Міністерство освіти і науки України
НТУУ «КПІ ім. Ігоря Сікорського»
Навчально-науковий інститут атомної та теплової енергетики
Кафедра цифрових технологій в енергетиці

Лабораторна робота №2

з дисципліни «Чисельні методи для розв’язання енергетичних задач»

Тема «Чисельне інтегрування функцій»

Варіант №22

Виконав:
Студент 3-го курсу НН ІАТЕ
гр. ТР-12
Ковальов Олександр

КИЇВ 2023

Варіант. $22\ \%\ 20 = 2$.

- Підінтегральна функція: $x^2\sqrt{1-x^2}$.
- Діапазон інтегрування – $[0; 1]$.
- Метод Трапецій, так як варіант парний.

Завдання.

1. Реалізувати програму, яка обчислює інтеграл за допомогою формули трапеції або Сімпсона, в залежності від варіанту. Точність обчислень має бути 0,0001. Мінімальну кількість кроків визначити за формулою. Оцінити похибку результату.

2. Реалізувати програму, яка обчислює інтеграл за допомогою квадратурної формули Гауса. Оцінити похибку результату.

3. Обчислити визначений інтеграл у Mathcad та порівняти реальну похибку кожного метода (різниця між розрахованим значенням інтегралу і значенням у MathCad) з аналітичною похибкою кожного методу. Реальна похибка має бути не більша ніж аналітична.

Хід роботи

Для початку, обрахуємо інтеграл в Mathcad Prime 9.

$$a := 0 \qquad b := 1 \qquad \int_a^b x^2 \sqrt{1-x^2}\, dx = 0.1963$$

1. Метод трапецій.

Для методу трапецій напишемо консольне програмне забезпечення, використовуючи Python 3.

```python
def algorithm(function, x_range, h, steps):
    lo, hi = x_range

    # Finding sum
    integration = function(lo) + function(hi)

    for i in range(1, steps):
        k = lo + i * h
        integration = integration + 2 * function(k)

    # Finding final integration value
    integration = integration * h / 2

    return integration
```

Відповідно, формула яка була інтерпретована в програмний код:

$$Area = \int_a^b ydx \approx \frac{1}{2}h[y_0 + 2(y_1 + y_2 + \cdots + y_{n-1}) + y_n]$$

$$where\ h = \frac{b-a}{n}$$

Для того, щоб порахувати кількість кроків, було застосоване правило Рунге. Тобто, алгоритм був запущений починаючи зі значення $h = 0.1$, і $h = h/2$. Кінцеві результати порівнювались, і якщо різниця по модулю між ними більша ніж задана точність – цикл продовжувався з довжиною кроку меншою в 2 рази:

```python
def get_steps(h, x_range):
    lo, hi = x_range
    return round((hi - lo) / h)


1 usage
def get_delta(algorithm, function, accuracy):
    MIN_H = 0.00001
    MAX_H = 1

    DEFAULT_X_RANGE = [0, 1]

    h = 0.1
    while MAX_H >= h > MIN_H:

        result1 = algorithm(function, DEFAULT_X_RANGE, h, get_steps(h, DEFAULT_X_RANGE))

        h = h / 2
        result2 = algorithm(function, DEFAULT_X_RANGE, h, get_steps(h, DEFAULT_X_RANGE))

        if abs(result1 - result2) < accuracy:
            return h

    return h * 2
```

Після запуску програми, бачимо, що алгоритм визначив довжину кроку $h = 0.0031$. Кількість кроків – 320. Результат: 0.19629823.

```
Method: Trapezoidal
Steps: 320
Length of step: 0.0031
Result: 0.19629823
```

Перевіримо різницю між результатами Mathcad Prime 9 та розробленою функцією.

$$a := 0 \qquad b := 1 \qquad result := \int_a^b x^2 \sqrt{1-x^2}\, dx = 0.1963$$

$$TrapezoidResult := 0.19629823$$

$$Error := \left| TrapezoidResult - result \right| = 0.00005135$$

Реальна похибка менше аналітичної (задана точність), тому результати влаштовують.

2. Квадратурний метод Гауса-Лежандра

Схема інтегрування Гаусса є дуже ефективним методом для виконання чисельного інтегрування за інтервалами. Насправді, якщо функція, яку потрібно інтегрувати, є поліномом відповідного ступеня, то схема інтегрування Гаусса дає точні результати. Схема інтегрування Гаусса реалізована майже в кожному програмному забезпеченні аналізу кінцевих елементів завдяки її простоті та ефективності обчислень.

