Akademia Górniczo-Hutnicza
Wydział Informatyki
Kierunek Informatyka

Metody Obliczeniowe w Nauce i Technice

Laboratorium 6

Kwadratury

Wojciech Michaluk, Kyrylo Iakymenko

Kraków, 12 kwietnia 2024

1 Wprowadzenie

Celem zadań w ramach tego laboratorium jest wykorzystanie metod całkowania numerycznego do obliczenia przybliżonej wartości liczby π oraz porównanie różnych kwadratur złożonych. W pierwszym zadaniu wykorzystujemy znaną równość:

$$\int_0^1 \frac{4}{1+x^2} dx = \pi.$$

Przeanalizujemy, jak zmiana liczby ewaluacji funkcji podcałkowej $f(x) = \frac{4}{1+x^2}$ oraz krok h wpływa na dokładność obliczeń na przykładzie *metody prostokątów* (midpoint), *metody trapezów* i *metody Simpsona*, porównując również empiryczne rzędy zbieżności z teoretycznymi wartościami.

W drugim zadaniu obliczamy wspomnianą całkę metodą Gaussa-Legendre'a. Wiedząc, że jej wartość wynosi π, analogicznie jak w zadaniu **pierwszym**, wyznaczamy błąd względny metody w zależności od liczby ewaluacji funkcji. Nastepnie, przedstawiamy wyniki na wykresie wspólnie z wykresami błędu metod z poprzedniego zadania.

2 Zadanie 1

2.1 Opis zadania

Do obliczenia tej całki wykorzystujemy trzy metody całkowania numerycznego:

1. złożoną kwadraturę otwartą prostokątów (metoda mid-point),
2. złożona kwadratura trapezów,
3. oraz złożona kwadratura Simpsona - metoda ta jest skuteczną techniką numeryczną, która pozwala na przybliżone obliczenie wartości całki, szczególnie w przypadku funkcji, dla których nie ma znanej całki analitycznej.

Złożone kwadratury charakteryzuja się one tym, że dzielimy nasz rozważany przedział $[0; 1]$ na pewną liczbę podprzedziałów, a w każdym z nich liczymy wartość korzystając z odpowiedniej metody prostej, następnie dodajemy te wartości, żeby uzyskać finalny wynik. Dla podprzedziału $[a_i, b_i]$ wzory kwadratur prostych wyglądają następująco:

- metoda prostokątów $\rightarrow \int_{a_i}^{b_i} f(x)dx \approx f(\frac{a_i+b_i}{2}) \cdot (b_i - a_i)$
- metoda trapezów $\rightarrow \int_{a_i}^{b_i} f(x)dx \approx \frac{1}{2}[f(a_i) + f(b_i)] \cdot (b_i - a_i)$
- metoda Simpsona $\rightarrow \int_{a_i}^{b_i} f(x)dx \approx \frac{1}{6}(b_i - a_i)[f(a_i) + 4f(\frac{a_i+b_i}{2}) + f(b_i)]$

Następnie wyznaczamy wartość bezwzględnego błędu względnego dla każdej z tych metod w zależności od liczby ewaluacji funkcji podcałkowej, która wynosi $n + 1 = 2^m + 2$ dla $m = 1, 2, ..., 25$. Zaprezentujemy wyniki na wspólnym wykresie, korzystając z logarytmicznej skali na obu osiach oraz je przeanalizujemy.

2.2 Opracowanie zadania

Zaczniemy od zdefiniowania badanych funkcji. W metodzie trapezów i w metodzie Simpsona korzystamy z bibliotecznych funkcji `trapz` oraz `simps` z pakietu `scipy.integrate`.

- Kwadratura otwarta prostokątów:

```
def midpoint_rule(f, a, b, m):
        h = (b - a) / m
        nodes = np.linspace(a + h / 2, b - h / 2, m)
        return h * np.sum(f(nodes)) # f to funkcja podcałkowa
```

Listing 1: Implementacja metody prostokątów

- Metoda trapezów:

```
def trapezoidal_rule(f, a, b, m):
        x = np.linspace(a, b, m)
        y = f(x)
        return trapz(y, x)
```

Listing 2: Implementacja metody trapezów

- Metoda Simpsona:

```
def simpsons_rule(f, a, b, m):
        x = np.linspace(a, b, m)
        y = f(x)
        return simps(y, x)
```

