FACULTAD
DE
CIENCIAS

INTERPOLACIÓN NUMÉRICA. ¿QUÉ HAY DETRÁS DE CHEBFUN?

(NUMERICAL INTERPOLATION. WHAT ARE THE MATHEMATICAL SUPPORTS OF CHEBFUN?)

Trabajo de fin de Grado
para acceder al

GRADO EN MATEMÁTICAS

Autor: Jorge Gómez González

Directora: Cecilia Pola Méndez

Junio-2017

Resumen

Este trabajo se centra en algunas cuestiones de interpolación polinomial relacionadas con la computación numérica y que serán ilustradas con el software Chebfun en MATLAB. En particular presentamos algunas propiedades de distintos conjuntos de nodos de interpolación como son, los puntos equiespaciados y los puntos de Chebyshev de primera y segunda especie. Por otra parte, también consideramos las ventajas de los polinomios de Chebyshev respecto de los monomios en el ámbito computacional. En cuanto al software Chebfun, analizamos cómo determina el grado adecuado del polinomio interpolador, cómo calcula los coeficientes que lo definen usando la transformada rápida de Fourier (FFT) y, finalmente, cómo evalúa dicho polinomio con un algoritmo estable y rápido basado en la fórmula del baricentro.

Palabras clave: interpolación polinomial, nodos Chebyshev, polinomios de Chebyshev, transformada rápida de Fourier, fórmula del baricentro.

Abstract

This work focuses on some polynomial interpolation issues close to numerical computation. These questions will be illustrated computationally with the help of Chebfun software package in MATLAB. In particular, we will consider several sets of interpolation nodes such as the equispaced points, the first-kind Chebyshev points and the second-kind ones and we will analyse some of their properties. On the other hand, we will highlight the advantages of Chebyshev polynomials with respect to the monomials. Moreover, we will analyse how the Chebfun software establishes the right degree of the polynomial interpolant and, how it computes its coefficients by using the fast Fourier transform (FFT) and, finally we will see how it evaluates that polynomial by using a fast and stable algorithm which relies on the barycentric formula.

Key words: polynomial interpolation, Chebyshev nodes, Chebyshev polynomials, Fast Fourier Transform, barycentric formula.

Índice general

1. Introducción **7**

2. Puntos y polinomios de Chebyshev **11**
2.1. Puntos de Chebyshev 11
2.2. Polinomios de Chebyshev 17
2.3. Constante de Lebesgue en interpolación 20

3. Chebfun **25**
3.1. Elementos chebfun 25
3.2. ¿Por qué elegir los puntos de Chebyshev de segunda especie? 28
3.3. Series y polinomios de Chebyshev 31
3.4. Transformada Rápida de Fourier 35
3.5. Fórmula del baricentro 39

Capítulo 1

Introducción

Este trabajo esta inspirado en los dos primeros apartados del artículo "Six Myths of Polynomial Interpolation and Quadrature", escrito por Lloyd N. Trefethen (ver [9]). En el primero de ellos se desmiente la no convergencia, en general, del interpolador polinómico cuando el número de nodos tiende a infinito. Se afirma que, para funciones Lipschitz continuas, los polinomios interpolantes en nodos de Chebyshev siempre convergen. El segundo apartado del artículo desmiente que la evaluación de interpoladores polinomiales sea numéricamente problemática en general, basándose en el uso de un algoritmo estable y computacionalmente rápido, conocido como fórmula del baricentro, que se desarrolla a partir de la fórmula de interpolación de Lagrange. En este punto es interesante el hecho de que el algoritmo implementado en MATLAB en las funciones `polyfit\polyval` está basado en el uso de matrices de Vandermonde, que como veremos no es estable.

Antes de continuar, recordamos que la **interpolación polinómica**, uno de los temas principales del análisis numérico, consiste en, a partir de un conjunto de $n+1$ datos, $\{(x_j, f_j)\}_{j=1}^{n}$ con $x_0 < x_1 < \ldots < x_n$, calcular el polinomio $p(x)$ de grado n verificando que $p(x_j) = f_j$ para $j = 1, \ldots, n$. Los valores x_j se denominan puntos o nodos de interpolación y los valores f_j pueden ser las imágenes de una función determinada en los nodos de interpolación anteriormente citados.

Hasta ahora, durante los estudios del Grado en Matemáticas, hemos visto diferentes formas de representar el polinomio interpolador, como son, la fórmula de Lagrange y la de Newton. En lo referente a la interpolación de Chebyshev se han introducido los polinomios de Chebyshev y los nodos de Chebyshev de primera especie, viendo su utilidad en la interpolacion de funciones como la de Runge. En este trabajo, por contra, nos centraremos en la interpolación en nodos de Chebyshev de segunda especie, siguiendo el libro "Approximation Theory and Approximation Practice"(ver [10]). Este es un libro sobre aproximación en el cual la mayoría de temas están computacionalmente ilustrados con el software Chebfun (ver [1]), que está implementado en MATLAB.

El software Chebfun, de código libre que nació en el año 2005 y ya va por la versión 5.7.0

en 2017, es una de las piezas claves de este trabajo. Nuestro objetivo consiste en estudiar su soporte matemático en el ámbito de la interpolación. Para empezar notemos que el comando `chebfun` de este software representa una función mediante el correspondiente polinomio interpolador en los $n+1$ nodos de Chebyshev de segunda especie:

$$p(x) = \sum_{k=0}^{n} c_k T_k(x),$$

donde $T_k(x)$ son los polinomios de Chebyshev. Además, una peculiaridad de Chebfun es que, a parte de haber creado comandos propios como `chebfun`, ha reescrito varios comandos de MATLAB, como por ejemplo `plot` o `norm`, para elementos chebfun que en MATLAB fueron diseñados inicialmente para matrices y vectores.

En concreto veremos las razones por las cuales en la práctica computacional Chebfun elige los nodos de Chebyshev de segunda especie, cuando teóricamente, los nodos de Chebyshev de primera especie son los que minimizan la cota del error de interpolación. Por otra parte, describiremos los polinomios de Chebyshev y veremos por qué son más útiles en el entorno de la interpolación que los monomios.

En lo referente a la forma que usa Chebfun para calcular los coeficientes de Chebyshev, c_k, hay que señalar que dicho software utiliza la transformada discreta de Fourier (DFT), cuyo algoritmo esta implementado en MATLAB como la transformada rápida de Fourier (ver función `fft`). En [10] no se detalla este proceso, pero nosotros veremos cómo se relaciona la interpolación de Chebyshev con la interpolación con polinomios trigonométricos y cómo usar la FFT para calcular dichos coeficientes.

Finalmente, cabe reseñar que el software Chebfun utiliza la fórmula de interpolación del baricentro en nodos de Chebyshev de segunda especie para evaluar numéricamente el polinomio interpolador. Como aparece en [4] la interpolación con la fórmula del baricentro, a pesar de no ser algo nuevo, es poco conocida en el mundo matemático. La razón por la cual se usa esta fórmula para evaluar el polinomio interpolador recae en su estabilidad numérica y en que el coste computacional es $O(n)$ frente al de la fórmula de Lagrange o de Newton que es de $O(n^2)$.

Este trabajo se estructura en tres capítulos. A continuación damos una breve descripción del contenido de los dos siguientes.

- El capítulo 2, en el cual se establecen los elementos matemáticos básicos usados por Chebfun, consta de tres secciones. En la primera de ellas se definen los distintos tipos de puntos de Chebyshev; en cuanto a los nodos de Chebyshev de segunda especie, se compara su formulación habitual en los libros de análisis con la formulación que utiliza Chebfun y, en cuanto a los nodos de Chebyshev de primera especie, se destaca su importancia en el marco de la interpolación, además se realiza una comparación de los errores de la interpolación con ambos nodos de Chebyshev y de los de la interpolación con nodos equiespaciados. En la segunda sección se definen los polinomios de Chebyshev y se describe su relación con ambos tipos de nodos de Chebyshev;

también se destaca su mejor condicionamiento frente a la base formada por los monomios. Finalmente, en la tercera sección se considera la constante de Lebesgue en interpolación y se comparan sus valores para ambos tipos de nodos de Chebyshev y para nodos equiespaciados.

- El capítulo 3, que se centra en lo referente al software Chebfun, está formado por cinco secciones. En la primera de ellas se describe cómo es un elemento del tipo chebfun con varios ejemplos. Dentro de la segunda sección podemos apreciar una razón, basada en el abaratamiento de coste computacional, por la cual Chebfun utiliza los nodos de Chebyshev de segunda especie en detrimento de los de primera especie y que está relacionada con la forma en la que se elige el grado del polinomio interpolador. En la tercera sección se establece que toda función Lipschitz continua tiene una única representación como serie de Chebyshev y se prueba la relación existente entre los coeficientes de la serie y los coeficientes del polinomio interpolador. Posteriormente en la cuarta sección se expone el proceso por el cual Chebfun calcula los coeficientes del polinomio interpolador utilizando la transformada rápida de Fourier como ya anunciamos. En el último apartado de este capítulo se describe la fórmula que utiliza Chebfun para evaluar el polinomio interpolador en un punto, que está basada en la fórmula de interpolación del baricentro para los nodos de Chebyshev de segunda especie, citada anteriormente.

Destacar que lo anteriormente expuesto está completado con varios ensayos numéricos, cuyos programas aparecen en el apéndice al final del trabajo y que además se han completado varias demostraciones que en la referencia original no aparecían totalmente desarrolladas.

A lo largo de la memoria, sin pérdida de la generalidad, se considera que las funciones están definidas en el intervalo $[-1, 1]$.

Capítulo 2

Puntos y polinomios de Chebyshev

En este capítulo vamos a considerar los elementos matemáticos que utiliza Chebfun (ver capítulo 3) para realizar la interpolación de funciones.

2.1. Puntos de Chebyshev

Definición 2.1.1 (Puntos de Chebyshev de segunda especie)**.** *Dado $n \in \mathbb{N}$, los $n+1$* ***puntos de Chebyshev de segunda especie*** *en el intervalo $[-1,1]$ vienen dados por la siguiente fórmula*

$$z_j = cos\left(\frac{j\pi}{n}\right), \quad 0 \leq j \leq n. \tag{2.1}$$

Estos puntos de Chebyshev pueden generarse a partir de la elección de puntos equiespaciados en la semicircunferencia unidad superior dentro del plano complejo, si posteriormente se elige la proyección de los puntos anteriormente citados en el eje de abcisas, es decir, la parte real, como se puede observar en la figura 2.1.

Ahora trataremos de justificar por qué el software Chebfun (ver capítulo 3) calcula los puntos de Chebyshev con la siguiente sentencia de MATLAB, que aparece en el código del fichero chebpts.m,

$$x = sin\left(\pi\frac{(-n+1:2:n-1)}{2(n-1)}\right),$$

que es equivalente a la fórmula que aparece a continuación,

$$x_j = sin\left(\frac{(2j-n-1)\pi}{2(n-1)}\right), \qquad 1 \leq j \leq n. \tag{2.2}$$

Proposición 2.1.1. *Dado $n \in \mathbb{N}$, la expresión (2.2) genera los n puntos de Chebyshev de segunda especie en $[-1,1]$.*

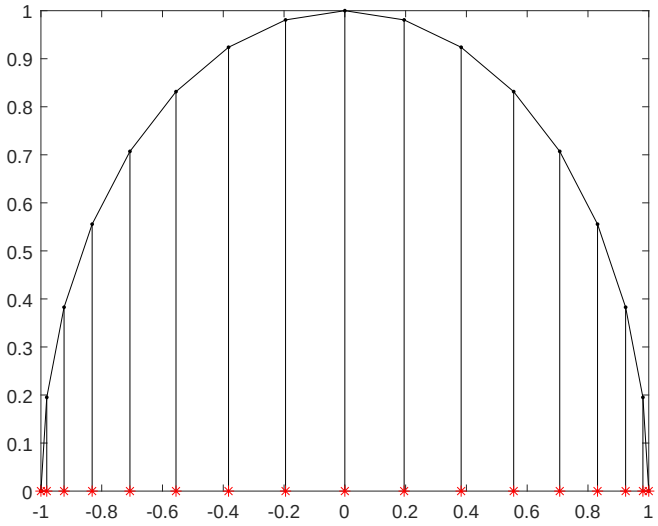

Figura 2.1: Puntos de Chebyshev para $n = 16$.

Demostración. Siguiendo la fórmula (2.1) los n puntos de Chebyshev de segunda especie son de la forma

$$\widehat{z}_j = cos\left(\frac{(j-1)\pi}{(n-1)}\right), \quad 1 \leq j \leq n. \tag{2.3}$$

Entonces, dado que

$$\frac{(j-1)\pi}{(n-1)} - \frac{\pi}{2} = \frac{2(j-1)\pi - (n-1)\pi}{2(n-1)} = \frac{(2j-n-1)\pi}{2(n-1)}, \tag{2.4}$$

aplicando la siguiente propiedad: $sin(a - (\pi/2)) = -cos(a)$ para $0 \leq a < 2\pi$ se cumple que

$$x_j = -\widehat{z}_j, \quad 1 \leq j \leq n.$$

□

Por ello de las dos maneras se pueden calcular los n puntos de Chebyshev de segunda especie, aunque con la fórmula (2.2) los argumentos están contenidos en el intervalo $[-\pi/2, \pi/2]$ y con la fórmula (2.3) en el intervalo $[0, \pi]$. Así que, teniendo en cuenta la aritmética computacional, es más conveniente utilizar la fórmula (2.2) ya que el intervalo de argumentos está centrado en el 0. Además como se ve en [5], MATLAB a la hora de calcular valores del seno y el coseno reduce el argumento al intervalo $[-\pi/4, \pi/4]$ y utiliza la relación $sin(\alpha) = -sin(-\alpha)$ para $\alpha \leq 0$. Veamos en el siguiente ejemplo cómo afecta todo lo mencionado a los resultados numéricos.

