GAMS - Modeling and Solving Optimization Problems

Abebe Geletu
Institute of Mathematics
Department of Operations Research & Stochastic
Ilmenau University of Technology

April 28, 2008

Contents

1 Linear Optimization Problems **2**
1.1 A review of Algorithms for Solving Linear Optimization Problems 2
1.2 Algorithms for LP 2
1.3 A Review of the Simplex Algorithm 3
1.3.1 Basic Idea of the Simplex-Algorithm 3
1.3.2 Duality and Optimality for LOP 6
1.3.3 Sensitivity Analysis 10
1.4 The Interior Point Method for LOP 13

2 Modeling and Solving (LOP)s under GAMS **17**
2.1 Introduction 17
2.2 Linear Optimization Problems 17
2.2.1 Simple Models 17
2.2.2 Writing Better GAMS Models 22
2.2.3 Solving a GAMS Model 26
2.2.4 Mixed Integer Linear Programming 30
2.2.5 Choosing a Solver and Setting Solver Options 33
2.2.6 Outputting Solver Report to a File 38
2.3 Importing Excel Data into GAMS 40

Chapter 1

Linear Optimization Problems

1.1 A review of Algorithms for Solving Linear Optimization Problems

We consider the following general form of a linear optimization problem

$$\boxed{\begin{array}{lcl} c^\top x & \longrightarrow & min \\ s.t. & & \\ Ax & \leq & b_1 \\ Bx & = & b_2 \\ L \leq & x & \leq U; \end{array}} \qquad \text{(LOP)}$$

where $A \in \mathbb{R}^{m_1 \times n}, B \in \mathbb{R}^{m_2 \times n}, b_1 \in \mathbb{R}^{m_1}, b_2 \in \mathbb{R}^{m_2}$ and $c, L, U \in \mathbb{R}^n$.

1.2 Algorithms for LP

Currently the most frequently used algorithms for the solution of LP's are:

- a simplex algorithm;
- a primal-dual interior point method.

The simplex algorithm is usually used to solve small- to **medium-scale** linear optimization problems. If the LP has a solution and the simplex algorithm fails to find this solution, then the problem at hand is a **large scale** problem. Moreover, a linear optimization problem with several thousands of variables along with sparse matrices is usually considered to be a large-scale problem. Large-scale LP's are solved using the **interior-point** method.

In the following we find a brief review of the classical primal simplex algorithm and the interior-point method. It is hoped this review will facilitate the interpretation and understanding of GAMS solver outputs. In fact, (LOP) algorithms in GAMS are most efficient variants of the simplex algorithm and the interior-point method, being implemented to exploit problem structure sometimes are combinations of simplex and interior point algorithms. Details of implemented algorithms could be obtained from the user guide of each solver. Currently more that 90 percent of GAMS solvers are capable of solving (LOP)'s.

1.3 A Review of the Simplex Algorithm

We can rewrite the (LOP) above into the following linear equivalent form by adding slack variables and transforming variables:

$$\boxed{\begin{array}{lll} c^\top x & \longrightarrow & min \\ s.t. & & \\ \quad Ax = b & & \\ \quad x \geq 0; & & \end{array}} \qquad \text{(LOP)}$$

and assume that $A \in \mathbb{R}^{m \times n}, m \leq n$. Without loss of generality, we also assume that $rank(A) = m$.

A system of equations $Ax = b$ with $rank(A) = m \leq n$ is under determined (less equations than number of variables). Thus, since $rank(A) = m$, we can fix the (n-m) variables so that we can solve for the remaining m variables. For this we can partition the equation (possibly after reordering the columns of the matrix A) such that

$$Ax = [B, N] \begin{pmatrix} x_B \\ x_N \end{pmatrix} = b$$

where $B \in \mathbb{R}^{m \times m}$ is an invertible matrix (also called Basis Matrix) with the partitioning

$$x = \begin{pmatrix} x_B \\ x_N \end{pmatrix}$$

the components of $x_B \in \mathbb{R}^m$ correspond to the columns of B, hence they are called *basic variables* and those of $x_N \in \mathbb{R}^{n-m}$ are *non-basic*. It follows that

$$Bx_B + Nx_N = b \Rightarrow x_B = B^{-1}b - B^{-1}Nx_N$$

Note that, if we fix $x_N = 0 \in \mathbb{R}^{n-m}$ we can easily determine x_B.

Accordingly, we can partition the objective function of (LOP) as

$$z(x) = c^\top x = c_B^\top x_B + c_N^\top x_N = c_B^\top (B^{-1}b - B^{-1}Nx_N) + c_N^\top x_N.$$

$\Rightarrow$

$$z(x) = c_B^\top B^{-1}b + \left(\underbrace{c_N^\top - c_B^\top B^{-1}N}_{r:=} \right) x_N$$

1.3.1 Basic Idea of the Simplex-Algorithm

Assume that the feasible set $\{x \in \mathbb{R}^n \mid Ax \leq b, x \geq 0\}$ of (LOP) is not empty.

- Start with a feasible vector $x = \begin{pmatrix} x_B \\ x_N \end{pmatrix}$ with $x_B \geq 0$ and $x_N = 0$.
- Test if x is optimal.
  - $\star$ If yes STOP.

$\star$ ELSE either (i) find a new feasible point that reduces the objective function or (ii) determine that the objective function is not bounded below (since our LOP is a minimization problem), i.e. there is no solution.

Step 0: Start with a feasible vector $x = \begin{pmatrix} x_B \\ x_N \end{pmatrix}$ with $x_B \geq 0$ and $x_N = 0$. Set I_B index set of the columns of A which correspond to x_B and I_N index set of the columns A which correspond to x_N.

Step k: If $c_N^\top - c_B^\top B^{-1} N \geq 0$

- Then x is optimal. STOP.
- Else determine a descent direction d and a step length α, so that $x + \alpha d$ is feasible, at least $n - m$ components of $x + \alpha d$ are equal to zero and $z(x + \alpha d) < z(x)$.

Step k+1: If the problem is bounded in the direction of d, then put $x = x + \alpha d$, update I_B and I_N and GOTO Step k. Otherwise the problem is unbounded. STOP.

Let I_B index set of the columns of A which belong to B and I_N index set of the corresponding the columns A which belong to N.

Question
How to determine a descent direction?
How to determine a step-length?

Note that if the (optimality) condition $r := c_N^\top - c_B^\top B^{-1} N \geq 0$ is not satisfied at the current vector, then in the expression

$$z(x) = \underbrace{c_B^\top B^{-1} b}_{a\ constant} + \left(c_N^\top - c_B^\top B^{-1} N \right) x_N$$

there is at least one component of $r := \left(c_N^\top - c_B^\top B^{-1} N \right) \in \mathbb{R}^{n-m}$ **which is less than zero**. This implies that, if we choose positive values for corresponding components of x_N, then we can reduce the objective function.

In fact, recall that

$$x_B = B^{-1} b - B^{-1} N x_N$$

$\Rightarrow$

$$\begin{aligned} x_B &= B^{-1} b - \quad B^{-1} N x_N \\ x_N &= \qquad\quad x_N \end{aligned}$$

$\Rightarrow$

$$\begin{pmatrix} x_B \\ x_N \end{pmatrix} = \begin{pmatrix} B^{-1} b \\ \mathbb{O}_{n-m} \end{pmatrix} + \begin{pmatrix} B^{-1} N \\ \mathbb{I}_{n-m} \end{pmatrix} x_N$$

We can write $B^{-1} N = B^{-1} [A_N^1, \ldots, A_N^{n-m}] = [B^{-1} A_N^1, \ldots, B^{-1} A_N^{n-m}]$, where, for each $j \in J$, A_N^j is a column of A that belong to N. From this follows that

$$\begin{pmatrix} x_B \\ x_N \end{pmatrix} = \begin{pmatrix} B^{-1}b \\ \mathbb{O}_{n-m} \end{pmatrix} + \sum_{j=1}^{n-m} x_N^j \begin{pmatrix} B^{-1}A_N^j \\ e_j \end{pmatrix}.$$

Here, e_j is the j-th unit vector in $\mathbb{R}^{n-m}$.

Theorem 1.3.1. *If for some $j \in J$, $r_j < 0$ (i.e. the j-th component of the vector $r = c_N^\top - c_B^\top B^{-1}N$), then the vector*

$$d^j = \begin{pmatrix} B^{-1}A_N^j \\ e_j \end{pmatrix}$$

is a feasible descent direction for the (LOP).

Consequently, we choose a vector d^j as a descent direction corresponding to $r_j < 0$ for some $j \in J$. Let $J_r := \{j \in J \mid r_j < 0\}$. Thus, for each $j \in J_r$ if we can determine $\alpha > 0$ such that

$$x^{new} = \underbrace{\begin{pmatrix} B^{-1}b \\ \mathbb{O}_{n-m} \end{pmatrix}}_{x^{old}} + \alpha d^j,$$

then we obtain

$$z(x^{new}) = z(x^{old} + \alpha d^j) < z(x^{old})$$

However, one of the following cases may arise:

- there is $j \in J_r$ such that all components of d^j are non-negative, i.e.,

$$d^j = \begin{pmatrix} B^{-1}A_N^j \\ e_j \end{pmatrix} \geq 0.$$

  In this case for each $\alpha > 0 : x^{old} + \alpha d^j \geq 0$ (since $x^{old} \geq 0$) $\Rightarrow$ for each $\alpha > 0 : x^{old} + \alpha d^j$ is feasible and d^j is a descent direction. This implies that the objective function reduces indefinitely in the direction d^j. That means the objective function is **unbounded below** and (LOP) has no solution.

- for each $j \in J_r$, d^j has a negative component. In this case, if the index set J_r is not a singleton, then we might have more than one possibilities to chose a descent direction, but we have to decide for a good one. There are several rules to chose d^j.

  (a) ***smallest index rule***:

  choose the direction d^{j_0} such that $j_0 \in J_r$ is the smallest index for which $r_{j_0} < 0$.

  (b) ***largest reduction rule (Dantzig's rule)***:

  choose the direction d^{j_0} such that $: |r_{j_0}| = max\{|r_j| \mid j \in J_r\}$.

  (i.e. the direction with the most negative r_j.)

Once we know that the problem is bounded below and have chosen a descent direction, then the next question will be: How to choose a step length? The step-length α is to be chosen in such a way that

- $x^{old} + \alpha d^{j_0}$ is feasible; and

- $x^{old} + \alpha d^{j_0}$ has at least $n - m$ zeros.

The classical technique used to choose the step-length α, with the above conditions being satisfied, is **the minimum ratio test**:

$$\alpha = \min_{i \in I_B} \left\{ -\frac{(x_B)_i}{d_i^{j_0}} \,\middle|\, d_i^{j_0} < 0 \right\}$$

Note that negative components of d^{j_0} can only appear among the components of d^{j_0} which correspond to the indices in I_B and $(x_B)_i$ is the i-th component of x_B.

