Temat: Metoda Monte Carlo			
Wykonał: Marcin Fabrykowski	Wydział: FiIS	Kierunek Inf. Stos.	Grupa: grupa 3

1. Wstęp
   Metoda Monte Carlo pozwala uprościć obliczenia matematyczne gdy są one skomplikowane, bądź niemożliwe do obliczenia w skończonym czasie, kosztem dokładności obliczeń. Polega ona na losowaniu próbek i sprawdzaniu tych próbek ze znanymi zależnościami. Przykładem mogą być symulacje zderzeń cząstek, gdzie losowane są parametry zderzenia, prędkość, kąt a następnie ze znanych zależności, obliczany jest wynik takiego zderzenia. Zakładając losowość generatora pseudolosowego można powiedzieć ze otrzymane wyniki są prawdopodobne. Innym przykładem może być obliczanie momentu bezwładności bryły. Tutaj również posługujemy się losowymi próbkami punktów w danym obszarze.
   Wyniki otrzymywane tą metodą są szybkie ale cechują się pewnym błędem wynikającym z losowości próbek

2. Wykonanie ćwiczenia
   Celem naszego ćwiczenia będzie obliczenie metodą Monte Carlo momentu bezwładności sześciany w dwóch przypadkach. Gdy oś obrotu przechodzi przez:

   (a) Przekątną bryły
   (b) Jedną z krawędzi

   Moment bezwładności obliczamy ze wzoru: $I = \int\limits_M r^2 dm$. Zakładamy jednorodność bryły: $dm = \sigma dV$ co daje:

   $$I = \sigma \int\limits_\Omega r^2 d\Omega$$

   Do metody Monte Carlo, będziemy potrzebowali wzoru który będzie uwzględniał trafienie próbki w badany obszar.

   $$I = \frac{V\sigma}{N} \sum_{n=1}^{N} r_i^2 \Theta_i$$

   gdzie N to liczba próbek, V objętość obszaru w którym będziemy losować próbki, Θ jest boolowską funkcją sprawdzającą czy dana próbka znajduję się w obszarze Ω będącym obszarem badanej przez nas bryły.
   Dodatkowa wiedza która będzie nam potrzebna, to jak policzyć odległość punktu od prostej.

$$r_i = \sqrt{\frac{|R_1 - R_i|^2 |R_2 - R_1|^2 - [(R_1 - R_i)(R_2 - R_1)]^2}{|R_2 - R_1|^2}}$$

W naszym ćwiczeniu założymy gęstość $\sigma = 1$ oraz liczbę próbek $N = 10^6$, obszar V przyjmujemy jako sześcian o boku $a = 4$ natomiast obszar Ω jako sześcian o boku $b = 2$. Oba sześciany są jednokładne i maja środku w punkcie $(0, 0, 0)$.
Program realizujący powyższe zadanie dla przekątnej bryły:

Listing 1: main.cpp

```
#include <iostream>
#include <cmath>
#include <cstdlib>
using namespace std;
class vec
{
    public:
        vec(float x,float y, float z):
            itsX(x),itsY(y),itsZ(z)
        {
        };
        vec operator-(const vec& co) const
        {
            vec wynik(*this);
            wynik.itsX-=co.itsX;
            wynik.itsY-=co.itsY;
            wynik.itsZ-=co.itsZ;
            return wynik;
        };
        //mnozenie skalarne
        vec operator*(const vec& co) const
        {
            vec wynik(0,0,0);
            wynik.itsX=this->itsX*co.itsX;
            wynik.itsY=this->itsY*co.itsY;
            wynik.itsZ=this->itsZ*co.itsZ;
            return wynik;

        };
        operator float()
        {
            return sqrt(itsX*itsX+itsY*itsY+itsZ*itsZ);
        };
//  private:
        float itsX;
        float itsY;
        float itsZ;
};
float r(const vec& r1, const vec& r2, const vec& ri)
{

    float wynik=0;
    wynik=((float)(r1-ri)*(float)(r1-ri))*((float)(r2-r1)*(float)(r2-r1));
    wynik-=(float)((r1-ri)*(r2-r1))*(float)((r1-ri)*(r2-r1));
    wynik/=((float)(r2-r1)*(float)(r2-r1));
    wynik=sqrt(wynik);
    if(wynik>sqrt(3))
    {
        cout<<ri.itsX<<","<<ri.itsY<<","<<ri.itsZ<<"  -  
            "<<wynik<<endl;
    };
```

```
/*
    float wynik;
    float tmp1=(r1-ri);
    tmp1=tmp1*tmp1;
    float tmp2=(r2-r1);
    tmp2=tmp2*tmp2;
    float tmp3=(vec(r1-ri)*vec(r2-r1));
    tmp3=tmp3*tmp3;
    float tmp4=r2-r1;
    tmp4=tmp4*tmp4;
    wynik=(tmp1*tmp2-tmp3)/tmp4;
    wynik=sqrt(wynik);
*/

    return wynik;
};
bool isIn(float x, float y, float z, float Ox, float Oy,
    float Oz)
{
//  cout<<x<<","<<y<<","<<z<<"\t"<<Ox<<","<<Oy<<","<<Oz<<endl;
    return ((fabs(Ox)>fabs(x))&&
        (fabs(Oy)>fabs(y))&&
        (fabs(Oz)>fabs(z)));
};
float los()
{
    return (float)rand()/RAND_MAX;
};
int main()
{
    srand(time(NULL));
    register int i;
    float Vx=2, Vy=2, Vz=2;
    float Ox=1, Oy=1, Oz=1;
    float I=0;
    float V=8*Ox*Ox*Ox;
    float N=1000;
    const vec r1(-1,-1,-1);
    const vec r2(1,1,1);
    for(i=0;i<N;i++)
    {
        float tmpX=los()*Vx*2-Vx;
        float tmpY=los()*Vy*2-Vy;
        float tmpZ=los()*Vz*2-Vz;
        vec tmp(tmpX,tmpY,tmpZ);
        if(isIn(tmpX,tmpY,tmpZ,Ox,Oy,Oz))
        {
//          cout<<tmpX<<","<<tmpY<<","<<tmpZ<<endl;
            float rTmp=r(r1,r2,tmp);
//          cout<<rTmp<<endl;
            I+=(rTmp*rTmp);
        };
    };
    I*=V;
    I=I/N;
    cout<<I<<endl;
};
```

Natomiast, aby obliczyć dla osi położonej na krawędzi (bądź dowolnego innego), należy odpowiednio zdefiniować wektor osi w linijkach 88 i 89. Po otrzymaniu wyniku obliczamy wariancję oszacowania. Korzystamy ze

wzoru:

$$s^2(N) = \frac{1}{N-1}\left[\sum_{i=1}^{N}(V\cdot\sigma\cdot r_i^2\cdot\Theta_i)^2 - \frac{1}{N}\left(\sum_{i=1}^{2}V\cdot\sigma\cdot r_i^2\cdot\Theta_i\right)^2\right]$$

Natomiast odchylenie standardowe:

$$s(I) = \sqrt{\frac{s^2}{N}}$$

3. Wnioski
   Metoda Monte Carlo cechuje się znacznym przyśpieszenie obliczeń względem obliczania dokładnego trudnych obliczeń. W przypadku obliczeń inżynierskich, gdzie dokładne wyniki nie są wymagane (zbyt dokładne) a ważna jest szybkość ich otrzymania jest ważniejsza, metoda ta doskonale spełnia te wymagania.