Ejemplo 2.1.1. *Calculamos los puntos de Chebyshev para $n = 5$ de dos formas, usando chebptscos.m un programa propio que calcula los puntos siguiendo la expresión* (2.3) *(ver*

apéndice) y usando chebpts.m (la función que usa Chebfun), obteniendo los siguientes resultados.

```
z=chebptscos(5)                    x=chebpts(5)
     1.000000000000000e+00              -1.000000000000000e+00
     7.071067811865476e-01              -7.071067811865475e-01
     6.123233995736766e-17                                   0
    -7.071067811865475e-01               7.071067811865475e-01
    -1.000000000000000e+00               1.000000000000000e+00
```

En el caso de chebpts obtenemos que $x_2 = -x_4$ y $x_3 = 0$, debido a lo explicado anteriormente, mientras que en el caso chebptscos $z_2 \neq z_4$ ya que son calculados evaluando $cos(\pi/4)$ y $-sin(\pi/4)$ respectivamente y z_3 aunque presenta un resultado computacionalmente aceptable no es el mismo que x_3.

Hasta ahora hemos visto los puntos de Chebyshev de segunda especie, que son los utilizados por Chebfun, pero también cabe destacar que existen los puntos de Chebyshev de primera especie que se definen a continuación (ver por ejemplo [6]).

Definición 2.1.2 (Puntos de Chebyshev de primera especie)**.** *Dado $n \in \mathbb{N}$, los n* ***puntos de Chebyshev de primera especie*** *son*

$$y_j = cos\left(\frac{(2j-1)\pi}{2n}\right), \qquad 1 \leq j \leq n.$$

Estos puntos son la proyección de los puntos medios, elegidos en la circunferencia unidad, de los puntos de Chebyshev de segunda especie, como se observa en la figura 2.2. Los puntos de Chebyshev de primera especie tienen importancia puesto que son los que minimizan una cota del error de interpolación como veremos en la sección siguiente.

Como podemos ver en [11] ambos tipos de puntos de Chebyshev son útiles en muchos ambitos del cálculo numérico como en la aproximación de funciones. Además, cabe destacar que en los libros clásicos de análisis numérico en los que se trata el tema de la aproximación de funciones únicamente se introducen los puntos de Chebyshev de primera especie, en cambio Trefethen en [10] trabaja principalmente con los puntos de Chebyshev de segunda especie y presta poca atención a los de primera especie, los cuales han sido introducidos por primera vez en la versión 5 de Chebfun.

En la figura 2.3 vemos la distribución de los puntos de Chebyshev de segunda especie en el intervalo $[-1, 1]$. Notemos que se acumulan en los extremos de dicho intervalo. Además tienen la siguiente propiedad, dados los $n+1$ puntos de Chebyshev de segunda especie, $\{x_j\}_{j=0}^{n}$ y considerando para cada uno de ellos la **distancia media geométrica**

$$d(x_j) = \sqrt[n-1]{\prod_{i=0\ i\neq j}^{n} |x_j - x_i|}$$

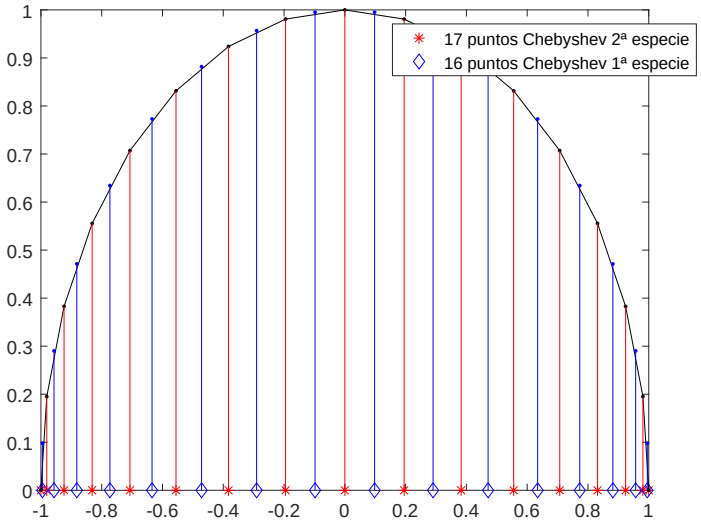

Figura 2.2: Puntos de Chebyshev de primera y segunda especie.

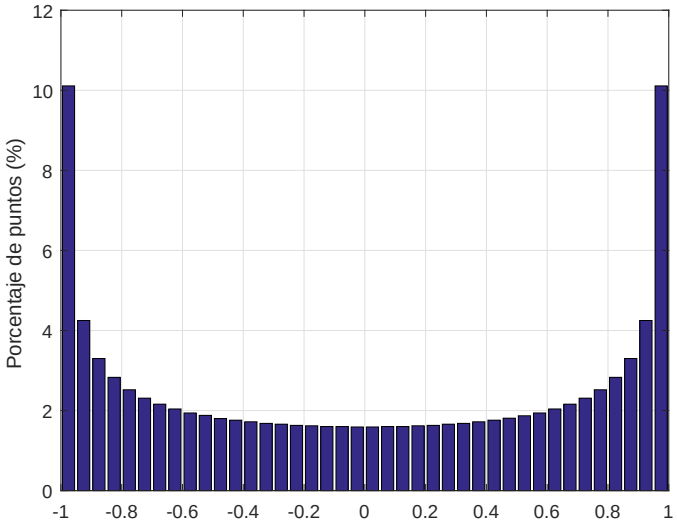

Figura 2.3: Histograma de la distribución de los 10000 puntos de Chebyshev de segunda especie.

entonces $d(x_1) = d(x_2) = \ldots = d(x_{n-1})$ y $d(x_0) = d(x_n)$. Además todas esas distancias se aproximan a $1/2$ al aumentar el número de nodos. Esta es una de las razones por la cual la interpolación en nodos de Chebyshev es preferible a la interpolación con puntos equiespaciados. En las figuras 2.4 y 2.5 se tiene una representación gráfica de la distribución de los nodos y de las distancias medias geométricas en ambos casos obtenidas con meandistance.m (ver apéndice). En la figura 2.6 se muestran los resultados correspondien-

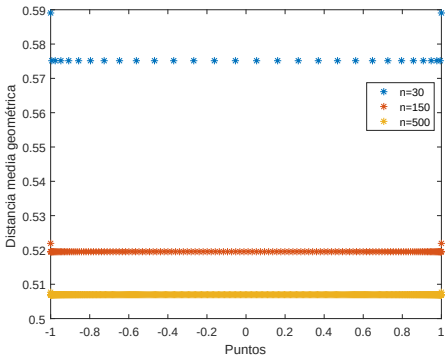

Figura 2.4: Distancia media geométrica de n puntos de Chebyshev de segunda especie.

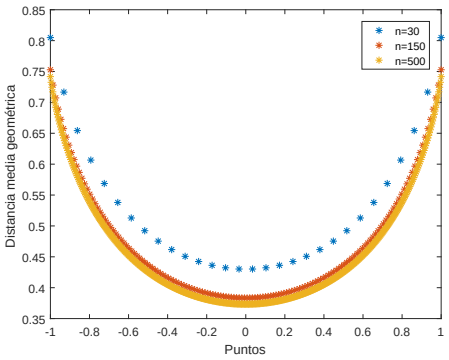

Figura 2.5: Distancia media geométrica de n puntos equiespaciados.

tes a los puntos de Chebyshev de primera especie, observándose que, al igual que en la figura 2.4, las distancias medias geométricas se aproximan a $1/2$ al aumentar el número de nodos, pero al contrario que para los de segunda especie la distancia media geométrica no es constante en los nodos y_k para $k = 1, \ldots, n-1$.

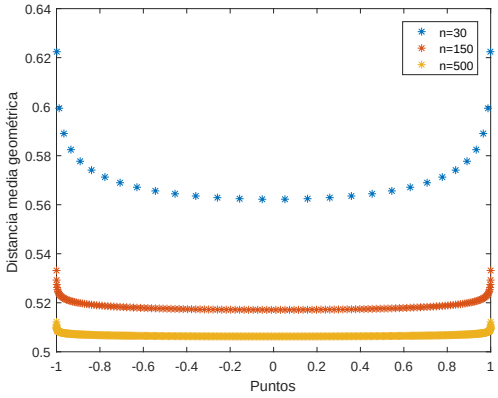

Figura 2.6: Distancia media geométrica de n puntos de Chebyshev de primera especie.

Como ya se vio en Cálculo Numérico I un ejemplo claro para apreciar que la interpolación en puntos de Chebyshev es más precisa que la interpolación en nodos equiespaciados es el

de la función de Runge. Por ello en el siguiente ejemplo vamos a hacer mención a dicha función y haremos una comparación de los errores con interpolación en nodos equiespaciados y en nodos de Chebyshev, tanto de primera como de segunda especie.

Ejemplo 2.1.2. *Dada la función* $f(x) = \dfrac{1}{1+25x^2}$ *en* $[-1,1]$*, denominada función de Runge, creamos por un lado los polinomios interpoladores para* $10, 12, 14, 16, 20$ *y* 30 *nodos de Chebyshev, tanto de primera como de segunda especie, utilizando en MATLAB la función* `chebfun`*, y por otro lado los polinomios interpoladores tomando puntos equiespaciados, usando el comando de MATLAB* `polyfit`*. Para el caso de* 12 *nodos, dibujamos la función* f *junto a los polinomios interpoladores en los tres casos y obtenemos la figura 2.7. Además, hemos calculado el error máximo en valor absoluto de la función de Runge respecto a cada polinomio interpolador en los diferentes puntos de interpolación, usando la función maxdifabs.m, en combinación con la funciones difigual.m, difcheb1.m y difcheb2.m (ver apéndice) y hemos expresado los datos en la tabla 2.1.*

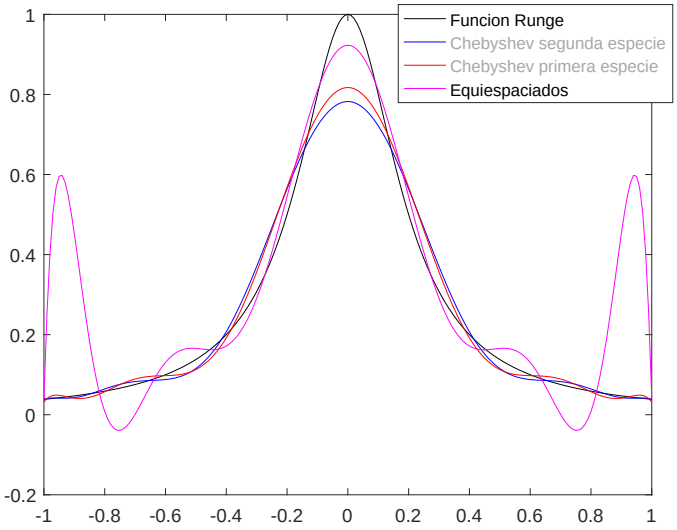

Figura 2.7: Polinomios interpoladores en 12 nodos junto a la función de Runge.

En la figura 2.7 observamos cómo en los extremos del intervalo es donde el polinomio interpolador en 12 nodos equiespaciados difiere más de la función, mientras que ocurre lo contrario con los polinomios interpoladores en los nodos de Chebyshev, que son en los extremos del intervalo donde mejor se ajustan a la función, esto se debe a la acumulación de los nodos de Chebyshev en los extremos del intervalo. Ahora, si nos fijamos en los datos que aparecen en la tabla 2.1, es reseñable que en todos los casos el polinomio que menor distancia tiene respecto a la función de Runge es el que interpola en nodos de Chebyshev de primera especie (esto va en sintonía con lo visto anteriormente para dichos puntos). Además el error con los distintos nodos de Chebyshev va disminuyendo a medida que aumenta el número de nodos de interpolación y, en cambio, para la interpolación en

n	Segunda especie	Primera especie	Equiespaciados
10	$3.1910e-01$	$2.6918e-01$	$3.0030e-01$
12	$2.1771e-01$	$1.8276e-01$	$5.5678e-01$
14	$1.4732e-01$	$1.2340e-01$	$1.0701e+00$
16	$9.9322e-02$	$8.3107e-02$	$2.1076e+00$
20	$4.4955e-02$	$3.7590e-02$	$8.5791e+00$
30	$6.1673e-03$	$5.1562e-03$	$3.3395e+02$

Tabla 2.1: Máximo error absoluto de los polinomios interpoladores en n nodos de Chebyshev de segunda especie, de Chebyshev de primera especie y equiespaciados.

puntos equiespaciados, la diferencia máxima respecto a la función de Runge va creciendo rápidamente de forma que para 30 puntos equiespaciados dicho error es mayor que 300, mientras que para 30 nodos de primera y segunda especie de Chebyshev, la distancia máxima del polinomio interpolador, en ese caso, es $5.1562e-03$ y $6.1673e-03$, respectivamente.

2.2. Polinomios de Chebyshev

Definición 2.2.1 (Polinomios de Chebyshev)**.** *Dado $n \in \mathbb{N}$, los **polinomios de Chebyshev** se definen recursivamente de la siguiente forma*

$$\begin{cases} T_0(x) = 1 \\ T_1(x) = x \\ T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \qquad (n \geq 1). \end{cases} \tag{2.5}$$

En el caso que acotemos x al intervalo $[-1, 1]$, los polinomios de Chebyshev se pueden expresar de la forma que sigue

$$T_n(x) = cos(ncos^{-1}(x)).$$

Los $n+1$ puntos de Chebyshev de segunda especie vistos en la definición 2.1.1 son los extremos (máximos o mínimos) del polinomio de Chebyshev $T_n(x)$ y los n puntos de primera especie considerados en la definición 2.1.2 son las raíces de dicho polinomio como vemos a continuación.

