POLITECHNIKA POZNAŃSKA

WYDZIAŁ INFORMATYKI I TELEKOMUNIKACJI
Instytut Informatyki

Praca dyplomowa licencjacka

ZASTOSOWANIE WSPÓŁCZESNYCH GENERATORÓW LICZB PSEUDOLOSOWYCH W SYMULACJACH UKŁADÓW Z NIEWIELKĄ LICZBĄ STOPNI SWOBODY ZA POMOCĄ METODY MONTE CARLO

Kacper Kalinowski, 145128

Promotor
prof. dr hab. Krzysztof W. Wojciechowski

POZNAŃ 2022

Tutaj będzie karta pracy dyplomowej;
oryginał wstawiamy do wersji dla archiwum PP, w pozostałych kopiach wstawiamy ksero.

Spis treści

Rozdział 1

Wstęp

1.1 Cel i zakres pracy

Celem pracy jest zapoznanie się z problematyką symulacji Monte Carlo modeli układów z niewielką liczbą swobody oraz opanowanie wybranych metod numerycznych (w tym niektórych metod symulacji komputerowych), które służą do badania takich modeli.

Szczegółowe zadania podjęte podczas realizacji pracy inżynierskiej:

- Zapoznanie się z literaturą dotyczącą tematyki badań, w szczególności generatorów liczb pseudolosowych i metod Monte Carlo przedstawionych w artykułach Profesora Wojciechowskiego [1] [2] [3] i łączone z Profesorem Tretiakovem [4] [5]
- Zapoznanie się z metodami programowania w FORTRANie i w C
- Przygotowanie i przetestowanie procedur do programów symulacyjnych
- Wykonanie symulacji wybranych modeli fizycznych
- Przygotowanie pracy opisującej wyniki badań

W swojej pracy korzystam ze źródeł literaturowych poruszających zagadnienia związane z tematem generatorów liczb pseudolosowych, między innymi: Mersenne Twister, Holiana SNWS oraz RAN2. Kolejne opisują układy fizyczne z małą liczbą stopni swobody takie jak na przykład: sprężyna, oscylator harmoniczny czy wahadło matematyczne.

1.2 Wprowadzenie

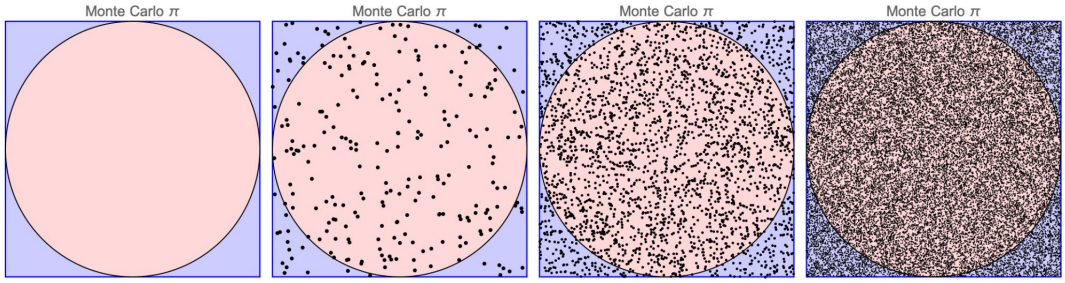
1.3 Uwagi

Rozdział 2

Podstawy teoretyczne

2.1 Metoda Monte Carlo

Ta metoda służy do matematycznego modelowania zdarzeń i obiektów, które są zbyt złożone, aby przewidzieć ich wynik za pomocą metod analitycznych. Najważniejszą rolą w tej metodzie jest losowy wybór wielkości charakterystycznej dla danego procesu, a dobór musi być zgodny ze znanym nam rozkładem. Typowe procesy obliczone metodą Monte Carlo to te, które są bardzo trudne lub niemożliwe do rozwiązania innymi metodami. Główne problemy, w których stosuje się te metody, to: [6] optymalizacja, całkowanie numeryczne, czy generowanie grafów z rozkładów prawdopodobieństwa.



Rysunek 2.1: Obliczanie π przy użyciu metody Monte Carlo
Źródło: https://thatsmaths.files.wordpress.com/2020/05/monte-carlo-wide-4panel-1.jpg

W fizyce metody Monte Carlo są bardzo przydatne w rozwiązywaniu problemów w układach z małą stopni liczbą swobody. Układami, którymi zająłem się w tej pracy są: wahadło matematyczne, oscylator harmoniczny i sprężyna.

W zasadzie wszystkie probabilistycznie interpretowalne problemy można rozwiązać tymi metodami. Zgodnie z Prawem Wielkich Liczb (LLN) całkę opisaną przez można aproksymować do wartości oczekiwanej zmiennej losowej, biorąc średnią empiryczną (średnią z próby) niezależnych prób tej zmiennej. [7]

Pomimo prostych koncepcji i algorytmów, złożoność obliczeniowa związana z symulacjami Monte Carlo może być bardzo duża. Ta metoda wymaga wielu próbek, aby uzyskać zadowalającą symulację i może skutkować długimi czasami przebiegu, jeśli przetwarzanie jednej próbki zajmuje dużo czasu. [8] Jest to poważne ograniczenie bardzo złożonych problemów, które można wyeliminować dzięki bardzo prostej strukturze algorytmicznej, zmniejszającej koszty obliczeń poprzez równoległe przetwarzanie na procesorach, klastrach, chmurach czy GPU. [9] [10]

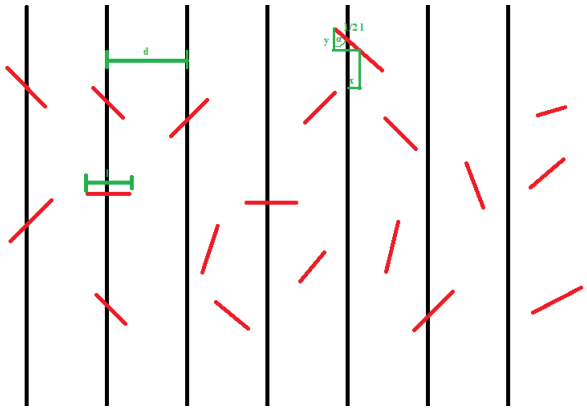
Metody Monte Carlo mogą się różnić między sobą, ale schemat bazowy zawsze jest podobny:

- Określenie dziedziny możliwych danych wejściowych
- Generowanie danych wejściowych w sposób losowy z rozkładu prawdopodobieństwa w tej dziedzinie.
- Wykonanie deterministycznych obliczeń na danych wejściowych
- Podsumowanie wyników

Zastosowanie metod Monte Carlo wymaga dużych ilości liczb losowych, obecnie wykorzystywane są do tego generatory liczb pseudolosowych.

2.1.1 Historia

Przed opracowaniem metody Monte Carlo testowane symulacje były wcześniej zrozumiałymi i znanymi problemami deterministycznymi, a do oszacowania niepewności tych symulacji wykorzystano próbkowanie statystyczne. Monte Carlo odwraca to podejście i wykorzystuje algorytmy probabilistyczne do rozwiązywania problemów deterministycznych. Pierwsze podejście do metody Monte Carlo zostało opracowane przez rozwiązanie problemu igły Buffona, gdzie π można oszacować, upuszczając igły na podłogę składającą się z równoległych, równomiernie rozmieszczonych pasków. W latach trzydziestych Enrico Fermi użył metody Monte Carlo do badania dyfuzji neutronów, ale jego praca nigdy nie została opublikowana. [11]



Rysunek 2.2: Igła Buffona
Źródło: Rysunek własny

Nieoceniony do powstania metod Monte Carlo jest wpływ polskiego naukowca. Pod koniec lat czterdziestych Stanisław Ulam, pracując nad projektem broni jądrowej, wynalazł nowoczesną wersję metody łańcuchowej Markowa Monte Carlo. W 1946 roku fizycy z Los Almos National Laboratory badali dyfuzję neutronów w rdzeniu broni jądrowej. [11] Posiadali oni większość potrzebnych do obliczeń danych takich jak średnia odległość jaką neutron przebyłby w materii zanim zderzyłby się z jądrem, oraz ilość energii, jaką neutron uwalnia po zderzeniu, mimo to nie byli w stanie rozwiązać problemu za pomocą powszechnie używanych deterministycznych metod matematycznych. Ulam zasugerował użycie eksperymentów losowych. Na pomysł ten wpadł rozważając sposoby ułożenia pasjansa podczas swojej rekonwalescencji po chorobie. Uznał on, że łatwiej będzie ułożyć go np. 100 razy i sprawdzić liczbę rozwiązań niż rozważać problem obliczeniami kombinatorycznymi. [12]

