Zadanie 18:

18*. Startując z kilku losowo wybranych punktów poczatkowych, spróbuj numerycznie znaleźć minima *czterowymiarowej* funkcji Rosenbrocka

$$f(x_1, x_2, x_2, x_4) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2 + 100(x_3 - x_2^2)^2 + 100(x_4 - x_3^2)^2 . \quad (15)$$

Kod w języku C++:

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

#define MAX 10   // maksymalny wymiar funkcji
#define No_of_Iter 120 // ilość iteracji
#define Max 64

double P0[MAX], Pmin[MAX];
double Ymin, Y0;
double H = 1.0;
double Err = 1.0;
double S[MAX], G[MAX];
int condiotion;

double Delta = 1e-64;   // tolerancja dla punktów
double Epsilon = 1e-64; // tolerancja dla wartości funkcji

double RosenbrockFunction(double *P){
    return ((1 - P[0]) * (1 - P[0]) + 100 * (P[1] - P[0] * P[0]) * (P[1] - P[0] * P[0]) + 100 * (P[2] - P[1] * P[1])
* (P[2] - P[1] * P[1]) + 100 * (P[3] - P[2] * P[2]) * (P[3] - P[2] * P[2]));
}

void Gradient(double *P){
    G[0] = (-2.0 + 2 * P[0] - 400 * P[0] * P[1] + 400 * P[0] * P[0] * P[0]);
    G[1] = (200 * P[1] - 200 * P[0] * P[0] - 400 * P[1] * P[2] + 400 * P[1] * P[1] * P[1]);
    G[2] = (200 * P[2] - 200 * P[1] * P[1] - 400 * P[2] * P[3] + 400 * P[2] * P[2] * P[2]);
    G[3] = (200 * P[3] - 200 * P[2] * P[2]);

    // "strona" gradientu
    S[0] = -G[0] / sqrt(G[0] * G[0] + G[1] * G[1] + G[2] * G[2] + G[3] * G[3]);
    S[1] = -G[1] / sqrt(G[0] * G[0] + G[1] * G[1] + G[2] * G[2] + G[3] * G[3]);
    S[2] = -G[2] / sqrt(G[0] * G[0] + G[1] * G[1] + G[2] * G[2] + G[3] * G[3]);
    S[3] = -G[3] / sqrt(G[0] * G[0] + G[1] * G[1] + G[2] * G[2] + G[3] * G[3]);
}

void Quadmin(int NC, double Delta, double Epsilon){
    double P1[MAX];
    double P2[MAX];
    double H0, H1, H2, Hmin, E0, E1, E2, Y1, Y2, D;
    int i, J;

    condiotion = 0;
    J = 0;

    for (i = 0; i < NC; i++){
        P1[i] = P0[i] + H * S[i];
        P2[i] = P0[i] + 2.0 * H * S[i];
    }

    Y1 = RosenbrockFunction(P1);
    Y2 = RosenbrockFunction(P2);

    while ((J < No_of_Iter) && (condiotion == 0)){
        if (Y0 <= Y1){
            Y2 = Y1;
            H = H / 2.0;

            for (i = 0; i < NC; i++){
                P2[i] = P1[i];
                P1[i] = P0[i] + H * S[i];
            }

            Y1 = RosenbrockFunction(P1);
```

```
        }

        else{
            if (Y2 < Y1){
                Y1 = Y2;
                H = 2.0 * H;

                for (i = 0; i < NC; i++){
                    P1[i] = P2[i];
                    P2[i] = P0[i] + 2.0 * H * S[i];
                }

                Y2 = RosenbrockFunction(P2);
            }else
                condiotion = -1;
        }
    }

    if (H < Delta)
        condiotion = 1;

    D = 4.0 * Y1 - 2.0 * Y0 - 2.0 * Y2;
    if (D < 0)
        Hmin = H * (4.0 * Y1 - 3.0 * Y0 - Y2) / D;

    else{
        condiotion = 4;
        Hmin = H / 3.0;
    }

    for (i = 0; i < NC; i++)
        Pmin[i] = P0[i] + Hmin * S[i];

