

Assignment 3 - Intégration et dérivation numérique

April 7, 2025

1 Assignment 3: Formule de quadrature d'ordres élevés

On considère une fonction $f : [a, b] \rightarrow \mathbb{R}$ dans $C^0([a, b])$. On est intéressé à approcher l'intégrale $I(f) = \int_a^b f(x) dx$ en utilisant des formules de quadrature d'ordres élevés.

```
[ ]: import matplotlib.pyplot as plt
import numpy as np
from scipy.special import erf, expi
from scipy.integrate import quad
```

1.1 Partie 1

Dans un premier temps, nous essayons d'approximer l'intégrale de f en exploitant des règles de quadrature simples (c-à-d non composites), de la forme

$$I(f) = \int_a^b f(x) dx \approx I_n(f) = \sum_{i=1}^n w_i f(x_i)$$

Ces règles ont été construites à partir de l'interpolation polynomiale d'ordre élevé sur des nœuds équidistribués, c-à-d

$$I_n(f) = \int_a^b \Pi_n(f)(x) dx ,$$

où $\Pi_n(f)$ est l'interpolée de Lagrange de f sur des noeuds équidistribués.

Plus précisément, nous considérons : * **nœuds** : équidistribués dans $[a, b]$, donc $x_i = a + i \left(\frac{b-a}{n-1} \right)$
* **poids** : calculés à partir des fonctions de base de Lagrange définies sur les nœuds afin de garantir le degré d'exactitude maximale, donc $w_i = \int_a^b \varphi_i(x) dx$.

On considère la fonction

$$\kappa(x) = 1 - \kappa_1(\kappa_2(x)) \quad \text{avec} \quad \kappa_1(y) = \frac{1}{1 - 2y + 4y^2} \quad \text{et} \quad \kappa_2(x) = \frac{x^2 - 4}{4} .$$

Approximer l'intégrale de κ dans $[-3, 3]$ pour $n = 2^j + 1$ noeuds, avec $j = 1, 2, \dots, 5$, et comparer le résultat avec une estimation précise de l'intégrale (donnée).

Discuter les résultats obtenus. En particulier, est-ce que l'estimation de l'intégrale s'améliore lorsqu'on considère plus de noeuds de quadrature ? Pourquoi ? (*Réponse 1*)

```
[ ]: def Lagrange(n):
    """
    Compute Lagrange quadrature nodes and weights on equidistant nodes in  $[-1,1]$ .

    Inputs: [n]
    n : Number of nodes.

    Outputs: [x, w]
    x : Equidistant nodes in  $[-1,1]$ .
    w : Quadrature weights.
    """

    x = np.linspace(-1, 1, n)
    w = np.zeros(n)

    for i in range(n):
        # Define Lagrange basis function
        def lagrange_basis(_x):
            res = 1
            for j in range(n):
                if i != j:
                    res *= (_x - x[j]) / (x[i] - x[j])
            return res

        # Integrate the Lagrange polynomial over  $[-1,1]$ 
        w[i], _ = quad(lagrange_basis, -1, 1)

    return x, w
```

```
[ ]: k2 = lambda x : (x**2 - 4) / 4
k1 = lambda y : 1 / (1 - 2*y + 4*y**2)
k = lambda x : 1 - k1(k2(x))
a, b = -3, 3

IexactK, _ = quad(k, a, b)
print(f'A precise estimate of the value of the integral is {IexactK:.7f}')

x = np.linspace(a, b, 1000)
y = k(x)

plt.plot(x, y, 'b')
plt.xlabel('$x$'); plt.ylabel('$f(x)$')
plt.title(r'Plot of $k(x)$')
plt.xlim([a,b])
plt.grid()
plt.show()
```

```
[ ]: n_range = [2**k + 1 for k in range(1,6)]

for n in n_range :
    nodes, weights = ## COMPLETE HERE ## (nodes and weights in [-1,1])

    nodes = ## COMPLETE HERE ## (rescale nodes to [a,b])
    weights = ## COMPLETE HERE ## (rescale weights to [a,b])

    intQuad = ## COMPLETE HERE ## (compute integral)
    errQuad = ## COMPLETE HERE ## (compute error)

    print(f"n={n:3d} - "
          f"Approximated integral: {intQuad:2.4f} - "
          f"Error: {errQuad:.2e}")
```

1.1.1 Commentaire

Réponse 1 À compléter

1.2 Partie 2

Maintenant, on considère des règles de quadrature composites, en subdivisant l'intervalle d'intégration $[a, b]$ en N sous-intervalles plus petits.