Rysunek 2.3: Stanisław Ulam w Los Almos
Źródło: https://upload.wikimedia.org/wikipedia/commons/thumb/8/82/Stanislaw$_U$lam.tif/lossy−page1 − 620px − Stanislaw$_U$lam.tif.jpg

Jako, że praca Von Neumana i Ulama była tajna, wymagała ona kryptonimu [13]. Ich kolega, Nicholas Metropolis, zasugerował użycie nazwy Monte Carlo, która odnosi się oczywiście do ówczesnej stolicy hazardu, Monako, i kasyna znajdującego się w niej o takiej właśnie nazwie. Metody Monte Carlo umożliwiły dalsze prace nad symulacjami do Projektu Manhattan, choć ich pełny potencjał nie mógł zostać jeszcze osiągnięty przez ograniczenia związane z technologią. Ci sami naukowcy zaprogramowali komputer ENIAC do wykonywania pierwszych w pełni zautomatyzowanych obliczeń Monte Carlo w 1948r. [14]

Teoria bardziej złożonej, cząsteczkowej metody Monte Carlo typu mean-field zaczęła się rozwijać w połowie lat 60.XX wieku, wraz z pracą Henry'ego P. McKeana Jr. nad interpretacjami Markowa pewnej klasy nieliniowych parabolicznych cząsteczkowych równań różniczkowych w mechanice płynów. [15] [16] Metody Monte Carlo typu mean-field genetic są wykorzystywane jako heurystyczne algorytmy przeszukiwania naturalnego (metaheurystyka). w obliczeniach ewolucyjnych. Początki tych technik sięgają lat 1950 i prac Alana Turinga nad maszynami uczącymi się typu genetycznego z mutacja i selekcją [17].

Sekwencyjne metody Monte Carlo dla zaawansowanego przetwarzania sygnałów i wnioskowaniu są zdecydowanie nowsze. W 1993 roku Gordon i inni opublikowali w swojej pracy [18] pierwsze zastosowanie algorytmu resamplingu Monte Carlo we wnioskowaniu statystycznym Bayesa. Autorzy nazwali swój algorytm "filtrem bootstrapowym"i wykazali, że w porównaniu z innymi metodami filtrowania, ich algorytm bootstrapowy nie wymaga żadnych założeń dotyczących przestrzeni stanów ani szumu systemu.

Nie ma jednoznacznej definicji Monte Carlo. Sawilowsky [19] rozróżnia symulację, metodę Monte Carlo i symulację Monte Carlo: symulacja to fikcyjna reprezentacja rzeczywistości, podczas gdy metoda Monte Carlo to technika, która może rozwiązać problemy matematyczne lub statystyczne, a symulacja Monte Carlo wykorzystuje wielokrotne próbkowanie w celu uzyskania statystycznych właściwości zjawiska (lub zachowania).

2.1.2 Liczby losowe

Główną ideą tej metody jest to, że wyniki są obliczane na podstawie powtarzanego próbkowania i analizy statystycznej. Symulacje Monte Carlo są w rzeczywistości eksperymentami losowymi, w przypadku, gdy, wyniki tych eksperymentów nie są dobrze znane. Symulacje te zazwyczaj zawie-

rają wiele nieznanych parametrów, z których wiele jest trudnych do uzyskania doświadczalnie. [20] Prawdziwie liczba losowa, nie zawsze jest użyteczna w metodach symulacji Monte Carlo. Wiele z najbardziej użytecznych technik wykorzystuje deterministyczne pseudolosowe sekwencje w celu ułatwienia testowania i ponownej symulacji. Jedyną cechą zwykle wymaganą do przeprowadzenia dobrej symulacji jest to, że pseudolosowa sekwencja liczb wydaje się w pewnym sensie „wystarczająco losowa". Jednym z najprostszych i najczęstszych sposobów jest sprawdzenie, czy liczby są równomiernie rozłożone, czy też podążają za innym pożądanym rozkładem przy wystarczającej liczbie elementów sekwencji. Często pożądane/niezbędne są również słabe korelacje pomiędzy kolejnymi próbami.

Sawilowsky wymienia cechy wysokiej jakości symulacji Monte Carlo: [19]

- generator liczb (pseudolosowych) ma określone cechy (np. długi ókres"przed powtórzeniem się sekwencji)
- generator liczb (pseudolosowych) wytwarza wartości, które przechodzą testy na losowość
- liczba próbek jest wystarczająca, aby zagwarantować dokładne wyniki
- zastosowano prawidłową technikę próbkowania
- zastosowany algorytm jest prawidłowy dla tego, co jest modelowane
- symuluje on dane zjawisko.

Algorytm próbkowania pseudolosowego służy do konwersji liczb pseudolosowych o jednorodnym rozkładzie na liczby o rozkładzie według określonego rozkładu prawdopodobieństwa.

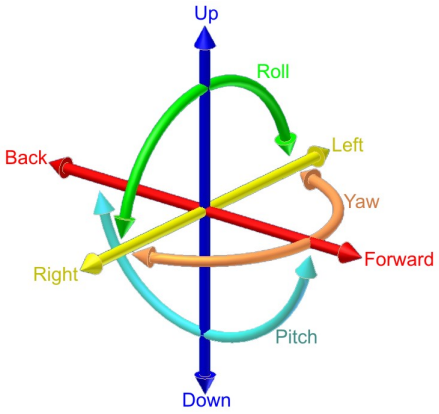
Ciągi niskodyskretne są często stosowane zamiast losowego próbkowania z przestrzeni. Dzieje się tak, ponieważ zapewniają one jednolite pokrycie i zazwyczaj mają szybszy rząd zbieżności niż symulacje Monte Carlo z losowymi lub pseudolosowymi sekwencjami. Metoda oparta na jej wykorzystaniu nazywana jest metodą Quasi-Monte Carlo.

Aby ocenić wpływ jakości liczb losowych na wyniki symulacji Monte Carlo, astrofizycy porównali bezpieczne kryptograficznie liczby pseudolosowe wygenerowane przy użyciu instrukcji Intel RDRAND z liczbami wygenerowanymi przez algorytmy takie jak Mersenne Twister, w symulacjach Monte Carlo dotyczących rozbłysków radiowych pochodzących od brązowych karłów. RDRAND to generator liczb pseudolosowych, który jest najbliższy prawdziwemu generatorowi liczb losowych. Nie znaleziono statystycznie istotnej różnicy między modelami generowanymi przez konwencjonalne generatory liczb pseudolosowych i RDRAND dla prób wygenerowania 10^7 liczb losowych.. [21]

2.2 Układy fizyczne o małej liczbie stopni swobody

Stopnie swobody to zmienne, które pozwalają nam opisać stan układu fizycznego, a liczba stopni swobody to minimalna wymagana liczba zmiennych niezależnych do jedoznacznego opisaniu stanu układu. [22] [23] Metody Monte Carlo są bardzo przydatne do symulacji zjawisk o dużych wejściowych niepewnościach i systemów z wieloma powiązanymi stopniami swobody.

