POLITECHNIKA CZĘSTOCHOWSKA

WYDZIAŁ INŻYNIERII MECHANICZNEJ I INFORMATYKI

Projekt na zaliczenie przedmiotu:

Metody Numeryczne

Imię Nazwisko: Artur Matuszczyk

Nr albumu: *135869*

Kierunek: *Informatyka*

Grupa dziek. / lab: *5/9*

Studia: stacjonarne

Poziom studiów: I

Częstochowa, *28.05.2022r.*

Wstęp

Projekt dotyczy zadania wyznaczonego na zaliczenie przedmiotu z Metod Numerycznych. Każdy z uczestników kursu powinien wykorzystać plik oznaczony nazwą swojego numeru indeksu z rozszerzeniem '.dat'. Poniżej zostały wypisane wszystkie podzadania do wykonania w trakcie trwania semestru, gdzie powinniśmy wykorzystać stworzone kody źródłowe napisane na zajęciach laboratoryjnych jak i swoje własne programy.

W moim kodzie źródłowym napisanym w c++ skróciłem wrzucony plik z danymi do tylko potrzebnych danych. W swoich programach wykorzystałem między innymi rozwiązanie Lagrange'a do obliczenia funkcji interpolacyjnej, metodę najmniejszych kwadratów do wyznaczenia funkcji liniowej aproksymacyjnej oraz obliczanie całek metodą trapezów. Pole powierzchni obliczyłem przy użyciu wzoru na długości odległości między odcinkami na układzie kartezjańskim.

Język programowania: C++, R

Środowisko: Visual Studio Code, Octave

Programy: Wykorzystane zostały kody źródłowe z zajęć laboratoryjnych oraz własne programy.

Wykresy: Octave, Excel

Zadania projektu

Dla wszystkich danych:

1. Zwizualizować dane w formie mapy 2D, gdzie kolorami są zaznaczone współrzędne Z.
2. Zwizualizować dane w formie powierzchni 3D.

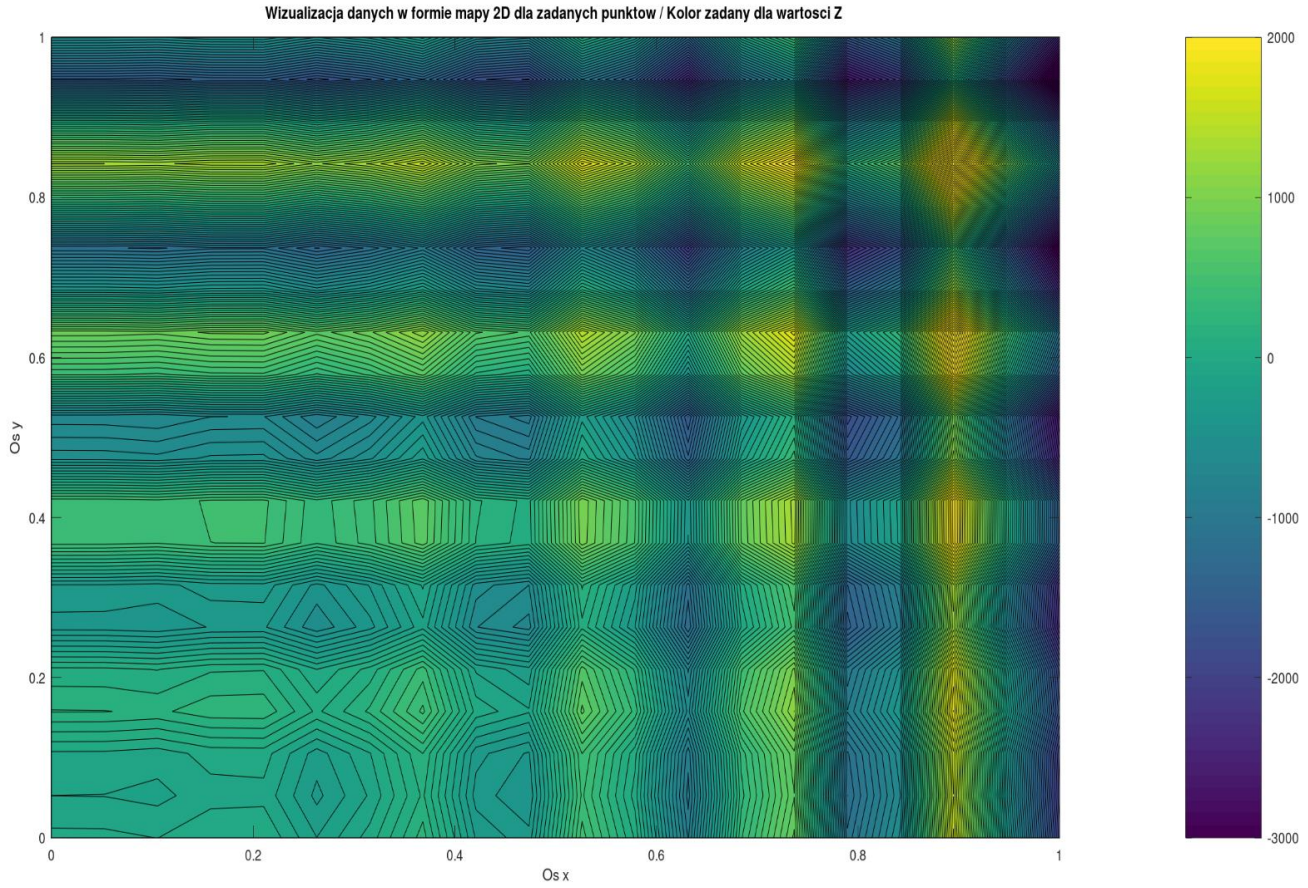
Dla wybranych danych:

1. Wyznaczyć funkcje interpolacyjne dla wybranego wiersza/kolumny.
2. Wyznaczyć funkcje aproksymacyjne dla wybranego wiersza/kolumny.
3. Wyznaczyć średnią arytmetyczną, medianę, oraz odchylenie standardowe dla wybranego wiersza/kolumny.
4. Obliczyć pole powierzchni dla wybranego wiersza/kolumny.
5. Obliczyć całkę z wyznaczonych wcześniej funkcji interpolacyjnych i aproksymacyjnych.
6. Wyznaczyć pochodne cząstkowych dla wybranego wiersza/kolumny.
7. Określić monotoniczność na podstawię wyznaczonych wcześniej pochodnych.

