Programowanie i metody numeryczne

Ćwiczenia 12.

Całkowanie Monte Carlo.

Zadanie 1. Całka oznaczona i liczba π.

Rozważmy zadanie obliczenia całki oznaczonej

$$I = \int_a^b f(x)\,\mathrm{d}x$$

całkowalnej funkcji $f : \mathbb{R} \supseteq X \to \mathbb{R}$ w przedziale $[a, b] \subset X$.

Napisz następujące funkcje, których zadaniem ma być obliczenie przybliżonej wartości tej całki – każda z funkcji powinna przyjmować jako argumenty implementację funkcji podcałkowej f, granice całkowania a i b oraz liczbę naturalną N i liczbę naturalną K o domyślnej wartości 100:

a) `mc_int` – metoda Monte Carlo wykorzystująca N losowo wybranych punktów z rozkładu jednostajnego,

b) `mc_int_stratified` – metoda Monte Carlo z losowaniem warstwowym: przedział całkowania ma być dzielony na N podprzedziałów, w każdym z nich ma być losowany jeden punkt,

c) `mc_int_importance_gauss` – metoda Monte Carlo z losowaniem istotnym opartym o rozkład normalny: N punktów ma być losowanych z rozkładu normalnego o średniej μ i odchyleniu standardowym σ; funkcja powinna przyjmować dodatkowe argumenty: μ i σ,

d) `qmc_int` – metoda Quasi Monte Carlo wykorzystująca N punktów z ciągu van der Corputa o podstawie b; funkcja powinna przyjmować dodatkowy argument: b.

Wszystkie funkcje powinny obliczać wartość całki I K–krotnie i zwracać średnią arytmetyczną wyników.

Korzystając z tej funkcji napisz program `pimc`, obliczający przybliżoną wartość liczby π na podstawie zależności

$$\pi = 2\int_{-1}^{1} \sqrt{1-x^2}\,\mathrm{d}x.$$

Program ma przyjmować jako argumenty wywołania liczbę naturalną M podzielną przez 100 oraz wartości parametrów K, b, μ i σ. Zadaniem programu jest obliczanie wartości π za pomocą każdej z napisanych przez Ciebie funkcji dla $N = 100, 200, \ldots, M$. Wynikiem działania programu powinno być osiem wykresów: dla każdej z funkcji program powinien wykreślać zależność obliczonej wartości π od N (rysując także linię odpowiadającą „dokładnej” wartości π dla porównania) oraz błąd, czyli moduł różnicy „dokładnej” i obliczonej wartości π, w funkcji N w skali logarytmicznej.

Jako generator liczb pseudolosowych wykorzystaj jeden z wysokiej jakości generatorów dostarczanych przez język programowania lub pakiet obliczeń naukowych, z którego korzystasz – na przykład *Mersenne Twister 19937*.

Poeksperymentuj z różnymi wartościami parametrów K, b, μ i σ. Która metoda obliczania daje najlepsze rezultaty? Jaką zależność przypomina błąd wartości całki w funkcji N dla każdej z metod?

Zadanie 2. Całki wielokrotne.

Rozważmy zadanie obliczenia całki wielokrotnej

$$I = \int\int\cdots\int_{\Omega} \left(1 - x_1^2 - x_2^2 - \ldots - x_m^2\right) \, \mathrm{d}x_1 \mathrm{d}x_2 \cdots \mathrm{d}x_m, \qquad \text{gdzie } \Omega = [0,\, 1]^m \,.$$

Napisz następujące funkcje, których zadaniem ma być obliczenie przybliżonej wartości tej całki – każda z funkcji powinna przyjmować jako argumenty liczbę naturalną N i liczbę naturalną K o domyślnej wartości 100:

a) `mc_int` – metoda Monte Carlo wykorzystująca N losowo wybranych punktów z rozkładu jednostajnego,

b) `mc_int_stratified` – metoda Monte Carlo z losowaniem warstwowym: obszar całkowania ma być dzielony na N podobszarów, w każdym z nich ma być losowany jeden punkt,

c) `mc_int_importance_gauss` – metoda Monte Carlo z losowaniem istotnym opartym o rozkład normalny: współrzędne N punktów mają być losowane z rozkładu normalnego o średniej μ i odchyleniu standardowym σ, takich samych dla każdej współrzędnej; funkcja powinna przyjmować dodatkowe argumenty: μ i σ,

d) `qmc_int_halton` – metoda Quasi Monte Carlo wykorzystująca N punktów z ciągu Halrona,

e) `qmc_int_sobol` – metoda Quasi Monte Carlo wykorzystująca N punktów z ciągu Sobol'a.

