Zadanie 3
Dana jest gęstość prawdopodobieństwa:

$$f(x,y) = \begin{cases} cx^m y^n, & 0 \leq x \leq 1 \wedge 0 \leq y \leq 1; \\ 0, & pozostale. \end{cases} \tag{1}$$

Należy dla parametrów $m = 1$ i $n = 3$:

- Wyznaczyć stałą c w taki sposób, aby rozkład gęstości był unormowany [1 pkt]
- Narysować gęstość prawdopodobieństwa f(x,y) [1 pkt]
- Wyznaczyć i narysować dystrybuantę F(x,y) [1 pkt]
- Wyznaczyć i narysować gęstość brzegową g(x) [1pkt]
- Wyznaczyć i narysować gęstość brzegową h(y) [1pkt].

Uwagi:

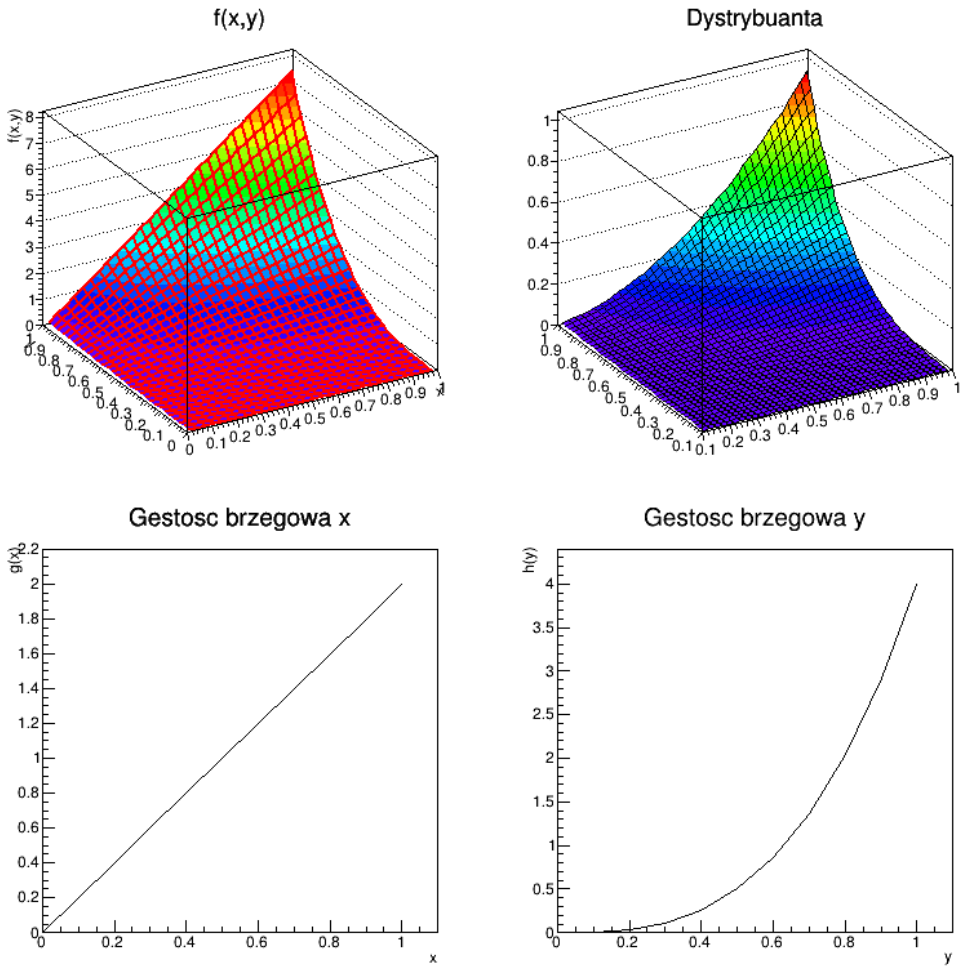
- Wszystkie wielkości (stała c, dystrybuanta, gęstości brzegowe) wyznaczamy numerycznie wykorzystując np. metodę Integral.
- Skrypt powinien być napisany w taki sposób aby wykonywał obliczenia dla dowolnego m i n.
- Należy oznaczyć i podpisać wszystkie osie.
- Funkcję rozkładu gęstości (obiekt TF2) tworzymy poprzez wykorzystanie konstruktora zawierającego wskaźnik na funkcję (odpowiednio zdefiniowanej dla wszystkich wartości x i y):

```
double fcn(double *x, double *params)
```

```
TF2(const char* name, void* fcn, Double_t xmin = 0, Double_t
    ↪ xmax = 1, Double_t ymin = 0, Double_t ymax = 1, Int_t
    ↪ npar = 0)
```

Przykład:

```
double function(double *x, double *params){
    if(x[0]>=0 && x[1]>=0) return params[0]*x[0]*x[1];
    return 0;
}
...
double xmin = 0;
double xmax = 10;
double ymin = 0;
double ymax = 10;
int nparams = 1;
TF2* fun1 = new TF2("fun1", function,xmin,xmax,ymin,ymax,
    ↪ nparams);
...
```



Rysunek 1: Przykładowe wykresy końcowe.

output:

```
Stala normujaca c = 8
```

Teoria:

1. Rozkład prawdopodobieństwa jest unormowany:

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)dxdy = 1$$

2. Dystrybuanta:

$$F(x,y) = \int_{-\infty}^{x}\int_{-\infty}^{y} f(x',y')dx'dy'$$

3. Brzegowa gęstość prawdopodobieństwa:

$$P(a \leq X \leq b, -\infty, \infty) = \int_a^b \left[\int_{-\infty}^{\infty} f(x,y)dy\right] dx = \int_a^b g(x)dx$$

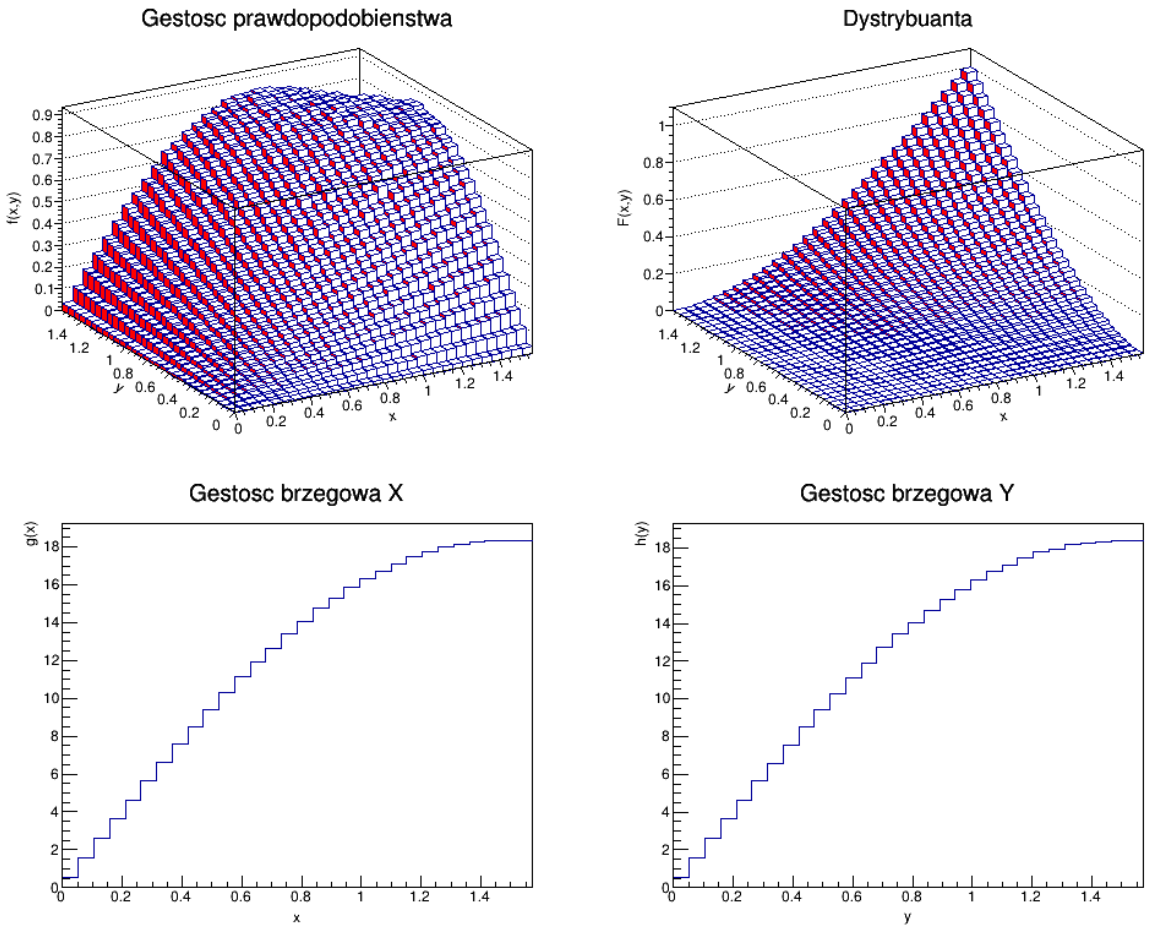
$$g(x) = \int_{-\infty}^{\infty} f(x,y)dy$$
$$h(y) = \int_{-\infty}^{\infty} f(x,y)dx$$

Zadanie 4
Dana jest gęstość prawdopodobieństwa:

$$f(x,y) = \begin{cases} c \cdot sin(x \cdot y), & 0 \leq x \leq \frac{\pi}{2} \wedge 0 \leq y \leq \frac{\pi}{2} \\ 0, & pozostale. \end{cases} \tag{1}$$

Należy:

1. Wyznaczyć stałą c w taki sposób, aby rozkład gęstości był unormowany [0.5 pkt]
2. Wylosować z rozkładu pary liczb (x,y) i wypełnić nimi histogram (obiekt TH2D) gęstości prawdopodobieństwa f(x,y)[0.5 pkt]
3. Unormować histogram gęstości i narysować [0.5 pkt]
4. Wyznaczyć i narysować histogram dystrybuanty F(x,y) [1 pkt]
5. Wyznaczyć i narysować histogramy gęstości brzegowej g(x) i h(y) [1pkt]
6. Wyznaczyć:
   (a) wartości oczekiwane *E(X)* i *E(Y)*; [0.5 pkt]
   (b) odchylenie standardowe $\sigma(X)$ i $\sigma(Y)$; [0.5 pkt]
   (c) kowariancję *cov(X,Y)*; [0.5 pkt]
   (d) Współczynnik korelacji *rho(X,Y)*. [0.5 pkt]



