525476 Métodos Espectrales: Bases, expansiones y operaciones rápidas

Leonardo E. Figueroa

CI^{2}MA y Departamento de Ingeniería Matemática
Universidad de Concepción

2022-2

Bases, expansiones y operaciones rápidas

Bases, expansiones y operaciones rápidas

El sistema de Fourier

La sucesión de funciones $\left(e^{\imath k \cdot}\right)_{k\in\mathbb{Z}}$ forma un sistema ortogonal en $L^2(0, 2\pi)$:

$$(\forall\, k, l \in \mathbb{Z}) \quad \int_0^{2\pi} e^{\imath kx}\, \overline{e^{\imath lx}}\, \mathrm{d}x = 2\pi\, \delta_{k,l}.$$

Definición 1

Se dice que una sucesión $(e_n)_{n\in\mathbb{N}_0}$ en un espacio de Hilbert H es una *base hilbertiana* o *sistema ortonormal completo* de H si satisface las propiedades

1. $\left(\forall\, m, n \in \mathbb{N}_0^2\right) \quad \langle e_m, e_n\rangle_H = \delta_{m,n}$,
2. la envoltura lineal de $(e_n)_{n\in\mathbb{N}_0}$ es densa en H.

Teorema 2

Sea $(e_n)_{n\in\mathbb{N}_0}$ una base hilbertiana de un espacio de Hilbert H. Entonces,

$$(\forall\, u \in H) \quad u = \lim_{n\to\infty} \sum_{k=0}^{n} \langle u, e_k\rangle_H e_k = \sum_{k=0}^{\infty} \langle u, e_k\rangle_H e_k$$

y

$$\|u\|_H^2 = \sum_{k=0}^{\infty} |\langle u, e_k\rangle_H|^2 .$$

Recíprocamente, dada una sucesión $(\alpha_n)_{n\in\mathbb{N}_0}$ en $\ell^2(\mathbb{C})$, la serie $\sum_{k=0}^{\infty} \alpha_k e_k$ converge a un elemento $u \in H$ tal que, para todo k en $\mathbb{N}_0$, $\langle u, e_k\rangle_H = \alpha_k$ y $\|u\|_H^2 = \sum_{k=0}^{\infty} |\alpha_k|^2$.

Demostración: Este es un resultado estándar de análisis funcional. □

Lema 3

$\left(e^{\imath k \cdot}/\sqrt{2\pi}\right)_{k\in\mathbb{Z}}$ es una base hilbertiana de $\mathrm{L}^2(0, 2\pi)$.

Demostración: No es difícil. □

Corolario 4 (Serie de Fourier)

Para todo $u \in \mathrm{L}^2(0, 2\pi)$,

$$u = \sum_{k\in\mathbb{Z}} \hat{u}_k \, e^{\imath k \cdot}, \quad \text{donde} \quad (\forall\, k \in \mathbb{Z}) \quad \hat{u}_k = \frac{1}{2\pi} \int_0^{2\pi} u(x)\, \overline{e^{\imath k x}}\, \mathrm{d}x.$$

También

$$\|u\|^2_{\mathrm{L}^2(0,2\pi)} = 2\pi \sum_{k\in\mathbb{Z}} |\hat{u}_k|^2 . \tag{1}$$

Demostración: Directo del lema 3 y el teorema 2. □

Definición 5

Dado $N \in \mathbb{N}$ con N par, definimos

$$S_N = \operatorname{span}\left((e^{\imath k \cdot})_{k=-N/2}^{N/2-1} \right).$$

Definimos también el proyector ortogonal $P_N \colon \mathrm{L}^2(0, 2\pi) \to S_N$; esto es, para todo $u \in \mathrm{L}^2(0, 2\pi)$,

$$(\forall\, x \in (0, 2\pi)) \quad P_N(u)(x) = \sum_{k=-N/2}^{N/2-1} \hat{u}_k \, e^{\imath k x}.$$

La convención de la suma en la definición 5 corresponde a las típicas implementaciones computacionales.

Debido al corolario 4, para todo $u \in \mathrm{L}^2(0, 2\pi)$,

$$\int_0^{2\pi} |u - P_N(u)|^2 \, \mathrm{d}x \xrightarrow{N\to\infty} 0;$$

esto es, la serie de Fourier truncada converge en $\mathrm{L}^2(0, 2\pi)$ a u. Una pregunta que surge es a **qué tasa ocurre la convergencia** de $P_N(u)$ a u. Para eso, observamos que si $u \in \mathrm{L}^2(0, 2\pi)$ tiene una derivada continua en $[0, 2\pi]$,

$$\begin{aligned}\hat{u}_k &= \frac{1}{2\pi} \int_0^{2\pi} u(x)\, e^{-\imath k x} \, \mathrm{d}x \\ &= \frac{-1}{2\pi\imath k} \left(u(2\pi) - u(0)\right) + \frac{1}{2\pi\imath k} \int_0^{2\pi} u'(x)\, e^{-\imath k x} \, \mathrm{d}x.\end{aligned}$$

Entonces, $\hat{u}_k = \mathcal{O}(|k|^{-1})$. Ahora, si u tiene **dos** derivadas continuas en $[0, 2\pi]$ **y** $u(2\pi) = u(0)$, entonces el primer término arriba

desaparece y puedo hacer integración por partes de nuevo para obtener $\hat{u}_k = \mathcal{O}(|k|^{-2})$. Aplicando este argumento sucesivamente se tiene que si u es m veces continuamente diferenciable en $[0, 2\pi]$ ($m \geq 1$) y si $u^{(j)}$ es periódica para $j \leq m-2$, entonces

$$\hat{u}_k = \mathcal{O}(|k|^{-m}).$$

Este mismo resultado también puede obtenerse (a partir del mismo argumento basado en integraciones por partes) asumiendo que u es meramente $m-1$ veces diferenciable casi en todas partes en $(0, 2\pi)$, que $u^{(m-1)}$ es de variación acotada en $[0, 2\pi]$ y que $u^{(j)}$ es periódica para $j \leq m-2$. En este caso la integral del lado derecho de la integración por partes realizada más arriba debe reemplazarse por la integral de Riemann–Stieltjes $\int_0^{2\pi} e^{-\imath kx}\, \mathrm{d}u(x)$.
Si $u \in \mathrm{C}^\infty([0, 2\pi])$ y todas sus derivadas son periódicas, entonces obviamente $\hat{u}_k = \mathcal{O}(|k|^{-m})$ para todo $m \in \mathbb{N}$.

Ahora queremos emplear estas tasas de decaimiento de los coeficientes de Fourier $\hat{u}_k$ para estimar la tasa de convergencia de la serie de Fourier $\sum_{k\in\mathbb{Z}} \hat{u}_k\, e^{\imath k \cdot}$ (equivalentemente, la tasa a la que $P_N(u)$ converge a u). Para eso hallaremos útil la estimación

$$(\forall\, (\sigma, L) \in (1,\infty)\times\mathbb{N}) \qquad \sum_{k=L+1}^{\infty} k^{-\sigma} \leq \frac{L^{1-\sigma}}{\sigma-1}. \tag{2}$$

Lema 6

Supongamos que $u \in \mathrm{L}^2(0, 2\pi)$ cumple alguna de las condiciones siguientes con $m \in \mathbb{N}$.

- u es m veces continuamente diferenciable en $[0, 2\pi]$ y $u^{(j)}$ es periódica para $j \leq m - 2$.
- u es $m - 1$ veces diferenciable casi en todas partes en $(0, 2\pi)$, $u^{(m-1)}$ es de variación acotada y $u^{(j)}$ es periódica para $j \leq m - 2$.

Entonces,

$$\|u - P_N(u)\|_{\mathrm{L}^2(0,2\pi)} = \mathcal{O}\left(N^{1/2-m}\right).$$

Demostración: Por la definición de los $\hat{u}_k$ dada en el corolario 4, la definición de P_N en la definición 5 y por la linealidad del producto interior

$$\frac{1}{2\pi}\int_0^{2\pi}\left(u(x)-P_N(u)(x)\right)\overline{e^{\imath kx}}\,\mathrm{d}x=\begin{cases}0 & \text{si } k\in\{-N/2,\ldots,N/2-1\}\\ \hat{u}_k & \text{en otro caso.}\end{cases}$$

Como $u-P_N(u)$ satisface la hipótesis del corolario 4,

$$\|u-P_N(u)\|^2_{\mathrm{L}^2(0,2\pi)}=2\pi\sum_{k<-N/2\ \vee\ k\geq N/2}|\hat{u}_k|^2\,. \tag{3}$$

De la discusión anterior y por las hipótesis de este lema, sabemos que existe un entero positivo K y una constante $C>0$ tal que $|k|\geq K$ implica que $|\hat{u}_k|\leq C\,|k|^{-m}$. Entonces, si N es lo

suficientemente grande, la suma en el lado derecho de (3) involucra solo términos de índice k con $|k| \geq K$ y así

$$\|u - P_N(u)\|^2_{\mathrm{L}^2(0,2\pi)} \leq C^2\, 2\pi \sum_{k<-N/2 \ \vee \ k\geq N/2} |k|^{-2m}$$

$$\leq C^2\, 4\pi \sum_{k=N/2}^{\infty} k^{-2m}.$$

Ahora, si además, $N \geq 4$ (lo que puede ser absorbido en la condición de que N sea lo suficientemente grande) y $m \geq 1$, $N/2 - 1 \in \mathbb{N}$ y $2m > 1$ por lo que podemos apelar a la desigualdad (2) para obtener

$$\|u - P_N(u)\|^2_{\mathrm{L}^2(0,2\pi)} \leq C^2 4\pi \frac{(N/2-1)^{1-2m}}{2m-1}.$$

Sacando raíces cuadradas y observando que existe una constante $\tilde{C} > 0$ tal que $(N/2-1)^{1/2-m} \leq \tilde{C} N^{1/2-m}$ obtenemos el resultado deseado. □

Observación 7

El lema 6 es suboptimal. Es posible eliminar el $1/2$ de $N^{1/2-m}$.

En muchas aplicaciones prácticas, no se pueden implementar métodos numéricos basados en series de Fourier **exactamente** de la manera sugerida por el tratamiento que hemos estado estudiando. Algunas de las dificultades son:

- Los coeficientes de Fourier de una función arbitraria no se conocen exactamente y deben ser aproximados de alguna manera.

- Se necesita una manera eficiente de conectar información en el dominio de la frecuencia ("k") con información en el dominio del espacio/tiempo ("x").
- Cualquier no-linealidad de cierta complejidad mínima conduce a grandes complicaciones.

La clave para superar estas dificultades es el uso de la transformada de Fourier discreta y la relacionada serie de Fourier discreta que introduciremos a continuación.
Dado $N \in \mathbb{N}$, N par, consideremos los nodos

$$(\forall\, j \in \{0, \ldots, N-1\}) \qquad x_j = \frac{2\pi j}{N}. \tag{4}$$

Los **coeficientes de Fourier discretos** de una función $u\colon [0, 2\pi] \to \mathbb{C}$ con respecto a estos nodos se definen por

$$(\forall\, k \in \{-N/2, \ldots, N/2-1\}) \qquad \tilde{u}_k := \frac{1}{N} \sum_{j=0}^{N-1} u(x_j)\, \overline{e^{\imath k x_j}}. \tag{5}$$

Esta fórmula es lo que sale de aplicar la regla del trapecio en los puntos $(x_j)_{j=0}^{N-1}$ para aproximar al coeficiente de Fourier $\hat{u}_k$.
Los valores puntuales $u(x_j)$ pueden obtenerse a partir de los coeficientes de Fourier discretos (esto es, (5) puede invertirse). La fórmula correspondiente es

$$(\forall\, j \in \{0, \ldots, N-1\}) \qquad u(x_j) = \sum_{k=-N/2}^{N/2-1} \tilde{u}_k \, e^{\imath k x_j}; \qquad (6)$$

Para probar (6) necesitamos el resultado que sigue.

Proposición 8 (Relación de ortogonalidad discreta)

$$(\forall\, p \in \mathbb{Z}) \quad \frac{1}{N} \sum_{j=0}^{N-1} e^{\imath p x_j} = \begin{cases} 1 & \text{si } p = 0 \text{ mód } N, \\ 0 & \text{en otro caso.} \end{cases}$$

Demostración: Dado cualquier $m \in \mathbb{Z}$,

$$\sum_{j=0}^{N-1} e^{\imath(p+mN)x_j} = \sum_{j=0}^{N-1} e^{\imath p x_j} e^{\imath mN 2\pi j/N} = \sum_{j=0}^{N-1} e^{\imath p x_j} \underbrace{e^{\imath 2\pi mj}}_{1}.$$

Por lo tanto, el lado izquierdo de la igualdad deseada depende de p solamente a través de p mód N. Basta considerar solamente los casos en que $p \in \{0, \ldots, N-1\}$.

Si $p = 0$, inmediatamente

$$\frac{1}{N} \sum_{j=0}^{N-1} e^{\imath\, 0\, x_j} = 1.$$

Notemos que

$$\begin{aligned}\sum_{j=0}^{N-1} e^{\imath p x_j} &= \sum_{j=0}^{N/2-1} e^{\imath p x_j} + \sum_{j=N/2}^{N-1} e^{\imath p 2\pi \frac{(-N)}{N}} e^{\imath p x_j} \\ &= \sum_{j=0}^{N/2-1} e^{\imath p \frac{2\pi j}{N}} + \sum_{j=N/2}^{N-1} e^{\imath p 2\pi \frac{j-N}{N}} \qquad (7) \\ &= \sum_{j=0}^{N/2-1} e^{\imath p \frac{2\pi j}{N}} + \sum_{j=-N/2}^{-1} e^{\imath p \frac{2\pi j}{N}} = \sum_{j=-N/2}^{N/2-1} e^{\imath p \frac{2\pi j}{N}}.\end{aligned}$$

Entonces, si $p \in \{1, \ldots, N-1\}$,

$$\overbrace{e^{\frac{\imath p\pi}{N}}}^{\neq 0} \overbrace{\operatorname{sen}\left(\frac{p\pi}{N}\right)}^{\neq 0} \sum_{j=0}^{N-1} e^{\imath p x_j} \overset{(7)}{=} e^{\frac{\imath p\pi}{N}} \operatorname{sen}\left(\frac{p\pi}{N}\right) \sum_{j=-N/2}^{N/2-1} e^{\imath p x_j}$$

$$= \sum_{j=-N/2}^{N/2-1} e^{\frac{\imath p\pi}{N}} e^{\imath p \frac{2\pi j}{N}} \frac{e^{\frac{\imath p\pi}{N}} - e^{-\frac{\imath p\pi}{N}}}{2\imath}$$

$$= \frac{1}{2\imath} \sum_{j=-N/2}^{N/2-1} \left(e^{\frac{\imath p 2\pi}{N}(j+1)} - e^{\frac{\imath p 2\pi}{N} j} \right)$$

$$\overset{\text{tel.}}{=} \frac{1}{2\imath} \left(e^{\frac{\imath p 2\pi}{N} N/2} - e^{\frac{\imath p 2\pi}{N}(-N/2)} \right)$$

$$= \frac{e^{\imath p\pi} - e^{-\imath p\pi}}{2\imath} = \operatorname{sen}(p\pi) = 0.$$

□

Ahora le pondremos nombre a las operaciones definidas por (5) y (6) probaremos que, efectivamente, una es inversa de la otra.

Definición 9

Dado $N \in \mathbb{N}$, N par, se definen la **transformada de Fourier discreta** $\mathrm{DFT}\colon \mathbb{C}^N \to \mathbb{C}^N$ y la **transformada de Fourier discreta inversa** $\mathrm{IDFT}\colon \mathbb{C}^N \to \mathbb{C}^N$, respectivamente por

$$\left(\forall\, u = (u_j)_{j=0}^{N-1} \in \mathbb{C}^N\right) \quad \mathrm{DFT}(u) := \left(\frac{1}{N}\sum_{j=0}^{N-1} u_j\, e^{-\imath k \frac{2\pi j}{N}}\right)_{k=-N/2}^{N/2-1}$$

y

$$\left(\forall\, \tilde{u} = (\tilde{u}_k)_{k=-N/2}^{N/2-1} \in \mathbb{C}^N\right) \quad \mathrm{IDFT}(\tilde{u}) := \left(\sum_{k=-N/2}^{N/2-1} \tilde{u}_k\, e^{\imath k \frac{2\pi j}{N}}\right)_{j=0}^{N-1}.$$

Proposición 10

Sea $N \in \mathbb{N}$, N par. Entonces, la transformada de Fourier discreta DFT y la transformada de Fourier discreta inversa IDFT son una la inversa de la otra.

Demostración: Sea $u = (u_j)_{j=0}^{N-1} \in \mathbb{C}^N$. Definimos a $\tilde{u} = (\tilde{u}_k)_{k=-N/2}^{N/2-1} := \mathrm{DFT}(u)$ y a $v = (v_j)_{j=0}^{N-1} := \mathrm{IDFT}(\tilde{u}) \in \mathbb{C}^N$. Entonces, por la Definición 9, para cada $j \in \{0, \ldots, N-1\}$,

$$v_j = \sum_{k=-N/2}^{N/2-1} \tilde{u}_k e^{\imath k \frac{2\pi j}{N}} = \sum_{k=-N/2}^{N/2-1} \frac{1}{N} \sum_{m=0}^{N-1} u_m e^{-\imath k \frac{2\pi m}{N}} e^{\imath k \frac{2\pi j}{N}}$$

$$= \sum_{m=0}^{N-1} u_m \frac{1}{N} \sum_{k=-N/2}^{N/2-1} e^{\imath (j-m) \frac{2\pi k}{N}} \overset{(7)}{=} \sum_{m=0}^{N-1} u_m \frac{1}{N} \sum_{k=0}^{N-1} e^{\imath (j-m) x_k}$$

$$= \sum_{m=0}^{N-1} u_m \, \delta_{0, j-m \bmod N} = \sum_{m=0}^{N-1} u_m \, \delta_{j,m} = u_j.$$

Así, hemos probado que $\mathrm{IDFT} \circ \mathrm{DFT}$ es el mapa identidad de $\mathbb{C}^N$.

La demostración correspondiente para $\mathrm{IDFT} \circ \mathrm{DFT}$ es análoga. □

Definición 11

Dada una función u definida en $[0, 2\pi]$ definimos la **serie de Fourier discreta** de u, denotada por $I_N(u)$, por

$$(\forall\, x \in \mathbb{R}) \qquad I_N(u)(x) := \sum_{k=-N/2}^{N/2-1} \tilde{u}_k \, e^{\imath k x},$$

donde los $\tilde{u}_k$ están dados por (5) o, equivalentemente por la Definición 9, $\tilde{u}_k = \mathrm{DFT}\left((u(x_j))_{j=0}^{N-1}\right)_k$.

Es claro que $I_N(u) \in S_N$. Por (6) —equivalentemente, por la Proposición 10—, es inmediato que $I_N(u) \in S_N$ e

$I_N(u)(x_j) = u(x_j)$; esto es, $I_N(u)$ es el interpolador trigonométrico de u en S_N en los nodos x_j.

Otra forma de escribir el interpolador/serie de Fourier discreta $I_N(u)$ se puede obtener escribiendo explícitamente a los $\tilde{u}_k$ de acuerdo a (5) y reordenando las sumas:

$$
\begin{aligned}
I_N(u)(x) &= \sum_{k=-N/2}^{N/2-1} \overbrace{\frac{1}{N} \sum_{j=0}^{N-1} u(x_j)\, e^{-\imath k x_j}}^{\tilde{u}_k}\; e^{\imath k x} \\
&= \sum_{j=0}^{N-1} u(x_j) \underbrace{\frac{1}{N} \sum_{k=-N/2}^{N/2-1} e^{\imath k (x - x_j)}}; \\
\end{aligned}
$$

esto es,

$$I_N(u) = \sum_{j=0}^{N-1} u(x_j)\,\psi_j, \tag{8}$$

donde

$$(\forall\, j \in \{0, \ldots, N-1\}) \quad \psi_j(x) := \frac{1}{N} \sum_{k=-N/2}^{N/2-1} e^{\imath k(x-x_j)}. \tag{9}$$

Notar que, para todo $j, \ell \in \{0, \ldots, N-1\}$,

$$\psi_j(x_\ell) = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} e^{\imath k(\frac{2\pi\ell}{N} - \frac{2\pi j}{N})} = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} e^{\imath(\ell-j)\frac{2\pi k}{N}}$$

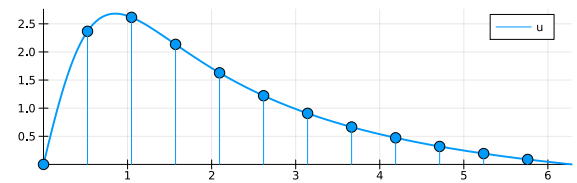
$$\overset{(7)}{=} \frac{1}{N} \sum_{k=0}^{N-1} e^{\imath(\ell-j)\frac{2\pi k}{N}} \overset{\text{Prop. 8}}{=} \delta_{j,\ell};$$

esto es, los ψ_j son las funciones delta discretas del subespacio S_N con respecto a los nodos x_j.

```
using Plots
using LinearAlgebra # Para función 'norm'

u(x::Float64) = x*(2π-x)/(x^2+1) # Función periódica cuyas derivadas ya no lo son
N = 12
nodes = [2π*j/N for j=0:N-1]  # Nodos equiespaciados
vu = u.(nodes) # Vector de valores nodales
```

```
12-element Vector{Float64}:
 0.0
 2.36683201936236
 2.615212325415677
 2.1347986825739973
 1.6287007520150147
 1.221744428142348
 0.9080003316496248
 0.6647981077210341
 0.47303995539283233
 0.318969609603995
 0.1929616046547867
 0.08824881746238264
```

```
p = plot(u, xlims=[0, 2π], color=1, linewidth=2, label="u", size=(600,200),
framestyle=:origin)
scatter!(p, nodes, vu, color=1, markersize=6, label="")
sticks!(p, nodes, vu, color=1, label="")
```




```
wavenumbers = -N/2:N/2-1
# Cómputo de valores modales empleando la DFT explícitamente (ESTO ES LENTO; EN LA
PRÁCTICA SE USA LA FFT)
vũ = [1/N*sum(vu .* exp.(-im*k*nodes)) for k in wavenumbers]
```

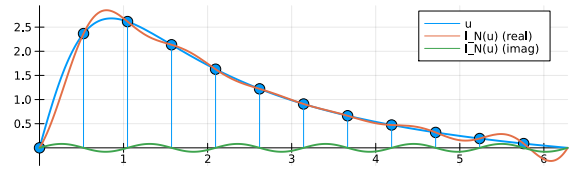
```
12-element Vector{ComplexF64}:
  -0.08145639131151511 + 3.090133807849151e-17im
  -0.08726269433127506 + 0.011252630631998517im
  -0.10533336510463337 + 0.032840185414268051m
  -0.13453612952602012 + 0.084975037445940821m
  -0.15249279941028826 + 0.215656693827855681m
 -0.005201259055111218 + 0.52767967505644311m
    1.0511088861661708 - 0.0im
 -0.005201259055111218 - 0.52767967505644311m
  -0.15249279941028826 - 0.215656693827855681m
  -0.13453612952602012 - 0.084975037445940821m
  -0.10533336510463337 - 0.032840185414268051m
  -0.08726269433127506 - 0.011252630631998517im
```


Figura 1: Ilustración en Julia del uso de la DFT y de la IDFT (1/3).

