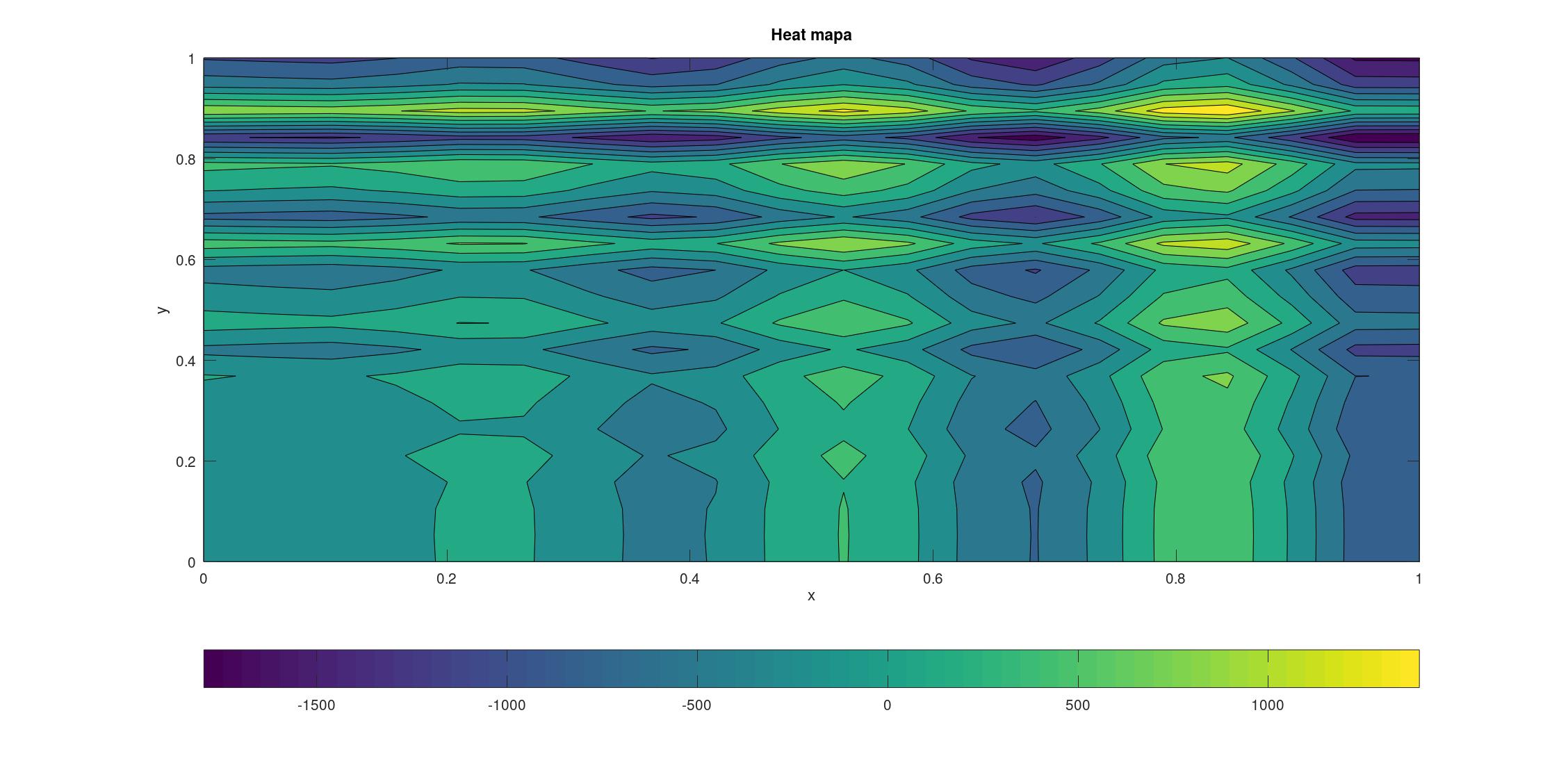
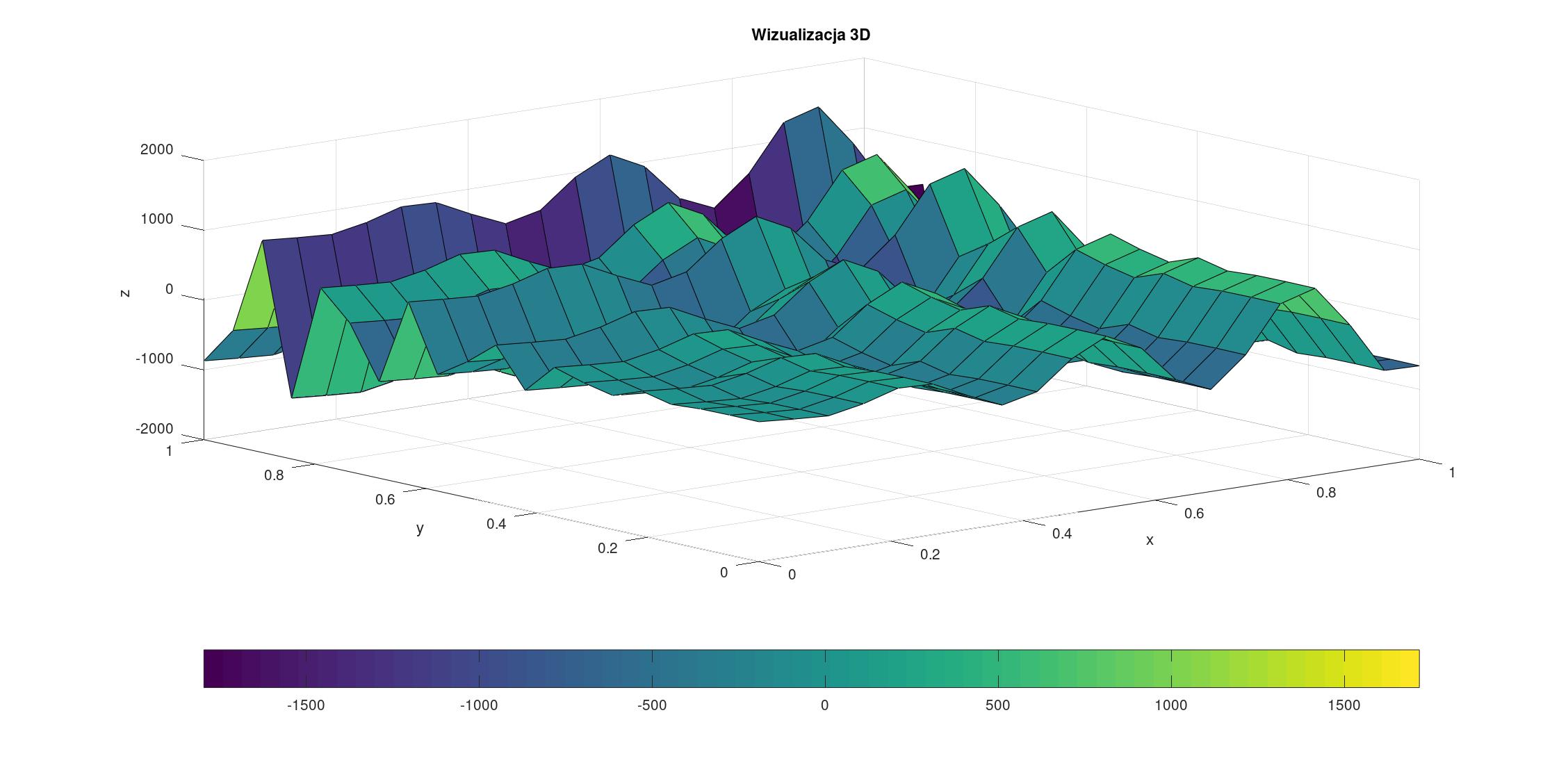
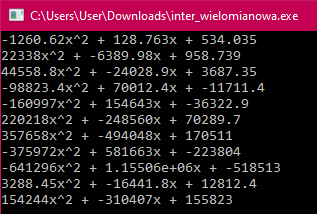
**Przybliżone metody rozwiązywania zagadnień początkowo-brzegowych**

**Kacper Szczerba.gr.5 nr\_indeksu:135192  
Informatyka, rok I, semestr II,**

**1.Wizualizacja danych w formie mapy 2D, kolory dla współrzędnej Z**



**Wizualizacje wykonałem w programie Octave na podstawie komend, które miałem na labolatoriach.**  
  
**2.Wizualizacja danych w formie powierzchni 3D zbudowanej z trójkątów / kwadratów lub innej metody  
  
Wizualizacje wykonałem w programie Octave na podstawie komend, które miałem na  
labolatoriach.  
  
