*Zadanie 14*

**Opis:**

Wykres wygenerowany za pomocą programu GNUplot. Program wykonany w języku C/C++.

**Kompilacja/uruchomienie programu:**

Aby skompilować program, którego kod znajduje się na końcu tego dokumentu można skopiować go do wybranego IDE np.: Visual Studio, Code Block itd. Bądź uruchomić z poziomu konsoli wybranymi komendami, tak jak zwykły program w C++. Program po takim uruchomieniu wypisze w słupku rozwiązania układu równań.

**Metoda rozwiązania/ dyskusja:**

Zadanie 14 polegalo na obliczeniu takiej całki: $\int_{-1}^{1} \frac{\exp(x^2)\,dx}{\sqrt{1-x^2}}$ trzema metodami: złożonej metody trapezów, Simpsona i reguły 3/8, stosując iteracyjne zagęszczanie podprzedziałów. Policzenie takiej całki będzie możliwe po zastosowaniu zamiany zmiennych. Tutaj dobrym pomysłem będzie zastosowanie zamiany na jakąś z funkcji trygonometrycznych ze względu na jej jasne przedziały.

Przyjąłem funkcje cosinus. X=cos(t) -> x є (-1;1), t od 0 do pi. Po podstawieniu dx=-sin(t)dt. Po przekształceniach postać naszej całki to już: $e^{cos^2 t}$.

Nie liczę za każdy razem wartości funkcji, korzystam z policzonych już wcześniej.

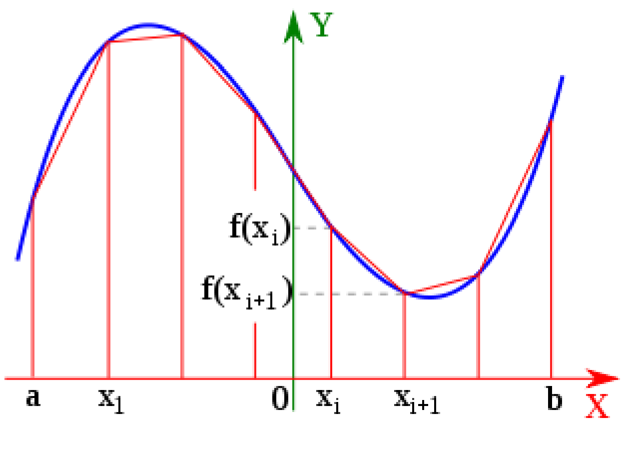
**Metoda Trapezów**

Metoda trapezów polega na aproksymacji danej krzywej linią łamaną w nią wpisaną. Przedział całkowania (a,b) dzielimy na n równych części o długościach h = $\frac{b-a}{2}$ .

$$\int_a^b f(x)\,dx \simeq \frac{b-a}{2}(f_0 + f_1)$$

Błąd metody trapezów: $h^3$ $E = -\frac{1}{12}(b-a)^3 f''(\zeta_0)\,,$

Przykladowy wykres:



**Metoda Simpsona:**

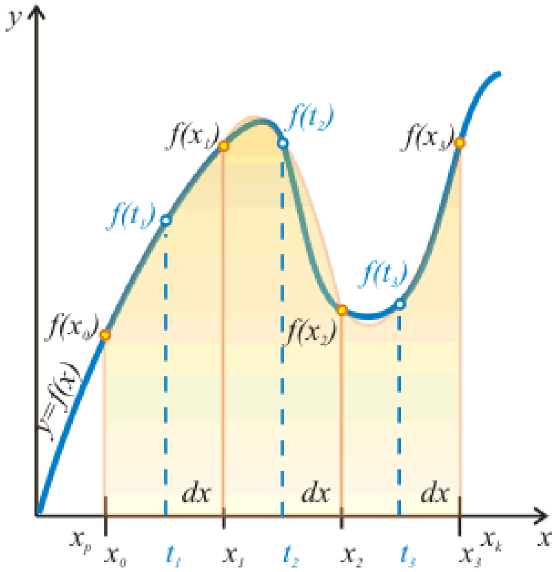
Wymaga podzielenia przedziału całkowania na parzystą liczbę podprzedziałów,

tzn. h=$\frac{b-a}{2n}$

$$\int_{x_i}^{x_{i+2}} f(x)dx \approx \frac{h}{3}[f_i + 4f_{i+1} + f_{i+2}]$$

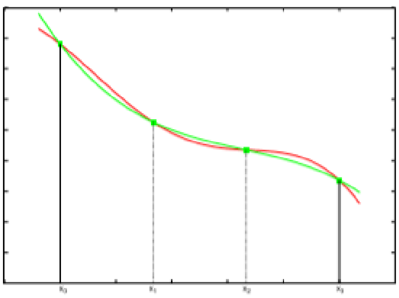
Dla całego przedziału (a,b):

$$\int_a^b f(x)dx \approx \frac{h}{3}[f_0 + 4(f_1 + f_3 + \ldots + f_{2n-1}) + 2(f_2 + f_4 + \ldots + f_{2n-2}) + f_{2n}]$$




