Contents

B  Mathematical Programming 3
   B.1  Linear programming ........................................... 3
       B.1.1  Fundamentals ............................................. 3
       B.1.2  The Simplex mechanism .................................. 6
       B.1.3  Sensitivity and duality .................................. 8
   B.2  Mixed-integer linear programming ............................. 11
       B.2.1  Branch and bound ......................................... 12
   B.3  Nonlinear programming. Optimality conditions .............. 13
       B.3.1  Unconstrained problems .................................. 14
       B.3.2  Constrained problems .................................... 14
   B.4  Unconstrained problems. Solution methods ................... 17
       B.4.1  Steepest-descent method ................................ 17
       B.4.2  Newton and Quasi-Newton methods ...................... 20
       B.4.3  Conjugate directions methods ............................ 22
       B.4.4  Coordinate descent method ............................... 22
   B.5  Constrained problems. Solution methods .................... 23
       B.5.1  Penalty and barrier methods .............................. 23
       B.5.2  Augmented Lagrangian methods .......................... 27
       B.5.3  Primal-dual interior point methods ..................... 29
Appendix B

Mathematical Programming

ANTONIO J. CONEJO

This appendix describes the fundamentals of mathematical programming, both linear and nonlinear. From a practical viewpoint, it is convenient to use an environment for the formulation and solution of mathematical programming (optimization) problems, such as GAMS [1] or AMPL [2], which allows focusing on modeling not on solution algorithms. Nevertheless, to assess results it is appropriate to be familiar with the working principles of the optimization algorithms. This is why this appendix provides an overview of practical algorithms to solve optimization problems. Prior to the description of the algorithms, this appendix also provides the theory underlying these algorithms.

Concerning models based on mathematical programming as well as solution algorithms, the books by Castillo et al. [3] and Conejo et al. [4] might interest the reader.

B.1 Linear programming

Linear programming allows formulating many problems related to control, operation, planning, economics and regulations of electric energy systems. On the other hand, the solution of a well-formulated linear programming problem (linear problem) can always be found. Thus, linear programming is a robust modeling technique. Moreover, it is possible to solve very large linear programming problems, involving hundreds of thousands of variables and constraints, using personal computers in seconds. Excellent books on linear programming includes Bazaraa et al. [5], Chvátal [6] and Luenberger [7].

B.1.1 Fundamentals

Consider the linear programming problem

\[
\begin{align*}
\text{minimize}_{x_1, x_2} & \quad z = -3x_1 - 5x_2 \\
\text{subject to} & \quad 3x_1 + 2x_2 \leq 18 \\
& \quad x_1 \leq 4, \quad x_2 \leq 6, \quad x_1 \geq 0, \quad x_2 \geq 0.
\end{align*}
\] (B.1)

Figure B.1 depicts the geometry of the feasible region (points that meet constraints)
of this problem. This feasible region is a polygon. As shown below, the corners of this polygon are of particular interest.

In problem (B.1), it can be observed that contour lines of the objective function are straight lines. Note also that the minimizer, feasible point in which the objective function attains its minimum value, coincides with a corner of the feasibility region.

![Figure B.1: Example of a linear programming problem.](image)

Motivated by the structure of problem (B.1), the canonical form of any linear programming problem is defined as

\[
\begin{align*}
\text{minimize} \quad & z = c^T x \\ 
\text{subject to} \quad & Ax \leq b, \quad x \geq 0, \quad x \in \mathbb{R}^n,
\end{align*}
\]  

(B.2)

where \(x\) is the \(n\)-dimensional unknown vector, \(c\) is the \(n\)-dimensional cost vector, \(A\) is the \(m \times n\) constraint matrix and \(b\) is the \(m\)-dimensional right-hand-side resource vector.

Linearity of both the objective function and the constraints confers problem (B.2) a singular structure. On one hand, constraint linearity makes the feasible region to be a convex polyhedron. On the other hand, the linearity of the objective function makes its gradients constant and thus the objective function always increases along certain directions and decreases along the opposite ones. Therefore, the frontier of the feasibility region and particularly its vertices are important. Contour curves abandon the feasible region through just one point (single optimal solution) or through a face of the polyhedron (multiple optimal solutions). It might also happens that contour curves never abandon the feasible region (unbounded problem).

An alternative form of expressing any linear programming problem, denominated standard form, is

\[
\begin{align*}
\text{minimize} \quad & z = c^T x \\ 
\text{subject to} \quad & Ax = b, \quad x \geq 0, \quad x \in \mathbb{R}^n.
\end{align*}
\]  

(B.3)
Without loss of generality vector $b$ is considered non-negative. A problem formulated in standard form requires $m$ to be smaller that or equal to $n$ (generally $m \leq n$).

As illustrated through problem (B.1), the feasible region of a linear programming problem is a convex polyhedron. Moreover, the standard form of a linear programming problem allows characterizing algebraically the vertices of this convex polyhedron, which are the candidate points to minimizers.

Since the most common form is the standard form, rules to convert any problem to standard form are stated below:

1) Constraints of the type greater than or equal to are converted to equality constraints through nonnegative surplus variables $(y_i)$. The constraint $\sum_j a_{ij}x_j \geq b_i$ is equivalent to the two constraints below

\[
\sum_j a_{ij}x_j - y_i = b_i \quad \text{and} \quad y_i \geq 0.
\] (B.4)

2) Similarly, constraint $\sum_j a_{ij}x_j \leq b_i$ is equivalent to the two constraints below

\[
\sum_j a_{ij}x_j + y_i = b_i \quad \text{and} \quad y_i \geq 0.
\] (B.5)

In this case, the variable $y_i$ is denominated slack variable.

3) If a variable is free, $-\infty < x_i < \infty$, it can be substituted by the difference of two nonnegative variables, i.e.,

\[
x_i = y_i - z_i, \quad y_i \geq 0, \quad z_i \geq 0.
\] (B.6)

4) Finally, note that maximizing a given objective function $z$ is equivalent to minimizing its opposite $-z$.

The Example (B.1) is written below in standard form ($x_3$, $x_4$ and $x_5$ are slack variables):

minimize $z = -3x_1 - 5x_2$
subject to
\[
\begin{align*}
x_1 + x_3 &= 4 \\
x_2 + x_4 &= 6 \\
3x_1 + 2x_2 + x_5 &= 18 \\
x_1 &\geq 0, \quad x_2 \geq 0, \quad x_3 \geq 0, \quad x_4 \geq 0, \quad x_5 \geq 0.
\end{align*}
\] (B.7)

Using the standard form of a linear programming problem, consider the partition

\[
A = [B|N], \quad x = \begin{bmatrix} x_B \\ x_N \end{bmatrix}, \quad c = \begin{bmatrix} c_B \\ c_N \end{bmatrix},
\] (B.8)

where $B$ is a $m$-dimensional full rank square matrix denominated the basis, $N$ is a $m \times (n-m)$ matrix, $x_B$ is a $m$-dimensional vector containing so-called basic variables (we say that these variables are in the basis), $x_N$ is a $(n-m)$-dimensional vector containing so-called nonbasic variables (we say that these variables are not in the basis), $c_B$ is a $m$-dimensional cost vector corresponding to basic variables and $c_N$ is a $(n-m)$-dimensional cost vector corresponding to nonbasic variables.

We say that a vector $x$ is a solution if $Ax = b$, and a feasible solution if $Ax = b$, $x \geq 0$, and a basic feasible solution if $Ax = b$, $x \geq 0$, $x_N = 0$. Observe that a basic solution, feasible
or not, can be generated computing $x_B = B^{-1}b$ and enforcing $x_N = 0$. It is relevant to note that the extreme points of \{Ax = b, x \geq 0\} are indeed basic feasible solutions. This is illustrated below.

We consider again the Example (B.1) and obtain all basic feasible solutions. We show below the interest of these solutions, which can be obtained constructing all possible matrices $B$. The number of matrices $B$ is given by the combination of 5 columns taken 3 at a time. All basic feasible solutions are (non feasible ones are not provided):

$$
\begin{bmatrix}
  x_1 \\
  x_2 \\
  x_3 \\
\end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\
  0 & 1 & 0 \\
  3 & 2 & 0 \\
\end{bmatrix}^{-1} \begin{bmatrix} 4 \\
  6 \\
  18 \\
\end{bmatrix} = \begin{bmatrix} 2 \\
  6 \\
  2 \\
\end{bmatrix}, \quad z = -36.
$$

$$
\begin{bmatrix}
  x_1 \\
  x_2 \\
  x_4 \\
\end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\
  0 & 1 & 1 \\
  3 & 2 & 0 \\
\end{bmatrix}^{-1} \begin{bmatrix} 4 \\
  6 \\
  18 \\
\end{bmatrix} = \begin{bmatrix} 4 \\
  3 \\
  3 \\
\end{bmatrix}, \quad z = -27.
$$

$$
\begin{bmatrix}
  x_1 \\
  x_4 \\
  x_5 \\
\end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\
  0 & 1 & 0 \\
  3 & 0 & 1 \\
\end{bmatrix}^{-1} \begin{bmatrix} 4 \\
  6 \\
  18 \\
\end{bmatrix} = \begin{bmatrix} 6 \\
  6 \\
  6 \\
\end{bmatrix}, \quad z = -12.
$$

$$
\begin{bmatrix}
  x_2 \\
  x_3 \\
  x_5 \\
\end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\
  1 & 0 & 0 \\
  2 & 0 & 1 \\
\end{bmatrix}^{-1} \begin{bmatrix} 4 \\
  6 \\
  18 \\
\end{bmatrix} = \begin{bmatrix} 6 \\
  6 \\
  6 \\
\end{bmatrix}, \quad z = -30.
$$

$$
\begin{bmatrix}
  x_3 \\
  x_4 \\
  x_5 \\
\end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\
  0 & 1 & 0 \\
  0 & 0 & 1 \\
\end{bmatrix}^{-1} \begin{bmatrix} 4 \\
  6 \\
  18 \\
\end{bmatrix} = \begin{bmatrix} 4 \\
  6 \\
  6 \\
\end{bmatrix}, \quad z = 0.
$$

We observe that each basic feasible solution corresponds to a corner of the feasible polygon of Figure B.1. The geometric analysis of the example shows that the minimizer is located at one of these corners. Therefore, we infer that the minimizer correspond to a basic feasible solution. Generally, it can be shown that this is so. Thus, a possible strategy to locate the minimizer is to obtain all basic feasible solutions and select the one with smallest objective function value. The fundamental theorem of linear programming, which we do not show (see, e.g., [7] for a formal proof), says precisely so. That is, if a linear programming problem has a solution, this solution is a basic feasible solution.