3.Wyznaczyć funkcje interpolacyjne** **dla wybranego wiersza lub kolumny z siatki.  
  
  
  
Kod programu:  
  
#include <iostream>**

**#include <math.h>**

**long 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, 0.789474, 0.842105, 0.894737, 0.947368, 1 };**

**long double z[20][20] = {**

**4.1,7.38498,3.68669,-23.2389,73.5913,-80.2073,12.8567,138.986,-282.392,275.12,-59.3128,-330.982,617.399,-616.6,130.887,542.012,-1129.78,1040.27,-342.569,-870.887,**

**-35.2535,-31.9686,-35.6668,-62.5925,34.2378,-119.561,-26.4968,99.6323,-321.746,235.767,-98.6664,-370.336,578.045,-655.953,91.5334,502.659,-1169.14,1000.91,-381.923,-910.241,**

**-66.3381,-63.0531,-66.7514,-93.677,3.15324,-150.645,-57.5814,68.5477,-352.83,204.682,-129.751,-401.42,546.961,-687.038,60.4488,471.574,-1200.22,969.829,-413.007,-941.325,**

**28.1257,31.4107,27.7124,0.786774,97.617,-56.1816,36.8824,163.011,-258.367,299.146,-35.2871,-306.956,641.425,-592.574,154.913,566.038,-1105.76,1064.29,-318.544,-846.862,**

**175.262,178.547,174.849,147.924,244.754,90.9551,184.019,310.148,-111.23,446.283,111.85,-159.82,788.561,-445.437,302.049,713.175,-958.622,1211.43,-171.407,-699.725,**

**158.586,161.871,158.172,131.247,228.077,74.2784,167.342,293.471,-127.907,429.606,95.1729,-176.496,771.885,-462.114,285.373,696.498,-975.298,1194.75,-188.083,-716.402,**

**-82.6359,-79.3509,-83.0492,-109.975,-13.1446,-166.943,-73.8792,52.2499,-369.128,188.385,-146.049,-417.718,530.663,-703.335,44.151,455.276,-1216.52,953.531,-429.305,-957.623,**

**-298.115,-294.83,-298.529,-325.454,-228.624,-382.423,-289.359,-163.23,-584.608,-27.0949,-361.528,-633.197,315.184,-918.815,-171.328,239.797,-1432,738.051,-644.785,-1173.1,**

**-183.87,-180.585,-184.284,-211.209,-114.379,-268.178,-175.114,-48.9845,-470.363,87.1501,-247.283,-518.952,429.429,-804.57,-57.0834,354.042,-1317.75,852.296,-530.54,-1058.86,**

**211.05,214.335,210.637,183.711,280.542,126.743,219.807,345.936,-75.4419,482.071,147.638,-124.032,824.349,-409.649,337.837,748.963,-922.834,1247.22,-135.619,-663.937,**

**455.734,459.019,455.32,428.395,525.225,371.426,464.49,590.62,169.241,726.754,392.321,120.652,1069.03,-164.966,582.521,993.646,-678.15,1491.9,109.065,-419.254,**

**209.609,212.894,209.196,182.27,279.101,125.302,218.366,344.495,-76.883,480.63,146.196,-125.473,822.908,-411.09,336.396,747.521,-924.275,1245.78,-137.06,-665.378,**

**-321.745,-318.46,-322.158,-349.084,-252.253,-406.052,-312.988,-186.859,-608.237,-50.7243,-385.158,-656.827,291.554,-942.444,-194.958,216.168,-1455.63,714.422,-668.414,-1196.73,**

**-538.234,-534.949,-538.647,-565.573,-468.742,-622.541,-529.477,-403.348,-824.726,-267.213,-601.647,-873.316,75.0652,-1158.93,-411.447,-0.321527,-1672.12,497.933,-884.903,-1413.22,**

**-119.807,-116.522,-120.22,-147.146,-50.3158,-204.114,-111.05,15.0787,-406.299,151.213,-183.22,-454.889,493.492,-740.507,6.97981,418.105,-1253.69,916.359,-466.476,-994.794,**

**534.035,537.32,533.621,506.696,603.526,449.727,542.791,668.92,247.542,805.055,470.622,198.953,1147.33,-86.6649,660.822,1071.95,-599.849,1570.2,187.366,-340.953,**

**679.822,683.107,679.409,652.483,749.314,595.515,688.579,814.708,393.33,950.843,616.41,344.74,1293.12,59.1228,806.609,1217.73,-454.062,1715.99,333.153,-195.165,**

**78.1882,81.4732,77.7749,50.8493,147.68,-6.11905,86.9449,213.074,-208.304,349.209,14.7754,-256.894,691.487,-542.511,204.975,616.1,-1055.7,1114.35,-268.481,-796.799,**

**-651.863,-648.578,-652.276,-679.202,-582.371,-736.17,-643.106,-516.977,-938.355,-380.842,-715.275,-986.945,-38.5636,-1272.56,-525.076,-113.95,-1785.75,384.304,-998.532,-1526.85,**

**-662.641,-659.356,-663.054,-689.98,-593.149,-746.948,-653.884,-527.755,-949.133,-391.62,-726.054,-997.723,-49.3418,-1283.34,-535.854,-124.729,-1796.52,373.526,-1009.31,-1537.63**

