Symulacje komputerowe w fizyce

Całkowanie Monte Carlo przy użyciu klasy TFoam z Pakietu ROOT

Justyna Peciak, Teresa Rodak

Wydział Fizyki, Matematyki i Informatyki, Fizyka techniczna
T. Kościuszko Cracow University of Technology

Luty 08, 2016

Plan prezentacji

Wprowadzenie

ROOT:

1. Całkowanie Monte Carlo
1. Biblioteka ROOT
   - Czym jest ROOT i do czego służy?
   - Zalety ROOT'a
   - Z czego składa się ROOT?
   - Przykładowe klasy
1. Klasa TFoam
   - Do czego używana jest klasa TFoam
   - Przykładowe metody klasy

Całkowanie Monte Carlo

Na czym polega całkowanie Metodą Monte Carlo?

Załóżmy, że chcemy obliczyć całke z funkcji f(x) w przedziale <xp;xk>. Definicja całki znaczonej mówi, że wartość całki równa jest polu obszaru pod wykresem krzywej w zadanym przedziale całkowania. Ustalamy że wartość funkcji w obszarze całkowania mieszczą się w przedziale <yp;yk>. Pole prostokąta wyznaczonego przez przedział całkownia: <xp;xk> oraz zakres wartości funkcji w tym przedziale :<yp;yk> jest prosty do wyznaczenia i wynosi:

$$P = |xk - xp| * |yp - yp|$$

Całkowanie Monte Carlo

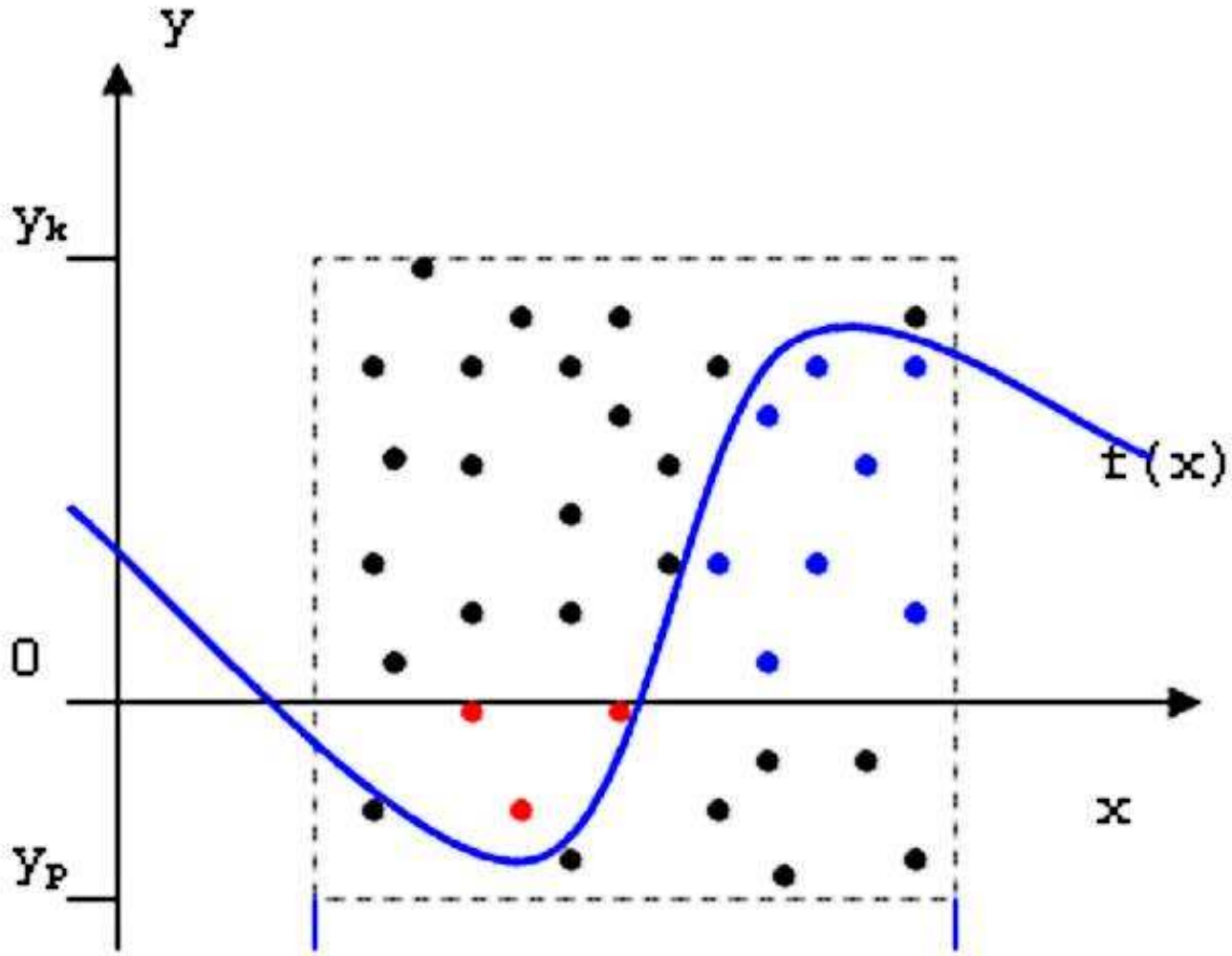
Na czym polega całkowanie Metodą Monte Carlo?