The above minimum will be taken for some $i_0 \in I_B$:

$$\alpha = -\frac{(x_B)_{i_0}}{d_{i_0}^{j_0}}$$

With this choice of step-length we obtain:

$$x^{new} = \begin{pmatrix} (x_B)_1 - \frac{(x_B)_{i_0}}{d_{i_0}^{j_0}} d_1^j \\ \vdots \\ (x_B)_{i_0} - \frac{(x_B)_{i_0}}{d_{i_0}^{j_0}} d_{i_0}^j \\ \vdots \\ (x_B)_m - \frac{(x_B)_{i_m}}{d_{i_0}^{j_m}} d_m^j \\ \vdots \\ -\frac{(x_B)_{i_0}}{d_{i_0}^{j_0}} \\ \vdots \end{pmatrix} = \begin{pmatrix} (x_B)_1 - \frac{(x_B)_{i_0}}{d_{i_0}^{j_0}} d_1^j \\ \vdots \\ 0 \\ \vdots \\ (x_B)_m - \frac{(x_B)_{i_m}}{d_{i_0}^{j_m}} d_m^j \\ \vdots \\ -\frac{(x_B)_{i_0}}{d_{i_0}^{j_0}} \\ \vdots \end{pmatrix}$$

Thus in x^{new} the i_0-th component corresponding to the older x_B becomes zero, and the j_0-th component corresponding to the older x_N becomes non-zero. This is what is commonly called pivoting or basis-exchange. This guarantees that there are at least $n - m$ zero components in x^{new}, which has a very important geometric significance in simplex method.

However, it may happen that x^{new} has a zero component among its basis elements. Such an x^{new} is said have a **degenerate** basis. In the subsequent step the step-length α will be equal to zero. The algorithm will be trapped into an endless loop, commonly known as **cycling**. There are several effective anti-cycling schemes used along with the simplex algorithm.

1.3.2 Duality and Optimality for LOP

For the linear optimization problem

$$\boxed{\begin{array}{lcl} z = c^\top x & \longrightarrow & min \\ s.t. & & \\ Ax & = & b \\ x & \geq & 0. \end{array}} \qquad \text{(LOP)}$$

its associated dual problem is given as

$$\boxed{\begin{array}{lcl} w = b^\top u & \longrightarrow & max \\ s.t. & & \\ A^\top u & \leq & c \\ u & \in & \mathbb{R}^m. \end{array}} \qquad \text{(LOP}_D\text{)}$$

Using a slack variable $s \in \mathbb{R}^n$ we write

$$\boxed{\begin{array}{lcl} w = b^\top u & \longrightarrow & max \\ s.t. & & \\ A^\top u + s & = & c \\ u \in \mathbb{R}^m, & s \geq 0 & . \end{array}} \qquad \text{(LOP}_D\text{)}$$

It is well known that, x^* is a solution of the primal (LOP) and (u^*, s^*) is a solution of dual (LOP') if and only if the Karush-Kuhn-Tucker (KKT) optimlaity condition is satisfied. The KKT conditions here can be written as

$$\begin{aligned} A^\top w^* + s^* &= c && (1.1) \\ Ax^* &= b && (1.2) \\ x_i^* s_i^* &= 0, i = 1, \ldots, n \text{(Complemetarity conditions)} && (1.3) \\ (x^*, s^*) &\geq 0. && (1.4) \end{aligned}$$

This is equivalent to the satisfaction of the strict complementarity; i.e.,

- for $i \in \{1, \ldots, n\} : x_i^* > 0 \Rightarrow s_i^* = 0$;
- for $i \in \{1, \ldots, n\} : s_i^* > 0 \Rightarrow x_i^* = 0$.

Important Terminologies

The components of the vector

$$r = c_N^\top - c_B^\top B^{-1} N$$

are called **reduced costs for the non-basic variables**. In particular, we have

$$r_j = c_j - c_B^\top B^{-1} A_j, j \in I_N.$$

By analogy, the **reduced costs for the basic variables** will be

$$r_i = c_i - c_B^\top B^{-1} A_i, i \in I_B.$$

However, since for each $i \in I_B, A_i$ is a column of A in the basis matrix B, we have $B^{-1} A_i = e_i$ (the i-th unit vector in $\mathbb{R}^m$) and

$$r_i = c_i - c_B^\top e_i = c_i - c_i = 0, i \in I_B.$$

This implies that the reduced costs associated with basic variables are equal to zero.

In GAMS the reduced costs are called **marginals**. Thus, in a solution x^* of an (LOP) the basic and non-basic variables can be identified by the values of the marginals.

marginal	variable
greater than zero	non-basic
equal to zero	basic
equal to zero (Eps)	non-basic

Duality and Optimality for LOP with Bounded Variables and the Simplex Method

$$\boxed{\begin{array}{lcl} z = c^\top x & \longrightarrow & min \\ s.t. & & \\ Ax & = & b \\ lb \leq & x & \leq ub. \end{array}} \qquad \text{(LOP)}$$

where $lb, ub \in \mathbb{R}^n$. Then the dual can be written as

$$\boxed{\begin{array}{ll} w = & b^\top u + (lb)^\top v - (ub)^\top y \longrightarrow max \\ s.t. & \\ & A^\top u - v + y = c \\ & u \in \mathbb{R}^m,\ v \geq 0,\ y \geq 0. \end{array}} \qquad \text{(LOP')}$$

The vectors $\overline{x}$ and $(\overline{u}, \overline{v}, \overline{y})$ are optimal solutions of the primal-dual pair (LOP) and (LOP') if and only if

$$\begin{aligned} A^\top \overline{u} + \overline{v} - \overline{y} &= c && (1.5)\\ A\overline{x} &= b && (1.6)\\ (\overline{x} - lb)^\top \overline{v} &= 0 && (1.7)\\ (ub - \overline{x})^\top \overline{y} &= 0 && (1.8)\\ \overline{v} \geq 0, \overline{y} &\geq 0. && (1.9) \end{aligned}$$

The equations (1.7)& (1.8) constitute the complementarity conditions. Thus, the above optimality conditions are equivalent to the satisfaction of strict complementarity w.r.t. (1.7)& (1.8).

Remark 1.3.2. *Thus given a solution $\overline{x}$ of (LOP) with bounded variables, it may happen that one of the components of $\overline{x}$ may take the lower bound or the upper bound. In general, the properties given below hold true. For some $i \in \{1, \ldots, n\}$*

(i) if $\overline{x}_i = lb_i$, then $(A^\top \overline{u})_i = \overline{u}^\top A_i < c_i$;

(ii) if $\overline{x}_i = ub_i$, then $(A^\top \overline{u})_i = \overline{u}^\top A_i > c_i$;

(iii) if $lb_i < \overline{x}_i < ub_i$, then $(A^\top \overline{u})_i = c_i$.

where A_i is the i-th column of A.

Proof. Suppose $\overline{x}_i = lb_i$. From this follows that $\overline{v}_i > 0$ and $\overline{x}_i < ub_i \Rightarrow \overline{y}_i = 0$ (by complementarity). The equation (1.5) yields

$$(A^\top \overline{u})_i + (\overline{v})_i - (\overline{y})_i + \overline{s}_i = c_i \Rightarrow (A^\top \overline{u})_i + (\overline{v})_i + \overline{s}_i = c_i \Rightarrow (A^\top \overline{u})_i < c_i$$

Similarly, you can verify (ii) and (iii).

□

The simplex method for the solution of the (LOP) with bounded variables

$$\boxed{\begin{array}{lcl} z = c^\top x & \longrightarrow & min \\ s.t. & & \\ Ax & = & b \\ lb \leq & x & \leq ub. \end{array}} \qquad \text{(LOP)}$$

must guarantee that at each step the iterates remain bounded between the vectors lb and ub. Thus, given a feasible vector $\hat{x}$ the **basic variables** are those components of the $\hat{x}$ that satisfy

- $(lb)_i < x_i < (ub)_i$
- the matrix B formed from A corresponding to this $x_i's, i \in I_B := \{i \in \{1, \ldots, n\} \,|(lb)_i < x_i < (ub)_i\}$, is a Basis matrix (i.e. the columns B are basis for $\mathbb{R}^m$).

All other components of $\hat{x}$ are non-basic. Let $I_N = \{1, \ldots, n\} \setminus I_B$ index set of non-basic variables. Such a feasible vector $\hat{x}$ is called a basic feasible vector (but may not be an optimal solution). To test the optimality of $\hat{x}$ we need to check the sign of the reduced costs (marginals). Thus, let (as before)

$$\hat{x} = \begin{pmatrix} \hat{x}_B \\ \hat{x}_N \end{pmatrix}$$

and the reduced costs $r = c_N^\top - c_B^\top B^{-1} N$.

Remark 1.3.3. . *If a basic feasible vector $\hat{x}$ is optimal, then the following hold true for the non-basic components of $\hat{x}$, for $j \in I_N$,*

(i) $\hat{x}_j = lb_j \Rightarrow r_j \geq 0$

(ii) $\hat{x}_j = ub_j \Rightarrow r_j \leq 0$

(ii) $lb_j < \hat{x}_j < ub_j \Rightarrow r_j = 0$

Proof. Given a solution $\hat{x}$ of (LOP), by duality theory is a solution $(\hat{u}, \hat{v}, \hat{y})$ of the dual (LOP') such that

$$A^\top \hat{u} + \hat{v} - \hat{y} = c$$

Using the partition $A = [B, N]$ and the definition of the basic variables $\hat{x}_B$ and applying the complementarity condition we can write

$$\begin{pmatrix} B^\top \\ N^\top \end{pmatrix} \hat{u} + \begin{pmatrix} \mathbb{O} \\ \hat{v}_N \end{pmatrix} + \begin{pmatrix} \mathbb{O} \\ \hat{u}_N \end{pmatrix} = \begin{pmatrix} c_B \\ \hat{c}_N \end{pmatrix} \Rightarrow B^\top \hat{u} = c_B.$$

Consequently, we can write the reduced costs as

$$r = c_N^\top - (\hat{u}^\top B) B^{-1} N = c_N^\top - \hat{u}^\top N$$

$\Rightarrow$

$$r_j = c_j - \hat{u}^\top A_j, j \in I_N.$$

Then the rest of the proof follows from Rem 1.3.2. □

Remark 1.3.3 implies that if any one of the above conditions is not satisfied, then $\hat{x}$ is not an optimal solution.

1.3.3 Sensitivity Analysis

Suppose we are given a linear optimization problems (LOP)

$$\boxed{\begin{array}{lcl} z = c^\top x & \longrightarrow & min \\ s.t. & & \\ Ax & = & b \\ x & \geq & 0. \end{array}} \qquad \text{(LOP)}$$

If (LOP) has a solution $\hat{x}$, then this solution is completely determined by the data (A, b, c). **Sensitivity (or postoptimal)** analysis deals with how the optimal solution of an (LOP) changes if there is a slight change on the data (A, b, c). Such small change on the data my happen, say, if the data (A, b, c) is a result of some experimental output.????????

The question here is: if we change (A, b, c) to $(A + \triangle A, b + \triangle b, c + \triangle c)$, does the solution $\hat{x}$ remain the same? If the answer is yes, then we say the problem is **stable** under small perturbation of the data (A, b, c). However, stability of the solution is only guaranteed only if the (perturbation) term $(\triangle A, \triangle b, \triangle c)$ lies in some bounded interval. Even then, sensitivity analysis based on the whole problem data is a difficult task (properly speaking, it lies in field of parametric linear programming). Thus, common sensitivity analysis for (LOP) is done separately based on the perturbation of either of c, b or A.