**};**

**typedef struct {**

**long double \*\*mat;**

**int m;**

**int n;**

**} matrix\_t;**

**void pamiecDel(matrix\_t \*b)**

**{**

**for (int i = 0; i < b->m; ++i)**

**delete []b->mat[i];**

**delete []b->mat;**

**}**

**int pomnoz\_macierz(matrix\_t \*c, matrix\_t \*a, matrix\_t \*b){**

**for(int i = 0; i < a->m; i++){**

**for(int j = 0; j < b->n; j++){**

**double suma = 0;**

**for(int s = 0; s < a->n; s++){**

**suma += a->mat[i][s] \* b->mat[s][j];**

**}**

**c->mat[i][j] = suma;**

**}**

**}**

**return 0;**

**}**

**void pamiecNew(matrix\_t \*b, const int& m\_, const int& n\_)**

**{**

**b->mat = new long double\* [m\_];**

**for (int i = 0; i < m\_; i++){**

**b->mat[i] = new long double[n\_];**

**}**

**b->m = m\_;**

**b->n = n\_;**

**}**

**matrix\_t odwroc\_macierz(matrix\_t \*a){**

**matrix\_t b;**

**pamiecNew(&b, a->m, a->n \* 2);**

**double c = 0;**

**double d = 0;**

**for(int i = 0; i < a->m; i++){**

**for(int j = 0; j < a->n; j++){**

**b.mat[i][j] = a->mat[i][j];**

**}**

**}**

**for (int i = 0; i < a->n; i++)**

**{**

**for (int j = a->n; j < 2 \* a->n; j++)**

**{**

**b.mat[i][j] = 0;**

**}**

**}**

**for (int i = 0; i < a->n; i++)**

**{**

**b.mat[i][i + a->n] = 1;**

**}**

**for (int s = 0; s < a->n; s++)**

**{**

**c = b.mat[s][s];**

**b.mat[s][s] = b.mat[s][s] - 1;**

**for (int j = s + 1; j < 2 \* a->n; j++)**

**{**

**d = (b.mat[s][j])/ c;**

**for (int i = 0; i < a->n; i++)**

**{**

**b.mat[i][j] = b.mat[i][j] - (d \* b.mat[i][s]);**

**}**

**}**

**}**

**matrix\_t ret;**

**pamiecNew(&ret, a->n, a->n);**

**for(int i = 0; i < a->n; i++){**

**for(int j = 0; j < a->n; j++){**

**ret.mat[i][j] = b.mat[i][j + a->n];**

**}**

**}**

**pamiecDel(&b);**

**return ret;**

**}**

**void inter\_wielomianowa(long double \*x, long double \*y){**

**matrix\_t Y;**

**pamiecNew(&Y, 3, 1);**

**matrix\_t X;**

**pamiecNew(&X, 3, 3);**

**for(int i = 0; i < X.m; i++){**

**for(int j = 0; j < X.n; j++){**

**X.mat[i][j] = pow(x[i], j);**

**}**

**}**

**for(int i = 0; i < Y.m; i++){**

**for(int j = 0; j < Y.n; j++){**

**Y.mat[i][j]= y[i];**

**}**

**}**

**matrix\_t X\_rev = odwroc\_macierz(&X);**

**matrix\_t F;**

**pamiecNew(&F, 3, 1);**

**pomnoz\_macierz(&F, &X\_rev, &Y);**

**long double a = F.mat[2][0];**

**long double b = F.mat[1][0];**

**long double c = F.mat[0][0];**

**std::cout << a << "x^2 + " << b << "x + " << c << '\n';**

**pamiecDel(&X);**

**pamiecDel(&Y);**

**pamiecDel(&X\_rev);**

**pamiecDel(&F);**

**}**

**int main(){**

**long double \*y = (long double \* ) &z[15];**

**for(int i = 0; i <= 18; i+=2){**

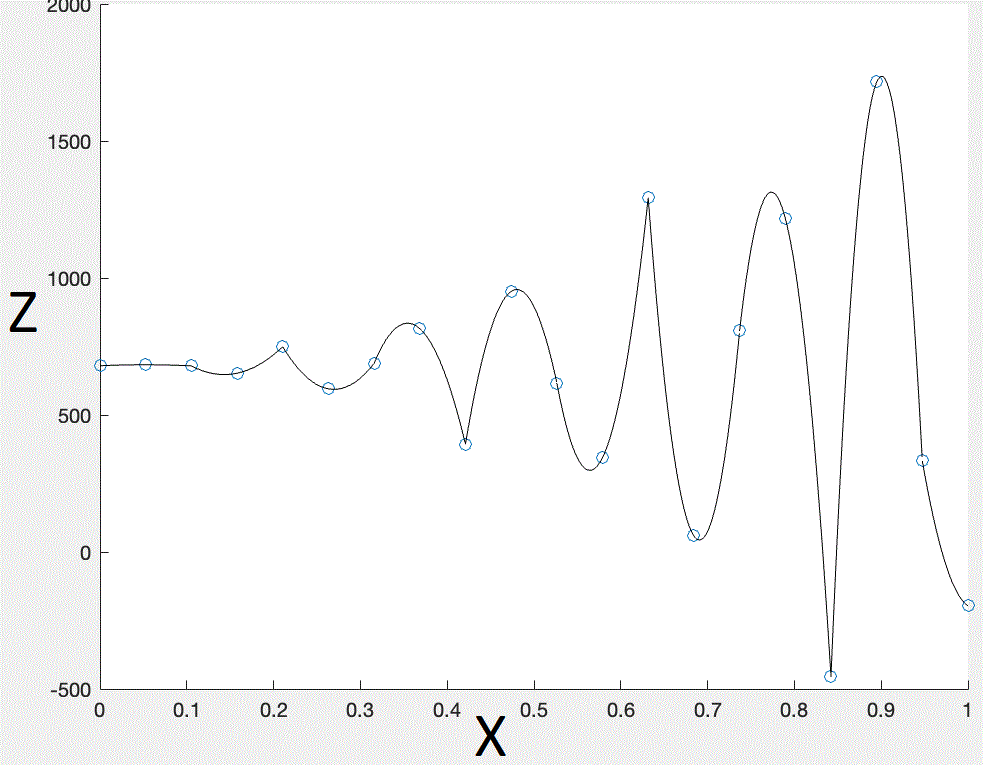
**inter\_wielomianowa(x + i, y + i);**

**}**

**inter\_wielomianowa(x + 17, y + 17);**

**}**

**Wykres wygląda następująco:**  
**4.Wyznaczyć funkcje aproksymacyjne** **dla wybranego wiersza lub kolumny z siatki.  
-240.746 \* x + 605.823  
Kod programu:  
#include <iostream>**



**#include <math.h>**

**using namespace std;**

**long double f(long double a, long double b, long double x)**

**{**

**return a\*x + b;**

**}**

**int main()**

**{**

**const int m = 20;**

**int z = 2;**

**long double Y[m] = {534.035,537.32,533.621,506.696,603.526,449.727,542.791,668.92,247.542,805.055,470.622,198.953,1147.33,-86.6649,660.822,1071.95,-599.849,1570.2,187.366,-340.953};**

**long double X[m] =**

**{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};**

**long double B[z][z+1];**

**long double p[z][1];**

**long double suma\_x = 0;**

**long double suma\_x2 = 0;**

**long double suma\_y = 0;**

**long double suma\_xy = 0;**

**B[0][0]= m;**

**for(int i = 0; i < m; i++)**

**{**

**suma\_x += X[i];**

**suma\_y += Y[i];**

**suma\_x2 += X[i]\*X[i];**

**suma\_xy += X[i]\*Y[i];**

**}**

**B[0][1] = suma\_x;**

**B[1][0] = suma\_x;**

**B[1][1] = suma\_x2;**

**B[0][2] = suma\_y;**

**B[1][2] = suma\_xy;**

**for(int s = 0; s <= z-1 ; s++)**

**{**

**for(int i = s+1; i < z; i++)**

**{**

**for(int j = s+1; j <= z+1; j++)**

**{**

**B[i][j]=B[i][j] - B[i][s]/B[s][s] \* B[s][j];**

**}**

**}**

**}**

**p[z-1][0] = B[z-1][z]/B[z-1][z-1];**

**for(int i = z - 2; i >= 0; i--){**

**long double suma = 0.0;**

**for(int s = i + 1; s < z; s++){**

**suma += B[i][s] \* p[s][0];**

**}**

**p[i][0] = (B[i][z] - suma) / B[i][i];**

**}**

**cout << "Wspolczynniki funkcji aproksymacyjnej: " << endl;**

**cout << "a = " << p[1][0] <<endl;**

**cout << "b = " << p[0][0] << '\n' << endl;**

**cout << "Kolejne punkty: " << endl;**

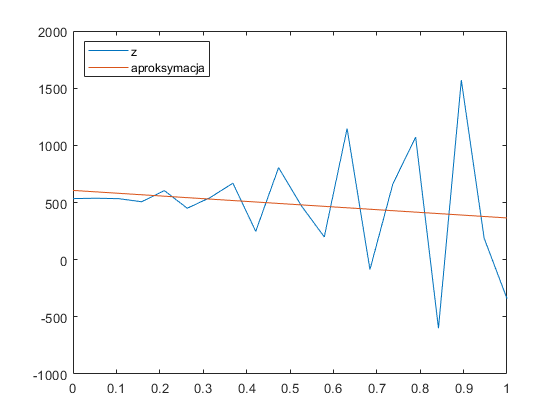
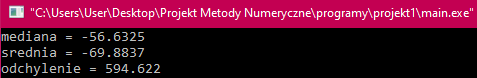
**for(int i = 0; i < m; i++)**

**{**

**cout << "f(" << i << ") = " << f(p[1][0], p[0][0], X[i]) << endl;**

**}**

**return 0;**

**}  
  
  
Wykres wygląda następująco:  
**  
**5.Wyznaczyć średnią, medianę, odchylenie standardowe.  
**

**Średnia, mediana i odchylenie standardowe zostały obliczone dzięki programowi „projekt1.cpp”, którego kod wstawiam poniżej:**  
#include <iostream>

#include <cmath>

using namespace std;

int main()

{

int n = 400;

double tab [n] = {

4.1,7.38498,3.68669,-23.2389,73.5913,-80.2073,12.8567,138.986,-282.392,275.12,-59.3128,-330.982,617.399,-616.6,130.887,542.012,-1129.78,1040.27,-342.569,-870.887,

-35.2535,-31.9686,-35.6668,-62.5925,34.2378,-119.561,-26.4968,99.6323,-321.746,235.767,-98.6664,-370.336,578.045,-655.953,91.5334,502.659,-1169.14,1000.91,-381.923,-910.241,

-66.3381,-63.0531,-66.7514,-93.677,3.15324,-150.645,-57.5814,68.5477,-352.83,204.682,-129.751,-401.42,546.961,-687.038,60.4488,471.574,-1200.22,969.829,-413.007,-941.325,

28.1257,31.4107,27.7124,0.786774,97.617,-56.1816,36.8824,163.011,-258.367,299.146,-35.2871,-306.956,641.425,-592.574,154.913,566.038,-1105.76,1064.29,-318.544,-846.862,

175.262,178.547,174.849,147.924,244.754,90.9551,184.019,310.148,-111.23,446.283,111.85,-159.82,788.561,-445.437,302.049,713.175,-958.622,1211.43,-171.407,-699.725,

158.586,161.871,158.172,131.247,228.077,74.2784,167.342,293.471,-127.907,429.606,95.1729,-176.496,771.885,-462.114,285.373,696.498,-975.298,1194.75,-188.083,-716.402,

-82.6359,-79.3509,-83.0492,-109.975,-13.1446,-166.943,-73.8792,52.2499,-369.128,188.385,-146.049,-417.718,530.663,-703.335,44.151,455.276,-1216.52,953.531,-429.305,-957.623,

-298.115,-294.83,-298.529,-325.454,-228.624,-382.423,-289.359,-163.23,-584.608,-27.0949,-361.528,-633.197,315.184,-918.815,-171.328,239.797,-1432,738.051,-644.785,-1173.1,