W mechanice klasycznej stopień swobody jest dowolną zmienną niezbędną do opisania stanu układu fizycznego. Oznacza to, że opisuje położenie jegoposzczególnych części w przestrzeni [23]. Liczba stopni swobody jest równa minimalnej liczbie zmiennych niezależnych wymaganych do jednoznacznego opisania stanu układu. Liczba ta zależy od liczby części składających się na układ i natury nałożonych więzów [23]. W przypadku układów mechanicznych stopnie swobody są współrzędnymi uogólnionymi. Na przykład ciało punktowe w przestrzeni ma trzy stopnie swobody, ciało ślizgające się pod dowolnej przestrzeni ma dwa stopnie swobody, a wahadło drgające w płaszczyźnie jeden. Ciało sztywne - 6 stopni swobody: 3 współrzędne translacyjne środka ciężkości i 3 współrzędne rotrotacyjne - współrzędne kątowe określające obrót bryły w przestrzeni [23]



Rysunek 2.4: 6 stopni swowbody
Źródło: https://upload.wikimedia.org/wikipedia/commons/f/fa/6DOF$_e n.jpg$

2.2.1 Wahadło matematyczne

Wahadło to ciało zawieszone w jednolitym polu grawitacyjnym, które może obracać się wokół osi poziomej, która nie przechodzi przez środek ciężkości zawieszonego ciała. Istnieją dwa podstawowe modele fizyczne wahadeł w mechanice klasycznej [24]:

1. matematyczne (proste) – opisujące wahadło jako punkt materialny zawieszony na nieważkiej linie,

2. fizyczne – opisujące wahadło jako ciało sztywne.

Ważną właściwością wahadła matematycznego i fizycznego jest to, że dla małych amplitud oscylacji okres drgań jest prawie całkowicie niezależny od amplitudy. Ta właściwość, zwana izochronizmem oscylacyjnym, została odkryta około 1602 roku przez Galileusza, który używał wahadła do mierzenia czasu. Ogólnie rzecz biorąc, wahadło jest oscylatorem anharmonicznym, którego okres i inne parametry zależą od jego amplitudy. Chociaż rozwiązanie ogólnego równania ruchu wahadłowego jest bardzo złożone, założenia upraszczające dla małych amplitud oscylacji można rozwiązać analitycznie.

Rysunek 2.5: Wahadło matematyczne
Źródło: https://edumax.com.pl/pol$_p l_W ahadlo - matematyczne - 5431_1.jpg$

Wahadło matematyczne to pojedynczy punkt poruszający się po kręgu w płaszczyźnie pionowej w jednorodnym polu grawitacyjnym, równanie jego ruchu określa wzór [**?**]:

$$\frac{\partial^2\theta(t)}{\partial t^2} + \frac{g}{\ell}\sin\theta(t) = 0$$

w którym:
$\theta(t)$ - kąt odchylenia wahadła do pionu w chwili t
g - przyspieszenie ziemskie
ℓ - długość liny

Drgania dla niskiej amplitudy Funkcję sinus można przybliżyć jej argumentem, gdy kąt jest odpowiednio mały (wzór Taylora) [**?**]

$$\sin\theta \approx \theta$$

wówczas ogólne równanie ruchu wahadła upraszcza się do postaci:

$$\frac{\partial^2\theta(t)}{\partial t^2} + \frac{g}{\ell}\theta(t) = 0$$

Powyższe równanie jest równaniem drgań harmonicznych. Rozwiązanie określa zależność kąta wahań od czasu i może być określone wzorem [23]:

$$\theta(t) = \theta_0 \sin(\omega t + \varphi)$$

w którym:
θ_0 - amplituda drgań,
$\omega = \sqrt{\frac{g}{\ell}}$ - częstość kołowa drgań
φ - faza początkowa drgań

okres drgań wynosi za to:

$$T = 2\pi\sqrt{\frac{g}{\ell}}$$

wynika z tego, że w przybliżeniu dla małych drgań wahadła okres drgań nie zależy od amplitudy, a tylko i wyłącznie od długości wahadła i siły ciążenia.

Okres drgań o dowolnej amplitudzie

Dla dużych ampiltud wahań okres drgań zależy od amplitudy θ_0 i rośnie wraz z jej wzrostem. Zależność okresu od amplitudy opisuje wzór [23]:

$$T(\theta_0) = 4\sqrt{\frac{\ell}{g}}K(\sin\tfrac{\theta_0}{2})$$

gdzie K jest całką eliptyczną, zupełną i pierwszego rodzaju.

Jego rozwinięciem jest wzór:

$$\begin{aligned} T(\theta_0) &= 2\pi\sqrt{\tfrac{\ell}{g}}\cdot\textstyle\sum_{n=0}^{\infty}[(\frac{(2n)!}{(2^n)\cdot n!})^2\cdot\sin^{2n}(\tfrac{\theta_0}{2})] = \\ &= 2\pi\sqrt{\tfrac{\ell}{g}}(1+(\tfrac{1}{2})^2\sin^2(\tfrac{\theta_0}{2})+(\tfrac{1\cdot3}{2\cdot4})^2\sin^4(\tfrac{\theta_0}{2})+\cdots) \end{aligned}$$

Rozwijając w szereg Maclaurina [25]:

$$T(\theta_0) = 2\pi\sqrt{\tfrac{\ell}{g}}(1+\tfrac{1}{16}\theta_0^2+\tfrac{11}{3072}\theta_0^4+\tfrac{172}{737280}\theta_0^6+\tfrac{22931}{1321205760}\theta_0^8+\cdots)$$

Gdy w wzorze podanym wyżej zignoruje się wyrazy sumy poza pierwszym, równym 1, to wychodzi wzór na okres niewielkich drgań wahadła.

Przybliżona zależność okresu od amplitudy

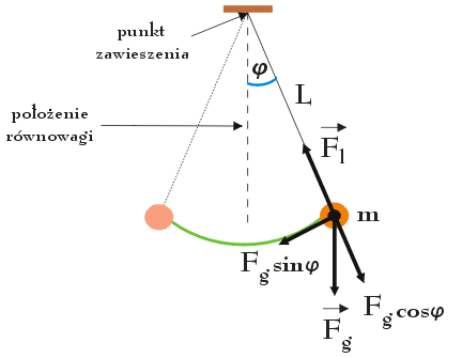
Problem ruchu wahadła można rozwiązać poprzez estymację funkcji sinus do dwóch wyrazów, wówczas równanie ruchu wahadła przyjmuje postać [26]:

$$\tfrac{d^2\theta}{dt^2}+\tfrac{g}{\ell}(\theta-\tfrac{1}{6}\theta^3)=0$$

Rozwiązanie tego równania ma w przybliżeniu postać

$$\theta(t) = \theta_0\cos\omega t+\theta_3\cos(3\omega t)$$

gdzie:
θ_0 – amplituda drgań o częstotliwości ω
$\theta_3 = \frac{1}{3}(\frac{\theta_0}{4})^3$ – amplituda drgań o częstotliwości 3ω
$\omega_0 = \sqrt{\frac{g}{l}}$,
$\omega = \omega_0(1-\frac{\theta_0^2}{16})$.



Rysunek 2.6: Schemat wahadła matematycznego
Źródło: https://efizyka.net.pl/wahadlo-matematyczne-i-fizyczne

Powyższe rozwiązanie wskazuje, iż wahadło matematyczne dla amplitud wystarczająco dużych nie jest oscylatorem harmonicznym, lecz:

- ruch wahadła jest złożeniem dwóch drgań harmonicznych mających częstotliwości ω oraz 3ω i amplitudy odpowiednio równe θ_0 oraz θ_3
- częstotliwość drgań ω zależy od θ_0 (czyli nie występuje izochronizm drgań charakterystyczny dla małych amplitud) i maleje wraz z jej wzrostem
- amplituda θ_3 wyższej harmonicznej zależy w trzeciej potędze od amplitudy θ_0

Z drugiej strony, dla dostatecznie małych wartości θ_0 częstotliwość drgań ω zbliża się do wartości ω_0 zaś amplituda wyższej harmonicznej staje się pomijalnie mała – otrzymuje się drganie harmoniczne o amplitudzie θ_0 i częstotliwości ω_0 która nie jest zależna od amplitudy.

