

Projekt 1 – Całkowanie

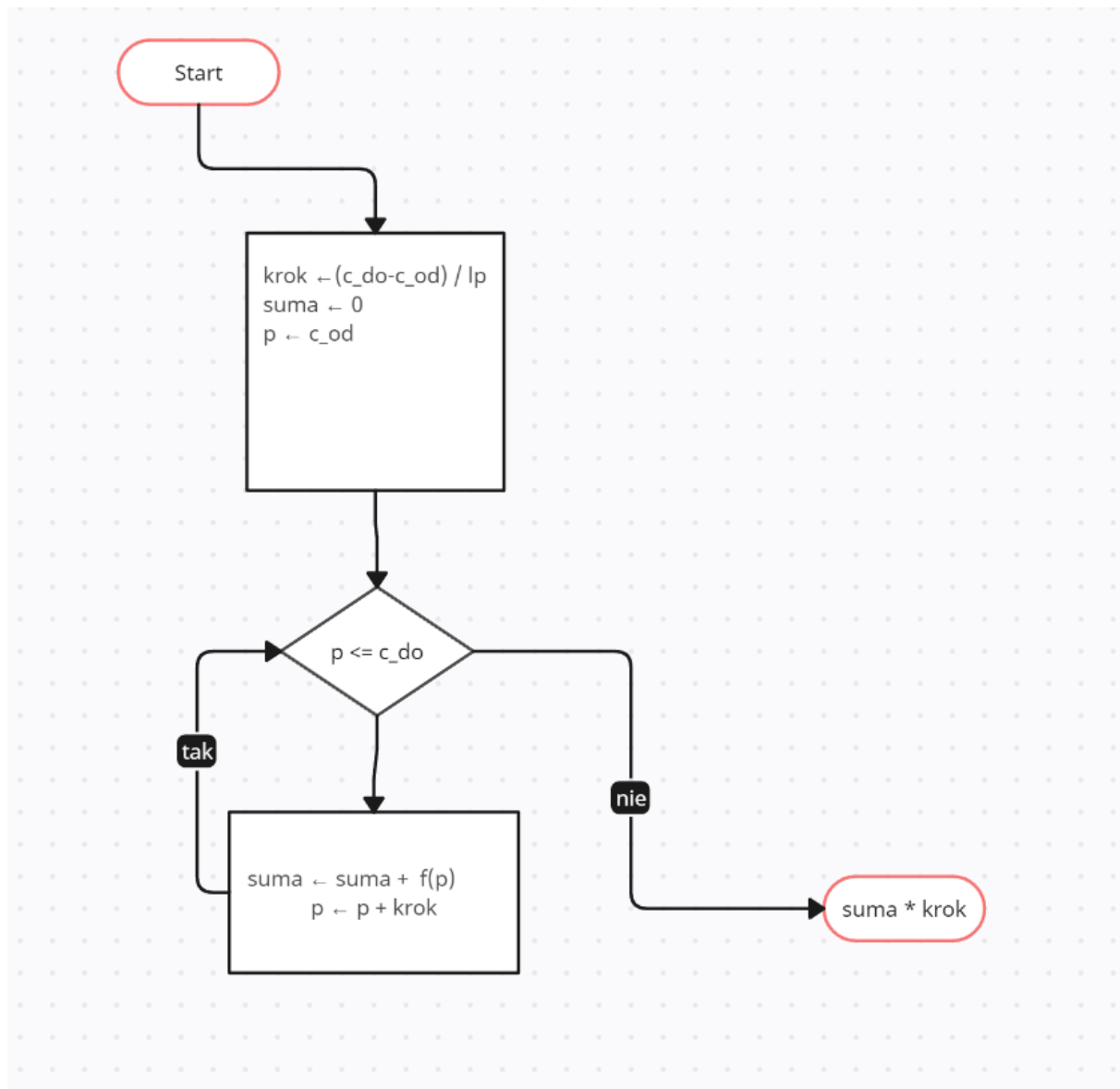
1. Opis zadania

Zadanie polega na zaimplementowaniu trzech algorytmów całkujących w pliku `calki.c`, który wraz z plikiem `calka.c` wchodzi w skład projektu. `Calka.c` przyjmuje dane od użytkownika, a następnie odwołując się do włączonego poprzez dyrektywę `#include „calki.c”` pliku wywołuje zaimplementowane w ramach zadania funkcje.