Finally, note that finding all basic feasible solutions is computationally costly, as an upper bound of the number of basic feasible solutions is $\frac{n!}{m!(n-m)!}$.

B.1.2 The Simplex mechanism

The algorithm below is denominated revised Simplex and allows solving linear programming problems, jumping form basic feasible solution to basic feasible solution while decreasing the objective function. The considered problem is

$$
\begin{align*}
\text{minimize}_x \quad & z = c^T x \\
\text{subject to} \quad & Ax = b, \quad x \geq 0, \quad x \in \mathbb{R}^n.
\end{align*}
$$

(B.9)
Using the partition (B.8), we write

\[
\begin{align*}
\text{minimize}_{x_B, x_N} & \quad z = c_B^T x_B + c_N^T x_N \\
\text{subject to} & \quad B x_B + N x_N = b \\
& \quad x_B \geq 0, \quad x_N \geq 0, \quad x_B \in \mathbb{R}^m, \quad x_N \in \mathbb{R}^{n-m}.
\end{align*}
\]

(B.10)

As matrix $B$ is invertible, we can write $x_B = B^{-1} b - B^{-1} N x_N$, and the objective function then becomes $z = c_B^T B^{-1} b - c_B^T B^{-1} N x_N + c_N^T x_N$.

We define below certain vectors and matrices that remain constant as long as the basis $B$ remains constant,

\[
\begin{align*}
\tilde{b} &= B^{-1} b, \quad \lambda^T = c_B^T B^{-1}, \quad Y = B^{-1} N, \quad d^T = \lambda^T N - c_N^T,
\end{align*}
\]

where $\tilde{b}$ is an $m$-dimensional column vector, $\lambda$ is an $m$-dimensional column vector, $Y$ is a $m \times (n-m)$ matrix, and $d$ is an $(n-m)$-dimensional column vector. Substituting the above matrix and vectors in (B.10), we obtain

\[
\begin{align*}
\text{minimize}_{x_B, x_N} & \quad z = c_B^T \tilde{b} - d^T x_N \\
\text{subject to} & \quad x_B = \tilde{b} - Y x_N \\
& \quad x_B \geq 0, \quad x_N \geq 0, \quad x_B \in \mathbb{R}^m, \quad x_N \in \mathbb{R}^{n-m},
\end{align*}
\]

(B.12)

where vector $d$ is the so-called reduced cost vector, and $\lambda$ is a highly relevant sensitivity vector (as analyzed below).

In problem (B.12), the objective function depends only on nonbasic variables and the equality constraints express basic variables as a function of nonbasic variables. This formulation allows switching from one basic feasible solution to another one. This is achieved increasing a nonbasic variable until a basic one reaches zero, which implies that the nonbasic and the basic variables swap status.

The simplex algorithm uses formulation (B.12) for moving among basic feasible solutions with the target of decreasing the objective function value until no further decrement is possible, which implies that the optimal solution has been reached. Observe that:

Observation 1. If the element $d_j$ of the reduce cost vector is positive, then the objective function value decreases as the nonbasic variable $x_{Nj}$ increases.

Observation 2. For fixed $j$, if some elements $Y_{ij}$’s are positive, the corresponding basic variables $x_{Bi}$ decrease provided that the nonbasic variable $x_{Nj}$ increases.

Observation 3. To move from basic feasible solution to basic feasible solution, the nonbasic variable $x_{Nj}$ can be increased until the first basic variable reaches zero, that is, until $\tilde{b}_i - Y_{ij} x_{Nj}$ becomes zero for the first $i$. The value of $x_{Nj}$ becomes the

\[
\text{minimum}_{1 \leq i \leq m} \left\{ \frac{\tilde{b}_i}{Y_{ij}} : Y_{ij} > 0 \right\},
\]

while $x_{Bi}$ becomes 0.
Taking into account the above observations, the revised Simplex algorithm works as follows:

Step 1. Obtain an initial basic feasible solution. We analyze below how to do that.

Step 2. Check if the current solution is optimal. This is so if all element of the reduced cost vector are nonpositive. If not, continue.

Step 3. Find out which nonbasic variable should enter the basis. This can be done selecting the nonbasic variable whose corresponding element in the reduced cost vector has the highest positive value. Note that other selection rules are possible.

Step 4. Select the basic variable leaving the basis through the criterion stated in observation 3 above.

Step 5. Build the new basis, get a new basic feasible solution and continue in step 2.

The Simplex algorithm needs an initial feasible solution to start with. An initial basic feasible solution for any linear programming problem can be obtained solving the linear programming problem below, which has a trivial initial basic feasible solution \((y = b, x = 0)\).

\[
\begin{align*}
\text{minimize}_{x,y} & \quad z = \sum_{i=1}^{m} y_i \\
\text{subject to} & \quad Ax + y = b; \quad x \geq 0, \quad y \geq 0, \quad x \in \mathbb{R}^n, \quad y \in \mathbb{R}^m.
\end{align*}
\]

(S.13)

Solving the problem above is referred to as Phase I of the Simplex algorithm, being the Phase II the actual solution of the considered problem.

Finally, note that it might happen that the value of a basic variable is zero. We say that such solution is degenerated. In rare cases, degeneracy might lead to a cycling behavior of the Simplex algorithm. However, appropriate trick to avoid such behavior are available. Check, for instance, [7].

B.1.3 Sensitivity and duality

Sensitivity

We analyze below the so-called sensitivities. These parameters allow deriving fruitful information from the solution of a linear programming problem. Sensitivities are of high practical interest as they provides information on how the objective function marginally changes as certain parameters of the problem change.

Being \(B^*\) the basis of the optimal solution of a linear programming problem, \(x_B^*\) the basic variable vector and \(z^*\) the objective function optimal value, then

\[
x_B^* = B^{*-1}b \quad \text{and} \quad z^* = c_B^T x_B^*.
\]

We consider marginal changes in the right-hand-side vector \(b\) so that the basis remains unchanged, that is, \(b^* \to b^* + \Delta b\), and \(B^*\) unchanged.

This marginal change in \(b\) produces marginal changes in the basic variable vector and in the objective function optimal value, \(x_B^* \to x_B^* + \Delta x_B\) and \(z^* \to z^* + \Delta z\).
Taking into account (B.14), these marginal changes can be expressed as
\[ \Delta x_B = B^* B^{-1} \Delta b \quad \text{and} \quad \Delta z = c_B^T \Delta x_B, \] (B.15)
and
\[ \Delta z = c_B^T B \Delta x_B = c_B^T B^* B^{-1} \Delta b. \] (B.16)

Taking into account that \( c_B^T B^* B^{-1} \) has been denoted by \( \lambda^* T \), then \( \Delta z = \lambda^* T \Delta b \). That is,
\[ \lambda^*_j = \frac{\Delta z}{\Delta b_j}, \quad \forall j = 1, 2, \ldots, m. \] (B.17)

Therefore, \( \lambda^*_j \) is the marginal change of the objective function originated by a marginal change in the right-hand-side vector corresponding with constraint \( j \), provided that the basis remain unchanged.

Duality

Duality is a mathematical concept that links two mathematical structures and allows extracting information from one of these structures based on information available from the other one.

Duality theory is introduced below through an illustrative example.

An oil factory manufactures an artificial oil that contains different amounts of \( m \) lubricants. The artificial oil is produced mixing up \( n \) natural oils easily found in the market at prices \( c_1, c_2, \ldots, c_n \). We denote by \( a_{ij} \) the quantity of lubricant \( i \) contained in one unit of natural oil \( j \). The minimum amounts of lubricants that should contain the artificial oil are \( b_1, b_2, \ldots, b_m \). If \( x_1, x_2, \ldots, x_n \) are the unknown quantities of natural oils to be mixed up so that the cost of the artificial oil is minimum and meets lubricant content requirements, the problem below allows computing optimal values for those unknown quantities,

\[
\begin{align*}
\text{minimize} & \quad z = \sum_{j=1}^n c_j x_j \\
\text{subject to} & \quad \sum_{j=1}^n a_{ij} x_j \geq b_i, \quad i = 1, 2, \ldots, m \\
& \quad x_j \geq 0, \quad j = 1, 2, \ldots, n.
\end{align*}
\] (B.18)

On the other hand, we consider that a lubricant factory manufactures the \( m \) lubricants required to obtain the artificial oil and sell them in the market at prices \( \lambda_1, \lambda_2, \ldots, \lambda_m \). For this business to thrive, the cost of producing any natural oil out of lubricants should be below (or equal to) the market price of that natural oil, that is,
\[ \sum_{i=1}^m \lambda_i a_{ij} \leq c_j; \quad j = 1, 2, \ldots, n. \] (B.19)
For the lubricant factory to maximize its profit, the lubricants should be sold at prices \(\lambda_1, \lambda_2, \ldots, \lambda_m\) obtained by solving the linear programming problem,

\[
\begin{align*}
\text{maximize}_{\lambda_1, \ldots, \lambda_m} & \quad z = \sum_{i=1}^{m} \lambda_i b_i \\
\text{subject to} & \quad \sum_{i=1}^{m} \lambda_i a_{ij} \leq c_j, \quad j = 1, 2, \ldots, n \\
& \quad \lambda_i \geq 0, \quad i = 1, 2, \ldots, m.
\end{align*}
\] (B.20)

The two problems above are dual problems and contain the same components but used differently. Note that they constitute two ways to look at the same reality. These two ways are complementary.

