Facultad de Matemática, Astronomía, Física y Computación, U.N.C.

Computación

Aula Virtual: https://famaf.aulavirtual.unc.edu.ar/course/view.php?id=747

Resguardo tutoriales: https://www.famaf.unc.edu.ar/~moreschi/docencia/Computacion/

Tutorial Problemas 12, 13 y 14 de la Guía N° 4

Problema 12: Regla del punto medio. Una regla muy simple para aproximar una integral es la regla del punto medio:

$$\int_a^b f(x)dx \simeq (b-a)f\Big(\frac{a+b}{2}\Big).$$

Cuando se subdivide el intervalo de interés en N subintervalos de igual longitud, tenemos la *Regla del punto medio compuesta*:

$$\int_a^b f(x) \simeq \frac{b-a}{N}\sum_{j=0}^{N-1} f(x_j), \qquad x_j = a + \frac{h}{2} + hj, \qquad h = \frac{b-a}{N}.$$

Esta regla, al igual que la del trapecio compuesta, resulta ser de convergencia cuadrática en h.

Escriba un programa en PYTHON `punto_medio(f,a,b,N)` que implemente la "Regla del punto medio compuesta" para aproximar la integral de una función $f(x)$ previamente definida. Trabaje en la forma más vectorizada posible (evitando bucles explícitos tanto como pueda).

Pruebe su programa integrando la función $f(x) = \frac{1}{4}x(10-x)+\text{sen}(\pi x)$ en el intervalo $[0, 5]$. Haga un dibujo donde se vea la función y superpuesto la función a trozos con la que se hace la estimación de la integral cuando se utiliza la regla del punto medio compuesta considerando 5 subintervalos. Verifique su programa realizando el test ε explicado en el teórico.

Problema 13: Regla del trapecio. Repita ejercicio anterior implementando el cálculo de la integral mediante la regla del trapecio. Compare numérica y graficamente los resultados.

Problema 14: Regla de Simpson. Repita el ejercicio anterior implementando el cálculo de la integral mediante la regla de Simpson compuesta. Compare numérica y gráficamente los resultados.

Tutorial:

- Guarde en el archivo p12.py las siguientes instrucciones:

```python
import numpy as np
from scipy import integrate
import matplotlib.pyplot as plt

def f(x):
    return x*(10.-x)/4. + np.sin(np.pi*x)


def intgr_f(x):
    return x**2*(15.-x)/12. - np.cos(np.pi*x)/np.pi