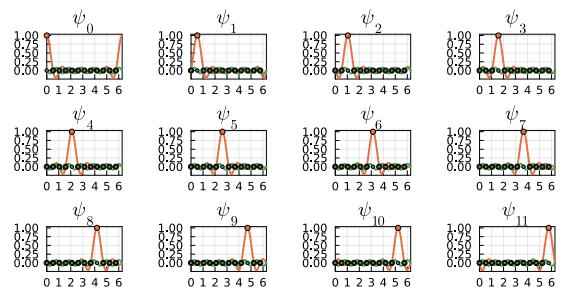
```
# Reconstrucción de valores nodales empleando la IDFT explícitamente (ESTO ES LENTO; EN
LA PRÁCTICA SE USA LA IFFT)
vu2 = [sum(vũ .* exp.(im*wavenumbers*x)) for x in nodes]
norm(vu - vu2)/norm(vu)
```

```
3.89524674571051l3e-16
```

```
interpolator(x::Float64) = sum(vũ .* exp.(im*wavenumbers*x)) # Función interpoladora
plot!(p, x -> real(interpolator(x)), xlims=[0, 2π], color=2, linewidth=2, label="I_N(u)
(real)")
plot!(p, x -> imag(interpolator(x)), xlims=[0, 2π], color=3, linewidth=2, label="I_N(u)
(imag)")
```



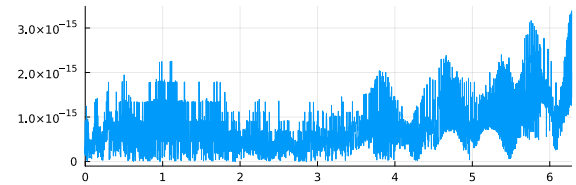

```
# Funciones delta discretas
ψ(x::Float64, j::Int64) = 1/N*sum(exp.(im*wavenumbers*(x - nodes[j+1])))
l = grid(3,4)
ψtitles = ["\$\\psi_{$j}\$" for spuriousIndex=1:1, j=0:N-1]
ψp = plot([x -> real(ψ(x, j)) for j=0:N-1], xlims=[0, 2π], layout=l, linewidth=2,
color=2, label="", size=(600,300), framestyle=:box, legendposition=:outerright,
title=ψtitles)
plot!(ψp, [x -> imag(ψ(x, j)) for j=0:N-1], xlims=[0, 2π], layout=l, linewidth=2,
color=3, label="")
scatter!(ψp, nodes, [Float64(i==j) for i=0:N-1, j=0:N-1], layout=l, markersize=3,
color=2, label="")
scatter!(ψp, nodes, zeros(N,N), layout=l, markersize=2, color=3, label="")
```



2

Figura 2: Ilustración en Julia del uso de la DFT y de la IDFT (2/3).

```
# Forma alternativa del interpolador y comparación
interpolator2(x::Float64) = sum(vu .* [ψ(x, j) for j=0:N-1])
plot(x -> abs(interpolator(x) - interpolator2(x)), xlims=[0, 2π], size=(600,200),
label="")
```



3

Figura 3: Ilustración en Julia del uso de la DFT y de la IDFT (3/3).

Proposición 12

Sea $N \in \mathbb{N}$, N par. Si $u \in \mathrm{L}^2(0, 2\pi)$ es tal que su serie de Fourier converge absoluta y puntualmente en los nodos x_j definidos en (4), entonces

$$(\forall\, k \in \{-N/2, \ldots, N/2-1\}) \qquad \tilde{u}_k = \hat{u}_k + \sum_{m \in \mathbb{Z}\setminus\{0\}} \hat{u}_{k+Nm}.$$

Demostración: De la definición de los $\tilde{u}_k$ en (5),

$$\tilde{u}_k = \frac{1}{N} \sum_{j=0}^{N-1} u(x_j)\, \overline{e^{\imath k x_j}} = \frac{1}{N} \sum_{j=0}^{N-1} \left[\sum_{\ell \in \mathbb{Z}} \hat{u}_\ell\, e^{\imath \ell x_j} \right] \overline{e^{\imath k x_j}}$$

$$= \sum_{\ell \in \mathbb{Z}} \hat{u}_\ell \frac{1}{N} \sum_{j=0}^{N-1} e^{\imath(\ell - k) x_j}$$

Usando la relación de ortogonalidad discreta (cf. Proposición 8) se obtiene el resultado deseado. □

La Proposición 12 demuestra que el modo k del interpolador trigonométrico de u depende no solamente del modo k de u, sino también de todos los modos que *alias* ("suplantan", "se mimetizan con") el modo k.

Para todo $m \in \mathbb{Z}$ el modo $k + mN$ es indistinguible del modo k en los nodos ya que $e^{\imath(k+mN)x_j} = e^{\imath k x_j}$, $j \in \{0, \ldots, N-1\}$.

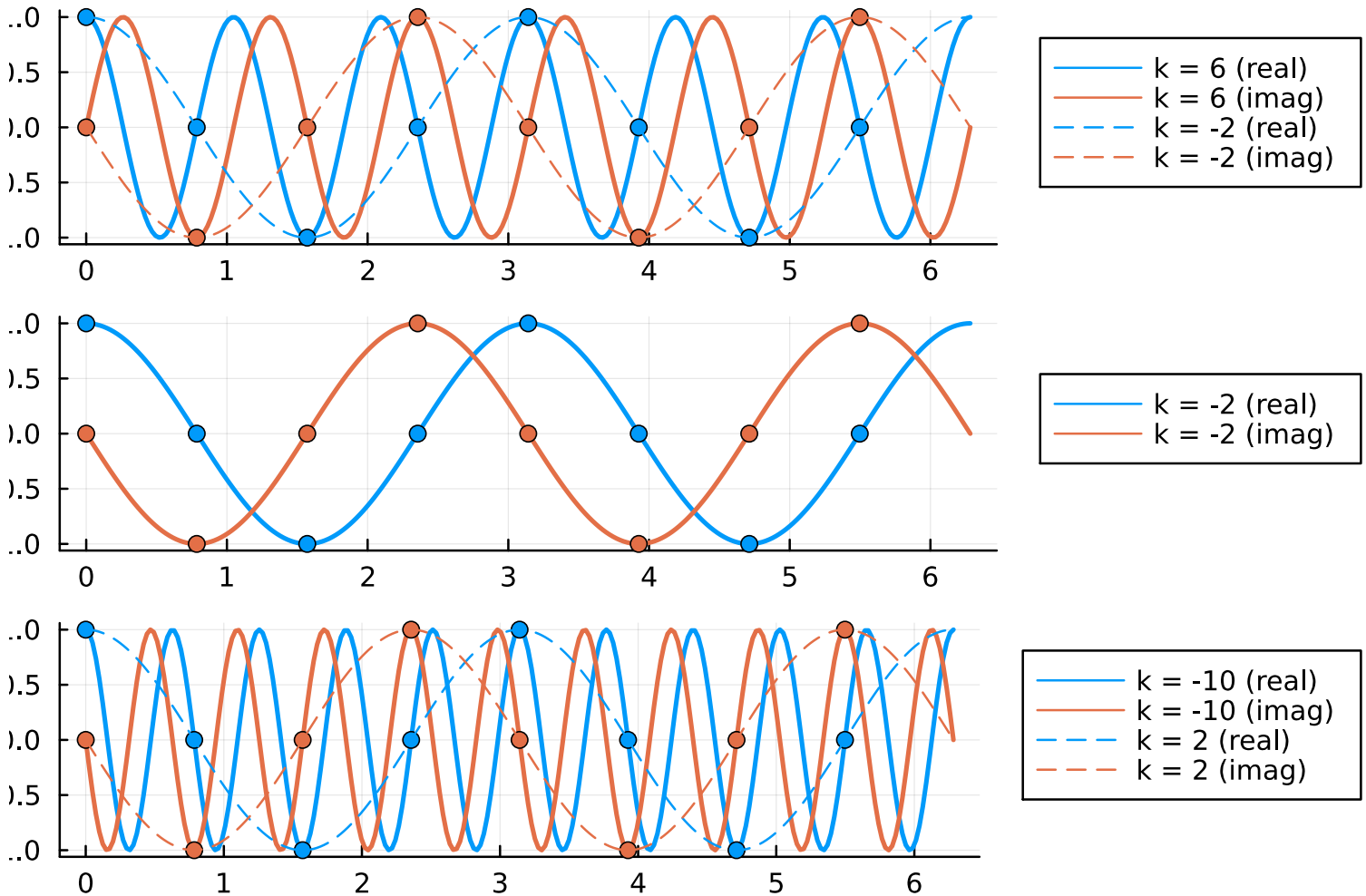


Figura 4: Ilustración del fenómeno del *aliasing*. En una grilla de $N = 8$ nodos las ondas planas $e^{\imath kx}$ con $k = 6$ (arriba) y $k = -10$ (abajo) no se pueden distinguir de aquella con $k = -2$ (al medio).

La tesis de la Proposición 12 se puede expresar como

$$\left(\forall\, u \in \mathrm{L}^2(0, 2\pi)\right) \qquad I_N(u) = P_N(u) + R_N(u), \tag{10}$$

donde

$$R_N(u)(x) := \sum_{k=-N/2}^{N/2-1} \left(\sum_{m \in \mathbb{Z}\setminus\{0\}} \hat{u}_{k+Nm} \right) e^{\imath k x}. \tag{11}$$

A $R_N(u)$ se le conoce como el *error de aliasing*. Es ortogonal al error de truncación $u - P_N(u)$ (¿por qué?), por lo que

$$\|u - I_N(u)\|^2_{\mathrm{L}^2(0,2\pi)} = \|u - P_N(u)\|^2_{\mathrm{L}^2(0,2\pi)} + \|R_N(u)\|^2_{\mathrm{L}^2(0,2\pi)}\,.$$

Por lo tanto, el error cometido al interpolar es siempre mayor o igual al error de truncación (¿obvio?). Pero en la mayor parte de los casos prácticos el error de interpolación se comporta de la misma manera que el error de truncación.

Otra propiedad de la DFT y de la IDFT es que son múltiplos escalares de transformaciones ortogonales:

Proposición 13

Para todo $u, v \in \mathbb{C}^N$,

$$\langle \mathrm{DFT}(u), \mathrm{DFT}(v) \rangle_{\mathbb{C}^N} = \frac{1}{N} \langle u, v \rangle_{\mathbb{C}^N}, \tag{12}$$

$$\langle \mathrm{IDFT}(u), \mathrm{IDFT}(v) \rangle_{\mathbb{C}^N} = N \langle u, v \rangle_{\mathbb{C}^N}. \tag{13}$$

Demostración: Usando que la DFT y la IDFT son una la inversa de la otra,

$$\begin{aligned}
\langle \mathrm{DFT}(u), \mathrm{DFT}(v) \rangle_{\mathbb{C}^N} &= \sum_{k=-N/2}^{N/2-1} \mathrm{DFT}(u)_k \overline{\left(\frac{1}{N} \sum_{\ell=0}^{N-1} v_\ell \, \overline{e^{\imath k x_\ell}} \right)} \\
&= \sum_{k=-N/2}^{N/2-1} \mathrm{DFT}(u)_k \left(\frac{1}{N} \sum_{\ell=0}^{N-1} \overline{v_\ell} \, e^{\imath k x_\ell} \right) \\
&= \frac{1}{N} \sum_{\ell=0}^{N-1} \overline{v_\ell} \sum_{k=-N/2}^{N/2-1} \mathrm{DFT}(u)_k \, e^{\imath k x_\ell} \\
&= \frac{1}{N} \sum_{\ell=0}^{N-1} \overline{v_\ell} \, \mathrm{IDFT}(\mathrm{DFT}(u))_\ell \\
&= \frac{1}{N} \sum_{\ell=0}^{N-1} \overline{v_\ell} \, u_\ell = \frac{1}{N} \langle u, v \rangle_{\mathbb{C}^N},
\end{aligned}$$

con lo que hemos probado (12). Usando este resultado, dados a y b en $\mathbb{C}^N$,

$$\begin{aligned}\langle \mathrm{IDFT}(a), \mathrm{IDFT}(b)\rangle_{\mathbb{C}^N} \\ = N\langle \mathrm{DFT}(\mathrm{IDFT}(a)), \mathrm{DFT}(\mathrm{IDFT}(b))\rangle_{\mathbb{C}^N} = N\langle a, b\rangle_{\mathbb{C}^N}.\end{aligned}$$

□

La manera en que se calculan las derivadas en un método espectral depende de si se está trabajando con una representación de una función en el espacio de la frecuencia o en el espacio del tiempo/espacio físico.

La diferenciación en el espacio de la frecuencia es muy sencilla: Si $u(x) = \sum_{k\in\mathbb{Z}} \hat{u}_k e^{\imath kx}$ es la serie de Fourier de u, entonces

$$u'(x) = \sum_{k\in\mathbb{Z}} \imath k\, \hat{u}_k\, e^{\imath kx} \tag{14}$$

es la serie de Fourier de la derivada de u. Por lo tanto,

$$(P_N(u))' = P_N(u'); \tag{15}$$

esto es, los operadores de truncación y de diferenciación conmutan. La serie (14) converge en $\mathrm{L}^2(0, 2\pi)$ si la derivada distribucional de u pertenece a $\mathrm{L}^2(0, 2\pi)$.

Si se representa a una función u por sus valores en los nodos definidos en (4), el procedimiento de diferenciación consiste en construir la serie de Fourier discreta de acuerdo a (5) y la Definición 11, multiplicar los coeficientes de Fourier discretos por $\imath k$ y pasar la representación en la frecuencia resultante a una representación espacial usando la fórmula de inversión (6).

Entonces, los valores $(\mathcal{D}_N(u))_j$ de la derivada aproximada en los nodos x_j están dados por

$$(\forall\, j \in \{0, \ldots, N-1\}) \quad (\mathcal{D}_N(u))_j = \sum_{k=-N/2}^{N/2-1} \tilde{u}_k^{(1)}\, e^{\imath k x_j}, \tag{16}$$

donde

$$(\forall\, k \in \{-N/2, \ldots, N/2-1\}) \quad \tilde{u}_k^{(1)} = \imath k\, \tilde{u}_k = \frac{\imath k}{N} \sum_{l=0}^{N-1} u(x_l)\, \overline{e^{\imath k x_l}}. \tag{17}$$

El procedimiento sugerido por (16)–(17) equivale a computar las evaluaciones en los nodos de la derivada de la serie de Fourier discreta de u; esto es,

$$\mathcal{D}_N(u) = I_N(u)'. \tag{18}$$

Notar que, en general,

$$\mathcal{D}_N(u) \neq P_N(u') = P_N(u)'.$$

Otro resultado negativo es que, en general,

$$I_N(u)' \neq I_N(u').$$

Sin embargo, se puede demostrar que $I_N(u)' - I_N(u')$ decae a la misma tasa con respecto a N que $u' - P_N(u')$.

Bases, expansiones y operaciones rápidas

Polinomios ortogonales en el intervalo unitario

Definición 14

Dado un abierto $\Omega \subset \mathbb{R}^d$ y dada una función medible $w\colon \Omega \to \overline{\mathbb{R}}$, positiva y finita casi en todas partes, definimos al espacio de Lebesgue ponderado $\mathrm{L}^2_w(\Omega) := w^{-1/2}\,\mathrm{L}^2(\Omega)$.

El espacio $\mathrm{L}^2_w(\Omega)$, equipado con su producto interior natural

$$\langle u, v\rangle_{\mathrm{L}^2_w(\Omega)} := \langle w^{1/2}u, w^{1/2}v\rangle_{\mathrm{L}^2(\Omega)} = \int_\Omega u(x)\,\overline{v(x)}\,w(x)\,\mathrm{d}x,$$

hereda de $\mathrm{L}^2(\Omega)$ su calidad de espacio de Hilbert.

Definición 15

Sea $\lambda > -1/2$. Definimos a la función **peso de Gegenbauer** de parámetro λ por $w_\lambda : (-1,1) \to \mathbb{R}$ por

$$w_\lambda(x) := (1-x^2)^{\lambda-1/2}.$$

Abreviamos $\mathrm{L}^2_\lambda := \mathrm{L}^2_{w_\lambda}(-1,1)$ y $\langle u, v\rangle_\lambda := \langle u, v\rangle_{\mathrm{L}^2_{w_\lambda}(-1,1)}$.

Como $\lambda > -1/2$ y $1-x^2 = (1+|x|)(1-|x|)$, dada $f \in \mathrm{L}^\infty(-1,1)$,

$$\begin{aligned}
\|f\|^2_\lambda &= \int_{-1}^{1} |f(x)|^2 (1-x^2)^{\lambda-1/2}\,\mathrm{d}x \\
&\leq \|f\|^2_{\mathrm{L}^\infty(-1,1)} \sup_{x\in(-1,1)} (1+|x|)^{\lambda-1/2} \int_{-1}^{1} (1-|x|)^{\lambda-1/2}\,\mathrm{d}x \\
&= \|f\|^2_{\mathrm{L}^\infty(-1,1)} \times \begin{cases} 2^{\lambda-1/2} & \text{si } \lambda \geq 1/2 \\ 1 & \text{si } \lambda \leq 1/2 \end{cases} \times \frac{2}{\lambda+1/2} < \infty.
\end{aligned}$$

Por lo tanto, $\mathrm{L}^\infty(-1,1) \subseteq \mathrm{L}^2_\lambda$. En particular, **todos los polinomios** pertenecen a L^2_λ.

Entonces podemos aplicar el proceso de Gram–Schmidt respecto al producto interior de L^2_λ a la sucesión de monomios $(x^n)_{n\in\mathbb{N}_0}$, respetando el orden y sin normalizar. Como aquella sucesión es linealmente independiente, obtendremos una sucesión de polinomios mutuamente L^2_λ-ortogonales $(p_n)_{n\in\mathbb{N}_0}$, donde cada p_n es de grado exactamente n.

Proposición 16

Para cada $n \in \mathbb{N}_0$, el polinomio ortogonal p_n posee n raíces simples, todas en el intervalo abierto $(-1,1)$.

Demostración: Si $n = 0$, el resultado es trivialmente cierto.

Sea $n \geq 1$. Entonces,

$$\int_{-1}^{1} p_n(x)(1-x^2)^{\lambda-1/2}\,\mathrm{d}x = \langle p_n, 1\rangle_\lambda = 0.$$

Como el integrando arriba no es idénticamente nulo, existe al menos un punto en $(-1,1)$ donde p_n cambia de signo. Denotemos por $x_1,\ldots,x_m$ a todos esos puntos. Entonces, la función $z(x) = p_n(x)(x-x_1)\cdots(x-x_m)(1-x^2)^{\lambda-1/2}$ no es idénticamente nula y no cambia de signo en $(-1,1)$. Si $m < n$,

$$\int_{-1}^{1} z(x)\,\mathrm{d}x = \int_{-1}^{1} p_n(x)(x-x_1)\cdots(x-x_m)(1-x^2)^{\lambda-1/2}\,\mathrm{d}x = 0$$

por la ortogonalidad entre p_n y los polinomios de grado m (pues éstos pueden escribirse como combinación lineal de $p_0,\ldots,p_m$) y hemos hallado una contradicción. Luego, $m \geq n$. Como $x_1,\ldots,x_m$

son raíces de p_n y éstas no pueden ser más que n, concluimos que $m = n$. □

Como los polinomios L^2_λ-ortogonales que produce el proceso de Gram–Schmidt tienen todas sus raíces en $(-1, 1)$, podemos reescalarlos para que en 1 tomen cualquier valor no nulo.

Observación 17 (La función gamma)

La función $\Gamma\colon \mathbb{C} \setminus (-\mathbb{N}_0) \to \mathbb{C}$ es analítica, no posee ceros y satisface la ecuación $\Gamma(z+1) = z\,\Gamma(z)$. Restringida a $(0, \infty)$ toma valores reales; en particular, para todo $n \in \mathbb{N}_0$, $n! = \Gamma(n+1)$. Otro valor particular es $\Gamma(1/2) = \sqrt{\pi}$. Si $\lambda > -1/2$,

$$\int_{-1}^{1} (1-x^2)^{\lambda-1/2}\,\mathrm{d}x = \frac{\sqrt{\pi}\,\Gamma(\lambda+1/2)}{\Gamma(\lambda+1)}. \tag{19}$$

Definición 18

Sea $\lambda > -1/2$, $\lambda \neq 0$. Definimos a la sucesión de **polinomios de Gegenbauer** o **polinomios ultraesféricos** $(C_n^{(\lambda)})_{n\in\mathbb{N}_0}$ como la sucesión que resulta de aplicar el proceso de Gram–Schmidt a la base de mononomios $(x \mapsto x^n)_{n\in\mathbb{N}_0}$ respecto al producto interior de L^2_λ, con la normalización

$$C_n^{(\lambda)}(1) = \frac{\Gamma(n+2\lambda)}{\Gamma(2\lambda)\,\Gamma(n+1)}.$$

Esta forma de normalizar colapsa en $\lambda = -1/2$ debido al $\Gamma(2\lambda)$ en el denominador. Pero polinomios $\mathrm{L}^2_{-1/2}$-ortogonales existen de lo más bien; con la normalización alternativa de tomar el valor 1 al evaluarse en 1 se trata de los polinomios de Chebyshev $(T_n)_{n\in\mathbb{N}_0}$, $T_n(x) = \cos(n \arccos(x))$.

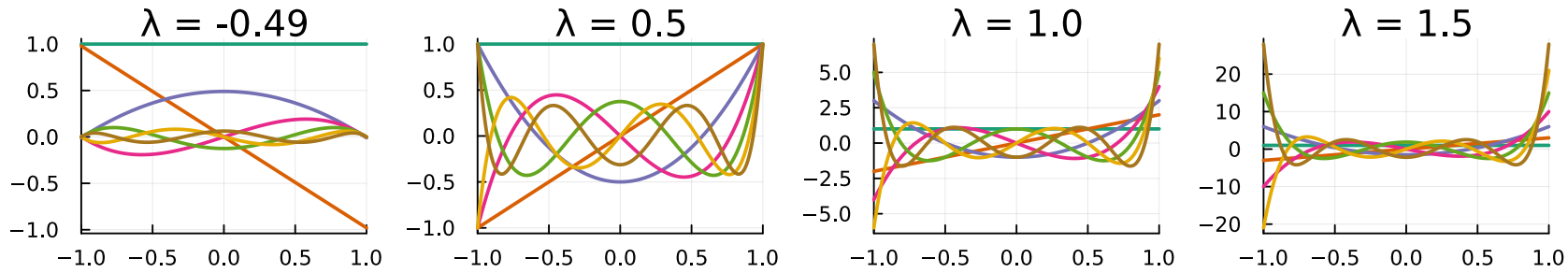


Figura 5: Gráficos de $C_n^{(\lambda)}$, $0 \leq n \leq 6$, para cuatro casos particulares del parámetro λ.