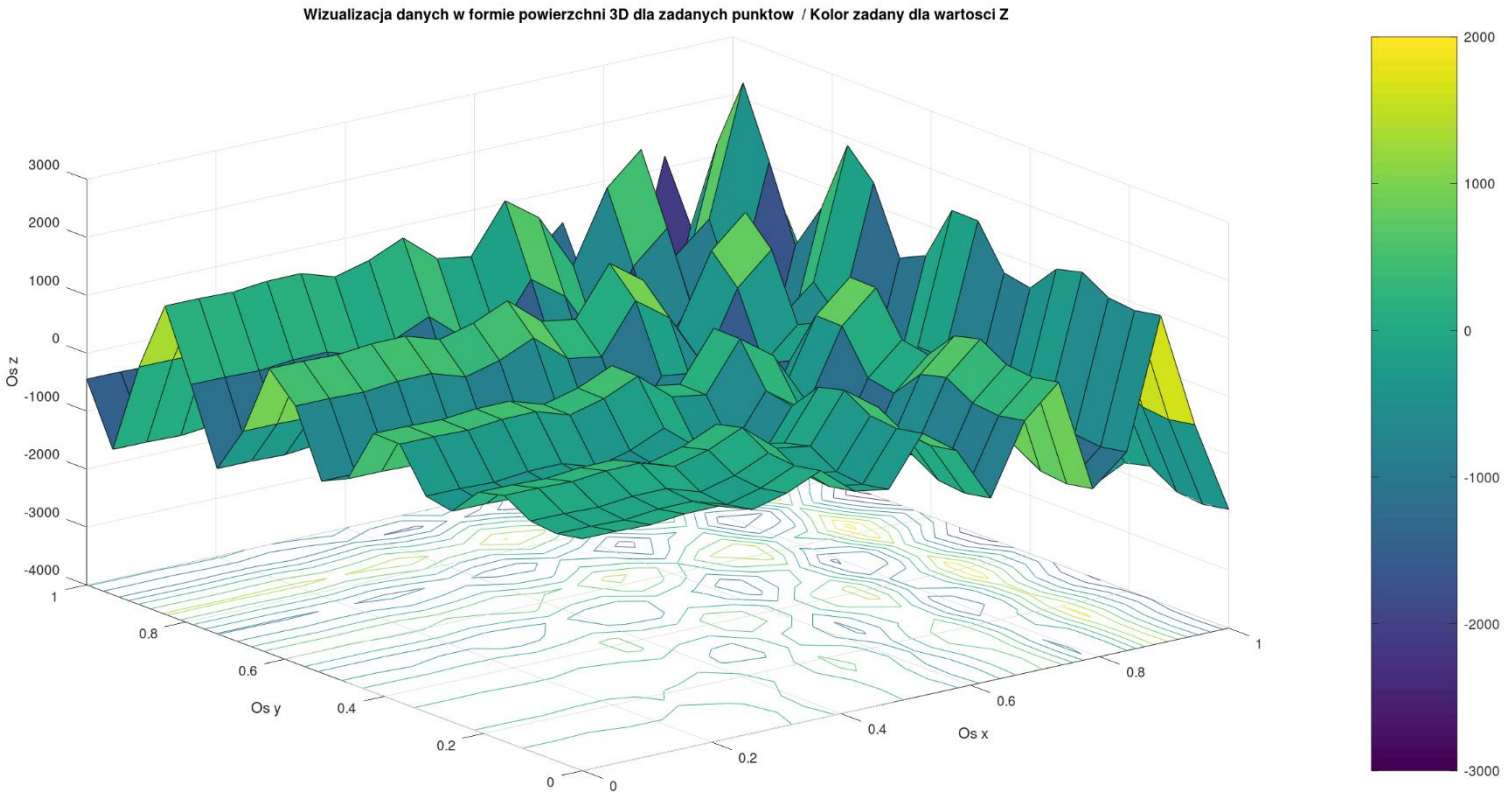
Dodatkowo:

1. Do wyznaczenia wiersza/kolumny stworzenie programu, który wyznaczy nam odchylenie od średniej, gdzie największe odchylenie jest naszym wyborem.

Wizualizacja Mapy 2D



Wizualizacja powierzchni 3D



Wybrany wiersz/kolumna

W moim sposobie wybierałem odchylenie standardowe względem wiersza x/osi OX.
Wybrałem te wartości, ponieważ odchylenia były największe dla x = 0,473684.

Wyglądają one następująco:

X	Y	Z
0.473684	0	-481.417
0.473684	0.0526316	-482.92
0.473684	0.105263	-501.606
0.473684	0.157895	-443.647
0.473684	0.210526	-438.929
0.473684	0.263158	-626.749
0.473684	0.315789	-472.809
0.473684	0.368421	-214.559
0.473684	0.421053	-699.632
0.473684	0.473684	-795.636
0.473684	0.526316	40.6906
0.473684	0.578947	-383.716
0.473684	0.631579	-1335.6
0.473684	0.684211	-128.182
0.473684	0.736842	405.786
0.473684	0.789474	-1562.33
0.473684	0.842105	-1055.31
0.473684	0.894737	1169.76
0.473684	0.947368	-842.536
0.473684	1	-2435.03

Kolorem jasnozielonym zaznaczyłem te wartości, które będę wykorzystywał do tworzenia zadań. Teraz wartości z tabeli y będą równe x, natomiast wartości z tabeli z będą przyjmować wartość y.

Wyniki interpolacji

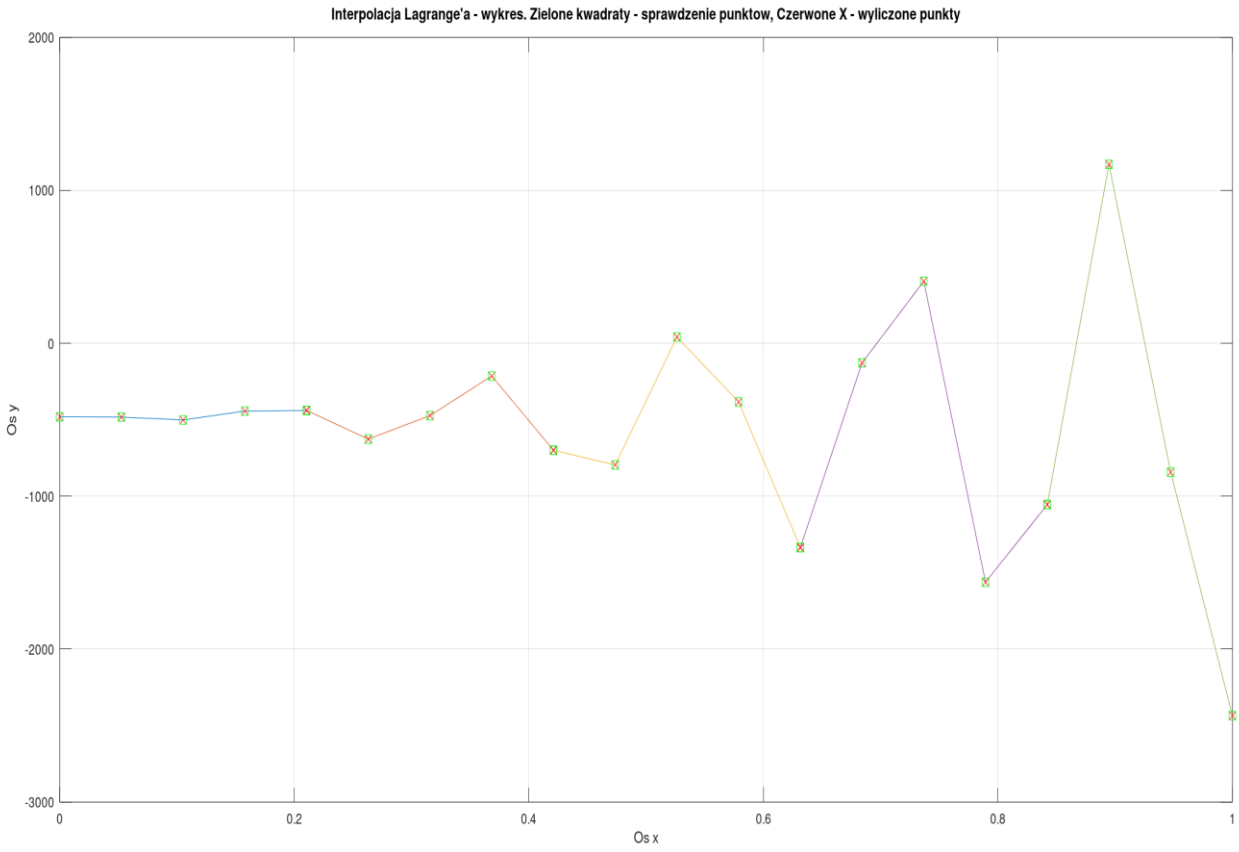
```
octave:1> interpolacja
Wzor na funkcje interpolujaca z przedzialu punktow <0.000000 , 0.210526 >:
W(x) = -1214770.681390x^4 + 490872.254639x^3 + -57052.560952+x^2 + 1791.558857x + -481.417000
Wzor na funkcje interpolujaca z przedzialu punktow <0.210526 , 0.421053 >:
W(x) = -3313195.297398x^4 + 3564879.560243x^3 + -1366838.995262+x^2 + 219748.781200x + -12876.535346
Wzor na funkcje interpolujaca z przedzialu punktow <0.421053 , 0.631579 >:
W(x) = 15890148.110068x^4 + -34287340.361557x^3 + 27456107.843447+x^2 + -9668050.341102x + 1262493.137603
Wzor na funkcje interpolujaca z przedzialu punktow <0.631579 , 0.842105 >:
W(x) = 36956213.174132x^4 + -107123914.757894x^3 + 115857019.643662+x^2 + -55409449.801239x + 9887308.733120
Wzor na funkcje interpolujaca z przedzialu punktow <0.842105 , 1.000000 >:
W(x) = 5323963.155411x^3 + -15055481.634138+x^2 + 14142300.978777x + -4413217.530051
```