-183.87,-180.585,-184.284,-211.209,-114.379,-268.178,-175.114,-48.9845,-470.363,87.1501,-247.283,-518.952,429.429,-804.57,-57.0834,354.042,-1317.75,852.296,-530.54,-1058.86,

211.05,214.335,210.637,183.711,280.542,126.743,219.807,345.936,-75.4419,482.071,147.638,-124.032,824.349,-409.649,337.837,748.963,-922.834,1247.22,-135.619,-663.937,

455.734,459.019,455.32,428.395,525.225,371.426,464.49,590.62,169.241,726.754,392.321,120.652,1069.03,-164.966,582.521,993.646,-678.15,1491.9,109.065,-419.254,

209.609,212.894,209.196,182.27,279.101,125.302,218.366,344.495,-76.883,480.63,146.196,-125.473,822.908,-411.09,336.396,747.521,-924.275,1245.78,-137.06,-665.378,

-321.745,-318.46,-322.158,-349.084,-252.253,-406.052,-312.988,-186.859,-608.237,-50.7243,-385.158,-656.827,291.554,-942.444,-194.958,216.168,-1455.63,714.422,-668.414,-1196.73,

-538.234,-534.949,-538.647,-565.573,-468.742,-622.541,-529.477,-403.348,-824.726,-267.213,-601.647,-873.316,75.0652,-1158.93,-411.447,-0.321527,-1672.12,497.933,-884.903,-1413.22,

-119.807,-116.522,-120.22,-147.146,-50.3158,-204.114,-111.05,15.0787,-406.299,151.213,-183.22,-454.889,493.492,-740.507,6.97981,418.105,-1253.69,916.359,-466.476,-994.794,

534.035,537.32,533.621,506.696,603.526,449.727,542.791,668.92,247.542,805.055,470.622,198.953,1147.33,-86.6649,660.822,1071.95,-599.849,1570.2,187.366,-340.953,

679.822,683.107,679.409,652.483,749.314,595.515,688.579,814.708,393.33,950.843,616.41,344.74,1293.12,59.1228,806.609,1217.73,-454.062,1715.99,333.153,-195.165,

78.1882,81.4732,77.7749,50.8493,147.68,-6.11905,86.9449,213.074,-208.304,349.209,14.7754,-256.894,691.487,-542.511,204.975,616.1,-1055.7,1114.35,-268.481,-796.799,

-651.863,-648.578,-652.276,-679.202,-582.371,-736.17,-643.106,-516.977,-938.355,-380.842,-715.275,-986.945,-38.5636,-1272.56,-525.076,-113.95,-1785.75,384.304,-998.532,-1526.85,

-662.641,-659.356,-663.054,-689.98,-593.149,-746.948,-653.884,-527.755,-949.133,-391.62,-726.054,-997.723,-49.3418,-1283.34,-535.854,-124.729,-1796.52,373.526,-1009.31,-1537.63,

};

double var = tab[0];

for(int j = 0; j < n; j++){

var = tab[0];

for(int i = 1; i < n; i++)

{

if(var > tab[i])

{

tab[i-1] = tab[i];

tab[i] = var;

}

else if(var < tab[i])

{

var = tab[i];

}

}

}

double mediana = 0;

int srodek = n/2;

if(srodek % 2 == 0)

{

mediana = (tab[srodek] + tab[srodek-1])/2;

}

else

{

mediana = tab[srodek+1];

}

cout << "mediana = " << mediana << endl;

double sumaS = 0;

double srednia = 0;

for(int i = 0; i < n; i++)

{

sumaS += tab[i];

}

srednia = sumaS/n;

cout << "srednia = " << srednia << endl;

double sumaO = 0;

double odchylenie = 0;

for(int i = 0; i < n; i++)

{

sumaO += (tab[i] - srednia)\*(tab[i] - srednia);

}

odchylenie = sqrt(sumaO/n);

cout << "odchylenie = " << odchylenie << endl;

const int m = 20;

double tabx [m][m] = {

{4.1,7.38498,3.68669,-23.2389,73.5913,-80.2073,12.8567,138.986,-282.392,275.12,-59.3128,-330.982,617.399,-616.6,130.887,542.012,-1129.78,1040.27,-342.569,-870.887},

{-35.2535,-31.9686,-35.6668,-62.5925,34.2378,-119.561,-26.4968,99.6323,-321.746,235.767,-98.6664,-370.336,578.045,-655.953,91.5334,502.659,-1169.14,1000.91,-381.923,-910.241},

{-66.3381,-63.0531,-66.7514,-93.677,3.15324,-150.645,-57.5814,68.5477,-352.83,204.682,-129.751,-401.42,546.961,-687.038,60.4488,471.574,-1200.22,969.829,-413.007,-941.325},

{28.1257,31.4107,27.7124,0.786774,97.617,-56.1816,36.8824,163.011,-258.367,299.146,-35.2871,-306.956,641.425,-592.574,154.913,566.038,-1105.76,1064.29,-318.544,-846.862},

{175.262,178.547,174.849,147.924,244.754,90.9551,184.019,310.148,-111.23,446.283,111.85,-159.82,788.561,-445.437,302.049,713.175,-958.622,1211.43,-171.407,-699.725},

{158.586,161.871,158.172,131.247,228.077,74.2784,167.342,293.471,-127.907,429.606,95.1729,-176.496,771.885,-462.114,285.373,696.498,-975.298,1194.75,-188.083,-716.402},

{-82.6359,-79.3509,-83.0492,-109.975,-13.1446,-166.943,-73.8792,52.2499,-369.128,188.385,-146.049,-417.718,530.663,-703.335,44.151,455.276,-1216.52,953.531,-429.305,-957.623},

{-298.115,-294.83,-298.529,-325.454,-228.624,-382.423,-289.359,-163.23,-584.608,-27.0949,-361.528,-633.197,315.184,-918.815,-171.328,239.797,-1432,738.051,-644.785,-1173.1},

{-183.87,-180.585,-184.284,-211.209,-114.379,-268.178,-175.114,-48.9845,-470.363,87.1501,-247.283,-518.952,429.429,-804.57,-57.0834,354.042,-1317.75,852.296,-530.54,-1058.86},

{211.05,214.335,210.637,183.711,280.542,126.743,219.807,345.936,-75.4419,482.071,147.638,-124.032,824.349,-409.649,337.837,748.963,-922.834,1247.22,-135.619,-663.937},

{455.734,459.019,455.32,428.395,525.225,371.426,464.49,590.62,169.241,726.754,392.321,120.652,1069.03,-164.966,582.521,993.646,-678.15,1491.9,109.065,-419.254},

{209.609,212.894,209.196,182.27,279.101,125.302,218.366,344.495,-76.883,480.63,146.196,-125.473,822.908,-411.09,336.396,747.521,-924.275,1245.78,-137.06,-665.378},

{-321.745,-318.46,-322.158,-349.084,-252.253,-406.052,-312.988,-186.859,-608.237,-50.7243,-385.158,-656.827,291.554,-942.444,-194.958,216.168,-1455.63,714.422,-668.414,-1196.73},

{-538.234,-534.949,-538.647,-565.573,-468.742,-622.541,-529.477,-403.348,-824.726,-267.213,-601.647,-873.316,75.0652,-1158.93,-411.447,-0.321527,-1672.12,497.933,-884.903,-1413.22},

{-119.807,-116.522,-120.22,-147.146,-50.3158,-204.114,-111.05,15.0787,-406.299,151.213,-183.22,-454.889,493.492,-740.507,6.97981,418.105,-1253.69,916.359,-466.476,-994.794},

{534.035,537.32,533.621,506.696,603.526,449.727,542.791,668.92,247.542,805.055,470.622,198.953,1147.33,-86.6649,660.822,1071.95,-599.849,1570.2,187.366,-340.953},

{679.822,683.107,679.409,652.483,749.314,595.515,688.579,814.708,393.33,950.843,616.41,344.74,1293.12,59.1228,806.609,1217.73,-454.062,1715.99,333.153,-195.165},

{78.1882,81.4732,77.7749,50.8493,147.68,-6.11905,86.9449,213.074,-208.304,349.209,14.7754,-256.894,691.487,-542.511,204.975,616.1,-1055.7,1114.35,-268.481,-796.799},

{-651.863,-648.578,-652.276,-679.202,-582.371,-736.17,-643.106,-516.977,-938.355,-380.842,-715.275,-986.945,-38.5636,-1272.56,-525.076,-113.95,-1785.75,384.304,-998.532,-1526.85},