Listing 3: Implementacja metody Simpsona

W poniższej tabeli pokazujemy częściowe wyniki aproksymacji naszej całki:

m	**Midpoint Rule**	**Trapezoidal Rule**	**Simpson's Rule**
1	3.150849	3.100000	3.133333
2	3.144926	3.131176	3.141569
3	3.143293	3.136963	3.141592
4	3.142621	3.138988	3.141593
5	3.142281	3.139926	3.141593
6	3.142086	3.140435	3.141593
7	3.141963	3.140742	3.141593
8	3.141881	3.140942	3.141593
9	3.141823	3.141078	3.141593
10	3.141782	3.141176	3.141593
11	3.141750	3.141248	3.141593
12	3.141726	3.141303	3.141593
13	3.141707	3.141346	3.141593
14	3.141692	3.141380	3.141593

Tabela 1: Wartości przybliżenia całki dla wszystkich metod dla początkowych wartości m.

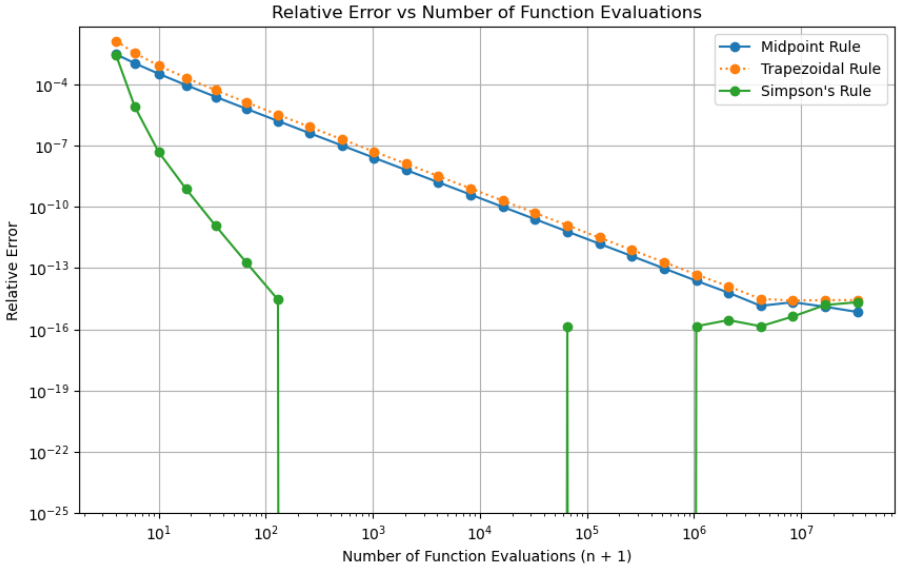
Nawet na oko widać, że metoda Simpsona jest najbardziej dokładna. Nawet dla $m = 1$ metoda Simpsona jest dokładniejsza w porównaniu do innych metod, a od $m = 4$ pierwsze 6 cyfr po przecinku się nie zmienia.

Teraz policzmy błędy względne naszych metod.

```
def relative_error(approx_val, actual_val):
        return np.abs((actual_val - approx_val) / actual_val)
```

Listing 4: Obliczanie błędu względnego

Przedstawiamy wyniki poniżej na wykresie w zależności od $n+1 = \frac{1}{h} + 1$ liczby ewaluacji funkcji w naszej metodzie liczenia całki.



Rysunek 1: Błąd względny metod numerycznego całkowania w zależności od liczby ewaluacji funkcji

Na wykresie widać oczywiste podobieństwo błędu metod prostokątów i trapezów. Wykresy ich błędów są w zasadzie nierozróżnialne. Wyróżnia się w naszym porównaniu metoda Simpsona. Szybciej zbiega do wartości bliskich zera i jako jedyna osiąga wartość 0 (co oznacza, że jej dokładność dla pewnej liczby ewaluacji byłaby większa od precyzji float64). Nie widzimy dokładnie wartości 0 ze względu na skalę logarytmiczną. Można by było użyć argumentu *symlog* przy ustawianiu skali na osi Y, ale sprawdziliśmy tę opcję i zmniejsza ona czytelność wykresu - stąd pozostajemy przy tym podejściu.