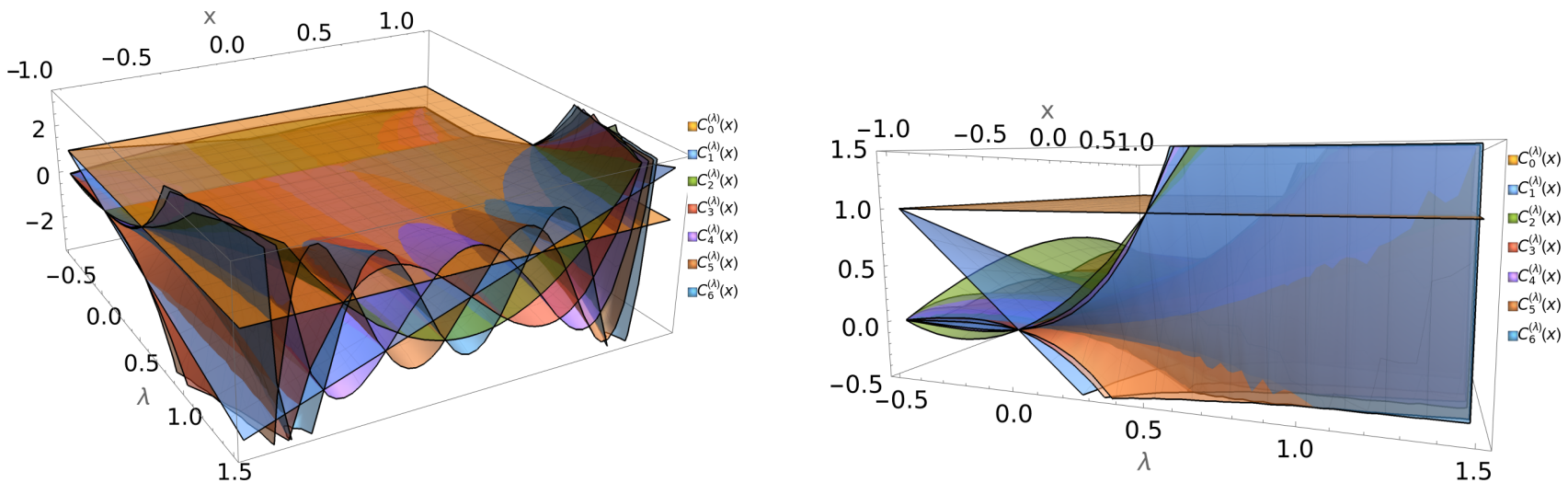


Figura 6: Gráficos de superficie de $C_n^{(\lambda)}(x)$ para $0 \leq n \leq 6$, desde distintos ángulos.

Los $C_n^{(1/2)}$ son los polinomios ortogonales respecto al producto interior

$$\int_{-1}^{1} u(x)\overline{v(x)}(1-x^2)^{1/2-1/2}\,\mathrm{d}x = \langle u, v\rangle_{\mathrm{L}^2(-1,1)},$$

normalizados de forma que $C_n^{(1/2)}(1) = \frac{\Gamma(n+2\times 1/2)}{\Gamma(2\times 1/2)\Gamma(n+1)} = 1$.
Concluimos que cada $C_n^{(1/2)}$ es el polinomio de Legendre L_n.

Los $C_n^{(1)}$ son los polinomios ortogonales respecto al producto interior

$$\int_{-1}^{1} u(x)\overline{v(x)}(1-x^2)^{1-1/2}\,\mathrm{d}x = \int_{-1}^{1} u(x)\overline{v(x)}\sqrt{1-x^2}\,\mathrm{d}x,$$

normalizados de forma que $C_n^{(1)}(1) = \frac{\Gamma(n+2\times 1)}{\Gamma(2\times 1)\Gamma(n+1)} = n+1$. Es fácil probar que los **polinomios de Chebyshev del segundo tipo** $U_n(x) := \frac{\operatorname{sen}((n+1)\arccos(x))}{\operatorname{sen}(\arccos(x))}$ hacen lo mismo. Concluimos que cada $C_n^{(1)}$ coincide con U_n.

Es fácil probar que

$$\lim_{\lambda\to 0} C_0^{(\lambda)}(1) = 1 \quad \text{y} \quad (\forall\, n \in \mathbb{N}) \ \lim_{\lambda\to 0} \frac{1}{\lambda} C_n^{(\lambda)}(1) = \frac{2}{n}.$$

Por lo tanto, es plausible esperar (y es cierta) la siguiente conexión entre límites en $\lambda = 0$ de polinomios de Gegenbauer y los polinomios de Chebyshev del primer tipo

$$\lim_{\lambda\to 0} C_0^{(\lambda)}(x) = T_0(x) \quad \text{y} \quad (\forall\, n \in \mathbb{N}) \ \lim_{\lambda\to 0} \frac{1}{\lambda} C_n^{(\lambda)}(x) = \frac{2}{n} T_n(x).$$

Para todo $\lambda > -1/2$ definimos al operador diferencial de segundo orden

$$L^{(\lambda)}(u) := -\frac{1}{w_\lambda}\left[w_{\lambda+1}u'\right]'. \tag{20}$$

Sustituyendo w_λ de acuerdo a la Definición 15, obtenemos

$$L^{(\lambda)}(u)(x) = -(1-x^2)u''(x) + (2\lambda+1)\,x\,u'(x), \tag{21}$$

lo que de paso demuestra que $L^{(\lambda)}$ preserva el grado de polinomios no constantes. Además, este operador es autoadjunto en el sentido preciso que sigue:

Proposición 19

Sea $\lambda > -1/2$. Entonces, para todo $u, v \in \mathrm{C}^2([-1,1])$,

$$\langle L^{(\lambda)}(u), v\rangle_\lambda = \langle u', v'\rangle_{\lambda+1} = \langle u, L^{(\lambda)}(v)\rangle_\lambda.$$

Demostración: Primero,

$$\begin{aligned}\langle L^{(\lambda)}(u), v\rangle_\lambda &= -\int_{-1}^{1} \frac{1}{w_\lambda}\left(w_{\lambda+1}u'\right)'\,\overline{v}\,w_\lambda = -\int_{-1}^{1}\left(w_{\lambda+1}u'\right)'\,\overline{v}\\ &\overset{\text{IPP}}{=} \int_{-1}^{1} u'\,\overline{v'}\,w_{\lambda+1} - \left[(1-x^2)^{\lambda+1/2}\,u'(x)\,\overline{v(x)}\right]\Big|_{x=-1}^{x=1}\\ &= \langle u', v'\rangle_{\lambda+1}.\end{aligned}$$

Por lo tanto,

$$\begin{aligned}\langle L^{(\lambda)}(u), v\rangle_\lambda = \langle u', v'\rangle_{\lambda+1} &= \overline{\langle v', u'\rangle_{\lambda+1}}\\ &= \overline{\langle L^{(\lambda)}(v), u\rangle_\lambda} = \langle u, L^{(\lambda)}(v)\rangle_\lambda.\end{aligned}$$

□

Ahora podemos probar que los polinomios de Gegenbauer $C_n^{(\lambda)}$ son autofunciones de $L^{(\lambda)}$.

Lema 20

Sea $\lambda > -1/2$. Entonces $C_n^{(\lambda)}$ (sustituir T_n si $\lambda = 0$) satisface el problema de Sturm–Liouville

$$L^{(\lambda)}\left(C_n^{(\lambda)}\right) = n(n+2\lambda)\, C_n^{(\lambda)}$$

y su versión débil

$$\left(\forall\, v \in \mathrm{C}^1([-1,1])\right) \quad \left\langle C_n^{(\lambda)\prime}, v' \right\rangle_{\lambda+1} = n(n+2\lambda)\left\langle C_n^{(\lambda)}, v \right\rangle_{\lambda}.$$

Demostración: Si $n = 0$ el resultado es obvio. Sea entonces $n \geq 1$. Entonces, $C_n^{(\lambda)}$ no es constante y por lo tanto $L^{(\lambda)}\left(C_n^{(\lambda)}\right)$ es de grado exactamente n, admitiendo la expansión

$$L^{(\lambda)}\left(C_n^{(\lambda)}\right) = \sum_{k=0}^{n} \alpha_k\, C_k^{(\lambda)},$$

donde ya sabemos que $\alpha_n \neq 0$. Por otro lado, para $0 \leq k \leq n-1$,

$$\alpha_k = \frac{\langle L^{(\lambda)}\big(C_n^{(\lambda)}\big), C_k^{(\lambda)}\rangle_\lambda}{\|C_k^{(\lambda)}\|_\lambda^2} \overset{\text{Prop. 19}}{=} \frac{\langle C_n^{(\lambda)}, L^{(\lambda)}\big(C_k^{(\lambda)}\big)\rangle_\lambda}{\|C_k^{(\lambda)}\|_\lambda^2} = 0,$$

ya que $L^{(\lambda)}\big(C_k^{(\lambda)}\big)$ es un polinomio de grado estrictamente menor que n. Así, la expansión anterior se reduce a $L^{(\lambda)}\big(C_n^{(\lambda)}\big) = \alpha_n C_n^{(\lambda)}$. Para determinar el coeficiente α_n faltante explotaremos que $C_n^{(\lambda)}$ puede expresarse como

$$C_n^{(\lambda)}(x) = \ell_n^{(\lambda)} x^n + \mathrm{LOT}(x),$$

donde aquí y de ahora en adelante usaremos LOT para referirnos a términos de orden inferior (*lower order terms*), los cuales pueden variar de expresión en expresión. Entonces,

$$\begin{aligned}\alpha_n \ell_n^{(\lambda)} x^n + \mathrm{LOT}(x) &= \alpha_n C_n^{(\lambda)}(x) = L^{(\lambda)}\big(C_n^{(\lambda)}\big)(x) \\ &\overset{(21)}{=} -(1-x^2)C_n^{(\lambda)\prime\prime}(x) + (2\lambda+1)xC_n^{(\lambda)\prime}(x) \\ &= -(1-x^2)\ell_n^{(\lambda)} n(n-1)x^{n-2} + (2\lambda+1)x\ell_n^{(\lambda)} n x^{n-1} + \mathrm{LOT}(x) \\ &= \ell_n^{(\lambda)}\left[n(n-1) + (2\lambda+1)n\right]x^n + \mathrm{LOT}(x),\end{aligned}$$

por lo que $\alpha_n = n(n+2\lambda)$ como se deseaba probar.

Teniendo la forma fuerte del problema de Sturm–Liouville, la forma débil es consecuencia de la Proposición 19. Que baste pedir que $v \in \mathrm{C}^1([-1,1])$ se desprende de examinar cuidadosamente la demostración de aquel resultado. □

Ahora expresaremos cada polinomio de Gegenbauer como serie de potencias en torno a $x = 1$.

Proposición 21

Sea $\lambda > -1/2$, $\lambda \neq 0$ y $n \in \mathbb{N}_0$. Entonces,

$$C_n^{(\lambda)}(x) = \sum_{\nu=0}^{n} \frac{1}{2^\nu (n-\nu)!\nu!} \frac{\Gamma(n+2\lambda+\nu)\,\Gamma(\lambda+1/2)}{\Gamma(2\lambda)\,\Gamma(\nu+\lambda+1/2)} (x-1)^\nu.$$

Demostración: Abreviemos $y = C_n^{(\lambda)}$. Por la forma fuerte del problema de Sturm–Liouville que satisface y (Lema 20) y adaptando la forma (21) del operador $L^{(\lambda)}$,

$$\begin{aligned} 0 &= -(1-x^2)y''(x) + (2\lambda+1)x\,y'(x) - n(n+2\lambda)y(x) \\ &= (x-1)\big((x-1)+2\big)y''(x) + (2\lambda+1)\big((x-1)+1\big)y'(x) \\ &\qquad - n(n+2\lambda)y(x). \end{aligned}$$

Sustituimos la expansión $y(x) = \sum_{\nu=0}^{n} a_\nu (x-1)^\nu$, obteniendo

$$
\begin{aligned}
0 = &\left((x-1)^2 + 2(x-1)\right) \sum_{\nu=0}^{n} a_\nu \nu(\nu-1)(x-1)^{\nu-2} \\
&+ \left((x-1)+1\right)(2\lambda+1) \sum_{\nu=0}^{n} a_\nu \nu (x-1)^{\nu-1} \\
&- n(n+2\lambda) \sum_{\nu=0}^{n} a_\nu (1-x)^\nu \\
= &\sum_{\nu=0}^{n} a_\nu \nu(\nu-1)(x-1)^\nu + \sum_{\nu=1}^{n} 2a_\nu \nu(\nu-1)(x-1)^{\nu-1} \\
&+ \sum_{\nu=0}^{n} (2\lambda+1) a_\nu \nu (x-1)^\nu + \sum_{\nu=1}^{n} (2\lambda+1) a_\nu \nu (x-1)^{\nu-1} \\
&- n(n+2\lambda) \sum_{\nu=0}^{n} a_\nu (1-x)^\nu .
\end{aligned}
$$

Ahora cambiamos el rango de índices de las sumas en $1 \leq \nu \leq n$ a $0 \leq \nu \leq n-1$.

$$0 = \sum_{\nu=0}^{n} a_\nu \nu(\nu-1)(x-1)^\nu + \sum_{\nu=0}^{n-1} 2a_{\nu+1}(\nu+1)\nu(x-1)^\nu$$
$$+ \sum_{\nu=0}^{n}(2\lambda+1)a_\nu \nu(x-1)^\nu + \sum_{\nu=0}^{n-1}(2\lambda+1)a_{\nu+1}(\nu+1)(x-1)^\nu$$
$$- n(n+2\lambda)\sum_{\nu=0}^{n} a_\nu(1-x)^\nu.$$

Resultó una combinación lineal de la base de polinomios $\{(x-1)^\nu\}_{\nu=0}^{n}$. Como la combinación lineal es nula, sus coeficientes son todos nulos.

Así, aislando al coeficiente que le corresponde a $(x-1)^\nu$, obtenemos, para $0 \le \nu \le n-1$,

$$\begin{aligned} 0 &= a_\nu \nu(\nu-1) + 2a_{\nu+1}(\nu+1)\nu + (2\lambda+1)a_\nu \nu \\ &\quad + (2\lambda+1)a_{\nu+1}(\nu+1) - n(n+2\lambda)a_\nu \\ &= a_\nu\left[\nu(\nu-1) + (2\lambda+1)\nu - n(n+2\lambda)\right] \\ &\quad + a_{\nu+1}\left[2(\nu+1)\nu + (2\lambda+1)(\nu+1))\right] \\ &= a_\nu\left[\nu(\nu+2\lambda) - n(n+2\lambda)\right] + a_{\nu+1}(\nu+1)(2\nu+2\lambda+1). \end{aligned}$$

Esto es, obtuvimos la relación de recurrencia

$$a_{\nu+1} = \frac{n(n+2\lambda) - \nu(\nu+2\lambda)}{(\nu+1)(2\nu+2\lambda+1)}\, a_\nu, \quad 0 \le \nu \le n-1, \tag{22a}$$

que podemos inicializar con

$$a_0 = y(1) = C_n^{(\lambda)}(1) = \frac{\Gamma(n+2\lambda)}{\Gamma(2\lambda)\,\Gamma(n+1)}. \tag{22b}$$

Probaremos por inducción que, para $0 \leq \nu \leq n$,

$$a_\nu = \frac{1}{2^\nu (n-\nu)!\nu!} \frac{\Gamma(n+2\lambda+\nu)\,\Gamma(\lambda+1/2)}{\Gamma(2\lambda)\,\Gamma(\nu+\lambda+1/2)}.$$

El caso $\nu = 0$ ya está listo en (22b). Ahora, asumiendo el caso ν, $0 \leq \nu \leq n-1$, probaremos el caso $\nu+1$.

$$\begin{aligned}
a_{n+1} &\overset{(22a)}{=} \frac{n(n+2\lambda)-\nu(\nu+2\lambda)}{(\nu+1)(2\nu+2\lambda+1)}\, a_\nu \\
&\overset{\text{HDI}}{=} \frac{n(n+2\lambda)-\nu(\nu+2\lambda)}{(\nu+1)(2\nu+2\lambda+1)} \frac{1}{2^\nu (n-\nu)!\nu!} \frac{\Gamma(n+2\lambda+\nu)\,\Gamma(\lambda+1/2)}{\Gamma(2\lambda)\,\Gamma(\nu+\lambda+1/2)} \\
&= \frac{[n(n+2\lambda)-\nu(\nu+2\lambda)]\,\Gamma(n+2\lambda+\nu)\,\Gamma(\lambda+1/2)}{(\nu+\lambda+1/2)\,2^{\nu+1}(n-\nu)!(\nu+1)!\,\Gamma(2\lambda)\,\Gamma(\nu+\lambda+1/2)} \\
&= \frac{[n^2+2\lambda n+n\nu-n\nu-\nu^2-2\lambda\nu]\,\Gamma(n+2\lambda+\nu)\,\Gamma(\lambda+1/2)}{2^{\nu+1}(n-\nu)!(\nu+1)!\,\Gamma(2\lambda)\,\Gamma((\nu+1)+\lambda+1/2)} \\
&= \frac{(n+2\lambda+\nu)(n-\nu)\,\Gamma(n+2\lambda+\nu)\,\Gamma(\lambda+1/2)}{2^{\nu+1}(n-\nu)!(\nu+1)!\,\Gamma(2\lambda)\,\Gamma((\nu+1)+\lambda+1/2)}.
\end{aligned}$$

Así,

$$a_{n+1} = \frac{(n+2\lambda+\nu)(n-\nu)\,\Gamma(n+2\lambda+\nu)\,\Gamma(\lambda+1/2)}{2^{\nu+1}(n-\nu)!(\nu+1)!\,\Gamma(2\lambda)\,\Gamma((\nu+1)+\lambda+1/2)}$$
$$= \frac{\Gamma(n+2\lambda+(\nu+1))\,\Gamma(\lambda+1/2)}{2^{\nu+1}(n-(\nu+1))!(\nu+1)!\,\Gamma(2\lambda)\,\Gamma((\nu+1)+\lambda+1/2)},$$

lo que es, finalmente, lo que queríamos demostrar. □

De acuerdo a la Proposición 21, para todo $\lambda > -1/2$ con $\lambda \neq 0$ y para todo $n \in \mathbb{N}_0$,

$$C_n^{(\lambda)}(x) = \ell_n^{(\lambda)}(x-1)^n + \mathrm{LOT}(x) = \ell_n^{(\lambda)} x^n + \mathrm{LOT}(x), \qquad (23a)$$

donde

$$\ell_n^{(\lambda)} = \frac{1}{2^n\, n!}\,\frac{\Gamma(2n+2\lambda)\,\Gamma(\lambda+1/2)}{\Gamma(2\lambda)\,\Gamma(n+\lambda+1/2)}. \qquad (23b)$$

Después de este festival de funciones gamma, podemos volver a obtener resultados elegantes.

Proposición 22 (Relación de paridad)

Sea $\lambda > -1/2$ y $n \in \mathbb{N}_0$. Entonces, con T_n en lugar de $C_n^{(\lambda)}$ si $\lambda = 0$,

$$C_n^{(\lambda)}(-x) = (-1)^n C_n^{(\lambda)}(x).$$

Demostración: Si $n = 0$, esto es obvio. Sea $n \geq 1$ entonces. Como para todo $x \in (-1, 1)$, $w_\lambda(-x) = w_\lambda(x)$,

$$(\forall\, q \in \mathbb{P}_{n-1}) \quad \langle C_n^{(\lambda)}(-x), q\rangle_\lambda = \langle C_n^{(\lambda)}, q(-x)\rangle_\lambda = 0.$$

Como $C_n^{(\lambda)} \in \mathbb{P}_n$ es ortogonal a $\mathbb{P}_{n-1}$, necesariamente existe una constante $\kappa_n \neq 0$ tal que $C_n^{(\lambda)}(-x) = \kappa_n C_n^{(\lambda)}(x)$. Luego,

$$\ell_n^{(\lambda)}(-1)^n x^n + \mathrm{LOT}(x) = \kappa_n\, \ell_n^{(\lambda)} x^n + \mathrm{LOT}(x),$$

lo que deja de manifiesto que $\kappa_n = (-1)^n$. □

Lema 23 (Recurrencia de tres términos)

Sea $\lambda > -1/2$, $\lambda \neq 0$. Entonces, $C_0^{(\lambda)}(x) = 1$, $C_1^{(\lambda)}(x) = 2\lambda x$ y, para todo $n \geq 1$,

$$2(n+\lambda)\, x\, C_n^{(\lambda)}(x) = (n+1) C_{n+1}^{(\lambda)}(x) + (n+2\lambda-1) C_{n-1}^{(\lambda)}(x).$$

Demostración: Las fórmulas para $C_0^{(\lambda)}$ y $C_1^{(\lambda)}$ salen directo de su valor en $x = 1$ (Definición 18) y, en el caso del segundo, de su condición de polinomio impar (Proposición 22).

Sea $n \geq 1$. Entonces, el polinomio $x\, C_n^{(\lambda)}(x) \in \mathbb{P}_{n+1}$ admite una expansión de la forma $\sum_{k=0}^{n+1} \alpha_k\, C_k^{(\lambda)}$. Si $k \leq n-2$, $\alpha_k = 0$ porque

$$\langle x\, C_n^{(\lambda)}(x), C_k^{(\lambda)} \rangle_\lambda = \langle C_n^{(\lambda)}, \overbrace{x\, C_k^{(\lambda)}(x)}^{\in \mathbb{P}_{n-1}} \rangle_\lambda = 0.$$

Además, $\alpha_n = 0$ porque $x\, C_n^{(\lambda)}$ y $C_n^{(\lambda)}$ son de paridades opuestas.

Se sigue que

$$x\, C_n^{(\lambda)}(x) = a_{n+1}\, C_{n+1}^{(\lambda)} + a_{n-1}\, C_{n-1}^{(\lambda)}. \tag{24}$$

Evaluando ambos lados de (24) en $x = 1$ obtenemos (Definición 18)

$$\frac{\Gamma(n+2\lambda)}{\Gamma(2\lambda)\,\Gamma(n+1)} = \frac{\Gamma(n+1+2\lambda)}{\Gamma(2\lambda)\,\Gamma(n+2)}\, a_{n+1} + \frac{\Gamma(n-1+2\lambda)}{\Gamma(2\lambda)\,\Gamma(n)}\, a_{n-1}.$$

Extrayendo el coeficiente de x^{n+1} en la expansión respecto a la base de monomios $(x^k)_{k=0}^{n+1}$ de cada lado de (24) con la ayuda de (23),

$$\frac{1}{2^n\, n!}\frac{\Gamma(2n+2\lambda)\,\Gamma(\lambda+1/2)}{\Gamma(2\lambda)\,\Gamma(n+\lambda+1/2)} = \frac{1}{2^{n+1}\,(n+1)!}\frac{\Gamma(2n+2+2\lambda)\,\Gamma(\lambda+1/2)}{\Gamma(2\lambda)\,\Gamma(n+1+\lambda+1/2)}\, a_{n+1}.$$

De estas dos últimas ecuaciones podemos despejar $a_{n+1} = \frac{n+1}{2(n+\lambda)}$ y $a_{n-1} = \frac{n-1+2\lambda}{2(n+\lambda)}$. Sustituyendo en (24) y reorganizando, obtenemos el resultado deseado.

Observación 24 (Recurrencia de tres términos para polinomios de Chebyshev)

Como ya vimos en el capítulo de ejemplos, por argumentos trigonométricos es fácil probar que $T_0(x) = 1$, $T_1(x) = x$ y

$$(\forall\, n \geq 1) \quad 2\,x\,T_n(x) = T_{n+1}(x) + T_{n-1}(x).$$

Lema 25 (Diferenciación con *shifts*)

Sea $\lambda > -1/2$, $\lambda \neq 0$. Entonces,

$$(\forall\, n \in \mathbb{N}) \quad C_n^{(\lambda)'} = 2\lambda\, C_{n-1}^{(\lambda+1)}. \tag{25}$$

También

$$(\forall\, n \in \mathbb{N}) \quad T_n' = n\, C_{n-1}^{(1)} = n\, U_{n-1}. \tag{26}$$

Demostración: Sea $\lambda > -1/2$, $\lambda \neq 0$. Si $n = 1$, (25) sale directo de las expresiones que tenemos para $C_1^{(\lambda)}$ y $C_0^{(\lambda+1)}$ (Lema 23). Sea ahora $n \geq 2$. Sea $q \in \mathbb{P}_{n-2}$, arbitrario y sea $Q \in \mathbb{P}_{n-1}$ cualquier antiderivada de q. Entonces,

$$\langle C_n^{(\lambda)'}, q\rangle_{\lambda+1} = \langle C_n^{(\lambda)'}, Q'\rangle_{\lambda+1} \overset{\text{P. 19}}{=} \langle C_n^{(\lambda)}, \overbrace{L^{(\lambda)}(Q)}^{\in\mathbb{P}_{n-1}}\rangle_{\lambda} = 0.$$

Así, $C_n^{(\lambda)'}$ es un polinomio de grado $n-1$ que es $\mathsf{L}^2_{\lambda+1}$-ortogonal a todos los de grado menor. Luego, existe alguna constante μ_n tal que $C_n^{(\lambda)'} = \mu_n C_{n-1}^{(\lambda+1)}$. Por (23),

$$\ell_n^{(\lambda)}\, n\, x^{n-1} + \mathsf{LOT}(x) = \mu_n\, \ell_{n-1}^{(\lambda+1)} x^{n-1} + \mathsf{LOT}(x)$$

y

$$\mu_n = \frac{n\,\ell_n^{(\lambda)}}{\ell_{n-1}^{(\lambda+1)}} = \frac{n\,\frac{1}{2^n\,n!}\,\frac{\Gamma(2n+2\lambda)\,\Gamma(\lambda+1/2)}{\Gamma(2\lambda)\,\Gamma(n+\lambda+1/2)}}{\frac{1}{2^{n-1}\,(n-1)!}\,\frac{\Gamma(2n+2\lambda)\,\Gamma((\lambda+1)+1/2)}{\Gamma(2\lambda+2)\,\Gamma(n+\lambda+1/2)}} = 2\lambda,$$

lo que termina de probar (25).

