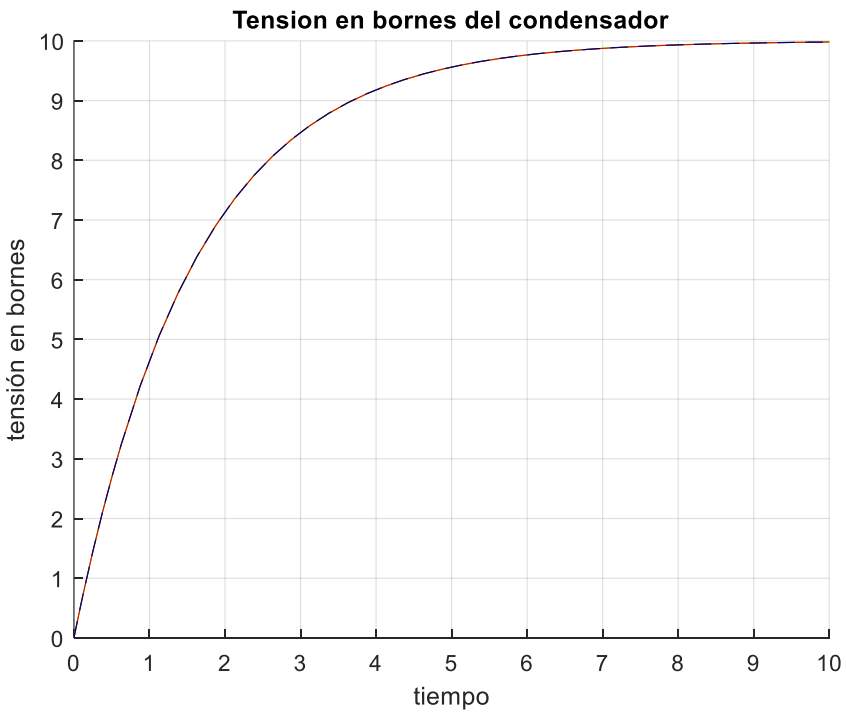


Figura 6-8: Gráfica V-t condensador
Aproximación mediante ode45 (rojo) y Spline de grado 2 (azul discontinuo) para un número de pasos de 69.

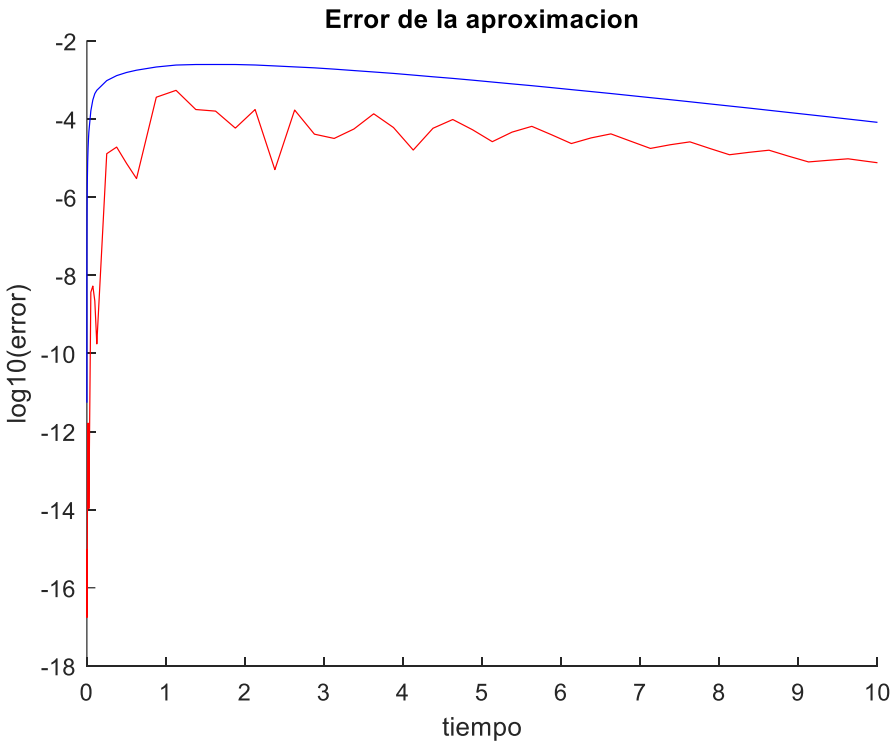


Figura 6-9: Error de las aproximaciones ode45 y Spline2
Error de la aproximación del ode45 (en rojo) / Spline de grado 2 (azul)

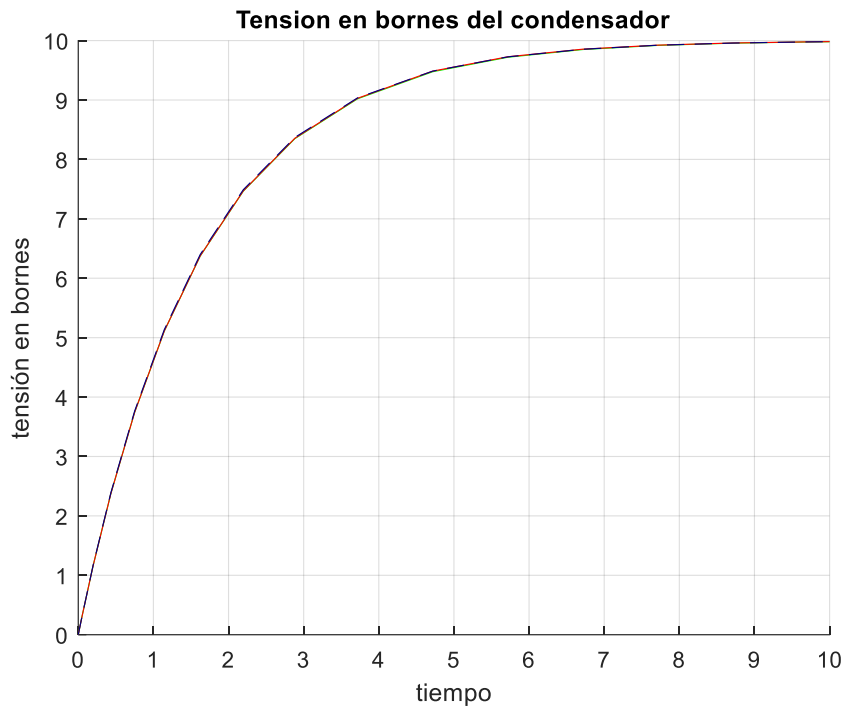


Figura 6-10 Gráfica V-t con ode23

Aproximación mediante ode23 (rojo) y Spline de grado 2 (azul discontinuo)
Para un número de pasos de 22

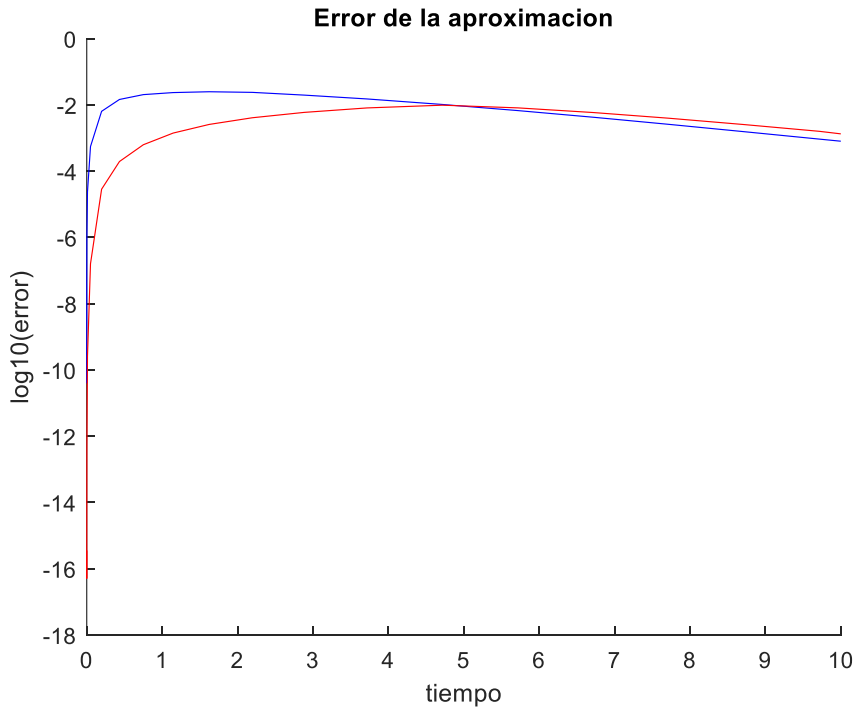


Figura 6-11: Error

Error de la aproximación del ode23 (en rojo) y Spline de grado 2 (azul)

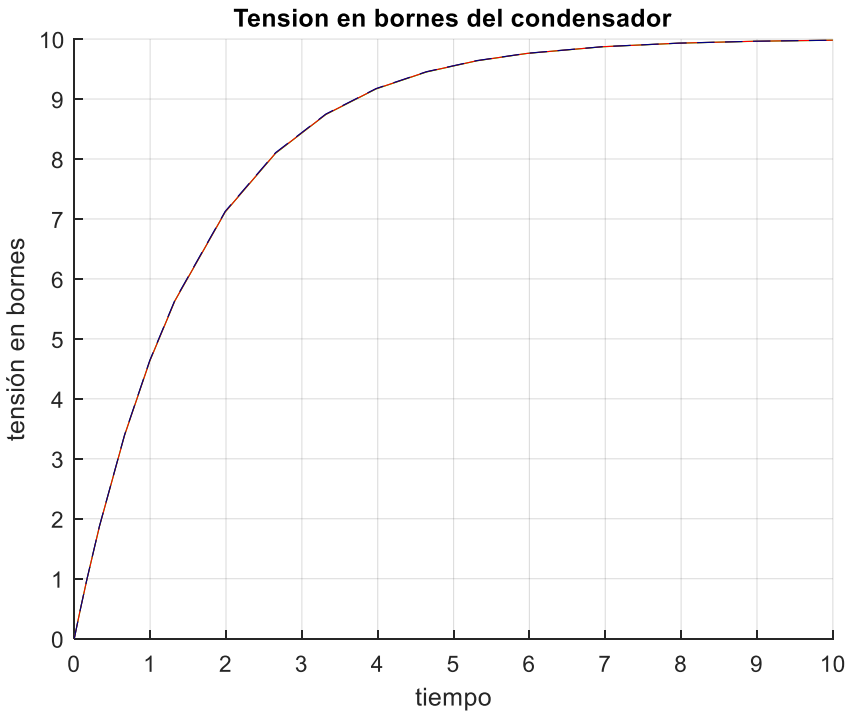


Figura 6-12: Tensión en bornes

Aproximación mediante ode113 (rojo) y Spline de grado 2 (azul discontinuo)
Para un número de pasos de 32

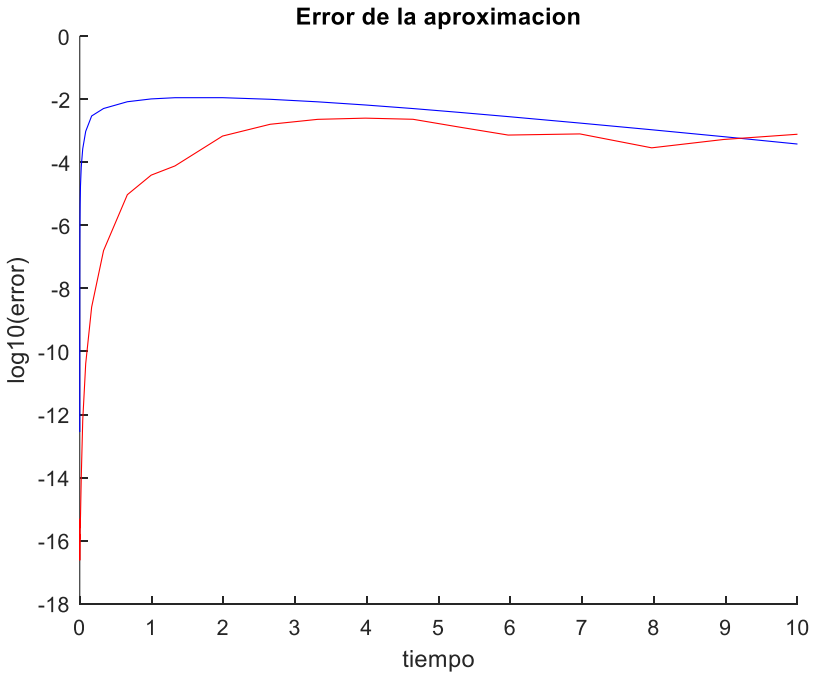


Figura 6-13: Error de aproximación

Error de la aproximación del ode23 (en rojo) y Spline de grado 2 (azul)

6.3. EXPERIMENTO 3: Splines G2 Vs Splines G3

El propósito de este experimento es el de comparar los métodos básicos de Splines con 2 puntos de colocación por cada intervalo y su mejora natural a 3 puntos de colocación por cada intervalo.

Como se ha visto, este método lo que consigue es, a partir de un punto y de la pendiente calculada en los demás, trazar el polinomio interpolador a trozos que aproxima el comportamiento de la función.

Vamos a comparar el método básico basado en Splines de grado 2 con el de Splines de grado 3. La filosofía de trabajo es la misma, sólo que el de grado 3 utiliza más información del campo de pendientes en el nodo intermedio del subintervalo.

Vamos a trabajar ahora con las siguientes funciones de prueba:

$$y = \frac{5}{7} \cdot e^{-7 \cdot \frac{x^2}{2}} + \frac{2}{7}$$

Esta función es la solución de la Ecuación diferencial

$$\begin{cases} y'(x) = -4 \cdot x \cdot y(x) + 8 \cdot x \\ y(0) = 1 \end{cases}$$

Esta función presenta características del comportamiento dinámico de los sistemas de primer orden en ingeniería. Tenemos una pendiente al principio del intervalo, que tiende a un valor estable de la función al final. Esto corresponde con multitud de sistemas de cualquier tipo.

$$y(x) = x + 3 \cdot e^{-x}$$

Esta función es la solución de la Ecuación diferencial

$$\begin{cases} y'(x) = -y(x) + x + 1 \\ y(0) = 1 \end{cases}$$

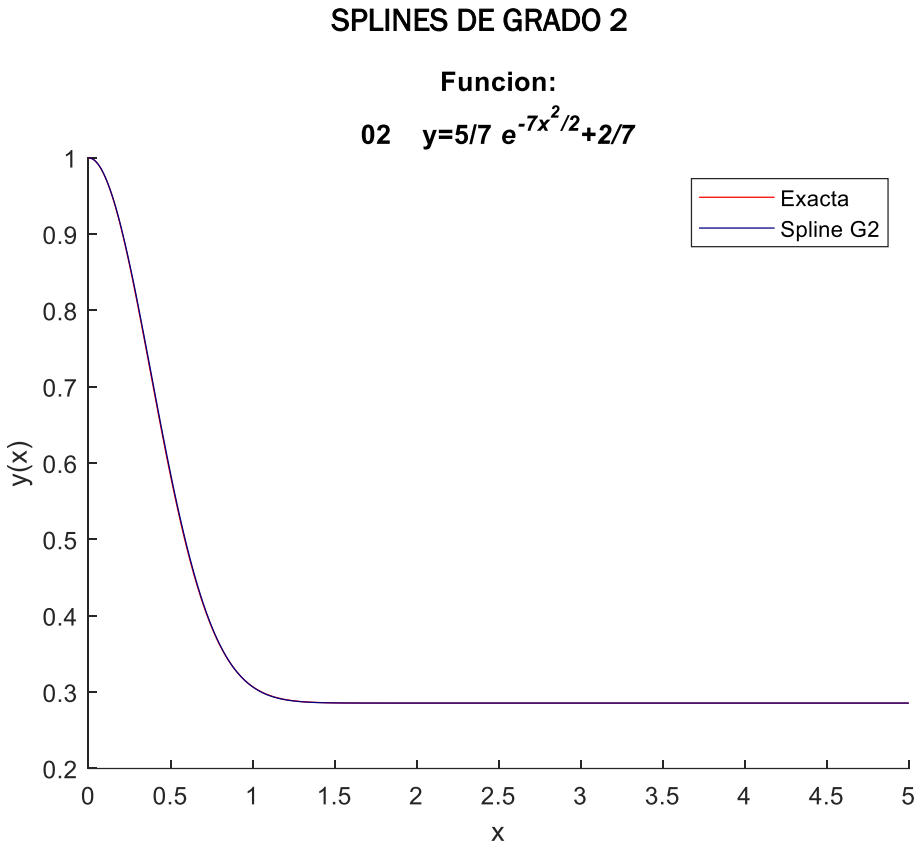


Figura 6-14 Spline de grado 2

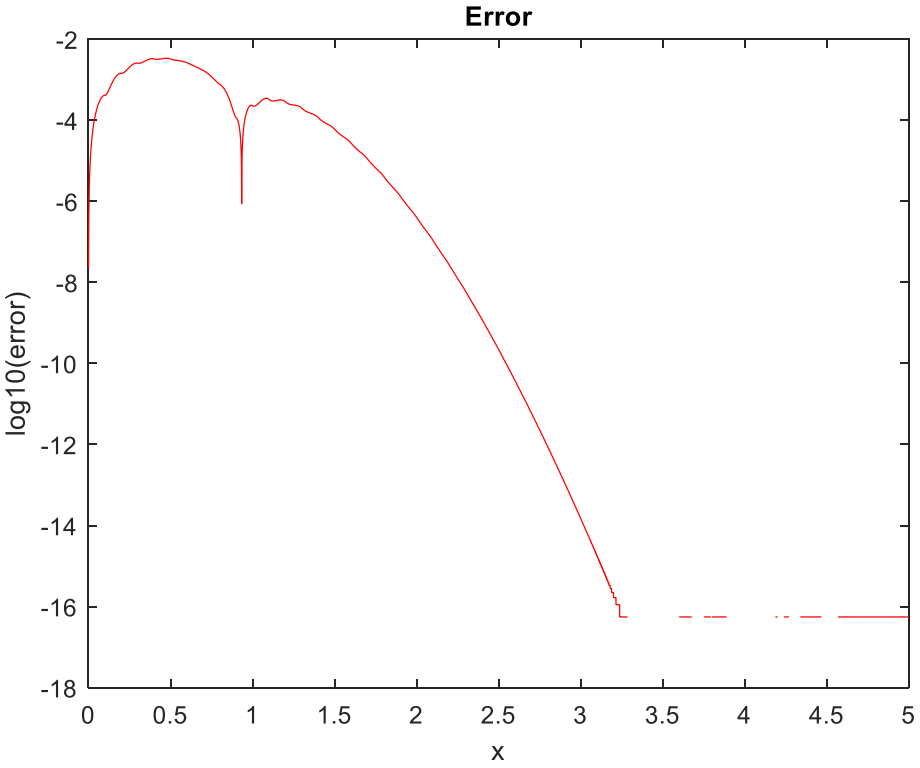


Figura 6-15: Error en la aproximación grado 2

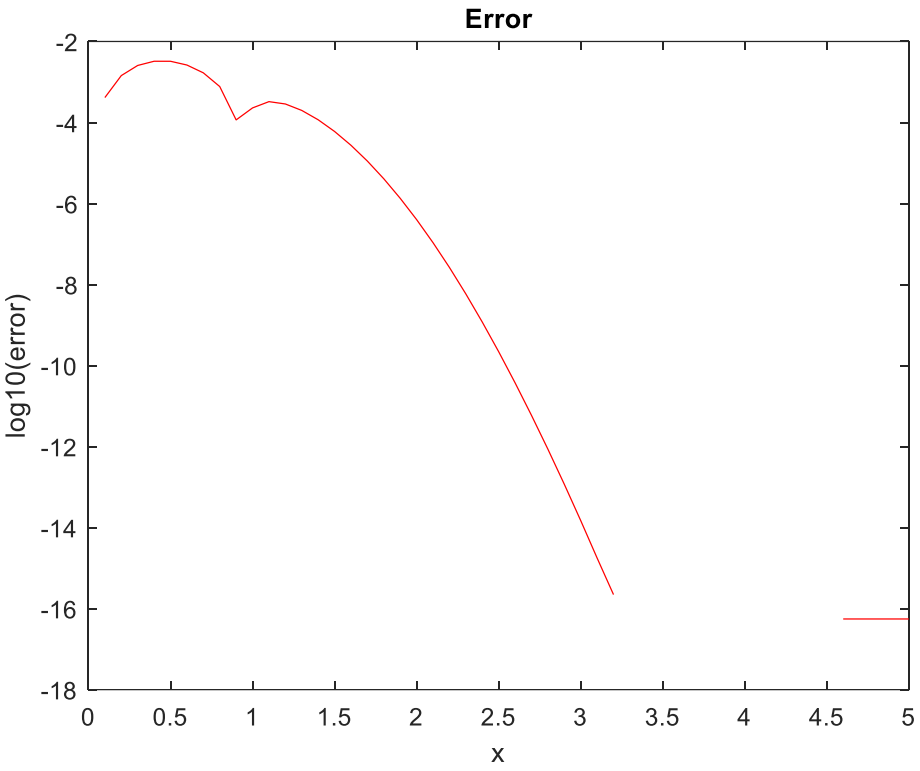


Figura 6-16: Error en los nodos

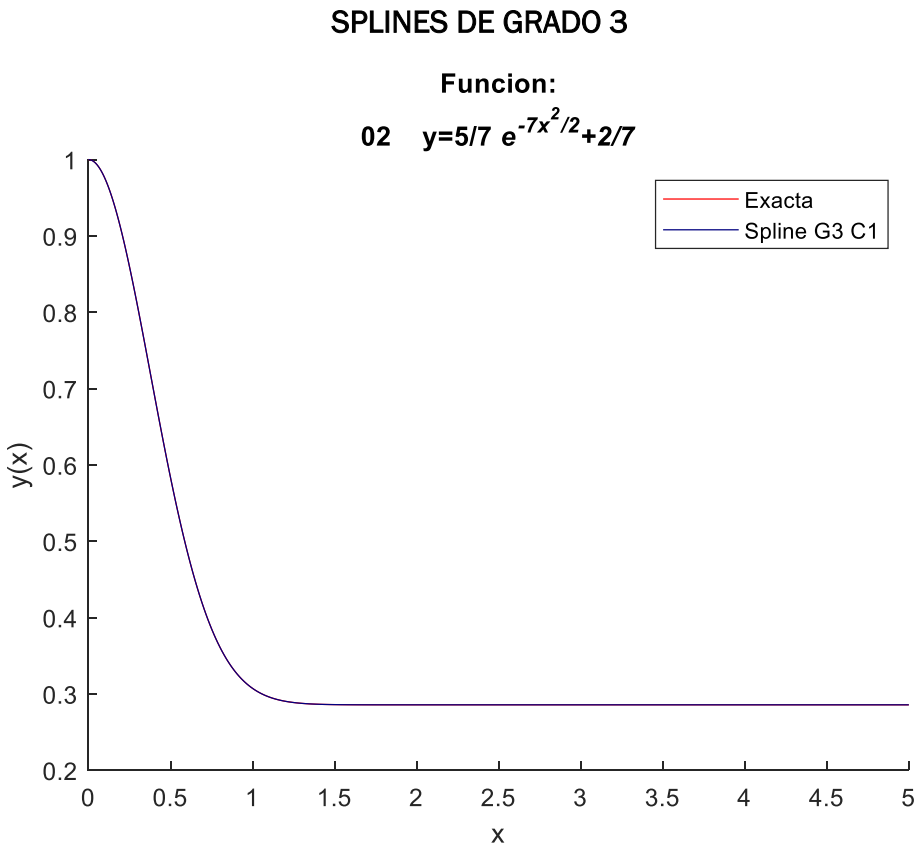


Figura 6-17: Spline de grado 3

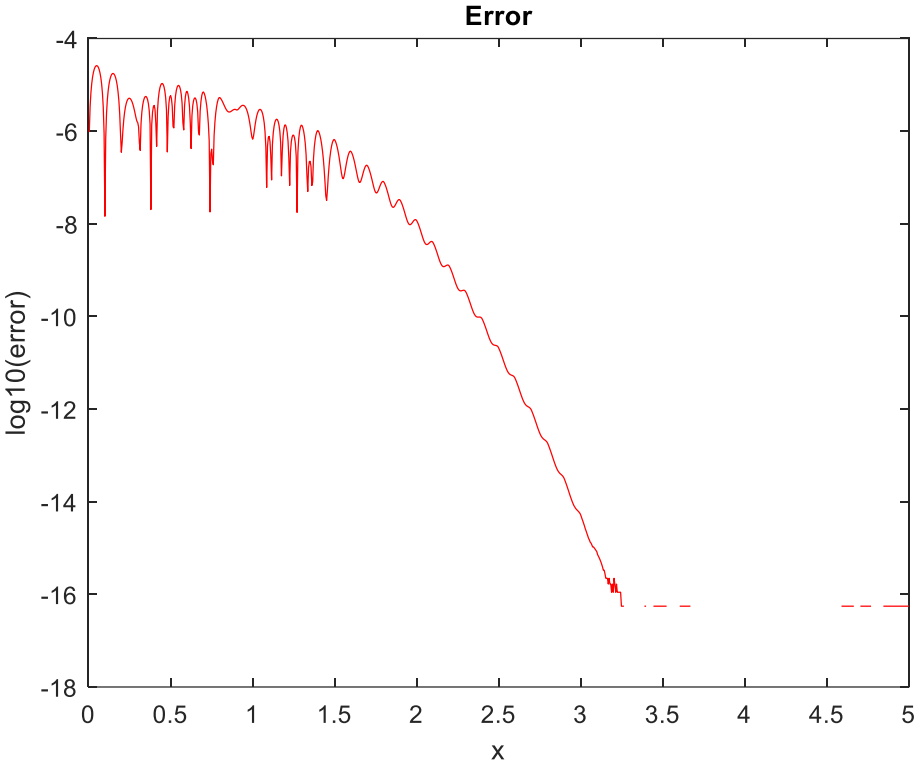


Figura 6-18: Error

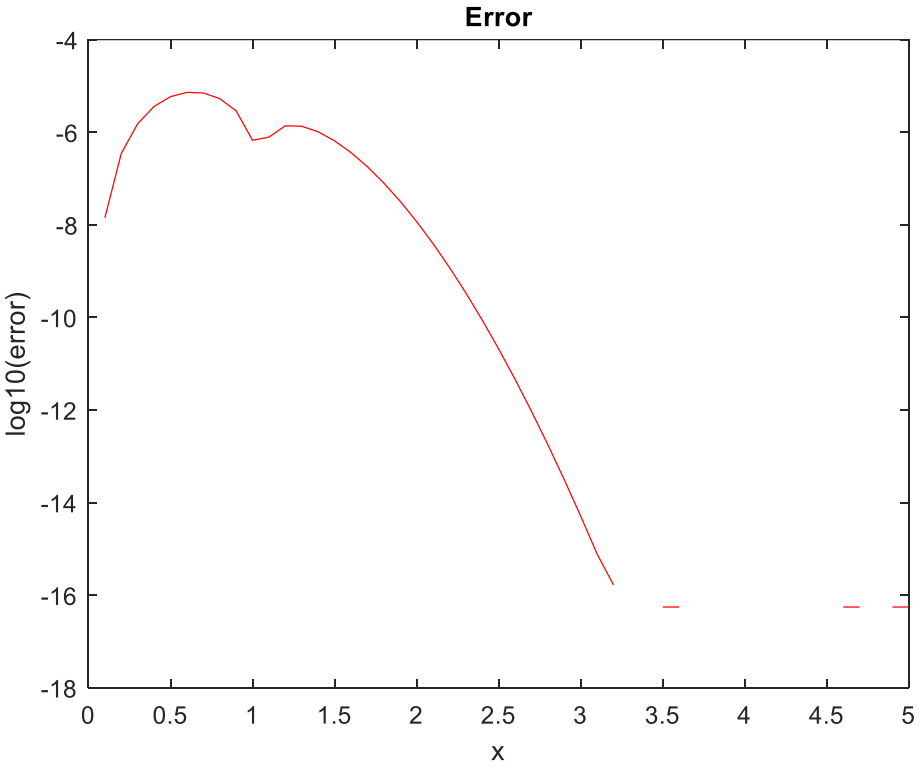


Figura 6-19: Error en los nodos

Como se aprecia, llega un momento en que el error está en torno a 10^{-16}. Con doble precisión el ordenador no es capaz de ofrecer resultados más precisos. Esto da cuenta de lo mucho que mejora el método en cuanto se usa de manera adecuada la información disponible

