Akademia Ekonomiczna we Wrocławiu
Wydział Gospodarki Regionalnej i Turystyki
w Jeleniej Górze

Andrzej Dudek

# Programowanie geometryczne jako narzędzie optymalizacji w ekonomii

Praca doktorska
napisana pod kierunkiem
dr hab. Michała Montygierda-Łoyby prof. AE
w Katedrze Ekonometrii i Informatyki

Jelenia Góra 1999 r.

# Spis rzeczy

**Wstęp** 4

**1 Programowanie geometryczne - podstawy matematyczne** 6
1.1 Programowanie geometryczne jako gałąź programowania matematycznego . . . . . 6
1.2 Przedmiot programowania geometrycznego - podstawowe definicje . . . . . 9
1.3 Podstawowe twierdzenia dotyczące programowania geometrycznego . . . . . 12
1.3.1 Twierdzenie I (o średniej arytmetycznej i geometrycznej) 12
1.3.2 Twierdzenie II (Nierówność geometryczna) . . . . . 13
1.3.3 Twierdzenie III (podstawowy lemat programowania geometrycznego) . . . . . 17
1.3.4 Twierdzenie IV (o warunku wypukłości funkcji) . . . . 20
1.3.5 Nierówność spełniana przez funkcje wypukłe . . . . . 21
1.3.6 Wniosek . . . . . 22
1.3.7 Dowód wklęsłości logarytmu funkcji dualnej . . . . . 23
1.4 Twierdzenie V (lemat Farkasa) . . . . . 25
1.5 Twierdzenie VI (Kuhna - Tuckera) . . . . . 28
1.6 Dualność programowania geometrycznego . . . . . 33
1.7 Zastosowanie metod programowania geometrycznego w wybranych problemach . . . . . 37

**2 Algorytmy programowania geometrycznego** 69
2.1 Zastosowanie metody Newtona-Raphsona w programowaniu geometrycznym . . . . . 69
2.2 Algorytm "Infeasible Interior Point" . . . . . 73

2.3 Opis programu PG.EXE jako implementacji algorytmu „Infeasible Interior Point" . . . . . 76
2.3.1 Format pliku wejściowego . . . . . 77
2.3.2 Zawartość pliku wyjściowego . . . . . 79

**3 Zastosowanie programowania geometrycznego w ekonomii 82**
3.1 Funkcje użyteczności Cobba - Douglasa a programowanie geometryczne . . . . . 82
3.1.1 Funkcje Cobba - Douglasa . . . . . 82
3.1.2 Funkcje typu Cobba - Douglasa wielu zmiennych . . . 83
3.2 Wykorzystanie metod programowania geometrycznego do minimalizacji kosztów . . . . . 85
3.2.1 Optymalne wykorzystanie materiałów . . . . . 86
3.2.2 Minimalizacja funkcji kosztów przy nieliniowych ograniczeniach. . . . . 100

**4 Rozszerzenia programowania geometrycznego 104**
4.1 Generalizacja programowania geometrycznego . . . . . 105
4.2 Programowanie geometryczne a szereg Taylora . . . . . 107
4.3 Funkcje o „pozymianowym" szeregu Taylora . . . . . 109

**Zakończenie 111**

**Literatura 113**

**Aneks - Treść programu PG.CPP 120**

# Wstęp

Programowanie geometryczne jest ciekawą, choć niezbyt jeszcze w Polsce znaną techniką znajdowania ekstremów funkcji. Nie zdobyło wprawdzie takiej popularności jak programowanie liniowe czy inne gałęzie programowania matematycznego, ale ze względu na swoją strukturę dobrze nadaje się do opisu problemów optymalizacyjnych, występujących w naukach ekonomicznych, jak np. optymalizacja kosztów.

W najprostszej formie znalezienie ekstremum funkcji przy pomocy technik programowania geometrycznego jest równoważne z rozwiązaniem układu równań. Zadania bardziej złożone mogą dobrze opisywać problemy z różnych dziedzin nauki (p. [4], [10], [16], [23], [24], [30], [82], [93]).

Celem tej pracy jest przedstawienie metod programowania geometrycznego oraz obszarów, w którym może ono znaleźć zastosowanie ze szczególnym uwzględnieniem problemów związanych z ekonomią. W polskiej literaturze dotyczącej badań operacyjnych, a w szczególności programowania nieliniowego istaniała luka dotycząca tego tematu, którą niniejsze opracowanie stara się wypełnić.

Praca składa się z czterech rozdziałów.

W pierwszym rozdziale przedstawiono podstawy matematyczne programowania geometrycznego, jego miejsce wśród innych gałęzi programowania matematycznego, najważniejsze twierdzenia na których jest ono oparte oraz kluczowe dla programowania geometrycznego twierdzenie o jego dualności. Ponadto w rozdziale znajduje się wiele przykładów zastosowań metod programowania geometrycznego do "klasycznych" problemów optymalizacyjnych. Autor się starał przedstawić programowanie geometryczne jako alternatywne narzędzie, pozwalające na rozwiązanie zadań maksymalizacji / minimalizacji bez odwoływania się do aparatu rachunku różniczkowego.

Rozdział drugi przedstawia praktyczną implementację programowania geo-

metrycznego. Została w nim przedstawiona grupa algorytmów wykorzystujących metodę Newtona do rozwiązania układu równań Karusha - Kuhna - Tuckera dla funkcji dualnej. Szczególny nacisk został położony na algorytm "Infeasible Interior Point" zaproponowany w 1994 w [60], który jest w chwili obecnej jednym z najbardziej efektywnych algorytmów programowania geometrycznego o złożoności obliczeniowej porównywalnej z algorytmami programowania liniowego.

Rozdział trzeci jest poświęcony zastosowaniom technik programowania geometrycznego w ekonomii. Zostały one podzielone na trzy zasadnicze grupy: optymalizację funkcji Cobba-Douglasa, optymalne wykorzystanie materiału oraz minimalizację funkcji kosztów przy nieliniowych ograniczeniach.

Rozdział czwarty to próby rozszerzenia programowania geometrycznego na funkcje inne niż wielomiany o dodatnich składnikach. Oprócz "klasycznych" wielomianowych rozszerzeń programowania geometrycznego (p. [30]) autor postuluje wykorzytanie metod p.g. do funkcji, których rozwinięcie w szereg Taylora składa się z pozymianów.

Wszystkie przykłady w pracy są dziełem autora. Niektóre z nich to znane zadania optymalizacyjne, jednak sposób ich rozwiązania przy pomocy programowania geometrycznego pochodzi od autora.

Integralną częścia pracy jest ponadto napisany w języku C++ program komputerowy PG.CPP, służący do optymalizacji przy użyciu programowania geometrycznego.

Korzystam z okazji aby złożyć podziękowania Panu Profesorowi Michałowi Montygierdowi - Łoybie za inspirację i pomoc okazaną w trakcie pisania pracy.

Osobne słowa podziękowania kieruję w stronę Pana Profesora Marka Walesiaka za cenne uwagi redakcyjne, szczególnie w ostatnim okresie pisania pracy.

# Rozdział 1

# Programowanie geometryczne - podstawy matematyczne

## 1.1 Programowanie geometryczne jako gałąź programowania matematycznego

**Program matematyczny** jest to problem optymalizacyjny polegający na (w formie standardowej) maksymalizacji funkcj f(x): $x \in X$ przy warunkach $g(x) \leq 0$, $h(x) = 0$, gdzie X jest podzbiorem $R^n$ i jest dziedziną funkcji f(x), g(x) i h(x).
f(x) nazywamy funkcją optymalizowaną (*ang. objective function*)
g(x),h(x) nazywamy funkcjami ograniczającymi (*ang. constrain functions*)

**Programowanie matematyczne** to dział matematyki stosowanej zajmujący się modelami optymalizacji decyzji w oparciu o ekstremalne własności funkcji wielu zmiennych i traktujący m.in. o takich zagadnieniach, jak: [1]

- twierdzenia o formie rozwiązań programów matematycznych, w szczególności twierdzenia o istnieniu bądź nieistnieniu rozwiązań w określonych sytuacjach;

- formulowanie praktycznych problemów jako programów matematycznych oraz porównywanie jakości takich formułowań;

[1] p. „Mathematical Programming - Glossary" dostępne w internecie pod adresem http://dy.fce.vutbr.cz/

- algorytmy szukające rozwiązań programu matematycznego lub stwierdzające, że nie ma rozwiązania optymalnego;

- twierdzenia dotyczące struktur modeli używanych w algorytmach, w szczególności ich własności, wykonalności, redundancji itp;

- analiza wyników, w tym sytuacji takich jak rozwiązania niewłaściwe;

- twierdzenia dotyczące błędów przybliżeń rozwiązań programów matematycznych wynikających z przyjęcia modelu nie oddającego w pełni problemu, błędów obliczeniowych komputerów, dokładości algorytmów;

- badania łączące programy matematyczne z innymi działami wiedzy w szczególności z rozwojem technik komputerowych.

Istnieje wiele odmian programowania matematycznego ze względu na strukturę funkcji optymalizowanej i funkcji ograniczających. Wśród najważniejszych z nich można wyróżnić m.in. następujące (por. [6], [11], [12], [13] str. 18-120, [21], [26], [27], [28], [36], [37], [40], [41], [43] str. 331-543, [44], [62], [63], [68], [69], [80] str. 13-104, [87]):

- bi-wypukłe (*biconvex*)
- bi-liniowe (*bilinear*)
- całkowitoliczbowe
- całkowitoliczbowe mieszne
- dynamiczne
- geometryczne (*geometric*)
- kwadratowe
- liniowe
- liniowe mieszane
- nieskończone
- pseudo-boolowskie (zero-jedynkowe)
- rozłączne
- semi-nieskończone
- semi-oznaczone
- wypukłe
- wypukłe odwrotne
- wypukłe złożone

Szczególnie popularne i często stosowane również w ekonomii ([19] str.221-

263, [64] str. 151) jest programowanie liniowe polegające na optymalizacji funkcji [2]

$$f(x) = c \cdot x$$

przy warunkach ograniczających:

$$Ax = b\,,\; x \geq 0$$

Typowe zadanie programowania geometrycznego może wyglądać następująco: znaleźć maksimum (p. [19] str 228.) funkcji

$$20 \cdot x_1 \quad +18 \cdot x_2 \quad +12 \cdot x_3 \quad +15 \cdot x_4 \quad +10 \cdot x_5$$

$$\begin{array}{lllll} 5 \cdot x_1 & +4 \cdot x_2 & & +2 \cdot x_4 & \leq 3500 \\ 4 \cdot x_1 & +3 \cdot x_2 & +1,5 \cdot x_3 & +2,5 \cdot x_4 & +2 \cdot x_5 \leq 3000 \\ 3 \cdot x_1 & +2 \cdot x_2 & +4 \cdot x_3 & +3 \cdot x_4 & +2 \cdot x_5 \leq 2400 \end{array}$$

Do rozwiązywania zadań programowania liniowego najczęściej używa się którejś z odmian popularnej metody simpleks (p. [80] str 18.)
W naszym przypadku rozwiązaniem optymalnym jest

$$x_1 = 0\,,\; x_2 = 875\,,\; x_3 = 110\,,\; x_4 = 0\,,\; x_5 = 105.$$

Programowanie gemetryczne, mimo podobieństwa w formie do programowania liniowego jest odmianą innego rodzaju programowania matematycznego, mianowicie programowania wypukłego (p. 2.1)

Program wypukły to (p. [43] str. 508) taki program matematyczny w którym funkcja optymalizowana jest wklęsła (lub w przypadku minimalizacji wypukła) zaś wszystkie funkcje ograniczające wypukłe.

W przeciwieństwie do programowania liniowego nie ma standardowej metody rozwiązania problemu programowania nieliniowego (w tym wypukłego). Algorytmy programowania wypukłego możemy podzielić na trzy główne grupy: (p. [43] str. 535)

- gradientowe;
- sekwencyjne, bez warunków ograniczających;
- sekwencyjne, aproksymacyjne.

W 2.1 zapoznamy się z przykładem algorytmu gradientowego zastosowanego dla problemów programowania geometrycznego.

[2] p.[43] str.35, [62], [80] str. 14, [91], [92]

## 1.2 Przedmiot programowania geometrycznego - podstawowe definicje

[3]

Niech $g_k(t)$, $k \in \{0, 1, 2, \ldots, p\}$ będą odwzorowaniami $m$-wymiarowej przestrzeni Euklidesowej w zbiór liczb rzeczywistych. Zdefiniujmy następujące dwa pojęcia, podstawowe dla programowania geometrycznego:

***I.*** Program pierwotny (primal program):

Znaleźć minimalną wartość funkcji $g_o(t)$ przy następujących ograniczeniach:

$$g_1(t) \leq 1,\ g_2(t) \leq 1,\ \ldots,\ g_p(t) \leq 1\,, \tag{1.1}$$

$$t_i > 0,\ i \in \{1, 2, \ldots, m\}\,,$$

gdzie $g_k(t)$ są tzw. pozymianami tj. funkcjami określonymi następująco:

$$g_k(t) = \sum_{i \in J[k]} c_i t_1^{a_{i1}} \cdot t_2^{a_{i2}} \cdots t_m^{a_{im}} \quad k \in \{0, 1, 2, \ldots, p\}\,,$$

przy tym:

$$J[k] = m_k, m_k + 1, m_k + 2, \ldots, n_k k \in \{0, 1, 2, \ldots, p\}\,,$$

$$m_0 = 1, m_1 = n_0 + 1, m_2 = n_1 + 1, \ldots, m_p = n_{p-1} + 1, n_p = n\,.$$

Dodajmy, że o ile wykładniki $a_{ij}$ mogą być dowolnymi liczbami rzeczywistymi, to współczynniki przy $c_i$ przy odpowiednich wyrazach pozymianów muszą być dodatnie (stąd nazwa „pozymian").

Każdemu programowi pierwotnemu odpowiada program dualny.

***II*** Program dualny (dual program):

Znaleźć maksymalną wartość funkcji $v(\delta)$ określonej następująco:

$$v(\delta) = [\prod_{i=1}^{n} (\frac{c_i}{\delta_i})^{\delta_i}] \prod_{k=1}^{p} \lambda_k(\delta)^{\lambda_k(\delta)}\,,$$

[3] p. [30] str. 1-4

gdzie:

$$\lambda_k(\delta) = \sum_{i \in J[k]} \delta_i \,,\; k \in \{0, 1, 2, \ldots, p\} \,.$$

Zbiory $J[k]$ są określone tak jak w programie pierwotnym. Współczynniki $c_i$ są dodatnie, a wektor zmiennych dualnych $\delta = (\delta_1, \delta_2, \ldots, \delta_n)$ spełnia następujące ograniczenia (przy założeniu, że dla $x = 0$ jest $x^x = x^{-x} = 1$):

$$\delta_i \geq 0 \,,\; i \in \{1, 2, \ldots, n\} \,,$$

warunek normalności:

$$\sum_{i \in J[0]} \delta_i = 1$$

i warunki ortogonalności:

$$\sum_{i=1}^{n} a_{ij} \cdot \delta_i = 0 \;\;,\; j \in \{1, 2, \ldots, m\} \,. \tag{1.2}$$

Sposób przechodzenia od programu pierwotnego do programu dualnego prześledzimy na przykładzie. Wcześniej jeszcze należy wprowadzić ważne pojęcie stopnia złożoności problemu (*DOD - Degree of Difficulty*). Jest on określony następująco: Stopień złożoności = łączna liczba wyrazów we wszystkich pozymianach - liczba zmiennych - 1. Jeśli DOD problemu jest równe 0, to znalezienie jego rozwiązania sprowadza się do rozwiązania układu równań liniowych. Takiego typu zadanie jest przedstawione w przykładzie 1.

**Przykład 1**

Znaleźć najmniejszą wartość funkcji

$$U_1(x, y) = \sqrt{x^2 + y^2} + \frac{1}{x^2 y} \,.$$

Aby sprowadzić ten problem do typowego zadania programowania geometrycznego należy wprowadzić dodatkową zmienną $z$ ograniczoną związkiem

$$z \geq x^2 + y^2 \,.$$

Ostatecznie więc program pierwotny możemy sformułować następująco: Zminimalizować funkcję

$$U = z^{\frac{1}{2}} + \frac{1}{x^2}y$$

przy założeniu, że

$$\frac{x^2}{x} + \frac{y^2}{z} \leq 1$$

i

$$x > 0, y > 0, z > 0\,.$$

Jest to zadanie programowania geometrycznego o 3 zmiennych i łącznie 4 składnikach pozymianów, a więc o stopniu złożoności równym 0 $(4-3-1)$. Problem ten sprowadza się do rozwiązania układu równań liniowych.

Program dualny dla tego problemu przedstawia się następująco:

Zmaksymalizować funkcję $v(\delta)$ określoną wzorem

$$v(\delta) = (\frac{1}{\delta_1})^{\delta_1}(\frac{1}{\delta_2})^{\delta_2}(\frac{1}{\delta_3})^{\delta_3}(\frac{1}{\delta_4})^{\delta_4}(\delta_3+\delta_4)^{(\delta_3+\delta_4)}\,,$$

gdzie nieujemne parametry $\delta_1, \delta_2, \delta_3, \delta_4$ spełniają warunki normalności:

$$\delta_1 + \delta_2 = 1$$

i ortogonalności

$$\frac{1}{2}\delta_1 - \delta_3 - \delta_4 = 0\,,$$
$$-2\delta_2 + 2\delta_3 = 0\,,$$
$$-\delta_2 + 2\delta_4 = 0\,.$$

Jak widać, jedyny wektor, który spełnia ograniczenia, ma współrzędne:

$$\delta_1 = \frac{3}{4}\,,$$

$$\delta_2 = \frac{1}{4}\,,$$

$$\delta_3 = \frac{1}{4}\,,$$

$$\delta_4 = \frac{1}{8}.$$

Maksymalną wartością $v$ jest więc $3^{\frac{9}{8}} \cdot 2^{-4}$.

Jak to będzie wynikać z twierdzeń przytoczonych poniżej, maksymalna wartość funkcji dualnej $V$ jest równocześnie minimalną wartością funkcji pierwotnej $U$. Co więcej, następne twierdzenia dają możliwość znalezienia zmiennych minimalizujacych program pierwotny na podstawie wartości funkcji $V$.

## 1.3 Podstawowe twierdzenia dotyczące programowania geometrycznego

Programowanie geometryczne opiera się na tzw. nierówności geometrycznej. Jest ona rozszerzeniem znanego twierdzenia o średniej arytmetycznej i geometrycznej.

### 1.3.1 Twierdzenie I (o średniej arytmetycznej i geometrycznej)

Dla dowolnych nieujemnych $U_1$, $U_2$ zachodzi nierówność:

$$\frac{1}{2}U_1 + \frac{1}{2}U_2 \geq U_1^{\frac{1}{2}} \cdot U_2^{\frac{1}{2}} \tag{1.3}$$

Równość w (1.3) zachodzi wtedy i tylko wtedy, gdy $U_1 = U_2$.

**Dowód**
Jest to wniosek z nierówności

$$(U_1 - U_2)^2 \geq 0$$

ponieważ

$$U_1^2 + U_2^2 - 2 \cdot U_1 U_2 \geq 0\,,$$
$$U_1^2 + U_2^2 + 2 \cdot U_1 U_2 \geq 4 \cdot U_1 U_2\,,$$
$$(U_1 + U_2)^2 \geq 4 \cdot U_1 U_2\,,$$

$$(U_1 + U_2) \geq 2 \cdot U_1^{\frac{1}{2}} U_2^{\frac{1}{2}},$$
$$\frac{1}{2}U_1 + \frac{1}{2}U_2 \geq U_1^{\frac{1}{2}} \cdot U_2^{\frac{1}{2}}.$$

□

Uogólnienie tego związku stanowi twierdzenie II.

### 1.3.2 Twierdzenie II (Nierówność geometryczna)

[4]

Dla dowolnego wektora $x$ składającego się z $N$ współrzędnych $x_i$ i dla wektora d o $N$ nieujemnych współrzędnych zachodzi nierówność:

$$\sum_{i=1}^{N} x_i \delta_i \leq \left(\sum_{i=1}^{N} \delta_i\right) \ln \sum_{i=1}^{N} e^{x_i} + \sum_{i=1}^{N} \delta_i \ln \delta_i - \left(\sum_{i=1}^{N} \delta_i\right) \ln \left(\sum_{i=1}^{N} \delta_i\right).$$

Zależność powyższą można zastąpić równością wtedy i tylko wtedy gdy istnieje nieujemna liczba $B$ taka, że dla każdego $i$ ($i \in \{1, 2, \ldots, N\}$) zachodzi równość $\delta_i = B \cdot e^{x_i}$
Aby powyższe twierdzenie było formalnie poprawne należy jeszcze przyjąć konwencję, że dla a=0 wyrażenie a ln a ma wartość 0.

**Dowód**
Punktem wyjściowym dla powyższego twierdzenia jest wypukłość funkcji wykładniczej. Fakt że $e^x$ jest wypukła w całej swojej dziedzinie jest dobrze udokumentowany w literaturze, co więcej da się bardzo prosto udowodnić ($f(x) = e^x, f''(x) = e^x, f''(x)$ jest większe od zera w całej swojej dziedzinie, więc $f(x)$ jest wypukłe w całej swojej dziedzinie). Z wypukłości funkcji $e^x$ i z tego że jest to funkcja stała względem różniczkowania, wynika, że dla dwóch dowolnych punktów $z$ i $x$ współczynnik kierunkowy linii prostej przechodzącej przez te punkty jest nie mniejszy niż wartość funkcji w tym punkcie więc:

$$e^z \leq \frac{e^z - e^x}{z - x}$$

czyli inaczej zapisując

$$e^z + (x - z)e^z \leq e^x$$

[4] p. [30] str. 110-114

Zauważmy, że z uwagi na ścisłą wypukłość $e^x$ nierówność ta staje się równością wtedy i tylko wtedy, gdy $x = z$. Niech $y = e^z$. Wtedy nierówność przyjmuje formę

$$y + (x - \ln y)y \leq e^x$$

i zmienia się w równość wtedy i tylko wtedy gdy $x = \ln y$
Jeśli więc weźmiemy dwa wektory $x = (x_1, x_2, \ldots, x_N)$ o współrzędnych rzeczywistych oraz $y = (y_1, y_2, \ldots, y_N)$ o współrzędnych dodatnich, to dla każdego $i \in \{1, 2, \ldots, N\}$ zachodzi nierówność

$$y_i + (x_i - \ln y_i)y_i \leq e^{x_i},$$

a po dodaniu stronami tych nierówności otrzymujemy:

$$\sum_{i=1}^{N} y_i + \sum_{i=1}^{N}(x_i - \ln y_i)y_i \leq \sum_{i=1}^{N} e^{x_i},$$

przy czym równość zachodzi wtedy i tylko wtedy gdy $x_i = \ln y_i$ dla każdego $i \in \{1, 2, \ldots, N\}$.
Weźmy dowolną liczbę dodatnią $\theta$ Przyjmując że wektor $\delta$ powstaje po podzieleniu współrzędnych $y$ przez $\theta$ (tj. $\delta_i = \frac{y_i}{\theta}$ $i \in \{1, 2, \ldots, N\}$) mamy:

$$\sum_{i=1}^{N} \theta\delta_i + \sum_{i=1}^{N}(x_i - \ln \theta\delta_i)\theta\delta_i \leq \sum_{i=1}^{N} e^{x_i}, \tag{1.4}$$

przy czym równość zachodzi wtedy i tylko wtedy gdy

$$x_i = \ln\theta + \ln\delta_i \quad i \in \{1, 2, \ldots, N\}. \tag{1.5}$$

Zapisując (1.4) w inny sposób mamy:

$$\sum_{i=1}^{N} \theta\delta_i + \sum_{i=1}^{N}(x_i - \ln\theta - \ln\delta_i)\theta\delta_i \leq \sum_{i=1}^{N} e^{x_i},$$

$$\theta\sum_{i=1}^{N} \delta_i(1 + x_i - \ln\delta_i) - \theta\ln\theta\sum_{i=1}^{N}\delta_i \leq \sum_{i=1}^{N} e^{x_i}.$$

Przyjmując, że lewa strona nierówności jest funkcją $\theta$ mamy:

$$f(\theta) \leq \sum_{i=1}^{N} e^{x_i} \tag{1.6}$$

gdzie:

$$f(\theta) = \theta \sum_{i=1}^{N} \delta_i (1 + x_i - \ln \delta_i) - \theta \ln \theta \sum_{i=1}^{N} \delta_i \tag{1.7}$$

Różniczkując funkcję (1.7) względem $\theta$ otrzymujemy:

$$f'(\theta) = \sum_{i=1}^{N} \delta_i (1 + x_i - \ln \delta_i) - \ln \theta \sum_{i=1}^{N} \delta_i - \sum_{i=1}^{N} \delta_i \,, \tag{1.8}$$

a po powtórnym zróżniczkowaniu (1.8)

$$f''(\theta) = -\frac{\sum_{i=1}^{N} \delta_i}{\theta} \tag{1.9}$$

Ponieważ druga pochodna $f(\theta)$ jest w całej swej dziedzinie ujemna, więc $f(\theta)$ posiada maksimum lokalne w punkcie $\theta_0$, które można obliczyć z (1.8), czyli w takim, dla którego:

$$\ln \theta_0 = \frac{\sum_{i=1}^{N} \delta_i (x_i - \ln \delta_i)}{\sum_{i=1}^{N} \delta_i} \,. \tag{1.10}$$

Punkt $\theta_0$ również musi spełniać ograniczenia wynikające z (1.6). Po połączeniu więc (1.6),(1.7) i (1.9) i wykorzystaniu faktu, że funkcja logarytmiczna jest funkcją rosnącą (lub inaczej mówiąc po „zlogarytmizowaniu" obu stron nierówności) otrzymujemy:

$$ln \sum_{i=1}^{N} \delta_i + \ln \theta \leq \ln \sum_{i=1}^{N} e^{x_i} \,,$$

$$ln \sum_{i=1}^{N} \delta_i + \frac{\sum_{i=1}^{N} \delta_i (x_i - \ln \delta_i)}{\sum_{i=1}^{N} \delta_i} \leq \ln \sum_{i=1}^{N} e^{x_i} \,,$$

$$\sum_{i=1}^{N} \delta_i \cdot ln \sum_{i=1}^{N} \delta_i + \sum_{i=1}^{N} \delta_i (x_i - \ln \delta_i) \leq \sum_{i=1}^{N} \delta_i \cdot \ln \sum_{i=1}^{N} e^{x_i} \,,$$

$$\sum_{i=1}^{N} \delta_i \cdot ln \sum_{i=1}^{N} \delta_i + \sum_{i=1}^{N} \delta_i x_i - \sum_{i=1}^{N} \delta_i \ln \delta_i \leq \sum_{i=1}^{N} \delta_i \cdot \ln \sum_{i=1}^{N} e^{x_i} \,,$$

$$\sum_{i=1}^{N} \delta_i x_i \leq \sum_{i=1}^{N} \delta_i \cdot \ln \sum_{i=1}^{N} e^{x_i} + \sum_{i=1}^{N} \delta_i \ln \delta_i - \sum_{i=1}^{N} \delta_i \cdot ln \sum_{i=1}^{N} \delta_i \,.$$

Z uwagi na (1.5) i (1.10) równość zachodzi wtedy i tylko wtedy gdy:

$$x_i = \ln\theta_0 + ln\delta_i \quad i \in \{1, 2, \ldots, N\},$$

$$x_i = ln\delta_i + \frac{\sum_{i=1}^{N} \delta_i (x_i - \ln\delta_i)}{\sum_{i=1}^{N} \delta_i} \quad i \in \{1, 2, \ldots, N\}.$$

Wprowadzając nową zmienną

$$B = e^{-\frac{\sum_{i=1}^{N} \delta_i (x_i - \ln\delta_i)}{\sum_{i=1}^{N} \delta_i}}$$

mamy (warto odnotować, że $B$ jest liczbą dodatnią)

$$x_i = ln\delta_i - \ln B \quad i \in \{1, 2, \ldots, N\}. \tag{1.11}$$

Otrzymaliśmy więc wynik, który odpowiada tezie twierdzenia. Ponieważ wektor $\delta$ ma współrzędne nieujemne, pozostaje więc do przedyskutowania sytuacja, gdy jedna lub kilka współrzędnych $\delta_i$ są równe 0. Wtedy $\ln\delta_i$ występujący w (1.11) nie byłby określony. Rozpatrzmy więc trzy przypadki:
1. Wszystkie $\delta_i$ są dodatnie. W takim przypadku (1.11) odpowiada dokładnie równości z tezy twierdzenia. Możemy więc uznać dowód za zakończony.
2. Wszystkie $\delta_i$ są równe 0. Jest to przypadek trywialny, gdyż zgodnie z przyjętą konwencją lewa i prawa strona nierówności jest równa 0 niezależnie od wektora $x$, więc zawsze zamiast nierówności zachodzi odpowiednia równość.
3. Co najmniej jedna współrzędna $\delta_i$ jest równa 0 i co najmniej jedna współrzędna jest różna od 0. W takim przypadku dla współrzędnej równej 0 lewa strona nierówności jest równa zeru, a prawa różna. W związku z tym nierówność jest zawsze ostra.
Powyższe przypadki wyczerpują możliwe sytuacje, więc możemy uznać dowód twierdzenia za zakończony.

