



Laboratorium 2

Symulacja Monte Carlo

dr inż. Jarosław Rulka
jaroslaw.rulka@wat.edu.pl

Wprowadzenie

- Stanisław Ulam, Nicolas Metropolis: *The Monte Carlo Methods*, 1949
- Metoda Monte Carlo (MC) – technika rozwiązywania problemu wykorzystująca losowe ciągi liczb
- Opiera się na generatorach liczb losowych

Wprowadzenie

- MC polega na przedstawieniu rozwiązania postawionego problemu w postaci **parametru** pewnej **hipotetycznej populacji** i używaniu **losowej sekwencji liczb** do tworzenia próbki tej populacji, na podstawie której można dokonać **statystycznego oszacowania wartości** badanego parametru;
- F – dokładne rozwiązanie problemu:
 - liczba, zbiór liczb, wartość logiczna – decyzja;
- Oszacowanie wyniku F:
$$\hat{F} = f(\{r_1, r_2, \dots, r_n\})$$
 - $\{r_1, r_2, \dots, r_n\}$ - zastosowane liczby (pseudo)losowe;

Wprowadzenie

- Problemy deterministyczne

- Całkowanie;

- Znajdowanie pól i objętości;

- Obliczanie liczby π ;

- Problemy probabilistyczne/stochastyczne:

- Symulacje procesów zależnych od zmiennych losowych:

- Systemy masowej obsługi;

- Metody biologicznie inspirowane, np. stadne;

- Szukanie ekstremum;

Zadanie 1

- Wyznaczyć całkę oznaczoną w przedziale $[a, b]$ zadany przez użytkownika. Inaczej jest to pole powierzchni pod funkcją podcałkową.
- Funkcje do całkowania

A.

$$\int_1^e \frac{3}{x} dx$$

$$\int \frac{dx}{x} = \ln|x| + C$$

$$\int \frac{3}{x} dx = 3 \int \frac{1}{x} dx = 3 \ln|x| + C$$

$$\int_1^e \frac{3}{x} dx = [3 \ln|x|]_1^e = 3 \ln|e| - 3 \ln|1| = 3 \ln e - 3 \ln 1 = 3 \cdot 1 - 3 \cdot 0 = 3$$

B. Funkcja zaproponowana przez studenta

Zadanie 1 cd.

- Oszacuj wartość całki funkcji podcałkowej metodą Monte Carlo poprzez:
 - losowy (z rozkładem równomiernym) wybór n punktów (x_i, y_i) , przy czym $x_i \in \langle a, b \rangle$, $y_i \in \langle 0, f_{max} \rangle$ (losowanie w prostokącie);
 - sprawdzenie, czy punkt $y_i \leq f(x_i)$ i zliczania takich punktów (licznik k);
 - pole powierzchni pod funkcją f (całka oznaczona)

$$P \approx \frac{k}{n}(b - a)f_{max}$$

- Zbadaj dokładność/błąd oszacowania zależnie od liczby punktów n ;

Zadanie1 – koncepcja rozwiązania

- Zdefiniuj interfejs funkcyjny `IFunc` z metodami:
 - `double func(double x),`
 - `double max(double a, double b).`
- Zdefiniuj 2 klasy konkretnych funkcji do całkowania, które implementują interfejs `IFunc`:
 - `Funkcja1,`
 - `Funkcja2.`
- Zdefiniuj klasę `Calka` z główną funkcją wyznaczającą całkę, która przyjmuje cztery parametry: granice przedziału `[a, b]`, obiekt funkcyjny implementujący interfejs `IFunc`, liczbę powtórzeń, i zwracającą wartość całki:
 - `double calculate(double a, double b, IFunc f, int rep),`
- Zdefiniuj klasę testową (uruchomieniową) `Main` w celu zademonstrowania działania programu.