{-662.641,-659.356,-663.054,-689.98,-593.149,-746.948,-653.884,-527.755,-949.133,-391.62,-726.054,-997.723,-49.3418,-1283.34,-535.854,-124.729,-1796.52,373.526,-1009.31,-1537.63}

};

int z = 2;

long double X[m] = {71.8237,119.777,90.8386,-65.1875,0.585689,286.255,233.286,-185.554,-192.383,355.266,475.602,-173.517,-446.913,273.465,742.997,5.91143,-677.902,26.9053,945.018,344.09};

long double Y[m] = {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};

long double B[z][z+1];

long double p[z][1];

long double suma\_x = 0;

long double suma\_x2 = 0;

long double suma\_y = 0;

long double suma\_xy = 0;

B[0][0]= m;

for(int i = 0; i < m; i++)

{

suma\_x += X[i];

suma\_y += Y[i];

suma\_x2 = suma\_x2 + (X[i]\*X[i]);

suma\_xy += X[i]\*Y[i];

}

B[0][1] = suma\_x;

B[1][0] = suma\_x;

B[1][1] = suma\_x2;

B[0][2] = suma\_y;

B[1][2] = suma\_xy;

for(int s = 0; s <= z-1 ; s++)

{

for(int i = s+1; i < z; i++)

{

for(int j = s+1; j <= z+1; j++)

{

B[i][j]=B[i][j] - B[i][s]/B[s][s] \* B[s][j];

}

}

}

p[z-1][0] = B[z-1][z]/B[z-1][z-1];

for(int i = z - 2; i >= 0; i--){

double suma = 0.0;

for(int s = i + 1; s < z; s++){

suma += B[i][s] \* p[s][0];

}

p[i][0] = (B[i][z] - suma) / B[i][i];

}

for(int i = 0; i < z; i++)

{

cout << "p"<< i << " = " << p[i][0] <<endl;

}

double funkcja = 0;

for(int i = 0; i < m; i++)

{

funkcja = i\*p[0][0] + p[1][0];

cout << /\*"x" << i << " = " <<\*/ funkcja << endl;

}

return 0;  
} **6.Obliczyć pole powierzchni.  
  
  
  
Pole powierzchni zostało obliczone dzięki programowi „pole.cpp”, którego kod wstawiam poniżej:**  
  
  
  
#include <iostream>

#include <cmath>

long 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, 0.789474, 0.842105, 0.894737, 0.947368, 1 };

long double y[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, 0.789474, 0.842105, 0.894737, 0.947368, 1 };

long double z[20][20] = {

4.1,7.38498,3.68669,-23.2389,73.5913,-80.2073,12.8567,138.986,-282.392,275.12,-59.3128,-330.982,617.399,-616.6,130.887,542.012,-1129.78,1040.27,-342.569,-870.887,

-35.2535,-31.9686,-35.6668,-62.5925,34.2378,-119.561,-26.4968,99.6323,-321.746,235.767,-98.6664,-370.336,578.045,-655.953,91.5334,502.659,-1169.14,1000.91,-381.923,-910.241,

-66.3381,-63.0531,-66.7514,-93.677,3.15324,-150.645,-57.5814,68.5477,-352.83,204.682,-129.751,-401.42,546.961,-687.038,60.4488,471.574,-1200.22,969.829,-413.007,-941.325,

28.1257,31.4107,27.7124,0.786774,97.617,-56.1816,36.8824,163.011,-258.367,299.146,-35.2871,-306.956,641.425,-592.574,154.913,566.038,-1105.76,1064.29,-318.544,-846.862,

175.262,178.547,174.849,147.924,244.754,90.9551,184.019,310.148,-111.23,446.283,111.85,-159.82,788.561,-445.437,302.049,713.175,-958.622,1211.43,-171.407,-699.725,

158.586,161.871,158.172,131.247,228.077,74.2784,167.342,293.471,-127.907,429.606,95.1729,-176.496,771.885,-462.114,285.373,696.498,-975.298,1194.75,-188.083,-716.402,

-82.6359,-79.3509,-83.0492,-109.975,-13.1446,-166.943,-73.8792,52.2499,-369.128,188.385,-146.049,-417.718,530.663,-703.335,44.151,455.276,-1216.52,953.531,-429.305,-957.623,

-298.115,-294.83,-298.529,-325.454,-228.624,-382.423,-289.359,-163.23,-584.608,-27.0949,-361.528,-633.197,315.184,-918.815,-171.328,239.797,-1432,738.051,-644.785,-1173.1,

-183.87,-180.585,-184.284,-211.209,-114.379,-268.178,-175.114,-48.9845,-470.363,87.1501,-247.283,-518.952,429.429,-804.57,-57.0834,354.042,-1317.75,852.296,-530.54,-1058.86,

211.05,214.335,210.637,183.711,280.542,126.743,219.807,345.936,-75.4419,482.071,147.638,-124.032,824.349,-409.649,337.837,748.963,-922.834,1247.22,-135.619,-663.937,

455.734,459.019,455.32,428.395,525.225,371.426,464.49,590.62,169.241,726.754,392.321,120.652,1069.03,-164.966,582.521,993.646,-678.15,1491.9,109.065,-419.254,

209.609,212.894,209.196,182.27,279.101,125.302,218.366,344.495,-76.883,480.63,146.196,-125.473,822.908,-411.09,336.396,747.521,-924.275,1245.78,-137.06,-665.378,

-321.745,-318.46,-322.158,-349.084,-252.253,-406.052,-312.988,-186.859,-608.237,-50.7243,-385.158,-656.827,291.554,-942.444,-194.958,216.168,-1455.63,714.422,-668.414,-1196.73,

-538.234,-534.949,-538.647,-565.573,-468.742,-622.541,-529.477,-403.348,-824.726,-267.213,-601.647,-873.316,75.0652,-1158.93,-411.447,-0.321527,-1672.12,497.933,-884.903,-1413.22,

-119.807,-116.522,-120.22,-147.146,-50.3158,-204.114,-111.05,15.0787,-406.299,151.213,-183.22,-454.889,493.492,-740.507,6.97981,418.105,-1253.69,916.359,-466.476,-994.794,

534.035,537.32,533.621,506.696,603.526,449.727,542.791,668.92,247.542,805.055,470.622,198.953,1147.33,-86.6649,660.822,1071.95,-599.849,1570.2,187.366,-340.953,

679.822,683.107,679.409,652.483,749.314,595.515,688.579,814.708,393.33,950.843,616.41,344.74,1293.12,59.1228,806.609,1217.73,-454.062,1715.99,333.153,-195.165,

78.1882,81.4732,77.7749,50.8493,147.68,-6.11905,86.9449,213.074,-208.304,349.209,14.7754,-256.894,691.487,-542.511,204.975,616.1,-1055.7,1114.35,-268.481,-796.799,

-651.863,-648.578,-652.276,-679.202,-582.371,-736.17,-643.106,-516.977,-938.355,-380.842,-715.275,-986.945,-38.5636,-1272.56,-525.076,-113.95,-1785.75,384.304,-998.532,-1526.85,

-662.641,-659.356,-663.054,-689.98,-593.149,-746.948,-653.884,-527.755,-949.133,-391.62,-726.054,-997.723,-49.3418,-1283.34,-535.854,-124.729,-1796.52,373.526,-1009.31,-1537.63

};

struct punkt {

double x;

double y;

double z;

};

long double pole\_trk(double a, double b, double c){

double p = (a + b + c) / 2;

long double pole = sqrt(p \* (p - a)\*(p - b)\*(p - c));

return pole;

}

long double pole(punkt &a, punkt &b, punkt &c, punkt &d){

double e = sqrt(pow(a.x - b.x,2) + pow(a.z - b.z,2));

double f = sqrt(pow(c.y - a.y,2) + pow(c.z - a.z, 2));

double pod = sqrt(pow(b.x - a.x,2) + pow(a.y - c.y, 2));

double g = sqrt(pow(c.z - b.z,2) + pow(pod,2));

double pole\_r = pole\_trk(e, f, g);

e = sqrt(pow(c.x - d.x,2) + pow(c.z - d.z,2));

f = sqrt(pow(d.y - b.y, 2) + pow(d.z - b.z, 2));

pole\_r += pole\_trk(e, f, g);

return pole\_r;

}

int main(){

double polecal = 0.0;

for(int i = 0; i < 19; i++){

for(int j = 0; j < 19; j++){

punkt a = {x[i], y[j], z[i][j]};

punkt b = {x[i + 1], y[j], z[i + 1][j]};

punkt c = {x[i], y[j + 1], z[i][j + 1]};

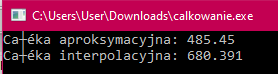
punkt d = {x[i + 1], y[j + 1], z[i + 1][j + 1]};

polecal += pole(a,b,c,d);

}

}

std::cout << polecal << '\n';

}  
**7.Obliczyć całkę z funkcji interpolacyjnych i aproksymacyjnych.  
  