Thus, we say that the dual problem of the linear programming problem

\[
\begin{align*}
\text{minimize}_x & \quad z = c^T x \\
\text{subject to} & \quad Ax \geq b; \quad x \geq 0, \quad x \in \mathbb{R}^n
\end{align*}
\] (B.21)

is the linear programming problem

\[
\begin{align*}
\text{maximize}_\lambda & \quad z = \lambda^T b \\
\text{subject to} & \quad \lambda^T A \leq c^T; \quad \lambda \geq 0, \quad \lambda \in \mathbb{R}^m.
\end{align*}
\] (B.22)

The duality relates the variables of one problem with the constraints of its dual, following strict rules. For linear programming problems duality is symmetric, i.e., the dual of the dual is the primal.

Conversion dual/primal rules are stated below without proof [3]. If the primal problem objective direction is minimizing, the dual problem objective direction is maximizing. Less than or equal to constraints of the primal problem lead to nonpositive variables in the dual problem, while greater than or equal to constraints in the primal problem lead to nonnegative variables in the dual problem. Equality constraints of the primal problem originate free variable in the dual problem.

If, on the other hand, the primal problem objective direction is maximizing, the objective direction of the dual problem is minimizing. Less than or equal to constraints in the primal problem originate nonnegative variables in the dual problem, while greater than or equal to constraints in the primal problem lead to nonpositive variables in the dual problem. Equality constraints in the primal problem originate free variables in the dual problem.

Additionally, note that the dual problem of the dual problem is the primal problem.

We formulate below four theorems that relate the primal and dual problems.

Consider the asymmetrical couple of dual problems

\[
\begin{align*}
\text{minimize}_x & \quad c^T x \\
\text{subject to} & \quad Ax = b, \quad x \geq 0, \quad x \in \mathbb{R}^n
\end{align*}
\] (B.23)

and

\[
\begin{align*}
\text{maximize}_\lambda & \quad \lambda^T b \\
\text{subject to} & \quad \lambda^T A \leq c^T, \quad \lambda \in \mathbb{R}^m.
\end{align*}
\] (B.24)
We say that this couple of linear programming problems is asymmetrical because the primal one only contains equality constraints while the dual one only contains inequality constraints.

The weak duality theorem states that if $x$ is a feasible solution for the primal problem (B.23) and $\lambda$ is a feasible solution for the dual problem (B.24), then $\lambda^T b \leq c^T x$ holds.

The strong duality theorem states that if $x^*$ is the minimizer of the primal problem (B.23), then, there exists a vector $\lambda^*$ that is the maximizer of the dual problem (B.24), and such that $\lambda^T b = c^T x^*$ holds.

Using the strong duality theorem is possible to show that the dual variables are sensitivities. This is why $\lambda$ is used to denote both dual variables and sensitivities.

The optimal value of the primal problem objective function is $z^* = c^T x^* = c^T B x^*_B$. The vector of optimal basic variables is $x^*_B = B^{* -1} b$. Then, $z^* = c^T x^* = c^T_B B^{* -1} b$. Using the strong duality theorem ($c^T x^* = \lambda^T b$), we conclude that $\lambda^T = c^T_B B^{* -1}$. That is, dual variables and sensitivities coincide.

Considering a couple of asymmetrical dual linear programming problems, the complementary slackness theorem states that

$$
\lambda^*_j (A_j x^* - b_j) = 0 \\
x^*_i (\lambda^T A_i - c_i) = 0.
$$

(B.25)

Note that the conditions above implies that if a sensitivity is nonnull then the corresponding constraint is binding (it is satisfied as an equality), and conversely, if a constraint is not binding (it is satisfied as an inequality) then the corresponding sensitivity is null.

## B.2 Mixed-integer linear programming

A mixed-integer linear programming problem is a linear programming problem that includes integer variables. If the integer variables are binary (most common case) the problem is denominated 0/1 mixed-integer linear programming problem. On the other hand, if all variables are integer, the problem is denominated strict mixed-integer linear programming problem.

A mixed-integer linear programming problem is then formulated as

$$
\begin{align*}
\text{minimize} & \sum_{j=1}^{n} c_j x_j \\
\text{subject to} & \sum_{j=1}^{n} a_{ij} x_j = b_i; \quad \forall i = 1, 2, \ldots, m \\
& x_j \geq 0; \quad \forall j = 1, 2, \ldots, n \\
& x_j \in I; \quad \forall j = 1, 2, \ldots, o \\
& x_j \in \mathbb{R}; \quad \forall j = o + 1, \ldots, n.
\end{align*}
$$

(B.26)

Textbook addressing mixed-integer linear programming problems include Nemhauser and Wolsey [8] and Castillo et al. [4] Solution technique are based on branch and bound strategies, and most recently on efficacious branch and cut strategies [9].
B.2.1 Branch and bound

The branch and bound technique is based on solving a sequence of progressively more constrained linear programming problems. These additional constraints partition the feasibility region into subregions facilitating the solution of the original mixed-integer linear programming problem. Initially, upper and lower bounds of the objective function optimal value are established and then, through branching strategies, the upper bound is progressively decreased and the lower bound increased as better solutions of the mixed-integer linear programming problem and its relaxations are generated. The gap between these upper and lower bounds provides a measure of the quality of the current solution.

If minimizing, note that a lower bound of the objective function optimal value is obtained by relaxing integrality constraints and solving the resulting problem. And an upper bound for the optimal objective function value is the objective function value corresponding to any solution meeting integrality constraints.

The branch and bound algorithm works as follows:

Step 1. Initialization. Establish an upper bound (\(\infty\)) and a lower bound (\(-\infty\)) of the objective function optimal value. Solve the initial problem relaxing integrality constraints. If the solution obtained meets integrality conditions, this solution is the minimizer and the procedure concludes. On the other hand, if the relaxed problem is infeasible, the original problem is infeasible and the procedure concludes. Otherwise, the algorithm continues in step 2.

Step 2. Branching. Selecting a variable that does not meet its integrality constraint, two problems (branches) are generated from the previous problem as indicated below. If the value of variable \(x_k\) is “a” (e.g. \(x_k = 5.3\)), the first branch problem is the original problem incorporating the constraint \(x_k \leq [a]\), where \([\,]\) denotes the integer part of \(a\) (e.g. \(x_k \leq 5\)); and the second branch problem is the original problem incorporating the constraint \(x_k \geq [a] + 1\) (e.g. \(x_k \geq 6\)). The above problems are added to a list of problems to be processed, either sequentially or in parallel. Observe that this partition procedure completely covers the whole feasible region.

Step 3. Solution. Solve the first problem in the list.

Step 4. Updating and bounds. If the solution of the current problem meets integrality constraints and its objective function value is below the current upper bound, the upper bound is updated, and the solution stored as the best solution so far. Else if integrality constraints are not satisfied and the objective function value is in between the lower and upper bounds, the lower bound is updated, and the current problem leads through branching to new problems, which are added to the list of problems to be processed.

Step 5. Cutting. If the solution of the considered problem meets integrality constraints, no further branching is possible and the branch is discarded. If the solution of the considered problem does not meet integrality constraints and the corresponding objective function value is higher than the upper bound, no better solution can be
found through further branching, and thus the branch receives no further consideration. If the current problem is infeasible no further branching makes sense and thus the branch is discarded.

Step 6. Optimality? If the list of problems to be processed is empty, the procedure concludes, the minimizer is the current best solution. If, on the other hand, the list is not empty, the procedure continues with step 3.

Observe that branching is stopped for three reasons, (i) problem infeasibility, (ii) meeting integrality constraints and (iii) the objective function value of a noninteger solution is above the current upper bound. We say that branching is stopped due to infeasibility, integrality and bounds, respectively.

At branching time, different variable are usually available for branching. How to select the most appropriate one generally requires a thorough knowledge of the problem under study.

Additionally, the problems waiting to be processed in the processing list can be considered using a deep-first strategy, a wide-first strategy or a mixed strategy. Selecting the right strategy is crucial to achieve an efficient solution. Physical knowledge of the system under consideration might lead to select the best processing strategy. A deep-first strategy quickly produces infeasible problems that leads to branch elimination, and generates good lower bounds as these bounds are provided by problems progressively more and more constrained. On the other hand, a wide-first strategy allows processing almost identical problems and that might constitute an advantage. Note finally that any processing strategy can be parallelized.

B.3 Nonlinear programming. Optimality conditions

We establish below the conditions to be met by a point to be a local minimizer of a nonlinear programming problem. Note that we characterize below local minimizers, not global ones, and recall that in linear programming, global minimizers are characterized, not local ones. Note also that we work with differentiable and sufficiently smooth functions. A nonlinear programming problem might include, or not, constraints, and the objective function and/or any constraint should be a nonlinear function. A point is a local minimizer if its corresponding objective function value is lower than the objective function of any point in its neighborhood. This definition is further formalized below. We formulate both necessary and sufficient conditions. Necessary conditions are met by minimizers but also by other points; that is, a point meeting necessary conditions is not necessarily a minimizer. Sufficient conditions are met only by minimizers; that is, a point meeting sufficient conditions is a minimizer. However, points not meeting sufficient conditions might be minimizers. We consider first and second order conditions. First order conditions involve just first order derivatives, while second order conditions involve both first and second order derivatives. Finally, it should be noted that understanding optimality conditions helps to understand how most algorithms to solve nonlinear programming problems work.

The material in this section is covered in detail by Bazaraa et al. [10], Bertsekas [11], Gill et al. [12] and Luenberger [7].
### B.3.1 Unconstrained problems

The unconstrained problem is

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad x \in \mathbb{R}^n,
\end{align*}
\]  

where \( f(x) : \mathbb{R}^n \to \mathbb{R} \), \( f(x) \in C^2 \) (up to second order derivatives continuous).

