Programación: aproximación de la transformada de Fourier en la recta real

Objetivos. Programar una función que calcule de manera aproximada los valores de $\widehat{f}$ en ciertos nodos equidistantes ξ_j usando los valores de f en ciertos nodos equidistantes x_k y aplicando la transformada finita de Fourier. Utilicemos el lenguaje de Matlab/Octave.

Requisitos. Integración numérica, transformada finita de Fourier, programación con arreglos.

1. Integración numérica con rectángulos izquierdos, repaso. Sea f una función continua en $[a, b]$ y sea $n \in \mathbb{N}$. Pongamos $h = (b-a)/n$ y denotemos por x_k a los nodos equidistantes $x_k := a + kh$, donde $k \in \{0, \dots, n\}$. Entonces

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{h} \sum_{k=0}^{n-1} f(x_k).$$

2. Transformada finita de Fourier, repaso. Sea $n \in \mathbb{N}$. Pongamos $\omega_n = \mathrm{e}^{-\frac{2\pi\,\mathrm{i}}{n}}$ y definimos la matriz cuadrada F_n mediante la regla

$$F_n = \left[\omega_n^{jk}\right]_{j,k=0}^{n-1}.$$

La matriz F_n se llama la *matriz de Fourier* de orden n, y el operador lineal $x \mapsto F_n x$ se llama la *transformada finita de Fourier* de orden n o la *transformada discreta cíclica de Fourier* de orden n. Se sabe que

$$\frac{1}{n} F_n^* F_n = I_n.$$

3. Aproximación de la transformada de Fourier en $\mathbb{R}$ por medio de la transformada finita de Fourier, idea. Sea m un número natural par. Pongamos $n = m^2$,

$$x_k = -\frac{m}{2} + \frac{k}{m} \quad (k \in \{0, \dots, n-1\}), \qquad \xi_j = -\frac{m}{2} + \frac{j}{m} \quad (j \in \{0, \dots, n-1\}). \tag{1}$$

Entonces

$$\widehat{f}(\xi) = \int_{\mathbb{R}} f(x)\,\mathrm{e}^{-2\pi\,\mathrm{i}\,x\xi}\,\mathrm{d}x \approx \int_{-m/2}^{m/2} f(x)\,\mathrm{e}^{-2\pi\,\mathrm{i}\,x\xi}\,\mathrm{d}x \approx \frac{1}{m} \sum_{k=0}^{n-1} f(x_k)\,\mathrm{e}^{-2\pi\,\mathrm{i}\,x_k\xi},$$

y

$$\widehat{f}(\xi_j) \approx v_j := \frac{1}{m} \sum_{k=0}^{n-1} f(x_k)\,\mathrm{e}^{-2\pi\,\mathrm{i}\,x_k\xi_j}\,.$$

4. Aproximación de la transformada de Fourier en $\mathbb{R}$ por medio de la transformada finita de Fourier, deducción de las fórmulas finales. Usando la notación y los convenios anteriores simplifique la expresión

$$-2\pi\,\mathrm{i}\,x_k\xi_j.$$

Muestre que

$$\mathrm{e}^{-2\pi\,\mathrm{i}\,x_k\xi_j} = (-1)^j(-1)^k\omega_n^{jk}.$$

Muestre que

$$v_j = \frac{1}{m}(-1)^j\sum_{k=0}^{n-1} f(x_k)(-1)^k\omega_n^{jk}.$$

Denotemos por s al vector $\left[(-1)^j\right]_{j=0}^{n-1}$ y por $\odot$ a la operación de multiplicación por componentes. Pongamos

$$u = \left[(-1)^k f(x_k)\right]_{k=0}^{n-1} = s\odot\left[f(x_k)\right]_{k=0}^{n-1}. \tag{2}$$