La ecuación (26) se prueba directamente por propiedades de funciones trigonométricas. □

El Lema 25 expresa que la diferenciación es una operación diagonal entre los polinomios de Gegenbauer de parámetro λ y los de parámetro $\lambda + 1$. La Proposición 26 que sigue expresa que el operador identidad es similarmente bidiagonal.

Adoptamos de ahora en adelante la convención de que, si $n < 0$, entonces $C_n^{(\lambda)} = T_n = 0$ y $\mathbb{P}_n = \{0\}$.

Proposición 26 (Operador identidad con *shift*)

Sea $\lambda > -1/2$, $\lambda \neq 0$. Entonces,

$$(\forall\, n \in \mathbb{N}_0) \quad C_n^{(\lambda)} = \frac{\lambda}{n+\lambda}\left(C_n^{(\lambda+1)} - C_{n-2}^{(\lambda+1)}\right). \tag{27}$$

También

$$\begin{cases} T_0 = C_0^{(1)} = U_0, \\ (\forall\, n \in \mathbb{N}) \quad T_n = \dfrac{1}{2}\left(C_n^{(1)} - C_{n-2}^{(1)}\right) = \dfrac{1}{2}\left(U_n - U_{n-2}\right). \end{cases} \tag{28}$$

Demostración: Sea $\lambda > -1/2$, $\lambda \neq 0$. Los casos $n = 0$ y $n = 1$ de (27) salen directo de las fórmulas explícitas que tenemos para los polinomios de Gegenbauer de esos grados (Lema 23). Dado $n \geq 2$, tenemos la expansión $C_n^{(\lambda)} = \sum_{k=0}^{n} \gamma_k C_k^{(\lambda+1)}$.

Si $k \leq n-3$, $\gamma_k = 0$ debido a que $\overbrace{\quad\quad}^{\in \mathbb{P}_{n-1}}$

$$\langle C_n^{(\lambda)}, C_k^{(\lambda+1)} \rangle_{\lambda+1} = \langle C_n^{(\lambda)}, \overbrace{(1-x^2) C_k^{(\lambda+1)}}^{\quad} \rangle_{\lambda} = 0.$$

También $\gamma_{n-1} = 0$ porque $C_n^{(\lambda)}$ y $C_{n-1}^{(\lambda+1)}$ son de paridades opuestas (Proposición 22). Se sigue que

$$C_n^{(\lambda)} = \gamma_n C_n^{(\lambda+1)} + \gamma_{n-2} C_{n-2}^{(\lambda+1)}. \tag{29}$$

Entonces, $\ell_n^{(\lambda)} x^n + \mathrm{LOT}(x) = \gamma_n \, \ell_n^{(\lambda+1)} x^n + \mathrm{LOT}(x)$, por lo que

$$\gamma_n = \frac{\ell_n^{(\lambda)}}{\ell_n^{(\lambda+1)}} \overset{(23)}{=} \frac{\frac{1}{2^n\, n!} \frac{\Gamma(2n+2\lambda)\,\Gamma(\lambda+1/2)}{\Gamma(2\lambda)\,\Gamma(n+\lambda+1/2)}}{\frac{1}{2^n\, n!} \frac{\Gamma(2n+2(\lambda+1))\,\Gamma((\lambda+1)+1/2)}{\Gamma(2(\lambda+1))\,\Gamma(n+(\lambda+1)+1/2)}} = \frac{\lambda}{n+\lambda}.$$

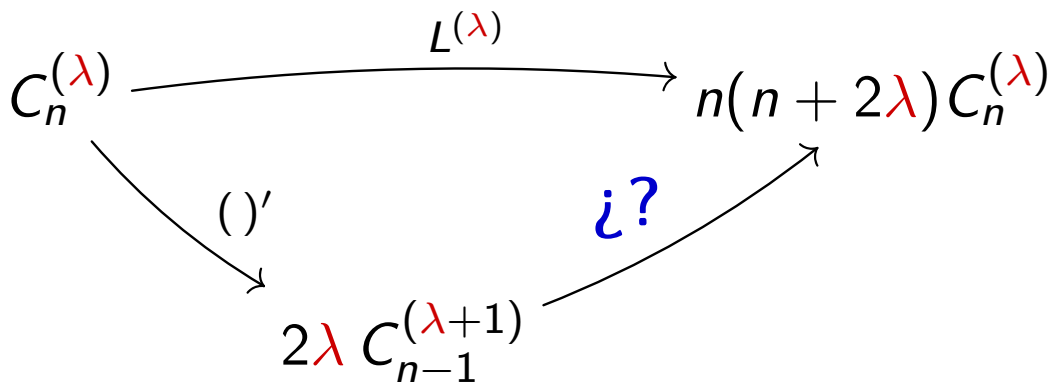
Sustituyendo el valor obtenido de γ_n en (29) y evaluando en $x = 1$ con el valor especificado en la Definición 18,

$$\frac{\Gamma(n+2\lambda)}{\Gamma(2\lambda)\, n!} = \frac{\lambda}{n+\lambda} \frac{\Gamma(n+2\lambda+2)}{\Gamma(2\lambda+2)\, n!} + \gamma_{n-2} \frac{\Gamma(n+2\lambda)}{\Gamma(2\lambda+2)\,(n-2)!},$$

de donde $\gamma_{n-2} = -\frac{\lambda}{n+\lambda}$, lo que termina de probar (27).

Las ecuaciones en (28) se prueban directamente por propiedades de funciones trigonométricas. □

Consideremos el siguiente esquema inspirado por el Lema 20 y el Lema 25:



Este esquema sugiere la existencia de un operador, dependiente de λ, que transforma a $C_{n-1}^{(\lambda+1)}$ en un múltiplo escalar de $C_n^{(\lambda)}$. Como $L^{(\lambda)}(u) = -\frac{1}{w_\lambda}\left[w_{\lambda+1}u'\right]'$ (cf. (20)), el candidato natural es

$$d^{(\lambda,\star)}(u) := -\frac{1}{w_\lambda}\left[w_{\lambda+1}u\right]'. \tag{30}$$

Proposición 27

Sea $\lambda > -1/2$, $\lambda \neq 0$. Entonces,

$$(\forall\, n \in \mathbb{N}) \quad d^{(\lambda,\star)}\left(C_{n-1}^{(\lambda+1)}\right) = \frac{n(n+2\lambda)}{2\lambda}\, C_n^{(\lambda)}.$$

Demostración:

$$d^{(\lambda,\star)}\left(C_{n-1}^{(\lambda+1)}\right) \overset{\text{L. 25}}{=} d^{(\lambda,\star)}\left(\frac{1}{2\lambda}C_n^{(\lambda)\prime}\right)$$

$$\overset{(20),(30)}{=} \frac{1}{2\lambda}L^{(\lambda)}(C_n^{(\lambda)}) \overset{\text{L. 20}}{=} \frac{n(n+2\lambda)}{2\lambda}C_n^{(\lambda)}.$$

□

Como consecuencia obtenemos otra fórmula útil para expresar los polinomios de Gegenbauer.

Proposición 28 (Fórmula de Rodrigues)

Sea $\lambda > -1/2$, $\lambda \neq 0$. Entonces,

$$(\forall\, n \in \mathbb{N}_0) \quad C_n^{(\lambda)} = \frac{(-1)^n\, 2^n}{n!} \frac{\Gamma(2\lambda+n)\,\Gamma(\lambda+n)}{\Gamma(2\lambda+2n)\,\Gamma(\lambda)} \frac{1}{w_\lambda} \frac{\mathrm{d}^n}{\mathrm{d}x^n}(w_{\lambda+n})\,.$$

Demostración: El caso $n = 0$ es directo. Sea $n \geq 1$. Entonces,

$$\begin{aligned}
&(-1)^n \frac{1}{w_\lambda} \frac{\mathrm{d}^n}{\mathrm{d}x^n}(w_{\lambda+n}) \\
&\overset{(30)}{=} d^{(\lambda,\star)} \circ d^{(\lambda+1,\star)} \circ \cdots \circ d^{(\lambda+n-2,\star)} \circ d^{(\lambda+n-1,\star)} \left(C_0^{(\lambda+n)} \right) \\
&\overset{\text{P. }27}{=} \frac{n(n+2\lambda)}{2\lambda} \times \frac{(n-1)(n-1+2(\lambda+1))}{2(\lambda+1)} \times \cdots \\
&\qquad \times \frac{2(2+2(\lambda+n-2))}{2(\lambda+n-2)} \times \frac{1(1+2(\lambda+n-1))}{2(\lambda+n-1)}\, C_n^{(\lambda)}.
\end{aligned}$$

Así, $\frac{(-1)^n}{w_\lambda}\frac{\mathrm{d}^n}{\mathrm{d}x^n}(w_{\lambda+n}) = a_n^{(\lambda)} C_n^{(\lambda)}$, donde

$$a_n^{(\lambda)} = \prod_{k=1}^{n} \frac{k(k+2(\lambda+n-k))}{2(\lambda+n-k)} \overset{\text{inducción}}{=} \frac{n!}{2^n}\frac{\Gamma(2n+2\lambda)\,\Gamma(\lambda)}{\Gamma(2\lambda+n)\,\Gamma(\lambda+n)}.$$

Reorganizando se obtiene el resultado deseado. □

Ahora podremos computar la norma al cuadrado de un polinomio de Gegenbauer. Necesitaremos agregar la siguiente **fórmula de duplicación de Legendre** a nuestra corta lista de propiedades de la función gamma:

$$(\forall\, z \in \mathbb{C} \setminus (-\mathbb{N}_0/2)) \quad \Gamma(z)\,\Gamma\left(z+\tfrac{1}{2}\right) = 2^{1-2z}\sqrt{\pi}\,\Gamma(2z). \tag{31}$$

Proposición 29

Sea $\lambda > -1/2$, $\lambda \neq 0$. Entonces,

$$(\forall\, n \in \mathbb{N}_0) \quad h_n^{(\lambda)} := \| C_n^{(\lambda)} \|_\lambda^2 = 2^{1-2\lambda}\, \frac{\pi}{n!}\, \frac{\Gamma(2\lambda + n)}{\Gamma(\lambda)^2\, (n + \lambda)}. \tag{32}$$

También

$$\begin{cases} \| T_0 \|_0^2 = \pi, \\ (\forall\, n \in \mathbb{N}) \quad \| T_n \|_0^2 = \dfrac{\pi}{2}. \end{cases} \tag{33}$$

Demostración: Las ecuaciones en (33) se prueban directamente por propiedades de funciones trigonométricas.

Sean $\lambda > -1/2$, $\lambda \neq 0$ y $n \in \mathbb{N}_0$. Como $C_n^{(\lambda)}(x) - \ell_n^{(\lambda)} x^n \in \mathbb{P}_{n-1}$,

$$\langle C_n^{(\lambda)},\, C_n^{(\lambda)}(x) - \ell_n^{(\lambda)} x^n \rangle_\lambda = 0.$$

Por lo tanto, abreviando como $\zeta_n^{(\lambda)}$ a aquella constante que hace que $C_n^{(\lambda)} = \zeta_n^{(\lambda)} \frac{(-1)^n}{w_\lambda} \frac{\mathrm{d}^n}{\mathrm{d}x^n}(w_{\lambda+n})$ (cf. Proposición 28),

$$\begin{aligned}
\|C_n^{(\lambda)}\|_\lambda^2 &= \langle C_n^{(\lambda)}, \ell_n^{(\lambda)} x^n \rangle_\lambda = \ell_n^{(\lambda)} \int_{-1}^{1} C_n^{(\lambda)}(x)\, x^n\, w_\lambda(x)\, \mathrm{d}x \\
&= \ell_n^{(\lambda)} \zeta_n^{(\lambda)} \int_{-1}^{1} \frac{(-1)^n}{w_\lambda(x)} \frac{\mathrm{d}^n}{\mathrm{d}x^n}(w_{\lambda+n}(x))\; x^n\, w_\lambda(x)\, \mathrm{d}x \\
&= \ell_n^{(\lambda)} \zeta_n^{(\lambda)} \int_{-1}^{1} (-1)^n \frac{\mathrm{d}^n}{\mathrm{d}x^n}(w_{\lambda+n}(x))\; x^n\, \mathrm{d}x \\
&\overset{\text{IPP}}{=} \ell_n^{(\lambda)} \zeta_n^{(\lambda)} \int_{-1}^{1} w_{\lambda+n}(x)\, n!\, \mathrm{d}x \\
&\overset{(19)}{=} \ell_n^{(\lambda)} \zeta_n^{(\lambda)}\, n! \frac{\sqrt{\pi}\,\Gamma(\lambda+n+1/2)}{\Gamma(\lambda+n+1)}.
\end{aligned}$$

Sustituyendo los valores conocidos de $\ell_n^{(\lambda)}$ (23) y $\zeta_n^{(\lambda)}$ y empleando las propiedades que conocemos de la función gamma —incluyendo la fórmula de duplicación de Legendre (31)—, se obtiene el resultado deseado. □

Habiendo aprendido a operar con ciertas bases de polinomios surge la pregunta de si podemos emplearlos para aproximar funciones.

Lema 30

Sea $\lambda > -1/2$. Si $\lambda \neq 0$, entonces $\left(C_n^{(\lambda)} / \sqrt{h_n^{(\lambda)}} \right)_{n \in \mathbb{N}_0}$ es una base hilbertiana de L^2_λ y $\{1/\sqrt{\pi}\} \cup (\sqrt{2/\pi}\, T_n)_{n \in \mathbb{N}}$ es una base hilbertiana de L^2_0.

Demostración: No es difícil. □

Corolario 31 (Serie de Fourier–Gegenbauer)

Sea $\lambda > -1/2$, $\lambda \neq 0$. Entonces, para todo $u \in \mathrm{L}^2_\lambda$,

$$u = \sum_{k=0}^{\infty} \hat{u}_k \, C_k^{(\lambda)} \qquad \text{y} \qquad \|u\|_\lambda^2 = \sum_{k=0}^{\infty} |\hat{u}_k|^2 \, h_k^{(\lambda)},$$

donde

$$(\forall\, k \in \mathbb{N}_0) \quad \hat{u}_k = \frac{1}{h_k^{(\lambda)}} \langle u, C_k^{(\lambda)} \rangle_\lambda .$$

Demostración: Directo del lema 30 y el teorema 2. □

Corolario 32 (Serie de Fourier–Chebyshev)

Para todo $u \in \mathrm{L}^2_0$,

$$u = \sum_{k=0}^{\infty} \hat{u}_k \, T_k \qquad \text{y} \qquad \|u\|_0^2 = \pi \left(|\hat{u}_0|^2 + \frac{1}{2} \sum_{k=1}^{\infty} |\hat{u}_k|^2 \right),$$

donde

$$\hat{u}_0 = \frac{1}{\pi} \langle u, T_0 \rangle_0 \qquad \text{y} \qquad (\forall\, k \in \mathbb{N}) \quad \hat{u}_k = \frac{2}{\pi} \langle u, T_k \rangle_0 .$$

Demostración: Directo del lema 30 y el teorema 2. □

Cuando sea necesario distinguir entre bases de expansión, le pondremos un superíndice a los coeficientes: $\hat{u}_k^{(\lambda)}$.

Lema 33 (Fórmula de diferenciación espectral)

Sea $u \in \mathrm{C}^1([-1,1])$. Si $\lambda > -1/2$, $\lambda \neq 0$, entonces

$$(\forall\, k \in \mathbb{N}_0) \quad \widehat{(u')}_k^{(\lambda+1)} = 2\lambda\, \hat{u}_{k+1}^{(\lambda)}. \tag{34}$$

En el caso $\lambda = 0$,

$$(\forall\, k \in \mathbb{N}_0) \quad \widehat{(u')}_k^{(1)} = (k+1)\, \hat{u}_{k+1}^{(0)}. \tag{35}$$

Demostración: La fórmula (34) sale de

$$\widehat{(u')}_k^{(\lambda+1)} = \frac{\langle u', C_k^{(\lambda+1)} \rangle_{\lambda+1}}{\langle C_k^{(\lambda+1)}, C_k^{(\lambda+1)} \rangle_{\lambda+1}} \overset{\text{L. 25}}{=} \frac{\frac{1}{2\lambda} \left\langle u', C_{k+1}^{(\lambda)\,\prime} \right\rangle_{\lambda+1}}{\frac{1}{(2\lambda)^2} \left\langle C_{k+1}^{(\lambda)\,\prime}, C_{k+1}^{(\lambda)\,\prime} \right\rangle_{\lambda+1}}$$

$$\overset{\text{L. 20}}{=} 2\lambda \frac{(k+1)(k+1+2\lambda)\, \langle u, C_{k+1}^{(\lambda)} \rangle_\lambda}{(k+1)(k+1+2\lambda)\, \langle C_{k+1}^{(\lambda)}, C_{k+1}^{(\lambda)} \rangle_\lambda} = 2\lambda\, \hat{u}_{k+1}^{(\lambda)}.$$

La fórmula (35) se obtiene de forma completamente análoga. □

La hipótesis $u \in \mathrm{C}^1([-1,1])$ del Lema 33 se puede relajar a que u pertenezca al espacio de tipo Sobolev $\{v \in \mathrm{L}^2_\lambda \mid v' \in \mathrm{L}^2_{\lambda+1}\}$.

Ahora, si $\lambda > -1/2$, $\lambda \neq 0$, $u \in \mathrm{C}^1([-1,1])$ y $k \in \mathbb{N}_0$,

$$\overbrace{\left|\widehat{(u')}_k^{(\lambda+1)}\right|^2 h_k^{(\lambda+1)}}^{\text{Término de } \|u'\|^2_{\lambda+1}} \overset{\text{L. 33}}{=} (2\lambda)^2 \left|\hat{u}_{k+1}^{(\lambda)}\right|^2 \langle C_k^{(\lambda+1)}, C_k^{(\lambda+1)}\rangle_{\lambda+1}$$

$$= \left|\hat{u}_{k+1}^{(\lambda)}\right|^2 \left\langle C_{k+1}^{(\lambda)\,\prime}, C_{k+1}^{(\lambda)\,\prime}\right\rangle_{\lambda+1}$$

$$\overset{\text{L. 20}}{=} (k+1)(k+1+2\lambda)\left|\hat{u}_{k+1}^{(\lambda)}\right|^2 \langle C_{k+1}^{(\lambda)}, C_{k+1}^{(\lambda)}\rangle_{\lambda}$$

$$= (k+1)(k+1+2\lambda) \underbrace{\left|\hat{u}_{k+1}^{(\lambda)}\right|^2 h_{k+1}^{(\lambda)}}_{\text{Término de } \|u\|^2_\lambda}. \tag{36}$$

La relaciones (34) y (35) entre los coeficientes de expansión de una función y los de su derivada son tremendamente útiles para efectos computacionales. Pero, para cuantificar cuánto poder de aproximación se gana con cada derivada, la relación más directamente importante es (36), como lo manifiesta la demostración de la Proposición 34 más abajo.

Sin embargo, antes de plantear dicho resultado, necesitamos introducir al **proyector ortogonal** $P_N^{(\lambda)} : \mathrm{L}^2_\lambda \to \mathbb{P}_N$, $N \in \mathbb{N}_0$. Este proyector ortogonal admite la fórmula (¿por qué?)

$$(\forall\, u \in \mathrm{L}^2_\lambda) \quad P_N^{(\lambda)}(u) = \sum_{k=0}^{N} \hat{u}_k^{(\lambda)} \begin{cases} C_k^{(\lambda)} & \text{si } \lambda \neq 0, \\ T_k & \text{si } \lambda = 0. \end{cases} \tag{37}$$

Esto es, $P_N^{(\lambda)}$ se puede caracterizar como el **operador-truncación** de la serie de Fourier–Gegenbauer/Chebyshev hasta el término de grado N.

Proposición 34 (Tasa de convergencia de serie de Fourier–*)

Sea $\lambda > -1/2$. Entonces, si $u \in \mathrm{C}^1([-1,1])$, para todo $N \in \mathbb{N}_0$,

$$\left\| u - P_N^{(\lambda)}(u) \right\|_\lambda \leq [(N+1)(N+1+2\lambda)]^{-1/2} \left\| u' \right\|_{\lambda+1}. \qquad (38)$$

En general, si $u \in \mathrm{C}^m([-1,1])$, existe una constante $C > 0$ que depende solamente de m y λ tal que, para todo $N \geq \text{máx}(1, m-1)$,

$$\left\| u - P_N^{(\lambda)}(u) \right\|_\lambda \leq C\, N^{-m} \left\| \frac{\mathrm{d}^m u}{\mathrm{d}x^m} \right\|_{\lambda+m}. \qquad (39)$$

Demostración: Supongamos que $\lambda \neq 0$. Entonces,

$$\begin{aligned}\left\|u'\right\|_{\lambda+1}^2 &\overset{\text{C. 31}}{=} \sum_{k=0}^{\infty} \left|\widehat{(u')}_k^{(\lambda+1)}\right|^2 h_k^{(\lambda+1)} \\ &\overset{(36)}{=} \sum_{k=0}^{\infty} (k+1)(k+1+2\lambda) \left|\hat{u}_{k+1}^{(\lambda)}\right|^2 h_{k+1}^{(\lambda)} \qquad (40) \\ &= \sum_{k=0}^{\infty} k(k+2\lambda) \left|\hat{u}_k^{(\lambda)}\right|^2 h_k^{(\lambda)}.\end{aligned}$$

Por otro lado, por el Corolario 31 y (37), $u - P_N^{(\lambda)}(u)$ admite la serie de Fourier–Gegenbauer

$$u - P_N^{(\lambda)}(u) = \sum_{k=N+1}^{\infty} \hat{u}_k^{(\lambda)}\, C_k^{(\lambda)}.$$

Por lo tanto,

$$\left\| u - P_N^{(\lambda)}(u) \right\|_\lambda^2 \overset{\text{Parseval}}{=} \sum_{k=N+1}^{\infty} \left| \hat{u}_k^{(\lambda)} \right|^2 h_k^{(\lambda)}$$

$$= \sum_{k=N+1}^{\infty} \frac{k(k+2\lambda)}{k(k+2\lambda)} \left| \hat{u}_k^{(\lambda)} \right|^2 h_k^{(\lambda)}$$

$$\leq \sup_{k \geq N+1} \frac{1}{k(k+2\lambda)} \sum_{k=N+1}^{\infty} k(k+2\lambda) \left| \hat{u}_k^{(\lambda)} \right|^2 h_k^{(\lambda)}$$

$$\overset{(40)}{\leq} \frac{1}{(N+1)(N+1+2\lambda)} \left\| u' \right\|_{\lambda+1}^2,$$

lo que, tomando raíces cuadradas, produce (38) en el caso $\lambda \neq 0$.
Iterando este resultado con cuidado se obtiene (39) en el caso $\lambda \neq 0$.
El caso $\lambda = 0$ es análogo, empleando un correspondiente análogo de (36). □

Definición 35 (Cuadratura de Gauss–Gegenbauer)

Sea $\lambda > -1/2$, $\lambda \neq 0$. Dado $N \in \mathbb{N}_0$, definimos los nodos $(x_j)_{j=0}^N$ y los pesos $(w_j)_{j=0}^N$ de la **regla de cuadratura de Gauss–Gegenbauer** de $N+1$ nodos como las raíces de $C_{N+1}^{(\lambda)}$ y las soluciones del sistema lineal de ecuaciones

$$(\forall\, k \in \{0, \ldots, N\}) \quad \sum_{j=0}^{N} x_j^k \, w_j = \int_{-1}^{1} x^k \, w_\lambda(x) \, \mathrm{d}x,$$

respectivamente.

