Optymalizacja IT gr. 2

Dariusz Homa

Wojciech Jurgielewicz

Michał Koleżyński

Projekt 5

Cel ćwiczenia

Celem ćwiczenia jest zapoznanie się z problematyką optymalizacji wielokryterialnej i wyznaczenie rozwiązań minimalnych w sensie Pareto.

Wykonanie

Kody funkcji

Metoda Powella

```
solution Powell(matrix(*ff)(matrix, matrix, matrix), matrix x0, double
epsilon, int Nmax, matrix ud1, matrix ud2)
{
  try
  {
    solution Xopt;

    int n = get_len(x0);

    matrix a(n, 2);
    matrix d = ident_mat(n);

    solution X, p, h;
    X = x0;

    double* section;

    while (true)
    {
      p = X;

      for (int i = 0; i < n; i++)
      {
        a.set_col(p.x, 0);
        a.set_col(d[i], 1);
        section = expansion(ff, 0, 1, 1.2, Nmax, ud1, a);
        h = golden(ff, section[0], section[1], epsilon, Nmax, ud1, a);
        p.x = p.x + h.x * d[i];
      }

      if (norm(X.x - p.x) < epsilon)
      {
        Xopt = p;
        Xopt.fit_fun(ff, ud1, ud2);

        Xopt.flag = 0;
        break;
      }

      if (solution::f_calls > Nmax)
      {
        Xopt = p;
        Xopt.fit_fun(ff, ud1, ud2);
        throw std::string("Przekroczono limit wywolan funkcji Powell :(");
      }

      for (int i = 0; i < n - 1; ++i)
        d.set_col(d[i + 1], i);
```

```
  d.set_col(p.x - X.x, n - 1);
      a.set_col(p.x, 0);
      a.set_col(d[n - 1], 1);

      section = expansion(ff, 0, 1, 1.2, Nmax, ud1, a);

      h = golden(ff, section[0], section[1], epsilon, Nmax, ud1, a);

      X = p.x + h.x * d[n - 1];
    }

    return Xopt;
  }
  catch (string ex_info)
  {
    throw("solution Powell(...):\n" + ex_info);
  }
}
```

Testowa funkcja celu

$$f_{1(x_1,x_2)} = a((x_1 - 2)^2 + (x_2 - 2)^2)$$

$$f_{2(x_1,x_2)} = \frac{1}{a}((x_1 + 2)^2 + (x_2 + 2)^2)$$

Ograniczenia:

$$x_1 \in [-10{,}10], x_2 \in [-10{,}10]$$

$$a \in \{1, 10, 100\}$$

Cel

- Wykonanie 101 optymalizacji w ramach znalezienia optymalnego rozwiązania funkcji celu

Parametry

- Dokładność $\epsilon = 10^{-3}$
- Maksymalna liczba wywołań funkcji: 10^4

Kod

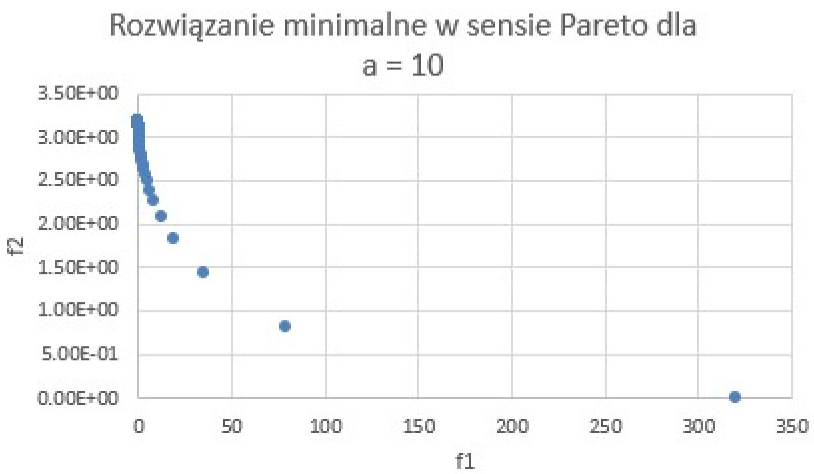
Funkcja celu

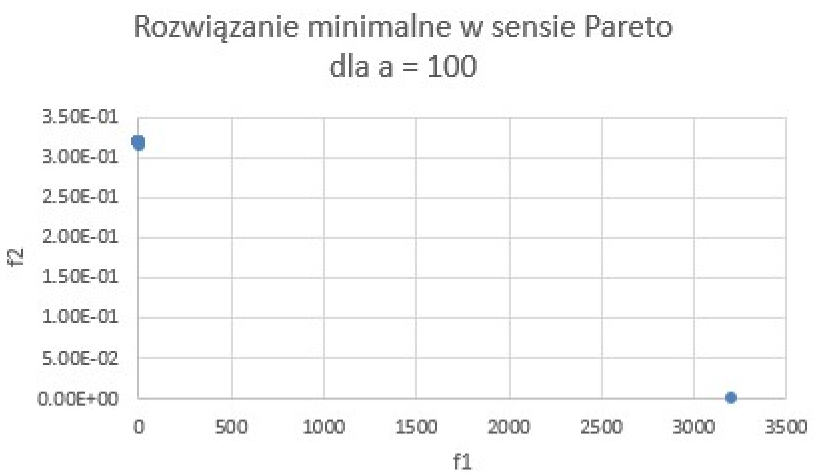
```
matrix fT5(matrix x, matrix ud1, matrix ud2)
{
  matrix y;

  if (isnan(ud2(0, 0)))
  {
    y = matrix(2, 1);
    y(0) = ud1(1) * (pow(x(0) - 2, 2) + pow(x(1) - 2, 2));
    y(1) = 1.0 / ud1(1) * (pow(x(0) + 2, 2) + pow(x(1) + 2, 2));
  }
  else
  {
    const double w = ud1(0);
    matrix yt;
    yt = fT5(ud2[0] + x * ud2[1], ud1);
    y = w * yt(0) + (1 - w) * yt(1);
  }

  return y;
}
```

