ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH

Đại số tuyến tính

Bài tập lớn

Ứng dụng đại số tuyến tính trong qui hoạch tuyến tính với bài toán lập kế hoạch sản xuất

---

GVHD: Nguyễn Tiến Thịnh
Nguyễn An Khương
SV thực hiện: Nguyễn Đình Khánh – 1810992
Nguyễn Tấn Đạt – 1810700
Nguyễn Huỳnh Long – 1811049
Phan Gia Anh – 1710009
Nguyễn Hoàng Khang – 1812545

Tp. Hồ Chí Minh, Tháng 11/2020

Mục lục

Báo cáo trình bày về một số lệnh MATLAB cơ bản thường dùng trong quá trình giải các bài toán đến đại số tuyến tính. Sau đó chúng tôi sẽ trình bày về ứng dụng của đại số tuyến tính vào bài toán quy hoạch tuyến tính trong bài toán lập kế hoạch sản xuất

Phần đầu tiên, chúng tôi sẽ trình bày về cú pháp, và công dụng của các lệnh cơ bản nhất khi làm việc với đại số tuyến tính trên công cụ MATLAB.

Trong các phần tiếp theo, chúng tôi sẽ trình bày ứng dụng của đại số tuyến trình trong việc lập kế hoạch sản xuất tối ưu trong thực tế. Đầu tiên, chúng tôi sẽ mô tả bài toán lập kế hoạch sản xuất trong thực tế, mô hình hóa toán học bài toán đó và giới thiệu cách giải quyết bằng qui hoạch tuyến tính. Phần tiếp theo sẽ trình bày về cách giải bài toán trên lý thuyết. Trong phần cuối của báo cáo, chúng em sẽ trình bày một phương hướng giải quyết với một bài toán cụ thể bằng công cụ Matlab.

1 Một số lệnh MATLAB cơ bản sử dụng trong các bài toán đại số tuyến tính

1.1 Số phức trong MATLAB

1.1.1 Lệnh real, imag

Ý nghĩa

- real: Lấy phần thực của số phức
- imag: Lấy phần ảo của số phức

Cú pháp

- phanthuc= real(z)
- phanao= imag(z)

1.1.2 Lệnh abs

Ý nghĩa

- Tìm modul của số phức

Cú pháp

- Y=abs(z)

1.1.3 Lệnh angle

Ý nghĩa

- Tìm agument của số phức với đơn vị là radian

Cú pháp

- Y=angle(z)

1.1.4 Lệnh conj

Ý nghĩa

- Lấy số phức liên hợp của số phức

Cú pháp

- Y=conj(z)

1.2 MA TRẬN TRONG MATLAB

1.2.1 Lệnh numel

Ý nghĩa

- Đếm số phần tử của a

Cú pháp

- numel(A)

1.2.2 Lệnh size

Ý nghĩa

- Cho biết số dòng và cột của một ma trận

Cú pháp

- size(A)

1.2.3 Lệnh inv

Ý nghĩa

- Tìm ma trận nghịch đảo của ma trận

Cú pháp

- inv(A)

1.2.4 Lệnh ghép hai ma trận theo cột

Ý nghĩa

- Ghép hai ma trận cùng số hàng

Cú pháp

A;B với A, B là hai ma trận cho trước

1.2.5 Lệnh ghép hai ma trận theo hàng

Ý nghĩa

- Ghép hai ma trận cùng số cột

Cú pháp

A B với A, B là hai ma trận cho trước

1.2.6 Lệnh A^k

Ý nghĩa

- lũy thừa ma trận

Cú pháp

- A^k

1.2.7 Xóa cột, hàng của ma trận A

Ý nghĩa

- Xóa cột thứ i của ma trận A

Cú pháp

- A(:,i)=[] để xóa hàng thứ i
- A(i,:)=[] để xóa cột thứ i

1.2.8 Câu lệnh A(:,n:end)

Ý nghĩa

- Cho phép lấy từ cột, hàng thứ n đến cột cuối của ma trận

Cú pháp

- Cú pháp cột: A(:,n:end)
- Cú pháp hàng: A(n:end,:)

1.2.9 Lệnh zeros(m,n)

Ý nghĩa

- Tạo ma trận 0 với m hàng n cột

Cú pháp

- zeros(m,n)

1.2.10 Lệnh eye(m,n)

Ý nghĩa

- Tạo ma trận đơn vị mở rộng

Cú pháp

- eye(m,n)

1.2.11 Lệnh ones(m,n)

Ý nghĩa

- Tạo ma trận toàn số 1 với m hàng n cột

Cú pháp

- ones(m,n)

1.2.12 Lệnh rank(A)

Ý nghĩa

- Tính hạng của ma trận

Cú pháp

- rank(A)

1.2.13 Lệnh trace(A)

Ý nghĩa

- Tính vết của ma trận

Cú pháp

- trace(A)

1.2.14 Lệnh A'

Ý nghĩa

- Tính ma trận chuyển vị của A

Cú pháp

- A'

1.2.15 Lệnh det(A)

Ý nghĩa

- Tính định thức của A

Cú pháp

- det(A)

1.2.16 Lệnh tril(A)

Ý nghĩa

- Trích ma trận tam giác dưới từ ma trận A

Cú pháp

- tril(A)

1.2.17 Lệnh triu(A)

Ý nghĩa

- Trích ma trận tam giác trên từ ma trận A

Cú pháp

- triu(A)

1.2.18 Lệnh [Q,R]=qr(Y) hoặc [L,U]=lu(Y)

Ý nghĩa

- Phân tích hai ma trận

Cú pháp

Q,R =qr(Y) : Phân tích Y thành tích hai ma trận Q và R

L,U =lu(Y) : Phân tích Y thành tích hai ma trận L và U, Với Y là ma trận cho trước

1.2.19 Lệnh A(i,j)

Ý nghĩa

- Tham chiếu phần tử dòng i cột j

Cú pháp

- A(i,j)

1.2.20 Lệnh A(i,:) và A(:,j)

Ý nghĩa

- Tham chiếu dòng i và tham chiếu cột j

Cú pháp

- A(i,:) và A(:,j)

1.2.21 Lệnh A(i :k, :) và A(:,j :k)

Ý nghĩa

- : Tham chiếu từ dòng i đến dòng k và tham chiếu từ cột j đến cột k

Cú pháp

- A(i :k, :) và A(:,j :k)

1.2.22 Lệnh rref(A)

Ý nghĩa

- Tạo ma trận bậc thang từ A

Cú pháp

- rref(A)

1.2.23 Lệnh FLIPLR

Ý nghĩa

- Chuyển các phần tử của các ma trận theo thứ tự cột ngược lại

Cú pháp

- b= fliplr(a)

1.2.24 Lệnh FLIPUD

Ý nghĩa

- Chuyển các phần tử của ma trận theo thứ tự hàng ngược lại

Cú pháp

- b= flipup(a)

1.2.25 Lệnh MAGIC

Ý nghĩa

- Tạo 1 ma trận vuông có tổng của các phần tử trong 1 hàng, 1

Cú pháp

- Tên ma trận = magic(n)

1.2.26 Lệnh PASCAL

Ý nghĩa

- Tạo ma trận theo quy luật tam giác Pascal

Cú pháp

- pascal (n)

1.2.27 Lệnh RAND

Ý nghĩa

- Tạo ma trận mà giá trị của các phần tử là ngẫu nhiên

Cú pháp

- y= rand(n) : ma trận vuông nxn
- y= rand(m,n) : ma trận mxn

1.2.28 Lệnh isemty

Ý nghĩa

- Kiểm tra xem ma trận có phải là ma trận rỗng không

Cú pháp

- isemty(A)

1.2.29 Lệnh DIAG

Ý nghĩa

- Tạo ma trận mới và xử lý đường chéo theo quy ước

Cú pháp

- v= diag(x): số hàng bằng số cột bằng số chiều của vector x, các phần tử của x nằm trên đường chéo chính
- v= diag(x,k): số hàng, cột của v bằng số chiều của vector x cộng với giá trị tuyệt đối của k. Nếu k âm, các phần tử sẽ nằm phía dưới đường chéo, ngược lại thì sẽ tải từ trên đường chéo xuống.

1.3 Một số lệnh trong không gian vector, không gian Euclide, trị riêng

1.3.1 Lệnh dot

Ý nghĩa

- Tính tích vô hướng hai vector

Cú pháp

- dot(u,v)

1.3.2 Lệnh cross

Ý nghĩa

- Tính tích hữu hướng

Cú pháp

- cross(u,v)

1.3.3 Lệnh length

Ý nghĩa

- Tính chiều dài của vector bất kì

Cú pháp

- length(v)

1.3.4 Lệnh norm

Ý nghĩa

- Tính độ dài vector bất kì

Cú pháp

- norm(v)

1.3.5 Lệnh qr

Ý nghĩa

- Trực chuẩn hóa họ vector cột A

Cú pháp

P, =qr(A)

1.3.6 Lệnh [P,D]=eig(A)

Ý nghĩa

- Chéo hóa

Cú pháp

P,D =eig(A)

1.3.7 Lệnh eig(H)

Ý nghĩa

- Trị riêng

Cú pháp

- eig(H)

1.3.8 Lệnh max(X), min(X)

Ý nghĩa

- Trả về giá trị lớn nhất và nhỏ nhất

Cú pháp

- max(A) và min(A)

2 Đặt vấn về: bài toán lập kế hoạch sản xuất tối ưu

Mọi công ty, doanh nghiệp, hộ sản xuất... đều có một mục đích chung là tạo ra lợi nhuận tốt nhất có thể để duy trì, phát triển sản xuất với nhiều mục đích khác nhau. Ngày nay, lập kế hoạch sản xuất với nhiều yêu cầu khắt khe về chất lượng, số lượng, nguồn lực, thời hạn, chi phí rẻ nhất có thể, lợi nhuận lớn nhất có thể đang là bài toán đau đầu với các nhà kinh tế nhằm duy trì sự sống, phát triển cho các công ty, xí nghiệp. Nhờ những nỗ lực tối ưu này mà xã hội nói chung và các ngành sản xuất nói riêng không ngừng phát triển trong thế giới đầy biến động và khắc nghiệt hiện nay.
Qui hoạch tuyến tính tỏ ra là một kĩ thuật toán học cực kì hữu ích trong bài toán này với ý tưởng chính là tìm giá trị cực đại trong miền giới hạn nhờ vào các định lý toán học. Kĩ thuật này còn được dùng trong các bài toán như vận chuyển, phân công công việc... Tuy nhiên trong giới hạn bài này chúng tôi sẽ chỉ giới thiệu ứng dụng trong bài toán lập kế hoạch sản xuất với việc áp dụng đại số tuyến tính để giải quyết bài toán. Qui hoạch tuyến tính lần đầu được giới thiệu tại [1] và được giải quyết rộng rãi bằng phương pháp đơn hình được giới thiệu tại [2].

3 Mô hình hoá toán học bài toán

Để giaỉ được bài toán này bằng công cụ toán học đòi hỏi phải mô hình hoá toán học bài toán hay phát biểu nó dưới dạng toán học để giải quyết.
Trong kinh tế, hàm sản xuất biểu thị được mối quan hệ giữa đầu vào và đầu ra. Vậy mục tiêu

của chúng ta sẽ là tối ưu hàm sản xuất này:

$$f(x) = C_1x_1 + C_2x_2 + ...C_nx_n = \sum_{i=1}^{n} C_ix_i \tag{1}$$

Trong đó, $c_i, \forall i \in \{1, 2, .., n\}$ là các hằng số thực và $x_i, \forall i \in \{1, 2, .., n\}$ là các biến biểu thị đầu vào của quá trình sản xuất. Những ràng buộc trong bài toán lập kế hoạch sản xuất phải là các ràng buộc tuyến tính. Các bất đẳng thức tuyến tính hay phương trình tuyến tính ở (2) được gọi là ràng buộc tuyến tính, còn ràng buộc (3) là ràng buộc không âm.

$$\sum_{j=0}^{n_i} a_{i,j} \leq b_i, \forall i \in \{1, 2, ...m\} \tag{2}$$

$$x_j \geq 0, \forall j \in \{1, 2, ..., n\} \tag{3}$$

Ta đặt:

$$c = \begin{pmatrix} c_1 \\ c_2 \\ . \\ . \\ c_n \end{pmatrix}, A_{m\times n} = \begin{pmatrix} a_{1,1} & a_{1,2} & ... & a_{1,n} \\ a_{2,1} & a_{2,2} & ... & a_{2,n} \\ . & . & . & . \\ . & . & . & . \\ a_{m,1} & a_{m,2} & ... & a_{m,n} \end{pmatrix}, b = \begin{pmatrix} b_1 \\ b_2 \\ . \\ . \\ b_n \end{pmatrix}, x = \begin{pmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{pmatrix} \tag{4}$$

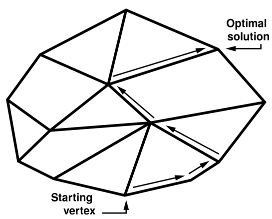
Khi đó, ta quy bài toán về dạng đại số như sau:
Tối đa:

$$c^T x \tag{5}$$

Phụ thuộc vào:

$$Ax \leq b \tag{6}$$
$$x \geq 0. \tag{7}$$

Về ý nghĩa hình học, các bất phương trình ràng buộc tuyến tính thiết lập một miền khả thi như Hình 1. Trong đa tạp lồi này, nếu hàm mục tiêu có có giá trị lớn nhất thì giá trị đó phải sẽ nằm trên các điểm cực hay là các điểm giao nhau của các bất phương trình ràng buộc. Mà số điểm giao là có hạn và theo định lý [3] thì sau một số bước nhất định ta hoàn toàn có thể tìm nghiệm tối ưu cho hàm mục tiêu. Chính vì thế, có thể dùng máy tính để giải quyết bài toán tối ưu này.
Với bài toán này, chúng ta có thể dùng rất nhiều phương pháp giải như thuật toán Karmakar [4] sử dụng inferior point, hay phổ biến hơn là phương pháp đơn hình.



Hình 1: *Miền khả thi trong miền tìm kiếm được tạo ra bởi các ràng buộc tuyến tính của bài toán*

4 Ứng dụng đại số tuyến tính trong phương pháp đơn hình giải bài toán lập kế hoạch sản xuất tối ưu

4.1 Phương pháp đơn hình

Ý tưởng của phương pháp đơn hình như sau: bắt đầu với một điểm cực và tìm kiếm từ điểm cực này sang điểm cực khác để cải thiện giá trị cho hàm mục tiêu tăng (hoặc giảm) đến khi đạt giá trị lớn nhất (hoặc nhỏ nhất).
Các bước của phương pháp đơn hình như sau [5]:

1. Mô hình hoá bài toán.
2. Biến đổi các ràng buộc bất đẳng thức thành đẳng thức bằng các biến phụ rồi lần lượt điền hệ số vào bảng.
3. Viết hàm mục tiêu ở dòng cuối cùng trong bảng
4. Biến âm nhỏ nhất trong dòng cuối cùng của bảng sẽ chỉ định pivot bắt đầu tìm kiếm.
5. Tính các thương số. Thương số nhỏ nhất xác định một hàng. Phần tử trong giao điểm của cột được xác định ở bước 4 và hàng được xác định trong bước này được xác định là phần tử xoay. Thương số được tính bằng cách chia cột ngoài cùng bên phải cho cột đã xác định ở bước 4. Thương số là số 0 hoặc số âm hoặc có số 0 ở mẫu số sẽ bị bỏ qua.
6. Thực hiện xoay vòng để làm cho tất cả các mục nhập khác trong cột này bằng không.
7. Khi không còn mục nhập âm nào ở hàng dưới cùng, ta đã hoàn thành; nếu không, ta bắt đầu lại từ bước 4.
8. **Thu nhận kết quả**. Lấy các biến bằng cách sử dụng các cột có 1 và 0. Tất cả các biến khác bằng không. Giá trị tối đa cần tìm kiếm xuất hiện ở góc dưới cùng bên phải.

4.2 Các sử dụng phương pháp đơn hình

Bài toán thực tế [6]: Nhà may váy cưới cao cấp A có sản xuất hai sản phẩm cùng lúc là áo cưới truyền thống và áo cưới kiểu Âu. Công ty không bao giờ muốn chi nhiều hơn tổng cộng 12 triệu trong một ngày cho nhân viên may. Qua nhận định trong quá trình làm việc, nhà may đã xác định rằng mỗi một sản phẩm áo cưới truyền thống nhà may làm ra cần 2 triệu tiền công, và mỗi áo cưới kiểu Âu thì nhà may cần chi tổng cộng 1 triệu cho tiền công. Nếu cô ấy kiếm được 40 triệu khi bán được một áo cưới truyền thống và 30 triệu với một áo cưới kiểu Âu. Giả sử rằng mọi áo cưới may ra đều bán được, nhà may nên phân chia tiền công thế nào để thu được lợi nhuận cao nhất?
Bước 1: Mô hình hoá bài toán
Đặt:
x_1: Số tiền công nhà may trả cho nhân viên làm áo cưới truyền thống
x_2: Số tiền công nhà may trả cho nhân viên làm áo cưới kiểu Âu
Vậy ta cần tối ưu hàm mục tiêu sau:

$$Z = 40x_1 + 30x_2 \tag{8}$$

với các ràng buộc:

$$x_1 + x_2 \leq 12 \tag{9}$$
$$2x_1 + x_2 \leq 16 \tag{10}$$
$$x_1 \geq 0, x_2 \geq 0 \tag{11}$$

Bước 2: Chuyển các ràng buộc bất đẳng thức thành ràng buộc đẳng thức
Chúng ta dùng thêm một biến phụ không âm (slack variable) để thêm vào vế nhỏ hơn của các bất đẳng thức để biến nó thành đẳng đồng thời cũng giả sử Z là giá trị lớn nhất của tối ưu của bài toán và chuyển vế về như sau:

$$\begin{array}{ll} \text{Hàm mục tiêu} & -40x_1 - 30x_2 + Z = 0 \\ \text{Ràng buộc:} & x_1 + x_2 + y_1 = 12 \\ & 2x_1 + x_2 + y_2 = 16 \\ & x_1 \geq 0; x_2 \geq 0 y_1 \geq 0; y_2 \geq 0 \end{array}$$

Bước 3: Thiết lập bảng đơn hình khởi điểm

x_1	x_2	y_1	y_2	Z	C
1	1	1	0	0	12
2	1	0	1	0	16
-40	-30	0	0	1	0

Áp dụng cách đặt như (4) ta có thể thu được một ma trận như sau:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 12 \\ 2 & 1 & 0 & 1 & 0 & 16 \\ -40 & -30 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Trong quá trình biến đổi, kết quả thu được sẽ phụ thuộc vào ma trận sau, ta có thể đọc là $y_1 = 12, y_2 = 16, Z = 0$.

$$\begin{bmatrix} y_1 & y_2 & Z & | & C \\ 1 & 0 & 0 & | & 12 \\ 0 & 1 & 0 & | & 16 \\ 0 & 0 & 1 & | & 0 \end{bmatrix}$$

Bước 4: Chọn điểm âm nhất trong hàng cuối cùng làm điểm khởi đầu của quá trình

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 12 \\ 2 & 1 & 0 & 1 & 0 & 16 \\ \text{-40} & -30 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Giá trị âm nhất trong hàng cuối cùng đại diện thành phần có hệ số lớn nhất hay sức ảnh hướng lớn nhất đối với hàm mục tiêu. Đây là lý do ta chọn điểm khởi đầu dựa vào giá trị âm nhỏ nhất ở hàng cuối cùng. **Bước 5: Xác định thương số để chọn thành phần trục**
Dựa vào phần tử vừa được chọn ở bước 4, ta lấy giao của cột chứa phần tử đó với tất cả các hàng còn lại. Sau đó lấy các phần tử ở cột cuối cùng trong ma trận chia cho các phần tử giao này, gọi là thương số đại diện cho hàng. Trong ví dụ này hàng 1 sẽ có thương số là $12 : 1 = 12$ và hàng 2 là $16 : 8 = 2$. Sau đó chọn hàng có thương số đại diện nhỏ hơn cả làm phần tử trục

mà ở đây sẽ là phần tử hàng 2 cột 1.

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 12 \\ 2 & 1 & 0 & 1 & 0 & 16 \\ -40 & -30 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Phần tử trục (pivot element) cho phép ta có thể thêm hay bớt một lượng các đơn vị thành phần, như đã đề cập thì phương pháp này sẽ tìm kiếm từ điểm góc này đến điểm góc khác và phần tử trục cho phép ta làm việc đó dễ dàng. **Bước 6: Thực hiện các phép biến đổi sơ cấp trên ma trận này đến khi chỉ còn phần tử tại vị trí chọn được ở bước 5 là bằng 1**

$$\begin{bmatrix} 0 & \frac{1}{2} & 1 & -\frac{1}{2} & 0 & 4 \\ 1 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 8 \\ 0 & -10 & 0 & 20 & 1 & 320 \end{bmatrix}$$

Tại thời điểm này, giá trị của hàm mục tiêu đang có giá trị là 320 với $x_1 = 8$ và $x_2 = 0$
Bước 7: Nếu hàng dưới cùng vẫn còn phần tử âm,quay lại bước 4, nếu không đến bước 8
Trong bài toán này, do hàng cuối vẫn còn -10 nên ta tiếp tục từ bước 4 và được kết quả như sau:

$$\begin{bmatrix} 0 & 1 & 2 & -1 & 0 & 8 \\ 1 & 0 & -1 & 1 & 0 & 4 \\ 0 & 0 & 10 & 10 & 1 & 400 \end{bmatrix}$$

Do hàng cuối không còn phần tử âm, ta đến bước 8.
Bước 8: Lấy kết quả từ bảng Ta chọn ra các cột chỉ có một phần tử 1 và các phần tử còn lại bằng 0.

$$\left[\begin{array}{ccc|c} x_1 & x_2 & Z & C \\ 0 & 1 & 0 & 8 \\ 1 & 0 & 0 & 4 \\ 0 & 0 & 1 & 400 \end{array}\right]$$

Giải hệ phương trình mà ma trận như trên thu được, $x_1 = 4$, $x_2 = 8$, và $Z = 400$ là giá trị tốt nhất có thể đạt được của hàm sản xuất (8) với các ràng buộc (9), (10), (11).

4.3 Một số lưu ý khi sử dụng phương pháp đơn hình

- Với bài toán tìm hàm min (trong các hàm sản xuất thường cố gắng tìm max thay vì min) thì ta có thể đổi dấu hàm mục tiêu để làm vì hàm mục tiêu là hàm tuyến tính nên không ảnh hưởng các tính chất.
- Ở bước 5, khi thực hiện nếu không chọn được phần tử trục (hệ số của ràng buộc tuyến tính là âm, tức là có thể tìm được một số lớn tuỳ thích tại x_i tương ứng với cột đó) thì ta nói hàm mục tiêu không có ràng buộc trên.
- Ở bước 4, lần đầu tiên thực hiện nếu không thể tìm được phần tử âm nào tức là hàm mục tiêu có dạng $f(x) = k - \sum_{i=1}^{n} b_i x_i, \forall i, b_i \geq 0, x_i \geq 0$. Suy ra hàm có giá trị lớn nhất là k. Thoả điều kiện dừng ngay lập tức.

5 Sử dụng công cụ MATLAB để giải bài toán bằng phương pháp đại số tuyến tính

Tài liệu

[1] George Danzig. The Dantzig simplex method for linear programming. IEEExplore Vol.2 Issue 1 (1947).

[2] Danzig Linear Programming and Extension. Princeton University Press (1963).ISBN 781400884179

[3] Murty, Katta G. (1983). Linear programming. New York: John Wiley & Sons, Inc. pp. xix+482. ISBN 978-0-471-09725-9. MR 0720547.

[4] Adler, Ilan; Karmarkar, Narendra; Resende, Mauricio G.C.; Veiga, Geraldo (1989). "An Implementation of Karmarkar's Algorithm for Linear Programming". Mathematical Programming. 44 (1–3): 297–335. doi:10.1007/bf01587095.

[5] Rupinder Sekhon, Roberta Bloom (2016). "Applied Finite Mathematics Third Edition". Linear Programming: The Simplex Method: 115.

[6] Rupinder Sekhon, Roberta Bloom (2016). "Applied Finite Mathematics Third Edition". p. 84.