Por la Proposición 16 ya sabemos que estos nodos efectivamente son $N+1$ en número y pertenecen todos al intervalo abierto $(-1, 1)$. El sistema que define a los pesos posee solución única porque la matriz asociada es una matriz de Vandermonde.

Definición 36 (Cuadratura de Gauss–Lobatto–Gegenbauer)

Sea $\lambda > -1/2$, $\lambda \neq 0$. Dado $N \in \mathbb{N}$, definimos los nodos $(x_j)_{j=0}^N$ y los pesos $(w_j)_{j=0}^N$ de la **regla de cuadratura de Gauss–Lobatto–Gegenbauer** de $N+1$ nodos como la unión de $\{-1, 1\}$ con las raíces de $C_N^{(\lambda)'}$ y las soluciones del sistema lineal de ecuaciones

$$(\forall\, k \in \{0, \ldots, N\}) \quad \sum_{j=0}^{N} x_j^k\, w_j = \int_{-1}^{1} x^k\, w_\lambda(x)\, \mathrm{d}x,$$

respectivamente.

Por la Proposición 16 y el Lema 25, sabemos que las raíces de $C_N^{(\lambda)'}$ son $N-1$ en número y pertenecen todas al intervalo abierto $(-1, 1)$. De nuevo el sistema que define a los pesos posee solución única.

Definición 37 (Cuadratura de Gauss–Chebyshev)

Dado $N \in \mathbb{N}_0$, definimos los nodos $(x_j)_{j=0}^N$ y los pesos $(w_j)_{j=0}^N$ de la **regla de cuadratura de Gauss–Chebyshev** de $N+1$ nodos como las raíces de T_{N+1} y las soluciones del sistema lineal de ecuaciones

$$(\forall\, k \in \{0, \ldots, N\}) \quad \sum_{j=0}^{N} x_j^k\, w_j = \int_{-1}^{1} x^k\, w_0(x)\, \mathrm{d}x,$$

respectivamente.

Mediante fórmulas trigonométricas y explotando la relación de ortogonalidad discreta de la Proposición 8, podemos probar las fórmulas explícitas

$$(\forall\, j \in \{0, \ldots, N\}) \qquad x_j = \cos\left(\frac{2j+1}{2(N+1)}\,\pi\right), \quad w_j = \frac{\pi}{N+1}.$$

Comprobemos estas fórmulas para los nodos y pesos de la cuadratura de Gauss–Chebyshev. Primero, claramente $T_{N+1}(x_j) = \cos\left(\frac{2j+1}{2}\pi\right) = 0$ para $0 \leq j \leq N$.

Segundo, por linealidad de la cuadratura y del funcional integral ponderada $f \mapsto \int_{-1}^{1} f(x) w_0(x)\,\mathrm{d}x$, da lo mismo comprobar la exactitud de la cuadratura en la base de los monomios, como reza la definición, o en la base de los polinomios de Chebyshev. Démonos un $k \in \{0, \ldots, N\}$ y computemos

$$\sum_{j=0}^{N} T_k(x_j)\, w_j = \frac{\pi}{N+1} \sum_{j=0}^{N} \cos\left(k \frac{2j+1}{2(N+1)}\pi\right)$$

$$= \frac{\pi}{2(N+1)} \left[\sum_{j=0}^{N} \cos\left(k \frac{2j+1}{2(N+1)}\pi\right) + \sum_{j=0}^{N} \cos\left(-k \frac{2j+1}{2(N+1)}\pi\right)\right].$$

Ahora, efectuando el cambio de índice $j \leftarrow -j-1$ (que es su propio inverso) en la segunda suma,

$$\sum_{j=0}^{N} \cos\left(-k\frac{2j+1}{2(N+1)}\pi\right) = \sum_{j=-(N+1)}^{-1} \cos\left(-k\frac{2(-j-1)+1}{2(N+1)}\pi\right)$$

$$= \sum_{j=-(N+1)}^{-1} \cos\left(k\frac{2j+1}{2(N+1)}\pi\right).$$

Por lo tanto, podemos fusionar las dos sumas, obteniendo

$$\sum_{j=0}^{N} T_k(x_j)\, w_j = \frac{\pi}{2(N+1)} \sum_{j=-(N+1)}^{N} \cos\left(k\frac{2j+1}{2(N+1)}\pi\right)$$

$$\overset{M:=2(N+1)}{=} \pi\, \Re\left(\frac{1}{M}\sum_{j=-M/2}^{M/2-1} \exp\left(\imath k\frac{2j+1}{M}\pi\right)\right).$$

Ahora,

$$\frac{1}{M}\sum_{j=-M/2}^{M/2-1}\exp\left(\imath k\frac{2j+1}{M}\pi\right)=\exp\left(\frac{\imath k\pi}{M}\right)\frac{1}{M}\sum_{j=-M/2}^{M/2-1}\exp\left(\imath k\frac{2\pi j}{M}\right)$$

$$\overset{\text{P. 8, (7)}}{=}\exp\left(\frac{\imath k\pi}{M}\right)\delta_{k \bmod M,0}.$$

Así, por fin

$$\sum_{j=0}^{N}T_k(x_j)\,w_j=\pi\cos\left(\frac{k\pi}{M}\right)\begin{cases}1 & \text{si } k=0,\\ 0 & \text{si } 1\le k\le M-1\end{cases}$$

$$=\begin{cases}\pi & \text{si } k=0,\\ 0 & \text{si } 1\le k\le 2N+1\end{cases}\quad\text{(¡rango extra!)}$$

$$=\int_{-1}^{1}T_k(x)\,w_0\,\mathrm{d}x.$$

Definición 38 (Cuadratura de Gauss–Lobatto–Chebyshev)

Dado $N \in \mathbb{N}$, definimos los nodos $(x_j)_{j=0}^N$ y los pesos $(w_j)_{j=0}^N$ de la **regla de cuadratura de Gauss–Lobatto–Chebyshev** de $N+1$ nodos como la unión de $\{-1, 1\}$ con las raíces de T_N' y las soluciones del sistema lineal de ecuaciones

$$(\forall\, k \in \{0, \ldots, N\}) \quad \sum_{j=0}^{N} x_j^k\, w_j = \int_{-1}^{1} x^k\, w_0(x)\, \mathrm{d}x,$$

respectivamente.

De forma análoga como se hace para la cuadratura de Gauss–Chebyshev, podemos probar las fórmulas explícitas

$$(\forall\, j \in \{0, \ldots, N\}) \quad x_j = \cos\left(\frac{j\pi}{N}\right),\ w_j = \begin{cases} \frac{\pi}{2N} & \text{si } j \in \{0, N\}, \\ \frac{\pi}{N} & \text{en otro caso.} \end{cases}$$

Lema 39 (Grado de exactitud de cuadraturas gaussianas)

Las reglas de cuadratura de Gauss–Gegenbauer/Chebyshev de $N+1$ nodos son exactas en $\mathbb{P}_{2N+1}$ y las de Gauss–Lobatto–Gegenbauer/Chebyshev de $N+1$ nodos son exactas en $\mathbb{P}_{2N-1}$.

Demostración: Recordemos que, dado un polinomio de p grado n y un polinomio *divisor* d no nulo de grado menor o igual a n, existe un único polinomio *cociente* q y un único polinomio *residuo* r tales que

$$p = d\,q + r \qquad \wedge \qquad (\deg(r) < \deg(d) \ \vee \ r \equiv 0).$$

Sean $(x_j)_{j=0}^N$ y $(w_j)_{j=0}^N$ los nodos que corresponden a alguna de las reglas de Gauss–Gegenbauer respecto al parámetro $\lambda > -1/2$, $\lambda \neq 0$ y sea $p \in \mathbb{P}_{2N+1}$. Si $p \in \mathbb{P}_N$, entonces, por construcción

(Definición 35), $\sum_{j=0}^{N} p(x_j)\, w_j = \int_{-1}^{1} p(x)\, w_\lambda(x)\, \mathrm{d}x$. Si $p \in \mathbb{P}_{2N+1} \setminus \mathbb{P}_N$, entonces $N+1 \leq \deg(p) \leq 2N+1$. Luego, existen un polinomio q y un polinomio r tales que $p = C_{N+1}^{(\lambda)}\, q + r$ con $\deg(r) < N+1$ o $r \equiv 0$. Notemos que necesariamente q es de grado $\deg(p) - (N+1)$, de modo que $0 \leq \deg(q) \leq N$. Así, explotando la ortogonalidad de $C_{N+1}^{(\lambda)}$, la exactitud de la regla de cuadratura en $\mathbb{P}_N$ y que los nodos son las raíces de $C_{N+1}^{(\lambda)}$,

$$\begin{aligned}
\int_{-1}^{1} p(x)\, w_\lambda(x)\, \mathrm{d}x &= \langle C_{N+1}^{(\lambda)}, \overline{q} \rangle_\lambda + \int_{-1}^{1} r(x)\, w_\lambda(x)\, \mathrm{d}x \\
&= \int_{-1}^{1} r(x)\, w_\lambda(x)\, \mathrm{d}x = \sum_{j=0}^{N} r(x_j)\, w_j \\
&= \sum_{j=0}^{N} \Big(C_{N+1}^{(\lambda)}(x_j)\, q(x_j) + r(x_j) \Big) w_j = \sum_{j=0}^{N} p(x_j)\, w_j.
\end{aligned}$$

El mismo argumento funciona para el caso Gauss–Chebyshev.

Sean ahora $(x_j)_{j=0}^N$ y $(w_j)_{j=0}^N$ los nodos que corresponden a alguna de las reglas de Gauss–Lobatto–Gegenbauer respecto al parámetro $\lambda > -1/2$, $\lambda \neq 0$ y sea $p \in \mathbb{P}_{2N-1}$. Si $p \in \mathbb{P}_N$, entonces, por construcción (Definición 36), $\sum_{j=0}^N p(x_j)\, w_j = \int_{-1}^1 p(x)\, w_\lambda(x)\, \mathrm{d}x$. Si $p \in \mathbb{P}_{2N-1} \setminus \mathbb{P}_N$, entonces $N+1 \leq \deg(p) \leq 2N-1$. Luego, existen un polinomio q y un polinomio r tales que

$$p(x) = C_N^{(\lambda)'}(x)(1-x^2)q(x) + r(x) \quad \wedge \quad \deg(r) < N+1 \text{ o } r \equiv 0.$$

Necesariamente q es de grado $\deg(p) - (N+1)$, de modo que $0 \leq \deg(q) \leq N-2$. Así, explotando la ortogonalidad de $C_N^{(\lambda)'} \propto C_{N-1}^{(\lambda+1)}$ (cf. Lema 25), la exactitud de la regla de

cuadratura en $\mathbb{P}_N$ y que los nodos son las raíces del polinomio $C_N^{(\lambda)'}(x)(1-x^2)$,

$$\begin{aligned}
&\int_{-1}^{1} p(x) w_\lambda(x)\,\mathrm{d}x \\
&= \int_{-1}^{1} \overbrace{C_N^{(\lambda)'}(x)}^{\propto C_{N-1}^{(\lambda+1)}(x)} \overbrace{q(x)}^{\in \mathbb{P}_{N-2}} \overbrace{(1-x^2) w_\lambda(x)}^{w_{\lambda+1}(x)}\,\mathrm{d}x + \int_{-1}^{1} r(x)\, w_\lambda(x)\,\mathrm{d}x \\
&= \int_{-1}^{1} r(x)\, w_\lambda(x)\,\mathrm{d}x = \sum_{j=0}^{N} r(x_j)\, w_j \\
&= \sum_{j=0}^{N} \left(C_N^{(\lambda)'}(x_j)\,(1-x_j^2)\, q(x_j) + r(x_j) \right) w_j = \sum_{j=0}^{N} p(x_j)\, w_j.
\end{aligned}$$

El mismo argumento funciona para el caso Gauss–Lobatto–Chebyshev. □

Observación 40

- El polinomio $s(x) = C_{N+1}^{(\lambda)}(x)^2 \in \mathbb{P}_{2N+2}$ en los casos Gauss–Gegenbauer y el polinomio $t(x) = C_N^{(\lambda)'}(x)^2(1-x^2) \in \mathbb{P}_{2N}$ en los casos Gauss–Lobatto–Gegenbauer revelan que los grados máximos de exactitud del Lema 39 no se pueden mejorar.
- Los polinomios $s(x)/(x-x_j)^2$ $(0 \leq j \leq N)$ en los casos Gauss–Gegenbauer y los polinomios $t(x)/(x-x_j)^2$ $(1 \leq j \leq N-1)$, $t(x)/(1-x)$ y $t(x)/(1+x)$ en los casos Gauss–Lobatto–Gegenbauer revelan que los pesos son todos positivos.
- Lo análogo corre para las cuadraturas Gauss–*–Chebyshev.

Sea $\lambda > -1/2$, $\lambda \neq 0$ y sean $(x_j)_{j=0}^N$ y $(w_j)_{j=0}^N$ los nodos y los pesos de la cuadratura de Gauss–[$\emptyset$|Lobatto]–Gegenbauer asociada. Los correspondientes **coeficientes de Fourier–Gegenbauer discretos** de una función $u\colon [-1,1] \to \mathbb{C}$ se definen por

$$(\forall\, k \in \{0,\dots,N\}) \quad \tilde{u}_k := \frac{1}{\gamma_k} \sum_{j=0}^{N} u(x_j)\, C_k^{(\lambda)}(x_j)\, w_j, \tag{41a}$$

donde

$$(\forall\, k \in \{0,\dots,N\}) \quad \gamma_k := \sum_{j=0}^{N} C_k^{(\lambda)}(x_j)^2\, w_j. \tag{41b}$$

Esto es, son aproximaciones de los coeficientes de Fourier–Gegenbauer (Corolario 31) $\hat{u}_k = \langle u, C_k^{(\lambda)} \rangle_\lambda / h_k^{(\lambda)}$, donde ambas integrales ahí implícitas se aproximan por la misma regla de cuadratura.

La correspondiente **serie de Fourier–Gegenbauer discreta** está entonces definida por (los $\tilde{u}_k$ son los de (41), por supuesto)

$$I_N(u) = \sum_{k=0}^{N} \tilde{u}_k \, C_k^{(\lambda)}. \tag{42}$$

En el caso $\lambda = 0$, si $(x_j)_{j=0}^N$ y $(w_j)_{j=0}^N$ son los nodos y pesos de la cuadratura de Gauss–[$\emptyset$|Lobatto]–Chebyshev, los correspondientes **coeficientes de Fourier–Chebyshev-discretos** de una función $u\colon [-1,1] \to \mathbb{C}$ se definen por

$$(\forall\, k \in \{0,\dots,N\}) \quad \tilde{u}_k := \frac{1}{\gamma_k} \sum_{j=0}^{N} u(x_j)\, T_k(x_j)\, w_j, \tag{43a}$$

donde

$$(\forall\, k \in \{0,\dots,N\}) \quad \gamma_k := \sum_{j=0}^{N} T_k(x_j)^2\, w_j. \tag{43b}$$

La correspondiente **serie de Fourier–Chebyshev discreta** está entonces definida por (ahora los $\tilde{u}_k$ son los de (43))

$$I_N(u) = \sum_{k=0}^{N} \tilde{u}_k \, T_k. \tag{44}$$

Lema 41 (Serie de Fourier-* discreta interpola)

Sea $\lambda > -1/2$ y sean $(x_j)_{j=0}^N$ y $(w_j)_{j=0}^N$ los nodos y los pesos de una cuadratura de Gauss–[$\emptyset$|Lobatto]–[Gegenbauer|Chebyshev]. Sea $u\colon [-1,1] \to \mathbb{C}$ y sea $I_N(u)$ su serie de Fourier–[Gegenbauer|Chebyshev] discreta correspondiente a la cuadratura. Entonces,

$$(\forall\, j \in \{0,\dots,N\}) \quad I_N(u)(x_j) = u(x_j).$$

Demostración:

Sea $\lambda > -1/2$, $\lambda \neq 0$. Sea $N \geq 0$ y sean $(x_j)_{j=0}^N$ y $(w_j)_{j=0}^N$ los nodos y pesos de la cuadratura de Gauss–Gegenbauer de parámetro λ. Sea $\check{I}_N(u) \in \mathbb{P}_N$ el único interpolador polinomial de u en esos nodos. Entonces, $\check{I}_N(u) = \sum_{k=0}^N \nu_k C_k^{(\lambda)}$, donde, para $0 \leq k \leq N$,

$$\nu_k = \frac{\langle \check{I}_N(u), C_k^{(\lambda)} \rangle_\lambda}{h_k^{(\lambda)}} = \frac{\int_{-1}^1 \overbrace{\check{I}_N(u)(x)\, C_k^{(\lambda)}(x)}^{\in \mathbb{P}_{2N}} w_\lambda(x)\,\mathrm{d}x}{\int_{-1}^1 \underbrace{C_k^{(\lambda)}(x)^2}_{\in \mathbb{P}_{2N}} w_\lambda(x)\,\mathrm{d}x}$$

$$\overset{\text{L. 39}}{=} \frac{\sum_{j=0}^N \check{I}_N(u)(x_j)\, C_k^{(\lambda)}(x_j)\, w_j}{\sum_{j=0}^N C_k^{(\lambda)}(x_j)^2\, w_j}$$

$$\overset{\substack{\text{interpolador} \\ \text{interpola}}}{=} \frac{\sum_{j=0}^N u(x_j)\, C_k^{(\lambda)}(x_j)\, w_j}{\sum_{j=0}^N C_k^{(\lambda)}(x_j)^2\, w_j} \overset{(41)}{=} \tilde{u}_k.$$

Así, $\check{I}_N(u) = \sum_{k=0}^N \tilde{u}_k C_k^{(\lambda)} \overset{(42)}{=} I_N(u)$.

Ahora sean $N \geq 1$ y sean $(x_j)_{j=0}^N$ y $(w_j)_{j=0}^N$ los nodos y pesos de la cuadratura de Gauss–Lobatto–Gegenbauer de parámetro λ. Sea $\check{I}_N(u) \in \mathbb{P}_N$ el único interpolador polinomial en esos nodos. Entonces, $\check{I}_N(u) = \sum_{k=0}^N \nu_k C_k^{(\lambda)}$, donde, por los mismos argumentos que en el caso anterior, $\nu_k = \tilde{u}_k$ para $0 \leq k \leq N-1$. Por otro lado,

$$
\begin{aligned}
0 &= \langle \overbrace{\check{I}_N(u) - \nu_N C_N^{(\lambda)}}^{\in \mathbb{P}_{N-1}}, C_N^{(\lambda)} \rangle_\lambda \\
&= \int_{-1}^{1} \left(\check{I}_N(u)(x) - \nu_N C_N^{(\lambda)}(x) \right) C_N^{(\lambda)}(x)\, w_\lambda(x)\, \mathrm{d}x \\
&\overset{\text{L. 39}}{=} \sum_{j=0}^{N} \left(\check{I}_N(u)(x_j) - \nu_N C_N^{(\lambda)}(x_j) \right) C_N^{(\lambda)}(x_j)\, w_j \\
&\overset{\substack{\text{interpolador} \\ \text{interpola}}}{=} \sum_{j=0}^{N} \left(u(x_j) - \nu_N C_N^{(\lambda)}(x_j) \right) C_N^{(\lambda)}(x_j)\, w_j .
\end{aligned}
$$

De aquí despejamos a ν_N, obteniendo

$$\nu_N = \frac{\sum_{j=0}^{N} u(x_j)\, C_N^{(\lambda)}(x_j)\, w_j}{\sum_{j=0}^{N} C_N^{(\lambda)}(x_j)^2\, w_j} \overset{(41)}{=} \tilde{u}_N,$$

por lo que en este caso también $\check{I}_N(u) = \sum_{k=0}^{N} \tilde{u}_k C_k^{(\lambda)} \overset{(42)}{=} I_N(u)$.

Las propiedades correspondientes para las series de Fourier–Chebyshev discretas se prueban de forma análoga. □

Observación 42 (Normas al cuadrado aproximadas)

Por el Lema 39, las normas al cuadrado aproximadas γ_k de (41) y (43) que participan en la definición de los coeficientes de Fourier–[Gegenbauer|Chebyshev] discretos respecto a los nodos de la cuadratura Gauss–[Gegenbauer|Chebyshev] de $N+1$ nodos coinciden con las normas al cuadrado exactas $h_k^{(\lambda)}$, $0 \leq k \leq N$.

Sigue ⟶

Observación 42 (cont.)

Si se usan los nodos de Gauss–Lobatto–[Gegenbauer|Chebyshev], tenemos $\gamma_k = h_k^{(\lambda)}$ para $0 \leq k \leq N-1$ solamente y, con la ayuda de la Observación 40, se puede probar que $\gamma_N > h_N^{(\lambda)}$.

Recordemos del Capítulo 1 que habíamos definido

$$\bar{c}_j = \begin{cases} 2 & \text{si } j \in \{0, N\}, \\ 1 & \text{si } 1 \leq j \leq N-1. \end{cases}$$

En el importantísimo caso Gauss–Lobatto–Chebyshev, la norma al cuadrado aproximada rebelde se puede evaluar de acuerdo a

$$\gamma_N \overset{(43)}{=} \sum_{j=0}^{N} T_N(x_j)^2 \, w_j = \frac{\pi}{N} \sum_{j=0}^{N} \bar{c}_j^{-1} \cos(j\pi)^2 = \pi > \frac{\pi}{2} = \| T_N \|_0^2 .$$

De acuerdo a la Observación 42, las demás normas al cuadrado aproximadas coinciden exactamente con las normas al cuadrado que suplantan en (43). Así, en resumen,

$$\gamma_0 = \pi, \qquad (\forall\, k \in \{1, \ldots, N-1\}) \quad \gamma_k = \pi/2, \qquad \gamma_N = \pi.$$

Por lo tanto, el mapa (43) que transforma los valores nodales $u(x_j)$ en los coeficientes de Fourier–Chebyshev discretos (esto es, valores modales) de su interpolador polinomial en este caso de nodos de Gauss–Lobatto–Chebyshev se puede escribir en la forma explícita

$$(\forall\, k \in \{0, \ldots, N\}) \quad \tilde{u}_k = \frac{2}{\pi \bar{c}_k} \sum_{j=0}^{N} u_j \cos\left(\frac{kj\pi}{N}\right) \frac{\pi}{\bar{c}_j N}$$

$$= \frac{2}{N \bar{c}_k} \sum_{j=0}^{N} \frac{1}{\bar{c}_j} u_j \cos\left(\frac{kj\pi}{N}\right),$$

donde $u_j = u(x_j)$. Esto coincide coincide con el mapa que define la ecuación (11) del Capítulo 1.

