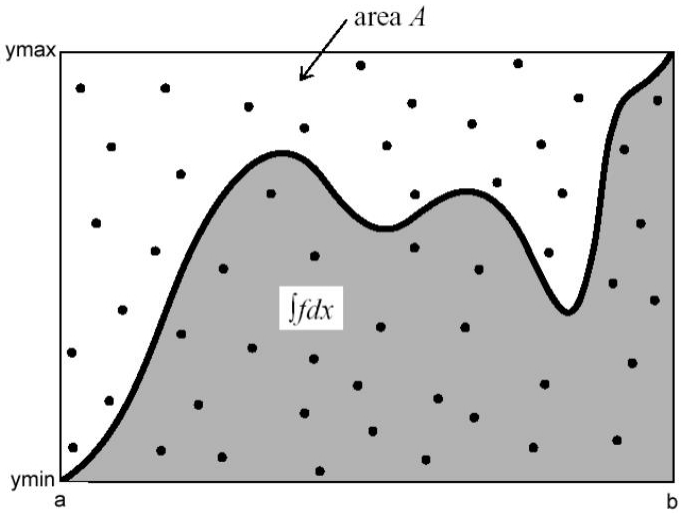
Propozycja II projektu zaliczeniowego z PiMN:

n-wymiarowe całki,

czyli metoda całkowania Monte Carlo

Opracowanie *Jędrzej Wardyn*

1 lipca 2024



Rysunek 1: Przykład całkowania Monte Carlo w jednym wymiarze

Wszelkie pytania proszę kierować na mail: `j.wardyn@uw.edu.pl`

1 Całkowanie numeryczne: Metoda Monte Carlo

Całkowanie numeryczne w jednym wymiarze możemy przeprowadzić na przykład metodami Newtona-Cotesa, metodami opartymi o kwadratury Gaussa, które w zależności od odwzorowania, wymaganej precyzji czy szybkości działania spełnia swoją funkcję.

Natomiast w wyższych wymiarach (od 3 wzwyż) preferowaną metodą całkowania staje się Monte Carlo z uwagi na skalowalność jak i prostotę w parallelizacji takiej metody, choć wciąż może być używana do przypadków 1D i 2D.
Tutaj na temat przewagi Metody Monte Carlo nad metodami kratowymi/siatkowymi w wyższych wymiarach

Celem tego projektu byłoby skonstruowanie programu, który mógłby wykorzystywać całkowanie Monte Carlo w wersji z próbkowaniem prostym oraz w wersji z próbkowaniem z uwagi na ważność [simple, importance sampling] i porównaniem ich zakresów stosowania w zależności od wymiaru. Więcej informacji na temat metod samplowania oraz metodzie Monte Carlo w ogólności można znaleźć w tej książce.

Celem sprawdzenia jakości wykonanego całkowania proponuję wstępnie przetestować całkowanie na funkcjach o znanych funkcjach pierwotnych.
Następnie, proszę przetestować jak sprawuje się importance sampling choćby na przykład $f(x) = x^{sin(x^2)}$ na jakimś sensownym przedziale. Prosiłbym o dobór kilku przykładów, w celu jak najlepszego pokazania tej metody. Jednym ale nie

jedynym źródłem inspiracji może być Encyklopedia funkcji matematycznych (odwzorowań) jak też wcześniej wspomniana książka na temat metody Monte Carlo.

Prosiłbym o należyty dobór funkcji w celu jak najlepszego pokazania stosowalności metody w jedynym i dwóch wymiarach i jakiś przykład dla trzech wymiarów. Program ma być jednak zaprojektowany tak, żeby można było przeprowadzać całkowanie na dowolnej liczbie wymiarów, a nie wyłącznie w 1D i 2D.