Teraz wyznaczmy wartości h_{min}, czyli największe wartości h, dla których błąd osiąga swoje minimum, dla wszystkich trzech metod. Z wykresu możemy odczytać pewne przybliżenie wartości n, ale zróbmy to bardziej dokładnie i konkretnie.

```
# funkcja licząca h_min dla podanych danych
def h_min(dataList):
        return 1 / evaluations[np.argmin(dataList)]
```

Listing 5: Wyznaczanie wartości h_{min}

Korzystamy z zależności $h = \frac{1}{n}$. Wyniki przedstawiamy poniżej.

Metoda	h_{min}
Prostokątów	$2.98 \cdot 10^{-8}$
Trapezów	$1.19 \cdot 10^{-7}$
Simpsona	$3.8 \cdot 10^{-3}$

Tabela 2: Wartości h_{min} dla wszystkich badanych metod

Jak można było oczekiwać, analizując wykres metody prostokątów i metody trapezów, mają bardzo podobną wartość h_{min}. Z drugiej strony metoda Simpsona wyróżnia się od innych w lepszą stronę dużo większym h_{min}, potwierdzenie czego można również zobaczyć na wykresie, patrząc na to, jak szybko wartości błędu zbiegają do 0. Oznacza to, że stosując metodę Simpsona, możemy używać dużo większych odstępów h (kilka rzędów większych w porównaniu do innych badanych metod), a co z tego wynika również mniejszej ilości ewaluacji funkcji, żeby uzyskać podobną (w naszym przypadku nawet lepszą) dokładność oszacowania. Nasze uzyskane wartości h_{min} są porównywalne (przynajmniej dla metody prostokątów i trapezów) z wartościami uzyskanymi w laboratorium 1.

Teraz przejdziemy do badania rzędów zbieżności naszych metod. W celu uzyskania dokładniejszej wartości, policzymy je dla wszystkich wartości m i zrobimy analizę uzyskanych wyników.

Empiryczny rząd zbieżności p obliczamy z zależności:

$$E(h) \approx Ch^p \mid \text{logarytmujemy obustronnie}$$
$$\log E(h) \approx \log C + p \log h,$$

gdzie C to pewna stała. Korzystając z tego wzoru dla pewnych dwóch wartości h_1 i h_2, otrzymujemy $p \approx \frac{\log(\frac{E(h_2)}{E(h_1)})}{\log(\frac{h_2}{h_1})}$.

Staramy się dobrać wartości h_1 i h_2 tak, aby pochodziły one z zakresu, w którym błąd metody przeważa nad błędem numerycznym - żeby wyniki obliczeń miały sens.
Przedstawiamy częściowe wyniki dla metody prostokątów oraz metody trapezów:

m	**Rząd zbieżności (metoda prostokątów)**
2	2.51907288
3	2.3012549
4	2.16400672
5	2.08586731
6	2.04397892
7	2.02226184
8	2.01120048
9	2.00561782
10	2.00281333
11	2.00140774
12	2.00070411
13	2.0003523
14	2.00017673

m	**Rząd zbieżności (metoda trapezów)**
2	3.41475228
3	2.71374569
4	2.35849453
5	2.17974635
6	2.09001423
7	2.04504478
8	2.02253214
9	2.01126855
10	2.00563490
11	2.00281761
12	2.00140892
13	2.00070417
14	2.00035273

Tabela 3: Rzędy zbieżności dla metody prostokątów i metody trapezów

Jak widzimy w obu przypadkach, nasz ciąg zmierza do wartości 2, która jest wartością teoretyczną rzędu zbieżności zarówno dla metody prostokątów, jak i dla metody trapezów.

Przechodzimy teraz do metody Simpson'a. Tutaj skorzystamy tylko z pierwszych 6 wartości m (począwszy od 2), gdyż dla kolejnych pojawiają się wartości niedodatnie.

m	**Rząd zbieżności (metoda Simpson'a)**
2	14.40310177
3	9.92266209
4	7.07294318
5	6.53905148
6	6.27040509
7	6.15800594

Tabela 4: Wartości empirycznego rzędu zbieżności dla metody Simpson'a w zależności od wartości m

Porównując uzyskany ciąg z wartością teoretyczną równą 4, dochodzimy do wniosku, że w naszym przypadku rząd zbieżności metody Simpson'a jest różny od wartości teoretycznej, gdyż w naszym przypadku zbiega raczej do wartości bliskiej 6. Może tak być dlatego, że precyzja obliczeń nie pozwala na sprawdzenie wartości rzędu zbieżności dla kolejnych wartości m. Ale warto zauważyć, że wartości, które możemy zbadać empirycznie cały czas maleją, co może wskazywać na to, że z czasem nasz ciąg rzeczywiście dojdzie do wartości teoretycznej.