**Local minimizer:** \( x^* \in \mathbb{R}^n \) is a local minimizer of \( f(x) \) if there exists an \( \epsilon > 0 \) such that

\[
\forall x \in \mathbb{R}^n \quad |x - x^*| < \epsilon \quad \Rightarrow \quad f(x^*) \leq f(x).
\]

If \( f(x^*) < f(x) \), the local minimizer is said to be strict.

First order optimality conditions for unconstrained problems are provided below.

**Theorem 1:**

Let be \( f(x) : \mathbb{R}^n \to \mathbb{R} \), such that \( f(x) \in C^1 \), and \( x \in \mathbb{R}^n \). If \( \nabla f(y) \neq 0 \), then \( y \) is not a local minimizer.

**Proof sketch 1:**

A proof sketch is provides for the cases of \( \mathbb{R}^1 \) and \( \mathbb{R}^2 \). In \( \mathbb{R}^1 \) the condition above implies a null derivative, i.e., a horizontal tangent line; in \( \mathbb{R}^2 \) the theorem requires the gradient to be null, i.e., a horizontal tangent plane.

Second order sufficient conditions are provided below.

**Theorem 2:**

Let be \( f(x) : \mathbb{R}^n \to \mathbb{R} \), such that \( f(x) \in C^2 \) and \( x^* \in \mathbb{R}^n \). If \( \nabla f(x^*) = 0 \) and \( \nabla^2 f(x^*) > 0 \), then \( x^* \) is a strict local minimizer of \( f(x) \).

**Proof sketch 2:**

A definite positive hessian matrix \( (\nabla^2 f(x^*)) > 0 \) at the considered point implies a locally convex geometry (round deep valley). This plus the null gradient condition \( (\nabla f(x^*)) = 0 \) imply a local minimizer.

### B.3.2 Constrained problems

A constrained nonlinear programming problem has the general form

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad h(x) = 0, \quad g(x) \leq 0, \quad x \in \mathbb{R}^n, 
\end{align*}
\]  

where \( f(x) : \mathbb{R}^n \to \mathbb{R} \), \( h(x) : \mathbb{R}^n \to \mathbb{R}^m \), \( g(x) : \mathbb{R}^n \to \mathbb{R}^p \), and \( m \leq n \).

Functions \( h(x) \) and \( g(x) \) have the form

\[
\begin{align*}
h(x) &= \begin{bmatrix} h_1(x) \\ h_2(x) \\ \vdots \\ h_m(x) \end{bmatrix} \quad \text{and} \quad g(x) = \begin{bmatrix} g_1(x) \\ g_2(x) \\ \vdots \\ g_p(x) \end{bmatrix}.
\end{align*}
\]
We consider that 
\[ f(x) \in C^2; \quad h_i(x) \in C^2 \quad \forall i = 1, 2, \ldots, m; \quad g_i(x) \in C^2 \quad \forall i = 1, 2, \ldots, p. \]

We define below a local minimizer and a strict local minimizer.

**Local minimizer:** \( x^* \in \mathbb{R}^n \) so that \( h(x^*) = 0 \) and \( g(x^*) \leq 0 \) is a local minimizer of \( f(x) \) subject to \( h(x) = 0, g(x) \leq 0 \), if there exists an \( \epsilon > 0 \) such that \( f(x^*) \leq f(x), \forall x \in \mathbb{R}^n \) with \( h(x) = 0, g(x) \leq 0 \) and \( |x - x^*| < \epsilon \). If \( f(x^*) < f(x) \) the local minimizer is said to be strict.

Regularity conditions defining regular points are stated below. These conditions are important because they allow characterizing minimizers. We denote by \( \Omega \) the set of binding inequality constraints.

**Regular point:** \( y \in \mathbb{R}^n \) satisfying \( h_i(y) = 0 \ (i = 1, 2, \ldots, m), g_j(y) \leq 0 \ (j = 1, 2, \ldots, p) \) is a regular point of these constraints if the gradients of the binding constraints evaluated at \( y \) are linearly independent. That is, \( \nabla h_i(y), i = 1, 2, \ldots, m; \nabla g_j(y), j \in \Omega \) are linearly independent. \( \Omega \) is the set of binding inequality constraints.

Regularity allows formulating optimality conditions as stated below. Nonregular points might be minimizers but cannot be characterized through optimality conditions below.

First order optimality conditions are given below.

**Theorem 3:**
KKT Conditions. If

(1) \( x^* \) is a local minimizer of \( f(x) \) subject to \( h(x) = 0 \) and \( g(x) \leq 0 \), and

(2) \( x^* \) is a regular point of \( h(x) = 0 \) and \( g(x) \leq 0 \);

then

there exist vectors \( \lambda \in \mathbb{R}^m \) and \( \mu \in \mathbb{R}^p, \mu \geq 0 \), such that

\[
\nabla f(x^*) + \lambda^T \nabla h(x^*) + \mu^T \nabla g(x^*) = 0 \\
\mu^T g(x^*) = 0.
\]

**Proof sketch 3:**
For the sake of illustration, we simply show the validity of these conditions in \( \mathbb{R}^2 \) for a problem containing just one inequality constraint. This inequality partitions \( \mathbb{R}^2 \) into two regions, one feasible and another infeasible. If the curvature of the constraint and the objective function contours coincide (parallel gradients and in the same direction), the minimizer is in the interior of the feasible region and then \( \mu = 0 \). On the other hand, if the curvature of the constraint and the objective function contours oppose each other (parallel gradient and different directions), the minimizer lies in the boundary of the feasible region, being the constraint binding, and \( \mu \) should be positive for the gradient and the objective function and the constraints to be linearly dependents. Since only the two situations above are possible, we conclude that \( \mu \geq 0 \).
Degenerated constraints are characterized below.

**Degenerated constraint:** A binding constraint is degenerate if its associate multiplier is null and non degenerate if otherwise.

In what follows we consider binding constraints to be non degenerate. This assumption simplifies the derivations below and generally it is not a practical limitation.

Optimality conditions are formulated in a convenient manner through the Lagrangian function defined below.

**Lagrangian:** The Lagrangian of the constrained problem (B.28) is defined as
\[ L(x, \lambda, \mu) = f(x) + \lambda^T h(x) + \mu^T g(x). \]

Using the Lagrangian function and the fact that the binding inequality constraints are non degenerate, the first order optimality conditions are
\[
\nabla_x L(x, \lambda, \mu) = 0,
\n\nabla_\lambda L(x, \lambda, \mu) = 0,
\text{If } g_i < 0 \Rightarrow \mu_i = 0 \text{ and if } g_i = 0 \Rightarrow \mu_i > 0.
\]

The above conditions constitute a system of nonlinear equalities and inequalities. Thus the solution of a nonlinear system of equalities and inequalities is equivalent to solving an equality/inequality constrained nonlinear programming problem. Finally, note that solving nonlinear systems of equalities and inequalities is not simple.

Second order sufficient conditions are formulated below.

**Theorem 4:**

If

1. \( x^* \) satisfies \( h(x^*) = 0 \) and \( g(x^*) \leq 0 \),
2. \( \lambda \in \mathbb{R}^m \) and \( \mu \geq 0 \in \mathbb{R}^p \) satisfy
   \[
   \nabla^2 f(x^*) + \sum_{i=1}^m \lambda_i \nabla^2 h_i(x^*) + \sum_{j \in \Omega} \mu_j \nabla^2 g_j(x^*) > 0,
   \]
   on the subspace
   \[
   \{ y : \nabla h(x^*)^T y = 0, \nabla g_j(x^*)^T y = 0 \ \forall j \in \Omega \}, \text{ where } \Omega \text{ is the set of indices of binding inequality constraints;}
   \]

then

\( x^* \) is a strict local minimizer of \( f(x) \) subject to \( h(x) = 0 \) and \( g(x) \leq 0 \).

The proof of the conditions above can be found, for instance, in [7] or [12]. Multiplier vectors \( \lambda \) and \( \mu \) are sensitivity parameters as shown below.
B.4. UNCONSTRAINED PROBLEMS. SOLUTION METHODS

**Theorem 5**: Consider the nonlinear mathematical programming problem

$$\begin{align*}
\text{minimize}_x & \quad f(x) \\
\text{subject to} & \quad h(x) = c, \quad g(x) \leq d,
\end{align*}$$

such that $f(x) \in C^2$, $h_i(x) \in C^2, (i = 1, 2, \ldots, m)$ and $g_j(x) \in C^2, (j = 1, 2, \ldots, p)$.

For $c = 0$ and $d = 0$, let $x^*$ be such that $h(x^*) = 0$ and $g(x^*) \leq 0$ and regular, and consider that $x^*$, $\lambda^*$ and $\mu^*$ meet second order sufficiency conditions for a strict local minimizer, and that no binding constraint is degenerate.

Then, $\forall c \in \mathbb{R}^m$ and $\forall d \in \mathbb{R}^p$ in a neighborhood of $(0,0)$, the solution of (B.31), $x$, can be parameterized as a function of $c$ and $d$, that is, $x(c, d)$ and the equalities below hold:

$$\begin{align*}
\nabla_c f(x(c, d)) \big|_{(0,0)} &= -\lambda^* T \\
\nabla_d f(x(c, d)) \big|_{(0,0)} &= -\mu^* T.
\end{align*}$$

Note that $x(0,0) = x^*$.

The proof of the theorem above can be found, for instance, in [7] or [12].

The results above show that $\lambda_i/\mu_i$ provides the change of the objective function as a results of a change in the right-hand-side of element $c_i/d_i$. Actually, $\lambda_i/\mu_i$ are dual variables.

B.4 Unconstrained problems. Solution methods

This section reviews algorithms to solve the unconstrained nonlinear programming problem

$$\begin{align*}
\text{minimize}_x & \quad f(x) \\
\text{subject to} & \quad x \in \mathbb{R}^n
\end{align*}$$

(B.32)

where $f(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ and $f(x) \in C^2$.

