Symulacje Monte Carlo

Temat: **Kwantowe Monte Carlo. Metoda wariacyjna**
Imię i nazwisko prowadzącego: **Grzegorz Pawlik**

Wykonawca :	*Gracjan Tokarz* 255531 *W*11
Termin zajec :	*Piatek*, 15 : 15
Data oddania sprawozdania :	04.12.2020*r*
Ocena końcowa:	

Adnotacje dotyczące wymaganych poprawek,oraz daty otrzymania poprawionego sprawozdania

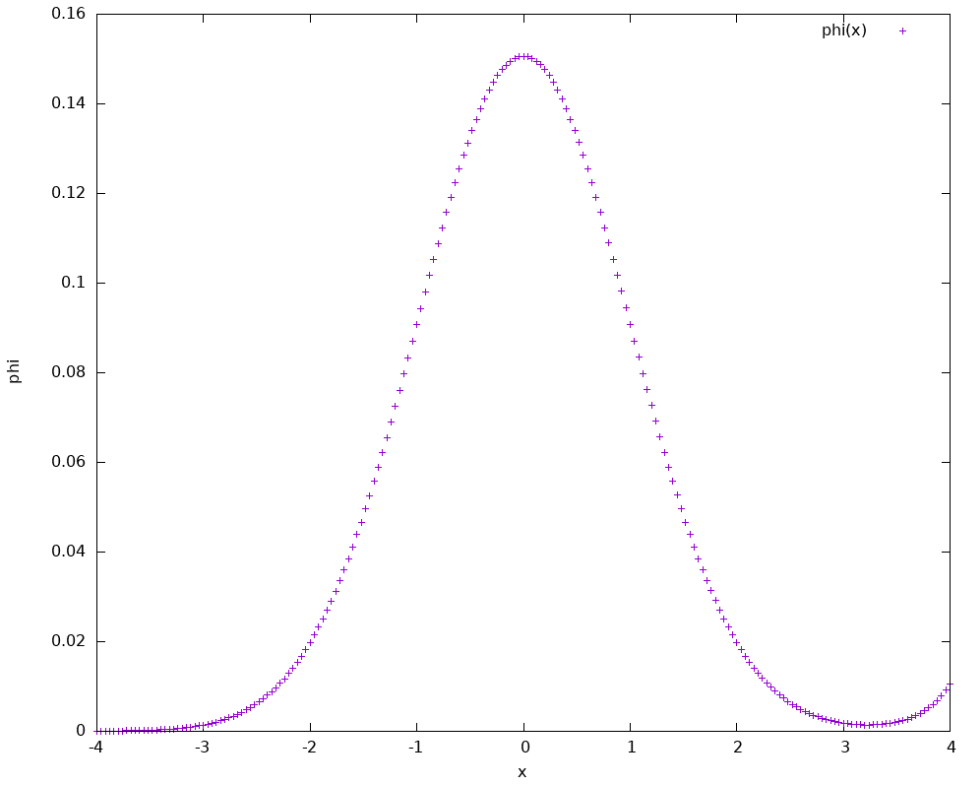
1 Kod źródłowy

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
#define N 200

float calcX(int index, float unit);
float calcV(float x);
void energyToFile(float E, int step);
void phiToFile(float phi, float x, float denominator);
float set(float phi, float dphi);

int main(){
        float unit = (float) 8 / N;
        int steps = 1000000000;
        int index;
        float phiStart = 0.1;
        float dPhi = 0.1;
        float phi[N+1];
        float U = 0, T = 0, denominator = 0, numerator, E, r, phiTrial;
        for (int i = 0; i <= N; i++) {//setting phi and U
                phi[i] = phiStart;
                U += (phi[i] * phi[i] * calcV( calcX(i, unit) ));
                denominator += phi[i] * phi[i];
        }

        for (int i = 1; i < N; i++){
                T += 0.5 * phi[i] * (2 * phi[i] - phi[i-1] - phi[i+1]);
        }
        T +=0.5*phi[0] * (2 * phi[0] - phi[1]);
        T += 0.5*phi[N] * (2 * phi[N] - phi[N-1]);
        T = T / (unit * unit);//kinetic energy
        numerator = U + T;
        E = numerator / denominator;//starting energy
        float deltaPhi, dU,dT, deltaPhiSqr, newE;
```

```
        for (int i = 0; i <= steps; i++){
                r = (float)rand()/RAND_MAX;
                index = rand() % N;
                phiTrial = phi[index] + (r - 0.5)*dPhi;
                deltaPhi = phiTrial - phi[index];
                deltaPhiSqr = phiTrial * phiTrial - phi[index] * phi[index];
                if (index == 0) dT = deltaPhiSqr - deltaPhi * phi[index+1];
                else if (index == N) dT = deltaPhiSqr - deltaPhi * phi[index-1];
                else dT = deltaPhiSqr  - deltaPhi * (phi[index+1] + phi[index-1]);
                dT = dT / (unit*unit);
                dU = deltaPhiSqr * calcV( calcX(index, unit) );
                newE = (numerator + dU + dT) / (denominator + deltaPhiSqr);
                if (newE < E && newE > 0){
                        phi[index] = phiTrial;
                        E = newE;
                        numerator += dU + dT;
                        denominator += deltaPhiSqr;
                }
                if(i%100000==0) energyToFile(E, i);
        }
        for (int i = 0; i <= N;i++){
                phiToFile(phi[i], calcX(i, unit), denominator );
        }
        return 0;
}
void phiToFile(float phi, float x, float denominator){
        FILE *file;
        file = fopen("phi.dat","a+");
        float phiNorm = phi/sqrt(denominator);
        fprintf(file ,"%f\t%f\n",x, phiNorm);
        fclose(file);
}
void energyToFile(float E, int step){
        FILE *file;
        file = fopen("energy.dat","a+");
        fprintf(file , "%f\t%d\n",step, E);
        fclose(file);
}
float calcV (float x){
        return 0.5*x*x;
}
float calcX(int index, float unit){
        if (index > N || index < 0){
                printf("%d - wrong index - out of scope\n", index);
                return 10;
        }
        else return (-4 + (index * unit));
}
```

2 Wykresy

2.1 Wykres unormowanej funkcji falowej



2.2 Wykres unergii układu w toku symulacji