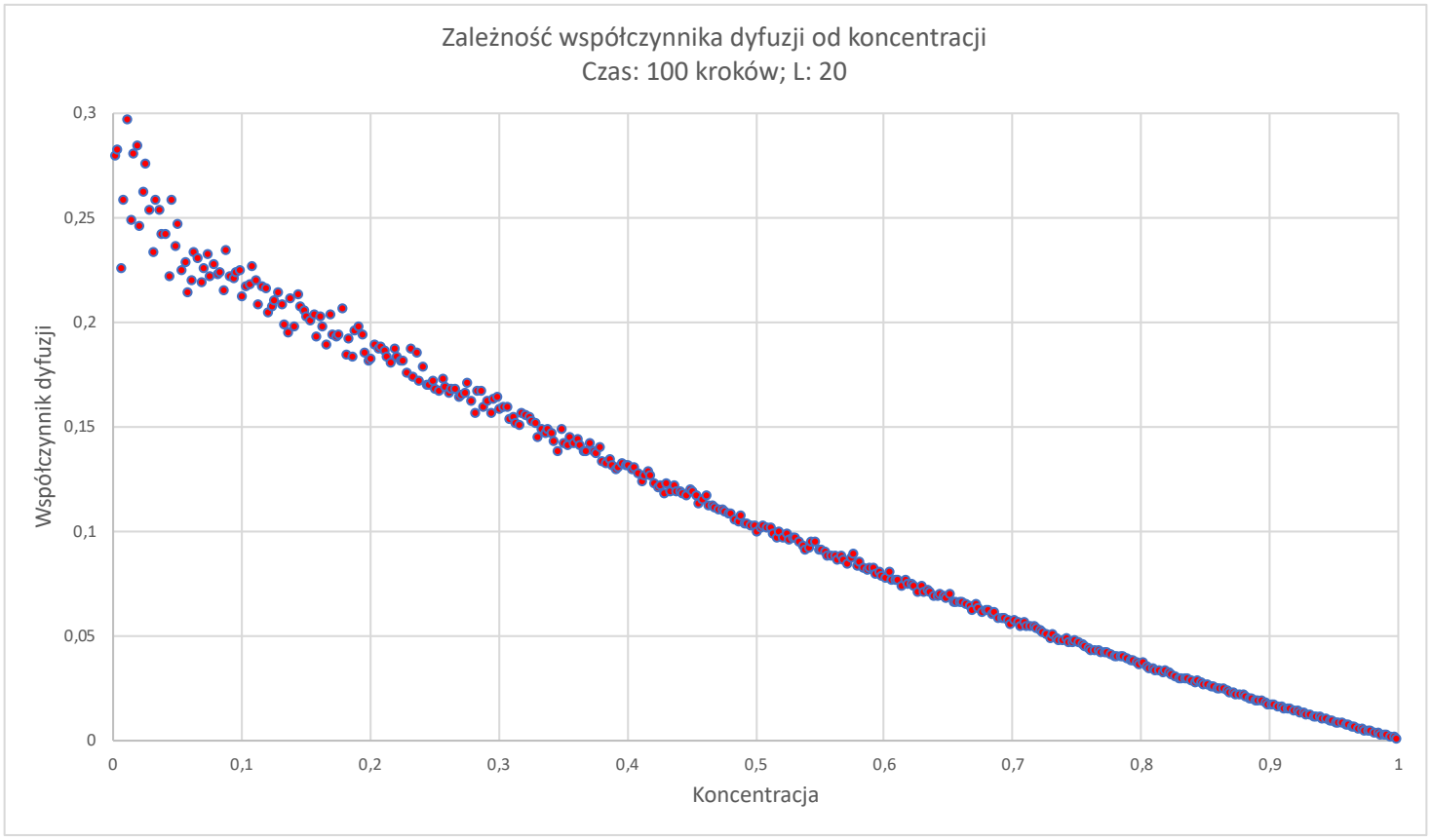
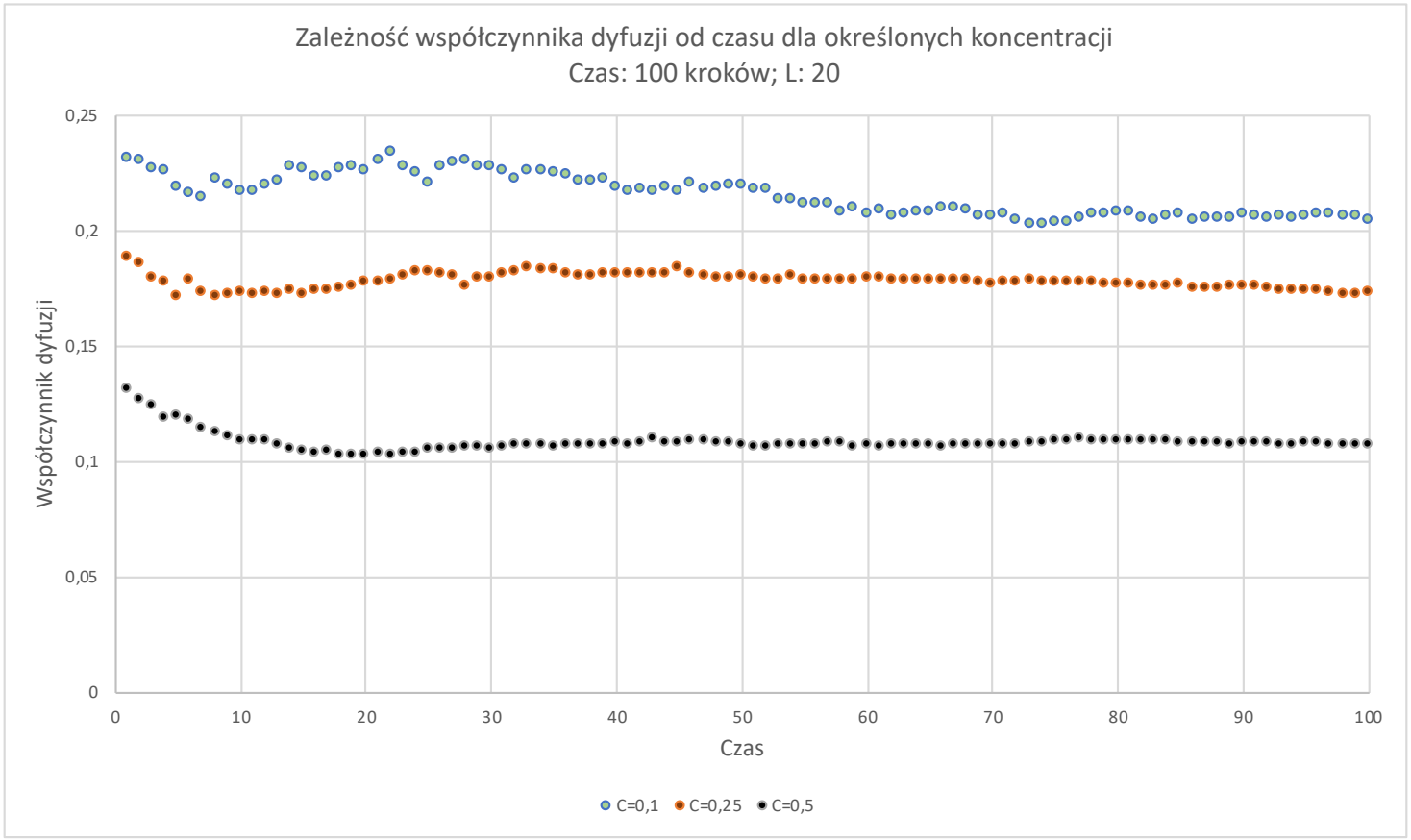

Zależność współczynnika dyfuzji od koncentracji
Czas: 100 kroków; L: 20
Współczynnik dyfuzji
0,3
0,25
0,2
0,15
0,1
0,05
0
0
0,1
0,2
0,3
0,4
0,5
0,6
0,7
0,8
0,9
1
Koncentracja


Zależność współczynnika dyfuzji od czasu dla określonych koncentracji
Czas: 100 kroków; L: 20
Współczynnik dyfuzji
0,25
0,2
0,15
0,1
0,05
0
0
10
20
30
40
50
60
70
80
90
100
Czas
C=0,1
C=0,25
C=0,5

```
//
//  main.cpp
//  Dyfuzja
//
//  Created by Mateusz Bulanda - Gorol on 05/03/2019.