Metoda Monte Carlo polega na wylosowaniu n punktów znajdujących się w obrębie wspomnianego prostokąta.
Przybliżoną wartość całki wyznaczamy z proporcji: stosunek pola pod krzywą do pola prostokąta jest równy stosunkowi liczby punktów pod krzywą do wszystkich wylosowanych punktów.

$$\frac{P_{prostokata}}{Calka} = \frac{n}{c}$$

$$calka = P_{prostokata} \cdot \frac{c}{n} = |x_k - x_p| \cdot |y_k - y_p| \cdot \frac{c}{n}$$

Całkowanie Monte Carlo



Całkowanie Monte Carlo

Błąd całkowania

Błędy całkowania maleją odwrotnie proporcjonalnie do pierwiastka z liczby próbek:

$$1/\sqrt{N}$$

Dokładność całkowania

Dokładność wyniku uzyskanego tą metodą jest zależna od liczby sprawdzeń i jakości użytego generatora liczb pseudolosowych. Zwiększanie liczby prób nie zawsze zwiększa dokładność wyniku, ponieważ generator liczb pseudolosowych ma skończenie wiele liczb losowych w cyklu.

Biblioteka ROOT

Czym jest ROOT i do czego służy?

Obecne eksperymenty fizyczne zbierają ogromne ilości danych. Danych tych nie da się analizować ręcznie choćby ze względu na ich ilość dlatego do ich obróbki wykorzystuje się komputery.

W CERN-ie (Europejska Organizacja Badań Jądrowych) opracowano więc narzędzie, które miało ułatwić analizowanie danych. Narzędziem tym jest własnie ROOT czyli platforma programistyczna na której opierają się najważniejsze eksperymenty fizyki wysokich energii (min. eksperymenty przy LHC - wielki zderzacz hadronów).

Biblioteka ROOT

Zalety ROOT'a

- Mniejsza ilość kodu do zapisania - wykorzystuje się dotychczasowe elementy kodu
- Kod jest bardziej niezawodny - elementy pochodzące ze szkieletu zostały już wcześniej przetestowane.
- Biblioteka pozwala na koncentrowanie się na konkretnych zagadnieniach - interfejs, grafika oraz połączenia pomiędzy elementami są dostarczane w ramach szkieletu

Biblioteka ROOT

Z czego składa się ROOT?

ROOT składa się z kilku części:

- Biblioteki c++ - dostarczają on gotowe klasy, które umożliwiają pisanie gotowych programów w c++ dzięki czemu można w znaczny sposób skrócić czas pracy. W środowisku ROOT dopasowanie funkcji, tworzenie histogramu, ustawienie różnych opcji rysowania itp. można zrobić zaledwie w kilku linijkach kodu.
- Interpreter CINT - można uruchomić wpisując w terminalu polecienie root. Umożliwia uruchamianie skryprów napisanych z biblioteki ROOT'a. Można też napisać własny kawałek kodu i go szybko uruchomić.
- Kompilator ACLiC - jest to kompilator, który z makr potrafi stworzyć pliki współdzielone.

Biblioteka ROOT

Przykładowe klasy

Biblioteka zawiera bardzo dużo klas:

- TObject - definiuje domyślne ustawienia dla wszystkich obiektów w ROOT
- TGraph, TGraphErrors, TGraph2D, TMultiGraph - klasy związane z wykresami
- TRandom1, TRandom2, TRandom3 - klasy generujące liczby losowe (według rozkładu 1,2,3-d)
- TMath - klasa z funkcjami matematycznymi
- TFoam - symulator generatora liczb losowych.