Proposición 2.2.1. *Dado $T_n(x)$ el polinomio de Chebyshev de grado $n \in \mathbb{N}$, se tiene que, $T_n(x_j) = (-1)^j$ para $j = 0, \ldots, n$ y $T_n(y_k) = 0$ para $k = 1, \ldots, n$ donde $\{y_k\}_{k=1}^n$ e $\{x_j\}_{j=0}^n$ son los puntos de Chebyshev de primera y segunda especie, respectivamente.*

Demostración. Sea $T_n(x) = cos(ncos^{-1}(x))$ el polinomio de Chebyshev de grado n. To-

mando $x_j = cos\left(\frac{j\pi}{n}\right), \quad 0 \leq j \leq n$, tenemos que

$$T_n(x_j) = cos\left(ncos^{-1}\left(cos\left(\frac{j\pi}{n}\right)\right)\right) = cos\,(j\pi) = (-1)^j.$$

Además evaluando el polinomio en los nodos $y_k = cos\left(\frac{(2k-1)\pi}{2n}\right), \quad 1 \leq k \leq n$, se obtiene

$$T_n(y_k) = cos\left(ncos^{-1}\left(cos\left(\frac{(2k-1)\pi}{2n}\right)\right)\right) = cos\left(\frac{(2k-1)\pi}{2}\right) = 0.$$

□

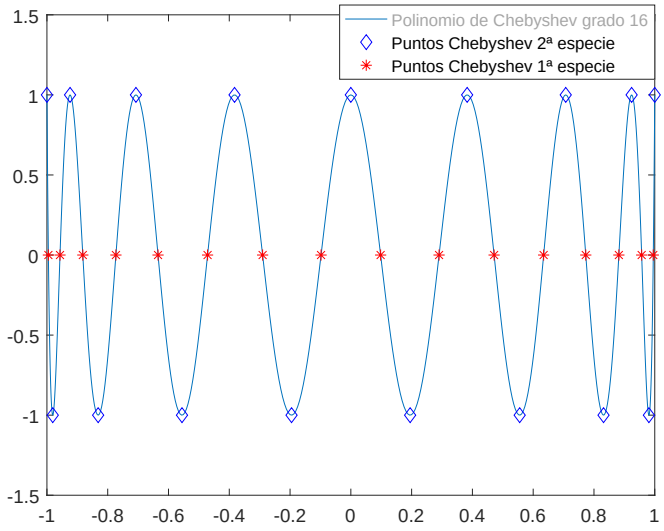

Figura 2.8: Puntos de Chebyshev asociados a $T_{16}(x)$.

Ahora veamos el papel de los puntos de Chebyshev de primera especie en relación al error de interpolación (ver por ejemplo [6]).

Teorema 2.2.1 (Error de interpolación)**.** *Dada f una función en $C^{n+1}[-1,1]$, y dado p el polinomio de grado $\leq n$ que interpola la función f en $n+1$ puntos distintos $x_0, x_1, \ldots, x_n$ en el intervalo $[-1,1]$. Para cada $x \in [-1,1]$ existe un punto $\xi_x \in (-1,1)$ tal que*

$$f(x) - p(x) = \frac{f^{n+1}(\xi_x)}{(n+1)!}\prod_{i=0}^{n}(x - x_i).$$

Teorema 2.2.2. *Los nodos de Chebyshev de primera especie minimizan la expresión*

$$\max_{|x|\leq 1} |\prod_{i=0}^{n}(x - x_i)|.$$

Demostración. 1. Por una parte vamos a probar que cualquier polinomio mónico p de grado n cumple que

$$\max_{|x|\leq 1} |p(x)| \geq 2^{1-n}.$$

Procediendo por reducción al absurdo suponemos que existe $p \in \mathcal{P}_n$ tal que $|p(x)| < 2^{1-n}$, $\forall x \in [-1,1]$. Entonces dado el polinomio $q = 2^{1-n}T_n \in \mathcal{P}_n$ mónico y tomando $x_i = cos(\frac{i\pi}{n})$ tenemos que

$$q(x_i) = 2^{1-n}T_n(x_i) = 2^{1-n}(-1)^i.$$

De esto deducimos que

$$(-1)^i q(x_i) = 2^{1-n} > |p(x_i)| \geq (-1)^i p(x_i),$$

que es equivalente a

$$(-1)^i(q(x_i) - p(x_i)) > 0, \quad 0 \leq i \leq n.$$

Por esto el polinomio $q - p \in \mathcal{P}_{n-1}$ oscila su signo $n+1$ veces en $[-1,1]$, luego $q-p$ tendría al menos n raíces en $[-1,1]$, con lo cual hemos llegado a una contradicción.

2. Por otro lado tenemos que

$$\max_{|x|\leq 1} 2^{1-n}|T_n(x)| \leq 2^{1-n}.$$

Esto se debe a que $|T_n(x)| = |cos(ncos^{-1}(x))| \leq 1$.

Ahora bien de los dos apartados anteriores obtenemos que $\|2^{1-n}T_n(x)\|_\infty = 2^{1-n}$. Además como las raíces de $2^{1-n}T_n(x)$ son, $y_i = cos\left(\dfrac{(2i-1)\pi}{2n}\right)$ donde $i = 1, \ldots, n$, entonces $2^{1-n}T_n(x) = \prod_{i=1}^n (x - y_i)$. □

Para $\mathcal{P}_n$ el espacio de los polinomios de grado $\leq n$, es usual utilizar la base formada por los monomios, pero en cálculo numérico es más conveniente la de los polinomios de Chebyshev. En el siguiente resultado se obtiene la relación entre los coeficientes de ambas representaciones.

Proposición 2.2.2. *Dado $p \in \mathcal{P}_n$, existen unos únicos coeficientes $a_0, a_1, ..., a_n$ tales que*

$$p = a_0 + a_1T_1 + ... + a_nT_n,$$

donde T_k es el k-ésimo polinomio de Chebyshev y existen unos únicos $b_0, b_1, ..., b_n$ tales que

$$p(x) = b_0 + b_1x + ... + b_nx^n.$$

Además existe una matriz triangular y regular C tal que $Ca = b$.

Demostración. Para cada polinomio de Chebyshev $T_k(x)$ se tienen unos coeficientes $c_{i,j}$ tales que

$$T_k(x) = c_{k,0} + c_{k,1}x + ... + c_{k,k}x^k,$$

por lo tanto se cumple la siguiente relación entre los coeficientes de ambas expresiones,

$$\begin{pmatrix} c_{0,0} & c_{1,0} & \cdots & c_{n,0} \\ 0 & c_{1,1} & \cdots & c_{n,1} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & c_{n,n} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{pmatrix}$$

Donde la matriz es regular debido a que es triangular y los elementos de la diagonal son distintos de cero, puesto que son los coeficientes asociados a x^k en el polinomio de Chebyshev T_k. □

En el siguiente ejemplo vamos a ver una razón por la cual Chebfun utiliza la base de los polinomios de Chebyshev, en vez de usar la base de los monomios.

Ejemplo 2.2.1. *Usando por un lado el comando* `chebpoly(n)` *del software Chebfun, que calcula el polinomio de Chebyshev de grado n y lo expresa como un elemento chebfun (ver sección 3.1), hemos calculado la quasimatriz*[1] *para la base de los polinomios de Chebyshev con dimensión* $n = 1, \ldots, 11$ *y por otro lado hemos calculado la quasimatriz para la base de los monomios de dimensión n tomando* `x=chebfun('x')` *y asignando a la columna* $j+1$ *de la quasimatriz el elemento chebfun* x^j *para* $j = 0, \ldots, n-1$*. Una vez realizado esto, hemos calculado el condicionamiento con el comando* `cond` *para ambas bases de polinomios y para cada una de las dimensiones n. Todo esto está ilustrado en el fichero de MATLAB basepolinomios.m (ver apéndice).*

Como se puede observar en la figura 2.9 el condicionamiento de la quasimatriz para la base de los monomios crece rápidamente cuando crece la dimensión, mientras que para los polinomios de Chebyshev crece muy lentamente.

2.3. Constante de Lebesgue en interpolación

De nuevo en esta sección observaremos un comportamiento más favorable de los nodos de Chebyshev frente a los equiespaciados. En el ámbito de la interpolación de funciones la constante de Lebesgue es utilizada a la hora de comparar el polinomio interpolador con la mejor aproximación polinomial de la función. Dicha constante no es más que la norma de un operador X que definimos a continuación. Sea un conjunto $S = \{x_0, x_1, \ldots, x_n\}$ de $n+1$ nodos distintos y $C([-1,1])$ el espacio de las funciones continuas en $[-1,1]$,

[1] Una quasimatriz es una matriz que consiste en n columnas donde cada una de ellas es un elemento chebfun, con lo cual una de sus dimensiones es discreta y la otra es continua.

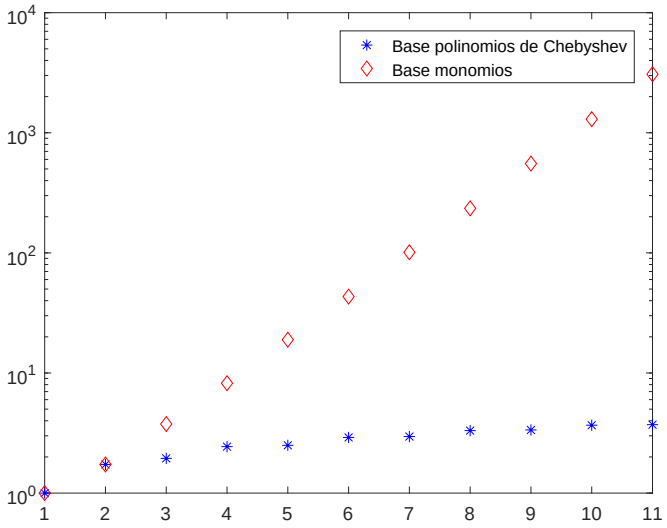

Figura 2.9: Relación entre el condicionamiento de la base de los polinomios de Chebyshev y la base de los monomios.

consideramos el operador que a cada función le asigna el polinomio interpolador en los nodos del conjunto S, esto es:

$$\begin{aligned} X : C([-1,1]) &\to \mathcal{P}_n \\ f &\mapsto X(f) = p \end{aligned}$$

donde $p(x_k) = f(x_k)$, $k = 0, 1, \ldots, n$. La aplicación X es lineal y es una proyección en el subespacio $\mathcal{P}_n$, es decir,

$$X(p) = p, \ \forall p \in \mathcal{P}_n.$$

Definición 2.3.1 (Constante de Lebesgue en interpolación)**.** *La norma del operador X se denomina* ***constante de Lebesgue en interpolación*** *y se denota como $\Lambda_n(S)$.*

En el siguiente teorema (ver por ejemplo [7]) se muestra la utilidad de la constante de Lebesgue en la comparación del polinomio interpolador con la mejor aproximación polinomial de la función.

Teorema 2.3.1. *Dado $f \in C([-1,1])$, sea d^* la distancia*

$$d^* = \min_{p \in \mathcal{P}_n} \|f - p\|_\infty.$$

Entonces $X(f)$ satisface que

$$\|f - X(f)\|_\infty \leq (1 + \Lambda_n(S))d^*.$$

Demostración. Dada p^* la mejor aproximación a f en $\mathcal{P}_n$: $\|f - p^*\|_\infty = d^*$. Por la linealidad de X y la propiedad de proyección tenemos la siguiente igualdad

$$f - X(f) = (f - p^*) - X(f - p^*).$$

Usando la desigualdad triangular, la propiedad $\|X(f)\|_\infty \leq \|X\|\|f\|_\infty$ y la definición de $\Lambda_n(S)$ y p^*, se obtiene la siguiente expresión

$$\begin{aligned} \|f - X(f)\|_\infty &\leq \|f - p^*\|_\infty + \|X(f - p^*)\|_\infty \\ &\leq [1 + \|X\|]\|f - p^*\|_\infty \\ &= (1 + \Lambda_n(S))d^* \end{aligned}$$

que completa la demostración. □

El teorema anterior nos lleva a estudiar el valor de $\Lambda_n(S)$. En el siguiente resultado (ver por ejemplo [7]) se usará la representación del polinomio interpolador con los polinomios de Lagrange,

$$p(x) = \sum_{k=0}^{n} f(x_k) l_k(x), \tag{2.6}$$

donde

$$l_k(x) = \prod_{j=0, j\neq k}^{n} (x - x_j)/(x_k - x_j), \quad 0 \leq k \leq n, \tag{2.7}$$

son los llamados **polinomios de Lagrange**.

Teorema 2.3.2. *La constante de Lebesgue en interpolación toma el valor*

$$\Lambda_n(S) = \max_{-1\leq x\leq 1} \sum_{k=0}^{n} |l_k(x)|.$$

Demostración. Por la definición de norma de un operador y dado que $X(f) = \sum_{k=0}^{n} f(x_k) l_k$ obtenemos que

$$\begin{aligned} \Lambda_n(S) = \|X\| &= \sup_{\|f\|_\infty \leq 1} \|X(f)\|_\infty \\ &= \sup_{\|f\|_\infty \leq 1} \max_{-1\leq x\leq 1} |\sum_{k=0}^{n} f(x_k) l_k(x)| \\ &= \max_{-1\leq x\leq 1} \sup_{\|f\|_\infty \leq 1} |\sum_{k=0}^{n} f(x_k) l_k(x)| \\ &= \max_{-1\leq x\leq 1} \sum_{k=0}^{n} |l_k(x)|. \end{aligned}$$

Para la última igualdad hemos usado que se verifica:

$$\sup_{\|f\|_\infty \leq 1} |\sum_{k=0}^{n} f(x_k) l_k(x)| = \sum_{k=0}^{n} |l_k(x)|.$$

Notemos que $\sum_{k=0}^{n} |l_k(x)|$ es una cota superior para $\sup_{\|f\|_\infty \leq 1} |\sum_{k=0}^{n} f(x_k) l_k(x)|$, entonces para que se de la igualdad debemos probar que dicha cota se alcanza. Para ello, para $x \in [-1, 1]$ tomamos una función $f \in C([-1, 1])$ de tal forma que

$$f(x_k) = \begin{cases} 1 & si \quad l_k(x) \geq 0 \\ \\ -1 & si \quad l_k(x) < 0 \end{cases}$$

entonces $f(x_k) l_k(x) = |l_k(x)| \quad \forall\, k \in 0, \ldots, n$, por tanto $|\sum_{k=0}^{n} f(x_k) l_k(x)| = \sum_{k=0}^{n} |l_k(x)|$, con lo cual se alcanza la cota superior. □

A continuación se establecen unas estimaciones para la constante de Lebesgue dependiendo del tipo de nodos elegidos en interpolación (ver [10]).