The material in this section can be extended through the textbooks by Bazaraa et al. [10], Gill et al. [12], and Luenberger [7].

From a practical viewpoint, we briefly describe the following methods: gradient, Newton, quasi-Newton, conjugate directions and coordinate descent. Unconstrained solution methods are also important to solve constrained problems.

The general working mechanism of these algorithms is as follows: (i) start at an initial point, (ii) identify a descent direction, and (iii) determine the advance over the descent direction so that the objective function value decreases. This process is repeated until no improvement in the objective function can be achieved. Finally, observe that this procedure locates a local minimum.

B.4.1 Steepest-descent method

Consider an initial point $x_0$, the minus-gradient as a descent direction, and an advance step $\alpha$; that is, $x_\alpha = x_0 - \alpha \nabla f(x_0)$, $\alpha > 0$, then $x_\alpha - x_0 = -\alpha \nabla f(x_0)$.

Using Taylor expansion, the objective function at $x_\alpha$ becomes

$$\begin{align*}
f(x_\alpha) & \approx f(x_0) + \nabla f(x_0)^T (x_\alpha - x_0) = f(x_0) - \nabla f(x_0)^T \alpha \nabla f(x_0) \\
& = f(x_0) - \alpha [\nabla f(x_0)]^2.
\end{align*}$$

(B.33)
For $\alpha$ small enough, if $\nabla f(x_0) \neq 0$ we conclude that $f(x_\alpha) < f(x_0)$; that is, the minus-gradient is a descent direction.

Next we consider descent directions different from the minus-gradient. Consider the step, $x_\alpha = x_0 + \alpha d$, such that (i) $\alpha \geq 0$, (ii) $\nabla f(x_0) \neq 0$, (iii) $d \in \mathbb{R}^n$ and (iv) $\nabla f(x_0)^T d < 0$. Using Taylor expansion and evaluating the objective function at $x_\alpha$ we obtain

$$f(x_\alpha) \approx f(x_0) + \nabla f(x_0)^T (x_\alpha - x_0)$$

and since $\nabla f(x_0)^T d < 0$ we conclude that $f(x_\alpha) < f(x_0)$ for $\alpha$ small enough.

It is also relevant to consider the step $x_\alpha = x_0 - \alpha D \nabla f(x_0)$, such that (i) $\alpha \geq 0$, and (ii) $D$ being a definite positive matrix. Using Taylor expansion and evaluating the objective function at $x_\alpha$ we get

$$f(x_\alpha) \approx f(x_0) + \nabla f(x_0)^T (x_\alpha - x_0)$$

and being $D > 0$ and $\nabla f(x_0)^T D \nabla f(x_0) > 0$, then $f(x_\alpha) < f(x_0)$ for $\alpha$ small enough.

Therefore, the steps below are descent steps:

**Steepest-descent type iteration 1**: $x_{k+1} = x_k + \alpha_k d_k$, $k = 1, 2, \ldots$, where (i) $\alpha_k \geq 0$, (ii) $\nabla f(x_k)^T d_k < 0$ if $\nabla f(x_k) \neq 0$, and (iii) $d_k = 0$ if $\nabla f(x_k) = 0$.

**Steepest-descent type iteration 2**: $x_{k+1} = x_k - \alpha_k D_k \nabla f(x_k)$, $k = 1, 2, \ldots$, where (i) $\alpha_k \geq 0$, and (ii) $D$ is a positive definite matrix. For $D_k$ equal to the identity matrix, the algorithm above is the so-called steepest-descent algorithm.

Steepest-descent methods work as follows:

Step 1. Select a descent direction, $d_k$ or $-D \nabla f(x_k)$.

Step 2. Select a search step $\alpha_k$ such that the objective function value decreases.

Alternatively to use the minus-gradient direction is to select a positive definite diagonal matrix $D_k$ to modify that minus-gradient direction. Each diagonal element of $D_k$ modifies specifically each gradient component, and the resulting descent direction might be particularly effective.

The search step is obtained either using a line search or a rule that guarantees a large enough descent. A line search works as stated below.

**Line search**: Once fixed $x_k$ and $d_k$, we need to find $\alpha_k$ being the argument that minimizes the function $\phi(\alpha)$, where $\phi(\alpha) = f(x_k + \alpha d_k)$ with $x_k$ and $d_k$ fixed.

Observe that $\alpha \in \mathbb{R}$ and therefore the search is in one dimension. Possible search techniques include: (i) Fibonacci search, (ii) golden search, (iii) quadratic function fitting, and (iv) cubic function fitting. These methods are described, for instance, in [7].

Alternatively, search rules are easy to implement and computationally cheap. These rules guarantee that the advance is neither too small nor too large. The Armijo and Goldstein rules can be found in [7].

Although laborious, it is not complicated to show that a sequence generated by a steepest-descent algorithm does converge to a local minimizer under not very restrictive
B.4. UNCONSTRAINED PROBLEMS. SOLUTION METHODS

conditions. That is, convergence is guaranteed and the convergence rate is linear in the following sense. A sequence \( \{x_k\} \) generated by a steepest-descent method meets

\[
\lim_{k \to \infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = \frac{1}{a}, \quad a > 0. \tag{B.36}
\]

That is, the distance to the minimizer at iteration \( k + 1 \) is \( a \) time smaller than the distance to the minimizer at iteration \( k \).

Additionally, it can be proved that

\[
f(x_{k+1}) - f(x^*) \leq \left( \frac{L - l}{L + l} \right)^2 [f(x_k) - f(x^*)] \tag{B.37}
\]

where \( L \) and \( l \) are respectively the largest and smallest eigenvalues of the Hessian matrix of the objective function evaluated at the minimizer. Thus the objective function error at iteration \( k + 1 \) is smaller than or equal to the objective function error at iteration \( k \) times a coefficient that depends on the eigenvalues of the Hessian of the objective function at the minimizer. If all eigenvalues are equal (\( L = l \)), the error vanishes in one step. In \( \mathbb{R}^2 \) the above implies a sphere-shaped geometry and the one step error elimination becomes intuitive. If on the other hand \( L \gg l \), the error decreases from iteration to iteration but by a very small quantity. In \( \mathbb{R}^2 \) the above implies a narrow ellipsoid-shaped geometry. It can be concluded that it is convenient to pursue a sphere-shaped geometry around the minimizer. However, no specific procedures to achieve this are available. Nevertheless, simple and generally efficacious rules are available.

A reasonable possibility, although not always efficacious, is to perform a change of variables. But how? A promising alternative is to carry out the change of variables in such a manner that all components of the minimizer are of the same order of magnitude. If each component \( x_j \) of \( x \) is bounded as \( a_j \leq x_j \leq b_j \), the change of variables

\[
y_j = \frac{2x_j}{b_j - a_j} - \frac{a_j + b_j}{b_j - a_j} \quad \forall j, \tag{B.38}
\]

makes each variable \( y_j \) to be between -1 and 1. This variable transformation is generally efficacious and advisable if no further information is available.

An alternative variable transformation is

\[
y_j = \frac{x_j - a_j}{b_j - a_j} \quad \forall j, \tag{B.39}
\]

which makes each variable \( y_j \) to be between 0 and 1.

The steepest-descent iterative procedure is stopped once one or several of the conditions below are satisfied:

Condition 1. The objective function value does not decrease sufficiently, that is,

\[
\frac{|f(x_{k+1}) - f(x_k)|}{|f(x_{k+1})| + 1} < \epsilon_1.
\]
Condition 2. The solution does not change sufficiently, that is,
\[ \frac{\|x_{k+1} - x_k\|}{\|x_{k+1}\| + 1} < \epsilon_2. \]

Condition 3. The gradient at the considered solution is close enough to zero, that is,
\[ \|\nabla f(x_{k+1})\| < \epsilon_3. \]

**B.4.2 Newton and Quasi-Newton methods**

A second order Taylor expansion of the objective function value at \( x_k \) is
\[ f(x) \approx f(x_k) + \nabla f(x_k)^T \Delta x + \frac{1}{2} \Delta x^T \nabla^2 f(x_k) \Delta x. \] (B.40)

The trick is to determine \( \Delta x \) so that the quadratic expression above attains its minimizer. The derivative with respect to \( \Delta x \) of (B.40) is made equal zero. Thus,
\[ \nabla f(x_k) + \nabla^2 f(x_k) \Delta x = 0 \quad \text{and} \quad \Delta x = -[\nabla^2 f(x_k)]^{-1} \nabla f(x_k). \] (B.41)

Therefore, the Newton iteration is
\[ x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k), \quad k = 1, 2, \ldots. \] (B.42)

The observations below are in order:

Observation 1. Around a local minimizer the Hessian matrix is definite positive and the Newton method is well defined.

Observation 2. Outside the proximity of a local minimizer, the Hessian matrix might not be positive definite or could even be singular.

Observation 3. Newton method exhibits quadratic convergence in the sense below. A Newton-generated sequence \( \{x_k\} \) meets
\[ \lim_{k \to \infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^2} = \frac{1}{b}, \quad b > 0. \] (B.43)

That is, the distance to the minimizer at iteration \( k + 1 \) is \( b \) times smaller than the squared distance to the minimizer at iteration \( k \).

The conclusion from the above properties is that Newton method is fast if it converges but it might not converge.

Approximate variants of Newton method include:

1. Factorization of the Hessian not at every iteration but just every several iterations and using available factors until they are updated.

3. Using a Hessian matrix containing just diagonal elements (considering zero the off-diagonal elements)

Quasi-Newton methods have the same structure as Newton methods but the inverse of the Hessian matrix is progressively approximated. These methods differ just in the way they approximate the inverse of the Hessian matrix. We sketch below several quasi-Newton methods.