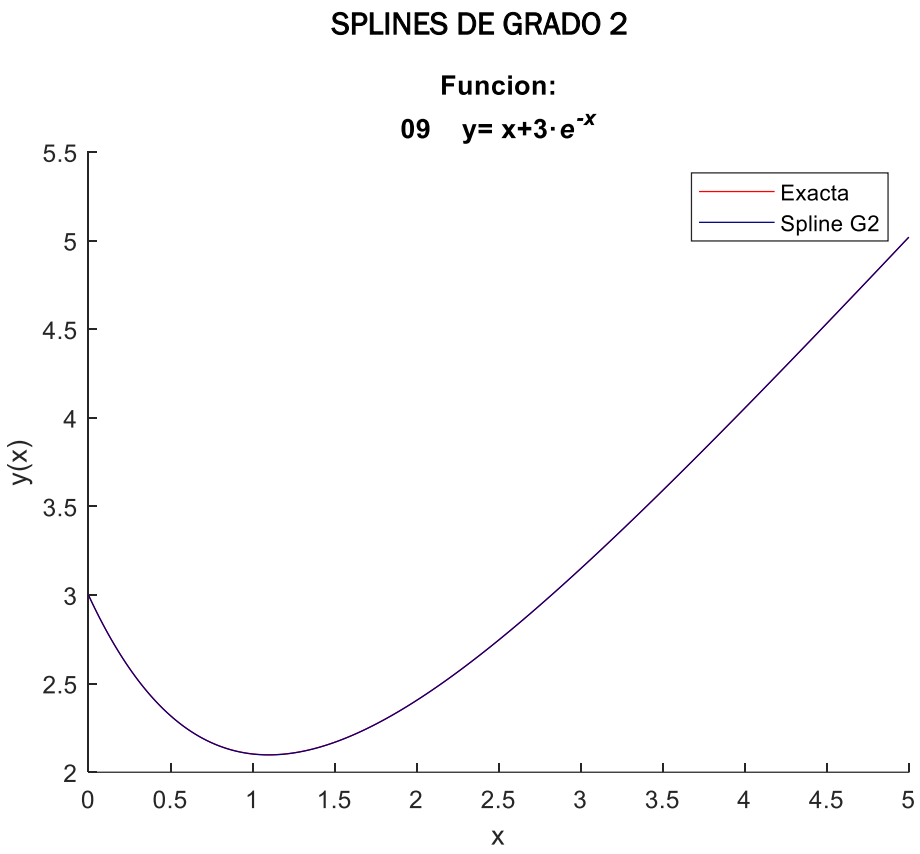


Figura 6-20: Spline grado 2

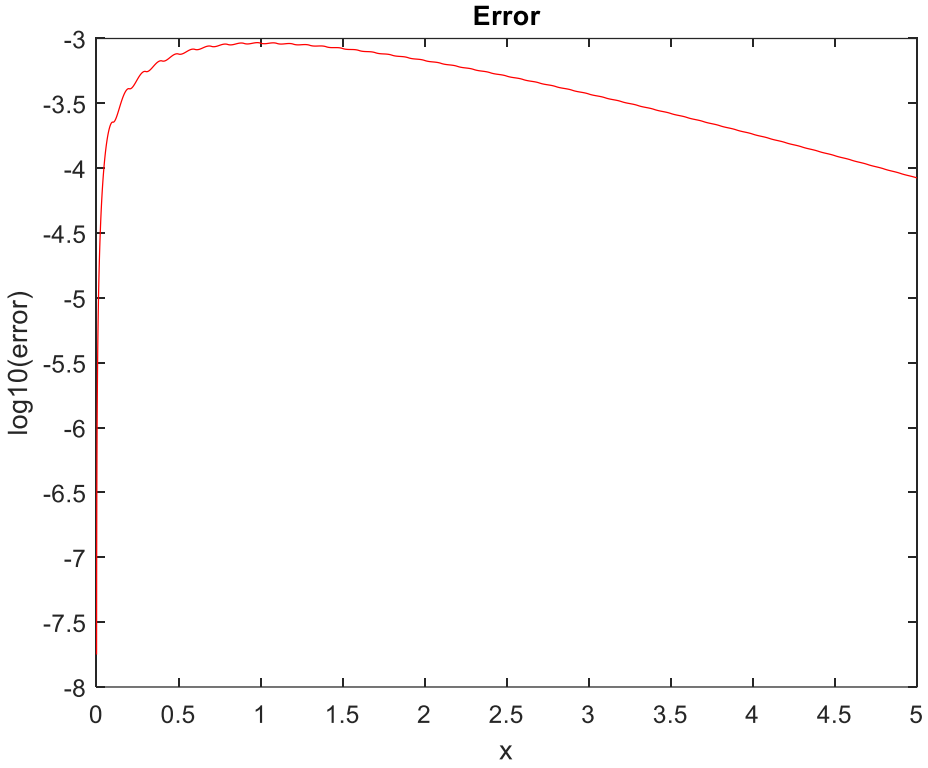


Figura 6-21: Errores

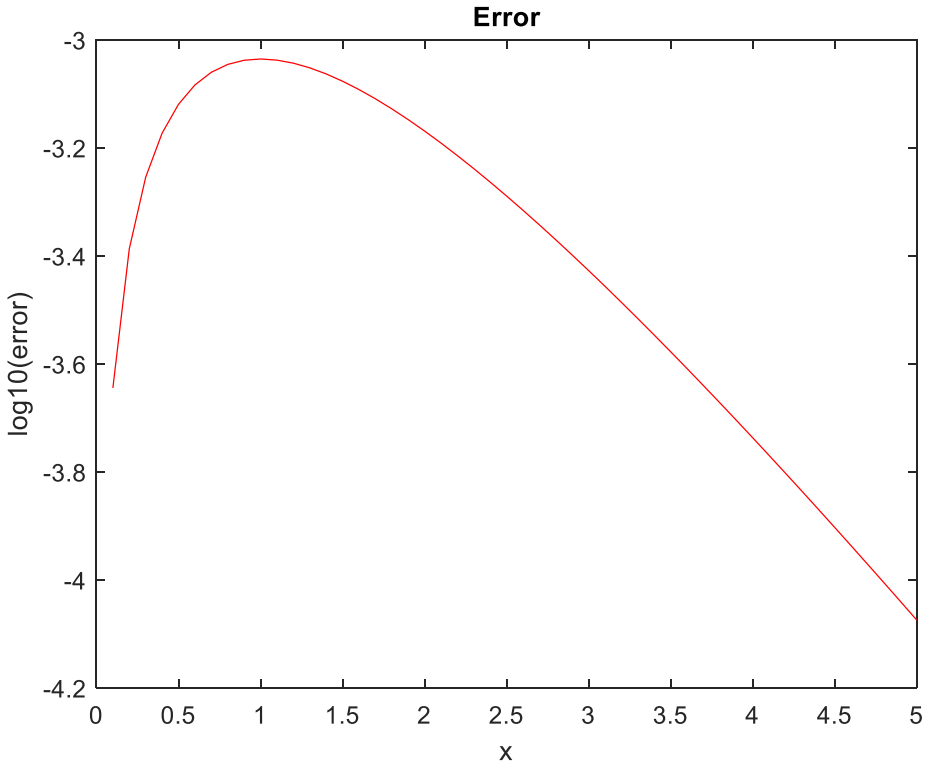


Figura 6-22: Error en los nodos

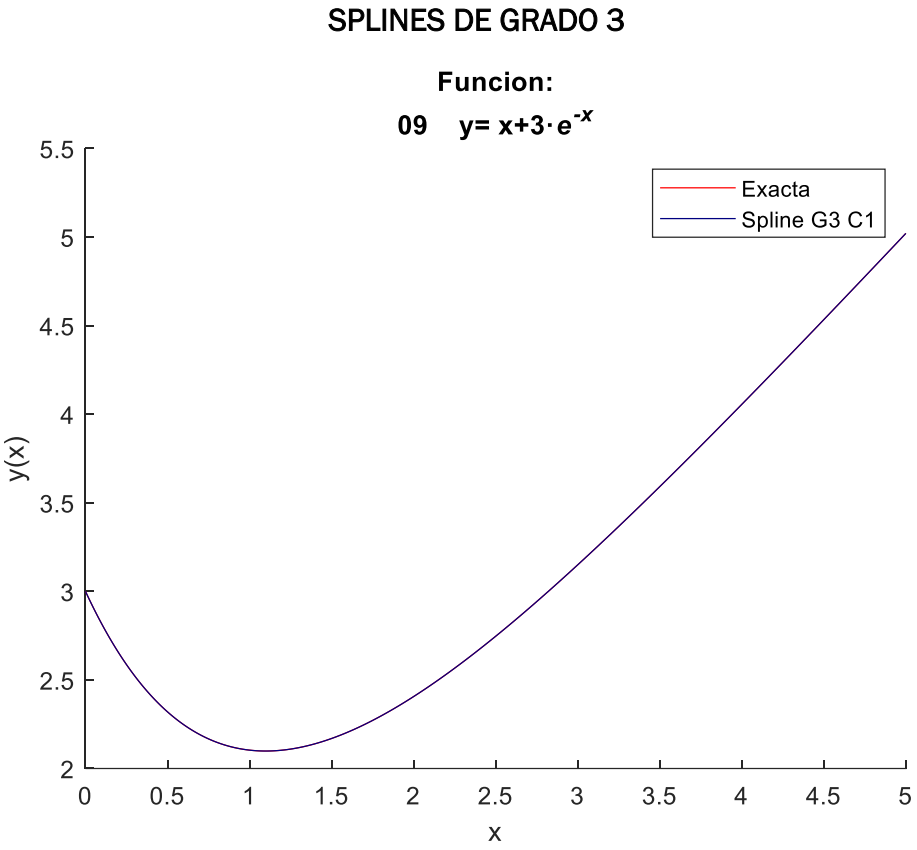


Figura 6-23: Spline G3

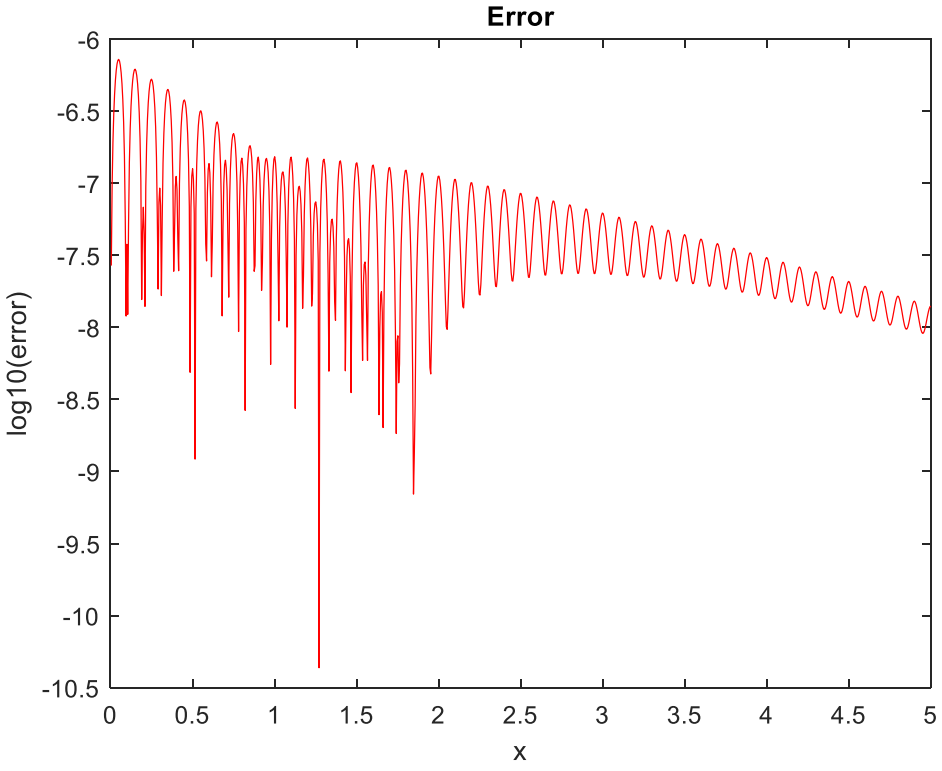


Figura 6-24: Error Spline G3

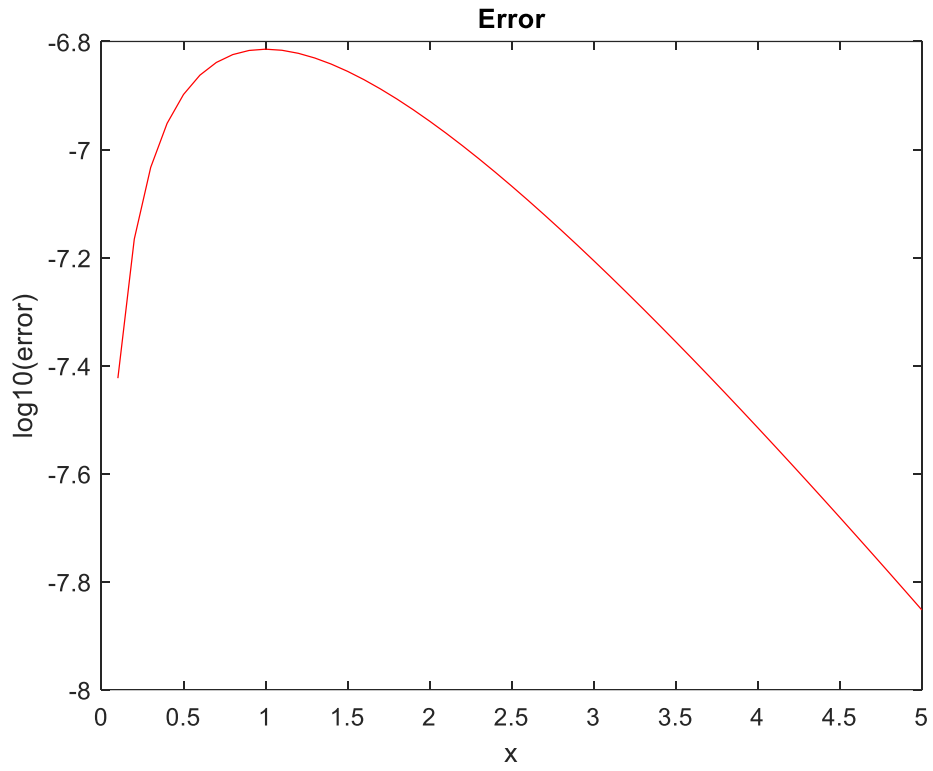


Figura 6-25: Error en los nodos

Tabla 6-2: Experimento SplineG2 y Spline G3

TABLA RESUMEN			
	$y = \frac{5}{7} \cdot e^{-7 \cdot \frac{x^2}{2}} + \frac{2}{7}$		
	Nº de pasos	$E_{máx}$	E_{medio}
Spline G2	50	0,0034	$3.4634 \cdot 10^{-4}$
Spline G3	50	$2,6017 \cdot 10^{-5}$	$1,1565 \cdot 10^{-6}$
	$y = x + 3 \cdot e^{-x}$		
	Nº de pasos	E_{max}	E_{medio}
Spline G2	50	$4,7977 \cdot 10^{-4}$	$4,7977 \cdot 10^{-4}$
Spline G3	50	$7,2390 \cdot 10^{-7}$	$7,2763 \cdot 10^{-8}$

Vemos dos fenómenos interesantes. El primero es que, aumentando tan sólo un punto de colocación se disminuye mucho el error. No hay que olvidar que, en base al primer experimento, se trata de un método de orden 4 frente a uno de orden 2.

El segundo fenómeno está en el error cometido. Normalmente los métodos numéricos comprueban el error en los nodos en los que realizan la

aproximación, como se hará en los sucesivos desarrollos. Sin embargo, las gráficas correspondientes a los errores están realizadas sobre una base de 1000 puntos en el intervalo, es decir, tomando unos puntos equidistantes. Con esto vemos una progresión del error que forma unas curvas en función del grado de los Splines.

Los puntos de mejor aproximación, corresponden con los extremos de los subintervalos, en los cuales imponemos la condición de colocación, quedando los puntos interiores algo más separados de la curva. Debido a la naturaleza de los polinomios, estos errores toman una distribución en arcos

6.4. EXPERIMENTO 4: Paso fijo Vs Paso Variable.

Vamos a continuar con las funciones del experimento anterior para poder comparar los resultados y probaremos los tres tipos de estrategia de cambio de paso.

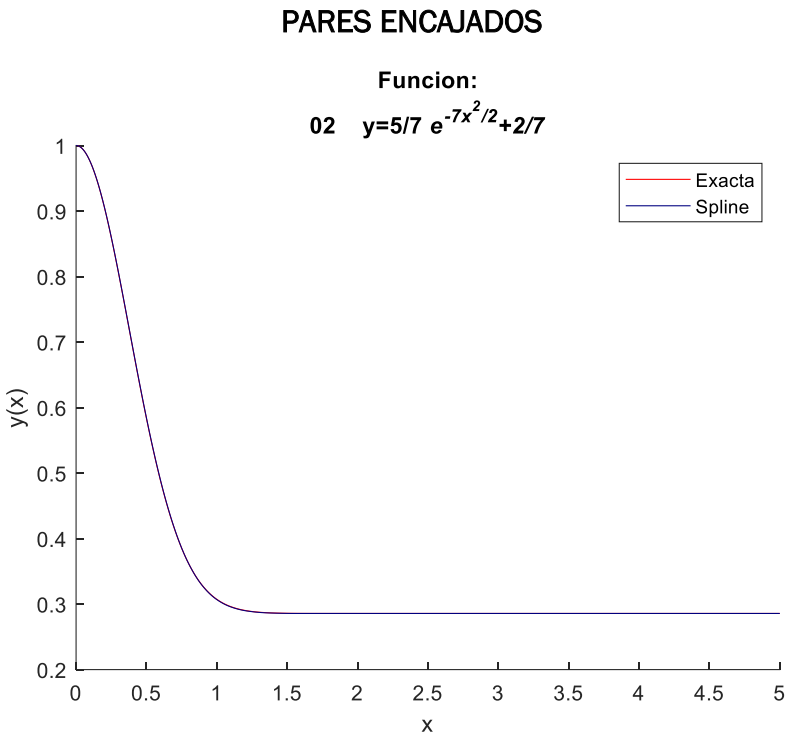


Figura 6-26: Pares encajados

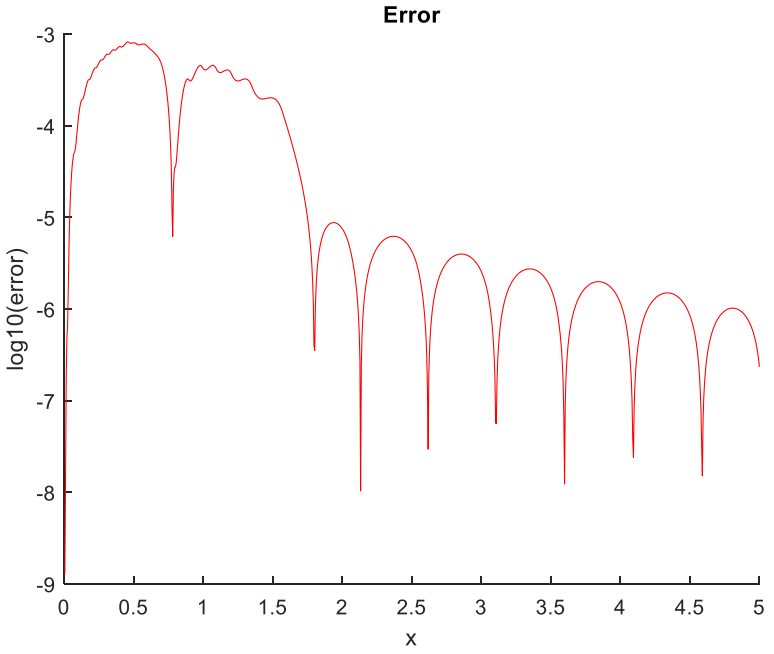


Figura 6-27: Error pares encajados

RICHARDSON GRADO 2

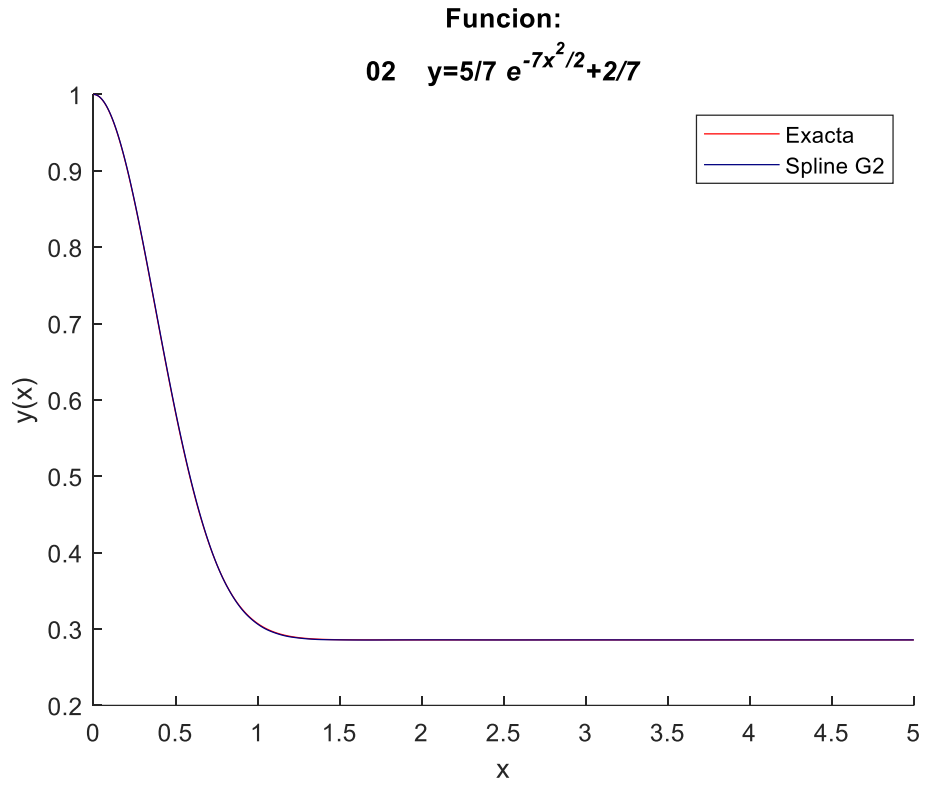
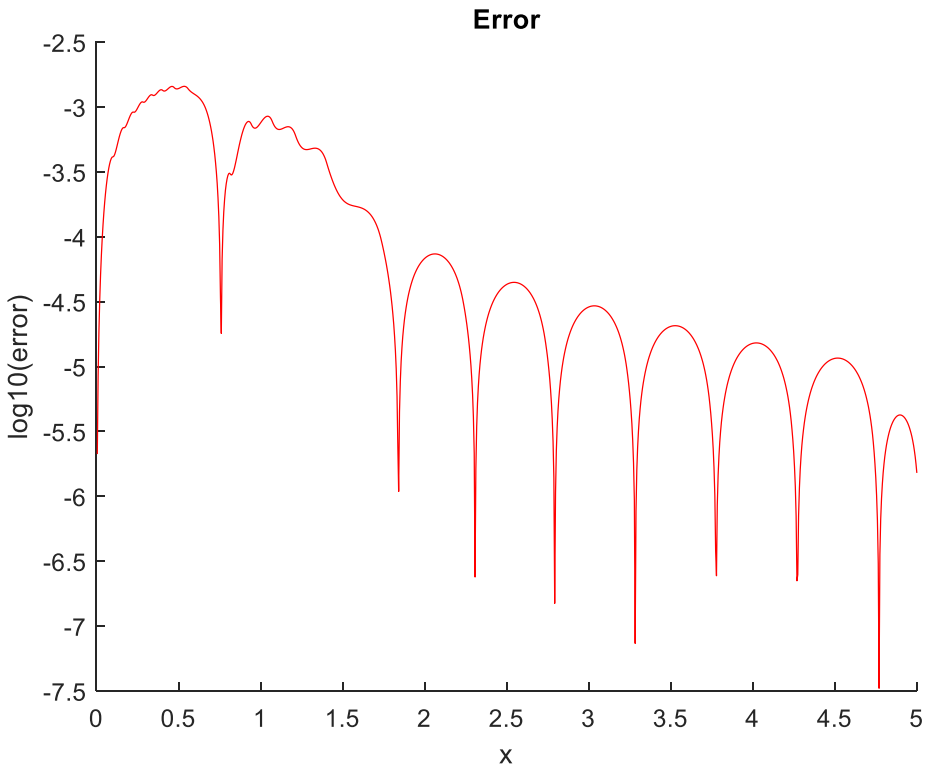