  
  
Kod programu:**#include <iostream>

#include <cmath>

int n = 19;

double a = 0;

double b = 1;

double h = 0.0526316;

double f(double x){

return -240.746 \* x + 605.823;

}

long double g(long double x, int i){

if(i >= 0 && i <= 1)

return -1260.44 \* x \* x + 128.754 \* x + 697.822;

if(i >= 2 && i <= 3)

return 22338.3 \* x \* x - 6390.1 \* x + 1104.53;

if(i >= 4 && i <= 5)

return 44558.8 \* x \* x -24028.9 \* x + 3833.13;

if(i >= 6 && i <= 7)

return -98823.4 \* x \* x + 70012.4 \* x - 11565.6;

if(i >= 8 && i <= 9)

return - 160997 \* x \* x + 154643 \* x - 36177.1;

if(i >= 10 && i <= 11)

return 220219 \* x \* x - 248561 \* x + 70435.7;

if(i >= 12 && i <= 13)

return 357658 \* x \* x - 494049 \* x + 170657;

if(i >= 14 && i <= 15)

return -375969 \* x \* x + 581659 \* x - 223657;

if(i >= 16 && i <= 17)

return -641297 \* x \* x +1155060 \* x - 518368;

if(i >= 18 && i <= 19)

return 154244 \* x \* x - 310409 \* x + 155969;

return 0;

}

int main(){

double x\_l, x\_p, x\_s;

double calka = 0.0;

for(int i = 0 ; i < n ; i++){

x\_l = a + i \* h;

x\_p = a + i \* h + h;

x\_s = a + i \* h + (h/2);

calka += (h / 6) \* (f(x\_l) + 4 \* f(x\_s) + f(x\_p));

}

std::cout << "Całka aproksymacyjna: " << calka << '\n';

calka = 0.0;

for(int i = 0 ; i < n ; i++){

x\_l = a + i \* h;

x\_p = a + i \* h + h;

x\_s = a + i \* h + (h/2);

calka += (h / 6) \* (g(x\_l, i) + 4 \* g(x\_s, i) + g(x\_p, i));

}

std::cout << "Całka interpolacyjna: " << calka << '\n';

}  
 **8.Wyznaczyć pochodne cząstkowe.  
  
Pochodna cząstkowa – dla danej funkcji wielu zmiennych pochodna względem jednej z jej zmiennych przy ustaleniu pozostałych. Pochodne cząstkowe znajdują zastosowanie np. w rachunku wektorowym oraz geometrii różniczkowej.**

**Do wyznaczenia pochodnej cząstkowej użyłem programu, którego kod wygląda następująco:**  
#include <iostream>

#include <fstream>

long 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, 0.789474, 0.842105, 0.894737, 0.947368, 1 };

long double y[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, 0.789474, 0.842105, 0.894737, 0.947368, 1 };

long double d[20][20] = {

4.1,7.38498,3.68669,-23.2389,73.5913,-80.2073,12.8567,138.986,-282.392,275.12,-59.3128,-330.982,617.399,-616.6,130.887,542.012,-1129.78,1040.27,-342.569,-870.887,

-35.2535,-31.9686,-35.6668,-62.5925,34.2378,-119.561,-26.4968,99.6323,-321.746,235.767,-98.6664,-370.336,578.045,-655.953,91.5334,502.659,-1169.14,1000.91,-381.923,-910.241,

-66.3381,-63.0531,-66.7514,-93.677,3.15324,-150.645,-57.5814,68.5477,-352.83,204.682,-129.751,-401.42,546.961,-687.038,60.4488,471.574,-1200.22,969.829,-413.007,-941.325,

28.1257,31.4107,27.7124,0.786774,97.617,-56.1816,36.8824,163.011,-258.367,299.146,-35.2871,-306.956,641.425,-592.574,154.913,566.038,-1105.76,1064.29,-318.544,-846.862,

175.262,178.547,174.849,147.924,244.754,90.9551,184.019,310.148,-111.23,446.283,111.85,-159.82,788.561,-445.437,302.049,713.175,-958.622,1211.43,-171.407,-699.725,

158.586,161.871,158.172,131.247,228.077,74.2784,167.342,293.471,-127.907,429.606,95.1729,-176.496,771.885,-462.114,285.373,696.498,-975.298,1194.75,-188.083,-716.402,

-82.6359,-79.3509,-83.0492,-109.975,-13.1446,-166.943,-73.8792,52.2499,-369.128,188.385,-146.049,-417.718,530.663,-703.335,44.151,455.276,-1216.52,953.531,-429.305,-957.623,

-298.115,-294.83,-298.529,-325.454,-228.624,-382.423,-289.359,-163.23,-584.608,-27.0949,-361.528,-633.197,315.184,-918.815,-171.328,239.797,-1432,738.051,-644.785,-1173.1,

-183.87,-180.585,-184.284,-211.209,-114.379,-268.178,-175.114,-48.9845,-470.363,87.1501,-247.283,-518.952,429.429,-804.57,-57.0834,354.042,-1317.75,852.296,-530.54,-1058.86,

211.05,214.335,210.637,183.711,280.542,126.743,219.807,345.936,-75.4419,482.071,147.638,-124.032,824.349,-409.649,337.837,748.963,-922.834,1247.22,-135.619,-663.937,

455.734,459.019,455.32,428.395,525.225,371.426,464.49,590.62,169.241,726.754,392.321,120.652,1069.03,-164.966,582.521,993.646,-678.15,1491.9,109.065,-419.254,

209.609,212.894,209.196,182.27,279.101,125.302,218.366,344.495,-76.883,480.63,146.196,-125.473,822.908,-411.09,336.396,747.521,-924.275,1245.78,-137.06,-665.378,

-321.745,-318.46,-322.158,-349.084,-252.253,-406.052,-312.988,-186.859,-608.237,-50.7243,-385.158,-656.827,291.554,-942.444,-194.958,216.168,-1455.63,714.422,-668.414,-1196.73,

-538.234,-534.949,-538.647,-565.573,-468.742,-622.541,-529.477,-403.348,-824.726,-267.213,-601.647,-873.316,75.0652,-1158.93,-411.447,-0.321527,-1672.12,497.933,-884.903,-1413.22,

-119.807,-116.522,-120.22,-147.146,-50.3158,-204.114,-111.05,15.0787,-406.299,151.213,-183.22,-454.889,493.492,-740.507,6.97981,418.105,-1253.69,916.359,-466.476,-994.794,

534.035,537.32,533.621,506.696,603.526,449.727,542.791,668.92,247.542,805.055,470.622,198.953,1147.33,-86.6649,660.822,1071.95,-599.849,1570.2,187.366,-340.953,

679.822,683.107,679.409,652.483,749.314,595.515,688.579,814.708,393.33,950.843,616.41,344.74,1293.12,59.1228,806.609,1217.73,-454.062,1715.99,333.153,-195.165,

78.1882,81.4732,77.7749,50.8493,147.68,-6.11905,86.9449,213.074,-208.304,349.209,14.7754,-256.894,691.487,-542.511,204.975,616.1,-1055.7,1114.35,-268.481,-796.799,

-651.863,-648.578,-652.276,-679.202,-582.371,-736.17,-643.106,-516.977,-938.355,-380.842,-715.275,-986.945,-38.5636,-1272.56,-525.076,-113.95,-1785.75,384.304,-998.532,-1526.85,

-662.641,-659.356,-663.054,-689.98,-593.149,-746.948,-653.884,-527.755,-949.133,-391.62,-726.054,-997.723,-49.3418,-1283.34,-535.854,-124.729,-1796.52,373.526,-1009.31,-1537.63

};

long double Px[20][20];

long double Py[20][20];

long double h = 0.0526316;

int main() {

for (int i = 1; i < 19; i++) {

for (int j = 1; j < 19; j++) {

Px[i][j] = (d[i + 1][j] - d[i - 1][j]) / (2 \* h);

Py[i][j] = (d[i][j + 1] - d[i][j - 1]) / (2 \* h);

}

}

std::ofstream plik\_out;

plik\_out.open("px.txt");

for (int i = 1; i < 19; i++) {

for (int j = 1; j < 19; j++) {

plik\_out << Px[i][j] << ' ';

}

plik\_out << '\n';

}

plik\_out.close();

plik\_out.open("py.txt");

for (int i = 1; i < 19; i++) {

for (int j = 1; j < 19; j++) {

plik\_out << Py[i][j] << ' ';

}

plik\_out << '\n';