□

Zajmiemy się teraz konsekwencjami tego twierdzenia. Jak zostało powiedziane wcześniej, nierówność geometryczną można uznać za uogólnienie twierdzenia o średniej geometrycznej i arytmetycznej. Faktycznie, jeśli na twierdzenie II nałożymy warunek normalności wektora $\delta$

$$\sum_{i=1}^{N} \delta_i = 1,$$

to otrzymujemy:

$$\sum_{i=1}^{N} \delta_i x_i \leq \sum_{i=1}^{N} e^{x_i} + \sum_{i=1}^{N} \delta_i \ln \delta_i - \ln 1. \tag{1.12}$$

Wprowadźmy dodatkowy wektor $T$ określony równaniami

$$x_i = ln(\delta_i \cdot T_i)\,, i \in \{1, 2, \ldots, N\}\,.$$

Wtedy (1.12) po prostych przekształceniach przyjmuje formę

$$\sum_{i=1}^{N} (x_i - \ln \delta_i)\delta_i \leq \ln \sum_{i=1}^{N} e^{x_i}\,,$$

$$\sum_{i=1}^{N} \delta_i \ln T_i \leq \ln \sum_{i=1}^{N} \delta_i T_i\,,$$

$$\prod_{i=1}^{N} T_i^{\delta_i} \leq \sum_{i=1}^{N} \delta_i T_i\,. \tag{1.13}$$

Z uwagi na (1.11) równość zachodzi wtedy i tylko wtedy, gdy istnieje taka liczba dodatnia $B$, że

$$\delta_i = B\delta_i T_i\,, i \in \{1, 2, \ldots, N\}\,.$$

Lewa strona nierówności nosi nazwę ważonej średniej geometrycznej dodatnich liczb $T_1, T_2, T_3, \ldots, T_N$ o wagach $\delta_1, \delta_2, \delta_3, \ldots, \delta_N$, a prawa strona ważonej średniej arytmetycznej.
Powyżej udowodniona nierówność jest kluczowa dla metod programowania geometrycznego i wystarczy do udowodnienia poniższego twierdzenia, zwanego podstawowym lematem programowania geometrycznego.

### 1.3.3 Twierdzenie III (podstawowy lemat programowania geometrycznego)

[5]

Jeśli wektor t spełnia ograniczenia programu pierwotnego $\mathcal{A}$, a wektor $\delta$ spełnia ograniczenia programu dualnego $\mathcal{B}$, to:

[5] p. [30] str. 114-117

$$g_0(t) \geq v(\delta) \tag{1.14}$$

Co więcej, w (1.14) zamiast nierówności zachodzi równość wtedy i tylko wtedy, gdy:

$$\delta_i = \begin{cases} \frac{c_i \cdot t_1^{a_{i1}} \cdot t_2^{a_{i2}} \ldots t_m^{a_{im}}}{g_0(t)} & i \in J[0] \\ \Lambda_k(\delta) \cdot c_i \cdot t_1^{a_{i1}} \cdot t_2^{a_{i2}} \cdots t_m^{a_{im}} & i \in J[k] \quad k \in \{1, 2, \ldots, p\} \end{cases} \tag{1.15}$$

**Dowód**

Ponieważ każdy składnik $u_i(t)$ pozymianów $g_k(t)$ jest dodatni, więc logarytm z $u_i(t)$ jest zawsze określony. Można więc zdefiniować

$$x_i = \ln u_i(t) \quad i \in J[k] \quad k \in \{1, 2, \ldots, p\}$$

i zastosować nierówność geometryczną

$$\sum_{J[K]} \ln u_i(t)\delta_i \leq \left(\sum_{J[K]} \delta_i\right) \cdot \ln \sum_{J[K]} e^{\ln u_i(t)} + \sum_{J[K]} \delta_i \ln \delta_i - \left(\sum_{J[K]} \delta_i\right) \cdot \ln \left(\sum_{J[K]} \delta_i\right).$$

wtedy

$$\left(\sum_{J[K]} \delta_i\right) \cdot \ln \sum_{J[K]} u_i(t) + \sum_{J[K]} \delta_i \ln \delta_i - \left(\sum_{J[K]} \delta_i\right) \cdot \ln \left(\sum_{J[K]} \delta_i\right) \geq \sum_{J[K]} \ln u_i(t)\delta_i,$$

$$\left(\sum_{J[K]} \delta_i\right) \cdot \ln g_k(t) \geq \sum_{J[K]} (\ln u_i(t) - \ln \delta_i)\delta_i + \left(\sum_{J[K]} \delta_i\right) \cdot \ln \left(\sum_{J[K]} \delta_i\right), \tag{1.16}$$

czyli

$$g_k(t)^{\Lambda_k(\delta)} \geq \left[\prod_{J[K]} \left(\frac{u_i(t)}{\delta_i}\right)^{\delta_i}\right] \Lambda_k(\delta)^{\lambda_k(\delta)}, \tag{1.17}$$

gdzie:

$$\Lambda_k(\delta) = \sum_{J[K]} \delta_i \,, k \in \{0, 1, 2, \ldots, p\}\,.$$

W szczególności dla $k = 0$, pamiętając o warunku normalności programu dualnego ($\Lambda_0(\delta) = 1$) otrzymujemy:

$$g_0(t) \geq \prod_{J[0]} \left(\frac{u_i(t)}{\delta_i}\right)^{\delta_i} \tag{1.18}$$

Z drugiej strony przy uwzględnieniu ograniczeń programu dualnego dla $k \in \{0, 1, 2, \ldots, p\}$ zachodzi nierówność podwójna

$$1 \geq g_k(t)^{\Lambda_k(\delta)} \geq \left[\prod_{J[K]} \left(\frac{u_i(t)}{\delta_i}\right)^{\delta_i}\right] \Lambda_k(\delta)^{\lambda_k(\delta)}. \tag{1.19}$$

Wymnożenie stronami (1.18) i odpowiednio p nierówności (1.19) daje:

$$g_0(t) \geq \left[\prod_{i=1}^{n} \left(\frac{u_i(t)}{\delta_i}\right)^{\delta_i}\right] \prod_{i=1}^{p} \Lambda_k(\delta)^{\lambda_k(\delta)},$$

$$g_0(t) \geq \left[\prod_{i=1}^{n} \left(\frac{c_i \cdot t_1^{a_{i1}} \cdot t_2^{a_{i2}} \cdots t_m^{a_{im}}}{\delta_i}\right)^{\delta_i}\right] \prod_{i=1}^{p} \Lambda_k(\delta)^{\lambda_k(\delta)},$$

$$g_0(t) \geq \left[\prod_{i=1}^{n} \left(\frac{c_i}{\delta_i}\right)^{\delta_i}\right] \left[\prod_{i=1}^{p} \lambda_k(\delta)^{\lambda_k(\delta)}\right] \cdot t_1^{\sum_{i=1}^{n} a_{i1}\delta_i} \cdot t_2^{\sum_{i=1}^{n} a_{i2}\delta_i} \cdots t_m^{\sum_{i=1}^{n} a_{im}\delta_i}.$$

Po uwzględnieniu warunków ortogonalności programu dualnego otrzymujemy

$$g_0(t) \geq \left[\prod_{i=1}^{n} \left(\frac{c_i}{\delta_i}\right)^{\delta_i}\right] \left[\prod_{i=1}^{p} \lambda_k(\delta)^{\lambda_k(\delta)}\right],$$

co kończy dowód pierwszej części twierdzenia.
Druga część twierdzenia jest bezpośrednią konsekwencją spostrzeżenia, że aby $g_0(t) = v(\delta)$, to zarówno nierówność (1.18) jak i wszystkie nierówności (1.19) muszą zmienić się w równości.
Korzystając z tego faktu oraz z drugiej części tezy twierdzenia II, można łatwo obliczyć współczynniki $\delta_i$. Obliczmy współczynniki dla $i \in J[0]$

$$\delta_i = B_0 u_i(t) \quad i \in J[0],$$

(z nierówności geometrycznej)

$$\sum_{i \in J[0]} \delta_i = \sum_{i \in J[0]} B_0 u_i(t),$$

dalej z warunku normalności

$$1 = B_0 g_0(t).$$

Ostatecznie więc:

$$\delta_i = \frac{c_i \cdot t_1^{a_{i1}} \cdot t_2^{a_{i2}} \cdots t_m^{a_{im}}}{g_0(t)} \quad i \in J[0]\,.$$

Analogicznie obliczamy $\delta_i$ dla $i \notin J[0]$.
Można więc uznać twierdzenie za udowodnione, że jeśli $g_0(t) = v(\delta)$, to $\delta_i$ przyjmują wartości takie, jak w tezie twierdzenia. Dowód wynikania w drugą stronę jako prostą konsekwencję nierówności geometrycznej, w tym miejscu pomijamy.

□

Przed przystąpieniem do udowodnienia dualności programowania geometrycznego wprowadźmy jeszcze kilka twierdzeń pomocniczych, (punkty 1.3.4 - 1.5) które będą nam potrzebne przy tym dowodzie.

### 1.3.4 Twierdzenie IV (o warunku wypukłości funkcji)

[6]

Funkcja $f$ mająca ciągłe drugie pochodne cząstkowe na wypukłym $S \subset E_N$ jest wypukła, jeżeli funkcja

$$\varphi(s) = f([1-s] \cdot x + s \cdot y) \quad , 0 \leq s \leq 1$$

spełnia nierówność

$$\frac{\partial^2 \varphi}{\partial s^2} \geq 0, \quad 0 \leq s \leq 1\,.$$

dla dowolnych x,y. Co więcej jeśli nierówność jest ostra (dla $x \neq y$), to funkcja $f()$ jest ściśle wypukła.

**Dowód**

$$\varphi(0) - \varphi(s) = \frac{\partial \varphi}{\partial s} \mid_s (-s) + \frac{1}{2} \frac{\partial^2 \varphi}{\partial s^2} \mid_{s'} \cdot s^2\,,$$

$$\varphi(1) - \varphi(s) = \frac{\partial \varphi}{\partial s} \mid_s (1-s) + \frac{1}{2} \frac{\partial^2 \varphi}{\partial s^2} \mid_{s''} \cdot (1-s)^2\,,$$

gdzie $s' \in [0, s]$, $s'' \in [s, 1]$, więc

[6] p. [31] str. 51-53

$$[1-s]\cdot[\varphi(0)-\varphi(s)]+s\cdot[\varphi[(1)-\varphi(s)] = \frac{1}{2}\frac{\partial^2}{\partial s^2}\,|_{s'}\,(1-s)s^2+\frac{1}{2}\frac{\partial^2}{\partial s^2}\,|_{s''}\,(1-s)^2 s\,,$$

czyli kładąc

$$\delta_1 = 1 - s\,,$$

$$\delta_2 = s\,,$$

mamy

$$0 \leq \delta_1, \delta_2 \leq 1\,,$$

a stąd

$$\delta_1 \cdot f(x) + \delta_2 \cdot f(y) - f(\delta_1 x + \delta_2 y) = \frac{1}{2}\frac{\partial^2}{\partial s^2}\,|_{s'}\,\delta_1\delta_2^2 + \frac{1}{2}\frac{\partial^2}{\partial s^2}\,|_{s''}\,\delta_1^2\delta_2\,,$$

ale prawa strona jest zawsze $\geq 0$ gdy

$$\frac{\partial^2}{\partial s^2} \geq 0\,,$$

więc

$$f(\delta_1 x + \delta_2 y) \leq \delta_1 \cdot f(x) + \delta_2 \cdot f(y)\,.$$

Ponadto, jeśli

$$\frac{\partial^2}{\partial s^2} > 0\,,$$

to

$$f(\delta_1 x + \delta_2 y) < \delta_1 \cdot f(x) + \delta_2 \cdot f(y)\,.$$

□

### 1.3.5 Nierówność spełniana przez funkcje wypukłe

[7]

Wszystkie funkcje wypukłe spełniają następującą nierówność

$$f(x) + \nabla f(x) \cdot [y - x] \leq f(y)\,.$$

[7] p. [31] str. 51-53

**Dowód**

$$f[(1-s)\cdot x + s\cdot y] \leq (1-s)\cdot f(x) + s\cdot f(y), \quad 0 < s \leq 1,$$
$$s\cdot f(x) + f([1-s]\cdot x + s\cdot y) - f(x) \leq s\cdot f(y), \quad 0 < s \leq 1,$$
$$f(x) + \frac{f([1-s]\cdot x + s\cdot y) - f(x)}{s} \leq f(y), \quad 0 < s \leq 1.$$

Po zróżniczkowaniu

$$f(x) + \lim_{s\to 0} \frac{f(x + s\cdot[y-x]) - f(x)}{s} \leq f(y), \quad 0 < s \leq 1,$$

$$f(x) + \frac{d\, f(x + s\cdot[y-x]) - f(x)}{d\, s} \mid_{s=0} \leq f(y), \quad 0 < s \leq 1,$$
$$f(x) + \nabla f(x)\cdot[y-x] \leq f(y).$$

□

## 1.3.6 Wniosek

[8]

Funkcja, która ma ciągłe drugie pochodne cząstkowe na wypukłym zbiorze $S$, jest ciągła na $S$, jeśli forma kwadratowa

$$Q(q;z) = \sum_{i,j=1}^{N} \frac{\partial^2 f}{\partial z_j \partial z_i} \mid_z q_i q_j$$

jest zawsze nieujemna. Co więcej, f jest ściśle wypukła, jeśli forma kwadratowa jest zawsze dodatnia.

**Dowód**
Niech $x, y \in S$

$$\varphi(s) = f([1-s]\cdot x + s\cdot y),$$
$$\frac{\partial \varphi}{\partial s} = \sum_{i=1}^{n} \frac{\partial f}{\partial z_i} \mid_z (y_i - x_i),$$

[8] p. [64] str. 284, [31] str. 51-53

$$\frac{\partial^2 \varphi}{\partial s^2} = \sum_{i,j=1}^{n} \frac{\partial^2 f}{\partial z_i z_j} \mid_z (y_i - x_i)(y_j - x_j)\,,$$

gdzie

$$z = [1 - s] \cdot x + s \cdot y\,,$$

$$\frac{\partial^2 \varphi}{\partial s^2} = Q(y - x; z)\,,$$

więc

$$\frac{\partial^2 \varphi}{\partial s^2} \geq 0\,,$$

jeśli $Q(y - x; z)$ jest zawsze nieujemne. Co więcej,

$$\frac{\partial^2 \varphi}{\partial s^2} > 0\,,$$

jeśli $Q(y - x; z)$ jest zawsze dodatnie i $x \neq y$, co kończy dowód.

□

### 1.3.7 Dowód wklęsłości logarytmu funkcji dualnej

[9]

Funkcja

$$log[v(\delta)] = \sum_{i=1}^{n} (\log c_i - \log \delta_i) + \sum_{k=1}^{p} \lambda_k(\delta) \cdot \lambda_k(\delta)$$

jest wklęsła na całej swojej dziedzinie.

**Dowód**

$$log[v(\delta)] = \sum_{i=1}^{n} (\log c_i - \log \delta_i) + \sum_{k=1}^{p} \lambda_k(\delta) \cdot \lambda_k(\delta)$$

Oznaczmy drugie pochodne cząstkowe:

[9] p. [60] str. 3-4

$$H_{ij} = \frac{\partial^2 \log v}{\partial \delta_i \, \partial \delta_j} \quad i,j \in \{1,2,\ldots,n\}\,.$$

Łatwo zauważyć, że Hesjan funkcji $v(\delta)$ da się zapisać jako

$$H = \begin{bmatrix} H_0 & 0 & 0 & \ldots & 0 \\ 0 & H_1 & 0 & \ldots & 0 \\ 0 & 0 & H_2 & \ldots & 0 \\ \ldots & & & & \\ 0 & 0 & 0 & \ldots & H_p \end{bmatrix},$$

gdzie

$$H_0 = \begin{bmatrix} -\delta_1^{-1} & 0 & & 0 \\ 0 & -\delta_2^{-1} & \ldots & 0 \\ \ldots & & & \\ 0 & 0 & \ldots & -\delta_{n_0}^{-1} \end{bmatrix},$$

a

$$H_k = \begin{bmatrix} \lambda_k^{-1} - \delta_{m_k}^{-1} & \lambda_k^{-1} & & \lambda_k^{-1} \\ \lambda_k^{-1} & \lambda_k^{-1} - \delta_{m_{k+1}}^{-1} & & \lambda_k^{-1} \\ \ldots & & & \\ \lambda_k^{-1} & \lambda_k^{-1} & \ldots & \lambda_k^{-1} - \delta_{n_k}^{-1} \end{bmatrix}, \quad k = 1,2,\ldots,p.$$

Forma kwadratowa powstała z $H$ ma postać

$$\sum_i \sum_j y_i \circ H_{ij} \circ y_j\,,$$

czyli

$$Q(y) = -\sum_{J[0]} \frac{y_i^2}{\delta_1} + \sum_{k=1}^{p} \left( \frac{\lambda_k^2(y)}{\lambda_k(\delta)} - \sum_{J[k]} \frac{y_i^2}{\delta_i} \right).$$

Korzystając z nierówności Cauchy'ego mamy

$$(\sum_{i \in J[K]} y_i)^2 = \sum_{i \in J[k]} (\frac{y_i}{\delta_i^{1/2}} \cdot \delta_i^{1/2})^2 \leq \sum_{i \in J[k]} \frac{y_i^2}{\delta_i} \cdot \sum_{i \in J[k]} \delta_i\,.$$

Oznacza to, że

$$\sum_{i \in J[k]} \frac{y_i^2}{\delta_i} - \frac{\lambda_k^2(y)}{\lambda_k(\delta)} \geq 0\,,$$

a ponieważ

$$\delta_i > 0\,,$$

więc cała forma kwadratowa jest niedodatnia a co za tym idzie (wniosek z 1.3.6) funkcja jest wklęsła.

□

## 1.4 Twierdzenie V (lemat Farkasa)

[10]

Niech $a_{ij}$ będzie macierzą {r,m}, a $b_j$ $j \in \{1, 2, \ldots, m\}$ liczbami rzeczywistymi. Wtedy, jeśli każde rozwiązanie układu nierówności

$$f_i(x) = \sum_{j=1}^{m} a_{ij} \cdot x_j \geq 0 \quad , i \in \{1, 2, \ldots, r\}\,,$$

jest równocześnie rozwiązaniem nierówności:

$$g(x) = \sum_{j=1}^{m} b_j \cdot x_j \geq 0\,,$$

to istnieją takie liczby

$$y_i \geq 0 \quad , i \in \{1, 2, \ldots, r\}\,,$$

że

$$\sum_{i=1}^{r} y_i \cdot a_{ij} = b_j \quad j \in \{1, 2, \ldots, m\}\,.$$

[10] p. [30] str. 244-247

**Dowód**

Przeprowadzimy dowód indukcyjny względem liczby zmiennych. Jeśli $m = 1$, to dowód jest oczywisty

$$a_{11} \cdot x_1 \geq 0 \qquad b_1 \cdot x_1 \geq 0$$
$$a_{12} \cdot x_1 \geq 0$$

$$\vdots$$
$$a_{1j} \cdot x_1 \geq 0$$
$$\Downarrow$$
$$a_{1j} = 0 \Rightarrow b_1 = 0$$
$$a_{1j} < 0 \Rightarrow b_1 > 0$$
$$a_{1j} > 0 \Rightarrow b_1 < 0$$

Załóżmy, że lemat jest prawdziwy dla $\bigwedge m < M$ zmiennych i spróbujmy udowodnić jego prawdziwość dla $M$ zmiennych.
Wprowadźmy nową zmienną

$$t = b_1 \cdot x_1 + b_2 \cdot x_2 + \ldots + b_M \cdot x_M \,.$$

Przynajmniej jeden ze współczynników $b_j$ powinien być różny od zera, gdyż inaczej trywialnym rozwiązaniem układu są $y_i \equiv 0$.
Z założeń lematu wiemy, że

$$f_i(x_1, \ldots, x_M) = F_i(x_1, \ldots, x_{M-1}) + c_i \cdot t \geq 0 \,. \tag{1.20}$$

Rozpatrzmy następujące przypadki:

$a) c_i = 0 \,,$

$b) c_i > 0 \,,$

$c) c_i < 0 \,.$

Rozbijmy nierówności (1.20) według tych przypadków. Po podzieleniu przez $\mid c_i \mid$ dla $c_i \neq 0$ otrzymamy odpowiednio

$$\begin{array}{ll} a) & F_i^0(x) \geq 0 \quad i \in \{1,2,\ldots,r^0\} \\ b) & F_i^+(x) + t \geq 0 \quad i \in \{1,2,\ldots,r^+\} \\ c) & F_i^-(x) - t \geq 0 \quad i \in \{1,2,\ldots,r^-\} \end{array}$$

Dodając nierówności b) i c) mamy

$$F_i^+(x) + F_j^-(x) \geq 0, \quad i \in \{1,2,\ldots,r^+\} \wedge \{1,2,\ldots,r^-\}$$

$r^+$ jest większe od 0 bo $f_m = 0 + 1 \cdot t \geq 0$ Załóżmy, ze $r^-$ też jest $> 0$
Oznaczmy $t_0 = -min(F_1^+(x), F_2^+(x), \ldots)$
Zauważmy, że dla $t > t_0$ spełnione są wszystkie nierównośći b).
Z definicji $t_0$ wynika, że dla pewnego $k$ jest

$$t_0 = -F_k^+(x)\,.$$

Zatem

$$F_i^-(x) - t_0 = F_i^-(x) + F_k^+(x)\,.$$

Jak łatwo sprawdzić, punkt $(x, t_0)$ spełnia a),b) i c) więc

$$t_0 \geq 0\,,$$

czyli

$$F_k \leq 0\,.$$

Powyższy fakt implikuje prawdziwość założeń twierdzenia dla $m = M - 1$, więc są takie stałe dodatnie $y_i, y_{ij}, z_i$, że:

$$\sum y_i \cdot F_i^0(x) + \sum\sum y_{ij} \cdot F^-(x) + F_j^x \equiv \sum z_i \cdot F_i^+(x)\,,$$

czyli oznaczając $\gamma = \sum z_i$ otrzymujemy

$$\sum \frac{yi}{\gamma} \cdot F_i^0(x) + \sum\sum \frac{i_{ij}}{\gamma}[F_i^-(x) - t + F_i^+(x) + t] + \sum \frac{z_i}{\gamma}[F_i^+(x) + t] = t\,,$$

co jest równoznaczne z tezą twierdzenia.

□

## 1.5 Twierdzenie VI (Kuhna - Tuckera)

[11]

Niech $G_k(z)$ $k = 0, 1, \ldots, p.$ będą funkcjami wypukłymi na $E_m$ mającymi ciągłe pochodne cząstkowe.
Następujący dwa problemy są równoważne: **problem I**
znaleźć punkt z minimalizujący funkcję $g_0(z)$ na zbiorze określonym nierównościami

$$G_k(z) \leq 0 \quad k \in \{1, \ldots, p\}$$

**problem II** znaleźć punkt $(z' , \mu')$ taki, że

$$a)\mu'_k \geq 0 \quad k \in \{1, .., p\},$$

$$b)\mu'_k < G_k(z') = 0 \quad k \in \{1, .., p\},$$

$$c)L(z', \mu) \leq (L(z', \mu') \leq (L(z, \mu'),$$

gdzie:
$L(z, \mu)$ jest funkcją Lagrange'a, zdefiniowana następująco:

$$L(z, \mu) = G_0(z) + \sum_{k=1}^{p} G_k(z),$$

z jest dowolnym wektorem z $E_m$, a $\mu$ takim wektorem, że $\mu_k \geq 0$.
Dodatkowo zakładamy, że istnieje co najmniej jeden punkt $z^*$, który spełnia ostro ograniczenia problemu pierwszego tj.

$$\bigwedge_{k \in \{1..p\}} Gk(z^*) < 0 .^{12}$$

**Dowód:**
Najpierw udowodnimy wynikanie w drugą stronę, tj. fakt, że rozwiązanie problemu **II** jest równocześnie rozwiązaniem problemu **I**.

[11] p. [43] str. 519-523, [30] str. 60-77

[12] To twierdzenie nosi również w niektórych opracowaniach nazwę warunków Karusha-Kuhna-Tuckera ponieważ zostało udowodnione niezależnie przez W. Karusha w 1939 r. oraz H.W. Kuhna i A.W. Tuckera w 1951 r. p. [43] str. 519-523

Załóżmy, że $(z', \mu')$ spełnia nierówności a)-c).
Ponieważ

$$L(z', \mu) \leq L(z', \mu'),$$

więc

$$L(z', \mu) - L(z', \mu') \leq 0,$$

czyli

$$G_0(z) + \sum_{k=1}^{p} \mu_k G_k(z) - [G_0(z) + \sum_{k=1}^{p} \mu' G_k(z) \leq 0,$$

$$\sum_{k=1}^{p} (\mu_k - \mu k') G_k(z) \leq 0$$

dla każdego wektora $\mu$ takiego, że wszystkie jego współrzędne są nie mniejsze od zera. W szczególności więc dla $\mu$ zdefiniowanego następująco (uwzględniając warunek a)

$$\mu_q = \begin{cases} \mu'_q & q \neq k, \\ \mu'_q + 1 & q = k. \end{cases}$$

a co za tym idzie

$$G_k(z') \leq 0 \quad k \in \{1, \ldots, p\}.$$

Czyli z spełnia ograniczenia problemu **I**.
Druga część nierówności c) mówi, że

$$\bigwedge_{z \in E_m} L(z, \mu') - L(z', \mu') \geq 0.$$

Inaczej zapisując:

$$G_0(z) + G_0(z') + \sum_{k=1}^{p} \mu'_k [Gk(z) - Gk(z')] \geq 0.$$

Wykorzystując założenie b) otrzymujemy

$$G_0(z) - G_0(z') + \sum_{k=1}^{p} \mu'_k Gk(z) \geq 0.$$

Jeśli uwzględnimy założenia problemu **I** oraz nierówność a), zauważamy, że

$$\sum_{k=1}^{p} \mu_k' Gk(z) \leq 0\,,$$

więc

$$G_0(z) - G_0(z') \leq 0\,,$$

czyli

$$G_0(z) \leq G_0(z')$$

dla każdego z spełniającego

$$G_k(z) < 0 \quad k \in \{1, \ldots, p\}$$

co kończy dowód wynikania w jedną stronę, a tym samym pierwszej części twierdzenia
Załóżmy teraz, że z' jest rozwiązaniem problemu **I**. Zdefiniujmy zbiór $K'$ jako zbiór indeksów tych funkcji, dla których $z'$ jest ich miejscem zerowym

$$K' = \{k \geq 1 : G_k(z') = 0\}\,.$$

Z twierdzenia IV wynika, że

$$G_k(z') + \nabla G_k(z')[z - z^*] < G_k(z), \quad k \in \{1, \ldots, p\}\,,$$

czyli

$$\nabla G_k(z')[z - z'] \leq G_k(z) \quad k \in K'\,,$$

więc

$$\nabla G_k(z')[z - z'] \leq 0 \quad k \in K'\,.$$

Dodatkowo wiemy, że istnieje takie $z^*$, że nierówność powyższa jest ostra:

$$\nabla G_k(z')[z^* - z'] < 0, \quad k \in K'\,.$$

Niech z będzie wybranym punktem, a s liczbą z przedziału (0,1). Łącząc dwie poprzednie nierówności otrzymujemy:

$$(1 - s)\nabla G_k(z')[z - z'] + s\nabla G_k(z')[z^* - z'] < 0 \quad k \in K'$$

lub

$$\nabla G_k(z')[(1-s)z + sz^* - z'] < 0, \quad k \in K',$$

a po przyjęciu oznaczenia

$$x = (1-s)z + sz^*, \quad 0, s \leq 1$$

zapisujemy nierówność w postaci:

$$\nabla G_k(z')[x - z'] < 0 \quad k \in K' \tag{1.21}$$

Niech $t$ będzie odpowiednio małą liczbą dodatnią. Wykorzystując ponownie twierdzenie IV dla liczb

$$z' + t[x - z'] \; i \; z'$$

mamy

$$G_k(z'+t[x-z'])+\nabla G_k(z'+t[x-z'])(z'-x) \leq G_k(z') \quad k \in \{1,\ldots,p\}, \tag{1.22}$$

czyli przy K' zdefiniowanym jw.

$$G_k(z' + t[x - z'])+ \leq \nabla G_k(z' + t[x - z'])(z' - x), \quad k \in K.' \tag{1.23}$$

Ponieważ

$$G_k(z') < 0 \;\; k \notin K'$$

i ponieważ $G_k(z)$ i ich pierwsze pochodne są ciągłe, więc przy odpowiednio małym, dodatnim t

$$G_k(z' + t[x - z']) < 0 \quad dla \; k \notin K'. \tag{1.24}$$

Z drugiej strony z (1.21) i (1.22) mamy

$$G_k(z' + t[x - z']) < 0 \quad dla \; k \in K',$$

więc

$$G_k(z' + t[x - z']) < 0 \quad dla \; k \in 1, 2, \ldots, p$$

Ponieważ z minimalizuje $G_0$, więc przy odpowiednio małym t

$$G_0(z' + t(x - z')) - G_0(z') \geq 0.$$

z (1.22) i (1.24) wynika, że

$$\nabla(z' + t(x - z')) \cdot (x - z') \geq 0 .$$

przy $t \longrightarrow 0$

$$\nabla G_0(z') \cdot (x - z') \geq 0 ,$$

czyli z definicji x

$$\nabla G_0(z') \cdot [(1 - s)z + sz^* - z'], \quad 0 < s \leq 1 ,$$

co redukuje się przy $s \longrightarrow 0$ do

$$\nabla G_0(z') \cdot (z - z') \geq 0 .$$

Pokazaliśmy więc, że

$$\nabla G_0(z') \cdot (z - z') \geq 0 ,$$

gdzie z spełnia nierówność

$$\nabla G_k(z') \cdot (z - z') \geq 0 \quad k \in K .$$

Stosując Lemat Farkasa (tw. V) wiemy, że są liczby

$$\mu'_k \geq 0, \quad k \in K'$$

takie, że

$$G_0(z') = - \sum_{k \in K'} \mu' \nabla G_k(z') .$$

Definiując

$$\mu'_k = 0 \quad k \notin K'$$

mamy

$$G_0(z') + \sum_{k=1}^{p} \mu' \nabla G_k(z') .$$

Sposób zdefiniowania $\mu_k$ pozwala stwierdzić, że warunki a) i b) problemu **II** są spełnione.
Pozostaje do udowodnienia warunek c). Z definicji funkcji $L$ wynika że

$$L(z', \mu) = G_0(z') + \sum_{k=1}^{p} G_k(z') ,$$

a więc zgodnie z b) otrzymamy

$$L(z', \mu') = G_0(z'),$$

czyli

$$L(z', \mu) \leq L(z', \mu')$$

dla $\mu_k > 0$ $k \in \{1, 2, \ldots, p\}$, gdyż $G_k(z') < 0$.
Druga część nierówności c) jest konsekwencją faktu, że funkcja $L(z, \mu')$ jako dodatnia kombinacja liniowa funkcji wypukłych jest wypukła, a z jako punkt stacjonarny funkcji wypukłej jest równocześnie punktem, w którym funkcja ma minimum, czyli

$$L(z', \mu') \leq L(z, \mu').$$

W ten sposób dowód wynikania w drugą stronę, a co z tego idzie całego twierdzenia Kuhna-Tuckera został zakończony.

