Temat: Poszukiwanie minimum wartości funkcji w dwóch wymiarach metodą Newtona			
Wykonał: Marcin Fabrykowski	Wydział: FiIS	Kierunek Inf. Stos.	Grupa: grupa 3

1. Wstęp
   Poszukiwanie minimum funkcji dwóch zmiennych metodą Newtona odbywa się analogicznie do poszukiwania minimum funkcji kwadratowej. Sprowadzenie poszukiwanej funkcji dwóch zmiennym do funkcji jednej zmiennej realizujemy poprzez podstawienie $r = [x, y]$. Doprowadzamy pierwotne równanie do postaci

   $$g(r) = \frac{1}{2}r^T Ar + r^T b$$

   gdzie A to macierz Hessego, natomiast $b = [b_1, b_2]$ jest wektorem wyrazów wolnych.

2. Wykonanie ćwiczenia
   Badamy funkcję

   $$f(x, y) = x^2 - 4x + 8 + y^2 - 4y + xy$$

   zauważamy że czynnik +8 nie wpływa na kształt funkcji a jedynie na jej przesunięcie. Nie ma to dla położenia minimum, dlatego definiujemy nową funkcję

   $$g(x, y) = f(x, y) - 8 = x^2 - 4x + y^2 - 4y + xy$$

   dążymy do postaci $g(r) = \frac{1}{2}r^T Ar + r^T b$. Z założenia $r = [x, y]$.

   Obliczamy macierz Hessego: $A = \begin{bmatrix} \frac{\partial^2 g}{\partial x^2} & \frac{\partial^2 g}{\partial x \partial y} \\ \frac{\partial^2 g}{\partial x \partial y} & \frac{\partial^2 g}{\partial y^2} \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$

   Teraz obliczamy czynnik $\frac{1}{2}r^T Ar$

$$\frac{1}{2}r^TAr = \frac{1}{2}\begin{bmatrix}x & y\end{bmatrix}\begin{bmatrix}2 & 1\\ 1 & 2\end{bmatrix}\begin{bmatrix}x\\ y\end{bmatrix} = \frac{1}{2}\begin{bmatrix}x & y\end{bmatrix}\begin{bmatrix}2x+y\\ x+2y\end{bmatrix} = \frac{1}{2}\left(2x^2+xy+xy+2y^2\right) =$$
x^2+y^2+xy

Porównując to z funkcją $g(x,y)$, zauważamy, że brakującym elementem jest $(-4x-4y)$. Przyrównujemy więc to do wektora wyrazów wolnych:

$$r^Tb = \begin{bmatrix}x & y\end{bmatrix}\begin{bmatrix}b_1\\ b_2\end{bmatrix} = x\cdot b_1 + y\cdot b_2 = -4x-4y \Rightarrow b = \begin{bmatrix}-4 & -4\end{bmatrix}$$

Następnie wyliczamy wartości funkcji $g(x,y)$ w obszarze $x\in(-10,10)$ oraz $y\in(-10,10)$

Zadanie to wykonuje program:

Listing 1: main.cpp

```
#include <iostream>
#include <stdio.h>
#include "nrutil.h"
#include <nrutil.c>
#include <gaussj.c>
using namespace std;
float f(float x,float y);
//TYLKO MACIERZ 2x2!!!
float** mul(float** a,float** b);
//TYLKO 2x2 jako 2x1!!!
float dl(float** a);
float** grad(float x,float y);
float** sub(float **a,float **b);
int main()
{
        printf("1)\n");
        FILE *plot;
        plot=fopen("plot.dat","w");
        int i,j;
        for(i=-100;i<100;i++)
        {
                for(j=-100;j<100;j++)
                {
                        fprintf(plot,"%f %f 
                           %f\n",(float)i/10,(float)j/10,f((float)i/10,(float)j/10
                };
                fprintf(plot,"\n");
        };
        fclose(plot);
        printf("2)\n");
        float** A=matrix(1,2,1,2);
        float** I=matrix(1,2,1,2);
        float** J=matrix(1,2,1,2);
        A[1][1]=2;
        A[1][2]=1;
```

```
        A[2][1]=1;
        A[2][2]=2;
        I[1][1]=1;
        I[1][2]=0;
        I[2][1]=0;
        I[2][2]=1;
        J[1][1]=-1;
        J[2][1]=0;
        J[1][2]=0;
        J[2][2]=01;

        gaussj(A,2,I,2);
        printf("A:\n");
        for(i=0;i<2;i++)
        {
                for(j=0;j<2;j++)
                {
                        printf("%f ",A[i+1][j+1]);
                };
                printf("\n");
        };
        float **b=matrix(1,2,1,2);
        b[1][1]=-4;
        b[1][2]=-4;
        b[2][1]=0;
        b[2][1]=0;
        float** A2=mul(A,J);
        float **r=mul(A2,b);
        printf("r:\n");
        for(i=0;i<2;i++)
        {
                for(j=0;j<2;j++)
                {
                        printf("%f ",r[i+1][j+1]);
                };
                printf("\n");
        };
        printf("3)\n");

        free_matrix(r,1,2,1,2);
        free_matrix(b,1,2,1,2);
        free_matrix(A,1,2,1,2);
        free_matrix(I,1,2,1,2);
        return 0;
};
float f(float x,float y)
{
        return x*x-4*x+8+y*y-4*y+x*y;
};
```

```
float** mul(float** a,float **b)
{
        float** wynik=matrix(1,2,1,2);
        wynik[1][1]=a[1][1]*b[1][1]+a[2][1]*b[1][2];
        wynik[1][2]=a[1][1]*b[2][1]+a[2][1]*b[2][2];
        wynik[2][1]=a[1][2]*b[1][1]+a[2][2]*b[1][2];
        wynik[2][2]=a[1][2]*b[2][1]+a[2][2]*b[2][2];
        return wynik;
};
float dl(float** a)
{
        return sqrt(a[2][1]*a[2][1]+a[2][2]*a[2][2]);
};
float** grad(float x,float y)
{
        float** wynik=matrix(1,2,1,2);
        wynik[1][1]=2*x-4+y;
        wynik[2][1]=2*y-4+x;
        wynik[1][2]=0;
        wynik[2][2]=0;
        return wynik;
};
float** sub(float **a,float **b)
{
        float **wynik=matrix(1,2,1,2);
        wynik[2][1]=a[2][1] - b[2][1];
        wynik[2][2]=a[2][2] - b[2][2];
        wynik[2][1]=a[2][1] - b[2][1];
        wynik[2][2]=a[2][2] - b[2][2];
        return wynik;
};
```

w efekcie czego otrzymujemy dane, które przedstawione graficznie pozwalają nam określić przybliżone miejsce zerowe funkcji

3. Wnioski
   Metoda Newtona sprawdza się przy wyznaczaniu miejsc zerowych funkcji dwóch zmiennych.

Rysunek 1: wartości funkcji g(x,y)