1.1 Co widziałbym, że ma się znaleźć:

1. Zaimplementowana metoda całkowania MC z samplingiem prostym dla dowolnego wymiaru
2. Zaimplementowana metoda całkowania MC z importance sampling dla dowolnego wymiaru
3. Wyrysowywanie dla przypadku 1D, żeby mieć jakiś pogląd jak to działa, ale jak mamy naprawdę dużą liczbę sampli (utrudniającą rysowanie) to wszystkich nie musimy rysować.
4. Input: system doboru z listy lub/i wpisywania funkcji dowolniewymiarowych, liczba sampli, jaka metoda jest stosowana i więcej parametrów jeśli jest taka potrzeba
5. Output: poza samą wartością całki w outpucie który powinien być jakoś zapisany tekstowo [można wszystkie zbiorczo do pliku .csv albo osobno] oraz parametrami inputu powinien się tam znaleźć czas liczenia dla danej całki daną metodą.
6. W osobnym lub tym samym programie powinna być przeprowadzona jakaś analiza z wykresami, dotycząca stosowalności poszczególnych metod numerycznych stosowanych w projekcie w zależności od wymiaru. Jak bardzo wydajne są te metody jak chodzi o czas, precyzję, stosunek czasu oraz precyzji do liczby użytych w całkowaniu punktów ($\frac{\text{precyzja}}{N}, \frac{\text{czas}}{N}$) i jak to zależy od wymiarowości problemu.
7. Proszę porównać w formie wykresu w jednym wymiarze $\frac{\text{precyzja}}{N}$ dla wybranej dowolnej z metod Newtona-Cotesa czy metod kwadratur Gaussa. Wystarczy porównać 3 istotnie odmienne od siebie odwzorowania (czyli **NIE** $\sin(x)$,$\sin(2x)$ i $\sin(3x)$ bo są zbyt podobne).

1.2 Dla uproszczenia:

1. Nie ma wymagania użycia C++, można również python lub inny język.
2. Utworzenie czegoś co przyjmowałby nam funkcje np jako argument main może być dość złożone (zatem niewymagane) więc input można zrobić częściowo za pomocą hardkodowania, czyli definicje funkcji będą już wpisane w kodzie i można je sobie po prostu wybrać. Jak długa będzie to lista, czy ma to jakieś optymalizacje zależy od tego kto podejmie się projektu, niemniej ma to realizować założenia opisane powyżej i poniżej.
3. Jakieś strasznie wyszukane przykłady dla 3 i więcej wymiarów jak i rysunki nie są wymagane, ale jakiś przykład choćby jednej całki dla 3D i 4D by się przydał.
4. Całki nie muszą być jakieś turboskomplikowane, szczególnie pod względem obliczeniowym, jeśli coś zajmuje za dużo czasu to proszę wykorzystać coś prostszego, czy to prostszą funkcję, obszar całkowania itp.
5. Jeżeli są problemy z implementacją liczenia czegoś dowolniewymiarowego, to proszę przyjąć skończony maksymalny wymiar ale w miarę duży na przykład 10.

Dla chętnych **[co może podbić punkty kapkę w sytuacji krytycznej]** Implementacja choćby 1D, ale metody samplowania quasi-Monte Carlo. Obowiązują jednakowe wymagania co do zwykłej.

2 Wymagania i obwieszczenia:

Poniżej umieszczone są warunki, które wpływają na ostateczną punktację projektu.

Kod projektu:

1. Powinien mieć wszelkie potrzebne pliki wraz z zewnętrznymi bibliotekami już na miejscu w jednym pakiecie, jeśli to konieczne, jeśli komplikacja jest spora proszę korzystać z Cmake.
2. Powinien posiadać czytelną i zrozumiałą instrukcję (w osobnym pliku) kompilacji i użytkowania programu.
3. **Powinien kompilować/uruchamiać się (brak tego warunku daje 0 punktów!).**
4. Powinien uruchamiać się bez błędów w stylu `segmentation fault`
5. Nie powinien skutkować jakimś szkodliwym działaniem, na przykład nadprodukcją zapisywanych plików zapełniających całą dostępną pamięć.
6. Przy wprowadzaniu lub użyciu danych, otwieraniu plików i innych czynnościach zależnych od pewnych czynników program powinien sprawdzać czy operacja została wykonana poprawnie, jeśli nie to powinien zostać pokazany odpowiedni do sytuacji komunikat błędu.
7. Kod programu powinien realizować wytyczne z opisu projektu.
8. (tam gdzie jest to możliwe) Warto jakby nie było zbytecznej redundancji
9. Warto by program nie był powolny lub nad wyraz zasobożerny polecam flagę (-O3 chyba, że macie wyraźne powody by jej nie użyć).
10. W kodzie dobrze jakby nie było części bezużytecznych (dodajcie `-Wall` do flag kompilacji), śmieci pozostałych po tworzeniu rozwiązania, proponuję wyczyścić kod a potem dodać komentarze.
11. W miarę możliwości kod ma być przejrzysty, opatrzony komentarzami w celu szybszego zrozumienia sprawdzającego, (co nie znaczy, że nie można używać jakichś trików). Komentarze wewnątrz pliku z kodem nie osobno!

Punkty 1-6 są najważniejsze, ale spełnienie ich nie gwarantuje maksymalnej ilości punktów.

Wszelkie pytania proszę kierować na mail: `j.wardyn@uw.edu.pl`