Rozwiązanie ogólnego równania ruchu Dokładne rozwiązanie ruchu wahadła dla dowolnej amplitudy można podać w postaci uwikłanej [27]:

$$dt = \sqrt{\frac{\ell}{2g}} \frac{d\theta}{(\cos\theta - \cos\theta_0)}$$

Wykonując całkowanie w zakresie od 0 do θ przy stałym kącie maksymalnego wychylenia θ_0, otrzymuje się

$$t(\theta) = \sqrt{\frac{\ell}{2g}} \int_0^{\theta} \frac{d\theta}{(\cos\theta - \cos\theta_0)}$$

Reakcja więzów Z definicji wahadła prostego, jego ruch jest ograniczony do ruchu po okręgu przez więzy. Suma składowych siły działających na ciało prostopadłe do trajektorii ruchu to siła dośrodkowa, której wartość wyraża wzór [28]

$F_r = -\frac{mv^2}{\ell}$,

przy czym znak "minus"jest ze względu na działanie siły w stronę środka okręgu, przeciwnie do zwrotu. Zależność tej siły od kąta θ można wyznaczyć z zasady zachowania energii i prędkości wahadła.

$v^2 = 2g\ell(\cos\theta - \cos\theta_0)$

θ_0 - kąt maksymalnego wychylenia wahadła.

Siłę naprężenia nici określa następujący wzór [28]: $T(\theta) = F_r - mg\cos\theta = -mg(3\cos\theta - 2\cos\theta_0)$

Przyjęty tutaj układ współrzędnych jest zgodny z więzami, więc nie musimy określać sił reakcji więzów, aby opisać ruch wahadła. Wyznaczenie tej siły jest konieczne, jeśli siła jest opisana w kartezjańskim układzie współrzędnych. Wybór układu współrzędnych podlegającego ograniczeniom jest podstawą do sformułowania mechaniki klasycznej w kategoriach mechaniki Lagrange'a.

2.2.2 Oscylator harmoniczny

Ruch harmoniczny to ruch oscylacyjny, w którym na ciało działa siła proporcjonalna do jego odchylenia od położenia równowagi, zawsze skierowana w kierunku jego punktu równowagi.

Oscylator harmoniczny to system oscylacyjny, który wykonuje ruch harmoniczny.Takie badanie ruchów można zredukować do modelu mechanicznego z jednym stopniem swobody. W takim układzie występuje siła sprężystości F(r) proporcjonalna do przemieszczenia r układu od położenia równowagi: $F(r) = -kr$, gdzie k jest tak zwaną stałą sprężystości.

Energia potencjalna oscylatora harmonicznego zależy od kwadratu jego odchylenia r względem jego położenia równowagi.

$V(r) = \frac{k}{2}r^2$

Wiele układów fizycznych można przedstawić przez model oscylatora w przybliżeniu, gdy układ oscyluje z małą siłą (tj. małą amplitudą) w pobliżu swojego położenia równowagi. Rozszerzając potencjał w szereg Taylora wokół minimum, aproksymacja składnika kwadratowego jest wystarczająco dokładna (jeśli składnik jest niezerowy). W praktyce oznacza to, że wiele rzeczywistych problemów można zredukować do pojęć z oscylatorami harmonicznymi.

Problem oscylatorów harmonicznych został pomyślnie rozwiązany zarówno w mechanice klasycznej, jak i kwantowej.

Oscylacje, które nie są harmonicznymi (to znaczy potencjałów, których nie można opisać ani przybliżyć przez żadną inną niż zależność kwadratową) nazywane są oscylacjami anharmonicznymi. Korekcja ruchu harmonicznego wynikającego z zależności potencjałów innych niż drugiego rzędu nazywana jest korekcją anharmoniczną.

Prosty oscylator harmoniczny to taki, który nie jest ani dodatkowo napędzany, ani w żaden sposób tłumiony. Składa się z masy m, na którą działa pojedyncza siła F, ciągnąca masę do punktu $x = 0$, w zależności tylko od położenia masy x i stałej k. Równowaga sił (drugie prawo Newtona) wynosi:

$$F = ma = m\frac{d^2x}{dt^2} = m\ddot{x} = -kx.$$

Rozwiązując to równanie różniczkowe, okazuje się, że ruch jest opisany funkcją

$$x(t) = A\cos(\omega t + \varphi),$$

gdzie $\omega = \sqrt{\frac{k}{m}}$.

Ruch ten jest okresowy i powtarza się ze stałą sinusoidalną amplitudą A. Oprócz amplitudy, ruch prostego oscylatora harmonicznego ma swój okres $T = 2\pi\omega$,to znaczy czasem trwania lub częstotliwością pojedynczej oscylacji $f = 1/T$ - liczba cykli w jednostce czasu. Pozycja w dowolnym momencie t zależy również od fazy ϕ, która określa punkt początkowy na sinusoidzie. Okres i częstotliwość zależą od masy m i wielkości stałej siły k, natomiast amplituda i faza zależą od położenia początkowego i prędkości.

Prędkość i przyspieszenie prostego oscylatora harmonicznych drgają z tą samą częstotliwością co położenie, ale z przesuniętymi fazami. Prędkość jest maksymalna przy zerowym przemieszczeniu, natomiast przyspieszenie jest w przeciwnym kierunku do przemieszczenia.

Energia potencjalna zmagazynowana w pozycji x prostego oscylatora harmonicznego jest równa:

$$U = \tfrac{1}{2}kx^2.$$

2.2.3 Sprężyna

Prawo Hooke'a:

Większość sprężyn jest zgodna z prawem Hooke'a. Prawo Hooke'a mówi, że siła odpychająca sprężyny jest liniowo proporcjonalna do jej odległości od jej długości w stanie równowagi, chyba że jest ona rozciągana lub ściskana poza granicę sprężystości:

$$F = -kx,$$

gdzie:

x to wektor przemieszczenia - odległość i kierunek odkształcenia sprężyny, jej długość w równowadze

F to wektor siły wypadkowej - wielkość i kierunek siły przywracającej wywieranej przez sprężynę

k jest modułem, stałą sprężyny lub stałą siły sprężyny, w zależności od materiału i konstrukcji

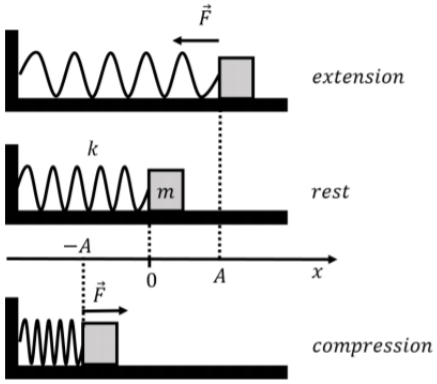
sprężyny. Znak ujemny oznacza, że siła wywierana przez sprężynę jest w kierunku przeciwnym do ugięcia.

Siła jest równa masie m pomnożonej przez przyspieszenie a, więc równanie siły dla sprężyny zgodnie z prawem Hooke'a to:

$$F = ma \Rightarrow -kx = ma$$

Przemieszczenie w funkcji czasu, x. Czas, jaki upływa między szczytami, nazywa się okresem. Masa sprężyny jest pomijalna, ponieważ jest mała w stosunku do dołączonej masy. Przyspieszenie jest drugą pochodną x względem czasu, więc

$$-kx = m\frac{d^2x}{dt^2}$$



Rysunek 2.7: Rozciąganie sprężyny
Źródło: [29]

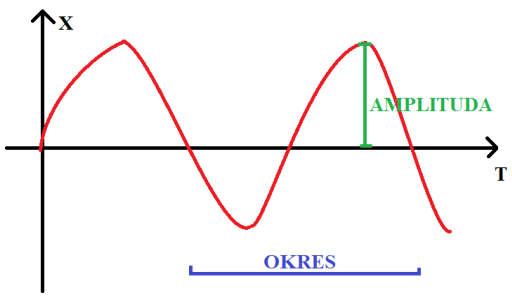
Jest to równanie różniczkowe liniowe drugiego rzędu x przemieszczenia w funkcji czasu

$$\frac{d^2x}{dt^2} + \frac{k}{m}x = 0$$

Jego rozwiązaniem jest suma sinusa i cosinusa:

$$x(t) = A\sin\left(t\sqrt{\frac{k}{m}}\right) + B\cos\left(t\sqrt{\frac{k}{m}}\right).$$

A i B to dowolne stałe, które można znaleźć biorąc pod uwagę początkowe przemieszczenie i prędkość masy.



