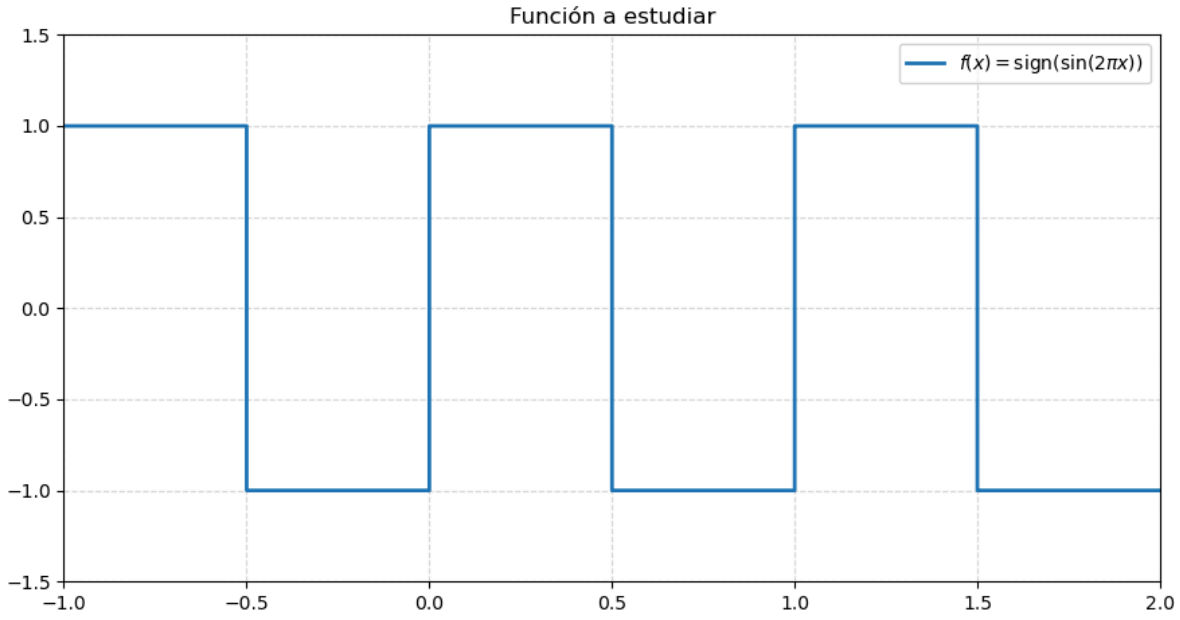
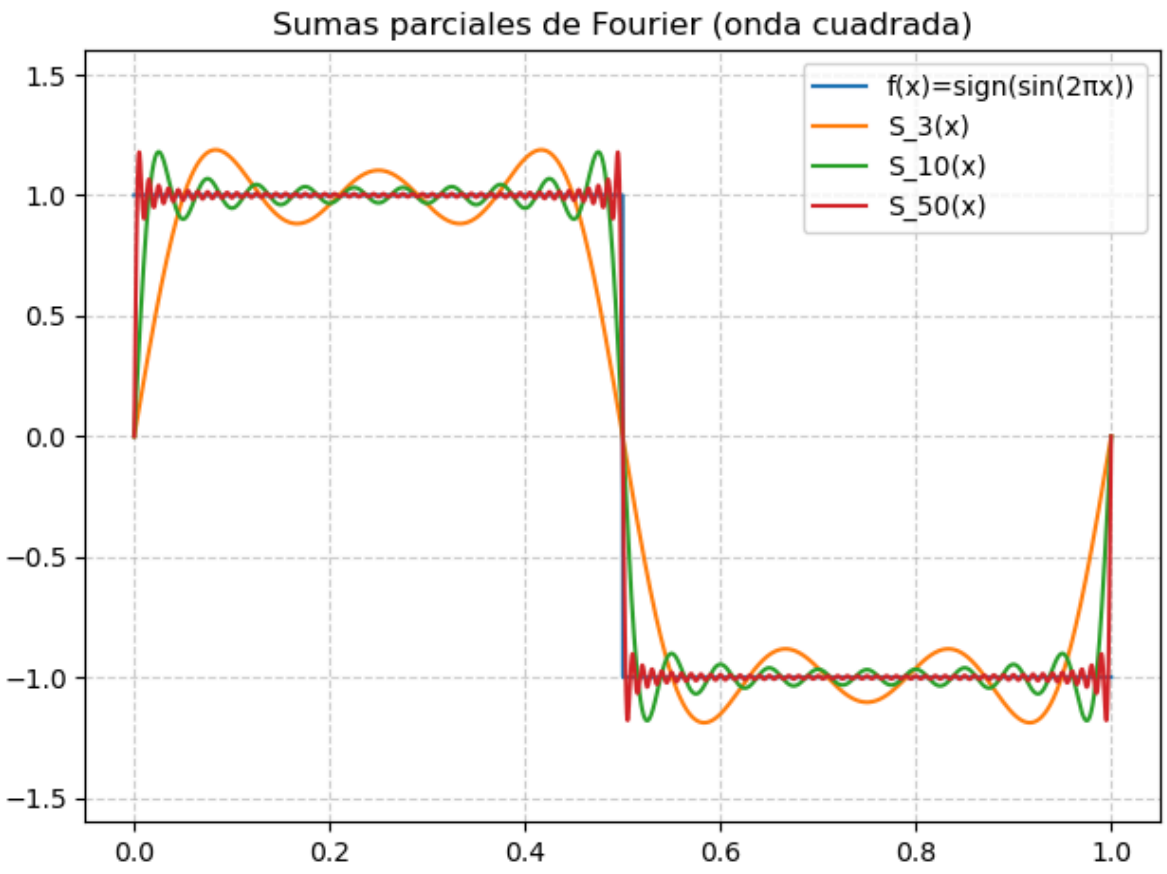
Fenómeno de Gibbs: comparación entre sumas parciales y Cesàro Autores: Paula Blanco, Covadonga Díaz, Claudia Pozurama Asignatura: EDP Curso 2025/2026