Figura 6-28: Richardson grado 2



RICHARDSON GRADO 3

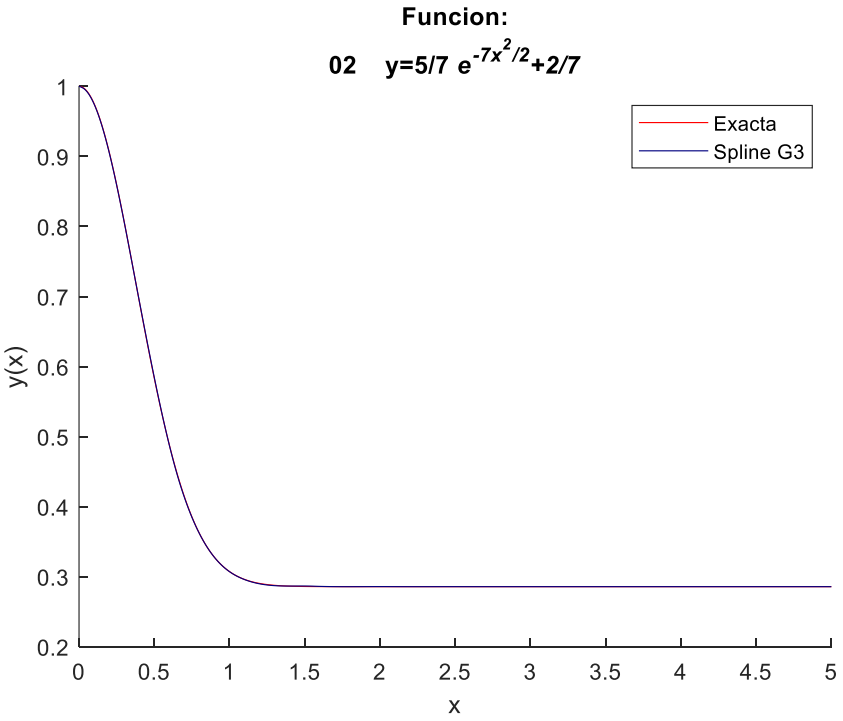


Figura 6-29: Richardson grado 3

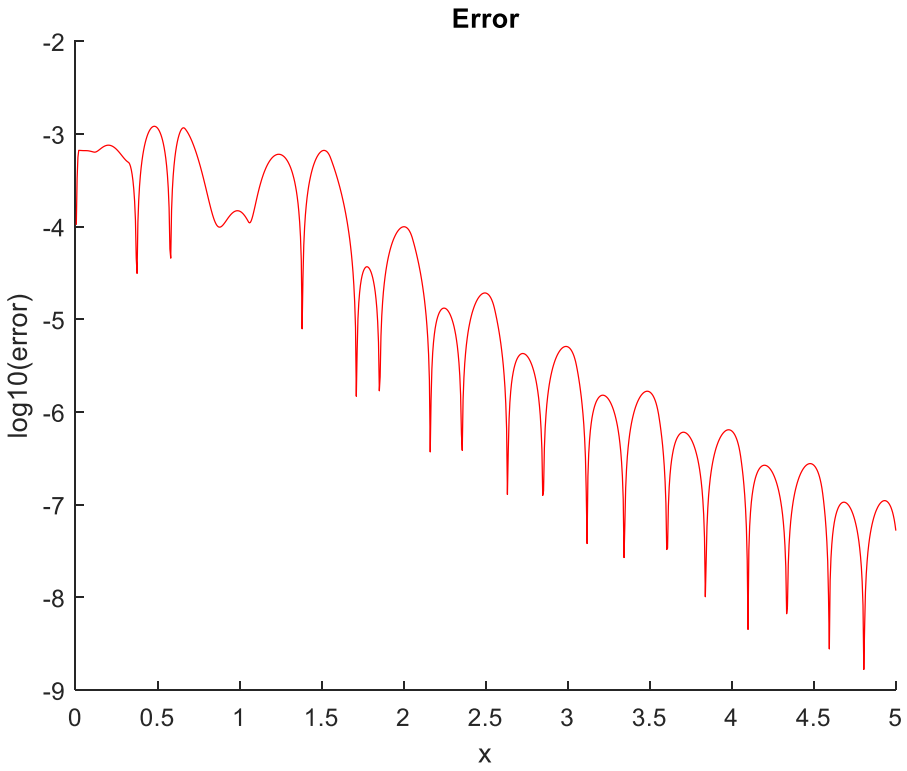


Figura 6-30: Error

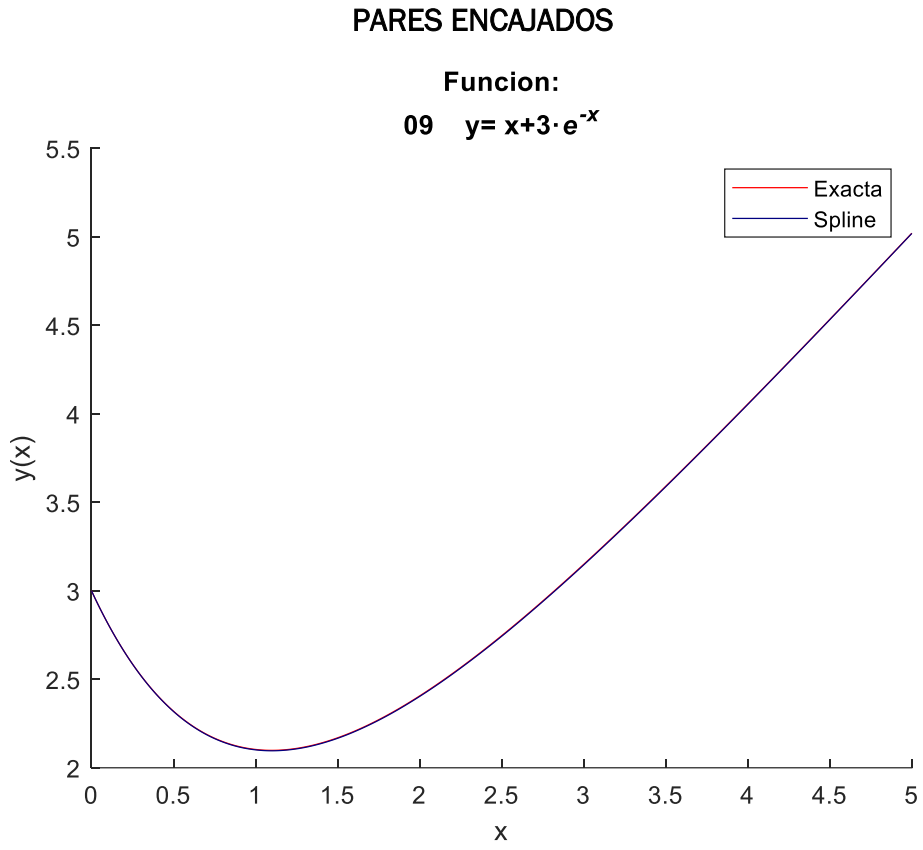


Figura 6-31: Pares encajados

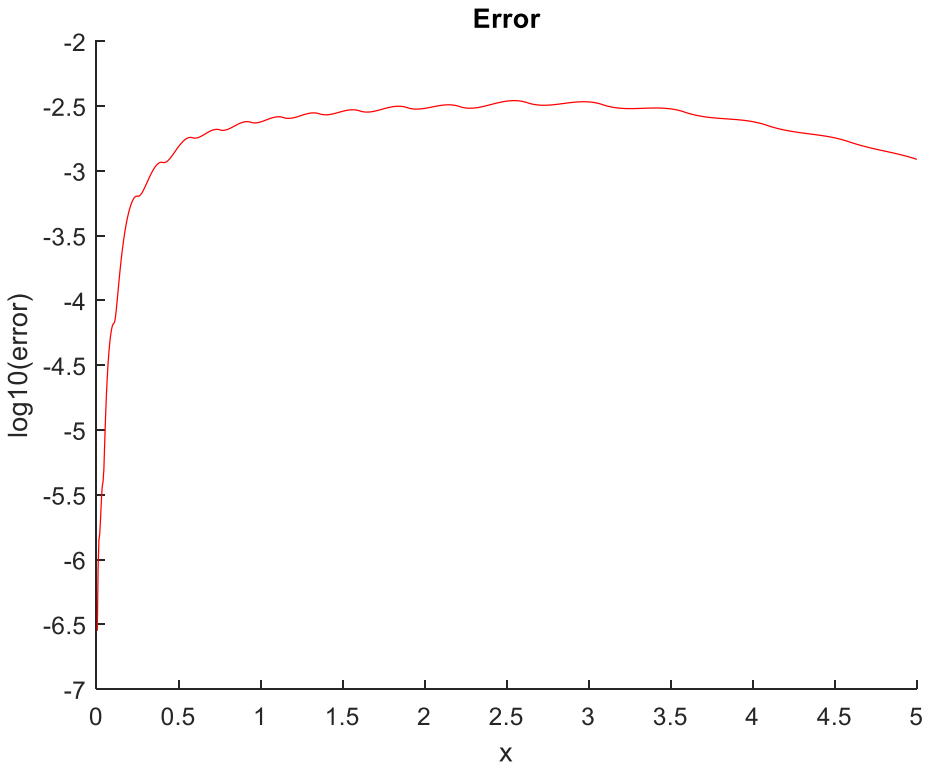


Figura 6-32: Error

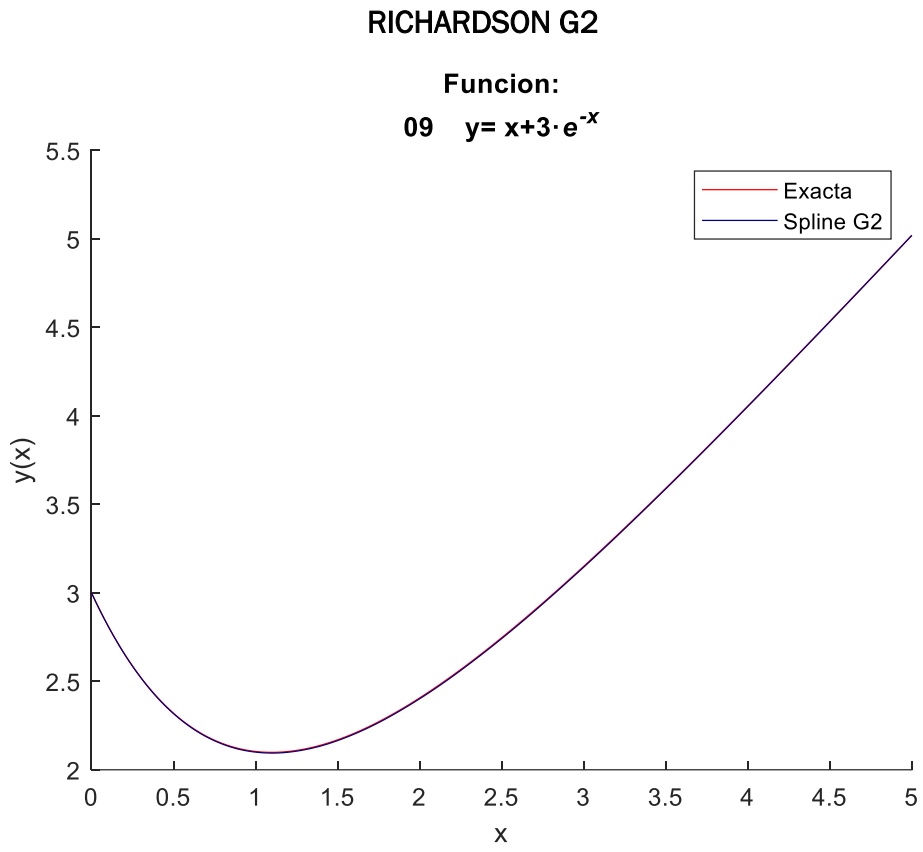


Figura 6-33: Richardson G2

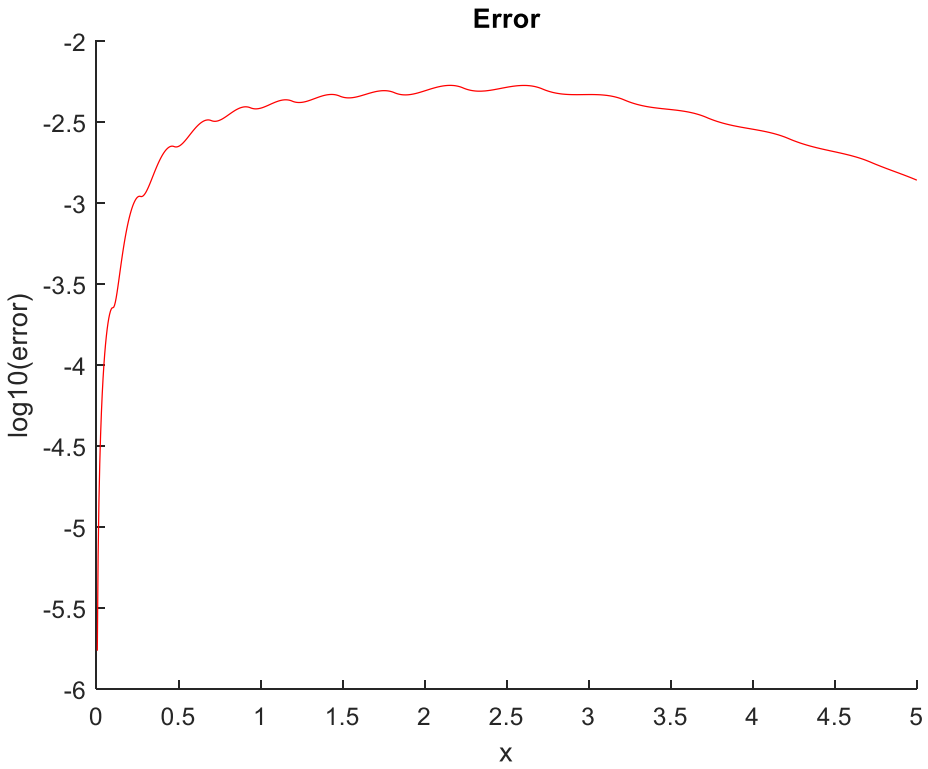


Figura 6-34: Errores

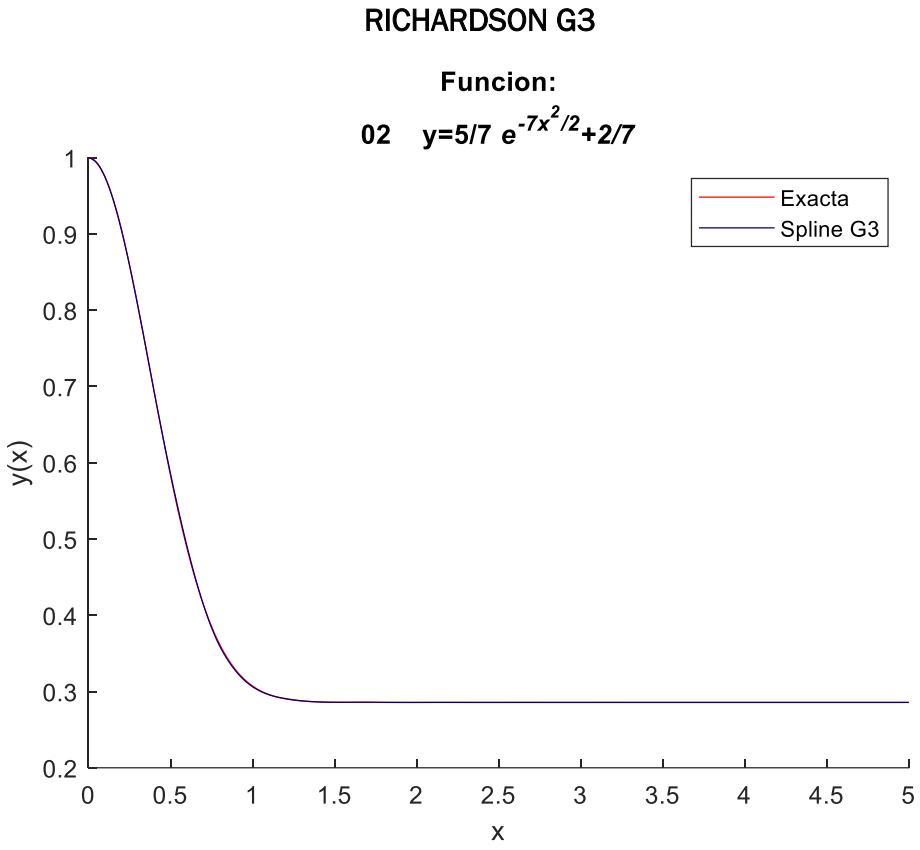


Figura 6-35: Richardson G3

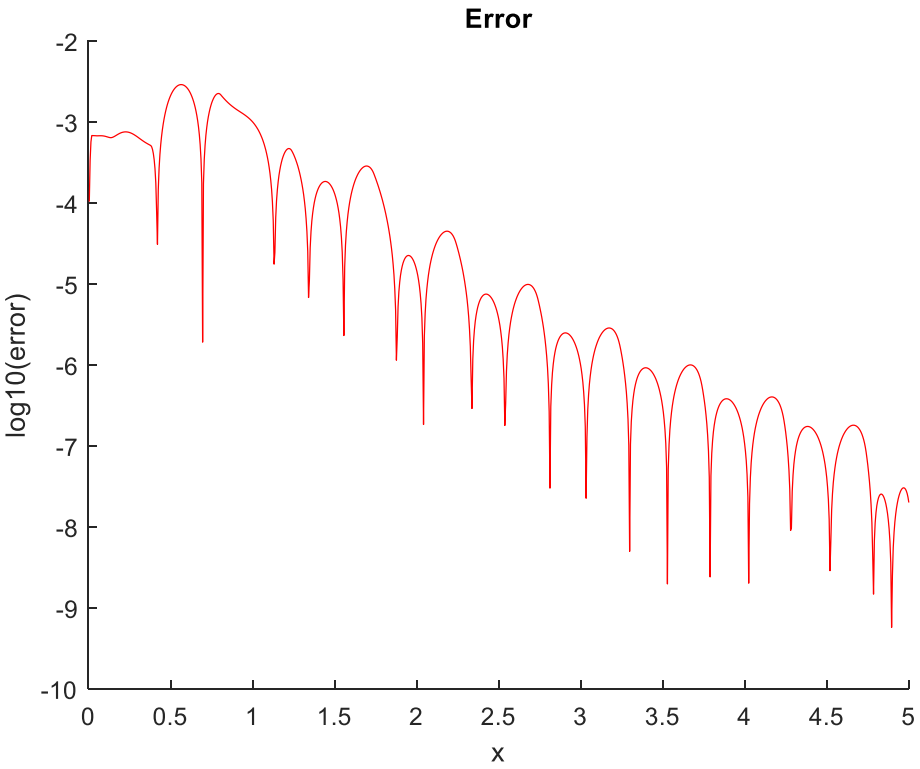


Figura 6-36: Errores

TABLA RESUMEN			
	$y = \frac{5}{7} \cdot e^{-7 \cdot \frac{x^2}{2}} + \frac{2}{7}$		
	$\boldsymbol{N^{\underline{o}}\ de\ pasos}$	$\boldsymbol{E_{máx}}$	$\boldsymbol{E_{medio}}$
Pares encajados	31	$8{,}1771 \cdot 10^{-4}$	$1{,}2486 \cdot 10^{-4}$
Richardson G2	22	$1{,}4 \cdot 10^{-3}$	$2{,}3992 \cdot 10^{-4}$
Richardson G3	15	$1{,}2 \cdot 10^{-3}$	$1{,}6612 \cdot 10^{-4}$
	$y = x + 3 \cdot e^{-x}$		
	$\boldsymbol{N^{\underline{o}}\ de\ pasos}$	$\boldsymbol{E_{max}}$	$\boldsymbol{E_{medio}}$
Pares encajados	19	0,0035	0,0024
Richardson G2	15	0,0053	0,0035
Richardson G3	14	0,0029	$2{,}8737 \cdot 10^{-4}$

Con la vista puesta en la tablas y figuras de este experimento, y mediante su comparación con el experimento anterior, podemos ver que el número de pasos se adapta a las necesidades del problema.

En estos casos se ha fijado una tolerancia correspondiente a 50 veces el error medio del experimento anterior.

Vemos que la carga de trabajo se reduce, en este caso, a menos de la mitad de evaluaciones en la mayoría de los casos y, además, podemos ofrecer una estimación del error.

6.5. EXPERIMENTO 5: Avance con métodos de orden alto o bajo.

Vamos a comparar las distintas formas de avanzar con los métodos cuando tenemos estrategia de cambio de paso.

La idea es que, con el método de orden más bajo, nos va a quedar una estructura más sencilla, aunque menos precisa. Esta va a ser consecuente con las tolerancias del error que fijamos.

Si avanzamos con el método de orden alto, la estructura es algo más densa, pero, en cambio, obtenemos unos errores inferiores a la tolerancia fijada.

Vamos a trabajar con las siguientes funciones de prueba:

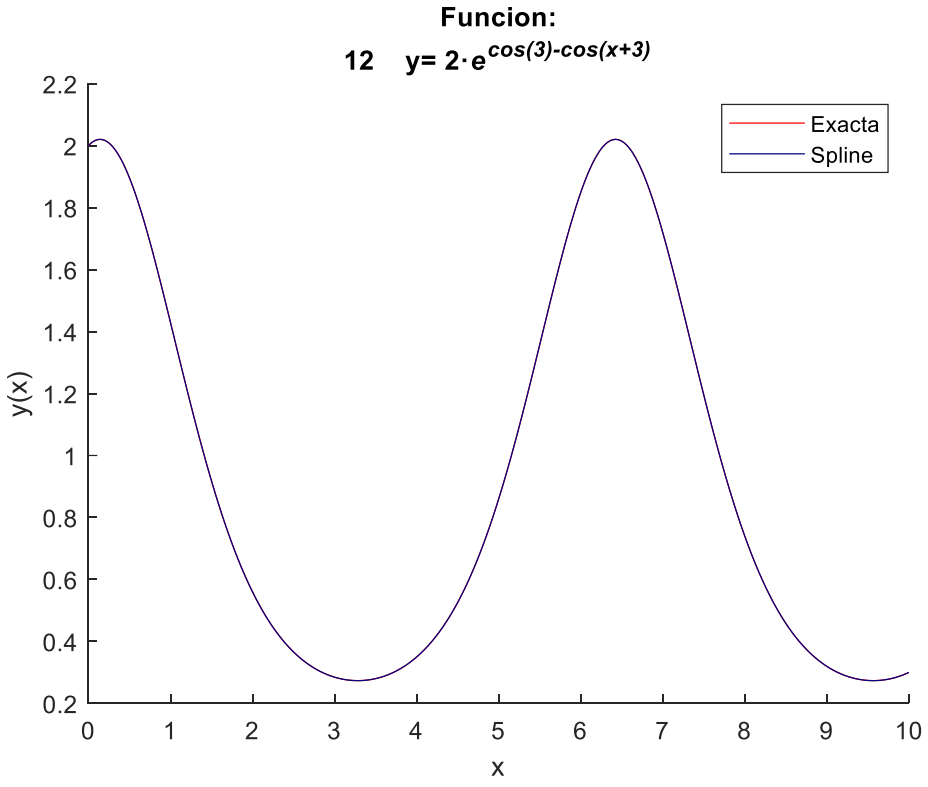
$$y = 2 \cdot e^{\cos(3) - \cos(x+3)}$$

Esta función es la solución de la Ecuación diferencial

$$\begin{cases} y'(x) = sen(x+3) \cdot y(x) \\ y(0) = 2 \end{cases}$$

En este caso vamos a tomar un intervalo amplio para poder ver las distintas zonas.

La gráfica de la función y la aproximación tiene este aspecto:



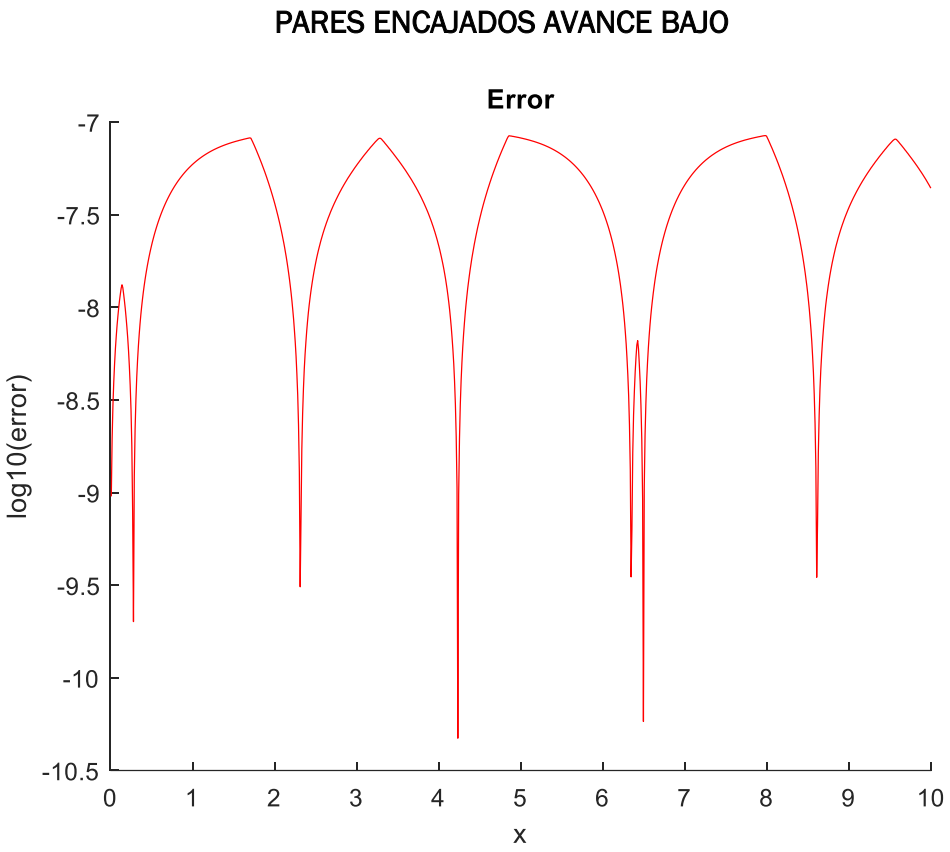


Figura 6-37: Pares encajados

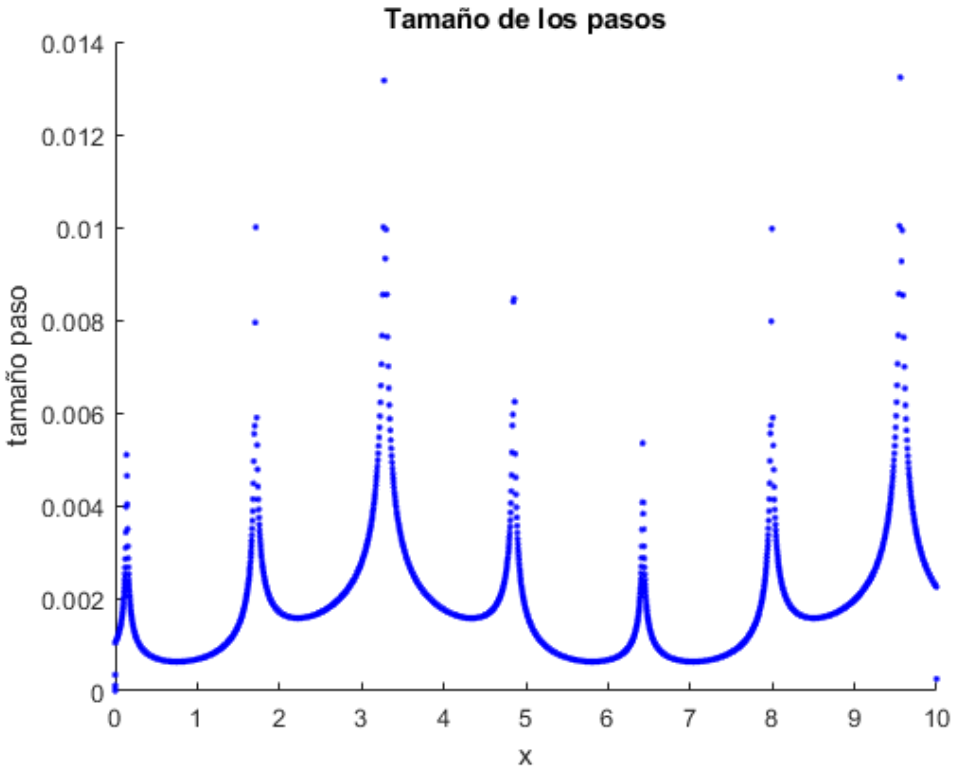


Figura 6-38: Tamaño paso

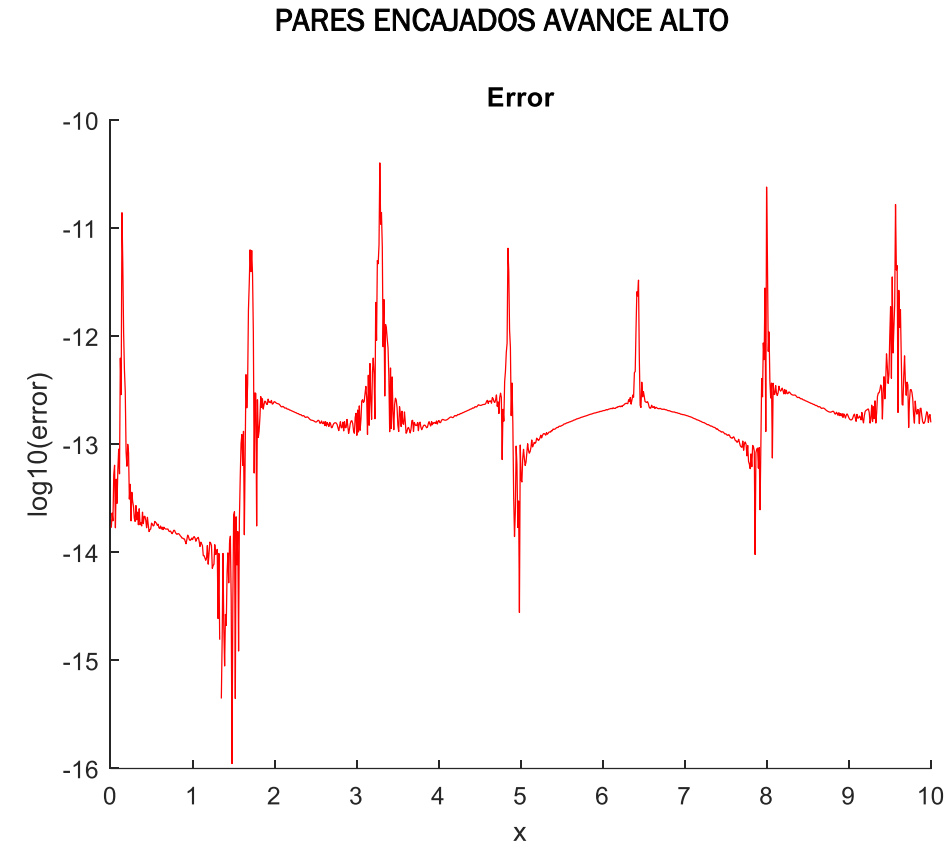


Figura 6-39: Pares encajados avance alto

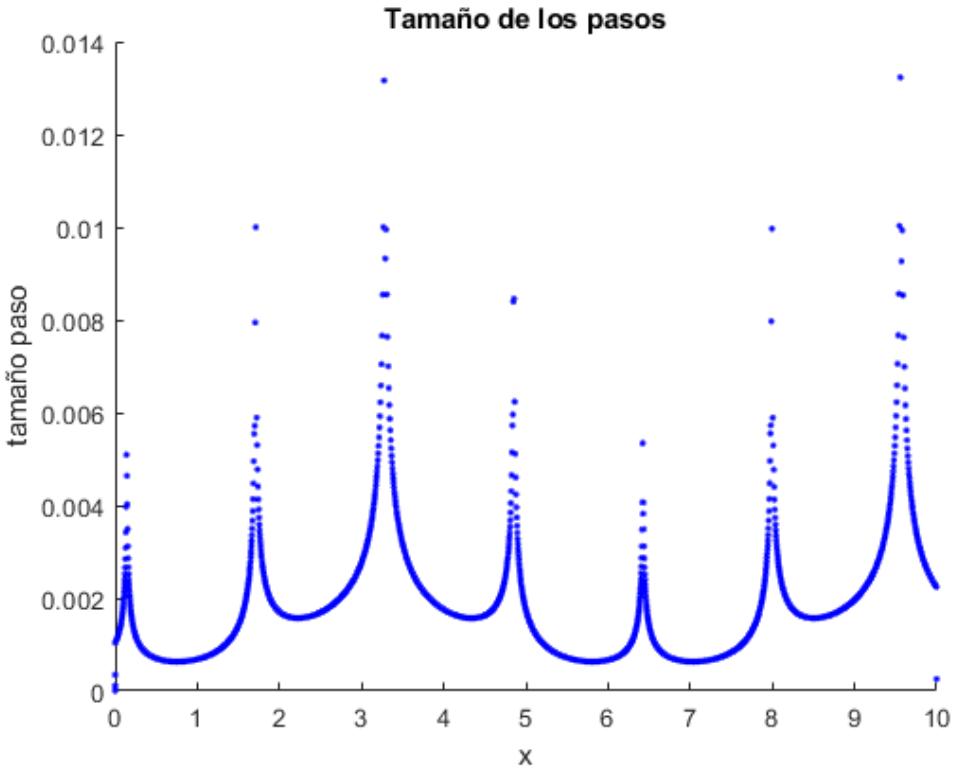


Figura 6-40: Tamaño paso

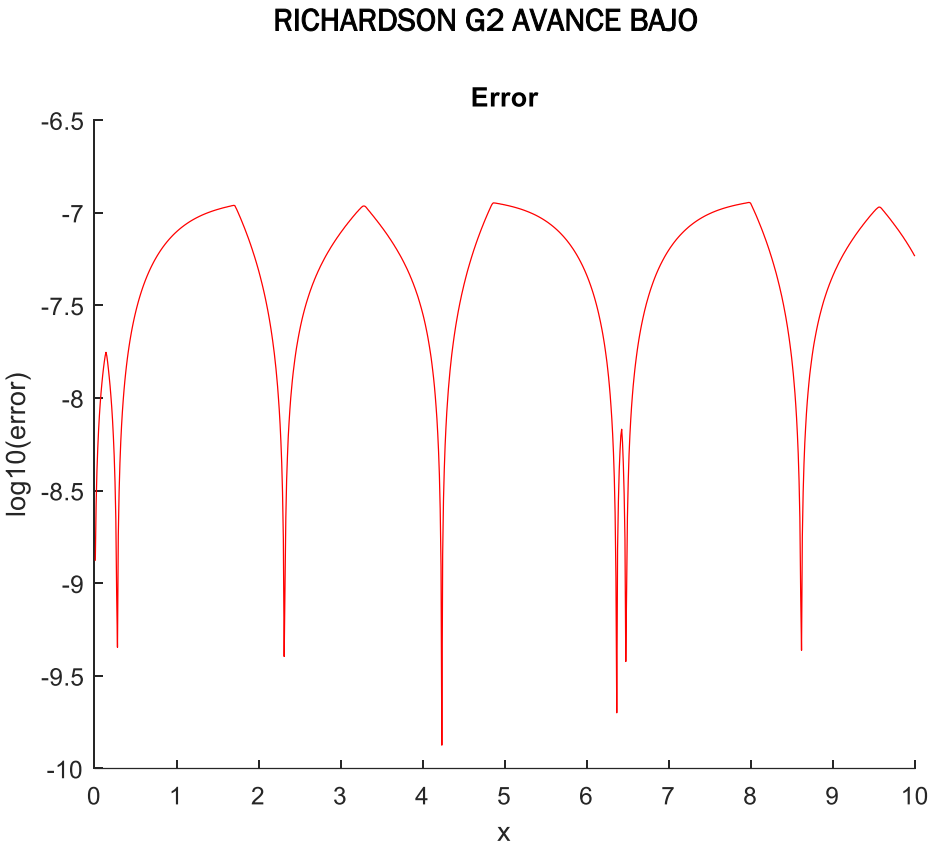


Figura 6-41: Richardson avance bajo

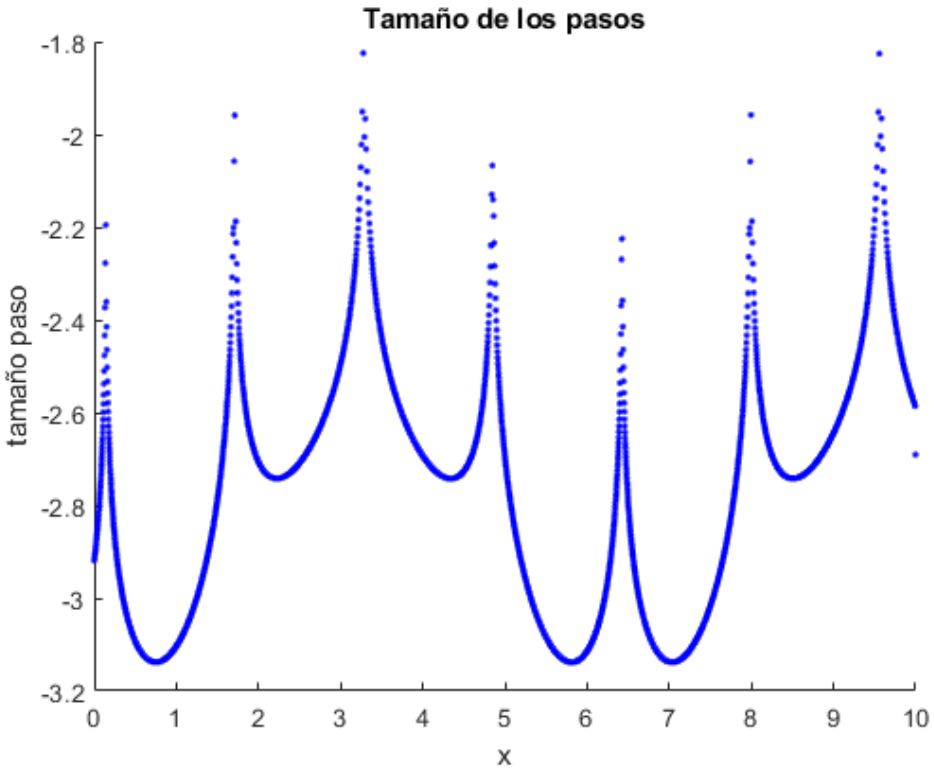


Figura 6-42: Tamaño paso

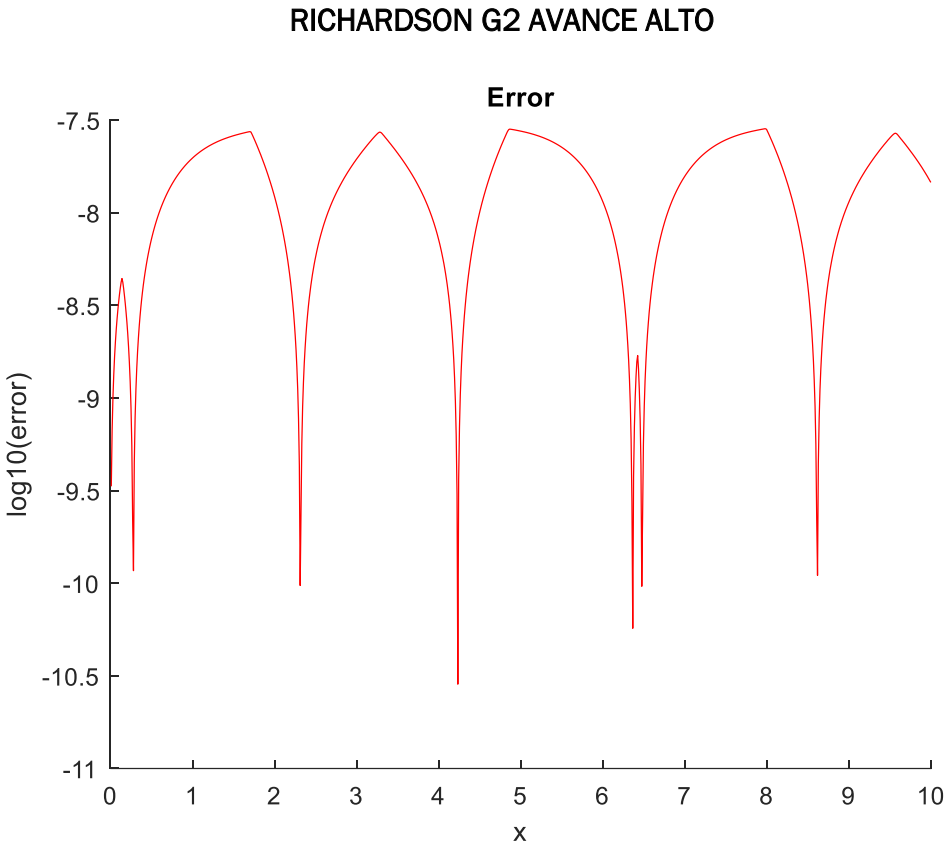


Figura 6-43: Richardson alto

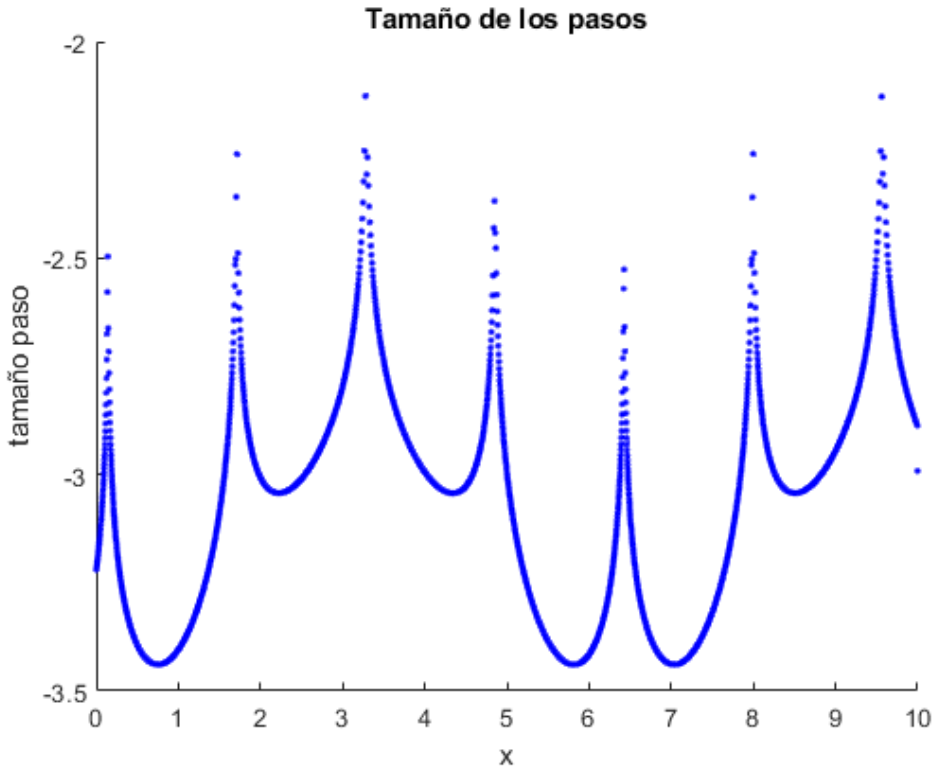


Figura 6-44: Tamaño paso

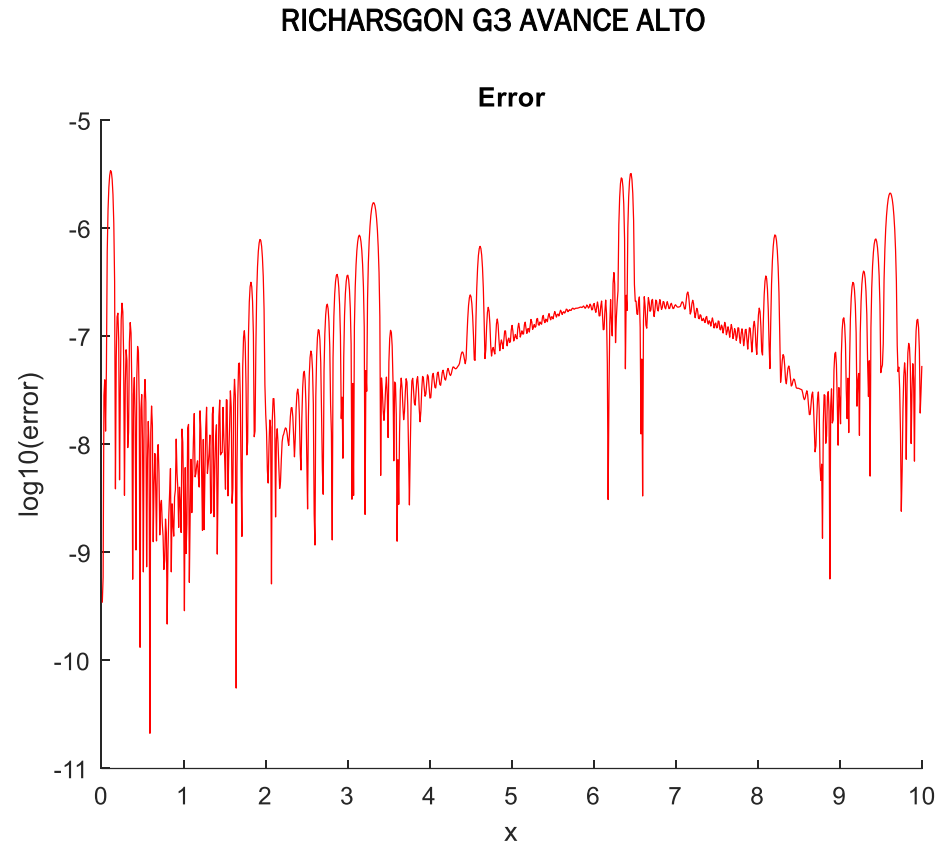


Figura 6-45: Richardson avance bajo

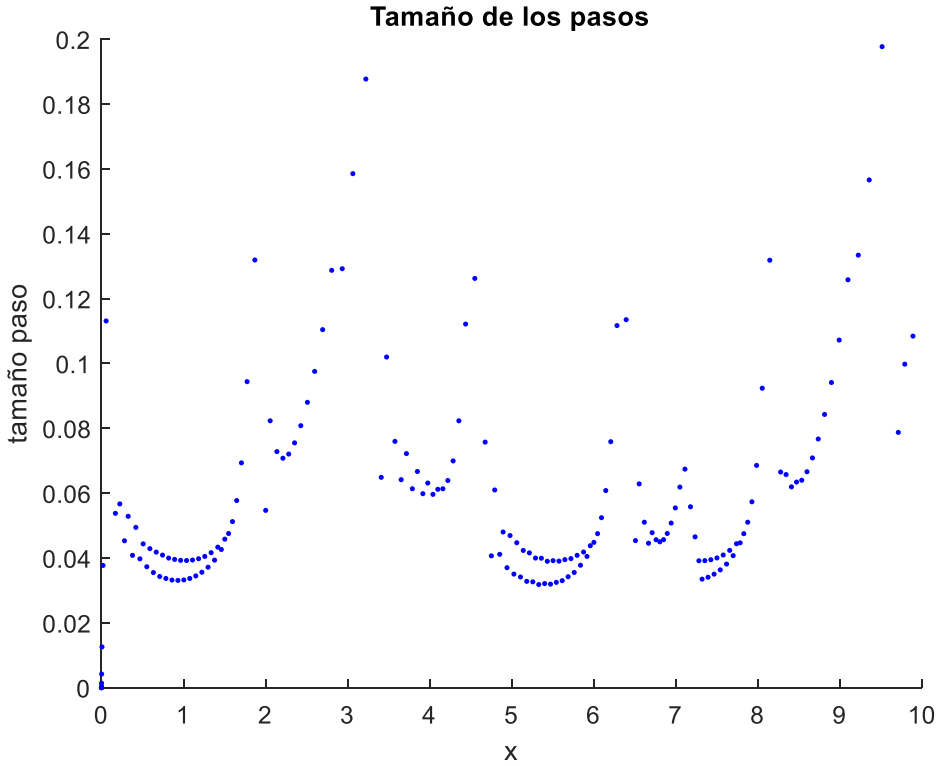


Figura 6-46: Tamaño paso

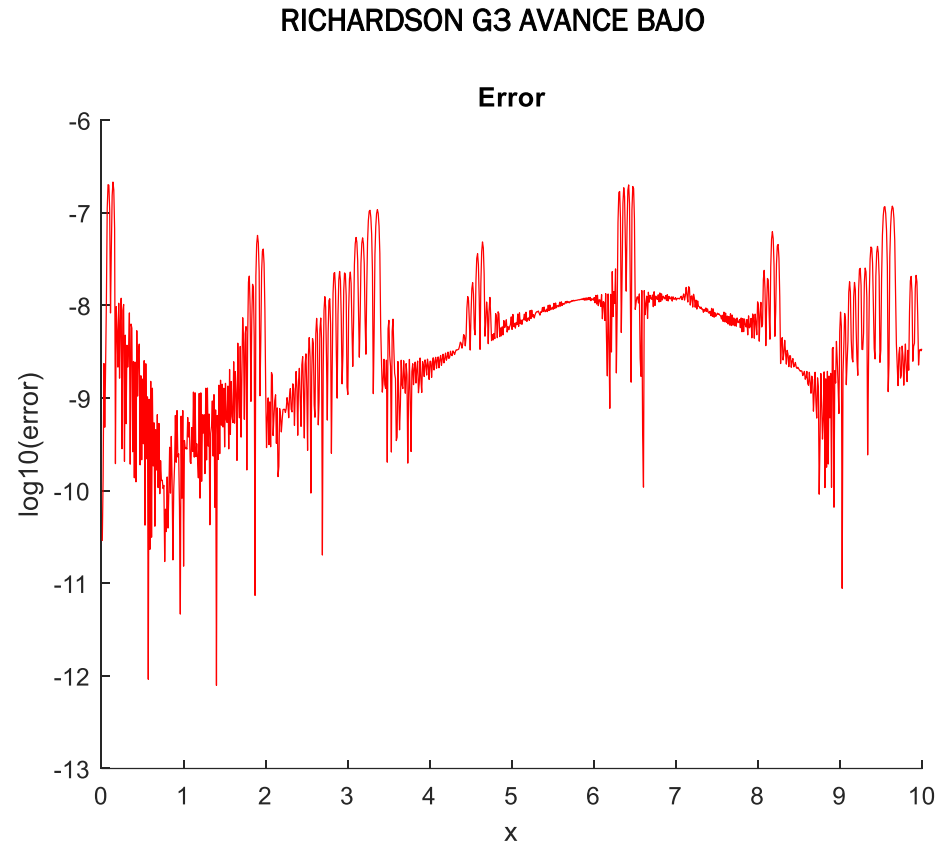


Figura 6-47 Richardson alto

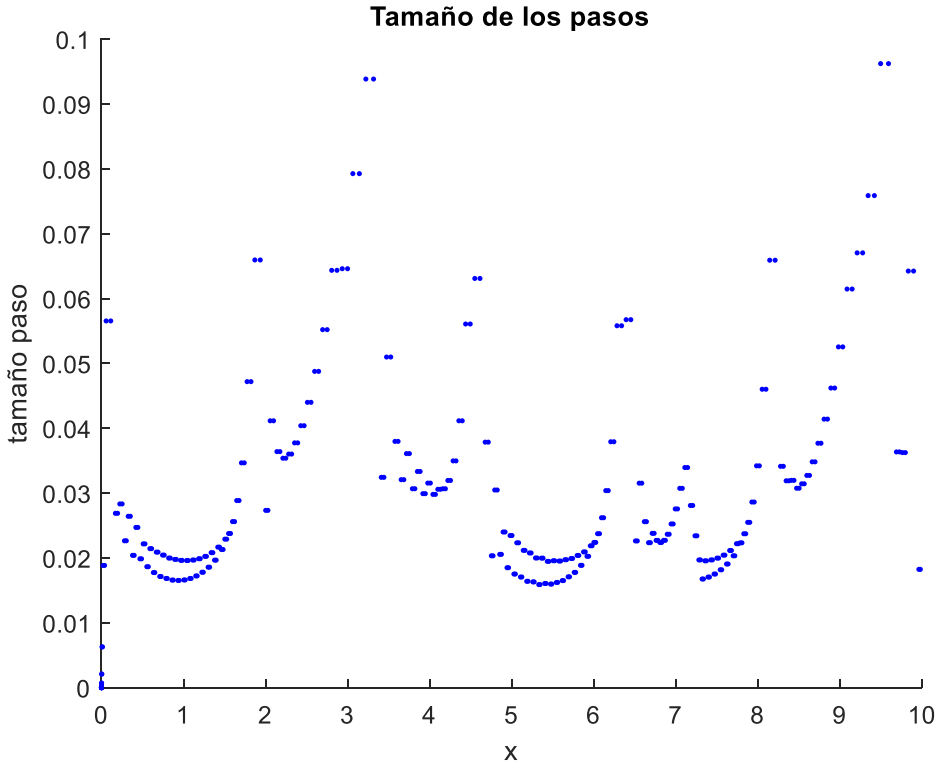


Figura 6-48: Tamaño paso

Tabla 6-3 Tipo de avance de los métodos de paso variable

$y = 2 \cdot e^{\cos(3)-\cos(x+3)}$			
AVANCE CON ORDEN BAJO			
	$N^{\underline{o}}$ **de pasos**	$E_{máx}$	E_{medio}
Pares encajados	7856	$8{,}4446 \cdot 10^{-8}$	$4{,}5701 \cdot 10^{-8}$
Richardson G2	6797	$1{,}1350 \cdot 10^{-7}$	$6{,}1354 \cdot 10^{-8}$
Richardson G3	177	$3{,}4047 \cdot 10^{-6}$	$2{,}0137 \cdot 10^{-7}$
AVANCE CON ORDEN ALTO			
	$N^{\underline{o}}$ **de pasos**	E_{max}	E_{medio}
Pares encajados	7856	$4{,}0085 \cdot 10^{-11}$	$4{,}0085 \cdot 10^{-11}$
Richardson G2	135934	$2{,}8373 \cdot 10^{-8}$	$1{,}5338 \cdot 10^{-8}$
Richardson G3	356	$2{,}14 \cdot 10^{-7}$	$1{,}12382 \cdot 10^{-8}$

A la vista de las tablas se confirman las hipótesis iniciales.

Las pruebas se han realizado con una tolerancia absoluta de 10^{-6}.

Como es de esperar, los métodos basados en extrapolación de Richardson presentan un número de pasos del doble que cuando se avanza con el método de orden bajo. Se puede ver que el método de pares encajados no presenta una disminución en los pasos, en cambio reduce mucho el error.

El método que presenta un menor número de pasos es el basado en extrapolación de Richardson con un Spline de grado 3 y con avance de orden bajo.

6.6. EXPERIMENTO 6: Adaptabilidad de los métodos

Vamos a considerar la siguiente función rígida.

$$y = \frac{4971 \cdot e^{-70x} - 70 \cdot \cos(x) + 4900 \cdot sen(x)}{4901}$$

Esta función es la solución de la Ecuación diferencial

$$\begin{cases} y'(x) = -70 \cdot y(x) + 70 \cdot sen(x) \\ y(0) = 1 \end{cases}$$

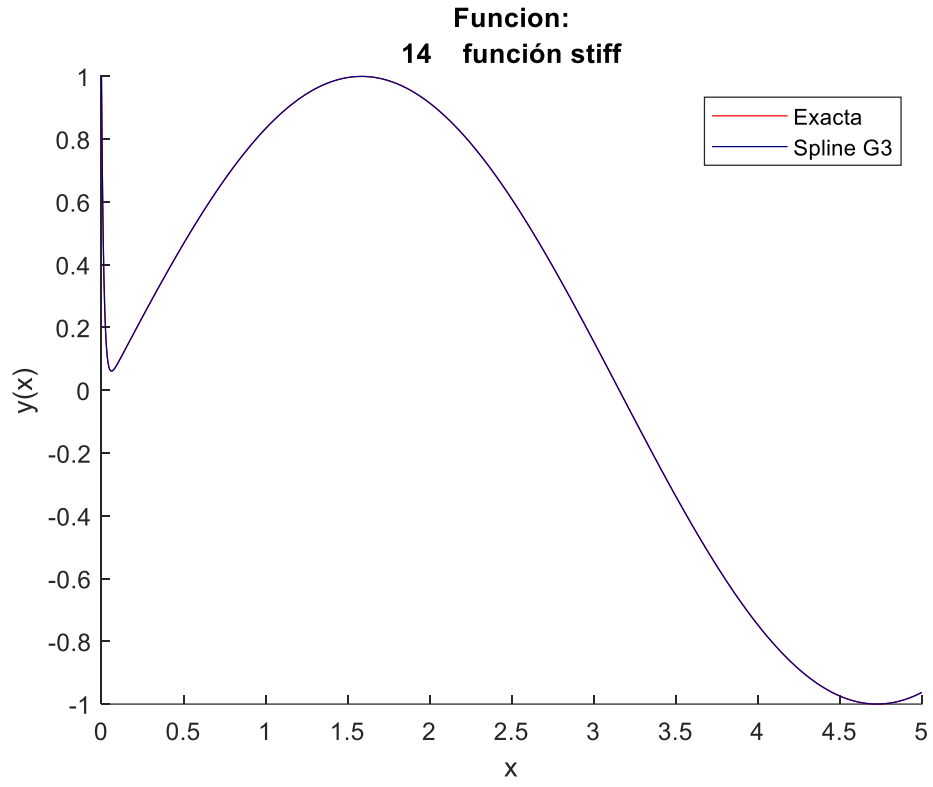


Figura 6-49: Función rígida

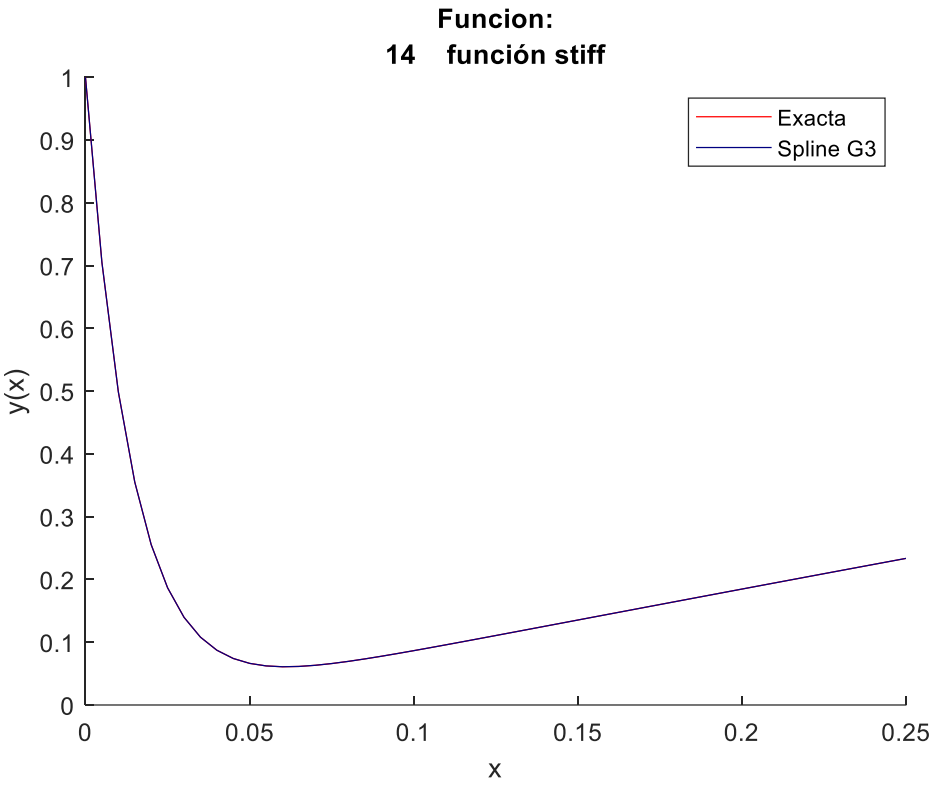


Figura 6-50: Detalle

Habiendo comprobado la validez de los métodos de adaptación de paso, vamos a ver cómo se pueden modificar.

Primero, vamos a ver la problemática de estas funciones. Se puede observar que tenemos una zona con la pendiente muy pronunciada y después otra en la que la pendiente es mucho más suave. Parece razonable que no se puede trabajar de igual modo en ambas partes.