Klasa TFoam

Do czego służy klasa TFoam?

TFoam to ogólnego przeznaczenia "komórkowy" generator zdarzeń - liczb losowych.

Wysoką wydajność generatora MC uzyskuje się poprzez podzielenie dziedziny całkowania na małe komórki. Komórki mogą być n-wymiarowe. Siatka komórek nazywana jest "pianą"(ang. foam) wtrwarzana jest w procesie binarnego podziału komórek.

Klasa TFoam

Przykładowe metody klasy

- TFoam::MakeEvent - generuje losowy punkt / wektor według rozkładu zdefiniowanego przez użytkownika.
- TFoam::TRandom - to podstawowa klasa ROOT'a do generowania liczb losowych.
- TFoam::GetMCvect - przy pomocy tej metody użytkownik może uzyskać punkt/wektor wygenerowany metoda Monte Carlo.
- TFoam::GetIntegMC - zwraca wartość całki obliczonej metodą Monte Carlo.
- TFoamIntegrand - funkcja całkująca.

Sposób instalacja biblioteki ROOT

Instalacja - Krok 1

Pierwszym krokiem przy instalacji ROOT'a jest pobranie ze strony paczki z plikami wykonywalnymi oraz należy je rozpakować w katalogu docelowym.

$gunziproot_v6.04.14.Linyx - ubuntu14 - x86_64 - gcc4.8.tar.gz$
$tarxvfroot_v6.04.14.Linyx - ubuntu14 - x86_64 - gcc4.8.tar$

Następnie pobieramy kod źródłowy. Najlepiej pobrać najnowszą wersję. Paczkę należy rozpakować w tym samym katalogu.

$tarzxvfroot_v2.25.xx.source.tar.gz$

Przechodzimy do tego katalogu i upewniamy się że znajduje się w nim plik Makefile.

Instalacja - Krok 2

Aby poprawnie zainstalować ROOT'a musimy jeszcze pobrać dodatkowe paczki. Można zrobić to poleceniem:

sudoapt – getinstallbuild – essentialgfortransubversionxorg – devlibxml2 – dev libmysqlclient – devlibfftw3 – devlibssl – devlibglu1 – mesa – devautomakeautoconf libtoolcurllibncurses5 – devbinutilslibX11 – devlibXpm – devlibXft – devlibXext – dev

Czasem trzeba doinstalować dodatkowe biblioteki.

Instalacja - Krok 3

Kolejną niezbędną czynnością jest ustawienie zmiennych środowiskowych. Są one niezbędne do poprawnej kompilacji i prawidłowego działania programu. Aby to zrobić otwieramy plik .bashrc, który znajduje się w katalogu domowym home/:

gedit /.bashrc

i na końcu tego pliku dopisujemy linijki:

```
export ROOTSYS=/opt/root
export PATH=\$PATH:\$ROOTSYS/bin
export LD_LIBRARY_PATH=\$LD_LIBRARY_PATH:\$ROOTSYS/lib
```

Instalacja - Krok 4

Teraz możemy już skonfigurować instalację komendą"

./configure

Kiedy już skonfigurujemy ROOT'a możemy przejść do kompilacji. W tym celu wystarczy wpisać w konsoli:

make

Kompilacja jest dość długim procesem - może trwać ponad godzinę. Na koniec wyświetla się informacja aby przed uruchomieniem ROOT'a wykonać polecenie:

.../root/bin/thisroot.sh

Instalacja - Krok 5

Przy wpisaniu w konsolę komendy *root* włącza się program.

Jeśli nie chcemy żeby wyświetlał się ekran startowy ROOT'a przy wpisywaniu komendy root w konsoli należy dodać -l.