Rysunek 1: Przykładowe wykresy końcowe.

output:

```
E(X) = 0.990629
E(Y) = 0.990713
sigma(X) = 0.37755
sigma(Y) = 0.377713
Cov(X,Y) = -0.0138571
Rho(X,Y) = -0.0971707
```

Teoria:

- Odchylenie standardowe - określa niepewność pomiaru, odległość danych od wartości średniej.
$$\sigma(X) = \sqrt{E[X^2] - E[X]^2}$$
- Kowariancja - określa wspólne zmiany zmiennych X i Y. Jeśli jest dodatnia to wartości Y na ogół rosną przy wzroście X, jeśli jest ujemne to wartości Y na ogół maleją przy wzroście X. Jeśli nie ma zależności to kowariancja jest równa 0.
$$cov(X, Y) = E[X \cdot Y] - E[X] \cdot E[Y]$$
- Współczynnik korelacji - ocenia liniową zależność między współrzędnymi X i Y. Jest wielkością znormalizowaną $-1 \leq \rho(X, Y) \leq 1$. Kowariancja i wpsółczynnik korelacji nie mówią o zależności przyczynowo-skutkowej.
$$\rho\,(X, Y) = \frac{cov(X, Y)}{\sigma(X) \cdot \sigma(Y)}$$

Podpowiedź: w zadaniu należy wykorzystać istniejące metody do wyznacznia wartości z punktu 6.

Zadanie 5
Wykonano pomiary trzech wielkości X1, X2 oraz X3. Wyniki pomiarów znajdują się w plikach: Dane1.dat, Dane2.dat, Dane3.dat.

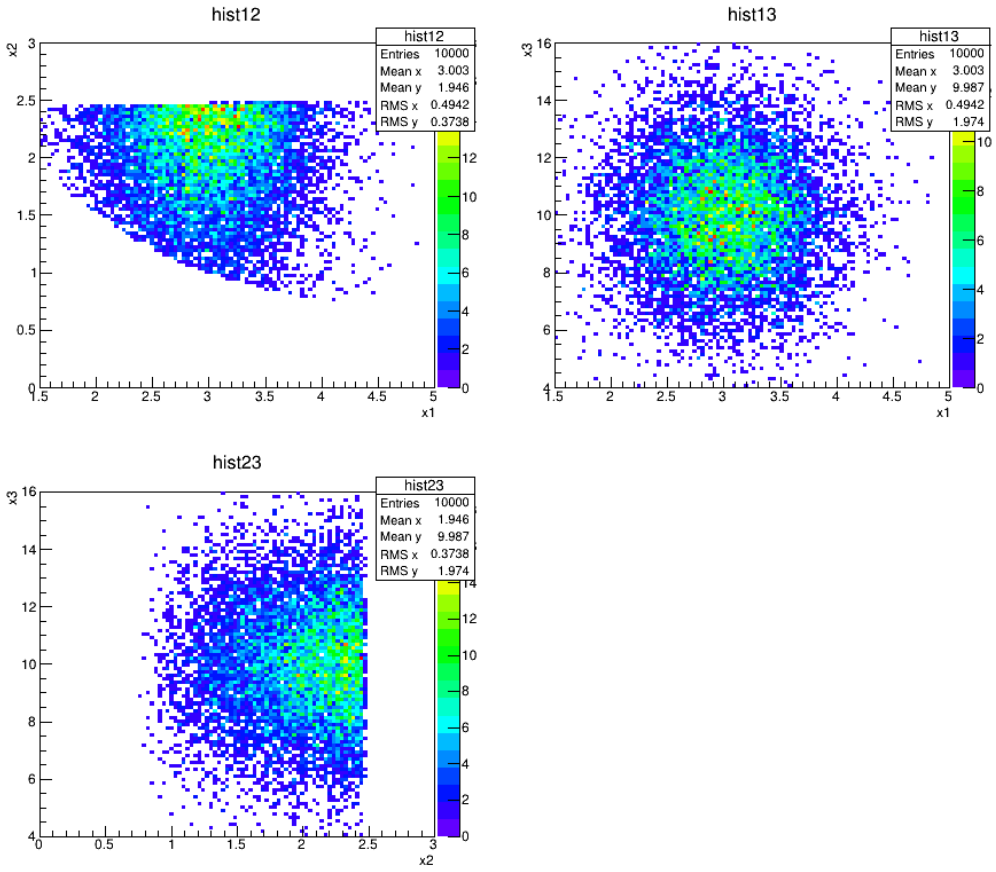
1. Należy określić wyniki pomiaru (średnia oraz niepewność - odchylenie standardowe) tych wielkości oraz narysować wykresy zależności między zmiennymi (X1,X2), (X2,X3) oraz (X1, X3). [1 pkt]

Wielkości Y1 i Y2 związane są z wielkościami X1, X2, X3 następującymi relacjami:

$$Y1 = 3 * X1 + 2 * X2 + X3$$
$$Y2 = 2 + 5 * X1 + 2 * X2$$

Należy wyznaczyć:

2. macierz kowariancji zmiennych X1, X2, X3 [1 pkt]
3. zmierzone wartości średnie zmiennych Y1, Y2 w pomiarze pośrednim [0.5 pkt]
4. macierz kowariancji zmiennych Y1, Y2 [1 pkt]
5. błędy z jakimi zmierzono pośrednio wielkości Y1, Y2 [1 pkt]
6. współczynnik korelacji pomiędzy zmiennymi Y1 i Y2 [0.5 pkt]



Rysunek 1: W celu narysowaniu zależności należy wykorzystać metodę Draw() z opcją *COLZ*.

output:

```
X1=3.00285, u(X1)=0.49416
X2=1.94644, u(X2)=0.373789
X3=9.98716, u(X3)=1.9737
rho(X1,X2)=-0.084933
rho(X1,X3)=0.0018259
rho(X2,X3)=-0.00333563

Po zaokragleniu:
X1=3.00, u(X1)=0.49
X2=1.95, u(X2)=0.37
X3=9.99, u(X3)=1.97
Macierz kowariancji wielkosci X

3x3 matrix is as follows

     |       0     |       1     |       2     |
-------------------------------------------------
   0 |      0.2442     -0.01569     0.001781
   1 |    -0.01569       0.1397    -0.002461
   2 |    0.001781    -0.002461        3.895

Macierz kowariancji wielkosci Y

2x2 matrix is as follows

     |       0     |       1     |
-------------------------------
   0 |       6.465         3.975
   1 |       3.975          6.35

Y1=22.8886, u(Y1)=2.57917
Y2=20.9071, u(Y2)=2.58142
rho(Y1,Y2)=0.596997
rho(Y2,Y1)=0.596997

Po zaokragleniu:
Y1=22.89, u(Y1)=2.58
Y2=20.91, u(Y2)=2.58
```

Uwagi:

- Wczytywanie danych z pliku tekstowego (tak samo jak w języku C++):

```
ifstream ifile;
ifile.open("dane.dat");
double val;
while(ifile>>val){
    cout<<val<<endl;
}
ifile.close();
```

- Do obliczeń na macierzach można wykorzystać klasę TMatrixD.

Teoria (wikibooks):

Przypomnienie z wykładu nr 4 (od slajdu 11):
Definiujemy $\mathbf{Y}$ jako liniową kombinację zmiennych $\mathbf{X}$

$$\begin{aligned} Y_1 &= a_1 + t_{11} * X_1 + t_{12} * X_2 + ... + t_{1n} * X_n \\ Y_2 &= a_2 + t_{21} * X_1 + t_{22} * X_2 + ... + t_{2n} * X_n \\ &\vdots \\ Y_r &= a_r + t_{r1} * X_1 + t_{r2} * X_2 + ... + t_{rn} * X_n \end{aligned}$$

W zapisie macierzowym: $\mathbf{Y} = T\mathbf{X} + \mathbf{a}$.
Wartość oczekiwana: $E(\mathbf{Y}) = \hat{\mathbf{y}} = T\hat{\mathbf{x}} + \mathbf{a}$
Macierz kowariancji: $C_{\mathbf{Y}} = E\left((\mathbf{Y} - \hat{\mathbf{y}})(\mathbf{Y} - \hat{\mathbf{y}})^T\right) = TC_{\mathbf{X}}T^T$

Zadanie 6

Część I: obliczanie liczby PI [1 pkt]

Wykorzystujemy metodę von Neumanna (**metodę von Neumanna** (wykład 5: slajdy 9-20) do obliczenia liczby PI. Jest to prosty przykład wykorzystania metod typu Monte Carlo. Należy napisać **funkcję**, która obliczy wartość PI. Losujemy dwie **x** i **y** z rozkładu jednorodnego na przedziale [0,1] i sprawdzamy czy punkt mieści się wewnątrz koła o promieniu 1. Na podstawie liczby zaakceptowanych punktów (punktów wewnątrz okręgu) i liczby odrzuconych punktów (poza okręgiem) oraz wzoru na powierzchnię koła należy obliczyć PI. Należy również narysować zaakceptowane oraz odrzucone punkty na jednym wykresie (wykorzystujemy dwa obiekty *TGraph*). Można również dodać histogram (*TH1D*) zaakceptowanych wartości x.

Część II: generowanie liczb pseudolosowych z dowolnego rozkładu metodą akceptacji i odrzucenia von Neumanna [3 pkt]

Metoda von Neumanna pozwala na wygenerowanie liczb pseudolosowych, gdy znany jest tylko rozkład g(y). W ogólności metoda działa nawet wtedy, gdy funkcja g(y) nie jest rozkładem gęstości prawdopodobieństwa (całka z niej nie wynosi 1). Pozwala to na bardzo szerokie wykorzystanie metody von Neumanna - przede wszystkim do obliczania całek oznaczonych ze skomplikowanych funkcji, gdy ich analityczne scałkowanie jest niemożliwe. Metody tego typu noszą nazwę wspomnianych wcześniej metod Monte Carlo.

