Detailed Notes – **Part 1**

Introduction to Mathematical Optimisation

1 What is Mathematical Optimisation?

Mathematical optimisation (alternatively, mathematical programming) is the is a field of applied mathematics that is concerned with solution of quantitative problems. Mathematical programming includes linear programming (LP), nonlinear programming (NLP) and integer programming (IP). Please note, that "programming" does not specifically refer to computer programming – indeed, these terms were first used before computer programming (as we know it today) really existed. However, producing an algorithm that could be used by a computer is the ultimate aim. We will look in depth at the mathematics of mathematical programming, and hopefully gain a good understanding of the methods used to solve them.

The general form of the mathematical programming problems is shown as follows:

$$\begin{array}{rcll} \text{max (or min)} & z = f(\mathbf{x}) & & \\ \text{subject to} & \mathbf{g}(\mathbf{x}) & \leq \text{(or } \geq\text{)} & \mathbf{0}, \\ & \mathbf{x} & \geq & \mathbf{0}. \end{array}$$

The bold type for $\mathbf{x}$ and $\mathbf{g}(\cdot)$ is entirely deliberate – it indicates that they are **vectors**. For example, a maximisation problem with four *decision variables* and three *constraints* would look like:

$$
\begin{array}{llcl}
\max & z = f(x_1, x_2, x_3, x_4) & & \\
\text{s.t.} & g_1(x_1, x_2, x_3, x_4) & \leq & 0, \\
& g_2(x_1, x_2, x_3, x_4) & \leq & 0, \\
& g_3(x_1, x_2, x_3, x_4) & \leq & 0, \\
& x_1, x_2, x_3, x_4 & \geq & 0.
\end{array}
$$

We refer to function f as the *objective function*.

The set $\mathbf{S} = \{\mathbf{x} : \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \text{ and } \mathbf{x} \geq \mathbf{0}\}$ is called the *feasible region*. Note that over this set we have

$$
z = \max_{\mathbf{x} \in \mathbf{S}} f(\mathbf{x}) = -\min_{\mathbf{x} \in \mathbf{S}} \big(-f(\mathbf{x}) \big)
$$

i.e. any maximisation problem can be written as a counterpart minimisation version, and vice versa.

The general approach to solving an optimisation problem is first to *formulate* the problem by expressing the objective and constraints in terms of the decision variables. Formulation usually commences with a clear definition of the decision variables, the variables over which the decision maker has control. It is then usual to express the objective, the quantity to be maximised or minimised, in terms of the decision variables. The constraints and/or the restrictions on the values that the decision variables can take are then constructed. For the case when the objective and all constraints are *linear*, the problem is referred to as a *linear program*.

Then we try to solve the formulated LP problem. In the case of an LP with only two decision variables, a graphical solution approach, which will be introduced in Section 2.2, would be sufficient to yield an optimal solution. For LP problems with a greater number of decision variables, numerical techniques such as the Simplex method or its variants are used. These are usually implemented in software packages, such as LINDO, LINGO, Excel, and CPLEX to name but a few. Other types of mathematical programs could also be solved using the Solver Add-in in Excel and some of the above optimisation packages.

1.1 Examples of Applications

Some simple practical applications of mathematical optimisation are :

- Blending petroleum products: the problem is to choose which end products to produce from some raw products; the objective is to maximise the profit, subject to constraints on quantity of components of petroleum available (LP).
- Portfolio design: the problem is to choose which assets to hold; the objective is to maximise the expected return subject to constraints on maximum levels of risk acceptable (NLP).
- Aircrew scheduling: the problem is to assign crew to flights; the objective is to minimise the cost to an airline (wages plus accommodation expenses, etc) subject to all flights having the required crew, all crew returning to their home bases, and all union and legal requirements on work schedules met (IP).

1.2 Examples of Mathematical Programs

Below are some simple examples of mathematical programs including LP, NLP, and IP.

$$\begin{array}{rrcl} \max & 2x_1 + 3x_2 & & \\ \text{s.t.} & x_1 + 3x_2 & \leq & 6 \\ & 3x_1 + 5x_2 & \leq & 15 \\ & x_1, x_2 & \geq & 0 \end{array}$$

$$\begin{array}{rrcl} \min & x_1^3 + 2x_1^2 x_2 + 5x_2^4 & & \\ \text{s.t.} & 2x_1 + x_2 & \geq & 4 \\ & 3x_1^2 + 2x_2 & \geq & 5 \\ & x_1, x_2 & \geq & 0 \end{array}$$

$$\begin{array}{rrcl} \max & x_1 - 2x_1 x_2 + 5x_2 & & \\ \text{s.t.} & x_1 + x_2 & \leq & 4 \\ & x_1, x_2 & \geq & 0, \quad x_1, x_2 \quad \text{integer} \end{array}$$

1.3 Complications

Consider the following NLP

$$\begin{array}{rrcl} \max & f(x) & = & x^3 - 3x \\ \text{s.t.} & x & \leq & 2, \\ & x & \geq & 0. \end{array}$$

On the feasible interval $0 \leq x \leq 2$, $f(x)$ has only one stationary point (where $f'(x) = 0$), which is $x_0 = 1$. Hence the required value can be found by comparing the value $f(1)$ with the values at the two end points, $x_1 = 0$ and $x_2 = 2$ (two extreme points of the interval $[0, 2]$), i.e.

$$\begin{aligned} \max_{x \in [0,2]} f(x) &= \max\{f(0), f(1), f(2)\} \\ &= \max\{\ 0, -2, 2\ \} = f(2) = 2. \end{aligned}$$

So, for mathematical optimisation problems, why can't we just check all the extreme points (a generalisation of the end points in the 1-D instances) and the stationary points?

Firstly, solving $f'(x) = 0$ or finding the extreme points could not be trivial if the objective function or the feasible set is "ugly"? Another even bigger problem that we come up against is that there could be a large number of these extreme points. Consider a 2-D problem where the feasible set is a square. Then there exist 4 extreme points. If $\mathbf{x}$ is a 3-D vector, we might have a cube as the feasible region, which has 8 extreme points. It is common that real-world LP problems have thousands of variables. Then they could have more than $2^{1000} \approx 1.07 \times 10^{301}$ extreme points!

So the kernel is how to find an optimal solution without a full enumeration of extreme points. Let's start from LP, where the stationary points are unnecessary to be taken into consideration due to the linearity of the objective function. To solve an LP, we obviously need to identify which extreme point might be a good solution, perhaps stepping from one to another in the direction that will improve the objective function value. Then we can be guided to approach an optimal solution. This is, actually, the basic concept of the Simplex method.

Before introducing LP and the Simplex method, we begin with a brief revision of fundamental linear algebra.

2 Revision of Basic Linear Algebra

2.1 Vectors and Matrices

Linear algebra is in some sense the "language" of LP in particular, and mathematical optimisation in general. It is a very important shorthand for the problems in higher dimensions that we will encounter.

Vector: A vector of dimension n is an ordered collection of n elements, which are called *components*.

Consider an n-dimensional variable vector

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}.$$

Vector $\mathbf{x}$ is called non-negative, denoted by $\mathbf{x} \geq \mathbf{0}$, if we have $x_i \geq 0$ for each $i \in \{1, 2, \cdots, n\}$.

In this subject, vectors are written by default as column vectors, unless otherwise specified. When we want a row vector, we write, for example

$$\mathbf{c}^T = (c_1, c_2, \ldots, c_n),$$

which is called $\mathbf{c}$ *transpose*.

If two vectors have the same dimension, then we can take the *dot product* (or *inner product*), which gives us a *scalar*. This can be written in a few ways:

$$\mathbf{c} \cdot \mathbf{x} = \mathbf{c}^T\mathbf{x} = c_1x_1 + c_2x_2 + \ldots + c_nx_n = \sum_{k=1}^{n} c_kx_k.$$

Mostly, the notation $\mathbf{c}^T\mathbf{x}$ will be adopted in this subject.

Matrix: A matrix is an array of numbers, for example,

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

is an $m \times n$ matrix (m rows and n columns).

For each $j \in \{1, 2, \cdots, n\}$, let $\mathbf{A}_j$ be the vector of the j^{th} column of matrix $\mathbf{A}$, i.e.

$$\mathbf{A}_j = \begin{pmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{mj} \end{pmatrix}.$$

Then matrix $\mathbf{A}$ can be written in the form

$$\mathbf{A} = (\mathbf{A}_1,\ \mathbf{A}_2,\ \cdots\ \mathbf{A}_n).$$

Matrices can also be transposed. The transpose of $\mathbf{A}$ is denoted by $\mathbf{A}^T$, is given by

$$\mathbf{A}^T = \begin{pmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{pmatrix},$$

which is an $n \times m$ matrix (n rows and m columns).

A matrix can be multiplied by another one if the number of columns in the first one is identical to the number of rows in the second one. If $\mathbf{A}$ is an $m \times n$ matrix, and $\mathbf{B}$ is an $n \times p$ matrix, then $\mathbf{AB}$ is an $m \times p$ matrix, given by

$$\mathbf{AB} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} + \cdots + a_{1n}b_{n1} & \cdots & a_{11}b_{1p} + a_{12}b_{2p} + \cdots + a_{1n}b_{np} \\ a_{21}b_{11} + a_{22}b_{21} + \cdots + a_{2n}b_{n1} & \cdots & a_{21}b_{1p} + a_{22}b_{2p} + \cdots + a_{2n}b_{np} \\ \vdots & & \vdots \\ a_{m1}b_{11} + a_{m2}b_{21} + \cdots + a_{mn}b_{n1} & \cdots & a_{m1}b_{1p} + a_{m2}b_{2p} + \cdots + a_{mn}b_{np} \end{pmatrix}.$$

Each element of the new matrix, $(\mathbf{AB})_{ij}$, is the dot product of the i^{th} row of $\mathbf{A}$ and the j^{th} column of $\mathbf{B}$. Note that, in general $\mathbf{AB} \neq \mathbf{BA}$.

An n-dimensional (column) vector can be regarded as an $n \times 1$ matrix, and its transpose is a $1 \times n$ matrix. Hence $\mathbf{c}^T\mathbf{x}$ is just a special case of matrix multiplication.

We will also use the product of a matrix and a vector as shown below frequently in this subject.

$$\mathbf{Ax} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{pmatrix}$$

$$= \begin{pmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{pmatrix}$$

One of the ways we use matrices is to abbreviate a whole list of linear inequalities, for example, we will write $\mathbf{Ax} \leq \mathbf{b}$, where $\mathbf{A}$ is an $m \times n$ matrix, $\mathbf{x}$ is an n-dimensional vector and $\mathbf{b}$ is an m-dimensional vector. This is equivalent to

$$\begin{array}{lcl} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \leq & b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & \leq & b_2 \\ & \vdots & \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & \leq & b_m \end{array}$$

We can solve a system of linear equations $\mathbf{Ax} = \mathbf{b}$ by Gaussian elimination (By using elementary row operations, the augmented matrix is reduced to row echelon form). It may have no solution, a unique solution, or an infinite number of solutions.[1]

[1] Please see p.29–32 in "Operations Research: Applications and Algorithms" (Winston, 2004).

Consider that $\mathbf{B}$ is a *square matrix* (say $m \times m$). Then its inverse matrix $\mathbf{B}^{-1}$ may exist, and

$$\mathbf{B}\mathbf{B}^{-1} = \mathbf{B}^{-1}\mathbf{B} = \mathbf{I},$$

where $\mathbf{I}$ is an $m \times m$ *identity matrix*.

If $\mathbf{B}^{-1}$ exists, then the system of equations $\mathbf{Bx} = \mathbf{b}$ has a unique solution which can be obtained by

$$\mathbf{x} = \mathbf{B}^{-1}\mathbf{b}.$$

In this case, we can use Gaussian-Jordan elimination (By using elementary row operations, the augmented matrix is reduced to reduced row echelon form). To solve $\mathbf{Bx} = \mathbf{b}$ by this method, we set up an *augmented matrix* $(\mathbf{B}|\mathbf{b})$ and use *elementary row operations* (EROs) to reduce the left part of the augmented matrix to the identity matrix.

The three EROs are:

1. Multiply one row by a constant.
2. Add a multiple of one row to another row.
3. Swap two rows.

This takes us from

$$(\mathbf{B}|\mathbf{b})$$

to

$$(\mathbf{B}^{-1}\mathbf{B}|\mathbf{B}^{-1}\mathbf{b}) = (\mathbf{I}|\mathbf{B}^{-1}\mathbf{b}).$$

So the solution is revealed in the right hand side of the resulting augmented matrix.

Consider the following example. To solve the system of equations

$$\begin{aligned} x_1 + 2x_2 + x_3 &= 5, \\ 2x_1 + 4x_2 &= 6, \\ x_1 + 3x_2 &= 6, \end{aligned}$$

we have the matrix form

$$\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 0 \\ 1 & 3 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 5 \\ 6 \\ 6 \end{pmatrix}.$$

So its augmented matrix $(\mathbf{B}|\mathbf{b})$ is

$$\left(\begin{array}{ccc|c} 1 & 2 & 1 & 5 \\ 2 & 4 & 0 & 6 \\ 1 & 3 & 0 & 6 \end{array}\right).$$

Then we conduct the EROs as follows:

$$\left(\begin{array}{ccc|c} 1 & 2 & 1 & 5 \\ 0 & 0 & -2 & -4 \\ 0 & 1 & -1 & 1 \end{array}\right) \begin{array}{l} \\ R_2' \leftarrow R_2 - 2R_1 \\ R_3' \leftarrow R_3 - R_1 \end{array}$$

$$\left(\begin{array}{ccc|c} 1 & 2 & 1 & 5 \\ 0 & 1 & -1 & 1 \\ 0 & 0 & -2 & -4 \end{array}\right) \begin{array}{l} \\ R_2'' \leftarrow R_3' \\ R_3'' \leftarrow R_2' \end{array}$$

$$\left(\begin{array}{ccc|c} 1 & 2 & 1 & 5 \\ 0 & 1 & -1 & 1 \\ 0 & 0 & 1 & 2 \end{array}\right) \begin{array}{l} \\ \\ R_3''' \leftarrow -\frac{1}{2}R_3'' \end{array}$$

$$\left(\begin{array}{ccc|c} 1 & 2 & 0 & 3 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 2 \end{array}\right) \begin{array}{l} R_1'''' \leftarrow R_1''' - R_3''' \\ R_2'''' \leftarrow R_2''' + R_3''' \\ \\ \end{array}$$

$$\left(\begin{array}{ccc|c} 1 & 0 & 0 & -3 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 2 \end{array}\right) \begin{array}{l} R_1''''' \leftarrow R_1'''' - 2R_2'''' \\ \\ \\ \end{array}$$

So the solution is

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -3 \\ 3 \\ 2 \end{pmatrix}.$$

The matrix inverse can be found by a similar process. Note, though, that finding solutions or finding the inverse can take a substantial amount of work!

2.2 Graphical Solution of LP Problems

If we have an LP with just two variables, then we can solve it by drawing a 2-D diagram. In particular, we draw each of the lines corresponding to the constraints, e.g. for the constraint $3x_1 + 2x_2 \leq 120$, we draw the line $3x_1 + 2x_2 = 120$. Consider the following example

$$\begin{aligned} 3x_1 + 2x_2 &\leq 120, \\ x_1 + x_2 &\leq 50, \\ x_1, x_2 &\geq 0. \end{aligned}$$

After drawing each constraint line in 2-D diagram, we then shade the feasible region (it's usually easiest to check whether the origin is on the feasible side or not). Then the feasible region with the following four extreme points $(0,0), (0,50), (40,0)$ and $(20,30)$ can be obtained.

Finally, we consider a series of lines defined by the objective function, e.g. if we have $\max z = 5x_1 + 4x_2$, then we consider $5x_1 + 4x_2 = c$ for any constant c. It is a level curve for the objective function. It is like a contour line on a map retaining the same altitude – all points on this line have the same objective function value. This level curve is called an *iso-profit* line for a maximisation problem, and an *iso-cost* line in a minimisation sense.

These lines are parallel to each other, so it is like sliding a ruler across the page as c increases in a maximisation sense. We finally get the last point (which is definitely an extreme point) or line segment (which connects two extreme points) where the ruler just touches the feasible region – that is an optimal solution(s). There are four possible outcomes from this procedure:

- There is a unique optimal solution;
- There are an infinite number of optimal solutions;
- There is no feasible solution to the LP, and hence no optimal solution exists;
- The problem is unbounded (and the solution can be boundlessly improved).

Further reading: Chapter 2 and 3 in the reference book "Operations Research: Applications and Algorithms" (Winston, 2004)