Écrivez une fonction qui implemente une formule de quadrature composite, étant donnée les noeuds (dans l'intervalle de reference $[-1, 1]$), les poids, les extrêmes de l'intervalle d'integration $[a, b]$ et le nombre de sous-intervalles N .

La fonction doit avoir la structure suivante:

```
def QuadratureComposite(nodes, weights, a, b, N, f) :
    # Function that implements a composite quadrature rule, being given in
    # input the nodes (in [-1,1]), the weights, the integration interval bounds,
    # the number of sub-intervals and the integrand function.

    # Inputs: [nodes, weights, a, b, N, f]
    # nodes   : quadrature nodes (in [-1,1])
    # weights  : quadrature weights
    # [a,b]   : integration interval
    # N       : number of sub-intervals
    # f       : function to integrate
    #
    # Outputs : [Lh]
    # Lh      : integral of f in [a,b], approximated with the prescribed composite quadrature
```

```
[ ]: def QuadratureComposite(nodes, weights, a, b, N, f) :
    """Function that implements a composite quadrature rule, being given in
    input the nodes (in [-1,1]), the weights, the integration interval bounds,
    the number of sub-intervals and the integrand function.

```

```

Inputs: [nodes, weights, a, b, N, f]
nodes   : quadrature nodes (in [-1,1])
weights : quadrature weights
[a,b]    : integration interval
N        : number of sub-intervals
f        : function to integrate

Outputs : [Lh]
Lh       : integral of f in [a,b], approximated with the prescribed
↳ composite quadrature rule
"""

M = len(nodes)
if len(nodes) != len(weights):
    raise ValueError(f"Invalid value of M: {M}")

H = ## COMPLETE HERE ## (size of the sub-intervals)
x = ## COMPLETE HERE ## (points defining sub-intervals)

Lh = 0
for k in range(N) :
    z = ## COMPLETE HERE ## (local quadrature points)

    Jgk = ## COMPLETE HERE ## (local quadrature rule)

    Lh += Jgk

Lh *= ## COMPLETE HERE ## (rescale Lh)

return Lh

```

1.2.1 Formules de quadrature de Gauss-Legendre-Lobatto.

Dans ce test, on prend les nœuds et les poids de quadrature de Gauss-Legendre-Lobatto (GLL), qui conviennent à l'intégration numérique d'ordre élevé.

La fonction `GaussLegendreLobatto` ci-dessous retourne les M nœuds (dans l'intervalle de référence $[-1, 1]$) et les M poids de quadrature pour les formules de GLL.

Remarque: seulement les cas $M = 2, 3, 4, 5$ sont considérés.

M	nœuds (dans $[1,1]$)	poids (dans le même ordre que les nœuds)
2	$-1, 1$	$1, 1$
3	$-1, 0, 1$	$\frac{1}{3}, \frac{4}{3}, \frac{1}{3}$
4	$-1, -\frac{\sqrt{5}}{5}, \frac{\sqrt{5}}{5}, 1$	$\frac{1}{6}, \frac{5}{6}, \frac{5}{6}, \frac{1}{6}$

M	noeuds (dans [1,1])	poids (dans le même ordre que les noeuds)
5	$-1, -\frac{\sqrt{21}}{7}, 0, \frac{\sqrt{21}}{7}, 1$	$\frac{1}{10}, \frac{49}{90}, \frac{32}{45}, \frac{49}{90}, \frac{1}{10}$

```
[ ]: def GaussLegendreLobatto(M):
    """Returns the GaussLegendreLobatto (GLL) quadrature nodes and weights for
    the given value of M.

    Inputs : [M]
    M       : number of quadrature nodes and weights

    Outputs :
    nq       : quadrature nodes (in [-1,1])
    wq       : quadrature weights
    """

    if M == 2:
        nq = np.array([-1.0, 1.0])
        wq = np.ones(M)
    elif M == 3:
        nq = np.array([-1.0, 0.0, 1.0])
        wq = np.array([1/3, 4/3, 1/3])
    elif M == 4:
        nq = np.array([-1.0, -np.sqrt(5)/5, np.sqrt(5)/5, 1.0])
        wq = np.array([1/6, 5/6, 5/6, 1/6])
    elif M == 5:
        nq = np.array([-1.0, -np.sqrt(21)/7, 0.0, np.sqrt(21)/7, 1.0])
        wq = np.array([1/10, 49/90, 32/45, 49/90, 1/10])
    else:
        raise ValueError(f"Invalid value of M: {M}")

    return nq, wq
```

