CA200 – Quantitative Analysis for Business Decisions

File name: CA200_Section_07_Linear Programming

Contents

7. Linear Programming

7.0 Introduction

Linear Programming (LP) is concerned with problems of allocation of scarce resources in the "best" possible manner. Typically,

EITHER • Maximise profit

OR • Minimise cost

Conditions and elements of a LP problem:-

1. **Decision variables** $\geq$ 0 (*usually*)
2. The "Best" criterion is described by a linear function of the decision variables. This is called the **Objective Function**
3. Rules (e.g. scarcity of resources) are described by linear functions of the decision variables. These are called the **Constraints** (or **Restrictions**)

Advantages

- Efficient solution methods available
- Data variation is easily handled (*comes under **"sensitivity analysis"***)
- Often possible to approximate non-linearities.

The classic method of solving linear programming problems is called the *Simplex Method*. However, we do not cover this in this course. Instead we lay the groundwork for:

- Formulation of linear programming problems
- Graphical solution of small 2-dimensional problems
- Possible software tools, such as ***R (***or ***AMPL*** if you want to give it a try***)*** to Solve some larger problems

Typical Areas of Application

• **Petro-chemical industries**, often now with an environmental focus

Investigating the Effect of Uncertainty on a LP Model of a Petrochemical Complex: Stability Analysis Approach , Al-Shammari A. (2011) **World Academy of Science, Engineering and Technology** 59 2011, 912-914 https://www.waset.org/journals/waset/v59/v59-176.pdf

A Mathematical Programming Model for optimum Economic Planning of the Saudi Arabian petrochemical industry, Alfares H.K. et al. (2002) The 6th Saudi Engineering Conference, KFUPM, Dhahran, December 2002, Vol. 4. 425-437
http://www.ccse.kfupm.edu.sa/~hesham/CP12_Petrochemical.pdf

• **Agri-food industries**

- farm planning
- feed mix
- product mix

e.g. Application of planning models in the agri-food supply chain: A review, European Journal of Operational Research, Volume 196, Issue 1, 1 July 2009, Pages 1-20 http://www.sciencedirect.com/science/article/pii/S0377221708001987

• **Distribution**

- warehouse location
- minimising transport costs

A Multi-Objective Mixed Integer Programming Model for Multi Echelon Supply Chain Network Design and Optimization
Paksoy T.et al. (2009) http://www3.iam.metu.edu.tr/iam/images/4/43/Preprint150.pdf

• **Public Services**

- water supply
- roads

e.g. A linear programming approach to water-resources optimization, Jacovkis P.M. et al. (1989) Zeitschrift für Operations Research, Volume 33, Issue 5, pages 341-362
http://www.springerlink.com/content/m15xwv065378777w/

7.1 Formulation of LP problems, with introductory examples

7.1.0 Three steps in formulation

There are three steps in formulation, of which Step 1 is the most crucial:

1. Identify decision variables (what can one *decide* about or *control*?)
2. Identify restrictions or constraints
3. Identify “best” criterion

7.1.1 Example 1: Product mix

The Handy-Dandy Company makes three types of kitchen appliances (*A*, *B* and *C*). To make each of these appliance types, just two inputs are required - labour and materials. Each unit of *A* made requires 7 hours of labour and 4 Kg of materials; for each unit of *B* made the requirements are 3 hours of labour and 4 Kg of materials, while for *C* the unit requirements are 6 hours of labour and 5 Kg of material. The company expects to make a profit of €40 for every unit of *A* sold, while the profit per unit for *B* and *C* are €20 and €30 respectively. Given that the company has available to it 150 hours of labour and 200 Kg of material each day, formulate this as a linear programming problem.

Formulation:

Step 1: Identify decision variables (what decisions can be made?)

For this production mix problem the company can decide on how many units of appliances A, B and C to make. Therefore, the **decision variables** are, say:

Let X_1 = number of units of appliance A to make per day

Let X_2 = number of units of appliance B to make per day

Let X_3 = number of units of appliance C to make per day

Step 2: Identify restrictions or constraints

Evidently, $X_1 \geq 0$, $X_2 \geq 0$, $X_3 \geq 0$ [*This is often the case but not always*]

Also, there are *limitations* (or *constraints*) on labour and materials:

Labour (per day): $7X_1 + 3X_2 + 6X_3 \leq 150$

Materials (per day): $4X_1 + 4X_2 + 5X_3 \leq 200$

Note: Here we have assumed *linearity* (scaling and additivity).