Квадратура Гаусса має на меті знайти «найменшу» кількість фіксованих точок для апроксимації інтеграла функції $f\colon [-1;1] \to \mathbb{R}$ такої, що:

$$I = \int_{-1}^{1} f \,\mathrm{d}x \approx \sum_{i=1}^{n} w_i f(x_i)$$

Де $\forall 1 \le i \le n : x_i \in [-1,1]$ і $w_i \in R$. Крім того, $\forall i\colon x_i$ називається точкою інтеграції, а w_i називається пов'язаною вагою. Кількість точок інтегрування та відповідні вагові коефіцієнти вибираються відповідно до складності функції f, яку потрібно інтегрувати. Оскільки загальний поліном ступеня $2n-1$ має $2n$ коефіцієнтів, можна знайти схему інтегрування Гаусса з n кількістю точок інтегрування та n кількістю пов'язаних ваг, щоб точно інтегрувати цю поліноміальну функцію на інтервалі $[-1,1]$.

Квадратуру Гаусса дуже легко реалізувати і вона забезпечує дуже точні результати з невеликою кількістю обчислень. Однак один недолік полягає в тому, що він не застосовується до даних, отриманих експериментально, оскільки значення функції в конкретних точках інтегрування не обов'язково будуть доступними. Щоб реалізувати схему інтегрування Гауса в Python, спочатку створюється таблиця, яка містить точки інтегрування та відповідні ваги. Рядок під номером i містить схему інтегрування Гаусса з $i-1$ точками інтегрування. Потім у Python створюється процедура, вхідною інформацією якої є функція f і необхідна кількість точок інтегрування. Потім процедура викликає відповідний рядок у «Таблиці Гауса» та обчислює зважену суму функції, обчисленої у відповідних точках інтегрування.

Наведена вище схема інтегрування Гаусса застосовується до функцій, які інтегруються на інтервалі $[-1,1]$. Проста заміна змінних може бути використана для інтегрування функції $g(z)$, де $z \in [a,b]$. У цьому випадку лінійна залежність між z і x може бути виражена як:

$$\frac{z-a}{b-a} = \frac{x-(-1)}{1-(-1)} = \frac{x+1}{2}$$

Тому:

$$\mathrm{d}z = \frac{(b-a)}{2}\mathrm{d}x$$

Це означає, що інтегрування можна перетворити з інтегрування по z на інтегрування по x таким чином:

$$\int_a^b g(z)\,\mathrm{d}z = \int_{-1}^{1} g(z(x))\,\frac{(b-a)}{2}\mathrm{d}x$$

Де:

$$z(x) = \frac{(b-a)(x+1)}{2} + a = \frac{(b-a)(x) + (b+a)}{2}$$

Тому:

$$\int_a^b g(z)\,\mathrm{d}z = \int_{-1}^{1} g\left(\frac{(b-a)(x) + (b+a)}{2}\right)\frac{(b-a)}{2}\mathrm{d}x$$

Тоді інтегрування може продовжуватися відповідно до вагових коефіцієнтів і значень точок інтегрування з $f(x)$, заданих як:

$$f(x) = g\left(\frac{(b-a)(x) + (b+a)}{2}\right)\frac{(b-a)}{2}$$

Для реалізації методу Гауса-Лежандра скористаємось Python 3 та Jupyter Notebook для візуалізації. Також, знадобляться бібліотеки pandas, sympy, scipy та numpy.