A. Perturbation of the coefficient vector of the objective function

Suppose we have already obtained an optimal solution x^* of (LOP), which is partitioned into basic and non-basic parts:

$$x^* = \begin{pmatrix} x_B \\ x_N \end{pmatrix},$$

then we want to study the sensitivity of x^* to a small perturbation of the coefficient vector c. Given a purturbation $\triangle c$ of c we ask: for what value of α does x^* remains a solution to the problem

$$\begin{array}{lll} z + \triangle z & = (c + \alpha \triangle c)^\top x \longrightarrow min & (LOP') \\ s.t. & & \\ & Ax = b & \\ & x \geq 0\ ? & \end{array}$$

Note that x^* remains feasible for the perturbed problem with the corresponding matrices B and N. Using the corresponding partition of the vector $c + \alpha \triangle c$

$$c + \alpha \triangle c = \begin{pmatrix} c_B \\ c_N \end{pmatrix} + \alpha \begin{pmatrix} \triangle c_B \\ \triangle c_N \end{pmatrix},$$

Thus x^* is optimal for (LOP') if its reduced costs at x^* is non-negative:

$$(c_N + \alpha \triangle c_N)^\top - (c_B + \alpha \triangle c_B)^\top B^{-1} N \geq 0$$

$\Rightarrow$

$$\left(c_N^\top - c_B^\top B^{-1} N\right) + \alpha \left(\triangle c_N^\top - \triangle c_B^\top B^{-1} N\right) \geq 0.$$

Let

$$r = c_N^\top - c_B^\top B^{-1}N \text{ and } \triangle r = \triangle c_N^\top - \triangle c_B^\top B^{-1}N.$$

Thus we have

$$r + \alpha \triangle r \geq 0.$$

But this hold true if $\underline{\alpha} \leq \alpha \leq \overline{\alpha}$, where

$$\underline{\alpha} = \max_{j \in I_N} \left\{ -\frac{r_j}{\triangle r_j} \mid \triangle r_j > 0 \right\} \text{ and } \overline{\alpha} = \min_{j \in I_N} \left\{ -\frac{r_j}{\triangle r_j} \mid \triangle r_j < 0 \right\}.$$

Consequently, x^* will remain an optimal solution also for (LOP') if α is chosen from the interval $[\underline{\alpha}, \overline{\alpha}]$. This is the **interval of stability** w.r.t. to a perturbation $\triangle c$ of the coefficient vector of the objective function.

B. Perturbation of the right-hand side vector

Let again x^* be a solution and z^* the optimal value of (LOP) . How do the the optimal solution and optimal value of (LOP) change if we perturb the right-hand side of the constraint $Ax = b$ with $\triangle b$? That is , we have

$$\begin{array}{lll} z(\alpha) & = c^\top x \longrightarrow min & (LOP') \\ s.t. & & \\ & Ax = b(\alpha), & \\ & x \geq 0, & \end{array}$$

where $\alpha \geq 0$; e.g. $b(\alpha) = b + \alpha \triangle b$ for a given $\triangle b$.

For $b(\alpha) = b + \alpha \triangle b$,

$$\begin{array}{lll} z(\alpha) & = c^\top x \longrightarrow min & (LOP') \\ s.t. & & \\ & Ax = b + \alpha \triangle b, & \\ & x \geq 0. & \end{array}$$

We attempt to characterize the solution of (LOP') in terms of those of (LOP). It is usually assumed that the solution $x^*(\alpha)$ of (LOP') to have the same basis matrix B and non-basic matrix N. This implies

$$[A, B] \begin{pmatrix} x^*(\alpha)_B \\ x^*(\alpha)_N \end{pmatrix} = b + \alpha \triangle b.$$

Taking the non-basic $x_N^*(\alpha)$ equal to zero, we have

$$x^*(\alpha)_B = B^{-1}(b + \alpha \triangle b) \Rightarrow x^*(\alpha) = \begin{pmatrix} B^{-1}(b + \alpha \triangle b) \\ \mathbb{O}_{n-m} \end{pmatrix}.$$

Hence, from the above assumption

- the reduced costs corresponding to $x^*(\alpha)$ will be

$$r = c_N^\top - c_N^\top B^{-1}N$$

  which is the same as that of x^*. Therefore, $r \geq 0$.

- Next, if we give conditions for the feasibility of $x^*(\alpha)$, then $x^*(\alpha)$ will be optimal for (LOP'), since $r \geq 0$. Nevertheless, $x^*(\alpha)$ is feasible if

$$B^{-1}(b + \alpha\triangle b) \geq 0 \Rightarrow B^{-1}b + \alpha B^{-1}\triangle b \geq 0 \Rightarrow \alpha B^{-1}\triangle b \geq -B^{-1}b.$$

Now, let's define

$$\overline{\alpha} := \min\left\{ \frac{-\left(B^{-1}b\right)_i}{\left(B^{-1}\triangle b\right)_i} \;\middle|\; \left(B^{-1}\triangle b\right)_i < 0, i \in I_B \right\}$$
$$\underline{\alpha} := \max\left\{ \frac{-\left(B^{-1}b\right)_i}{\left(B^{-1}\triangle b\right)_i} \;\middle|\; \left(B^{-1}\triangle b\right)_i > 0, i \in I_B \right\}.$$

It follows that for $\underline{\alpha} \leq \alpha \leq \overline{\alpha}$, the vector $x^*(\alpha) = \begin{pmatrix} B^{-1}(b + \alpha\triangle b) \\ \mathbb{O}_{n-m} \end{pmatrix}$ is feasible for (LOP') and, hence, optimal. (Note that $\underline{\alpha} \leq 0$ and $\overline{\alpha} \geq 0$).

Furthermore, the optimal value of the objective function of (LOP) will be

$$z^*(\alpha) = c^\top x^*(\alpha) = [c_B^\top, c_N^\top] \begin{pmatrix} B^{-1}(b + \alpha\triangle b) \\ \mathbb{O}_{n-m} \end{pmatrix} = c_B^\top B^{-1}b + \alpha c_B^\top B^{-1}\triangle b.$$

$\Rightarrow$

$$z^*(\alpha) = z^* + \alpha\, c_B^\top B^{-1}\triangle b.$$

Which implies that $z^*(\alpha)$ is a linear function on the interval $[\underline{\alpha}, \overline{\alpha}]$. In general, if $b(\alpha)$ is a linear perturbation, w.r.t. α, then $z^*(\alpha)$ can be shown to be a piece-wise linear and convex function (see [1] for more details).

Furthermore, the sensitivity of the objective function z of (LOP) is related to the solution of the dual problem (LOP$_D$). To see this let x^* and u^* be solutions of (LOP) and (LOP$_D$), respectively. In the (LOP$_D$), we have :

$$A^\top u^* + s^* = c$$

Using the partition of A, c and s^* into basic and non-basic blocks corresponding to x^*, we have

$$\begin{bmatrix} B^\top \\ N^\top \end{bmatrix} u^* + \begin{pmatrix} s_B^* \\ s_N^* \end{pmatrix} = \begin{pmatrix} c_B \\ c_N \end{pmatrix} \Rightarrow \begin{bmatrix} B^\top u^* + s_B^* \\ N^\top u^* + s_N^* \end{bmatrix} = \begin{pmatrix} c_B \\ c_N \end{pmatrix}$$

Since $x_B^* \geq 0$, it follows from the complementarity condition (1.3) that $s_B^* = 0$. Hence,

$$B^\top u^* = c_B \Rightarrow (u^*)^\top = c_B^\top B^{-1}.$$

Furthermore, we have

$$\begin{aligned} z^* &= c^\top x^* = c_B^\top x_B + c_N^\top x_N \\ &= c_B^\top B^{-1}b + (c_N^\top - c_B^\top B^{-1}N)x_N \\ &= (u^*)^\top b + r^\top x_N. \end{aligned}$$

Thus the variation of z^* w.r.t. b is associated only with the first term. Consequently,

$$\frac{\partial z^*}{\partial b_i} = u_i^*, i = 1, \ldots, m.$$

Thus the dual variables are measures of rate of change of the primal objective function as a result of perturbation of the components of the right hand-side vector b. For instance, in a transportation problem, these measure the change in the optimal cost a result of slight change in demand, etc. In general, in economic applications the $u_i^{*\prime}s, i = 1, \ldots, m$, are known to be **shadow prices**. In GAMS they are output as marginal values for individual constraints in $Ax = b$.

1.4 The Interior Point Method for LOP

It is believed that the interior point method is more efficient for solving large-scale (LOP)s, while the simplex method is well suited for problems of medium size. In this section we fined a brief discussion on the primal-dual interior point method for (LOP).

Let $A \in \mathbb{R}^{m\times n}, a \in \mathbb{R}^m, B \in \mathbb{R}^{p\times n}, b \in \mathbb{R}^p$. Then, for the linear programming problem

$$\boxed{\begin{array}{lcl} c^\top x & \longrightarrow & min \\ s.t. & & \\ Ax & \leq & a \\ Bx & = & b \\ lb \leq & x & \leq ub; \end{array}} \qquad \text{(LP)}$$

if we set $\widetilde{x} = x - lb$ we get

$$\boxed{\begin{array}{lcl} c^\top \widetilde{x} - c^\top lb & \longrightarrow & min \\ s.t. & & \\ A\widetilde{x} & \leq & a - A(lb) \\ B\widetilde{x} & = & b - B(lb) \\ 0 \leq & \widetilde{x} & \leq ub - lb; \end{array}} \qquad \text{(LP)}$$

Now, by adding slack variables $y \in \mathbb{R}^m$ and $s \in \mathbb{R}^n$ (see below), we can write (LP) as

$$\boxed{\begin{array}{lcccc} c^\top \widetilde{x} - c^\top lb & \longrightarrow & min & & \\ s.t. & & & & \\ A\widetilde{x} & +y & & = & a - A(lb) \\ B\widetilde{x} & & & = & b - B(lb) \\ \widetilde{x} & & +s & = & ub - lb \\ & \widetilde{x} \geq 0, & y \geq 0, & s \geq 0. & \end{array}} \qquad \text{(LP)}$$

Thus, using a single matrix for the constraints, we have

$$\begin{array}{l} c^\top \widetilde{x} - c^\top lb \longrightarrow min \\ s.t. \\ \quad \begin{pmatrix} A & I_m & O_{m\times n} \\ B & O_{p\times m} & O_{p\times n} \\ I_n & O_{n\times n} & I_n \end{pmatrix} \begin{pmatrix} \widetilde{x} \\ y \\ s \end{pmatrix} = \begin{pmatrix} a - A(lb) \\ b - B(lb) \\ ub - lb \end{pmatrix} \\ \qquad \widetilde{x} \geq 0, y \geq 0, s \geq 0. \end{array}$$

Since, a constant in the objective does not create a difficulty, we assume w.l.o.g that we have a problem of the form

$$\boxed{\begin{array}{lcl} c^\top x & \longrightarrow & min \\ s.t. & & \\ Ax & = & a \\ x & \geq & 0. \end{array}} \qquad \text{(LP')}$$

Now the dual of (LOP$_D$) is the problem

$$\boxed{\begin{array}{lcl} b^\top w & \longrightarrow & max \\ s.t. & & \\ A^\top w & \leq & c \\ w & \in & \mathbb{R}^m. \end{array}} \qquad \text{(LP}_D\text{)}$$

Using a slack variable $s \in \mathbb{R}^n$ we have

$$\boxed{\begin{array}{lcl} b^\top w & \longrightarrow & max \\ s.t. & & \\ A^\top w + s & = & c \\ w \in \mathbb{R}^m, & s \geq 0 & . \end{array}} \qquad \text{(LP}_D\text{)}$$

The problem (LP') and (LP$_D$) are called primal-dual pairs.