100 optymalizacji

```
for (double w = 0.0; w <= 1.0; w += 0.01)
  {
    matrix x0(2, 1);
    x0(0) = RandomNumberGenerator::Get().Double(-10, 10);
    x0(1) = RandomNumberGenerator::Get().Double(-10, 10);

    for (double a : a_tab)
    {
      ud1(0) = w;
      ud1(1) = a;

      solution result1 = Powell(fT5, x0, epsilon, Nmax, ud1);

      SAVE_TO_FILE("Test-" + std::to_string(a) + ".txt") << x0(0) << ";" <<
x0(1)<< ";" << result1.x(0) << ";" << result1.x(1) << ";" << result1.y(0) <<
";" << result1.y(1) << ";" << solution::f_calls << "\n";
      solution::clear_calls();
    }
  }
```

Omówienie wyników





Dla wyższych wartości współczynnika a rozwiązania minimalne są bardziej skupione. Jak widać ze wzorów, f1 jest proporcjonalne do a, natomiast f2 jest proporcjonalne do 1/a. Dla większego współczynnika a, wartości f1 są przez to rzędów wielkości wyższe od wartości f2. Ponieważ współczynnik wagi w skaluje funkcje f1 i f2 liniowo aby otrzymać funkcję jednokryterialną f(x) = w * f1(x) + (1-w) * f2(x), a wartości f1 są znacznie większe od f2 dla a=10 i a=100, optymalizacja przede wszystkim skierowana jest ku minimalizacji wartości f1 aby otrzymać rozwiązanie optymalne w sensie Pareto. W przypadku a = 100, wszystkie punkty są skupione przy minimum f1, z wyjątkiem tego odpowiadającego w = 0, gdyż wtedy f = f2 i przez to metoda Powella minimalizuje wyłącznie f2.

Wynika z tego fakt, iż przy minimalizacji funkcji wielokryterialnych metodą Powella znaczenie ma względny rozmiar wartości funkcji – rozwiązanie optymalne w sensie Pareto będzie zbliżone do rozwiązania optymalnego tej funkcji, które bardziej wpływa na kryterium ważone.

Problem rzeczywisty

Cel

Wykonanie 101 optymalizacji w celu znalezienia najbardziej optymalnych wymiarów belki dla których maksymalne możliwe ugięcie jest największe.

Parametry

- Gęstość $\rho = 7800 \frac{kg}{m^3}$
- Siła $P = 1000N$
- Moduł Younga $E = 207GPa$
- Maksymalne ugięcie belki $u_{max} = 5mm$
- Maksymalne naprężenie $\sigma_{max} = 300MPa$
- Dokładność 10^{-3}
- Maksymalna liczba wywołań funkcji $N_{max} = 10^4$