Rysunek 2.8: Amplituda i okres
Źródło: Rysunek własny

W prostym ruchu harmonicznym układu sprężyna-masa energia zmienia się między energią kinetyczną a potencjalną, ale energia całkowita układu pozostaje taka sama. Sprężyna spełniająca prawo Hooke'a o stałej sprężystości k będzie miała całkowitą energię układu E równą: [29]

$$E = \left(\frac{1}{2}\right) kA^2$$

Tutaj A jest amplitudą ruchu falowego, który powstaje w wyniku pulsującego zachowania się sprężyny.

Stała k sprężyny i jej przemieszczenie x mogą wyznaczyć energię potencjalną U takiego układu [29]:

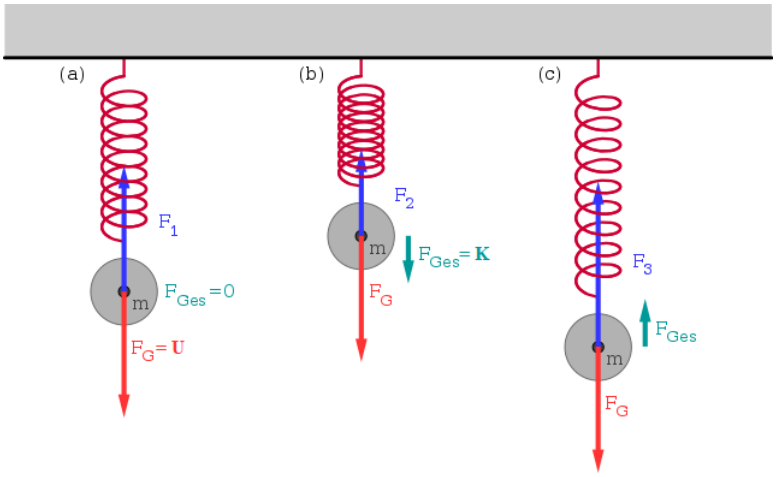
$$U = \left(\frac{1}{2}\right) kx^2$$

Prędkość v z jaką obiekt oscyluje i masa dołączonego obiektu m pozwalają na znalezienie energii kinetycznej K obiektu w prostym ruchu harmonicznym [29]:

$$K = \left(\frac{1}{2}\right) mv^2$$

Taki układ ma zawsze zachowaną energię, ponieważ nie ma w nim żadnych strat energii [29]:

$$E = K + U$$



Rysunek 2.9: Masa zawieszona na sprężynie i siły działające na nią
Źródło: https://upload.wikimedia.org/wikipedia/commons/0/0c/Vertical-mass-on-spring.svg

Częstotliwość kątowa ω obiektu, która może być podana w radianach na sekundę w ruchu harmonicznym prostym, można znaleźć za pomocą masy oscylującego obiektu m i stałej sprężystości k [30]:

$$\omega = \sqrt{\frac{k}{m}}$$

W takim ruchu harmonicznym, układ sprężyna-masa wykonuje jeden cykl w danej jednostce czasu, czyli okresie T: [31]

$$T = \frac{2\pi}{\omega} = 2\pi\sqrt{\frac{m}{k}}$$

Odwrotność okresu umożliwi nam znalezienie liczby oscylacji w czasie, czyli częstotliwości f: [29]

$$f = \frac{1}{T} = \frac{\omega}{2\pi} = \frac{1}{2\pi}\sqrt{\frac{k}{m}}$$

Teoria

W fizyce klasycznej sprężyna może być postrzegana jako urządzenie przechowujące energię potencjalną, zwłaszcza energię potencjalną sprężystości, poprzez rozciąganie wiązań między atomami w materiałach elastycznych.

Prawo sprężystości Hooke'a mówi, że wydłużenie (długość po rozciągnięciu minus długość po w stanie zrelaksowania) elastycznego pręta jest liniowo proporcjonalne do napięcia, siły użytej do jego rozciągnięcia. Podobnie skurcz (ujemne odkształcenie) jest proporcjonalne do ściskania (ujemne napięcie).

To prawo ma zastosowanie tylko wtedy, gdy odkształcenie (dodatnie lub ujemne) jest niewielkie w porównaniu z całkowitą długością pręta. Kiedy następuje odkształcenie poza granicę sprężystości, wiązania atomowe mogą zostać zerwane lub przegrupowane, powodując pęknięcie, spięcie lub trwałe odkształcenie sprężyny. Wiele materiałów nie ma dobrze zdefiniowanych granic sprężystości i prawo Hooke'a nie może być sensownie zastosowane. Co więcej, w przypadku materiałów hiperelastycznych liniowa zależność między siłą a przemieszczeniem ma znaczenie tylko w obszarach o niskim odkształceniu.

Prawo Hooke'a jest matematyczną konsekwencją faktu, że energia potencjalna pręta jest najmniejsza, gdy znajduje się on w swojej swobodnej długości. Jak widać, badając szereg Taylora, gładka funkcja jednej zmiennej zbliża się do funkcji kwadratowej, gdy jest testowana wystarczająco blisko jej punktu minimalnego. Dlatego siła, pochodna energii po przemieszczeniu, jest aproksymacją funkcji liniowej.

Siła w pełni ściśniętej sprężyny:

$$F_{max} = \frac{Ed^4(L-nd)}{16(1+\nu)(D-d)^3 n}$$

gdzie

E - Moduł Younga d - Średnica drutu sprężyny L - Długość swobodna sprężyny n - Aktywna liczba zwojów ν- Współczynnik Poissona D - Średnica zewnętrzna sprężyny

2.3 Generatory liczb losowych

Generator liczb losowych (RNG) — program komputerowy lub obwód elektroniczny, który generuje stałą, ergodyczną losową sekwencję elementów binarnych, zwykle zorganizowaną jako ciąg liczb losowych.

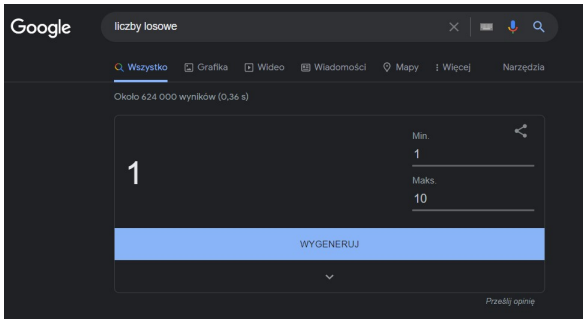
Wyniki generatora liczb losowych są zazwyczaj równomiernie rozłożonymi liczbami z zakresu [0,1). Z generatora o takim rozkładzie możemy uzyskać generator o dowolnym innym rozkładzie, obliczając odwrotność dystrybuanty pożądanego rozkładu z wyników pierwszego generatora. [32]

Istnieją dwa typy generatorów, w zależności od sposobu generowania liczb losowych:

- generator sprzętowy (TRNG), który działa na zasadzie odwzorowania właściwości i parametrów fizycznych procesów stochastycznych, głównie szumu elektrycznego. Takie generatory same nie wytwarzają liczb, a jedynie stany, które później są interpretowane jako liczby.
- Generatory programowe (PRNG) - działają na zasadzie deterministycznego obliczania ciągów liczbowych, które przypominają losowe

Główną zaletą generatora sprzętowego, która jest szczególnie ważna w kryptografii, jest nieprzewidywalność i nieodtwarzalność generowanych sekwencji. Wynika to z unikalnej implementacji fizycznych procesów stochastycznych w określonym czasie. [32]

Liczby generowane przez generatory programowe nazywane są liczbami pseudolosowymi, ponieważ w rzeczywistości nie są losowe, ale są wynikiem obliczeń matematycznych. Główną zaletą generatora liczb pseudolosowych jest jego szybkość, często z lepszymi właściwościami statystycznymi niż generatory sprzętowe. Należy jednak pamiętać, że kontrolując lub znając wartość podaną na wejście generatora i jego stan wewnętrzny, łatwo jest przewidzieć zwracaną liczbę.