1.3 Partie 3

Éstimez numériquement le degré d'exactitude des formules de quadrature simples (i.e. pas composite, $N = 1$ sous-intervalles) de Gauss-Legendre-Lobatto (GLL) avec $M = 4$. Est-ce que les résultats obtenus sont en accord avec la théorie? (Réponse 2)

```
[ ]: # Checking quadrature fonction
a, b = 1, 4

# with lambda functions, it is possible to determine a parameter (here d)
# at a later moment
monomial = lambda x : x**d
```

```
[ ]: # recording for which degrees the monomial integral is exact (up to epsilon)
exactDegree = -1
epsilon = 1e-12

M = 4
nodes, weights = ## COMPLETE HERE ## (GLL nodes in [-1,1])

N = 1
for d in range(10) :
    intQuad = ## COMPLETE HERE ## (approximate integral of 'monomial')
    intExact = ## COMPLETE HERE ## (exact integral of 'monomial')
    print(f'Quadrature on monomial x^{d} : {intQuad:.4f} - {intExact:.4f} =_
↪{intQuad-intExact:.6e}')

    if np.abs(intQuad-intExact) < epsilon :
        exactDegree = d

print(f'\nGLL composite quadrature with N = {N} and M = {M} is exact up to_
↪degree {exactDegree}')
```

1.3.1 Commentaire

Réponse 2 À compléter

1.4 Partie 4

Considérons maintenant la fonction $f(x) = e^{-x^2} + \frac{e^x}{x}$.

Approximer $I(f) = \int_{1/e}^e f(x) dx$ en utilisant les formules de quadrature de GLL composites avec $M = 2, 3, 4, 5$ et $N = 2^j$ sous-intervalles pour $j = 3, 4, 5, 6$.

Ensuite calculer les erreurs $E_{GLL}^{M,N}(f) := |I(f) - I_{GLL}^{M,N}(f)|$, où $I_{GLL}^{M,N}(f)$ représente l'approximation de l'intégral de f obtenue avec la formule de quadrature de GLL composite, avec N sous-intervalles et de degré M .

Dessiner les erreurs en fonction de H sur une échelle logarithmique sur les deux axes. Que peut-on déduire par rapport aux ordres de convergence ? Sont-ils en accord avec la théorie ? Motivez votre réponse (Réponse 3).

```
[ ]: f = lambda x : np.exp(-x**2) + np.exp(x) / x

a, b = 1 / np.exp(1), np.exp(1)
IexactF = (erf(b) - erf(a)) * np.sqrt(np.pi) / 2 + (exp(b) - exp(a))
print(f'The exact value of the integral of f is {IexactF:.7f}')
```

```
x = np.linspace(a,b,1000)
y = f(x)
```

```
plt.plot(x, y, 'b')
plt.xlabel('$x$'); plt.ylabel('$f(x)$')
plt.title(r'Plot of $f(x) = e^{-x^2} + \dfrac{e^x}{x}$')
plt.ylim([0, 1.1*np.max(y)])
plt.xlim([a,b])
plt.grid()
plt.show()
```

```
[ ]: Mrange = np.array([2, 3, 4, 5])
Nrange = np.array([2**k for k in range(4, 9)])

for M in Mrange :
    nodes, weights = ## COMPLETE HERE ## (GLL nodes in [-1,1])
    errQuad = []

    for N in Nrange :
        intQuad = ## COMPLETE HERE ## (approximate integral of f)
        errQuad.append() ## COMPLETE HERE ## (error)

    H = (b-a) / Nrange
    slopeQuad = ( np.log(errQuad[-1]) - np.log(errQuad[0]) ) / ( np.log(H[-1]) -
↪ np.log(H[0]) )

    print(f'Pour M = {M}, La convergence numérique est environ de {slopeQuad:.
↪ 2f}')

plt.figure(figsize=(6,3))

plt.loglog(H, errQuad, 'b-o')

plt.loglog(H, H**(2*M-2) * (errQuad[0]/H[0]**(2*M-2))**2, 'k:')
plt.loglog(H, H**(2*M-1) * (errQuad[0]/H[0]**(2*M-1)), 'r:')

plt.title(f"Errors for M={M}", fontweight='bold')
plt.legend(['GLL', f'$H^{2*M-2}$', f'$H^{2*M-1}$'])
plt.xlabel(r'$H$'); plt.ylabel('error')

xticks = [4*10**(-2), 6*10**(-2), 8*10**(-2), 10**(-1), 2*10**(-1)]
plt.xticks(xticks)
plt.grid(which='major', linestyle='--')
```