Muestre que

$$v_j = \frac{1}{m}(-1)^j(F_n u)_j. \tag{3}$$

En otras palabras,

$$v = \frac{1}{m}s\odot(F_n u).$$

5. Construcción de progresiones aritméticas en Matlab/Octave, repaso. Para repasar algunos elementos de la sintaxis, se recomienda ejecutar los siguientes comandos uno por uno en el intérprete de Matlab (o de GNU Octave).

```
1 : 5
(1 : 5)'
(0 : 4)'
(0 : 4)' * 0.1
repmat(2, 5, 1)
repmat(2, 5, 1) + (0 : 4)' * 0.1
```

6. Algunas operaciones por componentes, repaso.

```
[2; 3; 4] .* [11; 12; 13]
[2; 3; 4] .^ 2
repmat(-1, 5, 1)
s = repmat(-1, 5, 1) .^ ((0 : 4)')
```

7. Evaluación de una función en varios puntos, repaso.

```
ppp = (0 : 4)' * 0.1
f1 = @ (t) t .^ 2
vvv = f1(ppp)
norm(vvv, inf)
```

8. Transformada finita de Fourier en Matlab/Octave, repaso. Dado un vector a en $\mathbb{C}^n$, el producto $F_n a$ se puede calcular con el comando `fft(a)`. Dentro de la función `fft` está realizado un algoritmo muy eficiente llamado la *transformada rápida de Fourier* (fast Fourier transform). Ahora no vamos a estudiar este algoritmo, solamente comprobemos que `fft(a)` coincide con $F_n a$.

```
a = rand(5, 1)
b = fft(a)
norm(a)
norm(b) / sqrt(5)
omega = exp(- 2 * pi * 1i / 5)
repmat(omega, 5, 5)
ind = (0 : 4)' * (0 : 4)
F = repmat(omega, 5, 5) .^ ind
c = F * a
norm(b - c)
```

9. Programación: función que aproximada la transformada de Fourier. En este ejercicio y en el siguiente, suponemos que m es un número natural par, pongamos $n = m^2$ y definimos los vectores $x \in \mathbb{R}^n$, $\xi \in \mathbb{R}^n$, $u, v \in \mathbb{C}^n$ por las fórmulas (1), (2) y (3). Se recomienda guardar el siguiente código en el archivo `fourier_approx.m`, cambiando ??? por expresiones correctas.

```
function [v] = fourier_approx(f, m),
   n = m ^ 2;
   ind = (0 : n - 1)';
   x = repmat(- m / 2, ???, 1) + ind / ???;
   s = repmat(-1, ???, 1) .^ ???;
   u = f(x) .* ???;
   v = ???. * fft(???) / ???;
end
```

10. Programación: prueba con la función gaussiana. Hagamos una comprobación de nuestra función `fourier_approx` usando como f la función gaussiana

$$f(x) = \mathrm{e}^{-\pi x^2}\,.$$

Denotamos $\widehat{f}$ por g. En este ejemplo g coincide con f. Se recomienda guardar el siguiente código en el archivo `test_fourier_approx_1.m`, cambiando ??? por expresiones correctas.

```
function [er] = test_fourier_approx_1(m),
   f = @ (x) exp(-pi * (x .^ 2));
   g = f;
   vapprox = fourier_approx(f, m);
   n = m ^ 2;
   xi = repmat(- m / 2, ???, 1) + (0 : ???)' / ???;
   vexact = g(???);
   er = norm(??? - ???, inf);
end
```

Luego en el intérprete se pueden ejecutar los comandos (como $n = m^2$, elegimos los valores de m no demasiado grandes):

```
test_fourier_approx_1(16)
test_fourier_approx_1(128);
```

11. Programación: prueba con una función racional. Hagamos otra comprobación de nuestra función `fourier_approx` usando como f el siguiente caso particular del núcleo de Poisson:

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

Denotemos $\widehat{f}$ por g. En este ejemplo

$$g(\xi) = \mathrm{e}^{-2\pi|\xi|}.$$

Se recomienda guardar el siguiente código en el archivo `test_fourier_approx_2.m`, cambiando ??? por expresiones correctas.

```
function [er] = test_fourier_approx_2(m),
   f = @ (x) ???;
   g = @ (xi) ???;
   vapprox = fourier_approx(???, ???);
   n = ???;
   xi = ???;
   vexact = g(???);
   er = ???;
end
```