Zadanie 2

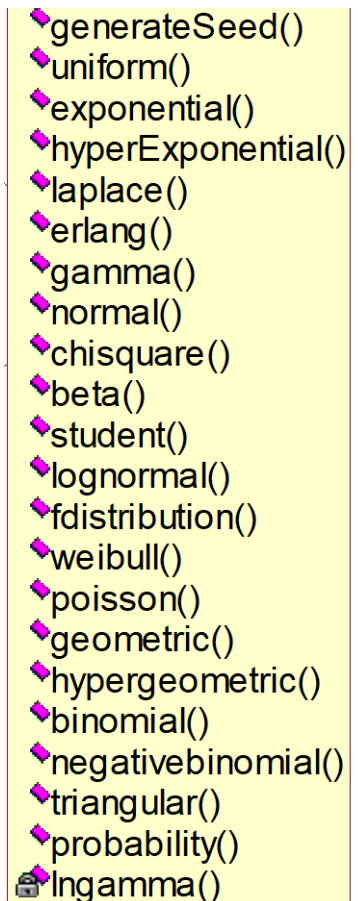
- Problem stochastyczny:
 - System składa się z N linii (stanowisk obsługi), z których każda może obsługiwać klientów;
 - W losowych chwilach czasu pojawiają się zgłoszenia:
 - każde zgłoszenie trafia najpierw na linię nr 1;
 - jeśli w chwili pojawienia się k -tego zgłoszenia (oznaczymy chwilę przez t_k) linia jest wolna, to zgłoszenie jest obsługiwane w ciągu czasu To_i minut (czas obsługi i -tej linii jest wartością stałą zadawaną osobno dla każdej linii);
 - jeśli natomiast w momencie t_k linia jest zajęta, wówczas zgłoszenie przekazywane jest na kolejną itd.;
 - jeśli wszystkie linie w chwili t_k są zajęte, układ odmawia obsługi zgłoszenia (odrzuca je);

Zadanie 2 – koncepcja rozwiązania

- Zdefiniować główną klasę definiującą system S_{mo} z metodą symulującą jego działanie:
 - `void symuluj(double czasZakon, int liczZglosz).`
 - Struktura N-elementowa obiektów klasy `Linia` zawierająca informacje o:
 - czasie obsługi To_i ;
 - czasie zajętości (do kiedy?):
 - $t_k + To_i$ – gdy zajęte przez zgłoszenie k-te do chwili $t_k + To_i$;
 - W metodzie `symuluj()` wykonuj w pętli:
 - losowanie czasów kolejnych zgłoszeń,
 - aktualizuj czas systemowy (symulacyjny `double`) `simTime`,
 - zajmij pierwszą wolną lub odrzuć zgłoszenie
 - sprawdź zajętości linii (porównując bieżący czas systemowy z czasem zajętości linii);
- Symulację zakończ po zadany czasie lub po wygenerowaniu zadanej liczby zgłoszeń (zależnie, co wystąpi wcześniej);
- Oszacuj, ile (względnie i bezwzględnie) zgłoszeń obsłuży system a ile razy odrzuci zgłoszenie;

Zadanie cd.

- Zaimplementuj odpowiednie klasy Java dla obu zadań w osobnych pakietach o nazwach zadanie1, zadanie2;
- Zastosuj klasę generatora liczb losowych RNGenerator z biblioteki `dissimlab2021`, w szczególności użyj metod:
 - `uniform(double a, double b)` – parametry wywołania to granice przedziału `[a,b]`;
 - `exponential(double a)` – parametrem jest wartość oczekiwana zmiennej losowej;



A vertical list of 21 methods from the `dissimlab2021` library, each preceded by a small purple diamond icon. The methods are: `generateSeed()`, `uniform()`, `exponential()`, `hyperExponential()`, `laplace()`, `erlang()`, `gamma()`, `normal()`, `chisquare()`, `beta()`, `student()`, `lognormal()`, `fdistribution()`, `weibull()`, `poisson()`, `geometric()`, `hypergeometric()`, `binomial()`, `negativebinomial()`, `triangular()`, `probability()`, and `lngamma()`.

- generateSeed()
- uniform()
- exponential()
- hyperExponential()
- laplace()
- erlang()
- gamma()
- normal()
- chisquare()
- beta()
- student()
- lognormal()
- fdistribution()
- weibull()
- poisson()
- geometric()
- hypergeometric()
- binomial()
- negativebinomial()
- triangular()
- probability()
- lngamma()