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt

# f(x) = sign(sin(2πx))
def f_square_pm1(x):
    return np.where(np.sin(2*np.pi*x) >= 0, 1.0, -1.0)


x_min, x_max = -1.0, 2.0
x = np.linspace(x_min, x_max, 8000)
y = f_square_pm1(x)

plt.figure(figsize=(9, 4.8))

# Función en azul
plt.step(x, y, where="post",
         linewidth=2,
         label=r"$f(x)=\mathrm{sign}(\sin(2\pi x))$")

plt.xlim(x_min, x_max)
plt.ylim(-1.5, 1.5)

plt.grid(True, linestyle="--", alpha=0.5)

plt.title("Función a estudiar")
plt.legend(loc="upper right")

plt.tight_layout()
plt.savefig("funcion_periodica.png", dpi=400, bbox_inches="tight")
plt.savefig("funcion_periodica.pdf", bbox_inches="tight")
plt.show()
```



In [3]:
```
import numpy as np
import matplotlib.pyplot as plt

# 1) Función: onda cuadrada
def f_square_pm1(x):
    y = np.sin(2*np.pi*x)
    return np.where(y >= 0, 1.0, -1.0)
```

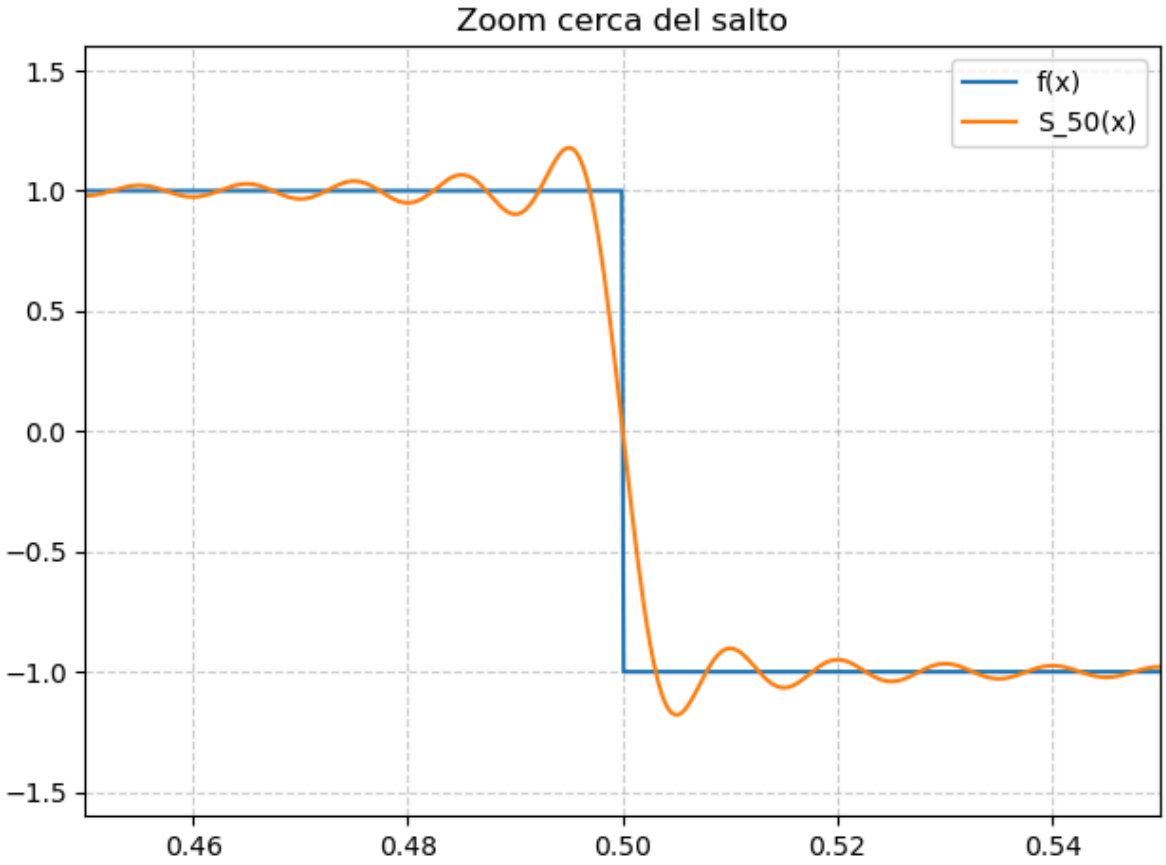
```
# 2) Suma parcial clásica S_N
def S_N(x, N):
    m = np.arange(1, N+1)
    k = 2*m - 1
    return (4/np.pi) * np.sum((1/k)[:,None] * np.sin(2*np.pi * k[:,None] * x[None,:

#3) Suma de Cesàro sigma_N
def sigma_N(x, N):
    acc = np.zeros_like(x, dtype=float)
    for n in range(1, N+1):
        acc += S_N(x, n)
    return acc / (N+1)


x = np.linspace(0, 1, 6000)
f = f_square_pm1(x)


plt.figure()
plt.plot(x, f, label="f(x)=sign(sin(2πx))")
for N in [3, 10, 50]:
    plt.plot(x, S_N(x, N), label=f"S_{N}(x)")
plt.ylim(-1.6, 1.6)
plt.title("Sumas parciales de Fourier (onda cuadrada)")
plt.legend()
plt.grid(True, linestyle="--", alpha=0.6)
plt.tight_layout()
plt.show()
```



In [4]:

```
# ---------- FIGURA: ZOOM CON N = 50 ----------
plt.figure()

N = 50
x = np.linspace(0, 1, 6000)
f = f_square_pm1(x)
```

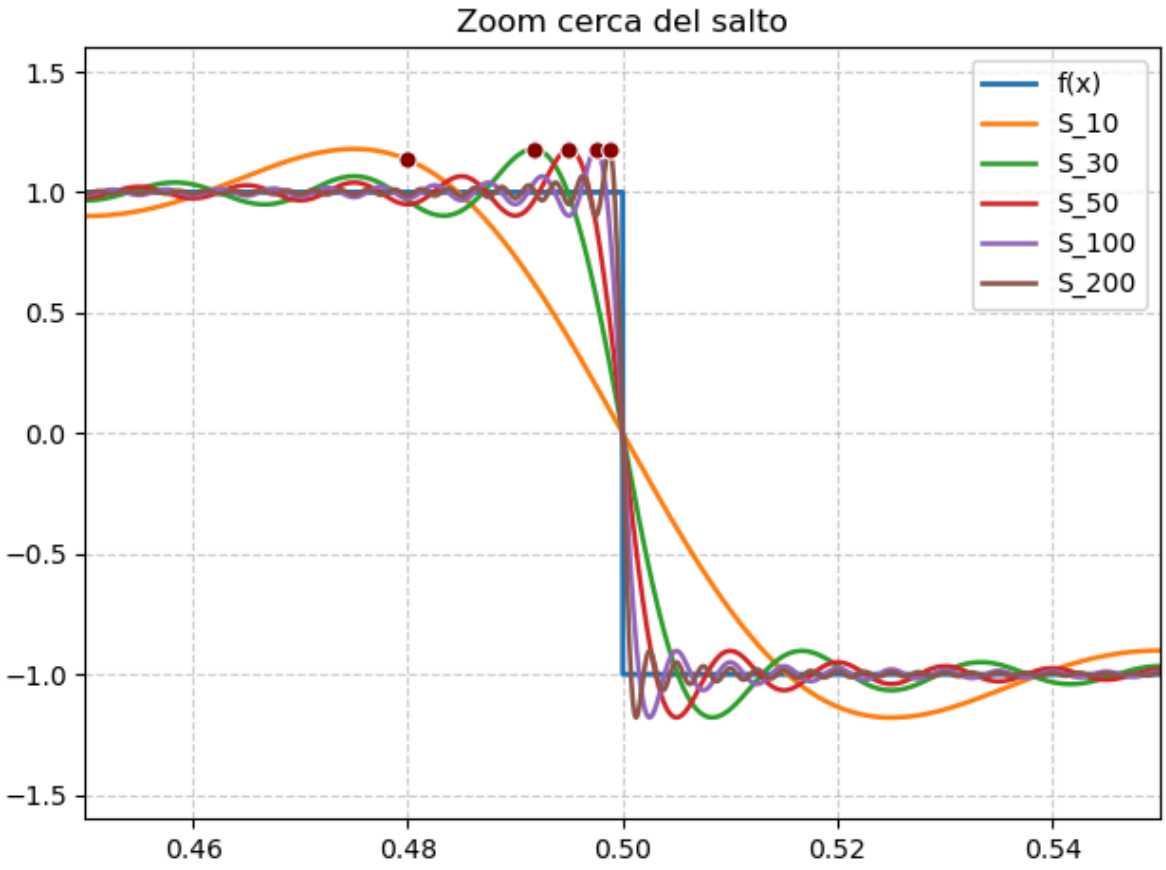
```
plt.plot(x, f, label="f(x)")
plt.plot(x, S_N(x, N), label=f"S_{N}(x)")

plt.xlim(0.45, 0.55)     # Zoom alrededor del salto
plt.ylim(-1.6, 1.6)