Optimality Condition

It is well known that a vector (x^*, w^*, s^*) is a solution of the primal-dual if and only if it satisfies the Karush-Kuhn-Tucker (KKT) optimlaity condition. The KKT conditions here can be written as

$$\begin{array}{l} Ax = a \\ A^\top w + s = c \\ x_i s_i = 0, i = 1, \ldots, n \text{(Complemetarity conditions)} \\ (x, y) \geq 0. \end{array}$$

This system can be written as

$$F(x, w, s) = \begin{bmatrix} Ax - a \\ A^\top w + s - c \\ XSe \end{bmatrix} = 0 \qquad (1.10)$$

$$(x, s) \geq 0, \qquad (1.11)$$

where $X = diag(x_1, x_2, \ldots, x_n), S = diag(s_1, s_2, \ldots, s_n) \in \mathbb{R}^{n \times n}, e = (1, 1, \ldots, 1)^\top \in \mathbb{R}^n$.

Primal-dual interior point methods generate iterates (x^k, w^k, s^k) that satisfy the system (1.10) & (1.11) so that (1.11) is satisfied strictly; i.e. $x^k > 0, s^k > 0$. That is, for each k, (x^k, s^k) lies in the interior of the nonnegative-orthant. Thus the naming of the method as **interior point method**. Interior point methods use a variant of the Newton method to solve the system (1.10) & (1.11).

Central Path

Let $\tau > 0$ be a parameter. The **central path** is a curve $\mathcal{C}$ which is the set of all points $(x, w, s) = (x(\tau), w(\tau), s(\tau)) \in \mathcal{C}$ that satisfy the parametric system :

$$\begin{aligned} &Ax = b, \\ &A^\top w + s = c, \\ &xs_i = \tau, i = 1, \ldots, n \\ &(x, s) > 0. \end{aligned}$$

The parameter $\tau \geq 0$ is called a **barrier parameter**. Hence, $\mathcal{C}$ is the set of all points (x, w, s) that satisfy

$$F(x, w, s) := \begin{bmatrix} Ax - a \\ A^\top w + s - c \\ XSe - \tau e \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, (x, s) > 0. \tag{1.12}$$

Obviously, if we let $\tau \downarrow 0$, the the system (1.12) goes close to the system (1.10) & (1.11).

Hence, the Newton method for the solution of (1.12) has the form

$$\begin{pmatrix} A & O_{n\times m} & O_{m\times n} \\ O_n & I_n & A^\top \\ V & X & O_{n\times m} \end{pmatrix} \begin{pmatrix} \triangle x \\ \triangle s \\ \triangle w \end{pmatrix} = - \begin{pmatrix} Ax - a \\ A^\top w + s - c \\ XSe - \tau e \end{pmatrix}$$

to determine a search direction $(\triangle x, \triangle s, \triangle w)$, where the matrix on the left is the Jacobian of the system. And the new iterate will be

$$(x^+, s^+, w^+) = (x, s, w) + \alpha(\triangle x, \triangle s, \triangle w),$$

where α is a step length, usually $\alpha \in (0, 1]$, chosen in such a way that $(x^+, s^+, w^+) \in \mathcal{C}$.

Practical primal-dual interior point methods attempt to reduce the duality gap between the primal and dual problems at the same time keeping the iterates in the interior of thier respective feasible regions. Thus, two parameters are used so that $\tau = \sigma\mu$, where $\sigma \in [0, 1]$ is a constant and

$$\mu = \frac{x^\top s}{n}$$

The term $x^\top s$ is the **duality gap** between the primal and dual problems. That is, μ is the measure of the (average) duality gap. Note also that, in general, $\mu \geq 0$; and $\mu = 0$ when x and s are primal and dual optimal, respectively.

Now, the Newton step $(\triangle x, \triangle s, \triangle w) = (\triangle x(\mu), \triangle s(\mu), \triangle w(\mu))$ is determined by solving:

$$\begin{pmatrix} A & O_{n\times m} & O_{m\times n} \\ O_n & I_n & A^\top \\ V & X & O_{n\times m} \end{pmatrix} \begin{pmatrix} \triangle x \\ \triangle s \\ \triangle w \end{pmatrix} = - \begin{pmatrix} Ax - a \\ A^\top w + s - c \\ XSe - \sigma\mu e \end{pmatrix} \tag{1.13}$$

The Newton step $(\triangle x, \triangle s, \triangle w)$ is also called a **centering direction** that pushes the iterates (x^+, s^+, w^+) towards the central path $\mathcal{C}$ along which the algorithm converges more rapidly. The parameter σ is called **the centering parameter**. If $\sigma = 0$, then the search direction is

known to be an **affine scaling** direction.

Primal-Dual Interior Point Algorithm

Step 0: Start with (x^0, s^0, w^0) with $(x^0, s^0) > 0$, $k = 0$

Step k: choose $\sigma_k \in [0, 1]$, set $\mu_k = (x^k)^\top s^k/n$ and solve

$$\begin{pmatrix} A & O_{n\times m} & O_{m\times n} \\ O_n & I_n & A^\top \\ V & X & O_{n\times m} \end{pmatrix} \begin{pmatrix} \triangle x^k \\ \triangle s^k \\ \triangle w^k \end{pmatrix} = - \begin{pmatrix} Ax^k - a \\ A^\top w^k + s^k - c \\ X^k S^k e - \sigma_k \mu_k e \end{pmatrix} \tag{1.14}$$

set

$$(x^{k+1}, s^{k+1}, w^{k+1}) \leftarrow (x^k, s^k, w^k) + \alpha_k(\triangle x^k, \triangle s^k, \triangle w^k)$$

choosing $\alpha_k \in [0, 1]$ so that $(x^{k+1}, s^{k+1}) > 0$.

If (convergence) STOP; else set $k \leftarrow k + 1$ GO To Step k.

Some primal-dual interior point algorithms use a predicator-corrector like method of Mehrotra [2] to guarantee that $(x^k, s^k) > 0$ for each $k, k = 1, 2, \ldots$

Predicator Step: A search direction (a predicator step) $d_p^k = (\triangle x^k, \triangle w^k, \triangle s^k)$ is obtained by solving the non-parameterized system (1.10) & (1.11).

Corrector Step: For a centering parameter σ is obtained from

$$d_c^k = [F^\top(x^k, w^k, s^k)]^{-1} F(x^k + \triangle x^k, w^k \triangle w^k, s^k + \triangle s^k) - \sigma \widehat{e}$$

where $\widehat{e} \in \mathbb{R}^{n+m+n}$, whose last n components are equal to 1.

Iteration: for a step length $\alpha \in (0, 1]$

$$(x^{k+1}, w^{k+1}, s^{k+1}) = (x^k, w^k, s^k) + \alpha(d_p^k + d_c^k).$$

Chapter 2

Modeling and Solving (LOP)s under GAMS

2.1 Introduction

GAMS = **G**eneralized **A**lgebraic **M**odeling **S**ystem. Properly speaking GAMS is a combination of a modeling system and a library of solvers. GAMS has it own programming language syntax. It allows a user to write a mathematical optimization problem in the syntax of GAMS. A GAMS model of an optimization problem is written independent of the type of solver to be used to solve it. This allows the modeler to run (solve) and test a given model using several solvers (algorithms) from the solvers library. That is, if we want to use a different solver, we do not need to remodel the problem. Thereby enabling the user to run one model with several solvers and compare algorithms based on their outputs. The individual solvers themselves are program packages that are supplied (to the GAMS Corporation) by several companies and institutions who hold the licenses for their packages. GAMS solvers are capable of solving optimization problems up to millions of variables.

The first step and most important job in solving an optimization problem under GAMS is the creation of an accurate GAMS model of a mathematical problem. This is the modeling step. When the modeling is finished, then an appropriate solver can be called to solve the model. If the modeling process has no error, then the solver produces a solution; otherwise, GAMS reports that there are errors in the modeling. When all is done well, the next step will be the interpretation and manipulation of solver outputs.

2.2 Linear Optimization Problems

2.2.1 Simple Models

Consider the linear optimization problem

$$\begin{array}{rcrcrcr} -2x_1 & + & 3x_2 & - & x_3 & \rightarrow & \min \\ -x_1 & + & x_2 & - & x_3 & \leq & 1 \\ 3x_1 & - & x_2 & + & 2x_3 & \leq & 8 \\ & & 2x_2 & + & 5x_3 & \leq & 10 \\ x_1 \geq 0 & , & x_2 \geq 0 & , & x_3 \geq 0. & & \end{array}$$

This can be modeled under GAMS as

```
Variables z;
Positive Variables x1,x2,x3;
Equations
obj, constr1, constr2, constr3;
obj.. z=E=-2*x1 + x2 -x3;
constr1..-x1-x2-x3=L=1;
constr2.. 3*x1-x2+2*x3=L=8;
constr3..2*x2+5*x3=L=10;
Model TestLp1 /ALL/;

    Solve TestLp1 Using LP Minimizing z;
```

Such a GAMS model of an optimization problem is usually saved using the file extension **.gms**.

In the above GAMS model the terms: **Variables** (the same as **Variable**), **Positive**, **Equations**, **Model**, **Solve** are reserved words or keywords. These are words that you should not use for any other purpose than as indicated in GAMS. GAMS keywords can be written in upper or lower case or mixed case letters. In general, GAMS is not case sensitive (say like MATLAB or c++). For instance: **variables** ,**Variables**, **VARIABLES**, **variAbles**, etc. all refer to the same keyword. But write keywords in a more convenient and readable way. For instance, use upper-case for the first letter of a keyword.

Variables

Always give identifying names for the variables in your problem. Define problem variables using the GAMS keyword **Variables**. Problem variables can be non-negative, discrete, etc. In this case use modifiers along with the GAMS keyword **Variables**, such as **Positive, Binary, SemiCont,** etc. In the model above, we have

- `Variables z;` - defines the (unbounded) variable z. The variable z stands for the objective function of (LOP). In GAMS objective functions of optimization problems should always be identified by an unbounded (unrestricted) variable.
- `Positive Variables x1,x2,x3;` - defines the non-negative variables x_1, x_2, x_3. Here we see the keyword **Variable** being used along with the modifier **Positive**.

Variable names can be up to 31 character long. Note also that **every GAMS statement should end with a semi-colon '; '.**

Equations

Equations and inequalities in a mathematical optimization problem should be all modeled as equations under GAMS. There are two steps in modeling equations.

naming equations: give names to each of your equations using keyword **Equations** or **Equation**;

Equations eqnName1, eqnName2,...;

Equation names can be up to 31 characters long.

defining equations: define each and every equation by using its algebraic description. Once you have declared an equation identifier using the keyword **Equations**, then you can define the actual equation as

eqnName1.. equation definition;
eqnName2.. equation definition;
⋮

When defining a named equation use always the '**..**' notation just after the name of the equation.

In the model given above the statement

```
Equations obj, constr1, constr2, constr3;
```

specifies four equations with names `obj`, `constr1`, `constr2`, `constr3`. After that each equation is defined using

```
obj.. z=E=-2*x1 + x2 -x3;
constr1..-x1-x2-x3=L=1;
constr2.. 3*x1-x2+2*x3=L=8;
constr3..2*x2+5*x3=L=10;
```

Equation and inequalities are indicated using the following symbols

=L=	$\leq$	Less than or Equal to
=E=	$=$	Equal to
=G=	$\geq$	Greater than or Equal to

Lower-case letters can be also used as `=l=`, `=e=`, `=g=`, correspondingly.