    Ymin = RosenbrockFunction(Pmin);

    H0 = fabs(Hmin);
    H1 = fabs(Hmin - H);
    H2 = fabs(Hmin - 2.0 * H);

    if (H0 < H)
        H = H0;

    if (H1 < H)
        H = H1;

    if (H2 < H)
        H = H2;

    if (H < Delta)
        condiotion = 1;

    E0 = fabs(Y0 - Ymin);
    E1 = fabs(Y1 - Ymin);
    E2 = fabs(Y2 - Ymin);

    if (E0 < Err)
        Err = E0;

    else if (E1 < Err)
        Err = E1;

    else if (E2 < Err)
        Err = E2;

    else if ((E0 == 0) && (E1 == 0) && (E2 == 0))
        Err = 0;

    if (Err < Epsilon)
        condiotion = 2;

    if ((condiotion == 2) && (H < Delta))
        condiotion = 3;

    J++;
}

int main(){
    int NC = 4; // liczba składników gradientu
    int counter = 0;
    int i, j, ILE_PKT;
```

```c
    printf("--------------------------------------------------------------------------------------------------- \
n");
    printf("---------------------------------------- Metoda Gradientów ---------------------------------------- \
n");
    printf("--------------------------------------------------------------------------------------------------- \
n");
    printf("\nPodaj ilość losowych punktów początkowych: ");
    scanf("%d", &ILE_PKT);

    printf("\n");
    printf("\n");

    printf("--------------------------------------------------------------------------------------------------- \
n");
    printf("--- Rozpoczęto losowanie %d punktów w kwadracie [-100, 100]x[-100, 100]x[-100, 100]x[-100, 100] --- \n",
ILE_PKT);
    printf("--------------------------------------------------------------------------------------------------- \
n");

    printf("\n");
    printf("\n");

    srand(time(NULL));

    for (j = 0; j < ILE_PKT; j++){
        P0[0] = ((rand() % 201) - 100 + rand() / ((double)RAND_MAX));
        P0[1] = ((rand() % 201) - 100 + rand() / ((double)RAND_MAX));
        P0[2] = ((rand() % 201) - 100 + rand() / ((double)RAND_MAX));
        P0[3] = ((rand() % 201) - 100 + rand() / ((double)RAND_MAX));

        printf("Wylosowano punkt: \n");
        printf("(%.10lf, %.10lf, %.10lf, %.10lf) \n", P0[0], P0[1], P0[2], P0[3]);

        Y0 = RosenbrockFunction(P0);

        while ((counter < Max) && (H < Delta) || (Err > Epsilon)){
            Gradient(P0);
            Quadmin(NC, Delta, Epsilon);

            for (i = 0; i < NC; i++)
                P0[i] = Pmin[i];

            Y0 = Ymin;

            counter++;
        }

        printf("---------------------------------------------------------------------------------------------------
\n");
        printf("Minimum lokalne (lub globalne) jest w punkcie: \n");
        printf("(%.10lf, %.10lf, %.10lf, %.10lf) \n", Pmin[0], Pmin[1], Pmin[2], Pmin[3]);
        printf("Wartość funkcji w minimum to: %lf \n\n", Ymin);
        printf("---------------------------------------------------------------------------------------------------
\n");

        if (condiotion == 0)
            printf("Zbieżność nie osiagnieta ponieważ osiągnięto maksymalną ilość iteracji \n");

        if (condiotion == 1)
            printf("Osiagnieto zbieżność odciętej \n");

        if (condiotion == 2)
            printf("Osiagnieto zbieżność odciętej \n");

        if (condiotion == 3)
            printf("Osiagnieto zbieżność dla obu współrzędnych \n");

        if (condiotion == 4)
            printf("Zbieżność jest wątpliwa, ponieważ pojawiło się dzielenie przez 0 \n");
        printf("\n");
        printf("\n");
    }

    return 0;
}
```

Przykładowe wyniki dla 4 wylosowanych punktów początkowych:

```
-----------------------------------------------------------------------------------------------------
----------------------------------------- Metoda Gradientów -----------------------------------------
-----------------------------------------------------------------------------------------------------

Podaj ilość losowych punktów początkowych: 4


-----------------------------------------------------------------------------------------------------
--- Rozpoczęto losowanie 4 punktów w kwadracie [-100, 100]x[-100, 100]x[-100, 100]x[-100, 100] ---
-----------------------------------------------------------------------------------------------------


Wylosowano punkt:
(36.8974899034, 13.7917621759, -48.0638049641, 66.4625002036)
-----------------------------------------------------------------------------------------------------
Minimum lokalne (lub globalne) jest w punkcie:
(0.9999999764, 0.9999999527, 0.9999999054, 0.9999998108)
Wartość funkcji w minimum to: 0.000000

-----------------------------------------------------------------------------------------------------
Osiagnieto zbieżność odciętej


Wylosowano punkt:
(52.3267610880, 20.6633782739, 45.4796955420, -81.3452159838)
-----------------------------------------------------------------------------------------------------
Minimum lokalne (lub globalne) jest w punkcie:
(0.9999999764, 0.9999999527, 0.9999999054, 0.9999998108)
Wartość funkcji w minimum to: 0.000000

-----------------------------------------------------------------------------------------------------
Osiagnieto zbieżność odciętej


Wylosowano punkt:
(-95.2826315618, -27.4653894168, -8.5813441158, -32.7136263413)
-----------------------------------------------------------------------------------------------------
Minimum lokalne (lub globalne) jest w punkcie:
(0.9999999764, 0.9999999527, 0.9999999054, 0.9999998108)
Wartość funkcji w minimum to: 0.000000

-----------------------------------------------------------------------------------------------------
Osiagnieto zbieżność odciętej


Wylosowano punkt:
(70.2526402693, 54.8836335916, -28.6557417371, -73.2075637943)
-----------------------------------------------------------------------------------------------------
Minimum lokalne (lub globalne) jest w punkcie:
(0.9999999764, 0.9999999527, 0.9999999054, 0.9999998108)
Wartość funkcji w minimum to: 0.000000

-----------------------------------------------------------------------------------------------------
Osiagnieto zbieżność odciętej
```

Metoda: Metoda Gradientów

Dwuwymiarowa funkcja Rosenbrocka jest używana do przedstawiania zachowań algorytmów optymalizacji. Jej minimum globalne znajduje się w punkcie (x,y) = (1,1), a wartość funkcji w tym punkcie wynosi f(x,y) = 0.

Wzór tej funkcji:

$$f(x,y) = (1-x)^2 + 100(y-x^2)^2$$

A jej wielowymiarowym rozwinięciem jest:

$$f(x) = \sum_{i=1}^{N-1} \left[(1-x_i)^2 + 100(x_{i+1} - x_i^2)^2\right] \quad \forall x \in \mathbb{R}^N$$

Metoda gradientów jest iteracyjnym algorytmem wyszukiwania minimum zadanej funkcji.
Założenia:

1. funkcja jest ciągła i różniczkowalna
2. funkcja w badanej dziedzinie jest ściśle wypukła

Algorytm zaczyna się wyborem punktu startowego, w którym to obliczany jest kierunek poszukiwań rozwiązania. Jeżeli następny punkt nie spełnia warunku stopu algorytmu całe postępowanie jest powtarzane.

Kryterium stopu:

$$\|\nabla f(\mathbf{x_k})\| \leqslant \epsilon,$$
$$\|\mathbf{x_{k+1}} - \mathbf{x_k}\| \leqslant \epsilon.$$

ϵ to zadana precyzja, a **||.||** zadana norma.