Wszystkie funkcje powinny obliczać wartość całki I K–krotnie i zwracać średnią arytmetyczną wyników.

Korzystając z tej funkcji napisz program `mcint`, obliczający przybliżoną wartość całki I. Program ma przyjmować jako argumenty wywołania liczbę naturalną M podzielną przez 100 oraz wartości parametrów K, μ i σ. Zadaniem programu jest obliczanie wartości I za pomocą każdej z napisanych przez Ciebie funkcji dla $N = 100,\, 200,\, \ldots,\, M$. Wynikiem działania programu powinno być dziesięć wykresów: dla każdej z funkcji program powinien wykreślać zależność obliczonej wartości I od N (rysując także linię odpowiadającą analitycznej wartości I dla porównania) oraz błąd, czyli moduł różnicy analitycznej i obliczonej wartości I, w funkcji N w skali logarytmicznej.

Jako generator liczb pseudolosowych wykorzystaj jeden z wysokiej jakości generatorów dostarczanych przez język programowania lub pakiet obliczeń naukowych, z którego korzystasz – na przykład *Mersenne Twister 19937*.

Poeksperymentuj z różnymi wartościami parametrów K, μ i σ. Która metoda obliczania daje najlepsze rezultaty? Jaką zależność przypomina błąd wartości całki w funkcji N dla każdej z metod?

Zadanie 3. Objętość n–wymiarowej kuli jednostkowej.

Rozważmy n–wymiarową, domkniętą kulę jednostkową

$$B_n = \left\{(x_1,\, x_2,\, \ldots,\, x_n) \in \mathbb{R}^n \,|\, x_1^2 + x_2^2 + \ldots + x_n^2 \leq 1\right\}.$$

Jej objętość możemy wyznaczyć numerycznie na podstawie wzoru

$$V_n = \int\int\cdots\int_{B_n} \mathrm{d}x_1 \mathrm{d}x_2 \cdots \mathrm{d}x_n = \int\int\cdots\int_{\Omega_n} \mathbb{1}_n\left(x_1,\, x_2,\, \ldots,\, x_n\right) \, \mathrm{d}x_1 \mathrm{d}x_2 \cdots \mathrm{d}x_n,$$

gdzie $\Omega_n = [0,\, 1]^n$ oraz

$$\mathbb{1}_n\left(x_1,\, x_2,\, \ldots,\, x_n\right) = \begin{cases} 1, & \text{gdy } (x_1,\, x_2,\, \ldots,\, x_n) \in B_n, \\ 0 & \text{w przeciwnym przypadku.} \end{cases}$$

Istnieje też analityczne wyrażenie na tę objętość:

$$V_n = \frac{\pi^{n/2}}{\Gamma\left(\frac{n}{2}+1\right)},$$

Γ jest tu funkcją Gamma Eulera.

a) Napisz funkcję `ball_exact`, przyjmującą jako argument liczbę naturalną n i zwracającą objętość V_n otrzymaną na podstawie wzoru analitycznego.

b) Napisz funkcję `ball_mc`, przyjmującą jako argumenty liczby naturalne n i N, która posługując się metodą Monte Carlo znajduje przybliżoną objętość V_n, wykorzystując N losowo wybranych punktów z rozkładu jednostajnego.

c) Korzystając z tych funkcji napisz program `ball`, który przyjmuje jako argumenty wywołania dwie liczby naturalne k i N. Program powinien obliczać objętości n–wymiarowych kul jednostkowych dla $n = 1, 2, \ldots, k$ dwukrotnie: metodą Monte Carlo, wykorzystując N losowo wybranych punktów, oraz analitycznie, a następnie narysować i wyświetlić na ekranie wykres zależności objętości kuli jednostkowej w n–wymiarowej przestrzeni euklidesowej od wymiaru przestrzeni n, zaznaczając zarówno wynik analityczny, jak i numeryczny.

