Exercice4_SDZ

January 15, 2022

4. Certaines fois, il n'est pas possible de calculer analytiquement une intégrale. Par exemple
$$\int_{-1}^{1} e^{-x^2} dx$$
doit être approximée par une méthode numérique.

   - La **méthode des trapèzes** remplace le calcul de l'intégrale définie par une somme (d'aires de trapèzes):
   $$\int_a^b f(x)dx \approx \frac{h}{2}\left(f(x_0) + 2f(x_1) + 2f(x_2) + \ldots + 2f(x_{n-1}) + f(x_n)\right)$$
   où $h = \frac{b-a}{n}$ et $x_k = a + k \cdot h$.

   - La **méthode de Simpson** remplace le calcul de l'intégrale définie par une somme (d'aires de paraboles)
   $$\int_a^b f(x)dx \approx \frac{h}{3}\left(f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \ldots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)\right)$$
   où $h = \frac{b-a}{n}$ et $x_k = a + k \cdot h$ (n pair).

   Utiliser les méthodes des trapèzes et de Simpson (dans Matlab, Python ou Excel) pour approximer
   $$\int_{-1}^{1} e^{-x^2} dx$$
   en faisant varier n ($n = 2, 4, 8, \ldots, 64$)

[2]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[60]:
```
def f(x):
    return np.exp(-x**2)


a, b = -1, 1
xup = np.linspace(a, b, 200)

N = np.array([2**(n+1) for n in range(6)])
```

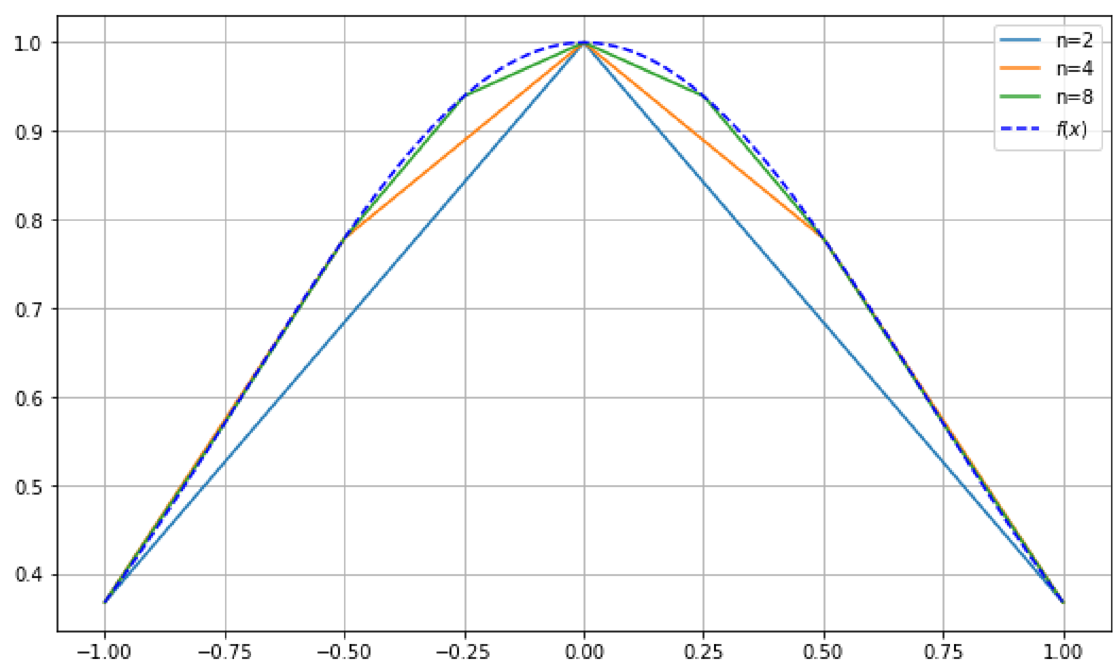
```
plt.figure(figsize=(10, 6))
trapeze = np.zeros_like(N, dtype=float)
simpson = np.zeros_like(N, dtype=float)

for i, n in enumerate(N):
    h = (b-a)/n
    x = np.linspace(a, b, n+1)

    data = f(x)
    # Méthode des trapèzes
    trapeze[i] = h * (np.sum(data) - 1/2 * (data[0] + data[-1]))
    if n <= 8:
        plt.plot(x, data, label=f'n={n}')

    # Méthode de Simpson
    simpson[i] = h/3 * (data[0] + data[-1] + 4 *
                        np.sum(data[1:-1:2]) + 2*np.sum(data[2:-1:2]))


plt.plot(xup, f(xup), 'b--', label='$f(x)$')
plt.legend()
plt.grid()
```



[61]:
```python
print(trapeze)
print(simpson)

plt.figure(figsize=(10,6))
plt.plot(trapeze, label="Méthode des trapèzes")
plt.plot(simpson, label="Méthode de Simpson")
plt.grid()
plt.legend()
plt.show()
```

```
[1.36787944 1.4627405  1.4859682  1.49173123 1.49316919 1.49352851]
[1.57858629 1.49436086 1.49371076 1.49365224 1.49364851 1.49364828]
```