plt.title("Zoom cerca del salto")
plt.legend()
plt.grid(True, linestyle="--", alpha=0.6)
plt.tight_layout()
plt.show()
```



In [5]:

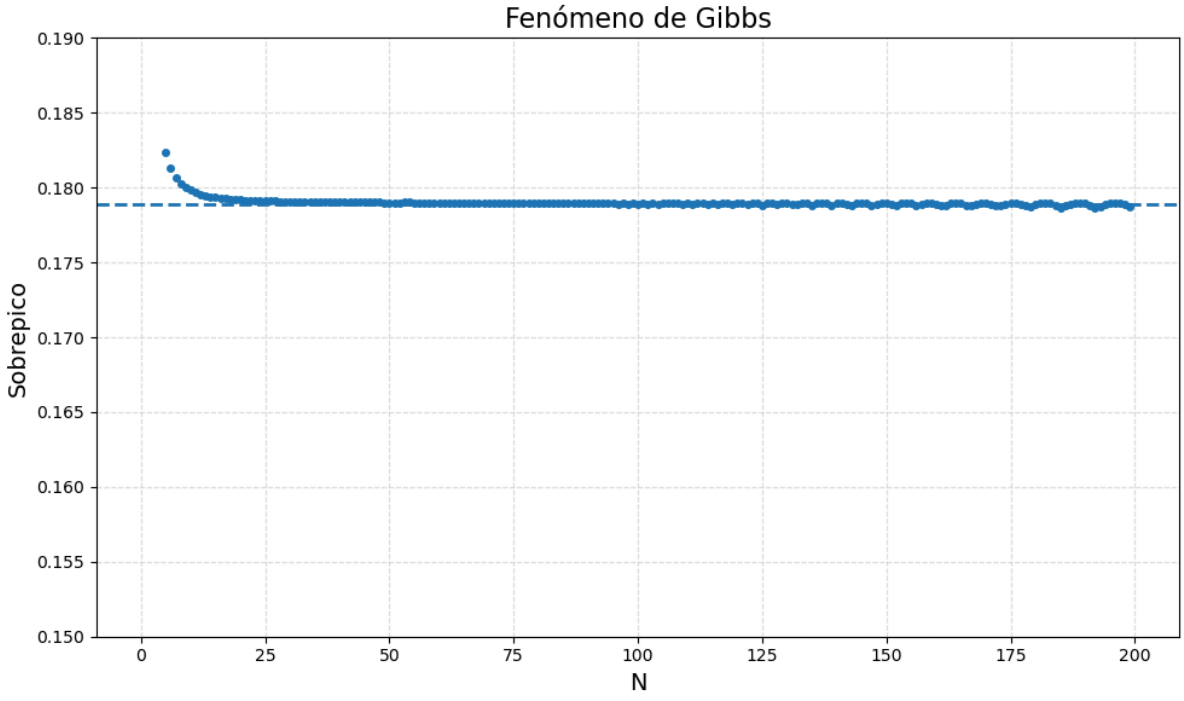
```
import numpy as np
import matplotlib.pyplot as plt

def f_square_pm1(x):
    return np.where(np.sin(2*np.pi*x) >= 0, 1.0, -1.0)

def S_N(x, N):
    m = np.arange(1, N+1)
    k = 2*m - 1
    return (4/np.pi) * np.sum((1/k)[:,None] * np.sin(2*np.pi*k[:,None]*x[None,:]),

x = np.linspace(0, 1, 50000)
f = f_square_pm1(x)

Ns = [10, 30, 50, 100, 200]
delta = 0.02
xc = 0.5

plt.figure()
plt.plot(x, f, linewidth=2, label="f(x)")

for N in Ns:
    SN = S_N(x, N)
    plt.plot(x, SN, linewidth=1.8, label=f"S_{N}")
```

```
    maskL = (x >= xc-delta) & (x < xc)
    iL = np.argmax(SN[maskL])
    xL = x[maskL][iL]
    yL = SN[maskL][iL]

    # Todos los puntos del mismo color
    plt.scatter(xL, yL,
                s=45,
                color="darkred",
                edgecolors="white",
                linewidth=0.8,
                zorder=5)

plt.xlim(0.45, 0.55)
plt.ylim(-1.6, 1.6)
plt.title("Zoom cerca del salto")
plt.grid(True, linestyle="--", alpha=0.6)
plt.legend()
plt.tight_layout()
plt.show()
```



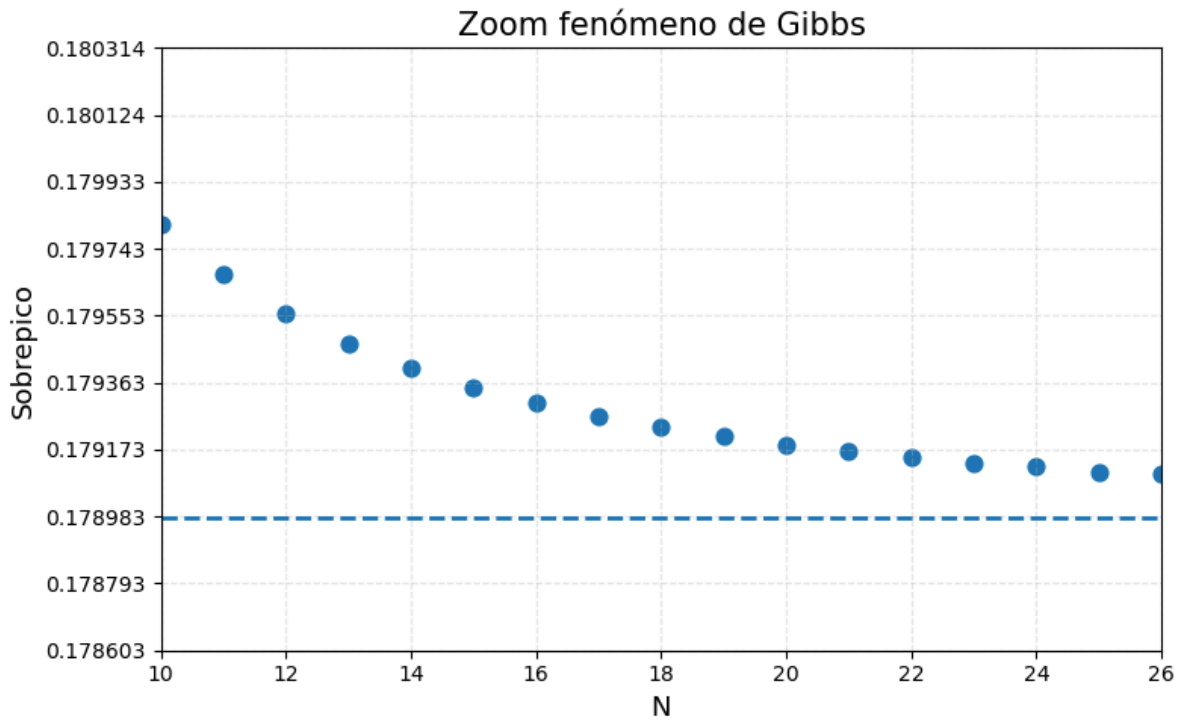
In [7]:
```
import numpy as np
import matplotlib.pyplot as plt