```

```
12
13 def punto_medio(func, a, b, N):
14     h = (b - a)/N
15     return h*( sum(func(a + 0.5*h + h*i ) for i in range(N)) )
16
17
18 def trapecio(func, a, b, N):
19     h = (b - a)/N
20     return h*(0.5*func(a) + sum(func(a+h*(i+1)) for i in range(N-1))
21               + 0.5*func(b))
22
23
24 def simpson(func, a, b, n):
25     h=(b-a)/(n-1)
26     k=0.0
27     x=a + h
28     for i in range(1,n//2 + 1):
29         k += 4*func(x)
30         x += 2*h
31     x = a + 2*h
32     for i in range(1,n//2):
33         k += 2*func(x)
34         x += 2*h
35     return (h/3)*(func(a)+func(b)+k)
36
37
38 a = 0.
39 b = 5.
40 print()
41 print('a =',a)
42 print('b =',b)
43 integralf_exac = intgr_f(b) - intgr_f(a)
44
45 # un cálculo separado --------------------------
46 N = 81
47 print('N =',N)
48 xx = np.arange(0, N)
49 xx = xx *(b-a)/(N-1)
50 y = f(xx)
51
52 int_punto_medio = punto_medio(f, a, b, N)
53 print('  int_punto_medio = ',int_punto_medio,
54       '   error =',(int_punto_medio-integralf_exac) )
55
56 int_trapecio = trapecio(f, a, b, N)
57 print('     int_trapecio = ',int_trapecio,
58             '   error =',(int_trapecio-integralf_exac) )
59
60 int_simpson = simpson(f, a, b, N)
61 print('      int_simpson = ',int_simpson,
62             '     error =',(int_simpson-integralf_exac) )
63
64 int_simpson_scipy = integrate.simps(y,xx)
65 print('int_simpson_scipy = ',int_simpson_scipy)
66
67 print('   integralf_exac = ',integralf_exac)
```

```
print()

# ------------------------------------------------------
# Escribimos en un archivo, para que nos quede el resultado.
file1 = outfile = open('datos/p12-punto_medio.txt', 'w')
file2 = outfile = open('datos/p12-trapecio.txt', 'w')
file3 = outfile = open('datos/p12-simpson.txt', 'w')

file1.write("# N     Integral          error\n")
file2.write("# N     Integral          error\n")
file3.write("# N     Integral          error\n")

xmedio=[]
ymedio=[]
xtrapecio=[]
ytrapecio=[]
xsimpson=[]
ysimpson=[]

for i in range(5):
    N = 10*2**i + 1
    integral = punto_medio(f, a, b, N)
    xmedio.append(N)
    ymedio.append(abs(integral-integralf_exac))
    file1.write( str(N)+"  "+str(integral)+"  "
                +str(integral-integralf_exac)+"\n" )
    integral = trapecio(f, a, b, N)
    xtrapecio.append(N)
    ytrapecio.append(abs(integral-integralf_exac))
    file2.write( str(N)+"  "+str(integral)+"  "
                +str(integral-integralf_exac)+"\n" )
    integral = simpson(f, a, b, N)
    xsimpson.append(N)
    ysimpson.append(abs(integral-integralf_exac))
    file3.write( str(N)+"  "+str(integral)+"  "
                +str(integral-integralf_exac)+"\n" )


file1.close()
file2.close()
file3.close()

plt.figure( figsize=(10, 7.5) )
plt.title('Errores en la integración numérica')
plt.xlabel('N')
plt.ylabel('errores')
plt.grid()
plt.loglog(xmedio , ymedio,'o-', label='error pmedio')
plt.loglog(xtrapecio , ytrapecio,'^-', label='error trapecio')
plt.loglog(xsimpson , ysimpson,'s-', label='error simpson')
plt.legend(loc="best")
plt.savefig('graficos/p12-errores.png', dpi=100)
plt.show()


# gráfico punto medio -------------------------
```

```
N = 4 # número de intervalos
print('N =',N)
x = np.arange(0, N)
hache = (b-a)/(N)
x = (x + 0.5) * hache
y = f(x)

print(' x=',x)
print(' y=',y)

xx = np.linspace(0,5,101)
yy = f(xx)

plt.figure( figsize=(10, 7.5) )
plt.title('Función y rectángulos de integración')
plt.xlabel('x')
plt.ylabel('f / rectángulos')
plt.bar(x, y, width=hache, label='Integral del punto medio', alpha=0.3)
plt.plot(xx , yy, 'r', linewidth=3, label='Función')
markerline, stemlines, baseline = plt.stem(x, y,
    label='Puntos de evaluación',use_line_collection=True)
# setting property of baseline
plt.setp(baseline, color='k', linewidth=0.5)
plt.grid()
plt.legend(loc="best")
plt.savefig('graficos/p12-f-punto_medio.png', dpi=100)
plt.show()

# gráfico trapecios --------------------------
#N = 5
print('N =',N)
x = np.arange(0, N+1)
x = (x) *(b-a)/(N)
y = f(x)

print(' x=',x)
print(' y=',y)

xx = np.linspace(0,5,101)
yy = f(xx)

plt.figure( figsize=(10, 7.5) )
plt.title('Función y trapecios de integración')
plt.xlabel('x')
plt.ylabel('f / trapecios')
plt.fill_between(x, y, label='Integral del trapecio', alpha=0.3)
plt.plot(xx , yy, 'r', linewidth=3, label='Función')
markerline, stemlines, baseline = plt.stem(x, y,
    label='Puntos de evaluación',use_line_collection=True)
# setting property of baseline
plt.setp(baseline, color='k', linewidth=0.5)
plt.grid()
plt.legend(loc="best")
plt.savefig('graficos/p12-f-trapecio.png', dpi=100)
plt.show()