ROOT w praktyce

ROOT w praktyce

Kod programu głównego

```
#include "Riostream.h"
#include "TFile.h"
#include "TFoam.h"
#include "TCanvas.h"
#include "TH2.h"
#include "TMath.h"
#include "TFoamIntegrand.h"
#include "TRandom3.h"
#include "Funkcja.cpp"
```

Funkcja

```
#include "TMath.h"

double Funkcja(int nDim, double *Xarg) ///Dwuwymiarowy rozkład cel.
{
double x=Xarg[0];
double y=Xarg[1];
double Fun=0;
  Fun +=TMath::Sin(x+1.0)*TMath::Cos(y-1.0);

 return Fun;
}
```

Kod programu głównego

```
using namespace std;

int main()
{
 cout<<"--- Tworzenie wykresu ---"<<endl;
 TH2D  *hst_xy = new TH2D("Legenda" ,  "Wykres x-y", 100,0,1.0, 100,0,1.0); ///Definicja parametrów wykresu -
                                                                              (Legenda, Tytuł wykresu,
                                                                               Wymiary osi x i y)

 Double_t *MCvect = new Double_t[2];     ///Dwuwymiarowy wektor generowany przez MC run.
 TRandom *PseRan= new TRandom3();        ///Tworzy generator liczb losowych.
 PseRan->SetSeed(4357);
 TFoam   *FoamX    = new TFoam("FoamX");//Tworzy symulator generatora liczn losowych.
 FoamX->SetkDim(2);                      ///Liczba wymiarów
 FoamX->SetnCells(500);                  ///Liczba celi (domyślnie jest to 2000)
 FoamX->SetRhoInt(Funkcja);              ///Ustawia dwuwymiarowy rozkład.
 FoamX->SetPseRan(PseRan);               ///Ustawia generator liczb losowych.
 FoamX->Initialize();                    ///Inicjalizuje symulator (może to chwilkę potrwać).


/// Tworzenie okna do wyświetlenia wykresu.
  TCanvas *cKanwa = new TCanvas("cKanwa","Calkowanie metoda Monte Carlo",800,600); ///Tworzy okno
                                                                                    wyświetlające wykres
                                                                                  ("klasa", "Tytuł", wymiary okna)
  cKanwa->cd();
```

Kod programu głównego

```
///Od teraz FoamX gotowy do generowania "wydarzeń"
 int nshow=5000;
 for(long loop=0; loop<10000; loop++)
        {
         FoamX->MakeEvent();            //Generuje MC zdarzenie.
         FoamX->GetMCvect(MCvect);      //Pobiera wygenerowny wektor (x,y).
         Double_t x=MCvect[0];
         Double_t y=MCvect[1];
         if(loop<10) cout<<"(x,y) =  ( "<< x <<", "<< y <<" )"<<endl;
         hst_xy->Fill(x,y);             /// Wykres rysowany w trakcie działania programu.
            if(loop == nshow)
              {
               nshow += 5000;
               hst_xy->Draw("lego2");    ///Rysowanie słupków.
               cKanwa->Update();         ///Aktualizowanie wykresu
              }
        }
  hst_xy->Draw("lego2");        ///Ostateczny wykres
  cKanwa->Update();

  Double_t MCresult, MCerror;   /// Zwraca całkę.
  FoamX->GetIntegMC( MCresult, MCerror);
  cout << " MCresult= " << MCresult << " +- " << MCerror <<endl; ///Zwraca wynika całkowania z uwzględnieniem
                                                                 błędu.
  cout<<"--- Koniec ---"<<endl;

 return 0;
}
```

Działanie programu

W zależności od przyjętej wartośći SetnCells() zmienia się dokładność całkowania - zmniejsza się błąd.

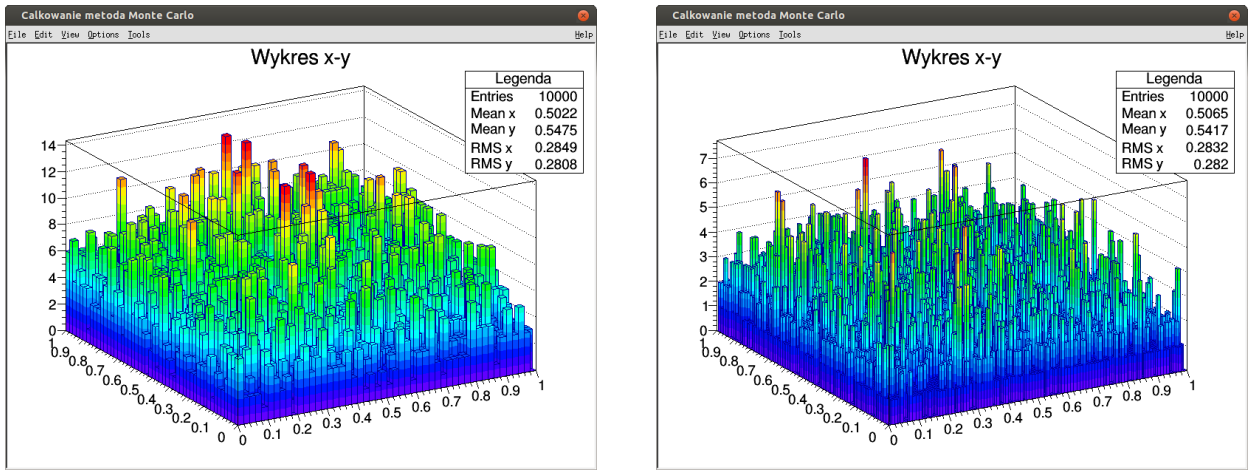
SetnCells(1000) -> $MCresult = 0.80482227 + -0.00015770558$
SetnCells(2000) -> $MCresult = 0.80470824 + -0.00013189493$
SetnCells(5000) -> $MCresult = 0.80466268 + -0.00010574415$