Wyniki pozostałych działań

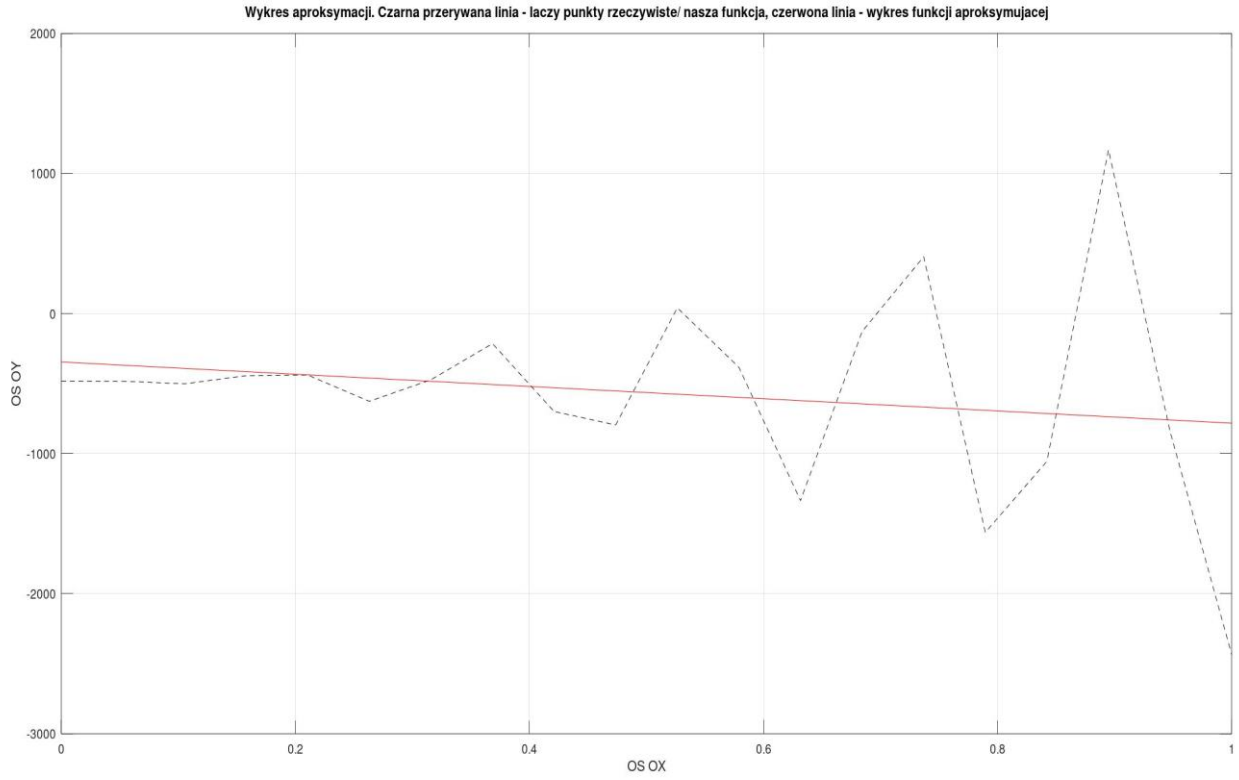
```
Najwieksze srednie odchylenie dla podanych danych wzgledem wierszy OX dla x=0.473684 i wynosi: 714.145521644625


----------Funkcja aproksymujaca----------


Wzor funkcji aproksymujacej: W(x)= -436.9510774x -345.74304


----------Srednia / Mediana / Odchylenie dla wybranego wiersza----------


Srednia: -564.21857
Mediana: -482.1685
Odchylenie standardowe: 714.1455216


----------Pole powierzchni----------


Pole powierzchni: 13522.95357


----------Calkowanie funkcji interpolacyjnych----------


Calka z funkcji interpolacyjnej wynosi: -517.5604153


----------Calkowanie funkcji aproksymujacej----------


Calka z funkcji aproksymujacej wynosi: -564.2185
```

```
------------------------------------------------------------Pochodne czastkowe------------------------------------------------------------


Pochodna z punktu: 0 wynosi : -28.55698858
Pochodna z punktu: 0.0526316 wynosi : -191.7957877
Pochodna z punktu: 0.105263 wynosi : 373.0926419
Pochodna z punktu: 0.157895 wynosi : 595.4323931
Pochodna z punktu: 0.210526 wynosi : -1739.471609
Pochodna z punktu: 0.263158 wynosi : -321.8604828
Pochodna z punktu: 0.315789 wynosi : 3915.810874
Pochodna z punktu: 0.368421 wynosi : -2154.801262
Pochodna z punktu: 0.421053 wynosi : -5520.23978
Pochodna z punktu: 0.473684 wynosi : 7033.07525
Pochodna z punktu: 0.526316 wynosi : 3913.24587
Pochodna z punktu: 0.578947 wynosi : -13074.78031
Pochodna z punktu: 0.631579 wynosi : 2427.55358
Pochodna z punktu: 0.684211 wynosi : 16543.19181
Pochodna z punktu: 0.736842 wynosi : -13624.42644
Pochodna z punktu: 0.789474 wynosi : -13880.43282
Pochodna z punktu: 0.842105 wynosi : 25954.89393
Pochodna z punktu: 0.894737 wynosi : 2021.356032
Pochodna z punktu: 0.947368 wynosi : -34245.55637
Pochodna z punktu: 1 wynosi : -30257.14394

------------------------------------------------------------Monotonicznosc------------------------------------------------------------


Pochodna funkcji w punkcie 0 maleje.
Pochodna funkcji w punkcie 0.0526316 maleje.
Pochodna funkcji w punkcie 0.105263 rosnie.
Pochodna funkcji w punkcie 0.157895 rosnie.
Pochodna funkcji w punkcie 0.210526 maleje.
Pochodna funkcji w punkcie 0.263158 maleje.
Pochodna funkcji w punkcie 0.315789 rosnie.
Pochodna funkcji w punkcie 0.368421 maleje.
Pochodna funkcji w punkcie 0.421053 maleje.
Pochodna funkcji w punkcie 0.473684 rosnie.
Pochodna funkcji w punkcie 0.526316 rosnie.
Pochodna funkcji w punkcie 0.578947 maleje.
Pochodna funkcji w punkcie 0.631579 rosnie.
Pochodna funkcji w punkcie 0.684211 rosnie.
Pochodna funkcji w punkcie 0.736842 maleje.
Pochodna funkcji w punkcie 0.789474 maleje.
Pochodna funkcji w punkcie 0.842105 rosnie.
Pochodna funkcji w punkcie 0.894737 rosnie.
Pochodna funkcji w punkcie 0.947368 maleje.
Pochodna funkcji w punkcie 1 maleje.
```