Należy stworzyć trzy bardzo podobne funkcje przyjmujące obiekt typu TF1. Powinny one przyjmować funkcję g na przedziale [min,max]:

- funkcje, która zwraca liczbę pseudolosową o rozkładzie g(y)

```
double losujVonNeumann(TF1* g, double min, double max)
```

- funkcje zwracająca wydajność metody akceptacji i odrzucenia von Neumanna dla danej funkcji g(y) i liczby losowań n

```
double wydajnoscVonNeumann(TF1* g, double min, double max,
    ↪ int n)
```

- funkcje, która zwraca obliczoną metodą von Neumanna całkę z g(y) na przedziale [min,max] przy użyciu n losowań

```
double calkaVonNeumann(TF1* g, double min, double max, int n
    ↪ )
```

Część III: metoda akceptacji i odrzucania von Neumanna z funkcją pomocniczą [1 pkt]

Zwiększenie wydajności metody von Neumanna można osiągnąc przez ograniczenia obszaru losowania wykorzystując funkcję pomocniczą. Należy napisać funkcję, która oblicza całkę wykorzystując metodę von Neumanna z funkcją pomocniczą:

```
double calkaVonNeumannZPomoc(TF1* g, TF1* s, double min, double max,
    ↪ int n, double &wydajnosc)
```

Funkcja ma rownież zwracać przez referencję wartość wydajności metody.

Obliczenia wykonujemy dla funkcji:

$$g(y) = 0.4 + \frac{sin\,(\pi * (x + 1.2))}{\pi * (x + 1.2) + 1} \tag{1}$$

Uwagi

1. Funkcję s(y) dobieramy w taki sposób aby zawsze była powyżej funkcji g(y) i łatwo można było wylosować liczbę pseudolosową metodą odwrotnej dystrybuanty (**w zadniu wykorzystujemy funkcję liniową**)

2. do losowania liczb z rozkładu jednorodnego wykorzystujemy klasę **TRandom** i funkcję **Uniform**. Można również spróbować wykorzystać istniejący obiekt tej klasy **gRandom**

3. Pamiętajmy, że zawsze akceptujemy wylosowane punkty pod zadaną krzywą w stosunku do wszystkich wylosowanych (stosunek ten ma się do siebie (dla N dążących do nieskończoności) jak stosunki powierzchni obszarów wewnątrz których zostało dokonane losowanie)

4. Funkcja kwadratowa i jej odwrotność:

$$a * x^2 + b * x + c = y$$

$$x = \pm\sqrt{\frac{y}{a} + \frac{b^2 - 4ac}{4a^2}} - \frac{b}{2a}$$

Funkcje pomocnicze

- Calka funkcji liniowej $y = ax + b$:

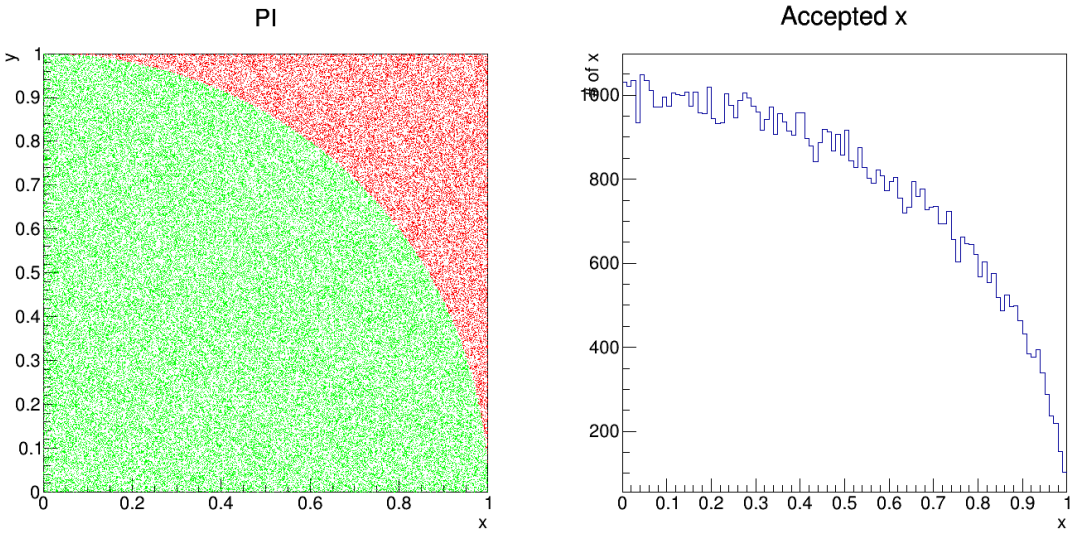
```
double calkaFunkcjiLiniowej(double x, double a, double b){
  return a * x*x / 2 + b * x;
}
```

- odwrotna dystrybuanta funkcji liniowej $y = ax + b$:

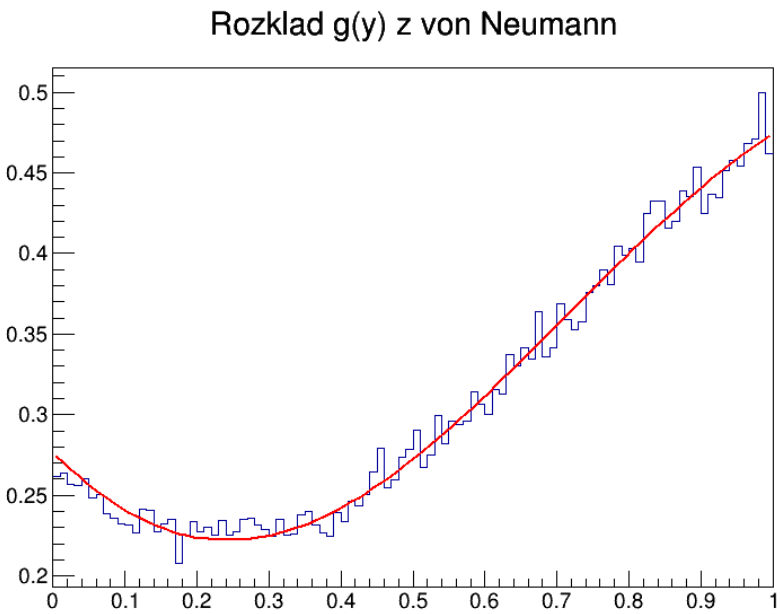
```
double odwrotnaDystrybuanta(double x, double a, double b){
  return  - b/(a) + TMath::Sqrt(2*x/a + (b*b)/(a*a) );
}
```

output

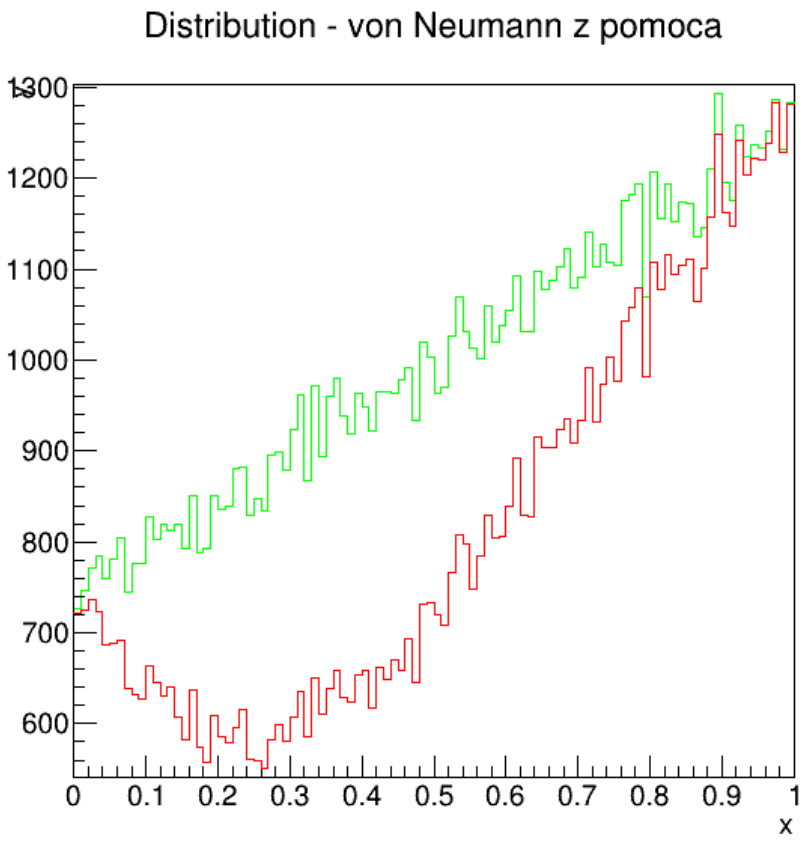
```
Wyliczone PI: 3.1398
Calka metoda Integral: 0.308157
# von Neumann #
Calka: 0.308123
Wydajnosc: 0.649832
# von Neumann z funkcja pomocnicza #
Calka: 0.308161
Wydajnosc: 0.820594
```



Rysunek 1: Obliczanie liczby PI



Rysunek 2: Rozklad g(y) nałożony na wyniki losowania von Neumanna



Rysunek 3: Rozklad g(y) wyznaczony metodą von Neumanna z liniową funkcją pomocniczą. Zielony histogram przedstawia wszystkie wylosowane wartości y_i natomiast czerwony wartości zaakceptowane.

Zadanie 7

Część I: liniowy kongruentny generator liczb losowych [2 pkt]

Należy napisać generator liczb pseudolosowych oraz zapisać wygenerowane liczby do pliku. Generator powinien opierać się na wzorze:

$$x[j+1] = (g * x[j] + c) \bmod m \tag{1}$$

Generator taki nazywamy generatorem LCG - czyli generatorem liniowym kongruentnym. Zadanie pewnej wartości poczatkowej x[0] definiuje nam zatem cały ciąg, który ponadto jest ciągiem okresowym. Okres zależy od doboru parametrów i przy spelnieniu kilku warunków osiąga maksymalnie wartość m. Warunki te to:

- c i m nie mają wspólnych dzielników
- b = g-1 jest wielokrotnością każdej liczby pierwszej, która jest dzielnikiem liczby m
- b jest wielokrontością 4 jeśli n też jest wielokrotnością 4.