Działanie programu

```
justyna@justyna-VirtualBox: ~/roocik/PROJEKT/src
Plik Edycja Widok Wyszukiwanie Terminal Pomoc
2222222222222222
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
F                                                                            F
F                    ***  TFoam::Initialize FINISHED!!!  ***                 F
F     nCalls =     199800 =            Total number of function calls       F
F     XPrime =     0.82346785   =      Primary total integral                F
F     XDiver =    0.018619036   =      Driver  total integral                F
F   mcResult =     0.80484882   =      Estimate of the true MC Integral      F
F                                                                            F
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
(x,y) =  ( 0.41510581, 0.2079734 )
(x,y) =  ( 0.21847249, 0.87668121 )
(x,y) =  ( 0.71214118, 0.61488836 )
(x,y) =  ( 0.29748162, 0.56792246 )
(x,y) =  ( 0.23337683, 0.43797535 )
(x,y) =  ( 0.68758711, 0.65575801 )
(x,y) =  ( 0.4029076, 0.67724973 )
(x,y) =  ( 0.45124346, 0.89701383 )
(x,y) =  ( 0.83685841, 0.36158518 )
(x,y) =  ( 0.81257419, 0.20492629 )
 MCresult= 0.80482227 +- 0.00015770558
--- Koniec ---
(int)0
root [1]
```

Działanie programu

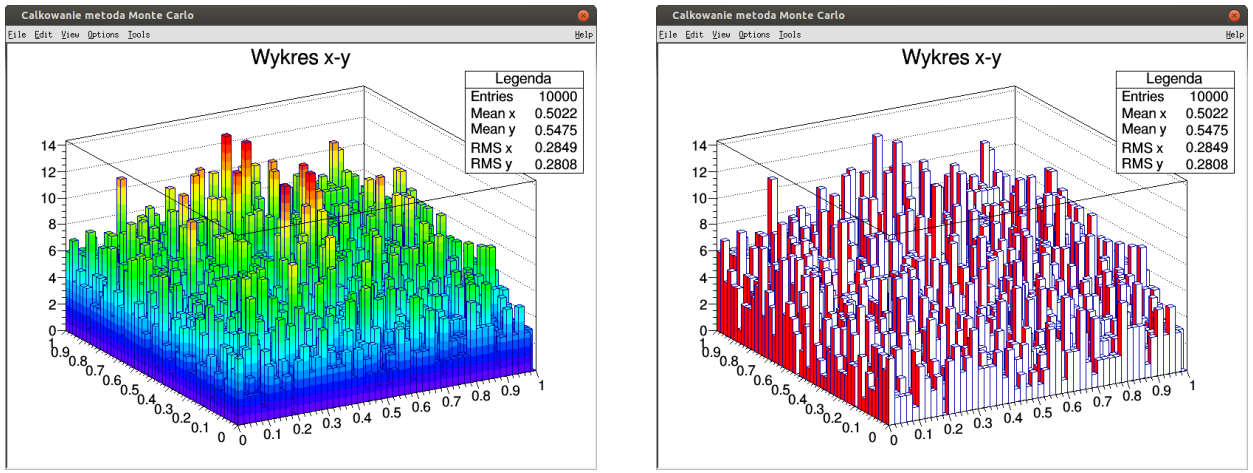
Zmieniając parametr drugi parametr w TH2D("Legenda" , "Wykres x-y", 50,0,1.0, 50,0,1.0) zagęszczamy siatkę.