Teorema 2.3.3 (Cotas para la constante de Lebesgue)**.** *La constante de Lebesgue, $\Lambda_n(S)$, para la interpolación polinomial asociada a un conjunto S de $n+1$ nodos en el intervalo $[-1, 1]$, cumple que*

$$\Lambda_n(S) \geq \frac{2}{\pi}(log(n+1) + log(4/\pi) + \gamma),$$

donde $\gamma \approx 0.577$ es la constante de Euler.
En el caso de la interpolación en puntos de Chebyshev también se cumple que,

$$\Lambda_n(S) \leq \frac{2}{\pi}(log(n+1)) + 1$$

y

$$\Lambda_n(S) \sim \frac{2}{\pi}(log(n)), \quad n \to \infty.$$

Y para la interpolación con puntos equiespaciados se satisface

$$\Lambda_n(S) > \frac{2^{n-2}}{n^2}$$

y

$$\Lambda_n(S) \sim \frac{2^{n+1}}{e \cdot n \cdot log(n)}, \quad n \to \infty.$$

En la figura 2.10 se pueden observar las cotas de la constante de Lebesgue que aparecen en el teorema anterior tomando de 10 a 40 puntos de interpolación. Cabe destacar que la cota

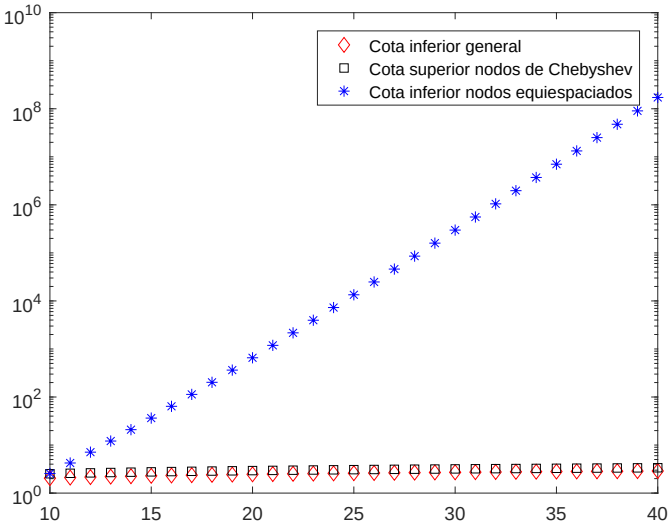

Figura 2.10: Cotas para la constante de Lebesgue de 10 a 40 nodos.

n	1	2	3	4	5	10	20	40
primera especie	1	1.4142	1.6667	1.8478	1.9889	2.4288	2.8698	3.3110
segunda especie	1	1	1.2500	1.6667	1.7988	2.3619	2.8371	3.2948

Tabla 2.2: Valor de la constante de Lebesgue para n puntos de Chebyshev de primera y segunda especie, respectivamente.

inferior de la constante de Lebesgue para puntos equiespaciados crece exponencialmente mientras que la cota superior para puntos de Chebyshev crece de manera mucho más lenta y se mantiene en una distancia inferior a 1/2 respecto a la cota inferior para cualquier conjunto de nodos.

En la tabla 2.2 se compara el valor de la constante de Lebesgue para un conjunto de n puntos de primera y segunda especie de Chebyshev (con la función constantelebesgue.m del apéndice). Vemos que para los puntos de primera especie la constante es mayor que para un conjunto, del mismo tamaño, de puntos de segunda especie , aunque cabe destacar que la diferencia no es muy grande y no podemos tomarlo como una razón para elegir entre uno de los dos conjuntos de puntos a la hora de usarlos para interpolar una función.

Capítulo 3

Chebfun

Chebfun (ver [1] y [3]) es un software de código abierto para el cálculo numérico con funciones, creado de los años 2002 al 2005 como parte de la investigación de la tesis doctoral de Zachary Battles que formaba parte del grupo de análisis numérico de la Universidad de Oxford, bajo la supervisión de Nick Trefethen. Posteriormente se han sucedido nuevas versiones de Chebfun en las que han colaborado numerosos científicos hasta llegar a la versión 5.7.0 publicada este mismo año. La versión que hemos utilizado para este trabajo es la 5.5.0 de julio de 2016. Chebfun tiene amplias capacidades para tratar con operadores lineales y no lineales diferenciables e integrables, muchas de las cuales se extienden de las funciones de MATLAB para vectores y matrices.

3.1. Elementos chebfun

Uno de los principales comandos del software Chebfun es `chebfun`, al cual se le pasa como argumento una función de la cual calcula el polinomio interpolador en puntos de Chebyshev de segunda especie (ver definición 2.1.1). El código `chebfun` por defecto realiza la interpolación en el intervalo $[-1, 1]$, aunque esto se puede cambiar añadiendo como argumento otro intervalo distinto después del argumento dado para la función.

Ejemplo 3.1.1. *Realizando la interpolación para la función $f = |x|^7$ obtenemos lo siguiente*

```
>> chebfun('abs(x)^7')
ans =

   chebfun column (1 smooth piece)
       interval       length     endpoint values
[      -1,       1]      197         1        1
vertical scale =   1
```

Como se observa en este ejemplo, MATLAB nos muestra el intervalo de interpolación (interval=$[-1, 1]$), el número de coeficientes del polinomio interpolador (length=197), el valor del polinomio interpolador en los extremos del intervalo (endpoint values = $[1, 1]$) y el valor absoluto máximo del polinomio interpolador (vertical scale=1). Chebfun guarda en un vector los coeficientes c_k correspondientes a expresar el polinomio interpolador en la forma:

$$\sum_{k=0}^{n} c_k T_k(x),$$

donde $T_k(x)$ es el polinomio de Chebyshev de grado k y $n+1$ el número total de puntos de interpolación. En el siguiente ejemplo vamos a presentar una función más complicada que nos ayude a entender mejor la información que nos muestra `chebfun`.

Ejemplo 3.1.2. *Elegimos la función $f(x) = sign(cos(5x)) + x^3$. Si realizamos la interpolación al igual que en el ejemplo 3.1.1, obtenemos lo siguiente*

```
>> f= @(x) sign(cos(5*x))+ x.^3;
>> p1=chebfun(f)
Warning: Function not resolved using 65537 pts.
Have you tried 'splitting on'?
p1 =

   chebfun column (1 smooth piece)
       interval       length     endpoint values
[      -1,       1]    65537  -3.5e-16        2
vertical scale =   2
```

En este caso observamos que Chebfun no ha podido encontrar un polinomio interpolador de grado menor a 65536 por lo que nos retorna un mensaje de error. Si realizamos lo que nos sugiere y usamos 'splitting on' obtenemos lo que aparece a continuación.

```
>> p2=chebfun(f,'splitting','on')

p2 =

   chebfun column (5 smooth pieces)
       interval       length     endpoint values
[      -1,   -0.94]        4  -2.1e-16     0.16
[   -0.94,   -0.31]        4      -1.8       -1
[   -0.31,    0.31]        4      0.97        1
[    0.31,    0.94]        4     -0.97    -0.16
[    0.94,       1]        4       1.8        2
vertical scale =   2    Total length = 20
```

Observamos que se ha realizado una interpolación polinomial a trozos (5 smooth pieces). Para cada trozo se nos muestra el intervalo donde está definido cada polinomio interpola-

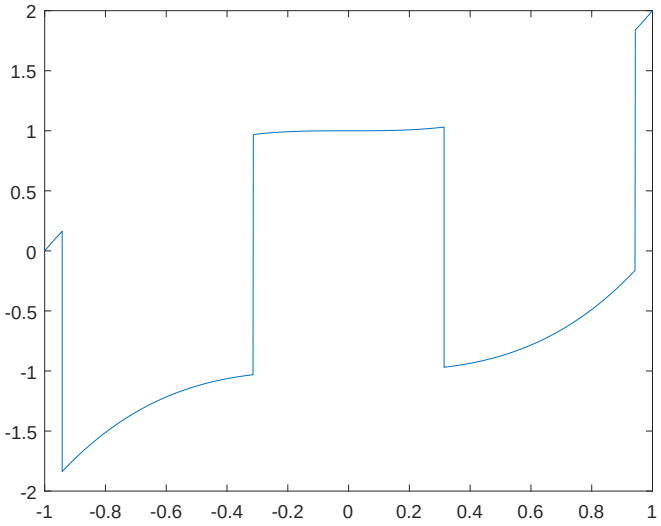

Figura 3.1: Función $sign(cos(5x)) + x^3$ en el intervalo $[-1, 1]$.

n	tiempo(seg.)
10	$9.3750e-01$
100	$1.7969e+00$
1000	$2.9531e+00$
10000	$2.2969e+00$

Tabla 3.1: Tiempo de cálculo del polinomio interpolador en n nodos de Chebyshev de segunda especie.

dor, así como el número de coeficientes que tiene y los valores que toma en los extremos del intervalo correspondiente. Si nos fijamos en la figura 3.1 vemos que Chebfun ha elegido como extremos de los intervalos los puntos donde la función f tiene singularidades y a partir de ahí ha generado los correspondientes polinomios interpoladores.

En el siguiente ejemplo vemos que `chebfun` también se puede utilizar para calcular el polinomio interpolador para un conjunto de datos y no únicamente es válido para funciones.

Ejemplo 3.1.3. *Hemos utilizado* `chebfun(2*rand(n,1)-1)` *para calcular el polinomio interpolador en n puntos de Chebyshev. En la siguiente tabla aparecen en la primera columna el número de datos aleatorios a interpolar, en la segunda columna el tiempo que usa MATLAB (medido con el comando* `cputime`*) en calcular el interpolante.*

Con este último ejemplo vemos cómo el tiempo que tarda Chebfun en calcular el polino-

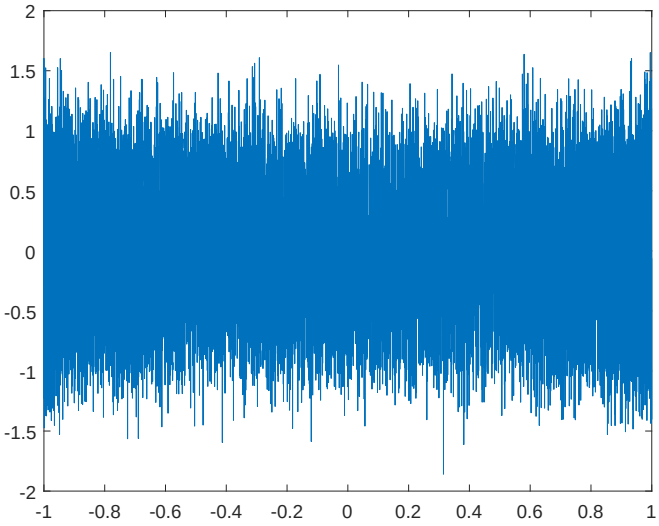

Figura 3.2: Polinomio interpolador con $n = 10000$.

mio interpolador para un conjunto grande de datos es muy pequeño: tarda menos de 2.3 segundos en calcular un polinomio de grado 9999, representado en la figura 3.2.

3.2. ¿Por qué elegir los puntos de Chebyshev de segunda especie?

Ya descartados los nodos equiespaciados, veamos en el siguiente resultado una razón por la cual el software Chebfun utiliza los puntos de Chebyshev de segunda especie en vez de los de primera especie.

Proposición 3.2.1. *1. Dado $n \in \mathbb{N}$ y siendo S_{n+1} el conjunto de los $n+1$ puntos de segunda especie de Chebyshev, se cumple que*

$$S_{n+1} \subseteq S_{2n+1}.$$

2. Dado $n \in \mathbb{N}$ y siendo $\tilde{S}_n$ el conjunto de los n puntos de primera especie de Chebyshev, se cumple que

$$\tilde{S}_n \subseteq \tilde{S}_{3n}.$$

Demostración. 1. Dado $n \in \mathbb{N}$, los $n+1$ puntos de segunda especie de Chebyshev son

$$x_j = cos\left(\frac{j\pi}{n}\right), \qquad 0 \leq j \leq n,$$

y los $2n+1$ puntos de segunda especie de Chebyshev son

$$\widehat{x}_k = cos\left(\frac{k\pi}{2n}\right), \qquad 0 \leq k \leq 2n.$$

Si tomamos los subíndices $k = 2j$ con $0 \leq j \leq n$ obtenemos que

$$\widehat{x}_{2j} = cos\left(\frac{2j\pi}{2n}\right) = cos\left(\frac{j\pi}{n}\right) = x_j.$$

2. Dado $n \in \mathbb{N}$, los n puntos de primera especie de Chebyshev son

$$y_j = cos\left(\frac{(2j-1)\pi}{2n}\right), \qquad 1 \leq j \leq n,$$

y los $3n$ puntos de primera especie de Chebyshev son

$$\widehat{y}_k = cos\left(\frac{(2k-1)\pi}{6n}\right), \qquad 1 \leq k \leq 3n.$$

Si tomamos los subíndices $k = 3j-1$ con $1 \leq j \leq n$ obtenemos que

$$\widehat{y}_{3j-1} = cos\left(\frac{3(2j-1)\pi}{6n}\right) = cos\left(\frac{(2j-1)\pi}{2n}\right) = y_j.$$

□

Dada una función, Chebfun sigue un procedimiento iterativo para determinar el número de nodos de Chebyshev que necesita para obtener un buen interpolante. A continuación describimos brevemente el procedimiento. Teniendo en cuenta la proposición 3.2.1, el software Chebfun utiliza el siguiente proceso de refinamiento: calcula en primer lugar los coeficientes del polinomio interpolante en $n = 2^4 + 1$ puntos de Chebyshev de segunda especie; si los últimos coeficientes caen al nivel del valor de la precisión de la máquina (*eps*), entonces el proceso termina y Chebfun determina el número de coeficientes útiles para calcular el correspondiente polinomio interpolador; en caso contrario, se aumenta el número de nodos y se realiza el mismo procedimiento para $n = 2^m + 1$ puntos de Chebyshev de segunda especie, con $m \in \mathbb{N}$ y $m = 5, 6, \ldots$. Si se llega a $m = 16$ el proceso termina sin éxito. (ver [2] y el ejemplo 3.1.2). Notemos que, el conjunto de nodos de Chebyshev de cada paso contiene al conjunto del paso anterior (primer apartado de la proposición 3.2.1) y por tanto esto conlleva a que se reduzcan los cálculos. En caso de que el software Chebfun tomase los puntos de Chebyshev de primera especie debería usar en cada paso el triple de puntos que en el paso anterior para poder reutilizar los cálculos realizados; esto significaría que en cada iteración el número de puntos crecería de forma más rápida con lo cual también crecería el tiempo de cálculo.