Rysunek 2.10: generator liczb losowych google
Źródło: google i zdjęcie własne

Generator liczb pseudolosowych

Generator liczb pseudolosowych lub inaczej PRNG – Program lub podprogram, który w oparciu o niewielką ilość informacji (ziaren, nasion) generuje deterministyczny ciąg bitów, który w określonych warunkach nie różni się od ciągów, które pochodzą z naprawdę przypadkowych źródeł.

Generator liczb losowych nie wytwarza prawdziwego łańcucha losowego - generator z zaczątkiem nasiona, które może przyjmować różne wartości k, może wytworzyć co najwyżej k odrębnych sekwencji. Ponadto, ponieważ rozmiar zmiennej reprezentującej stan wewnętrzny generatora jest ograniczony, po pewnym czasie bez dostarczania nowych danych z zewnątrz, powinien zakończyć cykl i zacząć generować te same wartości. Teoretyczne granice długości cyklu wynoszą 2^n, gdzie n to liczba bitów utrzymujących stan wewnętrzny.

Algorytmy probabilistyczne (takie jak Monte Carlo) wymagają tylko jednego źródła wartości, które jest zbliżone do liczb prawdziwie losowych, ale jakość liczb losowych może mieć znaczenie dla dokładności obliczeń. Dlatego każdy nowy generator do obliczeń numerycznych powinien być sprawdzany pod kątem jego właściwości statystycznych. Za pomocą jednego ze sprawdzonych generatorów można bezpośrednio obliczyć długość cyklu, a inne właściwości, takie jak równomierność rozkładu, są najczęściej znane. [33]

Potencjalne problemy

W praktyce, wyjście wielu popularnych PRNG wykazuje artefakty, które powodują, że nie przechodzą one testów wykrywania wzorców statystycznych. Obejmują one:

- Krótsze od oczekiwanych okresy dla niektórych stanów nasion (takie stany nasion można w tym kontekście nazwać ślabymi");
- Brak równomierności rozkładu dla dużych ilości generowanych liczb; Korelacja kolejnych wartości;
- Słaby rozkład wymiarowy sekwencji wyjściowej;

- Odległości pomiędzy miejscami występowania określonych wartości są rozłożone inaczej niż w rozkładzie sekwencji losowej.

Błędne wyniki, które pokazywane są przez wadliwy PRNG mogą wahać się od niezauważonych (i niejasnych) do bardzo oczywistych. Jednym z przykładów jest algorytm liczb losowych RANDU, który od dziesięcioleci jest używany w komputerach typu mainframe. Miał on poważną wadę, która pozostawała niewykryta przez bardzo długi czas.

Wyniki w wielu pracach badawczych z różnych dziedzin sprzed XXI wieku, które opierały się na losowym próbkowaniu lub symulacjach Monte Carlo, lub w inny sposób opierały się na PRNG, były znacznie mniej wiarygodne, ponieważ wykorzystywały PRNG niskiej jakości. [33]

Przykładowy prosty generator

Poniżej znajduje się bardzo prosty przykład PRNG napisany w JavaScript. Wykorzystuje on sekwencję mnożenia do uzyskania pozornie losowej wartości, która jest następnie normalizowana do zakresu od 0 do 1. W tym przykładzie 15485863 jest 1 000 000-tą liczbą pierwszą, a 2038074743 100 000 000-tą. [34]

```
1   class PRNG
2   {
3       seed = 0;
4
5       Seed(seed)
6       {
7           this.seed = seed;
8           let a = this.seed * 15485863;
9           return (a * a * a % 2038074743) / 2038074743; //Will return in range 0
                to 1 if seed >= 0 and -1 to 0 if seed < 0.
10      }
11
12      Next()
13      {
14          this.seed++;
15          let a = this.seed * 15485863;
16          return (a * a * a % 2038074743) / 2038074743;
17      }
18  }
```

Przykład zwraca bardzo podobne wyniki do funkcji Math.random() w JavaScript. [34]

2.3.1 Mersenne Twister

2.3.2 Holiana SNWS

2.3.3 RAN2

2.3.4 Hoover

2.4 Testy generatorów

Jak już zostało wcześniej wspomniane testy generatorów są ściśle istotną sprawą przy doborze odpowiedniego generatora do odpowiedniego zadania.

Test losowości (lub test na losowość), w analizie danych, to test używany do analizy rozkładu zbioru danych w celu ustalenia, czy można go uznać za losowy (nie ma wzorca). Modelowanie

probabilistyczne, podobnie jak w niektóre symulacje komputerowe, umożliwia weryfikację oczekiwanej losowości potencjalnych danych wejściowych za pomocą formalnych testów losowości w celu wykazania, że dane mogą być wykorzystane w przebiegu symulacji. W niektórych przypadkach dane pokazują oczywiste nielosowe wzorce, takie jak Tzw. "przebiegi danych"(np. losowe wartości oczekiwane 0-9, ale otrzymanie „3 2 1 0 3 2 1...", które nie przekracza 3). Jeśli wybrany zestaw danych nie przejdzie testu, można zmienić parametry lub użyć innych losowych danych, takich które pozwolą przejść test losowości.

Tło historyczne

Problem losowości jest ważnym tematem filozoficznym i teoretycznym. Testy losowości można wykorzystać do określenia, czy zbiór danych wykazuje dostrzegalne wzorce, które sugerują, że proces, który go wygenerował, jest znacząco nielosowy. W większości przypadków praktyczna analiza statystyczna polega w mniejszym stopniu na testowaniu losowości, a bardziej na znajdowaniu prawidłowości w danych. Wiele z obecnie używanych „generatorów liczb losowych" to w rzeczywistości generatory liczb pseudolosowych, ponieważ są one definiowane przez algorytmy. Generowane przez nie sekwencje nazywane są sekwencjami pseudolosowymi. Generatory te nie zawsze generują wystarczająco losowe sekwencje, ale zamiast tego mogą generować sekwencje z wzorcami. Na przykład niesławna metoda RANDU drastycznie zawodzi w wielu testach losowości, w tym w testach spektralnych.

Stephen Wolfram użył testu losowego „reguły 30" do przetestowania możliwości generowania liczb losowych, [35] okazało się, że ma efektywny rozmiar klucza znacznie mniejszy niż jego rzeczywisty rozmiar [36] i osiąga słabe wyniki w teście chi-squared. [37] Używanie źle zaprojektowanego generatora liczb losowych może podważyć ważność eksperymentu poprzez naruszenie założeń statystycznych. Chociaż istnieją powszechnie stosowane metody testowania statystycznego, takie jak standardy NIST, Yongge Wang wykazał, że standardy te są niewystarczające. Ponadto Yongge Wang [38] opracował metodę testową opartą na odległości statystycznej i prawie logarytmu ziterowanego. Korzystając z tej techniki, Yongge Wang i Tony Nicol [39] byli w stanie wkyryć luki w powszechnie używanych generatorach liczb pseudolosowych.

Specyficzne testy losowe

W praktyce stosuje się stosunkowo niewiele różnych typów generatorów liczb (pseudo)losowych, takich jak:

Linear congruential generator i Linear-feedback shift register Uogólniony generator Fibonacciego Generatory kryptograficzne Kwadratowy generator kongruencyjny Pseudolosowe sekwencje binarne

Istnieje wiele praktycznych miar losowości dla ciągu binarnego. Należą do nich pomiary oparte na testach statystycznych, przekształceniach, złożoności lub ich kombinacji. Popularnym i szeroko stosowanym zestawem testów jest „Diehard Battery of Tests" wprowadzony przez Marsaglia, rozszerzony przez L'Ecuyer i Simarda do pakietu „TestU01". Zastosowanie transformaty Hadamarda do pomiaru losowości zostało zaproponowane przez S. Kaka i rozwinięte przez Phillipsa, Yuena, Hopkinsa, Beta i Dai, Munda oraz Marsaglia i Zamana. [40]

Niektóre z tych testów złożoności liniowej zapewniają spektralne miary losowości. Mówi się, że T. Beth i Z-D. Dai wykazali, że złożoność Kołmogorowa i złożoność liniowa są praktycznie takie same, [41] ale Y. Wang później obalił ich twierdzenie. [42] Niemniej Wang wykazał również, że w

przypadku ciągów losowych Martina-Löfa złożoność Kołmogorowa jest zasadniczo taka sama jak złożoność liniowa.

