

| | | | |
|---|---|------------------------|--|
| Wydział WIMiIP | Imię i nazwisko Mateusz Witkowski | Rok II | Grupa 4 |
| Temat: Całkowanie metodą Simpsona i Monte Carlo | | | Prowadzący dr hab. inż. Hojny Marcin, prof. AGH |
| Data ćwiczenia 02.04.20 | Data oddania 08.04.2020 | Data zaliczenia | OCENA |

1. Cel ćwiczenia.

Celem ćwiczenia było napisanie implementacji całkowania przy pomocy metody Monte Carlo i Simpsona.

2. Wprowadzenie teoretyczne

Całkowanie numeryczne – jest to metoda numeryczna polegająca na obliczaniu całek oznaczonych z pewnym przybliżeniem. Całka oznaczona to pole powierzchni pod wykresem funkcji w zadanym przedziale. Proste metody całkowania numerycznego polegają na przybliżeniu całki za pomocą odpowiedniej sumy ważonej wartości całkowanej funkcji w kilku punktach. Aby uzyskać dokładniejsze przybliżenie dzieli się przedział całkowania na niewielkie fragmenty. Ostateczny wynik jest sumą oszacowań całek w poszczególnych podprzedziałach. Najczęściej przedział dzieli się na równe podprzedziały.

Mając funkcję $f(x)$ całkę oznaczoną zapisujemy w następujący sposób:

$$\int_{x_p}^{x_k} f(x) dx$$

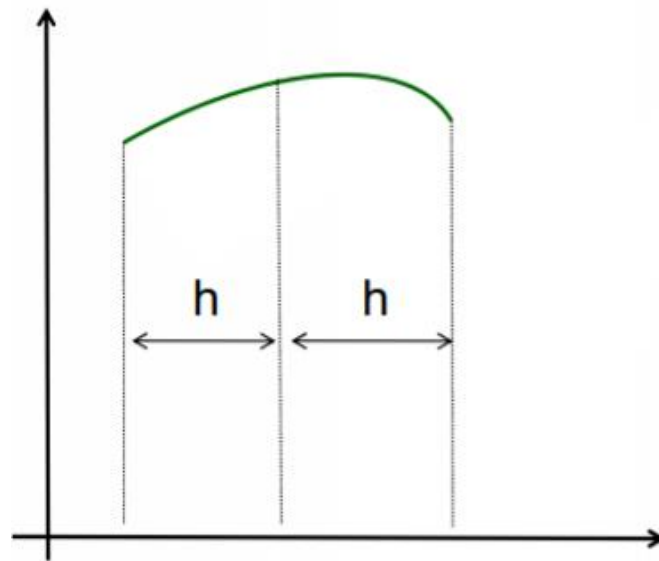
Gdzie:

x_p – dolna granica całkowania

x_k – górna granica całkowania

$f(x)$ – funkcja z której wyliczana jest całka

Metoda Simpsona jest jedna z najdokładniejszych metod całkowania przybliżonego. Funkcja podcałkowa w tym przypadku jest parabolą rozpiętą na krańcach przedziału oraz na jego środku. Metoda ta wykorzystuje interpolację wielomianem drugiego stopnia.



Rysunek 1. Metoda Simpsona -uproszczona

Szerokość pojedynczego podprzedziału wynika z ogólnej liczby podprzedziałów, dolnej (x_p) i górnej (x_k) granicy całki ze wzoru:

$$dx = \frac{x_k - x_p}{n} \quad (1)$$

Gdzie:

dx – szerokość podprzedziału

x_p – dolna granica całkowania

x_k – górna granica całkowania

n – liczba podprzedziałów

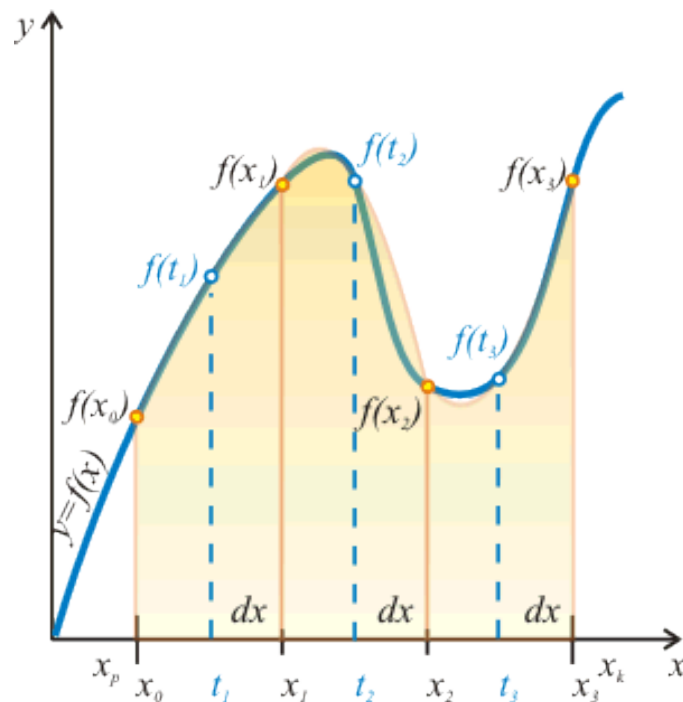
Znając szerokość możemy wyliczyć kolejne x :

$$x_i = x_p + i * dx, \text{ gdzie } i = 1, 2, \dots, n \quad (2)$$

W przypadku gdy nasz obszar całkowania jest podzielony tylko na dwie części możemy skorzystać ze wzoru:

$$\int_{x_p}^{x_k} f(x) dx \approx I = \left[f(x_p) + 4 \cdot f\left(\frac{x_p + x_k}{2}\right) + f(x_k) \right] \cdot \frac{dx}{3} \quad (3)$$

Często jednak w celu uzyskania dokładniejszych wyników dzielimy przedziały całkowania na większą ilość części, mamy do czynienia wtedy ze złożoną metodą Simpsona.



Rysunek 2. Złożona metoda Simpsona

W tym wariancie obliczanie przybliżenia całki polega na obliczeniu wartości funkcji dla punktów pośrednich (punkty leżące dokładnie pomiędzy dwoma sąsiadującymi punktami), które można wyliczyć w następujący sposób:

$$t_i = \frac{x_{i-1} + x_i}{2} \text{ dla } i = 1, 2, \dots, n \quad (4)$$

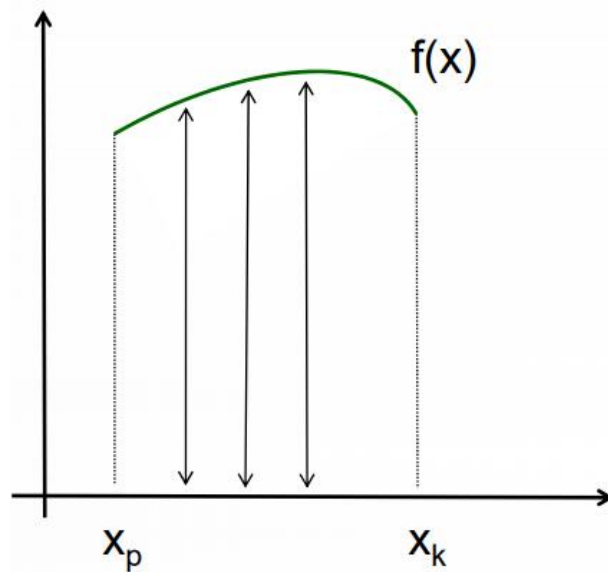
Lub:

$$t_i = x_p + i \cdot dx - \frac{dx}{2}, \text{ gdzie } i = 1, 2, \dots, n \quad (5)$$

Wzór możemy zapisać jako iloczyn szerokości podprzedziału podzielonej przez 6 i sumy wartości punktów granicznych całki, czterokrotną wartość sumy punktów pośrednich oraz dwukrotną wartość sumy punktów równo oddalonych o dx.

$$\int_{x_p}^{x_k} f(x) dx \approx \frac{dx}{6} \left(f(x_p) + 4 \cdot \sum_{i=0}^{n-1} f\left(x_p + i \cdot dx + \frac{dx}{2}\right) + 2 \cdot \sum_{i=1}^{n-1} f(x_p + i \cdot dx) + f(x_k) \right) \quad (6)$$

Metoda Monte Carlo – specyficzna metoda mająca zastosowania głównie w modelowaniu matematycznym tak bardzo złożonych procesów, że wykluczone jest podejście analityczne. Polega na losowaniu określonej liczby punktów z zakresu całkowania, a następnie obliczeniu średniej z wartości funkcji we wcześniej wylosowanych punktach.



