

Zastosowanie pakietu Geant4 w fizyce jądrowej

Aleksandra Fijałkowska

Wydział Fizyki, Uniwersytet Warszawski

aleksandra.fijalkowska@fuw.edu.pl

11 października 2018

Miejsce spotkań: czwartek 12.15 – 15.00 (??), sala 2.94

Przedmiot obejmuje 30 godzin rozłożonych w 3-godzinnych blokach, ostatnie zajęcia odbędą się przed Świętami Bożego Narodzenia.

Konsultacje: pokój 2.42

Literatura:

- ▶ Geant4 Book For Application Developers – Podstawowy podręcznik do pakietu Geant4
- ▶ Geant4 Installation Guide – Instrukcja instalacji
- ▶ Physics Reference Manual – Procesy fizyczne i ich modele wykorzystane w kodzie Geant4
- ▶ Bruce Eckel, *Thinking in C++*, Helion, 2000
- ▶ Czasem warto spojrzeć na kody źródłowe

Strona przedmiotu: github.com/olafijalkowska/Geant4

Zasady zaliczenia: Zaliczenie nastąpi w oparciu o końcowy projekt oraz obecność na zajęciach. Dopuszczam jedną nieusprawiedliwioną nieobecność. Zachęcam do samodzielnego ustalenia tematu projektów, mogę je też zaproponować.

- ▶ Programowanie obiektowe w języku C++ (test)
- ▶ Metody Monte Carlo
- ▶ Moduły kodu wykorzystującego pakiet Geant4
- ▶ Definicja geometrii detektora, kształty i materiały
- ▶ Określanie procesów fizycznych
- ▶ Określanie warunków początkowych zdarzenia (Event)
- ▶ Pojęcia Przebiegu (Run), Zdarzenia (Event) i Kroku (Step)
- ▶ Wyciągnięcie informacji z symulacji
- ▶ Manipulacja symulacją przy pomocy skryptów
- ▶ Opcjonalnie – Tworzenie geometrii z wykorzystaniem rysunków z CAD
- ▶ Nietypowe problemy – z pewnością się z nimi spotkacie

Geant4 oferuje bogatą bibliotekę przykładów, na których nie będziemy się skupiać. Proszę je jednak traktować jako pomoc przy pisaniu projektu.

- ▶ Kompilator C++ obsługujący standard C++11 lub nowszy
- ▶ Program CMake w wersji min. 3.3 znacznie ułatwiający kompilację
- ▶ Biblioteki Geant4 w wersji 10.4
(<https://geant4.web.cern.ch/support/download>)
- ▶ Biblioteki do grafiki (Qt, OpenGL)
- ▶ Edytor tekstu lub ulubione IDE (CLion)

Moja ulubiona konfiguracja CMake

GEANT 4

Aleksandra
Fijałkowska

Wstęp

Organizacja zajęć

Symulacje

Koncept

Metody Monte Carlo

```
aleksandra@aleksandra-UX32LN: ~/programy/geant4.10.04-build
Page 1 of 2
CMAKE_BUILD_TYPE Release
CMAKE_INSTALL_PREFIX /home/aleksandra/programy/geant4.10.04-install
GEANT4_BUILD_MULTITHREADED ON
GEANT4_INSTALL_DATA ON
GEANT4_INSTALL_DATADIR
GEANT4_USE_G3TOG4 OFF
GEANT4_USE_GDML OFF
GEANT4_USE_INVENTOR OFF
GEANT4_USE_OPENGL_X11 ON
GEANT4_USE_QT ON
GEANT4_USE_RAYTRACER_X11 OFF
GEANT4_USE_SYSTEM_CLHEP OFF
GEANT4_USE_SYSTEM_EXPAT ON
GEANT4_USE_SYSTEM_ZLIB OFF
GEANT4_USE_XM OFF
QT_QMAKE_EXECUTABLE /usr/bin/qmake
XERCESC_INCLUDE_DIR XERCESC_INCLUDE_DIR-NOTFOUND

CMAKE BUILD TYPE: Choose the type of build, options are: None Release TestRelease
Press [enter] to edit option CMake Version 3.5.1
Press [c] to configure
Press [h] for help Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)
```

Warto zapoznać się z możliwymi konfiguracjami pakietu Geant4.