Для початку копіюємо значення таблиці Гауса за посиланням (https://pomax.github.io/bezierinfo/legendre-gauss.html). Для отримання похибки близької до потрібної, найближча кількість точок інтегрування – 17:

```
# https://pomax.github.io/bezierinfo/legendre-gauss.html
GaussTable = [[[0], [2]],
              [[-1/np.sqrt(3), 1/np.sqrt(3)], [1, 1]],
              [[-np.sqrt(3/5), 0, np.sqrt(3/5)], [5/9, 8/9, 5/9]],
              [[-0.861136, -0.339981, 0.339981, 0.861136], [0.347855, 0.652145, 0.652145, 0.347855]],
              [[-0.90618, -0.538469, 0, 0.538469, 0.90618], [0.236927, 0.478629, 0.568889, 0.478629, 0.236927]],
              [[-0.93247, -0.661209, -0.238619, 0.238619, 0.661209, 0.93247], [0.171324, 0.360762, 0.467914, 0.467914,
               0.360762,  0.171324]],
              [[0.94910, -0.94910, 0.741531, -0.74153, -0.40584, 0.40584, 0], [0.12948, 0.129484, 0.27970, 0.27970, 0.38183,
               0.38183, 0.41795]],
              [[0.96028, -0.96028, 0.79666, -0.79666, 0.52553, -0.52553, 0.18343, -0.18343], [0.10122, 0.10122, 0.22238,
               0.22238, 0.31370, 0.31370, 0.36268, 0.36268]],
              [[0.613371, -0.61337, 0.32425, -0.32425, 0.96816, -0.96816, 0.83603, -0.83603, 0],[0.26061, 0.26061, 0.31234,
               0.31234, 0.08127, 0.08127, 0.18064, 0.18064, 0.33023]],
              [[0.97390, -0.97390, 0.86506, -0.86506, 0.67940, -0.67940, 0.43339, -0.43339, 0.14887, -0.14887], [0.06667,
               0.06667, 0.14945, 0.14945, 0.21908, 0.21908, 0.26926, 0.26926, 0.29552, 0.29552]],
              [[0.97822, -0.97822, 0.88706, -0.88706, 0.73015, -0.73015, 0.51909, -0.51909, 0.26954, -0.26954, 0],[0.05566,
               0.05566, 0.12558, 0.12558, 0.18629, 0.18629, 0.23319, 0.23319, 0.26280, 0.26280, 0.27292]],
              [[0.98156, -0.98156, 0.90411, -0.90411, 0.76990, -0.76990, 0.58731, -0.58731, 0.36783, -0.36783, 0.12523,
               -0.12523], [0.04717, 0.04717, 0.10693, 0.10693, 0.16007, 0.16007, 0.20316, 0.20316, 0.23349, 0.23349, 0.24914,
                0.24914]],
              [[0.98418, -0.98418, 0.91759, -0.91759, 0.80157, -0.80157, 0.64234, -0.64234, 0.44849, -0.44849, 0.23045,
               -0.23045, 0], [0.04048, 0.04048, 0.09212, 0.09212, 0.13887, 0.13887, 0.17814, 0.17814, 0.20781, 0.20781,
               0.22628, 0.22628, 0.23255]],
              [[0.98628, -0.98628, 0.92843, -0.92843, 0.82720, -0.82720, 0.68729, -0.68729, 0.51524, -0.51524, 0.31911,
               -0.31911, 0.10805, -0.10805], [0.03511, 0.03511, 0.08015, 0.08015, 0.12151, 0.12151, 0.15720, 0.15720,
               0.18553, 0.18553, 0.20519, 0.20519, 0.21526, 0.21526]],
```

Проводимо розрахунки відповідно до формул, наведених вище:

```
display(pd.DataFrame(GaussTable, columns=["Integration Points", "Corresponding Weights"]))

def IGAL(f, n, a, b):
  n = int(n)
  return sum([(b - a)/2*GaussTable[n - 1][1][i]*f((b - a)/2*(GaussTable[n - 1][0][i] + 1) + a) for i in range(n)])

def f(x): return x**2 * math.sqrt(1 - x ** 2)

lo = 0
hi = 1
Iexact, error = integrate.quad(f, lo, hi)
print("Iexact: ",Iexact)
Itable = [[i + 1, sp.N(IGAL(f, i + 1, lo, hi)), (Iexact - IGAL(f, i + 1, lo, hi))/Iexact] for i in range(len(GaussTable))]
Itable = pd.DataFrame(Itable, columns=["Number of Integration Points",  "Numerical Integration Results", "Relative Error"])

display(Itable)
Executed at 2023.12.10 21:20:33 in 112ms
```

Отримуємо результати:

7-17 17 rows × 2 columns pd.DataFrame CSV

	Integration Points	Corresponding Weights
9	[0.9739, -0.9739, 0.86506, -0.86506, 0.6794, -...	[0.06667, 0.06667, 0.14945, 0.14945, 0.21908, ...
10	[0.97822, -0.97822, 0.88706, -0.88706, 0.73015...	[0.05566, 0.05566, 0.12558, 0.12558, 0.18629, ...
11	[0.98156, -0.98156, 0.90411, -0.90411, 0.7699,...	[0.04717, 0.04717, 0.10693, 0.10693, 0.16007, ...
12	[0.98418, -0.98418, 0.91759, -0.91759, 0.80157...	[0.04048, 0.04048, 0.09212, 0.09212, 0.13887, ...
13	[0.98628, -0.98628, 0.92843, -0.92843, 0.8272,...	[0.03511, 0.03511, 0.08015, 0.08015, 0.12151, ...
14	[0.98799, -0.98799, 0.93727, -0.93727, 0.8482,...	[0.03075, 0.03075, 0.07036, 0.07036, 0.10715, ...
15	[0.9894, -0.9894, 0.94457, -0.94457, 0.86563, ...	[0.02715, 0.02715, 0.06225, 0.06225, 0.09515, ...
16	[0.99057, -0.99057, 0.95067, -0.95067, 0.88023...	[0.02414, 0.02414, 0.05545, 0.05545, 0.08503, ...