```

```
180
181 # sección gráficos de las parábolas que usa Simpson ----------
182
183 def ilagrange(x, xp, fp):
184     pol = 0.0
185     # suma sobre los puntos
186     for m in range(len(xp)):
187         l = 1.
188         for n in range(len(xp)):
189             if n == m:
190                 continue
191             l *= (x - xp[n])/(xp[m] - xp[n])
192         pol += fp[m]*l
193     return pol
194
195 # rango para los primeros tres puntos
196 # para generar una parábola
197 x1 = x[:3]
198 print('x1=',x1)
199 y1 = f(x1)
200
201 # rango para los últimos tres puntos
202 # para generar una parábola
203 x2 = x[2:]
204 print('x2=',x2)
205 y2 = f(x2)
206
207 # rangos para hacer los gráficos de las parábolas
208 xx1 = np.linspace(0,x1[-1],50)
209 print('xx1=',xx1)
210 xx2 = np.linspace(x2[0],x2[-1],50)
211 print('xx2=',xx2)
212
213 plt.figure( figsize=(10, 7.5) )
214 plt.title('Función y parábolas de integración')
215 plt.xlabel('x')
216 plt.ylabel('f / parábolas')
217 plt.fill_between(xx1, ilagrange(xx1,x1,y1),
218                  label='Integral de la parábola 1', alpha=0.3)
219 plt.fill_between(xx2, ilagrange(xx2,x2,y2),
220                  label='Integral de la parábola 2', alpha=0.3)
221 plt.plot(xx , yy, 'r', linewidth=3, label='Función')
222 markerline, stemlines, baseline = plt.stem(x, y,
223     label='Puntos de evaluación',use_line_collection=True)
224 # setting property of baseline
225 plt.setp(baseline, color='k', linewidth=0.5)
226 plt.grid()
227 plt.legend(loc="best")
228 plt.savefig('graficos/p12-f-simpson.png', dpi=100)
229 plt.show()
```

- Desde la terminal ejecute:

  python3 p12.py

  e interprete el resultado.

Alternativamente ejecute:

python3

y vaya agregando uno a uno los bloques del programa.

Obtendrá en la terminal algo como esto:

```
a = 0.0
b = 5.0
N = 81
  int_punto_medio =  21.471348676117806    error = 0.0013955704168928662
     int_trapecio =  21.46716290380018    error = -0.002790201900735355
      int_simpson =  21.4699583867955      error = 5.281094587417101e-06
int_simpson_scipy =  21.46995838679549
   integralf_exac =  21.469953105700913
```

y los siguientes gráficos:

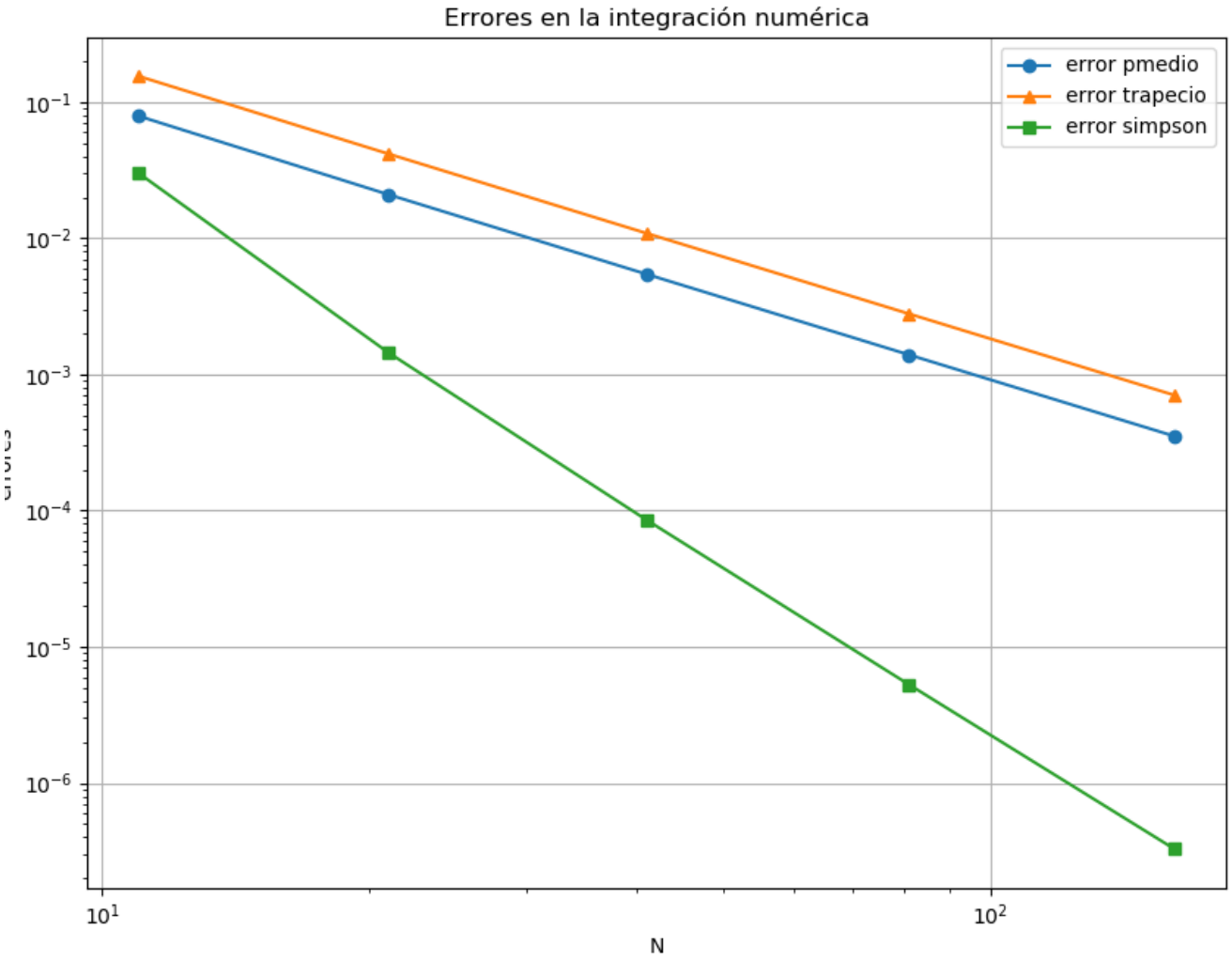

Figura 1: Errores en la integración numérica.