Si intentamos abordar la función con un método sin cambio de paso, por ejemplo, el de Splines de grado 3 y clase C1:

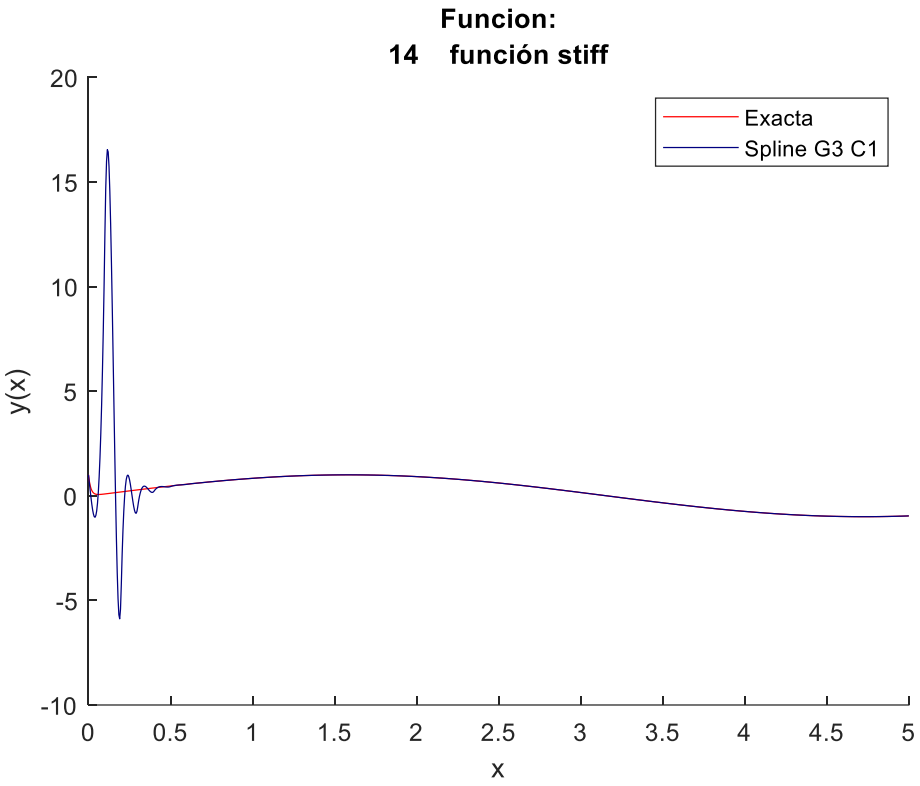


Figura 6-51: Paso 0,1

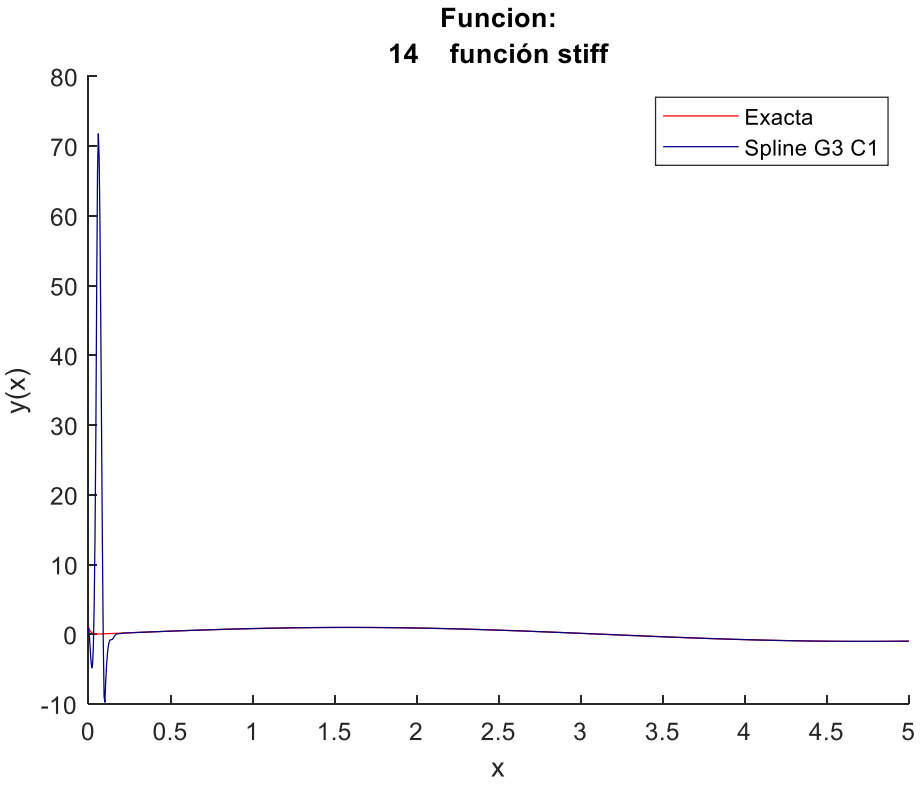


Figura 6-52: Paso 0,05

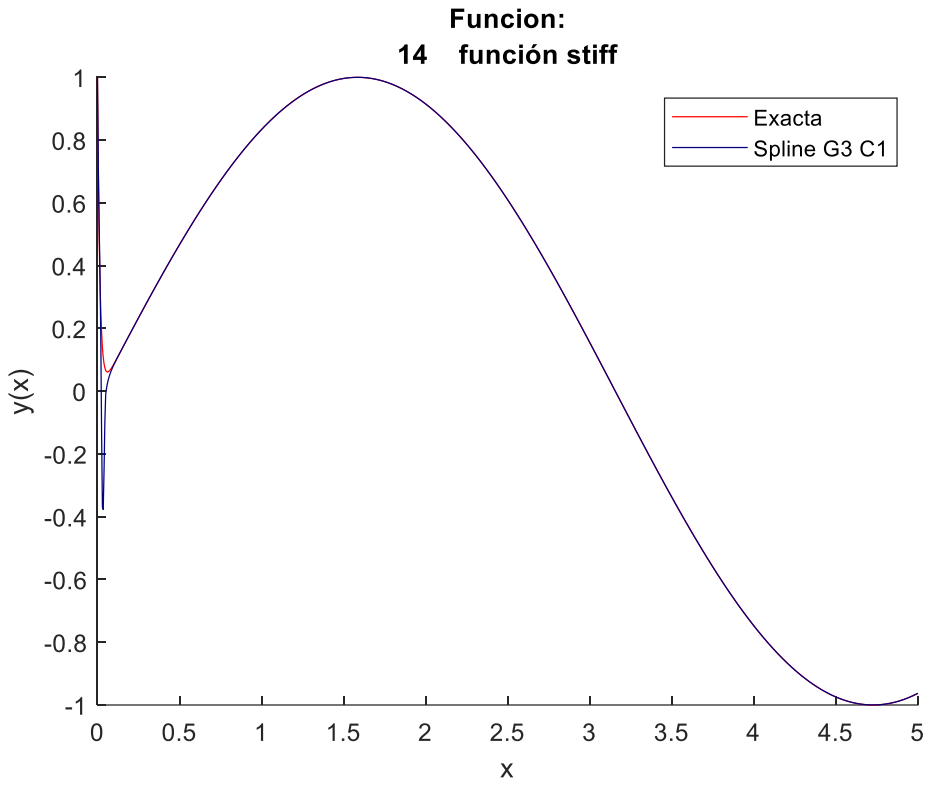


Figura 6-53: Paso 0,025

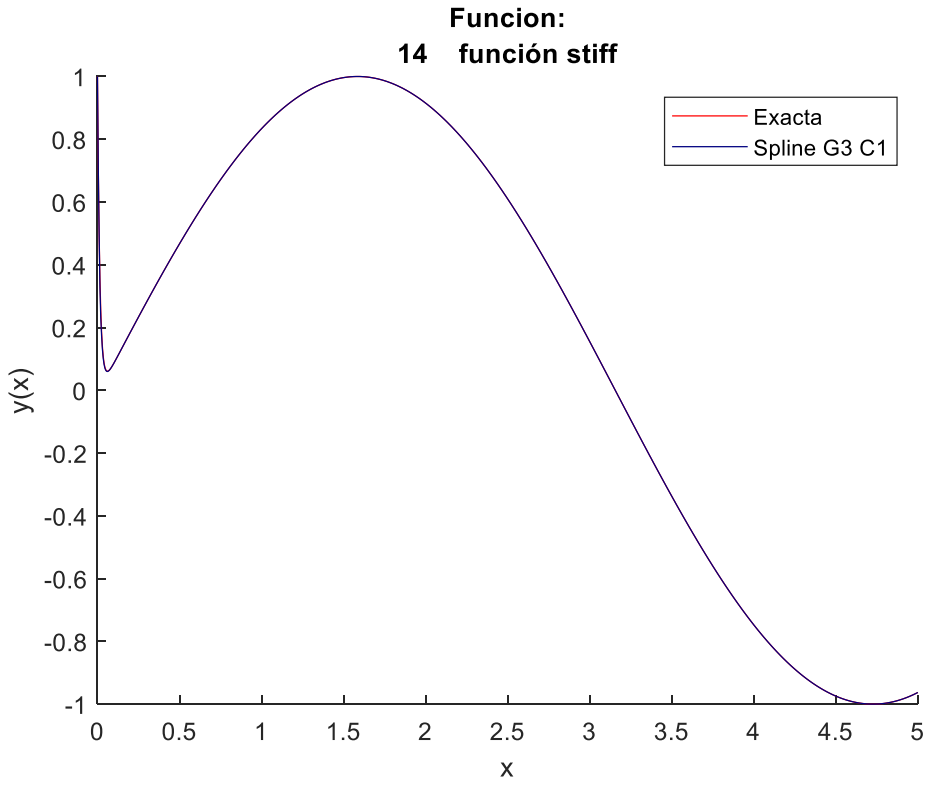


Figura 6-54: Paso 0,0125

En primer lugar, vamos a tomar el método de pares encajados con una tolerancia absoluta de 10^{-3}

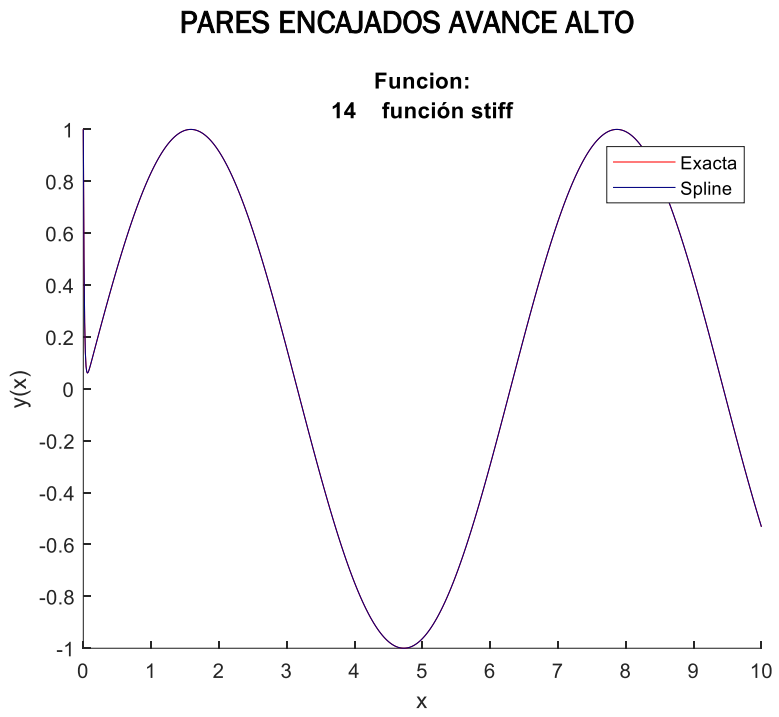


Figura 6-55: Aproximación a la función

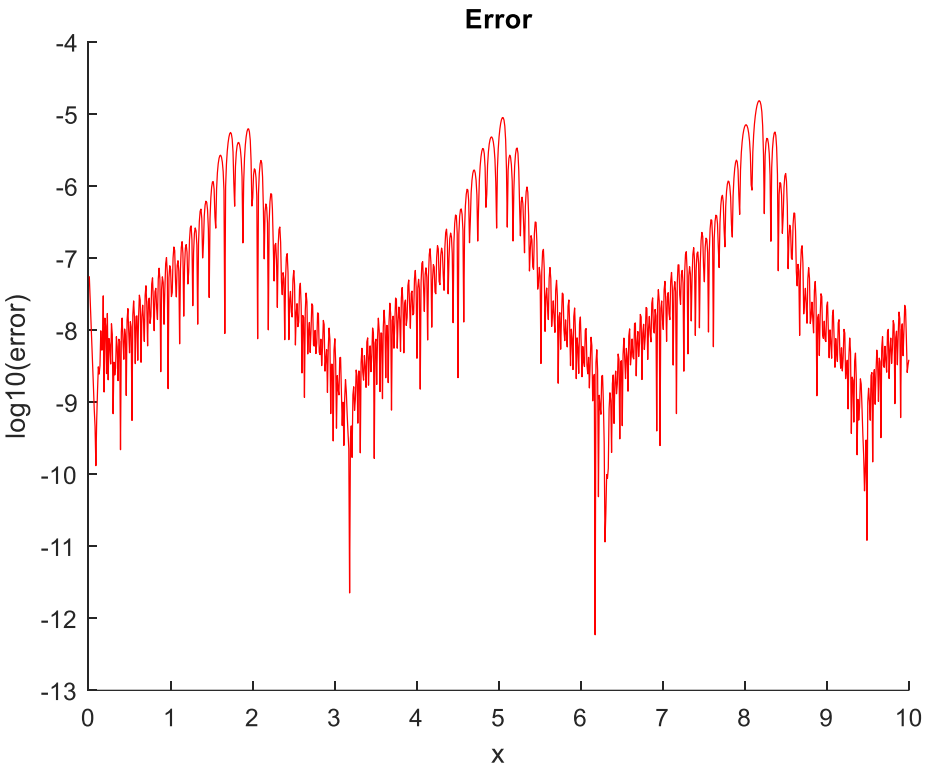


Figura 6-56: Error

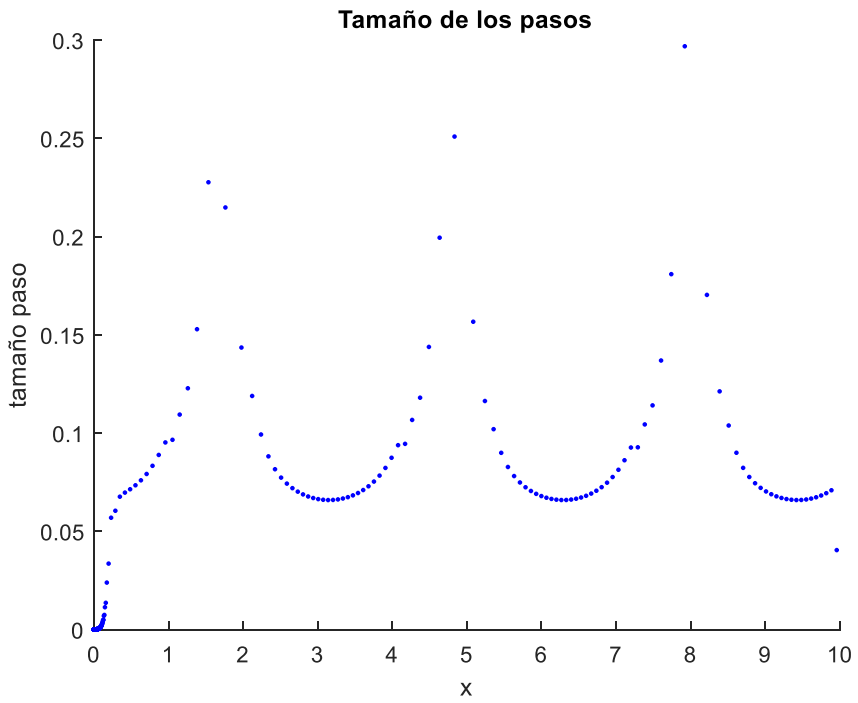


Figura 6-57: Adaptación del paso

Se puede ver que, en la zona de alta pendiente, los pasos que estamos obligados a dar son muy pequeños.

Viendo un detalle de la zona en cuestión, tenemos:

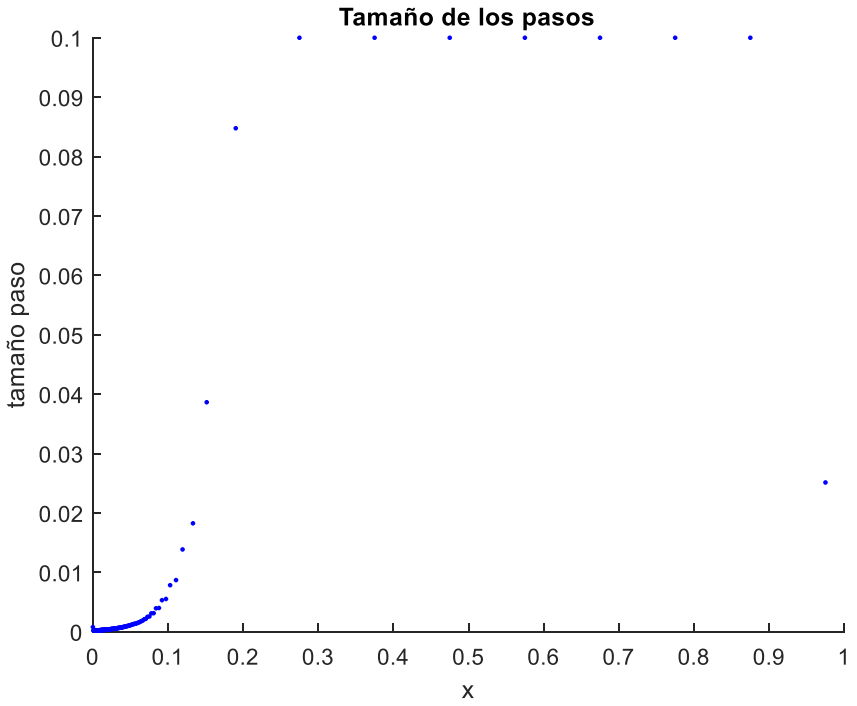


Figura 6-58: Tamaño de paso en la zona de cambio.

