Adam Przemysław Chojecki, 298814, grupa 1, projekt 1, zadanie 60

Wstęp

Zadanie polegało na zaimplementowaniu algorytmów typu Monte Carlo do całkowania funkcji na zbiorze $[0,1]\times[0,1]$. Zaimplementowano 6 metod. 4 losowe (basic, Imporftance Sampling, Latin Hypercube, Divide And Conquer) oraz 2 deterministyczne (quasiMC, Reimann).

Cała implementacja i testy przeprowadzone zostały za pomocą oprogramowania MATLAB, a kod źródłowy dostępny jest publicznie na repozytorium GitHub [1].

Krótki opis zaimplementowanych metod

Metody Basic, quasiMC, Reimann oraz Latin Hypercube korzystają z następującego wzoru:

$$\int_{[0,1]\times[0,1]} f(x,y)dxdy \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i,y_i) \tag{1}$$

Basic (pol. Podstawowa)

We wzorze (1) punkty $x_i, y_i \sim U(0,1)$ to niezależne zmienne losowe.

quasiMC

Deterministyczny algorytm. We wzorze (1) punkty $(x_i,y_i)_{i=1}^n$ są ciągiem Faure [2].

Reimann

Deterministyczny algorytm. We wzorze (1) punkty $(x_i,y_i)_{i=1}^n$ są z siatki $\{(\frac{j}{\lfloor\sqrt{n}\rfloor},\frac{k}{\lfloor\sqrt{n}\rfloor})\}_{j,k=1}^{\lfloor\sqrt{n}\rfloor}$.

Imporftance Sampling (pol. Próbkowanie Ważone)

Losuje punkty $x_i \sim Beta(\alpha_1,\beta_1)$, $y_i \sim Beta(\alpha_2,\beta_2)$, gdzie parametry α_j,β_j są podane przez użytkownika. Całka przybliżana jest wzorem $\frac{1}{n}\sum_{i=1}^n f(x_i,y_i)/(g_1(x_i)\cdot g_2(y_i))$, gdzie g_j to gęstość rozkładu $Beta(\alpha_j,\beta_j)$.

Metoda ta działa dobrze, jeśli gęstość odpowiedniego rozkładu $Beta$ dobrze przybliża funkcję podcałkową.

Latin Hypercube (pol. Łacińska hiperkostka)

We wzorze (1) punkty $x_i, y_i \sim U(0,1)$, ale kolejne pary są zależne od siebie tak, aby były równiej rozprowadzone po powierzchni kostki.

Divide And Conquer (pol. Dziel i rządź)

Algorytm wykorzystuje metodę warstwowania (ang. Stratified) [5], gdzie estymuje optymalny podział pozostałego budżetu. Następnie rekurencyjnie całkuje oddzielnie każdą ćwiartkę kwadratu podzielonego na 4 części.

Metoda ta działa dobrze dla funkcji podcałkowych takich, które mają mały fragment na którym funkcja się zmienia i duży na którym funkcja jest bliska stałej. Przykładami takich funkcji są funkcje schodkowe.