Symulacje pełnią istotną rolę na wielu etapach projektów naukowych

- ▶ Projektowanie – Ułatwiają znalezienie optymalnego układu eksperymentalnego
- ▶ Tworzenie i obrona projektu – Pomagają oszacować prawdopodobieństwo sukcesu badań, niejednokrotnie uwiarygodniając ambitne ale ryzykowne pomysły
- ▶ Przeprowadzenie pomiaru – Pozwalają na szybką ocenę otrzymanych wyników, wykrycie ewentualnych problemów
- ▶ Analiza danych – Ułatwiają przeprowadzenie analizy danych i ocenę niepewności

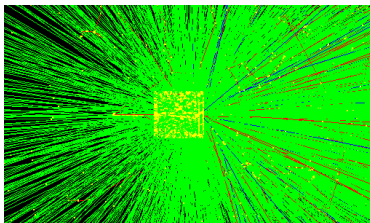
W trakcie tych zajęć skupimy się na symulowaniu procesów fizycznych zachodzących podczas przejścia cząstek przez materię.

- ▶ Elastyczny – Symulacja różnych materiałów, kształtów, różnego rodzaju promieniowania
- ▶ Aktualnie rozwijany – kolejne wersje przynoszą coraz udoskonalenie modeli, przekroje czynne pochodzą z aktualnych baz danych
- ▶ Napisany obiektowo w C++ (rozumiem sceptyków, należy jednak docenić, że nie jest napisany w FORTRAN-ie jak Geant3)
- ▶ Bardzo szeroko rozpowszechniony – Znajomość pakietu Geant4 jest ceniona w wielu grupach badawczych, bogata dostępność gotowych kodów, przykładów
- ▶ Projekt Open Source – Możliwość wprowadzenia swoich wasnych modeli i pomysłów

W trakcie tych zajęć skupimy się na symulowaniu procesów fizycznych zachodzących podczas przejścia cząstek przez materię.

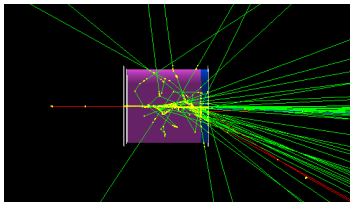
GEANT – "GEometry ANd Tracking"

- ▶ Geant posiada możliwość określania geometrii detektora i śledzenia cząstek w nim propagujących.
- ▶ Zanim symulacja zostanie wykonana użytkownik musi podać informacje niezbędne do jej inicjalizacji – zdefiniować geometrię, materiały, określić cząstki początkowe, ich energię i pęd, podać procesy fizyczne i ich przekroje czynne.
- ▶ Nadrzędną jednostką symulacji jest Run (seria?)
- ▶ Run składa się z szeregu zdarzeń (Event) przeprowadzonych dla określonych warunków początkowych (geometrii i procesów fizycznych)
- ▶ Run reprezentowany jest przez klasę G4Run
- ▶ Użytkownik może określić działania wykonywane na początku i końcu Run-u (np. otwarcie pliku wyjściowego, zapis danych i zamknięcie pliku)



Zdarzenie - Event

- ▶ Każdy Rus składa się z określonej przez użytkownika liczby zdarzeń (Event, klasa G4Event)
- ▶ Event rozpoczyna wysłanie zdefiniowanych przez użytkownika cząstek pierwotnych
- ▶ Na początku zdarzenia wszystkie cząstki pierwotne umieszczane są na stosie a następnie transportowane przez geometrię
- ▶ Niektóre procesy, którym ulegają cząstki pierwotne mogą powodować powstanie cząstek wtórnych (np. kreacja pary elektron-pozyton)
- ▶ Powstałe cząstki wtórne są odkładane na stos, a następnie jedna po drugiej transportowane przez detektor
- ▶ Po przetransportowaniu wszystkich cząstek przez geometrię program wykonuje polecenia określone w klasie G4UserEventAction (zapisanie danych do pliku) i kończy zdarzenie Zdarzeniem kieruje klasa G4EventManager.



