PEC2. Interpolación y derivación numérica

Juan José Rodríguez Aldavero
Métodos Numéricos en Ingeniería

Resumen

En este documento, correspondiente a la segunda Prueba de Evaluación Continua de la asignatura Métodos Numéricos en Ingeniería, se aplicarán métodos de interpolación sobre diferentes casos particulares y se estudiará la precisión procedente de aplicar un método de derivación numérica.

1. Ejercicio 1

Aplicamos distintos métodos de interpolación para estimar los valores de una función desconocida sobre tres puntos dados. En particular, conocemos los valores de la función sobre los siguientes 11 puntos test:

x_i	$f(x_i)$
-1.0	0.038462
-0.8	0.058824
-0.6	0.100000
-0.4	0.200000
-0.2	0.500000
0.0	1.000000
0.2	0.500000
0.4	0.200000
0.6	0.100000
0.8	0.058824
1.0	0.038462

y pretendemos estimar los valores sobre $x_1 = -0{,}95$, $x_2 = 0{,}11$ y $x_3 = 0{,}75$. Comenzamos realizando una interpolación mediante splines cúbicos. Esta interpolación está basada en calcular k polinomios cúbicos $s_k(x)$ denominados splines en la forma:

$$s_k(x) = S_{k,1} + S_{k,2}x + S_{k,3}x^2 + S_{k,4}x^3$$

que cumplen la particularidad de que la función resultante de unir estos polinomios trozo-a-trozo (*piecewise*) es continua, y además presenta una primera y segunda derivada continuas. Basándonos en la bibliografía aportada por la asignatura, vemos que podemos distinguir tres casos particulares para este método en función de las condiciones de contorno:

- Especificando la primera derivada, $s_0'(x_0) = S_{0,1}, s_N'(x_N) = S_{N,1}$
- Especificando la segunda derivada, $s_0''(x_0) = 2S_{0,2}, s_N''(x_N) = 2S_{N,2}$
- Extrapolando la segunda derivada.

Hemos dicho que para este caso particular desconocemos la función que da origen a los puntos test. En realidad, esto no es así ya que sabemos que esta función es:

$$f(x) = \frac{1}{1 + 25x^2}$$

y por tanto podemos emplear la interpolación por splines especificando la primera derivada en la forma:

$$f'(-1) \equiv S_{0,1} = 0{,}073965 \quad f'(1) \equiv S_{N,1} = -0{,}073965$$

Sin embargo es importante destacar que en un caso general esto no sería así y habría que buscar otra forma de realizar la interpolación. Gracias a esto, vemos en la bibliografía que realizar la interpolación por splines se reduce a un problema de álgebra lineal en el cual el objetivo es calcular los coeficientes de la parte lineal, $S_{k,2}$ dados por el sistema de ecuaciones:

$$\begin{bmatrix} 2h_0 & h_0 & 0 & \cdot & \cdot \\ h_0 & 2(h_0+h_1) & h_1 & \ddots & \vdots \\ 0 & \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & h_{N-2} & 2(h_{N-2}+h_{N-1}) & h_{N-1} \\ \cdot & \cdot & 0 & h_{N-1} & 2h_{N-1} \end{bmatrix} \begin{bmatrix} S_{0,2} \\ S_{1,2} \\ \cdot \\ S_{N-1,2} \\ S_{N,2} \end{bmatrix} = \begin{bmatrix} 3(dy_0 - S_{0,1}) \\ 3(dy_1 - dy_0) \\ \cdot \\ 3(dy_{N-1} - dy_{N-2}) \\ 3(S_{N,1} - dy_{N-1}) \end{bmatrix}$$

donde $h_k = x_{k+1} - x_k$ son las separaciones entre puntos, y $\frac{y_{k+1}-y_k}{h_k} = dy_k$ son las diferencias divididas. Podemos adaptar el código dado por la bibliografía para este caso particular, y este se adjuntará al trabajo. A partir de este, podemos obtener los siguientes valores:

$$y_1 = 0{,}040728 \quad y_2 = 1{,}000009 \quad y_3 = 0{,}099392$$

Procedemos a calcular estos valores a partir de un polinomio de grado máximo. Para ello utilizamos la interpolación de Newton, que es un método iterativo para construir un polinomio:

$$n_N(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots$$

Construido iterativamente en la forma:

$$n_1(x) = n_0(x) + a_1(x - x_0)$$

donde podemos dar los coeficientes en términos de diferencias divididas:

$$a_N = \frac{D^{N-1}f_1 - D^{N-1}f_0}{x_N - x_0} \equiv D^N f_0$$

De nuevo nos referimos al código de la asignatura para hallar los valores:

$$y_1 = 1{,}924 \quad y_2 = 0{,}8134 \quad y_3 = -0{,}2315$$

Vemos como el valor y_1 es muy elevado debido al efecto Runge

Comparamos los valores de la interpolación con los valores de la función exacta, dados por

$$f(x) = \frac{1}{1 + 25x^2}$$

$$y_1 = 0{,}042440 \quad y_2 = 0{,}767754 \quad y_3 = 0{,}066390$$

Finalmente repetimos el mismo procedimiento utilizando un mayor número de puntos test. Para el caso de los splines cúbicos obtenemos los valores:

$$y_1 = 0{,}039633 \quad y_2 = 0{,}800000 \quad y_3 = 0{,}075555$$

considerablemente más precisos que para el caso anterior debido a la mayor cantidad de puntos; y para el polinomio de Newton:

$$y_1 = -0{,}95 \quad y_2 = 0{,}767 \quad y_3 = -0{,}4471$$

Más preciso para el punto centrado pero considerablemente erróneo en los puntos próximos a la frontera, debido al efecto Runge. Podemos representar las gráficas de los dos polinomios interpoladores y la función original para ver:

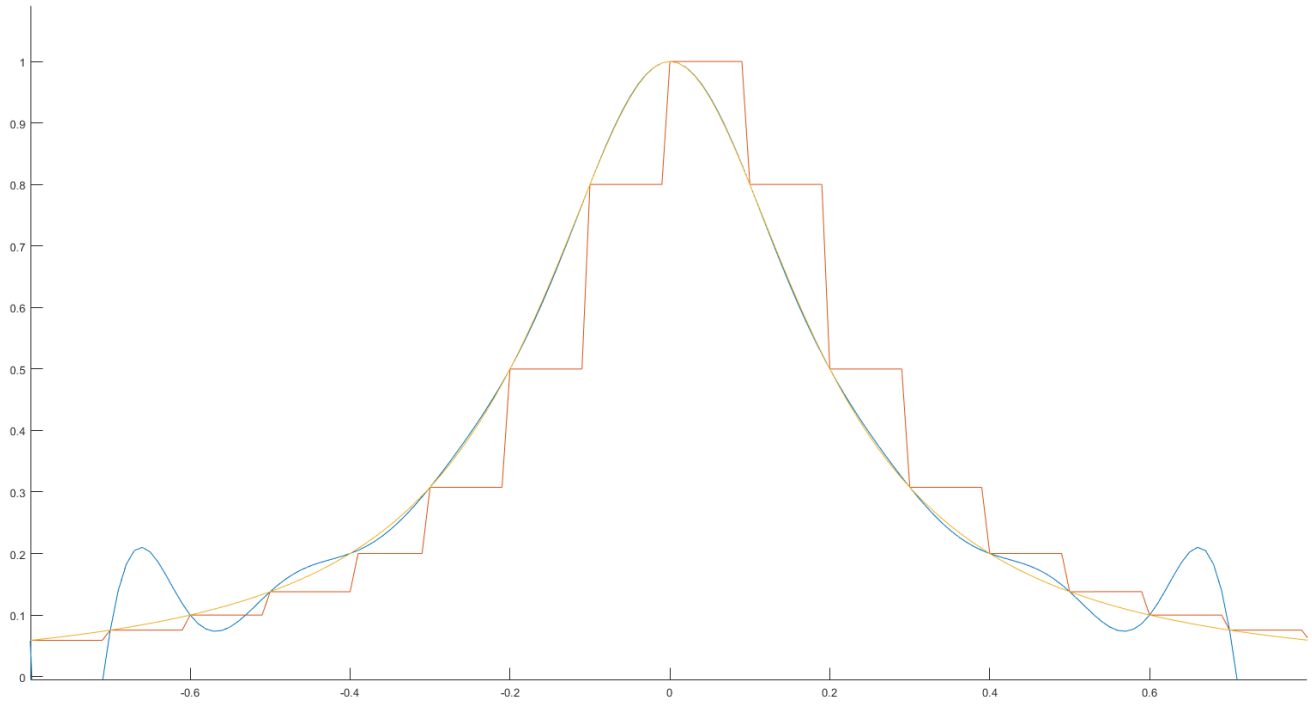

Figura 1: Amarillo: original. Azul: Newton. Rojo: splines.