}

plik\_out.close();

plik\_out.open("mon\_x.txt");

for (int i = 1; i < 19; i++) {

for (int j = 1; j < 19; j++) {

if (Px[i][j] < 0) {

plik\_out << -1 << ' ';

} else {

plik\_out << 1 << ' ';

}

}

plik\_out << '\n';

}

plik\_out.close();

plik\_out.open("mon\_y.txt");

for (int i = 1; i < 19; i++) {

for (int j = 1; j < 19; j++) {

if (Py[i][j] < 0) {

plik\_out << -1 << ' ';

}

else {

plik\_out << 1 << ' ';

}

}

plik\_out << '\n';

}

plik\_out.close();

}  
  
**Wyniki prezentują się następująco:  
\*dla X:**-669.161 -669.162 -669.162 -669.161 -669.158 -669.162 -669.164 -669.161 -669.161 -669.163 -669.161 -669.161 -669.161 -669.163 -669.161 -669.18 -669.189 -669.161

602.103 602.102 602.103 602.102 602.104 602.102 602.097 602.1 602.1 602.103 602.11 602.11 602.1 602.106 602.1 602.11 602.11 602.1

2295.2 2295.2 2295.21 2295.21 2295.2 2295.2 2295.2 2295.2 2295.21 2295.21 2295.2 2295.2 2295.21 2295.2 2295.21 2295.18 2295.21 2295.2

1239.37 1239.37 1239.37 1239.37 1239.37 1239.37 1239.37 1239.37 1239.37 1239.37 1239.37 1239.37 1239.37 1239.37 1239.37 1239.39 1239.37 1239.38

-2450.03 -2450.03 -2450.04 -2450.04 -2450.03 -2450.03 -2450.03 -2450.03 -2450.03 -2450.04 -2450.03 -2450.03 -2450.03 -2450.03 -2450.04 -2450.03 -2450.04 -2450.03

-4338.66 -4338.66 -4338.66 -4338.66 -4338.66 -4338.66 -4338.66 -4338.66 -4338.66 -4338.66 -4338.66 -4338.66 -4338.66 -4338.66 -4338.66 -4338.67 -4338.64 -4338.67

-961.724 -961.73 -961.723 -961.726 -961.732 -961.73 -961.726 -961.732 -961.731 -961.723 -961.723 -961.723 -961.732 -961.726 -961.723 -961.685 -961.732 -961.732

4837.07 4837.08 4837.07 4837.08 4837.08 4837.08 4837.08 4837.08 4837.07 4837.08 4837.07 4837.07 4837.08 4837.07 4837.08 4837.08 4837.1 4837.08

6076.24 6076.24 6076.24 6076.24 6076.24 6076.24 6076.24 6076.24 6076.23 6076.24 6076.24 6076.21 6076.24 6076.24 6076.24 6076.2 6076.24 6076.25

-13.6895 -13.6895 -13.6895 -13.6895 -13.6895 -13.6895 -13.6895 -13.6904 -13.6895 -13.699 -13.6895 -13.6895 -13.6895 -13.6895 -13.699 -13.6895 -13.68 -13.6895

-7386.05 -7386.04 -7386.05 -7386.04 -7386.04 -7386.04 -7386.05 -7386.04 -7386.04 -7386.05 -7386.05 -7386.02 -7386.04 -7386.05 -7386.04 -7386.06 -7386.04 -7386.05

-7104.51 -7104.51 -7104.51 -7104.51 -7104.51 -7104.51 -7104.51 -7104.51 -7104.51 -7104.51 -7104.51 -7104.5 -7104.48 -7104.51 -7104.5 -7104.52 -7104.54 -7104.51

1918.41 1918.41 1918.41 1918.4 1918.41 1918.41 1918.41 1918.41 1918.4 1918.41 1918.41 1918.41 1918.4 1918.41 1918.4 1918.43 1918.4 1918.41

10186.6 10186.5 10186.6 10186.5 10186.5 10186.5 10186.5 10186.5 10186.5 10186.6 10186.6 10186.5 10186.5 10186.6 10186.6 10186.6 10186.5 10186.6

7596.47 7596.47 7596.47 7596.48 7596.47 7596.47 7596.48 7596.47 7596.48 7596.48 7596.47 7596.46 7596.48 7596.47 7596.43 7596.46 7596.49 7596.47

-4330.54 -4330.54 -4330.54 -4330.54 -4330.54 -4330.54 -4330.54 -4330.54 -4330.54 -4330.54 -4330.54 -4330.51 -4330.54 -4330.54 -4330.57 -4330.58 -4330.57 -4330.54

-12651 -12651 -12651 -12651 -12651 -12651 -12651 -12651 -12651 -12651 -12651 -12651 -12651 -12651 -12651 -12651 -12651 -12651

-7037.87 -7037.87 -7037.88 -7037.87 -7037.87 -7037.87 -7037.87 -7037.87 -7037.87 -7037.88 -7037.87 -7037.87 -7037.87 -7037.87 -7037.87 -7037.79 -7037.83 -7037.87  
  
**\*dla Y**  
  
-3.92635 -290.927 664.093 -541.201 -576.978 2082.34 -2804.87 1293.28 2119.26 -5757.98 6428.76 -2713.36 -4621.86 11006.8 -11976.4 4733.38 7478.56 -18155.9

-3.92635 -290.927 664.094 -541.196 -576.979 2082.33 -2804.86 1293.28 2119.25 -5757.97 6428.76 -2713.37 -4621.86 11006.8 -11976.3 4733.42 7478.52 -18156

-3.92635 -290.927 664.093 -541.199 -576.978 2082.33 -2804.87 1293.28 2119.26 -5757.97 6428.76 -2713.37 -4621.86 11006.8 -11976.4 4733.39 7478.55 -18155.9

-3.9235 -290.918 664.097 -541.204 -576.982 2082.33 -2804.86 1293.28 2119.26 -5757.98 6428.75 -2713.36 -4621.86 11006.8 -11976.4 4733.42 7478.54 -18156

-3.933 -290.928 664.097 -541.201 -576.982 2082.33 -2804.86 1293.28 2119.26 -5757.97 6428.76 -2713.37 -4621.86 11006.8 -11976.4 4733.39 7478.54 -18155.9

-3.92635 -290.929 664.093 -541.196 -576.978 2082.33 -2804.86 1293.28 2119.25 -5757.98 6428.76 -2713.36 -4621.86 11006.8 -11976.4 4733.42 7478.54 -18156

-3.933 -290.928 664.097 -541.205 -576.982 2082.33 -2804.86 1293.28 2119.26 -5757.97 6428.76 -2713.37 -4621.86 11006.8 -11976.4 4733.41 7478.54 -18155.9

-3.933 -290.928 664.097 -541.205 -576.982 2082.34 -2804.86 1293.28 2119.26 -5757.97 6428.76 -2713.37 -4621.87 11006.8 -11976.3 4733.41 7478.49 -18156

-3.9235 -290.928 664.097 -541.196 -576.982 2082.33 -2804.86 1293.28 2119.26 -5757.98 6428.75 -2713.36 -4621.86 11006.8 -11976.4 4733.44 7478.54 -18156

-3.933 -290.928 664.097 -541.205 -576.982 2082.34 -2804.86 1293.27 2119.26 -5757.97 6428.73 -2713.37 -4621.83 11006.8 -11976.4 4733.41 7478.54 -18156

-3.9235 -290.928 664.097 -541.196 -576.982 2082.33 -2804.86 1293.28 2119.25 -5757.98 6428.76 -2713.36 -4621.86 11006.8 -11976.4 4733.46 7478.54 -18156

-3.9235 -290.928 664.097 -541.196 -576.982 2082.33 -2804.86 1293.28 2119.25 -5757.97 6428.76 -2713.36 -4621.86 11006.8 -11976.4 4733.41 7478.55 -18155.9