Makro powinno być napisane w taki sposób, aby wartości g i m łatwo można zmienić. Wynikiem makra powinien być plik o rozszerzeniu *.dat, w którym zapisane zostały wygenerowane liczby. Należy otrzymać trzy pliki z 10^4 liczbami pseudolosowymi o parametrach:

- m=97 i g=23,
- m=12799 i g=223,
- m=147483647 i g=16807.

Część II: test widmowy [1 pkt]

Należy sprawdzić jakość utworzonych generatorów. W tym celu należy narysować na płaszczyźnie punkty o współrzędnych (x_j, x_{j+1}). Należy wykorzystać **TH2D**. Otrzymamy w ten sposób obraz przypominający widmo generatora. Jeśli punkty rozłożone są równomiernie to generator działa popranie. Jeśli widać jednak okresowość, znaczy, że generator nie działa poprawnie. Wynikiem zadania powinny być trzy widma na podstawie liczb otrzymanych w części I.

Część III: generacja liczb losowych oparta na transformacji rozkładu jednorodnego [3 pkt]

Należy wygenerować 10000 liczb według rozkładu opisanego funkcją 2 (dla $\tau = 3$) wykorzystując metodę odwrtonej dystrybuanty.

$$f(x) = \begin{cases} \frac{1}{\tau} e^{-x/\tau}, & x \geq 0, \\ 0, & x < 0 \end{cases} \tag{2}$$

- Analitycznie (na kartce) wyznaczyć dystrybuantę rozkładu oraz jej funkcję odwrotną [1 pkt].
- Wygenerować rozkład f(x) - wrzucając wygenerowane wartości do histogramu - korzystając z: [1 pkt.]
  - liczby losowe wygenerowane w części I (pliki *dat),
  - standardowego generatora ROOT'a *gRandom->Rndm(1).*
- Narysować na jednym wyresie histogram (odpowiednio unormowany) oraz rozkład f(x) [1 pkt].

Końcowo należy przedstawić 4 histogramy.

Teoria:

Metoda odwrotnej dystrybuanty

Zakładamy, że prawdopodobieństwo g(y)dy jest równe f(x)dx, gdzie dx odpowiada wartością dy. Warunek jest spełniony dla odpowiednio małych dx. Wynika stąd, że:

$$g(y) = \frac{dx}{dy} f(x)$$

Teraz jeżeli założymy, że gęstość prawdopodobieństwa f(x) wynosi 1 w $0 \leq x \leq 1$ i $f(x) = 0$ dla $x < 0$ i $x > 1$ to powyższe równanie możemy zapisać w postaci:

$$g(y)dy = dx = dG(y),$$

gdzie G(y) jest dystrybuantą zmiennej losowej Y. Co po całkowaniu daje nam

$$x = G(y) \quad \rightarrow \quad y = G^{-}1(x). \tag{3}$$

Jeśli zmienna losowa X ma rozkład jednostajny na odcinku pomiędzy 0 i 1 oraz jeśli znana jest funkcja odwrotna G^{-1}(x) to funkcja g(y) opisuje gęstość prawdopodobieństwa rozkładu zmiennej losowej Y.

Uwagi:

- Liczba binów w histogramach powinna być równa 100
- Wczytywanie danych z pliku tekstowego (tak samo jak w języku C++):

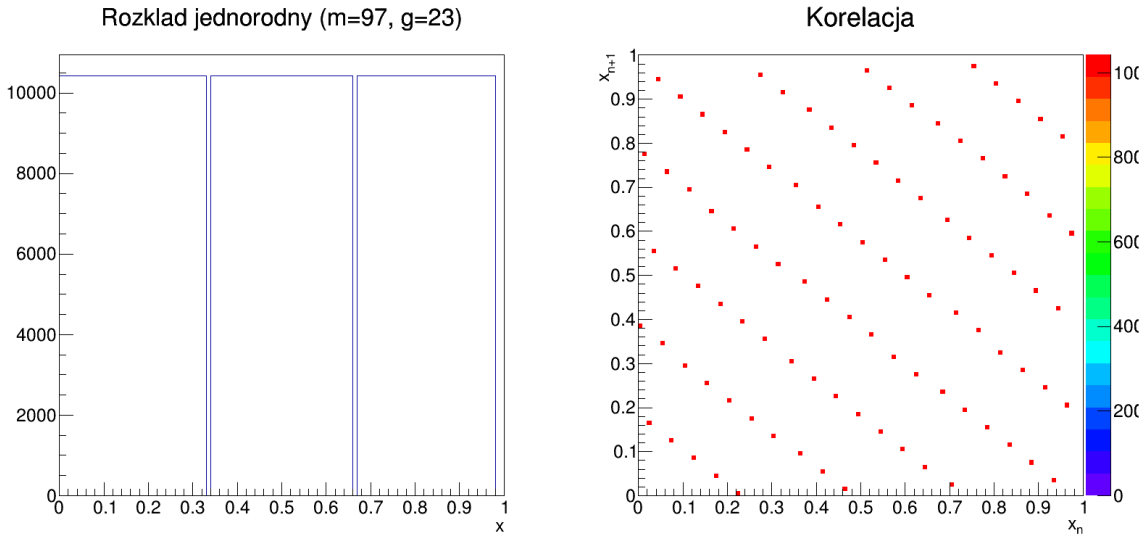
```
ifstream ifile;
ifile.open("dane.dat");
double val;
while(ifile>>val){
    cout<<val<<endl;
}
ifile.close();
```