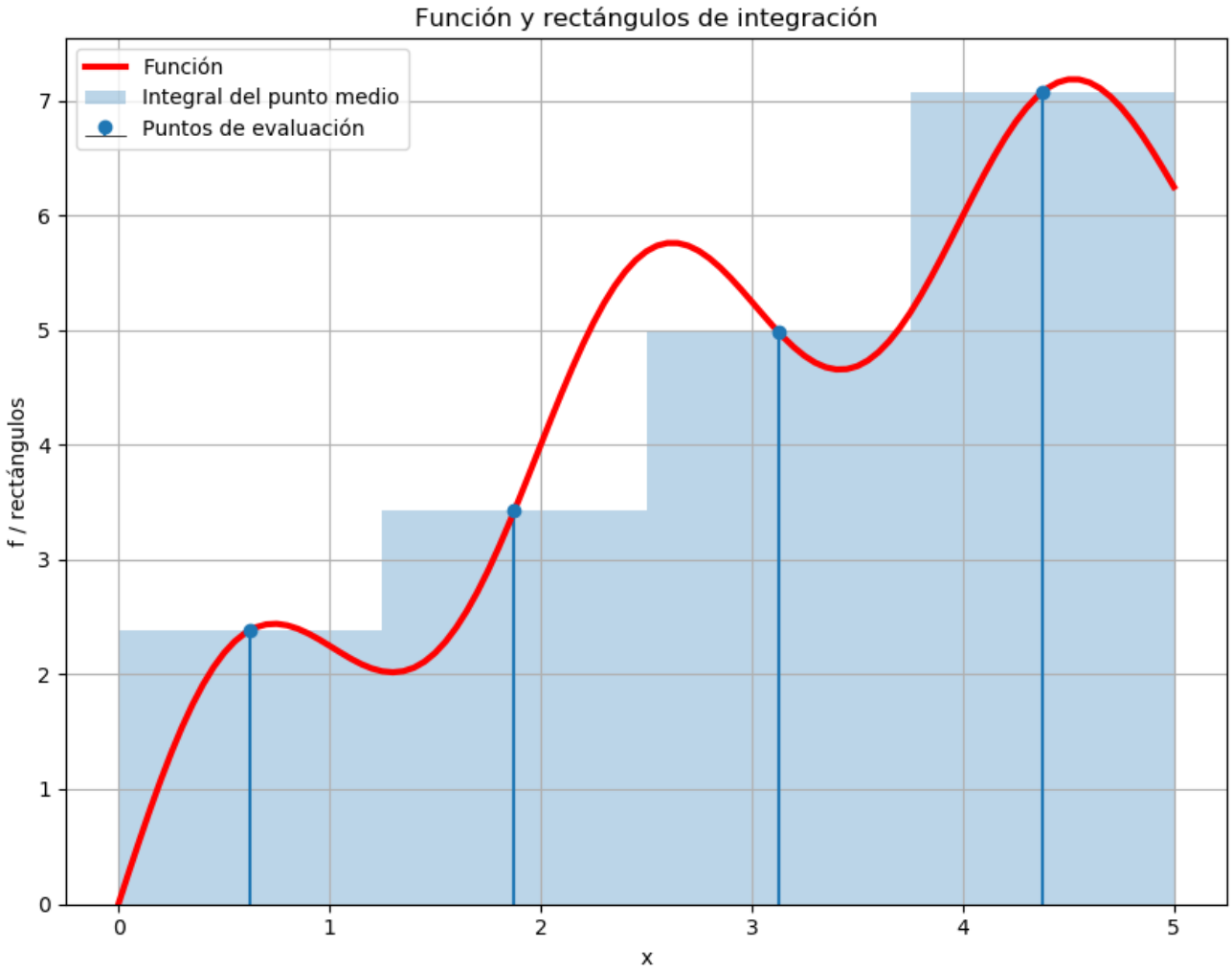

Figura 2: Área de integración con los puntos medios.

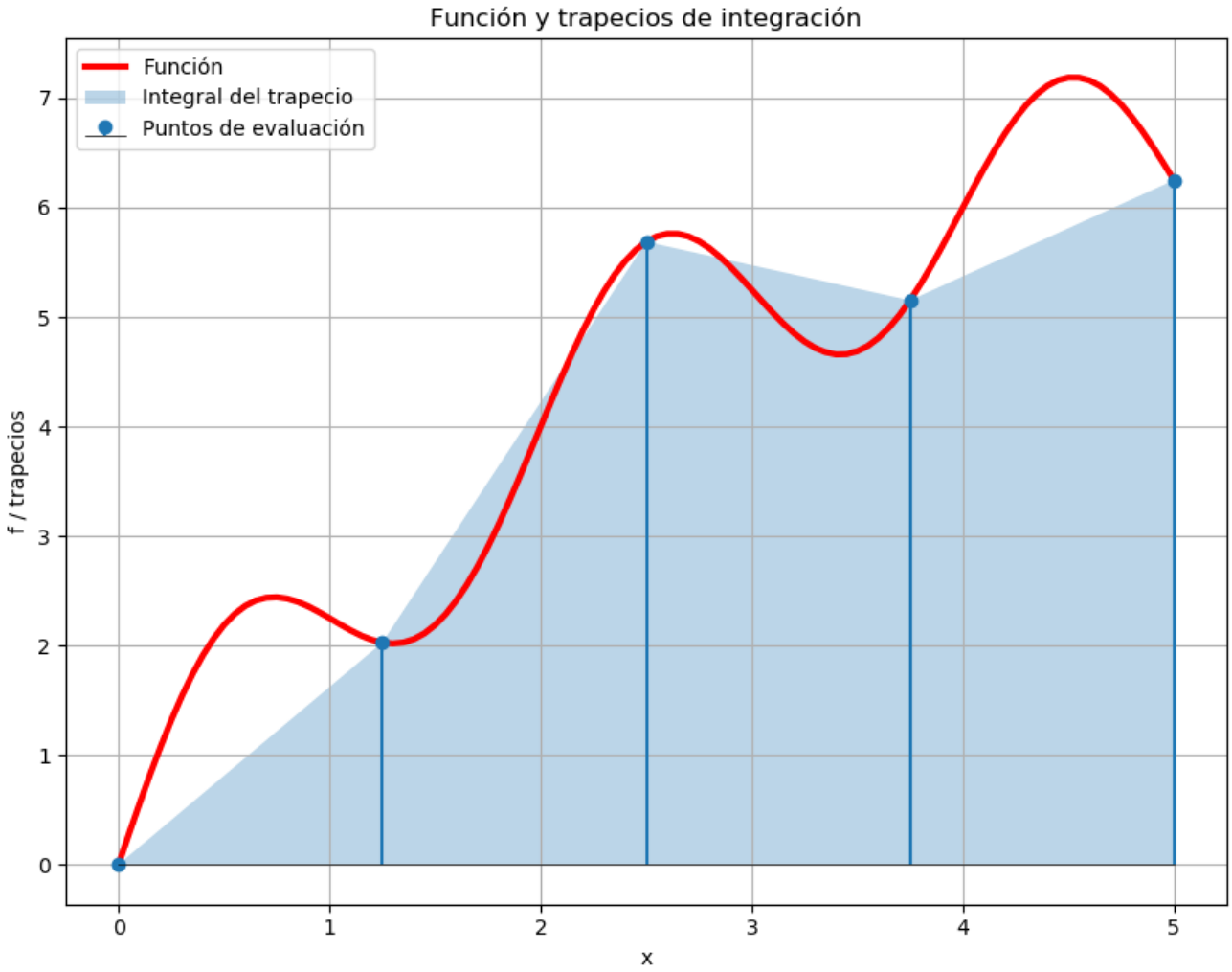

Figura 3: Área de integración con los trapecios.