Como sabemos que la serie de Fourier–Chebyshev discreta $I_N(u) = \sum_{k=0}^{N} \tilde{u}_k T_k$ resulta ser el interpolador polinomial de los valores nodales que se emplearon para construir los $\tilde{u}_k$ (Lema 41), tenemos que

$$(\forall\, j \in \{0, \ldots, N\}) \quad u_j = \sum_{k=0}^{N} \tilde{u}_k T_k(x_j) = \sum_{k=0}^{N} \tilde{u}_k \cos\left(\frac{kj\pi}{N}\right).$$

Parte de este mapa de valores modales a valores nodales apareció ya en la ecuación (13) del Capítulo 1.

Para futura referencia, fijemos un nombre y una notación para estos mapas.

Definición 43 (La DCT y la IDCT)

Dado $N \in \mathbb{N}$, se definen la **transformada de cosenos discreta** $\mathrm{DCT}\colon \mathbb{C}^{N+1} \to \mathbb{C}^{N+1}$ y la **transformada de cosenos discreta inversa** $\mathrm{IDCT}\colon \mathbb{C}^{N+1} \to \mathbb{C}^{N+1}$, respectivamente por

$$\left(\forall\, u = (u_j)_{j=0}^{N} \in \mathbb{C}^{N+1}\right)$$

$$\mathrm{DCT}(u) := \left(\frac{2}{N\bar{c}_k} \sum_{j=0}^{N} \frac{1}{\bar{c}_j}\, u_j \cos\left(\frac{kj\pi}{N}\right) \right)_{k=0}^{N}$$

y

$$\left(\forall\, \tilde{u} = (\tilde{u}_k)_{k=0}^{N} \in \mathbb{C}^{N+1}\right) \quad \mathrm{IDCT}(\tilde{u}) := \left(\sum_{k=0}^{N} \tilde{u}_k \cos\left(\frac{kj\pi}{N}\right) \right)_{j=0}^{N}.$$

Ya sabemos que la DCT y la IDCT son una la inversa de la otra.

Observación 44 (Confesiones sobre nomenclatura)

- Lo que llamamos *la* DCT, con algunos reescalamientos menores que no son los mismos en todas la literatura, se conoce como la DCT-I dentro de una familia que llega hasta la DCT-IV o DCT-VIII.
- Nuestra IDCT es tan parecida a la DCT que la literatura no suele ponerle un nombre aparte.
- Aquí definimos a la DCT como una función; existe un excelente *algoritmo* para implementarla que lleva el mismo nombre (a diferencia del par DFT/FFT).
- También existen las transformadas de senos discretas.

REDFT00 (DCT-I)

An `REDFT00` transform (type-I DCT) in FFTW is defined by:

$$Y_k = X_0 + (-1)^k X_{n-1} + 2\sum_{j=1}^{n-2} X_j \cos[\pi jk/(n-1)].$$

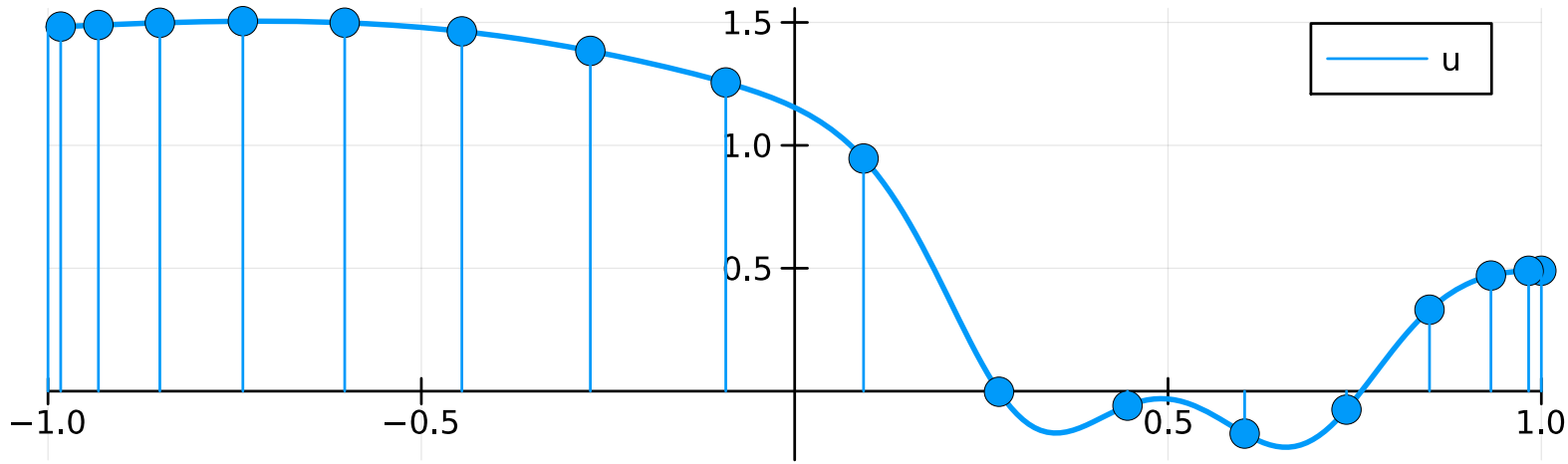
Figura 7: Extracto del manual de la librería FFTW (*Fastest Fourier Transform in the West*) con la definición de la DCT-I que computa.

Para $\lambda \neq 0$, los nodos y pesos de las cuadraturas de tipo Gauss–*–Gegenbauer introducidas en la Definición 35 y la Definición 36 pueden, en principio, calcularse computando las raíces de un polinomio ortogonal y posteriormente resolviendo un sistema lineal de ecuaciones. Pero hay una manera mejor.

```
using Plots
using LinearAlgebra # Para función 'norm'

u(x::Float64) = (0.25/(1 + (x+1/3)^2) + cos(4*exp(-10*(x-0.5)^2))/4) * (2.5-x) # Función de ejemplo
N = 17
nodes = [cos(j*π/N) for j=0:N]  # Nodos de Gauss-Lobatto-Chebyshev
vu = u.(nodes) # Vector de valores nodales

18-element Vector{Float64}:
  0.48996710738499505
  0.4898291069597808
  0.47037818262976505
  0.33147870091793447
 -0.07494126447191485
 -0.17251075302491004
 -0.058847027833319625
 -0.002376263608148019
  0.9464654634407871
  1.255893080520156
  1.3843362298334192
  1.4636804060756803
  1.498867877157313
  1.505073563606263
  1.498519295737415
  1.4895666460016306
  1.483066613249879
  1.4807692307692306
```

Figura 8: Ilustración en Julia del uso de la DFT y de la IDFT (1/6).

```
p = plot(u, xlims=[-1, 1], color=1, linewidth=2, label="u", size=(600,200), framestyle=:origin)
scatter!(p, nodes, vu, color=1, markersize=6, label="")
sticks!(p, nodes, vu, color=1, label="")
```

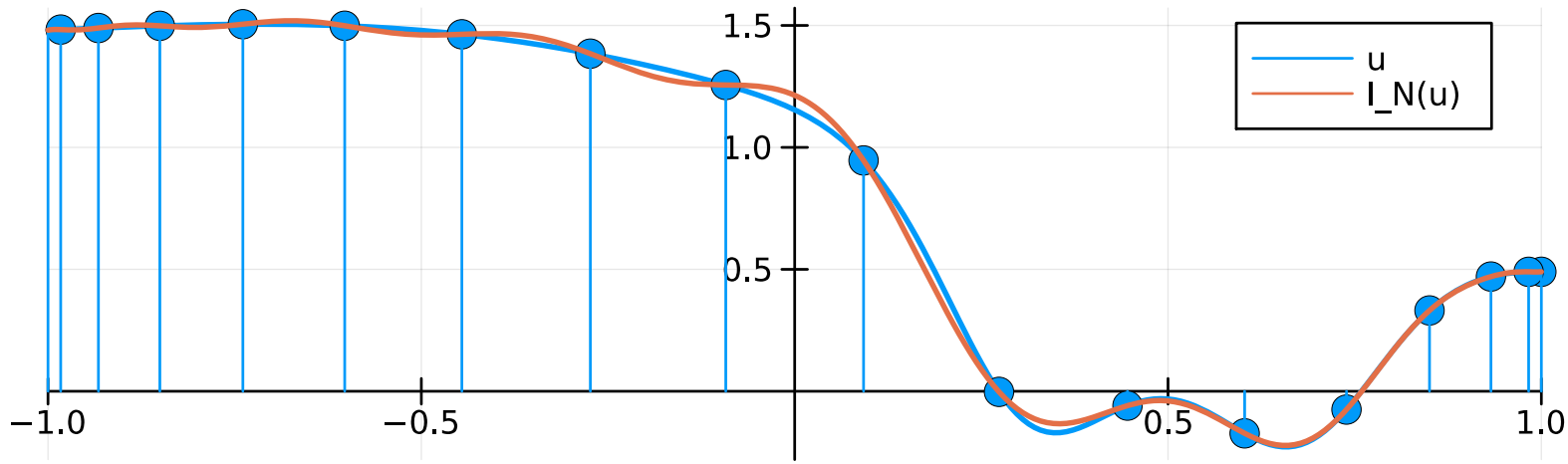



```
indexRange = 0:N
cbar = [if j==0||j==N; 2.0; else 1; end for j=0:N]
# Cómputo de valores modales empleando la DCT explícitamente (ESTO ES LENTO; EN LA PRÁCTICA SE USA
EL ALGORITMO DCT)
vũ = [2/N*sum(vu./cbar .* cos.(k*indexRange*π/N)) for k in indexRange] ./ cbar

18-element Vector{Float64}:
  0.852579295662873
```

Figura 9: Ilustración en Julia del uso de la DFT y de la IDFT (2/6).

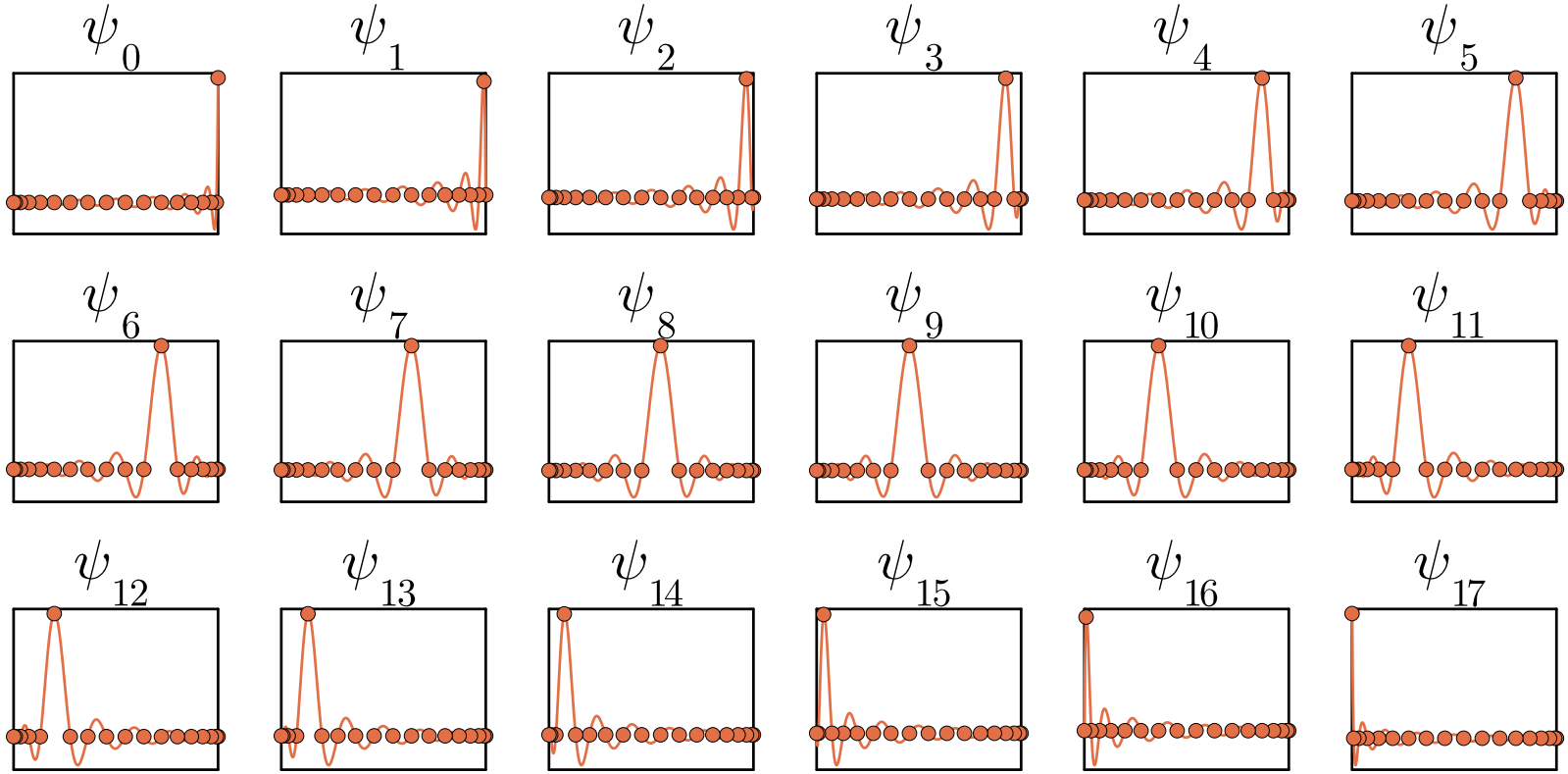
```
-0.7853995886691164
 0.07902473495869387
 0.4035625925797944
 0.14977448510112512
-0.10063863734645712
-0.11310519998063615
-0.005454160973198004
 0.03119145890755685
-0.025822572052798254
-0.05221168773519745
 0.006261510800351304
 0.06286814480113263
 0.03759851748940437
-0.027875395401098194
-0.042674007084970995
 0.0031223327626620566
 0.017165283564872923

# Reconstrucción de valores nodales empleando la IDCT explícitamente (ESTO ES LENTO; EN LA PRÁCTICA
SE USA EL ALGORITMO DCT)
vu2 = [sum(vũ .* cos.(indexRange*j*π/N)) for j in indexRange]
norm(vu - vu2)/norm(vu)

1.1183429834856038e-15

interpolator(x::Float64) = sum(vũ .* cos.(indexRange*acos(x))) # Función interpoladora
plot!(p, x -> interpolator(x), xlims=[-1, 1], color=2, linewidth=2, label="I_N(u)")
```

Figura 10: Ilustración en Julia del uso de la DFT y de la IDFT (3/6).




```
# Funciones delta discretas
ψ(x::Float64, j::Int64) = prod([(x - nodes[i+1])/(nodes[j+1] - nodes[i+1]) for i=[0:j-1;j+1:N]])
l = grid(3,6)
ψtitles = ["\$\\psi_{$j}\$" for spuriousIndex=1:1, j=0:N]
ψp = plot([x -> ψ(x, j) for j=0:N], xlims=[-1, 1], layout=l, linewidth=1, color=2, label="",
size=(600,300), framestyle=:box, ticks=[], title=ψtitles)
scatter(ψp, nodes, [Float64(i==j) for i=0:N, j=0:N], layout=l, markersize=3, color=2, label="")
```

Figura 11: Ilustración en Julia del uso de la DFT y de la IDFT (4/6).




```
# Forma alternativa del interpolador y comparación
interpolator2(x::Float64) = sum(vu .* [ψ(x, j) for j=0:N])
plot(x -> abs(interpolator(x) - interpolator2(x)), xlims=[-1, 1], size=(600,200), label="")
```

Figura 12: Ilustración en Julia del uso de la DFT y de la IDFT (5/6).

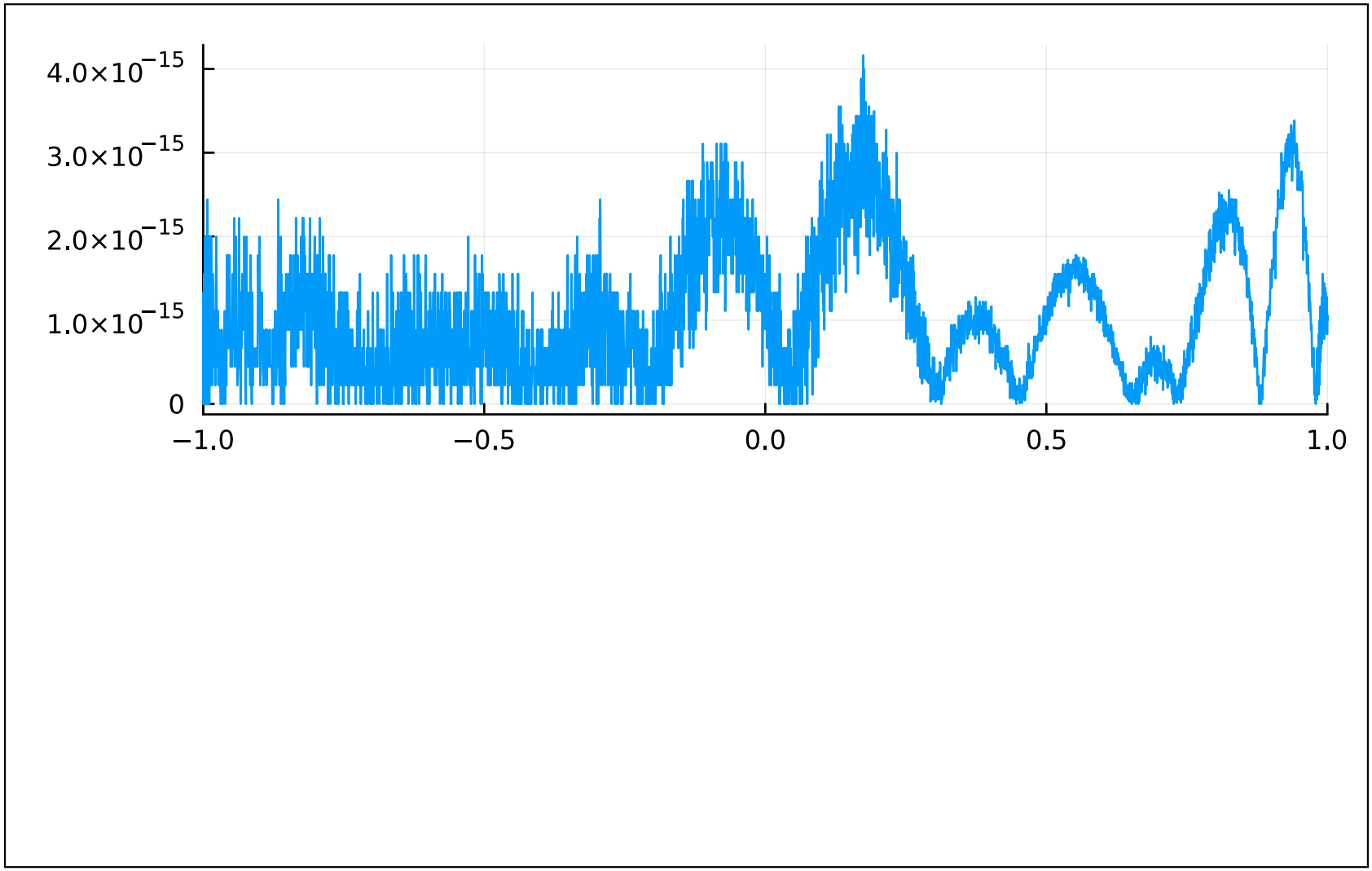


Figura 13: Ilustración en Julia del uso de la DFT y de la IDFT (6/6).

Sea $\lambda > -1/2$, $\lambda \neq 0$. Dado $n \geq 0$, sea $\hat{p}_n := C_n^{(\lambda)} / \sqrt{h_n^{(\lambda)}}$, de modo que $\|\hat{p}_n\|_\lambda = 1$. Reescalando la recurrencia de tres (dos) términos del Lema 23, sabemos que, para todo $n \geq 0$, existen constantes β_n y γ_n tales que

$$x\,\hat{p}_n(x) = \beta_n\,\hat{p}_{n-1}(x) + \gamma_n\,\hat{p}_{n+1}(x).$$

Como los $\hat{p}_n$ forman una base ortonormal, $\beta_n = \langle x\hat{p}_n, \hat{p}_{n-1}\rangle_\lambda$ y $\gamma_n = \langle x\hat{p}_n, \hat{p}_{n+1}\rangle_\lambda = \langle \hat{p}_n, x\hat{p}_{n+1}\rangle_\lambda = \langle x\hat{p}_{n+1}, \hat{p}_n\rangle_\lambda = \beta_{n+1}$. Así, dado $N \geq 0$,

$$\begin{aligned}
x\hat{p}_0(x) &= \beta_1\,\hat{p}_1(x),\\
x\hat{p}_1(x) &= \beta_1\,\hat{p}_0(x) + \beta_2\,\hat{p}_2(x),\\
x\hat{p}_2(x) &= \beta_2\,\hat{p}_1(x) + \beta_3\,\hat{p}_3(x),\\
&\vdots\\
x\hat{p}_{N-1}(x) &= \beta_{N-1}\,\hat{p}_{N-2}(x) + \beta_N\,\hat{p}_N(x),\\
x\hat{p}_N(x) &= \beta_N\,\hat{p}_{N-1}(x) + \beta_{N+1}\,\hat{p}_{N+1}(x),
\end{aligned}$$

Esto es,

$$\underbrace{\begin{pmatrix} 0 & \beta_1 & & & & & & \\ \beta_1 & 0 & \beta_2 & & & & & \\ & \beta_2 & 0 & \beta_3 & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & \beta_{N-2} & 0 & \beta_{N-1} & \\ & & & & & \beta_{N-1} & 0 & \beta_N \\ & & & & & & \beta_N & 0 \end{pmatrix}}_{:= J} \underbrace{\begin{pmatrix} \hat{p}_0(x) \\ \hat{p}_1(x) \\ \hat{p}_2(x) \\ \vdots \\ \vdots \\ \hat{p}_{N-2}(x) \\ \hat{p}_{N-1}(x) \\ \hat{p}_N(x) \end{pmatrix}}_{:= \hat{p}(x)}$$

$$= x \begin{pmatrix} \hat{p}_0(x) \\ \hat{p}_1(x) \\ \hat{p}_2(x) \\ \vdots \\ \vdots \\ \hat{p}_{N-2}(x) \\ \hat{p}_{N-1}(x) \\ \hat{p}_N(x) \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ 0 \\ \beta_{N+1}\, \hat{p}_{N+1}(x) \end{pmatrix}. \quad (45)$$

Por lo tanto, un número x^* es una raíz de $\hat{p}_{N+1}$ (y, consiguientemente, un nodo de la cuadratura de Gauss–Gegenbauer de parámetro λ de $N+1$ nodos) si y solo si

$$J\,\hat{p}(x^*) = x^*\,\hat{p}(x^*).$$

El cómputo de estos autovalores (y nodos y raíces) $x_0, \ldots, x_N$ se beneficia de la simetría de J. Además, como los x_j son distintos entre sí, la simetría de J obliga a que $\hat{p}(x_i) \cdot \hat{p}(x_j) = 0$ si $i \neq j$.

Ahora, los pesos $w_0, \ldots, w_N$ de la regla de cuadratura satisfacen

$$\left[\; \hat{p}(x_0) \;\middle|\; \hat{p}(x_1) \;\middle|\; \cdots \;\middle|\; \hat{p}(x_N) \;\right] \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} \sqrt{h_0^{(\lambda)}} \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Premultiplicando esta ecuación por la traspuesta de la matriz de autovectores y empleando la $\mathbb{R}^{N+1}$-ortogonalidad de éstos, obtenemos

$$\begin{bmatrix} \|\hat{p}(x_0)\|^2_{\mathbb{R}^{N+1}} & & & \\ & \|\hat{p}(x_1)\|^2_{\mathbb{R}^{N+1}} & & \\ & & \ddots & \\ & & & \|\hat{p}(x_N)\|^2_{\mathbb{R}^{N+1}} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_N \end{bmatrix}$$

$$= \sqrt{h_0^{(\lambda)}} \begin{bmatrix} \hat{p}_0(x_0) \\ \hat{p}_0(x_1) \\ \vdots \\ \hat{p}_0(x_N) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix},$$

por lo que

$$(\forall\, j \in \{0, \ldots, N\}) \quad w_j = \frac{1}{\|\hat{p}(x_j)\|^2_{\mathbb{R}^{N+1}}}.$$

Ahora, en la práctica, los algoritmos que computan descomposiciones espectrales de matrices devuelven los autovectores con una normalización distinta a la que aquí hemos asumido.