3 Zadanie 2

3.1 Opis zadania

W drugiej części zadania, stosujemy metodę Gaussa-Legendre'a do obliczenia wartości tej samej całki. Również analizujemy wartość bezwzględną błędu względnego w zależności od liczby ewaluacji funkcji podcałkowej i porównujemy wyniki z wcześniejszymi metodami.

W metodzie Gaussa-Legendre'a korzystamy z następującego sposobu obliczania całki:

$$\int_{-1}^{1} f(\xi)d\xi \approx \sum_{i=1}^{n} A_i f(\xi_i),$$

gdzie ξ_i to pierwiastki *n-tego* wielomianu Legendre'a, natomiast A_i to odpowiednie współczynniki, których sposób wyliczenia omawiamy w następnej sekcji. Ponieważ obliczamy całkę w przedziale $[0; 1]$, stosujemy transformację:

$$\int_{0}^{1} f(x)dx \approx \frac{1}{2} \sum_{i=1}^{n} A_i f(x_i) \text{ dla } x_i = \frac{1}{2} + \frac{1}{2}\xi_i.$$

3.2 Opracowanie zadania

W pierwszym podejściu do rozwiązania zadanego problemu postanowiliśmy ręcznie zaimplementować potrzebne funkcje, co okazało się niemałym wyzwaniem. Istnieje bowiem funkcja biblioteczna `scipy.special.roots_legendre`, która zwraca parę: (pierwiastki wielomianu, szukany wektor wag A). Rozwiązuje ona problem budowania wielomianu Legendre'a i szukania jego pierwiastków, a następnie rozwiązania równania macierzowego w celu znalezienia współczynników wagowych.

Chcąc zaimplementować funkcje samodzielnie, korzystamy z tego, że wielomiany Legendre'a są ortogonalne, co ułatwia dalsze obliczenia. Spełniają one rekurencyjną zależność:

$$\begin{aligned} P_0(x) &= 1 \\ P_1(x) &= x \\ P_{i+1} &= \frac{2i+1}{i+1} x P_i - \frac{i}{i+1} P_{i-1} \text{ dla } i \geq 1, \end{aligned}$$

którą wykorzystamy do wyznaczania ich współczynników.

Przedstawmy kod naszych funkcji:

```
@cache
def legendre_coeffs(i): #obliczanie współczynników wielomianu
    if i == 0: return np.array([1])
    elif i == 1: return np.array([1, 0])
    else:
        lshift = np.concatenate(((2 * i - 1) / i
          * legendre_coeffs(i - 1), np.zeros(1)))
        rshift = np.concatenate((np.zeros(2), (i - 1) / i
          * legendre_coeffs(i - 2)))
        return lshift - rshift

def legendre_roots(n): #obliczanie pierwiastków wielomianu
    coeffs = legendre_coeffs(n)
    return np.roots(coeffs)

def gauss_legendre(A, roots, n): #obliczanie całki metodą Gaussa - Legendre'a
    transform_roots = 0.5 + 0.5 * roots
    return 0.5 * A @ f(transform_roots)
```

Listing 6: Funkcje pomocnicze do metody Gaussa-Legendre'a

Współczynniki A otrzymujemy, rozwiązując równanie macierzowe:

$$\begin{bmatrix} P_0(x_1) & P_0(x_2) & \cdots & P_0(x_n) \\ P_1(x_1) & P_1(x_n) & \cdots & P_1(x_n) \\ \vdots & \vdots & \ddots & \vdots \\ P_{n-1}(x_1) & P_{n-1}(x_2) & \cdots & P_{n-1}(x_n) \end{bmatrix} \cdot \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix} = \begin{bmatrix} \int_{-1}^{1} P_0(x)dx \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

W tym przypadku x_i dla $i = 1, 2, ..., n$ to węzły wielomianu Legendre'a uporządkowane **malejąco**, natomiast wartości 0 w kolumnie po prawej stronie wynikają z ortogonalności. Pierwszy wyraz z kolei sprowadza się do trywialnej całki $\int_{-1}^{1} dx = 2$.