Te testy umożliwiają porównanie losowości ciągów. Ze względów probabilistycznych wszystkie ciągi o danej długości mają tę samą losowość. Jednak różne ciągi mają różną złożoność Kołmogorowa. Na przykład, biorąc pod uwagę następujące dwa ciągi:

Ciąg 1: 0101010101010101010101010101010101010101010101010101010101010101

Ciąg 2: 1100100001100001110111101110110011111010010000100101011110010110

Ciąg 1 umożliwia krótki opis językowy: "32 powtórzenia '01'". Ciąg 2 nie ma oczywistego prostego opisu poza zapisaniem krok po kroku samego łańcucha. Ma on długość 64 znaków i nie ma stosunkowo wydajnej reprezentacji funkcji bazowej. Korzystając z liniowego testu spektralnego Hadamarda, można stwierdzić, że pierwsza z tych sekwencji jest znacznie mniej losowa niż druga.

2.4.1 Diehard

Diehard nie jest pierwszym zestawem testów zaprojektowanych do analizy jakości generatorów liczb losowych, ale dzięki wydaniu na początku ery Internetu jest znacznie bardziej popularny niż którykolwiek z jego poprzedników. Profesor Marsaglia był wybitnym architektem generatorów liczb pseudolosowych i ich analizy przez kilka dziesięcioleci, zanim ta bateria testowa została opublikowana w 1995 roku. Późniejsze badania ujawniły pewne niedociągnięcia tych testów, ale nadal działają dobrze, zwłaszcza podczas testowania małych próbek (10-100 megabajtów). Jednakże testy te są odtwarzane za pomocą Die Harder, który ma znacznie bardziej przejrzysty interfejs użytkownika, dlatego zalecane jest używanie tego narzędzia do uruchamiania testów. [43]

Odstępy urodzeniowe
Wybierz losowe punkty na dużym interwale. Odstępy między punktami powinny mieć rozkład asymptotycznie wykładniczy. Nazwa oparta jest na paradoksie urodzin. [44]

Nakładające się permutacje
Przeanalizuj ciągi pięciu kolejnych liczb losowych. 120 możliwych uporządkowań powinno wystąpić ze statystycznie równym prawdopodobieństwem. [44]

Szeregi macierzy
Wybierz pewną liczbę bitów z pewnej liczby losowej, aby utworzyć macierz nad 0,1, a następnie wyznacz rangę macierzy. Policzyć rangi. [44]

"Monkey tests"
Traktuj ciągi pewnej liczby bitów jako słowa". Policzyć pokrywające się słowa w strumieniu. Liczba słów", które się nie pojawiają, powinna mieć znany rozkład. Nazwa wynika z twierdzenia o nieskończonej małpie. [44]

Liczenie jedynek
Policz bity 1 w każdym z kolejnych lub wybranych bajtów. Przekształć zliczenia na litery"i policz wystąpienia pięcioliterowych słów". [44]

Test parkingowy
Losowo rozmieść jednostkowe kółka w kwadracie 100×100. Kółko jest pomyślnie zaparkowane, jeśli nie zachodzi na istniejące już pomyślnie zaparkowane. Po 12 000 próbach liczba pomyślnie zaparkowanych kół powinna mieć rozkład normalny. [44]

Test minimalnej odległości
Losowo umieść 8000 punktów w kwadracie 10000×10000, a następnie znajdź minimalną odległość między parami. Kwadrat tej odległości powinien mieć rozkład wykładniczy z pewną średnią. [44]

Test losowych kul
Wybierz losowo 4000 punktów w sześcianie o krawędzi 1000. Na każdym punkcie wyśrodkuj sferę, której promień jest minimalną odległością od innego punktu. Najmniejsza objętość kuli powinna mieć rozkład wykładniczy z pewną średnią. [44]

Test ściskania
Pomnóż 231 przez losowe liczby typu "floatńa (0,1), aż osiągniesz 1. Powtórz to 100000 razy. Liczba cyfr "float"potrzebna do osiągnięcia 1 powinna mieć pewien rozkład. [44]

Test nakładających się sum
Wygeneruj długi ciąg losowych liczb typu "floatńa (0,1). Dodaj sekwencje 100 kolejnych "float". Sumy powinny mieć rozkład normalny z charakterystyczną średnią i wariancją. [44]

Test przebiegów
Wygeneruj długą sekwencję losowych zmiennych na (0,1). Policz przebiegi rosnące i malejące. Zliczenia powinny być zgodne z pewnym rozkładem. [44]

Test kości
Rozegraj 200000 gier w kości, licząc wygrane i liczbę rzutów na grę. Każde zliczenie powinno podążać za pewnym rozkładem. [44]

Większość testów w Diehard zwraca wartość p. Jeśli plik wejściowy zawiera naprawdę niezależne losowe bity, ta wartość powinna być konsekwentnie równa $[0, 1]$. Te wartości p uzyskuje się przez $p = F(X)$, gdzie F jest założonym rozkładem zmiennej losowej X, zwykle normalnym. Zakłada się jednak, że F jest tylko asymptotycznym przybliżeniem, ostatecznie dającym najgorsze dopasowanie. Dlatego sporadyczne wartości p bliskie 0 lub 1, takie jak 0,0012 lub 0,9983, nie powinny dziwić. Jeśli strumień bitów naprawdę poważnie zawiedzie, *ps* może przyjmować "0 lub 1"do 6 lub więcej miejsc po przecinku. Przy tak wielu testach nie jest nieprawdopodobne, że $p < 0,025 lub p > 0,975$ oznacza, że RNG „nie przeszedł testu przy 0,05". Wśród setek zdarzeń generowanych przez program "DIEHARD", wielu takich zdarzeń *ps* można się spodziewać, nawet mając doskonały generator liczb losowych. [43]

2.4.2 U01

"TestU01"to biblioteka oprogramowania zaimplementowana w języku ANSI C, która zapewnia zbiór narzędzi do empirycznego testowania losowości generatorów liczb losowych (RNG). [45] Biblioteka została po raz pierwszy pokazana w 2007 roku przez Pierre'a L'Ecuyera i Richarda Simarda na Uniwersytecie w Montrealu. [46]

Ta biblioteka implementuje kilka typów generatorów liczb losowych, w tym te proponowane w literaturze oraz te znajdujące się w szeroko stosowanym oprogramowaniu. Dostarcza wspólnych implementacji testów statystycznych dla klasycznych generatorów liczb losowych oraz kilku innych proponowanych w literaturze i kilku oryginalnych. Testy te można zastosować do wstępnie zdefiniowanych generatorów w bibliotece, generatorów niestandardowych i strumieni liczb losowych przechowywanych w plikach. Dostępne są również specjalne zestawy testów dla $[0, 1]$ jednolitych sekwencji liczb losowych lub sekwencji bitów. Dostępne są również podstawowe narzędzia do wykreślania wektorów punktowych generowanych przez generator.

Funkcje

TestU01 udostępnia cztery grupy modułów do analizy RNG:

Implementacja (zaprogramowanych) RNG; Wdrażanie specyficznych testów statystycznych; Wdrażanie baterii testów statystycznych; Stosowanie testów do całych rodzin RNG.

Gdy dany test jest stosowany do próbki o rozmiarze n wygenerowanej przez RNG, wartość p testu zwykle pozostaje ważna, gdy rozmiar próbki wzrasta, aż rozmiar próbki osiągnie $n0$. Następnie wartość p przesuwa się wykładniczo do 0 lub 1. Moduł 4 umożliwia zbadanie wzajemnych zależności między konkretnymi testami a strukturą zbiorów punktów generowanych przez określone rodziny RNG. Korzystając z tej techniki, możemy określić, jak duża musi być wielkość próbki w funkcji wielkości okresu generatora, zanim generator zacznie systematycznie nie przechodzić testów.