Rysunek : Zagęszczenie siatki

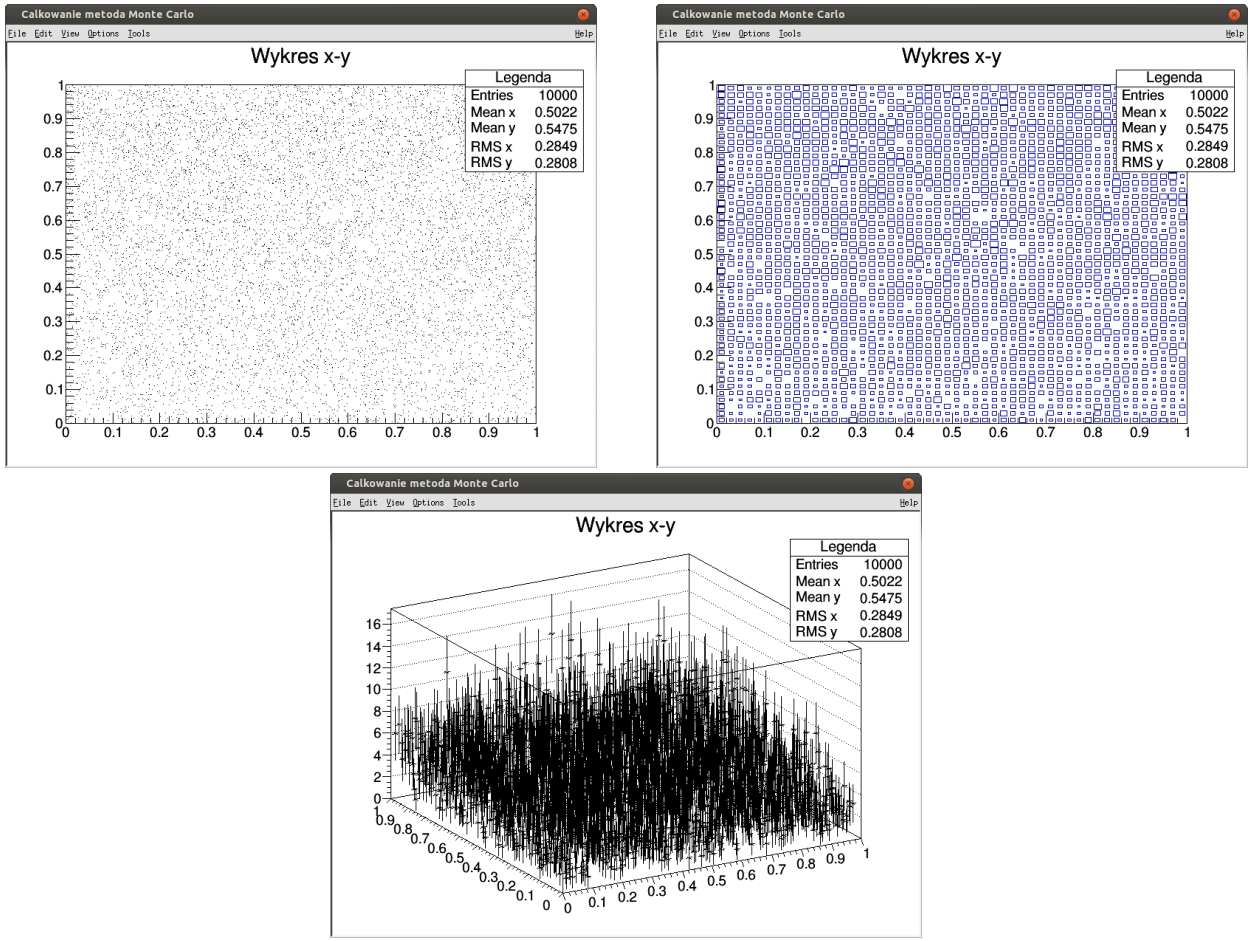
Działanie programu

Zmieniając parametr w Draw() zmieniamy typy wykresu.



Rysunek : Typy wykresów

Działanie programu



Rysunek : Typy wykresów

Bibliografia


https://root.cern.ch/


https://root.cern.ch/doc/master/annotated.html


https://root.cern.ch/doc/master/classTFoam.html


http://arxiv.org/abs/physics/0210061

Dziękujemy za uwagę