□

Przegląd lematów i twierdzeń potrzebnych do sformułowania teorii programowania geometrycznego możemy uznać za zakończony. Następne twierdzenie o dualności programowania geometerycznego, opublikowane po raz pierwszy w [29] str. 1307-1349 i powtórnie w [30] str. 80. które dowodzi że program pierwotny (1.1) i program dualny (1.2) wyznaczają to samo optymalne rozwiązanie (o ile takie istnieje), jest podstawą programowania geometrycznego

## 1.6 Dualność programowania geometrycznego

[13]

Załóżmy, że program $\mathcal{A}$ ma co najmniej jedno ostre rozwiązanie, a funkcja pierwotna $g_0(t)$ ma minimum, które spełnia ograniczenia pierwotne. Wtedy
1. program dualny $\mathcal{B}$ odpowiadający $\mathcal{A}$ ma co najmniej jeden punkt spełniający ograniczenia, a $v(\delta)$ posiada punkt maksymalny spełniający ograniczenia;
2. $v(\delta)_{max} = g_0(t)_{min}$
Jeśli $t'$ jest punktem minimalizującym $g_0$, to istnieją nieujemne mnożniki Lagrange'a takie, że funkcja

[13] p. [30] str. 164-194

$$L(t', \mu) \leq g_0(t') = L(t', \mu') \leq L(t, \mu')$$

dla dowolnych $t_j > 0$ i $\mu \geq 0$.

3. Wektor $\delta'$ maksymalizujący $\mathcal{B}$ ma współrzędne

$$\delta_i' = \begin{cases} c_i \cdot t_1^{a_{i1}} \cdot \ldots \cdot t_m^{a_{im}} & i \in J[0]\,, \\ \frac{\mu_k' \cdot c_i t_1^{a_{i1}} \cdot \ldots \cdot t_m^{a_{im}}}{g_0(t)} & i \in J[k]\ k \in \{1, \ldots, p\} \end{cases}$$

4. Jeśli $\delta'$ jest punktem maksymalizującym $\mathcal{B}$, to każdy $t'$ minimalizujący $\mathcal{A}$ spełnia układ równań

$$c_i \cdot t_1^{a_{i1}} \cdot \ldots \cdot t_m^{a_{im}} = \begin{cases} \delta_i' \cdot v(\delta') & i \in J[0]\,, \\ \frac{\delta_i'}{\lambda_k(\delta')} & i \in J[k]\ k \in \{1, \ldots, p\}. \end{cases}$$

**Dowód**

Zdefiniujmy

$$z_i = \ln t_i \quad i \in \{1, 2, \ldots, n\}\,.$$

Pierwotny program transformowany zdefiniujmy następująco: znaleźć minimalną wartość funkcji $g_0(z)$ przy ograniczeniach

$$g_k(z) \leq 1 \quad k \in \{1, 2, \ldots, p\}\,,$$

gdzie

$$g_k(z) = \sum_{i \in J[k]} c_i e^{\sum_{i=1}^{m} a_{ij} z_j} \quad k \in \{1, 2, \ldots, p\}\,,$$

$$J[k] = \{m_k, m_{k+1}, m_{k+2}, \ldots, n_k\} \quad k \in \{0, 1, \ldots, q\}$$

$$m_0 = 1, m_1 = n_0 + 1, m_2 = n_1 + 1, \ldots, m_p = n_{p-1} + 1, n_q = n\,,$$

gdzie $a_{ij} \in R$, $c_i > 0$

Łatwo udowodnić, że program $\mathcal{A}$ jest programem wypukłym, tj. takim, którego celem jest minimalizacja funkcji wypukłej na zbiorze wypukłym (p. 1.1)

Jeśli oznaczymy

$$G_k(z) \equiv \begin{cases} g_k(z) & k = 0\,, \\ g_k(z) - 1 & k \in \{1, 2, \ldots, p\}\,, \end{cases}$$

to możemy dla funkcji $G_k$ wykorzystać twierdzenie VI (tw. Kuhna-Tuckera). Oznacza to istnienie takiego wektora $\mu(\lambda)$ o $p$ współrzędnych, że funkcja

$$L(z,\mu) = g_0(z) + \sum_{k=1}^{p} \mu_k \cdot [g_k(z) - 1]$$

ma własność

$$L(z',\mu) \leq g_0(z') = L(z',\mu') \leq L(z,\mu')$$

dla dowolnych $z$ i $\mu$. Więc $z'$ jest punktem minimalizującym $L(z,\mu')$, a stąd

$$\frac{\partial L}{\partial z_q}(z',\mu') = 0 \quad q \in \{1,2,\ldots,m\},$$

$$\sum_{J[0]} a_{iq} c_i \cdot e^{\sum_{j=1}^{m} a_{ij} z'_j} + \sum_{k=1}^{p} \mu'_k \cdot \left( \sum_{J[k]} a_{iq} c_i \cdot e^{\sum_{j=1}^{m} a_{ij} z'_j} \right) = 0,$$

$$q \in \{1,2,\ldots,m\}.$$

Dzieląc te równania przez $g_0(z')$ i odpowiednio definiując wektor $\delta'$, widzimy że

$$\delta'_i = \begin{cases} c_i \cdot t_1^{a_{i1}} \cdot \ldots \cdot t_m^{a_{im}} & i \in J[0], \\ \frac{\mu'_k \cdot c_i t_1^{a_{i1}} \cdot \ldots \cdot t_m^{a_{im}}}{g_0(t)} & i \in J[k]\ k \in \{1,\ldots,p\} \end{cases}$$

spełniają warunki ortogonalności $\mathcal{B}$. Z powyższego oznaczenia wynika również, że $\delta'_i$ spełniają warunek normalności $\mathcal{B}$. Co więcej wszystkie $\delta'_i$ są dodatnie. Punkt $\delta'$ spełnia więc ograniczenia programu dualnego $\mathcal{B}$.
Przekształcając powyższą definicję mamy

$$\lambda_k(\delta') = \frac{\mu'_k g_k(z')}{g_0(z')} \quad k \in \{1,2,\ldots,p\}.$$

Z twierdzenia VI (Kuhna-Tuckera) wiemy, że

$$\mu'_k[g_k(z') - 1] = 0,$$

czyli

$$\lambda_k(\delta') = \frac{\mu'_k}{g_0(z')} \quad k \in \{1,2,\ldots,p\},$$

możemy więc ponownie zapisać $\delta_i'$ jako

$$\delta_i' = \begin{cases} c_i \cdot t_1^{a_{i1}} \cdot \ldots \cdot t_m^{a_{im}}, & i \in J[0], \\ \lambda_k)\delta')c_i t_1^{a_{i1}} \cdot \ldots \cdot t_m^{a_{im}}, & i \in J[k]\ k \in \{1, \ldots, p\}, \end{cases}'$$

co odpowiada wartościom $\delta_i'$ w twierdzeniu III (główny lemat programowania geometrycznego). Wnioskujemy więc, że

$$g_0(t') = v(\delta').$$

Pozostała do udowodnienia czwarta część twierdzenia. Załóżmy, że $t'$ minimalizuje program pierwotny $\mathcal{A}$ a $\delta'$ maksymalizuje program dualny $\mathcal{B}$. Zgodnie z tym co przed chwilą udowodniliśmy

$$g_0(t') = v(\delta').$$

więc

$$\delta_i' = \begin{cases} c_i \cdot t_1^{a_{i1}} \cdot \ldots \cdot t_m^{a_{im}}, & i \in J[0], \\ \lambda_k(\delta')c_i t_1^{a_{i1}} \cdot \ldots \cdot t_m^{a_{im}}, & i \in J[k]\ k \in \{1, \ldots, p\}, \end{cases}'$$

co po prostych przekształceniach daje nam czwarty podpunkt twierdzenia, a co za tym idzie kończy dowód dualności programowania geometrycznego.

□

## 1.7 Zastosowanie metod programowania geometrycznego w wybranych problemach

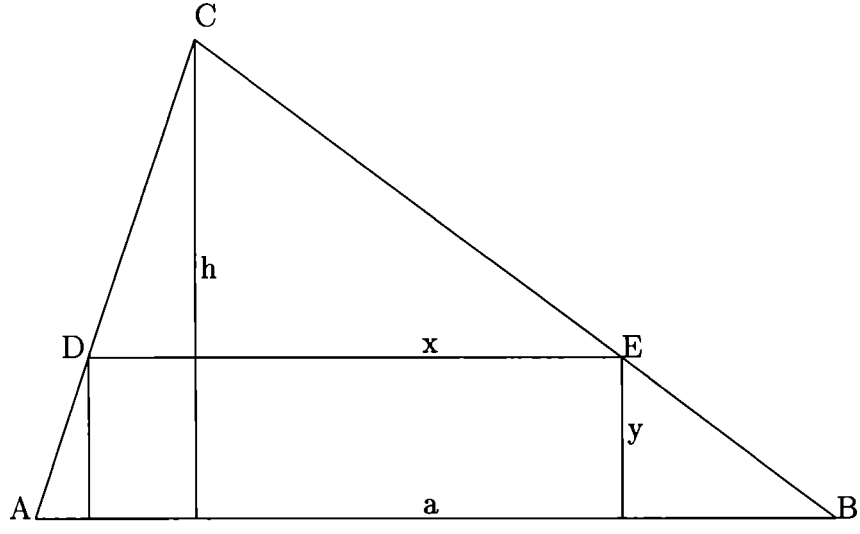
Metody programowania geometrycznego mogą być zastosowane do rozwiązywania wielu prostych zadań na znajdowanie minimum funkcji, bez wykorzystywania pochodnych. Szczególnie, zgodnie z nazwą, nadają się one do zadań związanych z polami i obwodami figur, gdyż funkcje, których ekstrema liczymy w tego typu problemach są najczęściej sumami pozymianów lub dadzą się w prosty sposób na takie sumy przekształcić. Zastanówmy się nad typowym takim zadaniem:

**Przykład 2**

Pośród wszystkich prostokątów wpisanych w trójkąt znaleźć taki, którego pole jest największe.

Pole prostokąta zgodnie z rysunkiem wyraża się wzorem:

$$P(x, y) = xy$$




równocześnie, ponieważ $\triangle ABC \sim \triangle DEC$, zachodzi równość:

$$\frac{a}{x} = \frac{h}{h-y},$$

skąd

$$ah - ay = xh\,,$$

$$xh + ay = ah\,,$$

oraz

$$\frac{x}{a} + \frac{y}{h} = 1\,.$$

Wystarczy jeszcze zauważyć, że pole prostokąta jest maksymalne wtedy gdy jego odwrotność osiąga minimum, aby sformułować problem programowania geometrycznego:

**zminimalizować funkcję:**

$$g(x,y) = \frac{1}{xy} = x^{-1}y^{-1}$$

przy założeniu, że

$$\frac{x}{a} + \frac{y}{h} = 1.$$

Jest to problem o łącznej liczbie wyrazów równej 3 i dwóch zmiennych, więc o stopniu złożoności DOD równym 3-(2+1)=0. Jego rozwiązanie sprowadza się do rozwiązania układu trzech równań liniowych.

Zgodnie z zasadami programowania geometrycznego wiemy, że wartością minimalną funkcji $g$ jest

$$\left(\frac{1}{\delta_1}\right)^{\delta}_{1}\left(\frac{1}{a\delta_2}\right)^{\delta}_{2}\left(\frac{1}{h\delta_3}\right)^{\delta}_{3}(\delta_2+\delta_3)^{\delta_2+\delta_3}$$

gdzie $\delta_i$ obliczamy z warunku normalności i warunków ortogonalnosci

$$\begin{array}{llll} \delta_1 & & & = 1\,, \\ -\delta_1 & +\delta_2 & & = 0\,, \\ -\delta_1 & & +\delta_3 & = 0\,. \end{array}$$

Rozwiązaniem tego układu, jak łatwo sprawdzić, są liczby $(1,1,1)$.
więc wartością minimalną $g(x,y)$ jest

$$\left(\frac{1}{1}\right)^{1}\left(\frac{1}{a}\right)^{1}\left(\frac{1}{h}\right)^{1}(1+1)^{1+1} = \frac{4}{ah}$$

Punkt, w, którym wartość minimalna została osiągnięta, wyznaczamy z równań

$$\frac{1}{xy} = \frac{4}{ah},$$

$$\frac{x}{a} = \frac{\delta_2}{\delta_2 + \delta_3} = \frac{1}{1+1} = \frac{1}{2},$$

$$\frac{y}{h} = \frac{\delta_3}{\delta_2 + \delta_3} = \frac{1}{1+1} = \frac{1}{2},$$

więc:

$$x = \frac{a}{2}, \; y = \frac{h}{2},$$

co ostatecznie daje nam odpowiedź, że maksymalne pole prostokąta wpisanego w trójkąt wynosi

$$\frac{ah}{4} = \frac{P_\triangle}{2}.$$

Boki maksymalnego prostokąta, wynoszą odpowiednio połowę wybranego boku trójkąta i połowę wysokości spuszczonej na ten bok.
Rozwiązanie to jest zgodne z rozwiązaniem, które otrzymalibyśmy rozwiązując to zadanie metodami rachunku różniczkowego.

Metody analogiczne do programowania geometrycznego stosowali już starożytni Grecy. Przykładem może być pytanie, który spośród prostokątów o tym samym polu ma najmniejszy obwód.

**Przykład 3**

$$O_b = 2a + 2b,$$

$$O_b = 4\left(\frac{a}{2} + \frac{b}{2}\right),$$

$$O_b = 4\left[\left(\frac{a}{2} + \frac{b}{2}\right)^2\right]^{\frac{1}{2}},$$

$$O_b = 4\left[\frac{ab}{2} + \frac{a^2}{4} + \frac{b^2}{4}\right]^{\frac{1}{2}},$$

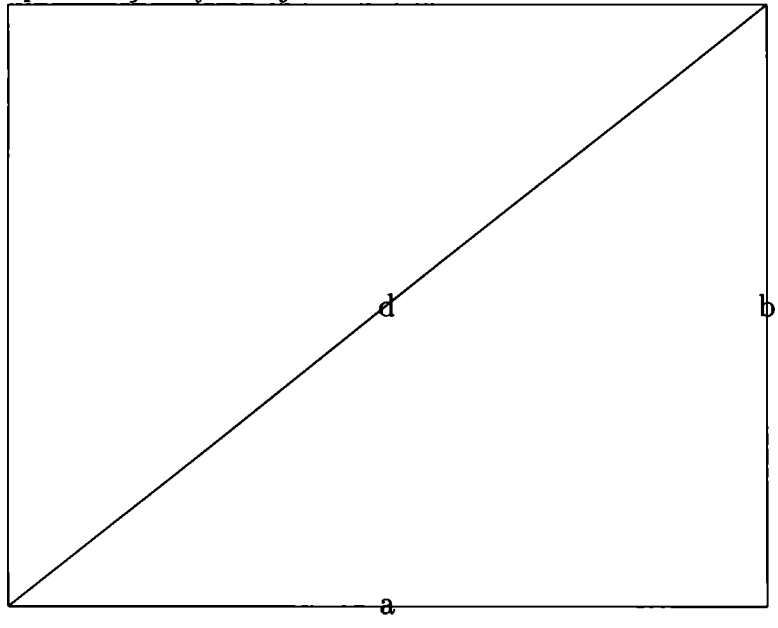
$$O_b = 4\left[ab - \frac{ab}{2} + \frac{a^2}{4} + \frac{b^2}{4}\right]^{\frac{1}{2}},$$

$$O_b = 4\left[ab + \left(\frac{a}{2} - \frac{b}{2}\right)^2\right]^{\frac{1}{2}}$$

Po przekształceniach widać, że obwód jest minimalny gdy wyrażenie pod kwadratem jest równe 0, więc gdy $a = b$, czyli figurą o najmniejszym obwodzie jest kwadrat.

**Przykład 4**

Wyznaczyć długości boków prostokąta o stałym obwodzie 2p tak, aby przekątna tego prostokąta była najkrótsza.




Pozornie problem minimalizacji długości przekątnej prostokąta nie jest zadaniem programowania geometrycznego, jednak po uwzględnieniu szeregu

równoważności możemy je sformułować (przy oznaczeniach jak na rysunku): jako zadanie minimalizacji $d$ przy ograniczeniach

$$2a + 2b = 2p\,,\; a^2 + b^2 = d^2\;,$$

$$\Updownarrow$$

albo minimalizacji $d^2$ przy ograniczeniach

$$2a + 2b = 2p\,,\; a^2 + b^2 = d^2\;,$$

$$\Updownarrow$$

albo minimalizacji $p^2 - 2 \cdot a \cdot b$ przy ograniczeniach

$$2a + 2b = 2p\;,$$

$$\Updownarrow$$

albo wreszcie minimalizacji

$$g(a,b) = \frac{1}{2 \cdot a \cdot b}$$

przy założeniu, że

$$\frac{a}{p} + \frac{b}{p} \leq 1,.$$

Problem posiada zadanie dualne maksymalizacji funkcji:

$$V(\delta) = \left(\frac{1}{2 \cdot \delta_1}\right)^{\delta_1} \cdot \left(\frac{1}{\delta_2}\right)^{\delta_2} \cdot \left(\frac{1}{\delta_3}\right)^{\delta_3} \cdot (\delta_2 + \delta_3)^{\delta_2 + \delta_3},$$

gdzie $\delta_i \quad \delta_i > 0 \quad i \in \{1,2,3\}$ spełniają równania:

$$\begin{array}{llll} \delta_1 & & & = 1, \\ -\delta_1 & +\delta_2 & & = 0, \\ -\delta_1 & & +\delta_3 & = 0, \end{array}$$

czyli:

$$\begin{aligned} \delta_1 &= 1, \\ \delta_2 &= 1, \\ \delta_3 &= 1, \end{aligned}$$

a co za tym idzie

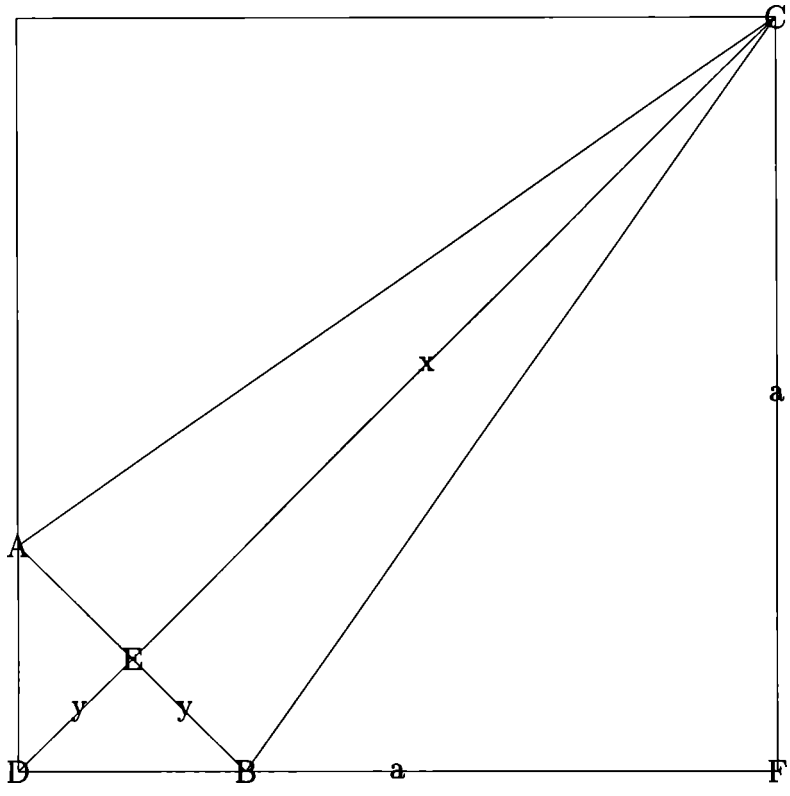
$$\begin{aligned} \frac{a}{p} &= \frac{1}{1+1}, \\ \frac{b}{p} &= \frac{1}{1+1}, \end{aligned}$$

$$\begin{aligned} a &= \frac{p}{2}, \\ b &= \frac{p}{2}. \end{aligned}$$

Szukanym prostokątem jest więc kwadrat.

**Przykład 5**

W kwadrat o boku długości a wpisujemy trójkąty równoramienne w ten sposób, że wierzchołek (nie należący do podstawy trójkąta) każdego z tych trójkątów jest równocześnie wierzchołkiem kwadratu. Który z tych trójkątów ma największe pole ?




Przy oznaczeniach jak na rysunku ($\overline{AE} = \overline{EB} = \overline{ED} = y$, $\overline{EC} = x$, $\overline{DF} = \overline{FC} = a$) widzimy, że pole trójkąta $ABC$ wynosi

$$\nabla_{ABC} = \frac{1}{2} \cdot 2 \cdot y \cdot x ,$$

gdzie $x, y \geq 0$ spełniają r"wnanie

$$x + y = a\sqrt{2} ,$$

Zadanie to jest więc problemem programowania geometrycznego polegającym na minimalizacji funkcji

$$g(x, y) = \frac{1}{x \cdot y} \quad , x, y \geq 0 ,$$

przy ograniczeniu

$$\frac{x\sqrt{2}}{2\cdot a}+\frac{y\sqrt{2}}{2\cdot a}\leq 1 \quad , x,y\geq 0\ .$$

Jego problem dualny to maksymalizacja funkcji:

$$V(\delta)=\left(\frac{1}{\delta_1}\right)^{\delta_1}\cdot\left(\frac{\sqrt{2}}{2\cdot a\cdot\delta_2}\right)^{\delta_2}\cdot\left(\frac{\sqrt{2}}{2\cdot a\cdot\delta_3}\right)^{\delta_3}\cdot(\delta_2+\delta_3)^{\delta_2+\delta_3},$$

gdzie $\delta_i \quad \delta_i>0 \quad i\in\{1,2,3\}$ spełniają związki:

$$\begin{array}{llll} \delta_1 & & & =1,\\ -\delta_1 & +\delta_2 & & =0,\\ -\delta_1 & & +\delta_3 & =0, \end{array}$$

stąd

$$\begin{array}{ll} \delta_1 & =1,\\ \delta_2 & =1,\\ \delta_3 & =1. \end{array}$$

Maksymalną wartością $V(\delta)$ i równocześnie minimalną wartością $g(x,y)$ jest

$$\frac{2}{a^2}$$

i jest ona osiągana przy:

$$\begin{array}{ll} \frac{x\sqrt{2}}{2\cdot a} & =\frac{1}{1+1},\\ \frac{y\sqrt{2}}{2\cdot a} & =\frac{1}{1+1}, \end{array}$$

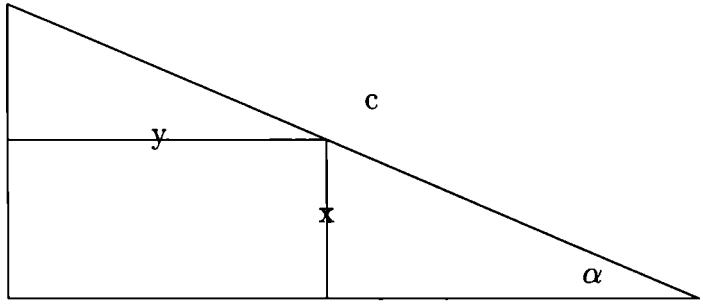
czyli przy

$$\begin{array}{ll} x & =\frac{a\sqrt{2}}{2},\\ y & =\frac{a\sqrt{2}}{2}. \end{array}$$

Szukanym trójkąt jest więc trójkątem prostokątnym, którego podstawę stanowi przekątna kwadratu.

**Przykład 6**

W trójkąt prostokątny o przeciwprostokątnej długości c i danym kącie ostrym $\alpha$ wpisujemy prostokąty w ten sposób, że dwa boki każdego z tych prostokątów zawierają się w obu przyprostokątnych. Który z tych prostokątów ma największe pole ?




Przy oznaczeniach takich jak na rysunku pole prostokąta wynosi $x \cdot y$ natomiast jego boki spełniają zależność:

$$\frac{x}{\sin\alpha} + \frac{y}{\cos\alpha} = c,$$

Formułując więc ten przykład jako problem programowania geometrycznego otrzymujemy zadanie:
**zminimalizować:**

$$g(x,y) = \frac{1}{x \cdot y} \quad , x, y \geq 0 \ ,$$

gdzie

$$\frac{x}{c \cdot \sin\alpha} + \frac{y}{c \cdot \cos\alpha} \leq 1 \quad , x, y \geq 0 \ ,$$

i formę dualną tego zadania
**zmaksymalizować:**

$$V(\delta) = \left(\frac{1}{\delta_1}\right)^{\delta_1} \cdot \left(\frac{1}{c \cdot \sin\alpha \cdot \delta_2}\right)^{\delta_2} \cdot \left(\frac{1}{c \cdot \cos\alpha \cdot \delta_3}\right)^{\delta_3} \cdot (\delta_2 + \delta_3)^{\delta_2+\delta_3},$$

gdzie $\delta_i \quad \delta_i > 0 \quad i \in \{1,2,3\}$ spełniają warunki

$$\begin{aligned} \delta_1 \qquad\quad &= 1, \\ -\delta_1 \quad +\delta_2 \quad &= 0, \\ -\delta_1 \quad +\delta_3 \quad &= 0, \end{aligned}$$

czyli:

$$\begin{aligned} \delta_1 &= 1, \\ \delta_2 &= 1, \\ \delta_3 &= 1. \end{aligned}$$

Maksymalną wartością $V(\delta)$ i równocześnie minimalną wartością $g(x,y)$ jest wielkość

$$\frac{4}{c^2 \cdot \sin\alpha \cdot \cos\alpha},$$

więc maksymalne pole prostokąta wynosi:

$$\frac{c^2 \cdot \sin 2\alpha}{8}$$

zaś boki prostokąta o maksymalnym polu spełniają związki

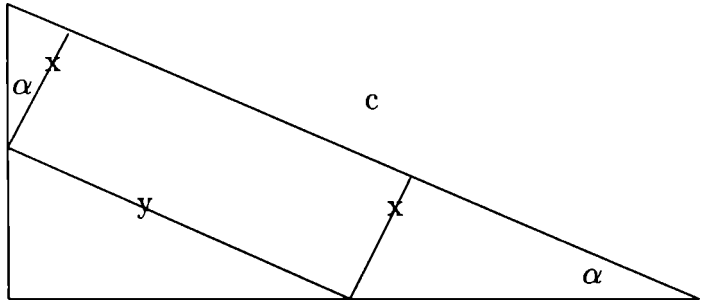
$$\begin{aligned} \tfrac{x}{c\cdot\sin\alpha} &= \tfrac{1}{1+1}, \\ \tfrac{y}{c\cdot\cos\alpha} &= \tfrac{1}{1+1}, \end{aligned}$$

czyli wynoszą

$$\begin{aligned} x &= \tfrac{c\cdot\sin\alpha}{2}, \\ y &= \tfrac{c\cdot\cos\alpha}{2}. \end{aligned}$$

**Przykład 7**

W trójkąt prostokątny o danym kącie ostrym $\alpha$ i przeciwprostokątnej długości c wpisujemy prostokąty w ten sposób, że jeden bok każdego z tych prostokątów zawiera się w przeciwprostokątnej trójkąta. Który z tych prostokątów ma największe pole?




Przy oznaczeniach takich jak na rysunku pole prostokąta wynosi $x \cdot y$ natomiast jego boki spełniają zależność:

$$\frac{x}{\tan\alpha} + y + x \cdot \tan\alpha = c,$$

Formułując więc to zadanie jako problem programowania geometrycznego otrzymujemy zadanie minimalizacji funkcji

$$g(x,y) = \frac{1}{x \cdot y} \quad , x, y \geq 0 \ ,$$

przy ograniczeniu

$$\frac{2 \cdot x}{c \cdot \sin 2\alpha} + fracyc \leq 1 \quad , x, y \geq 0 \ ,$$

i formę dualną tego zadania polegającą na maksymalizacji funkcji

$$V(\delta) = \left(\frac{1}{\delta_1}\right)^{\delta_1} \cdot \left(\frac{2}{c \cdot \delta_2 \cdot \sin 2\alpha}\right)^{\delta_2} \cdot \left(\frac{1}{c \cdot \delta_3}\right)^{\delta_3} \cdot (\delta_2 + \delta_3)^{\delta_2 + \delta_3},$$

gdzie $\delta_i \quad \delta_i > 0 \quad i \in \{1,2,3\}$ spełniają związki

$$\begin{aligned} \delta_1 \qquad\quad &= 1, \\ -\delta_1 \quad +\delta_2 \quad &= 0, \\ -\delta_1 \quad +\delta_3 \quad &= 0, \end{aligned}$$

czyli:

$$\begin{aligned} \delta_1 \quad &= 1, \\ \delta_2 \quad &= 1, \\ \delta_3 \quad &= 1. \end{aligned}$$

Maksymalną wartością $V(\delta)$ i równocześnie minimalną wartością $g(x,y)$ jest

$$\frac{16}{c^2 \cdot \sin 2\alpha}$$

więc maksymalne pole prostokąta wynosi:

$$\frac{c^2 \cdot \sin 2\alpha}{16},$$

Boki prostokąta o maksymalnym polu wynoszą:

$$\begin{aligned} \frac{2 \cdot x}{c \cdot \sin 2\alpha} \quad &= \frac{1}{1+1}, \\ \frac{y}{c} \quad &= \frac{1}{1+1}, \end{aligned}$$

czyli są równe

$$\begin{aligned} x \quad &= \frac{c \cdot \sin 2\alpha}{4}, \\ y \quad &= \frac{c}{2}. \end{aligned}$$

## Przykład 8

Daną liczbę dodatnią a rozłożyć na dwa takie składniki, aby:
a) suma kwadratów tych składników była najmniejsza,
b) suma sześcianów tych składników była najmniejsza.
Zadanie a) polega na minimalizacji wyrażenia

$$x^2 + y^2$$

przy założeniu:

$$x, y > 0, x + y = a,$$

czyli minimalizacji wyrażenia

$$a^2 - x \cdot y$$

przy założeniu:

$$x, y > 0, x + y = a,$$

albo minimalizacji wyrażenia

$$\frac{1}{x \cdot y},$$

przy ograniczeniu

$$\frac{x}{a} + \frac{y}{a} \leq 1 \ , \ x, y > 0,$$

Zadanie dualne polega tu na maksymalizacji wyrażenia

$$V(\delta) = \left(\frac{1}{\delta_1}\right)^{\delta_1} \cdot \left(\frac{1}{a \cdot \delta_2}\right)^{\delta_2} \cdot \left(\frac{1}{a \cdot \delta_3}\right)^{\delta_3} \cdot (\delta_2 + \delta_3)^{\delta_2 + \delta_3},$$

gdzie $\delta_i \quad \delta_i > 0 \quad i \in \{1, 2, 3\}$ spełniają związki

$$\begin{array}{llll} \delta_1 & & & = 1, \\ -\delta_1 & +\delta_2 & & = 0, \\ -\delta_1 & & +\delta_3 & = 0, \end{array}$$

czyli:

$$\begin{array}{ll} \delta_1 & = 1, \\ \delta_2 & = 1, \\ \delta_3 & = 1. \end{array}$$

Ostatecznie:

$$\begin{aligned} \frac{x}{a} &= \frac{1}{1+1}, \\ \frac{y}{a} &= \frac{1}{1+1}, \end{aligned}$$

$$\begin{aligned} x &= \frac{a}{2}, \\ y &= \frac{a}{2}. \end{aligned}$$

W problemie b) minimalizujemy wyrażenie

$$x^3 + y^3$$

przy założeniu:

$$x, y > 0, x + y = a,$$

co jest równoważne minimalizacji wyrażenia

$$a^3 - 3 \cdot a \cdot x \cdot y$$

przy założeniu

$$x, y > 0, x + y = a,$$

czyli minimalizacji wyrażenia

$$\frac{1}{x \cdot y}$$

przy ograniczeniu

$$\frac{x}{a} + \frac{y}{a} \leq 1 \ , \ x, y > 0$$

Zgodnie więc z wynikami z podpunktu a)

$$\begin{aligned} x &= \frac{a}{2}, \\ y &= \frac{a}{2}. \end{aligned}$$

**Przykład 9**

Wyznaczyć punkt o współrzędnych dodatnich, należący do hiperboli $y = 1/x$, tak aby jego odległość od punktu A(0,0) była najmniejsza.