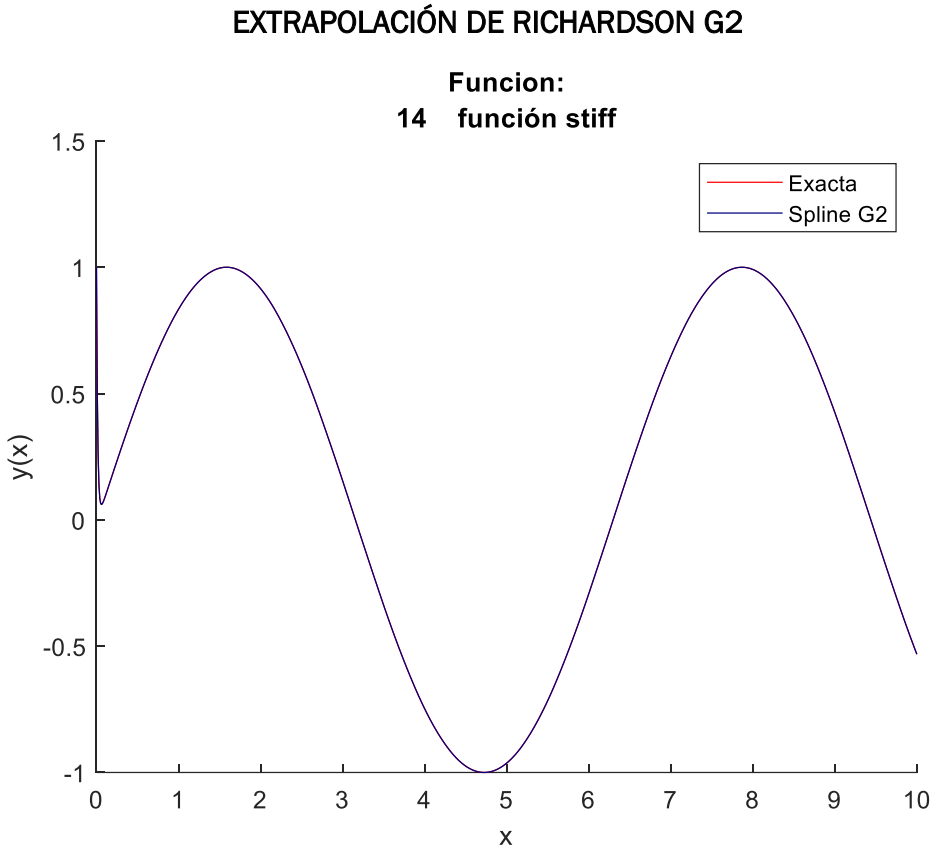


Figura 6-59: Richardson Grado 2

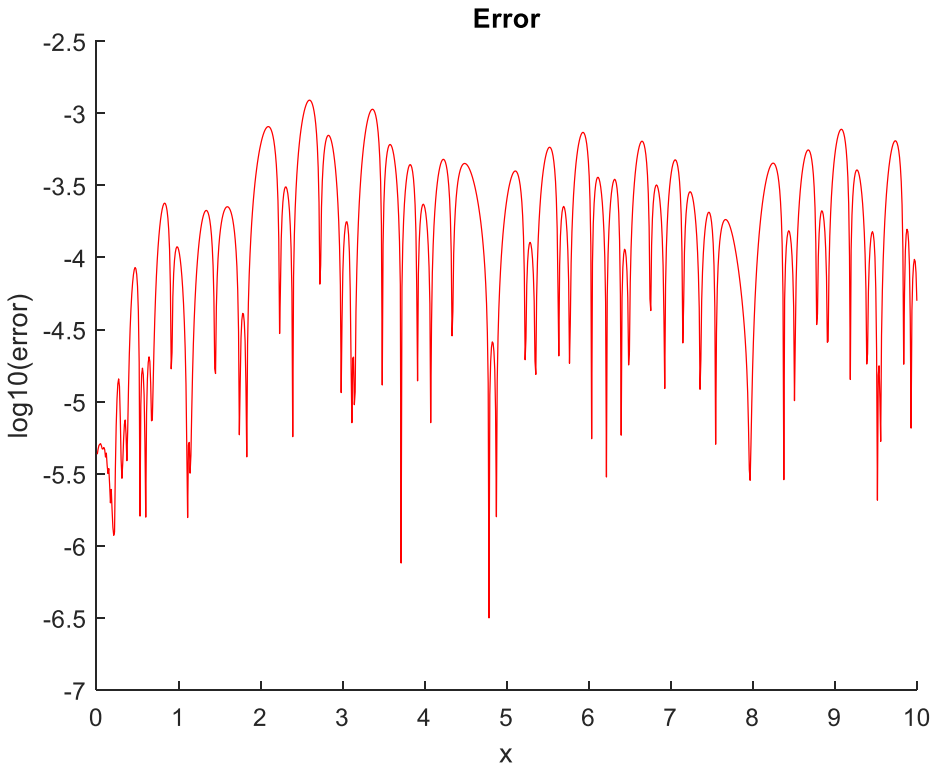


Figura 6-60: Error

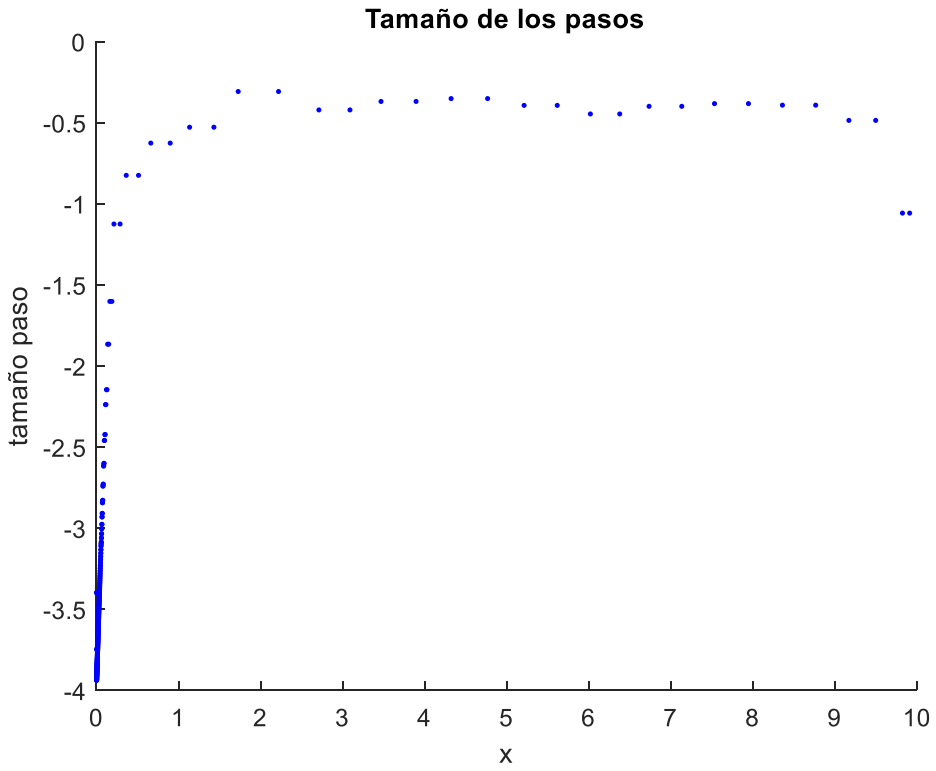


Figura 6-61: Tamaño de paso

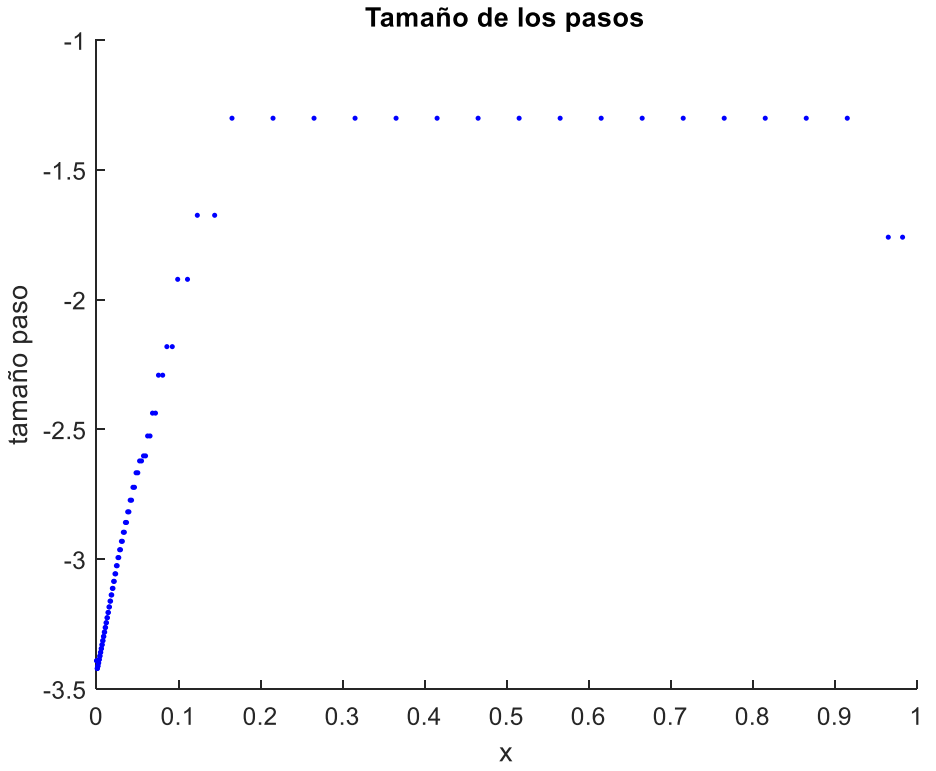


Figura 6-62: Detalle

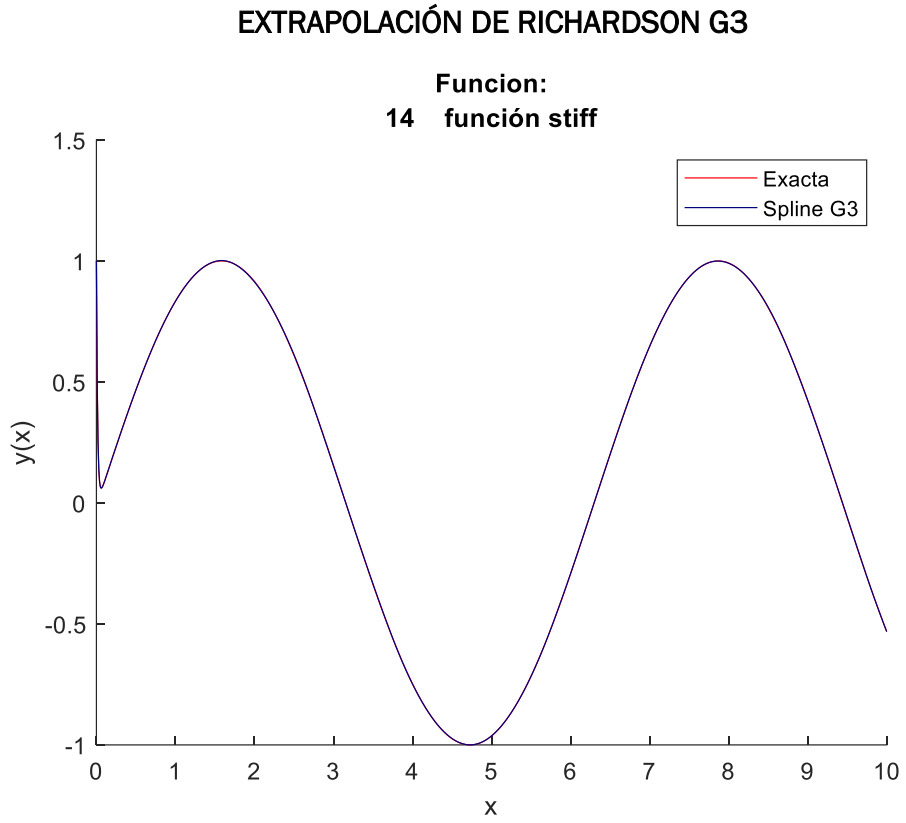


Figura 6-63: Función

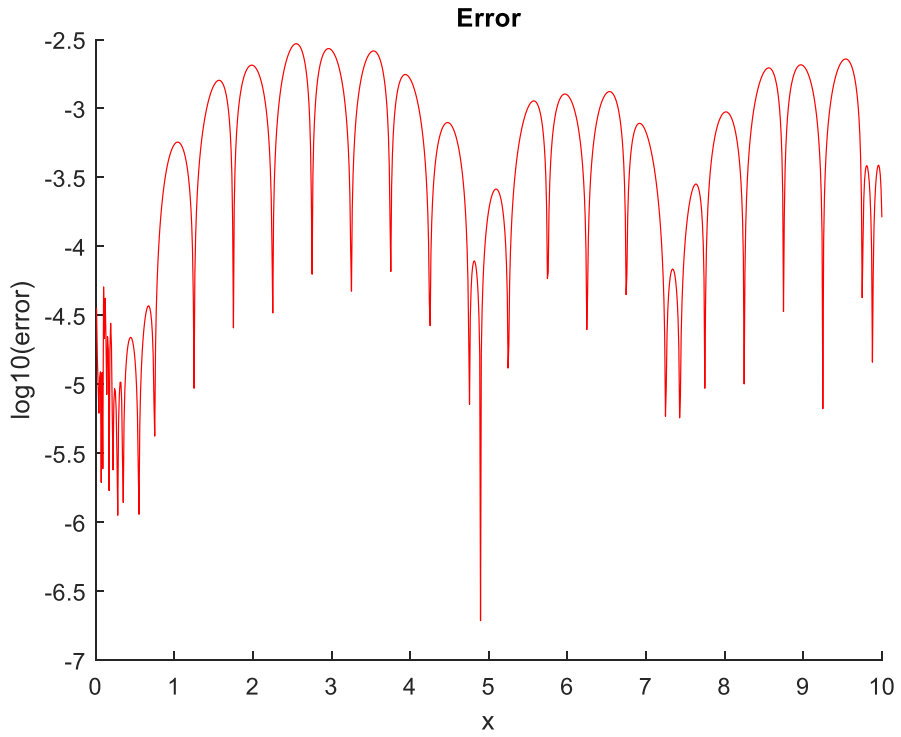


Figura 6-64: Error

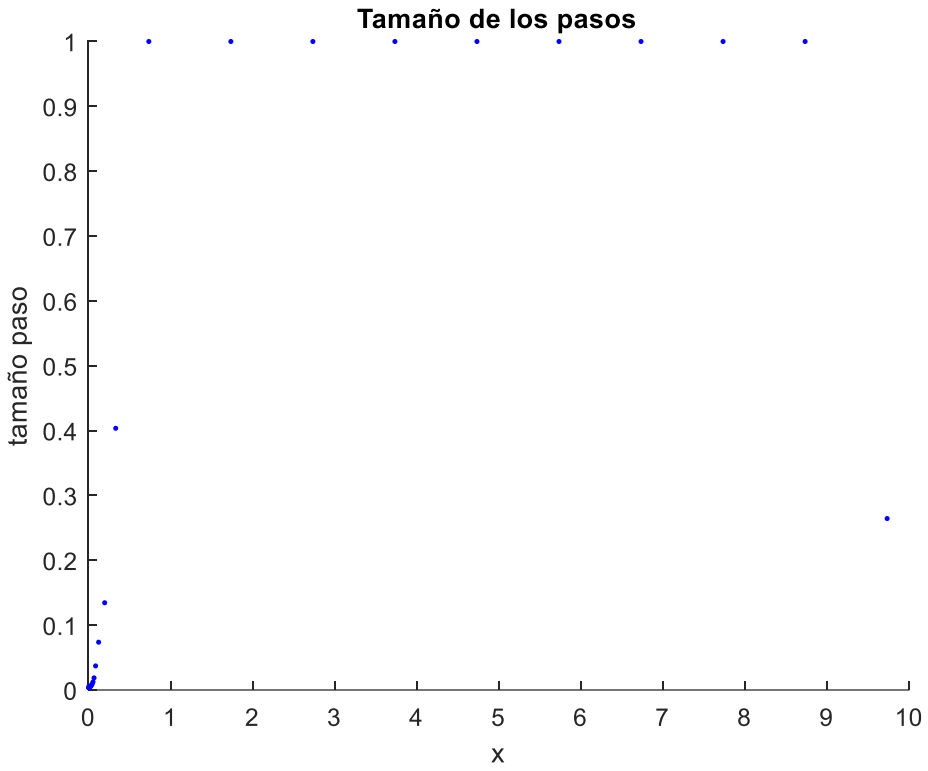


Figura 6-65: Pasos

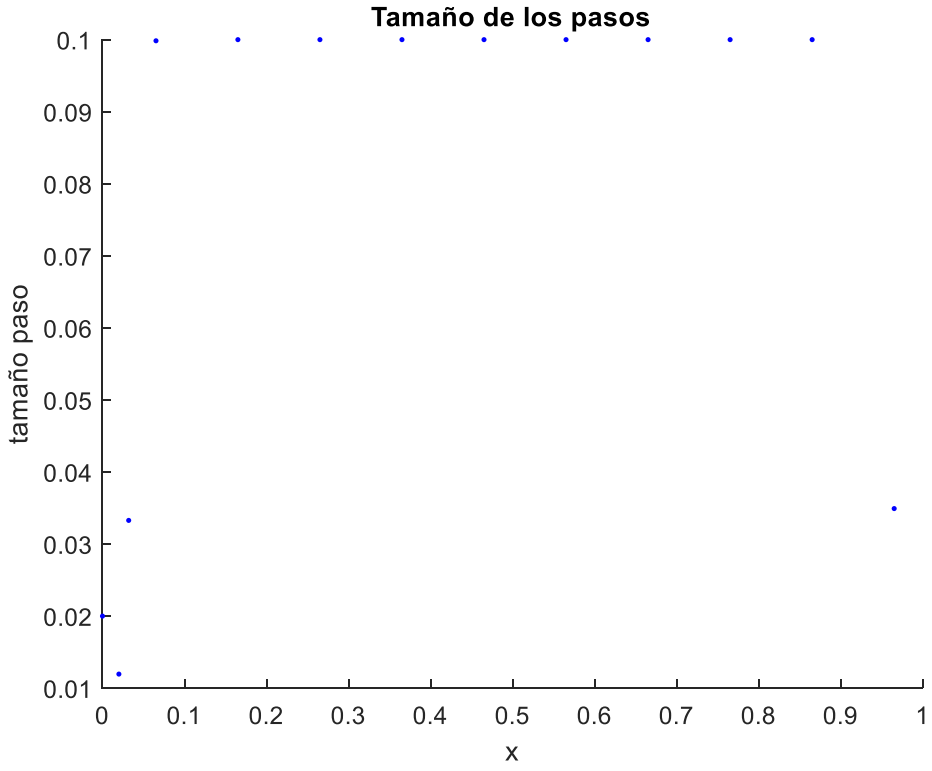


Figura 6-66: Detalle

Vemos que según escojamos un método u otro, nos van a ofrecer resultados ligeramente distintos.

Sin embargo, no podemos decir que uno sea mejor que otro. Todo dependerá de los parámetros que sea prioritario optimizar.

Se ha diseñado este experimento para conocer cómo afecta el cálculo de la estimación del error a las distintas estrategias.

6.7. EXPERIMENTO 7: Importancia del coeficiente de seguridad NO LINEAL

Hemos visto en el desarrollo teórico que se incluye un coeficiente de seguridad en el cálculo del parámetro α, que es el que permite el cambio de paso.

El cálculo así realizado, nos proporciona una estimación del paso necesario para satisfacer el error, pero esto conlleva riesgos de rechazo de pasos y, por tanto, de nuevos cálculos.

Conscientes de esto, se introduce este coeficiente y vamos a ver mediante la experimentación cómo afecta a los cálculos.

Vamos a usar el estimador de extrapolación de Richardson con un Spline de grado 2.

La función sobre la que vamos a trabajar es

$$y = 2 \cdot e^{-(\cos(3)-\cos(x+3))}$$

Esta función es la solución de la Ecuación diferencial

$$\begin{cases} y'(x) = sen(x+3) \cdot y(x) \\ y(0) = 2 \end{cases}$$

Para un intervalo de $[0, 5]$ y con una tolerancia absoluta de 10^{-3} y resolviendo de una manera no lineal, tenemos:

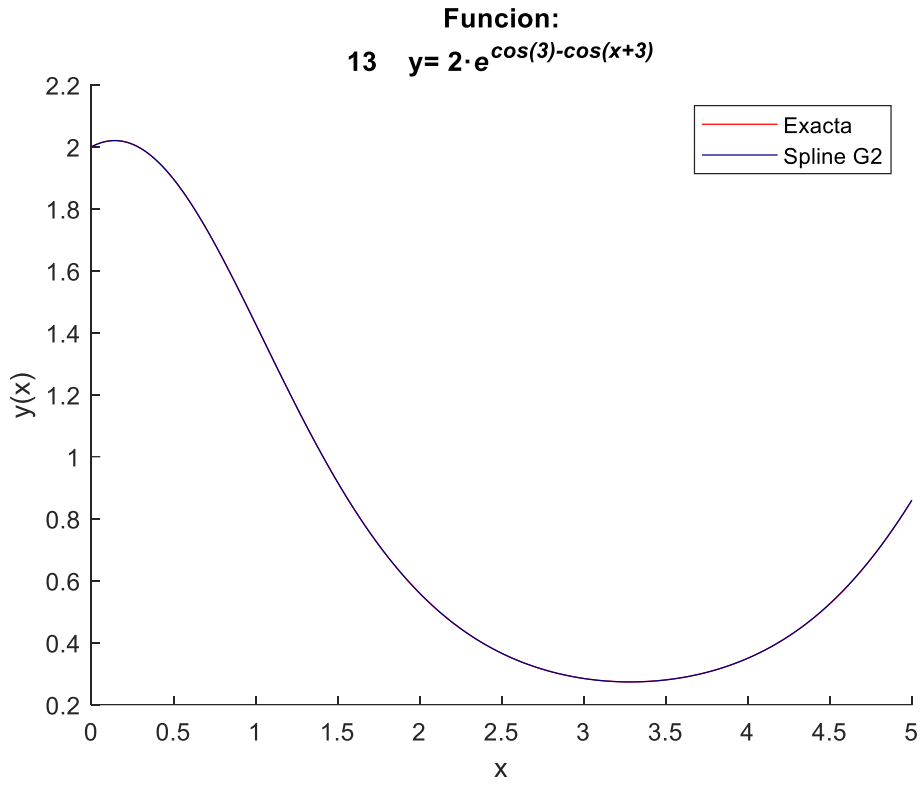


Figura 6-67: Función

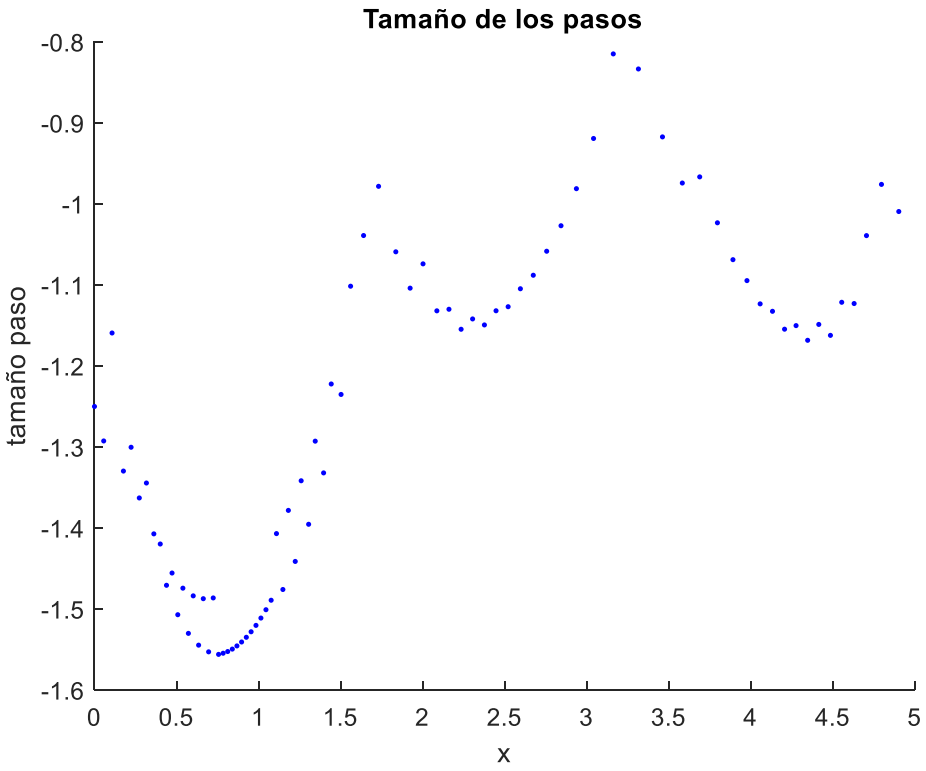


Figura 6-68: Tamaño de paso

Cs	Aceptados	Rechazados
1	70	56
0.98	72	52
0.96	73	50
0.94	74	46
0.92	75	41
0.90	77	43
0.85	80	21

Se puede ver cómo evoluciona la cantidad de pasos rechazados.

Al hacerse más grande el Coeficiente de seguridad, provoca que el intervalo sea más pequeño de lo que originalmente se ha calculado. De ahí que se vea un aumento en el número de intervalos aceptados.

Concretamente, si se ha reducido un 15% el tamaño de paso, es razonable que el número de subintervalos aumente en un 15% de una manera aproximada.

No obstante, el número de iteraciones rechazadas ha caído a la mitad.

Por otra parte. Al hacerse los intervalos más pequeños, disminuye el error local del método.

Este ejemplo se ha realizado como un sistema no lineal.

Todos los métodos están implementados en los programas para poder resolverse de una manera no lineal.

6.8. EXPERIMENTO 8: Coeficiente de exigencia dependiente del ángulo

Vamos a considerar la siguiente función rígida.

$$y = \frac{4971 \cdot e^{-70x} - 70 \cdot \cos(x) + 4900 \cdot sen(x)}{4901}$$

Esta función es la solución de la Ecuación diferencial

$$\begin{cases} y'(x) = -70 \cdot y(x) + 70 \cdot sen(x) \\ y(0) = 1 \end{cases}$$

Vamos a considerar la función en un intervalo $[-0.1, 0{,}1]$ y con una tolerancia absoluta de 10^{-3} y un coeficiente de seguridad $Cs = 0.85$

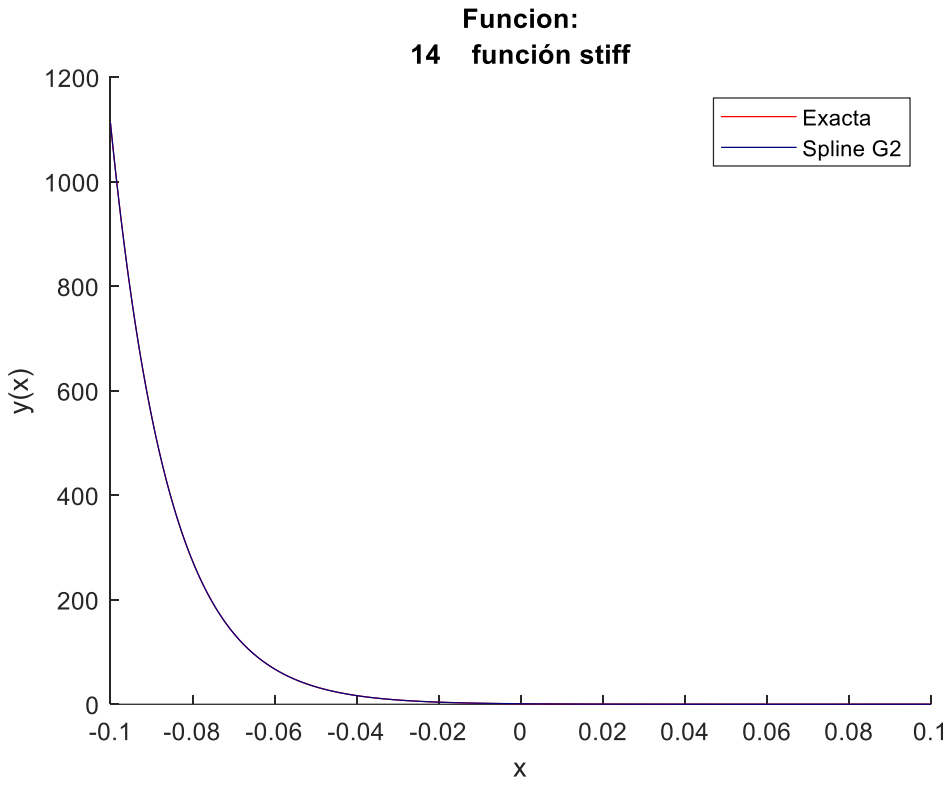


Figura 6-69: Función

Vamos a ver cómo evolucionan los parámetros de la aproximación

Ce	N.º Intervalos	N.º Rechazos	Error medio
100	2162	2153	0.0015
90	2295	2287	$6{,}5583 \cdot 10^{-5}$
80	2257	2250	$6{,}6523 \cdot 10^{-5}$
70	2187	2175	$6.9934 \cdot 10^{-5}$
60	1927	6	$8{,}4167 \cdot 10^{-5}$
50	1957	6	$8{,}0826 \cdot 10^{-5}$
40	1927	6	$8{,}4160 \cdot 10^{-5}$
30	2187	2174	$6{,}9934 \cdot 10^{-5}$
20	2257	2249	$6{,}6523 \cdot 10^{-5}$
10	2297	2290	$6{,}5611 \cdot 10^{-5}$
0	1897	1887	0.1281

Aunque se han encontrado funciones en las que es razonable usar este criterio, hay otras en las que no se puede aplicar de manera exitosa, ya que es un parámetro muy dependiente del problema.

Se puede ver cómo en las cifras centrales disminuye casi por completo el número de rechazos, pero aumenta de una manera brusca y no es consistente.

Se puede dejar abierta esta línea de investigación que puede ser interesante para la aplicación práctica de algunos problemas.

7. CONCLUSIONES

7.1. RESULTADOS OBTENIDOS

Podemos constatar que todos los objetivos que propusimos al principio de este TFG se han alcanzado satisfactoriamente.

Hemos mostrado cómo los métodos propuestos en este TFG permiten aproximar, mediante Splines obtenidos con la técnica de colocación, problemas de valores iniciales, que involucran Ecuaciones Diferenciales Ordinarias de Primer Orden.

Los métodos estudiados en esta memoria permiten obtener soluciones numéricas precisas, cuando se hace uso de un tamaño de paso adecuado al problema considerado.

Los métodos de colocación basados en Splines Cuadráticos, muestran empíricamente un orden de convergencia tres.

El orden de convergencia que alcanzan empíricamente los basados en Splines y Pseudo-Splines cúbicos, aplicados a los problemas considerados, es cuatro, esto es, son incluso superiores a lo que esperábamos.

Las aproximaciones numéricas que proporcionan los métodos propuestos en la memoria, vienen dadas en forma de funciones polinomiales a trozos muy sencillas de evaluar y de derivar, a un coste computacional bajo. Además, su grado de regularidad las hace especialmente indicadas para aplicaciones a la Industria y la Ingeniería. Por ejemplo, algunas aplicaciones industriales modeladas por problemas del tipo considerado en esta memoria, requieren que las soluciones tengan una primera o incluso una segunda derivada continua, y las soluciones numéricas que hemos obtenido cumplirían con estos requerimientos.

7.2. CONCLUSIONES

Mediante diversos experimentos numéricos, hemos mostrado que se han alcanzado los objetivos iniciales planteados en este proyecto. Además, a mayores, podemos extraer las siguientes conclusiones:

Hemos constatado la gran importancia que tienen problemas que vienen dados en términos de Ecuaciones Diferenciales de

Primer Orden, en muchos ámbitos, especialmente en el de Ingeniería, presentando algunos ejemplos de ello.

Ampliamos nuestros conocimientos relacionados con la resolución de Problemas de Valores Iniciales formulados en términos de Ecuaciones Diferenciales Ordinarias de Primer Orden. Para ello, hemos incluido en la memoria una introducción a algunos de los métodos numéricos clásicos más

utilizados y hemos propuesto y estudiados otros métodos obtenidos mediante una combinación del método de colocación con interpolación mediante funciones polinomiales a trozos.

Hemos implementado en Matlab satisfactoriamente los algoritmos propuestos, obteniendo programas sencillos y con coste computacional asociado bajo.

En algunos casos, los resultados obtenidos al aplicar los métodos implementados en Matlab a los problemas estudiados, son incluso mejores de lo que esperábamos inicialmente.

Se ha cumplido con el siempre deseable carácter formativo inherente a este tipo de proyectos. Es más, este trabajo nos ha permitido iniciarnos en primera persona, en la faceta investigadora (en este caso en Análisis Numérico).

7.3. LÍNEAS DE CONTINUACIÓN

- La línea de trabajo iniciada en este TFG podría ampliarse en el futuro:
- Buscando una extensión de los algoritmos considerados en esta memoria, que permita su aplicación a la resolución de Sistemas de Ecuaciones Diferenciales Ordinarias de Primer Orden.
- Estudiando generalizaciones de los métodos propuestos, que sean aplicables a problemas que involucren otros tipos de ecuaciones diferenciales, por ejemplo, ecuaciones con órdenes diferentes al considerado: Ecuaciones Diferenciales de Segundo Orden, Tercer Orden, etc.
- Estudiando la obtención de métodos basados en Splines y PseudoSplines, pero utilizando, en vez del método de colocación, otro tipo de técnicas distintas. Por ejemplo, podrían utilizarse técnicas de mínimos cuadrados, de Rayleigh-Ritz, etc.
- Perfeccionando y ampliando las técnicas de integración adaptativa con paso variable que hemos introducido en este trabajo.
- Completando, con un estudio matemático más teórico y riguroso, el análisis del orden de los métodos propuestos, que en esta memoria se han abordado desde un punto de vista más empírico y práctico.

8. BIBLIOGRAFÍA

Bernis, J. M. (2013). *Ecuaciones diferenciales y en diferencias finitas.* UNED-Tortosa.

Faires, R. L. (2017). *Análisis numérico.* Cengage.

Mora, J. F. (2012). *Circuitos eléctricos.* Pearson.

O'Connor, J. L. (2017). *Ingeniería de los algoritmos y métodos numéricos.* Circulo ROjo.

Parra, E. (10 de Julio de 2020). *Aula Moisan*. Obtenido de Transitorios eléctricos de primer orden: https://www.aulamoisan.com/software-moisan/transitorios

Rojo, J. (2020). *Pagina personal de Jesus Rojo. Métodos matemáticos I.* Obtenido de Métodos matemáticos: http://wmatem.eis.uva.es/~jesroj/

Ross, S. L. (1981). *Ecuaciones Diferenciales.* Reverté.

Varios. (15 de Julio de 2020). *Documentación web de Matlab*. Obtenido de https://es.mathworks.com

ANEXO 1: NOTACIÓN

El propósito del texto en los capítulos anteriores es sobre todo la claridad. Sin embargo, no hay que olvidar el rigor matemático de las definiciones, ya que en estos ámbitos no se permite ningún tipo de ambigüedad ni de interpretación de los distintos conceptos que usamos.

Notación de la derivada.

En el mundo matemático hay dos notaciones de uso común para representar la derivada de una función como la ley que asocia a cada punto de la función, la pendiente de la recta tangente a la función en dicho punto.

$f'(x)$	$\frac{df}{dx}$
Notación de Lagrange	Notación de Leibniz

Por el tipo de escritura en este texto, y debido a que las ecuaciones son ordinarias, es decir, solo tienen una variable independiente, he escogido la notación de Lagrange.

Sin embargo, no quiero dejar pasar la ocasión para señalar que la notación más adecuada sería la de Leibniz por tres razones:

- Es más cercana a la definición de derivada

En efecto, la definición de derivada se puede escribir como

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

- Es la forma en la que el ordenador trabaja
- Para funciones de más variables independientes, es necesario explicitar respecto de qué variable se efectúa la derivada.

También es de obligado cumplimiento señalar que, mientras que la función derivada es una función continua, cuando la evaluamos en pequeños incrementos de la variable independiente, estamos calculando en realidad la tasa de variación media de $f(x)$ entre los puntos x y $x + \Delta x$.

Esto se puede observar de una manera muy notable en el método de Euler explicado en el capítulo 2.

Intervalo de Interés

Es el intervalo en el que vamos a estudiar la ecuación diferencial.

Viene definido por sus extremos a y b, con $a, b \in \mathbb{R}$ y $a < b$

En este intervalo vamos a realizar una partición en subintervalos, de tal modo que

$$[a, b] = [x_1 = a, \quad x_2, \quad x_3, \dots, \quad x_n, \quad x_{n+1}]$$

A cada punto x_i se le llama nodo, y constituyen un punto frontera entre la definición de la función en ese subintervalo y el intervalo adyacente. Del mismo modo llamamos nodos a los puntos interiores a cada subintervalo en los que aplicaremos alguna condición de colocación.

Con carácter general, sólo deben satisfacer que la unión de todos los subintervalos $[x_i, x_{i+1}]$ sea igual a $[a, b]$, y que $x_i < x_{i+1}$.

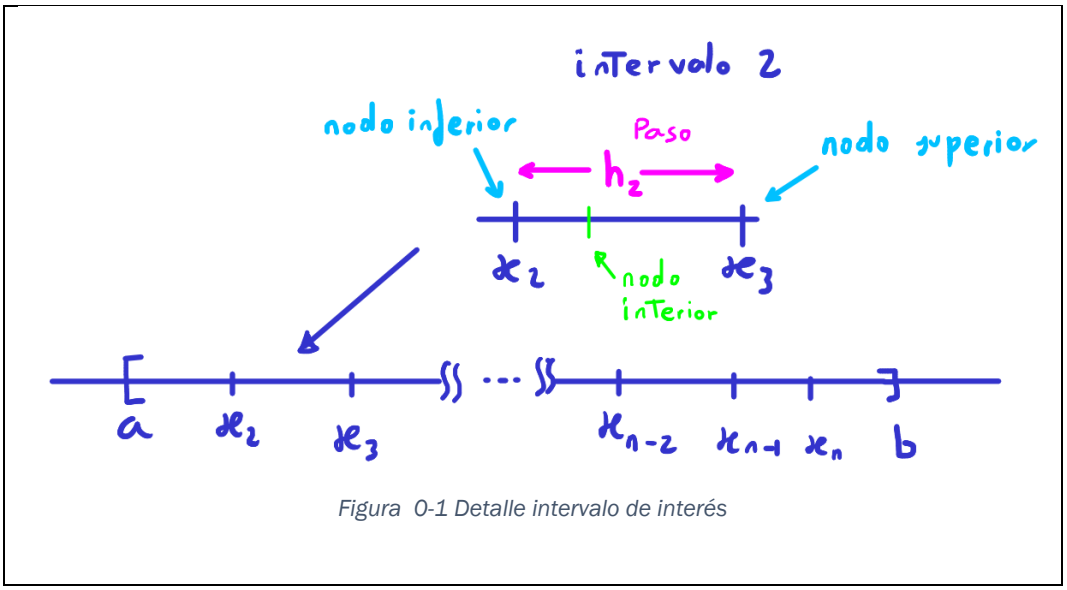


Figura 0-1 Detalle intervalo de interés

Paso

Cuando hablemos de un paso, vamos a estar hablando de la distancia en el eje de abscisas que vamos a avanzar desde el último nodo. De este modo, podemos decir que el paso es la distancia entre dos nodos consecutivos. Y como estamos trabajando con las propiedades de orden de los números reales, es una definición más que suficiente.

Lo denotaremos por

$$h_i = x_{i+1} - x_i$$

Error de truncamiento local

Es el error cometido al avanzar un único paso, suponiendo que al principio del intervalo no se comete error alguno.

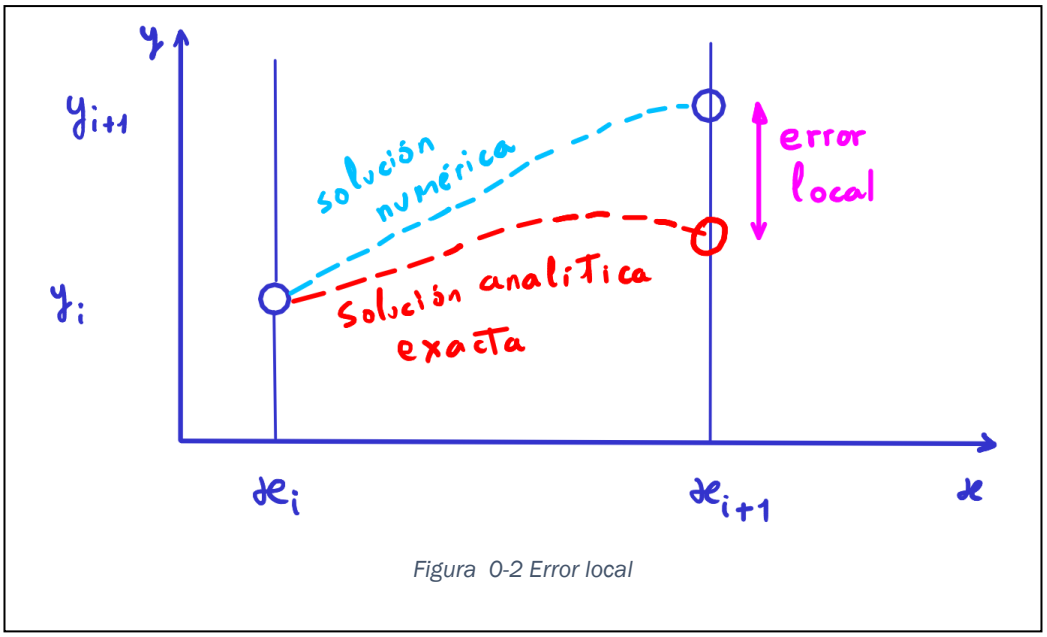


Figura 0-2 Error local

Sin embargo, hay que destacar los tipos de errores que intervienen. Se tiende a pensar que el error local es el error cometido por el método numérico y tiene relación con lo bueno que es.

Sin embargo, existen dos tipos de errores inherentes a estos métodos que son:

- Error de redondeo. Es el que se comete al redondear la última cifra. La única forma de minimizarlo es trabajando con más cantidad de cifras significativas. Matlab utiliza en sus cálculos la mayor cantidad de cifras posible, aunque muestre por pantalla una cifra redondeada.

- Error de truncamiento. Es el que tenemos cuando los cálculos nos dan más cifras significativas de las que se pueden calcular y en algún momento se corta por algún lado, no considerando ya las cifras inferiores.