Model

After defining the basic parts of an optimization problem under GAMS. The model should be given a name using the **Model** keyword:

Model *modelName* descriptive text /model content/;

where

- *modelName* is a name that you may like to give your model. This name can be up to 31 characters long.
- *descriptive text* is an optional explanatory comment. Such text is non-executable and is, usually, written as a clarification. Such a text can be up to 255 characters long.
- /*model content*/ indicates which equations (say constraints) to be included in the model that is currently being defined.

In the model above we have the statement

```
Model TestLp1 /ALL/;
```

The name of the model is `TestLp1`; it has not descriptive text; the model includes all equations that has been defined. Since we have

```
/ALL/
```

Meaning that the model contains all previously defined equations (constraints). However, we can drop some constraints through the model definition - only to optimize with the rest of them. For example, for the (LOP) above to optimize w.r.t. the first and the third constraints (dropping the second one) we may define another model as

```
Model TestLp2 /obj,constr1,constr2/;
```

This implies that, in a given GAMS file we are allowed to have several `Model` statements with different sets of equations. We might, therefore, modify our GAMS (LOP) model above as

```
Variables z;
Positive Variables x1,x2,x3;
Equations
obj, constr1, constr2, constr3;
obj.. z=E=-2*x1 + x2 -x3;
constr1..-x1-x2-x3=L=1;
constr2.. 3*x1-x2+2*x3=L=8;
constr3..2*x2+5*x3=L=10;
Model TestLp1 /ALL/;
Model TestLp2 with out the 2nd constraint/obj,constr1,constr2/;
 Solve TestLp1 Using LP Minimizing z;
 Solve TestLp2 Using LP Minimizing z;
```

In the statement

Model TestLp2 with out the 2nd constraint/obj,constr1,constr2/;

The text "*with out the 2nd constraint*" is a descriptive text which is a comment.

Solve

In order to solve a given optimization model which is already defined using a **Model** statement we use the **Solve** command. The general form of a **Solve** statement

Solve *modelName* **Using** *modelType* **minimizing/maximizing** *varName*

where

- **Solve** is a command which tells GAMS to solve a model.
- *modelName* is a name of a model that you have defined using a **Model** statement;

– **Using** is a GAMS keyword followed by a *modelType* specifier. A *modelType* can be any GAMS known model type. It can be any one of the following with the indicated description:

modelType	**Description**
LP	Linear Programming
MIP	Mixed Integer Linear Programming
NLP	Non-Linear Programming
MINLP	Mixed Integer non-linear Programming
RMIP	Relaxed Mixed Integer Programming
RMINLP	Relaxed Mixed Integer non-linear Programming
MCP	Mixed complementary Programming
MPEC	Mathematical Programming with Equilibrium Constraints
DNLP	Discontinuous Non Linear Program
CNS	Constrained Nonlinear System

Consequently, you choose a *modelType* specification that fits your mathematical model.

– use one of the keywords **minimizing** or **maximizing** corresponding on your mathematical problem. In the case of MCP or CNS *modelType*'s either **minimizing** or **maximizing** is not required.

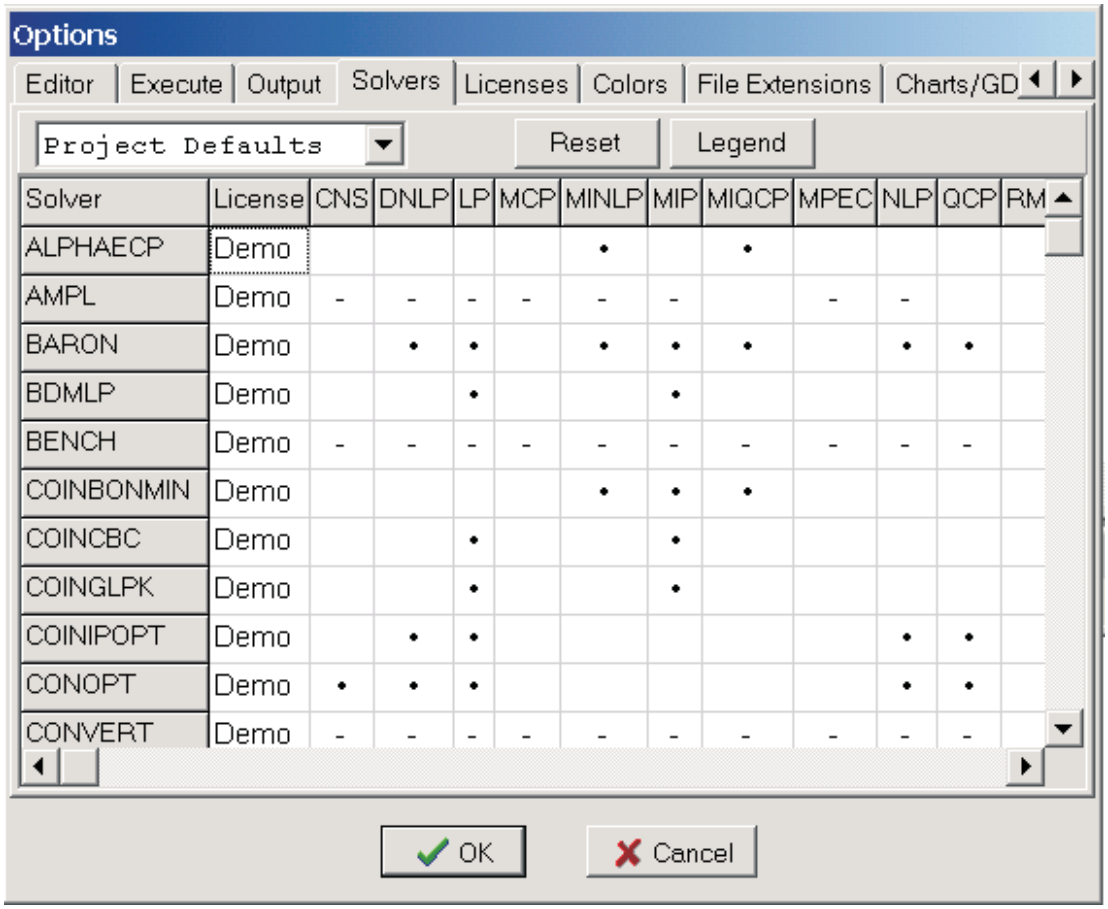
– *varName* is the name of the variable that represents the objective function in the model.

In the GAMS models given above we find the **Solve** statement

```
Solve TestLp1 Using LP Minimizing z;
```

Note that GAMS solves a model if there is only a **Solve** statement. Based on your *modeType* specification GAMS knows, by default, which solver to use to solve the problem. However, you can also choose a particular solver from available GAMS solvers. If you have a **Solve** statement, then GAMS allows you to select a different solver than the default solver. From the GAMS IDE choose

`File > options` and from the options widow use the **Solvers Tab** to specify a solver.



In general linear optimization problems are solved using

- the simplex algorithm; or
- the interior-point method

or a combination of them. Many GAMS solvers are capable of solving (LOP)s. Thus depending the size and complexity of a model a solver decides what algorithm to use.

2.2.2 Writing Better GAMS Models

That GAMS model that we have written to solve the (LOP)

$$\begin{array}{rcrcrcl} -2x_1 & + & 3x_2 & - & x_3 & \rightarrow & \min \\ -x_1 & + & x_2 & - & x_3 & \leq & 1 \\ 3x_1 & - & x_2 & + & 2x_3 & \leq & 8 \\ & & 2x_2 & + & 5x_3 & \leq & 10 \\ x_1 \geq 0 & , & x_2 \geq 0 & , & x_3 \geq 0. & & \end{array}$$

is not a good GAMS model. GAMS models should be written so that they are easy to upgrade, change and be written using GAMS general syntax. Thus a better GAMS model to the solve the above (LOP) is

```
 $Title  A Simple Linear Optimization Problem (simpleLp2)
$Ontext
         Abebe Geletu
         TU-Ilmenau
         FG OR & Stochastik
         Sommersemester 2008

This is a GAMS model to solve a simple linear optimization problem.
The purpose here is to demonstrate how to write good GAMS (LOP)
models.

$Offtext

*defining a symbole for an end of line comment
$eolcom ->

*all relevant indices are defined using the Sets keyword
Sets

         I row indices /row1,row2,row3/
         J column indices /col1,col2,col3/;

*objective function variable and problem variables
Variables z, x(J); positive  variable x;    ->non-negative variables


Parameters
         c(J) coefficient vector of objective
                 /col1 -2
                  col2 3
                  col3 -1/
         b(I) right-hand side vector of constraints
                 /row1 1
                  row2 8
                  row3 10/;
Table  A(I,J)   coefficient matrix
               col1      col2    col3
         row1   -1        1       -1
         row2    3       -1        2
         row3    0        2        5   ;

*Equation declaration
Equations
         obj  the objective function
         constr(I) the i-th constraint;

*Definition of objective and constraints
obj.. z=E=Sum(J,c(J)*x(J)); constr(I)..Sum(J,A(I,J)*x(J)) =L=b(I);
```

```
*Naming your model

Model TestLp2 /ALL/;

*solve the model
Solve TestLp2 Using LP Minimizing z;    ->solves the problem
```

Sets

The key-word **Sets** (or **Set**) is used to define indices. The code

```
Sets

         I row indices /row1,row2,row3/
         J column indices /col1,col2,col3/;
```

defines two index sets `I` and `J`. Here, `I` and `J` are identifiers of the two index sets. The indices `row1`, `row2`, `row3` are *elements* of set `I`; where as `col1`, `col2`, `col3` are *elements* of `J`. You can give any meaningful identifier for your sets (indices).

- Set identifiers can be up to 31 characters long, but the first character of the identifier must be a letter;
- element identifiers (an index) can be up to 10 characters long;
- element names can be quoted or unquoted.
- Unquoted elements
  - ⋆ must start with a letter or a number;
  - ⋆ if an element has two parts, then the parts must be written together (with no blank space)or must be joined with "-" .

  The following are legal element names

  ```
  Sets

           rowIndex row indices /row-1,row-2,row-3/
           columnIndex column indices /"col 1","col 2","col 3"/;
  ```

We can write the index set `I` and `J` in `simpleLp2.gms` in a more **compact form** as

```
Sets
         I row indices /row1*row3/
         J column indices /col1*col3/;
```

The operator "*" allows us to describe a very lengthy index set with a compact notation. For instance if there is an index set with indices from 1 up to 50, this can be described as

```
Sets
         myIndexSet  /1*50/;
```

Furthermore, through the use of elements of a set we can change, update, and manipulate data in parameters and tables as we wish.

Parameters

Usually, the key-word `Parameters` is used to define one dimensional data. Hence, using parameter we usually define vectors. Thus the following code fragment defines the coefficient vector of the objective function and the right-hand side vector of the inequality constraints:

```
Parameters
         c(J) coefficient vector of objective
                 /col1    -2
                  col2     3
                  col3    -1/
         b(I) right-hand side vector of constraints
                 /row1    1
                  row2    8
                  row3    10/;
```

This can also be written as

```
Parameters
         c(J) coefficient vector of objective
                 /col1  -2, col2  3, col3  -1/
         b(I) right-hand side vector of constraints
                 /row1   1
                  row2   8
                  row3   10/;
```

However, the former is more readable. Observe how we can use commas instead of line-breaks.

