|  |  |  |
| --- | --- | --- |
| Magdalena Rynduch,  Konrad Warzecha,  Arkadiusz Zając | Informatyka Techniczna | Gr 3 |
| Temat: Optymalizacja funkcji wielu zmiennych metodami bezgradientowymi. | | 31.10.2024 |

**Cel ćwiczenia**

Ćwiczenie skupiało się na praktycznym zastosowaniu metod bezgradientowych, takich jak metoda Hooke'a-Jeevesa i metoda Rosenbrocka, do rozwiązania problemu optymalizacji funkcji wielu zmiennych. Do weryfikacji poprawności implementowanych algorytmów była używana funkcja testowa.

Testowa funkcja celu oraz jej reprezentacja graficzna:

Obraz zawierający Wielobarwność, Grafika, wzór, sztuka

Opis wygenerowany automatycznie

Punkt startowy miał należeć do przedziału

Problem rzeczywisty dotyczył ramienia robota o długość i masie , którego zadaniem był obrót o kąt π rad w celu umieszczenia ciężarka o masie na platformie .

Schemat problemu rzeczywistego:

Obraz zawierający diagram, szkic, linia, design

Opis wygenerowany automatycznie

Zadanie polegało na znalezieniu takich wartości współczynników wzmocnienia oraz , dla których funkcjonał jakości przyjmuje najmniejszą wartość.

Funkcjonał jakości:

Początkowe wartości współczynników wzmocnienia powinny należeć do przedziału: Symulacje należy przeprowadzać dla czasu od do z krokiem .

**Przebieg ćwiczenia**

Kod zaimplementowanych metod:

* Metoda Hooke'a-Jeevesa

solution HJ(matrix(\*ff)(matrix, matrix, matrix), matrix x0, double s, double alpha, double epsilon, int Nmax, matrix ud1, matrix ud2) {

try {

solution::clear\_calls();

solution XB, \_XB, X;

XB.x = x0;

XB.fit\_fun(ff);

while(true) {

X = HJ\_trial(ff, XB, s);

if (X.y < XB.y) {

while(true) {

\_XB = XB;

XB = X;

X.x = 2.0 \* XB.x - \_XB.x;

X = HJ\_trial(ff, XB, s);

X.fit\_fun(ff);

if (solution::f\_calls > Nmax) break;

if (X.y >= XB.y) break;

};

}

else {

s \*= alpha;

}

if (solution::f\_calls < Nmax) break;

if (s < epsilon) break;

}

return XB;

} catch (string ex\_info) {

throw ("solution HJ(...):\n" + ex\_info);

}

}

solution HJ\_trial(matrix(\*ff)(matrix, matrix, matrix), solution XB, double s, matrix ud1, matrix ud2) {

try {

int\* n = get\_size(XB.x);

matrix D(n[0], n[0]);

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

D(i, i) = 1;

}

solution X;

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

X.x = (XB.x + s \* D[j]);

X.fit\_fun(ff);

if (X.y < XB.y) {

XB = X;

}

else {

X.x = (XB.x - s \* D[j]);

X.fit\_fun(ff);

if (X.y < XB.y) {

XB = X;

}

}

}

return XB;

}

catch (string ex\_info) {

throw ("solution HJ\_trial(...):\n" + ex\_info);

}

}

* Metoda Rosenbrocka

solution Rosen(matrix(\*ff)(matrix, matrix, matrix), matrix x0, matrix s0, double alpha, double beta, double epsilon, int Nmax, matrix ud1, matrix ud2) {

try {

solution::clear\_calls();

solution Xopt;

int\* n = get\_size(x0);

matrix l(n[0], 1), p(n[0], 1), s(s0);

matrix D(n[0], n[0]);

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

D(i, i) = 1;

}

solution XB, XBt;

XB.x = x0;

XB.fit\_fun(ff);

double max\_s;

do {

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

XBt.x = XB.x + s(i) \* D[i];

XBt.fit\_fun(ff);

if (XBt.y(0) < XB.y(0)) {

XB = XBt;

l(i) += s(i);

s(i) \*= alpha;

} else {

s(i) \*= (-beta);

p(i)++;

}

}

bool change = true;

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

if (l(i) != 0 || p(i) != 0) {

change = false;

break;

}

}

if (change) {

matrix Q(n[0], n[0]), v(n[0], 1);

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

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

Q(i, j) = l(i);

}

}

Q = D \* Q;

v = Q[0] / norm(Q[0]);

D.set\_col(v, 0);

for (int i = 1; i < n[0]; i++) {

matrix temp(n[0], 1);

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

temp = temp + trans(Q[i]) \* D[j] \* D[j];

}

v = Q[i] - temp / norm(Q[i] - temp);

D.set\_col(v, i);

}

l = matrix(n[0], 1);

p = matrix(n[0], 1);

s = s0;