En el siguiente ejemplo vamos a mostrar el número de coeficientes del polinomio interpolador en $n+1$ puntos de Chebyshev de segunda especie con Chebfun para una función dada y veamos que coincide con lo que hemos expuesto anteriormente.

Ejemplo 3.2.1. *Dada la función* $f(x) = sin(6x) + sin(60e^x)$*, si ejecutamos en MATLAB* `chebfun(f)`*, el software Chebfun nos devuelve el polinomio interpolador de grado* 150 *con sus respectivos* 151 *coeficientes representados en la figura 3.5. Por otra parte, si interpolamos la función* f *en* $65, 129$ *y* 257 *nodos ejecutando* `chebfun(f,65)`*,* `chebfun(f,129)` *y* `chebfun(f,257)`*, respectivamente, se obtienen los polinomios interpoladores de grado* $64, 128$ *y* 256*, cuyos coeficientes están representados en las figuras 3.3, 3.4 y 3.6, respectivamente.*

Si nos fijamos en el valor absoluto de los últimos cinco coeficientes del polinomio inter-

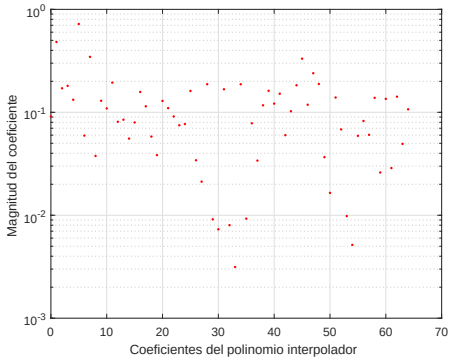

Figura 3.3: Magnitud de los coeficientes del polinomio interpolador en 65 nodos.

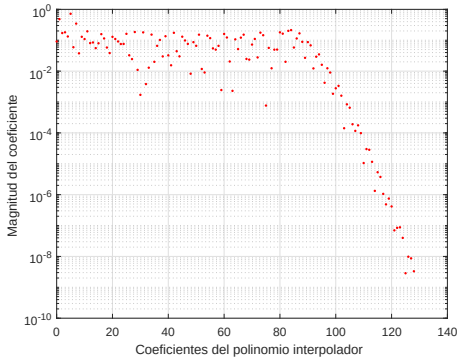

Figura 3.4: Magnitud de los coeficientes del polinomio interpolador en 129 nodos.

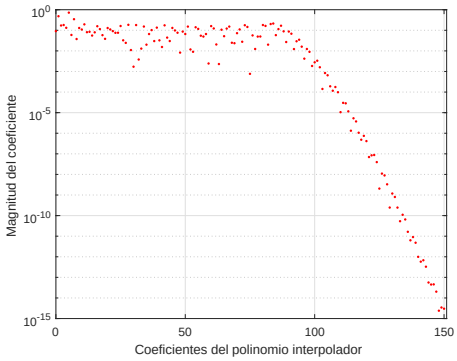

Figura 3.5: Magnitud de los coeficientes del polinomio interpolador en 151 nodos.

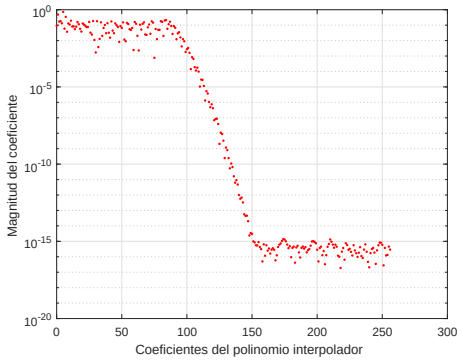

Figura 3.6: Magnitud de los coeficientes del polinomio interpolador en 257 nodos.

polador en 129 *puntos de Chebyshev de segunda especie* $(m = 7)$

$$3.9892e-08$$
$$2.8551e-09$$
$$9.8004e-09$$
$$8.6743e-09$$
$$3.3090e-09$$

vemos que están lejos del valor de la precisión de la máquina. Esto también lo podemos observar en la figura 3.4, para este caso, y en la figura 3.3 para los coeficientes del polinomio interpolador en 65 *nodos de Chebyshev. Entonces, si aumentamos el número de nodos a* 257 $(m = 8)$*, los últimos cinco coeficientes en valor absoluto del polinomio interpolador son*

$$3.7557e-16$$
$$1.2317e-16$$
$$1.3184e-16$$
$$4.5797e-16$$
$$2.9143e-16$$

y, por tanto, son cercanos al nivel de la precisión de la máquina. Además podemos ver en la figura 3.6 que existe un gran número de coeficientes que se mantienen cerca del nivel de la precisión de la máquina. Es por ello que, para la función dada Chebfun por defecto calcula un polinomio interpolador con 151 *coeficientes como podemos ver en la figura 3.5.*

3.3. Series y polinomios de Chebyshev

En esta sección estamos interesados en la generalización de la propiedad de que todo polinomio puede ser expresado de forma única como combinación lineal de los polinomios de Chebyshev. A continuación definimos el tipo de funciones para las que consideramos tal generalización.

Definición 3.3.1 (Función Lipschitz continua)**.** *Dada una función* $f : [-1, 1] \longrightarrow \mathbb{R}$*, se dice que es una* ***función Lipschitz continua*** *si existe una constante* $C \geq 0$ *tal que*

$$|f(x) - f(y)| \leq C|x - y|, \quad \forall\, x, y \in [-1, 1].$$

Notemos que, por ejemplo toda función de $C^1([-1, 1])$ es Lipschitz continua.

En el resultado siguiente (ver [10]), la generalización planteada vemos que conlleva la consideración de una serie involucrando a los polinomios de Chebyshev.

Teorema 3.3.1. *Si* $f : [-1, 1] \longrightarrow \mathbb{R}$ *es una función Lipschitz continua, entonces tiene una única representación como* ***serie de Chebyshev****:*

$$f(x) = \sum_{k=0}^{\infty} a_k T_k(x),$$

donde

$$a_k = \frac{2}{\pi} \int_{-1}^{1} \frac{f(x)T_k(x)}{\sqrt{1 - x^2}} dx, \quad \textit{para} \quad k \geq 1$$

y

$$a_0 = \frac{1}{\pi}\int_{-1}^{1} \frac{f(x)T_0(x)}{\sqrt{1-x^2}}dx.$$

Además la serie es absoluta y uniformemente convergente.

Ahora nuestro objetivo es relacionar los coeficientes a_k de la serie de Chebyshev, cuyo cálculo no es simple, con los coeficientes c_k del polinomio interpolador en $n+1$ puntos de Chebyshev de segunda especie

$$p_n(x) = \sum_{k=0}^{n} c_k T_k(x).$$

Como se observa en la figura 3.7 los polinomios de Chebyshev de grado $1, 5, 7, 11$ y 13 toman los mismos valores en los cuatro nodos de Chebyshev considerados. En relación con esto presentamos el siguiente teorema (ver por ejemplo [10]), que usaremos posteriormente y en el cual se establece la regla que siguen unos polinomios de Chebyshev determinados, por la cual toman los mismos valores en los puntos de Chebyshev de segunda especie.

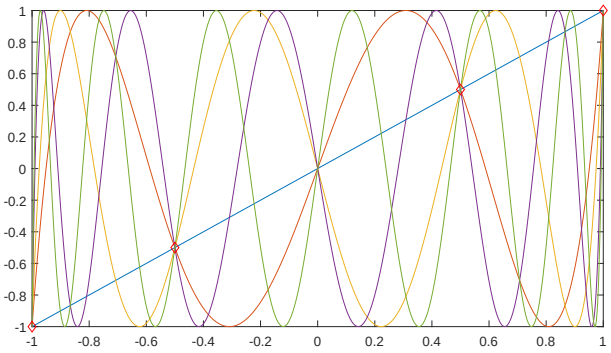

Figura 3.7: Polinomios $T_1, T_5, T_7, T_{11}, T_{13}$.

Teorema 3.3.2. *Para cualquier $n \geq 1$ y $0 \leq m \leq n$, los siguientes polinomios de Chebyshev toman los mismos valores en los $n+1$ puntos de Chebyshev,*

$$T_m, T_{2n-m}, T_{2n+m}, T_{4n-m}, T_{4n+m}, T_{6n-m}, \ldots$$

Ahora presentamos el teorema que relaciona los coeficientes de la serie de Chebyshev con los coeficientes del polinomio interpolador en $n+1$ nodos de Chebyshev de segunda especie (ver por ejemplo [10]).

Teorema 3.3.3. *Sean $f : [-1, 1] \longrightarrow \mathbb{R}$ una función Lipschitz continua y $p_n \in \mathcal{P}_n$ el polinomio interpolador en $n+1$ puntos de Chebyshev para $n \geq 1$. Si $\{a_k\}_{k=0}^{\infty}$ y $\{c_k\}_{k=0}^{n}$ son los coeficientes de Chebyshev de f y p_n respectivamente, entonces*

$$c_0 = a_0 + \sum_{s=1}^{\infty} a_{2sn},$$

$$c_k = a_k + \sum_{s=1}^{\infty} a_{k+2sn} + \sum_{s=1}^{\infty} a_{-k+2sn}, \quad para \;\; 1 \leq k \leq n-1$$

y

$$c_n = a_n + \sum_{s=1}^{\infty} a_{(2s+1)n}.$$

Demostración: usando el teorema 3.3.1 y teniendo en cuenta que,

$$f(1) = \sum_{k=0}^{\infty} a_k T_k(1) = \sum_{k=0}^{\infty} a_k,$$

la serie $\sum_{k=0}^{\infty} a_k$ es absolutamente convergente y por tanto convergen las series siguientes:

$$a_0 + \sum_{s=1}^{\infty} a_{2sn},$$

$$a_k + \sum_{s=1}^{\infty} a_{k+2sn} + \sum_{s=1}^{\infty} a_{-k+2sn}, \quad \text{para} \quad 1 \leq k \leq n-1$$

y

$$a_n + \sum_{s=1}^{\infty} a_{(2s+1)n}.$$

Denominamos a las sumas de las series anteriores como $\widehat{c}_0, \ldots, \widehat{c}_n$ y creamos el polinomio $\widehat{p}_n$ de la siguiente manera,

$$\widehat{p}_n(x) = \sum_{k=0}^{n} \widehat{c}_k T_k(x).$$

Evaluando este en los $n+1$ puntos de Chebyshev, que llamamos x_j, se tiene que,

$$\begin{aligned}
\widehat{p}_n(x_j) &= \sum_{k=0}^{n} \widehat{c}_k T_k(x_j) \\
&= a_0 T_0(x_j) + \sum_{s=1}^{\infty} a_{2sn} T_0(x_j) \\
&+ a_1 T_1(x_j) + \sum_{s=1}^{\infty} a_{1+2sn} T_1(x_j) + \sum_{s=1}^{\infty} a_{-1+2sn} T_1(x_j) + \dots \\
&+ a_{n-1} T_{n-1}(x_j) + \sum_{s=1}^{\infty} a_{-1+(2s+1)n} T_{n-1}(x_j) + \sum_{s=1}^{\infty} a_{1+(2s-1)n} T_{n-1}(x_j) \\
&+ a_n T_n(x_j) + \sum_{s=1}^{\infty} a_{(2s+1)n} T_n(x_j) \\
&= a_0 T_0(x_j) + \sum_{s=1}^{\infty} a_{2sn} T_{2sn}(x_j) \\
&+ a_1 T_1(x_j) + \sum_{s=1}^{\infty} a_{1+2sn} T_{1+2sn}(x_j) + \sum_{s=1}^{\infty} a_{-1+2sn} T_{-1+2sn}(x_j) + \dots \\
&+ a_{n-1} T_{n-1}(x_j) + \sum_{s=1}^{\infty} a_{-1+(2s+1)n} T_{-1+(2s+1)n}(x_j) + \sum_{s=1}^{\infty} a_{1+(2s-1)n} T_{1+(2s-1)n}(x_j) \\
&+ a_n T_n(x_j) + \sum_{s=1}^{\infty} a_{(2s+1)n} T_{(2s+1)n}(x_j) \\
&= \sum_{k=0}^{\infty} a_k T_k(x_j) = f(x_j), \ \textit{para } j = 0, \dots, n.
\end{aligned}$$

donde en la tercera igualdad se usa el teorema 3.3.2. Así hemos probado que $\widehat{p}_n$ es un polinomio de grado n interpolador de la función f en los $n+1$ puntos de Chebyshev de segunda especie. Como $p_n(x) = \sum_{k=0}^{n} c_k T_k(x)$ también verifica lo mismo y solo hay uno que lo cumple, $p_n = \widehat{p}_n$ y, por la unicidad de los coeficientes, se tiene que $c_i = \widehat{c}_i$ para $i = 0, \dots, n$. □

Así que el polinomio interpolador p_n no es la suma de los primeros $n+1$ términos de la serie de Chebyshev, aunque para funciones suficientemente regulares la diferencia es pequeña (ver capítulos 7, 8 y 16 de [10]). A continuación mostramos en el siguiente ejemplo la diferencia entre el polinomio interpolador de Chebyshev y el polinomio truncado, que consiste en elegir la suma de los citados $n+1$ primeros términos de la serie de Chebyshev.

Ejemplo 3.3.1. *Dada la función $f(x) = sin(3x) + exp(cos(x))$ que aparece en la función de MATLAB sinexp.m (ver apéndice), hemos calculado la mayor diferencia en valor*

absoluto respecto a f, tanto para el polinomio interpolador de grado n, como para el polinomio truncado de grado n (ver truncado.m, difpn.m y maxdiftrunc.m en el apéndice). Esto último está representado en la figura 3.8 para los valores $n = 1, \ldots, 20$. Como se puede observar el error es menor para el polinomio truncado en cada caso, pero la diferencia es pequeña y en ambos casos el error se acerca al valor de la precisión de la máquina al aumentar el número de nodos. Además en la figura 3.9 vemos el polinomio interpolador y el polinomio truncado de grado 3, en donde se observa que estos dos polinomios son distintos.