Eksperymenty numeryczne

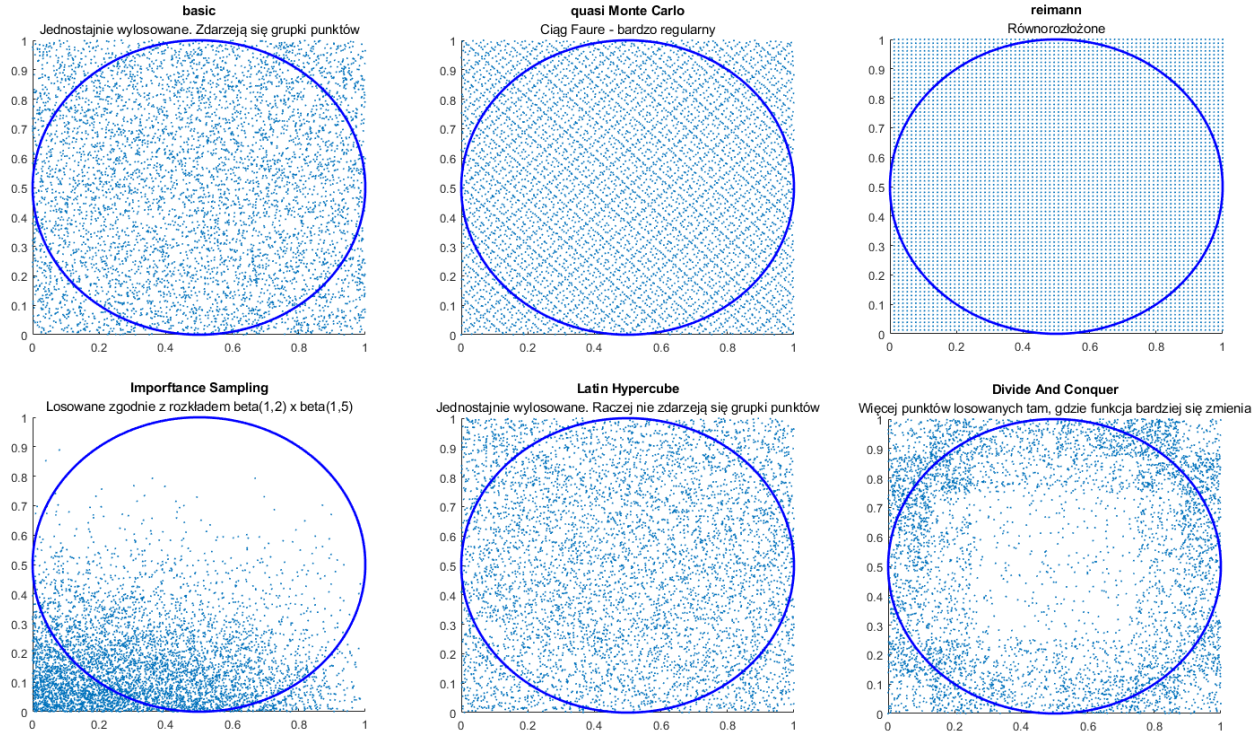
Różnice między algorytmami

Na rysunku 1 pokazano jakie punkty były wybierane przez poszczególne algorytmy do całkowania funkcji dającej 1 w zaznaczonym kole i 0 poza nim. Obrazek powstał w celu pokazania różnic między różnymi podejściami algorytmów.

W szczególności widać na nim, że algorytm quasiMC przyjmuje bardzo regularne punkty. Okazuje się, że jest to najlepszy spośród testowanych algorytmów przy ustalonej liczbie punktów. Niestety, wymaga on kilkaset razy więcej czasu niż pozostałe algorytmy, gdyż powolnym

jest wyliczanie ciągu Faure. Z tego powodu nadaje się w praktyce jedynie do całkowania funkcji czasochłonnych w obliczaniu, gdzie wykorzystuje się mało punktów do estymacji całki.

Innym ciekawym przykładem jest Divide And Conquer, która losuje więcej punktów z tych obszarów, które bardziej się zmieniają. W przypadku tej funkcji podcałkowej jest to obwód koła.

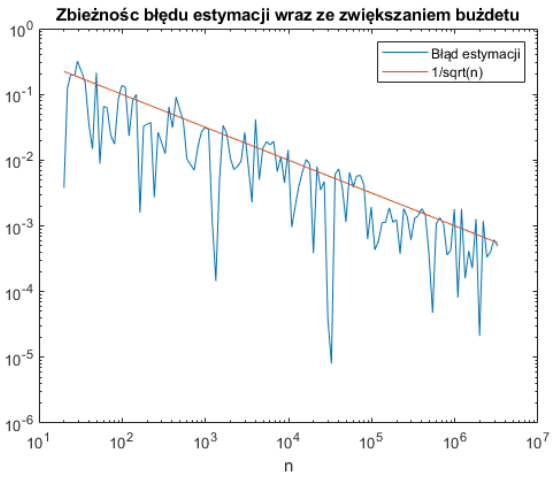


Rysunek 1: Przykładowe punkty, które różne algorytmy wybierały do estymowania całki z funkcji dającej 1 w zaznaczonym kole i 0 poza nim. Widzimy wyraźne różnice między podejściami każdego z algorytmów.

Zbieżność

W [3] pokazane jest z Centralnego Twierdzenia Granicznego, że bazowa wersja metody Monte Carlo estymuje w sposób nieobciążony (średnio wartość wyestymowanej całki jest dokładna), a odchylenie standardowe tego estymatora jest proporcjonalne do $\frac{1}{\sqrt{n}}$, gdzie n to budżet estymacji. Co ciekawe, odchylenie to jest niezależne od wymiaru przybliżanej całki. Wynik ten słusznie sugeruje, że metody Monte Carlo powinny dobrze radzić sobie nawet w sytuacji z dużą liczbą wymiarów.

Na obrazku 2 po prawej obserwujemy tą zbieżność dla funkcji podcałkowej $f(x, y) = x^3 \cdot y + 4 \cdot x \cdot y^2$. Widzimy, że obserwowany błąd estymacji całki maleje proporcjonalnie do $\frac{1}{\sqrt{n}}$.



Rysunek 2: Zbieżność błędu w podstawowej wersji MC wraz ze zwiększającym się budżetem.

Bibliografia

[1] Przemysław Chojecki. *Strona GitHub zawierająca kod do tego projektu.* `https://github.com/PrzeChoj/numerki`. [Online; 11-12-2022].

[2] Dirk P. Kroese, Thomas Taimre i Zdravko I. Botev. *Handbook of Monte Carlo Methods.* Wiley, 2011.

[3] Maciej Romaniuk. *Metody Monte Carlo.* Sty. 2019. ISBN: 978-83-7814-864-7.

[4] Dimitri Shvorob. *Generate a Faure sequence; MATLAB Central File Exchange.* `https://www.mathworks.com/matlabcentral/fileexchange/15373-generate-a-faure-sequence`. [Online; 11-12-2022]. 2016.

[5] Ad Ridder Zdravko Botev. *Variance Reduction.* `https://web.maths.unsw.edu.au/~zdravkobotev/variancereductionCorrection.pdf`. [Online; 11-12-2022].

Uwagi

1. Kod źródłowy tego projektu dostępny pod adresem [1] korzysta z:

    - generatora liczb pseudolosowych z rozkładu Beta pochodzącego z Statistics and Machine Learning Toolbox
    - generatora ciągu Fature dostępnego pod adresem [4] i będącego na licencji jego autora dostępnej w pliku ‘numerki/MonteCarlo/Algorithms/license.txt‘ na repozytorium projektu.

2. W kodzie źródłowym brak jest polskich znaków. Spowodowane jest to negatywnym doświadczeniem autora z problememi jakie sprawia używanie ich w kodach źródłowych.