//  Copyright © 2019 Mateusz Bulanda - Gorol. All rights reserved.
//
#include <iostream>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <fstream>

#define L 20        // wielkość planszy

using namespace std;

int MSC = 1000;
int N = 10;                 // liczba cząstek
int s;
bool pozycja [L][L];         // położenie
fstream plik, plik2;


class Czastka
{
private:
    int x, x0, xl, y, y0, yl, d;

public:
    Czastka()       // konstruktor rozmieszcza cząstki w przestrzeni
    {
        do
        {
            x=rand()%(L+1);
            y=rand()%(L+1);
        } while (pozycja[x][y]==1);
        pozycja[x][y]=1;
        plik<<x<<" "<<y<<"    ";  // zapisujemy położenie początkowe
        x0=x;
        xl=0;
        y0=y;
        yl=0;
    }
    void ruch() // wykonujemy ruch
    {
        for (int i=0; i<N; i++)
        {
            pozycja[x][y]=0;
            d=rand()%4;

            switch (d) {
                case 0:
                    if (x==19)
```

```
    {
        if (pozycja[0][y]==0)
        {
            x=0;
            xl++;
            pozycja[x][y]=1;
        }
        else
        {
            pozycja[x][y]=1;
        }
    }
    else
    {
        if (pozycja[x+1][y]==0)
        {
            x=x+1;
            xl++;
            pozycja[x][y]=1;
        }
        else
        {
            pozycja[x][y]=1;
        }
    }
    break;
case 2:
    if(x==0)
    {
        if (pozycja[19][y]==0)
        {
            x=19;
            xl--;
            pozycja[x][y]=1;
        }
        else
        {
            pozycja[x][y]=1;
        }
    }
    else
    {
        if (pozycja[x-1][y]==0)
        {
            x=x-1;
            xl--;
            pozycja[x][y]=1;
        }
        else
        {
            pozycja[x][y]=1;
        }
    }
case 1:
```

```
    if(y==19)
    {
        if (pozycja[x][0]==0)
        {
            y=0;
            yl++;
            pozycja[x][y]=1;
        }
        else
        {
            pozycja[x][y]=1;
        }
    }
    else
    {
        if (pozycja[x][y+1]==0)
        {
            y=y+1;
            yl++;
            pozycja[x][y]=1;
        }
        else
        {
            pozycja[x][y]=1;
        }
    }
    break;
case 3:
    if(y==0)
    {
        if (pozycja[x][19]==0)
        {
            y=19;
            yl--;
            pozycja[x][y]=1;
        }
        else
        {
            pozycja[x][y]=1;
        }
    }
    else
    {
        if (pozycja[x][y-1]==0)
        {
            y=y-1;
            yl--;
            pozycja[x][y]=1;
        }
        else
        {
            pozycja[x][y]=1;
        }
    }
```

```
            }

            plik<<endl<<x<<" "<<y<<"    ";    // zapisujemy kolejne
położenia cząstek
            plik2<<endl<<(xl-x0)*(xl-x0)+(yl-y0)*(yl-y0)<<"
"<<s;  //zapisujemy R i MSC

        }
    }
};

int main()
{
    srand((unsigned) time(NULL));
    plik.open ("położenia.txt", ios::out);
    plik2.open ("przemieszczenia.txt", ios::out);

    for (int i=0; i<L; i++)
    {
        for(int j=0; j<L; j++)
        {
            pozycja[i][j]=0;
        }
    }

    Czastka particle[N];

    for (s=0; s<MSC; s++)
    {
        particle[s].ruch();
    }

    plik.close();
    plik2.close();
    return 0;
}
```