Rysunek 3. Metoda Monte Carlo

Obliczenie średniej wygląda następująco:

$$f_{\text{średnie}} = \frac{f(x_1)}{n} + \frac{f(x_2)}{n} + \dots + \frac{f(x_n)}{n} \quad (7)$$

Wartość całki natomiast liczy się ze wzoru:

$$\int_{x_p}^{x_k} f(x) dx \approx I = f_{\text{średnie}} \cdot |x_k - x_p| \quad (8)$$

Gdzie:

x_p – dolna granica całkowania

x_k – górna granica całkowania

n – liczba podprzedziałów

x_1, x_2, \dots, x_n – wylosowane punkty z przedziału.

3. Kod programu

Na samym początku zdefiniowano funkcję pomocniczą służącą do obliczania wartości danej funkcji w przekazanym punkcie. Jej głównym zadaniem było zwiększenie przejrzystości kodu.

```
double funkcja(double x) {  
    double fx = x * x * x + 2;  
    return fx;  
}
```

Rysunek 4. Definicja funkcji pomocniczej

Kolejnym krokiem była definicja funkcji obliczającej całkę metodą Simpsona. Przyjmuje ona trzy parametry: dolną i górną granice oraz liczbę podprzedziałów. Wylicza szerokość i kolejne x_i oraz t_i przy pomocy wzorów (1) i (5), po czym korzystając ze wzoru (6) oblicza wynik.

```
double calkowanieMetodaSimpsona(double dolna_granica, double gorna_granica, int licznosc) {  
    double dx = (gorna_granica - dolna_granica) / licznosc; //Obliczenie wysokosci trapezu  
    double wynik = 0;  
    double x = dolna_granica;  
    double sumaH = 0;  
    double sumaH2 = 0;  
  
    for (size_t i = 1; i <= licznosc; i++)  
    {  
        x += dx; //Obliczanie kolejnych punktów  
        sumaH2 += funkcja(x - dx / 2); //Wyliczenie wartosci dla f(xp + i*dx - dx/2)  
        if (i < licznosc) { //if zapobiega dodaniu wartosci granicznych  
            sumaH += funkcja(x); //Wyliczenie wartosci dla f(xp + i*dx)  
        }  
    }  
  
    wynik = (dx / 6) * (funkcja(dolna_granica) + 2 * sumaH + 4 * sumaH2 + funkcja(gorna_granica));  
    return wynik;  
}
```

Rysunek 5. Funkcja metody Simpsona

Następną zdefiniowaną funkcją jest ta odpowiedzialna za metodę Monte Carlo. Przyjmuje dokładnie te same parametry. Punkty dobierane są w tym przypadku losowo dzięki `uniform_real_distribution < double > dist`, w ten sposób tworzony jest obiekt, do którego przekazujemy zakres w którym będą losowane liczby: `dist(dolna_granica, gorna_granica)`, następnie wywołujemy losowanie `dist(rnd)`. Dzięki `mt19937 rnd(std::time(NULL))` za każdym razem otrzymujemy inne liczby. Obliczamy średnią wzorem (7) oraz ostateczny wynik przy pomocy (8).

```
double calkowanieMonteCarlo(double dolna_granica, double gorna_granica, int licznosc) {
    double wynik = 0;
    double sredniaF = 0;
    double losowyX;
    cout << "Wylosowane punkty: " << endl;
    for (size_t i = 1; i <= licznosc; i++)
    {
        //utworzenie i przekazanie odpowiedniego przedziału do obiektu dist
        uniform_real_distribution < double > dist(dolna_granica, gorna_granica);
        losowyX = dist(rnd); //zapisanie wylosowanego punktu
        cout << losowyX << endl;
        sredniaF += funkcja(losowyX)/ licznosc; //Wyliczenie średniej
    }
    wynik = sredniaF*fabs(gorna_granica- dolna_granica); //Obliczenie wartosci całki
    return wynik;
}
```