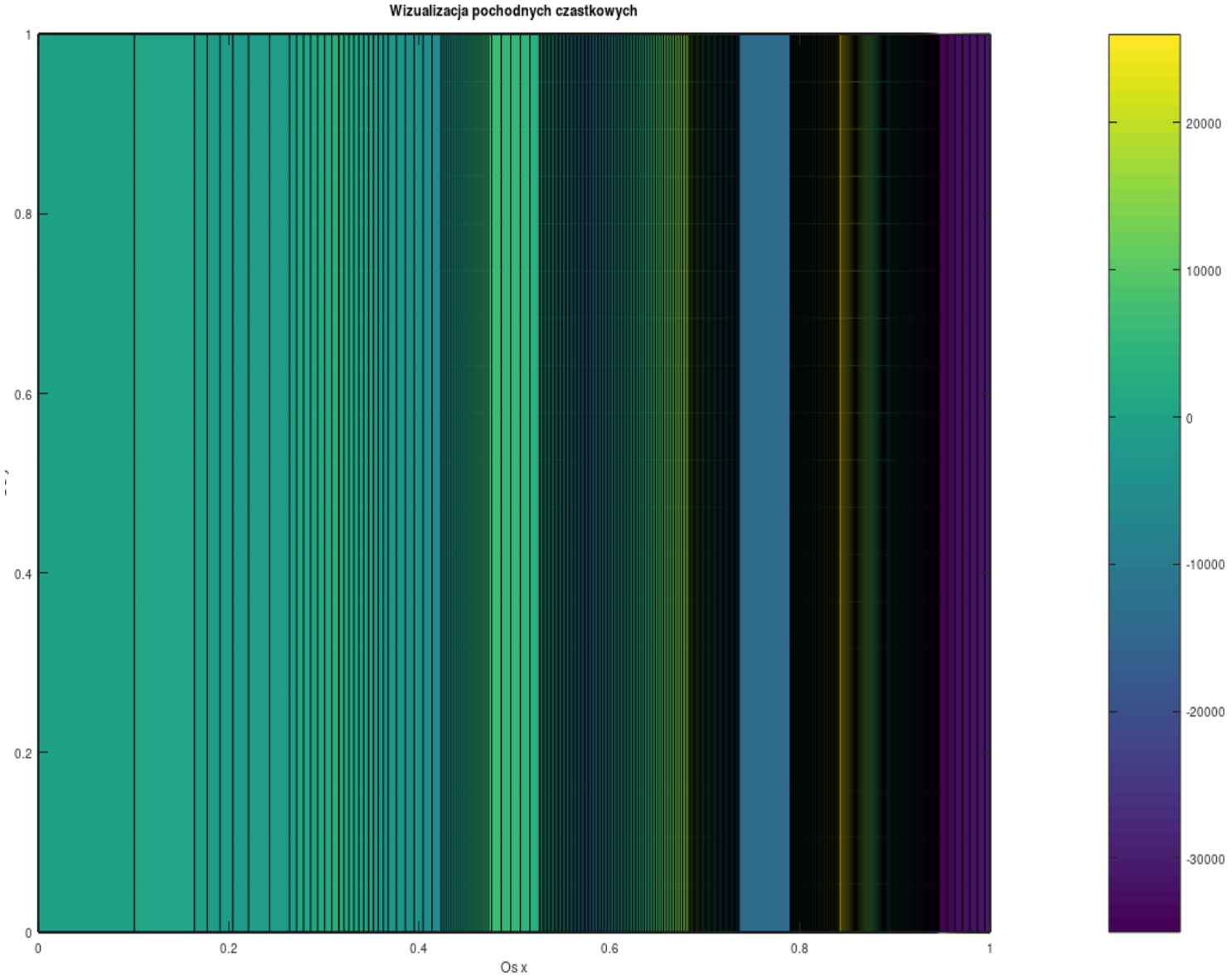
Wykres Funkcji Interpolacyjnych



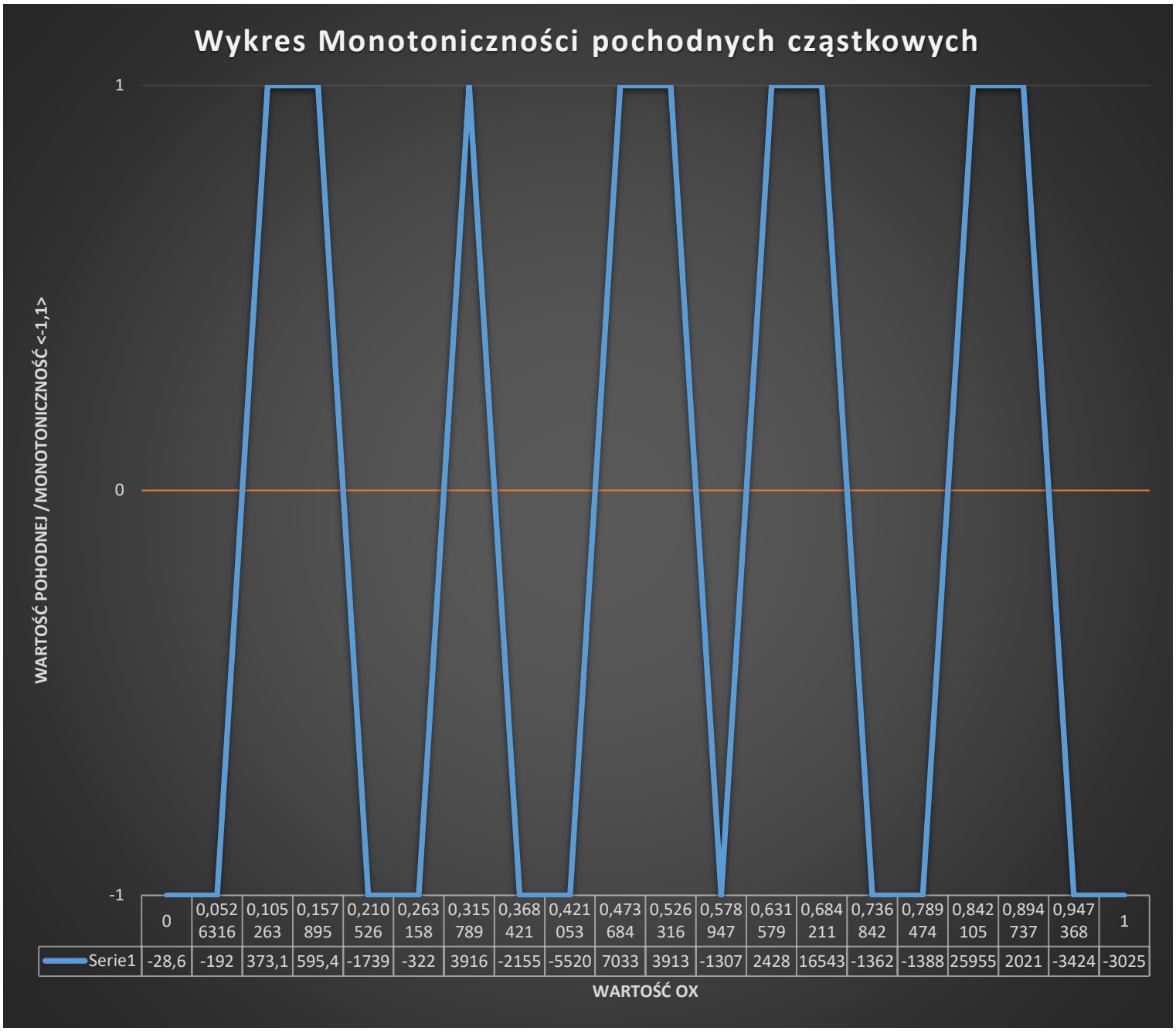
Wykres Funkcji Aproksymującej



Mapa Pochodnych Cząstkowych



Wykres Monotoniczności



	0	0,0526316	0,105263	0,157895	0,210526	0,263158	0,315789	0,368421	0,421053	0,473684	0,526316	0,578947	0,631579	0,684211	0,736842	0,789474	0,842105	0,894737	0,947368	1
Serie1	-28,6	-192	373,1	595,4	-1739	-322	3916	-2155	-5520	7033	3913	-1307	2428	16543	-1362	-1388	25955	2021	-3424	-3025