"TESTU01źapewnia wiele serii testów, w tym Śmall Crush"(składający się z 10 testów), Ćrush"(96 testów) i "Big Crush"(106 testów). Konkretne testy stosowane dla każdej baterii są szczegółowo opisane w podręczniku użytkownika. [45]

Ograniczenia TestU01 akceptuje tylko 32-bitowe dane wejściowe i interpretuje je jako wartości z zakresu [0, 1]. W rezultacie najbardziej znaczące bity są bardziej podatne na błędy niż najmniej znaczące. Ważne jest, aby przetestować uniwersalne generatory w postaci odwróconej bitowo, aby zweryfikować przydatność do zastosowań, które wykorzystują bity niskiego rzędu. [47]

Generator, który wytwarza 64 bity na wyjściu, również wymaga oddzielnych testów dla wysokich i niskich połów. [48]

2.4.3 Die Harder

Literatura

[1] K. W. Wojciechowski. Pseudorandom number generators based on the weyl sequence. *Computational Methods in Science and Technology*, Vol. 5:81–85, 1999.

[2] K.W. Wojciechowski. Monte carlo simulations of model particles forming phases of negative poisson ratio. In Bogdan Idzikowski, Peter Švec, and Marcel Miglierini, editors, *Properties and Applications of Nanocrystalline Alloys from Amorphous Precursors*, pages 241–252, Dordrecht, 2005. Springer Netherlands.

[3] Jakub Narojczyk and Krzysztof W. Wojciechowski. Computer simulation of poisson's ratio of soft polydisperse discs at zero temperature. 2015.

[4] K. V. Tretiakov and K. W. Wojciechowski. Efficient monte carlo simulations using a shuffled nested weyl sequence random number generator. *Phys. Rev. E*, 60:7626–7628, Dec 1999.

[5] K. V. Tretiakov and K. W. Wojciechowski. Monte carlo simulation of two-dimensional hard body systems with extreme values of the poisson's ratio. *physica status solidi (b)*, 242(3):730–741, 2005.

[6] Dirk P. Kroese, Tim J. Brereton, Thomas Taimre, and Zdravko I. Botev. Why the monte carlo method is so important today. *Wiley Interdisciplinary Reviews: Computational Statistics*, 6, 2014.

[7] N. Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, A. H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[8] Franklin Mendivil Ronald W. Shonkwiler. *Explorations in Monte Carlo Methods*. Springer New York, NY, Reading, MA, USA, 2009.

[9] Americo Cunha, Rafael Nasser, Rubens Sampaio, Hélio Lopes, and Karin Breitman. Uncertainty quantification through the monte carlo method in a cloud computing setting. *Computer Physics Communications*, 185(5):1355–1363, 2014.

[10] Jianming Wei and Frank Einar Kruis. A gpu-based parallelized monte-carlo method for particle coagulation using an acceptance–rejection strategy. *Chemical Engineering Science*, 104:451–459, 2013.

[11] Nick Metropolis. The beginning of the monte carlo method.

[12] Roger Eckhardt. Stan ulam, john von neumann, and the monte carlo method. *Los Alamos Science*, 15:131–136, 1987.

[13] Dobriyan Benov, Metodi Mazhdrakov, and Nikolai Valkanov. *The Monte Carlo Method. Engineering Applications*. 11 2018.

[14] Thomas Haigh, Mark Priestley, and Crispin Rope. Los alamos bets on eniac: Nuclear monte carlo simulations, 1947-1948. *IEEE Annals of the History of Computing*, 36(3):42–63, 2014.

[15] Pierre Del Moral and Julian Tugaut. Uniform propagation of chaos for a class of nonlinear diffusions. *Stochastic Analysis and Applications*, 37, 03 2013.

[16] Henry McKean. A class of markov processes associated with nonlinear parabolic equations. *Proceedings of the National Academy of Sciences of the United States of America*, 56:1907–11, 01 1967.

[17] A. M. TURING. I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX(236):433–460, 10 1950.

[18] A.F.M. Smith N.J. Gordon, D.J. Salmond. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140:107–113(6), April 1993.

[19] Shlomo Sawilowsky. Invited debate: Target article you think you've got trivials? *Journal of Modern Applied Statistical Methods Copyright ©*, 200:218–225, 01 2003.

[20] Mohammad Hasan Shojaeefard, Abolfazl Khalkhali, and Sadegh Yarmohammadisatri. An efficient sensitivity analysis method for modified geometry of macpherson suspension based on pearson correlation coefficient. *Vehicle System Dynamics*, 55(6):827–852, 2017.

[21] Matthew Route. Radio-flaring ultracool dwarf population synthesis. *The Astrophysical Journal*, 845(1):66, aug 2017.

[22] Stopnie swobody encyklopedia pwn. https://encyklopedia.pwn.pl/haslo/;3979995. Accessed: 2022-08-23.

[23] W Krolikowski and W Rubinowicz. *Mechanika teoretyczna*. Wydawnictwo Naukowe PWN, 2012.

[24] David Halliday, Robert Resnick, and Jearl Walker. *PODSTAWY FIZYKI. TOM 1*. Wydawnictwo Naukowe PWN, 2015.

[25] Gaetan Kerschen, Douglas Adams, and Alex Carrella. *Topics in Nonlinear Dynamics, Volume 1: Proceedings of the 31st IMAC, A Conference on Structural Dynamics, 2013*. 01 2013.

[26] F. G. Major. *Oscillations and Fourier Analysis*, pages 23–43. Springer New York, New York, NY, 2007.

[27] Lew D Landau and Jewgienij M Lifszyc. *Fizyka statystyczna Część 1*. 2011.

[28] Andrzej Kajetan Wróblewski and Janusz Andrzej Zakrzewski. *Wstęp do fizyki tom I*. 1984.

[29] Libretexts. 13.1: The motion of a spring-mass system, Nov 2020.

[30]

[31] Simple harmonic motion.

[32] Ryszard Wieczorkowski, Robert i Zieliński. *Komputerowe generatory liczb losowych*. Wydawnictwa Naukowo-Techniczne PWN, 1997.

[33] James E. Gentle. *Random Number Generation and Monte Carlo Methods*. Springer New York, NY, 2003.

[34] Pseudorandom number generator, Aug 2022.

[35] Stephen Wolfram. *A new kind of science*. Wolfram Media, 2019.

[36] Willi Meier and Othmar Staffelbach. Analysis of pseudo random sequences generated by cellular automata. In Donald W. Davies, editor, *Advances in Cryptology — EUROCRYPT '91*, pages 186–199, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

[37] Moshe Sipper and Marco Tomassini. Generating parallel random number generators by cellular programming. *International Journal of Modern Physics C*, 07, 10 1996.

[38] Yongge Wang. On the design of lil tests for (pseudo) random generators and some experimental results. 01 2014.

[39] Yongge Wang and Tony Nicol. Statistical properties of pseudo random sequences and experiments with php and debian openssl. pages 454–471, 09 2014.

[40]

[41] Thomas Beth and Zong-Duo Dai. On the complexity of pseudo-random sequences - or: If you can describe a sequence it can't be random. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology — EUROCRYPT '89*, pages 533–543, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

[42] Yongge Wang. Linear complexity versus pseudorandomness: On beth and dai's result. In *Advances in Cryptology - ASIACRYPT '99, International Conference on the Theory and Applications of Cryptology and Information Security, Singapore, November 14-18, 1999, Proceedings*, volume 1716 of *Lecture Notes in Computer Science*, pages 288–298. Springer, 1999.

[43] The tests - a study of entropy.

[44] Diehard tests, Jan 2022.

[45] Testu01.

[46] Pierre L'Ecuyer and Richard Simard. A software library in ansi c for empirical testing of random number generators. *ACM Transactions on Mathematical Software - TOMS*, 01 2007.

[47] Sebastiano Vigna. An experimental exploration of marsaglia's xorshift generators, scrambled. *ACM Trans. Math. Softw.*, 42(4), jun 2016.

[48] Melissa E. O'Neill. Pcg : A family of simple fast space-efficient statistically good algorithms for random number generation. 2014.

Dodatek A

Kody generatorów liczb losowych

Dodatek B

Kody programów testujących

Instytut Informatyki, Wydział Informatyki i Telekomunikacji
Politechnika Poznańska

Skład przy użyciu systemu LaTeX na platformie Overleaf.