Error de truncamiento global

Es el error que se comete en la aproximación al avanzar un número n de pasos y, en el caso más extremo, al final del intervalo de interés, que es en teoría el punto más susceptible de tener un error mayor.

Ocurre con frecuencia, que los errores no son acumulativos y a veces los errores que se cometen por exceso, se compensan con los que se cometen por defecto, quedando el error de truncamiento global de una magnitud menor que la suma de los errores de truncamiento locales.

ANEXO 2: CÓDIGOS DE MATLAB

01 – MÉTODO DE EULER

```
function [Y] = EULER_TFG_2020(nodos,qX,rX,z1)

% Descripcion
% La funcion EULER_TFG_2020 implementa un algoritmo para aproximar la
% solución de % una EDO de primer orden lineal mediante aproximaciones
% obtenidas con el método de Euler.
%
%
% INPUTS
% n = vector en el que está definida la partición del intervalo de
%     interes.
% qX,rX = Funciones de definicion de la EDO
%z_1 = Valor de la condicion inicial.

% OUTPUT
% Y = Vector de aproximaciones a f(x)

%
format long     % muestra los coeficientes con formato largo

n=size(nodos,1)-1;

% Predimensionamiento de los vectores

Q= zeros(n+1, 1);      % funcion q(x) evaluada en los nodos
R= zeros(n+1,1);       % funcion r(x) evaluada en los nodos
Y=zeros(size(nodos));
Y(1)=z1;
for i=1:n+1
    Q(i)= qX(nodos(i));  %funcion q(x) evaluada en los (n+1) nodos
    R(i)= rX(nodos(i));  %funcion r(x) evaluada en los (n+1) nodos
end

% Primer intervalo
p=nodos(2)-nodos(1);

B=Q(1)*z1+R(1);
Y(2)=z1+p*B;

% Resto de intervalos

for i=2:n
A=Y(i);
B=Q(i)*A+R(i);
Y(i+1)=A+(nodos(i+1)-nodos(i))*B;


end

end
```

02 – METODO DE RUNGE KUTTA

```
function [y] = RK4_TFG_2020(nodos,qX,rX,z1)
% Proporciona la aproximación mediante un método Runge-Kutta de cuarto
% orden a la solución de un problema de valores iniciales.
% Adaptación de Ido Schwartz

% INPUTS
% n = vector en el que está definida la partición del intervalo de
%     interes.
% qX,rX = Funciones de definicion de la EDO
%z_1 = Valor de la condicion inicial.

% OUTPUT
% Y = Vector de aproximaciones a f(x)


n=size(nodos,1)-1;

% Predimensionamiento de los vectores


y=zeros(size(nodos));
y(1)=z1;
for i=1:n+1
    Q(i)= qX(nodos(i));  %funcion q(x) evaluada en los (n+1) nodos
    R(i)= rX(nodos(i));  %funcion r(x) evaluada en los (n+1) nodos
end                                      % initial condition

F_xy = @(x,y) qX(x)*y+rX(x);
for i=1:n
    h=nodos(i+1)-nodos(i);
    k_1 = F_xy(nodos(i),y(i));
    k_2 = F_xy(nodos(i)+0.5*h,y(i)+0.5*h*k_1);
    k_3 = F_xy((nodos(i)+0.5*h),(y(i)+0.5*h*k_2));
    k_4 = F_xy((nodos(i)+h),(y(i)+k_3*h));
    y(i+1) = y(i) + (1/6)*(k_1+2*k_2+2*k_3+k_4)*h;
end


end
```

03 – SPLINE GRADO 2 LINEAL PARTICION UNIFORME

```
function [coeficientes] = SPLINE2P_TFG_2020(nodos,qX,rX,z1)

% Descripcion
% spline2P_TFG,2020 implementa un algoritmo para aproximar la solución de
% una EDO de primer % orden lineal mediante aproximaciones con splines de
% grado 2.
%
% Esta funcion calcula los coeficientes de un spline de grado 2 en el
% intervalo [a,b], basado en una particion de n+1 nodos contenidos en el
% vector nodos=(nodo_1, nodo_2, ..., nodo_n, nodo_n+1), siendo nodo_1=a y
% nodo_n+1=b
%
% En cada particion [nodo_i, nodo_i+1] el polinomio de grado, a lo sumo 2
% que aproxima la solución de la EDO  esta definido por:
% s_i(x)=a_i+b_i*(x-nodo_i)+c_i*(x-nodo_i)^2
%
%
% INPUTS
% n = vector en el que está definida la partición del intervalo de Interes.
% qX,rX = Funciones de definicion de la EDO
% z_1 = Valor de la condicion inicial.

% OUTPUT
% c= Matriz de dimension n filas y 3 columnas que contiene los coeficientes
% que determinan de manera univoca el spline cuadrático que aproxima la
% solucion de la EDO. La fila k-esima contiene los coeficientes de S_k en
% la base: [(x-x_i)^2, (x-x_i), 1]

%
format long     % muestra los coeficientes con formato largo

n=size(nodos,1)-1;

% Predimensionamiento de los vectores

A= zeros(n,1);          % Vector que almacena los coeficientes a_i
B= zeros(n,1);          % Vector que almacena los coeficientes b_i
C= zeros(n,1);          % Vector que almacena los coeficientes c_i
Q= zeros(n+1, 1)        % funcion q(x) evaluada en los nodos
R= zeros(n+1,1)         % funcion r(x) evaluada en los nodos

for i=1:n+1
    Q(i)= qX(nodos(i));  %funcion q(x) evaluada en los (n+1) nodos
    R(i)= rX(nodos(i));  %funcion r(x) evaluada en los (n+1) nodos
end

% Primer intervalo
p=nodos(2)-nodos(1);
A(1)=z1;
B(1)=Q(1)*A(1)+R(1);
C(1)=(Q(2)*(A(1)+B(1)*p)+R(2)-B(1))/(2*p-Q(2)*(p^2));

% Resto de intervalos

for i=2:n

    p1=nodos(i)-nodos(i-1);
    p2=nodos(i+1)-nodos(i);

    A(i)= A(i-1)+B(i-1)*p1+C(i-1)*p1^2;
    B(i)= B(i-1)+2*C(i-1)*p1;
    C(i)=(Q(i+1)*(A(i)+B(i)*p2)+R(i+1)-B(i))/(2*p2-Q(i+1)*(p2^2));

end
coeficientes=[C,B,A]
end
```

04 - PSEUDO-SPLINE GRADO 3, CLASE C1 LINEAL PARTICION UNIFORME

```
function [coeficientes] = SPLINE3PC1_TFG_2020(nodos,qX,rX,z1)
% Descripcion
% SPLINE3PC1_TFG_2020 implementa un algoritmo para aproximar la solución de
% una EDO de primer orden lineal mediante aproximaciones con Pseudo-Splines
% de grado 3.
%
% Esta funcion calcula los coeficientes de un Pseudo-Spline de grado 3 en el
% intervalo [a,b], basado en una particion de n+1 nodos contenidos en el
% vector nodos=(nodo_1, nodo_2, ..., nodo_n, nodo_n+1), siendo nodo_1=a y
% nodo_n+1=b
%
% En cada particion [nodo_i, nodo_i+1] el polinomio de grado, a lo sumo 3
% que aproxima la solución de la EDO  esta definido por:
% s_i(x)=a_i+b_i*(x-nodo_i)+c_i*(x-nodo_i)^2+d_i*(x-nodo_i)^3
%
%
% INPUTS
% n = vector en el que está definida la partición del intervalo de
%     interes.
% qX,rX = Funciones de definicion de la EDO
%z_1 = Valor de la condicion inicial.

% OUTPUT
% c= Matriz de dimension n filas y 4 columnas que contiene los coeficientes
% que determinan de manera univoca el spline cuadrático que aproxima la
% solucion de la EDO. La fila k-esima contiene los coeficientes de S_k en
% la base: [(x-x_i)^3, (x-x_i)^2, (x-x_i), 1]

%
format long     % muestra los coeficientes con formato largo

n=size(nodos,1)-1;

% Predimensionamiento de los vectores

A= zeros(n,1);        % Vector que almacena los coeficientes a_i
B= zeros(n,1);        % Vector que almacena los coeficientes b_i
C= zeros(n,1);        % Vector que almacena los coeficientes c_i
D= zeros(n,1);


% Primer intervalo
% Definimos los nodos y sus valores asociados
x1=nodos(1);
x3=nodos(2);
x2=x1+(x3-x1)/2;
h1=x2-x1;
h2=x3-x1;
q1=qX(x1);
q2=qX(x2);
q3=qX(x3);
r1=rX(x1);
r2=rX(x2);
r3=rX(x3);
% Ecuaciones de definicion del problema
a=z1;
b=q1*a+r1;
% Simplificamos expresiones de colocacion
alpha1=2*h1-q2*h1^2;
alpha2=3*h1^2-q2*h1^3;
beta1= q2*(a+b*h1)+r2-b;

alpha3=2*h2-q3*h2^2;
alpha4= 3*h2^2-q3*h2^3;
beta2= q3*(a+b*h2)+r3-b;
% COnstruimos sistema de ecuaciones
S=[alpha1, alpha2;alpha3,alpha4]\[beta1;beta2];
c=S(1);
d=S(2);
% Introducimos los valores en su lugar
```

```
A(1)=a;
B(1)=b;
C(1)=c;
D(1)=d;
% Parametros que pasan al siguiente intervalo
continuidad_funcion=a+b*h2+c*h2^2+d*h2^3;
continuidad_derivada= b+2*c*h2+3*d*h2^2;

% Resto de intervalos

for i=2:n


x1=nodos(i);
x3=nodos(i+1);
x2=x1+(x3-x1)/2;
h1=x2-x1;
h2=x3-x1;
q1=qX(x1);
q2=qX(x2);
q3=qX(x3);
r1=rX(x1);
r2=rX(x2);
r3=rX(x3);
% Ecuaciones de definicion del problema
a=continuidad_funcion;
b=continuidad_derivada;
% Simplificamos expresiones de colocacion
alpha1=2*h1-q2*h1^2;
alpha2=3*h1^2-q2*h1^3;
beta1= q2*(a+b*h1)+r2-b;

alpha3=2*h2-q3*h2^2;
alpha4= 3*h2^2-q3*h2^3;
beta2= q3*(a+b*h2)+r3-b;
% COnstruimos sistema de ecuaciones
S=[alpha1, alpha2;alpha3,alpha4]\[beta1;beta2];
c=S(1);
d=S(2);
% Introducimos los valores en su lugar
A(i)=a;
B(i)=b;
C(i)=c;
D(i)=d;

% Parametros que pasan al siguiente intervalo
continuidad_funcion=a+b*h2+c*h2^2+d*h2^3;
continuidad_derivada= b+2*c*h2+3*d*h2^2;

end
coeficientes=[D,C,B,A];
end
```

05 - SPLINE GRADO 3, CLASE C2 LINEAL PARTICION UNIFORME

```
function [coeficientes] = SPLINE2PC2_TFG_2020(nodos,qX,rX,z1)

% Descripcion
% SPLINE2PC2_TFG_2020 implementa un algoritmo para aproximar la solución
% de % una EDO de primer orden lineal mediante aproximaciones con Splines
% de grado 3.
%
% Esta funcion calcula los coeficientes de un Spline de grado 3 en el
% intervalo [a,b], basado en una particion de n+1 nodos contenidos en el
% vector nodos=(nodo_1, nodo_2, ..., nodo_n, nodo_n+1), siendo nodo_1=a y
% nodo_n+1=b
%
% En cada particion [nodo_i, nodo_i+1] el polinomio de grado, a lo sumo 3
% que aproxima la solución de la EDO  esta definido por:
% s_i(x)=a_i+b_i*(x-nodo_i)+c_i*(x-nodo_i)^2+d_i*(x-nodo_i)^3
%
%
% INPUTS
% n = vector en el que está definida la partición del intervalo de
%     interes.
% qX,rX = Funciones de definicion de la EDO
%z_1 = Valor de la condicion inicial.

% OUTPUT
% c= Matriz de dimension n filas y 4 columnas que contiene los coeficientes
% que determinan de manera univoca el spline cuadrático que aproxima la
% solucion de la EDO. La fila k-esima contiene los coeficientes de S_k en
% la base: [(x-x_i)^3, (x-x_i)^2, (x-x_i), 1]

%
format long     % muestra los coeficientes con formato largo

n=size(nodos,1)-1;

% Predimensionamiento de los vectores

A= zeros(n,1);        % Vector que almacena los coeficientes a_i
B= zeros(n,1);        % Vector que almacena los coeficientes b_i
C= zeros(n,1);        % Vector que almacena los coeficientes c_i
D= zeros(n,1);


% Primer intervalo
% Definimos los nodos y sus valores asociados
x1=nodos(1);
x3=nodos(2);
x2=x1+(x3-x1)/2;
h1=x2-x1;
h2=x3-x1;
q1=qX(x1);
q2=qX(x2);
q3=qX(x3);
r1=rX(x1);
r2=rX(x2);
r3=rX(x3);
% Ecuaciones de definicion del problema
a=z1;
b=q1*a+r1;
% Simplificamos expresiones de colocacion
alpha1=2*h1-q2*h1^2;
alpha2=3*h1^2-q2*h1^3;
beta1= q2*(a+b*h1)+r2-b;

alpha3=2*h2-q3*h2^2;
alpha4= 3*h2^2-q3*h2^3;
beta2= q3*(a+b*h2)+r3-b;
% COnstruimos sistema de ecuaciones
S=[alpha1, alpha2;alpha3,alpha4]\[beta1;beta2];
c=S(1);
d=S(2);
% Introducimos los valores en su lugar
A(1)=a;
```

```
B(1)=b;
C(1)=c;
D(1)=d;
% Parametros que pasan al siguiente intervalo
continuidad_funcion=a+b*h2+c*h2^2+d*h2^3;
continuidad_derivada= b+2*c*h2+3*d*h2^2;
continuidad_derivada2=2*c+6*h2*d;

% Resto de intervalos

for i=2:n


x1=nodos(i);
x2=nodos(i+1);
h1=x2-x1;

q2=qX(x2);
r2=rX(x2);

% Ecuaciones de definicion del problema
a=continuidad_funcion;
b=continuidad_derivada;
c=continuidad_derivada2/2;
d=(q2*(a+b*h1+c*h1^2)+r2-b-2*c*h1)/(3*h1^2-q2*h1^3);

% Introducimos los valores en su lugar
A(i)=a;
B(i)=b;
C(i)=c;
D(i)=d;

% Parametros que pasan al siguiente intervalo
continuidad_funcion=a+b*h1+c*h1^2+d*h1^3;
continuidad_derivada= b+2*c*h1+3*d*h1^2;
continuidad_derivada2=2*c+6*h1*d;

end
coeficientes=[D,C,B,A];
end
```

06 – FUNCIONES AUXILIARES DE LOS MÉTODOS PARA PASO VARIABLE

Aquí se detallan las funciones en las que se apoyan los métodos de paso variable, siendo susceptibles de futuras líneas de investigación en lo referente a sistemas de ecuaciones y sistemas de ecuaciones no lineales.

Se van a poner las funciones, pero está bien dar, además, una visión general del método.

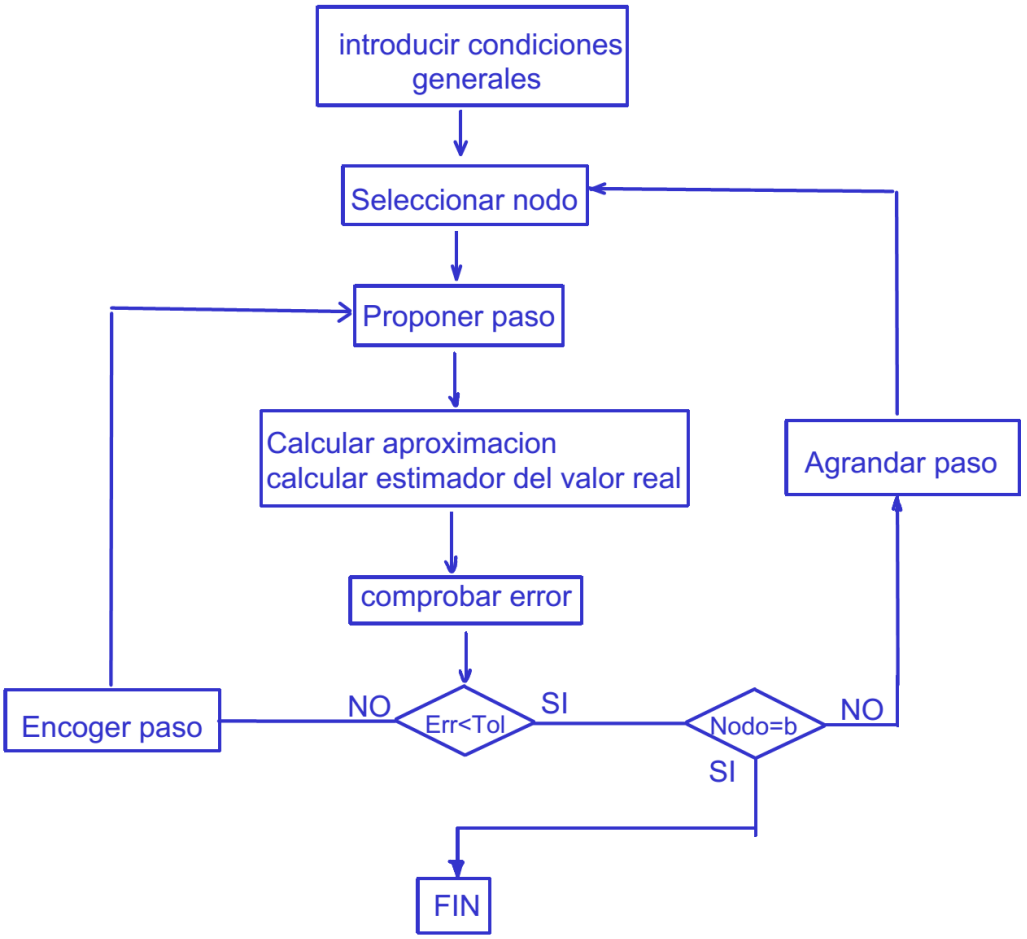


Figura 0-1 Esquema de métodos con paso variable

FUNCION DE AVANCE DE PASO

```
function [PUNTO] = AvanzaIntervalo(x, h, b)
% Devuelve un punto propuesto
PUNTO=x+h;

if abs(b-PUNTO)<=0.1*h || PUNTO>b
    PUNTO=b;
end

end
```

FUNCIÓN QUE CALCULA EL SPLINE INICIAL

```
function [S] = SplineInicial_TFG_2020(x1, x2, z1, qX, rX, grado, Tipo)
%Calcula los grados del polinomio en el primer intervalo

% Selector de las distintas opciones

if strcmp(grado, 'G2') && strcmp(Tipo, 'Lineal')     % Selector
caso= 1;
elseif strcmp(grado, 'G2') && strcmp(Tipo, 'NoLineal')
caso=2;
elseif strcmp(grado, 'G3') && strcmp(Tipo, 'Lineal')
caso=3;
else
caso=4
end        %selector

% Se calculan los elementos comunes
    a= z1;
    b= qX(x1)*a+rX(x1);

% Ramificaciones del programa
switch caso

case 1  % Grado 2 y Lineal

    % a b y c se obtienen de la resolucion de un sistema 3x3 lineal

    h1= x2-x1;
    c= (qX(x2)*(a+b*h1)-b+rX(x2))/(2*h1-qX(x2)*h1^2);

    S=[a;b;c]

case 2   % Grado2 y No Lineal

   % a y b asignacion directa
   % c se estima inicialmente en 0 y se resuelve de manera no lineal

    h1= x2-x1;

    c_est=(qX(x2)*(a+b*h1)-b+rX(x2))/(2*h1-qX(x2)*h1^2);

    F1= @(c) qX(x2)*(a+b*h1+c*h1^2)+rX(x2)-b-2*c*h1;
    c= fsolve(F1, c_est);

    S=[a;b;c];

case 3    % Grado 3 y Lineal
```

```
    % preasignacion de elementos

    h2= x2-x1; h1= h2/2;
    q2=qX(x1+h2); r2= rX(x1+h2);
    q3= qX(x2); r3= rX(x2);

    % Escritura de ecuaciones
    alpha1=2*h1-q2*h1^2; alpha2= 3*h1^2-q2*h1^3;
    alpha3=2*h2-q3*h2^2; alpha4= 3*h2^2-q2*h1^3;
    beta1= q2*(a+b*h1)+r2-b; beta2= q3*(a+b*h2)+r3-b;

    % Sistema de ecuaciones
    C=[alpha1, alpha2; alpha3, alpha4]\[beta1;beta2];

    % Asignacion de salida
    S=[a; b; C(1); C(2)];


otherwise  % Grado 3 No lineal

    % preasignacion de elementos

    h2= x2-x1; h1= h2/2;
    q2=qX(x1+h1); r2= rX(x1+h1);
    q3= qX(x2); r3= rX(x2);

    alpha1=2*h1-q2*h1^2; alpha2= 3*h1^2-q2*h1^3;
    alpha3=2*h2-q3*h2^2; alpha4= 3*h2^2-q3*h2^3;
    beta1= q2*(a+b*h1)+r2-b; beta2= q3*(a+b*h2)+r3-b;


    % Escritura de las ecuaciones
    Fun= @(u) [alpha1*u(1)+alpha2*u(2)-beta1;alpha3*u(1)+alpha4*u(2)-
beta2];

    % Estimadores iniciales
    alpha1=2*h1-q2*h1^2; alpha2= 3*h1^2-q2*h1^3;
    alpha3=2*h2-q3*h2^2; alpha4= 3*h2^2-q3*h2^3;
    beta1= q2*(a+b*h1)+r2-b; beta2= q3*(a+b*h2)+r3-b;
    C_est=[alpha1, alpha2; alpha3, alpha4]\[beta1;beta2];

    [u,v]= fsolve(Fun, [C_est(1);C_est(2)]);

    % Solucion
    S=[a; b; u(1); u(2)];

end  % switch

% Salida de datos
S=fliplr(S')

end  %function
```

FUNCIÓN QUE CALCULA EL SPLINE SIGUIENTE

```
function [S] = SplineSiguiente_TFG_2020(x0, x1, x2, Sant, qX, rX, grado,
clase, Tipo)

%Calcula los grados del polinomio en los intervalos siguientes

% Selector de las distintas opciones

if strcmp(grado, 'G2') && strcmp(Tipo, 'Lineal')     % Selector
    caso= 1;
elseif strcmp(grado, 'G2') && strcmp(Tipo, 'NoLineal')
    caso=2;
elseif strcmp(grado, 'G3') && strcmp(Tipo, 'Lineal')
    caso=3;
else
    caso=4;
end        %selector

% Se calculan los elementos comunes


% Ramificaciones del programa
switch caso

    case 1  % Grado 2 y Lineal
        % Obtenemos los coeficientes a, b y c de manera directa
        a= Sant*[(x1-x0)^2;(x1-x0);1];
        b= Sant*[2*(x1-x0);1;0];
        h1= x2-x1;
        c= (qX(x2)*(a+b*h1)-b+rX(x2))/(2*h1-qX(x2)*h1^2);

        % Introducimos resultados en la salida
        S=[a;b;c];

    case 2   % Grado2 y No Lineal

        % Obtenemos a y b de manera directa
        a= Sant*[(x1-x0)^2;(x1-x0);1];
        b= Sant*[2*(x1-x0);1;0];

        h1= x2-x1;
        % Estimamos c para arranque del método
        c_est=(qX(x2)*(a+b*h1)-b+rX(x2))/(2*h1-qX(x2)*h1^2);

        % Construimos las entradas para fsolve
        F1= @(c) qX(x2)*(a+b*h1+c*h1^2)+rX(x2)-b-2*c*h1;
        c= fsolve(F1, c_est);

        % Introducimos resultados en la salida
        S=[a;b;c];

    case 3    % Grado 3 y Lineal

        a= Sant*[(x1-x0)^3;(x1-x0)^2;(x1-x0);1];
        b= Sant*[3*(x1-x0)^2; 2*(x1-x0);1;0];

        if strcmp(clase, 'C1')
            %preasignacion
            h2= x2-x1; h1= h2/2;
            q2=qX(x1+h1); r2= rX(x1+h1);
            q3= qX(x2); r3= rX(x2);

            % Escritura de ecuaciones
```

```
            alpha1=2*h1-q2*h1^2; alpha2= 3*h1^2-q2*h1^3;
            alpha3=2*h2-q3*h2^2; alpha4= 3*h2^2-q3*h2^3;
            beta1= q2*(a+b*h1)+r2-b; beta2= q3*(a+b*h2)+r3-b;

            % Sistema de ecuaciones
            C=[alpha1, alpha2; alpha3, alpha4]\[beta1;beta2];

            % Introducimos resultados en la salida
            S=[a; b; C(1); C(2)];



        else  % Lineal clase C2
            % preasignacion
            h1= x2-x1;
            q2=qX(x2); r2= rX(x2);

            c= Sant*[3*(x1-x0); 1;0;0];

            d=(q2*(a+b*h1+c*h1^2)+r2-b-2*c*h1)/(3*h1^2-q2*h1^3);
            % Introducimos resultados en la salida
            S=[a; b; c; d];
        end



    otherwise  % Grado 3 No lineal

        % Obtenemos a y b de manera directa
        a= Sant*[(x1-x0)^3;(x1-x0)^2;(x1-x0);1];
        b= Sant*[3*(x1-x0)^2; 2*(x1-x0);1;0];


        if strcmp(clase, 'C1')  %No lineal Clase C1
            % preasignacion
            h2= x2-x1; h1= h2/2;
            q2=qX(x1+h1); r2= rX(x1+h1);
            q3= qX(x2); r3= rX(x2);

            % construccion de la estructura del sistema
            alpha1=2*h1-q2*h1^2; alpha2= 3*h1^2-q2*h1^3;
            alpha3=2*h2-q3*h2^2; alpha4= 3*h2^2-q3*h2^3;
            beta1= q2*(a+b*h1)+r2-b; beta2= q3*(a+b*h2)+r3-b;


            % Escritura de las ecuaciones
            Fun= @(u) [alpha1*u(1)+alpha2*u(2)-
beta1;alpha3*u(1)+alpha4...
                *u(2)-beta2];

            % Estimadores iniciales
            alpha1=2*h1-q2*h1^2; alpha2= 3*h1^2-q2*h1^3;
            alpha3=2*h2-q3*h2^2; alpha4= 3*h2^2-q3*h2^3;
            beta1= q2*(a+b*h1)+r2-b; beta2= q3*(a+b*h2)+r3-b;
            C_est=[alpha1, alpha2; alpha3, alpha4]\[beta1;beta2];

            % Resolución
            [u,v]= fsolve(Fun, [C_est(1);C_est(2)]);

            % Introducimos resultados en la salida
            S=[a; b; u(1); u(2)];

        else   %No lineal clase C2

            % Preasignacion
```

```
            h1= x2-x1;
            q2=qX(x2); r2= rX(x2);

            % Obtencion directa de c
            c= Sant*[3*(x1-x0); 1;0;0];

            % estimacion de d
            d_est=(q2*(a+b*h1+c*h1^2)+r2-b-2*c*h1)/(3*h1^2-q2*h1^3);

            % Entradas para fsolve
            Fun= @(u) b+2*c*h1+3*u*h1^2-q2*(a+b*h1+c*h1^2+u*h1^3)-r2;
            [u, v]= fsolve(Fun, d_est);

            % Introducimos resultados en la salida
            S=[a; b; c; u(1)];

        end


end  % switch

% Salida de datos en el orden correcto
S=fliplr(S');

end  %function
```

FUNCIÓN AUXILIAR DE REPRESENTACIÓN

```
function [Triplete] = EligeColor(color)


switch color
    case 'rojo1'
      Triplete=[1, 0, 0];
    case 'rojo5'
      Triplete=[0.9792, 0.5000, 0.4453];
    case 'azul1'
      Triplete=[0, 0, 0.5000];
    case 'azul2'
      Triplete=[0, 0, 0.8008];
    case 'azul3'
      Triplete=[0.1172, 0.5625, 1.0000];
    case 'azul5'
      Triplete=[0.5273, 0.8047, 0.9792];
    case 'rosa1'
      Triplete=[1, 0, 1];
    otherwise
end
```

FUNCIÓN SPLINE G2

```
function [NODOS,COEFICIENTES] = SplineG2_TFG_2020(a, b, h, z1, qX, rX,
Tipo)
% Esta funcion nos devuelve el vector de nodos de la particion resultante
% en el intervalo que va desde el nodo inicial hasta el extremo final
% que consideramos y asociado a cada subintervalo nos devuelve los
% coeficientes del polinomio asociado.

grado='G2';
clase='C1';

% Inicializamos nodos y coeficientes

Nodos01=[a];
Coeficientes01=zeros(1,3);

while Nodos01(end)<b   % Hasta el fin del intervalo

    indice=size(Nodos01,1); %Numero de elemento

    Nodo_prop=AvanzaIntervalo(Nodos01(end), h, b);  % Nodo propuesto

    if indice==1  % Subintervalo Inicial

        Coef_Provisional= SplineInicial_TFG_2020(Nodos01(end), Nodo_prop,
z1, qX, rX, grado, Tipo);

    else       % Subintervalos siguientes

        x0=Nodos01(indice-1); x1=Nodos01(indice); x2=Nodo_prop;
        Sant=Coeficientes01(indice-1,:);
        Coef_Provisional= SplineSiguiente_TFG_2020(x0, x1, x2, Sant, qX,
rX, grado, clase, Tipo);

    end  %if 01

    Nodos01(indice+1,1) = Nodo_prop;
    Coeficientes01(indice,:)=Coef_Provisional;

end  %while 01

% Definimos la salida
NODOS= Nodos01;
COEFICIENTES= Coeficientes01;

end
```