Kod źródłowy interpolacji

```
w1 = [0,-481.417];
w2 = [0.0526316,-482.92];
w3 = [0.105263,-501.606];
w4 = [0.157895,-443.647];
w5 = [0.210526,-438.929];

w6 = [0.210526,-438.929];
w7 = [0.263158,-626.749];
w8 = [0.315789,-472.809];
w9 = [0.368421,-214.559];
w10 = [0.421053,-699.632];

w11 = [0.421053,-699.632];
w12 = [0.473684,-795.636];
w13 = [0.526316,40.6906];
w14 = [0.578947,-383.716];
w15 = [0.631579,-1335.6];

w16 = [0.631579,-1335.6];
w17 = [0.684211,-128.182];
w18 = [0.736842,405.786];
w19 = [0.789474,-1562.33];
w20 = [0.842105,-1055.31];

w21 = [0.842105,-1055.31];
w22 = [0.894737,1169.76];
w23 = [0.947368,-842.536];
w24 = [1,-2435.03];

function [w] = interpolacja5pkt(w1,w2,w3,w4,w5)
w = [w1;w2;w3;w4;w5];
format short
fi=zeros(1,5);
for i=1:5
    suma=1;
    for j=1:5
        if i != j
```

```
            suma=suma*(w(i,1)-w(j,1));
        end
    endfor
    fi(i)=suma;
endfor
a=zeros(1,5);
for i=1:5
    a(i)=w(i,2)/fi(i);
endfor
wsp=zeros(1,5);
wsp(1)=
a(1)*w(2,1)*w(3,1)*w(4,1)*w(5,1)+a(2)*w(1,1)*w(3,1)*w(4,1)*w(5,1)+a(3)*w(2,1)*
w(1,1)*w(4,1)*w(5,1)+a(4)*w(2,1)*w(3,1)*w(1,1)*w(5,1)+a(5)*w(2,1)*w(3,1)*w(4,
1)*w(1,1);
wsp(2)= -
(a(1)*w(2,1)*w(3,1)*w(4,1)+a(1)*w(5,1)*w(3,1)*w(4,1)+a(1)*w(5,1)*w(2,1)*w(4,1)
+a(1)*w(5,1)*w(2,1)*w(3,1))-
(a(2)*w(1,1)*w(3,1)*w(4,1)+a(2)*w(5,1)*w(3,1)*w(4,1)+a(2)*w(5,1)*w(1,1)*w(4,1)
+a(2)*w(5,1)*w(1,1)*w(3,1))-
(a(3)*w(2,1)*w(1,1)*w(4,1)+a(3)*w(5,1)*w(1,1)*w(4,1)+a(3)*w(5,1)*w(2,1)*w(4,1)
+a(3)*w(5,1)*w(2,1)*w(1,1))-
(a(4)*w(2,1)*w(3,1)*w(1,1)+a(4)*w(5,1)*w(3,1)*w(1,1)+a(4)*w(5,1)*w(2,1)*w(1,1)
+a(4)*w(5,1)*w(2,1)*w(3,1))-
(a(5)*w(2,1)*w(3,1)*w(4,1)+a(5)*w(1,1)*w(3,1)*w(4,1)+a(5)*w(1,1)*w(2,1)*w(4,1)
+a(5)*w(1,1)*w(2,1)*w(3,1));
wsp(3)=
(a(1)*w(2,1)*w(3,1)+a(1)*w(4,1)*w(3,1)+a(1)*w(4,1)*w(2,1)+a(1)*w(5,1)*w(3,1)+
a(1)*w(5,1)*w(2,1)+a(1)*w(5,1)*w(4,1))+(a(2)*w(1,1)*w(3,1)+a(2)*w(4,1)*w(3,1)
+a(2)*w(4,1)*w(1,1)+a(2)*w(5,1)*w(3,1)+a(2)*w(5,1)*w(1,1)+a(2)*w(5,1)*w(4,1))
+(a(3)*w(2,1)*w(1,1)+a(3)*w(4,1)*w(1,1)+a(3)*w(4,1)*w(2,1)+a(3)*w(5,1)*w(1,1)
+a(3)*w(5,1)*w(2,1)+a(3)*w(5,1)*w(4,1))+(a(4)*w(2,1)*w(3,1)+a(4)*w(1,1)*w(3,1
)+a(4)*w(1,1)*w(2,1)+a(4)*w(5,1)*w(3,1)+a(4)*w(5,1)*w(2,1)+a(4)*w(5,1)*w(1,1)
)+(a(5)*w(2,1)*w(3,1)+a(5)*w(4,1)*w(3,1)+a(5)*w(4,1)*w(2,1)+a(5)*w(1,1)*w(3,1
)+a(5)*w(1,1)*w(2,1)+a(5)*w(1,1)*w(4,1));
wsp(4)= -(a(1)*w(2,1)+a(1)*w(3,1)+a(1)*w(4,1)+a(1)*w(5,1))-
(a(2)*w(1,1)+a(2)*w(3,1)+a(2)*w(4,1)+a(2)*w(5,1))-
(a(3)*w(2,1)+a(3)*w(1,1)+a(3)*w(4,1)+a(3)*w(5,1))-
(a(4)*w(2,1)+a(4)*w(3,1)+a(4)*w(1,1)+a(4)*w(5,1))-
(a(5)*w(2,1)+a(5)*w(3,1)+a(5)*w(4,1)+a(5)*w(1,1));
wsp(5)= (a(1)+a(2)+a(3)+a(4)+a(5));
```

```
fprintf("Wzor na funkcje interpolujaca z przedzialu punktow <%f , %f >: ", w(1,1), w(5,1));
disp("");
fprintf("W(x) = %fx^4 + %fx^3 + %f+x^2 + %fx + %f\n",wsp(5),wsp(4), wsp(3),wsp(2),wsp(1));
x=[w(1,1),w(2,1),w(3,1),w(4,1),w(5,1)];
y=zeros(1,5);
for i=1:5
  fi(1) = (x(i)-w(2,1))*(x(i)-w(3,1))*(x(i)-w(4,1))*(x(i)-w(5,1));
  fi(2) = (x(i)-w(1,1))*(x(i)-w(3,1))*(x(i)-w(4,1))*(x(i)-w(5,1));
  fi(3) = (x(i)-w(2,1))*(x(i)-w(1,1))*(x(i)-w(4,1))*(x(i)-w(5,1));
  fi(4) = (x(i)-w(2,1))*(x(i)-w(1,1))*(x(i)-w(3,1))*(x(i)-w(5,1));
  fi(5) = (x(i)-w(2,1))*(x(i)-w(1,1))*(x(i)-w(4,1))*(x(i)-w(3,1));
  y(i) = a(1)*fi(1) + a(2)*fi(2) + a(3)*fi(3) + a(4)*fi(4) + a(5)*fi(5);
endfor
plot(x,y);
grid on;
hold on;
vector_x=[w(1,1),w(2,1),w(3,1),w(4,1),w(5,1)];
vector_y=[0,0,0,0,0];
for i=1:5
  for j=1:5
    vector_y(1,j) = vector_y(1,j)+wsp(i)*w(j,1)^(i-1);
  endfor
endfor
plot(vector_x,vector_y,'rx');
% tutaj nam sie robi sprawdzenie naszych punktow
for i=1:5
  plot(w(i,1),w(i,2),'sg');
endfor
title("Interpolacja Lagrange'a - wykres. Zielone kwadraty - sprawdzenie punktow, Czerwone X - wyliczone punkty");
ylabel("Os y");
xlabel("Os x");
endfunction;
interpolacja5pkt(w1,w2,w3,w4,w5);
interpolacja5pkt(w6,w7,w8,w9,w10);
interpolacja5pkt(w11,w12,w13,w14,w15);
interpolacja5pkt(w16,w17,w18,w19,w20);
 function [w] = interpolacja4pkt(w1,w2,w3,w4)
 w = [w1;w2;w3;w4];
```

```
 fi=zeros(1,4);
 for i=1:4
    suma=1;
    for j=1:4
       if i != j
          suma=suma*(w(i,1)-w(j,1));
       end
    endfor
    fi(i)=suma;
 endfor
 a=zeros(1,4);
 for i=1:4
    a(i)=w(i,2)/fi(i);
 endfor
 wsp=zeros(1,4);
 wsp(1)=-(a(1)*w(4,1)*w(3,1)*w(2,1))-(a(2)*w(4,1)*w(3,1)*w(1,1))-
(a(3)*w(4,1)*w(1,1)*w(2,1))-(a(4)*w(1,1)*w(3,1)*w(2,1));

wsp(2)=+(a(1)*w(3,1)*w(2,1)+a(1)*w(4,1)*w(2,1)+a(1)*w(4,1)*w(3,1))+(a(2)*w(3,
1)*w(1,1)+a(2)*w(4,1)*w(1,1)+a(2)*w(4,1)*w(3,1))+(a(3)*w(1,1)*w(2,1)+a(3)*w(4
,1)*w(2,1)+a(3)*w(4,1)*w(1,1))+(a(4)*w(3,1)*w(2,1)+a(4)*w(1,1)*w(2,1)+a(4)*w(
1,1)*w(3,1));
 wsp(3)=-(a(1)*w(2,1)+a(1)*w(3,1)+a(1)*w(4,1))-
(a(2)*w(1,1)+a(2)*w(3,1)+a(2)*w(4,1))-(a(3)*w(2,1)+a(3)*w(1,1)+a(3)*w(4,1))-
(a(4)*w(2,1)+a(4)*w(3,1)+a(4)*w(1,1));
 wsp(4)=a(1)+a(2)+a(3)+a(4);
 fprintf("Wzor na funkcje interpolujaca z przedzialu punktow <%f , %f >: ",
w(1,1), w(4,1));
 disp("");
 fprintf("W(x) = %fx^3 + %f+x^2 + %fx + %f\n",wsp(4), wsp(3),wsp(2),wsp(1));
 x=[w(1,1),w(2,1),w(3,1),w(4,1)];
 y=zeros(1,4);
 for i=1:4
   fi(1) = (x(i)-w(2,1))*(x(i)-w(3,1))*(x(i)-w(4,1));
   fi(2) = (x(i)-w(1,1))*(x(i)-w(3,1))*(x(i)-w(4,1));
   fi(3) = (x(i)-w(2,1))*(x(i)-w(1,1))*(x(i)-w(4,1));
   fi(4) = (x(i)-w(2,1))*(x(i)-w(1,1))*(x(i)-w(3,1));
   y(i) = a(1)*fi(1) + a(2)*fi(2) + a(3)*fi(3) + a(4)*fi(4);
 endfor
 plot(x,y);
 grid on;
```

```
 hold on;
 vector_x=[w(1,1),w(2,1),w(3,1),w(4,1)];
 vector_y=[0,0,0,0];
 for i=1:4
   for j=1:4
     vector_y(1,j) = vector_y(1,j)+wsp(i)*w(j,1)^(i-1);
   endfor
 endfor
 plot(vector_x,vector_y,'rx');
 % tutaj nam sie robi sprawdzenie naszych punktow
 for i=1:4
   plot(w(i,1),w(i,2),'gs');
 endfor
title("Interpolacja Lagrange'a - wykres. Zielone kwadraty - sprawdzenie
punktow, Czerwone X - wyliczone punkty");
 ylabel("Os y");
 xlabel("Os x");
 endfunction;
interpolacja4pkt(w21,w22,w23,w24);
```