Tables

The key-word `Table` is used to define two or more dimensional data. For the (LOP) the coefficient matrix of the constraints has been defined as:

```
Table  A(I,J)   coefficient matrix
                col1      col2    col3
         row1   -1         1       -1
         row2    3        -1        2
         row3    0         2        5   ;
```

Comments

A comment is a descriptive text that you might use to document your GAMS models, at the same time, to add clarity to your modeling statements. There are four ways of adding comments

- **Header comment**. Such comments are usually put at the beginning of the model file. Such comments are written between the pair `$Ontext` and `$Offtext`.

```
$Ontext
        a model documentation text
$Offtext
```

Comment of this type are usually written to provide information on the name of the modeler (author), date of modeling, etc. and to describe the type of problem that has been modeled into the GAMS code.

- **One line comment** . If you want to write a comment on its own in a line, then you must put the "*" symbol in the **first column** of the line. For instance

  `*objective function variable and problem variables`

  is a one line comment.

- **Explanatory text**. Such text can be used in the program code itself. It is usually used to add clarity to the definition of variable and declaration statements. When properly written, GAMS knows which words belong to the comment and which to the actual modeling code. For example in the code fragment

```
Parameters
         c(J) coefficient vector of objective
                 /col1  -2, col2  3, col3  -1/
         b(I) right-hand side vector of constraints
                 /row1   1
                  row2   8
                  row3   10/;
```

  the text `coefficient vector of objective` and
  `right-hand side vector of constraints` are used for explanation.

- **End-of-line comments**. End-of-line comments are are usually marked by a user chosen symbol. But this marking symbol must be first defined using the command `$eolcom`. In the model simpleLp2.gms I have used the symbol "->" as a marker after defining it as

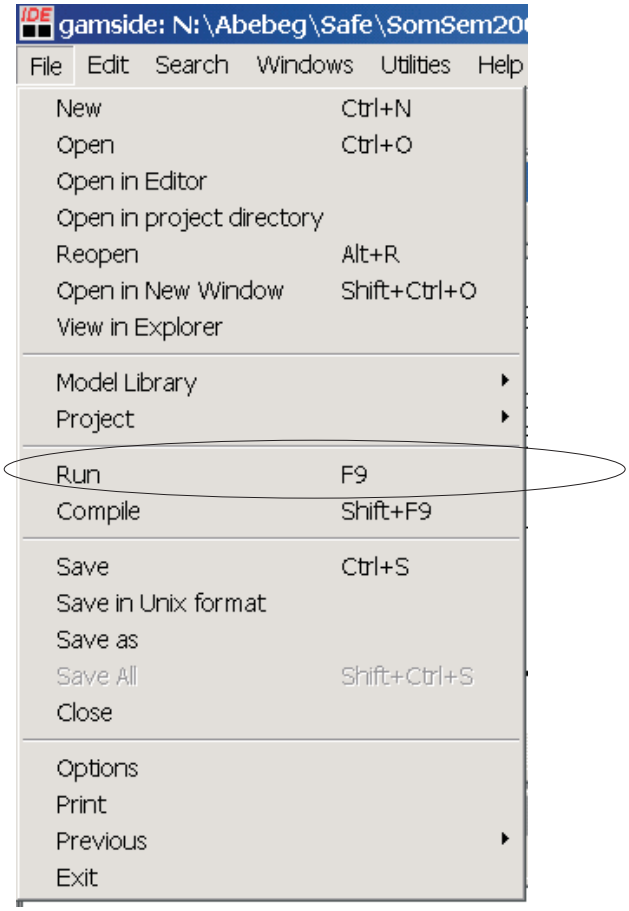
```
$eolcom ->
```

  One end of line comment has been used in

```
positive  variable x;     ->non-negative variables
```

2.2.3 Solving a GAMS Model

To solve a GAMS model either press the **F9** function key from the keyboard; or select Run from the File menu



or use the **Run GAMS** button

If the model has no syntax errors, then GAMS opens two new windows: the **listing window** and the **process window**.

- The listing window displays a file with **.lst** extension. In our case the listing file associated with `simpleLp2.gms` is `simpleLp2.lst`. The **.lst** file contains
  - ⋆ a numbered list (echo) of the instructions we haver written in the .gms file
  - ⋆ model statistics
  - ⋆ solution report of our model (Solve Summary)
- the process window shows the progress of the GAMS execution. After GAMS finished running the process window can be used to position the cursor at important positions in the **.lst** file and **.gms** files. In the process window
  - ⋆ double clicking on a *blue line* to position the cursor in the **.lst** file.

- ⋆ if the are (syntax or logical) errors in the GAMS model, then double click on a *red line* so that the cursor will be positioned in the **.gms** file where error might have occurred.

In the **.lst** file, the notations

.LO	lower bound
.L	(starting or current)level
.UP	upper bound.

indicate the status of a variable. Perhaps the most important part of the **.lst** file is the *solution report*.

```
Optimal solution found. Objective :          -5.333333

                        LOWER     LEVEL     UPPER    MARGINAL

---- EQU obj             .         .         .        1.000

  obj  the objective function

---- EQU constr  the i-th constraint

            LOWER     LEVEL     UPPER    MARGINAL

   row1     -INF     -2.667     1.000      .
   row2     -INF      8.000     8.000    -0.667
   row3     -INF       .       10.000      .

                        LOWER     LEVEL     UPPER    MARGINAL

---- VAR z              -INF     -5.333     +INF       .

---- VAR x

         LOWER        LEVEL        UPPER    MARGINAL

   col1      .        2.667         +INF         .
   col2      .         .            +INF       2.333
   col3      .         .            +INF       0.333

**** REPORT SUMMARY :        0     NONOPT
                             0 INFEASIBLE
                             0  UNBOUNDED
```

The solution of the (LOP) is given at

```
---- VAR x

         LOWER        LEVEL        UPPER    MARGINAL
```

```
col1      .         2.667          +INF          .
col2      .          .             +INF        2.333
col3      .          .             +INF        0.333
```

That is $x^* = (2.677, 0, 0)$ is the optimal solution. The dots "." represent 0 values. The variables are declared positive, i.e. the LOWER (bounds) = 0; but they are not bounded from above, hence, UPPER (bound) = +INF. The current level (at optimality) is indicated by LEVEL, corresponding to each index

$$\begin{pmatrix} x(col1) \\ x(col2) \\ x(col3) \end{pmatrix} = \begin{pmatrix} 2.677 \\ 0 \\ 0 \end{pmatrix}.$$

The values under MARGINAL corresponding to each index, represent the **reduced costs** associated with each variable. Marginals corresponding to basic variable are equal to zero. Marginals corresponding to non-basic variables are greater than zero.

Constraints are also associated with MARGINAL values as in

```
---- EQU constr  the i-th constraint

           LOWER     LEVEL     UPPER    MARGINAL

row1       -INF     -2.667     1.000       .
row2       -INF      8.000     8.000     -0.667
row3       -INF        .      10.000       .
```

UPPER indicates the the right hand-side bound on the constraints. LEVEL the current value of the constraints (at the optimal solution). The MARGINAL values here represent **shadow costs**. Note that the first and second constraints are not active, hence marginal values are zeros. The second constraint is active and its marginal value equal to -0.677. That is

$$\frac{\partial z}{\partial b_2} = -0.667.$$

A one unit change in the right hand-side of the second constraints, changes the optimal objective value by -0.667. Sometimes, we find the Marginal values being given as EPS. This represents a value close to zero, but not equal to zero.

2.2.4 Mixed Integer Linear Programming

A mixed integer linear programming problem has the general form

$$
\begin{array}{|ll|}
\hline
c^\top x + d^\top y + e^\top w & \longrightarrow \quad min \\
s.t. & \\
A_1 x + B_1 y + E_1 w \leq b_1, & \\
A_2 x + B_2 y + E_2 w = b_2, & \\
L_1 \leq x \leq U_1, & \\
L_2 \leq y \leq U_2, & \\
x \in \mathbb{R}^n_+, y \in \mathbb{Z}^p_+, w \in \{0,1\}. & \\
\hline
\end{array}
\qquad \text{(MILP)}
$$

Here the variables are interpreted as follows:

- x - continuous variables
- y - integer variables
- w - binary variables.

Variables which are integer or binary are also known as *discrete variables*. Under GAMS mixed integer linear programming problems are identified with the GAMS model type (MIP). These model types does not include non-linear terms.

Well known algorithms for the solution of MILP's:

- Branch & Bound
- Benders Decomposition
- Cutting Plane (Gomory) algorithm
- Branch & Cut

Usually these algorithms are used in combination with the simplex algorithm and/or the interior-point method.

There are more than ten GAMS solvers that can solve (MIP) model types. For instance the solvers:

BARON, BDMLP, CPLEX, LINDOGLOBAL, MOSEK, OSL, XPRESS

are some of the most frequently used ones. However, the student (demo) version of GAMS can solve (MILP)'s with at most 10 discrete variables.

Example 2.2.1. *Solve the following (MILP)*

$$
\begin{aligned}
& 2x_1 + 2x_2 + 4x_3 + 3w_1 + 4w_2 \to \min \\
& s.t. \\
& \quad -x_1 + x_2 + x_3 + 3w_1 - w_2 \geq 6 \\
& \quad x_1 - 3x_2 + x_3 + 2w_1 + 2w_2 \geq 5 \\
& \quad\qquad\qquad\qquad\quad w_1 \leq 2 \\
& \quad x_1, x_2, x_3 \geq 0 \\
& \quad w_1, w_2 \geq 0, \textit{integer}.
\end{aligned}
$$

The above (MILP) can be modeled in (GAMS) as

```
$Title Generating File Outputs (myMilp1.gms) $Ontext
        This demonstrates how to model a simple
        mixed integer linear programming problem
        2x1 + 2x2 + 4x3 + 3w1 + 4w2 -> min
        s.t.
            -x1 + x2  + x3 + 3w1 - w3 >= 6
             x1 - 3x2 + x3 + 2w1 + 2w2 >= 5
                                 w1 <= 2
            x1,x2,x3 >= 0,
            w1, w2 >=0 and integer
$Offtext
$eolcom ->

Sets

         supI  variables index        /i1*i5/
         I1(supI) cont. vars index      /i1*i3/    -> subset of supI
         I2(supI) intger vars index     /i4,i5/    -> a subsets supI
         constInd constraints index    /con1,con2/   ;

Variable  z   objective; positive  variable x(I1);
->non-negative continuous variables

integer  variables w(I2);     ->integer variables, by default
non-negative

Parameters
         cx(I1) coefficient vector of continuous variables
                /i1    2
                 i2    2,
                 i3    4/
         cw(I2) coefficient vector of integer variables
                /i4    3
                 i5    4/
         b(constInd) /
                     con1  6
                     con2  5/;

Table  A(constInd,supI)   coefficient matrix
                I1     I2    I3     I4    I5
         con1   -1      1     1      3    -1
         con2    1     -3     1      2     2;

*Equation declaration

Equations
         obj  the objective function
```

```
        constr(constInd) the i-th constraint;

*Definition of objective and constraints
obj.. z=E=Sum(I1,cx(I1)*x(I1)) + Sum(I2,cw(I2)*w(I2));
constr(constInd)..Sum(I1,A(constInd,I1)*x(I1)) +
Sum(I2,A(constInd,I2)*w(I2)) =G=b(constInd); w.up('i5')=5;
*Naming your model

Model TestMILP1 /ALL/;
*solve the model
Solve TestMILP1 Using MIP Minimizing z;
```

Note the following

- In the definition of the indices (index sets) in

```
Sets

        supI  variables index        /i1*i5/
        I1(supI) cont. vars index      /i1*i3/    -> subset of supI
        I2(supI) intger vars index     /i4,i5/    -> a subsets supI
```

  the sets `I1` and `I2` are defined as subsets of the set `supI`. In GAMS if you have a (an index) set `superSetName`, then you can define a subset of this set with the syntax:

```
Set
      Set subsetName(superSetName)  /list of elements/;
```

  The `superSetName` must have been defined earlier. The set `superSetName` allows us to define a single matrix corresponding to both continuous and discrete variables. While the subsets `I1` and `I2` keep track of continuous and discrete variables, respectively.