1.4.1 Commentaire

Réponse 3 À compléter

1.5 Partie 5

Répéter la Partie 4, mais en considérant la fonction $g_K : [-1, 1] \rightarrow \mathbb{R}$ définie comme suit

$$g_K(x) = \begin{cases} e^x & \text{if } x \leq -K \\ c_0 x^2 + c_1 |x| & \text{if } -K < x \leq K \\ e^{-x} & \text{if } x > K \end{cases} \quad (1)$$

avec $K = 0.70$, $c_0 = -\frac{K+1}{K^2}e^{-K}$, $c_1 = e^{-K}(1 + \frac{2}{K})$ et en prenant $N = 2^k$, $k = 3, 4, 5, 6$ sous-intervalles.

On a que l'intégrale exacte de g_K est égale à

$$I_{g_K} = \int_{-1}^1 g_K(x) dx = 2e^{-K} - \frac{2}{e} + \frac{2}{3}c_0 K^3 + c_1 K^2.$$

Quels sont les ordres de convergence numérique obtenus? Sont-ils égaux à ceux obtenus avec la fonction f ? Pourquoi?(Réponse 4)

```
[ ]: a,b = -1, 1
K = 0.70
c0 = -(K+1)/K**2 * np.exp(-K)
c1 = (1+2/K) * np.exp(-K)

g = lambda x : (np.exp(x)*(x<=-K) +
                (c0 * x**2 + c1 * np.abs(x))*(x>-K)*(x<=K) +
                np.exp(-x)*(x>K))

IexactG = (2*np.exp(-K) - 2*np.exp(-1) + 2*c0/3*K**3 + c1*K**2) / 1 # exact
↳value of the integral of g
print(f'The exact value of the integral of g is {IexactG:.7f}')

x = np.linspace(a,b,1001)
y = g(x)

plt.plot(x, y, 'b')
plt.xlabel(r'$x$'); plt.ylabel(r'$g(x)$')
plt.ylim([0, 1.1*np.max(y)])
plt.xlim([a,b])
plt.grid()
plt.show()
```

```
[ ]: Mrange = np.array([2, 3, 4, 5])
Nrange = np.array([2**k for k in range(4,9)])

for M in Mrange :
    nodes, weights = ## COMPLETE HERE ## (GLL nodes in [-1,1])
    errQuad = []
```

```

for N in Nrange:
    intQuad = ## COMPLETE HERE ## (approximate integral of g_K)
    errQuad.append() ## COMPLETE HERE ## (error)

H = (b-a) / Nrange
slopeQuad = ( np.log(errQuad[-1]) - np.log(errQuad[0]) ) / ( np.log(H[-1]) -
↪ np.log(H[0]) )

print(f'Pour M = {M}, La convergence numérique est environ de {slopeQuad:.
↪ 2f}')

plt.figure(figsize=(5,3))

plt.loglog(H, errQuad, 'b-o')

plt.loglog(H, H**2 * (errQuad[0]/H[0]**2)*2, 'k:')
plt.loglog(H, H**3 * (errQuad[0]/H[0]**3)/2, 'r:')
plt.loglog(H, H**(2*M-2) * (errQuad[0]/H[0]**(2*M-2))/2, 'g:')

plt.title(f"Errors for M={M}")
plt.legend(['GLL', '$H^2$', '$H^3$', '$H^{{2M-2}}$'])
plt.xlabel('H'); plt.ylabel('err')

xticks = [2*10**(-2), 4*10**(-2), 6*10**(-2), 8*10**(-2), 10**(-1),
↪ 2*10**(-1)]
plt.xticks(xticks)
plt.grid(which='major', linestyle='--')

```

1.5.1 Commentaire

Réponse 4 À compléter

2 Quelques petites questions finales (pas évaluées)

- What types of collaboration strategies did your group use?
 - Work in pairs on different sections.
 - Work individually on different sections.
 - Work together on the same section with one notebook opened.
 - Work together on the same section with multiple notebooks opened.
 - Other (please specify).
- How effective was your collaboration strategy today? Please rate from 1 (not at all) to 5 (very effective).
- How supported did you feel by your TA during the session today? Please rate from 1 (not at all) to 5 (very effective).

Please report your answers here. **Thank you!**