Krok - Step

Proces transportowania cząstek pierwotnych i wtórnych odbywa się w krokach (G4Step).

- ▶ Dla każdego z możliwych procesów dyskretnych **losuje** się odległość oddziaływania (w oparciu o przekroje czynne)
- ▶ Najmniejsza z odległości jest wybrana jako krok fizyczny (*physical step length*)
- ▶ Program oblicza odległość od granicy bryły, w której krok się odbywa - krok geometryczny (*geometric step length*)
- ▶ Zanim proces dyskretny zostanie wykonany program realizuje wszystkie aktywowane procesy ciągłe, wpływają one min. na zmianę energii kinetycznej cząstki, powstanie cząstek wtórnych,
- ▶ Jeśli energia kinetyczna cząstki spadnie do zera kończy się śledzenie cząstki, w przeciwnym razie wykonuje się proces dyskretny, mogą powstać cząstki wtórne, zmienić się kinematyka cząstki itp.
- ▶ Program wykonuje polecenia określone w klasie G4UserSteppingAction (światło w Nal) i zapamiętuje dane w Trajektorii
- ▶ Przed rozpoczęciem nowego kroku program wyznacza nowe wartości średniej drogi swobodnej

Wstęp

Organizacja zajęć

Symulacje

Koncept

Metody Monte Carlo

Losowanie? Monte Carlo!

GEANT 4

Aleksandra
Fijałkowska

Wstęp

Organizacja zajęć

Symulacje

Koncept

Metody Monte Carlo

Zanim przejdziemy dalej - test z C++ (bez stresu i jakichkolwiek konsekwencji)

Losowanie? Monte Carlo!

GEANT 4

Aleksandra
Fijałkowska

Wstęp

Organizacja zajęć

Symulacje

Koncept

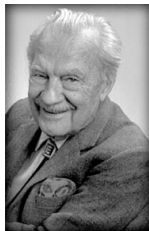
Metody Monte Carlo



(a) S. Ulam



(b) J. von Neumann



(c) N. Metropolis

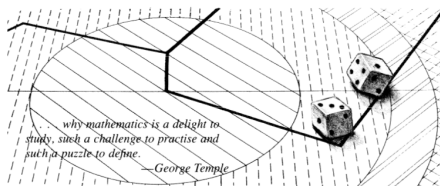


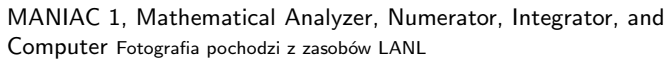
(d) E. Fermi

Fotografie pochodzą z zasobów Wikipedii

THE BEGINNING *of the* MONTE CARLO METHOD

by N. Metropolis





W naszym przypadku metody Monte Carlo będą wykorzystywane do symulacji procesów, które są statystyczne.

Metody te mają szersze zastosowanie. Pierwotnie były wykorzystywane do rozwiązywania skomplikowanych problemów dla których może istnieć rozwiązanie analityczne.

Podstawowym założeniem metody jest stwierdzenie, że **losowa** próbka wybrana z całej populacji przedstawia zbliżone własności do całej populacji.

Błąd metody maleje ze wzrostem liczności próbki (Prawo Bernoulliego).



$$1 + 2 + 3 + \dots + 34 + 35 + 36 = 666$$

W zależności od systemu na kole znajduje się jedno 0 (system europejski) lub 0 i 00 (amerykański).