a) Metoda prostokątów

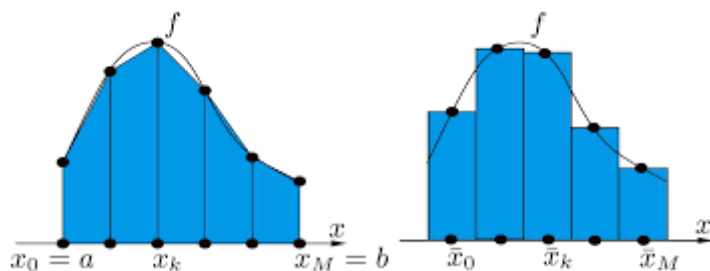
Polega ona na podzieleniu przedziału całkowania na określoną liczbę podprzedziałów (n). Wzrost liczby podprzedziałów implikuje wzrost czasu wykonania. Musimy więc znaleźć złoty środek między wydajnością a dokładnością. Robimy to ręcznie. Następnie dla każdego podprzedziału obliczamy wartość funkcji na jego początku. Po czym obliczamy pole prostokąta o podstawie równej długości podziału i wysokości równej obliczonej wartości.

Wykonujemy takie kroki dla każdego podprzedziału i otrzymujemy n różnych pól prostokątów których suma w przybliżeniu równa się polu pod wykresem.



b) Metoda trapezów

W swoim działaniu przypomina metodę prostokątów. Tutaj również mamy do czynienia z podziałem przedziału całkowania na n podprzedziałów. Liczymy natomiast pola trapezów (takich jak ilustruje zdjęcie po lewej stronie).



Metoda ta zapewnia lepsze przybliżenie wartości całki, ponieważ trapezy lepiej pokrywają obszar ograniczony przez funkcję. Można również wykonać pewną optymalizację:

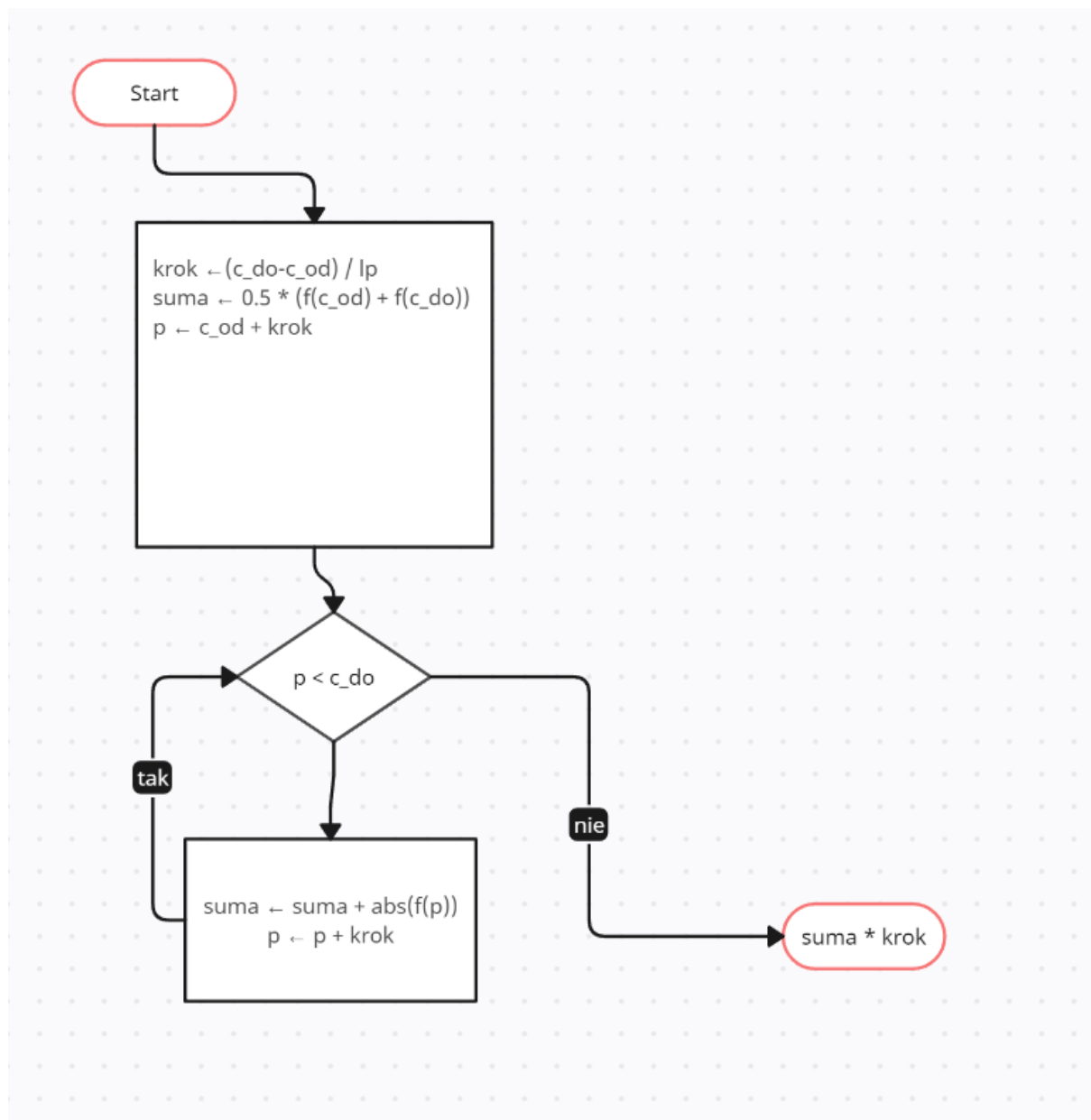
$$s = \frac{f_0 + f_1}{2} \cdot dx + \frac{f_1 + f_2}{2} \cdot dx + \frac{f_2 + f_3}{2} \cdot dx + \dots + \frac{f_{n-2} + f_{n-1}}{2} \cdot dx + \frac{f_{n-1} + f_n}{2} \cdot dx$$

$$s = \frac{dx}{2} \cdot (f_0 + f_1 + f_1 + f_2 + f_2 + f_3 + \dots + f_{n-2} + f_{n-1} + f_{n-1} + f_n)$$

$$s = \frac{dx}{2} \cdot (f_0 + 2f_1 + 2f_2 + \dots + 2f_{n-1} + f_n)$$

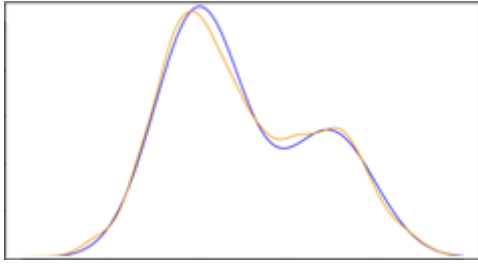
$$s = dx \cdot \left(f_1 + f_2 + \dots + f_{n-1} + \frac{f_0 + f_n}{2} \right)$$

S to wartość całki. Dzięki takiemu przekształceniu, na samym początku programu możemy zainicjować naszą sumę jako średnia arytmetyczna pól pierwszego i ostatniego trapezu, a następnie tylko dodawać odpowiednie pola w ramach pętli. Unikamy mnożenia przez długość przedziału (będącą jednocześnie wysokością trapezu) i wykonujemy je na całej sumie przy końcu funkcji.