Rysunek 6. Funkcja metody Monte Carlo

Na koniec został funkcja main w której zostały zadeklarowane zmienne, pobrano dane od użytkownika, wywołano odpowiednie funkcje oraz wyświetlono wyniki.

```
int main() {
    double xp, xk;
    int n;

    cout << "Podaj dolna granice ";
    cin >> xp;
    cout << endl;
    cout << "Podaj gorna granice ";
    cin >> xk;
    cout << endl;
    cout << "Podaj ilosc czesci ";
    cin >> n;
    cout << endl;

    cout << "Wynik dla metody prostokatow " << calkowanieMetodaProstokatow(xp, xk, n) << endl;
    cout << "Wynik dla metody trapezow " << calkowanieMetodaTrapezow(xp, xk, n) << endl;
    cout << "Wynik dla metody Simpsona " << calkowanieMetodaSimpsona(xp, xk, n) << endl;
    cout << "Wynik dla metody Monte Carlo " << calkowanieMonteCarlo(xp, xk, n) << endl;

    return 0;
}
```

Rysunek 7. Funkcja main

Cały kod:

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <random>
using namespace std;
mt19937 rnd(std::time(NULL));

double funkcja(double x) {
    double fx = x * x * x + 2;
    return fx;
}

double calkowanieMetodaProstokatow(double dolna_granica, double gorna_granica, int licznosc) { ... }
double calkowanieMetodaTrapezow(double dolna_granica, double gorna_granica, int licznosc) { ... }
double calkowanieMetodaSimpsona(double dolna_granica, double gorna_granica, int licznosc) {

    double dx = (gorna_granica - dolna_granica) / licznosc; //Obliczenie wysokosci trapezu
    double wynik = 0;
    double x = dolna_granica;
    double sumaH = 0;
    double sumaH2 = 0;

    for (size_t i = 1; i <= licznosc; i++)
    {
        x += dx;
        sumaH2 += funkcja(x - dx / 2); //Obliczanie kolejnych punktów
        //Wylicznienie wartosci dla f(xp + i*dx - dx/2)
        if (i < licznosc) {
            sumaH += funkcja(x); //if zapobiega dodaniu wartosci granicznych
            //Wylicznienie wartosci dla f(xp + i*dx)
        }
    }

    wynik = (dx / 6) * (funkcja(dolna_granica) + 2 * sumaH + 4 * sumaH2 + funkcja(gorna_granica));
    return wynik;
}

double calkowanieMonteCarlo(double dolna_granica, double gorna_granica, int licznosc) {

    double wynik = 0;
    double sredniaF = 0;
    double losowyX;
    cout << "Wylosowane punkty: " << endl;
    for (size_t i = 1; i <= licznosc; i++)
    {
        //utworzenie i przekazanie odpowiedniego przedzialu do obiektu dist
        uniform_real_distribution < double > dist(dolna_granica, gorna_granica);
        losowyX = dist(rnd); //zapisanie wylosowanego punktu
        cout << losowyX << endl;
        sredniaF += funkcja(losowyX) / licznosc; //Wylicznienie sredniej
    }

    wynik = sredniaF * fabs(gorna_granica - dolna_granica); //Oblicznienie wartosci calki
    return wynik;
}
```

Rysunek 8. Cały kod cz.1

```

int main() {
    double xp, xk;
    int n;