Założymy uproszczoną wersję zakładów, można obstawiać jedno pole i w razie wygranej uzyskuje się 35-krotność postawionych pieniędzy.


```
#ifndef RULETKA_H
#define RULETKA_H
#include <vector>
#include <random>
class Ruletko
{
public:
    Ruletko(std::default_random_engine& engine);
    ~Ruletko();
    virtual int zakrec() = 0;
protected:
    std::default_random_engine myEngine;
};

class Uczciwa: public Ruletko
{
public:
    Uczciwa(std::default_random_engine& engine);
    ~Uczciwa(){};
    virtual int zakrec();
};

class Amerykanska: public Ruletko
{
public:
    Amerykanska(std::default_random_engine& engine);
    ~Amerykanska(){};
    virtual int zakrec();
};

class Europejska: public Ruletko
{
public:
    Europejska(std::default_random_engine& engine);
    ~Europejska(){};
    virtual int zakrec();
};

#endif
```

```
#include "Ruletko.h"

Ruletko::Ruletko(std::default_random_engine& engine)
{
    myEngine = engine;
}

Ruletko::~Ruletko() {}

Uczciwa::Uczciwa(std::default_random_engine& engine): Ruletko(engine) {}

Europejska::Europejska(std::default_random_engine& engine): Ruletko(engine) {}

Amerykanska::Amerykanska(std::default_random_engine& engine): Ruletko(engine) {}

int Uczciwa::zakrec()
{
    std::uniform_int_distribution<int> dist(1, 36);
    int randomElement = dist(myEngine);
    return randomElement;
}

int Europejska::zakrec()
{
    std::uniform_int_distribution<int> dist(0, 36);
    int randomElement = dist(myEngine);
    return randomElement;
}

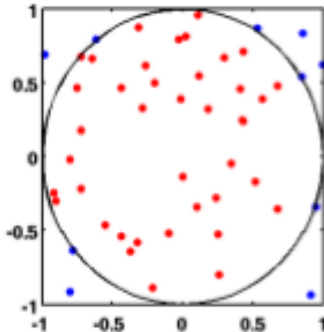
int Amerykanska::zakrec()
{
    std::uniform_int_distribution<int> dist(0, 37);
    int randomElement = dist(myEngine);
    if(randomElement == 37)
        randomElement = 0;
    return randomElement;
}
```

Przykłady - objętość N-wymiarowej kuli

Wyznacz objętość n -wymiarowej kuli o promieniu R metodą Monte Carlo.

Sugestia:

- ▶ Wsłosuj N punktów z n -wymiarowego pudełka o boku $2R$
- ▶ Wyznacz liczbę punktów (M), dla których odległość od punktu 0 znajduje się jest mniejsza od R (te znajdują się wewnątrz kuli)
- ▶ Objętość kuli $V = \frac{M}{N} \cdot (2R)^n$, gdzie $(2R)^n$ jest objętością pudełka



Wstęp

Organizacja zajęć

Symulacje

Koncept

Metody Monte Carlo

Orkiestra składająca się ze 100 muzyków gra koncert. Każdy muzyk gra na innym instrumencie. Po zagranie koncertu schodzą ze sceny i chowają swoje instrumenty do pudełek. Na każdym pudełku jest etykieta jaki instrument powinien znaleźć się w pudełku. Pudełka są identyczne, na tyle duże, że każdy z instrumentów może się tam zmieścić. Muzycy są jednak zmęczeni i chowają instrumenty niedbale (nie zwracają uwagi na etykiety). Gdy wszystkie instrumenty są schowane do pudełek przybiega menadżer i krzyczy: "co Wy robicie! musicie zagrać jeszcze jeden utwór". Muzycy muszą znaleźć swoje instrumenty. Muszą to robić sekwencyjnie, jeden po drugim. Nie mogą się komunikować, oznaczać pudełek, zostawiać otwartych itp. Każdy muzyk podchodzi do wybranego przez siebie pudełka, otwiera je, sprawdza czy to jego instrument. Jeśli znalazł, zapamiętuje naklejoną etykietę i zamyka pudełko. Jeśli nie znalazł, może powtórzyć to dla innego pudełka. Każdy muzyk może sprawdzić maksymalnie 50 pudełek ($100/2$). W jaki sposób powinni sprawdzać pudełka aby zmaksymalizować szanse na powodzenie?

Wstęp

Organizacja zajęć

Symulacje

Koncept

Metody Monte Carlo

Wstęp

Organizacja zajęć

Symulacje

Koncept

Metody Monte Carlo

Pytania?