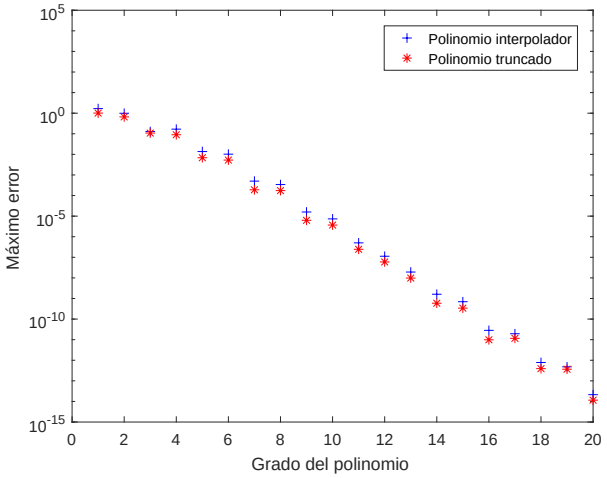

Figura 3.8: Máxima diferencia en valor absoluto para el polinomio interpolador y el polinomio truncado.

3.4. Transformada Rápida de Fourier

Chebfun utiliza interpolantes del tipo

$$p(x) = \sum_{k=0}^{n} c_k T_k(x) \tag{3.1}$$

para aproximar las funciones. En esta sección vamos a ver cómo calcula Chebfun los coeficientes del polinomio interpolador usando la transformada discreta de Fourier (DFT) que se define a continuación.

Definición 3.4.1. *Dados N valores reales f_i, donde $i = 0, \ldots, N-1$, la **transformada discreta de Fourier** (DFT) viene dada por*

$$Y_k = \sum_{j=0}^{N-1} f_j e^{\frac{-i2\pi kj}{N}}, \qquad 0 \leq k \leq N-1.$$

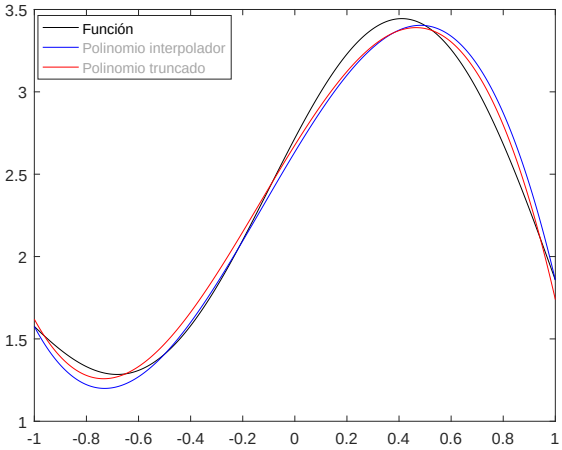

Figura 3.9: Función f junto a los polinomios interpolador y truncado de grado 3.

El algoritmo que se usa en la práctica para calcular la DFT se denomina **transformada rápida de Fourier** (FFT) que en MATLAB está implementada en la función `fft`. Usando la variable $\theta = cos^{-1}(x) \in [0, 2\pi]$ la expresión (3.1) toma la forma

$$p(\theta) = \sum_{k=0}^{n} c_k cos(k\theta),$$

que es un caso particular de la siguiente definición (ver [7]).

Definición 3.4.2. *(Polinomio trigonométrico) Dado $n \in \mathbb{N}$ se define como* ***polinomio trigonométrico*** *una función de la forma*

$$q(x) = \frac{1}{2}a_0 + \sum_{k=1}^{n}[a_k cos(kx) + b_k sin(kx)], \tag{3.2}$$

donde a_k y b_k para $k = 0, 1, \ldots, n$ son coeficientes reales.

El grado del polinomio trigonométrico (3.2) es el mayor entero k tal que al menos uno de los coeficientes a_k o b_k es distinto de cero.

El siguiente resultado (ver por ejemplo [8]) permite relacionar los coeficientes de este polinomio con la DFT.

Teorema 3.4.1. *Dados N valores reales $\{f_i\}_{i=0}^{N-1}$ y N nodos igualmente espaciados en $[0, 2\pi]$:*

$$\theta_j = j\frac{2\pi}{N}, \quad 0 \leq j \leq N-1,$$

el único polinomio trigonométrico de grado $M = N/2$ *de la forma*

$$p(\theta) = \tilde{a}_0 + 2\sum_{k=1}^{M-1}[\tilde{a}_k cos(k\theta) + \tilde{b}_k sin(k\theta)] + \tilde{a}_M cos(M\theta),$$

que verifica $p(\theta_j) = f_j$, *para* $j = 0, \ldots, N-1$, *viene determinado por*

$$\tilde{a}_k = \frac{1}{N}\sum_{j=0}^{N-1} f_j cos(\frac{2\pi kj}{N}), \quad 0 \leq k \leq M, \tag{3.3}$$

y

$$\tilde{b}_k = \frac{1}{N}\sum_{j=0}^{N-1} f_j sin(\frac{2\pi kj}{N}), \quad 1 \leq k \leq M-1.$$

Usando la fórmula de Moivre: $e^{kix} = cos(kx) + isin(kx)$, los coeficientes $\tilde{a}_k$ se pueden expresar en relación a la DFT de la siguiente forma.

Corolario 3.4.1. *Dados los valores* $Y_k = \sum_{j=0}^{N-1} f_j e^{\frac{-i2\pi kj}{N}}$ *de la transformada discreta de Fourier, para* $k = 0, \ldots, N-1$, *los coeficientes del polinomio interpolador trigonométrico descritos en la expresión* (3.3) *verifican*

$$\begin{cases} \tilde{a}_0 = \frac{1}{N} Y_0, \\ \\ \tilde{a}_k = \frac{1}{N} real(Y_k) \quad si \quad 1 \leq k \leq M-1, \\ \\ \tilde{a}_M = \frac{1}{N} Y_M. \end{cases} \tag{3.4}$$

Veamos la utilidad de lo expuesto anteriormente para el cálculo de los coeficientes del polinomio (3.1) interpolador en $n+1$ nodos de Chebyshev de segunda especie: $x_k = cos(\theta_k)$, donde $\theta_k = \frac{k\pi}{n}$ para $k = 0, 1, \ldots, n$. Dada una función f que toma valores $f_k = f(x_k)$ en los nodos citados, realizamos el cambio de variable $x = cos(\theta)$ para trasladarnos del intervalo $[-1, 1]$ al intervalo $[0, 2\pi]$ y, utilizando la función

$$F(\theta) = f(cos(\theta)),$$

nos lleva ahora a determinar los coeficientes del polinomio trigonométrico

$$p(\theta) = \sum_{k=0}^{n} c_k cos(k\theta),$$

tal que $p(\theta_k) = F(\theta_k)$, para $k = 0, \ldots, n$. Ahora bien, si queremos utilizar la transformada rápida de Fourier (FFT) para calcular los coeficientes del polinomio $p(\theta)$ necesitamos que el número de nodos sea el doble del grado del polinomio como vimos en el teorema 3.4.1,

para nuestro caso $2n$ puesto que el grado de $p(\theta)$ es n. Con ese objetivo, elegimos los siguientes $2n$ nodos igualmente espaciados en el intervalo $[0, 2\pi]$:

$$\widehat{\theta}_j = \frac{j\pi}{n}, \quad 0 \leq j \leq 2n-1.$$

Si ahora evaluamos la función F en los últimos $n-1$ nodos, tenemos que

$$F(\widehat{\theta}_{n+l}) = F(\widehat{\theta}_{n-l}) = f_{n-l}, \quad 1 \leq l \leq n-1, \tag{3.5}$$

debido a que $cos(\widehat{\theta}_{n+l}) = cos(\widehat{\theta}_{n-l})$. En consecuencia, tomando $f_{n+l} = f_{n-l}$ para $l = 1, \ldots, n-1$ y usando el teorema 3.4.1, el único polinomio trigonométrico de grado n de la forma

$$\widehat{p}(\theta) = \widehat{a}_0 + 2\sum_{k=1}^{n-1}[\widehat{a}_k cos(k\theta) + \widehat{b}_k sin(k\theta)] + \widehat{a}_n cos(n\theta)$$

que verifica $\widehat{p}(\widehat{\theta}_j) = f_j$ para $j = 0, \ldots, 2n-1$ viene determinado por

$$\widehat{a}_k = \frac{1}{2n}\sum_{j=0}^{2n-1} f_j cos(\frac{\pi kj}{n}), \quad 0 \leq k \leq n,$$

$$\widehat{b}_k = \frac{1}{2n}\sum_{j=0}^{2n-1} f_j sin(\frac{\pi kj}{n}), \quad 1 \leq k \leq n-1.$$

Veamos ahora que $\widehat{b}_k = 0$ para $k = 1, 2, \ldots, n-1$. En efecto, usando $f_{n+l} = f_{n-l}$, además de que $sin(\frac{\pi k}{n}(n+l)) = -sin(\frac{\pi k}{n}(n-l))$, se obtiene

$$\sum_{j=0}^{2n-1} f_j sin(\frac{\pi kj}{n}) = 0,$$

para $k = 1, 2, \ldots, n-1$. Así el polinomio resultante es

$$\widehat{p}(\theta) = \widehat{a}_0 + 2\sum_{k=1}^{n-1}\widehat{a}_k cos(k\theta) + \widehat{a}_n cos(n\theta),$$

que a su vez toma los valores f_k en los nodos $\theta_k = \widehat{\theta}_k$ para $k = 0, \ldots, n$. Entonces por la unicidad del polinomio interpolador, podemos expresar los coeficientes c_k de la siguiente manera

$$\begin{cases} c_0 = \widehat{a}_0, \\ c_k = 2\widehat{a}_k \quad \text{si} \quad 1 \leq k \leq n-1, \\ c_n = \widehat{a}_n, \end{cases}$$

de donde, si además usamos la expresión (3.4), que relaciona el valor de los coeficientes $\widehat{a}_k$ con la transformada discreta de Fourier, se concluye que

$$\begin{cases} c_0 = \frac{1}{2n} Y_0, \\ c_k = \frac{1}{n} real(Y_k) \quad \text{si} \quad 1 \leq k \leq n-1, \\ c_n = \frac{1}{2n} Y_n. \end{cases} \tag{3.6}$$

k	Chebfun	coeficientesfft.m
0	0	$-4.2701e-18$
1	$8.8010e-01$	$8.8010e-01$
2	0	$-2.8817e-17$
3	$-3.9127e-02$	$-3.9127e-02$
4	0	$1.4073e-18$
5	$4.9952e-04$	$4.9952e-04$
6	0	$5.0533e-18$
7	$-3.0047e-06$	$-3.0047e-06$
8	0	$-2.6655e-17$
9	$1.0499e-08$	$1.0499e-08$
10	0	$1.2810e-17$
11	$-2.3960e-11$	$-2.3960e-11$
12	0	0
13	$3.8538e-14$	$3.8520e-14$

Tabla 3.2: Coeficientes del polinomio interpolador de Chebyshev.

Esta relación entre los coeficientes del polinomio interpolador (3.1) y la transformada discreta de Fourier la hemos puesto en práctica en el siguiente ejemplo usando el comando `fft` de MATLAB para calcular los coeficientes de la expresión (3.1) siguiendo lo expuesto anteriormente (ver coeficientesfft.m en el apéndice).

Ejemplo 3.4.1. *Dada la función $f(x) = sin(x)$ si ejecutamos en MATLAB* `chebfun(f)` *obtenemos un polinomio interpolador de Chebyshev de grado $n = 13$ cuyos 14 coeficientes c_k aparecen en la segunda columna de la tabla 3.2. Además, por otra parte, si calculamos los 14 coeficientes siguiendo el procedimiento anterior con la expresión (3.6), obtenemos los coeficientes de la tercera columna de la tabla 3.2. Calculando la norma infinito de la diferencia de los vectores de ambos coeficientes nos da un resultado de $7.2162e-17$, lo cual es satisfactorio teniendo en cuenta la precisión de la máquina.*

3.5. Fórmula del baricentro

Esta sección se centra en elegir una fórmula adecuada para evaluar numéricamente el polinomio interpolador de Chebyshev (3.1). Dados $n+1$ nodos de interpolación, $\{x_j\}_{j=0}^n$ junto a los correspondientes datos $\{f_j\}_{j=0}^n$, nuestro objetivo es evaluar el polinomio $p \in \mathcal{P}_n$ tal que $p(x_j) = f_j$ para $j = 0, \ldots, n$. Habitualmente la primera fórmula de interpolación polinomial que se enseña para usar en la práctica a nivel de Grado es la de Newton, mientras que la fórmula de Lagrange únicamente se usa en la teoría, esto puede tener

sentido debido a que como vemos en [4], la **fórmula de Lagrange** viene dada por

$$p(x) = \sum_{j=0}^{n} f_j l_j(x), \tag{3.7}$$

donde

$$l_j(x) = \frac{\prod_{k=0,k\neq j}^{n}(x - x_k)}{\prod_{k=0,k\neq j}^{n}(x_j - x_k)}, \quad 0 \leq j \leq n, \tag{3.8}$$

y su coste es $O(n^2)$ operaciones para evaluar $p(x)$ en cada valor de x.

Por otra parte, si utilizamos la **fórmula de Newton**:

$$\begin{aligned} p(x) =& f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \dots \\ &+ f[x_0, x_1, \dots, x_n](x - x_0)(x - x_1)\dots(x - x_{n-1}). \end{aligned}$$

donde $f[x_j] = f_j$ y

$$f[x_j, x_{j+1}, \dots, x_{k-1}, x_k] = \frac{f[x_{j+1}, \dots, x_k] - f[x_j, \dots, x_{k-1}]}{x_k - x_j},$$

para lo cual hay que calcular la tabla de diferencias divididas de Newton

$$\begin{array}{lllllll} f[x_0] & & & & & & \\ & f[x_0, x_1] & & & & & \\ f[x_1] & & f[x_0, x_1, x_2] & & & & \\ & f[x_1, x_2] & & & f[x_0, x_1, x_2, x_3] & & \\ f[x_2] & & f[x_1, x_2, x_3] & & & \ddots & \\ & f[x_2, x_3] & & & \vdots & & f[x_0, x_1, \dots, x_n] \\ f[x_3] & & \vdots & & f[x_{n-3}, \dots, x_n] & \iddots & \\ & \vdots & f[x_{n-2}, x_{n-1}, x_n] & & & & \\ \vdots & f[x_{n-1}, x_n] & & & & & \\ f[x_n] & & & & & & \end{array}$$

Como vemos son necesarias tres operaciones por cada elemento de la tabla, a partir de la segunda columna, resultando un total de $O(n^2)$ operaciones, pero este cálculo únicamente hay que realizarlo una vez puesto que no depende del valor de x. Finalmente, para terminar de evaluar la fórmula de Newton precisamos de $O(n)$ operaciones. Como vemos, a priori, es más barato, computacionalmente hablando, la fórmula de Newton que la de Lagrange.