def f_square_pm1(x):
    y = np.sin(2*np.pi*x)
    return np.where(y >= 0, 1.0, -1.0)

def S_N(x, N):
    m = np.arange(1, N+1)
    k = 2*m - 1
    return (4/np.pi) * np.sum(
        (1/k)[:,None] * np.sin(2*np.pi * k[:,None] * x[None,:]),
        axis=0
    )
```

```
x = np.linspace(0, 1, 20000)

delta = 0.05
mask = (x > 0.5 - delta) & (x < 0.5 + delta)

N_values = np.arange(1, 200)
overshoot = []

for N in N_values:
    SN = S_N(x, N)
    local_max = np.max(SN[mask])
    overshoot.append(local_max - 1)


plt.figure(figsize=(10,6))

plt.plot(N_values, overshoot, 'o', markersize=4)  # puntos
plt.axhline(0.1789, linestyle='--', linewidth=2)

plt.xlabel("N", fontsize=14)
plt.ylabel("Sobrepico", fontsize=14)
plt.title("Fenómeno de Gibbs", fontsize=16)


plt.ylim(0.15, 0.19)

plt.grid(True, linestyle="--", alpha=0.4)
plt.tight_layout()
plt.show()
```

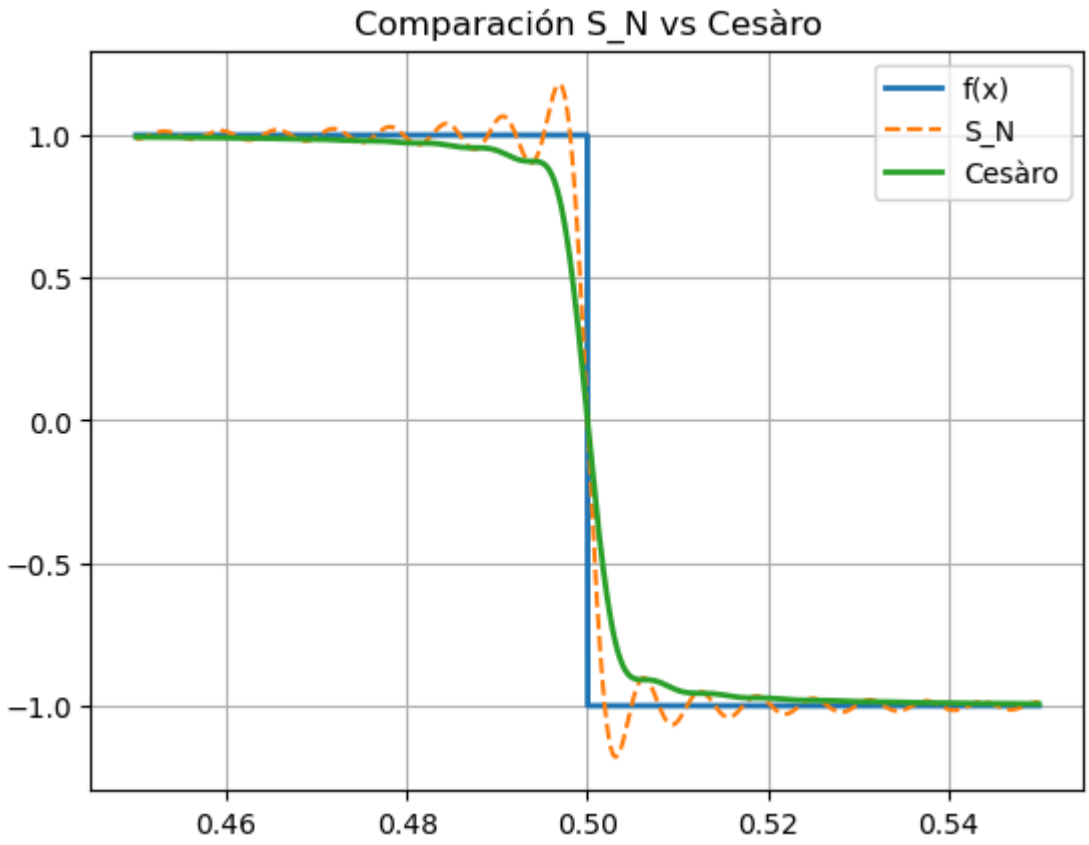


In [8]:
```
import numpy as np
import matplotlib.pyplot as plt

def f_square_pm1(x):
    y = np.sin(2*np.pi*x)
    return np.where(y >= 0, 1.0, -1.0)

def S_N(x, N):
    m = np.arange(1, N+1)
    k = 2*m - 1
```

```
    return (4/np.pi) * np.sum(
        (1/k)[:, None] * np.sin(2*np.pi * k[:, None] * x[None, :]),
        axis=0
    )


x = np.linspace(0, 1, 30000)

delta = 0.05
mask = (x > 0.5 - delta) & (x < 0.5 + delta)