Davidon-Fletcher-Powell quasi-Newton method works as follows:

Step 1. Consider an arbitrary positive definite matrix $H_0$ and an initial point $x_0$. Set $k = 0$.

Step 2. Set $g_k = \nabla f(x_k)$ and $d_k = -H_k g_k$.

Step 3. Minimize $f(x_k + \alpha d_k)$ with respect to $\alpha > 0$ to obtain $x_{k+1} = x_k + \alpha_k d_k$ and also $g_{k+1} = \nabla f(x_{k+1})$. If convergence is attained (use one or several stopping rules), stop; otherwise, continue.

Step 4. Set $q_k = g_{k+1} - g_k$, and compute

$$H_{k+1} = H_k + \frac{p_k p_k^T}{q_k^T q_k} \left( H_k q_k q_k^T H_k - \frac{H_k q_k q_k^T H_k}{q_k^T q_k} \right).$$

(B.44)

Update $k$ and continue in step 2.

It should be noted that the inverse of the Hessian matrix is approximated using information pertaining to gradients in successive iterations.

Broyden-Fletcher-Goldfarb-Shanno quasi-Newton method is similar to the above method but updating the inverse Hessian as

$$H_{k+1} = H_k + \left( 1 + \frac{q_k^T H_k q_k}{q_k^T p_k} \right) \frac{p_k p_k^T}{q_k^T q_k} - \frac{p_k q_k^T H_k}{q_k^T q_k} - \frac{H_k q_k q_k^T H_k}{q_k^T q_k}.$$  

(B.45)

If we enforce $H_k = I$ in the expression above, the resulting method does not require storing $H_k$ and its behavior is generally good enough (memoryless Quasi-Newton method).

Additionally, it can be shown that the Hessian updating below provides a good scaling for the inverse Hessian (self-scaling Quasi-Newton method).

$$H_{k+1} = \left( H_k - \frac{H_k q_k q_k^T H_k}{q_k^T q_k} \right) \frac{p_k^T q_k}{q_k^T q_k} + \frac{p_k p_k^T}{q_k^T q_k}.$$  

(B.46)

Finally, it should be noted that the constructing procedures of the inverse Hessian provided above guarantee that this matrix is positive definite. It should also be noted that the line search in step 3 should be precise to adequately approximate the inverse Hessian.
Conjugate direction methods [12, 7] are based on extrapolating properties exhibited by quadratic functions. For a quadratic function \( f(x) = x^T A x + B x + C \) with \( A \) symmetric and definite positive, there exists a set of independent directions (conjugate directions) that if successively used as descent directions allows finding the minimizer in a finite number of steps.

To derive the iterative procedure of a conjugate direction method is complex but the actual structure of the iteration is simple. Fletcher-Reeves conjugate direction algorithm works as follows:

Step 1. Consider a point \( x_0 \), compute \( g_0 = \nabla f(x_0) \) and set \( d_0 = -g_0 \).

Step 2. For \( k = 0, 1, \ldots, n - 1 \) carry out the steps below:

1. If convergence is attained (use on or several stopping rules), stop; otherwise, continue.
2. Set \( x_{k+1} = x_k + \alpha_k d_k \) where \( \alpha_k \) minimizes in \( \alpha \) the function \( f(x_k + \alpha d_k) \).
3. Calculate \( g_{k+1} = \nabla f(x_{k+1}) \).
4. While \( k \neq n - 1 \), do \( d_{k+1} = -g_{k+1} + \beta_k d_k \) where
   \[
   \beta_k = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}.
   \]  
   (B.47)

Step 3. Replace \( x_0 \) with \( x_n \) and go to step 2.

Polak-Ribiere conjugate direction method is similar to the above method but updating parameter \( \beta_k \) as

\[
\beta_k = \frac{(g_{k+1} - g_k)^T g_{k+1}}{g_k^T g_k}.
\]  

(B.48)

Updating rule (B.48) usually performs better than (B.47).

Coordinate descent method

Methods that do not use derivatives converge generally slowly and their convergence is not guaranteed. These methods are of interest if the derivatives of the objective function are not available. One of such method is the coordinate descent method that works as follows:

Step 1. Consider a solution \( x_0 \).

Step 2. Improve iteratively \( f(x_0) \) solving for \( i = 1, 2, \ldots, n \)

\[
\text{minimize}_{x_i} \quad f(x_1, x_2, \ldots, x_n)
\]  

(B.49)

Step 3. Repeat the above step until convergence or until a large enough number of iterations has been carried out.
Note that not to compute the gradient is computationally an advantage but not being able to use the rich gradient information is a disadvantage.

Note that a coordinate descent method is a descent-direction method whose successive search directions are single-variable directions.

### B.5 Constrained problems. Solution methods

This section considers computational procedures to solve the problem

\[
\begin{align*}
\text{minimize}_x & \quad f(x) \\
\text{subject to} & \quad x \in S
\end{align*}
\]  

where \( f(x) : \mathbb{R}^n \to \mathbb{R}, \quad f(x) \in C^2 \) and \( S \subset \mathbb{R}^n \).

Set \( S \) represent equality and inequality constraints and bounds on variables. For equality constraints only, the set \( S \) has the form \( S = \{ x : h_i(x) = 0, \forall i = 1, 2, \ldots, m \} \). For inequality constraints only, the set \( S \) has the form \( S = \{ x : g_i(x) \leq 0, \forall i = 1, 2, \ldots, p \} \).

Additional information on solution methods for constrained nonlinear programming problems can be found in Bazaraa et al. [10], Gill et al. [12], Luenberger [7] and Bertsekas [11].

Among the many techniques available to solve constrained nonlinear programming problems, this appendix considers penalty and barrier methods, augmented Lagrangian methods and the primal-dual interior point method.

#### B.5.1 Penalty and barrier methods

**Penalty methods**

The constrained mathematical programming problem

\[
\begin{align*}
\text{minimize}_x & \quad f(x) \\
\text{subject to} & \quad x \in S
\end{align*}
\]  

is substituted by the unconstrained problem below that incorporates into the objective function a penalty term,

\[
\begin{align*}
\text{minimize}_x & \quad f(x) + c P(x)
\end{align*}
\]  

where \( c \in \mathbb{R}, \ c > 0; \ P(x) : \mathbb{R}^n \to \mathbb{R} \) so that \( P(x) \) is a continuous function, \( P(x) \geq 0 \ \forall x \in \mathbb{R}^n \) and \( P(x) = 0 \) if and only if \( x \in S \).

In the case of just equality constraints, \( S = \{ x : h_i(x) = 0, \forall i = 1, 2, \ldots, m \} \), and the penalty function \( P(x) \) might have the form \( P(x) = \frac{1}{2} \sum_{i=1}^{m} (h_i(x))^2 \). In the case of just inequality constraints, \( S = \{ x : g_i(x) \leq 0, \forall i = 1, 2, \ldots, p \} \), and the penalty function \( P(x) \) might have the form \( P(x) = \frac{1}{2} \sum_{i=1}^{p} (\text{maximum } [0, g_i(x)])^2 \).

Note that the search for the minimizer occurs outside the feasible region, which is approached as \( c \to \infty \).

Penalty methods work as follows:
Step 1. Set a sequence \( \{c_k\} \), \( k = 1, 2, \ldots \), so that \( c_k > 0 \) and \( c_{k+1} > c_k \).

Step 2. Build the penalized function \( q(c_k, x) = f(x) + c_k P(x) \).

Step 3. For each \( k \) solve the problem minimize\( x \) \( q(c_k, x) \) and obtain \( x_k \). Obtain a new penalty parameter (from the sequence described in step 1) and use \( x_k \) as the starting point in step 2. Stop once one or several stopping rules are satisfied.

It can be easily shown that the limit point of a sequence of points \( \{x_k\} \) generated by a penalty procedure is a solution of the original problem (B.51).

Penalty parameters and Lagrange multiplies are closely related as stated below. Consider the problems:

\[
\begin{align*}
\text{minimize}_x & \quad f(x) \\
\text{subject to} & \quad h(x) = 0, \\
& \quad (B.53)
\end{align*}
\]

and

\[
\begin{align*}
\text{minimize}_x & \quad f(x) + c_k \gamma(h(x)) \\
& \quad (B.54)
\end{align*}
\]

where \( h(x) : \mathbb{R}^n \to \mathbb{R}^m \) and \( \gamma(h) : \mathbb{R}^m \to \mathbb{R} \) is the penalty function.

The first order optimality condition of (B.54) at \( x_k \) is

\[
\nabla f(x_k) + c_k \nabla h \gamma(h(x_k)) \nabla h(x_k) = 0, \\
& \quad (B.55)
\]

and the first order optimality condition of problem (B.53) at \( x_k \) is

\[
\nabla f(x_k) + \lambda^T_k \nabla h(x_k) = 0. \\
& \quad (B.56)
\]

Imposing that both conditions must hold at the minimizer implies

\[
\lambda^T_k = c_k \nabla h \gamma(h(x_k)). \\
& \quad (B.57)
\]

Note that \( x_k \to x^* \) leads to \( \lambda_k \to \lambda^* \), where \( x^* \) is a regular point of \( h(x) \) and a solution of (B.53) and \( \lambda^* \) is the multiplier vector associated with that solution.

Inequality constrained problems are considered next.

Consider the problems:

\[
\begin{align*}
\text{minimize}_x & \quad f(x) \\
\text{subject to} & \quad g(x) \leq 0, \\
& \quad (B.58)
\end{align*}
\]

and

\[
\begin{align*}
\text{minimize}_x & \quad f(x) + c_k \gamma(g(x)) \\
& \quad (B.59)
\end{align*}
\]

where \( g(x) : \mathbb{R}^n \to \mathbb{R}^m \) and \( \gamma(g) : \mathbb{R}^m \to \mathbb{R} \) is the penalty function.