Step 3: Identify "best" criterion

It seems clear that *profit* should be *maximised*, that is the **objective function** is

$$\text{Maximise } 40X_1 + 20X_2 + 30X_3$$

Note: Here also we have assumed *linearity* (scaling and additivity).

7.1.2 Example 2: Inspection problem

A company employs *two grades* of quality control inspector to examine pieces being produced on a production line. A **grade one** inspector can inspect at the rate of **25 pieces per hour**, with **98% accuracy** and for this they are paid €16 per hour. **Grade two** inspectors inspect pieces at the slower rate of **15 per hour and with 95% accuracy**, and they are paid €12 per hour. The company can call on up to **8** grade one inspectors and up to **10** grade two inspectors, but inspectors who are not called upon do not have to be paid. Any **errors** which are made in the inspection process cost the company **€8** each. The company requires that **at least 1800** pieces must be inspected each day (**8 hours**). Formulate this as a linear programming problem.

Formulation:

Step 1: Identify decision variables (what decisions can be made?)

For this problem the company can decide on **how many of each grade of inspector** to call up. Therefore, the decision variables are, say,

Let X_1 = no. of grade 1 inspectors to call up **per day** (*appropriate time horizon here*)

Let X_2 = no. of grade 2 inspectors to call up per day

Step 2: Identify restrictions or constraints

We have, $0 \leq X_1 \leq 8$

$0 \leq X_2 \leq 10$

Also, there is the additional constraint,

No. of pieces (per day): $8\times25X_1 + 8\times15X_2 = 200X_1 + 120X_2 \geq 1800$

Step 3: Identify "best" criterion

Here the emphasis is on *costs*, which must be *minimised*. The costs are for labour and for (average) number of errors. Thus, **per day**,

Labour:	Inspectors grade 1:	$16X_1$ per hour ….or $128X_1$ per day
	Inspectors grade 2:	$12X_2$ per hour …..or $96X_2$ per day
Average no. errors:	Inspectors grade 1:	$0.02\times200X_1 = 4X_1 \Rightarrow 32X_1$ cost
	Inspectors grade 2:	$0.05\times120X_2 = 6X_2 \Rightarrow 48X_2$ cost

Therefore, the **objective** is to **minimise the total cost per day**, that is,

$$\text{Minimise } 160X_1 + 144X_2$$

Note 1: Here, we have assumed linearity (scaling and additivity) in establishing *both* the "No. of pieces" constraint and the objective function.

Note 2: This is really an *integer* LP problem as X_1 and X_2 must be whole numbers.

7.1.3 Example 3: A pair of related problems

Problem I: The following nutrient content and cost data are given for cereals A & B:

	Cereal A	**Cereal B**	**Mean Daily Req't (MDR)**
Thiamin	0.20 (mg/oz)	0.10 (mg/oz)	1.0 (mg)
Iron	0.80 (mg/oz)	1.20 (mg/oz)	7.2 (mg)
Price/ounce	2	5/3	

The problem is to **satisfy the MDR at minimum cost** by using some of Cereal A and some of Cereal B.

Problem II: A salesman has Thiamin and Iron in pill form and would like the customer to take pills instead of cereal to satisfy the MDR. The **salesman's objective** is to **maximise profit at a competitive price**.

Formulation of the problems in a "mathematical model":

Problem I: We must introduce **decision variables** to state the problem mathematically. What we ***want to decide on*** is the (unknown) amounts of Cereal A and Cereal B so we start by letting

y_1 = *number of ounces* the customer should use of Cereal A

y_2 = *number of ounces* the customer should use of Cereal B

Then the Total Cost is going to be $2y_1 + 5/3y_2$ so that **our objective (objective function)** is to

Minimise $z = 2y_1 + 5/3y_2$

The customer must also satisfy MDR for both Thiamin and Iron which translates to **satisfying the constraints**

$0.2y_1 + 0.1y_2 \geq 1$ or $2y_1 + y_2 \geq 10$ (Thiamin MDR)

$0.8y_1 + 1.2y_2 \geq 7.2$ or $8y_1 + 12y_2 \geq 72$ (Iron MDR)

Finally, we must have $y_1 \geq 0$ and $y_2 \geq 0$.