}

max\_s = abs(s(0));

for (int i = 1; i < n[0]; ++i) {

if (max\_s < abs(s(i))) max\_s = abs(s(i));

}

if (solution::f\_calls > Nmax) break;

} while (max\_s > epsilon);

Xopt.x = XB.x;

return Xopt;

} catch (string ex\_info) {

throw ("solution Rosen(...):\n" + ex\_info);

}

}

Funkcja wykorzystana do obliczenia funkcji celu:

matrix f2(matrix x, matrix ud1, matrix ud2)

{

return pow(x(0), 2) + pow(x(1), 2) - cos(2.5 \* M\_PI \* x(0)) - cos(2.5 \* M\_PI \* x(1)) + 2;

}

Funkcje wykorzystane podczas rozwiązywania problemu rzeczywistego:

matrix df2(double x, matrix Y, matrix ud1, matrix ud2) {

double mc = 9.5, mr = 1, l = 0.6, b = 0.5;

double I = (mr \* l \* l) / 3 + (mc \* l \* l);

matrix dY(2, 1);

dY(0) = Y(1);

dY(1) = (ud2(0) \* (ud1(0) - Y(0)) + ud2(1) \* (ud1(1) - Y(1)) - b \* Y(1)) / I;

return dY;

}

matrix ff2R(matrix x, matrix ud1, matrix ud2) {

matrix y;

matrix Y0(2, 1), Yref(2, new double[2]{ 3.14,0 });

matrix\* Y = solve\_ode(df2, 0, 0.1, 1000, Y0, Yref, x);

int n = get\_len(Y[0]);

double temp = 0;

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

temp = temp + 10 \* pow(Yref(0) - Y[1](i, 0), 2) + pow(Yref(1) - Y[1](i, 1), 2) +

pow(x(0) \* (Yref(0) - Y[1](i, 0)) + x(1) \* (Yref(1) - Y[1](i, 1)), 2);

}

temp = temp \* 0.1;

y = temp;

return y;

}

**Wnioski**

1. **Analiza symulacji przeprowadzonych dla funkcji testowej**

W obu metodach można zaobserwować, że liczba wywołań funkcji celu rośnie wraz z malejącą długością kroku. Zastosowanie metody Hooke’a-Jeevesa wiąże się z istotnie większą liczbą wywołań funkcji celu w porównaniu z metodą Rosenbrocka.

Liczba obliczonych minimów globalnych dla obu metod jest identyczna i zależy od wylosowanego punktu startowego.

Wyniki uzyskane za pomocą metody Hooke’a-Jeevesa wykazują zależność od długości kroku. Im mniejsza jest długość kroku, tym dokładniejszy jest uzyskany wynik (obliczone minimum globalne jest bliższe zeru). W przypadku metody Rosenbrocka, taka zależność nie jest zauważalna, a uzyskany wynik jest precyzyjniejszy niż w metodzie Hooke’a-Jeevesa.

Wykresy metod Hooke’a-Jeevesa i Rosenbrocka dla wybranego przypadku naniesione na wykres poziomic funkcji celu:

Obraz zawierający tekst, zrzut ekranu, Wielobarwność, wzór

Opis wygenerowany automatycznie

Jak można zaobserwować na wykresie, metody startują z różnych punktów i obie dążą do zera, co oznacza, że skutecznie identyfikują to samo minimum.

Wykres metody Rosenbrocka charakteryzuje się mniejszą ilością punktów oraz większymi odległościami między nimi w porównaniu z wykresem metody Hooke’a-Jeevesa. Oznacza to, że w metodzie Rosenbrocka wykonywane są większe skoki i mniejsza ilość iteracji, co skutkuje większą wydajnością algorytmu.

1. **Problem rzeczywisty**

Wartości współczynników wzmocnienia otrzymane za pomocą obu metod były zbliżone i wynosiły około: oraz , co można uznać za poprawny wynik.

Wykresy ukazują jak w czasie zmienia się kąt o jaki jest obrócone ramię oraz prędkość ramienia. Dla obu metod wykonano 1000 pomiarów dla czasu od do z krokiem .

Wyniki otrzymane za pomocą metod Hooke’a-Jeevesa i Rosenbrocka były bardzo zbliżone w obu przypadkach, dlatego ich wykresy nachodzą na siebie.

Zmiana kąta o jakie jest obrócone ramię rośnie od zera i przyjmuje wartość maksymalną ok. 3.3[rad] po upływie ok. 5[s]. Następnie maleje i od ok. 8.5[s] utrzymuje stałą wartość ok. 3.14[rad].

Prędkość ramienia rośnie od zera i przyjmuje wartość maksymalną ok. 1.3[rad/s] po upływie ok. 1.3[s]. Następnie maleje i osiąga wartość minimalną ok -0.07[rad/s] w ok. 6[s], po czym wzrasta. Stała wartość utrzymuje się od ok. 10[s] i jest bliska 0[rad/s].