Kod źródłowy pozostałych obliczeń

```
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
double sredniaArytm(const double z[]){
    double srednia=0.00;
    for(unsigned int i=0; i<20; i++){
        srednia+=z[i];
    }
    return srednia/20;
}
//------------------------------Koniec funkcji - Wyznaczanie X-------------------------//
void zamien(double &x, double &y){
    double pom=y;
    y=x;
    x=pom;
}
double sortowanieMediana(double y[]){
    for(unsigned int i=0;i<20;i++){
        for(unsigned int j=i+1; j<20; j++ ){
            if(y[i]>y[j]) zamien(y[i],y[j]);
        }
    }
    return (y[9]+y[10])/2.00;
}
//------------------------------Koniec funkcji -
Srednia/Mediana/Odchylenie_Standardowe-------------------------//
double odlegloscPunktow(double xa, double xb, double ya, double yb){
    return sqrt(pow(xb-xa,2)+pow(yb-ya,2));
}
//------------------------------Koniec funkcji - Pole Powierzchni-------------------------
-//
double calkowanieInterpolacja(double tabX[], double x){
        if(x<=0.210526) return
tabX[0]*pow(x,4)+tabX[1]*pow(x,3)+tabX[2]*pow(x,2)+tabX[3]*pow(x,1)+tabX[4]
*1;
```

```
        else if(x>0.210526 && x<=0.421053) return
tabX[5]*pow(x,4)+tabX[6]*pow(x,3)+tabX[7]*pow(x,2)+tabX[8]*pow(x,1)+tabX[9]
*1;
        else if(x>0.421053 && x<=0.631579) return
tabX[10]*pow(x,4)+tabX[11]*pow(x,3)+tabX[12]*pow(x,2)+tabX[13]*pow(x,1)+ta
bX[14]*1;
        else if(x>0.631579 && x<=0.842105) return
tabX[15]*pow(x,4)+tabX[16]*pow(x,3)+tabX[17]*pow(x,2)+tabX[18]*pow(x,1)+ta
bX[19]*1;
        else return
tabX[20]*pow(x,3)+tabX[21]*pow(x,2)+tabX[22]*pow(x,1)+tabX[23]*1;
    }
double calkowanieAproksymacja(double tabX[], double x ){
        return tabX[0]*x+tabX[1];
    }
//-------------------------------Koniec funkcji - Calkowanie funkcji-----------------------
---//
double wyliczPochodna(double y1, double y2, double z1, double z2){
    return ((z1-z2)/(y1-y2));
}
//-------------------------------Koniec funkcji - pochodne czastkowe---------------------
------//
void monotonicznosc(double pochodne[], double x[]){
    for(unsigned int i=0; i<20; i++){
        if(pochodne[i]<0) cout<<"Pochodna funkcji w punkcie "<<x[i]<<" maleje.\n";
        else cout<<"Pochodna funkcji w punkcie "<<x[i]<<" rosnie.\n";
    }
    cout<<endl<<endl;
}
//-------------------------------Koniec funkcji - monotonicznosc-------------------------
//
void przerywnik(string tekst){
    cout<<endl<<endl;
    cout<<setw(80)<<setfill('-')<<tekst<<"-------------------------------------------------
-------"<<endl;
    cout<<endl<<endl;
}
int main(int argc, char** argv){
    double mainZ[400]={4.8, 3.29722, -15.3889, 42.5705, 47.2888, -140.531,
13.4088, 271.659, -213.414, -309.418, 526.908, 102.501, -849.382, 358.036,
892.003, -1076.11, -569.089, 1655.97, -356.319, -1948.81, -78.1213, -79.6241,
```