Problem II: In this case the salesman must ***decide on the price*** and this leads to introducing the **decision variables**

x_1 = price per mg of Thiamin

x_2 = price per mg of Iron

Then, assuming the customer buys no more than is required of each nutrient, the total price is going to be $1.0x_1 + 7.2x_2$ so that the **objective** (i.e **objective function**) is to

Maximise $w = x_1 + 7.2x_2$

In this case the **constraints** arise from needing to be "competitive" with the price for the nutrients taken in cereal form. For y_1 ounces of cereal A one gets $0.2y_1$ mg of Thiamin and $0.8y_1$ mg of Iron at a cost of $2y_1$; (from Cereals Table). The *corresponding cost* of these amounts of nutrients in pill form is $(0.2y_1)\,x_1 + (0.8y_1)x_2$ and so we must have the "competitive" **constraint**

$(0.2y_1)\,x_1 + (0.8y_1)\,x_2 \leq 2y_1$ or $0.2x_1 + 0.8x_2 \leq 2$

Similarly, for Cereal B, we have

$(0.1y_2)\,x_1 + (1.2y_2)\,x_2 \leq 5/3y_2$ or $0.1x_1 + 1.2x_2 \leq 5/3$

For convenience, we can *re-write the constraints* as

$2x_1 + 8x_2 \leq 20$

$x_1 + 12x_2 \leq 50/3$

Finally, we also have $x_1 \geq 0$ and $x_2 \geq 0$.

Note: Often called the *Primal* and *Dual* problems. This means that they are inter-related.

7.2 Graphical solution of simple problems; Some Terminology

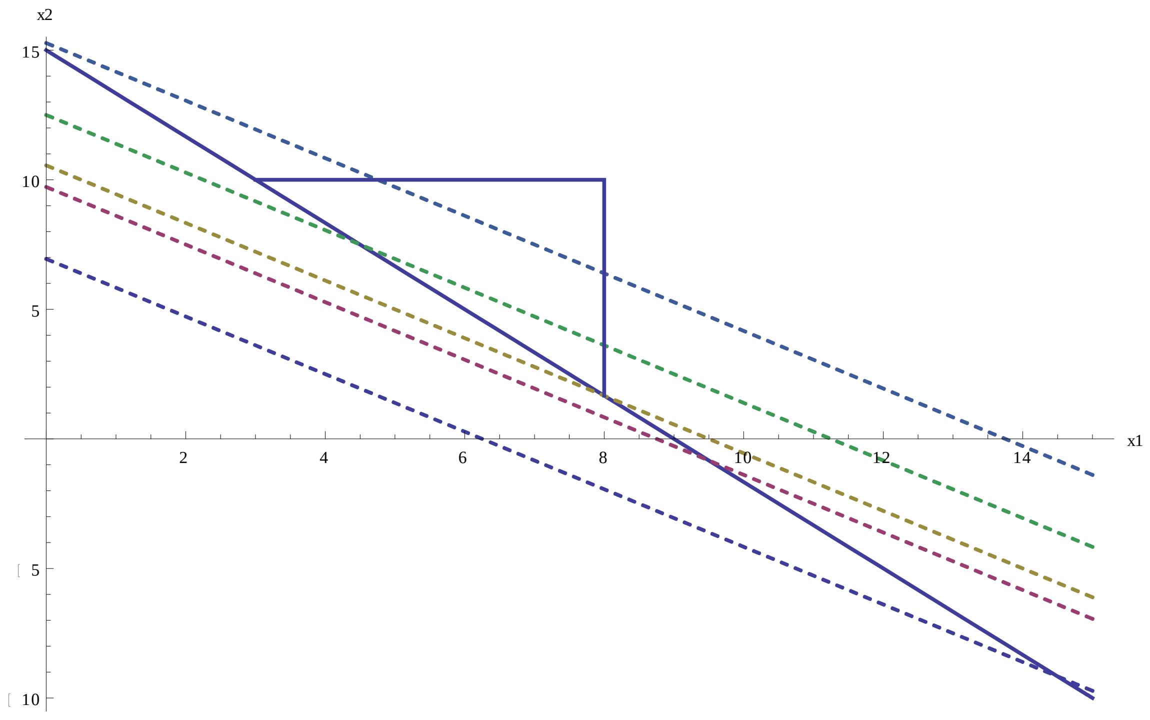
If a LP problem involves *just two* decision variables, as in Examples 2 and 3 of section 7.1.2, 7.1.3 then it can be solved graphically.