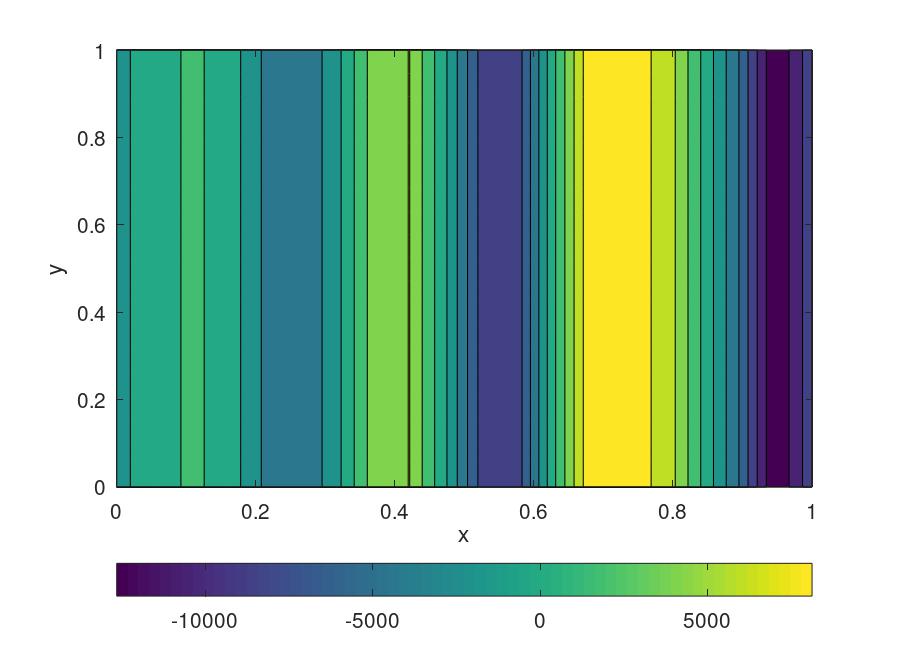
-3.9235 -290.928 664.097 -541.196 -576.982 2082.33 -2804.86 1293.28 2119.25 -5757.98 6428.76 -2713.33 -4621.86 11006.8 -11976.4 4733.42 7478.56 -18155.9

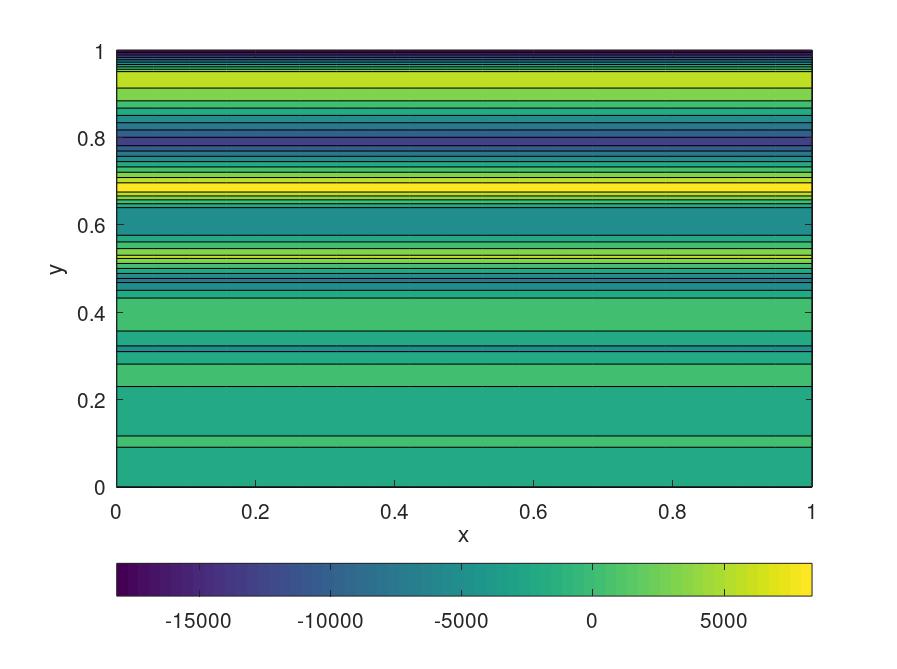
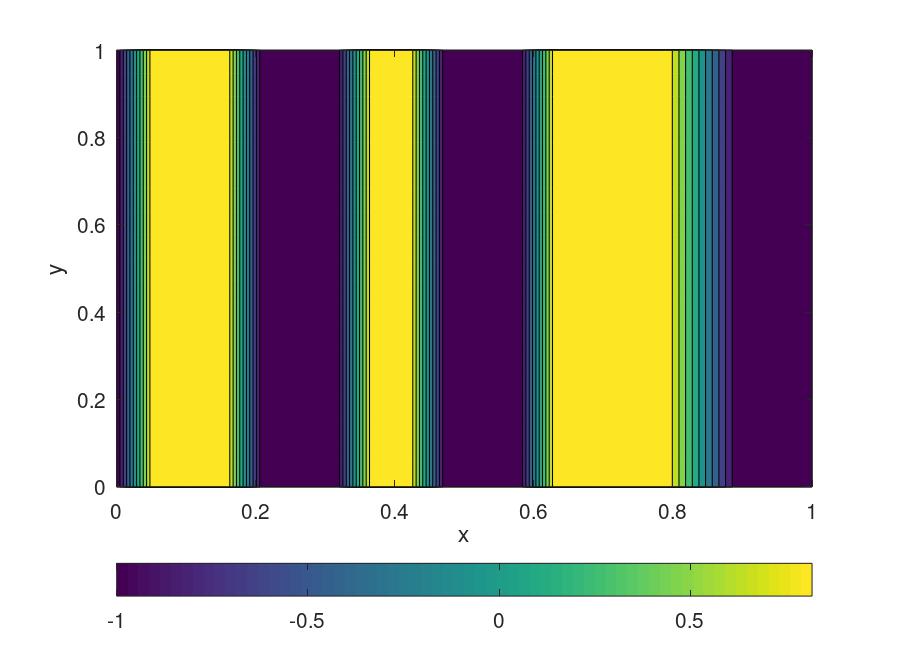
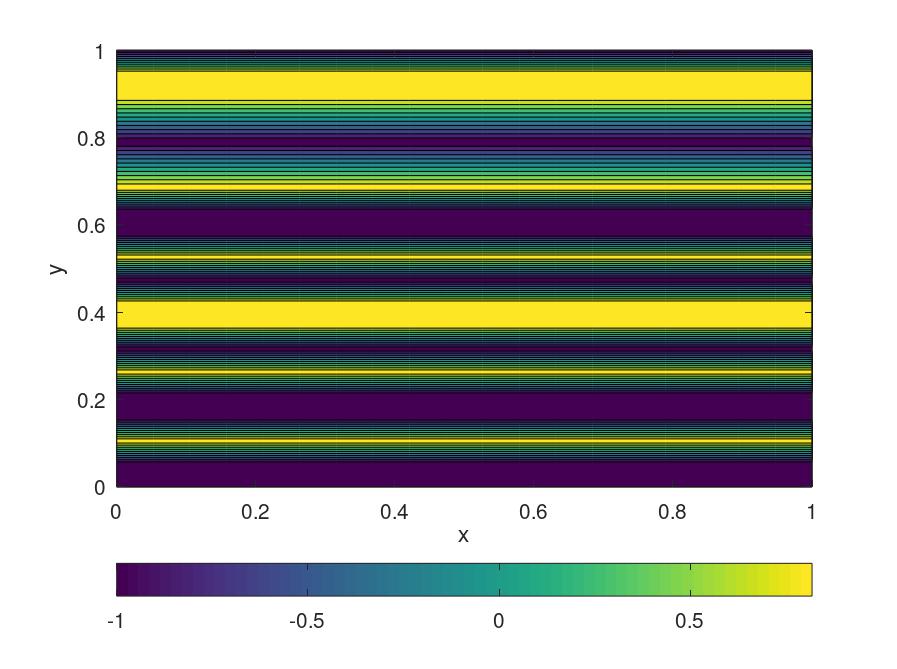
-3.9235 -290.928 664.09 -541.196 -576.975 2082.33 -2804.86 1293.28 2119.25 -5757.97 6428.76 -2713.37 -4621.86 11006.8 -11976.4 4733.41 7478.53 -18155.9

-3.933 -290.928 664.097 -541.205 -576.982 2082.33 -2804.86 1293.28 2119.26 -5757.97 6428.72 -2713.37 -4621.82 11006.8 -11976.4 4733.37 7478.54 -18155.9

-3.9235 -290.928 664.097 -541.196 -576.982 2082.33 -2804.86 1293.28 2119.26 -5757.98 6428.74 -2713.36 -4621.85 11006.8 -11976.4 4733.47 7478.54 -18156

-3.92635 -290.927 664.098 -541.199 -576.983 2082.33 -2804.86 1293.28 2119.25 -5757.98 6428.76 -2713.36 -4621.86 11006.8 -11976.4 4733.37 7478.58 -18155.9

-3.9235 -290.928 664.097 -541.196 -576.982 2082.33 -2804.86 1293.28 2119.26 -5757.98 6428.76 -2713.34 -4621.87 11006.8 -11976.4 4733.41 7478.57 -18156  
 **Heatmapa pochodnej X: **

**Heatmapa pochodnej Y:  
  
  
  
  
9.Określić monotoniczność.  
   
Monotoniczność X:   
  
  
  
Monotoniczność Y:  
  
**