Iexact: 0.196349540849358

12-17 17 rows × 3 columns pd.DataFrame CSV

	Number of Integration Points	Numerical Integration Results	Relative Error
11	12	0.196419032123673	-0.000354
12	13	0.196406197651897	-0.000289
13	14	0.196390388166801	-0.000208
14	15	0.196383903678668	-0.000175
15	16	0.196375675779901	-0.000133
16	17	0.196370237638804	-0.000105

Результат: 0.196370237638804, реальна похибка: $|-0.000105|$.

Похибка вираховувалась за допомогою порівняння з точним значенням. Точне значення було отримане за допомогою методу бібліотеки SciPy.

Порівняння результатів з Mathcad:

$$GaussResult := 0.196370237638804$$
$$ErrorGauss := |GaussResult - result| = 0.00002$$

Реальна похибка менша, ніж задана точність, і менша ніж після використання методу трапецій.

Висновок: За результатами цієї лабораторної роботи були набуті практичні навички в області чисельного інтегрування. Була проведена робота з методом трапецій та методом Гауса-Лежандра. Було виявлено, що останній метод вимагає менше ітерацій при обчисленні результату, і при цьому дає точніший результат.

Лістинг

trapezoidal_method.py:

```
import utils


def start(function, x_range, accuracy):
    h = utils.get_delta(algorithm, function, accuracy)
    steps = utils.get_steps(h, x_range)

    result = algorithm(function, x_range, h, steps)
    print(f"Method: Trapezoidal\n"
          f"Steps: {steps:d}\n"
          f"Length of step: {h:.4f}\n"
          f"Result: {result:.8f}\n")


# Implementation of trapezoidal method
def algorithm(function, x_range, h, steps):
    lo, hi = x_range

    # Finding sum
    integration = function(lo) + function(hi)

    for i in range(1, steps):
        k = lo + i * h
        integration = integration + 2 * function(k)

    # Finding final integration value
    integration = integration * h / 2

    return integration
```

utils.py:

```
def get_steps(h, x_range):
    lo, hi = x_range
    return round((hi - lo) / h)


def get_delta(algorithm, function, accuracy):
    MIN_H = 0.00001
    MAX_H = 1

    DEFAULT_X_RANGE = [0, 1]

    h = 0.1
    while MAX_H >= h > MIN_H:

        result1 = algorithm(function, DEFAULT_X_RANGE, h, get_steps(h,
DEFAULT_X_RANGE))

        h = h / 2
        result2 = algorithm(function, DEFAULT_X_RANGE, h, get_steps(h,
DEFAULT_X_RANGE))

        if abs(result1 - result2) < accuracy:
            return h

    return h * 2
```