En general lo que tendremos es una colección de autopares $(x_j, q(x_j))$, $0 \leq j \leq N$, donde los autovectores obtenidos son múltiplos escalares de los que usamos en nuestro análisis; esto es, $q(x_j) = \alpha_j \, \hat{p}(x_j)$ para alguna constante $\alpha_j \neq 0$. Como $q_0(x_j) = \alpha_j \, \hat{p}_0(x_j) = \alpha_j / \sqrt{h_0^{(\lambda)}}$, despejamos $\alpha_j = q_0(x_j) \sqrt{h_0^{(\lambda)}}$ y

$$(\forall\, j \in \{0, \ldots, N\}) \quad w_j = \frac{1}{\|\hat{p}(x_j)\|^2_{\mathbb{R}^{N+1}}}$$

$$= \frac{1}{\|\alpha_j^{-1} q(x_j)\|^2_{\mathbb{R}^{N+1}}} = h_0^{(\lambda)} \frac{q_0(x_j)^2}{\|q(x_j)\|^2_{\mathbb{R}^{N+1}}}. \quad (46)$$

Ahora que ya sabemos qué hacer con la matriz de Jacobi J de (45), preocupémonos de su construcción. Tenemos

$$\sqrt{h_n^{(\lambda)} h_{n-1}^{(\lambda)}}\,\beta_n = \sqrt{h_n^{(\lambda)} h_{n-1}^{(\lambda)}}\,\langle x\hat{p}_n, \hat{p}_{n-1}\rangle_\lambda = \langle x\, C_n^{(\lambda)}, C_{n-1}^{(\lambda)}\rangle_\lambda$$
$$= h_{n-1}^{(\lambda)} \frac{\langle x\, C_n^{(\lambda)}, C_{n-1}^{(\lambda)}\rangle_\lambda}{h_{n-1}^{(\lambda)}} \overset{\text{L. 23}}{=} h_{n-1}^{(\lambda)} \frac{n+2\lambda-1}{2(n+\lambda)},$$

por lo que

$$\beta_n = \frac{\sqrt{h_{n-1}^{(\lambda)}}}{\sqrt{h_n^{(\lambda)}}} \frac{n+2\lambda-1}{2(n+\lambda)}.$$

El festival de funciones gamma que involucra β_n puede causar problemas numéricos para n grande. Por ello, es mejor computar estos números de forma recursiva a partir de las ecuaciones (obtenidas empleando el valor conocido (32) de las normas al cuadrado $h_n^{(\lambda)}$):

$$\begin{aligned} \beta_1 &= \frac{\operatorname{sign}(\lambda)}{\sqrt{2(\lambda+1)}}, \\ (\forall\, n \in \mathbb{N}) \quad \beta_{n+1} &= \frac{\sqrt{(n+1)(n+2\lambda)(n+\lambda-1)}}{\sqrt{n\,(n+\lambda+1)(n+2\lambda-1)}}\,\beta_n. \end{aligned} \tag{47}$$

A continuación ilustramos la obtención de nodos y pesos de reglas de cuadratura de Gauss–Gegenbauer mediante este procedimiento, el cual se denomina **método de Golub–Welsch**. También comprobamos que las reglas de $N+1$ nodos obtenidas efectivamente son exactas (salvo errores de punto flotante) para calcular la integral ponderada de polinomios hasta grado $2N+1$.

```
using LinearAlgebra
using SpecialFunctions

N = 8

8

# Con el peso de Chebyshev del segundo tipo
λ = 1.0
β = zeros(N)
β[1] = sign(λ)/sqrt(2(λ+1))
for n=1:N-1
        β[n+1] = sqrt((n+1)*(n+2λ)*(n+λ-1)/(n*(n+λ+1)*(n+2λ-1)))*β[n]
end
J = diagm(1 => β, -1 => β)

9×9 Matrix{Float64}:
 0.0  0.5  0.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.5  0.0  0.5  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.5  0.0  0.5  0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.5  0.0  0.5  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.5  0.0  0.5  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.5  0.0  0.5  0.0  0.0
 0.0  0.0  0.0  0.0  0.0  0.5  0.0  0.5  0.0
 0.0  0.0  0.0  0.0  0.0  0.0  0.5  0.0  0.5
 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.5  0.0
```

Figura 14: Ilustración en Julia del método de Golub–Welsch (1/6).

```
JSD = eigen(J)
nodes = JSD.values

9-element Vector{Float64}:
 -0.9510565162951523
 -0.8090169943749472
 -0.5877852522924725
 -0.30901699437494723
 -4.440892098500626e-16
  0.3090169943749477
  0.5877852522924736
  0.8090169943749475
  0.9510565162951535

h0 = 2^(1-2λ) * π * gamma(2λ)/(gamma(λ)^2*λ);
# Los vectores propios que devuelve la función 'eigen' vienen normalizados
weights = h0 * JSD.vectors[1,:].^2

9-element Vector{Float64}:
 0.029999540371608065
 0.10853935671135308
 0.20561990864762597
 0.28415972498737085
 0.31415926535898026
 0.2841597249873704
 0.2056199086476258
 0.10853935671135348
 0.029999540371608076
```

Figura 15: Ilustración en Julia del método de Golub–Welsch (2/6).

```
denominator = sin.(acos.(nodes))
quadratures = [(sin.((k+1)*acos.(nodes)) ./ denominator) · weights for k=0:2N+1]
integrals = [if k==0; h0; else 0.0; end for k=0:2N+1]
[quadratures integrals]

18×2 Matrix{Float64}:
  1.5708       1.5708
  7.91034e-16  0.0
 -5.82867e-16  0.0
  3.19189e-15  0.0
 -1.11022e-16  0.0
 -2.06779e-15  0.0
 -5.13478e-15  0.0
  3.44169e-15  0.0
 -4.48946e-15  0.0
  6.06836e-15  0.0
 -4.85029e-15  0.0
  5.10009e-15  0.0
  3.20577e-15  0.0
 -4.92661e-15  0.0
 -1.31839e-15  0.0
  3.01148e-15  0.0
  3.40006e-15  0.0
 -5.34989e-15  0.0

# Con el peso Legendre
λ = 0.5
β = zeros(N)
```

Figura 16: Ilustración en Julia del método de Golub–Welsch (3/6).

```
β[1] = sign(λ)/sqrt(2(λ+1))
for n=1:N-1
        β[n+1] = sqrt((n+1)*(n+2λ)*(n+λ-1)/(n*(n+λ+1)*(n+2λ-1)))*β[n]
end
J = diagm(1 => β, -1 => β)

9×9 Matrix{Float64}:
 0.0      0.57735   0.0       0.0       ...  0.0       0.0       0.0
 0.57735  0.0       0.516398  0.0            0.0       0.0       0.0
 0.0      0.516398  0.0       0.507093       0.0       0.0       0.0
 0.0      0.0       0.507093  0.0            0.0       0.0       0.0
 0.0      0.0       0.0       0.503953       0.0       0.0       0.0
 0.0      0.0       0.0       0.0       ...  0.501745  0.0       0.0
 0.0      0.0       0.0       0.0            0.0       0.50128   0.0
 0.0      0.0       0.0       0.0            0.50128   0.0       0.500979
 0.0      0.0       0.0       0.0            0.0       0.500979  0.0

JSD = eigen(J)
nodes = JSD.values

9-element Vector{Float64}:
 -0.9681602395076245
 -0.8360311073266349
 -0.6133714327005888
 -0.3242534234038086
 -1.1102230246251565e-16
  0.32425342340380925
  0.6133714327005908
```

Figura 17: Ilustración en Julia del método de Golub–Welsch (4/6).

```
  0.8360311073266359
  0.9681602395076261

h0 = 2^(1-2λ) * π * gamma(2λ)/(gamma(λ)^2*λ);
weights = h0 * JSD.vectors[1,:].^2

9-element Vector{Float64}:
 0.08127438836157477
 0.18064816069485742
 0.26061069640293466
 0.3123470770400029
 0.3302393550012609
 0.3123470770400021
 0.2606106964029343
 0.1806481606948582
 0.08127438836157432

quadratures = zeros(2N+2);
valuesOld = ones(N+1)
valuesCurrent = 2λ*nodes
quadratures[1] = valuesOld⋅weights
quadratures[2] = valuesCurrent⋅weights
for n=1:2N
        valuesNew = (2*(n+λ)*nodes.*valuesCurrent - (n+2λ-1)*valuesOld) / (n+1)
        quadratures[(n+1)+1] = valuesNew⋅weights
        # Los 'global' le confirman a Julia que sí, queremos sobreescribir en variables definidas
fuera del ciclo 'for'
        global valuesOld = valuesCurrent
```

Figura 18: Ilustración en Julia del método de Golub–Welsch (5/6).

```
        global valuesCurrent = valuesNew
end
integrals = [if k==0; h0; else 0.0; end for k=0:2N+1]
[quadratures integrals]

18×2 Matrix{Float64}:
  2.0          2.0
  6.52256e-16  0.0
 -1.11022e-15  0.0
  2.22045e-15  0.0
 -1.52656e-16  0.0
 -2.08167e-16  0.0
 -2.13718e-15  0.0
  1.43982e-15  0.0
 -2.52923e-15  0.0
  2.5992e-15   0.0
 -2.37831e-15  0.0
  2.07473e-15  0.0
 -2.15106e-16  0.0
  4.61436e-16  0.0
 -2.13024e-15  0.0
  2.4078e-15   0.0
 -1.11716e-15  0.0
  2.08167e-16  0.0
```

Figura 19: Ilustración en Julia del método de Golub–Welsch (6/6).

El método de Golub–Welsch se puede adaptar para computar los nodos y los pesos de cuadraturas de Gauss–**Lobatto**–Gegenbauer.

Disponemos de fórmulas explícitas para los nodos y pesos de cuadraturas de Gauss–*–Chebyshev, por lo que no necesitamos del método de Golub–Welsch en ese caso (igual funciona). **Pero**, reproduzcamos la ecuación vectorial (45) en el caso Chebyshev, **sin** normalizar y que el último polinomio que aparezca sea el de grado N esta vez. De la Observación 24, para $N \geq 1$,

$$
\begin{aligned}
x\,T_0(x) &= 1\,T_1(x),\\
x\,T_1(x) &= 1/2\,T_0(x) + 1/2\,T_2(x),\\
x\,T_2(x) &= 1/2\,T_1(x) + 1/2\,T_3(x),\\
&\vdots\\
x\,T_{N-2}(x) &= 1/2\,T_{N-3}(x) + 1/2\,T_{N-1}(x),\\
x\,T_{N-1}(x) &= 1/2\,T_{N-2}(x) + 1/2\,T_N(x).
\end{aligned}
$$

Esto es,

$$\underbrace{\begin{pmatrix} 0 & 1 & & & & & & \\ 1/2 & 0 & 1/2 & & & & & \\ & 1/2 & 0 & 1/2 & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & 1/2 & 0 & 1/2 & \\ & & & & & 1/2 & 0 & 1/2 \\ & & & & & & 1/2 & 0 \end{pmatrix}}_{:=\tilde{J}} \underbrace{\begin{pmatrix} T_0(x) \\ T_1(x) \\ T_2(x) \\ \vdots \\ \vdots \\ T_{N-3}(x) \\ T_{N-2}(x) \\ T_{N-1}(x) \end{pmatrix}}_{:=T(x)}$$

$$= x \begin{pmatrix} T_0(x) \\ T_1(x) \\ T_2(x) \\ \vdots \\ \vdots \\ T_{N-3}(x) \\ T_{N-2}(x) \\ T_{N-1}(x) \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ 0 \\ 1/2\, T_N(x) \end{pmatrix}. \quad (48)$$

Sea p un polinomio de grado exactamente N cuyos coeficientes p_k en su expansión de Fourier–Chebyshev $\sum_{k=0}^{N} p_k T_k$ son conocidos. Entonces, si x^* es una raíz de p,

$$T_N(x^*) = -\sum_{k=0}^{N-1} \frac{p_k}{p_N} T_k(x^*),$$

por lo que

$$\begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1/2\, T_N(x^*) \end{pmatrix} = \underbrace{\frac{1}{2p_N} \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 0 \\ -p_0 & -p_1 & \cdots & -p_{N-1} \end{pmatrix}}_{:= \tilde{J}_p} \underbrace{\begin{pmatrix} T_0(x^*) \\ T_1(x^*) \\ \vdots \\ T_{N-1}(x^*) \end{pmatrix}}_{= T(x^*)}. \quad (49)$$

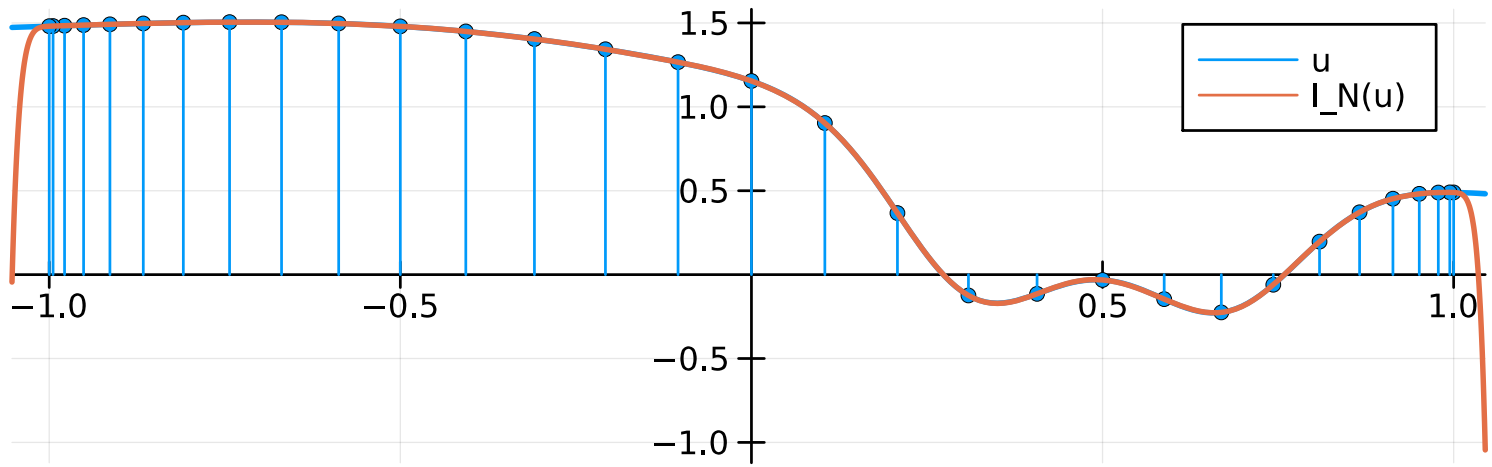
Combinando (48) y (49) obtenemos que las raíces del polinomio p son autovalores de $\tilde{J} + \tilde{J}_p$, a la cual se le denomina *matriz colega* del polinomio p.

Se puede probar el siguiente resultado más fuerte: Los autovalores de la matriz colega son, con sus multiplicidades, exactamente las raíces del polinomio p.

```
using Plots
using LinearAlgebra

u(x::Float64) = (0.25/(1 + (x+1/3)^2) + cos(4*exp(-10*(x-0.5)^2))/4) * (2.5-x) # Función de ejemplo
N = 30 # Más nodos
nodes = [cos(j*π/N) for j=0:N]  # Nodos de Gauss-Lobatto-Chebyshev
vu = u.(nodes) # Vector de valores nodales

indexRange = 0:N
cbar = [if j==0||j==N; 2.0; else 1; end for j=0:N]
# Cómputo de valores modales del polinomio de interpolación obtenido empleando la DCT explícitamente
(ESTO ES LENTO; EN LA PRÁCTICA SE USA EL ALGORITMO DCT)
vũ = [2/N*sum(vu./cbar .* cos.(k*indexRange*π/N)) for k in indexRange] ./ cbar

# Esto requiere una explicación
function interpolator(x::Float64) # Función polinomio de interpolación
        bOlder = 0.0;
        bOld = 0.0;
        for k=N:-1:1
                bNew = vũ[k+1] + 2*x*bOld - bOlder;
                bOlder = bOld;
                bOld = bNew;
        end
        vũ[1] + x*bOld - bOlder
end

# Gráfico
p = plot(u, xlims=[-1.054, 1.045], color=1, linewidth=2, label="u", size=(600,200),
```

Figura 20: Ilustración en Julia de uso de matriz colega (1/7).

```
framestyle=:origin)
scatter!(p, nodes, vu, color=1, markersize=3, label="")
sticks!(p, nodes, vu, color=1, label="")
plot!(p, x -> interpolator.(x), xlims=[-1.053, 1.045], color=2, linewidth=2, label="I_N(u)")
```




```
# Construcción de matriz colega
colleague = diagm(-1=>0.5*ones(N-1), 1=>0.5*ones(N-1))
colleague[1,2] = 1.0
for i = 1:N
        colleague[N,i] -= vũ[i]/vũ[N+1]/2.0
end
colleague
```

Figura 21: Ilustración en Julia de uso de matriz colega (2/7).

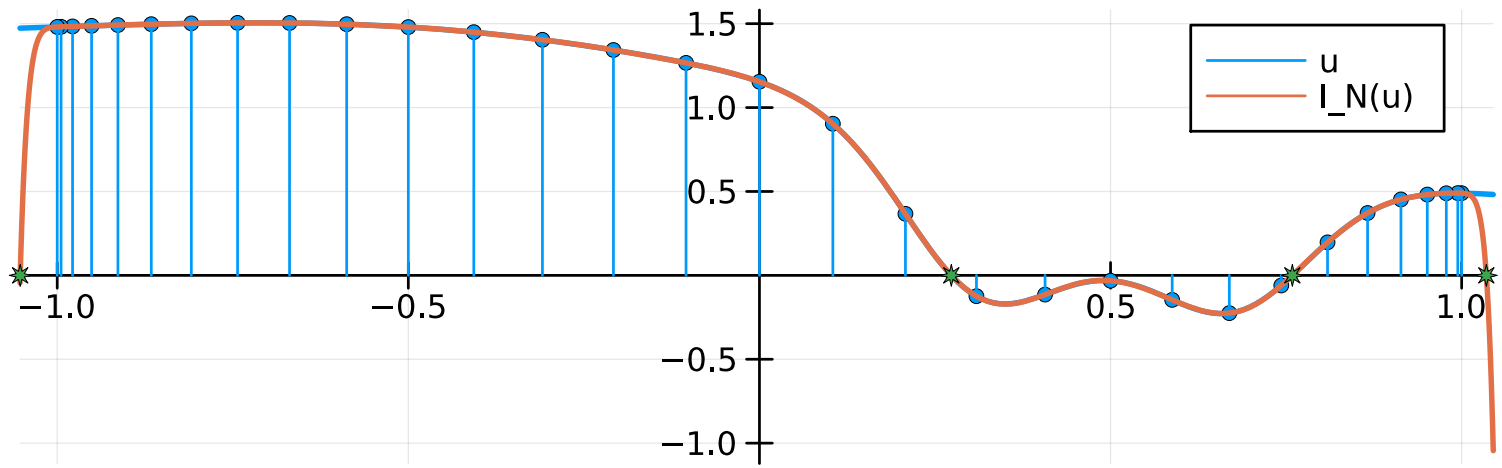
```
30×30 Matrix{Float64}:
   0.0       1.0     0.0      0.0     0.0     ...  0.0     0.0       0.0
   0.5       0.0     0.5      0.0     0.0          0.0     0.0       0.0
   0.0       0.5     0.0      0.5     0.0          0.0     0.0       0.0
   0.0       0.0     0.5      0.0     0.5          0.0     0.0       0.0
   0.0       0.0     0.0      0.5     0.0          0.0     0.0       0.0
   0.0       0.0     0.0      0.0     0.5     ...  0.0     0.0       0.0
   0.0       0.0     0.0      0.0     0.0          0.0     0.0       0.0
   0.0       0.0     0.0      0.0     0.0          0.0     0.0       0.0
   0.0       0.0     0.0      0.0     0.0          0.0     0.0       0.0
   0.0       0.0     0.0      0.0     0.0          0.0     0.0       0.0
   ⋮                                            ⋱
   0.0       0.0     0.0      0.0     0.0          0.0     0.0       0.0
   0.0       0.0     0.0      0.0     0.0          0.0     0.0       0.0
   0.0       0.0     0.0      0.0     0.0          0.0     0.0       0.0
   0.0       0.0     0.0      0.0     0.0          0.0     0.0       0.0
   0.0       0.0     0.0      0.0     0.0     ...  0.0     0.0       0.0
   0.0       0.0     0.0      0.0     0.0          0.5     0.0       0.0
   0.0       0.0     0.0      0.0     0.0          0.0     0.5       0.0
   0.0       0.0     0.0      0.0     0.0          0.5     0.0       0.5
 498.039  -458.725  45.858  235.618  87.9531       1.2191  1.21419  -0.450389

# Sus valores propios son las raíces del polinomio
roots = eigen(colleague).values

30-element Vector{ComplexF64}:
 -1.0527044089838609 + 0.0im
```

Figura 22: Ilustración en Julia de uso de matriz colega (3/7).

```
 -1.0318732527128986 - 0.06173448224075163im
 -1.0318732527128986 + 0.06173448224075163im
 -0.9693811378750417 - 0.11903675543381714im
 -0.9693811378750417 + 0.11903675543381714im
 -0.8664636328295219 - 0.1690692529374445im
 -0.8664636328295219 + 0.1690692529374445im
 -0.7265242876508644 - 0.2099386022070839im
 -0.7265242876508644 + 0.2099386022070839im
  -0.554826750383651 - 0.23985458044907484im
                     ⋮
  0.5718326852735985 + 0.2111372482967323im
  0.7588894668619902 + 0.0im
  0.8442153897035868 - 0.14186178976691055im
  0.8442153897035868 + 0.14186178976691055im
  0.9534240740436073 - 0.09468674639929728im
  0.9534240740436073 + 0.09468674639929728im
  1.0150172143157419 - 0.049117188187440805im
  1.0150172143157419 + 0.049117188187440805im
   1.035151782134667 + 0.0im

# Selección de las raíces reales
realRoots = real(roots[imag(roots) .== 0.0])

4-element Vector{Float64}:
 -1.0527044089838609
  0.27325300053763296
  0.7588894668619902
```

Figura 23: Ilustración en Julia de uso de matriz colega (4/7).

```
  1.035151782134667

scatter!(p, realRoots, interpolator.(realRoots), color=3, markershape=:star8, label="")
```




```
function complexInterpolator(x::ComplexF64) # Variante compleja de la función polinomio de
interpolación
        bOlder = 0.0;
        bOld = 0.0;
        for k=N:-1:1
                bNew = vũ[k+1] + 2*x*bOld - bOlder;
                bOlder = bOld;
                bOld = bNew:
```