Spójrzmy na kod:

```
ns = [5, 10, 50, 100] #przykładowe wartości

for n in ns:
    x = np.flip(np.sort(legendre_roots(n)))

    S = np.array([[np.polyval(legendre_coeffs(i), x[j]) for i in range(n)]
      for j in range(n)])
    R = np.concatenate((np.array([2]), np.zeros(n - 1)))
    A = np.linalg.solve(S, R) # SA = R

    value = gauss_legendre(A, x, n)
```

Listing 7: Czy obliczone wartości będą dokładne?

Jak się okazuje, jest wiele problemów, głównie wynikających z użycia funkcji `np.roots`. Oblicza ona pierwiastki wielomianu, bazując na wartościach własnych odpowiadającej wielomianowi macierzy. Powoduje to nie tylko to, że pierwiastki są w "losowej" kolejności (i trzeba je posortować), ale także obliczone wartości są bardzo niedokładne, a dla $n \geq 45$ zaczynaja pojawiać się składowe urojone w pierwiastkach. Z tego powodu wyniki znacznie odbiegają od prawidłowej wartości, która wynosi π.

Aby zwrócić uwagę na to ciekawe zjawisko, prezentujemy uzyskane wyniki poniżej.

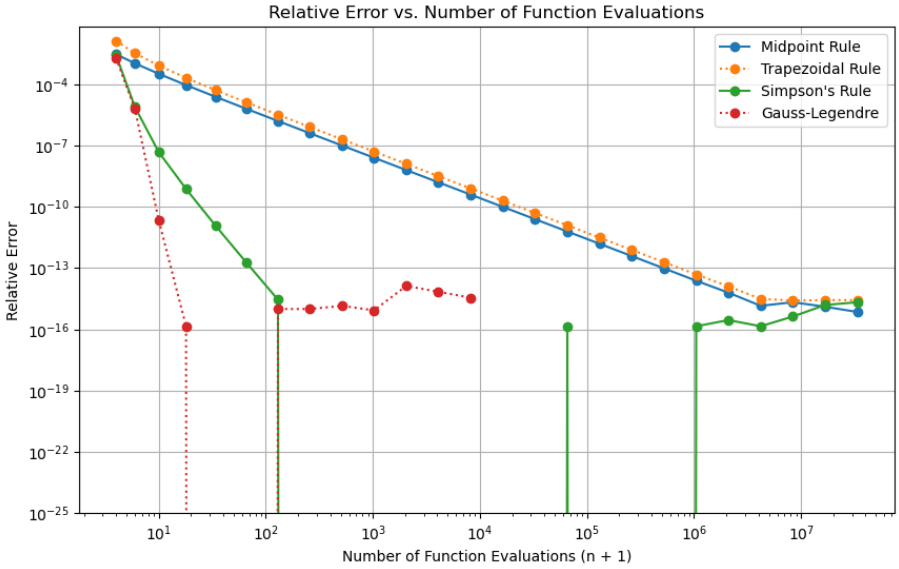
n	**Obliczona wartość**
5	5.068129
10	5.123556
50	(0.326415 - 0.439230i)
100	(-2.867405 $\cdot 10^{-7}$ + 1.735485i $\cdot 10^{-7}$)

Tabela 5: Wyniki dla ręcznej implementacji - niezbyt dokładne!

Korzystając ze wspomnianej funkcji bibliotecznej, obliczenia stają się znacznie prostsze (i dokładniejsze). Dla przykładu, wyniki obliczone dla wartości n z powyższej tabeli są identyczne dla wszyskich n z dokładnością do 6 cyfr po przecinku i wynoszą 3.141593.

Można zauważyć, że w przypadku tej metody dla poprawnej implementacji już dla stosunkowo małych n uzyskuje się zaskakująco dobrą dokładność obliczeń. W celu porównania tego podejścia z metodami z zadania 1, przeprowadzamy obliczenia dla stosunkowo "dużej" liczby węzłów, bo dla 2^m węzłów, gdzie $m = 1, 2, .., 13$. Nie przyjmujemy tutaj dokładnie takiego samego zakresu, gdyż dla użytych wartości błąd numeryczny już w pewnym momencie zaczyna przeważać nad błędem metody oraz z powodu dużego kosztu obliczeniowego - już dla samego $m = 14$ obliczenia trwały kilka sekund.