   Czy jakość wyniku numerycznego zmienia się wraz z wymiarem przestrzeni, n?

d) Zapewne zauważyłeś, że dla dużych n metoda Monte Carlo radzi sobie coraz gorzej. Wynika to stąd, że stosunek objętości kuli B_n to opisanego na niej sześcianu Ω_n jest coraz mniejszy – kula zajmuje coraz mniejszą część tego sześcianu, a więc coraz więcej punktów wypada poza nią.

   Rozwiązaniem jest zastosowanie losowania istotnego. Napisz funkcję `ball_mc_importance`, działającą identycznie, jak funkcja `ball_mc`, losująca jednak współrzędne punktów z rozkładu normalnego o średniej 0 i odchyleniu standardowym 1/2.

   Napisz także program `ballim`, działający identycznie, jak program ball, posługujący się jednak funkcją `ball_mc_importance` zamiast funkcji `ball_mc`.

   Czy jakość przybliżenia się poprawiła?

Jako generator liczb pseudolosowych wykorzystaj jeden z wysokiej jakości generatorów dostarczanych przez język programowania lub pakiet obliczeń naukowych, z którego korzystasz – na przykład *Mersenne Twister 19937*.

Zadanie 4. Pole powierzchni ograniczonej lemniskatą Bernoulliego.

Napisz funkcję `lemniscate_area`, która posługując się metodą Monte Carlo znajduje przybliżoną wartość pola powierzchni ograniczonej lemniskatą Bernoulliego, opisaną równaniem

$$\left(x^2+y^2\right)^2 \leq 2\left(x^2-y^2\right),$$

wykorzystując N losowo wybranych punktów. Wybierz odpowiedni, prosty obszar całkowania, w którym zawarta jest lemniskata. Funkcja `lemniscate_area` przyjmuje argument N. Wartością zwracaną przez funkcję `LemniscateArea` powinna być obliczona przez nią przybliżona wartość szukanego pola powierzchni.

Korzystając z tej funkcji napisz program `lemniscate`, który przyjmuje jako argument wywołania liczbę naturalną określającą wartość N. Program powinien wypisywać na standardowe wyjście obliczoną przez funkcję `lemniscate_area` wartość pola powierzchni ograniczonej lemniskatą Bernoulliego.

Jako generator liczb pseudolosowych wykorzystaj jeden z wysokiej jakości generatorów dostarczanych przez język programowania lub pakiet obliczeń naukowych, z którego korzystasz – na przykład *Mersenne Twister 19937*.

Zadanie 5. Paradoks kawalera de Méré.

Zadanie to nie dotyczy całkowania – jego celem jest porównanie generatorów liczb pseudolosowych.

Paradoks kawalera de Méré to problem z dziedziny rachunku prawdopodobieństwa postawiony przez francuskiego matematyka–amatora Antoine'a Gombauda (1607 – 1684). Dotyczy on serii rzutów symetrycznymi sześciennymi kostkami do gry i polega na rozstrzygnięciu, które spośród następujących dwóch zdarzeń jest bardziej prawdopodobne:

- zdarzenie A: wyrzucenie przynajmniej jednej szóstki w czterech rzutach jedną kostką,
- zdarzenie B: wyrzucenie przynajmniej jednej pary szóstek w dwudziestu czterech rzutach dwiema kostkami.

Analityczne rozwiązanie tego problemu podał Blaise Pascal (1623 – 1662).

Napisz funkcję `demere`, która posługując się metodą Monte Carlo znajduje przybliżone prawdopodobieństwa opisanych wyżej zdarzeń A i B. Funkcja `demere` przyjmuje argument N – liczbę punktów losowych, i zwraca obliczone prawdopodobieństwa.

Korzystając z tego szablonu napisz program `demere`, który czterokrotnie wypisze na standardowe wyjście prawdopodobieństwa zdarzeń A i B: za pierwszym razem program powinien podać prawdopodobieństwa znalezione przez Ciebie w literaturze, a następnie ich przybliżone wartości obliczone przez funkcję `DeMere` za pomocą trzech różnych generatorów liczb pseudolosowych: liniowego generatora kongruencyjnego, generatora *Mersenne Twister 19937* oraz dwóch innych generatorów, których implementacje uda Ci się znaleźć lub napisać.

Czy wybór generatora znacząco wpływa na jakość wyniku?

Opracowanie: Bartłomiej Zglinicki.