- Zapisywanie danych do pliku tekstowego (tak samo jak w języku C++):

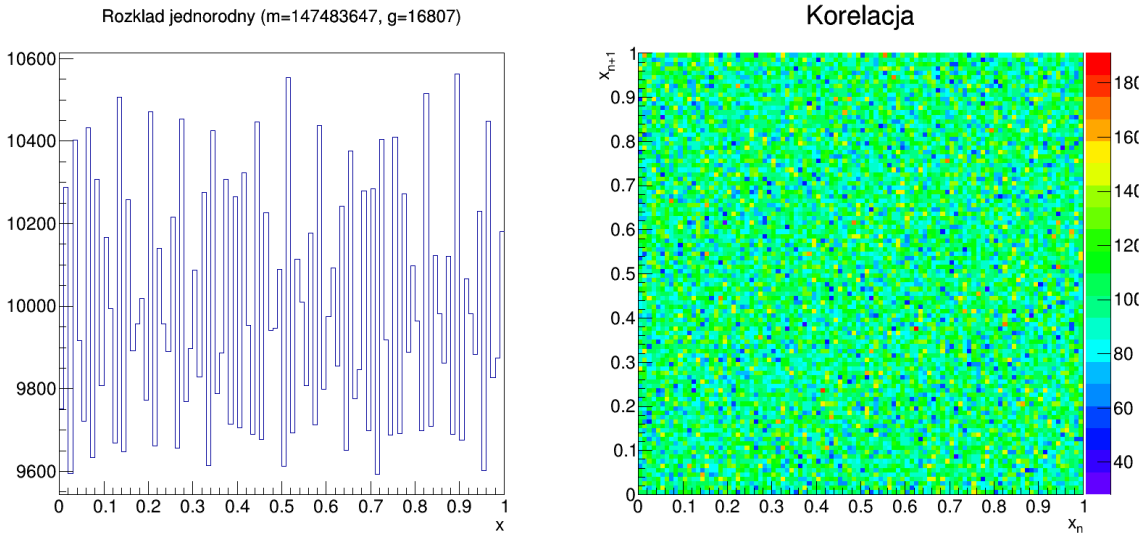
```
ofstream ofile;
ofile.open("dane.dat");
for(int i=0; i<N; i++){
    ofile<<val[i]<<endl;
}
ofile.close();
```

- Zapisywanie wykresu w ROOT, można wykonać przez zapisanie całego canvasu: c->SaveAs(ńazwa.pn

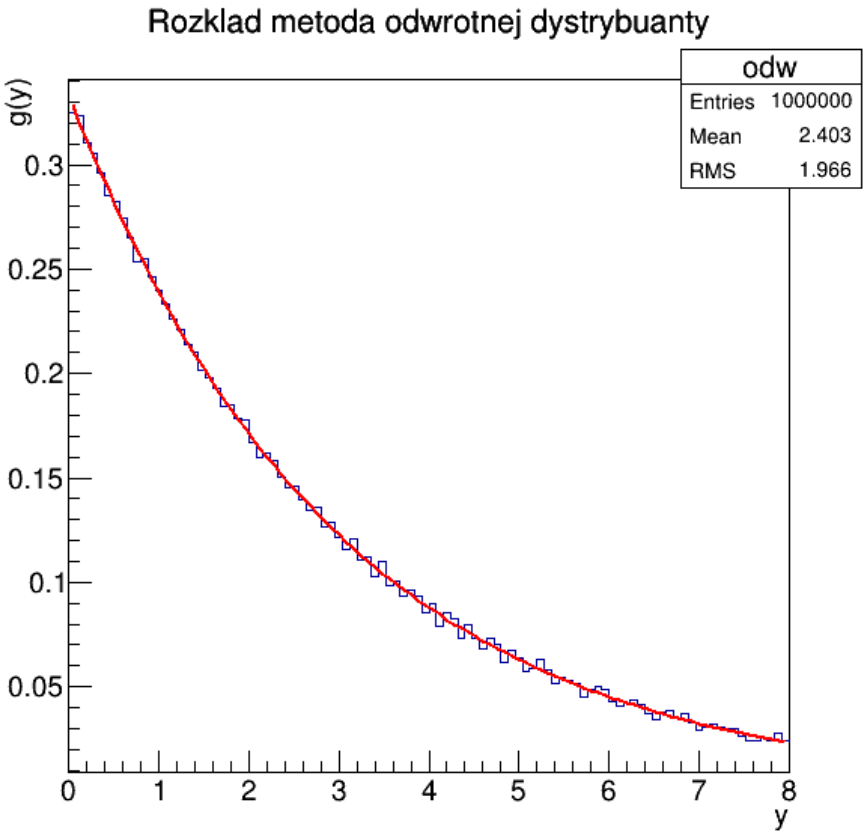
Przykładowe histogramy końcowe



Rysunek 1: histogramy rozkładu i korelacji dla zestawu 1.



Rysunek 2: Histogramy rozkładu i korelacji dla zestawu 3.



Rysunek 3: Rozkład zmiennej losowej wygenerowanej metodą odwrotnej dystrybuanty z funkcji opisanej wzorem 2.