# N del 10 al 25
N_values = np.arange(10, 27)

overshoot = []
for N in N_values:
    SN = S_N(x, N)
    local_max = np.max(SN[mask])
    overshoot.append(local_max - 1)

overshoot = np.array(overshoot)

gibbs_limit = 0.17898

# -------- GRAFICA --------
plt.figure(figsize=(8,5))

plt.scatter(N_values, overshoot, s=60)
plt.axhline(gibbs_limit, linestyle='--', linewidth=2)


plt.xlim(10, 26)
plt.xticks(np.arange(10, 27, 2))


ymin = overshoot.min() - 0.0005
ymax = overshoot.max() + 0.0005
plt.ylim(ymin, ymax)
plt.yticks(np.linspace(ymin, ymax, 10))

plt.xlabel("N", fontsize=13)
plt.ylabel("Sobrepico", fontsize=13)
plt.title("Zoom fenómeno de Gibbs", fontsize=15)

plt.grid(True, linestyle="--", alpha=0.3)
plt.tight_layout()
plt.show()
```

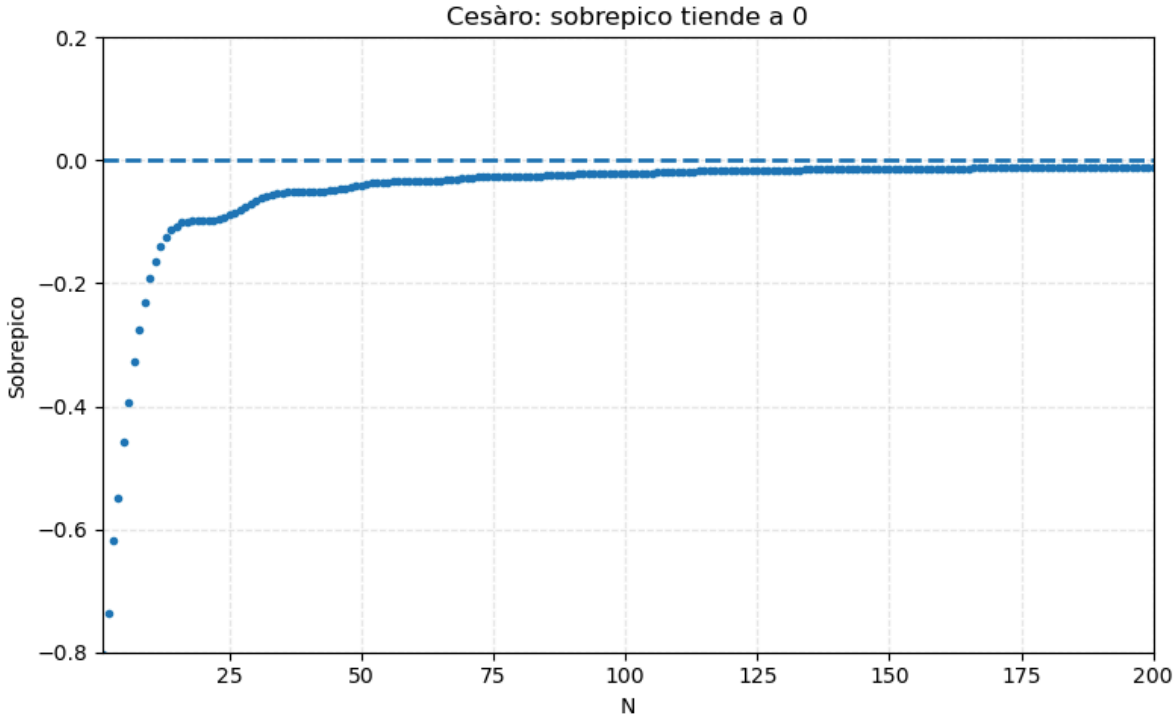


In [13]:
```python
def sigma_N(x, N):
    acc = np.zeros_like(x)
    for k in range(1, N + 1):
        acc += S_N(x, k)
    return acc / N

N = 80
SN = S_N(x, N)
sigN = sigma_N(x, N)

plt.figure()
plt.plot(x[mask], f[mask], linewidth=2, label="f(x)")
plt.plot(x[mask], SN[mask], linestyle="--", linewidth=1.5, label="S_N")
plt.plot(x[mask], sigN[mask], linewidth=2, label="Cesàro")
plt.legend()
plt.title("Comparación S_N vs Cesàro")
plt.grid(True)
plt.show()
```



In [14]:
```python
import numpy as np
import matplotlib.pyplot as plt

# --- Fejér (Cesàro) directo para la onda cuadrada sign(sin(2πx)) ---
def sigma_N_fejer(x, N):
    # armónicos n=1..N, pero solo impares contribuyen
    n = np.arange(1, N+1)
    n = n[n % 2 == 1]                 # impares
    w = 1 - n/(N+1)                   # peso de Fejér
    # b_n = 4/(π n) y multiplicamos por el peso w
    return (4/np.pi) * np.sum((w/n)[:, None] * np.sin(2*np.pi * n[:, None] * x[None

# Mallado fino
x = np.linspace(0, 1, 30000)

# Ventana alrededor del salto x0 = 0.5
x0 = 0.5
delta = 0.05
mask = (x > x0 - delta) & (x < x0 + delta)

# N para la gráfica
N_values = np.arange(1, 201)

# Sobrepico Cesàro (superior): H_N = max sigma_N - 1
H = []
for N in N_values:
    sig = sigma_N_fejer(x, N)
    local_max = np.max(sig[mask])
    H.append(local_max - 1)