Ograniczenia

- Długość belki: $l \in [200,1000]mm$
- Średnica belki: $d \in [10,50]mm$

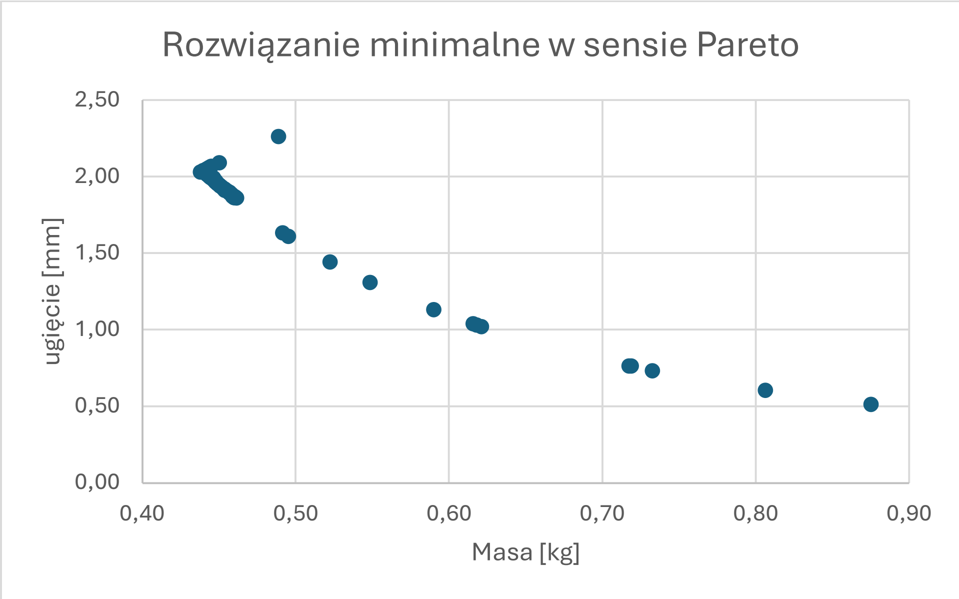
Kod

```
matrix fR5(matrix x, matrix ud1, matrix ud2)
{
  matrix y;

  const double rho = 7800;  // Gestosc (kg/m3)
  const double P = 1000;    // Dzialajaca sila (N)
  const double E = 207e9;   // Modul Younga (Pa)

  if (isnan(ud2(0, 0))) {
    const double l = x(0);
    const double d = x(1);

    y = matrix(3, 1);
    y(0) = rho * l * M_PI * (pow(d, 2) / 4);          // Masa
    y(1) = (64.0 * P * pow(l, 3)) / (3 * E * M_PI * pow(d, 4)); // Ugiecie
    y(2) = (32.0 * P * l) / (M_PI * pow(d, 3));         // Naprezenie
maksymalne
  }
  else
  {
    matrix xt = ud2[0] + x * ud2[1];
    matrix yt = fR5(xt, ud1);

    y = ud1 * (yt(0) - 0.12) / (15.3 - 0.12) + (1 - ud1) * (yt(1) - 4.2e-5) /
(3.28 - 4.2e-5);

    const double c = 1e10;

    if (xt(0) < 0.2) y = y + c * pow(0.2 - xt(0), 2);
    if (xt(0) > 1) y = y + c * pow(xt(0) - 1, 2);
    if (xt(1) < 0.01) y = y + c * pow(0.01 - xt(1), 2);
    if (xt(1) > 0.05) y = y + c * pow(xt(1) - 0.05, 2);
    if (xt(1) > 0.005) y = y + c * pow(xt(1) - 0.005, 2);

    if (yt(1) > 0.005) y = y + c * pow(yt(1) - 0.005, 2);
    if (yt(2) > 300e6) y = y + c * pow(yt(2) - 300e6, 2);
  }

  return y;
}
```

```
matrix ud1(1);

for (double w = 0.0; w <= 1.00; w += 0.01)
{
  ud1(0) = w;

  matrix x0(2, 1);
  x0(0) = RandomNumberGenerator::Get().Double(0.2, 1);    // l <200, 1000>
  x0(1) = RandomNumberGenerator::Get().Double(0.01, 0.05);  // d <10, 50>

  solution result1 = Powell(fR5, x0, epsilon, Nmax, ud1);

  SAVE_TO_FILE("Real.txt") << x0(0) * 1000 << ";" << x0(1) * 1000 << ";" <<
result1.x(0) * 1000 << ";" << result1.x(1) * 1000 << ";" << result1.y(0) <<
";" << result1.y(1) * 1000 << ";" << solution::f_calls << "\n";
  solution::clear_calls();
}
```

Omówienie wyników

- Dla zadanego problemu rzeczywistego dla danych początkowych oraz początkowych przedziałów wartości otrzymujemy optymalne wymiary belki równe przy uśrednieniu i odrzuceniu wyników zbyt odbiegających od reszty $l = 200{,}28\ mm$ oraz $d = 19{,}95\ mm$
- Niektóre wyniki stanowczo odbiegały od innych, dlatego też wyliczona z nich masa oraz ugięcie zaburzały wynik, dlatego zostały one odrzucone.
- Z wyliczonych wartości masy i ugięcia widzimy, że wraz ze wzrostem masy jej ugięcie maleje, co ma sens, gdyż łatwiej jest ugiąć linijkę plastikową niż linijkę drewnianą o takich samych wymiarach (co ma bezpośrednie odniesienie do gęstości, oraz samych właściwości materiału).



Wnioski

- Metoda Powella jest wrażliwa na zmianę danych wejściowych. Porównując otrzymane rozmiary belki dla problemu rzeczywistego ciężko znaleźć jest parę identycznych wartości, niektóre wyniki odstają wręcz od reszty.
- Z otrzymanych wyników otrzymujemy wykres przedstawiający realną zależność między wartościami. Rozwiązania optymalne w sensie Pareto na wykresie są w większości ułożone wzdłuż krzywej, co oznacza że wraz ze wraz ze wzrostem masy zmniejsza się ugięcie belki.
- Metoda Powella jest w stanie dokonać optymalizacji wielokryterialnej, o ile pierwotnie funkcje optymalizowane zostaną sprowadzone to funkcji ważonej. Optymalne rozwiązanie będzie zależeć od przypisanym wagom obu funkcji. Ponadto, funkcja celu przyjmująca największe wartości będzie najznaczniej wpływać na wartość funkcji ważonej, zatem rozwiązanie optymalne funkcji ważonej w sensie Pareto będzie zbliżone w stronę optimum tej konkretnej funkcji celu.