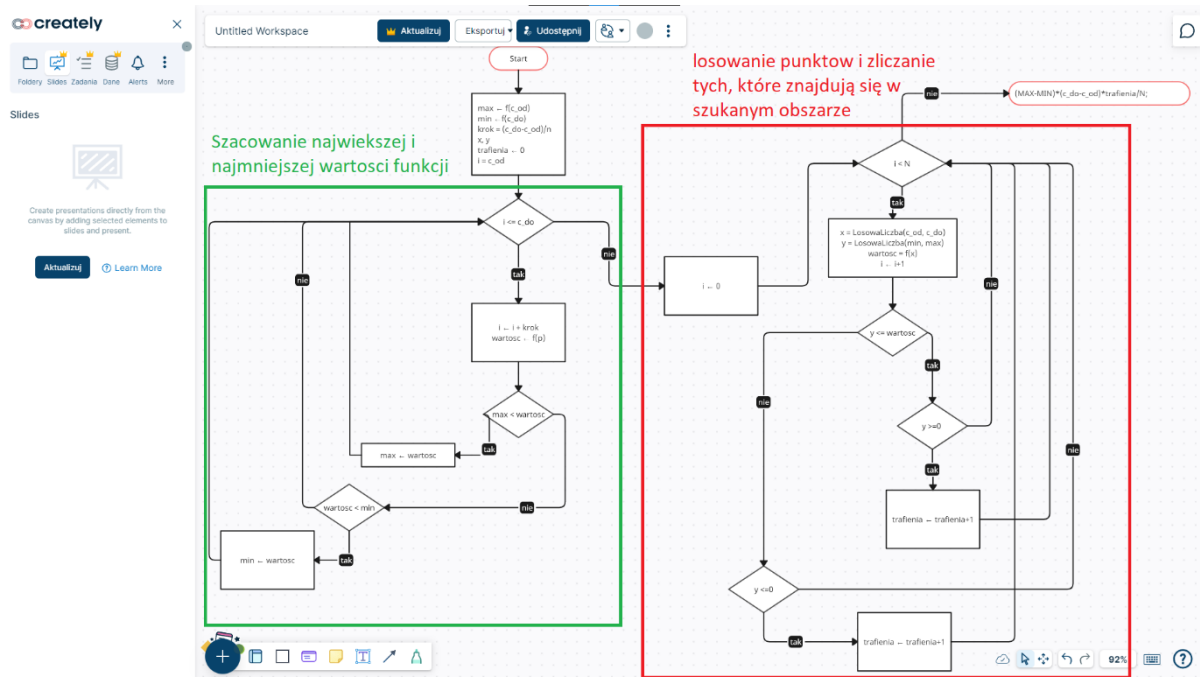


c) Metoda Monte Carlo

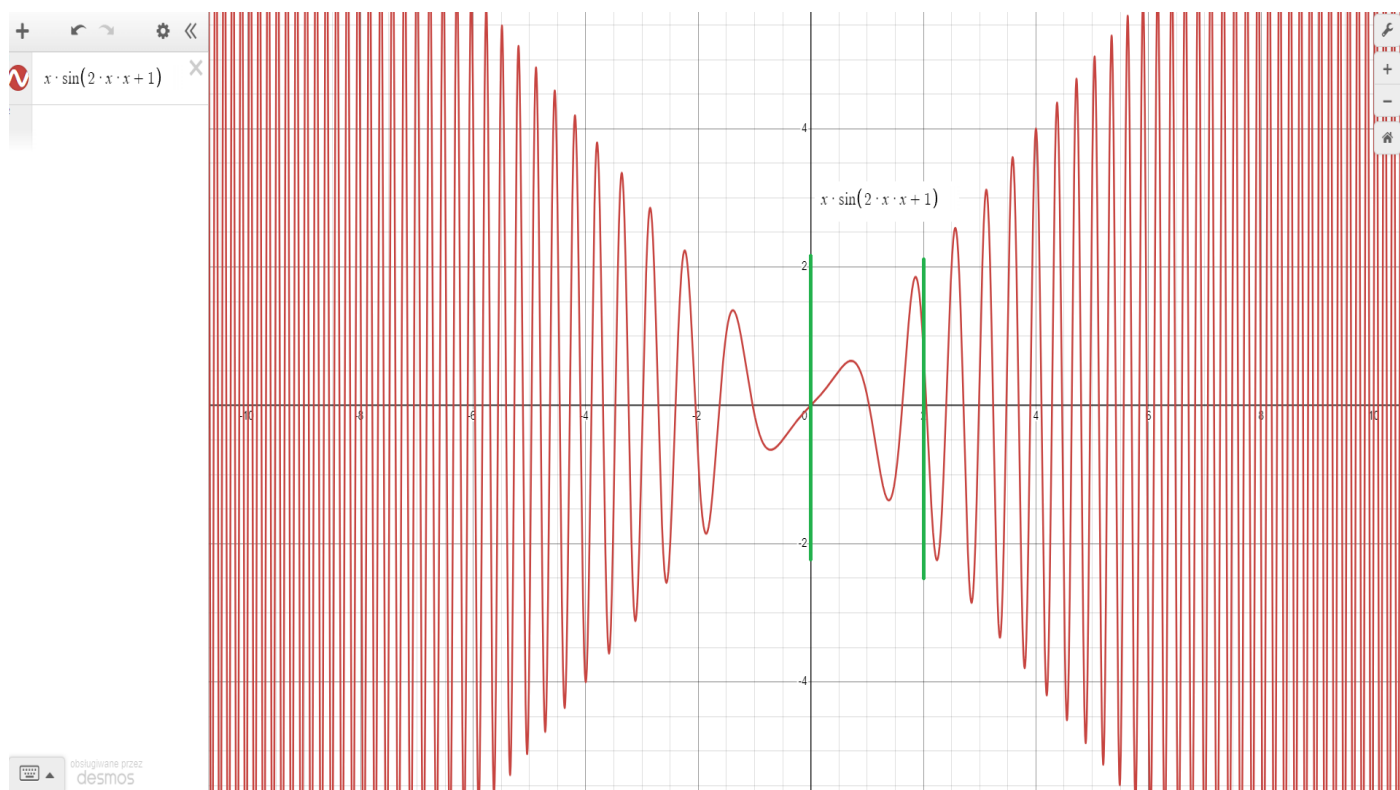
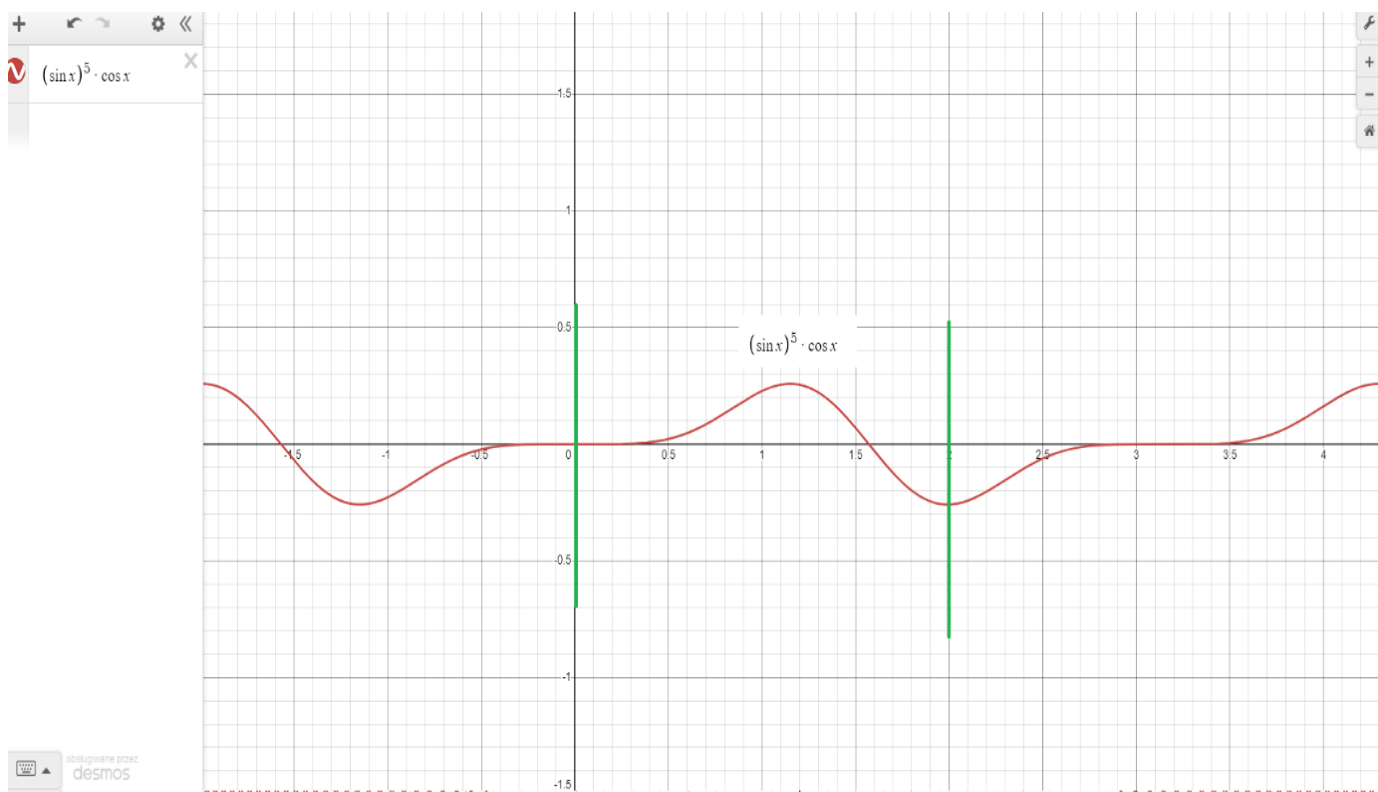
Polega ona na oszacowaniu całki za pomocą probabilistyki. W pierwszym kroku szacujemy największą i najmniejszą wartość funkcji w danym przedziale całkowania. Następnie w oparciu o te dwa punkty i punkty graniczne przedziału całkowania tworzymy prostokąt:



Następnie losujemy N punktów znajdujących się w obrębie tego prostokąta. Punkt leżący w szukanym obszarze zwiększa licznik trafień. Z uwagi na to, że całka z funkcji ujemnej jest ujemna, punkty znajdujące się w obszarze ograniczonym przez funkcję, ale poniżej osi ox (wtedy gdy jest ona ujemna) dekrementują wspomniany licznik. Ostatecznie, aby obliczyć całkę mnożymy wartość pola prostokąta przez iloczyn $\text{trafienia}/N$.



2. Wykresy badanych funkcji z zaznaczonymi przedziałami całkowania



3. Rozwiązanie analityczne z obliczeniami


Do obliczenia całek wykorzystałem program WolframAlpha.

Przedział całkowania: $[0, 2]$.

Funkcja f1:

Definite integral

$$\int_0^2 (-3.13 x x x x + 14.5 x x - 3) dx = 12.6347$$

 *Wolfram|Alpha Step-by-step solution*

Definite integral:

STEP 1


Compute the definite integral:







$$\int_0^2 \left(-\frac{313 x^4}{100} + \frac{29 x^2}{2} - 3 \right) dx$$

Integrate the sum term by term and factor out constants:

$$= -\frac{313}{100} \int_0^2 x^4 dx + \frac{29}{2} \int_0^2 x^2 dx - 3 \times \int_0^2 1 dx$$

Funkcja f2:



 NATURAL LANGUAGE  MATH INPUT  EXTENDED KEYBOARD  EXAMPLES  UPLOAD  RANDOM

Definite integral More digits

$$\int_0^2 (\cos(2 x x) 2 \sin(8 x) - \cos(5 x) + 1) dx = \frac{1}{2} \sqrt{\pi} \left(2 C\left(\frac{4}{\sqrt{\pi}}\right) - C\left(\frac{8}{\sqrt{\pi}}\right) \right) \sin(8) + \left(S\left(\frac{8}{\sqrt{\pi}}\right) - 2 S\left(\frac{4}{\sqrt{\pi}}\right) \right) \cos(8) + 2 - \frac{\sin(10)}{5} \approx 2.83344$$

$S(x)$ is the Fresnel S integral
 $C(x)$ is the Fresnel C integral

Funkcja f3:

FROM THE MAKERS OF WOLFRAM LANGUAGE AND MATHEMATICA



int [//math:pow(sin(x), 5)*cos(x);//] from [//math:0//] to [//math:2//]

NATURAL LANGUAGE

MATH INPUT

EXTENDED KEYBOARD EXAMPLES UPLOAD RANDOM

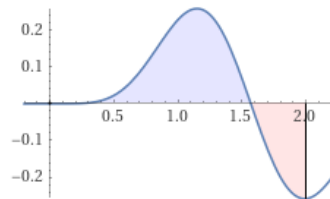
Definite integral

More digits

Step-by-step solution

$$\int_0^2 \sin^5(x) \cos(x) dx = \frac{\sin^6(2)}{6} \approx 0.094207$$

Visual representation of the integral



Funkcja f4:

FROM THE MAKERS OF WOLFRAM LANGUAGE AND MATHEMATICA



int [//math:x*sin(2*x*x+1)//] from [//math:0//] to [//math:2//]

NATURAL LANGUAGE

MATH INPUT

EXTENDED KEYBOARD EXAMPLES UPLOAD RANDOM

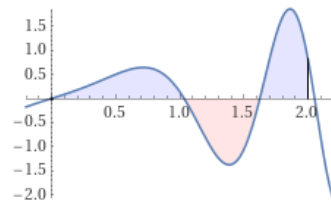
Definite integral

More digits

Step-by-step solution

$$\int_0^2 x \sin(2x^2 + 1) dx = \frac{1}{4} (\cos(1) - \cos(9)) \approx 0.36286$$

Visual representation of the integral



Indefinite integral

Step-by-step solution

$$\int x \sin(2x^2 + 1) dx = -\frac{1}{4} \cos(2x^2 + 1) + \text{constant}$$

4. Tabela z wynikami zaimplementowanych metod i wynikiem rozwiązania analitycznego

Przedział całkowania: [0, 2]	Prostokąty	Trapezy	MC	Rozwiązanie analityczne
f1: 3.13*x*x*x*x+14.5*x*x-3;	12.626733	12.634653	12.760860	12.6347
f2: cos(2*x*x)*2*sin(8*x)-cos(5*x)+1	2.831512	2.833435	2.836248	2.83344
f3: pow(sin(x), 5)*cos(x);	0.094466	0.094207	0.094929	0.094207
f4: x*sin(2*x*x+1)	0.362029	0.362853	0.363750	0.36286

5. Podsumowanie – wnioski

Spośród trzech badanych metod całkowania wyróżnia się metoda trapezów. Jest najdokładniejsza oraz zapewnia dobry czas działania. Metoda MC ustępuje dokładnością pozostałym algorytmom. Jest również mniej wydajna – osiągnięcie wyników przedstawionych na tabelce wymagało u mnie aż 40 000 iteracji pętli celem znalezienia min i max funkcji oraz 70 000 iteracji podczas których były losowane punkty. Dla porównania, funkcje prostokąty i trapezy dysponowały pętlami wykonującymi iteracje zaledwie 1000 razy i osiągnęły lepszą dokładność. Nie licząc czasu podania wartości c_od i c_do, program wykonał się w 0,036 sek, jednak przy zadaniach wymagających wielokrotnego całkowania metoda Monte Carlo nie jest dobrym wyborem.