-98.3103, -40.3509, -35.6325, -223.452, -69.5125, 188.737, -296.335, -392.34, 443.987, 19.5801, -932.303, 275.115, 809.082, -1159.03, -652.011, 1573.05, -439.24, -2031.73, -27.4166, -28.9194, -47.6055, 10.3539, 15.0723, -172.748, -18.8077, 239.442, -245.631, -341.635, 494.691, 70.2849, -881.598, 325.819, 859.787, -1108.33, -601.306, 1623.76, -388.535, -1981.03, 244.097, 242.594, 223.908, 281.867, 286.586, 98.7656, 252.706, 510.955, 25.8826, -70.1216, 766.205, 341.798, -610.085, 597.333, 1131.3, -836.813, -329.792, 1895.27, -117.022, -1709.51, 130.457, 128.954, 110.268, 168.228, 172.946, -14.8741, 139.066, 397.316, -87.7571, -183.761, 652.565, 228.158, -723.725, 483.693, 1017.66, -950.453, -443.432, 1781.63, -230.662, -1823.15, -365.03, -366.533, -385.219, -327.259, -322.541, -510.361, -356.421, -98.1712, -583.244, -679.248, 157.078, -267.328, -1219.21, -11.7938, 522.173, -1445.94, -938.919, 1286.15, -726.149, -2318.64, -272.28, -273.782, -292.468, -234.509, -229.791, -417.611, -263.671, -5.421, -490.494, -586.498, 249.829, -174.578, -1126.46, 80.9564, 614.924, -1353.19, -846.169, 1378.9, -633.398, -2225.89, 456.009, 454.506, 435.82, 493.78, 498.498, 310.678, 464.618, 722.868, 237.795, 141.791, 978.117, 553.711, -398.172, 809.245, 1343.21, -624.901, -117.88, 2107.18, 94.8904, -1497.6, 468.778, 467.275, 448.589, 506.548, 511.267, 323.447, 477.387, 735.636, 250.564, 154.56, 990.886, 566.479, -385.404, 822.014, 1355.98, -612.132, -105.111, 2119.95, 107.659, -1484.83, -481.417, -482.92, -501.606, -443.647, -438.929, -626.749, -472.809, -214.559, -699.632, -795.636, 40.6906, -383.716, -1335.6, -128.182, 405.786, -1562.33, -1055.31, 1169.76, -842.536, -2435.03, -689.361, -690.864, -709.55, -651.591, -646.872, -834.692, -680.752, -422.503, -907.575, -1003.58, -167.253, -591.66, -1543.54, -336.125, 197.842, -1770.27, -1263.25, 961.814, -1050.48, -2642.97, 439.195, 437.692, 419.006, 476.965, 481.683, 293.863, 447.803, 706.053, 220.98, 124.976, 961.303, 536.896, -414.987, 792.43, 1326.4, -641.715, -134.695, 2090.37, 78.0757, -1514.41, 915.256, 913.753, 895.067, 953.027, 957.745, 769.925, 923.865, 1182.11, 697.042, 601.038, 1437.36, 1012.96, 61.0746, 1268.49, 1802.46, -165.654, 341.367, 2566.43, 554.137, -1038.35, -334.634, -336.137, -354.823, -296.864, -292.145, -479.965, -326.025, -67.7754, -552.848, -648.852, 187.474, -236.933, -1188.82, 18.602, 552.569, -1415.54, -908.523, 1316.54, -695.753, -2288.24, -1148.77, -1150.27, -1168.95, -1110.99, -1106.28, -1294.1, -1140.16, -881.907, -1366.98, -1462.98, -626.657, -1051.06, -2002.95, -795.529, -261.562, -2229.67, -1722.65, 502.41, -1509.88, -3102.37, 132.884, 131.381, 112.695, 170.654, 175.372, -12.4477, 141.492, 399.742, -85.3306, -181.335, 654.992, 230.585, -721.298, 486.119, 1020.09, -948.026, -441.006, 1784.06, -228.235, -1820.73, 1322.09, 1320.58, 1301.9, 1359.86, 1364.57, 1176.75, 1330.69, 1588.94, 1103.87, 1007.87, 1844.19, 1419.79, 467.904, 1675.32, 2209.29, 241.175, 748.196, 2973.26, 960.967, -631.524, 99.548, 98.0452, 79.3591, 137.318, 142.037, -45.7832, 108.157, 366.407, -118.666, -

```
214.67, 621.656, 197.249, -754.634, 452.784, 986.751, -981.362, -474.341,
1750.72, -261.571, -1854.06, -1493.39, -1494.89, -1513.58, -1455.62, -1450.9,
-1638.72, -1484.78, -1226.53, -1711.6, -1807.61, -971.281, -1395.69, -2347.57,
-1140.15, -606.186, -2574.3, -2067.28, 157.786, -1854.51, -3447, -447.915, -
449.418, -468.104, -410.144, -405.426, -593.246, -439.306, -181.056, -666.129,
-762.133, 74.1933, -350.213, -1302.1, -94.6788, 439.288, -1528.82, -1021.8,
1203.26, -809.034, -2401.52};
    double
x[20]={0,0.0526316,0.105263,0.157895,0.210526,0.263158,0.315789,0.368421
,0.421053,0.473684,0.526316,0.578947,0.631579,0.684211,0.736842,0.78947
4,0.842105,0.894737,0.947368,1};
    double sredniaAryt[20]={}, secondZ[20]={}, odchylenia[20]={},
szukanieX[20][2]={},  y[20]={}, yy[20]={};
    unsigned k=0, pom=0;
    for(unsigned int i=0; i<400; i+=20){
        for(unsigned int j=0; j<20; j++){
            secondZ[j]=mainZ[i+j];
        }
        sredniaAryt[k]=sredniaArytm(secondZ);
        for(unsigned int m=0; m<20; m++){
            secondZ[m]=pow(secondZ[m]-sredniaAryt[k],2);   //Policzenie roznicy
kwadratow do wzoru
        }
        odchylenia[k]=sqrt(sredniaArytm(secondZ));
        szukanieX[k][0]=odchylenia[k];
        szukanieX[k][1]=x[k];
        k++;
    } //Wyznaczenie maksymalnego odchylenia standardowego
        double max=szukanieX[0][0];
        for(unsigned int i=1;i<20;i++){
            if(max<szukanieX[i][0]){
                max=szukanieX[i][0];
                pom=i;
            }
        }
        cout<<"Najwieksze srednie odchylenie dla podanych danych wzgledem