Example 2: Minimise $160X_1 + 144X_2$

subject to $200X_1 + 120X_2 \geq 1800$

and $0 \leq X_1 \leq 8$

$0 \leq X_2 \leq 10$



In this figure, the solid sloped line is $200X_1 + 120X_2 = 1800$ and we know that the solution must lie *on or above* this line. The *triangular region*, bounded by this line and the lines $X_1 = 8$ and $X_2 = 10$, is the set of ***FEASIBLE*** solutions of the problem, that is those solutions for which the *constraints are satisfied*.

The dashed lines, from higher to lower, are the lines

$160X_1 + 144X_2 = 2200$, $160X_1 + 144X_2 = 1800$, $160X_1 + 144X_2 = 1520$, $160X_1 + 144X_2 = 1400$ and $160X_1 + 144X_2 = 1000$.

These *illustrate the behaviour* of the objective function. Looking at the objective "Minimise $160X_1 + 144X_2$" we can see that the minimum *feasible solution* occurs with value 1520 when $X_1 = 8$ and $X_2 = 5/3$.

Note: This means that 8 (the maximum) Grade 1 inspectors and 5/3 Grade 2 inspectors should be called up. *In practice, a whole number is usual.*

Exercise on Example 3:

(a) Show that the corners of the feasible region for Problem I are (0,10), (3, 4) and (9,0) and that the minimum of z is 38/3.

(b) For Problem II show that the feasible region is bounded and that the maximum of w occurs at corner (20/3, 5/6) and that its value is (also) 38/3

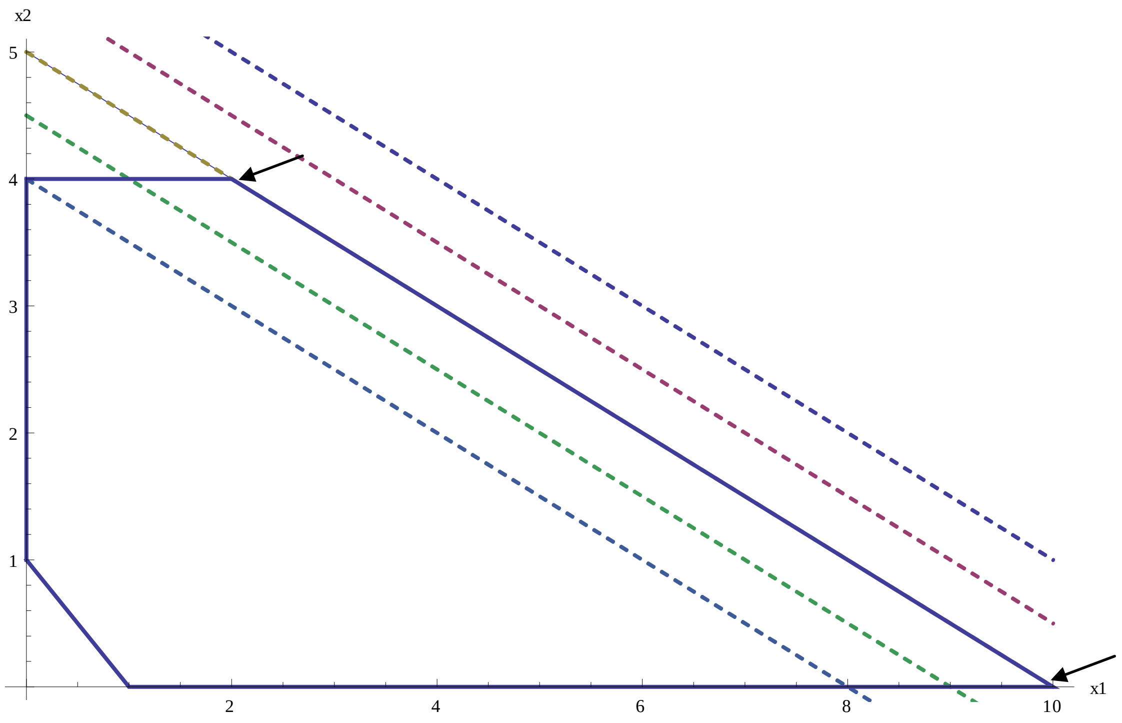
Some terminology:

A number of situations that may arise can be illustrated graphically, in superficially similar examples, as follows.

(1) Alternative/Multiple Optimal Solutions

Consider the problem

Maximise	$X_1 + 2X_2$
subject to	$2X_1 + 4X_2 \leq 20$
	$X_1 + X_2 \geq 1$
	$X_2 \leq 4$
and	$X_1, X_2 \geq 0$



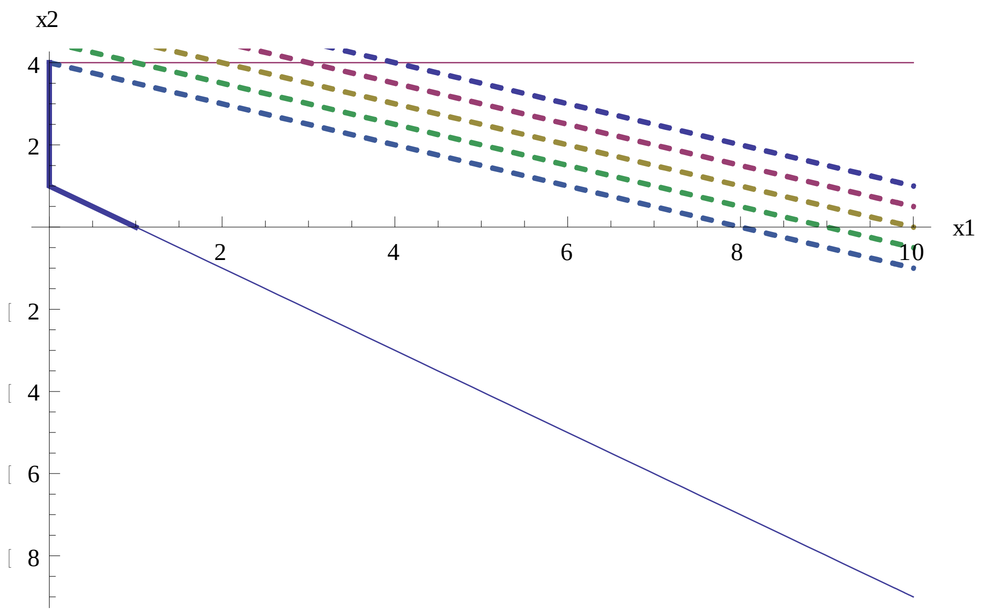
The ***feasible region*** is on the boundary and inside the polygon. The dashed lines indicate the behaviour of the objective function, so the lowest line shown e.g. is X_1 +

$2X_2 = 8$. As $X_1 + 2X_2$ = some Constant and $2X_1 + 4X_2 = 20$ are parallel, the maximum is at any point on the line joining (2,4) to (10,0) and has value 10.

(2) Unbounded Solutions

Consider the problem (same as previous except one constraint is omitted)

Maximise	$X_1 + 2X_2$
subject to	$X_1 + X_2 \geq 1$
	$X_2 \leq 4$
and	$X_1, X_2 \geq 0$



Constraints are as shown in bold. A whole set of possible realisations of the objective function apply, but are not subject to any overall constraint to the right. The *feasible region* is clearly unbounded (which might mean an error in the formulation – or decision basis)!

(3) Infeasible Problem

Consider the problem (similar to previous two but with two inequality signs reversed)

Maximise	$X_1 + 2X_2$
subject to	$X_1 + X_2 \leq 1$
	$X_2 \geq 4$
and	$X_1, X_2 \geq 0$

The diagram of (2) can be referred to except that in this case a solution must both lie in the small triangular region near the origin *and above* the line $X_2 = 4$, which is impossible. Again, probably means an error in the formulation, emphasising its importance.

7.3 Large scale problems – Tool usage (R, AMPL etc.

In practice, the real difficulties in LP are in formulation rather than in solving as there are various ‘solver tools’ available; (essentially part of Numerical Analysis).

7.3.1 Sketch of *typical* modelling language and approach

A typical sequence of events for a practical linear programming application is summarised as follows:

(a) **Formulate a model**, the abstract system of variables, objectives, and constraints that represent the general form of the problem to be solved.

(b) **Collect data** that define a specific problem instance.

(c) Generate a **specific objective function** and constraint equations from the model and data.

(d) **Solve** the **problem instance** by running a program, or *solver*, to apply an algorithm that finds optimal values of the variables.

(e) **Analyse** the results.

(f) **Refine** the **model** and **data** as necessary, and **repeat**.

This ‘solver’ sequence has a somewhat different emphasis than our examples to date, with its separation of general model (a) and specific data (b) & (c).
Regarding (d), marketed solvers are typically black boxes; can not modify much.
Step (e) could well include “sensitivity analysis” which we have already mentioned.
Basically, it means looking at slightly different data values to see how sensitive (or changeable) the solution is from set to set. Finally, (f) is a normal step in any modelling or simulation exercise and the use of a tool makes it relatively simple to do this (and the (e) step), whereas re-doing calculations manually would be time-consuming and error-prone.

Software Choices

R is a scripting language, with which you are now familiar. It is designed around a true computer language and allows users to add to functionality by defining new functions. Much of the system is written also in R, which makes it easy for users to follow the algorithmic choices made. For computationally-intensive tasks, C, C++ and Fortran code can be linked and called at run time. Advanced users can write C code to manipulate R objects directly. It is not strictly a *statistics system*, though many people describe it as such, rather it is an environment within which statistical and other mathematical techniques can be implemented. Extensions are possible via a number of *package* add-ons.

AMPL is an algebraic modelling language. If giving the latter a try, students should just aim to familiarise themselves with whatever features they may need for this LP section.

Note: The Tool or Language for LP commands or other analyses is ***not*** prescribed for the module examination. You can give an example or answer in R, AMPL or other**.**

N.B. The AMPL sub-sections 7.3.2 and 7.3.3 here will *only* be relevant to those deciding to give AMPL a try. Otherwise, Problem Set-up using R will be covered in Practicals and also relies on the *three steps* – Section 7.1 and the *general principles*, as outlined in the previous sub-section 7.3.1. References in Section 7.4 below are general and may be of use for further reading on LP and Operations Research.

7.3.2 Introduction to AMPL

If you would like to try the linear programming tool ***AMPL***, the School computer labs have a version installed, but a student version is freely downloadable from the site http://www.ampl.com/ In fact, this site is quite informative and students should make use of it, if planning to try out this option.

AMPL Example 1:

A simple two-variable example introduced in Section 1.1 of the ***AMPL*** book (on-line) is formulated in the book (on-line – ***check***) and its mathematical statement is

Maximize $25\,X_B + 30\,X_C$

Subject to

$(1/200)\,X_B + (1/140)\,X_C \leq 40$

$0 \leq X_B \leq 6000$

$0 \leq X_C \leq 4000$

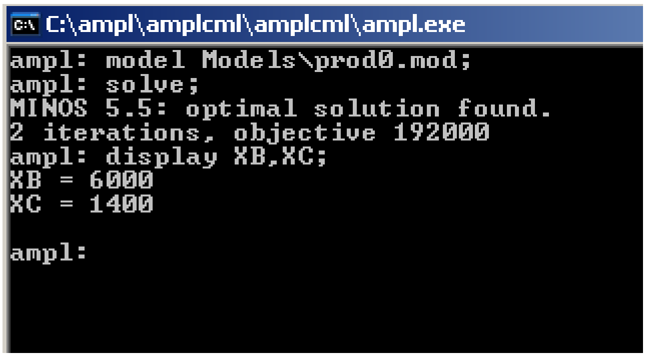
The problem has to do with how to allocate next week's time in a steel plant between producing bands (B) and coils (C). X_B and X_C represent, respectively, the number of tons of bands and coils to produce.

The simplest use of ***AMPL*** in solving this problem (*which actually mixes up steps (a) to (c) above, almost ignoring step (a) in fact, so could do it with Notepad*)) is to create a file (usually) of type ".mod" as follows. The actual file used is "prod0.mod" (included in the ***AMPL*** download) and its contents are,

```
var XB;
var XC;
maximize Profit: 25 * XB + 30 * XC;
subject to Time: (1/200) * XB + (1/140) * XC <= 40;
subject to B_limit: 0 <= XB <= 6000;
subject to C_limit: 0 <= XC <= 4000;
```

This is in ***AMPL*** syntax but its attraction is that it is a very direct "translation" from the ordinary mathematical statement. The use of ";" and ":" as separators and some key words is clearly apparent. As summarised in the ***AMPL*** book (p5 – on-line), i.e.:
"*The* ***AMPL*** *linear program that you type into the file parallels the algebraic form in every respect. It specifies the decision variables, defines the objective, and lists the constraints. It differs mainly in being somewhat more formal and regular, to facilitate computer processing. Each variable is named in a* ***var*** *statement, and each constraint by a statement that begins with* ***subject to*** *and a name like Time or B_limit for the constraint. Multiplication requires an explicit * operator, and the ≤ relation is written <=.*"

The next step is to run the model (see page 5 of ***AMPL*** book – on-line) which is done using a small number of ***AMPL*** commands put in bold italics for emphasis (***model prod0.mod*** (N.B. – see note on file paths - below), ***solve***, ***display XB, XC***, ***quit***). The solver invoked by 'solve' is a default one unless otherwise specified. When the ***AMPL*** command line is started up (by running the application called ampl) the prompt "ampl:" appears. We have



```
C:\ampl\amplcml\amplcml\ampl.exe
ampl: model Models\prod0.mod;
ampl: solve;
MINOS 5.5: optimal solution found.
2 iterations, objective 192000
ampl: display XB,XC;
XB = 6000
XC = 1400

ampl:
```

To finish type ***quit***.

<u>Note</u>: It is necessary to be clear about setting file paths. In the above, the "ampl" application is in folder "amplcml" and file "prod0.mod" is in a sub-folder of this called "Models"; thus, "Models\prod0.mod" is relative to "amplcml".

AMPL Example 2: As a second example we take a problem that we previously considered, which has the formulation:

Maximise $40X_1 + 20X_2 + 30X_3$

subject to

<u>Labour (per day)</u>:	$7X_1 + 3X_2 + 6X_3 \leq 150$
<u>Materials (per day)</u>:	$4X_1 + 4X_2 + 5X_3 \leq 200$

and

$X_1 \geq 0,\ X_2 \geq 0,\ X_3 \geq 0$

As for **AMPL Example 1**, the simplest use of ***AMPL*** in solving this problem is to create a file of type ".mod" as follows. The actual file created is "EX_000.txt" which is located on, say, C drive of computer.

```
ampl - HJR                              [contains "EX_000.txt"]
      - amplcml                         [contains "sw.exe" & ampl.exe]
                  - amplcml
                              - MODELS  [contains "prod0.mod" of Example 1]
                              - TABLES
```

The contents of "EX_000.txt" are,

```
var X1;
var X2;
var X3;
maximize Profit: 40*X1 + 20*X2 + 30*X3;
subject to Labour   : 7*X1 + 3*X2 + 6*X3 <=150;
subject to Materials: 4*X1 + 4*X2 + 5*X3 <=200;
subject to 1_limit:0<=X1;
subject to 2_limit:0<=X2;
subject to 3_limit:0<=X3;
```

The next step is to run the model. We have –as an alternative to the screen shot

```
C:\ampl\amplcml\ampl.exe

ampl: model c:\ampl\HJR\EX_000.txt;
ampl: solve;
MINOS 5.5: optimal solution found.
2 iterations, objective 1000
ampl: display X1,X2,X3;
X1 = 0
X2 = 50
X3 = 0

ampl:
```

To finish type ***quit***.

Note: filepaths!

7.3.3 Interfaces and Getting started with AMPL

AMPL provides various interfaces (see section 1.7 of book – on-line) but we just focus on the basic command line interface. Students may prefer to find out about and use one of the available GUIs.

AMPL Example 3: Production Models: Maximising profits (Ch1, *AMPL* book)

This example is taken from Chapter 1 of the ***AMPL*** book (Fourer et al, Second edition). This chapter is available on-line, if required. In this section, we just list some key points ; (go to the book chapter for details).

(i) This is AMPL Example 1 of sub-section 7.3.2 (above). We had

Math. Formulation	**AMPL** (file prod0.mod)
Maximize $25\,X_B + 30\,X_C$	var XB;
Subject to	var XC;
$(1/200)\,X_B + (1/140)\,X_C \leq 40$	maximize Profit: 25 * XB + 30 * XC;
$0 \leq X_B \leq 6000$	subject to Time: (1/200) * XB + (1/140) * XC <= 40;
$0 \leq X_C \leq 4000$	subject to B_limit: 0 <= XB <= 6000;
	subject to C_limit: 0 <= XC <= 4000;

For present purposes we need to explain the context – ***AMPL*** book page 2.

In brief, X_B = number of tons of steel bands to be produced

X_C = number of tons of steel coils to be produced

One can produce 200 tons of bands per hour and 140 tons of coils so that it will take $X_B/200$ and $X_C/140$ hours to produce X_B and X_C tons, respectively. There is an overall limit of 40 hours.

(ii) ***AMPL*** book then (page 6) generalises the problem to

Given: P, a set of products

a_j = tons per hour of product j, for each $j \in P$

b = hours available at the mill

c_j = profit per ton of product j, for each $j \in P$

u_j = maximum tons of product j, for each $j \in P$

Define variables: X_j = tons of product j to be made, for each $j \in P$

Maximize:

$$\sum_{j \in P} c_j X_j$$

Subject to:

$$\sum_{j \in P} (1/a_j)\, X_j \leq b$$

$$0 \leq X_j \leq u_j, \text{ for each } j \in P$$

The components of this are sets, parameters, variables, **objective** and **constraints**.
These appear in the corresponding ***AMPL*** file prod.mod:

```
set P;
param a {j in P};
param b;
param c {j in P};
param u {j in P};
var X {j in P};
maximize Total_Profit: sum {j in P} c[j] * X[j];
subject to Time: sum {j in P} (1/a[j]) * X[j] <= b;
subject to Limit {j in P}: 0 <= X[j] <= u[j];
```

The various points of ***AMPL*** syntax introduced in this are explained in the book including key words set, param, var, maximise, subject to and sum. Also, the indexing expression {j in P} meaning “for each j in P”.
Another point to note is that the above does not contain any data values, (unlike our very first examples). Instead, ***AMPL*** allows separation of *general model* from *specific data*. The data can be put in a separate file (say prod.dat) such as

```
set P := bands coils;
param:    a     c     u     :=
bands     200   25    6000
coils     140   30    4000 ;
param b := 40;
```

Refer to book for how to execute this.

7.4 Some References

There are many books on Operations Research in general, and on Linear Programming in particular in DCU library or available on-line. The books by Taha and Luenberger are well known, especially.
Others include

- “Linear Programming” V. Chvatal (Freeman, 1986) [***in library***]
- “Model Building in Mathematical Programming” by H. P. Williams [***in library***]

- "Applied Mathematical Programming" Bradley, Hax, and Magnanti, [***Available in electronic form***]
- "The Traveling Salesman Problem: A Computational Study" by David L. Applegate, R. E. Bixby, V. Chvátal & W. J. Cook (2007) [***in library***]
- "Operations Research" F. S. Hillier, Gerald J. Lieberman. (1974) [***in library***]
- "Combinatorial Optimization: Algorithms and Complexity" C.H. Papadimitriou and K. Steiglitz [***in library***]
- "Urban Operations Research" Larson and Odoni. [***in library***]
- "Deterministic Operations Research" (2010) D. Rader [***in library***]
- "Serious Play" M. Schrage [***in library]***
- "The Fifth Discipline" P. Senge [***in library***]
- "The Predictioneer's Game" by B. Bueno de Mesquita[***in library***]
- "Optimization algorithms for Networks and Graphs" E. Minieka. [***in library***]
- "Operations Research: An Introduction" H. Taha [***in library, various editions***]
- "Elements of statistical learning" (*"If, by OR, you also want to include modern-day and the future that is data-driven analytics, then [this] is a must .."*) [***Available in electronic form***]
- "Linear Programming and Network Flows" M. S. Bazaraa and J. J. Jarvis {Appears to be downloadable for free, as are several others with similar titles}.
- AMPL: A Modeling Language for Mathematical Programming [Hardcover], Fourer, Gay, Kernighan (2002) **[*in DCU library but at least some of it is available on line as is the related software*]**

This list is intended to give an idea of what is available. Some of the books are straightforward technical books while some are more for the general reader or as background (e.g. - Serious Play by Schrage, The Predictioneer's Game by Bueno de Mesquita). Taha's book is a good general source and there is a related piece of software (***TORA***) on the School's lab machines, if you want to check it out. It is recommended especially to go to the web-site associated with the ***AMPL*** book.
Finally, be aware that there is a lot of useful material elsewhere on-line.