**Metoda 3/8**:

Metoda 3/8



$$\int_a^b f(x)\,dx \simeq \frac{b-a}{8}(f_0 + 3f_1 + 3f_2 + f_3)$$

$$E = -\frac{3}{80}\left(\frac{b-a}{3}\right)^5 f^{(4)}(\zeta)$$

Błąd Metody 3/8

**Wyniki działania programów:**

**Wynik działania programów:**

**Każdy program dał taki sam wynik w ostatniej iteracji, co daję nadzieje ze jest dobrze 😋**

```
1 5.8406634381
2 5.5101370350
3 5.5084297778
4 5.5084297739
5 5.5084297739
```

**Coraz dokładniejsza wartość z każdą iteracją**

**Kod programów:**

METODA TRAPEZÓW:

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>

#define _USE_MATH_DEFINES
#include <cmath>

const double e = 0.0000000001;

double funkcja(double x) {
	double cCos = 0.0;
	cCos = cos(x);
	return (exp(cCos * cCos));
}

double f(const double S1, double S2, double h) {
	return (h * (0.5 * S1 + S2));
}
int main() {
	double ostatni_I = 0.0;
	double N = 1.0;
	double b = M_PI;
	double a = 0.0;
	double h = b - a;
	double res;
	int i = 0;
	const double S1 = funkcja(a) + funkcja(b);
	double S2 = 0.0;
	res = f(S1, S2, h);
	ostatni_I = res;
	while (1) {
		i++;
		h /= 2;
		for (int j = 0; j < N; j++) {
			S2 += funkcja(a + (2 * j + 1) * h);
		}
		res = f(S1, S2, h);
		printf("%d %.10f \n", i, res);
		N *= 2;
		if (fabs(ost_I - res) < e) break;
```

```
		ostatni_I = res;
	}
	return 0;}
```

**METODA 3/8:**

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define _USE_MATH_DEFINES

#include <math.h>
#include <cmath>

const double e = 0.0000000001;
double funkcja(double x) {
	double cCos = 0.0;
	cCos = cos(x);
	return (exp(cCos * cCos));
}
double f(const double S1, double S2, double S3, double h) {
	return (3 * h * (S1 + 3 * S2 + 2 * S3) / 8);
}
int main() {
	double ostatni_I = 0.0;
	double N = 1.0;
	double b = M_PI;
	double a = 0.0;
	double h = (b - a) / 3;
	double res = 0;
	int i = 0;
	const double S1 = funkcja(a) + funkcja(b);
	double S2 = 0.0, S3 = 0.0;
	ostatni_I = res;
	while (1) {
		i++;
		S2 = 0.0;
		for (int j = 0; j < N; j++) {
			S2 += (funkcja(a + (3 * j + 1) * h) + funkcja(a + (3 * j + 2) *
h));
		}
		res = f(S1, S2, S3, h);
		printf("%d %.10f \n", i, res);
		if (fabs(ostatni_I - res) < e) break;
		ostatni_I = res;
		N *= 3;
		S3 += S2;
		h /= 3;
	}
	system("pause");
	return 0;
}
```

**METODA SIMPSONA:**

```
//Metoda Simpsona

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define _USE_MATH_DEFINES
#include <math.h>


const double e = 0.0000000001;

double funkcja(double x) {
	double cCos = 0.0;
	cCos = cos(x);
	return (exp(cCos * cCos));
}

double f(const double S1, double S2,double S3, double h) {
	return (h * (S1 + 4 * S2 + 2 * S3)/3);
}
int main() {
	double ostatni_I = 0.0;
	double N = 1.0;
	double b = M_PI;
	double a = 0.0;
	double h = b - a;
	double res;
	int i = 0;
	const double S1 = funkcja(a) + funkcja(b);
	double S2 = 0.0;
	double S3 = 0.0;
	res = f(S1, S2,S3, h);
	ostatni_I = res;
	while (1) {
		i++;
		h /= 2;
		S2 = 0.0;
		for (int j = 0; j < N; j++) {
			S2 += funkcja(a + (2 * j + 1) * h);
		}
		res = f(S1, S2, S3, h);
		printf("%d %.10f \n", i, res);
		N *= 2;
		S3 += S2;
		if ((fabs(ostatni_I - res) < e)) break;
		ostatni_I = res;
	}
	return 0;
}
```