Zadanie sprowadza się do minimalizacji funkcji:

$$f(x,y) = x^2 + y^2$$

przy ograniczeniu

$$\frac{1}{x \cdot y} \leq 1 \ , x, y > 0,$$

czyli do maksymalizacji

$$V(\delta) = \left(\frac{1}{\delta_1}\right)^{\delta_1} \cdot \left(\frac{1}{\delta_2}\right)^{\delta_2} \cdot \left(\frac{1}{\delta_3}\right)^{\delta_3} \cdot (\delta_2 + \delta_3)^{\delta_2+\delta_3},$$

gdzie $\delta_i \quad \delta_i > 0 \quad i \in \{1,2,3\}$ spełniają związki

$$\begin{array}{lll} \delta_1 + & \delta 2 & = 1, \\ 2 \cdot \delta_1 & -\delta_3 & = 0, \\ 2 \cdot \delta_2 & -\delta_3 & = 0, \end{array}$$

czyli:

$$\begin{array}{ll} \delta_1 & = \frac{1}{2}, \\ \delta_2 & = \frac{1}{2}, \\ \delta_3 & = 1. \end{array}$$

Najmniejsza odległość wynosi więc 2 a współrzędne szukanego punktu wyznaczone z równań

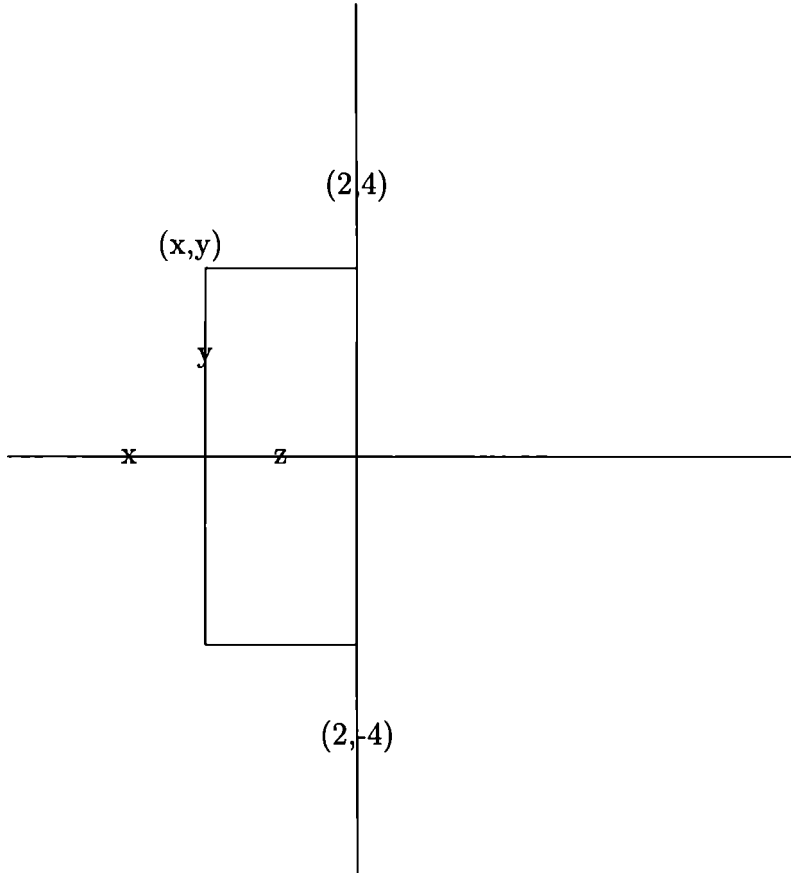
$$\begin{array}{ll} x^2 & = \frac{1}{2} \cdot 2, \\ y^2 & = \frac{1}{2} \cdot 2, \end{array}$$

wynoszą

$$\begin{array}{ll} x & = 1\,, \\ y & = 1\,. \end{array}$$

**Przykład 10**

Dana jest parabola $y^2 = 8 \cdot x$ i odcinek o końcach A(2,4), B(2,-4). Rozważmy rodzinę tych wszystkich prostokątów, których dwa wierzchołki należą do danego odcinka, zaś dwa pozostałe należą do danej paraboli. Który z tych prostokątów ma największe pole ?




Pole szukanego prostokąta wynosi zgodnie z oznaczeniami z rysunku $2 \cdot y \cdot z$ przy czym punkt $(x, y)$ należy do paraboli $y = 8 \cdot x^2$ oraz:

$$z + x = 2.$$

Formułując problem w języku programowania geometrycznego mamy zadanie minimalizacji funkcji

$$g(x,y,z) = \frac{1}{2 \cdot y \cdot z}$$

przy ograniczeniach

$$\frac{x}{2} + \frac{z}{2} \leq 1 \ ,$$

$$\frac{y^2}{8 \cdot x} \leq 1 \ , x, y, z > 0 \ .$$

Problem dualny odpowiadający temu zadaniu to maksymalizacja funkcji

$$V(\delta) = \left(\frac{1}{2 \cdot \delta_1}\right)^{\delta_1} \cdot \left(\frac{1}{2 \cdot \delta_2}\right)^{\delta_2} \cdot \left(\frac{1}{2 \cdot \delta_3}\right)^{\delta_3}$$

$$\cdot (\delta_2 + \delta_3)^{\delta_2 + \delta_3} \cdot \left(\frac{1}{8 \cdot \delta_4}\right)^{\delta_4} \cdot (\delta_4)^{\delta_4},$$

gdzie $\delta_i \quad \delta_i > 0 \quad i \in \{1,2,3,4\}$ spełniają związki:

$$\begin{array}{llll} \delta_1 & & & = 1, \\ -\delta_1 & +\delta_3 & & = 0, \\ \delta_2 & & -\delta_4 & = 0, \\ -\delta_1 & & +2\delta_4 & = 0, \end{array}$$

czyli:

$$\begin{array}{ll} \delta_1 & = 1, \\ \delta_2 & = \frac{1}{2}, \\ \delta_3 & = 1, \\ \delta_4 & = \frac{1}{2}. \end{array}$$

Współrzędne szukanego punktu wyznaczamy z równań

$$\begin{array}{ll} \frac{x}{2} & = \frac{\frac{1}{2}}{\frac{3}{2}} \\ \frac{y^2}{8 \cdot x} & = 1 \end{array}$$

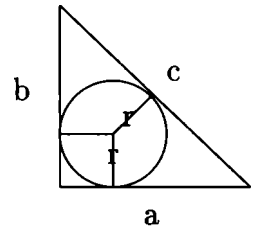
$$\begin{array}{ll} x & = \frac{2}{3}, \\ y & = \frac{4 \cdot \sqrt{3}}{3}. \end{array}$$

Szukany prostokąt ma współrzędne

$$\left(\frac{2}{3}, \frac{4\sqrt{3}}{3}\right), \left(2, \frac{4\sqrt{3}}{3}\right), \left(2, -\frac{4\sqrt{3}}{3}\right), \left(\frac{2}{3}, -\frac{4\sqrt{3}}{3}\right).$$

**Przykład 11**

Wyznacz długości boków trójkąta prostokątnego opisanego na okręgu o danym promieniu $r = 1\,cm$ tak, aby długość przeciwprostokątnej trójkąta była najmniejsza




Z właściwości okręgu wpisanego w trójkąt prostokątny wiemy, że:

$$a^2 + b^2 = c^2 \ ,$$

oraz

$$\nabla_{\overline{abc}} = \frac{1}{2} \cdot a \cdot b = \frac{1}{2} \cdot (a + b + c) \cdot r \ \ , a, b, c > 0 \ ,$$

Po prostych przekształceniach i uwzględnieniu równości $r = 1$ zadanie da się więc sformułować jako minimalizacja funkcji

$$f(a, b, c) = c$$

przy ograniczeniach:

$$\frac{a^2}{c^2} + \frac{b^2}{c^2} \leq 1,$$

$$\frac{1}{a}+\frac{1}{b}+\frac{c}{a\cdot b}\leq 1,$$
$$a,b,c>0\ ,$$

Problem dualny odpowiadający temu zadaniu to maksymalizacja funkcji

$$V(\delta)=\left(\frac{1}{\delta_1}\right)^{\delta_1}\cdot\left(\frac{1}{\delta_2}\right)^{\delta_2}\cdot\left(\frac{1}{\delta_3}\right)^{\delta_3}\cdot(\delta_2+\delta_3)^{\delta_2+\delta_3}$$
$$\cdot\left(\frac{1}{\delta_4}\right)^{\delta_4}\cdot\left(\frac{1}{\delta_5}\right)^{\delta_5}\cdot\left(\frac{1}{\delta_6}\right)^{\delta_6}\cdot(\delta_4+\delta_5+\delta_6)^{\delta_4+\delta_5+\delta_6},$$

gdzie $\delta_i$ $(\delta_i>0,\quad i\in\{1,2,3,4,5,6\})$ spełniają równania

$$\begin{array}{llll} \delta_1 & & & =1,\\ 2\cdot\delta_2 & -\delta_4 & -\delta_6 & =0,\\ 2\cdot\delta_3 & -\delta_5 & -\delta_6 & =0,\\ \delta_1 & -2\cdot\delta_2 & -2\cdot\delta_3\ +\delta_6 & =0, \end{array}$$

W programie pierwotnym występują trzy zmienne przy łącznej liczbie pozymianów równej 6. Stopień złożoności tego zadania wynosi więc $DOD = 6-3-1=2>0$. W celu obliczenia $\delta_i$ $i\in\{1,2,3,4,5,6\}$ musimy więc przyrównać do 0 pochodne cząstkowe z funkcji przeciwnej do logarytmu z $V(\delta)$ analogicznie jak w przykładzie ze strony [88]. Po wykonaniu odpowiednich obliczeń otrzymujemy:

$$\begin{array}{ll} \delta_1 & =1,\\ \delta_2 & =\frac{\sqrt{2}}{4},\\ \delta_3 & =\frac{\sqrt{2}}{4},\\ \delta_4 & =1-\frac{\sqrt{2}}{2},\\ \delta_5 & =1-\frac{\sqrt{2}}{2},\\ \delta_6 & =\sqrt{2}-1\,. \end{array}$$

Długości boków trójkąta obliczymy z równań

$$\begin{array}{ll} \frac{1}{a} & =\frac{1-\frac{\sqrt{2}}{2}}{1-\frac{\sqrt{2}}{2}+1-\frac{\sqrt{2}}{2}+\sqrt{2}-1},\\ \frac{1}{b} & =\frac{1-\frac{\sqrt{2}}{2}}{1-\frac{\sqrt{2}}{2}+1-\frac{\sqrt{2}}{2}+\sqrt{2}-1},\\ \frac{c}{a\cdot b} & =\frac{\sqrt{2}-1}{1-\frac{\sqrt{2}}{2}+1-\frac{\sqrt{2}}{2}+\sqrt{2}-1}, \end{array}$$

ostatecznie

$$\begin{aligned} a &= 2+\sqrt{2}, \\ b &= 2+\sqrt{2}, \\ c &= 2+2\cdot\sqrt{2}. \end{aligned}$$

**Przykład 12**

Wyznaczyć długości boków prostokąta wpisanego w elipsę $\frac{x^2}{16}+\frac{y^2}{9}=1$ , tak aby jego pole było nawiększe.

Przed sformułowaniem zadania programowania geometrycznego zauważmy, że boki prostokąta są równoległe do osi elipsy oraz, że oś rzędnych i odciętych są osiami symetrii prostokąta.

Program pierwotny można więc sformułować jako zadanie minimalizacji funkcji

$$\frac{1}{4\cdot x\cdot y} \ ,$$

przy ograniczeniu

$$\frac{x^2}{16}+\frac{y^2}{9}\leq 1 \ \ , x,y>0 \ ,$$

Formułując zaś program dualny otrzymujemy następujące zadanie: **zmaksymalizować wyrażenia:**

$$V(\delta)=\left(\frac{1}{4\cdot\delta_1}\right)^{\delta_1}\cdot\left(\frac{1}{16\cdot\delta_2}\right)^{\delta_2}\cdot\left(\frac{1}{9\cdot\delta_3}\right)^{\delta_3}\cdot(\delta_2+\delta_3)^{\delta_2+\delta_3},$$

gdzie $\delta_i$ $(\delta_i>0, \quad i\in\{1,2,3\})$ spełniają równania

$$\begin{aligned} \delta_1 \qquad\quad &= 1, \\ -\delta_1 \ +2\cdot\delta_2 \ &= 0, \\ -\delta_1 \ +2\cdot\delta_3 \ &= 0, \end{aligned}$$

czyli:

$$\begin{aligned}\delta_1 &= 1,\\ \delta_2 &= \tfrac{1}{2},\\ \delta_3 &= \tfrac{1}{2}.\end{aligned}$$

Maksymalną wartością $V(\delta)$ i równocześnie minimalną wartością $g(x,y)$ jest $\frac{1}{24}$

Stąd maksymalne pole prostokąta równa się $24\,j^2$
zaś boki prostokąta o maksymalnym polu wyznaczone z równań

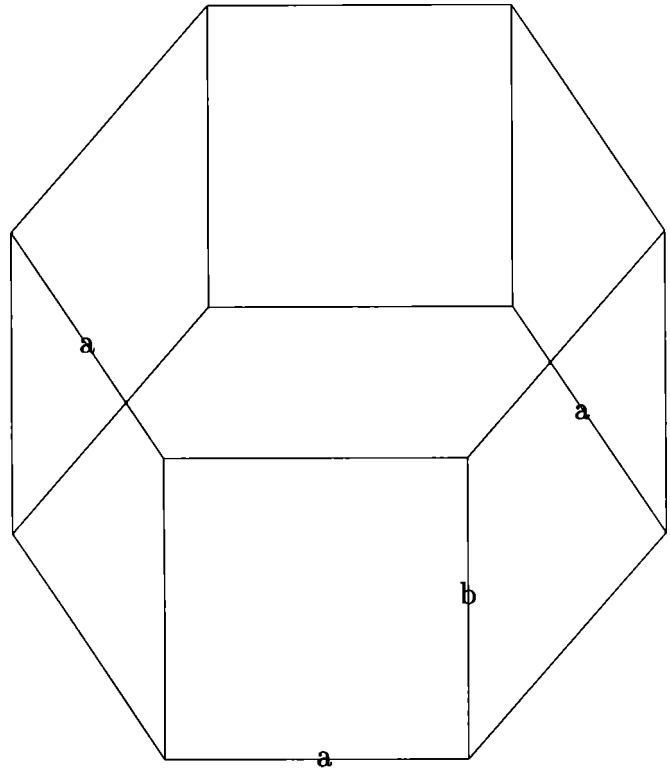
$$\begin{aligned}\tfrac{x^2}{16} &= \tfrac{\frac{1}{2}}{1},\\ \tfrac{y^2}{9} &= \tfrac{\frac{1}{2}}{1},\end{aligned}$$

wynoszą

$$\begin{aligned}2\cdot x &= 4\cdot\sqrt{2},\\ 2\cdot y &= 3\cdot\sqrt{2}.\end{aligned}$$

**Przykład 13**

Podstawą graniastosłupa prostego jest n-kąt foremny, zaś suma długości wszystkich krawędzi graniastosłupa jest równa p. Wyznaczyć długość krawędzi tak, aby objętość graniastosłupa była największa.




Objętość graniastosłupa przy oznaczeniach takich jak na rysunku wynosi:

$$V = Pp \cdot b = \frac{n}{2} \cdot \cos\frac{\pi}{n} \cdot a^2 \cdot b,$$

przy czym:

$$2 \cdot n \cdot a + n \cdot b = p \ , a, b > 0 \ .$$

Nasz problem sprowadza się do minimalizacji funkcji

$$f(a,b) = \frac{2}{n \cdot \cos\frac{\pi}{n} \cdot a^2 \cdot b} \ ,$$

gdzie $a, b > 0$ spełniają związek

$$\frac{2 \cdot n \cdot a}{p} + \frac{n \cdot b}{p} \leq 1 \ ,$$

lub do maksymalizacji funkcji

$$V(\delta) = \left(\frac{2}{n \cdot \cos\frac{\pi}{n} \cdot \delta_1}\right)^{\delta_1} \cdot \left(\frac{2 \cdot n}{p \cdot \delta_2}\right)^{\delta_2} \cdot \left(\frac{n}{p \cdot \delta_3}\right)^{\delta_3} \cdot (\delta_2 + \delta_3)^{\delta_2+\delta_3},$$

gdzie $\delta_i \quad \delta_i > 0 \quad i \in \{1, 2, 3\}$ spełniają równania

$$\begin{array}{lll} \delta_1 & & = 1, \\ -2 \cdot \delta_1 & +\delta_2 & = 0, \\ -\delta_1 & +\delta_3 & = 0, \end{array}$$

czyli:

$$\begin{array}{ll} \delta_1 & = 1, \\ \delta_2 & = 2, \\ \delta_3 & = 1. \end{array}$$

Krawędzie graniastosłupa o największej objętości spełniają więc: związki

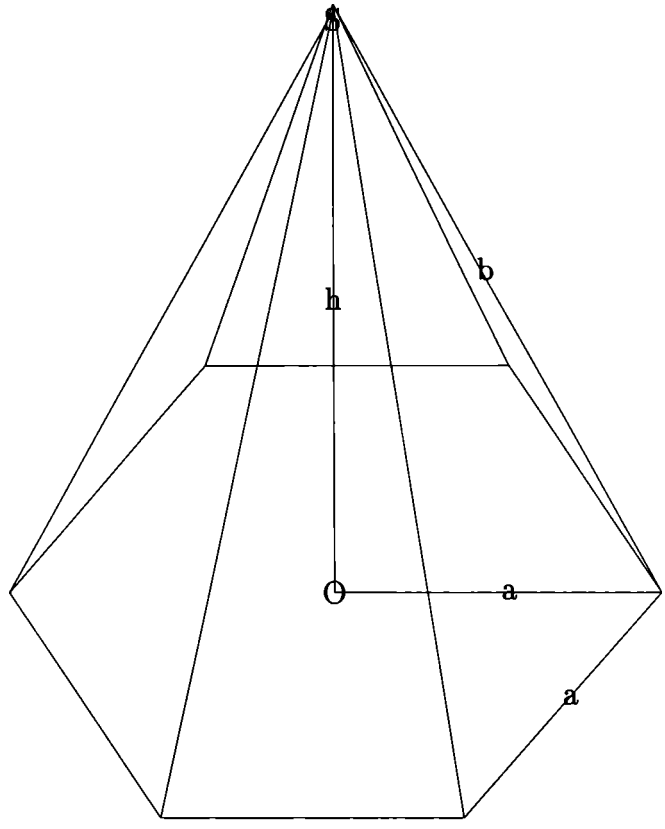
$$\begin{array}{ll} \frac{2 \cdot n \cdot a}{p} & = \frac{2}{2+1}, \\ \frac{n \cdot b}{p} & = \frac{1}{2+1}, \end{array}$$

czyli

$$\begin{array}{ll} a & = \frac{p}{3 \cdot n}, \\ b & = \frac{p}{3 \cdot n}. \end{array}$$

## Przykład 14

Podstawą ostrosłupa prawidłowego jest 6-kąt foremny, zaś suma wszystkich krawędzi ostrosłupa jest równa p. Wyznaczyć długości krawędzi, tak aby objętość ostrosłupa była największa.




Z treści zadania wynikają następujące zależności:

$$V_{Ostroslupa} = \frac{1}{3} \cdot \frac{6 \cdot \sqrt{3}}{4} \cdot a^2 \cdot h \ ,$$

$$h^2 + a^2 = b^2 \ ,$$

$$6 \cdot a + 6 \cdot b = p \ .$$

Maksymalizacja objętości jest więc równoważna minimalizacji $f(a, b, h)$ , $a, b, h > 0$ danej zależnością

$$f(a, b, h) = \frac{2 \cdot \sqrt{3}}{3} \cdot a^2 \cdot h$$

przy ograniczeniach

$$\frac{h^2}{b^2} + \frac{a^2}{b^2} \leq 1 \ ,$$

$$\frac{6 \cdot a}{p} + \frac{6 \cdot b}{p} \leq 1 \ ,$$

a to z kolei jest równoważne maksymalizacji $V(\delta)$

$$V(\delta) = \left(\frac{2 \cdot \sqrt{(3)}}{3 \cdot \delta_1}\right)^{\delta_1} \cdot \left(\frac{1}{\delta_2}\right)^{\delta_2} \cdot \left(\frac{1}{\delta_3}\right)^{\delta_3} \cdot (\delta_2 + \delta_3)^{\delta_2+\delta_3}$$

$$\cdot \left(\frac{6}{p \cdot \delta_4}\right)^{\delta_4} \cdot \left(\frac{6}{p \cdot \delta_5}\right)^{\delta_5} \cdot (\delta_4 + \delta_5)^{\delta_4+\delta_5}$$

gdzie $\delta_i \quad (\delta_i > 0, \quad i \in \{1,2,3,4,5\})$ spełniają związki

$$\begin{array}{llll} \delta_1 & & & = 1, \\ -2 \cdot \delta_1 & +2 \cdot \delta_3 & +\delta_4 & = 0, \\ -\delta_1 & 2 \cdot \delta_2 & & = 0, \\ -2 \cdot \delta_2 & -2 \cdot \delta_3 & +\delta_5 & = 0. \end{array}$$

W programie pierwotnym występują trzy zmienne przy łącznej liczbie pozymianów równej 5. Stopień złożoności tego zadania wynosi więc $DOD = 5 - 3 - 1 = 1 > 0$. W celu obliczenia $\delta_i \quad i \in \{1,2,3,4,5\}$ musimy więc przyrównać do 0 pochodne cząstkowe z funkcji przeciwnej do logarytmu z $V(\delta)$ analogicznie jak w przykładzie ze strony [88]. Po wykonaniu odpowiednich obliczeń otrzymujemy:

$$\begin{array}{ll} \delta_1 & = 1, \\ \delta_2 & = \frac{1}{2}, \\ \delta_3 & = \frac{2}{5}, \\ \delta_4 & = \frac{6}{5}, \\ \delta_5 & = \frac{9}{5}. \end{array}$$

Długości krawędzi szukanego ostrosłupa obliczone z równań:

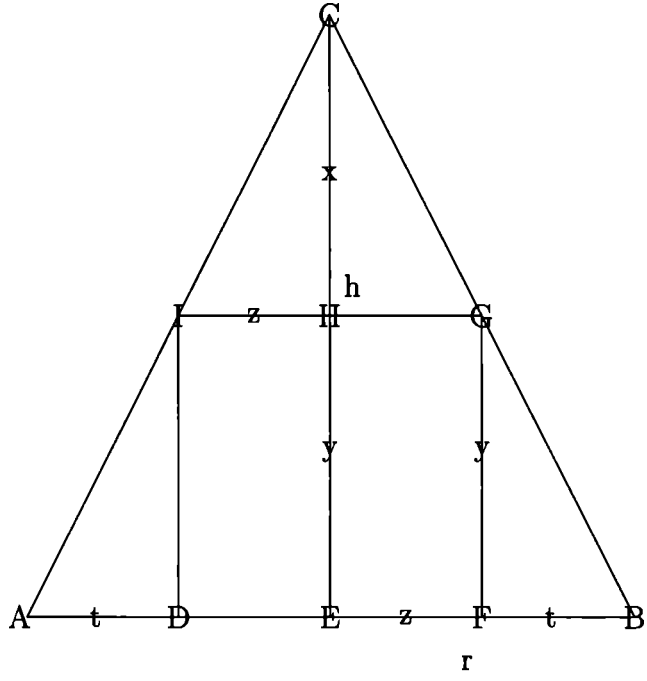
$$\begin{array}{ll} \frac{6 \cdot a}{p} & = \frac{\frac{6}{5}}{\frac{6}{5}+\frac{9}{5}}, \\ \frac{6 \cdot b}{p} & = \frac{\frac{9}{5}}{\frac{6}{5}+\frac{9}{5}}, \end{array}$$

wynoszą

$$\begin{aligned} a &= \tfrac{p}{15}, \\ b &= \tfrac{p}{10}. \end{aligned}$$

## Przykład 15

Wyznaczyć długość promienia podstawy i wysokości walca wpisanego w stożek o danym promieniu podstawy r, którego długość wysokości jest równa h, tak, aby objętość walca była największa.