- **In GAMS** `integer variables` **are assumed by default to be between 0 and 100**; i.e. they are by default non-negative. Consequently, if you have an upper- or a lower-bound on the integer variables, say as in

$$w1 <= 2$$

  This has been modeled in GAMS as

```
w.up('i4')=5;
```

  For example if we have an additional bound constraint on w_2 like

$$-10 \leq w_2 \leq 30$$

  Instead of defining a GAMS equation constraint for this, we could write

```
w.up('i5')=30; ->upper bound
w.lo('i5')=-10; >lower bound
```

Hence, using these options we can tell GAMS to consider negative integers as well.

Example 2.2.2. *Machine Scheduling Problem.*
A single machine has to execute $n(n \geq 2)$ *jobs.*

- *The processing time for job* i *is* t_i*(minutes).*
- *After each job the machine has to be set up in order to process the next job. The setup time from job* i *to job* j *is* c_{ij} *minutes.*
- *After executing all jobs the machine has to be put into the initial situation* i_0*. The setup from* i_0 *to job* i *takes* c_{0i} *minutes; and from job* j *to the initial situation takes* c_{j0} *minutes.*

2.2.5 Choosing a Solver and Setting Solver Options

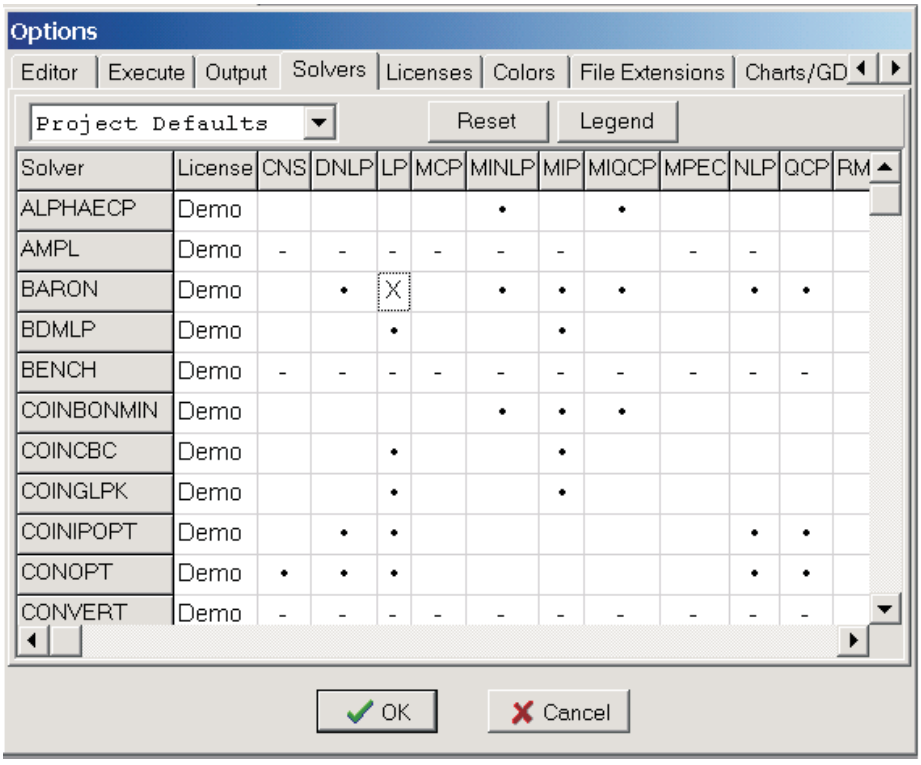
Choosing a Solver

We know that there are several solvers for a given GAMS model type. For instance, inorder to solve a GAMS lp model we could use either

CPLEX, BARON, SNOPT, MOSEK, MINOS, KNITRO, CONOPT, etc.

For GAMS lp models if we do not specify a solver, then GAMS chooses CPLEX by default. However, we can specify another lp solver in three different ways.

1. From the user interface use the **File** menu choose **options** then from the **options window** click on the **solvers tab**. From the list of the solvers choose a solver that is capable of solving the given GAMS model. (Put an x by selecting (with mouse) the positing where a GAMS model type and a solver meet).



GAMS options window with BARON selected for lp.

2. In the GAMS model of your problem choose a solver by using the syntax

```
option GAMS_ModelType = solverName;
```

This option statement must come before the particular solve statement. For example we can use

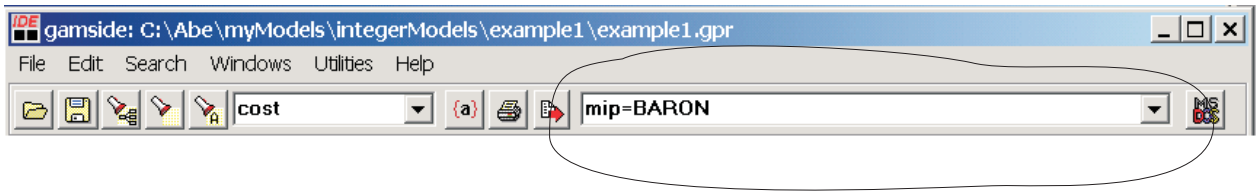
```
option lp = BARON;
```

Consequently, the `option` statement allows us to run and test a given GAMS model using several solvers in a single GAMS execution. For example in the model `myMilp1.gms` we could use the following option statements:

```
Solve TestMILP1 Using MIP Minimizing z;

option lp = BARON;

Solve TestMILP1 Using MIP Minimizing z;

option lp = MOSEK;

Solve TestMILP1 Using MIP Minimizing z;
```

The first solve calls the default `mip` solver (cplex), the second BARON and the third MOSEK. However, when calling several solvers in a single GAMS execution, the listing file will be cluttered with solver outputs. Thus, the best way to manage solver outputs is to send the most important outputs into an external output file.

3. The third way to specify a solver is by using the **command line parameter box**:



The solver option statements which are specified by using the last two methods will be saved along with the GAMS model and are available when we open the model in GAMS another time.

Setting Solver Parameters

Every solver under GAMS has certain parameters associated to it. The parameters in turn have their corresponding possible values. To find out what parameters a given solver has, you need to consult the documentation of that particular solver. Thus, when you decide to use a given solver for your GAMS model, GAMS uses default parameter values associated with that solver. Usually, the default parameter values are enough to properly solve a given GAMS model. However, you can also change the default parameter values into values of your wish.

To tell GAMS to use other solver parameter values than the default ones, you need to create an **option file**. Furthermore, to cause a solver to use an option file, it is necessary to set the **optfile** model attribute to a *positive value*

```
model modelName /all/ ;
option modelType = solverName;
modelName.optfile = 1 ;
solve modelName using modelType maximizing/minimizing Obj ;
```

Where

- `modelName` - the name that you gave to your model
- `Obj` - the objective function variable
- `modelType` - any one of lp, mip, nlp, etc.;
- `solverName` - represents names of solvers like CPLEX, BARON, MINOS, etc.

Creating an Option File

The option file that you need to create and save under your project folder must have the following from

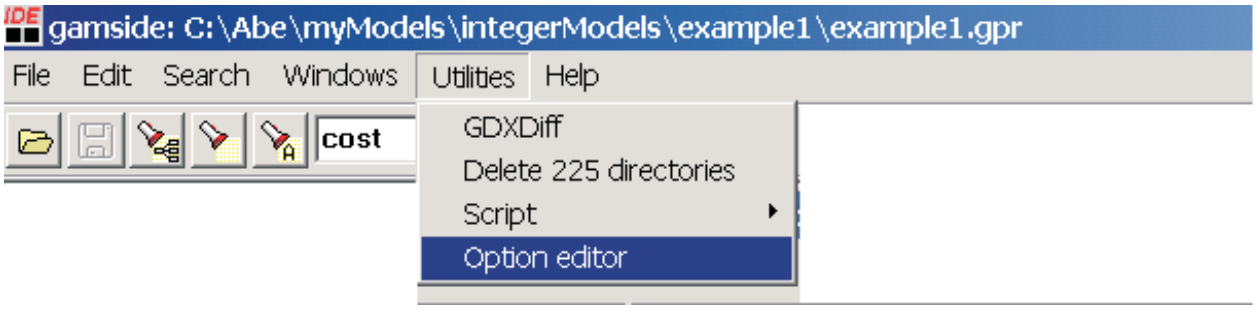
```
solverName.ext
```

The file extension `.ext` depends on the value that assigned to `modelName.optfile` attribute. Some of these values are listed below:

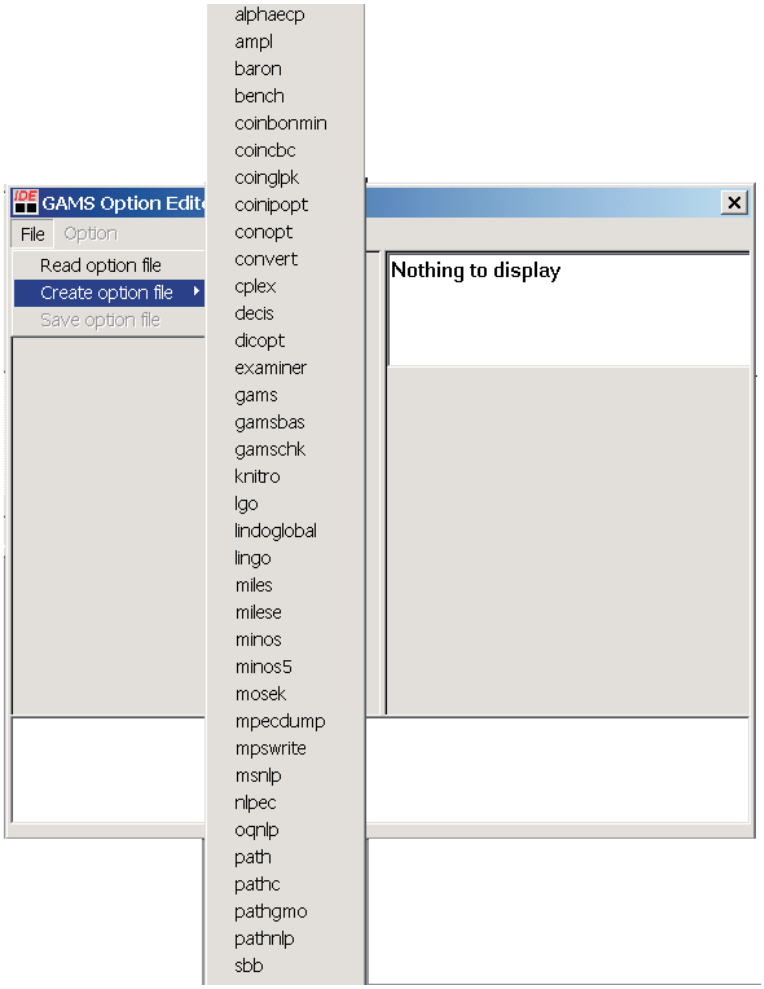
modelName.optfile	.ext	Example
0	default (no option)	
1	.opt	CEPLEX.opt
2	.op2	CEPLEX.op2
3	.op3	BARON.op3
⋮	⋮	⋮
10	.o10	MINOS.010
⋮	⋮	⋮
99	.o99	CEPLEX.o99
100	.100	CEPLEX.100
⋮	⋮	⋮
999	.999	CEPLEX.999

Consequently, you can create several option files for a given GAMS model and you can call each of them by change the `modelName.optfile` attribute accordingly.