The first order optimality condition of (B.59) at \( x_k \) is

\[
\nabla f(x_k) + c_k \nabla g \gamma(g(x_k)) \nabla g(x_k) = 0, \\
& \quad (B.60)
\]
and the first order optimality conditions of (B.58) at \( x_k \) are

\[
\nabla f(x_k) + \mu_k^T \nabla g(x_k) = 0, \quad \mu_k \geq 0, \\
\mu_k^T g(x_k) = 0. 
\]

(B.61)

At the minimizer, it should hold

\[
\mu_k^T = c_k \nabla g(\gamma(x_k)), \quad \mu_k \geq 0.
\]

(B.62)

Note that as \( x_k \rightarrow x^* \) leads to \( \mu_k \rightarrow \mu^* \), where \( x^* \) is a regular point of \( g(x) \) and a solution of (B.58) and \( \mu^* \) is the multiplier vector associated to that solution.

Note also that non binding constraints have null multipliers.

The structure of the Hessian matrix is analyzed below. Consider the penalized objective function

\[
q(c, x) = f(x) + c \, \gamma(e(x))
\]

(B.63)

where \( e(x) \) includes both equality and inequality constraints.

The gradient of (B.63) is

\[
\nabla q(c, x) = \nabla f(x) + c \, \nabla e(\gamma(e(x))) \, \nabla e(x),
\]

(B.64)

and the Hessian matrix of (B.63) is

\[
\nabla^2 q(c, x) = \nabla^2 f(x) + c \, \nabla e(\gamma(e(x))) \, \nabla^2 e(x) + c \, \nabla e(x)^T \, \nabla^2 e(\gamma(e(x))) \, \nabla e(x).
\]

(B.65)

Denominating \( \nabla^2 \mathcal{L}(x) = \nabla^2 f(x) + \lambda^T \, \nabla^2 e(x) \) where \( \lambda^T = c \, \nabla e(\gamma(e(x))) \), the Hessian becomes

\[
\nabla^2 q(c, x) = \nabla^2 \mathcal{L}(x) + c \, \nabla e(x)^T \, \nabla^2 e(\gamma(e(x))) \, \nabla e(x).
\]

(B.66)

The observations below are in order:

Observation 1. Matrix \( \nabla^2 \mathcal{L}(x) \) approximates the Hessian of the Lagrangian function of problem (B.51) at the minimizer as \( x_k \) gets closer to the minimizer \( x^* \). Therefore neither the limit matrix nor its condition number depend on \( c \).

Observation 2. Matrix \( [c \, \nabla e(x)^T \, \nabla^2 e(\gamma(e(x))) \, \nabla e(x)] \) approaches infinity as \( c \) approaches infinity.

Therefore, as \( c \) increases the Hessian matrix (B.66) becomes worse and worse conditioned.

**Barrier methods**

To use a barrier method, it is convenient that the feasible region of the considered problem is defined through inequalities.

The constrained problem

\[
\minimize_x \quad f(x) \\
\text{subject to} \quad x \in S
\]

(B.67)
can be substituted by the unconstrained problem below that incorporates in the objective function a barrier term,

\[ \text{minimize}_x \ f(x) + \frac{1}{c} B(x) \]  

(B.68)

where \( c \in \mathbb{R}; \ c > 0; \ B(x) : \mathbb{R}^n \rightarrow \mathbb{R} \) is continuous and \( B(x) > 0 \ \forall x \in \mathbb{R}^n \), and \( B(x) \rightarrow \infty \) if \( x \) approaches the feasibility boundary of \( \mathcal{S} \).

For just inequality constraints, the barrier function \( B(x) \) might take the form

\[ B(x) = -\sum_{i=1}^{p} \frac{1}{g_i(x)} \]  

(B.69)

Barrier methods work as follows:

Step 1. Establish a sequence \( \{c_k\}, \ k = 1, 2, \ldots, \) so that \( c_k > 0 \ , \ c_{k+1} > c_k \).

Step 2. Build the barrier function \( r(c_k, x) = f(x) + \frac{1}{c_k} B(x) \).

Step 3. Solve the problem

\[ \text{minimize}_x \ r(c_k, x) \]  
\[ \text{subject to} \quad x \in \text{interior of} \ \mathcal{S} \]  

(B.70)

and obtain \( x_k \). The procedure conclude once one or several stopping criteria are satisfied. Otherwise, the algorithm continues obtaining a new barrier parameter \( c_k \) and using \( x_k \) as the starting point in step 2.

It is easy to prove that the limit point of a sequence \( \{x_k\} \) generated by a barrier method is a solution of problem (B.67).

Barrier penalty parameters and Lagrange multipliers are closely related. Consider the problems:

\[ \text{minimize}_x \ f(x) \]  
\[ \text{subject to} \quad g(x) \leq 0, \]  

and

\[ \text{minimize}_x \ f(x) + \frac{1}{c_k} \eta(g(x)) \]  

(B.71)

(B.72)

where \( g(x) : \mathbb{R}^n \rightarrow \mathbb{R}^p \) and \( \eta(g) : \mathbb{R}^p \rightarrow \mathbb{R} \) is the barrier function.

The first order optimality condition of (B.72) at \( x_k \) is

\[ \nabla f(x_k) + \frac{1}{c_k} \nabla g \eta(g(x_k)) \nabla g(x_k) = 0, \]  

(B.73)

and the first order optimality conditions of (B.71) at \( x_k \) are

\[ \nabla f(x_k) + \mu_k^T \nabla g(x_k) = 0, \]  
\[ \mu_k \geq 0, \]  
\[ \mu_k^T g(x_k) = 0. \]  

(B.74)
Requiring that both conditions are satisfied at the minimizer implies
\[ \mu_k^T = \frac{1}{c_k} \nabla g \eta(g(x_k)) , \quad \mu_k \geq 0 . \] (B.75)

Note that \( x_k \to x^* \) leads to \( \mu_k \to \mu^* \), where \( x^* \) is a regular point of \( g(x) \) and a solution of (B.71) and \( \mu^* \) is the vector of Lagrangian multipliers associated with that solution. Note also that non-binding constraints have null multipliers.

It is simple to prove that the Hessian of the objective function incorporating a barrier function suffers a similar ill-conditioning as a objective function that incorporates a penalty term.

### B.5.2 Augmented Lagrangian methods

The augmented Lagrangian method or method of the multipliers is considered next. Only equality constrained problems are considered. The extension to both equality and inequality constrained problems is described, for instance, in [13]. Consider the problem

\[
\begin{align*}
\text{minimize}_x & \quad f(x) \\
\text{subject to} & \quad h(x) = 0.
\end{align*}
\] (B.76)

The augmented Lagrangian function is defined as
\[
\mathcal{L}(x, \lambda) = f(x) + \lambda^T h(x) + \frac{1}{2}c[h(x)]^2
\] (B.77)

where \( c \in \mathbb{R} \) is a positive constant.

Consider also the equality constrained optimization problem

\[
\begin{align*}
\text{minimize}_x & \quad f(x) + \lambda^T h(x) \\
\text{subject to} & \quad h(x) = 0.
\end{align*}
\] (B.78)

Observe that the quadratic penalty function plus the objective function of problem (B.78) is the augmented Lagrangian of problem (B.76).

Additionally, the equality constrained problem below is considered

\[
\begin{align*}
\text{minimize}_x & \quad f(x) + \frac{1}{2}c[h(x)]^2 \\
\text{subject to} & \quad h(x) = 0.
\end{align*}
\] (B.79)

Observe that the Lagrangian function of problem (B.79) is the augmented Lagrangian of problem (B.76).

Based on the above observations, the ingredients of the augmented Lagrangian method are described below.

The first order optimality condition of problem (B.76) is
\[
\nabla f(x) + \lambda^* \nabla h(x) = 0.
\] (B.80)
On the other hand, consider the equality constrained problem

\[
\begin{align*}
\text{minimize}_x \quad & f(x) + \lambda_k^T h(x) \\
\text{subject to} \quad & h(x) = 0,
\end{align*}
\]  

(B.81)

The first order optimality condition for problem (B.81) is

\[
\nabla f(x) + \lambda_k^T \nabla h(x) + \lambda^T \nabla h(x) = 0.
\]  

(B.82)

where $\lambda$ is the optimal Lagrange multiplier vector.

Note that the solutions of the two problems previously formulated are identical. Considering the first order optimality conditions for both of them, we conclude that $\lambda = \lambda^* - \lambda_k$.

On the other hand, the penalized problem associated with problem (B.81) is

\[
\begin{align*}
\text{minimize}_x \quad & f(x) + \lambda_k^T h(x) + \frac{1}{2}c[h(x)]^2.
\end{align*}
\]  

(B.83)

The optimal multiplier vector of this problem can be analytically obtained (see (B.57)) and is

\[
\lambda = c \nabla h \left[ \frac{1}{2} (h(x))^2 \right] = c h(x_k),
\]  

(B.84)

where $x_k$ is the solution of the penalized problem (B.83).

The above Lagrange multiplier vectors are related as follows

\[
\lambda = \lambda^* - \lambda_k \quad \text{and} \quad \lambda = c h(x_k)
\]  

(B.85)

It can be concluded that $\lambda^* = \lambda_k + c h(x_k)$, and an appropriate updating rule for $\lambda^*$ is

\[
\lambda_{k+1} = \lambda_k + c h(x_k).
\]  

(B.86)

Therefore, the augmented Lagrangian method to solve problem (B.76) works as follows:

Step 1. Consider an appropriate penalty constant $c$ and initialize the multiplier vector $\lambda_k$.

Step 2. Compute $x_k$ solving the unconstrained problem:

\[
\begin{align*}
\text{minimize}_x \quad & f(x) + \lambda_k^T h(x) + \frac{1}{2}c[h(x)]^2.
\end{align*}
\]  

Step 3. Update the multiplier vector as $\lambda_{k+1} \leftarrow \lambda_k + c h(x_k)$.

Step 4. Repeat steps 2 and 3 until convergence in $\lambda$ is attained.