H = np.array(H)

# --- Gráfica ---
plt.figure(figsize=(8,5))
plt.plot(N_values, H, 'o', markersize=3)           # solo puntos
plt.axhline(0.0, linestyle='--', linewidth=2)      # referencia en 0
```

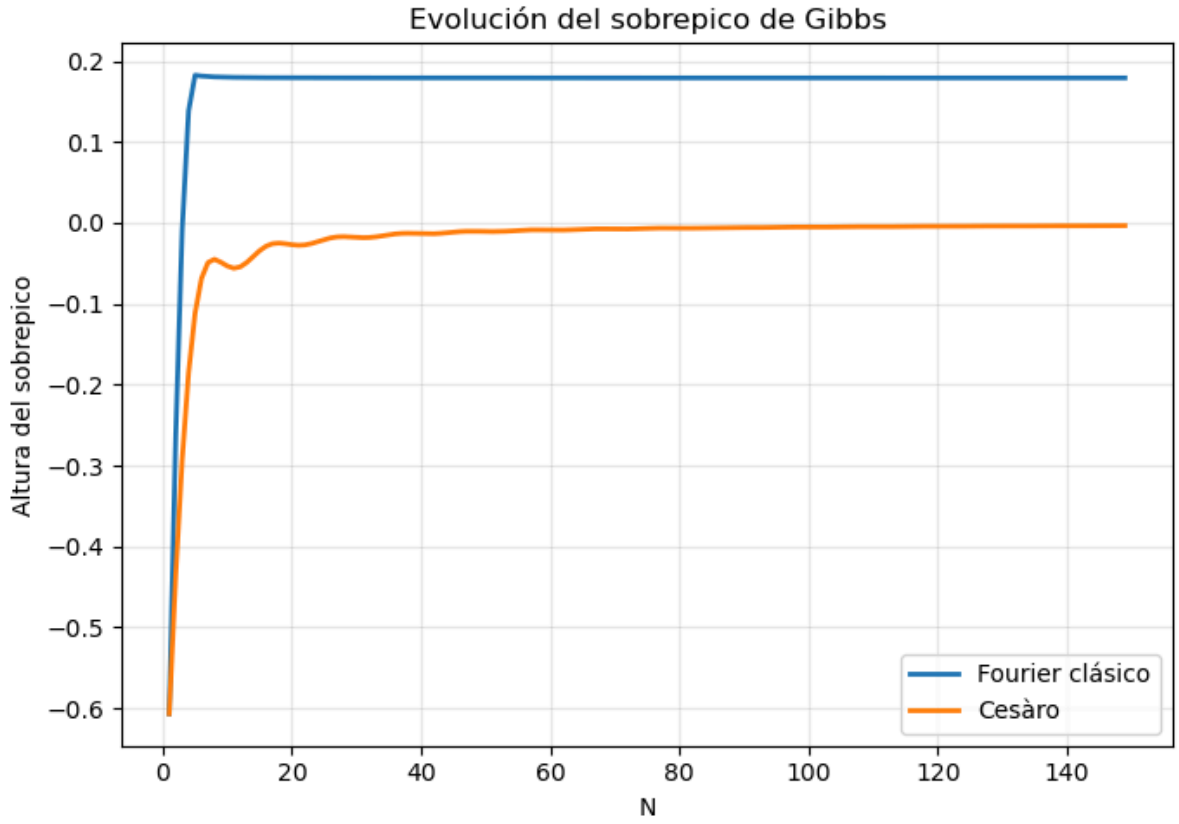
```
plt.xlabel("N")
plt.ylabel("Sobrepico")
plt.title("Cesàro: sobrepico tiende a 0")
plt.grid(True, linestyle="--", alpha=0.3)

# Ajuste de ejes para que se vea la caída
plt.xlim(1, 200)
plt.ylim(-0.8, 0.2)

plt.tight_layout()
plt.show()
```



In [15]:

```
import numpy as np
import matplotlib.pyplot as plt

def f_square_pm1(x):
    y = np.sin(2*np.pi*x)
    return np.where(y >= 0, 1.0, -1.0)

def S_N(x, N):
    m = np.arange(1, N + 1)
    k = 2*m - 1
    return (4/np.pi) * np.sum((1/k)[:, None] * np.sin(2*np.pi*k[:, None]*x[None, :]

x = np.linspace(0, 1, 60000, endpoint=False)
x0 = 0.5
delta = 0.05


mask = (x > x0 - delta) & (x < x0)

Ns = np.arange(1, 150)

G_fourier = []
G_cesaro = []

acc = np.zeros_like(x)