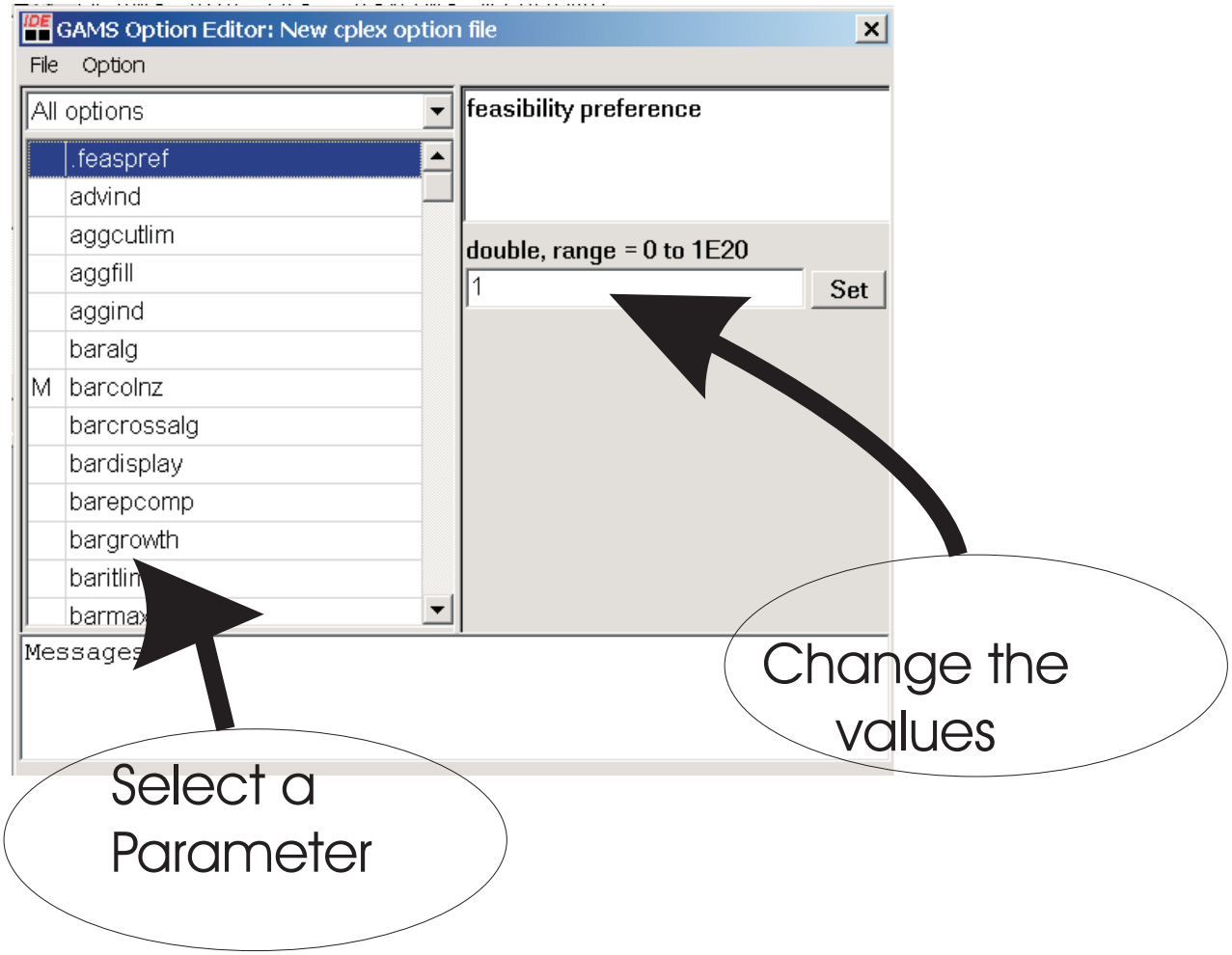
The easiest way to create an option file is to use the GAMS option files editor. To open the option file editor from the GAMS menu **Utilities > Option Editor**



From the **Option File Editor** window point on the **Create Option file** submenu. Then you will see a list of all available GAMS solvers for which you can create an option file. From this list select the solver for which you wanted to create an option file.



For instance the following a shows a list of parameters and their corresponding default values for CPLEX as displayed in GAMS File Editor window



Once again to find out the meaning if each parameter and their values you have to read the manual for each solver. After editing the value of a parameter save the option file using the file name `solverName.ext` as has been discussed earlier. For example if we have had the following code in `myMilp1.gms`,

```
model TestMILP1 /All/;
option lp = BARON;
TestMILP1.optfile = 2;
Solve TestMILP1 Using MIP Minimizing z;
```

then the option file would have been saved using the file name `baron.op2`.

2.2.6 Outputting Solver Report to a File

Usually, GAMS solver output are not easy to read. Particularly, if you'r looking for only a solution report, then the better way is to tell GAMS to produce an output file which is easier to read. Thus, you might decide to produce a text file containing only part of the solver output.

```
File filePointer /fileName/;
Put filePointer;
Put item1;
   .
   .
   .
Put itemN;
Putclose filePointer;
```

- the words `File, Put, Putclose` are (GAMS) key words. `File` defines a file into which you want to save items. Every item you want to save into your file must be preceded with the key word `Put`. When you have finished putting data into a file, then you must close the file using `Putclose`.
- `fileName` - your file will be saved with this name. This can have file extension `.txt` (text file), or `.xls` (Excel file), or `.opt` (option file), etc. Furthermore, you might need to specify the folder path into which you want to save your file. If you do not specify a path, GAMS saves the file into your current project directory.
- `filePointer` - a name you give to refer to your output file in your GAMS program. This name acts just like a file pointer.

The following GAMS program explains how you can generate a simple text report file:

```
$Title Generating File Outputs (myreport.gms)
$Ontext
        This file explains how to generate an output
        report file myreport2.txt from GAMS in such a formatted way
        so that it is easily readable.

$Offtext $eolcom->
Sets
         I row indices /row1,row2,row3/
         J column indices /col1,col2,col3/;

*objective function variable and problem variables
Variables z, x(J); positive  variable x;     ->non-negative
variables

Parameters
         c(J) coefficient vector of objective
                /col1   -2
                 col2    3,
```

```
                  col3   -1/
         b(I) right-hand side vector of constraints
                 /row1    1
                  row2    8
                  row3    10/;
Table  A(I,J)   coefficient matrix
               col1      col2    col3
         row1   -1        1       -1
         row2    3       -1        2
         row3    0        2        5   ;

*Equation declaration
Equations
         obj  the objective function
         constr(I) the i-th constraint;

*Definition of objective and constraints
obj.. z=E=Sum(J,c(J)*x(J)); constr(I)..Sum(J,A(I,J)*x(J)) =L=b(I);

*Naming your model
Model TestLp2 /ALL/;

*solve the model
Solve TestLp2 Using LP Minimizing z;    ->solves the problem

Option x:2;    ->displays x with 2 decimal places

File myreport2_file /myreport2.txt/;
Put myreport2_file;

Put ' ' :<20,
    'x1':<>10,
    'x2 ':<>10,
    'x3':<>10/;
Put
    'Optimal Soultion:':<>20,
    x.l('col1'):<>10,
    x.l('col2'):<>10,
    x.l('col3'):<>10/;
Put
    'Reduced Costs:':<>20,
    x.m('col1'):<>10,
    x.m('col2'):<>10,
    x.m('col3'):<>10/;
Put /; put ' ' :<20,
    'constr1':<>10,
    'constr2 ':<>10,
    'constr3':<>10/;
```

```
Put
    'Constraint Level:':<>20,
    constr.l('row1'):<>10,
    constr.l('row2'):<>10,
    constr.l('row3'):<>10/;


Put
    'Shadow Prices:':<>20,
    constr.m('row1'):<>10,
    constr.m('row2'):<>10,
    constr.m('row3'):<>10/;

*close the output file
Putclose   myreport2_file;
```

When putting items into a file, you can decide in what format they should be saved. In general you have the syntax:

```
Put item:justificationSymbol width:decimalPlaces;
```

where

- `width` - number of character spaces to be reserved for the item to be saved.
- `justificationSymbol` is any one of the symboly >, <>, < to determine the justification of the item in the width reserved.

>	right justified
<>	centered
<	left justfied

- `decimalPlace` - determines the number of decimal places in a numerical ouput.
- Use / to force line brake in the output.

2.3 Importing Excel Data into GAMS

There are two ways to import and use data from an Excel spreadsheet into GAMS. Either using the GAMS GDX facility or using the XLS2GMS toolbox. However, in many cases importing external data into GAMS does not go as smoothly as you may wish. Here we will see how to import an Excel data into GAMS.

Before you use data from an excel spreadsheet, you need to extract the required part of the data and save it into a include file. You do this using the following syntax:

```
$call =XLS2gms i=inputExcelFile o=outputIncludeFile r=sheetNum!range
```

where

- `$call` - used in call an external software routine to perform a task. In this case `call` calls the external toolkit `XLS2GMS`.
- `XLS2gms` a toolkit that converts excel data into GAMS include data. Depending on your version of GAMS use the full path for XLS2GMS under your GAMS installation, for instance like:

```
c:\Programme\GAMS22.5\xls2gms
```

- `i=inputExcelFile` - `i` stands for input; `inputExcelFile` is the name of the input file with `.xls` extension. It may be necessary to specify the full path where this file has been saved in, like:

```
i=c:\GAMSProjects\prject1\myData.xls
```

- `o=outputIncludeFile` - `o` stands for output; `outputIncludeFile` is name of the file with file extension `.inc` into which to save the extracted data.
- a given Excel file may contain several spreadsheets numbered `Sheet1`, `Sheet2`, etc. Thus, you can tell GAMS which sheet to read and the range of the data using

```
r=sheetNum!range.
```

  For instance as in : `r=Sheet2!r=a3:g6`.

Bibliography

[1] B. Jansen, C. Roos, T. Terlaky, An interior point approach and parametric analysis in linear programming. Delft University of Technology, Faculty of Technical Mathematics and Informatics, Report 92-21.

[2] S. Mehrotra, On th implementation of a (primal-dual) interior point method. SIAM J. on Optimization, pp. 575-601, V. 2, No.4, 1992.