Poniżej przedstawiamy wyniki na wspólnym wykresie, wraz z metodami z zadania 1.



Rysunek 2: Błąd względny metod numerycznego całkowania w zależności od liczby ewaluacji funkcji

Patrząc na uzyskany wykres widzimy, że wartości błędu kwadratury Gaussa-Legendre'a bardzo szybko zbiegają do wartości interpretowanych jako 0 (tzn. mniejszych niż dokładność reprezentacji `np.float64`). Z kolei dla wartości n rzędu 10^2 błąd numeryczny zaczyna odgrywać znaczącą rolę i osiąga bardzo podobny poziom co błędy innych metod - różnica jest taka, że jest on osiągany dla znacznie mniejszych wartości n.

4 Podsumowanie i wnioski

Podczas tego laboratorium eksperymentalnie zbadaliśmy różne metody numerycznego całkowania w celu obliczenia przybliżonej wartości liczby π oraz porównania ich skuteczności. W pierwszym zadaniu wykorzystaliśmy kwadratury złożone: metody prostokątów (midpoint), trapezów i Simpsona, analizując wpływ zmiany liczby ewaluacji funkcji podcałkowej na dokładność obliczeń. Wyniki pokazały, że metoda Simpsona wykazała się największą dokładnością, już dla niewielkiej liczby ewaluacji funkcji była ona niezwykle dokładna, a pozostałe metody zachowywały się podobnie, jeśli chodzi o błąd względny w zależności od liczby ewaluacji funkcji podcałkowej i na wykresie w skali logarytmicznej schodziły w zasadzie liniowo. Następnie, obliczyliśmy punkty gdzie wartości błędów względnych przyjęły minimum dla tych metod, co pozwoliło na wyznaczenie optymalnych wartości kroku h dla każdej z nich.

Analizując rzędy zbieżności metod doszliśmy do wniosku, że zarówno metoda prostokątów, jak i metoda trapezów dążą do rzędu zbieżności wynoszącego 2, co jest zgodne z teorią. Natomiast dla metody Simpsona, choć teoretycznie rząd zbieżności powinien być 4, otrzymane wyniki były bardziej podobne do ciągu dążącego do 6. To zjawisko może być związane z ograniczeniami precyzji obliczeń numerycznych. Również warto zwrócić uwagę na to, że nie byliśmy w stanie wyliczyć rzędu zbieżności metody Simpson'a dla wartości m większej od 7 dlatego, że błąd aproksymacji Simpson'a dla tych m czasami przyjmuje wartość 0.

W drugim zadaniu przetestowaliśmy metodę Gaussa-Legendre'a do obliczenia przybliżonej wartości π oraz przeprowadziliśmy analogiczną analizę błędu bezwzględnego w zależności od liczby ewaluacji funkcji podcałkowej, porównując wyniki z wcześniej badanymi metodami. Wykorzystanie tej metody, zwłaszcza przy użyciu odpowiednich bibliotek, pozwoliło osiągnąć bardzo dokładne wyniki nawet dla stosunkowo niewielkiej liczby węzłów. Jednak dla dużych wartości n błąd numeryczny stawał się znaczący i w połączeniu ze znacznie większym kosztem obliczeniowym, stosowanie tej metody może być zalecane tylko w przypadkach, gdy ilość ewaluacji funkcji podcałkowej jest stosunkowo mała (już dla $n \approx 10^4$ obliczenia zajmowały kilka sekund).

Podsumowując, eksperymenty wykazały, że metoda Gaussa-Legendre'a charakteryzowała się największą dokładnością w przypadku niewielkiej liczby ewaluacji funkcji, natomiast metoda Simpson'a mogła zapewnić dosyć dokładne wyniki w porównaniu do innych metod, zachowując przy tym stosunkowo niski koszt obliczeniowy, choć dla dużych wartości n błąd numeryczny okazał się znacząco podobny do błędu metody prostokątów i trapezów.

Literatura

[1] Materiały pomocnicze do laboratorium zamieszczone na platformie Teams w katalogu *lab06/lab6-intro.pdf*.

[2] Treść przedstawiona na wykładzie o kwadraturach.