wierszy OX dla x="<<szukanieX[pom][1]<<" i wynosi:
"<<setprecision(15)<<max<<endl;
        cout<<setprecision(10);
        unsigned int pomocnicza1=0, wartoscI=20*pom;
            for(unsigned int j=0;j<20;j++){
```

```
            if(pom==j){
                for(unsigned int i=wartoscI;i<wartoscI+20;i++){
                y[pomocnicza1]=mainZ[i];
                yy[pomocnicza1]=mainZ[i];
                pomocnicza1++;
                }
            }
        }
    //-------------------------Wyznaczanie funkcji aproksymujacej - Metoda Najmniejszych Kwadratow ------------------------------------//
        double suma_x=0.00, suma_y=0.00, iloczynXY=0.00, sumaKwadratowX=0.00;
        for(unsigned int i=0; i<20; i++){
            suma_x+=x[i]; //Suma X
            suma_y+=y[i]; //Suma y
            iloczynXY+=(x[i]*y[i]);
            sumaKwadratowX+=x[i]*x[i];
        }
        double aa = ((20*iloczynXY)-(suma_x*suma_y))/(20*sumaKwadratowX-(suma_x*suma_x));
        double bb = (0.05)*(suma_y-(aa*suma_x));
        przerywnik("Funkcja aproksymujaca");
        cout<<"Wzor funkcji aproksymujacej: W(x)= "<<aa<<"x ";
        if(bb>0.00) cout<<"+ "<<bb<<endl;
        else cout<<bb<<endl;
    //-------------------------Wyznaczanie sredniej/mediany/odchylenia standardowego-------------------------------------//
        przerywnik("Srednia / Mediana / Odchylenie dla wybranego wiersza");
        cout<<"Srednia: "<<sredniaAryt[pom]<<endl;
        //Mediana
        cout<<"Mediana: "<<sortowanieMediana(yy)<<endl;
        cout<<"Odchylenie standardowe: "<<max <<endl;
    //-------------------------Pole Powierzchni-------------------------------------//
        double polePowierzchni=0.00;
        przerywnik("Pole powierzchni");
        for(unsigned int i=1; i<20; i++){
            polePowierzchni+=odlegloscPunktow(x[i-1],x[i],y[i-1],y[i]);
        }
        cout<<"Pole powierzchni: "<<polePowierzchni<<endl;
    //-------------------------Całkowanie funkcji interpolacyjnych ------------------------------------//
```

```
        double xFourToZerox4[24]={-1214770.681390, 490872.254639, -
57052.560952, 1791.558857, -481.417000,-3313195.297398, 3564879.560243,
-1366838.995262, 219748.781200, -12876,15890148.110068, -
34287340.361557, 27456107.843447, -9668050.341102, 1262493.137603,
36956213.174132, -107123914.757894,  115857019.643662, -
55409449.801239,  9887308.733120, 5323963.155411,-
15055481.634138,14142300.978777,-4413217.530051};
        //Macierz pomocnicza do sumowania wartosci calek
        const unsigned int n=10000;
        double doObliczenWartosci[10001]={}, a=0.00, b=1.00,
calkaInterpolacja=0.00,p=(double)(b-a)/n,pomx=0.00, pomx1=0.00;
        doObliczenWartosci[0]=0;
        //Obliczenie wartosci kazdego 'x' do metody trapezow
        for(unsigned int i=1; i<=n; i++)
doObliczenWartosci[i]=doObliczenWartosci[i-1]+p;
        //Wyznaczanie calki z danego x'a podstawiajac pod funkcje interpolacyjne
        for(unsigned int i=0; i<=n-1; i++){
            pomx=calkowanieInterpolacja(xFourToZerox4,doObliczenWartosci[i]);
            pomx1=calkowanieInterpolacja(xFourToZerox4,
doObliczenWartosci[i+1]);
            calkaInterpolacja+=(((pomx+pomx1)/2)*p);
        }
        przerywnik("Calkowanie funkcji interpolacyjnych");
        cout<<"Calka z funkcji interpolacyjnej wynosi: "<<calkaInterpolacja<<"\n";
    //-----------------------------------------------Całkowanie funkcji aproksymacyjnej --
-----------------------------------------------//
        //Do obliczeń posłużyłem się już istniejącą macierzą z wyliczania całki z
funkcji interpolacji oraz innymi zmiennymi
        double xApro[2]={-436.951,-345.743},calkaAproksymacja=0.00;
        pomx=0.00, pomx1=0.00;
        for(unsigned int i=0; i<=n-1; i++){
            pomx=calkowanieAproksymacja(xApro,doObliczenWartosci[i]);
            pomx1=calkowanieAproksymacja(xApro, doObliczenWartosci[i+1]);
            calkaAproksymacja+=((pomx+pomx1)/2)*p;
        }
        przerywnik("Calkowanie funkcji aproksymujacej");
        cout<<"Calka z funkcji aproksymujacej wynosi:
"<<calkaAproksymacja<<"\n";
    //-----------------------------------------------Wyznaczanie pochodnych
cząstkowych i monotoniczność----------------------------------------------------//
        double pochodne[20]={};
```

```
	pochodne[0]=((y[1]-y[0])/(x[1]-x[0])), pochodne[19]=((y[19]-y[18])/(x[19]-
x[18]));
	for(unsigned int i=1; i<19; i++){
	    pochodne[i]=wyliczPochodna(x[i+1],x[i-1],y[i+1],y[i-1]);
	}
	przerywnik("Pochodne czastkowe");
	for(unsigned int i=0; i<20;i++) cout<<"Pochodna z punktu: "<<x[i]<<"
wynosi : "<<pochodne[i]<<"\n";
	przerywnik("Monotonicznosc");
	monotonicznosc(pochodne,x);
	return 0;
}
```

Koniec

Wykonał: Artur Matuszczyk