It can be proved that for $c$ large enough, the above procedure converges to a solution of problem (B.76). On the other hand, $c$ should be small enough to avoid ill-conditioning. It is worth to mention that the penalty parameter $c$ can be increased with the iterations to facilitate convergence.

More sophisticated multiplier updating procedures that incorporate second derivative information are available [12], [13]. The augmented Lagrangian method can be extended to consider inequality constraints as illustrated in [13] or [4].
B.5.3 Primal-dual interior point methods

Interior point methods were firstly developed to tackle linear programming problems. However, these algorithms can successfully solve general convex nonlinear problems. The primal-dual interior point algorithm for linear programming problems is developed below. Additional information is available in [14].

Consider the problem

\[
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad Ax = b, \quad x \geq 0,
\end{align*}
\] (B.87)

where \( x, c \in \mathbb{R}^n \), \( b \in \mathbb{R}^m \) and \( A \in \mathbb{R}^{m \times n} \).

The dual problem of problem (B.87) is

\[
\begin{align*}
\text{maximize} & \quad b^T y \\
\text{subject to} & \quad A^T y \leq c.
\end{align*}
\] (B.88)

Converting inequality constraints into equalities using slack variables, problem (B.88) above becomes

\[
\begin{align*}
\text{maximize}_{y,z} & \quad b^T y \\
\text{subject to} & \quad A^T y + z = c, \quad z \geq 0.
\end{align*}
\] (B.89)

The dual variables are \( y \in \mathbb{R}^m \) and \( z \in \mathbb{R}^n \).

To eliminate nonnegativity constraint, logarithmic barriers are used, i.e.,

\[
\begin{align*}
\text{maximize}_{y,z} & \quad b^T y + \mu \sum_{j=1}^{n} \ln z_j \\
\text{subject to} & \quad A^T y + z = c.
\end{align*}
\] (B.90)

The interior point method relies on solving different instances of problem (B.90) for increasing values of the parameter \( \mu \). This parameter is defined so that \( \mu_0 > \mu_1 > \mu_2 > \cdots > \mu_\infty = 0 \). The following result shows the interest of the above algorithm.

Parameter sequence \( \{\mu_k\} \) results in a sequence of problems (B.90), whose solutions, \( \{x_k\} \), progressively approach the solution of the original problem (B.89). This result is shown, for instance, in [14].

The lagrangian of problem (B.90) is

\[
\mathcal{L}(x, y, z, \mu) = b^T y + \mu \sum_{j=1}^{n} \ln z_j - x^T (A^T y + z - c).
\] (B.91)

The first order optimality conditions for problem (B.90) are

\[
\nabla_x \mathcal{L}(\cdot) = A^T y + z - c = 0; \quad \nabla_y \mathcal{L}(\cdot) = Ax - b = 0; \quad \nabla_z \mathcal{L}(\cdot) = XZe - \mu e = 0
\] (B.92)

where

\[
X = \text{diag}(x_1, x_2, \ldots, x_n); \quad Z = \text{diag}(z_1, z_2, \ldots, z_n); \quad e = (1, 1, \ldots, 1)^T.
\] (B.93)
Note that the dimension of \( e \) is \( n \times 1 \).

To solve by Newton the nonlinear system of equations (B.92), \( x, y \) and \( z \) are substituted by \( x + \Delta x, y + \Delta y \) and \( z + \Delta z \), respectively. Ignoring second order terms, system (B.92) becomes

\[
\begin{align*}
Z \Delta x + X \Delta z &= \mu e - XZe \\
A \Delta x &= 0 \\
A^T \Delta y + \Delta z &= 0.
\end{align*}
\] (B.94)

Primal and dual search directions are obtained solving (B.94) for \( \Delta y, \Delta z \) and \( \Delta x \), that is

\[
\begin{align*}
\Delta y &= -(AXZ^{-1}A^T)^{-1}AZ^{-1}v(\mu) \\
\Delta z &= -A^T \Delta y \\
\Delta x &= Z^{-1}v(\mu) - XZ^{-1} \Delta z
\end{align*}
\] (B.95)

where \( v(\mu) = \mu e - XZe \).

The Newton iteration is then

\[
\begin{align*}
x^{k+1} &= x^k + \alpha_p \Delta x \\
y^{k+1} &= y^k + \alpha_d \Delta y \\
z^{k+1} &= z^k + \alpha_d \Delta z
\end{align*}
\] (B.96)

where \( 0 \leq \alpha_p \leq 1, 0 \leq \alpha_d \leq 1 \) are primal and dual search steps, respectively.

The primal search step concerns variable \( x \), while the dual one concerns variables \( y \) and \( z \). The selection of \( \alpha_p \) and \( \alpha_d \) is carried out so that \( x \) and \( z \) remain positive (\( y \) does not need to be positive). To enforce positiveness, parameter \( \sigma \) (0 < \( \sigma \) < 1) is used. Thus

\[
\begin{align*}
\alpha_x &= \text{minimum} \left\{ \frac{-\Delta z_i}{\Delta x_i} \text{ so that } \Delta x_i < -\delta \right\}, \quad \alpha_p = \text{minimum} \{1, \sigma \alpha_x\} \\
\alpha_z &= \text{minimum} \left\{ \frac{-\Delta y_j}{\Delta z_j} \text{ so that } \Delta z_j < -\delta \right\}, \quad \alpha_d = \text{minimum} \{1, \sigma \alpha_z\}
\end{align*}
\] (B.97, B.98)

where \( \delta \) is a tolerance parameter (e.g., \( \delta = 0.0001 \)) and \( \sigma \) is generally equal to 0.99995.

The above adjustments are important if \( \Delta x_i \) and \( \Delta z_j \) are large enough.

The duality gap is the difference between the objective function value of the primal problem and the objective function value of the dual one. For feasible values \( x \) and \( y \), the duality gap is \( c^T x - b^T y \). This duality gap is an appropriate metric of proximity to optimality.

If the available initial solution is not feasible, Newton system (B.94) becomes

\[
\begin{align*}
Z \Delta x + X \Delta z &= \mu e - XZe \\
A \Delta x &= b - Ax \\
A^T \Delta y + \Delta z &= c - A^T y - z
\end{align*}
\] (B.99)

whose solution is

\[
\begin{align*}
\Delta y &= -(AXZ^{-1}A^T)^{-1}(AZ^{-1}v(\mu) - AXZ^{-1}r_d - r_p) \\
\Delta z &= -A^T \Delta y + r_d \\
\Delta x &= Z^{-1}v(\mu) - XZ^{-1} \Delta z
\end{align*}
\] (B.100)
where

\[ r_p = b - Ax, \quad r_d = c - A^T y - z \]  \hspace{1cm} (B.101)

are primal and dual residuals, respectively.

If the initial solution is infeasible, both feasibility and optimality are achieved simultaneously. That is, the residuals and the duality gap approach zero simultaneously.

Barrier penalty parameter \( \mu \) can be updated as follows. Using optimality conditions (B.92) leads to

\[ \mu = \frac{z^T x}{n} \]  \hspace{1cm} (B.102)

Values for \( \rho \) are experimentally obtained. A reasonable selection is [15]

\[ \rho = \begin{cases} 
0.1 \text{ if } c^T x > b^T y \\
0.2 \text{ if } c^T x < b^T y. 
\end{cases} \]  \hspace{1cm} (B.103)

A reasonable stopping criterion is to achieve a small enough duality gap, i.e.,

\[ \frac{|c^T x^k - b^T y^k|}{\max\{1, |c^T x^k|\}} < \varepsilon, \]  \hspace{1cm} (B.104)

where \( \varepsilon \) is a per unit tolerance parameter. The numerator of the above expression is the duality gap whereas the denominator is the maximum between 1 and the absolute value of the primal objective function. This stopping criterion avoids division by zero.

The extension of the above results to linear programming problems that include upper and lower bounds on variables is easily achieved. See, for instance, [15].

The primal-dual interior point algorithm works as follows:

Step 1. Set \( k = 0 \) and select an initial solution \((x^0, y^0, \mu_0)\) and a tolerance parameter \( \varepsilon \). This initial solution can be primal/dual feasible or not.

Step 2. Compute the vector of slack dual variables \( z^k = c - A^T y^k \), and build the diagonal matrices \( X^k = \text{diag}(x_1^k, x_2^k, \ldots, x_n^k) \) and \( Z^k = \text{diag}(z_1^k, z_2^k, \ldots, z_n^k) \).

Step 3. Calculate vector \( v(\mu_k) = \mu_k e - X^k Z^k e \), and residual vectors \( r_p = Ax - b \) and \( r_d = c - A^T y - z \). If the current solution is primal and dual feasible, residual vectors are zero.

Step 4. Calculate search directions \( \Delta y, \Delta z \) and \( \Delta x \), solving the corresponding Newton system. Use (B.100).

Step 5. Calculate the steps \( \alpha_p \) and \( \alpha_d \). Use (B.97) and (B.98).

Step 6. Update primal and dual variables using (B.96).
Step 7. If the duality gap is small enough (expression (B.104)), stop, a solution has been found within a level of accuracy of $\varepsilon$. Otherwise, set $k \leftarrow k + 1$ and continue with step 8 below.

Step 8. Update the barrier parameter $\mu_k$ (equations (B.102) and (B.103)) and continue in step 2.
Bibliography


Index

Constrained problems. Solution methods, 23
  augmented Lagrangian methods, 27
  barrier methods, 25
  penalty methods, 23
  primal-dual interior point methods, 29

Linear programming, 3
  duality, 9
  fundamentals, 3
  sensitivity, 8
  Simplex, 6

Mixed-integer linear programming, 11
  branch and bound, 12

Nonlinear programming, 13
  constrained problems, 14
  optimality conditions, 13
  unconstrained problems, 14

Unconstrained problems. Solution methods, 17
  conjugate directions methods, 22
  coordinate descent method, 22
  Newton method, 20
  quasi-Newton methods, 20
  steepest-descent method, 17