FUNCIÓN SPLINE G3 C1

```
function [NODOS,COEFICIENTES] = SplineG3C1_TFG_2020(a, b, h, z1, qX, rX,
Tipo)
% Esta funcion nos devuelve el vector de nodos de la particion resultante
% en el intervalo que va desde el nodo inicial hasta el extremo final
% que consideramos y asociado a cada subintervalo nos devuelve los
% coeficientes del polinomio asociado.
grado='G3';
clase='C1';

% Inicializamos nodos y coeficientes
Nodos01=[a];
Coeficientes01=zeros(1,4);

while Nodos01(end)<b   % Hasta el fin del intervalo

    indice=size(Nodos01,1); %Numero de elemento

    Nodo_prop=AvanzaIntervalo(Nodos01(end), h, b);  % Nodo propuesto

    if indice==1  % Subintervalo Inicial
        Coef_Provisional= SplineInicial_TFG_2020(Nodos01(end), Nodo_prop,
z1, qX, rX, grado, Tipo);

    else       % Subintervalos siguientes siguientes

        x0=Nodos01(indice-1);
        x1=Nodos01(indice);
        x2=Nodo_prop;
        Sant=Coeficientes01(indice-1,:);
        Coef_Provisional= SplineSiguiente_TFG_2020(x0, x1, x2, Sant, qX,
rX, grado, clase, Tipo);

    end  %if 01
    Nodos01(indice+1,1) = Nodo_prop;
    Coeficientes01(indice,:)=Coef_Provisional;

end  %while 01

% Definimos la salida
NODOS= Nodos01;
COEFICIENTES= Coeficientes01;


end
```

FUNCIÓN SPLINE G3 C2

```
function [NODOS,COEFICIENTES] = SplineG3C2_TFG_2020(a, b, h, z1, qX, rX,
Tipo)
% Esta funcion nos devuelve el vector de nodos de la particion resultante
% en el intervalo que va desde el nodo inicial hasta el extremo final
% que consideramos y asociado a cada subintervalo nos devuelve los
% coeficientes del polinomio asociado.
grado='G3';
clase='C2';

% Inicializamos nodos y coeficientes
Nodos01=[a];
Coeficientes01=zeros(1,4);

while Nodos01(end)<b   % Hasta el fin del intervalo

    indice=size(Nodos01,1); %Numero de elemento

    Nodo_prop=AvanzaIntervalo(Nodos01(end), h, b);  % Nodo propuesto

    if indice==1  % Subintervalo Inicial
        Coef_Provisional= SplineInicial_TFG_2020(Nodos01(end), Nodo_prop,
z1, qX, rX, grado, Tipo);

    else       % Subintervalos siguientes siguientes

        x0=Nodos01(indice-1);
        x1=Nodos01(indice);
        x2=Nodo_prop;
        Sant=Coeficientes01(indice-1,:);
        Coef_Provisional= SplineSiguiente_TFG_2020(x0, x1, x2, Sant, qX,
rX, grado, clase, Tipo);

    end  %if 01
    Nodos01(indice+1,1) = Nodo_prop;
    Coeficientes01(indice,:)=Coef_Provisional;

end  %while 01

% Definimos la salida
NODOS= Nodos01;
COEFICIENTES= Coeficientes01;


end
```

FUNCION PARES ENCAJADOS 23

```
function [NODOS,COEFICIENTES] = ParesEncajadosMSpline23_TFG_2020(a, b, h,
z1, qX, rX, Tipo, Atol, Ce)
% Esta función implementa el método de pares encajados para resolver
% una ecuación diferencial adaptando el paso mediante un método de orden
% 2 y un método de orden 4 (Splines de grado 2 y PseudoSplines de grado
% 3.
% Permite seleccionar el método de avance, y parámetro de control por
% angulo y coeficientes de seguridad.


%Alto= false;
Alto= true;

Cs=0.9;

if Alto

    clase='C1';
    Aceptado=0;
    Rechazado=0;
    Eacumulado=0;

    Utol=Atol/(b-a);  %Tolerancia unitaria
    hmax=(b-a)/10;

    % Inicializamos nodos y coeficientes
    Nodos01=[a];
    Coeficientes01=zeros(1,4);

    while Nodos01(end)<b   % Hasta el fin del intervalo

        indice=size(Nodos01,1); %Numero de elemento
        Nodo_prop=AvanzaIntervalo(Nodos01(end), h, b);  % Nodo propuesto


        if indice==1  % Subintervalo Inicial

            Coef_Provisional_1= SplineInicial_TFG_2020(Nodos01(end),
Nodo_prop, z1, qX, rX, 'G2', Tipo);
            Coef_Provisional_2= SplineInicial_TFG_2020(Nodos01(end),
Nodo_prop, z1, qX, rX, 'G3', Tipo);


        else       % Subintervalos siguientes siguientes

            x0=Nodos01(indice-1); x1=Nodos01(indice); x2=Nodo_prop;
            Sant=Coeficientes01(indice-1,:);
            Coef_Provisional_2= SplineSiguiente_TFG_2020(x0, x1,
Nodo_prop, Sant, qX, rX,'G3', clase, Tipo);
            Coef_Provisional_1=zeros(1,3);
            Coef_Provisional_1(3)=Coef_Provisional_2(4);
            Coef_Provisional_1(2)=Coef_Provisional_2(3);

Coef_Provisional_1(1)=(qX(x2)*(Coef_Provisional_1(3)+Coef_Provisional_1(2
)*(x2-x1))+rX(x2)-Coef_Provisional_1(2))/(2*(x2-x1)-qX(x2)*(x2-x1)^2);

        end  %if 01

        % definimos los tamaños de intervalo de los Spline 1 y 3
        h1=Nodo_prop-Nodos01(end);

        % Calculamos el valor estimado  MODO NORMAL
        y1=Coef_Provisional_1*[h1^2;h1;1];
```

```
        y2=Coef_Provisional_2*[h1^3;h1^2;h1;1];

        % Calculamos el error estimado (Mitad del intervalo)
        %y1=Coef_Provisional*[h2^3; h2^2;h2;1];
        %y2=Coef2*[h2^3;h2^2;h2;1];

        % Comprobamos error
        Stol=Utol*h1;
        Err=abs(y1-y2);

        % modificamos el error Si Ce=0 --> distancia entre funciones
        %                      Si Ce=100 --> Mayor exigencia (Richardson)
        Ce=(100-Ce);

angle=Ce*(atan(Coef_Provisional_2(3)+2*Coef_Provisional_2(2)*h1+3*Coef_Pr
ovisional_2(1)*h1^2))/100;
        modificador=cos(angle);
        Err=modificador*Err;

        if Err<Stol

            Aceptado=Aceptado+1;
            Nodos01(indice+1,1) =  Nodo_prop;
            Coeficientes01(indice,:)=Coef_Provisional_2;

            Eacumulado=Eacumulado+Err;

        elseif Err>Stol

             Rechazado=Rechazado+1;

         end

        alpha=Cs*(Stol/Err)^(1/2);

        if alpha<1/5

            alpha=1/5;

        elseif alpha>3

            alpha=3;
        end

        h=alpha*h;

        if h>hmax

            h=hmax;

        end



    end  %while 01

    % Definimos la salida
    NODOS= Nodos01;
    COEFICIENTES= Coeficientes01;



else
    % Avanzamos con el de grado bajo
```

```
    clase='C1';
    Aceptado=0;
    Rechazado=0;
    Eacumulado=0;

    Utol=Atol/(b-a);  %Tolerancia unitaria
    hmax=(b-a)/10;

    % Inicializamos nodos y coeficientes
    Nodos01=[a];
    Coeficientes01=zeros(1,3);

    while Nodos01(end)<b   % Hasta el fin del intervalo

        indice=size(Nodos01,1); %Numero de elemento
        Nodo_prop=AvanzaIntervalo(Nodos01(end), h, b);  % Nodo propuesto


        if indice==1  % Subintervalo Inicial

            Coef_Provisional_1= SplineInicial_TFG_2020(Nodos01(end),
Nodo_prop, z1, qX, rX, 'G2', Tipo);
            Coef_Provisional_2= SplineInicial_TFG_2020(Nodos01(end),
Nodo_prop, z1, qX, rX, 'G3', Tipo);


        else       % Subintervalos siguientes siguientes

            x0=Nodos01(indice-1); x1=Nodos01(indice); x2=Nodo_prop;
            Sant=Coeficientes01(indice-1,:);
            Coef_Provisional_1= SplineSiguiente_TFG_2020(x0, x1,
Nodo_prop, Sant, qX, rX,'G2', clase, Tipo);
            Sant=[0, Sant];
            Coef_Provisional_2= SplineSiguiente_TFG_2020(x0, x1,
Nodo_prop, Sant, qX, rX,'G3', clase, Tipo);
        end  %if 01

        % definimos los tamaños de intervalo de los Spline 1 y 3
        h1=Nodo_prop-Nodos01(end);

        % Calculamos el valor estimado  MODO NORMAL
        y1=Coef_Provisional_1*[h1^2;h1;1];
        y2=Coef_Provisional_2*[h1^3;h1^2;h1;1];

        % Calculamos el error estimado (Mitad del intervalo)
        %y1=Coef_Provisional*[h2^3; h2^2;h2;1];
        %y2=Coef2*[h2^3;h2^2;h2;1];

        % Comprobamos error
        Stol=Utol*h1;
        Err=abs(y1-y2);

        % modificamos el error Si Ce=0 --> distancia entre funciones
        %                      Si Ce=100 --> Mayor exigencia (Richardson)
        Ce=(100-Ce);

angle=Ce*(atan(Coef_Provisional_2(3)+2*Coef_Provisional_2(2)*h1+3*Coef_Pr
ovisional_2(1)*h1^2))/100;
        modificador=cos(angle);
        Err=modificador*Err;

        if Err<Stol

            Aceptado=Aceptado+1;
            Nodos01(indice+1,1) =  Nodo_prop;
```

```
            Coeficientes01(indice,:)=Coef_Provisional_1;

            Eacumulado=Eacumulado+Err;

        elseif Err>Stol

            Rechazado=Rechazado+1;

        end

        alpha=Cs*(Stol/Err)^(1/2);

        if alpha<1/5

            alpha=1/5;

        elseif alpha>3

            alpha=3;
        end

        h=alpha*h;

        if h>hmax

            h=hmax;

        end


    end  %while 01

    % Definimos la salida
    NODOS= Nodos01;
    COEFICIENTES= Coeficientes01;


end

end
```

FUNCIÓN RICHARDSON G2

```
function [NODOS,COEFICIENTES] = RichMSplineG2_TFG_2020(a, b, h, z1, qX,
rX, Tipo, Atol, Ce)
% Esta funcion nos devuelve el vector de nodos de la particion resultante
% en el intervalo que va desde el nodo inicial hasta el extremo final
% que consideramos y asociado a cada subintervalo nos devuelve los
% coeficientes del polinomio asociado. Tiene la capacidad de adaptar el
% paso y estimar el error mediante el método de extrapolación de
% Richardson
% Permite seleccionar el método de avance, y parámetro de control por
% angulo y coeficientes de seguridad.

Alto=true;
%Alto=false;

Cs=1;

if Alto
    grado='G2';
    clase='C1';
    Aceptado=0;
    Rechazado=0;
    Eacumulado=0;

    Utol=Atol/(b-a);  %Tolerancia unitaria
    hmax=(b-a)/10;

    % Inicializamos nodos y coeficientes
    Nodos01=[a];
    Coeficientes01=zeros(1,3);

    while Nodos01(end)<b   % Hasta el fin del intervalo

        indice=size(Nodos01,1); %Numero de elemento
        Nodo_prop=AvanzaIntervalo(Nodos01(end), h, b);  % Nodo propuesto

        prop2=Nodos01(end)+(Nodo_prop-Nodos01(end))/2;
        prop3=Nodo_prop;

        if indice==1  % Subintervalo Inicial

            Coef_Provisional= SplineInicial_TFG_2020(Nodos01(end),
Nodo_prop, z1, qX, rX, grado, Tipo);
            Coef2=SplineInicial_TFG_2020(Nodos01(end), prop2, z1, qX, rX,
grado, Tipo);
            Coef3= SplineSiguiente_TFG_2020(Nodos01(end), prop2, prop3,
Coef2, qX, rX, grado, clase, Tipo);

        else       % Subintervalos siguientes siguientes

            x0=Nodos01(indice-1); x1=Nodos01(indice); x2=Nodo_prop;
            Sant=Coeficientes01(indice-1,:);
            Coef_Provisional= SplineSiguiente_TFG_2020(x0, x1, x2, Sant,
qX, rX, grado, clase, Tipo);

            Coef2= SplineSiguiente_TFG_2020(x0, x1, prop2, Sant, qX, rX,
grado, clase, Tipo);
            Coef3= SplineSiguiente_TFG_2020(x1, prop2, prop3, Coef2, qX,
rX, grado, clase, Tipo);

        end  %if 01

        % definimos los tamaños de intervalo de los Spline 1 y 3
        h1=Nodo_prop-Nodos01(end);
```

```
        h2=prop3-prop2;

        % Calculamos el valor estimado  MODO NORMAL
        y1=Coef_Provisional*[h1^2;h1;1];
        y2=Coef3*[h2^2;h2;1];

        % Calculamos el error estimado (Mitad del intervalo)
        %y1=Coef_Provisional*[h2^2;h2;1];
        %y2=Coef2*[h2^2;h2;1];

        % Comprobamos error
        Stol=Utol*h1;
        Err=abs(y1-y2);

        %modificamos el error

        Ce=(100-Ce);

angle=Ce*(atan(Coef_Provisional(2)+2*Coef_Provisional(1)*h1))/100;
        modificador=cos(angle);
        Err=modificador*Err;

        if Err<Stol

            Aceptado=Aceptado+1;
            Nodos01(indice+1,1) = prop2;
            Nodos01(indice+2,1) = prop3;
            Coeficientes01(indice,:)=Coef2;
            Coeficientes01(indice+1,:)=Coef3;
            Eacumulado=Eacumulado+Err;


        elseif Err>Stol

            Rechazado=Rechazado+1;

        end

        alpha=Cs*(Stol/Err)^(1/2);

        if alpha<1/5

            alpha=1/5;

        elseif alpha>3

            alpha=3;
        end

        h=alpha*h;

        if h>hmax

            h=hmax;
        end



    end  %while 01

    % Definimos la salida
    NODOS= Nodos01;
    COEFICIENTES= Coeficientes01;
```

```
else

    grado='G2';
    clase='C1';
    Aceptado=0;
    Rechazado=0;
    Eacumulado=0;

    Utol=Atol/(b-a);  %Tolerancia unitaria
    hmax=(b-a)/10;

    % Inicializamos nodos y coeficientes
    Nodos01=[a];
    Coeficientes01=zeros(1,3);

    while Nodos01(end)<b   % Hasta el fin del intervalo

        indice=size(Nodos01,1); %Numero de elemento
        Nodo_prop=AvanzaIntervalo(Nodos01(end), h, b);  % Nodo propuesto

        prop2=Nodos01(end)+(Nodo_prop-Nodos01(end))/2;
        prop3=Nodo_prop;

        if indice==1  % Subintervalo Inicial

            Coef_Provisional= SplineInicial_TFG_2020(Nodos01(end),
Nodo_prop, z1, qX, rX, grado, Tipo);
            Coef2=SplineInicial_TFG_2020(Nodos01(end), prop2, z1, qX, rX,
grado, Tipo);
            Coef3= SplineSiguiente_TFG_2020(Nodos01(end), prop2, prop3,
Coef2, qX, rX, grado, clase, Tipo);

        else   % Subintervalos siguientes siguientes

            x0=Nodos01(indice-1); x1=Nodos01(indice); x2=Nodo_prop;
            Sant=Coeficientes01(indice-1,:);
            Coef_Provisional= SplineSiguiente_TFG_2020(x0, x1, x2, Sant,
qX, rX, grado, clase, Tipo);

            Coef2= SplineSiguiente_TFG_2020(x0, x1, prop2, Sant, qX, rX,
grado, clase, Tipo);
            Coef3= SplineSiguiente_TFG_2020(x1, prop2, prop3, Coef2, qX,
rX, grado, clase, Tipo);

        end  %if 01

        % definimos los tamaños de intervalo de los Spline 1 y 3
        h1=Nodo_prop-Nodos01(end);
        h2=prop3-prop2;

        % Calculamos el valor estimado  MODO NORMAL
        y1=Coef_Provisional*[h1^2;h1;1];
        y2=Coef3*[h2^2;h2;1];

        % Calculamos el error estimado (Mitad del intervalo)
        %y1=Coef_Provisional*[h2^2;h2;1];
        %y2=Coef2*[h2^2;h2;1];

        % Comprobamos error
        Stol=Utol*h1;
        Err=abs(y1-y2);

        %modificamos el error

        Ce=(100-Ce);
```

```
angle=Ce*(atan(Coef_Provisional(2)+2*Coef_Provisional(1)*h1))/100;
        modificador=cos(angle);
        Err=modificador*Err;

        if Err<Stol

            Aceptado=Aceptado+1;
            Nodos01(indice+1,1) = Nodo_prop;
            Coeficientes01(indice,:)=Coef_Provisional;
            Eacumulado=Eacumulado+Err;

        elseif Err>Stol

            Rechazado=Rechazado+1;

            alpha=Cs*(Stol/Err)^(1/2);

            if alpha<1/5

                alpha=1/5;

            elseif alpha>3

                alpha=3;

            end

            h=alpha*h;

            if h>hmax

                h=hmax;

            end

        end  %while 01

        % Definimos la salida
        NODOS= Nodos01;
        COEFICIENTES= Coeficientes01;

    end

end
```

FUNCIÓN RICHARDSON G3

```
function [NODOS,COEFICIENTES] = RichMSplineG3C1_TFG_2020(a, b, h, z1, qX,
rX, Tipo, Atol, Ce)
% Esta funcion nos devuelve el vector de nodos de la particion resultante
% en el intervalo que va desde el nodo inicial hasta el extremo final
% que consideramos y asociado a cada subintervalo nos devuelve los
% coeficientes del polinomio asociado. Tiene la capacidad de adaptar el
% paso y estimar el error mediante el método de extrapolación de
% Richardson

% Permite seleccionar el método de avance, y parámetro de control por
% angulo y coeficientes de seguridad.


%Alto= false;
Alto= true;

Cs=1;

if Alto



    grado='G3';
    clase='C1';
    Aceptado=0;
    Rechazado=0;
    Eacumulado=0;

    Utol=Atol/(b-a);  %Tolerancia unitaria
    hmax=(b-a)/10;

    % Inicializamos nodos y coeficientes
    Nodos01=[a];
    Coeficientes01=zeros(1,4);

    while Nodos01(end)<b   % Hasta el fin del intervalo

        indice=size(Nodos01,1); %Numero de elemento
        Nodo_prop=AvanzaIntervalo(Nodos01(end), h, b);  % Nodo propuesto

        prop2=Nodos01(end)+(Nodo_prop-Nodos01(end))/2;
        prop3=Nodo_prop;

        if indice==1  % Subintervalo Inicial

            Coef_Provisional= SplineInicial_TFG_2020(Nodos01(end),
Nodo_prop, z1, qX, rX, grado, Tipo);

            Coef2=SplineInicial_TFG_2020(Nodos01(end), prop2, z1, qX, rX,
grado, Tipo);
            Coef3= SplineSiguiente_TFG_2020(Nodos01(end), prop2, prop3,
Coef2, qX, rX, grado, clase, Tipo);

        else       % Subintervalos siguientes siguientes

            x0=Nodos01(indice-1); x1=Nodos01(indice); x2=Nodo_prop;
            Sant=Coeficientes01(indice-1,:);
            Coef_Provisional= SplineSiguiente_TFG_2020(x0, x1, x2, Sant,
qX, rX, grado, clase, Tipo);

            Coef2= SplineSiguiente_TFG_2020(x0, x1, prop2, Sant, qX, rX,
grado, clase, Tipo);
```

```
            Coef3= SplineSiguiente_TFG_2020(x1, prop2, prop3, Coef2, qX,
rX, grado, clase, Tipo);

        end  %if 01

        % definimos los tamaños de intervalo de los Spline 1 y 3
        h1=Nodo_prop-Nodos01(end);
        h2=prop3-prop2;
        % Calculamos el valor estimado  MODO NORMAL
        y1=Coef_Provisional*[h1^3;h1^2;h1;1];
        y2=Coef3*[h2^3;h2^2;h2;1];

        % Calculamos el error estimado (Mitad del intervalo)
        %y1=Coef_Provisional*[h2^3; h2^2;h2;1];
        %y2=Coef2*[h2^3;h2^2;h2;1];
        % Comprobamos error
        Stol=Utol*h1;
        Err=abs(y1-y2);

        %modificamos el error

        Ce=(100-Ce);

angle=Ce*(atan(Coef_Provisional(3)+2*Coef_Provisional(2)*h1+3*Coef_Provis
ional(1)*h1))/100;
        modificador=cos(angle);
        Err=modificador*Err;

        if Err<Stol

            Aceptado=Aceptado+1;
            Nodos01(indice+1,1) = prop2;
            Nodos01(indice+2,1) = prop3;

            Coeficientes01(indice,:)=Coef2;
            Coeficientes01(indice+1,:)=Coef3;
            Eacumulado=Eacumulado+Err;

        elseif Err>Stol

            Rechazado=Rechazado+1;

          end

        alpha=Cs*(Stol/Err)^(1/2);

        if alpha<1/5

            alpha=1/5;

        elseif alpha>3

            alpha=3;

        end

        h=alpha*h;

        if h>hmax

            h=hmax;
        end

    end  %while 01
```

```
    % Definimos la salida

    NODOS= Nodos01;

    COEFICIENTES= Coeficientes01;


else


    grado='G3';
    clase='C1';
    Aceptado=0;
    Rechazado=0;
    Eacumulado=0;

    Utol=Atol/(b-a);  %Tolerancia unitaria
    hmax=(b-a)/10;

    % Inicializamos nodos y coeficientes
    Nodos01=[a];
    Coeficientes01=zeros(1,4);

    while Nodos01(end)<b   % Hasta el fin del intervalo

        indice=size(Nodos01,1); %Numero de elemento
        Nodo_prop=AvanzaIntervalo(Nodos01(end), h, b);  % Nodo propuesto

        prop2=Nodos01(end)+(Nodo_prop-Nodos01(end))/2;
        prop3=Nodo_prop;

        if indice==1  % Subintervalo Inicial

            Coef_Provisional= SplineInicial_TFG_2020(Nodos01(end),
Nodo_prop, z1, qX, rX, grado, Tipo);

            Coef2=SplineInicial_TFG_2020(Nodos01(end), prop2, z1, qX, rX,
grado, Tipo);
            Coef3= SplineSiguiente_TFG_2020(Nodos01(end), prop2, prop3,
Coef2, qX, rX, grado, clase, Tipo);

        else       % Subintervalos siguientes siguientes

            x0=Nodos01(indice-1); x1=Nodos01(indice); x2=Nodo_prop;
            Sant=Coeficientes01(indice-1,:);
            Coef_Provisional= SplineSiguiente_TFG_2020(x0, x1, x2, Sant,
qX, rX, grado, clase, Tipo);

            Coef2= SplineSiguiente_TFG_2020(x0, x1, prop2, Sant, qX, rX,
grado, clase, Tipo);
            Coef3= SplineSiguiente_TFG_2020(x1, prop2, prop3, Coef2, qX,
rX, grado, clase, Tipo);

        end  %if 01

        % definimos los tamaños de intervalo de los Spline 1 y 3
        h1=Nodo_prop-Nodos01(end);
        h2=prop3-prop2;

        % Calculamos el valor estimado  MODO NORMAL
        y1=Coef_Provisional*[h1^3;h1^2;h1;1];
        y2=Coef3*[h2^3;h2^2;h2;1];

        % Calculamos el error estimado (Mitad del intervalo)
        %y1=Coef_Provisional*[h2^3; h2^2;h2;1];
```

```
        %y2=Coef2*[h2^3;h2^2;h2;1];

        % Comprobamos error
        Stol=Utol*h1;
        Err=abs(y1-y2);

        %modificamos el error

        Ce=(100-Ce);

angle=Ce*(atan(Coef_Provisional(3)+2*Coef_Provisional(2)*h1+3*Coef_Provis
ional(1)*h1))/100;
        modificador=cos(angle);
        Err=modificador*Err;

        if Err<Stol
            paso=h1;

            Aceptado=Aceptado+1;
            Nodos01(indice+1,1) = Nodo_prop;
            Coeficientes01(indice,:)=Coef_Provisional;
            Eacumulado=Eacumulado+Err;
            % disp('Se acepta el paso')
        elseif Err>Stol
            paso=h1;
            Err;
            Stol;
            Rechazado=Rechazado+1;
            %   disp('Se rechaza el paso');
        end

        alpha=Cs*(Stol/Err)^(1/2);
        if alpha<1/5
            alpha=1/5;
        elseif alpha>3
            alpha=3;
        end
        h=alpha*h;

        if h>hmax
            h=hmax;
        end

    end  %while 01

    % Definimos la salida
    NODOS= Nodos01;
    COEFICIENTES= Coeficientes01;

end
end
```

ANEXO 3: FUNCIONES ESPECIALES DE MATLAB

(Varios, 2020)

mkpp

Esta función implementada en Matlab recoge la información de la partición creada y de los coeficientes de los polinomios asociados a cada subintervalo y construye una función a trozos para el intervalo de interés.

Su sintaxis es

```
pp = mkpp(breaks,coefs)
```

Lo normal es trabajar con coeficientes en orden creciente, pero esta función los tiene implementados al contrario, es decir, la última columna de coeficientes corresponde a los valores aproximados de la función, la penúltima a los que se encuentran afectados por la x, la antepenúltima por la x^2, … y así sucesivamente.

Para una partición de 100 subintervalos nos muestra lo siguiente.

```
struct with fields:

    form: 'pp'
  breaks: [1×101 double]
   coefs: [100×3 double]
  pieces: 100
   order: 3
     dim: 1
```

Básicamente esto nos indica el tipo de estructura especial (Piecewise Polynomial), la secuencia de nodos, la secuencia de coeficientes y sus características.

Se puede consultar más información en el siguiente enlace de documentación de Matlab.

https://es.mathworks.com/help/curvefit/the-ppform.html

Guía de selección de solvers en Matlab

Tabla 0-1 Guia de selección de solvers en Matlab

Solver	Problem Type	Accuracy	When to Use
ode45	Nonstiff	Medium	Most of the time. ode45 should be the first solver you try.
ode23		Low	ode23 can be more efficient than ode45 at problems with crude tolerances, or in the presence of moderate stiffness.
ode113		Low to High	ode113 can be more efficient than ode45 at problems with stringent error tolerances, or when the ODE function is expensive to evaluate.
ode15s	Stiff	Low to Medium	Try ode15s when ode45 fails or is inefficient and you suspect that the problem is stiff. Also use ode15s when solving differential algebraic equations (DAEs).
ode23s		Low	ode23s can be more efficient than ode15s at problems with crude error tolerances. It can solve some stiff problems for which ode15s is not effective.
			ode23s computes the Jacobian in each step, so it is beneficial to provide the Jacobian via odeset to maximize efficiency and accuracy.
			If there is a mass matrix, it must be constant.
ode23t		Low	Use ode23t if the problem is only moderately stiff and you need a solution without numerical damping.
			ode23t can solve differential algebraic equations (DAEs).
ode23tb		Low	Like ode23s, the ode23tb solver might be more efficient than ode15s at problems with crude error tolerances.
ode15i	Fully implicit	Low	Use ode15i for fully implicit problems $f(t,y,y') = 0$ and for differential algebraic equations (DAEs) of index 1.

https://es.mathworks.com/help/matlab/math/choose-an-ode-solver.html

(Varios, 2020)

fsolve

Es la función de referencia que usamos para la resolución de sistemas de ecuaciones.

Es una función generalista y de amplio alcance. La particularidad que tiene es que se adapta a casi cualquier sistema de ecuaciones y las resuelve con unos valores de tolerancia muy pequeños. Esto es, que no encontramos la solución exacta de nuestro sistema, sino que la vamos aproximando hasta que sea lo suficientemente pequeña.

Podemos hacernos a la idea de que es una adaptación del famoso método de Newton para encontrar el cero de una ecuación. Lo que hace es aproximar la función de manera local y extrapolarla hacia un conjunto de soluciones en las que se espera que la solución se encuentre más cerca. Luego se va repitiendo iterativamente este sistema hasta conseguir que haya una convergencia.

```
x = fsolve(fun,x0)
```

fun

Es el sistema de ecuaciones que queremos resolver

X0

Es el estimador inicial que usa el método