Otra opción para evaluar el polinomio interpolador (3.1) es calcular sus coeficientes con la transformada rápida de Fourier (FFT) (ver sección 3.4) y posteriormente realizar la suma

$$p(x) = \sum_{k=0}^{n} c_k T_k(x),$$

n	n^2	$n * log(n)$
10	10^2	$2.3026e+01$
10^2	10^4	$4.6052e+02$
10^3	10^6	$6.9078e+03$
10^4	10^8	$9.2103e+04$
10^5	10^{10}	$1.1513e+06$
10^6	10^{12}	$1.3816e+07$

Tabla 3.3: Valores del coste computacional en base al número de datos n.

lo que conlleva realizar $O(nlog(n))$ operaciones para la evaluación en un único punto.

En la tabla 3.3 ilustramos lo que suponen esos costes para distintos valores de n. A continuación veremos cómo usando la fórmula del baricentro se puede reducir el coste del cálculo de $p(x)$ con la interpolación de Lagrange a $O(n)$. Para ello comenzamos introduciendo unos resultados que posteriormente utilizaremos en la demostración de la fórmula de interpolación del baricentro para un conjunto de nodos cualquiera (ver [10]).

Definición 3.5.1 (Polinomio nodal)**.** *Dados $n+1$ puntos de interpolación $\{x_j\}_{j=0}^n$, definimos el **polinomio nodal** $l \in \mathcal{P}_{n+1}$ como*

$$l(x) = \prod_{k=0}^{n}(x - x_k). \tag{3.9}$$

Lema 3.5.1. *La derivada del polinomio nodal asociado a $n+1$ puntos de interpolación, $\{x_j\}_{j=0}^n$ cumple:*

$$l'(x) = \sum_{i=0}^{n}\left(\prod_{k=0\ k\neq i}^{n}(x - x_k)\right).$$

Demostración. Para realizar esta demostración vamos a usar inducción en n.

- Para $n = 2$ el polinomio nodal $l(x)$ es

  $$l(x) = (x - x_0)(x - x_1)(x - x_2),$$

  con lo cual su derivada es

  $$\begin{aligned} l'(x) &= (x - x_0)(x - x_1) + (x - x_2)[(x - x_0)(x - x_1)]' \\ &= (x - x_0)(x - x_1) + (x - x_0)(x - x_2) + (x - x_1)(x - x_2) \\ &= \sum_{i=0}^{2}\left(\prod_{k=0\ k\neq i}^{2}(x - x_k)\right). \end{aligned}$$

- Supuesto cierto para $n-1$ probémoslo para n.
  Tomamos el polinomio nodal $l(x)$ visto en la fórmula (3.9) y derivamos de la forma

que sigue

$$\begin{aligned}
l'(x) &= \left(\prod_{k=0}^{n}(x-x_k)\right)' = \left((x-x_n)\prod_{k=0}^{n-1}(x-x_k)\right)' \\
&= \prod_{k=0}^{n-1}(x-x_k) + (x-x_n)\left(\prod_{k=0}^{n-1}(x-x_k)\right)' \\
&= \prod_{k=0}^{n-1}(x-x_k) + (x-x_n)\left(\sum_{i=0}^{n-1}\left(\prod_{k=0\ k\neq i}^{n-1}(x-x_k)\right)\right) \\
&= \prod_{k=0}^{n-1}(x-x_k) + \left(\sum_{i=0}^{n-1}\left(\prod_{k=0\ k\neq i}^{n}(x-x_k)\right)\right) \\
&= \sum_{i=0}^{n}\left(\prod_{k=0\ k\neq i}^{n}(x-x_k)\right).
\end{aligned}$$

□

Lema 3.5.2. *Dado un conjunto de $n+1$ puntos $\{x_j\}_{j=0}^n$, los polinomios de Lagrange asociados a ellos verifican $\sum_{j=0}^n l_j(x) \equiv 1$.*

Demostración. $l_j(x)$ es el único polinomio en $\mathcal{P}_n$ que verifica

$$l_j(x_k) = \begin{cases} 1 & si \quad k = j, \\ 0 & si \quad k \neq j. \end{cases}$$

Por tanto se tiene que el polinomio $q(x) = \sum_{j=0}^n l_j(x) \in \mathcal{P}_{n+1}$ y cumple que $q(x_k) = 1$ para $k = 0, \ldots, n$ y por la unicidad del polinomio interpolador $q \equiv 1$. □

Teorema 3.5.1 (Fórmula de interpolación del baricentro)**.** *El polinomio interpolante de un conjunto de datos $\{f_j\}_{j=0}^n$ en $n+1$ puntos $\{x_j\}_{j=0}^n$ puede expresarse de la siguiente forma*

$$p(x) = \frac{\displaystyle\sum_{j=0}^{n}\frac{\lambda_j f_j}{(x-x_j)}}{\displaystyle\sum_{j=0}^{n}\frac{\lambda_j}{(x-x_j)}}, \quad si \quad x \neq x_j, \tag{3.10}$$

donde

$$\lambda_j = \frac{1}{\prod_{k\neq j}(x_j - x_k)}.$$

Demostración. La demostración consistirá en reescribir la fórmula (3.7). Si observamos el numerador de los polinomios de Lagrange, $l_j(x)$, definidos en la expresión (3.8), vemos que

si lo multiplicamos por $(x - x_j)$ obtenemos el polinomio nodal, $l(x)$, visto en la expresión (3.9). Además el denominador de $l_j(x)$ lo podemos expresar como $l'(x_j)$. Así obtenemos una formulación alternativa para los polinomios de Lagrange:

$$l_j(x) = \frac{l(x)}{l'(x_j)(x - x_j)} = l(x)\frac{\lambda_j}{x - x_j}, \tag{3.11}$$

donde $\lambda_j = 1/l'(x_j)$.

Ahora bien, usando la expresión (3.11) para los polinomios de Lagrange y el lema anterior, tenemos que

$$l(x)\sum_{j=0}^{n}\frac{\lambda_j}{(x - x_j)} = 1.$$

A continuación dividiendo la fórmula (3.11) entre la expresión anterior podemos eliminar $l(x)$ y la expresión no varía puesto que dividimos entre la unidad, por ello finalmente tenemos que los polinomios de Lagrange son de la siguiente forma,

$$l_j(x) = \frac{\dfrac{\lambda_j}{x - x_j}}{\displaystyle\sum_{k=0}^{n}\frac{\lambda_k}{(x - x_k)}}. \tag{3.12}$$

Y ahora sí, usando la expresión de los polinomios de Lagrange (3.12) en la fórmula de la interpolación de Lagrange (3.7) obtenemos el resultado esperado

$$p(x) = \frac{\displaystyle\sum_{j=0}^{n}\frac{\lambda_j f_j}{(x - x_j)}}{\displaystyle\sum_{j=0}^{n}\frac{\lambda_j}{(x - x_j)}}.$$

□

En este punto cabe destacar que calcular todos los coeficientes λ_j requiere $O(n^2)$ operaciones, pero al no depender de los datos f_j únicamente es necesario calcularlos una vez. Esto es una ventaja respecto a la fórmula de Newton puesto que esta última requiere el cálculo de la tabla de diferencias divididas para cada conjunto de datos f_j. Si nos centramos en los puntos de Chebyshev de segunda especie veremos que la expresión (3.10) se reduce de forma que únicamente son necesarias $O(n)$ operaciones para evaluar el polinomio interpolador. En primer lugar probaremos dos resultados que utilizaremos posteriormente.

Lema 3.5.3. *Dados $\{x_j\}_{j=0}^{n}$ para $n \geq 1$, los $n+1$ puntos de Chebyshev de segunda especie en $[-1, 1]$, el polinomio nodal $l(x) \in \mathcal{P}_{n+1}$ asociado a ellos cumple que*

$$l(x) = 2^{-n}(T_{n+1}(x) - T_{n-1}(x)).$$

Demostración. Usando el teorema 3.3.2, si tomamos $m = n-1$, entonces T_{n-1} y $T_{2n-(n-1)} = T_{n+1}$ toman los mismos valores en los $n+1$ puntos de Chebyshev de segunda especie, por ello

$$2^{-n}(T_{n+1}(x_j) - T_{n-1}(x_j)) = 0. \tag{3.13}$$

Ahora veamos que el polinomio $2^{-n}(T_{n+1}(x) - T_{n-1}(x))$ es mónico, para ello si usamos la fómula recursiva para los polinomios de Chebyshev (2.5) tenemos que

$$2^{-n}(T_{n+1}(x) - T_{n-1}(x)) = 2^{-n}((2^n x^{n+1} + \ldots) - T_{n-1}(x)) = x^{(n+1)} + \ldots.$$

Luego el polinomio $2^{-n}(T_{n+1}(x) - T_{n-1}(x))$ es de grado $n+1$, mónico y, usando (3.13), tiene por raíces los $n+1$ nodos de Chebyshev de segunda especie. Esas tres propiedades sirven para determinar unívocamente a un polinomio. Luego como el polinomio nodal también las verifica, tenemos la igualdad del enunciado. □

Lema 3.5.4. *Dados $\{x_j\}_{j=0}^n$ los $n+1$ puntos de Chebyshev se tiene que*

1. *Para $1 \leq j \leq n-1$, $T'_{n+1}(x_j) - T'_{n-1}(x_j) = 2n(-1)^j$.*
2. *Para $j = 0$ y $j = n$, $T'_{n+1}(x_j) - T'_{n-1}(x_j) = 4n(-1)^j$.*

Demostración. Como ya hemos visto el polinomio de Chebyshev de grado $n+1$ puede expresarse de la siguiente forma $cos((n+1)cos^{-1}(x))$. Ahora para probar cada uno de los casos del enunciado usaremos estrategias diferentes.

1. Si tomamos $\theta(x) = cos^{-1}(x)$, tenemos que $T_{n+1}(x) = cos((n+1)\theta(x))$ y $T_{n-1}(x) = cos((n-1)\theta(x))$. A continuación calculamos la derivada para cada uno de los polinomios

$$T'_{n+1}(x) = T'_{n+1}(\theta(x))\theta'(x) = \frac{(n+1)sin((n+1)\theta(x))}{sin(\theta(x))},$$

$$T'_{n-1}(x) = \frac{(n-1)sin((n-1)\theta(x))}{sin(\theta(x))}.$$

Si restamos las expresiones $T'_{n+1}(x)$ y $T'_{n-1}(x)$ y denominamos la resta como $A(x)$ tenemos que

$$A(x) = \frac{1}{sin(\theta(x))}\left[(n+1)sin((n+1)\theta(x)) - (n-1)sin((n-1)\theta(x))\right].$$

Veamos el valor de $A(x)$ en los nodos de Chebyshev de segunda especie x_j para $j = 1, \ldots, n-1$. Como $x_j = cos((j\pi)/n)$, $\theta(x_j) = cos^{-1}(x_j) = (j\pi)/n$, y por tanto

$$\begin{aligned} A(x_j) &= \frac{1}{sin(\frac{j\pi}{n})}\left[(n+1)sin\left((n+1)\frac{j\pi}{n}\right) - (n-1)sin\left((n-1)\frac{j\pi}{n}\right)\right] \\ &= \frac{1}{sin(\frac{j\pi}{n})}\left[(n+1)sin\left(j\pi + \frac{j\pi}{n}\right) - (n-1)sin\left(j\pi - \frac{j\pi}{n}\right)\right]. \end{aligned}$$

Ahora denominando $\alpha_j = j\pi + \frac{j\pi}{n}$ y $\beta_j = j\pi - \frac{j\pi}{n}$ tenemos que:

- Si j es par, $sin(\alpha_j) = sin(j\pi/n)$, $sin(\beta_j) = -sin(j\pi/n)$ y $A(x_j) = 2n$.
- Si j es impar, $sin(\alpha_j) = -sin(j\pi/n)$, $sin(\beta_j) = sin(j\pi/n)$ y $A(x_j) = -2n$.

2. Para los nodos de Chebyshev $x_0 = 1$ y $x_n = -1$ vamos a usar la expresión de la definición de derivada. Por ello tenemos que

$$\begin{aligned} T'_{n+1}(1) &= \lim_{h\to 0^-} \frac{T_{n+1}(1+h) - T_{n+1}(1)}{h} \\ &= \lim_{h\to 0^-} \frac{cos((n+1)cos^{-1}(1+h)) - cos((n+1)cos^{-1}(1))}{h}. \end{aligned}$$

Como el numerador y el denominador tienden a cero, aplicamos la regla de l'Hôpital obteniendo

$$T'_{n+1}(1) = \lim_{h\to 0^-} \frac{(n+1)sin((n+1)cos^{-1}(1+h))}{\sqrt{1-(1+h)^2}},$$

nuevamente aplicamos la regla de l'Hôpital puesto que tenemos un cociente en el que ambos términos tienden a cero y resulta que

$$\begin{aligned} T'_{n+1}(1) &= \lim_{h\to 0^-} \frac{(n+1)^2 cos((n+1)cos^{-1}(1+h)) \frac{1}{\sqrt{1-(1+h)^2}}}{\frac{2(1+h)}{2\sqrt{1-(1+h)^2}}} \\ &= \lim_{h\to 0^-} \frac{(n+1)^2 cos((n+1)cos^{-1}(1+h))}{1+h} = (n+1)^2. \end{aligned} \tag{3.14}$$

$$T'_{n+1}(-1) = \lim_{h\to 0^+} \frac{(n+1)^2 cos((n+1)cos^{-1}(-1+h))}{-1+h} = (n+1)^2(-1)^n. \tag{3.15}$$

Para calcular $T'_{n-1}(1)$ y $T'_{n-1}(-1)$ podemos seguir el mismo procedimiento que en el caso anterior con los siguientes resultados

$$T'_{n-1}(1) = (n-1)^2 \tag{3.16}$$

y

$$T'_{n-1}(-1) = (n-1)^2(-1)^n. \tag{3.17}$$

Finalmente restando de las expresiones (3.14) y (3.16) tenemos que

$$T'_{n+1}(1) - T'_{n-1}(1) = (n+1)^2 - (n-1)^2 = 4n,$$

y, equivalentemente, restando (3.15) y (3.17) obtenemos que

$$T'_{n+1}(-1) - T'_{n-1}(-1) = (n+1)^2(-1)^n - (n-1)^2(-1)^n = 4n(-1)^n.$$

□

En el siguiente teorema vamos a presentar el caso particular de la fórmula de interpolación del baricentro (3.10) para los $n+1$ puntos de Chebyshev de segunda especie (ver por ejemplo [10]).

