Spis treści

1 Zastosowanie Matlab'a

1.1 Wstęp

Zadania programowania liniowego ZPL postaci:

$$\min_x f^T x \text{ , z ograniczeniami: } A \cdot x \le b$$
$$Aeq \cdot x = beq$$
$$lb \le x \le ub$$

gdzie:
f, x, b, beq, lb oraz ub są wektorami, a A i Aeq - macierzami.

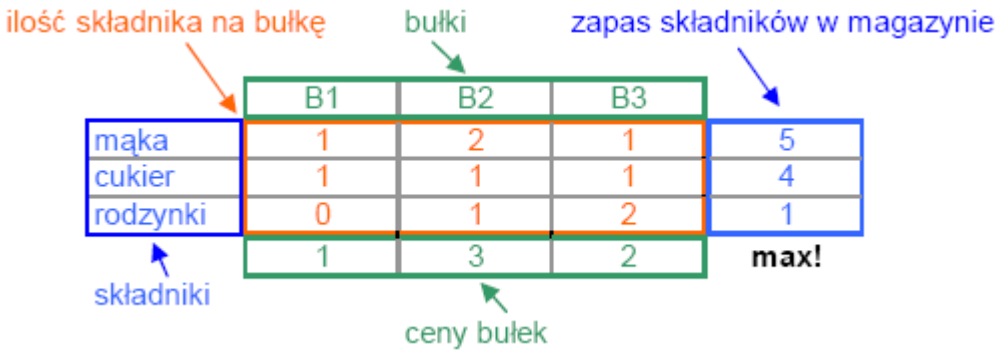
pozwala rozwiązywać funkcja Matlaba **linprog**.

Składnia
x = linprog(f,A,b,Aeq,beq)
x = linprog(f,A,b,Aeq,beq,lb,ub)
x = linprog(f,A,b,Aeq,beq,lb,ub,x0)
x = linprog(f,A,b,Aeq,beq,lb,ub,x0,options)
[x,fval] = linprog(...)
[x,fval,exitflag] = linprog(...)
[x,fval,exitflag,output] = linprog(...)
[x,fval,exitflag,output,lambda] = linprog(...)

Opis
linprog rozwiązuje ZPL.
x = linprog(f,A,b) - rozwiązuje zdania: min f'*x, z ograniczeniami typu: A*x <= b.
x = linprog(f,A,b,Aeq,beq) - rozwiązuje powyższe zadanie z dodatkowymi ograniczeniami równościowymi: Aeq*x = beq. Jeśli nie występują ograniczenia nierównościowe wstawiamy A=[] oraz b=[] .
x = linprog(f,A,b,Aeq,beq,lb,ub) - pozwala określić zakresy dopuszczalnych wartości dla zmiennych decyzyjnych x, co skutkuje tym, że rozwiązanie spełnia warunek lb <= x <= ub. Jeśli nie występują ograniczenia równościowe wstawiamy Aeq=[] oraz beq=[] .
x = linprog(f,A,b,Aeq,beq,lb,ub,x0) - ustala punkt startowy x0. Opcja ta jest dostępna jedynie dla algorytmów typu MediumScale (parametr LargeScale ma wartość 'off', zmiany wartości parametrów realizuje funkcja optimset). Domyślnie stosowany algorytm typu LargeScale ignoruje zadany punkt startowy.
x = linprog(f,A,b,Aeq,beq,lb,ub,x0,options) - postać umożliwiająca ustalenie parametrów dla algorytmu optymalizacji na wartościach określonych w strukturze options. Do określania wartości poszczególnych parametrów służy funkcja optimset.
[x,fval] = linprog(...) - zwraca wartść funkcji celu fun dla uzyskanego rozwiązania x: fval = f'*x.
[x,lambda,exitflag] = linprog(...) - zwraca wartość exitflag, która określa warunki zakończenia obliczeń.
[x,lambda,exitflag,output] = linprog(...) - zwraca strukturę output, zawierającą informacje o przeprowadzonej optymalizacji.
[x,fval,exitflag,output,lambda] = linprog(...) - zwraca strukturę lambda, której pola zawierają wartości mnożników Lagrange'a dla rozwiązania x.

1.2 Zagadnienie standardowe

Obliczmy zadanie, które wcześniej wyliczyliśmy metodą simpleks.
Poniżej przedstawiona została tabelka z danymi z zadania.



Tabelka.1. Dane do zadania

Jest to standardowe zagadnienie programowania liniowego przy funkcji celu dążącej do maximum.

Postać standardowa układu:

$1x_1 + 3x_2 + 2x_3$ --> MAX

$1x_1 + 2x_2 + 1x_3 <= 5$
$1x_1 + 1x_2 + 1x_3 <= 4$
$0x_1 + 1x_2 + 2x_3 <= 1$

$x_1 >= 0, x_2 >= 0, x_3 >= 0$

Użyjemy funkcji **linprog** o składni:
[x, fval] = **linprog**(f, A, b, Aeq, beq, lb, ub)
w wyniku jej działania otrzymamy wektor x z rozwiązaniem oraz zysk pod zmienną fval.

Przygotujmy dane wejściowe do funkcji linprog:

-> wektor f
Wektor współczynników funkcji celu lub patrząc na tabelkę - ostatni zielony wiersz:
f=[1 3 2];

-> macierz A
Macierz współczynników lewej strony nierówności lub patrząc na tabelkę - środkowa pomarańczowa ramka:
A=[1 2 1
1 1 1
0 1 2];

-> wektor b
Wektor liczb znajdujących sie po prawej stronie nierówności lub patrząc na tabelkę - ostatnia niebieska kolumna:
b=[5
4
1];

-> Macierz Aeq i wektor beq dotyczy równości - w naszym wypadku pozostawiamy je puste:
Aeq=[];
beq=[];

-> Wektor lb:
Wektor ograniczeń dolnych rozwiązania. Każda zienna rozwiązania ma być większa od zera:
lb=[0 0 0];

-> Wektor ub:
Wektor ograniczeń górnych rozwiązania. Nie nakładamy ograniczeń górnych na zmienne rozwiązania:
ub=[];
lub
ub=[inf inf inf]; **inf** - nieskończenie duża liczba

Wstawienie danych do funkcji linprog:
[x, fval] = **linprog**(-f, A, b, Aeq, beq, lb, ub)

Ponieważ standardowo funkcja linprog liczy minimum f-kcji celu, musimy wektor f wstawić ze znakiem minus - wówczas funkcja wyliczy maximum.

wynik z funkcji:
x =
3.0000
1.0000
0.0000

fval =
-6.0000

Wstawiliśmy znak minus przed wektor f dzięki czemu uzyskaliśmy maximum funkcji celu ale dlatego też uzyskaliśmy zysk ze znakiem minus. Licząc maximum funkcji celu czytamy wartość fval pomijając minus:

$x_1 = 3$
$x_2 = 1$
$x_3 = 0$

zysk = 6

W efekcie uzyskaliśmy ten sam wynik co w metodzie simpleks ale dużo szybciej.

1.3 Zagadnienie transportowe

Obliczmy przy pomocy Matlab'a zagadnienie transportowe, które wyliczyliśmy wcześniej ręcznie.
Poniżej przedstawiona została tabelka z danymi z zadania.

dostawcy ↓

20	30	10	40		
P1	P2	P3	P4	odbiorcy ↓	
5	2	1	5	S1	10
3	1	1	4	S2	15
1	1	2	3	S3	30
2	1	5	1	S4	10
2	1	2	6	S5	35

koszty ↑

Tabelka.1. Dane do zadania

Postać standardowa układu:

$20x_1 + 30x_2 + 10x_3 + 40x_4$ --> MIN

$5x_1 + 2x_2 + 1x_3 + 5x_4 = 10$
$3x_1 + 1x_2 + 1x_3 + 4x_4 = 15$
$1x_1 + 1x_2 + 2x_3 + 3x_4 = 30$
$2x_1 + 1x_2 + 5x_3 + 1x_4 = 10$
$2x_1 + 1x_2 + 2x_3 + 6x_4 = 35$

$x_1 >= 0, x_2 >= 0, x_3 >= 0, x_4 >= 0, x_5 >= 0$

Użyjemy funkcji **linprog** o składni:
[x, fval] = **linprog**(f, A, b, Aeq, beq, lb, ub)

w wyniku jej działania otrzymamy wektor x z rozwiązaniem oraz zysk pod zmienną fval.

Przygotujmy dane wejściowe do funkcji linprog:

-> wektor f
Wektor współczynników lewej strony nierówności lub patrząc na tabelkę - liczby koloru zielonego. Przy czym wartości z tabelki przepisujemy do wektora beq wierszami:
f=[5 2 1 5 3 1 1 4 1 1 2 3 2 1 5 1 2 1 2 6];

-> Macierz A i wektor b
Dotyczą one nierówności wiec w naszym przypadku nie maja miejsca:
A=[];
b=[];

-> Macierz Aeg

Macierz przygotowyjemy w następujący sposób:
- ilość wierszy macierzy = m+n
- ilość kolumn macierzy = m*n

gdzie:
m - liczba odbiorców (m=5),
n - liczba dostawców (n=4).

Pierwsze m-wierszów wypełniamy następująco:

Dzielimy sobie każdy wiersz na m-grup po n-liczb (tutaj 5 grup po 4 liczby każda). Następnie wypełniamy wszystkie zerami tylko grupę **i** w wierszu **i** wypełniamy jedynkami, gdzie **i** to numer kolejnego wiersza (i=1,2,...,m).

Kolejne wiersze od m do m+n wypełniamy następująco:

Również dzielimy sobie każdy wiersz na m-grup po n-liczb (tutaj 5 grup po 4 liczby każda). Następnie wypełniamy wszystkie zerami tylko **i**-tą liczbę w każdej grupie w wierszu **i**+m wypełniamy jedynkami, gdzie **i** = 1,2,...,n.

```
Aeq=[ 1 1 1 1  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0
      0 0 0 0  1 1 1 1  0 0 0 0  0 0 0 0  0 0 0 0
      0 0 0 0  0 0 0 0  1 1 1 1  0 0 0 0  0 0 0 0
      0 0 0 0  0 0 0 0  0 0 0 0  1 1 1 1  0 0 0 0
      0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0  1 1 1 1
      1 0 0 0  1 0 0 0  1 0 0 0  1 0 0 0  1 0 0 0
      0 1 0 0  0 1 0 0  0 1 0 0  0 1 0 0  0 1 0 0
      0 0 1 0  0 0 1 0  0 0 1 0  0 0 1 0  0 0 1 0
      0 0 0 1  0 0 0 1  0 0 0 1  0 0 0 1  0 0 0 1 ];
```

-> Wektor beq

Wektor liczb występujących po prawej stronie równań oraz współczynników funkcji celu.
Patrząc na tabelkę - ostatnia pomarańczowa kolumna oraz pierwszy, niebieski wiersz:
beq=[10 15 30 10 35 20 30 10 40];

-> Wektor lb:

Wektor ograniczeń dolnych rozwiązania. Każda zienna rozwiązania ma być większa od zera:
lb=[0 0 0 0 0];

-> Wektor ub:

Wektor ograniczeń górnych rozwiązania. Nie nakładamy ograniczeń górnych na zmienne rozwiązania:
ub=[];
lub
ub=[inf inf inf inf inf]; **inf** - nieskończenie duża liczba

Wstawienie danych do funkcji linprog:

[x, fval] = **linprog**(f, A, b, Aeq, beq, lb, ub)

wynik z funkcji:

x =
0.0000
0.0000
10.0000
0.0000
0.0000
6.2311
0.0000
8.7689
8.7689
0.0000
0.0000
21.2311
0.0000
0.0000
0.0000
10.0000
11.2311
23.7689
0.0000
0.0000

fval =
180.0000

Otrzymaliśmy więc wynik:

0.0000	0.0000	10.0000	0.0000
0.0000	6.2311	0.0000	8.7689
8.7689	0.0000	0.0000	21.2311
0.0000	0.0000	0.0000	10.0000
11.2311	23.7689	0.0000	0.0000

przy koszcie = 180

Rozwiązanie optymalne różni się od naszego. Koszt otrzymaliśmy ten sam.