    cout << "Podaj dolna granice ";
    cin >> xp;
    cout << endl;
    cout << "Podaj gorna granice ";
    cin >> xk;
    cout << endl;
    cout << "Podaj ilosc czesci ";
    cin >> n;
    cout << endl;
    cout << "Wynik dla metody prostokatow " << calkowanieMetodaProstokatow(xp, xk, n) << endl;
    cout << "Wynik dla metody trapezow " << calkowanieMetodaTrapezow(xp, xk, n) << endl;
    cout << "Wynik dla metody Simpsona " << calkowanieMetodaSimpsona(xp, xk, n) << endl;
    cout << "Wynik dla metody Monte Carlo " << calkowanieMonteCarlo(xp, xk, n) << endl;
    return 0;
}

```

Rysunek 9. Cały kod cz.2

4. Testy

W celu sprawdzenia poprawności działania programu zostały wykonane trzy testy przy pomocy programu WolframAlpha. Każdej funkcji została policzona obydwiema metodami oraz kalkulatorem z wyżej wymienionego programu. Na koniec by sprawdzić która z metod jest dokładniejsza, policzono i porównano błędy względne przy użyciu wzoru:

$$\delta = \left| \frac{x - x_0}{x} \right| * 100\%$$

Gdzie:

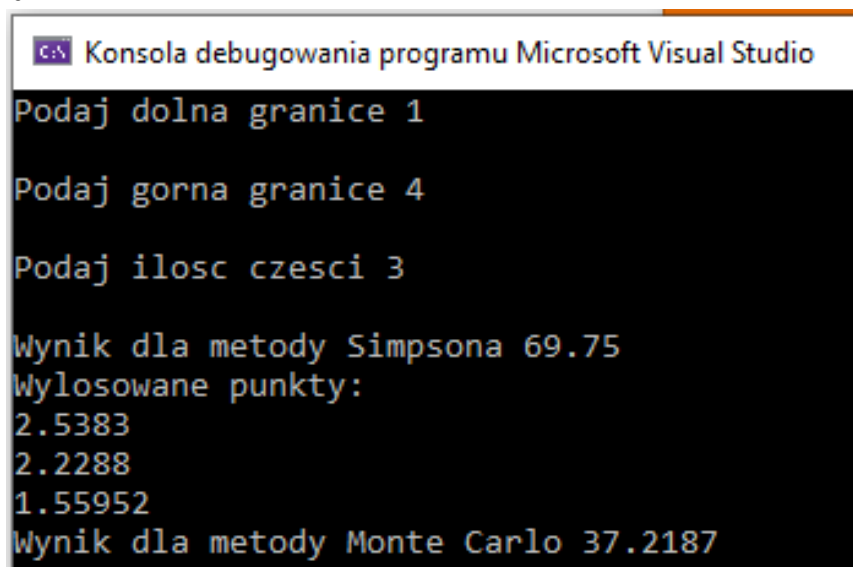
x – wartość dokładna – wynik z Wolframa

x_0 – zmierzona wartość – wyliczona przez program

Test 1:

| Funkcja | Dolna granica | Górna granica | Liczba fragmentów |
|-----------|---------------|---------------|-------------------|
| $x^3 + 2$ | 1 | 4 | 3 |

Wyniki programu:



```
Konsola debugowania programu Microsoft Visual Studio

Podaj dolna granice 1
Podaj gorna granice 4
Podaj ilosc czesci 3

Wynik dla metody Simpsona 69.75
Wylosowane punkty:
2.5383
2.2288
1.55952
Wynik dla metody Monte Carlo 37.2187
```

Rysunek 10. Wyniki całkowania $f(x) = x^3 + 2$ dla $n = 3$

Wynik WolframAlpha:

$$\int_1^4 (x^3 + 2) dx = \frac{279}{4} = 69.75$$

Błędy względne:

- Dla metody Simpsona: $\delta = \left| \frac{69.75 - 69.75}{69.75} \right| * 100\% = 0\%$
- Dla metody Monte Carlo: $\delta = \left| \frac{69.75 - 37.2187}{69.75} \right| * 100\% = 46.63986\%$

Test 2:

| Funkcja | Dolna granica | Górna granica | Liczba fragmentów |
|------------------|---------------|---------------|-------------------|
| $-4x^3 + 7x + 1$ | 4 | 16 | 20 |

Wyniki programu:

```
Konsola debugowania programu Microsoft Visual Studio
Podaj gorna granice 16
Podaj ilosc czesci 20
Wynik dla metody Simpsona -64428
Wylosowane punkty:
8.01968
14.7474
8.94288
14.0138
10.2554
7.21454
7.66615
6.78295
10.4334
10.4498
8.32745
11.6449
6.97527
6.18468
5.29591
7.38872
8.44623
7.48001
9.40907
10.3883
Wynik dla metody Monte Carlo -42296.3
```

Rysunek 11. Wyniki całkowania $f(x) = -4x^3 + 7x + 1$ dla $n = 20$

Wynik WolframAlpha:

$$\int_4^{16} (-4x^3 + 7x + 1) dx = -64428$$

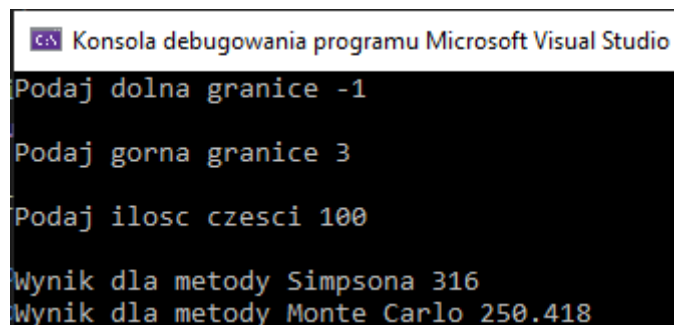
Błędy względne:

- Dla metody Simpsona: $\delta = \left| \frac{-64428 + 64428}{-64428} \right| * 100\% = 0\%$
- Dla metody Monte Carlo: $\delta = \left| \frac{-64428 + 42296.3}{-64428} \right| * 100\% = 34.3511\%$

Test 3:

| Funkcja | Dolna granica | Górna granica | Liczba fragmentów |
|--------------------|---------------|---------------|-------------------|
| $3x^5 - 5x^3 + 13$ | -1 | 3 | 100 |

Wyniki programu:



```
Konsola debugowania programu Microsoft Visual Studio
Podaj dolna granice -1
Podaj gorna granice 3
Podaj ilosc czesci 100
Wynik dla metody Simpsona 316
Wynik dla metody Monte Carlo 250.418
```

Rysunek 12. Wyniki całkowania $f(x) = 3x^5 - 5x^3 + 13$ dla $n = 100$

Wynik WolframAlpha:

$$\int_{-1}^3 (3x^5 - 5x^3 + 13) dx = 316$$

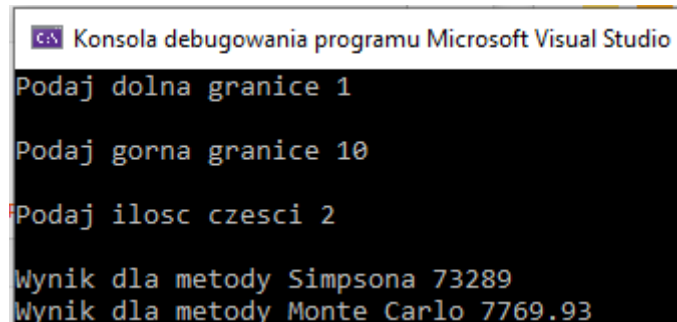
Błędy względne:

- Dla metody Simpsona: $\delta = \left| \frac{316 - 316}{316} \right| * 100\% = 0\%$
- Dla metody Monte Carlo: $\delta = \left| \frac{316 - 250.418}{316} \right| * 100\% = 20.7538\%$

Test 4:

| Funkcja | Dolna granica | Górna granica | Liczba fragmentów |
|----------------------|---------------|---------------|-------------------|
| $4x^4 - 3x^3 + 2x^2$ | 1 | 10 | 2 |

Wyniki programu:



```
Konsola debugowania programu Microsoft Visual Studio
Podaj dolna granice 1
Podaj gorna granice 10
Podaj ilosc czesci 2
Wynik dla metody Simpsona 73289
Wynik dla metody Monte Carlo 7769.93
```

Rysunek 13. Wyniki całkowania $f(x) = 4x^4 - 3x^3 + 2x^2$ dla $n = 2$

Wynik WolframAlpha:

$$\int_1^{10} (4x^4 - 3x^3 + 2x^2) dx = \frac{1463319}{20} \approx 73166.$$

Błędy względne:

- Dla metody Simpsona: $\delta = \left| \frac{73166 - 73289}{73166} \right| * 100\% = 0.16811\%$
- Dla metody Monte Carlo: $\delta = \left| \frac{73166 - 7769.93}{73166} \right| * 100\% = 89.38041\%$

5. Wnioski

Wykonane testy potwierdziły, że metoda Simpsona jest niezwykle dokładna nawet dla niewielkiej ilości punktów, błąd względny tylko raz był różny od zera i to przy podziale dla $n = 2$. Świadczy to o tym, że metoda ta jest bardzo wydajna, gdyż przy małym nakładzie pracy programu (mały podział) jest w stanie podać dokładne wyniki. Metoda Monte Carlo, która opiera się o liczby losowe, jak wynika z powyższych testów nie daje zadowalających rezultatów. Właściwy wynik jest niemal niemożliwy do uzyskania tym sposobem. Metoda Simpsona jest najprecyzyjniejszą i najwydajniejszą metodą jaką poznaliśmy jak dotąd na zajęciach.