Teorema 3.5.2 (Fórmula del baricentro en puntos de Chebyshev)**.** *El polinomio interpolante de un conjunto de datos* $\{f_j\}_{j=0}^n$ *en* $n+1$ *puntos de Chebyshev* $\{x_j\}_{j=0}^n$ *es* [1]

$$p(x) = \frac{\sum_{j=0}^{n}{}' \frac{(-1)^j f_j}{(x-x_j)}}{\sum_{j=0}^{n}{}' \frac{(-1)^j}{(x-x_j)}}, \quad si\ x \neq x_j. \tag{3.18}$$

Demostración. La ecuación (3.18) es un caso particular de la ecuación (3.10). Para probarlo basta verificar que para los $n+1$ puntos de Chebyshev de segunda especie se verifica

$$\lambda_j = \begin{cases} \frac{2^{n-1}}{n}(-1)^j & \text{si} \quad 1 \leq j \leq n-1, \\ \\ \frac{2^{n-2}}{n}(-1)^j & \text{si} \quad j=0 \quad \text{ó} \quad j=n. \end{cases} \tag{3.19}$$

En efecto, en primer lugar recordemos que $\lambda_j = 1/l'(x_j)$ y usando el lema 3.5.3 se tiene que

$$\lambda_j = \frac{2^n}{T'_{n+1}(x_j) - T'_{n-1}(x_j)}.$$

Si a esta última igualdad le añadimos lo probado en el lema 3.5.4 obtenemos lo expuesto en la expresión (3.19). □

Para finalizar esta sección presentamos un ejemplo en el cual comparamos el algoritmo numérico de evaluación del polinomio interpolador implementado en MATLAB con la función `polyval` con la fórmula del baricentro usada en Chebfun.

Ejemplo 3.5.1. *Consideramos la función* $f(x) = cos(\alpha x)$ *dependiente de un parámetro que toma los valores* $\alpha = 10, 20, \ldots, 90, 100$*. Para cada valor de* α *evalúamos el polinomio interpolador en* $x = 0$ *de dos formas (ver evaluacionpolinomio.m en el apéndice), con Chebfun y con* `polyval`*. Primero calculamos el polinomio interpolador* (3.1) *quedando el grado determinado por Chebfun y a continuación para ese grado evaluamos el correspondiente polinomio con los dos métodos. En la tabla 3.4, para cada valor de* α *presentamos el grado del polinomio en la segunda columna y los valores de las dos evaluaciones en las dos siguientes columnas. Podemos observar cómo el polinomio interpolador con Chebfun,* p*, toma valor 1 en* $x = 0$ *para cualquier valor de* α*, mientras que usando,* pf*, vemos cómo a partir de* $\alpha = 60$ *el valor del polinomio para* $x = 0$ *comienza a diferir de* 1*, que es el valor correcto de la función* f *en el cero. El mal funcionamiento de* `polyval` *es debido al mal condicionamiento de la matrices de Vandermonde usadas durante el proceso (ver última columna de la tabla).*

[1] Los primas en el sumatorio significan que los términos $j = 0$ y $j = n$ hay que multiplicarlos por $1/2$.

α	n	$p(0)$	$pf(0)$	$cond(pf)$
10	34	1	1	$3.9654e+12$
20	50	1	1	$5.5580e+17$
30	64	1	1	$1.7528e+19$
40	76	1	1	$2.1249e+18$
50	88	1	1	$5.8994e+18$
60	102	1	$1.0038e+00$	$5.2122e+18$
70	112	1	$1.2567e+00$	$3.7374e+19$
80	124	1	$2.6304e+01$	$1.0905e+19$
90	136	1	$1.9604e+01$	$1.4178e+20$
100	148	1	$-3.4252e+02$	$1.1033e+19$

Tabla 3.4: Comparando Chebfun y polyval de MATLAB.

Apéndice

En esta sección recopilamos las funciones MATLAB que hemos implementado para realizar los experimentos numéricos. Al tratarse de una trascripción literal del código MATLAB, las palabras aparecen sin tildes.

- basepolinomios.m

```
clear all
format short e
x=chebfun('x');
for n=0:10,
    T2=chebpoly(0:n);%Polinomios de Chebyshev
    y(n+1)=cond(T2);
    M2=T2;%Matriz de los monomios
    for j=0:n,
        M2(:,j+1)=x.^j;
    end
    z(n+1)=cond(M2);
end
semilogy([1:11],y,'*b')
hold on
semilogy([1:11],z,'dr')
legend('Base polinomios de Chebyshev','Base monomios')
```

- chebptscos.m

```
%Calcula nodos de Chebyshev de segunda especie con la formula del coseno
function [x] = chebptscos(n)
    for j=1:n
        x(j)=cos((j-1)/(n-1)*pi);
    end
end
```

- coeficientesfft.m

```
format short e;
f=@(x) sin(x);
```

```
f2=chebfun(f);%chebfun de la funcion f
coefcheb=chebcoeffs(f2);%coeficientes calculados por chebfun
long=length(coefcheb);
n=long-1;%grado polinomio interpolador
z=chebpts(long);%nodos de interpolacion
z=z(long:-1:1);
y1=f(z(n:-1:2));
y=[f(z);y1];%2n valores de la funcion

valoresfft=real(fft(y));
%Coeficientes con transformada de Fourier
coeffft=valoresfft(1:long);
coeffft(1)=coeffft(1)/(2*n);
coeffft(long)=coeffft(long)./(2*n);
for i=2:long-1
    coeffft(i)=coeffft(i)/n;
end
coeffft
error=norm(coefcheb-coeffft,inf)
```

- constantelebesgue.m

```
function [lambda]=constantelebesgue(pts,kind)
%pts es el numero de nodos de Chebyshev
%kind:
% =1 , para calcular la constante de Lebesgue para puntos de primera
%    especie.
% =2 , para calcular la constante de Lebesgue para puntos de segunda
%    especie.
%
chebkind(kind);
y=0;
for k=1:pts
    for j=1:pts
        %Polinomio j de Lagrange
        leb{j}=chebfun([zeros(1,j-1) 1 zeros(1,pts-j)]');
    end
    y=y+abs(leb{k});
end
lambda=norm(y,inf) %Constante de Lebesgue
end
```

- difcheb1.m

```
function [h]=difcheb1(t)
global n;
```

```
chebkind(1);%Puntos Chebyshev primera especie
fc1 = chebfun('1./(1+25*x.^2)',n);%Polinomio interpolador
h=-abs(runge(t)-fc1(t));
end
```

- difcheb2.m

```
function [h]=difcheb2(t)
global n;
chebkind(2);%Puntos Chebyshev segunda especie
fc2 = chebfun('1./(1+25*x.^2)',n);%Polinomio interpolador
h=-abs(runge(t)-fc2(t));
end
```

- difigual.m

```
function [h2]=difigual(t)
global n;
xx=linspace(-1,1,n);%Puntos equiespaciados
fe=polyfit(xx,runge(xx),n-1);%Polinomio interpolador
fe2=polyval(fe,t);
h2=-abs(runge(t)-fe2);
end
```

- difpn.m

```
function [h]=difpn(t)
global n;
chebkind(2);%Puntos Chebyshev segunda especie
fn = chebfun('sin(3*t)+exp(cos(t))',n+1);%Polinomio interpolador
h=-abs(sinexp(t)-fn(t));
end
```

- evaluacionpolinomio.m

```
clear all;
format short e;
y=chebfun('x');
p1=[];
p2=[];
condvander=[];
grado=[];
for k=[10:10:100]
    i=k/10;%Indice vector
    f = @(x) cos(k*x);%Funcion cos(kx)
    p=cos(k*y);%Interpolacion con chebfun
    p1(i)=p(0);
    grado(i)=length(p)-1;%Grado polinomio interpolador
```

```
    xx=chebpts(length(p));%Nodos de Chebyshev
    pf=polyfit(xx,f(xx),grado(i));
    M=fliplr(vander(xx));
    condvander(i)=cond(M);
    p2(i)=polyval(pf,0); %Valor en el 0 con la interpolacion
    %basada en la matriz de Vandermonde
end
grado,p1,p2,condvander
```

- maxdifabs.m

```
function [error] = maxdifabs(n)%Calcula la distancia maxima entre
%la funcion de Runge y los polinomios interpoladores en puntos de
%Chebyshev de segunda y primera especie y en puntos esquiespaciados
%Chebyshev segunda especie
format short e;
[y2,FVAL2,EXITFLAG,OUTPUT]=fminbnd('difcheb2',-1,1);
if EXITFLAG~=1
    print('WARNING')
end
%%%Chebyshev primera especie
[y1,FVAL1,EXITFLAG,OUTPUT]=fminbnd('difcheb1',-1,1);
if EXITFLAG~=1
    print('WARNING')
end
%%%Equiespaciados
e=0;
for i=[-1:0.1:0.9]
    [y,FVALE,EXITFLAG,OUTPUT]=fminbnd('difigual',i,i+0.1);
    if EXITFLAG~=1
        print('WARNING')
    end
    if FVALE<e
        e=FVALE;
    end
end
%%%
error=[FVAL2,FVAL1,e];
```

- maxdiftrunc.m

```
function [error] = maxdiftrunc(n)%Calcula la distancia maxima
%entre la funcion sinexp y el polinomios interpolador en
%puntos de Chebyshev de segunda especie y el polinomio truncado.
%%%Polinomio interpolador
[y1,FVAL,EXITFLAG,OUTPUT]=fminbnd('difpn',-1,1);
```

```
if EXITFLAG~=1
    print('WARNING')
end
%Polinomio truncado
format short e;
[y2,FVALT,EXITFLAG,OUTPUT]=fminbnd('truncado',-1,1);
if EXITFLAG~=1
    print('WARNING')
end
error=[FVAL,FVALT];
```

- meandistance.m

```
function y=meandistance(x)
    %%Funcion meandistance
    %Input: x vector
    %Output: Grafica con los puntos del vector x en el eje de
    %abcisas y en el eje de ordenadas la media geometrica de las
    %distancias de cada elemento de x respecto a los demas
    l=length(x);
    for s=1:l
       z=abs(x-x(s));%Distancia en valor absoluto
       z(s)=1;
       y(s)=nthroot(prod(z),l-1);%Raiz l-1
    end
    plot(x,y,'*r')
    xlabel('Puntos')
    ylabel('Distancia media geometrica')
end
```

- runge.m

```
function [f]=runge(t)
f=1./(1+25*(t).^2);
end
```

- sinexp.m

```
function [f]=sinexp(t)
f=sin(3*t)+exp(cos(t));
end
```

- truncado.m

```
function [h]=truncado(t)
global n;
%Polinomio truncado grado n
ft = chebfun('sin(3*t)+exp(cos(t))','trunc',n+1);
```

```
h=-abs(sinexp(t)-ft(t));%Diferencia en valor absoluto
end
```

Notación

Descripción	Expresión
Puntos de Chebyshev de segunda especie	$cos\left(\frac{j\pi}{n}\right)$
Puntos de Chebyshev de primera especie	$cos\left(\frac{(2j-1)\pi}{2n}\right)$
Polinomio de Chebyshev de grado n	$T_n(x) = cos(ncos^{-1}(x))$
Constante de Lebesgue para el conjunto S	$\Lambda_n(S)$
Polinomios de Lagrange	$l_k(x) = \prod_{j=0, j\neq k}^{n} (x - x_j)/(x_k - x_j)$
Polinomio nodal	$l(x) = \prod_{k=0}^{n}(x - x_k)$
Transformada discreta de Fourier	$Y_k = \sum_{j=0}^{N-1} f_j e^{\frac{-i2\pi kj}{N}}$
Valor de precisión de la máquina	$eps = 2.2204e - 16$

Bibliografía

[1] Chebfun. http://www.chebfun.org/.

[2] J. L. AURENTZ and L. N. TREFETHEN. Chopping a Chebyshev series. *ACM Transactions on Mathematical Software*, 43(4), 2017.

[3] Z. BATTLES and L. N. TREFETHEN. An extension of MATLAB to continuous functions and operators. *SIAM Journal on Scientific Computing*, 25(5):1743–1770, 2004.

[4] J. BERRUT and L. N. TREFETHEN. Barycentric Lagrange interpolation. *SIAM Journal on Scientific Computing*, 46(3):501–517, 2004.

[5] M. CLEVE. The Tetragamma function and numerical craftsmanship, Technical Note. The Mathworks Inc. https://es.mathworks.com/company/newsletters/articles/the-tetragamma-function-and-numerical-craftsmanship.html, 2002.

[6] D.R. KINCAID and W. CHENEY. *Numerical Analysis : Mathematics of Scientific Computing*. Pacific Grove, CA: Brooks/Cole, 1991.

[7] M.J.D. POWELL. *Approximation Theory and Methods*. Cambridge Univeristy Press, 1981.

[8] J. STOER and R. BULIRSCH. *Introduction to Numerical Analysis*. Texts in Applied Mathematics. Springer New York, 2002.

[9] L. N. TREFETHEN. Six myths of polinomial interpolation and quadrature. *Mathematics Today*, 47(4):184–188, 2012.

[10] L. N. TREFETHEN. *Approximation Theory and Approximation Practice*. SIAM Journal on Scientific Computing, 2013.

[11] K. XU. The Chebyshev points of the first kind. *Applied Numerical Mathematics*, (102):17–30, 2016.