for N in Ns:
```

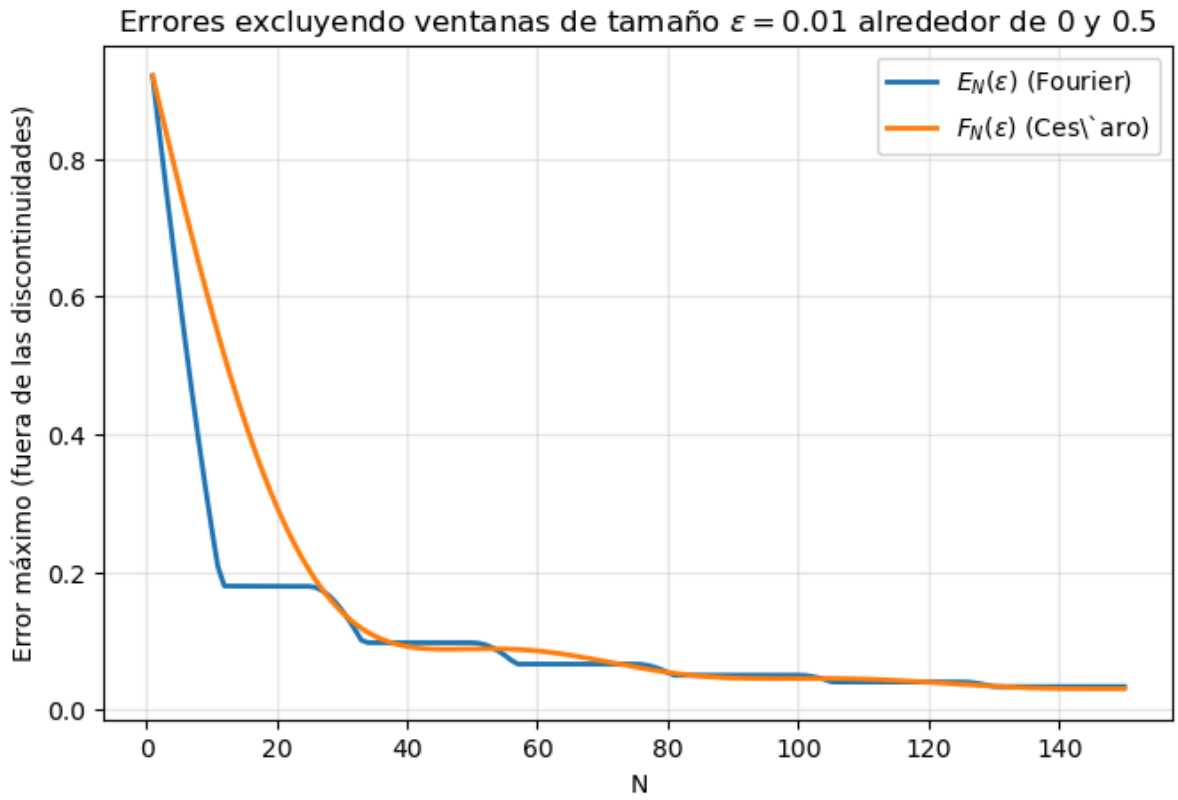
```
    SN = S_N(x, N)
    acc += SN
    sigmaN = acc / N

    # sobrepico real (por encima de 1)
    G_fourier.append(np.max(SN[mask]) - 1)
    G_cesaro.append(np.max(sigmaN[mask]) - 1)

plt.figure(figsize=(7,5))
plt.plot(Ns, G_fourier, linewidth=2, label="Fourier clásico")
plt.plot(Ns, G_cesaro, linewidth=2, label="Cesàro")

plt.xlabel("N")
plt.ylabel("Altura del sobrepico")
plt.title("Evolución del sobrepico de Gibbs")
plt.legend()
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.savefig("sobrepico_vs_N.png", dpi=300)
plt.show()
```



In [16]:

```
import numpy as np
import matplotlib.pyplot as plt

def f_square_pm1(x):
    y = np.sin(2*np.pi*x)
    return np.where(y >= 0, 1.0, -1.0)

def S_N(x, N):
    m = np.arange(1, N + 1)
    k = 2*m - 1
    return (4/np.pi) * np.sum((1/k)[:, None] * np.sin(2*np.pi*k[:, None]*x[None, :]

x = np.linspace(0, 1, 60000, endpoint=False)
f = f_square_pm1(x)

eps = 0.01
```

```
# discontinuidades en [0,1): 0 y 0.5
dpoints = [0.0, 0.5]

mask = np.ones_like(x, dtype=bool)
for xj in dpoints:
    # distancia en el círculo (por periodicidad): min(|x-xj|, 1-|x-xj|)
    dist = np.minimum(np.abs(x - xj), 1 - np.abs(x - xj))
    mask &= (dist >= eps)

Nmax = 150
Ns = np.arange(1, Nmax + 1)

err_S = np.zeros_like(Ns, dtype=float)
err_C = np.zeros_like(Ns, dtype=float)

acc = np.zeros_like(x)

for i, N in enumerate(Ns):
    SN = S_N(x, N)
    acc += SN
    sigmaN = acc / N

    err_S[i] = np.max(np.abs(SN[mask] - f[mask]))
    err_C[i] = np.max(np.abs(sigmaN[mask] - f[mask]))

plt.figure(figsize=(7,5))
plt.plot(Ns, err_S, linewidth=2, label=r"$E_N(\varepsilon)$ (Fourier)")
plt.plot(Ns, err_C, linewidth=2, label=r"$F_N(\varepsilon)$ (Ces\`aro)")
plt.xlabel("N")
plt.ylabel("Error máximo (fuera de las discontinuidades)")
plt.title(rf"Errores excluyendo ventanas de tamaño $\varepsilon={eps}$ alrededor de
plt.grid(True, alpha=0.3)
plt.legend()

plt.subplots_adjust(left=0.12, right=0.98, top=0.88, bottom=0.12)
plt.savefig("error_vs_N.png", dpi=300)
plt.show()
```


Errores excluyendo ventanas de tamaño $\varepsilon = 0.01$ alrededor de 0 y 0.5
$E_N(\varepsilon)$ (Fourier)
$F_N(\varepsilon)$ (Ces\`aro)
Error máximo (fuera de las discontinuidades)
0.0
0.2
0.4
0.6
0.8
0
20
40
60
80
100
120
140
N

In [ ]: