Operations Research Society of South Africa

Submitted for publication in ORiON

Operasionele Navorsings-vereniging van Suid-Afrika

An example for the LaTeX package `ORiONeng.sty`[*]

Authors' identities suppressed: Blind refereeing copy

Abstract

The format for papers published in ORiON is provided. Examples of how environments, such as tables, should be presented are also provided. The editorial format and standards of ORiON are maintained throughout this paper. The purpose is not to provide a mathematically correct or coherent content to this document, but rather to create a paper using specific examples of standard LaTeX elements.

Key words: Style sheet, examples, LaTeX

1 Introduction

This paper contains examples of several functions used in ORiON's LaTeX package. The purpose of this paper is not to provide readers with sensible content or proper logic. The theorems and definitions used in this paper may therefore appear not to make sense. The purpose is, however, not to provide the correct theorems and definitions, but rather to provide examples of these entities. The examples demonstrate how the standard environments and elements are created within the ORiON package. This paper also provides examples of the preferred format in which equations, references and other standard elements are handled in ORION.

The examples provided in this paper are not exhaustive. Should an author wish to include specific or exceptional situations in a paper, the business manager of ORiON can be contacted for guidance. Frequently asked questions will be addressed as examples in further updates of this paper.

There are various manuals available for LaTeX and TeX[2, 5]. Should these not provide sufficient guidance, the internet can also be used as a source of information. Text is entered into the source document in the same manner as for standard LaTeX documents. The environments in Table 1 have already been defined.

[*]This document is funded by ORSSA.

`algorithm`	`axiom`	`corollary`
`definition`	`example`	`heuristic`
`lemma`	`proposition`	`theorem`

Table 1: *Pre-defined environments in* `ORiONeng.sty`

2 Text

To solve LPs with the help of the simplex algorithm, knowledge of linear algebra is required. Amongst other things, the concept of linearly dependent vectors is important.

Definition 1 *If $\underline{a}_j$ with $j = 1, 2, 3, \ldots, r$ is a set of vectors, then the equation*

$$k_1\underline{a}_1 + k_2\underline{a}_2 + \ldots + k_r\underline{a}_r = \underline{0}$$

will have at least one solution, which is $k_1 = k_2 = \ldots = k_r = 0$. If this is the only solution, the set, $\underline{a}_j$, is called linearly independent. If other solutions exist, the set is called linearly dependent.

A set of column vectors, $\underline{a}_j$, with $j = 1, 2, 3, \ldots, k$, is linear independent or linearly dependent. Assume the set of vectors is linearly dependent. Under these circumstances it is possible to reduce the number of positive variables step by step until the columns which correspond with the remaining positive variables are linearly independent. If $\underline{a}_j$, with $j = 1, 2, 3, \ldots, n$, is linearly dependent, real numbers α_j (not all equal to zero) exist, so that

$$\sum_{j=1}^{k} \alpha_j \underline{a}_j = \underline{0}. \tag{1}$$

Equation (1) can be used to reduce an x_j, say x_r, in

$$\sum_{j=1}^{k} x_j \underline{a}_j = \underline{b}, \qquad x_j > 0 \tag{2}$$

to zero. Suppose any vector $\underline{a}_r$ of the k vectors in equation (1), for which $\alpha_r \neq 0$, can be expressed in terms of the remaining $k - 1$ vectors. Thus

$$\underline{a}_r = -\sum_{j \neq r} \frac{\alpha_j}{\alpha_r} \underline{a}_j. \tag{3}$$

Substitute equation (3) into equation (2), so that

$$\sum_{j=1}^{p} \left(x_j - x_r \frac{\alpha_j}{\alpha_r} \right) \underline{a}_j = \underline{b}. \tag{4}$$

This solution does not have more than $k-1$ non-zero variables. If $\underline{a}_r$ is selected arbitrarily, there would be no assurance that some of the variables may not perhaps be negative. If $\underline{a}_r$ is, however, selected carefully, the remaining $k - 1$ variables will still be non-negative. Select $\underline{a}_r$ so that

$$x_j - x_r \frac{\alpha_j}{\alpha_r} \geq 0 \quad j = 1, \ldots, p. \tag{5}$$

For any j, where $\alpha_j = 0$, equation (5) will automatically hold. If $\alpha_j \neq 0$, then equation (5), divided by α_j, will give one of the following inequalities:

$$\begin{aligned} \frac{x_j}{\alpha_j} - \frac{x_r}{\alpha_r} &\geq 0 \quad \text{if } \alpha_j > 0, \text{ or} && (6) \\ \frac{x_j}{\alpha_j} - \frac{x_r}{\alpha_r} &\leq 0 \quad \text{if } \alpha_j < 0. && (7) \end{aligned}$$

From (6) and (7) follows that $\frac{x_r}{\alpha_r}$ can not be greater than any $\frac{x_j}{\alpha_j} > 0$ and also not less than any $\frac{x_j}{\alpha_j} < 0$. If we choose the vector $\underline{a}_r$ so that

$$\frac{x_r}{\alpha_r} = \min_j \left\{ \frac{x_j}{\alpha_j}, \quad \alpha_j > 0 \right\}, \tag{8}$$

then all the variables in equation (4) will be non-negative. A feasible solution with no more then $k-1$ non-zero variables has been found. If this process of reduction is continued, the remaining x_j's will eventually be linearly independent and the solution will be a basic feasible solution.

The theorem which summarises the theory of the solution of LPs with the simplex algorithm, is the fundamental theorem of LP.

Theorem 2 (Fundamental theorem of linear programming (LP)) *This theorem consists of two parts:*

1. *If a feasible solution exists for an LP, a basic feasible solution exists.*
2. *If an optimal solution exists for an LP, an optimal basic feasible solution exists.*

Only a partial proof for Theorem 2 is presented.

Proof: Assume that a feasible solution exists with $k < n$ variables having, a positive value. Number these variables in such a way that the first k variables have a positive value. The solution can then be formulated as

$$\sum_{j=1}^{k} x_j \underline{a}_j = \underline{b}$$

with $x_j > 0$ where $j = 1, 2, \ldots, k$ and $x_j = 0$ for $j = k+1, \ldots, n$. The vectors $\underline{a}_j$ which are associated with the positive values can either be linearly independent or linearly dependent [6].

For a more detailed proof, see [4].

It is obviously much easier to solve LPs using software packages. A few of these programs are listed in Table 2.

Software	Advantages
LINDO & LINGO	Programmable
	Built-in mathematical functions
QM for Windows	User-friendly interface
	Interface with Excel

Table 2: *LP software with their respective advantages.*

3 Algorithms

The binary search algorithm starts its search in the centre of an alphabetical list of words. The algorithm compares the word in the middle of the list with the word that must be found (WRD) to test whether the middle word is, in fact, WRD. If this is not the case, it means that WRD is either after or before the word in the middle of the alphabetical list. If WRD comes alphabetically before the middle word, the word halfway between the first word and the middle word is tested to determine whether that word is WRD, while the other half (after the middle word) of the list is discarded. If WRD comes after the middle word, the word halfway between the middle and the last word is tested, while the first half of the list is discarded. This process of dividing the remaining list in half is repeated until the word is located or the list has been exhausted. This implies that only one half of the list has to be tested after the first step, one quarter of the list after the second step, one eighth of the list after the third step, etc. (The middle word is the word in the $\left\lfloor \frac{1+n}{2} \right\rfloor$-th position in the list.)

Algorithm 3 (Binary search algorithm) *Find a word in an alphabetical list of words by comparing the middle word with the word that must be found and by dividing the list, if required, into sublists.*

```
1. Let $\alpha$ and $\beta$ be the first and last positions of the sublist. Initially $\alpha = 1$ and $\beta = n$.
   Thus $\alpha \leftarrow 1$ and $\beta \leftarrow n$. WRD $\leftarrow$ “the word to be located”.
2. Repeat this step until $\alpha \leq \beta$, else go to 3.
    (a) Determine $k$, the middle position (between $\alpha$ and $\beta$).
            $k \leftarrow \left\lfloor \frac{\alpha + \beta}{2} \right\rfloor$
    (b) If $W(k)$ = WRD, stop. Output: $k$.
    (c) If WRD $> W(k)$, then $\beta \leftarrow k - 1$. Go to 2a.
    (d) If WRD $< W(k)$, then $\alpha \leftarrow k + 1$. Go to 2a.
3. Stop. Output: “Not in the list”.
```

Let us examine the (worst case) complexity of the binary search algorithm. Note that step 1 and step 3 are both performed only once. In both of these steps, a constant number of calculations are performed. This means that the complexity of these steps is $O(1)$. Let

$B(n)$ be the maximum number of times that step 2 must be repeated. After step 2 has been performed for the first time, the remaining list will have a maximum of $\lfloor \frac{n}{2} \rfloor$ elements to consider. This means that

$$B(n) = 1 + B\left(\left\lfloor \frac{n}{2} \right\rfloor\right). \tag{9}$$

We use induction to prove that

$$B(n) \leq 1 + \log_2 n \tag{10}$$

for each positive integer n. Equation (10) applies if $n = 1$. Accept that this result holds for an integer, k. We must then show that this inequality holds for $k + 1$. We therefore accept that

$$B(k) \leq 1 + \log_2 k. \tag{11}$$

From equation (9) we know that

$$B(k+1) = 1 + B\left(\left\lfloor \frac{k+1}{2} \right\rfloor\right),$$

but from equation (11) it follows that

$$B\left(\left\lfloor \frac{k+1}{2} \right\rfloor\right) \leq 1 + \log_2\left(\left\lfloor \frac{k+1}{2} \right\rfloor\right).$$

We also know that

$$\log_2\left(\left\lfloor \frac{k+1}{2} \right\rfloor\right) \leq \log_2\left(\frac{k+1}{2}\right).$$

This means that

$$\begin{aligned} B(k+1) &\leq 2 + \log_2\left(\frac{k+1}{2}\right) \\ &\leq 2 + \log_2(k+1) - \log_2(2) \\ &\leq 1 + \log_2(k+1), \end{aligned}$$

which gives the desired result that $B(n) = O(\log_2 n)$.

4 Matrices

The variable, x_3, enters the base with a value of $\frac{-20}{-1} = 20$, which is smaller than or equal to the upper bound. Thus a simplex iteration can be performed. The column below x_3 is given by

$$B_1^{-1} p_3 = \begin{bmatrix} -1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ 0 \end{bmatrix}.$$

The elementary matrix is given by

$$E_1 = \begin{bmatrix} \left(\frac{1}{-1}\right) & 0 & 0 \\ -\left(\frac{-1}{-1}\right) & 1 & 0 \\ -\left(\frac{0}{-1}\right) & 0 & 1 \end{bmatrix}.$$

We can then calculate

$$B_2^{-1} = E_1 B_1^{-1} = \begin{bmatrix} -1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 1 & -1 \\ 0 & 0 & -1 \end{bmatrix}.$$

After the iteration, the respective values are given by

$$A' = \begin{bmatrix} -1 & -1 & 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 1 & 0 \\ -1 & -1 & 0 & 0 & 0 & 1 \end{bmatrix} \quad B_2^{-1} = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 1 & -1 \\ 0 & 0 & -1 \end{bmatrix} \quad \underline{b} = \begin{bmatrix} -60 \\ 30 \\ -80 \end{bmatrix}$$

$$\begin{aligned} \underline{c}_{bv}(2) &= [-1, 0, -1] & \underline{c}_{nbv}(2) &= [-2, 0, 0] \\ \underline{x}_{bv}(2) &= [x_3, s_2, \widetilde{x}_2] & \underline{x}_{nbv}(2) &= [\widetilde{x}_1, s_3, \widetilde{s}_1]. \end{aligned}$$

Go to step 5.

Step 5:
The current right hand sides are given by

$$\underline{b}(2) = B_2^{-1}\underline{b} = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 1 & -1 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} -60 \\ 30 \\ -80 \end{bmatrix} = \begin{bmatrix} 20 \\ 50 \\ 80 \end{bmatrix}.$$

The basic variables x_3, s_2 and $\widetilde{x}_2$ have the values 20, 50 and 80 respectively. None of these variables occur at a value above their upper bounds.

Step 6:
All the right hand sides are positive, which means that an optimal solution has been found.

The optimal solution is given by

$$\begin{aligned} x_1 &= 80 - \widetilde{x}_1 = 80 - 0 = 80 \\ x_2 &= 80 - \widetilde{x}_2 = 80 - 80 = 0 \\ x_3 &= 20 \\ s_1 &= 0 - \widetilde{s}_1 = 0 - 0 = 0 \\ s_2 &= 50 \\ s_3 &= 0. \end{aligned}$$

The optimal objection function value is

$$\begin{aligned} \text{Objection function} &= c_1x_1 + c_2x_2 + c_3x_3 \\ &= (-2)(80) + (-1)(0) + (1)(20) \\ &= -140. \end{aligned}$$

References

[1] Chartrand G & Oellermann OR, 1993, *Applied and Algorithmic Graph Theory*, McGraw Hill, Singapore, pp 43–46

[2] Doob M, 1996, *TEX starting from 1*, Springer-Verlag, New York.

[3] Gass SI & Harris CM, 1996, *Encyclopedia of Operations Research and Management Science*, Kluwer Academic Publishers, Boston.

[4] Hadley G, 1962, *Linear programming*, Addison-Wesley Publishing Company Inc., Reading, Massachusetts.

[5] Lamport L, 1994, *LATEX A document preparation system: User's guide and reference manual*, Addison-Wesley Longman Inc., Reading, Massachusetts.

[6] Nickolson WK, 1995, *Linear algebra with applications*, third edition, PWS Publishing Company, Boston.