gauss_quadrature.ipynb:

```
import math
```

```
      import numpy as np
      import sympy as sp
      import pandas as pd
      from scipy import integrate

      # https://pomax.github.io/bezierinfo/legendre-gauss.html
      GaussTable = [[[0], [2]],
                    [[-1/np.sqrt(3), 1/np.sqrt(3)], [1, 1]],
                    [[-np.sqrt(3/5), 0, np.sqrt(3/5)], [5/9, 8/9, 5/9]],
                    [[-0.861136, -0.339981, 0.339981, 0.861136], [0.347855, 0.652145,
0.652145, 0.347855]],
                    [[-0.90618, -0.538469, 0, 0.538469, 0.90618], [0.236927, 0.478629,
0.568889, 0.478629, 0.236927]],
                    [[-0.93247, -0.661209, -0.238619, 0.238619, 0.661209, 0.93247],
[0.171324, 0.360762, 0.467914, 0.467914, 0.360762,  0.171324]],
                    [[0.94910, -0.94910, 0.741531, -0.74153, -0.40584, 0.40584, 0],
[0.12948, 0.129484, 0.27970, 0.27970, 0.38183, 0.38183, 0.41795]],
                    [[0.96028, -0.96028, 0.79666, -0.79666, 0.52553, -0.52553, 0.18343,
-0.18343], [0.10122, 0.10122, 0.22238, 0.22238, 0.31370, 0.31370, 0.36268, 0.36268]],
                    [[0.613371, -0.61337, 0.32425, -0.32425, 0.96816, -0.96816,
0.83603, -0.83603, 0],[0.26061, 0.26061, 0.31234, 0.31234, 0.08127, 0.08127, 0.18064,
0.18064, 0.33023]],
                    [[0.97390, -0.97390, 0.86506, -0.86506, 0.67940, -0.67940, 0.43339,
-0.43339, 0.14887, -0.14887], [0.06667, 0.06667, 0.14945, 0.14945, 0.21908, 0.21908,
0.26926, 0.26926, 0.29552, 0.29552]],
                    [[0.97822, -0.97822, 0.88706, -0.88706, 0.73015, -0.73015, 0.51909,
-0.51909, 0.26954, -0.26954, 0],[0.05566, 0.05566, 0.12558, 0.12558, 0.18629, 0.18629,
0.23319, 0.23319, 0.26280, 0.26280, 0.27292]],
                    [[0.98156, -0.98156, 0.90411, -0.90411, 0.76990, -0.76990, 0.58731,
-0.58731, 0.36783, -0.36783, 0.12523, -0.12523], [0.04717, 0.04717, 0.10693, 0.10693,
0.16007, 0.16007, 0.20316, 0.20316, 0.23349, 0.23349, 0.24914, 0.24914]],
                    [[0.98418, -0.98418, 0.91759, -0.91759, 0.80157, -0.80157, 0.64234,
-0.64234, 0.44849, -0.44849, 0.23045, -0.23045, 0], [0.04048, 0.04048, 0.09212,
0.09212, 0.13887, 0.13887, 0.17814, 0.17814, 0.20781, 0.20781, 0.22628, 0.22628,
0.23255]],
                    [[0.98628, -0.98628, 0.92843, -0.92843, 0.82720, -0.82720, 0.68729,
-0.68729, 0.51524, -0.51524, 0.31911, -0.31911, 0.10805, -0.10805], [0.03511, 0.03511,
0.08015, 0.08015, 0.12151, 0.12151, 0.15720, 0.15720, 0.18553, 0.18553, 0.20519,
0.20519, 0.21526, 0.21526]],
                    [[0.98799, -0.98799, 0.93727, -0.93727, 0.84820, -0.84820, 0.72441,
-0.72441, 0.57097, -0.57097, 0.39415, -0.39415, 0.20119, -0.20119, 0], [0.03075,
0.03075, 0.07036, 0.07036, 0.10715, 0.10715, 0.13957, 0.13957, 0.16626, 0.16626,
0.18616, 0.18616, 0.19843, 0.19843, 0.20257]],
                    [[0.98940, -0.98940, 0.94457, -0.94457, 0.86563, -0.86563, 0.75540,
-0.75540, 0.61787, -0.61787, 0.45801, -0.45801, 0.28160, -0.28160, 0.09501, -0.09501],
[0.02715, 0.02715, 0.06225, 0.06225, 0.09515, 0.09515, 0.12462, 0.12462, 0.14959,
0.14959, 0.16915, 0.16915, 0.18260, 0.18260, 0.18945, 0.18945]],
                    [[0.99057, -0.99057, 0.95067, -0.95067, 0.88023, -0.88023, 0.78151,
-0.78151, 0.65767, -0.65767, 0.51269, -0.51269, 0.35123, -0.35123, 0.17848, -0.17848,
0], [0.02414, 0.02414, 0.05545, 0.05545, 0.08503, 0.08503, 0.11188, 0.11188, 0.13513,
0.13513, 0.15404, 0.15404, 0.16800, 0.16800, 0.17656, 0.17656, 0.17944]]
                    ]

      display(pd.DataFrame(GaussTable, columns=["Integration Points", "Corresponding
Weights"]))

      def IGAL(f, n, a, b):
        n = int(n)
        return sum([(b - a)/2*GaussTable[n - 1][1][i]*f((b - a)/2*(GaussTable[n -
1][0][i] + 1) + a) for i in range(n)])

      def f(x): return x**2 * math.sqrt(1 - x ** 2)

      lo = 0
```

```
      hi = 1
      Iexact, error = integrate.quad(f, lo, hi)
      print("Iexact: ",Iexact)
      Itable = [[i + 1, sp.N(IGAL(f, i + 1, lo, hi)), (Iexact - IGAL(f, i + 1, lo,
hi))/Iexact] for i in range(len(GaussTable))]
      Itable = pd.DataFrame(Itable, columns=["Number of Integration Points",
"Numerical Integration Results", "Relative Error"])

      display(Itable)
```