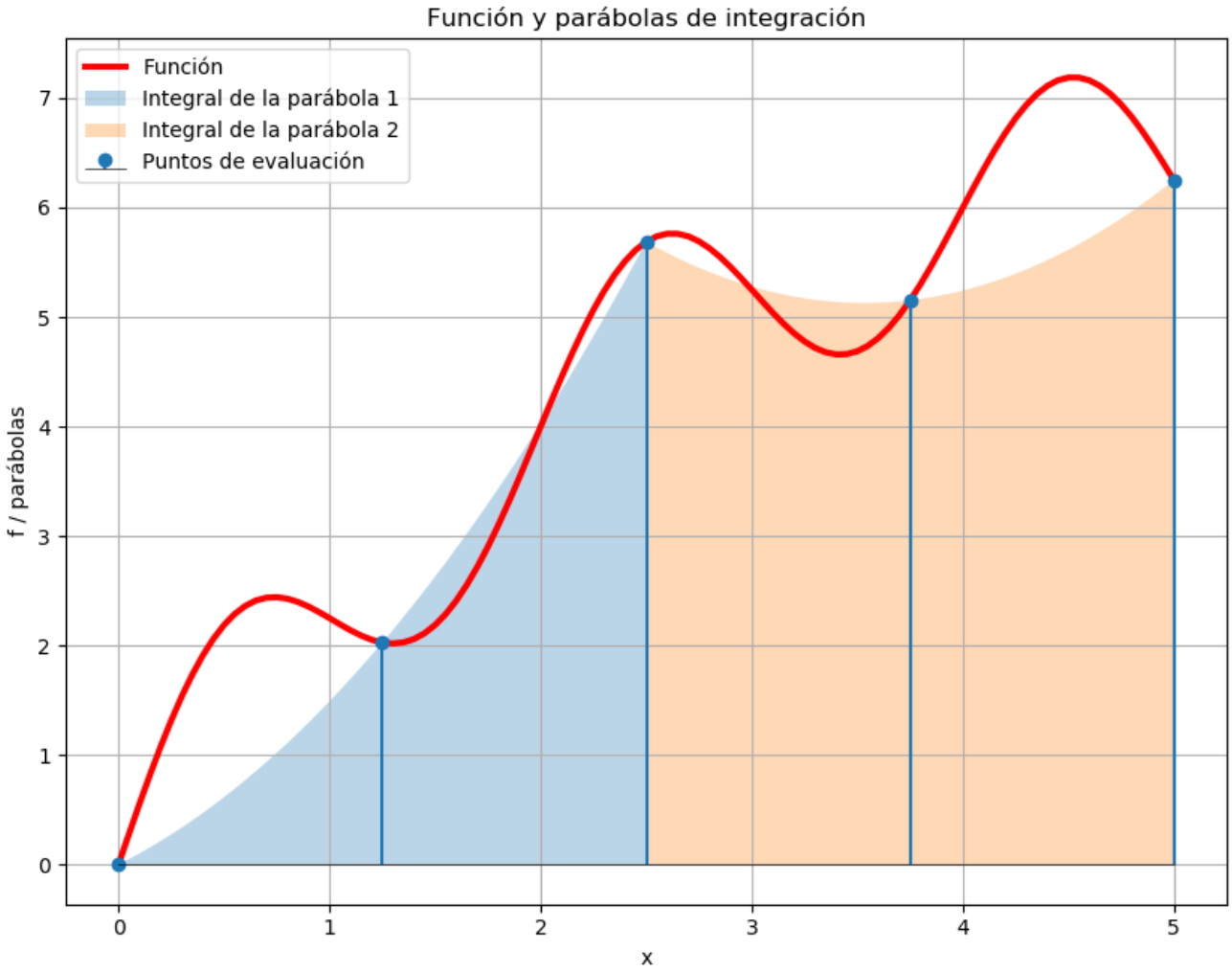

Figura 4: Área de integración con las parábolas usadas en el método de Simpson.