$\overline{AE} = \overline{EB} = r$ , $\overline{AD} = \overline{FB} = t$ , $\overline{DE} = \overline{EF} = \overline{HG} = \overline{IH} = z$ ,

$\overline{CE} = h$ , $\overline{DI} = \overline{EH} = \overline{FG} = y$, $\overline{HC} = x$
Przy oznaczeniach takich jak na rysunku objętość walca wynosi:

$$V = \pi \cdot z^2 \cdot y \ ,$$

zaś z podobieństwa trójkątów $\triangle_{FBG} \sim \triangle_{HGC}$ wynika, że:

$$\frac{t}{y} = \frac{z}{x} \ .$$

Ponadto:

$$x + y = h \ ,$$
$$z + t = r \ .$$

Problem programowania geometrycznego możemy więc sformułować jako minimalizację funkcji

$$f(x, y, z, t) = \frac{1}{\pi \cdot z^2 \cdot y} \ , \quad x, y, z, t > 0$$

przy ograniczeniach

$$\frac{y}{h} + \frac{x}{h} \leq 1,$$
$$\frac{z}{r} + \frac{t}{r} \leq 1,$$
$$\frac{z \cdot y}{t \cdot x} \leq 1.$$

Problem dualny odpowiadający temu zadaniu to maksymalizacja funkcji

$$V(\delta) = \left(\frac{1}{\pi \cdot \delta_1}\right)^{\delta_1} \cdot \left(\frac{1}{h \cdot \delta_2}\right)^{\delta_2} \cdot \left(\frac{1}{h \cdot \delta_3}\right)^{\delta_3} \cdot (\delta_2 + \delta_3)^{\delta_2 + \delta_3}$$
$$\cdot \left(\frac{1}{r \cdot \delta_4}\right)^{\delta_4} \cdot \left(\frac{1}{r \cdot \delta_5}\right)^{\delta_5} \cdot (\delta_4 + \delta_5)^{\delta_4 + \delta_5} \cdot \left(\frac{1}{\delta_6}\right)^{\delta_6} \cdot (\delta_6)^{\delta_6} ,$$

gdzie $\delta_i \quad \delta_i > 0 \quad i \in \{1, 2, 3, 4, 5, 6\}$ spełniają związki:

$$\begin{array}{llll} \delta_1 & & & = 1, \\ -2 \cdot \delta_1 & +\delta_4 & +\delta_6 & = 0, \\ \delta_5 & -\delta_6 & & = 0, \\ \delta_3 & -\delta_6 & & = 0, \\ -\delta_1 & +\delta_2 & +\delta_6 & = 0. \end{array}$$

W programie pierwotnym występują trzy zmienne przy łącznej liczbie pozymianów równej 6. Stopień złożoności tego zadania wynosi więc $DOD = 6 - 4 - 1 = 2 > 0$. W celu obliczenia $\delta_i$ $(i \in \{1, 2, 3, 4, 5, 6\})$ musimy więc przyrównać do 0 pochodne cząstkowe z funkcji przeciwnej do logarytmu z $V(\delta)$ analogicznie jak w przykładzie ze strony [88]. Po wykonaniu odpowiednich obliczeń otrzymujemy:

$$\begin{aligned} \delta_1 &= 1, \\ \delta_2 &= \tfrac{1}{3}, \\ \delta_3 &= \tfrac{2}{3}, \\ \delta_4 &= \tfrac{4}{3}, \\ \delta_5 &= \tfrac{2}{3}, \\ \delta_6 &= \tfrac{2}{3}. \end{aligned}$$

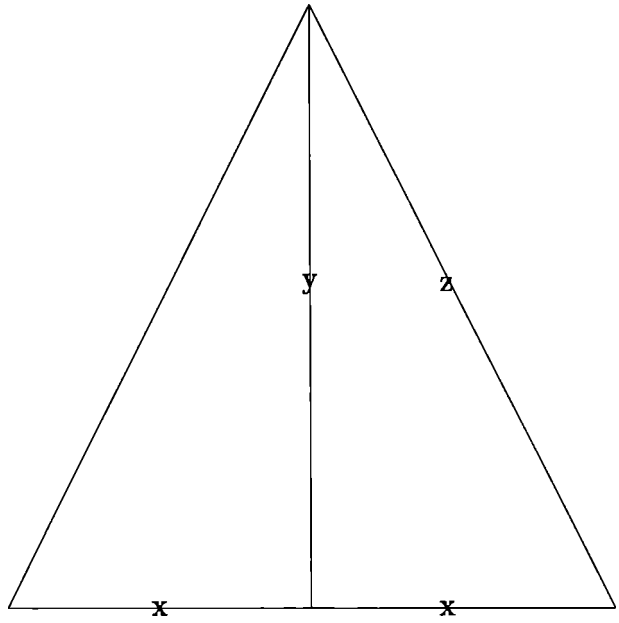
Promień z i wysokość y walca wpisanego w stożek wyznaczymy z zależności:

$$\begin{aligned} \tfrac{z}{r} &= \tfrac{\frac{4}{3}}{\frac{4}{3}+\frac{2}{3}}, \\ \tfrac{y}{h} &= \tfrac{\frac{1}{3}}{\frac{1}{3}+\frac{2}{3}}, \end{aligned}$$

$$\begin{aligned} z &= \tfrac{2}{3} \cdot r, \\ y &= \tfrac{1}{3} \cdot h. \end{aligned}$$

**Przykład 16**

Obwód trójkąta równoramiennego jest równy 2p. Jaką długość winny mieć boki tego trójkąta aby:
a) objętość bryły powstałej przez obrót trójkąta dokoła jego podstawy była największa,
b) objętość bryły powstałej przez obrót trójkąta dokoła jego osi symetrii, była największa.




a) Objętość bryły powstałej przez obrót trójkąta dokoła jego podstawy wynosi przy oznaczeniach jak na rysunku

$$V = 2 \cdot \frac{1}{3}\pi \cdot y^2 \cdot x \ .$$

Z treści zadania oraz z prawa Pitagorasa wynikają ponadto zależności:

$$2 \cdot x + 2 \cdot z = 2 \cdot p \ ,$$

$$x^2 + y^2 = z^2 \ .$$

Zapisując to w języku programowania geometrycznego otrzymujemy następujący problem:

**zminimalizować funkcję**

$$f(x,y,z) = \frac{3}{2 \cdot \pi \cdot y^2 + \pi \cdot x} \quad , x,y,z > 0$$

przy ograniczeniach

$$\frac{x}{p} + \frac{z}{p} \leq 1 \ ,$$

$$\frac{x^2}{z^2} + \frac{y^2}{z^2} \leq 1 \ .$$

Odpowiadający mu problem dualny to maksymalizacja funkcji

$$V(\delta) = \left(\frac{3}{2 \cdot \pi \cdot \delta_1}\right)^{\delta_1} \cdot \left(\frac{1}{p \cdot \delta_2}\right)^{\delta_2} \cdot \left(\frac{1}{p \cdot \delta_3}\right)^{\delta_3} \cdot (\delta_2 + \delta_3)^{\delta_2 + \delta_3}$$

$$\cdot \left(\frac{1}{\delta_4}\right)^{\delta_4} \cdot \left(\frac{1}{\delta_5}\right)^{\delta_5} \cdot (\delta_4 + \delta_5)^{\delta_4 + \delta_5},$$

gdzie $\delta_i \quad (\delta_i > 0, \quad i \in \{1,2,3,4,5\})$ spełniają zależności

$$\begin{array}{llll} \delta_1 & & & = 1, \\ -\delta_1 & +\delta_2 & +2 \cdot \delta_4 & = 0, \\ -2 \cdot \delta_1 & & +2 \cdot \delta_5 & = 0, \\ \delta_3 & -2 \cdot \delta_4 & -2 \cdot \delta_5 & = 0. \end{array}$$

W programie pierwotnym występują trzy zmienne przy łącznej liczbie pozymianów równej 5. Stopień złożoności tego zadania wynosi więc $DOD = 5 - 3 - 1 = 1 > 0$. W celu obliczenia $\delta_i \quad i \in \{1,2,3,4,5\}$ musimy więc przyrównać do 0 pochodne cząstkowe z funkcji przeciwnej do logarytmu z $V(\delta)$ analogicznie jak w przykładzie ze strony [88]. Po wykonaniu odpowiednich obliczeń mamy:

$$\begin{aligned} \delta_1 &= 1 \ , \\ \delta_2 &= \tfrac{3}{4} \ , \\ \delta_3 &= \tfrac{9}{4} \ , \\ \delta_4 &= \tfrac{1}{8} \ , \\ \delta_5 &= 1 \ . \end{aligned}$$

Podstawa 2x i bok y szukanego trójkąta wyznaczamy z równań:

$$\begin{aligned}
\frac{x}{p} &= \frac{\frac{3}{4}}{\frac{3}{4}+\frac{9}{4}} \ , \\
\frac{y}{p} &= \frac{\frac{9}{4}}{\frac{3}{4}+\frac{9}{4}} \ , \\
2x &= \frac{1}{2}\cdot p \ , \\
y &= \frac{1}{2}\cdot p \ .
\end{aligned}$$

b) Objętość bryły powstałej przez obrót trójkąta dokoła jego osi symetrii wynosi

$$V = \frac{1}{3}\pi \cdot y \cdot x^2 \ .$$

Podobnie jak w przykładzie a) zachodzą związki

$$2 \cdot x + 2 \cdot z = 2 \cdot p \ ,$$

$$x^2 + y^2 = z^2 \ .$$

Możemy więc sformułować zadanie jako następujący problem:
**zminimalizować funkcję**

$$f(x,y,z) = \frac{3}{\pi \cdot y + \pi \cdot x^2} \ , x,y,z > 0$$

przy ograniczeniach

$$\frac{x}{p} + \frac{z}{p} \leq 1 \ ,$$

$$\frac{x^2}{z^2} + \frac{y^2}{z^2} \leq 1 \ .$$

Odpowiadający mu problem dualny to maksymalizacja funkcji

$$V(\delta) = \left(\frac{3}{\pi \cdot \delta_1}\right)^{\delta_1} \cdot \left(\frac{1}{p \cdot \delta_2}\right)^{\delta_2} \cdot \left(\frac{1}{p \cdot \delta_3}\right)^{\delta_3} \cdot (\delta_2 + \delta_3)^{\delta_2+\delta_3}$$

$$\cdot \left(\frac{1}{\delta_4}\right)^{\delta_4} \cdot \left(\frac{1}{\delta_5}\right)^{\delta_5} \cdot (\delta_4 + \delta_5)^{\delta_4+\delta_5},$$

gdzie $\delta_i \quad \delta_i > 0 \quad i \in \{1,2,3,4,5\}$ spełniają

$$\begin{array}{llll} \delta_1 & & & = 1, \\ -2 \cdot \delta_1 & +\delta_2 & +2 \cdot \delta_4 & = 0, \\ -\delta_1 & & +2 \cdot \delta_5 & = 0, \\ \delta_3 & -2 \cdot \delta_4 & -2 \cdot \delta_5 & = 0\,. \end{array}$$

Analogicznie jak w przykładzie a) obliczamy optymalne wartości $\delta$

$$\begin{aligned} \delta_1 &= 1\ , \\ \delta_2 &= \tfrac{6}{5}\ , \\ \delta_3 &= \tfrac{9}{5}\ , \\ \delta_4 &= \tfrac{2}{5}\ , \\ \delta_5 &= \tfrac{1}{2}\ . \end{aligned}$$

Podstawa 2x i bok y szukanego trójkąta wynoszą:

$$\begin{aligned} \tfrac{x}{p} &= \frac{\frac{6}{5}}{\frac{6}{5}+\frac{9}{5}}\ , \\ \tfrac{y}{p} &= \frac{\frac{9}{5}}{\frac{6}{5}+\frac{9}{5}}\ , \\ 2x &= \tfrac{4}{5} \cdot p\ , \\ y &= \tfrac{3}{5} \cdot p\ . \end{aligned}$$

# Rozdział 2

# Algorytmy programowania geometrycznego

## 2.1 Zastosowanie metody Newtona-Raphsona w programowaniu geometrycznym

Jak wynika z definicji 1.1 i z dowodu teorii dualności programowania geometrycznego (p.1.6) zagadnienie programowania geometrycznego można sformułować następująco:
**Zminimalizować:**

$$f(x)$$

przy ograniczeniu $A \cdot x = b \ , x \in R_+^{n_p}$
gdzie

$$f(x) = \sum_{i=1}^{n_0} x_i \cdot \ln(\frac{x_i}{c_i}) + \sum_{k=1}^{p} \sum_{i=n_{k-1}+1}^{n_k} x_i \cdot \ln(\frac{x_i}{c_i \cdot \lambda_k})$$

$$\lambda_k = \sum_{i=n_{k-1}+1}^{n_k} x_i \ , k = 1, 2, \ldots, p \ ,$$

$$A = \begin{pmatrix} 1 & \cdots & 1 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ a_{1,1} & \cdots & a_{n_0,1} & a_{n_0+1,1} & \cdots & a_{n_1,1} & \cdots & a_{n_{p-1}+1,1} & \cdots & a_{n_p,1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{1,m} & \cdots & a_{n_0,m} & a_{n_0+1,m} & \cdots & a_{n_1,m} & \cdots & a_{n_{p-1}+1,m} & \cdots & a_{n_p,m} \end{pmatrix}, \tag{2.1}$$

$$b^T = (1, 0, 0, \ldots, 0) \in R^{m+1} \tag{2.2}$$

Po prostych obliczeniach stwierdzamy, że gradient $\nabla f(x)$ równa się

$$\frac{\partial f(x)}{\partial x_i} = \begin{cases} \ln(x_i) - \ln(c(i) + 1, & i = 1, \ldots, n_0 \\ \ln(x_i) - \ln(c(i) - \ln(\lambda_k), & i = n_0 + 1, \ldots, n_p \end{cases}, \tag{2.3}$$

Hesjan $f(x)$ wynosi (p. 1.3.7) zaś

$$H = \begin{bmatrix} H_0 & 0 & 0 & \ldots & 0 \\ 0 & H_1 & 0 & \ldots & 0 \\ 0 & 0 & H_2 & \ldots & 0 \\ \ldots & & & & \\ 0 & 0 & 0 & \ldots & H_p \end{bmatrix}, \tag{2.4}$$

gdzie

$$H_0 = \begin{bmatrix} x_1^{-1} & 0 & \ldots & 0 \\ 0 & x_2^{-1} & \ldots & 0 \\ \ldots & & & \\ 0 & 0 & \ldots & x_{n_0}^{-1} \end{bmatrix}, \tag{2.5}$$

a

$$H_k = -\begin{bmatrix} \lambda_k^{-1} - x_{m_k}^{-1} & \lambda_k^{-1} & \ldots & \lambda_k^{-1} \\ \lambda_k^{-1} & \lambda_k^{-1} - x_{m_{k+1}}^{-1} & \ldots & \lambda_k^{-1} \\ \ldots & & & \\ \lambda_k^{-1} & \lambda_k^{-1} & \ldots & \lambda_k^{-1} - x_{n_k}^{-1} \end{bmatrix} \quad k = 1, 2, \ldots, p, \tag{2.6}$$

W p. 1.3.7 wykazane zostało, że funkcja przeciwna do $f(x)$ jest wklęsła, co oznacza, że $f(x)$ jest wypukłe.
ograniczenie $A \cdot x = b$ , $x \in R_+^{n_p}$ jest natomiast ograniczeniem liniowym

Tak więc problem każdy programowania geometrycznego jest równocześnie programem wypukłym z ograniczeniami liniowymi i jako taki posiada formę dualną Wolfe'go [1]. Stawiamy zadanie w formie następującej:
**zmaksymalizować funkcję:**

$$b^T y - \nabla f(x)^T x + f(x)$$

przy ograniczeniu

$$A^T \cdot y - \nabla f(x) + z = 0$$

gdzie $x, y, z$ należą odpowiednio do:

$$x \in R_+^{n_p} \ , \ y \in R^{m+1} \ , \ z \in R_{++}^{n_p}.$$

W celu ustalenia punktu, w którym $f(x)$ osiąga minimum przyrównamy do zera ograniczenie programu pierwotnego, ograniczenie formy dualnej Wolfe-'go oraz dodatkowo warunek uzupełniający $X \cdot z = 0$ ; $X = diag(x)$, gdzie symbol $diag(x)$ oznacza macierz kwadratową, której kolejne elementy na przekątnej głównej są elementami wektora x (tj. $X \in R^{n_p} \times R^{n_p}$ ; $X_{ii} = x_i$ ; $X_{ij} = 0 \ \ dla \ \ i \neq j$ ; $i, j \in 1, 2, \ldots, n_p$)
Zadanie programowania geometrycznego będzie sprowadzać się więc do rozwiązania układu równań:

$$\begin{array}{rrrcl} & A \cdot x & & = & b \\ A^T \cdot y & -\nabla f(x) & +z & = & 0 \\ & & X \cdot z & = & 0 \end{array} \tag{2.7}$$

gdzie

$$x \in R_+^{n_p} \ , \ y \in R^{m+1} \ , \ z \in R_{++}^{n_p}$$

zwanego układem KKT (Karush-Kuhn-Tucker-a).

Z uwagi na występowanie w układzie równań czynnika nieliniowego $\nabla f(x)$ nie da się do jego rozwiązania zastosować żadnej ze metod eliminacyjnych takich jak np. metoda Gaussa-Jordana.

Najbardziej efektywnym sposbem rozwiązania układu wydaje się zastosowanie metody Newtona-Raphsona, która jest dobrze opisana i często stosowana w Naukach Ekonomicznych [2]

[1] p. [85] str. 239-244
[2] p. [84]

W naszym przypadku użyjemy formy zmodyfikowanej metody Newtona-Raphsona, w której w przeciwieństwie do metody klasycznej[3] kolejne wyrazy ciągu iteracyjnego nie są obliczane ze wzoru:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)},$$

lecz z uwzględnieniem dodatkowego parametru skalującego $\alpha$, którego wartość jest w następnych krokach algorytmu wyznaczana przy pomocy jednej z metod przeszukiwania liniowego (*ang. line search*) np. śledzenia wstecz (*ang. backtracking*) lub złotego podziału. [4] Przyjmując za początkowe wartości

$$x0 = (e, e, \ldots, e) \;;\; y_0 = (0, 0, \ldots, 0) \;;\; z_0 = (e, e, \ldots, e)$$

W kolejnych iteracjach rozwiązujemy najpierw układ równań Newtonowskich:

$$\begin{array}{rrcl} A\Delta_x & & = & \eta r_P^k \\ A^T\Delta_y \quad -H^k\Delta_x & +\Delta_z & = & \eta r_D^k \\ Z^k\Delta_x & +X^k\Delta_z & = & \gamma\mu^k \cdot e - X^k z^k \end{array} \tag{2.8}$$

gdzie:
$\gamma$ i $\eta$ są parametrami należącymi do przedziału [0,1]
reszta pierwotna $r_P$ i dualna $r_D$ *primal residual, dual residual* są zdefiniowane:

$$r_P = b - Ax\,, \tag{2.9}$$

$$r_D = \nabla f(x) - A^T y - z\,, \tag{2.10}$$

zaś reszta uzupełniająca *complemantarity residual* to

$$\mu = \frac{x^T \cdot z}{n_p} \tag{2.11}$$

po rozwiązaniu układu 2.8 wyznaczamy wartość $\alpha$ tak aby podstawienie

$$(x^{k+1}, y^{k+1}, z^{k+1}) \leftarrow (x^k, y^k, z^k) + \alpha \cdot (\Delta_x^k, \Delta_y^k, \Delta_z^k)$$

redukowało w sposób maksymalny wartości reszty pierwotnej i dualnej $r_p$ i $r_d$.

[3] p. [64] str. 220
[4] p. [32]

## 2.2 Algorytm "Infeasible Interior Point"

Na metodzie opisanej w poprzednim paragrafie oparta jest cała grupa algorytmów, [5] z których najbardziej efektywnym jest algorytm "Infeasible Interior Point" zaproponowany w 1994 r. przez K. Kortanka, Xiaojie Xu i Yinyu Ye w [60], który pozwala na numeryczne rozwiązywanie zadań programowania geometrycznego w czasie porównywalnym z algorytmami programowania liniowego. [6]

Dodatkowym założeniem twórców algorytmu jest ograniczenia kolejnych wartości $x$ i $z$ do

$$\begin{array}{rcl} Xz & \geq & \sigma\mu \\ \| Xz - \mu e \| & \leq & \beta\mu \end{array} \tag{2.12}$$

Kolejne kroki tego algorytmu przedstawiają się więc następująco:

**Krok 1: Podstawienia początkowe**

Przyjmujemy:

$$x_0 = (e, e, \ldots, e) \ ; y_0 = (0, 0, \ldots, 0) \ z_0 = (\mu_0 \cdot e, \mu_0 \cdot e, \ldots, \mu_0 \cdot e)$$

Ustalamy wartości stałych używanych w algorytmie:

$\sigma, \beta$ - (p. wzór 2.12)

$\epsilon_P, \epsilon_D, \epsilon_C$ - toleracja pierwotnej, dualnej i uzupełniającej do określenia dokładności wyników

$\Theta_P, \Theta_D, \Theta_C$ - stałych używanych, w przeszukiwaniu liniowym (krok 5.c algorytmu)

$F_\alpha \in (0, 1)$ - współczynnika skalującego dla $\alpha$ - (p. krok 4.d algorytmu)

$F_z$ - współczynnika skalującego dla zmiennej $z$ - (p. krok 7 algorytmu)

---

[5] Innego typu algorytmy rozwiązywania zadań programowania geometrycznego są opisane np. w [7] ,[42], [73]

[6] p. [60] str 22-23, [91] , [91]

**dopóki nie są spełnione warunki**

$$\frac{\| r_P \|}{1+\| x \|} \leq \epsilon_P \ , \ \frac{\| r_D \|}{1+\| z \|} \leq \epsilon_D \ , \ \frac{x^T z}{1+\| x \| + \| z \|} \leq \epsilon_C \ ,$$

Powtarzaną są kroki 2-7

**Krok 2: Utworzenie układu KKT**

Dokonujemy obliczeń potrzebnych do utworzenia aktualnego układu KKT (p.2.7)

obliczamy dla aktualnych wartości $x^k, y^k, z^k$ gradient f(x) (p. 2.3), Hessjan f(x) (p. 2.4-6), resztę pierwotną i resztą dualną (p. 2.9 i 10),

**Krok 3: Znalezienie parametrów $\eta$ i $\gamma$**

a) Przyjmujemy $\eta = 1$ i $\delta = 0$ i rozwiązujemy układ (2.8) wyznaczając $(\Delta_x^a, \Delta_y^a, \Delta_z^a)$

b) Obliczmy $\alpha^a = min(\alpha_x^a, \alpha_z^a)$ gdzie

$$\alpha_x^a = \max\{-\frac{x_j}{(\Delta_x^a)_j} : (\Delta_x^a)_j < 0\}$$

$$\alpha_z^a = \max\{-\frac{z_j}{(\Delta_z^a)_j} : (\Delta_z^a)_j < 0\}$$

c) Obliczamy

$$ma^a = \frac{(x + \alpha_a \Delta_x^a)^T (z + \alpha_a \Delta_z^a)}{n_p}$$

d) i ostatecznie ustalamy $\gamma$ i $\eta$ na:

$$\gamma = 0,5 \cdot (\mu_a \mu) \ , \ \eta = 1 - \gamma \ .$$

**Krok 4: Wyznaczenie $\Delta_x, \Delta_y, \Delta_z$**

Rozwiązujemy układ:

$$\begin{array}{rrcl} A\Delta_x & & = & \eta r_P^k \\ A^T\Delta_y \quad -H^k\Delta_x & +\Delta_z & = & \eta r_D^k \\ Z^k\Delta_x & +X^k\Delta_z & = & \gamma\mu^k \cdot e - X^k z^k \end{array}$$

i wyznaczamy $\Delta_x, \Delta_y, \Delta_z$

**Krok 5: Wyznaczenie parametru $\alpha$**

a) Obliczmy $\alpha^R = min(\alpha_x, \alpha_z)$ gdzie

$$\alpha_x^a = \max\{-\frac{x_j}{(\Delta_x)_j} : (\Delta_x)_j < 0\}$$

$$\alpha_z^a = \max\{-\frac{z_j}{(\Delta_z)_j} : (\Delta_z)_j < 0\}$$

b) Wyznaczamy $\alpha^N$ tak, aby po podstawieniu $x \leftarrow x + \alpha \cdot \Delta_x$ , $z \leftarrow z + \alpha \cdot \Delta_z$ spełninone było:

$$\begin{array}{rcl} Xz & \geq & \sigma\mu \\ \| Xz - \mu e \| & \leq & \beta\mu \end{array}$$

c) Wykorzystując algorytmy przeszukiwania liniowego i znajdujemy $\alpha_L$ spełniające:

$$\begin{array}{rcl} \| b - A \cdot (x + \alpha\Delta_x) \| & \leq & \Theta_P \cdot (x + \alpha\Delta_X)^T \cdot (z + \alpha\Delta_z) \\ \| \nabla f(x + \alpha\Delta_x) - A^T(y + \alpha\Delta_y) - (z + \alpha\Delta_z) \| & \leq & \Theta_D \cdot (x + \alpha\Delta_X)^T \cdot (z + \alpha\Delta_z) \\ (x + \alpha\Delta_x)^T(z + \alpha\Delta_z) & \leq & \Theta_C \cdot x^T z \end{array}$$

d) Ostatecznie ustalamy $\alpha$

$$\alpha = min\{F_\alpha \cdot \alpha_R, \alpha_N, \alpha_L\}$$

**Krok 6: Uaktualnienie zmiennych**

$$(x, y, z) \leftarrow (x, y, z) + \alpha \cdot (\Delta_x, \Delta_y, \Delta_z)$$

**Krok 7: Przeskalowanie $z^k$**

Aby uniezależnić działanie algorytmu od wartości stałych $\Theta_P, \Theta_D, \Theta_C$ i zredukować $r_D$ zmienimy wartość zmiennej $z$ zgodnie ze wzorem:

a) Niech $s = \nabla f(x) - A^T y$

b) Ustalamy:

$$z_i = \begin{cases} z_i & gdy \quad s_i < 0 \\ s_i & gdy \quad s_i \geq 0 \quad \wedge \quad s_i \in \left(\frac{z_i}{F_z}, z_i \cdot F_z\right) \\ \frac{z_i}{F_z} & gdy \quad s_i \geq 0 \quad \wedge \quad s_i \leq \frac{z_i}{F_z} \\ z_i \cdot F_z & gdy \quad s_i \geq 0 \quad \wedge \quad s_i \geq z_i \cdot F_z \end{cases}$$

gdzie $i \in \{1, 2, \ldots n_P\}$ a $F_z$ jest stałą większą od jedności ustaloną w p. 1 algorytmu.[7]

## 2.3 Opis programu PG.EXE jako implementacji algorytmu „Infeasible Interior Point"

Program PG jest implementacją algorytmu „Infeasible Interior Point" w języku C++. Do jego kompilacji użyto pakietu Microsoft Visual C 1.5 w połączeniu z biblioteką newmat v. 0.9 służącą do operacji na macierzach. [8]

Do swojej pracy program potrzebuje systemu operacyjnego Microsoft Windows 3.x, Microsoft Windows 95/98 lub Microsoft Windows NT.

Program po uruchomieniu pozwala na wybór pliku z danymi. Pliki tego rodzaju mają domyślnie rozszerzenie *.dat. Wybór odbywa się poprzez standardowe okno wyboru plików systemu MS Windows.

Po zakończeniu pracy program umieszcza wyniki swojego działania w pliku WYNIK.TXT

[7] w [60] proponowaną wartością $F_z$ jest $10^2$

[8] przykładowe kody programów w języku C i w Fortranie związanych z programowaniem nieliniowym znajdują się w [18], [45], [76], [77] p. również [67]

W przypadku gdy zadanie p.g. nie posiada optymalnego rozwiązania program (lub nie jest możliwe znalezienie go ze względów numerycznych np. wartość optymalna przekroczy możliwości zapisu danych typu *double* [9]), skończy swoją pracę komunikatem, że została przekroczona maksymalna liczba iteracji i nie wypisuje żadnego rozwiązania do pliku WYNIK.TXT

Format pliku z danymi, jak i format pliku z wynikami jest opisany poniżej.

Treść programu znajduje się w dodatku A.

### 2.3.1 Format pliku wejściowego

W kolejnych liniach pliku z danymi (*.dat) dla programu PG powinny znajdować się:

wiersz 1 - liczba zmiennych problemu pierwotnego,
wiersz 2 - łączna liczba ograniczeń (p),
wiersz 3 - liczba pozymianów w funkcji minimalizowanej,
wiersz 4 - liczba pozymianów w 1-szej nierówności ograniczającej,
wiersz 5 - liczba pozymianów w 2-giej nierówności ograniczającej,
⋮
wiersz 4+p - liczba pozymianów w p-tej nierówności ograniczającej.

W następnych wierszach opisywane są kolejno pozymiany składające się na funkcję minimalizowaną i na funkcje ograniczające w formie:

- współczynnik przy pozymianie ($c_{ij}$) ,
- liczba czynników (zmiennych) występujących w pozymianie ,
- numer zmiennej ($x_1, x_2, \ldots x_n$) i potęga w której ta zmienna występuje ($\alpha$) w pierwszym czynniku pozymianu ,
- numer zmiennej ($x_1, x_2, \ldots x_n$) i potęga w której ta zmienna występuje ($\alpha$) w drugim czynniku pozymianu ,

⋮

Przykładowo, jeżeli chcemy obliczyć wartość minimalną funkcji:

$$f(x,y) = y^3 \cdot \left(1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \frac{x^6}{6!} + \frac{x^7}{7!} + \frac{x^8}{8!}\right) + \frac{1}{y} + \frac{1}{x^2} \ ,$$

przy ograniczeniu:

$$x^2 + y^2 \leq 1 \ ,$$

[9] p. [57]

To zawartość pliku EXP.DAT powinna wyglądać następująco:

```
2
1
11
2
1.0
1
2  3.0
1.0
2
1  1.0
2  3.0
0.5
2
1  2.0
2  3.0
0.16666
2
1  3.0
2  3.0
0.04166
2
1  4.0
2  3.0
0.00833
2
1  5.0
2  3.0
.00138
2
1  6.0
2  3.0
0.000198
2
1  7.0
2  3.0
0.0000248
```

```
2
1  8.0
2  3.0
1.0
1
2  -1.0
1.0
1
1  -2.0
1.0
1
1  2.0
1.0
1
2  2.0
```

### 2.3.2 Zawartość pliku wyjściowego

W pliku wyjściowym programu GP znajdują się kolejno:

- Winieta pliku;
- Funkcja pierwotna i ograniczenia;
- Parametry programu pierwotnego;
  - Liczba zmiennych;
  - Liczba funkcji ograniczajacych;
  - Liczba pozymianow w funkcji;
  - Laczna liczba pozymianow;
  - Stopien zlozonosci(DOD);
- Optymalne $x_i$;
- Maksymalna wartosc funkcji dualnej.

Dla przykładu z poprzedniego paragrafu plik WYNIK.TXT wygląda następująco:

```
PROGRAMOWANIE GEOMETRYCZNE - OPTYMALIZACJA
Andrzej Dudek      1998/99
Akademia Ekonomiczna, Wroclaw
```

```
WGRiT Jelenia Gora

Program jest czescia pracy doktorskiej:
"Programowanie Gemetryczne jako narzedzie optymalizacji w ekonomii"
Pisanej pod opieka Profesora Michala Montygierda - Loyby
------------------------------------------------------------------

Program Pierwotny:

Zminimalizowac funkcje:

f(x) = 1.000*x2^(3.000) + 1.000*x1^(1.000)*x2^(3.000) + 0.500
*x1^(2.000)*x2^(3.000) + 0.167*x1^(3.000)*x2^(3.000) + 0.042
*x1^(4.000)*x2^(3.000) + 0.008*x1^(5.000)*x2^(3.000) + 0.001
*x1^(6.000)*x2^(3.000) + 0.000*x1^(7.000)*x2^(3.000) + 0.000
*x1^(8.000)*x2^(3.000) + 1.000*x2^(-1.000) + 1.000*x1^(-2.000)

Przy ograniczeniach:

1.000*x1^(2.000) + 1.000*x2^(2.000) <= 1
------------------------------------------------------------------

Liczba zmiennych:            2
Liczba funkcji ograniczajacych:   1
Liczba pozymianow w funkcji:     11
Laczna liczba pozymianow:   13
Degree of Dificulty:           10
------------------------------------------------------------------

Program Dualny:

delta1 = 0.039110
delta2 = 0.033505
delta3 = 0.014878
delta4 = 0.005078
delta5 = 0.001976
delta6 = 0.001082
delta7 = 0.000759
```

```
delta8 = 0.000611
delta9 = 0.000529
delta10 = 0.528631
delta11 = 0.376423
delta12 = 0.328361
delta13 = 0.121424

Wynik = ((25.569) ^ (0.039) * ((29.846) ^ (0.034) * ((33.607) ^ (0.015)
* ((32.820) ^ (0.005) * ((21.087) ^ (0.002) * ((7.700) ^ (0.001) *
((1.818) ^ (0.001) * ((0.324) ^ (0.001) * ((0.047) ^ (0.001) * ((1.892)
^ (0.529) * ((2.657) ^ (0.376) * ((3.045) ^ (0.328) * ((8.236) ^ (0.121)
* ((0.450) ^ (0.450)= 3.624012

--------------------------------------------------------------------------

X1 = 0.8567
X2 = 0.5208
```

# Rozdział 3

# Zastosowanie programowania geometrycznego w ekonomii

## 3.1 Funkcje użyteczności Cobba - Douglasa a programowanie geometryczne

### 3.1.1 Funkcje Cobba - Douglasa

Programowanie geometryczne może być z powodzeniem stosowane do rozpatrywania wielu zagadnień ekonomicznych. Funkcje występujące w tego typu problemach często mogą w prosty sposób być przekształcone do pozymianów. Jako przykład takiego typu problemu przedstawmy zadanie zmaksymalizowania funkcji użyteczności typu Cobba - Douglasa

Funkcje użyteczności typu Cobba-Douglasa są określone następująco [1]

$$f(x,y) = x^c \cdot y^d$$

Przy ograniczeniu

$$\alpha_1 \cdot x + \alpha_2 \cdot y = m\,.$$

Po prostych przekształceniach możemy sformułować zadanie programowania geometrycznego następująco:

Zminimalizować funkcję

$$g(x,y) = x^{-c} \cdot y^{-d}$$

[1] p. [83], [8], [78]

przy założeniu, że

$$\frac{\alpha_1}{m}x + \frac{\alpha_2}{m}y \leq 1\,.$$

Zastosowanie metod programowania geometrycznego pozwala stwierdzić, że $g(x,y)$ osiąga minimum dla

$$\left(\frac{1}{\delta_1}\right)^{\delta_1}\left(\frac{\alpha_1}{m\delta_2}\right)^{\delta_2}\left(\frac{\alpha_2}{m\delta_3}\right)^{\delta_3}(\delta_2+\delta_3)^{\delta_2+\delta_3}\,,$$

przy czym

$$\begin{array}{llll} \delta_1 & & & = 1\,, \\ -c\delta_1 & +\delta_2 & & = 0\,, \\ -d\delta_1 & & \delta_3 & = 0\,. \end{array}$$

stąd

$$\delta_1 = 1, \delta_2 = c, \delta_3 = d\,.$$

zatem minimalną wartością funkcji f jest

$$\left(\frac{\alpha_1}{mc}\right)^{-c}\left(\frac{\alpha_2}{md}\right)^{-d}(c+d)^{c+d}\,,$$

lub inaczej, pamiętając, że minimalizowaliśmy funkcję odwrotną

$$g_0 = \left(\frac{mc}{\alpha_1(c+d)}\right)^{-c}\left(\frac{md}{\alpha_2(c+d)}\right)^{-d}\,,$$

Daje to wynik zgodny z tym, jaki otrzymalibyśmy stosując metody różniczkowe.

### 3.1.2 Funkcje typu Cobba - Douglasa wielu zmiennych

Uogólniając poprzedni przykład możemy zastanowić się nad maksymalizacją funkcji:

$$f(x) = \prod_{i=1}^{n} x^{\alpha_i}\ \ , x \in R_+^n\ \ ,$$

Przy ograniczeniach

$$\sum_{i=1}^{n} p_i \cdot x_i = m$$

[2] Formułując inaczej problem mamy zadanie minimalizacji funkcji

$$g(x) = \prod_{i=1}^{n} x^{-\alpha_i} \ , x \in R_+^n \ ,$$

przy ograniczeniach

$$\sum_{i=1}^{n} \frac{p_i \cdot x_i}{m} \leq 1.$$

Jest to więc problem programowania geometrycznego o n - zmiennych i łącznej liczbie pozymianów równej 1+n, czyli stopniu złożoności DOD=n+1-n-1=0. Optymalizacja g(x) sprowadza więc do rozwiązania układu równań liniowych.

Zamiast minimalizować $g(x)$ możemy **maksymalizować:**

$$V(\delta) = \left(\frac{1}{\delta_1}\right)^{\delta_1} \cdot \prod_{i=2}^{n+1} \left(\frac{p_{i-1}}{m \cdot \delta_i}\right)^{\delta_i} \cdot \left(\sum_{i=2}^{n+1} \delta_i\right)^{\sum_{i=2}^{n+1} \delta_i} \ ,$$

przy czym zachodzą związki

$$\begin{array}{lll} \delta_1 & & = 1 \, , \\ -\alpha_i \cdot \delta_1 & +\delta_{i+1} & = 0 \end{array} \quad i \in \{1, 2, \ldots, n\} \ .$$

stąd

$$\delta_1 = 1, \delta_2 = \alpha_1, \delta_3 = \alpha_2, \ldots, \delta_{n+1} = \alpha_n \, .$$

Punktem, w którym g(x) osiąga minimum, a f(x) maksimum jest więc:

$$\begin{array}{ll} \frac{p_i}{m} \cdot x_i & = \frac{\alpha_i}{\sum_{j=1}^{n} \alpha_j} \ , \\ & \qquad i \in \{1, 2, \ldots, n\} \ . \\ x_i & = \frac{\alpha_i}{p_i \cdot \sum_{j=1}^{n} \alpha_j} \cdot m \ . \end{array}$$

[2]p. [78] str 18.

## 3.2 Wykorzystanie metod programowania geometrycznego do minimalizacji kosztów

Wykorzystanie metod programowania geometrycznego w ekonomii można podzielić na trzy kategorie
a) Funkcje typu Cobba-Douglasa
b) Optymalne wykorzystanie materiałów.
c) Minimalizacja funkcji kosztów przy nieliniowych ograniczeniach.

Funkcjami typu Cobba-Douglasa zajmowaliśmy się w poprzednim paragrafie Przykładowe zadanie wykorzystujące te funkcje może mieć postać (przykład 18).

**Przykład 17**

Funkcja produkcji dziennej w firmie produkującej batony czekoladowe ma następującą postać:

$$f(S_1, S_2) = S_1^{\frac{1}{2}} \cdot S_2^{\frac{1}{3}}$$

$S_1$-nakłady surowca 1-czekolady
$S_2$-nakłady surowca 2-karmelu

Jeżeli cena batonów wynosi 1 zł, cena kilograma czekolady 10zł, cena kg karmelu 20 zł, przy stałych kosztach produkcji 5000 zł/dzień, jakie rozmiary produkcji dziennej zoptymalizują wynik finansowy firmy ?

Zadanie w jezyku programowania geometrycznego da się sformułować następująco:

**Zminimalizować funkcję:**

$$g(S_1, S_2) = S_1^{-\frac{1}{2}} \cdot S_2^{-\frac{1}{3}},$$

gdzie

$$\frac{10}{5000} \cdot S_1 + \frac{20}{5000} \cdot S_2 \leq 1\,.$$

Zadanie dualne ma postać:
**Zmaksymalizować funkcję:**

$$\left(\frac{1}{\delta_1}\right)^{\delta_1}\left(\frac{10}{5000\delta_2}\right)^{\delta_2}\left(\frac{20}{5000\delta_3}\right)^{\delta_3}(\delta_2+\delta_3)^{\delta_2+\delta_3},$$

przy czym

$$\begin{array}{llll} \delta_1 & & & = 1, \\ -\frac{1}{2}\cdot\delta_1 & +\delta_2 & & = 0, \\ -\frac{1}{3}\delta_1 & & \delta_3 & = 0. \end{array}$$

stąd

$$\delta_1 = 1, \delta_2 = \frac{1}{2}, \delta_3 = \frac{1}{3}.$$

zatem minimalną wartością funkcji f jest:

$$\left(\frac{10}{5000\cdot\frac{1}{2}}\right)^{-\frac{1}{2}}\left(\frac{20}{5000\cdot\frac{1}{3}}\right)^{-\frac{1}{3}}(\frac{1}{2}+\frac{1}{3})^{\frac{1}{2}+\frac{1}{3}},$$

i jest ona osiągana przy $S_1, S_2$ określonych równościami:

$$\begin{array}{rl} \frac{10}{5000}\cdot S_1 & = \frac{c}{c+d}, \\ \frac{20}{5000}\cdot S_2 & = \frac{d}{c+d}, \end{array}$$

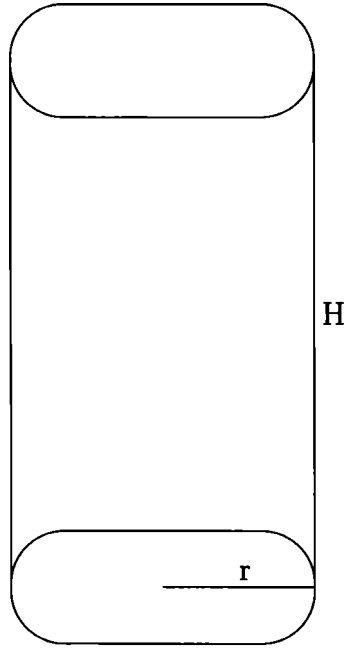
$$\begin{array}{rl} S_1 & = \frac{\frac{1}{2}}{\frac{1}{2}+\frac{1}{3}}\cdot 500, \\ S_2 & = \frac{\frac{1}{3}}{\frac{1}{2}+\frac{1}{3}}\cdot 250. \end{array}$$

Prześledźmy na przykładach pozostałe domeny, w których programowanie geometryczne ma zastosowanie (przykłady 18-23).

### 3.2.1 Optymalne wykorzystanie materiałów

**Przykład 18**

Należy zaprojektować otwarte naczynie w kształcie walca do przewozu płynnych materiałów o pojemności $500\pi\ m^3$. Koszt materiału na boczną ścianę walca wynosi 50 zł/$m^2$. Koszt podstawy to 100 zł/$m^2$.




Uwzględniając, że

$$500\pi = \pi r^2 H$$

otrzymujemy zadanie:

**Zminimalizować**

$$g(H, r) = 50 \cdot 2\pi r H + 100 \cdot \pi \cdot r^2$$

przy założeniu, że

$$\frac{500}{r^2 H} \leq 1\,.$$

Minimalna wartość tej funkcji będzie wynosiła:

$$\left(\frac{100\pi}{\delta_1}\right)^{\delta_1} \left(\frac{100\pi}{\delta_2}\right)^{\delta_2} \left(\frac{500\pi}{\delta_3}\right)^{\delta_3} (\delta_3)^{\delta_3}\,,$$

gdzie $\delta_1, \delta_2, \delta_3$ są wyznaczone równaniami:

$$\begin{aligned} \delta_1 &+\delta_2 & &= 1, \\ \delta_1 & & -\delta_3 &= 0, \\ \delta_1 &+2\delta_2 &-2\delta_3 &= 0, \end{aligned}$$

którego rozwiązaniem są liczby:

$$\delta_1 = \frac{2}{3},$$

$$\delta_2 = \frac{1}{3},$$

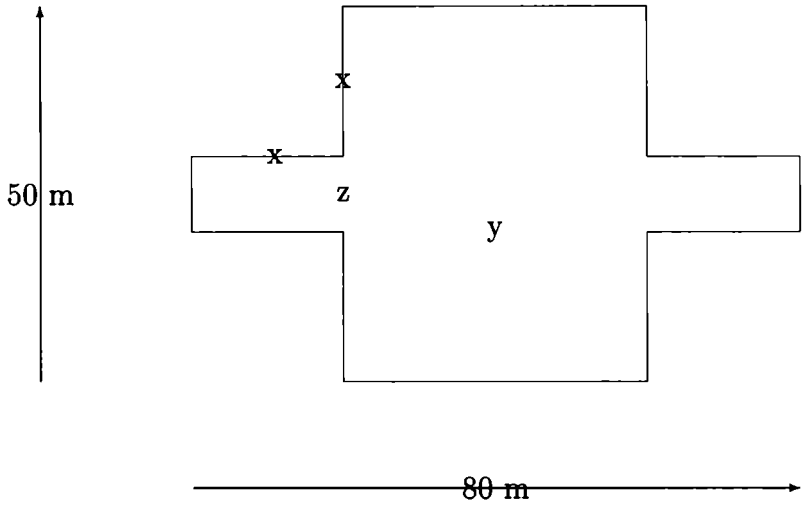
$$\delta_3 = \frac{2}{3}.$$

Czyli minimalna wartość wynosi:

$$(150\pi)^{\frac{2}{3}}(300\pi)^{\frac{1}{3}}(750)^{\frac{2}{3}}(\frac{2}{3})^{\frac{2}{3}}.$$

**Przykład 19**

Z prostokątnej blachy o wymiarach 80 na 50 m należy utworzyć naczynie w kształcie otwartego prostopadłościanu. Jakie powinny być wymiary tego naczynia aby jego pojemność była największa z możliwych?




Zgodnie z oznaczeniami z rysunku, pojemność projektowanego naczynia wynosi $x \cdot y \cdot z$. Nietrudno zauważyć że pomiędzy poszczególnymi bokami prostopadłościanu zachodzą związki $2 \cdot x + y = 80m$ i $2 \cdot x + z = 50m$

Nasze zadanie da się więc sformułować następująco.

**Zmaksymalizować:**

$$f(x, y, z) = x \cdot y \cdot z$$

przy ograniczeniach

$$x, y, z > 0,$$

$$2 \cdot x + y \leq 80m,$$

$$2 \cdot x + z \leq 50m,$$

lub po drobnych modyfikacjach
**zminimalizować:**

$$g(x) = \frac{1}{x_1 \cdot x_2 \cdot x_3}, \qquad x \in R_+^3$$

przy ograniczeniach

$$x_i > 0 \quad , i \in \{1, 2, 3\},$$

$$\frac{x_1}{40} + \frac{x_2}{80} \leq 1,$$

$$\frac{x_1}{25} + \frac{x_3}{50} \leq 1.$$

Przy trzech zmiennych w powyższym zadaniu występuje pięć pozymianów, jest to więc przykład zadania programowania geometrycznego o stopniu złożoności równym 5-3-1=1.
**Równoważnym zadaniem dualnym jest maksymalizacja funkcji**

$$V(\delta) = \left(\frac{1}{\delta_1}\right)^{\delta_1} \cdot \left(\frac{1}{40 \cdot \delta_2}\right)^{\delta_2} \cdot \left(\frac{1}{80 \cdot \delta_3}\right)^{\delta_3} \cdot \left(\frac{1}{25 \cdot \delta_4}\right)^{\delta_4} \cdot \left(\frac{1}{50 \cdot \delta_5}\right)^{\delta_5}$$

$$\cdot(\delta_2 + \delta_3)^{\delta_2+\delta_3} \cdot (\delta_4 + \delta_5)^{\delta_4+\delta_5} \quad \delta \in R^5, \ \delta \geq 0,$$

przy ograniczeniach

$$\begin{array}{llll} \delta_1 & & & = 1, \\ -\delta_1 & +\delta_2 & +\delta_4 & = 0, \\ -\delta_1 & +\delta_3 & & = 0, \\ -\delta_1 & +\delta_3 & & = 0, \end{array}$$

Po dokonaniu prostych przekształceń otrzymujemy:

$$\begin{array}{lll} \delta_1 & & = 1, \\ \delta_2 & +\delta_4 & = 1, \\ \delta_3 & & = 1, \\ \delta_3 & & = 0. \end{array}$$

A więc naszym zadaniem jest maksymalizacja funkcji:

$$V(\delta_2) = 1 \cdot \left(\frac{1}{40 \cdot \delta_2}\right)^{\delta_2} \cdot \left(\frac{1}{80}\right) \cdot \left(\frac{1}{25 \cdot (1-\delta_2)}\right)^{1-\delta_2} \cdot \left(\frac{1}{50}\right)$$

$$\cdot(\delta_2+1)^{\delta_2+1} \cdot (2-\delta_2)^{2-\delta_2} \quad \delta_2 \in <0,1>,$$

lub równoważnie minimalizacja funkcji przeciwnej do logarytmu z $V(\delta_2)$

$$v(\delta_2) = -\ln(1) + \delta_2 \cdot \ln(40 * \delta_2) + \ln(80) + (1-\delta_2) \cdot \ln(25 \cdot (1-\delta_2)) + \ln(50)$$

$$-(1+\delta_2) \cdot \ln(1+\delta_2) - (2-\delta_2) \cdot \ln(2-\delta_2) \ ,$$

Po obliczeniu pochodnej z funkcji $v(\delta_2)$

$$v'(\delta_2) = 3 \cdot \ln(2) - \ln(5) + \ln(\delta_2) - \ln(1-\delta_2) - \ln(1+\delta_2) + ln(2-\delta_2)\,,$$

i przyrównaniu jej do zera

$$\ln(\frac{8}{5}) - \ln((1-\delta_2)(1+\delta_2) + \ln(\delta_2(2-\delta_2)) = 0\,,$$

otrzymujemy

$$\frac{8 \cdot (2 \cdot \delta_2 - \delta_2^2)}{5 \cdot (1-\delta_2^2)} = 1\,,$$

$$3 \cdot \delta_2^2 - 16 \cdot \delta_2 + 5 = 0\,,$$

skąd wnioskujemy, że $v(\delta_2)$ osiąga wartość minimalną dla $\delta_2 = \frac{1}{3}$ ($\delta_4 = \frac{2}{3}$), czyli zgodnie z twierdzeniami programowania geometrycznego wartością maksymalną $V(\delta)$ a co za tym idzie wartością minimalną g(x) jest

$$\left(\frac{3}{40}\right)^{\frac{1}{3}} \cdot \frac{1}{80} \cdot \left(\frac{3}{50}\right)^{\frac{2}{3}} \cdot \frac{1}{50} \cdot \left(\frac{4}{3}\right)^{\frac{4}{3}} \cdot \left(\frac{5}{3}\right)^{\frac{5}{3}} = 18000^{-1}$$

Wartość ta jest osiągana przy następujących długościach $x_1, x_2, x_3$:

$$x_1 \cdot x_2 \cdot x_3 = 18000\,,$$

$$\frac{x_1}{40} = \frac{\frac{1}{3}}{\frac{4}{3}}\,,$$

$$\frac{x_2}{80} = \frac{1}{\frac{4}{3}}\,,$$

$$\frac{x_1}{25} = \frac{\frac{2}{3}}{\frac{5}{3}},$$

$$\frac{x_3}{50} = \frac{1}{\frac{5}{3}},$$

czyli:

$$x_1 = 10,$$

$$x_2 = 60,$$

$$x_3 = 30,$$

lub wracając do oznaczeń z rysunku przy

$$x = 10\,m.\ ,$$

$$y = 60\,m.\ ,$$

$$z = 30\,m.\ .$$

Odpowiada to wynikom otrzymanym przy pomocy "klasycznych" metod ekstremalizacji funkcji.

Również problem odwrotny do powyższego da się rozwiązać przy pomocy programowania geometrycznego

Rozpatrzmy następujący przykład

**Przykład 20**

Z prostokątnej blachy należy utworzyć naczynie w kształcie otwartego prostopadłościanu o pojemności 54000 litrów. Jakie powinny być wymiary tego naczynia aby koszty jego wytworzenia były jak najmniejsze jeżeli $m^2$ blachy kosztuje 20 zł, a koszt robocizny jest stały i wynosi 500 zł.

Przy oznaczeniach jak w poprzednim przykładzie możemy sformułować to zadanie jakonastępująco:

**Zminimalizować:**

$$f(x,y,z) = \frac{20zl \cdot (y + 2 \cdot x) \cdot (z + 2 \cdot x)}{1000} + 500zl\,,$$

przy spełnionym warunku

$$x, y, z > 0$$

$$x \cdot y \cdot z = 54000\, dm^3.,$$

lub po pominięciu stałych, nie mających wpływu na wartość ekstremum, w formie:

**Zminimalizować:**

$$g(x) = 4 \cdot x_1^2 + 2 \cdot x_1 \cdot x_2 + 2 \cdot x_1 \cdot x_3 + x_2 \cdot x_3\,,$$

przy ograniczeniu

$$x_i > 0 \quad , i \in \{1, 2, 3\},$$

$$\frac{54000}{x_1 \cdot x_2 \cdot x_3} \leq 1\,,$$

Jest to więc zadanie programowania geometrycznego o stopniu złożoności równym 5-3-1=1.

Odpowiadający mu program dualny to

**Zmaksymalizować:**

$$V(\delta) = \left(\frac{4}{\delta_1}\right)^{\delta_1} \cdot \left(\frac{2}{\delta_2}\right)^{\delta_2} \cdot \left(\frac{2}{\delta_3}\right)^{\delta_3} \cdot \left(\frac{1}{\delta_4}\right)^{\delta_4} \cdot \left(\frac{54000}{\delta_5}\right)^{\delta_5} \cdot (\delta_5)^{\delta_5}$$

$$\delta \in R^5 \quad , delta \geq 0\,,$$

przy ograniczeniach

$$\begin{array}{rrrrrl} \delta_1 & +\delta_2 & +\delta_3 & +\delta_4 & & = 1\,, \\ 2 \cdot \delta_1 & +\delta_2 & +\delta_3 & & -\delta_5 & = 0\,, \\ & \delta_2 & & +\delta_4 & -\delta_5 & = 0\,, \\ & & \delta_3 & +\delta_4 & -\delta_5 & = 0\,. \end{array}$$

Zapisując inaczej ograniczenia otrzymujemy:

$$\begin{array}{ll} \delta_1 & = \frac{1}{3} - \delta_2\,, \\ \delta_3 & = \delta_2\,, \\ \delta_4 & = \frac{2}{3} - \delta_2\,, \\ \delta_5 & = \frac{2}{3}\,. \end{array}$$

Funkcja $V(\delta)$ przybiera więc postać

$$V(\delta_2) = \left(\frac{4}{\frac{1}{3}-\delta_2}\right)^{\frac{1}{3}-\delta_2} \cdot \left(\frac{2}{\delta_2}\right)^{\delta_2} \cdot \left(\frac{2}{\delta_2}\right)^{\delta_2} \cdot \left(\frac{1}{\frac{2}{3}-\delta_2}\right)^{\frac{2}{3}-\delta_2}$$

$$\cdot \left(\frac{54000}{\frac{2}{3}}\right)^{\frac{2}{3}} \cdot \left(\frac{2}{3}\right)^{\frac{2}{3}} \quad \delta_2 \in \left\langle 0, \frac{1}{3} \right\rangle .$$

Funkcja przeciwna do logarytmu z $V(\delta_2)$ wynosi

$$v(\delta_2) = \left(\frac{1}{3}-\delta_2\right) \cdot \ln\left(\frac{1}{12} - \frac{\delta_2}{4}\right) + 2 \cdot \delta_2 \cdot \ln\left(\frac{\delta_2}{2}\right)$$

$$+ \left(\frac{2}{3}-\delta_2\right) \cdot \ln\left(\frac{2}{3}-\delta_2\right) - \frac{2}{3} \cdot \ln(54000)\,,$$

i osiąga minimum w punkcie, w którym jej pochodna

$$v'(\delta_2) = 2 \cdot \ln(3) - \ln(1-3\cdot\delta_2) + 2 \cdot \ln(\delta_2) - \ln(2-3\cdot\delta_2)$$

wynosi 0:

$$\ln\left(\frac{9\cdot\delta_2^2}{(1-3\cdot\delta_2)\cdot(2-3\cdot\delta_2)}\right) = 0\,,$$

$$\frac{9\cdot\delta_2^2}{(1-3\cdot\delta_2)\cdot(2-3\cdot\delta_2)} = 1\,,$$

$$9\cdot\delta_2^2 = 2 - 9\cdot\delta_2 + 9\cdot\delta_2^2\,,$$

$$\delta_2 = \frac{2}{9}\,,$$

czyli:

$$\delta_1 = \tfrac{1}{9}\,,$$

$$\delta_2 = \tfrac{2}{9}\,,$$

$$\delta_3 = \tfrac{2}{9}\,,$$

$$\delta_4 = \tfrac{4}{9}\,,$$

$$\delta_5 = \tfrac{2}{3}\,.$$

Jak łatwo policzyć wartością maksymalna $V(\delta)$ i równocześnie minimalną $g(x)$ jest 8100

Pozostaje zbadać dla jakich długości boków ta wartość zostaje osiągnięta. W tym celu rozwiążemy układ równań:

$$\begin{aligned} 4 \cdot x_1^2 &= \tfrac{1}{9} \cdot 8100\,, \\ 2 \cdot x_1 \cdot x_2 &= \tfrac{2}{9} \cdot 8100\,, \\ 2 \cdot x_1 \cdot x_3 &= \tfrac{1}{9} \cdot 8100\,, \\ x_2 \cdot x_3 &= \tfrac{4}{9} \cdot 8100\,, \\ \tfrac{54000}{x_1 \cdot x_2 \cdot x_3} &= 1\,, \end{aligned}$$

który po elementarnych przekształceniach daje odpowiedź

$$\begin{aligned} x_1 &= 15\,, \\ x_2 &= 60\,, \\ x_3 &= 60\,, \end{aligned}$$

lub uwzględniając oznaczenia z rysunku

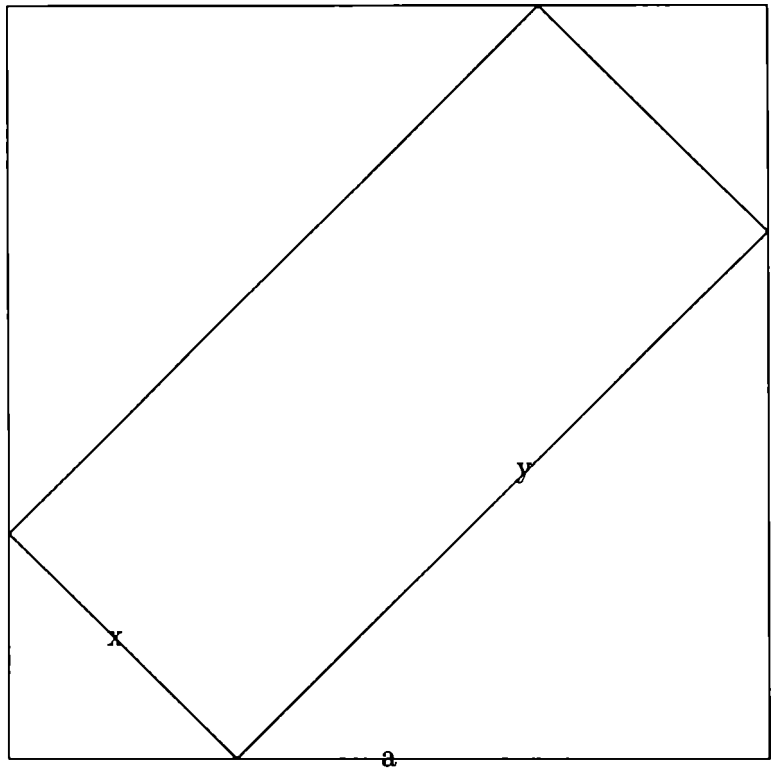
$$\begin{aligned} x &= 15\,dm.\,, \\ y &= 60\,dm.\,, \\ z &= 60\,dm.\,. \end{aligned}$$

Sprawdzenie, że koszt wyprodukowania takiego naczynia wyniesie 662 zł. jest już prostym ćwiczeniem arytmetycznym.

**Przykład 21**

Z kawałka blachy w kształcie kwadratu o boku długości a wycinamy prostokąty w ten sposób, że do każdego boku kwadratu należy tylko jeden wierzchołek prostokąta, zaś każdy z boków prostokąta jest równoległy do

przekątnej kwadratu. Jakie muszą być wymiary prostokąta aby straty materiału poniesione przy wycinaniu były minimalne ?




Zgodnie z oznaczeniami na rysunku zadanie da się sformułować w postaci maksymalizacji funkcji

$$f(x,y) = x \cdot y$$

przy założeniu

$$x + y = a\sqrt{2}\ ,$$

lub równoważnie w postaci minimalizacji funkcji $g(x,y)$, $x, y \geq 0$:

$$g(x,y) = \frac{1}{x \cdot y}\ ,$$

przy ograniczeniu

$$\frac{x\sqrt{2}}{2\cdot a}+\frac{y\sqrt{2}}{2\cdot a}\leq 1 \quad ,x,y\geq 0\ ,$$

Problem ten jako zadanie programowania geometrycznego ma swą formę dualną:

**zmaksymalizować funkcję:**

$$V(\delta)=\left(\frac{1}{\delta_1}\right)^{\delta_1}\cdot\left(\frac{\sqrt{2}}{2\cdot a\cdot\delta_2}\right)^{\delta_2}\cdot\left(\frac{\sqrt{2}}{2\cdot a\cdot\delta_3}\right)^{\delta_3}\cdot(\delta_2+\delta_3)^{\delta_2+\delta_3},$$

gdzie $\delta_i \quad \delta_i>0 \quad i\in\{1,2,3\}$ spełniają równania

$$\begin{array}{llll} \delta_1 & & & =1,\\ -\delta_1 & +\delta_2 & & =0,\\ -\delta_1 & & +\delta_3 & =0, \end{array}$$

Czyli:

$$\begin{array}{ll} \delta_1 & =1,\\ \delta_2 & =1,\\ \delta_3 & =1. \end{array}$$

Maksymalną wartością $V(\delta)$ i równocześnie minimalną wartością $g(x,y)$ jest

$$\frac{2}{a^2}.$$

i jest ona osiągana przy:

$$\begin{array}{ll} \frac{x\sqrt{2}}{2\cdot a} & =\frac{1}{1+1},\\ \frac{y\sqrt{2}}{2\cdot a} & =\frac{1}{1+1}, \end{array}$$

tj.

$$\begin{array}{ll} x & =\frac{a\sqrt{2}}{2},\\ y & =\frac{a\sqrt{2}}{2}., \end{array}$$

Szukanym prostokątem jest więc kwadrat.

**Przykład 22**

Naczynie w kształcie stożka ma mieścić V litrów wody. Trzeba zaprojektować je tak, aby koszty wykonania były jak najmniejsze przy jednakowej cenie surowca zarówno na podstawę naczynia jak i na jego pole boczne.

Niech r oznacza promień podstawy stożka, h jego wysokość a l - tworzącą. Pole powierzchni całkowitej i objętość stożka wynoszą odpowiednio:

$$V = \frac{1}{3} \cdot \pi \cdot r^2 \cdot h \ ,$$

$$P = \pi \cdot r^2 + \pi \cdot r \cdot l \ .$$

Ponadto między wysokością,promieniem a tworzącą stożka zachodzi związek:

$$h^2 + r^2 = l^2 \ .$$

Możemy sformułować program pierwotny minimalizacji funkcji

$$f(h, r, l) = \pi \cdot r^2 + \pi \cdot r \cdot l \ , h, r, l > 0 \ ,$$

przy ograniczeniach:

$$\frac{h^2}{l^2} + \frac{r^2}{l^2} \leq 1 \ ,$$

$$\frac{3 \cdot V}{\pi \cdot r^2 \cdot h} \leq 1 \ , r, h, l > 0 \ ,$$

i dualny maksymalizacji funkcji

$$V(\delta) = \left(\frac{\pi}{\delta_1}\right)^{\delta_1} \cdot \left(\frac{\pi}{\delta_2}\right)^{\delta_2} \cdot \left(\frac{1}{\delta_3}\right)^{\delta_3} \cdot \left(\frac{1}{\delta_4}\right)^{\delta_4}$$

$$\cdot (\delta_3 + \delta_4)^{\delta_3 + \delta_4} \cdot \left(\frac{3 \cdot V}{\pi \delta_5}\right)^{\delta_5} \cdot (\delta_5)^{\delta_5},$$

gdzie $\delta_i$ $(\delta_i > 0, \quad i \in \{1, 2, 3, 4, 5\})$ spełniają

$$\begin{array}{llll} \delta_1 & +\delta_2 & & = 1, \\ 2 \cdot \delta_1 & +2 \cdot \delta_2 \ +2 \cdot \delta_4 & -2 \cdot \delta_5 & = 0, \\ \delta_2 & -2 \cdot \delta_3 \ -2 \cdot \delta_4 & & = 0, \\ & 2 \cdot \delta_3 & -\delta_5 & = 0. \end{array}$$

W programie pierwotnym występują trzy zmienne przy łącznej liczbie pozymianów równej 5. Stopień złożoności tego zadania wynosi więc $DOD = 5 - 3 - 1 = 1 > 0$. W celu obliczenia $\delta_i \quad i \in \{1,2,3,4,5\}$ musimy więc przyrównać do 0 pochodne cząstkowe z funkcji przeciwnej do logarytmu z $V(\delta)$ analogicznie jak w przykładzie ze strony [88]. Po wykonaniu odpowiednich obliczeń otrzymujemy:

$$\begin{aligned}
\delta_1 &= \tfrac{1}{4}, \\
\delta_2 &= \tfrac{3}{4}, \\
\delta_3 &= \tfrac{1}{3}, \\
\delta_4 &= \tfrac{1}{24}, \\
\delta_5 &= \tfrac{2}{3}.
\end{aligned}$$

Wysokość i promień szukanego stożka obliczymy z zależności:

$$\begin{aligned}
\tfrac{h^2}{l^2} &= \tfrac{\frac{1}{3}}{\frac{1}{3}+\frac{1}{24}}, \\
\tfrac{h^2}{l^2} &= \tfrac{\frac{1}{24}}{\frac{1}{3}+\frac{1}{24}}, \\
h &= \tfrac{2\cdot\sqrt{2}}{3}\cdot l, \\
r &= \tfrac{1}{3}\cdot l, \\
h &= 2\cdot\sqrt{2}\cdot\sqrt[3]{\tfrac{3\cdot\sqrt{2}}{4\cdot\pi}\cdot V}, \\
r &= \sqrt[3]{\tfrac{3\cdot\sqrt{2}}{4\cdot\pi}\cdot V}.
\end{aligned}$$

**Przykład 23**

Na produkcję naczynia w kształcie walca przeznaczone jest P $m^2$ blachy. Należy zaprojektować to naczynie tak, aby miało jak największą objętość.

Niech r oznacza promień podstawy walca, a h jego wysokość. Objętość i pole powierzchni całkowitej walca wynoszą odpowiednio

$$V = \pi \cdot r^2 \cdot h,$$

$$P = 2 \cdot \pi \cdot r^2 + 2 \cdot \pi \cdot r \cdot h.$$

Możemy sformułować program pierwotny minimalizacji funkcji

$$f(h,r) = \frac{1}{\pi \cdot r^2 \cdot h}$$

przy ograniczeniach

$$\frac{2 \cdot \pi \cdot r^2}{P} + \frac{2 \cdot \pi \cdot r \cdot h}{P} \leq 1 \ , r, h > 0 \ ,$$

i dualny maksymalizacji funkcji

$$V(\delta) = \left(\frac{1}{\pi \cdot \delta_1}\right)^{\delta_1} \cdot \left(\frac{2 \cdot \pi}{P \cdot \delta_2}\right)^{\delta_2} \cdot \left(\frac{2 \cdot \pi}{P \cdot \delta_3}\right)^{\delta_3} \cdot (\delta_2 + \delta_3)^{\delta_2 + \delta_3},$$

gdzie $\delta_i \quad (\delta_i > 0, \quad i \in \{1, 2, 3\})$ spełniają związki

$$\begin{array}{llll} \delta_1 & & & = 1, \\ -2 \cdot \delta_1 & +2 \cdot \delta_2 & +\delta_3 & = 0, \\ -\delta_1 & & +\delta_3 & = 0. \end{array}$$

czyli

$$\begin{array}{ll} \delta_1 & = 1, \\ \delta_2 & = \frac{1}{2}, \\ \delta_3 & = 1. \end{array}$$

Promień i wysokość walca o największej objętości obliczymy z zależności

$$\begin{array}{ll} \frac{2 \cdot \pi \cdot r^2}{P} & = \frac{\frac{1}{2}}{\frac{1}{2}+1}, \\ \frac{2 \cdot \pi \cdot r \cdot h}{P} & = \frac{1}{\frac{1}{2}+1}, \end{array}$$

$$\begin{array}{ll} r & = \sqrt{\frac{P}{6 \cdot \pi}}, \\ h & = 2 \cdot \sqrt{\frac{P}{6 \cdot \pi}}. \end{array}$$

### 3.2.2 Minimalizacja funkcji kosztów przy nieliniowych ograniczeniach.

Programowanie geometryczne może służyć również jako narzędzie umożliwiające optymalizację nieliniowych funkcjii kosztów, również w przypadku pewnych funkcji dla których obliczenie dokładnej wartości punktu minimalizującego metodami tradycyjnego rachunku różniczkowego jest niemożliwe ze względu na trudności rachunkowe.

**Przykład 24**

Ustalono, że dobrym przybliżeniem funkcji kosztów pewnego przedsiębiorstwie jest:

$$K(x_1, x_2, x_3) = 1000 \cdot x1 + 1300\sqrt[4]{x_2^3 \cdot x_3} + 15000$$

gdzie:
$x_1$ - miesięczny nakład pracy
$x_2$ - miesięczny nakład surowca
$x_3$ - miesięczne koszty ziemi

Natomiast funkcja produkcji dla tego przedsiębiorstwa wyraża się wzorem

$$P(x_1, x_2, x_3) = \frac{5}{2} \cdot \left( 2\sqrt{x_1 \cdot x_3} \cdot \sqrt[3]{x_2}^2 + \frac{3}{\sqrt[3]{x_1 \cdot x_2^2 \cdot x_3}} \right) * 450$$

Maksymalne rozmiary zbytu wytwarzanych produktów to 2500 szt/miesiąc
Proces technologiczny w tym przedsiębiorstwie jest elastyczny i umożliwia substytucję nakładów.
Określ wielkości nakładu pracy, surowca i kosztów ziemi minimalizujące koszty w tym przedsiębiorstwie.

Próba zminimalizowania K(x) przy pomocy funkcji Lagrange'a

$$L(x, \lambda) = 1000 \cdot x_1 + 1300 \cdot x_2^{\frac{3}{4}} x_3 \frac{1}{4} - \lambda \cdot \left( x_1^{\frac{1}{2}} \cdot x_2^{\frac{2}{3}} \cdot x_3^{\frac{1}{2}} + x_1^{-\frac{1}{3}} \cdot x_2^{-\frac{2}{3}} \cdot x_3^{-1\frac{1}{3}} - \frac{20}{9} \right)$$

prowadzi do układu równań:

$$\begin{aligned}
\frac{\partial L}{\partial x_1} &= 1000 - \lambda \cdot \left( \frac{1}{\sqrt{x_1}} \cdot x_2^{\frac{2}{3}} \cdot x_3 - \frac{1}{x_1^{\frac{4}{3}} \cdot x_2^{\frac{2}{3}} \cdot x_1^{\frac{1}{3}}} \right) &= 0 \\
\frac{\partial L}{\partial x_2} &= \frac{975}{\sqrt[4]{x_2}} \cdot \sqrt[4]{x_3} - \lambda \cdot \left( \frac{4}{3} \cdot \frac{\sqrt{x_1}}{\sqrt[3]{x_2}} \cdot \sqrt{x_3} - \frac{2}{x_1^{\frac{1}{3}} \cdot x_2^{\frac{5}{3}} \cdot x_1^{\frac{1}{3}}} \right) &= 0 \\
\frac{\partial L}{\partial x_3} &= 325 \cdot \frac{x_2^{\frac{3}{4}}}{x_3^{\frac{3}{4}}} - \lambda \cdot \left( \sqrt{x_1} \cdot \frac{x_2^{\frac{2}{3}}}{\sqrt{x_3}} - \frac{1}{x_1^{\frac{1}{3}} \cdot x_2^{\frac{2}{3}} \cdot x_1^{\frac{4}{3}}} \right) &= 0 \\
\frac{\partial L}{\partial \lambda} &= -2 \cdot \sqrt{x1 \cdot x3} \cdot x_2^{\frac{2}{3}} - \frac{3}{x_1^{\frac{1}{3}} \cdot x_2^{\frac{2}{3}} \cdot x_1^{\frac{1}{3}}} + \frac{20}{9} &= 0
\end{aligned},$$

dla którego trudności rachunkowe nie pozwalają znaleźć dokładnego rozwiązanie a jedynie rozwiązania przybliżone.

Natomiast formułując to zadanie w języku programowania geometrycznego otrzymujemy problem:

**zmaksymalizować**

$$K(x_1, x_2, x_3) = 1000 \cdot x1 + 1300\sqrt[4]{x_2^3 \cdot x_3}$$

(stałą 15000 jako nie mającą wpływu na zmienne minimalizujące koszty możemy odrzucić)

Przy ograniczeniu

$$\frac{9}{20} \cdot 2\sqrt{x_1 \cdot x_3} \cdot \sqrt[3]{x_2}^2 + \frac{9}{20} \cdot \frac{3}{\sqrt[3]{x_1 \cdot x_2^2 \cdot x_3}} \leq 1 \ , \ x_i > 0 \ , \ i \in \{1, 2, 3\}$$

Oraz odpowiadający mu problem dualny:

$$V(\delta) = \left(\frac{1000}{\delta_1}\right)^{\delta_1} \cdot \left(\frac{1300}{\delta_2}\right)^{\delta_2} \cdot \left(\frac{9 \cdot 2}{20 \cdot \delta_3}\right)^{\delta_3} \cdot \left(\frac{9 \cdot 3}{20 \cdot \delta_4}\right)^{\delta_4}$$

$$\cdot(\delta_3 + \delta_4)^{\delta_3 + \delta_4}$$

*Gdzie* $\delta_i \quad \delta_i > 0 \quad i \in \{1, 2, 3, 4\}$ *spełniają:*

$$\begin{array}{llll} \delta_1 & +\delta_2 & & = 1 \\ \delta_1 & +\frac{1}{2} \cdot \delta_3 & -\frac{1}{3} \cdot \delta_4 & = 0 \\ \frac{3}{4} \cdot \delta_1 & +\frac{2}{3} \cdot \delta_3 & -\frac{2}{3} \cdot \delta_4 & = 0 \\ \frac{1}{4} \cdot \delta_1 & +\frac{1}{2} \cdot \delta_3 & -\frac{1}{3} \cdot \delta_4 & = 0 \end{array}$$

Czyli

$$\begin{array}{ll} \delta_1 & = \frac{1}{5} \\ \delta_2 & = \frac{4}{5} \\ \delta_3 & = \frac{3}{5} \\ \delta_4 & = \frac{3}{2}, \end{array}$$

Minimalną wartością kosztów jest więc:

$$M = \left(\frac{1000}{0,2}\right)^{0,2} \cdot \left(\frac{1300}{0,8}\right)^{0,8} \cdot \left(\frac{3}{2}\right)^{0,6}$$

$$\cdot \left(\frac{9}{10}\right)^{1,5} \cdot (2,1)^{2,1} \approx 10523,47887$$

Natomiast nakłady pracy surowca i wydatków na ziemię minimalizujące koszty wynoszą:

$$\begin{aligned}
1000 \cdot x_1 &= M \cdot 0,2 \\
1300 \cdot x_2^{\frac{3}{4}} \cdot x_3^{\frac{1}{4}} &= M \cdot 0,8 \\
\tfrac{9}{10} \cdot \sqrt{x_1 \cdot x_3} \cdot \sqrt[3]{x_2^2} &= \tfrac{0,6}{0,6+1,5} \\
\\
x_1 &= \tfrac{M}{5000} \\
x_2 &= e^{\frac{12 \cdot \left(\ln \frac{0,8 \cdot M}{1300} - 2 \cdot \ln \frac{20}{63} \cdot \sqrt{\frac{5000}{M}}\right)}{5}} \\
x_3 &= e^{\frac{32}{5} \cdot \ln \frac{0,8 \cdot M}{1300} - \frac{6}{5} \cdot \ln \frac{20}{63} \cdot \sqrt{\frac{5000}{M}}}
\end{aligned}$$

# Rozdział 4

# Rozszerzenia programowania geometrycznego

Istotnym ograniczeniem programowania geometrycznego jest fakt, iż ta technika, mimo swojej niewątpliwej elegancji, dotyczy tylko funkcji składających się z pozymianów. Wydaje się, że jednak, że założenie o nieujemności współczynników przy tkwi tak głęboko w podstawach matematycznych programowania geometrycznego nie da się przeprowadzić uogólnienia p.g. na w funkcje składające się z dowolnych wielomianów. Niemniej autorzy prac związanych z programowaniem geometrycznych ([30], [3]) starali się rozszerzyć je na jak największą klasę funkcji.

Już w [30] str. 97-101 zaproponowane zostało uogólnienie programowania geometrycznego na funkcje postaci

$$G(t) = \sum_i \prod_j \frac{[q_{ij}(t)]^{\alpha_{ij}}}{[1 - p_{ij}(t)]^{\beta_{ij}}},$$

To uogólnienie przedstawione jest w pierwszej części bieżącego rozdziału.

Druga część tego rozdziału zawiera propozycję powiązania metod programowania geometrycznego z rozwinięciem funkcji w szereg Taylora / MacLaurina. Jest to propozycja autora niniejszej pracy, zastosowanie jej do wybranych problemów dało całkiem obiecujące rezultaty.

Wydaje się, że całościowe potraktowanie tego problemu wykraczałoby poza zakres tematyczny niniejszej pracy, ograniczyłem się więc do przedstawienia przykładu takiego rozwinięcia [1] oraz listy najważniejszych funkcji

[1] Jednego z kilkunastu obliczonych przy pomocy programu PG.EXE - w większości z

o „pozymianowym" szeregu Taylora, pozostawiając do dalszych badań matematyczno / informatyczną analizę złożoności obliczeniowej i dokładności algorytmów wykorzystujących to rozwinięcie

## 4.1 Generalizacja programowania geometrycznego

Programowania geometryczne da się rozszerzyć na następującą grupę zagadnień
A. Nierówność

$$G(tf(t) + [q(t)]^2 h(t)a > 0$$

jest równoważna układowi zależności

$$g(\tau) = f(t) + t_0^a h(t)\,,$$

$$t_0^{-1} q(t) \leq 1\,.$$

B. Funkcja

$$G(t) = f(t) + \frac{q(t)}{[u(t) - h(t)]^\alpha}\,,$$

gdzie $u(t)$ ma jeden składnik jest równoważna układowi

$$g(\tau) = f(t) + \frac{q(t)}{t_0^\alpha}\,,$$

$$\frac{t_0}{u(t)} + \frac{h(t)}{u(t)} \leq 1\,.$$

C.
Niech

$$G(t) = f(t) - u(t)$$

gdzie $u(t)$ ma jeden składnik.

---

nich otrzymałem wyniki o tej podobnej dokładności już przy wykorzystaniu 6-8 pierwszych wyrazów rozwinięcia MacLaurina

Załóżmy, że minimalna wartość $G(\tau)$ jest mniejsza od 0 i niech

$$\tau' = [u(t') - f(t'), t'_1, t'_2, \ldots, t'_m] .$$

G(t) jest minimalna gdy

$$h(\tau) = t_0$$

jest maksymalna, przy założeniach

$$t_0 + f(t) - u(t) \leq 0 .$$

Problem da się sformułować jako minimalizacja funkcji

$$g(\tau) = \frac{1}{h(\tau)} = \frac{1}{t_0}$$

przy ograniczeniach

$$\frac{t_0}{u(t)} + \frac{f(t)}{u(t)} \leq 1 .$$

Ogólnie generalizacja programowania geometrycznego, to konstrukcje funkcji

$$G(t) = \sum_i \prod_j \frac{[q_{ij}(t)]^{\alpha_{ij}}}{[1 - p_{ij}(t)]^{\beta_{ij}}} ,$$

gdzie
$q_{ij}, p_{ij}$ - pozymiany,
$\alpha_{ij}, \beta_{ij}$ - dodatnie,
$1 - p_{ij} > 0$.

## 4.2 Programowanie geometryczne a szereg Taylora

Metody programowania geometrycznego mogą być wykorzystane przy optymalizacji funkcji, dla których szereg Taylora [2] (lub dostatecznie dużo pierwszych wyrazów szeregu) składa się z pozymianów. Ilustracją optymalizacji takiego rodzaju może być przykład 25.

**Przykład 25**

Zminimalizować funkcję:

$$f(x,y) = e^x y^3 + \frac{1}{y} + \frac{1}{x^2} \quad x, y > 0 \ ,$$

Przy ograniczeniu:

$$x^2 + y^2 \leq 1$$

Korzystając z rozwinięcia Taylora/ MacLaurina

$$e^x = \cdot \left(1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \frac{x^6}{6!} + \frac{x^7}{7!} + \frac{x^8}{8!} + O(\Theta)\right) \ ,$$

Otrzymujemy problem przybliżony:
Zminimalizuj funkcję:

$$f(x,y) \cong y^3 \cdot \left(1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + \frac{x^6}{720} + \frac{x^7}{5040} + \frac{x^8}{40320}\right) + \frac{1}{y} + \frac{1}{x^2} \ ,$$

przy ograniczeniu:

$$x^2 + y^2 \leq 1 \ ,$$

[2] [64] str. 223

czyli zagadnienie programowania geometrycznego o 2 zmiennych i 13 pozymianach czyli o stopniu złożoności DOD=13-2-1=10 posiadającemu zadanie dualne polegające na maksymalizacji funkcji

$$V(\delta) = \left(\frac{1}{\delta_1}\right)^{\delta_1} \cdot \left(\frac{1}{\delta_2}\right)^{\delta_2} \cdot \left(\frac{1}{2 \cdot \delta_3}\right)^{\delta_3} \cdot \left(\frac{1}{6 \cdot \delta_4}\right)^{\delta_4} \cdot \left(\frac{1}{24 \cdot \delta_5}\right)^{\delta_5} \cdot \left(\frac{1}{120 \cdot \delta_6}\right)^{\delta_6}$$
$$\cdot \left(\frac{1}{720 \cdot \delta_7}\right)^{\delta_7} \cdot \left(\frac{1}{5040 \cdot \delta_8}\right)^{\delta_8} \cdot \left(\frac{1}{40320 \cdot \delta_9}\right)^{\delta_9}$$
$$\cdot \left(\frac{1}{\delta_{10}}\right)^{\delta_{10}} \cdot \left(\frac{1}{\delta_{11}}\right)^{\delta_{11}} \cdot \left(\frac{1}{\delta_{12}}\right)^{\delta_{12}} \cdot \left(\frac{1}{\delta_{13}}\right)^{\delta_{13}} \cdot (\delta_{12} + \delta_{13})^{\delta_{12}+\delta_{13}}$$

przy ograniczeniach:

$$\begin{array}{lllllllllll}
\delta_1 & +\delta_2 & +\delta_3 & +\delta_4 & +\delta_5 & +\delta_6 & +\delta_7 & +\delta_8 & +\delta_9 & +\delta_{10} & +\delta_{11} & = 1 \\
3\delta_1 & +3\delta_2 & +3\delta_3 & +3\delta_4 & +3\delta_5 & +3\delta_6 & +3\delta_7 & +3\delta_8 & +3\delta_9 & -\delta_{10} & +2\delta_{13} & = 0 \\
\delta_2 & +2\delta_3 & +3\delta_4 & +4\delta_5 & +5\delta_6 & +6\delta_7 & +7\delta_8 & +8\delta_9 & -2\delta_{11} & +\delta_{12} & & = 0
\end{array}$$

Optymalizując $V(\delta)$ przy pomocy programu PG stwierdzamy że maksymalną jej wartościa jest 3.624012 (p. 2.3.2)
Natomiast przybliżonymi wartościami $\delta$ , $x$ i $y$ są

$$\begin{array}{lcl}
\delta_1 & = & 0.039110 \\
\delta_2 & = & 0.033505 \\
\delta_3 & = & 0.014878 \\
\delta_4 & = & 0.005078 \\
\delta_5 & = & 0.001976 \\
\delta_6 & = & 0.001082 \\
\delta_7 & = & 0.000759 \\
\delta_8 & = & 0.000611 \\
\delta_9 & = & 0.000529 \\
\delta_{10} & = & 0.528631 \\
\delta_{11} & = & 0.376423 \\
\delta_{12} & = & 0.328361 \\
\delta_{13} & = & 0.121424
\end{array}$$

$$\begin{array}{lcl}
x & = & 0.8567 \\
y & = & 0.5208
\end{array}$$

Rozwiązując podobny problem za pomocą pakietu SOLVER z popularnego arkusza kalkulacyjnego MS Excel 97 otrzymany został wynik 3,62410.

Porównując to z otrzymanym przez nas rozwiązaniem stwierdzamy, że rząd przybliżenia to $10^{-5}$ co jest dopuszczalnym błędem w praktycznych zadaniach optymalizacyjnych.

## 4.3 Funkcje o „pozymianowym" szeregu Taylora

Istnieje cała grupa funkcji, dla których wszystkie lub dostatecznie dużo pierwszych wyrazów rozwinięcia Taylora/McLarina to pozymiany i dla których przybliżoną optymalizację można przeprowadzić przy pomocy metod programowania geometrycznego.
Oto niektóre z nich:

$$
\begin{aligned}
acos(-x) &= \tfrac{1}{2} \cdot x + +x + \tfrac{1}{6} \cdot x^3 + \tfrac{3}{40} \cdot x^5 + +O(\Theta) \\
-ln(1-x) &= x + \tfrac{1}{2} \cdot x^2 + \tfrac{1}{3} \cdot x^3 + \tfrac{1}{4} \cdot x^4 + \tfrac{1}{5} \cdot x^5 + O(\Theta) \\
-asin(-x) &= x + +\tfrac{1}{6} \cdot x^3 + \tfrac{3}{40} \cdot x^5 + \tfrac{5}{112} \cdot x^7 + O(\Theta) \\
asin(x) \cdot e^x &= x + x^2 + \tfrac{2}{3} \cdot x^3 + \tfrac{1}{3} \cdot x^4 + \tfrac{1}{5} \cdot x^5 + O(\Theta) \\
sin(x) \cdot e^x &= x + x^2 + \tfrac{1}{3} \cdot x^3 + \tfrac{1}{30} \cdot x^5 + O(\Theta) \\
\tfrac{1}{\sqrt{1-x}} &= 1 + +\tfrac{1}{2} \cdot x + \tfrac{3}{8} \cdot x^2 + \tfrac{5}{16} \cdot x^3 + \tfrac{35}{128} \cdot x^4 + \tfrac{63}{256} \cdot x^5 + O(\Theta) \\
\tfrac{1}{1-x} &= 1 + x + x^2 + x^3 + x^4 + x^5 + O(\Theta) \\
tan(x) &= x + \tfrac{1}{3} \cdot x^3 + \tfrac{2}{15} \cdot x^5 + O(\Theta) \\
\tfrac{1}{sin(x)} &= \tfrac{1}{x} + \tfrac{1}{6} \cdot x + \tfrac{7}{360} \cdot x^3 + O(\Theta) \\
\tfrac{1}{cos(x)} &= 1 + \tfrac{1}{2} \cdot x^2 + \tfrac{5}{24} \cdot x^4 + O(\Theta) \\
\tfrac{e^x}{sin(x)} &= \tfrac{1}{x} + 1 + \tfrac{2}{3} \cdot x + \tfrac{1}{3} \cdot x^2 + \tfrac{13}{90} \cdot x^3 + O(\Theta)
\end{aligned}
$$

$$
\begin{aligned}
-ln\left(\tfrac{1-x}{1+x}\right) &= 2\cdot x + \tfrac{2}{3}\cdot x^3 + \tfrac{2}{5}\cdot x^5 + \tfrac{2}{7}\cdot x^7 + O(\Theta) \\
ln\left(\tfrac{1}{cos(x)}\right) &= \tfrac{1}{2}\cdot x^2 + \tfrac{1}{12}\cdot x^4 + \tfrac{1}{45}\cdot x^6 + O(\Theta) \\
sinh(x) &= x + \tfrac{1}{3!}\cdot x^3 + \tfrac{1}{5!}\cdot x^5 + \tfrac{1}{7!}\cdot x^7 + O(\Theta) \\
cosh(x) &= 1 + + \tfrac{1}{2!}\cdot x^2 + \tfrac{1}{4!}\cdot x^4 + \tfrac{1}{6!}\cdot x^6 + O(\Theta) \\
atanh(x) &= x + + \tfrac{1}{3}\cdot x^3 + \tfrac{1}{5}\cdot x^5 + \tfrac{1}{7}\cdot x^7 + O(\Theta) \\
\tfrac{e^x}{\sin(x)\cdot y^2} + x\cdot\tan(y) &= \tfrac{1}{y^2\cdot x} + \tfrac{1}{y^2} + \tfrac{3\cdot x}{\cdot y^2} + \tfrac{x^2}{3\cdot y^2} + \tfrac{13\cdot x^3}{90\cdot y^2} + \tfrac{x^4}{18\cdot y^2} + \tfrac{19\cdot x^5}{945\cdot y^2} \\
&\quad + x\cdot y + \tfrac{x\cdot y^3}{3} + x\cdot y + \tfrac{2\cdot x\cdot y^5}{15} + \tfrac{17\cdot x\cdot y^7}{3} + O(\Theta)
\end{aligned}
$$

# Zakończenie

Chociaż programowanie matematyczne nie rozwija się już tak gwałtownie, jak w latach 50-tych czy 60-tych, to na pewno nie wszystko w nim zostało powiedziane. Niektóre dziedziny, mimo, iż stoją na uboczu głównego kierunku badań, kryją w sobie jeszcze wiele obszarów słabo lub w ogóle nie poznanych.

Jedną z takich dziedzin jest programowanie geometryczne, które wydaje się ze względu na swoją strukturę naturalnym rozszerzeniem programowania liniowego. Niestety, dość istotnym jego ograniczeniem jest fakt, iż wszyskie składniki funkcji minimalizowanej jak i funkcji ograniczających, muszą być dodatnie. Niemniej, nawet przy tym ograniczeniu, możemy z powodzeniem stosować go w wielu dziedzinach.

Niniejsza praca stanowi drobny przyczynek do rozwoju programowania geometrycznego.

Wyniki pracy można podzielić na dwie grupy:

Pierwsza to zaprezentowanie metod programowania geometrycznego w formie kompletnego wykładu od podstaw matematycznych poprzez zasadnicze twierdzenia, na których się ono opiera, aż do sformułowania teorii dualności programowania . Jest to pierwsze, wg stanu wiedzy autora, tego typu opracowanie w literaturze polskojęzycznej.

Druga grupa to indywidualny wkład autora w rozwój p.g., czyli m.in.
a) zwrócenie uwagi na korelację między programowaniem geometrycznym a funkcjami żyteczności Cobba-Douglasa;
b) zaproponowanie programowania geometrycznego jako narzędzia do optymalizacji funkcji kosztów przy złożonych funkcjach ograniczających;
c) przedstawienie programowania geometrycznego jako narzędzia pozwalającego na rozwiązanie wielu zadań maksymalizacji / minimalizacji bez odwoływania się do aparatu rachunku różniczkowego, a jedynie poprzez rozwiązanie układu równań liniowych.

Praktyczną częścią pracy jest również program PG.CPP napisany w języku C++, służący do numerycznego rozwiązywania zadań programowania geometrycznego.

Ciekawym problemem jest zasygnalizowane w rozdziale IV wykorzystanie programowania geometrycznego do optymalizacji funkcji, których dostatecznie wiele pierwszych wyrazów szeregu Taylora/MacLaurina jest pozymianami.

Temat ten, jako wykraczający poza granice niniejszej pracy wymaga osobnego opracowania ze strony matematyczno / informatycznej (np rozbudowy programu PG.CPP tak, aby automatycznie rozpoznawał funkcje o „pozymianowym" szeregu Taylora). Niemniej wyniki uzyskane przy przybliżonej optymalizacji wydają się obiecujące, a dość dobra dokładność rozwiązań pozwala przypuszczać, że może ona znaleźć zastosowanie praktyczne.

Podsumowując uważam, że programowanie geometryczne jest wygodną metodą szukania ekstremów funkcji. Jest często bardziej elastyczne w użyciu od tradycyjnych metod. Dobrze opisuje liczne zagadnienia związane z naukami ekonomicznymi, dla których odpowiednie funkcje po prostych przekształceniach można wyrazić przez pozymiany. Jest to więc technika, którą warto poznać i stosować w ilościowych problemach nauk ekonomicznych.

# Literatura


[1] Adamczyk Ludwik (i in.): „Metody ilościowe w ekonomii", Wrocław, AE, 1999.

[2] Arrow K.J., Hurwicz L.: „Reduction of Constrained Maxima to Saddle-point Problems", in Proceedings of the Third Berkeley Symposium on Probability, vol. V, J. Neyman (ed.), University of California Press, Berkeley, CA, 1956.

[3] Avriel M.: „Advances in Geometric Programming", Plenum Press New York, 1980.

[4] Azarm S., Duong T., Golshani D., Tuan T.: „Optimization via Geometric Programming", University of Maryland, 1995.

[5] Bahn O.,Goffin J.L., Vial J.: „Experimental behavior of an interior point cutting plane algorithm for convex programming: an aplication to a geometric programming", *Discrete Applied Mathematics*, 49 2-23, 1994.

[6] Bazaraa M.S., Shetty C.M. , Sherali: „Nonlinear Programming, Theory & Applications", Wiley & Sons, 1979/1994.

[7] Beck P.A.,Ecker J.G.: „A modified concave simplex algorithm for geometric programming", *Journal of Optimization Theory an Applications* 15, 1975.

[8] Begg D. „Ekonomia", Warszawa, 1993.

[9] Beightler Charles, Phillips D.T., Wilde D.J.: „Foundations of Optimization", Pretince-Hall, Englewood Cliffs, 1967.

[10] Beightler Charles S.: „Applied Geometric Programming", John Wiley & Sons, 1976.

[11] Bertsekas, Dimitri P.: „Dynamic Programming and Optimal Control", Belmont, Athena Scientific, 1995.

[12] Bertsekas, Dimitri P.: „Nonlinear Programming", Belmont, Athena Scientific, 1995.

[13] Błażewicz Jacek, Cellary Wojciech, Słowiński Roman, Węglarz Jan: „Badania Operacyjne Dla Informatyków", Wydawnictwa Naukowo - Techniczne, 1983.

[14] Borchers, Mitchell: „An Improved Branch and Bound Algorithm for Mixed Integer Nonlinear Programs", *Computers and Operations Research,* Vol 21/4 1994.

[15] Charnes A., Cooper W.W.K., Kortanek K.O.: „Duality in Semi-Infinite Programs and Some Works of Haar and Caratheodory", *Management Science 9:2*, 1993.

[16] Chibani L.: „Optimum Design of Structures: With Special Reference to Alternative Loads Using Geometric Programming", *Lecture Notes in Engineering*, Vol 50 1989.

[17] Coleman, Li: „Large Scale Numerical Optimization", SIAM Books.

[18] Conn A.R.: „LANCELOT: a Fortran Package for Large-Scale Nonlinear Optimization", *Springer Series in Computational Mathematics*, vol. 17, 1992.

[19] Czechowski Tadeusz: „Wprowadzenie do Zastosowań Matematyki w Ekonomii", Państwowe Wydawnictwo Naukowe 1973.

[20] Dantzig G.B., Cottle R.W., Eisenberg E.: „Symetric Dual Nonlinear Programs" ,*Pacific Journal of Mathematics 15*, 1965.

[21] Dennis J.B.: „Mathematical Programming and Electrical Networks", Wiley & Sons, New York, 1959.

[22] Dennis J.B., Schnabel: „Numerical Methods for Unconstrained Optimization and Nonlinear Equations", Prentice Hall, 1983.

[23] Dembo R.S: ,,A Set of Geometric Programming Test Problems and Their Solutions", *Mathematical Programming* 10,192-213, 1976.

[24] Dembo R.S: ,,Dual to primal conversion in geometric programming", *Mathematical Programming* 26,243-252, 1976.

[25] Dieter G.: ,,Engineering Desig", Mc-Graww-Hill Inc., 1983.

[26] Dorn W.S.: ,,Duality in Quadratic Programming", *Quarterly of Applied Mathematics* 18:2, 1960.

[27] Dorn W.S.: ,,Self-dual Quadratic Programs", *SIAM Journal on Applied Mathematics* 9:1, 1961.

[28] Du, Sun: ,,Advances in Optimization and Approximation", Kluwer, 1994.

[29] Duffin R.J., Peterson E.L.: ,,Duality Theory for Geometric Programming", *SIAM Journal on Applied Mathematics* 14:6, 1966.

[30] Duffin R.J., Peterson E.L., Zener C.: ,,Geometric Programming", John-Willey & Sons Inc. New York 1967.

[31] Eggleston: ,,Convexity", Cambridge at the University Press 1958.

[32] El-Bakry A.S., R.A. Tapia, T Tsuchiya, Zhang Y.: ,,On the Formulation and Theory of the Primal-Dual Newton Interior Point Method for Nonlinear Programming", Department of Computional &Applied Mathematics, Rice University, Houston, 1994.

[33] Fenchel W.: ,,Convex Cones, Sets and Functions", Mathematics Department, Princeton University, Princeton, 1951.

[34] Fiacco, McCormick: ,,Sequential Unconstrained Minimalization Techniques", SIAM Books.

[35] Fletcher, R.: ,,Practical Methods of Optimization", Wiley, 1987.

[36] Floudas, Pardalos: ,,Recent Advances in Global Optimization", Princeton University Press, 1992.

[37] Gale D., Kuhn H.W., Tucker A.W.: „Linear Programming and the Theory of Games, in Activity Analysis of Production and Allocation", T.C. Koopmans (ed.), Wiley, New York, 1951.

[38] Greenberg H.J.: „The Generalized Penalty-Function/Surrogate Model", *Operations Research 21*, 1973.

[39] Greenberg H.J., Pierskalla W.P.: „Quasi-conjugate Functions and Surrogate Duality" ,Cahiers Centre studes de Rech. Oper. 15, 1973.

[40] Gill, Murray, Wright: „Practical Optimization", Academic Press, 1981.

[41] Goldstein E.G.: „Theory of Convex Programming", American Mathematical Soc., Providence 1972.

[42] Grotschel Martin: „Geometric Algorithms and Combinatorial Optimization", *Algorithms and Combinatorics, 2*, 1993.

[43] Hiller Frederick S., Lieberman Gerald J.: „Introduction to Operations Research", V wydanie, McGraw-Hill Publishing Company, New York 1990.

[44] Himmelblau: „Applied Nonlinear Programming", McGraw-Hill, 1972.

[45] Hock, Schittkowski: „Test Examples for Nonlinear Programming Codes", Springer-Verlag, 1981.

[46] Hooke, Jeeves: „Direct Search Solution of Numerical and Statistical Problems", *Journal of the ACM*, Vol.8, 212-229, 1961.

[47] Hooker J.N.: „Inference Duality as a Basis for Sensitivity Analysis, Principles and Practice of Constraint Programming", *Lecture Notes in Computer Science 1118*, Springer, Berlin, 1996.
(dostępne pod adresem
http://www.gsia.cmu.edu/afs/andrew/gsia/jh38/papers.html).

[48] Horst, Tuy: „Global Optimization", Springer-Verlag, 1993.

[49] Horst, Pardalos: „Handbook of Global Optimization", Kluwer, 1995.

[50] Horst, Pardalos, Thoai: „Introduction to Global Optimization", Kluwer, 1995.

[51] Jeroslow R.G.: „Cutting Plane Theory: Algebraic Methods", *Discrete Mathematics 23*,1978.

[52] Jeroslow R.G.: „Minimal Inequalities", *Mathematical Programming* 17, 1979.

[53] Johnson E.L.: „Cyclic Groups, Cutting Planes and Shortest Paths in Mathematical Programming", Academic Press, New York, 1973.

[54] Johnson E.L.: „On the Group Problem for Mixed Integer Programming", Mathematical Programming Study 2, 1974.

[55] Kahaner, Moler, Nash: „Numerical Methods and Software", Prentice - Hall.

[56] Kamerschen D.R.: „Ekonomia", Fundacja Gospodarcza NSZZ „Solidarność", 1991.

[57] Kerningham B.W., Ritchie D.M: „Język C.", WNT, Warszawa, 1990.

[58] Klimczak Bożena: „Mikroekonomia", Wydawnictwo Akademii Ekonomicznej we Wrocławiu, 1992.

[59] Kojima M. Noma T. Yoshise A.: „Global Convergence in Infeasible Interior Points Algoritms", *Mathematical Programming* 65, 1994

[60] Kortanek K.O., Xiaxojie Xu, Yinyu Ye: „An Infeasible Interior Point Algorithm for Solving Primal and Dual Geometric Programs", The University of Iowa 1994.

[61] Lau H.T.: „A Numerical Library in C for Scientists and Engineers", CRC Press, 1994.

[62] Luenberger: „Introduction to Linear and Nonlinear Programming", Addison Wesley, 1984.

[63] Mangasarian O.L.: „Nonlinear Programming", McGraw-Hill, New York, 1969.

[64] Montygierd - Łoyba Michał: „Podstawy Matematyki dla ekonomistów", Wydawnictwo A.E. we Wrocławiu, 1995.

[65] More, Noye, Fletcher: „Numerical Solution of Bound Constrained Problems", *Computational Techniques, Applications*, 29-37, 1988.

[66] More, Toraldo: „Algorithms for Bound Constrained Quadratic Programming" *Problems, Numerische Mathematik 55, 377-400*, 1989.

[67] More, Wright: „Optimization Software Guide", SIAM, 1993.

[68] Nash S., Sofer A.: „Linear and Nonlinear Programming", McGraw-Hill, 1996.

[69] Nemhauser, Rinnooy, Kan, Todds: „Optimization", North-Holland, 1989.

[70] Nocedal J.„Summary of algorithms for unconstrained optimization", *Acta Numerica*, 1992".

[71] Passy U.: „A note on Modular Design", *Operations-Research* Vol 18, 562-564, 1970.

[72] Pardalos, Wolkowicz: „Quadratic Assignment and Related Problems", *American Mathematical Society, DIMACS series in discrete mathematics*, 1994.

[73] Powell, M.J.D.: „A Fast Algorithm for Nonlinearly Constrained Optimization Calculations", *Springer-Verlag Lecture Notes in Mathematics*, vol. 630.

[74] Press, Flannery, Teukolsky, Vetterling: „Numerical Recipes", Cambridge, 1986.

[75] Rockfellar R.T.: „Convex Analysis", Priceton University Press, 1970

[76] Schittkowski: „Nonlinear Programming Codes", Springer-Verlag, 1980.

[77] Schittkowski: „More Test Examples for Nonlinear Programming Codes", *Lecture Notes in Economics and Math. Systems* 282, Springer 1987.

[78] Stanisz Tadeusz: „Funkcje Jednej Zmiennej w Badaniach Ekonomicznych", PWN 1986.

[79] Stanisz Tadeusz: „Zastosowania Matematyki w Ekonomii", Kraków, 1993.

[80] Sysło M.M., Narshongh D., Kowalik J.: „Algorytmy optymalizacji dyskretnej", PWN

[81] Torn, Zilinskas: „Global Optimization", Springer-Verlag, 1989.

[82] Tuan T., Golshani D., Duong T., Azarm S.: „Optimization Via Geometric Programming", 1996.

[83] Varian H. R.: „Mikroekonomia", PWN Warszawa 1995.

[84] Welfe Aleksander, Zatoń Wojciech: „Wykorzystanie Metody Newtona do Symulacji Modeli Ekonometrycznych", *Przegląd Statystyczny 2/1993*;

[85] Wolfe P.: „A Duality Theorem for Nonlinear Programming", *Quarterly of Applied Mathematics* 19:3,1961.

[86] Wilde D.J.: „Globally Optimum Design", Willey-Interscience, New York, 1978

[87] Wismer, Chattergy: „Introduction to Nonlinear Optimization", 1978.

[88] Wolsey L.A.: „Integer Programming Duality: Price Functions and Sensitivity Analysis", *Mathematical Programming* 20:2,

[89] Wright M.: „Interior Methods For Constrained Optimization",*Acta Mathematica*, Cambridge University Press, 1992.

[90] Wright M.: „Direct Search Methods: Once Scorned, Now Respectable"

[91] Xiaxojie Xu: „An $\mathcal{O}\left(\sqrt{\backslash} \cdot \mathcal{L}\right)$-Iteration Step Infeasible Path - Following Algorithm for Solving Linear Programming", The University of Iowa 1994.

[92] Xiaxojie Xu,Yinyu Ye: „A Generalized Homogeneous and Self Dual Algorithm for Linear Programming", The University of Iowa 1994.

[93] Zener Clarence Melvin: „Engineering design by geometric programming"

# Aneks - Treść programu PG.CPP

```
#define TEST  //katalog z danymi

#include <afx.h>
#include <stdlib.h>
#include <setjmp.h>
#include <windows.h>
#include <commdlg.h>
#include <memory.h>
#include <string.h>
#include <stdio.h>

#include <math.h>
#include <newmat.h>

//#define maxliczbazmiennych 50 //
#define maxliczbafunkcji 50      //
//#define maxliczbawyrazow 500
        // Laczna we wszystkich funkcjach (terms number)

#define factor 0.9995                    // Alg 4
#define fac 100                          // Alg 5
#define duzo 100000000
#define MaxIter 100

FILE *wyniki;
```

```
CStdioFile dane;
CFileException e;

int m;
        // liczba_zmiennych;
int p;
        // liczba_funkcji;
int dod;
        // Degree of difficulty
int n[maxliczbafunkcji];
        // Kolejne wyrazy w funkcjach

int i,j,k;

ColumnVector c;

Matrix Alpha;
ColumnVector Lambda;

ColumnVector gradient;
Matrix hessian;

Matrix A;
ColumnVector x;
ColumnVector B;

//
Matrix lnAlpha;
ColumnVector lnX;
ColumnVector lnB;

ColumnVector y,z;
ColumnVector r_p,r_d;
ColumnVector delta_x,delta_y,delta_z;
double mi;
double sigma,beta;
double eta,gamma;
double thetac,thetap,thetad;
```

```
double wynik;

BOOL wczytaj_dane();
BOOL zapisz_wyniki();
double potega (double wsp, double pot);
void rozwiaz();

//double f();
void _Lambda();
void _gradient();
void _hessian();
void _mi();
Matrix diag(ColumnVector x);
double Norm2(Matrix M);
void znajdz_kierunek();

void main()
{
wczytaj_dane();
if (dod!=0)rozwiaz();
zapisz_wyniki();
};


BOOL wczytaj_dane()
{
int ilepoteg,ktorapotega;
float potega;
OPENFILENAME ofn;
char
szDirName[256];
char szFile[256];
UINT  cbString;
char  szFilter[256];

    GetSystemDirectory(szDirName, sizeof(szDirName));
    #ifdef TEST
    strcpy(szDirName,"c:\\andrzej\\geomet~1\\xgp\\");
```

```
#endif
szFile[0] = '\0';

strcpy(szFilter,"Pliki z danymi ");
strcat(szFilter,"*.dat ");
cbString=sizeof(szFilter);
szFilter[cbString-1]='\0';

memset(&ofn, 0, sizeof(OPENFILENAME));

ofn.lStructSize = sizeof(OPENFILENAME);
ofn.hwndOwner = NULL;
ofn.lpstrFilter = szFilter;
ofn.nFilterIndex = 1;
ofn.lpstrFile= "*.dat";
ofn.nMaxFile = sizeof(szFile);
ofn.lpstrFileTitle = NULL;
ofn.nMaxFileTitle = 0;
ofn.lpstrInitialDir = szDirName;
ofn.Flags = OFN_SHOWHELP | OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST;

if (GetOpenFileName(&ofn))
{
if( !dane.Open(ofn.lpstrFile,CFile::modeRead, &e ) )
{
#ifdef _DEBUG
afxDump << "File could not be opened "
                                << e.m_cause << "\n";
#endif
    return FALSE;
         }

    }
else
     return FALSE;
fscanf (dane.m_pStream,"%d\n",&m);
if (m==0) fscanf(dane.m_pStream,"%d\n",&m);
fscanf (dane.m_pStream,"%d\n",&p);
```

```
fscanf (dane.m_pStream,"%d\n",&n[0]);
for (i=1;i<=p;i++)
{
fscanf (dane.m_pStream,"%d\n",&n[i]);
n[i]+=n[i-1];
}
Alpha.ReSize(n[p],m);
Alpha=0;
Lambda.ReSize(n[p]);
c.ReSize(n[p]);
for (i=1;i<=n[p];i++)
{
fscanf (dane.m_pStream,"%f\n",&potega);
c(i)=(double)potega;
fscanf (dane.m_pStream,"%d\n",&ilepoteg);
for (j=1;j<=ilepoteg;j++)
{
fscanf (dane.m_pStream,"%d%f\n",
                        &ktorapotega,&potega);
Alpha(i,ktorapotega)=(double)potega;
}
    }
dane.Close;
A.ReSize(m+1,n[p]);
x.ReSize(n[p],1);
B.ReSize(m+1);
B(1)=1;
for (j=1;j<=n[0];j++) A(1,j)=1;
for (j=n[0]+1;j<=n[p];j++) A(1,j)=0;
for (i=1;i<=m;i++)
{
B(i+1)=0;
for (j=1;j<=n[p];j++) {
A(i+1,j)=Alpha(j,i);}
    }
    dod=n[p]-m-1;
if (dod==0)
    {
```

```
x=A.i()*B;
    }
    return TRUE;
};

BOOL zapisz_wyniki()
{

double s;
    wyniki = fopen("Wynik.txt","w+");

    fprintf (wyniki,"PROGRAMOWANIE GEOMETRYCZNE - OPTYMALIZACJA\n");
    fprintf (wyniki,"Andrzej Dudek
                                        1998/99 \n");
    fprintf (wyniki,"Akademia Ekonomiczna, Wroclaw\n") ;
    fprintf (wyniki,"WGRiT Jelenia Gora\n\n");
    fprintf (wyniki,"Program jest czescia pracy doktorskiej:\n");
    fprintf (wyniki,"\"Programowanie Gemetryczne jako
                        narzedzie optymalizacji w ekonomii\"\n");
    fprintf (wyniki,"Pisanej pod opieka
                        Profesora Michala Montygierda - Loyby\n\n");
    fprintf (wyniki,"----------------------------------------
                        ------------------------------\n\n");
    fprintf(wyniki,"Program Pierwotny: \n\n");
    fprintf(wyniki,"Zminimalizowac funkcje:
                                                \n\nf(x) = ");
for (i=1;i<=n[0];i++)
{
    fprintf(wyniki,"%.3f",c(i));
for (j=1;j<=m;j++)
{
if (Alpha(i,j)!=0)
    fprintf(wyniki,"*x%d^(%.3f)",j,Alpha(i,j));

}
if (i!=n[0])
    fprintf(wyniki," + ");
```

```
    }
    fprintf(wyniki,"\n\n");
    fprintf(wyniki,"Przy ograniczeniach:\n\n");
for (k=0;k<=p-1;k++)
{
for (i=n[k]+1;i<=n[k+1];i++)
{
    fprintf(wyniki,"%.3f",c(i));
for (j=1;j<=m;j++)
{
if (Alpha(i,j)!=0)
    fprintf(wyniki,"*x%d^(%.3f)",j,
                                              Alpha(i,j));

}
if (i!=n[k+1])
    fprintf(wyniki," + ");

     }
     fprintf(wyniki," <= 1\n");
    }
    fprintf (wyniki,"\n");
    fprintf (wyniki,"-----------------------------------------------
                        ------------------------\n\n");
    fprintf (wyniki,"Liczba zmiennych:            %d\n",m);
        fprintf (wyniki,"Liczba funkcji ograniczajacych:  %d\n",p);
   fprintf (wyniki,"Liczba pozymianow w funkcji:     %d\n",n[0]);
   fprintf (wyniki,"Laczna liczba pozymianow:   %d\n",n[p]);
   fprintf (wyniki,"Degree of Dificulty:           %d\n\n",dod);
    fprintf (wyniki,"-----------------------------------------------
                        ------------------------\n\n");
   fprintf (wyniki,"Program Dualny:           \n\n");
#ifdef Tylko_liniowo
if (dod==0)
{
#endif
   for (i=1;i<=n[p];i++)
{
```

```
fprintf (wyniki,"delta%d = %f\n",i,x(i));
}
wynik=1;
fprintf (wyniki,"\nWynik = ");
for (i=1;i<=n[p];i++)
{
   fprintf (wyniki,"((%.3f) ^ (%.3f) * ",(c(i)/x(i)),x(i));
   wynik*=potega(c(i)/x(i),x(i));
}
for (i=0;i<=p-1;i++)
{
s=0;
for(j=n[i]+1;j<=n[i+1];j++)
{
s+=x(j);
   }
   fprintf (wyniki,"((%.3f) ^ (%.3f)",s,s);
if (i!=p-1)
{
   fprintf (wyniki," * ");
   }
Lambda(i+1)=s;
   wynik*=potega(s,s);
    }
fprintf (wyniki,"= %f\n",wynik);
    fprintf (wyniki,"----------------------------------
                        ------------------------------------\n\n");
lnAlpha.ReSize(m,m);
lnB.ReSize(m,1);
lnX.ReSize(m,1);

for (i=1;i<=m;i++)
{
for (j=1;j<=m;j++)
{
lnAlpha(i,j)=Alpha(i,j);
if (i<=n[0])
lnB(i)=log(x(i)*wynik/c(i));
```

```
else
{
int kk;   // ktore k do lambdy
for (k=0;k<=p-1;k++)
if (n[k]<i<=n[k+1]) kk=k+1;
lnB(i)=log(x(i)/Lambda(kk)/c(i));
}
}
}
lnX=lnAlpha.i()*lnB;
   for (i=1;i<=m;i++)
{
fprintf (wyniki,"X%d = %.4f\n",i,exp(lnX(i)));
}
#ifdef Tylko_liniowo
}
#endif
fclose(wyniki);
    return TRUE;

};


double potega (double podst, double pot)
{
double wyn;
wyn=exp(pot*log(podst));
return wyn;
}


void _Lambda()
{
double s;
for (i=0;i<=p-1;i++)
{
s=0;
for(j=n[i]+1;j<=n[i+1];j++)
```

```
{
s+=x(j);
   }
Lambda(i+1)=s;
    }
}

void _gradient()
{
gradient.ReSize(n[p]);
    for (i=1;i<=n[0];i++) gradient(i)=log(x(i)/c(i))+1;
    for (j=0;j<=p-1;j++)
     for (i=n[j]+1;i<=n[j+1];i++)
     {
     gradient(i)=log(x(i)/(c(i)*Lambda(j+1)));
     }
}

void _hessian()
{
hessian.ReSize(n[p],n[p]);
for (i=1;i<=n[p];i++)
for (j=1;j<=n[p];j++) hessian(i,j)=0;
for (i=1;i<=n[0];i++) hessian (i,i)=1/x(i);
    for (j=0;j<=p-1;j++)
     for (i=n[j]+1;i<=n[j+1];i++)
     for (k=n[j]+1;k<=n[j+1];k++)
if(i==k)
{
hessian(i,k)=1/x(i)-1/Lambda(j+1);
                }
                else
                {
hessian(i,k)=-1/Lambda(j+1);
}
}
```

```
void _mi()
{
Matrix M(1,1);
mi=(x.t()*z).AsScalar()/n[p];

}

Matrix diag(ColumnVector xx)
{
Matrix M;
M.ReSize(xx.Nrows(),xx.Nrows());
M=0;
for (i=1;i<=xx.Nrows();i++) M(i,i)=xx(i);
return M;
}

double Norma(Matrix M)
{
double norm;
norm=0;
for (i=1;i<=M.Nrows();i++)
for (j=1;j<=M.Ncols();j++) norm+=fabs(M(i,j));
return norm;
}

double Norma2(Matrix M)
{
double norm;
norm=0;
for (i=1;i<=M.Nrows();i++)
for (j=1;j<=M.Ncols();j++) norm+=M(i,j)*M(i,j);
norm=sqrt(m);
return norm;
}

void znajdz_kierunek()
{
Matrix M,M1,Zero;
```

```
ColumnVector Delta;
ColumnVector LewaStrona;
ColumnVector E;


Zero.ReSize(m+1,m+1);
Zero=0;
M1=Zero|A;
Zero.ReSize(m+1,n[p]);
Zero=0;
M1=M1|Zero;
M=M1;
    M1=A.t()|-hessian;
Zero.ReSize(n[p],n[p]);
Zero=0;
for (i=1;i<=n[p];i++) Zero(i,i)=1;
M1=M1|Zero;
M=M&M1;
Zero.ReSize(n[p],m+1);
Zero=0;

M1=Zero|diag(z);
    M1=M1|diag(x);
M=M&M1;

E.ReSize(n[p]);
E=log(1);
LewaStrona=(eta*r_p)&(eta*r_d)&(gamma*mi*E-diag(x)*z);

Delta=delta_y&delta_x&delta_z;


if ((M.LogDeterminant()).Sign()) Delta=M.i()*LewaStrona;
else
{
    wyniki = fopen("Bledy.txt","w+");
MessageBox(NULL,"Sprzeczny uklad","Sprzeczny uklad",
                            MB_ICONSTOP);
```

```
fprintf(wyniki,"Wyznacznik ukladu = 0  !!!!!\n");
for (i=1;i<=2*n[p]+m+1;i++)
{
for (j=1;j<=2*n[p]+m+1;j++)
        {
                                if (M(i,j)>=0) fprintf (wyniki," %.2f ",
                                M(i,j)); else fprintf(wyniki,"%.2f ",M(i,j))
}
fprintf(wyniki,"\n");
}
fprintf(wyniki,"Wyznacznik ukladu = 0  !!!!!\n");
fclose (wyniki);
exit(0);
}
for(i=1;i<=m+1;i++) delta_y(i)=Delta(i);
for(i=1;i<=n[p];i++) delta_x(i)=Delta(i+m+1);
for(i=1;i<=n[p];i++) delta_z(i)=Delta(i+m+1+n[p]);
}


bool  krok_alpha(double alpha)
{
ColumnVector xx;
xx=x;
x=x+alpha*delta_x;
_gradient();
x=xx;

if ( ((x+alpha*delta_x).t()*(z+alpha*delta_z)).AsScalar() >
                thetac*((x.t()*z).AsScalar()) )
return FALSE;

if (Norma(B-A*(x+alpha*delta_x)) > thetap*((x+alpha*delta_x).t()
                *(z+alpha*delta_z)).AsScalar())
return FALSE;

if (Norma(gradient-A.t()*(y+alpha*delta_y)-(z+delta_z)) >
                thetap*((x+alpha*delta_x).t()*(z+alpha*delta_z)).AsScalar())
```

```
return FALSE;

_gradient();
return TRUE;
}

void rozwiaz()
{

double kryterium_p,epsilon_p,kryterium_d,epsilon_d,kryterium_c,epsilon_c;
double alpha,alpha_x,alpha_z;
double Theta_c,Theta_p,Theta_d;
ColumnVector s;


int iteracja;

// **********************Poczatek - podstawienia


x.ReSize(n[p]);
y.ReSize(m+1);
z.ReSize(n[p]);
r_p.ReSize(m+1);
r_d.ReSize(n[p]);
delta_x.ReSize(n[p]);
delta_y.ReSize(m+1);
delta_z.ReSize(n[p]);
for (i=1;i<=n[p];i++)
{
x(i)=exp(1);
z(i)=exp(1);
}
for (i=1;i<=m+1;i++)
{
y(i)=0;
}
sigma=potega(10,-8);
```

```
    beta=potega(10,3);
epsilon_p=potega(10,-8);
epsilon_d=potega(10,-8);
epsilon_c=potega(10,-12);

kryterium_p=1;
kryterium_d=1;
kryterium_c=1;

_Lambda();
    _gradient();
    _hessian();
r_p=B-A*x;
r_d=gradient-A.t()*y-z;
_mi();

thetac=2;
thetap=100*Norma(r_p)/((x.t()*z).AsScalar());
thetad=10000*Norma(r_d)/((x.t()*z).AsScalar());

iteracja=0;
        // **********************Petla ************************

    while( ((kryterium_p>=epsilon_p) || (kryterium_d>=epsilon_d)
    ||(kryterium_c>=epsilon_c)) && (iteracja<=MaxIter))
    {

// ********************** KROK 2 - Podstawienia
        _Lambda();
     _gradient();
     _hessian();
r_p=B-A*x;
r_d=gradient-A.t()*y-z;
_mi();
if (iteracja==0)
{
Theta_c=2;
Theta_p=100*(r_p.Norm1()/(x.t()*z).AsScalar());
```

```
Theta_d=10000*(r_d.Norm1()/(x.t()*z).AsScalar());
        }
// ********************** KROK 3 - Wybor Parametrow
eta=1;
gamma=0;
znajdz_kierunek();
alpha_x=0;
alpha_z=0;
        for (i=1;i<=n[p];i++)
        {
         if ((delta_x(i)<0) && (-x(i)/delta_x(i)>alpha_x))
         alpha_x=-x(1)/delta_x(1);
         if ((delta_z(i)<0) && (-z(i)/delta_z(i)>alpha_z))
         alpha_z=-z(1)/delta_z(1);
}
if (alpha_x<alpha_z) alpha=alpha_x; else alpha=alpha_z;
gamma=0.5*(((x+alpha*delta_x).t()*(z+alpha*delta_z))
                        .AsScalar()/n[p])/mi;
eta=1-gamma;

// ********************** KROK 4 - Wybor Kierunku
znajdz_kierunek();


// ********************** KROK 5 - Skala Przesuniecia
                alpha_x=duzo;                                   //
                alpha_z=duzo;                                   //
        for (i=1;i<=n[p];i++)
        {
         if ((delta_x(i)<0) && (-x(i)/delta_x(i)<alpha_x))
         alpha_x=-x(i)/delta_x(i);
         if ((delta_z(i)<0) && (-z(i)/delta_z(i)<alpha_z))
         alpha_z=-z(i)/delta_z(i);
}
if (alpha_x<alpha_z) alpha=alpha_x; else alpha+alpha_z;
alpha*=factor;
        }
// ********************** KROK 6  Przejscie w Optymalnym Kierunku
```

```
y+=alpha*delta_y;
x+=alpha*delta_x;
z+=alpha*delta_z;


// ********************** KROK 7 Slack Reset
     s=gradient-A.t()*y;
    for (i=1;i<=n[p];i++)
        {
if (s(i)>=0)
{
if (s(i)<=(z(i)/fac))
z(i)/=fac;
else
if (s(i)>=(z(i)*fac))
z(i)/=fac;
else
z(i)=s(i);
}
}
kryterium_p=r_p.Norm1()/(1+x.Norm1());
kryterium_d=r_d.Norm1()/(1+z.Norm1());
kryterium_c=(x.t()*z).AsScalar()/(1+x.Norm1()+z.Norm1());

iteracja++;
if (iteracja==MaxIter)
                        MessageBox(NULL,"Max Iteration Number Reached"
                        ,"End",MB_ICONSTOP);
    }
}
```