Figura 24: Ilustración en Julia de uso de matriz colega (5/7).

```
        end
        vũ[1] + x*bOld - bOlder
end

# Las raíces obtenidas efectivamente son raíces
complexInterpolator.(roots)

30-element Vector{ComplexF64}:
   1.971756091734278e-13 + 0.0im
  -1.971756091734278e-13 - 3.361755318564974e-13im
  -1.971756091734278e-13 + 3.361755318564974e-13im
  -1.396660564978447e-13 + 2.220446049250313e-13im
  -1.396660564978447e-13 - 2.220446049250313e-13im
     1.0635936575909e-13 - 3.530509218307998e-14im
     1.0635936575909e-13 + 3.530509218307998e-14im
  -9.381384558082573e-15 + 4.418687638008123e-14im
  -9.381384558082573e-15 - 4.418687638008123e-14im
  -3.930189507173054e-14 + 2.6645352591003757e-14im
                         ⋮
  -5.440092820663267e-15 + 1.008915173628111e-14im
  -6.661338147750939e-16 + 0.0im
  -1.099120794378905e-14 - 3.552713678800501e-15im
  -1.099120794378905e-14 + 3.552713678800501e-15im
  -9.103828801926284e-15 - 2.930988785010413e-14im
  -9.103828801926284e-15 + 2.930988785010413e-14im
 -1.5232259897857148e-13 - 9.015010959956271e-14im
 -1.523225989785714e-13 + 9.015010959956271e-14im
```

Figura 25: Ilustración en Julia de uso de matriz colega (6/7).

```
  -6.616929226765933e-14 + 0.0im

# Curvas de nivel de (el logaritmo del valor absoluto de) el polinomio de interpolación junto a sus raíces
realRange = extrema(real(roots))
imagRange = extrema(imag(roots))
inflateFactor = 1.025;
realRange2 = (realRange[1]+realRange[2])/2 .+ 0.5*inflateFactor*(realRange[2]-realRange[1]).*(-1,1)
imagRange2 = (imagRange[1]+imagRange[2])/2 .+ 0.5*inflateFactor*(imagRange[2]-imagRange[1]).*(-1,1)
xx = range(realRange2[1], realRange2[2], length=1024)
yy = range(imagRange2[1], imagRange2[2], length=512)
q = contour(xx, yy, (x,y) -> log(abs(complexInterpolator(x+im*y))),
size=(600,600*(imagRange[2]-imagRange[1])/(realRange[2]-realRange[1])), title="log( | I_N(u)(z) | )",
titlefontsize=8, levels=30, fill=true)
scatter!(q, real(roots), imag(roots), color=3, markershape=:star8, label="")
```

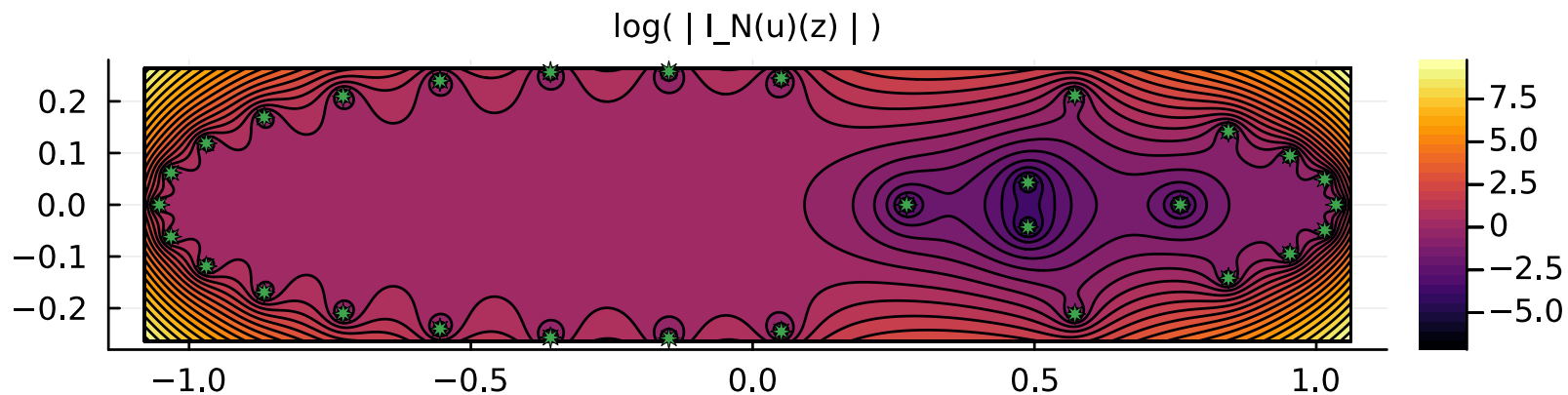


Figura 26: Ilustración en Julia de uso de matriz colega (7/7).

Sea ahora $p \in \mathbb{P}_N$ un polinomio cuyos coeficientes p_k en su expansión de Fourier–Chebyshev $\sum_{k=0}^{N} p_k T_k$ son conocidos. Nuestro siguiente propósito es **evaluar** al polinomio p en un punto arbitrario x en forma eficiente.

Para ello, tomemos otra vez más algunas instancias de la recurrencia de tres términos para polinomios de Chebyshev (Observación 24), esta vez organizada de otra forma. Dado $N \geq 1$,

$$\begin{aligned} T_0(x) &= 1, \\ -1x\,T_0(x) + T_1(x) &= 0, \\ T_0(x) - 2x\,T_1(x) + T_2(x) &= 0, \\ \vdots &= \vdots \\ T_{N-2}(x) - 2x\,T_{N-1}(x) + T_N(x) &= 0. \end{aligned}$$

Esto es,

$$\underbrace{\begin{pmatrix} 1 & & & & \\ -1x & 1 & & & \\ 1 & -2x & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2x & 1 \end{pmatrix}}_{:= A(x)} \underbrace{\begin{pmatrix} T_0(x) \\ T_1(x) \\ T_2(x) \\ \vdots \\ T_N(x) \end{pmatrix}}_{:= \check{T}(x)} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \qquad (50)$$

Sea $b(x) = \begin{pmatrix} b_0(x) & \cdots & b_N(x) \end{pmatrix}^{\mathsf{T}}$ la solución del sistema

$$\underbrace{\begin{pmatrix} 1 & -1x & 1 & & & \\ & 1 & -2x & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2x & 1 \\ & & & & 1 & -2x \\ & & & & & 1 \end{pmatrix}}_{= A(x)^{\mathsf{T}}} \underbrace{\begin{pmatrix} b_0(x) \\ b_1(x) \\ \vdots \\ b_{N-2}(x) \\ b_{N-1}(x) \\ b_N(x) \end{pmatrix}}_{= b(x)} = \underbrace{\begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_{N-2} \\ p_{N-1} \\ p_N \end{pmatrix}}_{:= \tilde{p}}. \tag{51}$$

Entonces,

$$p(x) = \sum_{k=0}^{N} p_k T_k(x) = \tilde{p} \cdot \check{T}(x) \overset{(51)}{=} \left(A(x)^{\mathsf{T}} b(x) \right) \cdot \check{T}(x)$$

$$= b(x) \cdot \left(A(x) \check{T}(x) \right) \overset{(50)}{=} b(x) \cdot e_0 = b_0(x).$$

Así, para evaluar p en x basta resolver el sistema triangular superior con solamente tres diagonales posiblemente no nulas (51) por sustitución regresiva y extraer la primera componente de la solución. Lo encapsulamos en el **Algoritmo de Clenshaw:**

- Entradas: Vector $(p_0, \ldots, p_N)$, escalar x.
- $b_{N+2} \leftarrow 0$, $b_{N+1} \leftarrow 0$.
- Para $k = N, N-1, \ldots, 1$:
  - $b_k \leftarrow p_k + 2x p_{k+1} - p_{k+2}$.
- Devolver $b_0 \leftarrow p_0 + 1x p_1 - p_2$.

El Algoritmo de Clenshaw cuesta $\mathcal{O}(N)$ flops para un escalar x.

Con el *algoritmo* DCT se puede evaluar a un polinomio de grado N a partir de sus coeficientes de Fourier–Chebyshev en $N+1$ nodos de Gauss–Lobatto–Chebyshev en $\mathcal{O}(N \log(N))$ flops. Al algoritmo de Clenshaw esto mismo le cuesta $\mathcal{O}(N^2)$ flops, lo que es más costoso, pero otorga libertad sobre dónde evaluar.

- Este Algoritmo de Clenshaw es el procedimiento, que en aquel entonces tenía la explicación pendiente, que aparece en la Figura 20.
- El Algoritmo de Clenshaw se puede adaptar para evaluar polinomios a partir de sus coeficientes respecto a cualquier base que satisfaga una recurrencia de tres términos.

Si se conoce a un polinomio por sus valores en los nodos de Gauss–Lobatto–Chebyshev (valores *nodales*) y se desea evaluarlo en un punto arbitrario x, se puede aplicar la *función* DCT (Definición 43) (¡usar el *algoritmo* DCT que cuesta solamente $\mathcal{O}(N \log(N))$ flops!), obtener sus coeficientes de Fourier–Chebyshev ~~discretos~~ y aplicar el Algoritmo de Clenshaw que cuesta $\mathcal{O}(N)$ flops. El costo total serían $\mathcal{O}(N \log(N))$ flops.

Sin embargo, existe una alternativa que evita pasar por una representación modal.

Sean $(x_j)_{j=0}^N$ los nodos de la cuadratura de Gauss–Lobatto–Chebyshev de $N+1$ puntos ($x_j = \cos(j\pi/N)$). Si $p \in \mathbb{P}_N$, podemos expresarlo en su forma de Lagrange

$$p(x) = \sum_{j=0}^{N} p(x_j)\,\psi_j(x), \quad \text{donde} \quad \psi_j(x) := \frac{\prod_{\substack{i=0 \\ i\neq j}}^{N}(x - x_i)}{\prod_{\substack{i=0 \\ i\neq j}}^{N}(x_j - x_i)}.$$

El numerador en la definición de la función delta discreta $\psi_j(x)$ puede expresarse como $l(x)/(x - x_j)$, donde

$$l(x) := \prod_{i=0}^{N}(x - x_i).$$

Definiendo a los *pesos baricéntricos* $\check{w}_j$ de acuerdo a

$$(\forall\, j \in \{0, \ldots, N\}) \quad \check{w}_j := 1 \Bigg/ \prod_{\substack{i=0 \\ i\neq j}}^{N}(x_j - x_i)\ ,$$

obtenemos la *primera forma de la fórmula baricéntrica*:

$$(\forall\, p \in \mathbb{P}_N) \quad p(x) = l(x) \sum_{j=0}^{N} \frac{\check{w}_j}{x - x_j}\, p(x_j).$$

Dividiendo esta representación de p por la correspondiente representación de $1 \in \mathbb{P}_N$ se obtiene la *segunda forma de la fórmula baricéntrica* o ***la*** *fórmula baricéntrica*:

$$(\forall\, p \in \mathbb{P}_N) \quad p(x) = \frac{\displaystyle\sum_{j=0}^{N} \frac{\check{w}_j}{x - x_j}\, p(x_j)}{\displaystyle\sum_{j=0}^{N} \frac{\check{w}_j}{x - x_j}}. \tag{52}$$

Notemos que

$$l'(x_j) = l'(x)\big|_{x=x_j} = \sum_{k=0}^{N} \prod_{\substack{i=0 \\ i\neq k}}^{N} (x - x_i)\Big|_{x=x_j} = \prod_{\substack{i=0 \\ i\neq j}}^{N} (x_j - x_i),$$

por lo que

$$\check{w}_j = 1/l'(x_j).$$

La verdad es que, hasta aquí, todo esto de las fórmulas baricéntricas vale para cualquier colección de $N+1$ nodos distintos. Pero ahora notamos que, en el caso particular de los nodos de Gauss–Lobatto–Chebyshev,

$$l(x) = 2^{-N}(T_{N+1} - T_{N-1});$$

en efecto, ambos polinomios son del mismo grado, tienen a los $x_j = \cos(j\pi/N)$ por raíces y de la recurrencia de tres términos para polinomios de Chebyshev (Observación 24) se infiere que el coeficiente principal de T_{N+1} en su expansión por monomios es 2^N.

Empleando que $T_k' = k\, U_{k-1}$ (cf. (26)) y algunas identidades trigonométricas obtenemos que

$$l'(x) = 2^{1-N}\left(N\, T_N(x) + U_{N-1}(x)\, x\right).$$

Así, $l'(x_j) = 2^{1-N} N (-1)^j \bar{c}_j$ y consiguientemente,

$$\check{w}_j = \frac{2^{N-1}}{N}\frac{(-1)^j}{\bar{c}_j}.$$

La segunda forma de la fórmula baricéntrica, cancelando como podemos los factores comunes entre todos los pesos $\check{w}_j$, queda

$$(\forall\, p \in \mathbb{P}_N) \quad p(x) = \frac{\displaystyle\sum_{j=0}^{N} \frac{(-1)^j/\bar{c}_j}{x - x_j}\, p(x_j)}{\displaystyle\sum_{j=0}^{N} \frac{(-1)^j/\bar{c}_j}{x - x_j}}. \tag{53}$$

- Aplicar la fórmula baricéntrica (53) para evaluar un polinomio cuyos valores nodales son conocidos cuesta $\mathcal{O}(N)$ flops.
- Estas fórmulas se comportan excelente incluso cerca de los nodos.
- En este caso de nodos de cuadratura de Gauss–Lobatto–Chebyshev, los pesos $\check{w}_j$ no discrepan en magnitud más que por un factor 2. Los pesos correspondientes en el caso de nodos equiespaciados también son conocidos en forma explícita y discrepan entre sí hasta por un factor de $\binom{N}{N/2}$, lo que es terrible.

Bases, expansiones y operaciones rápidas

Algoritmos FFT y DCT

La DFT puede ser computada mediante la *Transformada rápida de Fourier* (FFT por sus siglas en inglés) que es un algoritmo cuya forma más sencilla calcula la DFT en $5N\log_2(N) - 6N$ operaciones de punto flotante con números reales (flops) en lugar de las $2N^2 - N$ operaciones de punto flotante con números complejos que implicaría una DFT calculada mediante una multiplicación matriz-vector.

Recordemos a qué le llamamos DFT en la Definición 9: Es aquella operación que a un vector $u = (u_j)_{j=0}^{N-1} \in \mathbb{C}^N$ asocia el vector $\tilde{u} = (\tilde{u}_k)_{k=-N/2}^{N/2-1} \in \mathbb{C}^N$ definido por

$$(\forall\, k \in \{-N/2, \ldots, N/2-1\}) \qquad \tilde{u}_k = \frac{1}{N}\sum_{j=0}^{N-1} u_j\, e^{-\imath k 2\pi j/N}.$$

Deduciremos un algoritmo en el espíritu del descrito por Cooley y Tukey en 1965 para efectuar este cálculo que es una variante de las muchas que hay de la FFT; será aparente que emplea $\mathcal{O}(N \log_2(N))$ operaciones aritméticas si N es una potencia de 2.

Primero, notemos que podemos ignorar el $\frac{1}{N}$ que está adelante de la suma porque incorporarlo emplea N multiplicaciones complejas. También, debido al hecho de que

$$e^{-\imath k 2\pi j/N} = e^{-\imath (k+N) 2\pi j/N}$$

podemos, a costa de un reordenamiento no emplea flops, hacer correr a k en $\{0, \ldots, N-1\}$ al igual que j, y el resultado será un reordenamiento de $\tilde{u}$; entonces, podemos concentrarnos en el cómputo del vector $v = (v_k)_{k=0}^{N-1}$, definido por

$$(\forall\, k \in \{0, \ldots, N-1\}) \qquad v_k = \sum_{j=0}^{N-1} u_j\, e^{-\imath k 2\pi j/N}. \qquad (54)$$

Esto es lo que típicamente computan los algoritmos de tipo FFT que se hallan en las librerías de *software*; en particular, es lo que que computan las funciones `FFTW.fft` de Julia, `numpy.fft.fft` de Python y `fft` de Matlab. Ahora,

$$
\begin{aligned}
v_k &= \sum_{m=0}^{N/2-1} u_{2m}\, e^{-\imath k 2\pi (2m)/N} + \sum_{m=0}^{N/2-1} u_{2m+1}\, e^{-\imath k 2\pi (2m+1)/N} \\
&= \underbrace{\sum_{m=0}^{N/2-1} u_{2m}\, e^{-\imath k 2\pi m/(N/2)}}_{:=E_k} + e^{-\imath k 2\pi/N} \underbrace{\sum_{m=0}^{N/2-1} u_{2m+1}\, e^{-\imath k 2\pi m/(N/2)}}_{:=O_k},
\end{aligned}
$$

donde E_k es la DFT del vector de entradas de índice par de u y O_k es la DFT del vector de entradas de índice impar de u. Ahora, como los E_k son N en total, pero el vector que participa en su definición es de largo $N/2$ solamente, debe haber alguna

redundancia; en efecto, $E_{k+N/2} = E_k$, por lo que solo es necesario computar (usando una DFT para vectores de largo $N/2$) los E_k para $k \in \{0, \ldots, N/2 - 1\}$. Similarmente, $O_{k+N/2} = O_k$. Agregando la observación de que

$$e^{-\imath(k+N/2)2\pi/N} = e^{-\imath\pi} e^{-\imath k 2\pi/N} = -e^{-\imath k 2\pi/N},$$

se tiene que

$$v_k = \begin{cases} E_k + e^{-\imath k 2\pi/N} O_k & \text{si } 0 \leq k \leq N/2 - 1, \\ E_{k-N/2} - e^{-\imath(k-N/2)2\pi/N} O_{k-N/2} & \text{si } N/2 \leq k \leq N - 1. \end{cases}$$

Con este procedimiento tenemos que una DFT de largo N se puede calcular al costo de dos DFT de largo $N/2$, más una cantidad de sumas y multiplicaciones proporcional a N (las exponenciales

complejas pueden computarse calculando $e^{-\imath 2\pi/N}$ al principio y luego todas las otras son potencias de ésta). Lo expresamos como

$$\text{costo}(N) = 2\,\text{costo}(N/2) + C\,N,$$

donde, obviamente, costo(N) es el costo de la DFT de tamaño N usando este procedimiento.
Denotemos por C_0 el costo la DFT de largo $1 = 2^0$. Es fácil comprobar por inducción entonces que

$$(\forall\, p \in \mathbb{N}_0) \quad \text{costo}(2^p) = C\,p\,2^p + C_0\,2^p.$$

Si $N = 2^p$, entonces
$\text{costo}(N) = C\,N\,\log_2(N) + C_0\,N = \mathcal{O}(N \log_2(N))$.

Cabe mencionar que implementaciones de producción de este algoritmo FFT no suelen emplear recursiones explícitas (en general, se prefiere evitarlas). Es necesario enfatizar que existen algoritmos

tipo FFT para calcular la DFT para N que no es potencia de 2; suelen tener costos del mismo orden $\mathcal{O}(N \log_2(N))$. La verdad, el estudio y la implementación de algoritmos de tipo FFT son temas de estudio enormes. Sin ir más lejos, no hemos mencionado aspectos de costos de comunicación, paralelización/*threading*, estabilidad numérica, entre otros, que son esenciales a la hora de producir algoritmos de FFT competitivos.

Dado que la transformada discreta de Fourier y su inversa tienen formas muy parecidas, no es sorpresa que para esta última también existan algoritmos del tipo descrito más arriba, usualmente llamados IFFT por sus siglas en inglés, que cuestan $\mathcal{O}(N \log_2(N))$, donde N es el tamaño del vector a transformar. Normalmente se

implementa la inversa de (54); esto es, dado un vector $(v_k)_{k=0}^{N-1}$ se obtiene el vector $(u_j)_{j=0}^{N-1}$ definido por

$$(\forall\, j \in \{0, \ldots, N-1\}) \qquad u_j = \frac{1}{N} \sum_{k=0}^{N-1} v_k \, e^{\imath j 2\pi k/N}. \tag{55}$$

Esto hacen, por ejemplo, `FFTW.ifft` de Julia, `numpy.fft.ifft` de Python y `ifft` de Matlab. Al igual que (54) con respecto a (5), la transformación (55) involucra un reordenamiento (pero ahora de la entrada; no de la salida) y un re-escalamiento con respecto a (6). Las librerías de *software* suelen venir con los reordenamientos implementados (por ejemplo, `FFTW.fftshift` y `FFTW.ifftshift` en Julia; `numpy.fft.fftshift` y `numpy.fft.ifftshift` en Python; `fftshift` e `ifftshift` en Matlab).

En la Definición 43 definimos formalmente la transformación que a un vector $u \in \mathbb{R}^{N+1}$ asocia el vector $\tilde{u} \in \mathbb{R}^{N+1}$ definido por

$$(\forall\, k \in \{0, \ldots, N\}) \quad \tilde{u}_k = \frac{2}{N\bar{c}_k} \sum_{j=0}^{N} \frac{1}{\bar{c}_j} \cos\left(\frac{\pi jk}{N}\right) u_j.$$

También comentamos en la Observación 44 que esencialmente esta transformación se conoce como la *transformación de coseno discreta* (DCT por sus siglas en inglés) de tipo I.

Exhibiremos una manera de calcular dos DCT reales de tamaño $N+1$ usando una IDFT/IFFT de largo $2N$ de la forma (55). Sean $u^{(0)} \in \mathbb{R}^{N+1}$ y $u^{(1)} \in \mathbb{R}^{N+1}$. Definimos $v \in \mathbb{C}^{2N}$ por

$$(\forall\, l \in \{0, \ldots, 2N-1\}) \quad v_l = \begin{cases} u_l^{(0)} + \imath\, u_l^{(1)} & \text{si } 0 \leq l \leq N, \\ v_{2N-l} & \text{si } N+1 \leq l \leq 2N-1. \end{cases}$$

Definimos a $u \in \mathbb{C}^{2N}$ como la IDFT (del tipo (55)) del vector v definido anteriormente:

$$(\forall\, j \in \{0, \ldots, 2N-1\}) \qquad u_j = \frac{1}{2N} \sum_{l=0}^{2N-1} v_l \, e^{\imath j 2\pi l/(2N)}.$$

También definimos a $\tilde{v} \in \mathbb{C}^{N+1}$ como la DCT del vector complejo $(v_l)_{l=0}^{N}$ (esto es, el vector compuesto por las primeras $N+1$ entradas de v):

$$(\forall\, k \in \{0, \ldots, N\}) \quad \tilde{v}_k = \frac{2}{N\bar{c}_k} \sum_{l=0}^{N} \frac{1}{\bar{c}_l} \cos\left(\frac{\pi l k}{N}\right) v_l.$$

Usando identidades trigonométricas es fácil demostrar que

$$(\forall\, k \in \{0, \ldots, N\}) \quad \tilde{v}_k = \frac{2}{\bar{c}_k} u_k.$$

Como la DCT es lineal, podemos extraer las DCT de $u^{(0)}$ y de $u^{(1)}$ de las partes real e imaginaria de las primeras $N+1$ componentes del lado derecho de la expresión anterior.

Cabe notar que como la DCT tiene casi la misma forma que su inversa IDCT, no suele ser necesario distinguir entre la DCT-I y su inversa. El procedimiento detallado anteriormente, con mínimas modificaciones, sirve para calcularla.

El procedimiento aquí sugerido emplea $\mathcal{O}(N\log_2(N))$ operaciones aritméticas. Sin embargo, para obtener dos DCT reales de $2N+2$ variables de salida en total, estamos empleando una IFFT de tamaño $2N$ que involucra $4N$ variables reales de salida, lo que sugiere que estamos empleando aproximadamente el doble de operaciones aritméticas de lo que es estrictamente necesario en general. Hay dos cosas que se pueden hacer:

- Existe una manera de emplear una DFT/FFT de tamaño N para calcular las dos DCT reales. Los detalles están en el apéndice B de Canuto, Hussaini, Quarteroni & Zang: *Spectral methods* (2006).
- Existen rutinas tipo FFT especializadas para transformaciones como nuestras DCT e IDCT; en Julia está `FFTW.r2r`; en Python está `scipy.fftpack.dct`. Matlab tiene la función `dct` en su *Signal Processing Toolbox*.