El gráfico de los splines cúbicos no se muestra correctamente por dificultades propias al manejar el lenguaje Matlab.

2. Ejercicio 2

Dada la función

$$f(x) = \arctan(x)$$

procedemos a calcular el polinomio interpolante de Newton de grado $n \in \{2, 4, 10, 20\}$. Para ello, calculamos un conjunto de n puntos equiespaciados entre los valores $x \in (-1, 1)$ y aplicamos el método de Newton anteriormente descrito. El resultado es el siguiente:

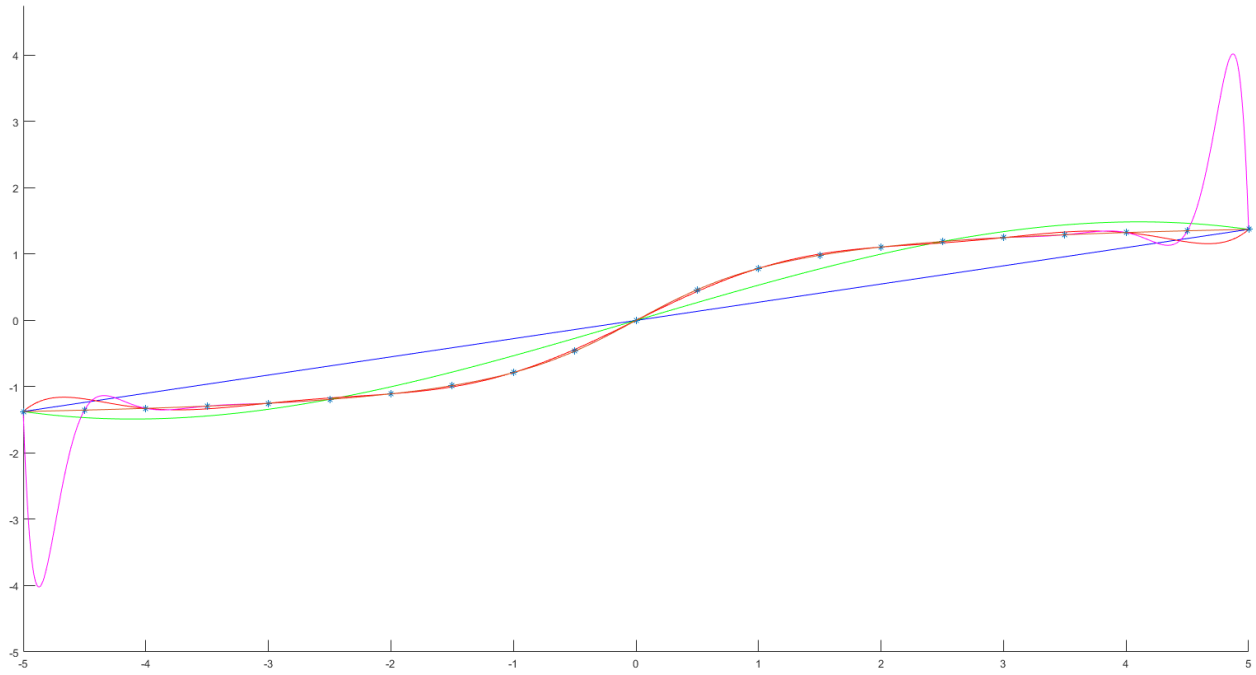

Figura 2: Rojo: función original. El resto, polinomios de Newton

Vemos como los polinomios de Newton sufren el efecto Runge y tienen desviaciones importantes en los extremos, sobretodo a medida que aumentamos el grado del polinomio.

A continuación hacemos lo mismo pero utilizando nodos de Chebyshev en lugar de nodos equiespaciados. Estos nodos tienen la característica de disminuir el efecto de Runge. :

Por claridad lo hacemos únicamente para el polinomio de grado 20. El resultado es:

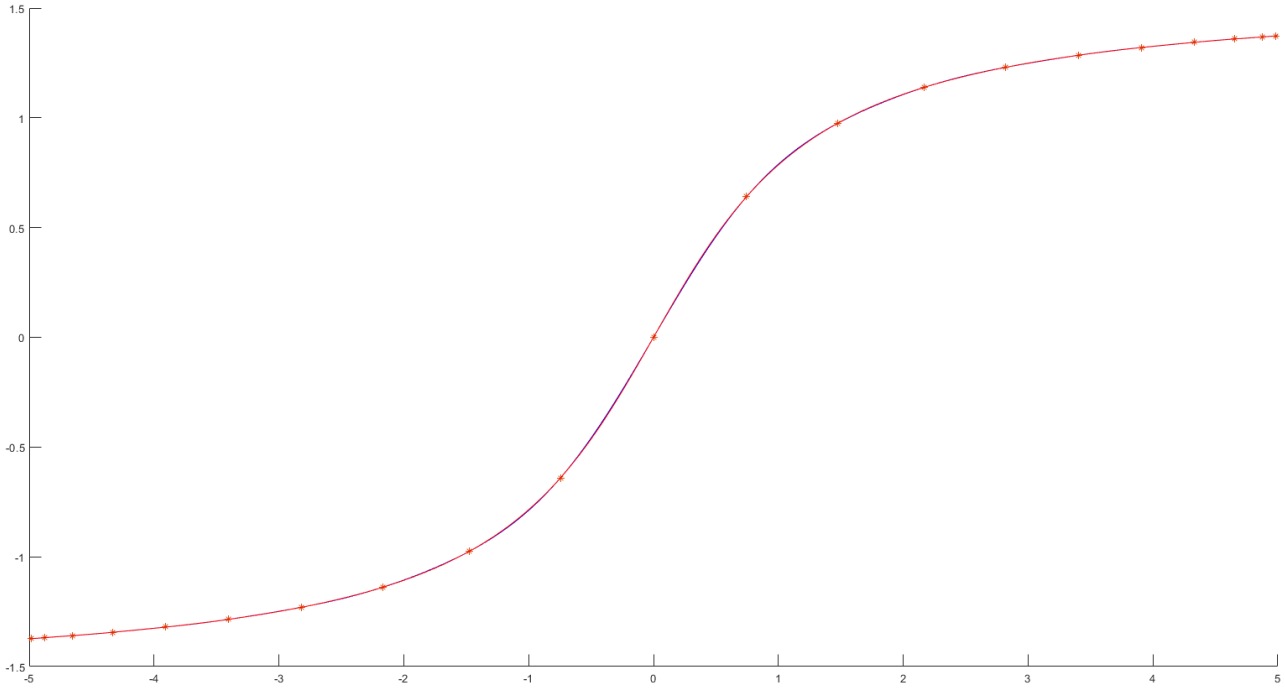

Figura 3: Rojo: función original. Azul: polinomio de Newton sobre nodos de Chebyshev.

3. Ejercicio 3

Estudiamos la precisión de las fórmulas aproximadas de derivación

$$f'(x) \simeq \frac{f(x+h) - f(x)}{h}$$

$$f'(x) \simeq \frac{-25f(x) + 48f(x+h) - 36f(x+2h) + 16f(x+3h) - 3f(x+4h)}{12h}$$

para la función

$$f(x) = e^{-3x}\cos(2x)$$

en el punto $x = 0$ con valores $h = 4^{-n}$, $1 \leq n \leq 15$. Para ello, empleamos un script en Matlab proporcionado con este documento. El resultado es:

	Valor exacto	Primera aproximación	Segunda aproximación
$n = 1$	-3	-2.341837401976866	-3.086893644263229
$n = 2$	-3	-2.839027886019505	-3.001255639566224
$n = 3$	-3	-2.960589909689432	-3.000006505053553
$n = 4$	-3	-2.990211781200344	-3.000000027187980
$n = 5$	-3	-2.997557167851824	-3.000000000107548
$n = 6$	-3	-2.999389559102838	-2.999999999999091
$n = 7$	-3	-2.999847406523259	-3.000000000000606
$n = 8$	-3	-2.999961852678098	-2.999999999997575
$n = 9$	-3	-2.999990463227732	-2.999999999883585
$n = 10$	-3	-2.999997615814209	-3.000000000038805
$n = 11$	-3	-2.999999403953552	-3
$n = 12$	-3	-2.999999850988388	-3
$n = 13$	-3	-2.999999962747097	-2.999999980131785
$n = 14$	-3	-2.999999970197678	-2.999999920527140
$n = 15$	-3	-3	-3.000000039736430

Vemos como la primera aproximación converge más lentamente mientras que la segunda lo hace con mayor velocidad hasta llegar al resultado exacto (dentro de el error de representación de la maquina $\xi_r = 2^{-48} \approx 0{,}36 \cdot 10^{-14}$) entre las iteraciones 5 y 6. Sin embargo, una vez alcanza este valor oscila alrededor de él, posiblemente debido a factores relacionados con la propagación del error sobre h.